Universidad de Granada

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Programa de Doctorado en Ciencias de la Computación
y Tecnología Informática

Estudios para clasificación con datos con ruido

Tesis Doctoral

José Antonio Sáez Muñoz

Granada, Octubre de 2014

Editor: Editorial de la Universidad de Granada
Autor: José Antonio Sáez Muñoz
D.L.: GR 2383-2014
ISBN: 978-84-9083-278-3

Universidad de Granada

Estudios para clasificación con datos con ruido

MEMORIA PRESENTADA POR

José Antonio Sáez Muñoz

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Octubre de 2014

DIRECTORES

Francisco Herrera Triguero y Julián Luengo Martín

Departamento de Ciencias de la Computación
e Inteligencia Artificial

La memoria titulada "*Estudios para clasificación con datos con ruido*", que presenta D. José Antonio Sáez Muñoz para optar al grado de doctor, ha sido realizada dentro del Programa Oficial de Doctorado en "*Ciencias de la Computación y Tecnología Informática*", en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección de los doctores D. Francisco Herrera Triguero y D. Julián Luengo Martín.

El doctorando y los directores de la tesis garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis, y hasta donde nuestro conocimiento alcanza, en la realización del trabajo se han respetado los derechos de otros autores a ser citados cuando se han utilizado sus resultados o publicaciones.

Granada, Octubre de 2014

El Doctorando

Fdo: José Antonio Sáez Muñoz

Los directores

Fdo: Francisco Herrera Triguero

Fdo: Julián Luengo Martín

Esta tesis doctoral ha sido desarrollada con la financiación del proyecto TIN2011-28488 del Ministerio de Ciencia e Innovación. También ha sido subvencionada bajo el Programa de Becas de Formación de Profesorado Universitario del Ministerio de Educación, en su Resolución del 11 de Octubre de 2010, bajo la referencia AP2009-2930.

Dedicada a mis padres José Antonio y Rosario;
y mis abuelos Paco, Mariana, Luis y Fina.

Resumen

En esta tesis se han abordado varios problemas relacionados con la presencia de ruido en los datos en tareas de clasificación. Dos líneas principales de investigación, vinculadas con las dos propuestas clásicas para tratar el ruido, constituyen su contenido: estudios y propuestas para el tratamiento de datos con ruido tanto a nivel de algoritmos como a nivel de preprocesamiento de datos. Los objetivos llevados a cabo fueron:

1. Proponer y analizar alternativas para hacer que los clasificadores funcionen mejor con datos con ruido, con independencia del clasificador seleccionado y las características de los datos.
2. Profundizar en el problema del ruido de atributos, que usualmente ha sido menos estudiado que el problema de ruido de clases en la literatura.
3. Analizar la necesidad de la aplicación de las técnicas de preprocesamiento del ruido basándonos en las propiedades de los datos y diseñar nuevos métodos de preprocesamiento del ruido.
4. Estudiar y proponer medidas de evaluación del comportamiento de los clasificadores con datos con ruido.

Con estos objetivos en mente, se han aplicado sistemas de clasificación basados en múltiples clasificadores para tratar con datos con ruido, obteniendo buenos resultados. Igualmente, se ha propuesto un esquema de ponderación de características basado en métodos de imputación y test estadísticos. Esta combinación permite reducir el impacto del ruido de atributos y ha mostrado superar a otros métodos de tratamiento del ruido de atributos del estado del arte.

También se ha estudiado la relación entre las características de los datos, analizando sus medias de complejidad, y la eficiencia de los filtros del ruido, llegando a la conclusión de que el filtrado del ruido es beneficioso cuando se trata con problemas con un alto grado de solapamiento entre las clases. Además, se han propuesto varios métodos de preprocesamiento de ruido, principalmente basados en el uso de múltiples clasificadores y filtros, que han mostrado buenos resultados al tratar con datos con ruido.

Finalmente, se han planteado varias medidas de evaluación del comportamiento de los clasificadores con datos con ruido. Se han analizado las propiedades de cada una de ellas, llegando a la conclusión de que es necesario considerar el rendimiento y la robustez en dichas medidas si se desea obtener un buen estimador del comportamiento de los clasificadores al entrenar en problemas con ruido.

Table of Contents

Chapter I

PhD dissertation

1. Introduction

In the last decades, the advance of technology has produced an increment in the amount of data to process in applications of many fields (such as medicine or economy). This fact has implied the need of new techniques to manage and analyze this vast amount of data. Since these tasks exceed the human capabilities, the development of automatic procedures to address these issues has been essential [BHS09, Bra07].

Thus, both the acquisition of new knowledge from large amounts of data and its analysis have become tasks highly relevant. In the literature, these works are usually included within the *Knowledge Discovery in Databases* (KDD) [Fri97, TSK05] process. The main aim of the knowledge obtained through the KDD process is the utility, that is, the knowledge must be useful to the problem addressed. The steps of the KDD process are described in the following (see Figure 1):

1. **Data selection.** It is addressed from the previous knowledge of experts in the problem.
2. **Data preprocessing.** It includes the cleaning of the data (treatment of errors and corruptions) and its reduction (selection of the features and the more representative examples of the problem) in order to facilitate the *data mining* stage.
3. **Data transformation.** Available data are transformed into a structured representation that reflects the knowledge of the problem in the initial data.
4. **Data mining.** Patterns of knowledge are constructed from the analysis of the data.
5. **Data interpretation.** This stage aims to understand the patterns obtained in a way in which they could be useful for the users.

The main phase of the KDD process is its fourth stage, that is, the *data mining* [TSK05] stage. This is focused on the identification of patterns and the prediction of relationships among the data. Generally, data mining techniques are classified into *descriptive* (when they are used to discover patterns in the data) and *predictive* (when they are used to predict the behavior of a model through the analysis of the data). These processes (descriptive and predictive) are usually carried out using *machine learning* algorithms [Alp10], which are defined as those mechanisms that are able to induce

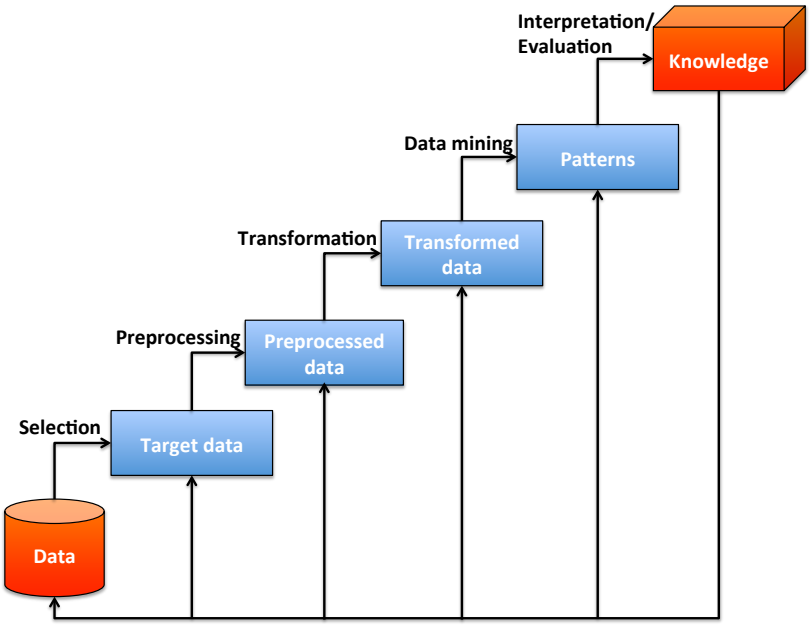


Figure 1: The KDD process.

knowledge from the data. Two main types of problems are considered depending on the information available by the machine learning algorithm for the extraction of knowledge:

1. **Supervised learning problems.** This establishes a relationship between the input variables and the output (target variable), which are all known, in order to predict the output of new examples whose target variable is unknown. This problem is divided into two different problems depending on the type of target variable: (i) *classification* [DHS00], in which the values of the target variable (also known as class) are finite; and (ii) *regression* [CM98], in which the values of the target variable are continuous.

2. **Unsupervised learning problems.** In this problem, in which the values of the target variable are unknown, the main aim is to describe the relationships among the data. This is mainly divided into two groups of problems: (i) *clustering* [Har75], in which the data are separated into different groups that share similar characteristics; and (ii) *association* [AIS93], in which the relationships among transactional data are identified.

Classification methods learn to categorize items (*examples*) into several predefined classes from a model (*classifier*) that is constructed from a set of examples (the *training* set) and then, it is applied to predict the class of examples of another set (the *test* set), which has not been used in the learning phase. The *performance* of the classifier can be measured using different metrics, such as the *accuracy* (which is viewed as the confidence of the classifier learned), the *learning time* required to build classifier or its *interpretability* (the clarity and credibility of the classifier from the human point of view).

Since classifiers are built from a training data set, their classification accuracy is directly influenced by the quality of these data. Data quality depends on several components [WSF95], e.g., the source and the input procedure, inherently subject to errors. Thus, real-world data sets usually

contain imperfect data, formally known as *noise* [ZW04], that may hinder the classifiers built and, therefore, the interpretations and decisions obtained from them.

Two types of noise in a given data set can be distinguished based on the two types of variables of a classification problem (output variable or *class* and input variables or *attributes*) [Wu96]:

1. **Class noise.** It occurs when an example is incorrectly labeled due to the presence of noise in the class label of the training examples.
2. **Attribute noise.** It refers to corruptions in the values of one or more attributes of the data set. Examples of attribute noise are: erroneous attribute values, unknown or missing values (MV) [JL13], and irrelevant attributes to build the classifier.

Given the loss of accuracy produced by noise, two ways have been proposed in the specialized literature in order to mitigate its effects:

1. **Algorithm level approaches.** They are mainly based on the adaptation of the algorithms to properly handle the noise [Qui93]. These methods are known as *robust learners* and they are characterized by being less influenced by noisy data.
2. **Data level approaches**. They preprocess the data sets aiming to remove or correct the noisy examples [BF99].

Even though both techniques can provide good results, they present some drawbacks. Thus, for example, the adaptation of the algorithms depends on the particular algorithm to modify. Therefore, the same adaptation is not directly extensible to other learning algorithms, since it depends of the characteristics of the method. This approach requires to change an existing method, which neither is always possible nor easy to develop. For these reasons, it is important to investigate other mechanisms closely related to the algorithm level approaches, which could lead to decrease the effects caused by noise without neither needing to adapt each specific algorithm.

Several works have claimed that simultaneously using classifiers of different types and combining their predictions may improve classification performance on difficult problems, such as satellite image classification or fingerprint recognition [HHS94, KPD90]. If multiple classifiers are combined, complementing each other, the generalization capabilities of the system may improve. This fact is a key factor in problems with noise since the overfitting to the new characteristics introduced by the noisy examples is avoided [Ten99]. Therefore, the application of these techniques based on the usage of multiple classifiers when working with noisy data would be interesting. Different strategies can be used in order to perform this task. One of them is to combine the predictions of different classifiers that are created with different classification algorithms over the same training set. These techniques are traditionally known as *Multiple Classifier Systems* (MCSs) [HHS94]. Another possibility is to employ decomposition strategies such as, for example, the *One-vs-One* (OVO) [KPD90] decomposition, in which different classifiers are built with same learning algorithm on different training sets that are created considering each pair of classes separately. Thus, the aforementioned modifications at the algorithm level are avoided using these techniques (MCSs or decomposition strategies), while a better performance is expected against noisy data.

Regarding to the data level approaches, *noise filters* are commonly used in order to detect and eliminate class noise [BF99]. Some of them, which are based on the usage of multiple classifiers, show some advantages detecting noisy examples compared to other simpler filters. However, there are some filtering paradigms that offer possibilities that methods based on multiple classifiers found

in the literature still do not incorporate. For example, iterative noise filters [KR07] are based on the interesting idea that the noise detected at a stage does not influence the noise detection at posterior stages. On the other hand, filters that measure the degree of noise of each example [GLD00] allow one to control the conservativeness level of the filter, that is, if they remove a higher or lower number of examples. The combination of these types of noise filters taking advantage of the strengths of the different paradigms should be an important advance in the field of the filtering techniques. Similarly, the application of these powerful noise filters based on multiple classifiers can be extended to classification problems with noise in which the number of examples of each class is not the same (*imbalanced classification*) [HYKL06]. According to our best knowledge, this has not been studied in the literature yet, where many of the approaches are based on combining a re-sampling technique (which somehow balances the number of examples of each class) with a subsequent filtering that is performed with noise filters that are usually very simple attending to the complexity of the problem addressed [Wil72, Tom76].

In light of the aforementioned, it is possible to deduce that noise preprocessing techniques found in the literature are mainly noise filtering methods. Some works show that the effectiveness of the filtering (the precision of the filter to remove noisy examples) mainly depends on the characteristics of the data [WZ08], although this issue has not been deeply studied. It would be useful to detect which are these characteristics of the data that make the filter works better or worse. Furthermore, the possibility of correcting, rather than eliminating, the noisy examples has not received enough attention in the literature yet. Since many existing noise filters create a model to label the training examples (removing those examples in which the predicted and the original class do not match), their transformation into a noise corrector may be immediate by replacing the original class of the example for that predicted by the filter.

Note that both types of techniques, data level and algorithm level approaches, mainly focus on class noise, whereas attribute noise has not been enough studied because its detection and treatment are more complex. Even though class noise is more disruptive to classification performance [ZW04], attribute noise is more common than class noise and therefore, it is interesting to deepen in its study. In the case of those attributes with less importance to build the model, there are a series of techniques based on the *k-Nearest Neighbors* (k-NN) [McL04] classifier aiming to cope the problems that they produce. These are formally known as *feature weighting methods* [WAM97]. Their purpose is to weight the different attributes in order to help building the classifier. Most of these methods do not show a very good performance in many noisy classification problems and new proposals in this field should be of great interest.

Finally, the study of evaluation measures of the classifiers dealing with noisy data has been slightly studied in the literature and lacks of consensus. Therefore, its analysis and the proposal of measures of evaluation of the behavior of classifiers in these circumstances is very important to progress in this field.

This thesis develops the open issues raised in the introduction, with the aim of deepening in the field of classification with noisy data. After this introductory section, the next section (Section 2.) is devoted to describe in detail the four main areas related: noisy data in classification (Section 2.1), noise filters (Section 2.2), the usage of multiple classifiers and decomposition strategies in classification (Section 2.3) and feature weighting techniques (Section 2.4). All of them are fundamental areas for defining and describing the proposals presented as a results of this thesis.

Next, the justification of this memory will be given in Section 3., describing the open problems addressed. The objectives pursued when tackling them are described in Section 4.. Section 5. presents a summary on the works that compose this memory. A joint discussion of results is provided in Section 6., showing the connection between each of the objectives and how they have

been reached. A summary of the conclusions drawn is provided in Section 7.. Finally, in Section 8. we point out several open future lines of work derived from the results achieved.

The second part of the memory is constituted by eight journal publications and two conference publications. These publications are the following:

- Sáez J. A., Galar M., Luengo J., and Herrera F. (2013) Tackling the Problem of Classification with Noisy Data using Multiple Classifier Systems: Analysis of the Performance and Robustness. *Information Sciences* 247: 1–20
- Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems* 38(1): 179–206
- Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Statistical computation of feature weighting schemes through data estimation for nearest neighbor classifiers. *Pattern Recognition* 47(12): 3941–3948
- Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Improving the behavior of the nearest neighbor classifier against noisy data with feature weighting schemes. In *Hybrid Artificial Intelligence Systems*, volumen 8480 of *Lecture Notes in Computer Science*, pp. 597–606. Springer International Publishing
- Sáez J. A., Luengo J., and Herrera F. (2013) Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition* 46(1): 355–364
- Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) INFFC: An iterative noise filter based on the fusion of classifiers with noise sensitiveness control (submitted)
- Sáez J. A., Luengo J., Stefanowski J., and Herrera F. (2014) SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, doi: 10.1016/j.ins.2014.08.051. *Information Sciences* (in press)
- Sáez J. A., Luengo J., Shim S., and Herrera F. (2014) Class Noise Reparation by an Aggregated Noise Filter Ensemble Voting Algorithm (submitted)
- Sáez J. A., Luengo J., and Herrera F. (2011) Fuzzy Rule Based Classification Systems versus Crisp Robust Learners Trained in Presence of Class Noise's Effects: a Case of Study. In *11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), Córdoba (Spain)*, pp. 1229–1234
- Sáez J. A., Luengo J., and Herrera F. (2014) Evaluating the classifier behavior with noisy data considering performance and robustness: the Equalized Loss of Accuracy measure (submitted)

Introducción

En las últimas décadas, el avance de la tecnología ha conllevado un enorme aumento en la cantidad de datos a procesar en aplicaciones de muy diversos campos (como la medicina o la economía). Este hecho ha supuesto la necesidad de nuevas técnicas para gestionar y analizar esta ingente cantidad

de datos. Dado que estas tareas sobrepasan las capacidades humanas, ha sido imprescindible el desarrollo de procedimientos automáticos para su abordaje [BHS09, Bra07].

Así pues, tanto la adquisición de nuevo conocimiento a partir de grandes cantidades de datos como su análisis se han convertido en tareas de gran relevancia en la actualidad. En la literatura especializada, estas labores se engloban normalmente dentro del proceso de Descubrimiento de Conocimiento en Bases de Datos (*Knowledge Discovery in Databases*, KDD) [Fri97, TSK05]. El conocimiento obtenido mediante el proceso de KDD tiene como requisito fundamental la utilidad, de forma que este conocimiento tenga sentido para el problema abordado. Las fases principales del KDD se describen a continuación (ver Figura 2):

1. **Selección de datos.** Esta etapa se realiza a partir del conocimiento previo de expertos en el problema a tratar.
2. **Preprocesamiento de datos.** Éste incluye la limpieza de los datos (tratamiento de errores/corrupciones) y su reducción (selección tanto de las características como de los ejemplos más representativos del problema) para facilitar la cuarta etapa de *minería de datos*.
3. **Transformación de datos.** Se transforman los datos disponibles en una representación estructurada que refleje el conocimiento del problema presente en los datos iniciales.
4. **Minería de datos.** Se construyen patrones de conocimiento a partir del análisis de los datos.
5. **Interpretación de datos.** Esta etapa tiene como objetivo la comprensión de los patrones obtenidos, de forma que puedan ser útiles para los usuarios.

La fase principal del proceso de KDD es su cuarta etapa de *minería de datos* [TSK05]. Ésta se centra en la identificación de patrones y la predicción de relaciones entre los datos. Normalmente, las técnicas de minería de datos se clasifican en *descriptivas* (cuando se usan para descubrir patrones entre los datos) y *predictivas* (cuando se usan para predecir el comportamiento de un modelo a través del análisis de los datos). Estos procesos (descriptivos y predictivos) se llevan a cabo normalmente mediante el uso de algoritmos de *aprendizaje automático* [Alp10], definidos como mecanismos capaces de inducir conocimiento a partir de datos. En función de la información de la que disponga el algoritmo de aprendizaje automático para llevar a cabo la extracción de conocimiento, se pueden considerar principalmente dos tipos de problemas:

1. **Aprendizaje supervisado.** Es aquel que pretende establecer una relación entre las variables de entrada y salida (variable objetivo), siendo todas conocidas, con el fin de predecir la salida de nuevos ejemplos cuya variable objetivo es desconocida. En función del tipo de variable objetivo, este problema se subdivide en dos problemas diferentes: (i) *clasificación* [DHS00], donde los valores de la variable objetivo (conocida como clase) son discretos; y (ii) *regresión* [CM98], donde los valores de la variable objetivo son continuos.
2. **Aprendizaje no supervisado.** En éste, donde los valores de la variable objetivo no son conocidos, se pretenden describir las relaciones entre los datos. Se divide principalmente en dos grupos de problemas: (i) *clustering* [Har75], donde se desea separar los datos en distintos grupos que comparten caractersticas similares; y (ii) *asociación* [AIS93], donde se trata de identificar las relaciones entre datos transaccionales.

Los métodos de clasificación aprenden a categorizar elementos (más conocidos como *ejemplos*) dentro de varias clases predefinidas mediante la creación de un modelo, conocido como *clasificador*.

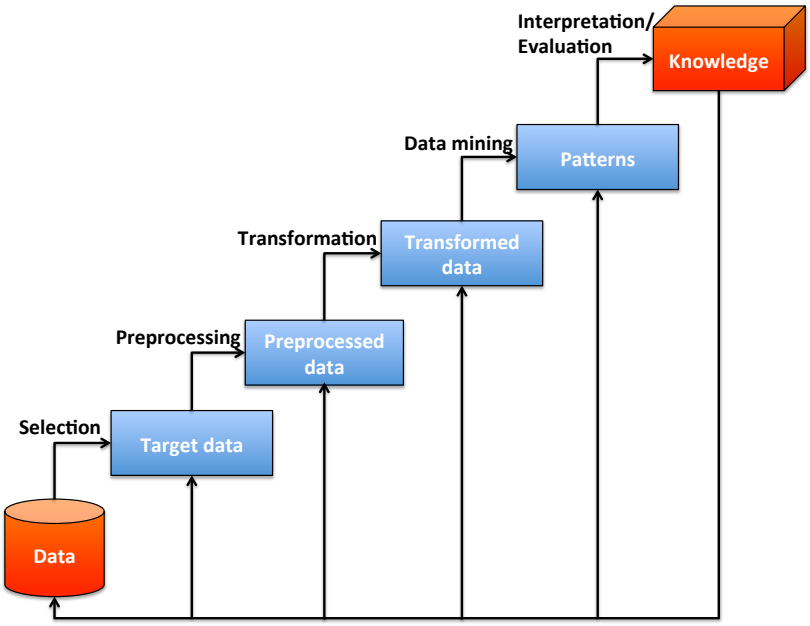


Figure 2: El proceso de KDD.

Éste se construye a partir de un conjunto de ejemplos (el conjunto de *entrenamiento*) y después es aplicado para predecir la clase de los ejemplos de otro conjunto (el conjunto de *test*) que no han sido previamente utilizados en la fase de aprendizaje. El *rendimiento* del clasificador se puede medir de distintas formas, entre las que cabe mencionar la *precisión* (entendida como la confianza del clasificador aprendido), el *tiempo de aprendizaje* requerido para construir el clasificador o su *interpretabilidad* (la claridad y credibilidad del clasificador desde el punto de vista humano).

Dado que el clasificador se construye a partir de un conjunto de datos de entrenamiento, su rendimiento se ve directamente influido por la calidad de dichos datos. Esta calidad depende de varios componentes [WSF95] como, por ejemplo, los procedimientos de obtención y entrada de datos, que están inherentemente sujetos a errores. Así pues, los conjuntos de datos que se usan normalmente contienen datos con corrupciones, formalmente conocidas como *ruido* [ZW04], que pueden perjudicar a los clasificadores construidos y, por tanto, a las interpretaciones y decisiones obtenidas a partir de ellos.

En base a los dos tipos de variables existentes en un problema de clasificación (variables de entrada o *atributos* y variable objetivo o *clase*), se distinguen dos tipos de ruido [Wu96]:

1. **Ruido de clase.** Éste ocurre cuando un ejemplo está incorrectamente etiquetado.
2. **Ruido de atributos.** Éste se refiere a corrupciones en los valores de uno o más atributos en el conjunto de datos. Algunos ejemplos de ruido de atributos son: valores de atributos erróneos, valores de atributos perdidos o desconocidos (conocidos como *missing values*, MV [JL13]) y atributos irrelevantes o de escasa importancia para construir el clasificador.

Debido a la perdida de precisión producida por el ruido, en la literatura especializada se han propuesto tradicionalmente dos formas de mitigar sus efectos:

1. **Enfoques a nivel de algoritmos.** Éstos se basan principalmente en la adaptación de

los algoritmos con el fin de manejar de forma adecuada el ruido [Qui93]. Estos métodos se conocen como *robust learners* y se caracterizan por verse menos afectados por el ruido.

2. **Enfoques a nivel de datos**. Mediante el preprocesamiento de los datos se intentan eliminar o corregir los ejemplos con ruido [BF99].

Si bien ambos tipos de enfoques pueden proporcionar buenos resultados, existen algunas limitaciones. Así, por ejemplo, la adaptación de algoritmos depende del algoritmo concreto a modificar. Por tanto, el mismo resultado no es directamente aplicable a otros algoritmos de aprendizaje, ya que el beneficio proviene de la propia adaptación. Este enfoque requiere el cambio de un método existente, que no es siempre posible ni fácil de realizar, y puede estar orientado a un tipo de ruido con unas características determinadas. Por estos motivos, es interesante investigar mecanismos cercanos a los enfoques a nivel de algoritmos que reduzcan los efectos del ruido sin la necesidad de adaptar cada algoritmo concreto ni tener que hacer supuestos sobre las características del ruido.

Algunos trabajos exponen que usar simultáneamente distintos clasificadores y combinar sus predicciones puede mejorar el rendimiento en problemas de clasificación difíciles como son la clasificación de imágenes de satélites o el reconocimiento de huellas dactilares [HHS94, KPD90]. Al combinar múltiples clasificadores, de forma que se complementen unos a otros, es posible mejorar las capacidades de generalización del sistema, lo cual es una cuestión clave en problemas con ruido ya que se evita el sobreajuste a las nuevas características introducidas por los ejemplos con ruido [Ten99]. Así pues, sería interesante la aplicación de estas técnicas basadas en el uso de múltiples clasificadores cuando los datos presentan ruido, para lo cual es posible seguir diferentes estrategias. Una de ellas es combinar las predicciones de distintos clasificadores creados con diferentes algoritmos de clasificación sobre el mismo conjunto de entrenamiento. Estas técnicas son conocidas tradicionalmente como *Sistemas de Múltiples Clasificadores* (*Multiple Classifier Systems*, MCSs) [HHS94]. Otra posibilidad es emplear estrategias de descomposición como puede ser, por ejemplo, la estrategia Uno-frente-Uno (*One-vs-One*, OVO) [KPD90], la cual construye diferentes clasificadores con el mismo algoritmo de aprendizaje sobre diferentes conjuntos de entrenamiento creados considerando cada par de clases por separado. De la forma que sea (MCSs o estrategias de descomposición), utilizando estas técnicas se evitan las modificaciones a nivel de algoritmos que se han comentado, mientras que se espera un mejor comportamiento frente a los datos con ruido.

En cuanto a los enfoques a nivel de datos, el uso de múltiples clasificadores ya ha sido propuesto en algunos trabajos con el fin de detectar y eliminar el ruido de clase [BF99]. Estas técnicas se conocen como *filtros del ruido*. Los filtros del ruido basados en múltiples clasificadores muestran algunas ventajas en cuanto a la capacidad de detección del ruido frente a otros que pueden considerarse más sencillos. Sin embargo, hay algunos paradigmas de filtrado que ofrecen posibilidades que los métodos basados en múltiples clasificadores existentes en la literatura todavía no incorporan. Por ejemplo, los filtros de ruido iterativos [KR07] se basan en la interesante idea de que el ruido detectado en una etapa no influya en la detección de ruido en etapas posteriores. Por otro lado, los filtros que utilizan medidas del grado de ruido de cada ejemplo [GLD00] permiten controlar de forma sencilla el nivel de conservación del filtro, es decir, si se eliminan un número mayor o menor de ejemplos. La combinación de estos tipos de filtros de ruido con el objetivo de aprovechar las ventajas de los diferentes paradigmas sería un importante avance dentro de este campo. De igual modo, la aplicación de estos potentes filtros del ruido basados en múltiples clasificadores puede extenderse a problemas de clasificación con ruido donde el número de ejemplos de cada clase es desigual (*problemas no balanceados*) [HYKL06]. Esto, de acuerdo con nuestro conocimiento, todavía no se ha estudiado en la literatura, donde muchos de los enfoques se basan en combinar una técnica de remuestreo (que balancea en cierto grado el número de ejempos de cada clase) con un filtrado

posterior realizado con filtros que suelen ser muy sencillos y no tan potentes, en determinadas ocasiones, como podría exigir un problema de tal complejidad [Wil72, Tom76].

Como se ha podido deducir de lo anteriormente expuesto, las técnicas de preprocesamiento del ruido que se encuentran en la literatura son principalmente métodos de filtrado, es decir, son técnicas que eliminan ejemplos con ruido en el conjunto de entrenamiento. Algunos trabajos muestran que la eficacia del filtrado (la precisión del filtro al eliminar los ejemplos con ruido) depende fundamentalmente de las características de los datos con los que se trabaja [WZ08], aunque esto no ha sido profundamente estudiado. Sería útil detectar cuáles son estas carceterístcas de los datos que pueden hacer que el filtrado funcione mejor o peor. Por otro lado, la posibilidad de corregir, en vez de eliminar, los ejemplos con ruido, tampoco ha sido aún lo suficientemente estudiada en la literatura. Dado que una de las características de muchos de los filtros de ruido existentes es que crean un modelo para etiquetar los ejemplos de entrenamiento (de forma que eliminan un ejemplo si la clase predicha y la original en el ejemplo no coinciden), su transformación a un corrector de ruido es inmediata sin más que sustituir la clase del ejemplo por la predicha por el filtro.

Debe notarse que, tanto los enfoques a nivel de datos como los enfoques a nivel de algoritmos que se encuentran en la literatura, se centran principalmente en el ruido de clase, mientras que el ruido de atributos no ha sido tan estudiado debido a que su detección y tratamiento son más complejos. Ya que el ruido de atributos es bastante más común que el de clase, es de interés profundizar en él. En el caso de aquellos atributos con escasa importancia e información para la construcción del modelo, existen una serie de técnicas basadas en el clasificador *k-Nearest Neighbors* (k-NN) [McL04] cuyo objetivo es hacer frente a los problemas que producen. Son los conocidos como métodos de ponderación de características o *feature weighting methods* [WAM97], cuyo objetivo es ponderar los distintos atributos para ayudar a construir el modelo. La mayoría de estos métodos no muestran muy buen comportamiento en gran variedad de problemas de clasificación y nuevas propuestas en este campo serían de gran interés.

Finalmente, el estudio de medidas de evaluación de los clasificadores cuando se trabaja con datos con ruido no ha sido suficientemente abordado en la literatura y carece de consenso. Por tanto, su análisis y la propuesta de medidas de evaluación del comportamiento de los clasificadores en estas circunstancias es muy importante para avanzar en este campo.

La presente tesis prentende desarrollar las cuestiones abiertas planteadas durante la introducción, con el objetivo de profundizar en el campo de la clasificación con datos con ruido. Tras esta sección introductoria, la siguiente sección (Sección 2.) está dedicada a describir en detalle cuatro principales áreas relacionadas: los datos con ruido en clasificación (Sección 2.1), los filtros de ruido (Sección 2.2), el uso de múltiples clasificadores y estrategias de descomposición en clasificación (Sección 2.3) y las técnicas de ponderación de características (Sección 2.4). Todas ellas son áreas fundamentales para definir y describir las propuestas presentadas como resultado de esta tesis.

Después, la justificación de esta memoria se presenta en la Sección 3., describiendo los problemas abiertos abordados. Los objetivos perseguidos al abordar dichos problemas son descritos en la Sección 4.. La Sección 5. presenta un resumen de los trabajos que componen esta memoria. Se aporta una discusión conjunta de resultados en la Sección 6., mostrando la conexión entre cada uno de los objetivos y como ha sido alcanzado cada uno de ellos. En la Sección 7. se incluye un resumen de las conclusiones alcanzadas. Finalmente, en la Sección 8. se destacan varias líneas de trabajo futuro abiertas, derivadas de los resultados alcanzados.

La segunda parte de la memoria se constituye de ocho publicaciones en revistas y otras dos en congresos. Estas publicaciones son las siguientes:

- Sáez J. A., Galar M., Luengo J., and Herrera F. (2013) Tackling the Problem of Classification with Noisy Data using Multiple Classifier Systems: Analysis of the Performance and Robustness. *Information Sciences* 247: 1–20

- Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems* 38(1): 179–206

- Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Statistical computation of feature weighting schemes through data estimation for nearest neighbor classifiers. *Pattern Recognition* 47(12): 3941–3948

- Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Improving the behavior of the nearest neighbor classifier against noisy data with feature weighting schemes. In *Hybrid Artificial Intelligence Systems*, volumen 8480 of *Lecture Notes in Computer Science*, pp. 597–606. Springer International Publishing

- Sáez J. A., Luengo J., and Herrera F. (2013) Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition* 46(1): 355–364

- Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) INFFC: An iterative noise filter based on the fusion of classifiers with noise sensitiveness control (submitted)

- Sáez J. A., Luengo J., Stefanowski J., and Herrera F. (2014) SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, doi: 10.1016/j.ins.2014.08.051. *Information Sciences* (in press)

- Sáez J. A., Luengo J., Shim S., and Herrera F. (2014) Class Noise Reparation by an Aggregated Noise Filter Ensemble Voting Algorithm (submitted)

- Sáez J. A., Luengo J., and Herrera F. (2011) Fuzzy Rule Based Classification Systems versus Crisp Robust Learners Trained in Presence of Class Noise's Effects: a Case of Study. In *11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), Córdoba (Spain)*, pp. 1229–1234

- Sáez J. A., Luengo J., and Herrera F. (2014) Evaluating the classifier behavior with noisy data considering performance and robustness: the Equalized Loss of Accuracy measure (submitted)

2. Preliminaries

In this section we describe all the background information involved in this thesis. Firstly, Section 2.1 presents an introduction to noisy data in the field of classification. Secondly, Section 2.2 describes some previous works on noise filters. Thirdly, Section 2.3 shows a snapshot on classification with multiple classifiers and decomposition strategies. Finally, Section 2.4 provides information about feature weighting methods for classification, its main characteristics and some examples of such techniques that have been proposed in the literature.

2.1 Introduction to noisy data in classification

Data gathered from real-world problems are never perfect and often suffer from corruptions that may hinder the performance of the system in terms of the classification accuracy, building time, size and interpretability of the classifier [ZKS04]. Noise mainly affects the data acquisition and preprocessing phases, having two main sources [ZW04]: implicit errors introduced by measurement tools, such as different types of sensors; and random errors introduced by batch processes or experts when the data are gathered, such as in a document digitalization process. Hence, classification problems containing noise are complex problems and accurate solutions are often difficult to achieve.

A large number of components determine the quality of a data set [WSF95]. Among them, the class labels and the attribute values directly influence the quality of a classification data set. The quality of the class labels refers to whether the class of each example is correctly assigned; otherwise, the quality of the attributes refers to their capability of properly characterizing the examples for classification purposes. Based on these two information sources, two types of noise can be distinguished in a given data set [Wu96] (see Figure 3):

1. **Class noise**. This occurs when an example is incorrectly labeled. Class noise can be attributed to several causes, such as subjectivity during the labeling process, data entry errors, or inadequacy of the information used to label each example. Two types of class noise can be distinguished: (i) *contradictory examples* – duplicate examples have different class labels –, and (ii) *misclassifications* – examples that are labeled as a class different from the real one.

2. **Attribute noise**. This refers to corruptions in the values of one or more attributes. Examples of attribute noise are: erroneous attribute values, missing or unknown attribute values and irrelevant attributes to build the classifier.

In this thesis, class noise refers to misclassifications, whereas attribute noise generally refers to erroneous attribute values (even though those attributes less important to build the model are also studied) because they are the most common in real-world data [ZW04].

Noise hinders the knowledge extraction from the data and spoils the models obtained using that noisy data when they are compared to the models learned from clean data from the same problem, which represent the real implicit knowledge of the problem [ZW04]. In this sense, *robustness* [Hub81] is the capability of an algorithm to build models that are insensitive to data corruptions and suffer less from the impact of noise; that is, the more robust an algorithm is, the more similar the models built from clean and noisy data are.

Checking the effect of noisy data on the performance of classifier learning algorithms is necessary to improve their reliability and has motivated the study of how to generate and introduce noise into the data. Noise generation can be characterized by three main characteristics:

1. **The place where the noise is introduced.** Noise may affect the input attributes or the output class, impairing the learning process and the resulting model.

2. **The noise density distribution.** The way in which the noise is present can be, for example, uniform or Gaussian.

3. **The magnitude of generated noise values.** The extent to which the noise affects the data set can be relative to each data value of each attribute, or relative to the minimum, maximum and standard deviation for each attribute.

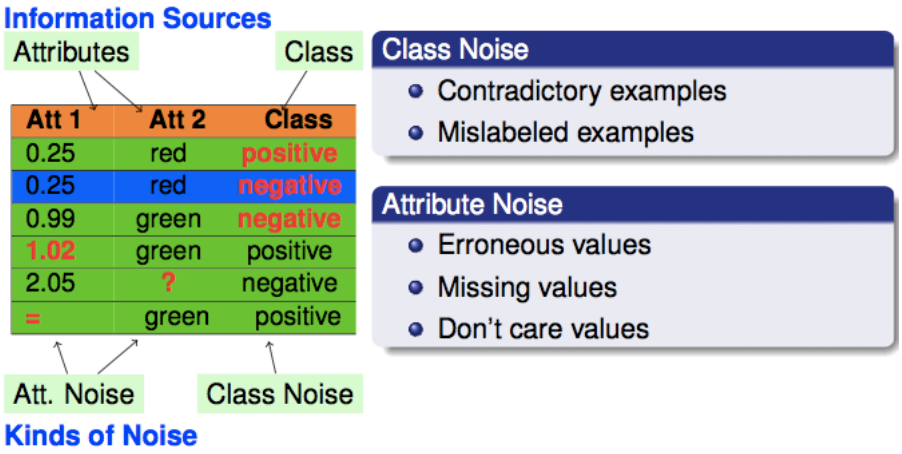


Figure 3: Information sources and types of noise in a classification data set.

Based on these three characteristics, the two types of noise (class and attribute noise) have been modeled in the literature using four main noise schemes; in such a way, the presence of those types of noise will allow one to simulate the behavior of the classifiers in these two scenarios:

1. **Class noise** usually occurs on the boundaries of the classes, where the examples may have similar characteristics –although it can occur in any other area of the domain. In this thesis, class noise is introduced using an *uniform class noise scheme* [Ten99] (randomly corrupting the class labels of the examples) and a *pairwise class noise scheme* [ZWC03] (labeling examples of the majority class with the second majority class). Considering these two schemes, noise affecting any pair of classes and only the two majority classes are simulated, respectively.

2. **Attribute noise** can originate from several sources, such as transmission constraints, faults in sensor devices, irregularities in sampling and transcription errors [Ten04]. The erroneous attribute values can be totally unpredictable, i.e., random, or imply a low variation with respect to the correct value. In the literature, the *uniform attribute noise scheme* [ZWY04] and the *Gaussian attribute noise scheme* are used in order to simulate each one of the possibilities, respectively.

Since errors in real-world data sets are therefore common, techniques that eliminate noise or reduce its impact are needed [Wu96]. Two main alternatives have been proposed in the literature to deal with noisy data:

- **Algorithm level approaches** [Qui93, Coh95]. Also known as *robust learners*, these are techniques characterized by being less influenced by noisy data. Examples of a robust learner are C4.5 [Qui93] or RIPPER [Coh95]. These classifiers have been adapted to properly handle the noise. Thus, for example, C4.5 uses pruning strategies to reduce the chances that the trees are overfitting due to noise in the training data.

- **Data level approaches** [BF99, KR07]. The most well-known type of methods within this group is that of the *noise filters*, described in the next section. They identify noisy instances which can be eliminated from the training data.

2.2 Noise filters

Noise filters are preprocessing methods to detect and eliminate noisy examples in the training set [BF99]. The result of noise elimination in preprocessing is a reduced and cleaned training set which is then used as an input to a machine learning algorithm. The separation of noise detection and learning has the advantage that noisy instances do not influence the classifier building design [GLD00].

Noise filters are generally oriented to deal with instances with class noise from the training data. Elimination of such instances has been shown to be advantageous [GBLG99]. However, the elimination of instances with attribute noise seems counterproductive [ZW04] since they still contain valuable information in other attributes which can help to build the classifier.

Some of these filters are based on the computation of measures over the training set. For example, the proposal in [GLD00] is based on the elimination of noisy examples that reduce the *Complexity of the Least Complex Correct Hypothesis* value of the training set.

Other filtering methods are based on the sensitivity of the k-NN classifier [McL04] to noisy data, particularly if k is low [KK07]. Within this group, the *Edited Nearest Neighbor* (ENN) [Wil72] noise filter is one of the most well-known methods. It removes those examples which class does not agree with that of the majority of its k nearest neighbors. One of its variations is the *All k-Nearest Neighbors* (AllKNN) [Tom76] filter. It applies the NN rule k times varying the number of neighbors considered between 1 to k. If one instance is misclassified by the NN rule, it is registered as removable from the training set. Then all those that do meet the criteria are removed at once.

An interesting group of filtering techniques are based on the predictions of classifiers. The *Classification Filter* (CF) proposed in [GBLG99] performs a partition of the training set into n subsets, then a set of classifiers is trained from the union of any $n-1$ subsets; the classifiers are used to classify the examples in the excluded subset, eliminating the examples that are incorrectly classified. This filter has the risk to remove too many instances due to the usage of a single classifier. To solve this problem, ensembles of classifiers are used to identify mislabeled instances; the proposals of [BF99, KR07] are two of the most representative and well known methods within this field (described hereafter).

The *Ensemble Filter* (EF), proposed in [BF99], uses a set of different learning algorithms to remove the potentially noisy instances. This filter is based on the idea that, if some examples have been mislabeled and it is assumed that the label errors are independent of the particular classifiers learned from the data, collecting predictions from different classifiers could provide a better estimation of mislabeled examples than collecting information from a single classifier. In order to perform the removal of noisy examples, first, the training data is classified using an n-fold cross-validation with each classification algorithm and then, the noisy examples are identified using a voting scheme (consensus, which removes an example if it is misclassified by all the classifiers; or majority, which removes an example if it is misclassified by more than half of the classifiers).

Another example of ensemble filters is the *Iterative-Partitioning Filter* (IPF) [KR07], which proposes a similar technique, but removes the noisy data iteratively using several classifiers built with the same learning algorithm. The authors of this method claimed that a iterative elimination of noisy examples implies that the examples removed in one iteration do not influence the detection in subsequent ones, resulting in a more accurate noise filtering, Thus, in each iteration, the current training data set is split into n equal sized subsets and the C4.5 classifier is built over each of these n subsets to evaluate the whole training set. Then, the incorrectly labeled examples are removed from it (according to the majority or consensus scheme) and a new iteration is started.

2.3 Multiple classifiers and decomposition strategies for classification tasks

Given a set of problems, finding the best overall classification algorithm is sometimes difficult because some classifiers may excel in some cases and perform poorly in others. Moreover, even though the optimal match between a learning method and a problem is usually searched, this match is generally difficult to achieve and perfect solutions are rarely found for complex problems [HHS94]. Thus, several motivations to combine several classifiers are found in the literature:

- To avoid the necessity of choosing a specific learning method for a concrete classification problem; all of them might be used, taking advantage of the strengths of each method, while avoiding its weaknesses.
- To avoid the choice of some arbitrary but important initial condition, e.g., those involving the parameters of the learning method.
- To introduce some randomness to the training process in order to obtain different alternatives that can be combined to improve the results obtained by the individual classifiers.

There are several strategies to use more than one classifier for a single classification task [HHS94]. Among them, *parallel approaches* focus the majority of classifier combination research due to its simplicity and the fact that they enable one to take advantage of the aforementioned factors. In this approach, all available classifiers are used for the same input example in parallel. The outputs from each classifier are then combined to obtain the final prediction. Thus, some of the most well-known decisions combination proposals are the following:

- **Majority vote (MAJ)** [MKK87]. This is a simple but powerful approach, where each classifier gives a vote to the predicted class and the most voted one is chosen as the output.
- **Weighted majority vote (W-MAJ)** [SG84]. Similarly to MAJ, each classifier gives a vote for the predicted class, but in this case, the vote is weighted depending on the competence (accuracy) of the classifier in the training phase.
- **Naive Bayes (NB)** [TMM$^+$81]. This method assumes that the base classifiers are mutually independent. Hence, the predicted class is that one obtaining the highest posterior probability. In order to compute these probabilities, the confusion matrix of each classifier is considered.
- **Behavior-Knowledge Space (BKS)** [HS95]. This is a multinomial method that indexes a cell in a look-up table for each possible combination of classifiers outputs. A cell is labeled with the class to which the majority of the instances in that cell belong to. A new instance is classified by the corresponding cell label; in case that for the cell is not labeled or there is a tie, the output is given by MAJ.

The most common strategy to train all the base classifiers is using the same training data set with all of them and to compute the parameters of the aggregation methods, as it is recommended in [Kun04]. Using a separate set of examples to obtain such parameters can imply some important training data to be ignored and this fact is generally translated into a loss of accuracy of the final system built.

Another possibility to use multiple classifiers for a classification task is to employ the same classification algorithm over different data samples taken from the same training data and, then,

training a classifier in each of these samples and combining their predictions. Thus, if the classification problem has several classes (*multi-class classification problem*), a way to build these different data samples is to decompose the multi-class problem into a set of easier to solve binary subproblems, aiming to reduce the complexity of the original problem. Each one of the classes of the binary subproblem will be the joint of one or more classes of the original problem. This methodology is formally known as *decomposition strategies* [GFB+11], [Fur02]. Several motivations for the usage of these binary decomposition strategies in multi-class classification problems can be found in the literature:

- The separation of the classes becomes easier (less complex), since less classes are considered in each subproblem [MM96], [Fur02].
- Classification algorithms, whose extension to multi-class problems is not easy, can address multi-class problems using decomposition techniques [Fur02].
- Decomposition allows one to easily parallelize the classifier learning, since the binary subproblems are independent and can be solved with different processors.

Dividing a problem into several new subproblems, which are then independently solved, implies the need of a second phase where the outputs of each problem need to be aggregated. Therefore, decomposition includes two steps:

1. *Problem division.* The problem is decomposed into several binary subproblems which are solved by independent binary classifiers, called *base classifiers*. The most studied decomposition strategies in the literature are: *One-vs-One* (OVO) [KPD90], which trains a classifier to distinguish between each pair of classes, and *One-vs-All* (OVA) [AMMR95], which trains a classifier to distinguish each class from all other classes.
2. *Combination of the outputs* [GFB+11]. The different outputs of the binary classifiers must be aggregated in order to output the final class prediction.

Among the decomposition strategies, OVO is the most common approach due to the several advantages shown in the literature with respect to OVA [GFB+11], [Fur02]:

- OVO creates simpler borders between classes than OVA.
- OVO generally obtains a higher classification accuracy and a shorter training time than OVA because the new subproblems are easier and smaller.
- OVA has more of a tendency to create imbalanced data sets which can be counterproductive.
- The application of the OVO strategy is widely extended and most of the software tools considering binarization techniques use it as default.

The OVO decomposition strategy consists of dividing a classification problem with M classes into $M(M-1)/2$ binary subproblems. A classifier is trained for each new subproblem only considering the examples from the training data corresponding to the pair of classes (λ_i, λ_j) with $i < j$ considered. When a new instance is going to be classified, it is presented to all the the binary classifiers. This way, each classifier discriminating between classes λ_i and λ_j provides a confidence

degree $r_{ij} \in [0, 1]$ in favor of the former class (and hence, r_{ji} is computed by $1 - r_{ij}$). These outputs are represented by a score matrix R:

$$R = \begin{pmatrix} - & r_{12} & \cdots & r_{1M} \\ r_{21} & - & \cdots & r_{2M} \\ \vdots & & & \vdots \\ r_{M1} & r_{M2} & \cdots & - \end{pmatrix} \tag{I.1}$$

The final output is derived from the score matrix by different aggregation models. The most used and simplest combination is the application of a voting strategy:

$$Class = \underset{i=1,\ldots,M}{\arg\max} \sum_{1 \leq j \neq i \leq M} s_{ij} \tag{I.2}$$

where s_{ij} is 1 if $r_{ij} > r_{ji}$ and 0 otherwise. Therefore, the class with the largest number of votes will be predicted. This strategy has proved to be competitive with different classifiers obtaining similar results in comparison with more complex strategies [GFB+11].

2.4 Feature weighting schemes in nearest neighbor classification

The Nearest Neighbor (NN) classifier [CH67] is one of the most widely used methods in classification tasks due to its simplicity and good behavior in many real-world domains [WKRQ+07]. It is a nonparametric classifier which simply uses the full training data set to establish a classification rule, based on the most similar or nearest training instance to the query example. Data preparation [Pyl99] provides a number of ways to improve the performance of NN, such as Prototype Selection [GDCH12] or Feature Selection [LM07, XZB14]. A different, yet powerful approach is Feature Weighting [WAM97].

The main objective of Feature Weighting methods is to reduce the sensitivity of the NN rule to redundant, irrelevant or noisy features. This is achieved by modifying its similarity function [CGG+09] with the inclusion of weights. These weights can be regarded as a measure of how useful a feature is with respect to the final classification task. The higher a weight is, the more influence the associated feature will have in the decision rule used to compute the classification of a given example. Therefore, an adequate scheme of weights could be used to highlight the best features of the domain of the problem, diminishing the impact of redundant, irrelevant and noisy ones. Thus, the accuracy of the classifier could be greatly improved if a proper selection of weights is made.

In the case of the NN classifier, most of the techniques developed to include Feature Weighting schemes have been focused on incorporating the weights in the distance measure, mainly to Euclidean distance (see Equation I.3, where X and Y are two instances and M is the number of features that describes them). In spite of its simplicity, the usage of Euclidean distance has been preferred in many research approaches, since it is easy to optimize and shows a good discriminative power in most classification tasks. In fact, it is the most commonly used similarity measure in the instance based learning field [AKA91].

$$d(X, Y) = \sqrt{\sum_{i=0}^{M} (x_i - y_i)^2} \tag{I.3}$$

Feature Weighting methods often extend this definition through the inclusion of weights associated with each feature (W_i, usually $W_i \in [0,1]$). These modify the way in which the distance measure is computed (Equation I.4), increasing the relevance (the squared difference between feature's values) of those features with greater weights associated with them (near to 1.0).

$$d_w(X,Y) = \sqrt{\sum_{i=0}^{M} W_i \cdot (x_i - y_i)^2} \tag{I.4}$$

The application of this technique to the NN classifier has been widely addressed. The most complete study undertaken to this end can be found in [WAM97], in which a review of several Feature Weighting methods for Lazy Learning algorithms [Aha97] is presented (with most of them applied to improve the performance of the NN rule). In this review, Feature Weighting techniques are categorized by several dimensions, regarding the weight learning bias, the weight space (binary or continuous), the representation of features employed, their generality and their degree of employment of domain specific knowledge.

A wide range of classical Feature Weighting techniques are available in the literature, both classical (see [WAM97] for a complete review) and recent [PV06, FI08]. The most well known compose the family of *Relief-based* algorithms. The Relief algorithm [KR92] has been widely studied and modified, producing several interesting variations of the original approach. Some of them [SK03, Sun07] are based on ReliefF [Kon94], which is the first adaptation of Relief as a Feature Weighting approach. In addition to these approaches, Feature Weighting methods are also very useful when considered as a part of larger supervised learning schemes. In these approaches, Feature Weighting can be regarded as an improved version of Feature Selection. Again, if the weights scheme is properly chosen, Feature Weighting can play a decisive role in enhancing the performance of the NN classifier in these techniques [DTGH12].

3. Justification

As it has been shown in the previous sections, the presence of noise in data is a common problem that produces several negative consequences in classification problems. Given the negative effects produced by noise, the need of techniques to deal with it and the analysis of their impact and characteristics have always had a notable relevance in the literature.

However, if new studies on the framework of noisy data in classification are desired to be developed, the following key issues should be taken into consideration:

- There are two main approaches to deal with noisy data in classification: algorithm level (robust learners) and data level approaches (noise preprocessing techniques) [ZW04].
- Algorithm level approaches depend on the classification algorithm and require to adapt an existing method, which neither is always possible nor easy to develop. In many cases, they also depend on the characteristics of the noise. For these reasons, it is important to investigate other mechanisms, which could lead to decrease the effects caused by noise without adapting each specific algorithm or having to make assumptions about the type of noise present in the data.
- Furthermore, most of the works in the literature focus on the class noise problem, whereas the attribute noise problem is less studied because its treatment is usually more complex.

Since attributes with missing values have been widely studied in the literature [ZW04], new research efforts on the erroneous attribute values and attributes with low influence in the classifier building will imply a higher novelty in this field.

- Data level approaches, concretely noise filters, usually depend on the characteristics of the data and the noise [WZ08]. However, these characteristics, which determine the efficacy of the noise filters, are not enough concreted in any work of the literature. Furthermore, even though there are many proposals of paradigms of noise filtering in standard classification (for example, classification filtering, distance-based filtering or ensemble-based filtering), an interesting trend would be to propose advanced noise filtering techniques trying to take advantages of the strengths of the different paradigms. It would be also interesting to develop new proposals of advanced noise filtering techniques adapted to the characteristic of classification problem, such as the degree of balance among the classes (imbalanced classification), when the data are mainly characterized by the presence of noisy data.

- Most of the data level approaches are focused on the elimination of noisy examples from the training data by means of the well-known noise filters. However, instance filtering can be harmful as it may eliminate more examples than necessary or produce an information loss [KB81]. For this reason, a direct alternative would be to correct the class label of those suspicious examples that are potentially noisy when possible, trying to avoid the aforementioned problem of noise filtering techniques.

- Finally, there are few proposals of evaluation metrics on the classifier behavior when the classification problem is mainly characterized by the presence of noisy data. The concept of robustness must be also considered in such evaluation metrics and it would be systematically included in comparisons of classifiers build from noisy data. Therefore, the study of such metrics and new proposals to measure the behavior of classifiers with noisy data are necessary.

All these issues refers to a common topic, which is also the main subject of this thesis: deepen in the problem of noisy data in classification developing different studies and proposals related to the two main ways to deal with it that have been proposed in the literature (algorithm level and data level approaches).

4. Objectives

After studying the current state of all the areas described in the previous sections, it is possible to focus on the actual objectives of the thesis. They will include the research and analysis of the background fields described before, and the development of studies analyzing different aspects related to the problem of noisy data and proposals of advanced models to deal with noise in classification based on their most promising properties of each field. More specifically, the objectives are:

- **To propose and analyze non-preprocessing alternatives to make the classifiers perform better with noisy data, with independence of the classifier selected and the characteristics of the noise.** Since both knowing the real characteristic of the noise and adapting each single classification method are tasks hard to perform at the algorithm level approaches, proposals (such as techniques based on the usage of multiple classifiers) that may increase the performance of classification techniques without modifications of the

methods and assumptions on the characteristics of data will be very useful. These proposals should be well justified and have properties that improve the classification capabilities of classifiers trained from noisy data.

- **To deepen in the attribute noise problem, which is usually less studied than the class noise problem in the literature.** Erroneous attribute values and those with less importance to the classification task are maybe the two less studied types of attribute noise. Feature Weighting schemes are a group of techniques designed to reduce the importance of noisy or irrelevant attributes, whereas the most important features for the classification task are maximized. Therefore, this type of techniques seems to be well adapted to deal with the aforementioned types of attribute noise. Its study and the development of new proposals in this field seem to be very promising approaches to better understand the problem of attribute noise in classification.
- **To analyze the necessity of the application of noise preprocessing techniques based on the properties of the data and to design new competitive noise preprocessing methods.** It would be interesting to determine which are the properties of the data that makes the usage of a noise preprocessing technique to be efficient, since their behavior is basically determine by these [WZ08]. In order to achieve this goal, data complexity metrics [BH06] of the classification problem, which are a recent proposal to represent characteristics of the data which are considered difficult in classification tasks (such as the overlapping among classes, their separability or the linearity of the decision boundaries) could be used. Furthermore, it would be interesting to propose new preprocessing methods considering advanced noise filtering paradigms, to propose new paradigms (for example, based on the correction of noise, instead of its elimination) or to design new filtering strategies adapted to concrete properties of the data (such as the imbalance ratio among the classes).
- **To study and propose evaluation metrics of the behavior of classifiers with noisy data.** In spite of its great relevance, evaluation metrics when training with noisy data have not been enough studied in the literature. Hence, it is important to analyze which are the aspects to evaluate when training a classifier from noisy data (such as the traditional performance, but also the robustness of the classifier) and reflect them into new evaluation metrics under such circumstances.

5. Summary

This thesis is composed by ten works, organized into four different parts. Each part is devoted to pursue one of the objectives described, contributing as a whole in the deepening in the problem of noisy data in classification by means of different studies and proposals.

- Non-preprocessing alternatives based on multiple classifiers to deal with noisy data.
- Feature weighting schemes to treat with attribute noise in nearest neighbor classifiers.
- Data level approaches and proposals to deal with noisy data.
- Evaluation measures of the behavior of classifiers with noisy data.

This section shows a summary of the different proposals presented in this dissertation, describing the associated publications and their main contents.

5.1 Non-preprocessing alternatives based on multiple classifiers to deal with noisy data

In the literature, it is claimed that building several classifiers from noisy training data and combining their predictions is an interesting method of overcoming the individual problems produced by noise in each classifier [HHS94]. This statement is usually not supported by thorough empirical studies considering problems with different types and levels of noise. Neither they show what type of noise is better handled by multiple classifier techniques. Furthermore, in noisy environments, the noise robustness of the methods can be more important than the performance results themselves and, therefore, it must be carefully studied. It should be mentioned that, in real situations, the existence of noise in the data sets is usually unknown. Hence, tools (such as multiple classifier techniques) which are able to manage the presence of noise in the data sets, despite its type or quantity (or unexistence), are of great interest. Furthermore, these strategies can be used with any of the existing classifiers in the literature.

In our first contribution in this topic, we aim to reach conclusions on the aforementioned aspects focusing on the analysis of the behavior, in terms of performance and robustness, of several Multiple Classifier Systems against their individual classifiers when these are trained with noisy data, studying to what extent the behavior of these MCSs depends on that of the individual classifiers. In order to accomplish this study, several classification algorithms (SVM [CV95], C4.5 [Qui93] and k-NN [McL04]), of varying noise robustness, will be chosen and compared with respect to their combination. All these classification algorithms will be combined using different decisions combination methods [MKK87, SG84, TMM$^+$81, HS95] to create MCSs of different sizes and characteristics. Two forms to create diversity will be considered: (i) considering different individual classifiers trained with the whole training data (heterogeneous base classifiers) and (ii) considering only one baseline classifier trained with different random samples of the training data of equal size as the original training data (using bagging).

Multi-class problems with noisy data are analyzed in our second contribution. In this case, an interesting approach to reduce the effect of noise is to decompose the problem into several binary subproblems, reducing the complexity and, consequently, dividing the effects caused by noise into each of these subproblems. These techniques are referred to as binary decomposition strategies [LdCG08]. The most studied schemes in the literature are: OVO [KPD90], which trains a classifier to distinguish between each pair of classes, and OVA [AMMR95], which trains a classifier to distinguish each class from all other classes. We will analyze the usage of the OVO strategy, which generally out-stands over OVA [GFB$^+$11, RK04], and check its suitability with noisy training data. Several well-known classification algorithms (C4.5 [Qui93], RIPPER [Coh95] and k-NN [McL04]), with or without decomposition, will be trained on them in order to check when decomposition is advantageous.

In order to reach meaningful conclusions based on the characteristics of the noise, a large collection of real-world data sets will be considered and different types of noise, present in real-world data, and several noise levels will be introduced into them, since these are usually unknown in real-world data. Two different types of noise, class and attribute noise, and four different schemes to introduce them will be considered [ZW04]. A large number of noise levels - from 5% to 50%, by increments of 5% - will be also studied. In the case of MCSs, the experimentation will consist of a total of 1640 noisy data sets and 800 data sets will be created in the study considering the OVO decomposition strategy. The results obtained will be contrasted using the proper statistical tests, as recommended in the specialized literature [Dem06].

The journal articles associated to this part are:

- Sáez J. A., Galar M., Luengo J., and Herrera F. (2013) Tackling the Problem of Classification with Noisy Data using Multiple Classifier Systems: Analysis of the Performance and Robustness. *Information Sciences* 247: 1–20
- Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems* 38(1): 179–206

5.2 Feature weighting schemes to treat with attribute noise in nearest neighbor classifiers

The NN classifier [CH67] is one of the most widely used methods in classification tasks due to its simplicity and good behavior in many real-world domains [WKRQ$^+$07]. The most frequently used similarity function for the NN classifier is Euclidean distance [AKA91]. However, it is very sensitive to noisy, redundant and irrelevant features, which may cause its performance to deteriorate [CGG$^+$09]. Feature weighting methods [WAM97] try to overcome this problem by incorporating weights into the similarity function to increase or reduce the importance of each feature, according to how they behave in the classification task. By contrast to Feature Selection [LM07, XZB14, Ker14, LQDZ13], the usage of weighting schemes provides the classifiers with a way of considering features *partially*, giving them some degree of importance in the classification task. This is usually preferred since weak, yet useful features may still be considered, instead of forcing the methods to either accept or completely ignore them.

Our aim is to propose a novel approach for weighting features in the NN classification, based on the usage of imputation methods [LGH12, FKD08]. These are commonly employed to estimate those feature values in a data set that are unknown, formally known as missing values [JL13], using the rest of the data available. Therefore, imputation methods enable us to sample values from an estimated distribution of the original data set, in which the value of each feature is conditioned to the rest of the features or all the data. The generated samples of each feature can be compared with the original values in order to detect the relevance of each feature, depending on the accuracy of the estimation for that feature performed by the imputation method. The Kolmogorov-Smirnov statistic [Smi39] may then be used to evaluate the differences between the original distribution of the feature's values and that of the imputed ones. It is thus possible to measure how well the values of each feature can be predicted using the rest of the data.

In our first contribution, we focus of classification problems with low or inexistent noise. In these problems, we want to reduce the importance of irrelevant attributes while maximizing the importance of the rest of attributes in order to facilitate the classifier building. Thus, we use the aforementioned procedure to give more importance to those features with high changes between their original and estimated value distributions - these features keep most of the structural information of the data and are not easily predictable using the rest of the data, which reduces the effect of those features that are easily predictable, and which are therefore likely to be redundant or irrelevant.

In our second contribution, we focus on classification problems with a notable quantity of noise. In this case, we adapt the above procedure inverting the weighting scheme: we give less importance to those features with high changes between their original and estimated value distributions and therefore they contain too much noise and we increment the effect of those features that are easily predictable, and which have therefore a less amount of noise.

In order to complete these studies, we will perform an experimentation in which our first proposal

will be compared with several classic and recent proposals of feature weighting, considering 25 supervised classification problems taken from the Keel-dataset repository [AFFL$^+$11]. Our second proposal will be compared with the classic NN classifier considering the aforementioned 25 data sets, in which different types and levels of noise are introduced.

The publications associated to this part are:

- Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Statistical computation of feature weighting schemes through data estimation for nearest neighbor classifiers. *Pattern Recognition* 47(12): 3941–3948
- Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Improving the behavior of the nearest neighbor classifier against noisy data with feature weighting schemes. In *Hybrid Artificial Intelligence Systems*, volumen 8480 of *Lecture Notes in Computer Science*, pp. 597–606. Springer International Publishing

5.3 Data level approaches and proposals to deal with noisy data

Noise filters [Wil72, BF99] are employed to remove corrupted data and improve the classification performance, particularly of instance-based learners such as the NN classifier [CH67]. However, their efficacy depends on the properties of the data [WZ08]. These can be analyzed by what are known as data complexity measures [BH06], which are characteristics of the data which are considered difficult in classification tasks, such as the overlapping among classes or their separability. Our first contribution in this field consists of studying the relation between the complexity metrics of a data set and the efficacy of several noise filters to improve the performance of the noise-sensitive NN classifier. A methodology is proposed to extract a rule set based on data complexity measures that enables one to predict in advance whether the use of noise filters will be statistically beneficial. This prediction can help, for example, to determine an appropriate noise filter for a concrete noisy data set –that filter providing a significant advantage in terms of the results– or to design new noise filters which select more or less aggressive filtering strategies considering the characteristics of the data.

The second contribution to the field of noise filtering techniques is to propose a new noise filtering method, based on combining different noise filtering paradigms in order to increase the accuracy of the classification algorithms used later. First, the filtering is based on the fusion of the predictions of several classifiers used to detect the presence of noise. Thus, we consider the combination of classifiers instead of using only one to detect noise. Second, the proposed method follows an iterative noise filtering scheme that allows us to avoid using detected noisy examples in the next phases of the filtering. Third, we also introduce a noisy score to control the filtering sensitiveness, and hence, the amount of noisy examples removed in each iteration can be adapted to the necessities of the practitioner. In this way, we take advantage of the three different paradigms. The validity of the proposed method will be studied in an exhaustive experimental study, considering 25 real-world data sets, into which different class noise levels will be introduced (from 5% to 30%, by increments of 5%). The filtered data sets will be then used to create classifiers with several learning methods of a different and well-known behavior against noise. We will compare the new filtering method against the state-of-the-art methods to deal with noisy data sets.

In our third contribution, we focus on the effects of noise in classification data sets that have an unequal class distribution among their examples. This problem is known as imbalanced classification [HYKL06]. The *Synthetic Minority Over-sampling Technique* (SMOTE) [CBHK02] is one of

the most well-know data pre-processing methods to cope with it and to balance the different number of examples of each class. However, if imbalanced problems suffer from noise, certain intrinsic limitations of SMOTE can aggravate the problem produced by noisy examples and current generalizations of SMOTE are not correctly adapted to their treatment. Our work proposes the extension of SMOTE through a new element, the iterative ensemble-based noise filter IPF [KR07] (which has been used in standard classification only), which can overcome the problems produced by noisy examples in imbalanced data sets. This extension results in SMOTE-IPF. The properties of this proposal will be discussed in a comprehensive experimental study. It will be compared against a basic SMOTE and its most well-known generalizations. The experiments will be carried out both on a set of synthetic data sets with different levels of noise and shapes of borderline examples as well as real-world data sets. Furthermore, the impact of introducing additional different types and levels of noise into these real-world data will be studied.

The problem of instance filtering is that it can be harmful as it may eliminate more examples than necessary or produce an information loss [KB81]. This is studied in our fourth contribution, in which we introduce a new proposal where the goal is not to filter the instances but to correct the mislabeled ones if possible. Only in the case where the instance is completely skewed it is eliminated from the data set. Relabeling noisy instances is performed by the use of an ensemble of specialized class noise filters. By aggregating the information they provide for each instance it is possible to repair the class label in many cases and to discard fewer instances than any of the individual filters would do. The dual behavior of the method –label repairing and noise filtering– is intended to achieve the desired balance between overcleansing and maintaining questionable instances. In order to check the validity of the proposal, a thorough empirical study will be developed. We will introduce class noise into 25 base data sets and creating a total of 175 data sets. Several class noise levels will be introduced into them, since these are usually unknown in real-world data, from 5% to 30%, by increments of 5%. The test accuracy of k-NN with the proposal and the baseline data sets will be compared using Wilcoxon's statistical test [GFLH10] in order to check the significance of the differences found. The analysis will be also continued and extended by using a learner considered robust to noise as is C4.5 [Qui93] and a SVM also considered being noise sensitive [Vap98].

The journal articles associated to this part are:

- Sáez J. A., Luengo J., and Herrera F. (2013) Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition* 46(1): 355–364
- Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) INFFC: An iterative noise filter based on the fusion of classifiers with noise sensitiveness control (submitted)
- Sáez J. A., Luengo J., Stefanowski J., and Herrera F. (2014) SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, doi: 10.1016/j.ins.2014.08.051. *Information Sciences* (in press)
- Sáez J. A., Luengo J., Shim S., and Herrera F. (2014) Class Noise Reparation by an Aggregated Noise Filter Ensemble Voting Algorithm (submitted)

5.4 Evaluation measures of the behavior of classifiers with noisy data

One may wonder how to know which systems are more suitable or are better adapted to deal with noisy data. Even though some classifiers have been related to this capability of working with imperfect data, this fact is usually based on only checking the accuracy of those and other classifiers

over a concrete collection of data sets, with independence of the type and noise level present in the data. This analysis procedure has a double disadvantage in noisy environments. First of all, the study of the performance alone does not provide enough information on the classifier behavior affected by the noise [KZ94, Kha96]. Moreover, a study with a controlled (probably artificial) noise level for each data set is also necessary to reach meaningful conclusions when evaluating the classifier behavior against noise [ZW04].

In our first contribution, we propose a simple and intuitive metric: the *Relative Loss of Accuracy* (RLA). This metric is mainly based on the concept of robustness [KZ94, Kha96], an important issue in noisy environments that must be carefully studied. It computes the percentage variation between the performance of the classifier trained with noisy data and that trained with clean data. This metric will be computed in an case of study where we will compare a Fuzzy Rule Based Classification System (FURIA [HH09]) versus a crisp robust learner (C4.5 [Qui93]), both trained on data sets with different levels of class noise.

The main problem of RLA is that it may carry some problems when two different classifiers are compared. In our second contribution, we propose a new single score to perform an analysis of the classifier behavior with noisy data from a double point of view focusing on the classic performance assessment of the methods but also on their robustness. Since performance and robustness are different concepts, the conclusions that they provide may also be different. In this second work, we analyze the existing robustness measures in the classification framework focusing on their advantages and disadvantages and motivate the necessity of combining the robustness and performance concepts to obtain an unified conclusion on the expected behavior of the methods with noisy data. We will propose a new behavior-against-noise measure to characterize the behavior of a method with noisy data, the *Equalized Loss of Accuracy* (ELA) measure, which tries to minimize the problems of considering performance and robustness measures individually. In order to complete our analysis, we will perform an experimental evaluation of the behavior and representativeness of the different measures, considering several classifiers with a known behavior against noise (concretely, C4.5 [Qui93] and SVM [CV95]). The behavior of such classifiers described by using ELA will be tested using 32 data sets from the KEEL-dataset repository [AFFL$^+$11], over which we will introduce a 10% of noise level into the class labels in a controlled way [ZW04].

The publications associated to this part are:

- Sáez J. A., Luengo J., and Herrera F. (2011) Fuzzy Rule Based Classification Systems versus Crisp Robust Learners Trained in Presence of Class Noise's Effects: a Case of Study. In *11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), Córdoba (Spain)*, pp. 1229–1234

- Sáez J. A., Luengo J., and Herrera F. (2014) Evaluating the classifier behavior with noisy data considering performance and robustness: the Equalized Loss of Accuracy measure (submitted)

6. Discussion of results

The following subsections summarize and discuss the results of each specific stage of the thesis.

6.1 Non-preprocessing alternatives based on multiple classifiers to deal with noisy data

In our first work, we analyze how well several MCSs behave with noisy data by means the realization of a huge experimental study. The results obtained show that the MCSs studied statistically improve the performance of their single classification algorithms when dealing with noisy data in the majority of cases. The improvement depends on many factors, such as the type and level of noise. Moreover, the performance of the MCSs built with heterogeneous classifiers depends on the performance of their single classifiers, so it is recommended to study the behavior of each single classifier before building the MCS. Generally, the MCSs studied perform better with class noise than with attribute noise. The *robustness results* show that the MCS built with heterogeneous classifiers will not be more robust than the most robust among their single classification algorithms. In fact, the robustness can always be shown as an average of the robustnesses of the individual methods. BagC4.5 is an exception: it becomes more robust than its single classifier C4.5. The study of several decisions combination methods shows that the majority vote scheme is a simple yet powerful alternative to other techniques when working with noisy data.

The main reason for the better performance of MCSs with noisy data can be attributed to the increment of the capability of generalization when an MCS is built. Each single classifier is known to make errors, but since they are different, that is, they have different behaviors over different parts of the domain, misclassified examples are not necessarily the same [KHDM98], and the same is true for noisy instances. This fact enables MCSs to achieve a better generalization from the examples of a problem and leads to better avoiding the overfitting of noisy data and, therefore, to obtain more accurate solutions.

The other contribution to this topic analyzes the suitability of the usage of the OVO decomposition when dealing with noisy training data sets in multi-class problems. The results obtained show that the OVO decomposition improves the baseline classifiers in terms of accuracy when data is corrupted by noise in all the noise schemes studied. The robustness results are particularly notable with the more disruptive noise schemes - the random class noise scheme and the random attribute noise scheme - where a larger amount of noisy examples and with higher corruptions are available, which produces greater differences (with statistical significance).

Three hypotheses aim to explain the better performance and robustness of the methods using OVO when dealing with noisy data: (i) the distribution of the noisy examples in the subproblems, (ii) the increase of the separability of the classes and (iii) the possibility of collecting information from different classifiers.

6.2 Feature weighting schemes to treat with attribute noise in nearest neighbor classifiers

Our works in this topic are based on a new scheme for feature weighting developed to improve the performance of the NN classifier, in which the weights are computed by combining imputation methods and the Kolmogorov-Smirnov statistic.

In our first contribution, we focus on a problem of low quantity of erroneous attribute values, where we want to reduce the importance of irrelevant attributes, so we give more importance to those features that keep most of the structural information of the data and we reduce the effect of those features that are likely to be redundant or irrelevant. From the experimental results it is possible to conclude that our feature weighting scheme is not very sensitive to the imputation

method selected, since the results obtained in every case are quite similar regardless of the specific imputation technique chosen, and statistical differences among them have not been found. The results obtained show that all our approaches enhance the performance of NN to a greater degree than the rest of the feature weighting methods analyzed. They also show a robust behavior in several domains, in contrast to the rest of the classifiers, which demonstrate a variable performance when different data sets are considered. The statistical analysis performed confirms our conclusions.

In our second contribution, we focus on classification problems with a notable quantity of noisy examples (viewed as erroneous values). We have assigned a lower weight to that features that were more affected by the presence of noise (those features whose original and imputed distribution of values were more different). We have reduced the importance of these features that contain the more harmful noise and increased the importance of those features that are easily predictable, and which have a less amount of noise. The results obtained show that our approach enhance the performance of NN in the presence of several types and levels of noise. The statistical analysis supports our conclusions, even though in some cases the null hypothesis is not rejected.

6.3 Data level approaches and proposals to deal with noisy data

Our first contribution proposes a methodology to extract a rule set based on data complexity measures to predict in advance when a noise filter will statistically improve the results of the nearest neighbor classifier. The rule set proposed provides a good prediction performance of the noise filtering efficacy, showing that the conditions under which a noise filter works well are similar for other noise filters. The analysis of the rule set provided shows that, generally, noise filtering statistically improves the classifier performance of the nearest neighbor classifier when dealing with problems with a high value of overlapping among the classes. However, if the problem has several clusters with a low overlapping among them, noise filtering is generally unnecessary and can indeed cause the classification performance to deteriorate. The validation process carried out shows that the final rule set provided is fairly accurate in predicting the efficacy of noise filters before their application and it produces an improvement with respect to the indiscriminate usage of noise filters.

In our second contribution, we propose a new iterative noise filtering method based on the fusion of the predictions of several classifiers. We also introduce a noisy score to control the filtering sensitiveness and remove more or less noisy examples according to the practitioner's necessities. We compare our proposal (INFFC) against other well-known filters found in the literature over a large collection of real-world data sets with different levels of class noise. The validity of the proposed method is studied in an exhaustive experimental study. From the experimental results it is possible to conclude that our noise filter enhance the performance of the rest of the noise filters and also no preprocessing. It also shows a robust behavior in several domains, in contrast to the rest of the filters, which demonstrate a variable performance when different data sets are considered. The statistical analysis performed supports our conclusions.

In our third contribution, we focus on the presence of noisy and borderline examples in imbalanced data. SMOTE is extended with a new element, the IPF noise filter (resulting in SMOTE-IPF), to control the noise introduced by the balancing between classes produced by SMOTE and to make the class boundaries more regular. The results obtained show that our proposal has a notably better performance when dealing with imbalanced data sets with noisy and borderline examples in the different scenarios considered. Our proposal especially outperforms the rest of the methods with the more complex to learn data sets in each group of data sets: the non-linear synthetic data sets and the attribute noise real-world data sets. These observations are supported by statistical tests. One must consider that the ensemble-nature of IPF, which constitutes a robust and accu-

rate way of detecting mislabeled examples, the iterative noise detection and elimination processes carried out and the possibility of controlling the diversity between classifiers are the key points of IPF which finally produce a more accurate filtering process. All these factors help SMOTE-IPF to obtain better performances than other re-sampling techniques in the scenarios considered.

Finally, in our fourth contribution we propose a method based on a ensemble of noise filters whose main objective is the correction of class noise in the data, even though it also removes some examples where the correction is not reliable. The results obtained shows that the use of our preprocessing improves the performance results without preprocessing when data is corrupted by noise (for all the noise levels for the noise-sensitive methods NN and SVM and for intermediate-high noise levels for the robust method C4.5). When no noise is induced into the data, preprocessing or not CNC show a equivalent behavior. All these observations are supported by statistical tests. We also analyze the working of our proposal in terms of the number of examples corrected and removed, showing that a great efficacy of the correction process and very low rates of removed examples, which is desirable in many domains.

6.4 Evaluation measures of the behavior of classifiers with noisy data

In our works related to this topic, we propose two evaluation metrics: RLA and ELA. These proposals are based on the idea that performance and robustness are two independent concepts that imply different conclusions.

In our first contribution, we propose the RLA metric, which is mainly focused on the concept of robustness. We use RLA to compare two classifiers (FURIA and C4.5). The results obtained indicate that FURIA has better test accuracy and a higher robustness (as indicated by RLA) than C4.5 when training with data with class noise.

However, RLA may present some problems when different classifiers are compared, which is discussed in our second contribution. In this work, we analyze the existing robustness measures pointing out their main drawbacks. We propose ELA, which considers both concepts together (performance and robustness). This seems to be crucial in order determine the expected behavior of the classifiers against noise. We experimentally compare the ELA and RLA measures, showing that the evaluation of the ELA and RLA metrics agree in some cases, but in other cases the behavior of the RLA is not so desirable since it is only based on the percentage variation of the performance without and with noise. The results obtained show that ELA is able to overcome some of the problems that RLA produces, being useful to represent the behavior of the classifiers against noise.

7. Concluding remarks

This thesis has addressed several problems related to the presence of noisy data in classification tasks. Two main research lines, related to the two classic proposals to deal with noisy data, compose this dissertation: studies and proposals for the treatment of noisy data at the *algorithm level* and *data level*.

The *algorithm level* approaches to deal with noisy data present some problems: the adaptation of each particular algorithm to train with noisy data and, sometimes, knowing the characteristics of noise to perform that adaptation. In order to avoid the aforementioned problems, Multiple Classifier Systems and the One-vs-One decomposition strategy have been applied providing good results when noisy data are considered.

Among the *algorithm level* approaches, a new classifier based on the nearest neighbor rule has been proposed to deal with noisy data. This method weights the different attributes to reduce the importance of those that do not contribute to successfully create the classifier. It is based on a novel idea that combines imputation methods and the Kolmogorov-Smirnov test to measure the changes between the distributions of the original and imputed values. The proposed model has been able to outperform other feature weighting methods proposed in the literature and the classic nearest neighbor classifier with good results.

Another part of this thesis has been devoted to the development of studies and proposals to deal with noisy data at the *data level*. The characteristics of the data that determine the performance of noise filters haven been studied, analyzing their data complexity metrics. This study is very important due to the large number of noise filters found in the literature. Within this research line, several noise filtering methods have been proposed. The first one (INFFC) combines several filtering paradigms (filters based on multiple classifiers, iterative filtering and the use of noise metrics) in order to take advantage of each one. We have also proposed another preprocessing method based on the filtering of noise (SMOTE-IPF) in the field of imbalanced classification. Both have outperformed the rest of noise filtering techniques of the state-of-the-art which they have been compared with, showing their potential in this field. On the other hand, a class noise correction technique has been proposed (CNC), based on a ensemble of multiple filters, which attempts to correct, rather than eliminate, the examples identified as noise. This last proposal has been a point of innovation in the field of preprocessing of noisy data, due to most of the techniques proposed in the literature just filter instances.

Finally, we have focused on the study of measures of evaluation of classifiers dealing with noisy data. The proposal of measures evaluating the classifiers in these circumstances seems to be important to progress in this field. We have proposed several measures of evaluation, such as RLA and ELA, and the properties of each one have been analyzed. Even though both measures can be used, RLA works better when the robustness of a single classifier is evaluated, but it has some problems when different classifiers are compared. Thus, ELA is able to solve some of the problems that RLA implies and it is useful to represent the behavior of different classifiers with noisy data.

Conclusiones

En esta tesis se han abordado varios problemas relacionados con la presencia de ruido en los datos en tareas de clasificación. Dos líneas principales de investigación, vinculadas con las dos propuestas clásicas para tratar el ruido, conforman esta disertación: estudios y propuestas para el tratamiento de datos con ruido tanto a nivel de algoritmos como a nivel de preprocesamiento de datos.

Los enfoques a nivel de algoritmos para tratar con datos con ruido presentan algunos problemas: la adaptación de cada algoritmo particular para entrenar con datos con ruido y, en muchas ocasiones, la necesidad de conocer las características del ruido para realizar dicha adaptación. Con el fin de evitarlos, se han aplicado técnicas basadas en Sistemas de Múltiples Clasificadores y la estrategia de descomposición Uno-frente-Uno, que permiten evitar estos problemas mostrando buenos resultados al tratar con el ruido.

Dentro de los enfoques a nivel de algoritmos, también se ha propuesto un nuevo clasificador basado en la regla del vecino más cercano para tratar con datos con ruido. Este método pondera los distintos atributos para reducir la importancia de aquellos que no contribuyen a crear correctamente el modelo. Se basa en una novedosa idea que combina métodos de imputación y el test Kolmogorov-Smirnov para medir los cambios entre las distribuciones de valores original e imputada. El modelo

propuesto ha sido capaz de superar a otros métodos de ponderación de características existentes en la literatura y al clasificador del vecino más cercano con buenos resultados.

Otra parte de esta tesis se ha dedicado al campo de estudios y propuestas para tratar con datos con ruido a nivel de datos. Se ha estudiado cuáles son las características de los datos, analizando sus medias de complejidad, que determinan el mejor o peor funcionamiento de las técnicas de filtrado del ruido. Este estudio es importante debido a la gran cantidad de técnicas de filtrado existentes. Dentro de esta línea también se han propuesto varios métodos de filtrado del ruido. El primero de ellos (INFFC) combina varios paradigmas de filtrado (basado en múltples classificadores, filtrado iterativo y usando medidas del ruido), con el fin de aprovechar las ventajas de cada uno de ellos. También hemos propuesto otro método de preprocesamiento basado en el filtrado del ruido (SMOTE-IPF) en el campo de la clasificación no balanceada. Ambos han mostrado superar al resto de ténicas de filtrado en el estado del arte con las que han sido comparados, lo cual muestra su potencial en este campo. Por otro lado, se ha propuesto una técnica de corrección de datos con ruido de clase (CNC), basada en un ensemble de múltiples filtros, que intenta corregir, en vez de eliminar, los ejemplos identificados como ruido. Esta última propuesta ha supuesto un punto de innovación en el campo del preprocesamiento de datos con ruido, debido a que la mayoría de técnicas propuestas en la literatura son filtros de ruido.

Por último, nos hemos centrado en el estudio de medidas de evaluación de los clasificadores cuando se trabaja con datos con ruido. La propuesta de medidas de evaluación de clasificadores en estas circunstancias parece importante para avanzar en este campo. Hemos propuesto varias medidas de evaluación, como RLA y ELA, y se han analizado las propiedades de cada una de ellas. RLA parece funcionar correctamente cuando se quiere medir la robustez de un único clasificador, pero presenta algunos problemas cuando se pretenden comparar varios clasificadores distintos. Así, los resultados obtenidos muestran que ELA es capaz de solventar algunos de los problemas que RLA supone, siendo útil para representar el comportamiento de distintos clasificadores con datos con ruido y permitir su comparación.

8. Future work

The results achieved in this PhD thesis may open new future trends in different challenging problems. In what follows, we present some research lines that can be addressed starting from the current studies:

- **Combination of multiple classifier strategies with noise preprocessing techniques when dealing with noisy data.** Our papers have shown good results of Multiple Classifier Systems and the One-vs-One decomposition strategy when training with noisy data, particularly with the most disruptive noise schemes [SGLH13, SGLH14a]. These techniques allow one to employ noise preprocessing methods, such as the noise filters [Wil72, BF99], combining in this way these two different strategies. Thus, noise affecting each one of the classifiers created by the multiple classifier strategies could be reduced and, therefore, the prediction capabilities of the final system may improve.

- **Treatment of erroneous attribute values.** The problem of erroneous class values or mislabeled examples, which is a type of class noise, has been widely addressed in the literature [ZW04]. Thus, for example, there are many noise filtering techniques that have been proposed to cope with it [Wil72, BF99, KR07, GBLG99]. However, the problem of erroneous attribute values as a type of attribute noise has been less studied in the specialized literature,

since its identification and treatment are more complex than in the case of mislabeled examples. It has been proven that the elimination of instances with erroneous attribute values is counterproductive since they still contain valuable information in other attributes which can help to build the classifier [ZW04]. However, a possible alternative could be to correct these erroneous attribute values instead of removing the complete noisy example from the training data. This procedure may imply an improvement to the model built without the loss of useful information that the elimination of examples suppose. In order to accomplish this idea, it could be interesting to follow a similar scheme to that of our work [SDLH14a], trying to correct those attribute values that present higher differences between the real value of the attribute of an example and its estimated value obtained by means of the usage of imputation methods [BM03].

- **Design of new fuzzy algorithms adapted to noisy data.** Fuzzy Rule-Based Classification Systems (FRBCSs) [INN04] are widely employed due to their ability to build a linguistic model interpretable to the users with the possibility of mixing different information such as that proceeding from expert knowledge and information from mathematical models or empirical measures. Traditionally, fuzzy systems have been related to the capability of working with imperfect data, even though most of them are not specially designed to deal with noisy data. Fuzzy rules and inference processes of fuzzy systems which differ from those with classic crisp systems may enable one to better avoid the overfitting of noisy instances and obtain more accurate and robust classifiers. If the presence of noise in the data is explicitly modeled in the fuzzy systems, it is more likely that its behavior with noisy data to be improved.

- **Study of the problem of noise in big data.** The problem of *big data* affects to classical and advanced data mining techniques [BBBE11]. It refers to the challenges and advantages derived from collecting and processing vast amounts of data. Formally, it is defined as the quantity of data that exceeds the processing capabilities of a given system. The main challenges are to deal with the increasing scale of data at the level of number of instances, at the level of features or characteristics and the complexity of the problem. Nowadays, with the availability of cloud platforms we dispose of sufficient processing units to extract valuable knowledge from massive data. Therefore, the adaptation of data mining techniques to emerging technologies, such as distributed computation, will be a mandatory task to overcome their limitations. Noise filters found in the literature, and also these ones proposed in this thesis, are not directly extensible to the big data problem since they are limited by the memory and processor capabilities of the system used. Therefore, they require of intelligent strategies that allow one to use them with huge amounts of data.

- **Tackling other learning paradigms with noise preprocessing techniques.** Besides standard classification problems, there are many other challenges in machine learning which are of great interest to the research community [YW06]. For example, in this thesis a noise preprocessing technique for imbalance classification problems has been proposed [SLSH14b]. We consider that the lessons learned in this thesis can be used to explore new problems. For example, this is the case of the *one-class classification* [ZYY+14], in which the classifier is built on the basis of examples coming only from a single class, while it must discriminate between the known examples and new, unseen examples. Applying noise preprocessing techniques in this topic may be useful to reduce computational complexity and sensitivity to noisy data of the models. Another example is *multi-label classification* [TK07], in which one instance can be assigned to multiple classes. Due to the increase in the complexity of the classification process in this scenario, the adaptation of noise preprocessing techniques is a very challenging and interesting task.

Chapter II

Publications: published and submitted papers

1. Non-preprocessing alternatives based on multiple classifiers to deal with noisy data

The journal papers associated to this part are:

1.1 Tackling the Problem of Classification with Noisy Data using Multiple Classifier Systems: Analysis of the Performance and Robustness

Sáez J. A., Galar M., Luengo J., and Herrera F. (2013) Tackling the Problem of Classification with Noisy Data using Multiple Classifier Systems: Analysis of the Performance and Robustness. *Information Sciences* 247: 1–20

- Status: **Published**.
- Impact Factor (JCR 2013): 3.893
- Subject Category: Computer Science, Information Systems. Ranking 8 / 135 (**Q1**).

Information Sciences 247 (2013) 1–20

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Tackling the problem of classification with noisy data using Multiple Classifier Systems: Analysis of the performance and robustness

José A. Sáez [a,*], Mikel Galar [b], Julián Luengo [c], Francisco Herrera [a]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, Granada 18071, Spain
[b] Department of Automática y Computación, Universidad Pública de Navarra, Pamplona 31006, Spain
[c] Department of Civil Engineering, LSI, University of Burgos, Burgos 09006, Spain

ARTICLE INFO

Article history:
Received 25 May 2012
Received in revised form 13 May 2013
Accepted 2 June 2013
Available online 13 June 2013

Keywords:
Noisy data
Class noise
Attribute noise
Multiple Classifier System
Classification

ABSTRACT

Traditional classifier learning algorithms build a unique classifier from the training data. Noisy data may deteriorate the performance of this classifier depending on the degree of sensitiveness to data corruptions of the learning method. In the literature, it is widely claimed that building several classifiers from noisy training data and combining their predictions is an interesting method of overcoming the individual problems produced by noise in each classifier. This statement is usually not supported by thorough empirical studies considering problems with different types and levels of noise. Furthermore, in noisy environments, the noise robustness of the methods can be more important than the performance results themselves and, therefore, it must be carefully studied. This paper aims to reach conclusions on such aspects focusing on the analysis of the behavior, in terms of performance and robustness, of several Multiple Classifier Systems against their individual classifiers when these are trained with noisy data. In order to accomplish this study, several classification algorithms, of varying noise robustness, will be chosen and compared with respect to their combination on a large collection of noisy datasets. The results obtained show that the success of the Multiple Classifier Systems trained with noisy data depends on the individual classifiers chosen, the decisions combination method and the type and level of noise present in the dataset, but also on the way of creating diversity to build the final system. In most of the cases, they are able to outperform all their single classification algorithms in terms of global performance, even though their robustness results will depend on the way of introducing diversity into the Multiple Classifier System.

1. Introduction

Classifier learning algorithms aim to extract the knowledge from a problem from the available set of labeled examples (training set) in order to predict the class for new, previously unobserved, examples [8]. Classic learning algorithms [36,4] build a unique model, called a classifier, which attempts to generalize the peculiarities of the training set. Therefore, the success of these methods, that is, their ability to classify new examples, highly depends on the usage of a concrete feature descriptor and a particular inference procedure, and directly on the training data.

* Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
E-mail addresses: smja@decsai.ugr.es (J.A. Sáez), mikel.galar@unavarra.es (M. Galar), jluengo@ubu.es (J. Luengo), herrera@decsai.ugr.es (F. Herrera).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.06.002

Real-world data, which is the input of the classifier learning algorithms, are affected by several components [42,52,37]; among them, the presence of noise is a key factor. Noise is an unavoidable problem, which affects the data collection and data preparation processes in Data Mining applications, where errors commonly occur [48,50]. The performance of the classifiers built under such circumstances will heavily depend on the quality of the training data, but also on the robustness against noise of the classifier itself. Hence, classification problems containing noise are complex problems and accurate solutions are often difficult to achieve with a unique classifier system – particularly if this classifier is noise-sensitive.

Several works have claimed that simultaneously using classifiers of different types, complementing each other, improves classification performance on difficult problems, such as satellite image classification [27], fingerprint recognition [30] and foreign exchange market prediction [33]. Multiple Classifier Systems (MCSs) [15,14,32,45] are presented as a powerful solution to these difficult classification problems, because they build several classifiers from the same training data and therefore allow the simultaneous usage of several feature descriptors and inference procedures. An important issue when using MCSs is the way of creating *diversity* among the classifiers [24], which is necessary to create discrepancies among their decisions and hence, to take advantage from their combination.

MCSs have been traditionally associated with the capability of working accurately with problems involving noisy data [15]. The main reason supporting this hypothesis could be the same as one of the main motivations for combining classifiers: the improvement of the generalization capability (due to the complementarity of each classifier), which is a key question in noisy environments, since it might allow one to avoid the overfitting of the new characteristics introduced by the noisy examples [39]. Most of the works studying MCSs and noisy data are focused on techniques like bagging and boosting [7,25,20], which introduce diversity considering different samples of the set of training examples and use only one baseline classifier. For example, in [7] the suitability of randomization, bagging and boosting to improve the performance of C4.5 was studied. The authors reached the conclusion that with a low noise level, boosting is usually more accurate than bagging and randomization. However, bagging outperforms the other methods when the noise level increases. Similar conclusions were obtained in the paper of Maclin and Opitz [25]. Other works [20] compare the performance of boosting and bagging techniques dealing with imbalanced and noisy data, reaching also the conclusion that bagging methods generally outperforms boosting ones. Nevertheless, explicit studies about the adequacy of MCSs (different from bagging and boosting, that is, those introducing diversity using different base classifiers) to deal with noisy data have not been carried out yet. Furthermore, most of the existing works are focused on a concrete type of noise and on a concrete combination rule. On the other hand, when data are suffering from noise, a proper study on how the robustness of each single method influences the robustness of the MCS is necessary, but this fact is usually overlooked in the literature.

This paper aims to develop a thorough analysis of the behavior of several MCSs with noisy training data with respect to their individual components (classifiers), studying to what extent the behavior of these MCSs depends on that of the individual classifiers. The classic hypothesis about the good behavior of MCSs with noisy data will be checked in detail and the conditions under which the MCSs studied work well with noisy data will be analyzed. In order to reach meaningful conclusions based on the characteristics of the noise, a large collection of real-world datasets will be considered and different types of noise, present in real-world data, and several noise levels will be introduced into them, since these are usually unknown in real-world data. Two different types of noise, class and attribute noise, and four different schemes to introduce them will be considered. The experimentation will consist of a total of 1640 datasets. The results taken from these datasets will be analyzed taking into account two different factors: (i) the *performance*, and (ii) the *robustness*, i.e., the capability of the classifier to be insensitive to the increments in the noise level, of each method in each noisy dataset. The results obtained will also be contrasted using the proper statistical tests, as recommended in the specialized literature [16,17,10,11].

The choice of the single classification algorithms used to build the MCSs is based on their behavior with noisy data. In such a way, one is able to extract meaningful conclusions from the point of view of noise. Three algorithms have been selected, among the top ten algorithms in Data Mining [47], each one belonging to a different learning paradigm, and having a well-known differentiated robustness to noise: a *Support Vector Machine* (SVM) [5], C4.5 [36] and *k-Nearest Neighbors* (*k*-NN) [29].

All these classification algorithms will be combined using different decisions combination methods [28,38,41,18] to create MCSs of different sizes and characteristics. Two forms to create diversity will be considered: (i) considering different individual classifiers trained with the whole training data (heterogeneous base classifiers) and (ii) considering only one baseline classifier trained with different random samples of the training data of equal size as the original training data (using bagging). In this way, whether the performance and noise-robustness of the individual components is related to that of the corresponding MCS will be verified. All the conclusions and lessons learned from the analysis of the empirical results will be included in a specific section at the end of this paper.

A web-page with all the complementary material associated with this paper is available at http://www.sci2s.ugr.es/mcs_noise, including the basic information of this paper, all the datasets created and the complete results obtained for each classification algorithm, in such a way that this work becomes easily reproducible by other researchers.

The rest of this paper is organized as follows. Section 2 presents an introduction to classification with noisy data. Section 3 gives the motivations for the usage of MCSs. Next, Section 4 describes the experimental framework. Section 5 includes the experimental results and their analysis. Section 6 studies the results of different decisions combination methods. Section 7 presents the lessons learned and, finally, Section 8 presents some concluding remarks.

2. Noisy data in classification problems

First, this section introduces the problem of noisy data in classification (Section 2.1) and then, it describes how to simulate different types of noise present in real-world data (Section 2.2).

2.1. Introduction to noisy data

Data gathered from real-world problems are never perfect and often suffer from corruptions that may hinder the performance of the system in terms of the classification accuracy, building time, size and interpretability of the classifier [51]. Noise mainly affects the data acquisition and preprocessing phases, having two main sources [52]: implicit errors introduced by measurement tools, such as different types of sensors; and random errors introduced by batch processes or experts when the data are gathered, such as in a document digitalization process.

A large number of components determine the quality of a dataset [42]. Among them, the class labels and the attribute values directly influence the quality of a classification dataset. The quality of the class labels refers to whether the class of each example is correctly assigned; otherwise, the quality of the attributes refers to their capability of properly characterizing the examples for classification purposes – obviously, if noise affects attribute values, this capability of characterization and therefore, the quality of the attributes, is reduced. Based on these two information sources, two types of noise can be distinguished in a given dataset [46,3]:

1. **Class noise**. This occurs when an example is incorrectly labeled. Class noise can be attributed to several causes, such as subjectivity during the labeling process, data entry errors, or inadequacy of the information used to label each example. Two types of class noise can be distinguished: (i) *contradictory examples* [13] – duplicate examples have different class labels –, and (ii) *misclassifications* [53] – examples that are labeled as a class different from the real one.
2. **Attribute noise**. This refers to corruptions in the values of one or more attributes. Examples of attribute noise are: erroneous attribute values, missing or unknown attribute values, and incomplete attributes or "do not care" values.

In this paper, class noise refers to misclassifications, whereas attribute noise refers to erroneous attribute values, because they are the most common in real-world data [52]. Furthermore, erroneous attribute values, unlike other types of attribute noise, such as missing values (which are easily detectable), have received less attention in the literature.

Noise hinders the knowledge extraction from the data and spoils the models obtained using that noisy data when they are compared to the models learned from clean data from the same problem, which represent the real implicit knowledge of the problem [52]. In this sense, *robustness* [19] is the capability of an algorithm to build models that are insensitive to data corruptions and suffer less from the impact of noise; that is, the more robust an algorithm is, the more similar the models built from clean and noisy data are. Thus, a classification algorithm is said to be more robust than another if the former builds classifiers which are less influenced by noise than the latter. Robustness is considered more important than performance results when dealing with noisy data, because it allows one to know *a priori* the expected behavior of a learning method against noise in those cases where the characteristics of noise are unknown.

In order to analyze the degree of robustness of the classifiers in the presence of noise, we will compare the performance of the classifiers learned with the original (without induced noise) dataset with the performance of the classifiers learned using the noisy dataset. Therefore, those classifiers learned from noisy datasets that are more similar (in terms of results) to the noise-free[1] classifiers will be the most robust ones. An example of a robust learner is the C4.5 decision tree learning algorithm [36] considered in this paper, which uses pruning strategies to reduce the chances of trees being influenced by the noise in the training data [34,35]. One of the objectives of this paper is to check whether the performance and robustness of C4.5 trained with noisy data can be improved by incorporating it into an MCS.

2.2. Simulating the noise of real-world datasets

Checking the effect of noisy data on the performance of classifier learning algorithms is necessary to improve their reliability and has motivated the study of how to generate and introduce noise into the data. Noise generation can be characterized by three main characteristics [52]:

1. **The place where the noise is introduced.** Noise may affect the input attributes or the output class, impairing the learning process and the resulting model.
2. **The noise distribution.** The way in which the noise is present can be, for example, uniform [39,54] or Gaussian [53,52].
3. **The magnitude of generated noise values.** The extent to which the noise affects the dataset can be relative to each data value of each attribute, or relative to the minimum, maximum and standard deviation for each attribute [53,54,52].

[1] Noise-free refers to the original datasets without induced noise, which might contain noise that is not quantifiable.

In contrast to other studies in the literature, this paper aims to clearly explain how noise is defined and generated (see Section 4.2 for more details), and also to properly justify the choice of the noise introduction schemes. Furthermore, the noise generation software has been incorporated into the KEEL tool [2] for its free usage. The two types of noise considered in this work, class and attribute noise, have been modeled using four different noise schemes; in such a way, the presence of those types of noise will allow one to simulate the behavior of the classifiers in these two scenarios:

1. **Class noise** usually occurs on the boundaries of the classes, where the examples may have similar characteristics – although it can occur in any other area of the domain. In this paper, class noise is introduced using an *uniform class noise scheme* [39] (randomly corrupting the class labels of the examples) and a *pairwise class noise scheme* [53,52] (labeling examples of the majority class with the second majority class). Considering these two schemes, noise affecting any pair of classes and only the two majority classes are simulated, respectively.
2. **Attribute noise** can proceed from several sources, such as transmission constraints, faults in sensor devices, irregularities in sampling and transcription errors [40]. The erroneous attribute values can be totally unpredictable, i.e., random, or imply a low variation with respect to the correct value. We use the *uniform attribute noise scheme* [54,52] and the *Gaussian attribute noise scheme* in order to simulate each one of the possibilities, respectively. We introduce attribute noise in accordance with the hypothesis that interactions between attributes are weak [52]; as a consequence, the noise introduced into each attribute has a low correlation with the noise introduced into the rest.

3. Multiple Classifier Systems for classification tasks

This section focuses on describing the usage of MCSs for classification tasks. Section 3.1 presents the motivations for the usage of MCSs, whereas Section 3.2 describes the different ways usually considered to build an MCS. Next, Section 3.3 shows different methods to combine the outputs of the classifiers. Finally, Section 3.4 explains how the MCSs of this paper are built.

3.1. Single classifiers against their combination

Given a set of problems, finding the best overall classification algorithm is sometimes difficult because some classifiers may excel in some cases and perform poorly in others. Moreover, even though the optimal match between a learning method and a problem is usually searched, this match is generally difficult to achieve and perfect solutions are rarely found for complex problems [14,15]. This is a reason for using MCSs [15,14,32], since it is not necessary to choose a specific learning method. All of them might be used, taking advantage of the strengths of each method, while avoiding its weaknesses. Furthermore, there are other motivations to combine several classifiers [14]:

- To avoid the choice of some arbitrary but important initial condition, e.g., those involving the parameters of the learning method.
- To introduce some randomness to the training process in order to obtain different alternatives that can be combined to improve the results obtained by the individual classifiers.
- To use complementary classification methods improves dynamic adaptation and flexibility.

3.2. Using several classifiers for a classification problem

There are several strategies to use more than one classifier for a single classification task [15]:

- *Dynamic classifier selection*. This is based on the fact that one classifier may outperform all others using a global performance measure but it may not be the best in all parts of the domain. Therefore, this type of methods divide the input domain into several parts and aim to select the classifier with the best performance in that part.
- *Multi-stage organization*. This builds the classifiers iteratively. At each iteration, a group of classifiers operates in parallel and their decisions are then combined. A dynamic selector decides which classifiers are to be activated at each stage based on the classification performances of each classifier in previous stages.
- *Sequential approach*. A classifier is used first and the other ones are used only if the first does not yield a decision with sufficient confidence.
- *Parallel approach*. All available classifiers are used for the same input example in parallel. The outputs from each classifier are then combined to obtain the final prediction.

Although the first three approaches have been explored to a certain extent, the majority of classifier combination research focuses on the fourth approach, due to its simplicity and the fact that it enables one to take advantage of the factors presented in the previous section. For these reasons, this paper focus on the fourth approach.

3.3. Decisions combination in multiple classifiers systems

As has been previously mentioned, parallel approaches need a posterior phase of combination after the evaluation of a given example by all the classifiers. Many decisions combination proposals can be found in the literature, such as the intersection of decision regions [12], voting methods [28], prediction by top choice combinations [43], use of the Dempster–Shafer theory [26,49] or ranking methods [15]. In concrete, we will study the following four combination methods for the MCSs built with heterogeneous classifiers:

1. **Majority vote (MAJ)** [28]. This is a simple but powerful approach, where each classifier gives a vote to the predicted class and the most voted one is chosen as the output.
2. **Weighted majority vote (W-MAJ)** [38]. Similarly to MAJ, each classifier gives a vote for the predicted class, but in this case, the vote is weighted depending on the competence (accuracy) of the classifier in the training phase.
3. **Naive Bayes (NB)** [41]. This method assumes that the base classifiers are mutually independent. Hence, the predicted class is that one obtaining the highest posterior probability. In order to compute these probabilities, the confusion matrix of each classifier is considered.
4. **Behavior-Knowledge Space (BKS)** [18]. This is a multinomial method that indexes a cell in a look-up table for each possible combination of classifiers outputs. A cell is labeled with the class to which the majority of the instances in that cell belong to. A new instance is classified by the corresponding cell label; in case that for the cell is not labeled or there is a tie, the output is given by MAJ.

We always use the same training dataset to train all the base classifiers and to compute the parameters of the aggregation methods, as it is recommended in [23]. Using a separate set of examples to obtain such parameters can imply some important training data to be ignored and this fact is generally translated into a loss of accuracy of the final MCS built.

In MCSs built with heterogeneous classifiers, all of them may not return a confidence. Even though each classifier can be individually modified to return a confidence for its predictions, such confidences will come from different computations depending on the classifier adapted and their combination could become meaningless. Nevertheless, in MCSs built with the same type of classifier, this fact does not occur and it is possible to combine their confidences since these are homogeneous among all the base classifiers [23]. Therefore, in the case of Bagging, given that the same classifier is used to train all the base classifiers, the confidence of the prediction can be used to compute a weight and, in turn, these weights can be used in a weighted voting combination scheme.

3.4. Multiple Classifier Systems used in the experimentation

The choice of the learning algorithms used in this paper – SVM [5], C4.5 [36] and *k*-NN [29] – is based on their good behavior in a large number of real-world problems; moreover, they were selected because these methods have a highly differentiated and well known noise-robustness, which is important in order to properly evaluate the performance of MCSs in the presence of noise. This section briefly describes their operating procedures, with special reference to their noise handling mechanisms, allowing one to better understand the results of the impact of noise on them in the experimentation (Section 5).

- **C4.5 decision tree generator** [36]. C4.5 is considered a robust learner tolerant to noisy data. It iteratively builds a decision tree that correctly classifies the largest number of examples (defined by the *window size*). As indicated in [9], the main problem of *windowing* techniques is that the process may incorporate all noisy examples into the learning *window*, because they may be misclassified by an apparently good rule. In order to avoid this problem as well as over-fitting to noisy data, C4.5 uses pruning strategies to reduce the chances of classifiers being affected by noisy examples from the training data [34,35]. For example, in this paper, a post-pruning process is carried out, which discards unreliable parts from the fully grown decision tree. Nevertheless, this process is based on statistical assumptions that might be unreliable, since they depend on the training data. Therefore, when the noise level is relatively high, even a robust learner such as C4.5 may obtain a poor performance.
- **Support Vector Machine** [5]. SVM builds a hyperplane that separates examples of different classes maximizing the distance, that is, the margin, of the examples lying near the separating hyperplane, which are called support vectors. A better generalization is generally achieved when the distances from the examples of both classes to the hyperplane are larger, because the maximization of the margin of separation reduces the empirical risk instead of the expected risk. SVM relies on the support vectors, which are identified from the training instances, to derive the decision model. Thus, the hyperplanes found by SVM can be easily altered including or excluding a single noisy example [31]. Furthermore, the implicit interdependence among the input attributes – since they are fused into two factors in the learning phase – may also cause difficulties when noise is introduced into the training data, which disrupts the interrelations and correlations between the attributes. Thus, SVM should *a priori* be more noise-sensitive than C4.5. We have used the Puk kernel for SVM, whose expression is given in Eq. (1):

$$K(x_i, x_j) = \frac{1}{\left[1 + \left(\frac{2\sqrt{\|x_i - x_j\|^2}\sqrt{2^{(1/\omega)}-1}}{\sigma}\right)^2\right]^{\omega}}, \quad (1)$$

where x_i and x_j are two vectors, and σ and ω control the half-width and the tailing factor of the peak of the function.

- ***k*-Nearest Neighbors** [29]. This method searches the k most similar examples to the given one using a distance function in order to determine its class. The function used in this paper is the Heterogeneous Value Distance Metric (HVDM), which computes the distance between the values of each attribute based on the training data distribution. In the experimentation, three different values of k are considered: 1, 3 and 5. The value of k determines a higher or lower sensitivity to noise [22]. Thus, if only one nearest neighbor is considered, k-NN relies on each single example to derive the decision model and the decision regions can be easily altered by individual examples. Otherwise, higher k values make the model more robust. In order to find the k nearest neighbors in the space of numerical and nominal attributes, the Heterogeneous Value Difference Metric (HVDM) metric is applied [44].

Considering the previous classifiers (SVM, C4.5 and k-NN), the following MCS are built:

1. **MCS3-*k***. These MCSs are composed by 3 individual classifiers (SVM, C4.5 and k-NN). Three values of k are considered (1, 3 and 5), so we will create three different MCS3-k denoted as MCS3-1, MCS3-3 and MCS3-5.
2. **MCS5.** This is composed by 5 different classifiers: SVM, C4.5 and k-NN with the 3 different values (1, 3 and 5).
3. **BagC4.5.** This MCS considers C4.5 as baseline classifier. In this case, for the decisions combination method we use as confidence the accuracy of the leaf predicting the class, which is computed as the percentage of correctly classified train examples from the total number of covered ones.

Therefore, the MCSs built with heterogeneous classifiers (MCS3-k and MCS5) will contain a noise-robust algorithm (C4.5), a noise-sensitive method (SVM) and a method whose robustness varies depending on the value of k (k-NN). Hence, how the behavior of each MCS3-k and the MCS5 depends on that of each single method can be analyzed. In the case of BagC4.5, it is compared to its base classifier, that is, C4.5.

The configuration parameters used to build the classifiers considered are shown in Table 1. The same parameter setup for each classification algorithm, considered either as part of a MCS or individually, has been used for all the datasets. This will help us to know to what extent the same classifier trained with the same parameters over the same training data can imply an advantage incorporating it into a MCS with respect to its usage alone.

4. Experimental framework

First, this section describes the base datasets used in the experiments (Section 4.1). Second, the processes to induce noise into them are introduced (Section 4.2). Finally, the methodology for the analysis of the results is explained in Section 4.3.

4.1. Base datasets

The experimentation is based on forty real-world classification problems from the KEEL-dataset repository[2] [1]. Table 2 shows the datasets sorted by the number of classes (#CL). Moreover, for each dataset, the number of examples (#EX) and the number of attributes (#AT), along with the number of numeric and nominal attributes are presented. Some of the largest datasets (*nursery*, *page-blocks*, *penbased*, *satimage*, *splice*) were stratified at 10% in order to reduce the computational time required for training, given the large amount of executions carried out in this paper. For datasets containing missing values (such as *automobile* or *dermatology*), instances with missing values were removed from the datasets before the partitioning.

4.2. Introducing noise into datasets

In the previous datasets, as in most of the real-world datasets, the initial amount and type of noise present are unknown. Therefore, no assumptions about the base noise type and level can be made. For this reason, these datasets are considered to be noise free, in the sense that no recognizable noise has been induced into them. In order to control the amount of noise in each dataset and check how it affects the classifiers, noise is introduced into each dataset in a supervised manner. Four different noise schemes proposed in the literature, as explained in Section 2, are used in order to introduce a noise level x% into each dataset:

1. **Introduction of class noise.**
   - **Uniform class noise** [39]. x% of the examples are corrupted. The class labels of these examples are randomly replaced by another one from the M classes.

[2] http://www.keel.es/datasets.php.

Table 1
Parameter configuration for the classification algorithms.

SVM	C4.5	*k*-NN
• Cost: $C = 100$, Tolerance: $t = 0.001$ • Parameter for the round-off error: $\epsilon = 10^{-12}$ • Type of Kernel: *Puk* with $\sigma = 1$, $\omega = 1$ • Data preprocessing: *Normalization in* $[0,1]$ • Fit logictics models to the output	• Confidence level: $c = 0.25$ • Minimal instances per leaf: $i = 2$ • Prune after the tree building	• Number of neighbors: $k = 1,3$ and 5 • Distance function: *HVDM*

Table 2
Summary description for the classification datasets.

Dataset	#EX	#AT	#CL	Dataset	#EX	#AT	#CL	Dataset	#EX	#AT	#CL	Dataset	#EX	#AT	#CL
banana	5300	2(2/0)	2	spambase	4597	57(57/0)	2	hayes-roth	160	4(4/0)	3	glass	214	9(9/0)	7
german	1000	20(13/7)	2	twonorm	7400	20(20/0)	2	car	1728	6(0/6)	4	shuttle	2175	9(9/0)	7
heart	270	13(13/0)	2	wdbc	569	30(30/0)	2	lymphography	148	18(3/15)	4	zoo	101	16(0/16)	7
ionosphere	351	33(33/0)	2	balance	625	4(4/0)	3	vehicle	846	18(18/0)	4	satimage	643	36(36/0)	7
magic	19020	10(10/0)	2	splice	319	60(0/60)	3	nursery	1296	8(0/8)	5	segment	2310	19(19/0)	7
monk	432	6(6/0)	2	contraceptive	1473	9(9/0)	3	page-blocks	548	10(10/0)	5	ecoli	336	7(7/0)	8
phoneme	5404	5(5/0)	2	iris	150	4(4/0)	3	cleveland	297	13(13/0)	5	led7digit	500	7(0/7)	10
pima	768	8(8/0)	2	new-thyroid	215	5(5/0)	3	automobile	159	25(15/10)	6	penbased	1099	16(16/0)	10
ring	7400	20(20/0)	2	thyroid	720	21(6/15)	3	dermatology	358	33(1/32)	6	yeast	1484	8(8/0)	10
sonar	208	60(60/0)	2	wine	178	13(13/0)	3	flare	1066	11(0/11)	6	vowel	990	13(13/0)	11

- **Pairwise class noise** [53,52]. Let X be the majority class and Y the second majority class, an example with the label X has a probability of $x/100$ of being incorrectly labeled as Y.

2. **Introduction of attribute noise.**
   - **Uniform attribute noise** [54,52]. $x\%$ of the values of each attribute in the dataset are corrupted. To corrupt each attribute A_i, $x\%$ of the examples in the data set are chosen, and their A_i value is assigned a random value from the domain $\mathbb{D}_i$ of the attribute A_i. An uniform distribution is used either for numerical or nominal attributes.
   - **Gaussian attribute noise**. This scheme is similar to the uniform attribute noise, but in this case, the A_i values are corrupted, adding a random value to them following a Gaussian distribution of $mean = 0$ and $standard\ deviation = (max - min)/5$, being max and min the limits of the attribute domain $(\mathbb{D}_i)$. Nominal attributes are treated as in the case of the uniform attribute noise.

In order to create a noisy dataset from the original one, the noise is introduced into the training partitions as follows:

1. A level of noise $x\%$, of either class noise (uniform or pairwise) or attribute noise (uniform or Gaussian), is introduced into a copy of the full original dataset.
2. Both datasets, the original one and the noisy copy, are partitioned into 5 equivalent folds, that is, with the same examples in each one.
3. The training partitions are built from the noisy copy, whereas the test partitions are formed from examples from the base dataset, that is, the noise free dataset.

We introduce noise, either class or attribute noise, only into the training sets since we want to focus on the effects of noise on the training process. This will be carried out observing how the classifiers built from different noisy training data for a particular dataset behave, considering the accuracy of those classifiers, with the same clean test data. Thus, the accuracy of the classifier built over the original training set without additional noise acts as a reference value that can be directly compared with the accuracy of each classifier obtained with the different noisy training data. Corrupting the test sets also affects the accuracy obtained by the classifiers and therefore, our conclusions will be not only limited to the effects of noise on the training process.

The accuracy estimation of the classifiers in a dataset is obtained by means of 5 runs of a stratified 5-fold cross-validation. Hence, a total of 25 runs per dataset, noise type and level are averaged. 5 partitions are used because, if each partition has a large number of examples, the noise effects will be more notable, facilitating their analysis.

As a consequence, a large collection of new noisy datasets are created from the aforementioned forty base datasets. Both types of noise are independently considered: class and attribute noise. For each type of noise, the noise levels ranging from $x = 0\%$ (base datasets) to $x = 50\%$, by increments of 5%, are studied. Therefore, 400 noisy datasets are created with each of the four aforementioned noise schemes. The total number of datasets in the experimentation is 1640. Hence, considering the 5x5fcv of the 1640 datasets, 41000 executions are carried out for each classification algorithm. All these datasets are available on the web-page associated with this paper.

4.3. Methodology of analysis

In order to check the behavior of the different methods when dealing with noisy data, the results of each MCS are compared with those of their individual components using two distinct properties:

1. The performance of the classification algorithms on the test sets for each level of induced noise, defined as its accuracy rate. For the sake of brevity, only averaged results are shown (the rest can be found on the web page associated with this paper), but it must be taken into account that the conclusions drawn in this paper are based on the proper statistical analysis, which considers all the results (not averaged).
2. The robustness of each method is estimated with the *relative loss of accuracy* (RLA) (Eq. (2)), which is used to measure the percentage of variation of the accuracy of the classifiers at a concrete noise level with respect to the original case with no additional noise:

$$RLA_{x\%} = \frac{Acc_{0\%} - Acc_{x\%}}{Acc_{0\%}}, \tag{2}$$

where $RLA_{x\%}$ is the relative loss of accuracy at a noise level $x\%$, $Acc_{0\%}$ is the test accuracy in the original case, that is, with 0% of induced noise, and $Acc_{x\%}$ is the test accuracy with a noise level $x\%$.

In order to properly analyze the performance and RLA results, Wilcoxon's signed rank statistical test is used, as suggested in the literature [6]. This is a non-parametric pairwise test that aims to detect significant differences between two sample means; that is, between the behavior of the two algorithms involved in each comparison. For each type and noise level, the MCS and each single classifier will be compared using Wilcoxon's test and the p-values associated with these comparisons will be obtained. The p-value represents the lowest level of significance of a hypothesis that results in a rejection and it allows one to know whether two algorithms are significantly different and the degree of this difference. Both performance and robustness are studied because the conclusions reached with one of these metrics need not imply the same conclusions with the other.

5. Analysis of the results: Noisy data in Multiple Classifier Systems

This section presents the results of performance and robustness of the MCSs trained with noisy data with respect to their individual classifier components. Section 5.1 analyzes the results obtained with class noise, whereas Section 5.2 is devoted to the attribute noise. Each one of these sections is divided into two main parts: the first part describes the results obtained comparing each MCS with each of its components, whereas the second one analyzes these results for each type of noise. Due to the large amount of results, they should be properly outlined. As a consequence, only the general behavior of the methods will be described in order to reach meaningful conclusions (without observing the characteristics of each single dataset). This section is devoted to the study of the MAJ scheme for the final class prediction in order to consider its results as a reference, since it is the simplest method; other decisions combination schemes are analyzed in Section 6.

5.1. First scenario: Datasets with class noise

Table 3 shows the performance (top part of the table) and robustness (bottom part of table) results of each classification algorithm at each noise level on datasets with class noise. Each one of these parts in the table (performance and robustness parts) is divided into other two parts: one with the results of the uniform class noise and another with the results of the pairwise class noise. A star '$*$' next to a p-value indicates that the corresponding single algorithm obtains more ranks than the MCS in Wilcoxon's test comparing the individual classifier and the MCS. Note that the robustness can only be measured if the noise level is higher than 0%, so the robustness results are presented from a noise level of 5% and higher.

The performance results in this table are summarized below:

- **Performance results with uniform class noise**.
  1. **Results of MCS3-*k***.
     - **MCS3-*k* vs. SVM**. MCS3-k is statistically better than SVM regardless of the value of k. Moreover, the higher the value of k, the lower the p-values are.
     - **MCS3-*k* vs. C4.5**. The performance of MCS3-k with respect to C4.5 heavily depends on the value of k. In MCS3-1 statistical differences are only found at the lowest noise levels (up to 5%), whereas MCS3-3 maintains these differences up to 30%. For the rest of the noise levels, both MCS3-1 and MCS3-3 are statistically equivalent to C4.5, although MCS3-3 generally obtains more ranks than C4.5. Finally, MCS3-5 is statistically better than all its individual components at all noise levels.
     - **MCS3-*k* vs. *k*-NN**. Statistical differences are found between MCS3-k and k-NN regardless of the value of k. Furthermore, as the value of k increases, so does the corresponding p-value at the different noise levels (note that very low p-values are obtained in the case of $k = 1$).

Table 3
Performance and robustness results on datasets with class noise.

x%	Results										*p*-values													
	Single methods					Multi-classifiers					MCS3-1 vs.			MCS3-3 vs.			MCS3-5 vs.			MCS5 vs.					BagC4.5 vs.
	SVM	C4.5	1-NN	3-NN	5-NN	MCS3-1	MCS3-3	MCS3-5	MCS5	BagC4.5	SVM	C4.5	1-NN	SVM	C4.5	3-NN	SVM	C4.5	5-NN	SVM	C4.5	1-NN	3-NN	5-NN	C4.5
Performance																									
Uniform class noise																									
0%	83.25	82.96	81.42	82.32	82.32	85.42	85.48	85.52	83.74	85.18	5.2E−03	1.8E−03	7.1E−04	1.3E−02	5.0E−04	3.1E−03	1.0E−02	2.8E−04	5.0E−03	4.4E−01	1.9E−02	1.4E−04	9.7E−06	2.2E−03	1.5E−07
5%	81.04	82.48	78.68	81.67	82.06	84.11	84.86	85.03	83.15	85.10	2.6E−04	3.0E−02	1.3E−06	2.4E−05	6.1E−04	7.8E−04	5.5E−06	2.2E−04	4.6E−03	6.6E−03	1.6E−02	4.2E−08	2.7E−06	3.5E−02	6.6E−08
10%	79.58	82.08	76.28	80.84	81.56	83.00	84.25	84.57	82.60	84.87	1.1E−04	3.9E−01	1.3E−07	1.5E−06	3.5E−03	7.8E−04	2.1E−07	4.5E−04	1.1E−02	1.7E−03	4.8E−02	3.9E−08	1.1E−06	8.1E−02	1.3E−07
15%	78.26	80.88	73.71	79.71	80.94	81.60	83.21	83.63	81.74	84.20	1.5E−04	7.1E−01	6.1E−08	1.3E−06	3.1E−03	4.7E−04	1.7E−07	6.1E−04	5.3E−02	2.1E−03	4.5E−02	3.6E−08	1.3E−06	2.9E−01	1.3E−07
20%	76.55	79.97	71.22	78.18	80.09	80.09	81.98	82.69	80.55	83.58	5.8E−05	9.5E−01*	1.0E−07	8.1E−07	2.0E−02	2.7E−04	1.7E−07	8.6E−04	6.2E−02	8.6E−04	7.4E−02	4.5E−08	1.1E−06	8.5E−01	6.3E−07
25%	75.05	79.02	68.78	76.68	79.27	78.53	80.87	81.78	79.50	82.80	3.7E−04	4.7E−01*	7.0E−08	2.1E−06	4.8E−02	2.2E−04	1.0E−07	1.2E−03	1.2E−01	2.4E−03	5.3E−02	3.6E−08	2.2E−06	9.3E−01*	8.1E−07
30%	73.82	77.90	65.88	75.09	78.18	77.10	79.69	80.75	78.18	81.79	2.8E−04	3.5E−01*	6.1E−08	1.7E−06	8.3E−02	1.9E−04	2.3E−07	2.0E−03	1.0E−01	1.9E−03	3.2E−01	3.6E−08	5.2E−06	5.2E−01*	4.0E−06
35%	72.31	76.28	63.51	72.95	76.67	75.11	77.87	79.16	76.44	80.39	1.7E−03	2.1E−01*	6.5E−08	2.9E−06	1.5E−01	9.2E−05	5.3E−07	5.4E−03	1.0E−01	3.1E−03	5.0E−01	3.6E−08	4.9E−06	3.3E−01*	4.6E−06
40%	70.69	74.51	61.00	70.71	75.09	73.20	76.04	77.56	74.72	78.85	7.2E−03	2.0E−01*	7.0E−08	1.8E−05	1.8E−01	7.8E−05	2.1E−06	5.6E−03	9.0E−02	5.2E−03	4.9E−01	3.6E−08	1.6E−06	3.8E−01*	3.9E−05
45%	69.10	72.07	58.68	68.00	72.57	70.68	73.61	75.27	72.16	77.06	4.3E−02	2.0E−01*	1.4E−07	1.2E−04	1.7E−01	7.3E−05	3.1E−06	6.6E−03	7.8E−02	2.4E−02	5.0E−01	3.6E−08	1.8E−06	5.5E−01*	9.2E−05
50%	67.07	69.22	55.55	65.39	70.40	67.64	70.82	72.70	69.43	74.31	5.4E−01	1.4E−01*	1.1E−07	6.7E−04	1.6E−01	2.3E−04	2.3E−05	4.8E−03	1.4E−01	8.1E−02	7.1E−01	3.6E−08	5.5E−06	2.4E−01*	1.3E−04
Pairwise class noise																									
0%	83.25	82.96	81.42	82.32	82.32	85.42	85.48	85.52	83.74	85.18	5.2E−03	1.8E−03	7.1E−04	1.3E−02	5.0E−04	3.1E−03	1.0E−02	2.8E−04	5.0E−03	4.4E−01	1.9E−02	1.4E−04	9.7E−06	2.2E−03	1.5E−07
5%	81.77	82.59	79.58	81.83	82.10	84.75	85.19	85.28	83.54	84.97	1.6E−04	1.0E−02	2.1E−06	1.2E−04	4.3E−04	7.4E−04	9.7E−05	1.7E−04	4.6E−03	2.7E−02	1.1E−02	1.6E−07	9.3E−07	1.8E−03	3.0E−07
10%	80.74	82.17	77.73	80.87	81.61	83.95	84.61	84.86	82.99	84.62	1.2E−04	5.8E−02	1.2E−07	6.2E−05	2.3E−03	4.3E−05	2.0E−05	3.9E−04	2.8E−03	1.7E−02	3.5E−02	6.1E−08	5.3E−07	6.1E−03	2.1E−07
15%	79.72	81.54	75.99	79.60	80.84	83.09	83.89	84.29	82.23	83.75	7.3E−05	1.6E−01	7.0E−08	2.0E−05	5.6E−03	4.9E−06	7.1E−06	5.2E−04	6.4E−04	1.2E−02	9.0E−02	4.5E−08	2.8E−07	5.2E−03	1.2E−06
20%	79.11	80.87	74.25	77.81	79.33	82.21	83.06	83.50	80.99	83.01	2.0E−04	3.5E−01	8.2E−08	3.3E−05	1.9E−02	3.8E−07	1.2E−05	1.8E−03	6.7E−06	4.4E−02	4.3E−01	4.5E−08	2.3E−07	2.9E−04	2.5E−06
25%	77.80	79.59	72.27	75.75	77.52	80.81	81.64	82.14	79.54	81.46	3.3E−04	4.5E−01	6.5E−08	9.7E−05	7.6E−02	3.3E−07	2.6E−05	2.8E−03	2.7E−06	8.5E−02	6.6E−01	4.5E−08	1.7E−07	1.1E−04	1.1E−04
30%	76.64	78.81	70.46	73.45	75.19	79.52	80.24	80.76	77.48	79.74	2.1E−03	8.2E−01	5.6E−08	2.7E−04	4.0E−01	2.0E−07	6.9E−05	8.8E−02	7.1E−07	5.5E−01	3.3E−01*	4.2E−08	1.4E−07	2.2E−05	5.0E−02
35%	75.17	76.90	68.88	71.05	72.64	77.65	78.13	78.66	75.20	77.26	2.2E−02	8.6E−01	1.0E−07	1.4E−02	6.2E−01	9.5E−08	3.4E−03	2.4E−01	4.6E−07	6.0E−01*	1.4E−01*	1.2E−07	4.5E−08	1.8E−05	4.0E−01
40%	73.13	74.83	66.58	68.17	69.25	75.25	75.61	75.93	72.21	74.15	3.8E−02	8.0E−01*	4.5E−08	1.9E−02	9.7E−01*	1.3E−07	1.4E−02	5.8E−01	3.0E−07	2.8E−01*	6.2E−02*	1.4E−07	1.5E−07	5.5E−06	4.3E−01*
45%	70.38	71.18	64.95	65.58	66.05	71.82	71.97	72.01	68.85	69.95	2.7E−01	7.5E−01	1.1E−07	2.2E−01	6.5E−01	2.0E−07	1.8E−01	5.1E−01	4.0E−07	1.5E−01*	2.3E−01*	8.7E−07	1.5E−07	1.4E−06	4.7E−01*
50%	65.92	60.29	63.06	62.71	62.42	64.46	64.23	64.00	63.67	62.40	2.6E−02*	2.6E−05	1.1E−01	1.8E−02*	3.3E−05	7.8E−02	1.1E−02*	6.6E−05	1.2E−01	7.6E−02*	4.1E−04	5.2E−01	2.5E−02	3.7E−02	2.4E−03
Robustness																									
Uniform class noise																									
5%	2.72	0.58	3.35	0.84	0.33	1.59	0.73	0.61	0.76	0.07	3.2E−03	2.3E−05*	3.3E−07	9.2E−06	2.3E−01*	3.4E−01	3.5E−06	9.5E−01	1.8E−01*	8.7E−05	6.1E−01*	3.6E−08	6.2E−03	4.5E−04*	1.2E−02
10%	4.44	1.10	6.16	1.85	0.95	2.91	1.47	1.12	1.43	0.36	7.2E−03	1.5E−06*	1.0E−07	4.6E−05	9.3E−02*	1.5E−02	3.1E−06	7.1E−01	2.7E−01*	1.2E−04	4.8E−01*	3.6E−08	1.3E−03	3.3E−04*	3.3E−04
15%	6.07	2.64	9.13	3.07	1.67	4.59	2.72	2.31	2.35	1.20	3.6E−02	2.8E−04*	9.3E−07	9.7E−05	6.9E−01*	3.8E−02	1.9E−05	3.4E−01	1.9E−01*	6.7E−04	4.7E−01	4.2E−08	1.0E−04	1.9E−03*	2.3E−04
20%	8.16	3.78	12.10	4.88	2.64	6.40	4.24	3.47	3.85	1.94	1.5E−02	3.1E−05*	1.2E−06	1.1E−04	2.6E−01*	4.4E−02	2.3E−05	3.8E−01	2.8E−01*	2.7E−04	7.4E−01	5.2E−08	5.5E−04	4.1E−04*	3.9E−04
25%	9.89	5.00	14.97	6.84	3.70	8.26	5.54	4.52	5.10	2.93	7.4E−02	1.1E−05*	1.2E−06	2.6E−04	2.5E−01*	3.5E−02	2.7E−05	3.7E−01	5.0E−01*	9.0E−04	9.4E−01*	3.9E−08	5.2E−05	1.4E−03*	2.4E−04
30%	11.38	6.36	18.71	8.60	4.97	9.95	6.92	5.78	6.72	4.08	8.8E−02	4.6E−05*	1.8E−07	1.5E−04	2.8E−01*	2.8E−02	8.6E−06	3.8E−01	5.9E−01*	1.2E−03	4.7E−01*	3.6E−08	1.8E−03	6.7E−04*	8.4E−03
35%	13.16	8.39	21.26	10.99	6.65	12.33	9.10	7.72	8.67	5.83	3.3E−01	3.3E−05*	6.6E−07	5.5E−04	2.6E−01*	3.3E−02	1.3E−04	4.3E−01	9.8E−01*	4.8E−03	5.5E−01*	3.9E−08	7.1E−04	5.8E−04*	4.6E−03
40%	15.08	10.54	24.15	13.59	8.31	14.54	11.20	9.55	10.56	7.60	6.6E−01	8.2E−05*	1.5E−06	2.8E−03	3.0E−01*	9.9E−03	1.7E−04	2.5E−01	9.7E−01*	1.1E−02	6.1E−01*	3.9E−08	1.1E−04	1.4E−03*	5.0E−03
45%	17.07	13.49	27.07	16.96	11.54	17.56	14.10	12.30	13.86	9.77	8.7E−01*	9.7E−05*	8.1E−06	1.2E−02	3.5E−01*	1.2E−02	3.5E−04	3.8E−01	8.6E−01	6.0E−02	6.2E−01*	5.2E−08	2.2E−04	1.4E−02*	7.8E−03
50%	19.47	17.00	30.58	19.56	13.49	21.08	17.41	15.35	16.81	12.99	1.9E−01*	1.3E−04*	1.3E−05	5.1E−02	4.9E−01*	2.1E−02	1.6E−03	1.5E−01	8.4E−01*	1.5E−01	7.1E−01*	4.2E−08	8.2E−04	5.9E−03*	8.8E−03
Pairwise class noise																									
5%	1.72	0.47	2.12	0.58	0.18	0.79	0.34	0.28	0.17	0.27	2.8E−02	1.3E−02*	8.7E−07	9.4E−04	8.0E−01	4.1E−02	7.8E−04	5.5E−01	6.7E−01*	9.4E−04	2.4E−01	1.2E−07	9.9E−05	7.6E−01*	5.4E−01
10%	2.97	1.00	4.20	1.59	0.78	1.73	1.05	0.81	0.76	0.69	6.6E−03	2.0E−04*	6.7E−06	3.9E−04	2.4E−01*	4.2E−03	6.6E−05	5.6E−01	4.6E−01	1.1E−03	8.6E−01	1.8E−07	4.6E−06	7.3E−01*	1.2E−01
15%	4.11	1.82	6.23	3.06	1.64	2.81	1.95	1.53	1.68	1.81	1.4E−02	1.7E−03*	2.1E−06	1.9E−04	1.3E−01*	1.6E−03	5.5E−05	5.1E−01	4.2E−02	2.7E−03	3.9E−01*	1.2E−07	4.0E−06	8.1E−01*	7.8E−01*
20%	4.86	2.66	8.21	5.10	3.33	3.86	2.97	2.51	3.08	2.67	7.2E−02	1.7E−03*	1.0E−05	3.7E−03	9.0E−02*	4.3E−04	4.1E−04	7.4E−01*	3.0E−03	6.8E−02	4.0E−02*	2.1E−07	3.3E−06	2.4E−01	7.2E−01*
25%	6.41	4.33	10.57	7.32	5.24	5.55	4.68	4.16	4.69	4.53	2.5E−01	2.1E−02*	6.7E−06	2.7E−02	4.2E−01*	4.1E−04	4.4E−03	7.5E−01	3.7E−03	3.5E−01	8.8E−02*	1.6E−07	9.9E−07	5.0E−02	3.1E−01*
30%	7.81	5.33	12.75	9.96	7.80	7.11	6.35	5.81	7.16	6.56	3.8E−01	2.7E−03*	6.3E−06	6.4E−02	7.4E−02*	7.3E−05	1.2E−02	3.6E−01*	8.6E−04	5.9E−01*	5.8E−04*	2.6E−07	4.6E−06	3.6E−02	5.0E−03*
35%	9.59	7.68	14.28	12.51	10.53	9.30	8.88	8.32	9.64	9.50	3.8E−01	6.9E−03*	6.6E−05	3.6E−01	1.5E−02*	1.1E−04	1.3E−01	1.1E−01*	3.9E−04	2.7E−01*	1.0E−03*	1.6E−05	8.1E−06	2.3E−02	3.2E−03*
40%	12.01	10.19	17.20	15.93	14.53	12.08	11.83	11.52	13.17	13.09	5.5E−01	7.8E−03*	4.4E−05	4.4E−01	5.6E−03*	5.5E−05	3.3E−01	1.2E−02*	6.9E−05	1.4E−01*	9.7E−05*	6.6E−05	8.6E−06	5.8E−04	1.2E−04*
45%	15.19	14.59	18.96	18.84	18.03	16.05	15.96	16.01	16.92	17.93	7.3E−01*	1.1E−01*	1.0E−03	8.1E−01*	1.2E−01*	3.3E−04	7.7E−01*	9.6E−02*	1.4E−04	8.8E−02*	1.7E−03*	1.9E−02	1.1E−04	2.7E−04	8.7E−05*
50%	20.30	26.70	21.18	21.94	22.11	24.13	24.54	24.90	22.70	26.41	1.4E−04*	2.6E−03	1.1E−01*	5.8E−05*	5.9E−03	7.8E−01*	3.1E−05*	2.4E−02	9.4E−01	4.1E−02*	1.6E−02	8.8E−03*	7.4E−01*	5.8E−01	8.4E−01

2. **Results of MCS5.** MCS5 statistically outperforms SVM (except without induced noise) and C4.5 (up to 25%; they are equivalent at the rest of the noise levels). Statistical differences are also found with 1-NN and 3-NN at all the noise levels and with 5-NN at the lowest noise levels (below of 15%; both are equivalent at the rest of the noise levels). The *p*-values increase with larger values of *k*.
3. **Results of BagC4.5.** BagC4.5 is statistically better than C4.5 at all the noise levels.

- **Performance results with pairwise class noise**.
  1. **Results of MCS3-*k*.**
     - **MCS3-*k* vs. SVM**. MCS3-*k* statistically outperforms its individual components when the noise level is below 45%, whereas it only performs statistically worse than SVM when the noise level reaches 50% (regardless of the value of *k*). The *p*-values at the different noise levels decrease as *k* increases.
     - **MCS3-*k* vs. C4.5**. MCS3-*k* obtains more ranks than C4.5 in most of the cases; moreover, it is statistically better than C4.5 when the noise level is below 15% ($k = 1$), 30% ($k = 3$) and 35% ($k = 5$).
     - **MCS3-*k* vs. *k*-NN**. MCS3-*k* statistically outperforms *k*-NN regardless of the value of *k*. The *p*-values at the different noise levels increase with larger values of *k*.

  2. **Results of MCS5.** In general, MCS5 is statistically better than SVM up to 25% and C4.5 up to 15%; there are not differences at the rest of the noise levels. Statistical differences are also found with *k*-NN regardless the value of *k* (the *p*-values increase with larger values of *k*).
  3. **Results of BagC4.5.** BagC4.5 statistically outperform C4.5 up to a 30% of noise level.

Hereafter, the robustness results obtained are summarized:

- **Robustness results with uniform class noise**.
  1. **Results of MCS3-*k*.**
     - **MCS3-*k* vs. SVM**. MCS3-1 is significantly more robust than SVM up to a noise level of 30%. Both are equivalent from 35% onwards-even though MCS3-1 obtains more ranks at 35–40% and SVM at 45–50%. In the cases of $k = 3$ and $k = 5$, MCS3-*k* statistically outperforms SVM.
     - **MCS3-*k* vs. C4.5**. The robustness of C4.5 excels with respect to MCS3-1, observing the differences found. Regarding MCS3-3 and MCS3-5, they are equivalent to C4.5-although C4.5 obtains more ranks with $k = 3$ and MCS3-*k* with $k = 5$.
     - **MCS3-*k* vs. *k*-NN**. MCS3-*k* is statistically better than *k*-NN (with $k = 1$ and $k = 3$, even though the *p*-values for $k = 3$ are higher than those for $k = 1$). Otherwise, MCS3-5 is statistically equivalent to 5-NN at all the noise levels.

  2. **Results of MCS5.** In general, MCS5 is statistically more robust than SVM, 1-NN and 3-NN, equivalent to C4.5 and less robust than 5-NN.
  3. **Results of BagC4.5.** BagC4.5 statistically outperforms C4.5.
- **Robustness results with pairwise class noise**.
  1. **Results of MCS3-*k*.**
     - **MCS3-*k* vs. SVM**. MCS3-1 statistically overcomes SVM up to a 20% noise level, they are equivalent up to 45% and MCS3-1 is outperformed by SMV at 50%. Similar behavior is shown with MCS3-3 and MCS3-5, but the statistical differences in favor of the MCS3-*k* remain up to 30%.
     - **MCS3-*k* vs. C4.5**. C4.5 is statistically more robust than MCS3-1 (except in highly affected datasets, 45–50%). MCS3-3 and C4.5 are equally robust in most of the cases, even though C4.5 excels at some noise levels. No differences are found between MCS3-5 and C4.5, obtaining larger *p*-values than with the other values of *k*.
     - **MCS3-*k* vs. *k*-NN**. The superiority of MCS3-*k* against *k*-NN is notable, as it is statistically better at all noise levels; in addition, the values increase with the value of *k*.

  2. **Results of MCS5.** MCS5 is statistically more robust than SVM up to 20% of noise level, equivalent up to 40% and worse at 45–50%. It statistically outperforms C4.5 up to 15% and, in general, it is worse onwards. Statistical differences with 1-NN and 3-NN are found (except at the highest noise level). MCS5 is also statistically better than 5-NN between the noise levels 25–45% and it is equivalent at the rest of the noise levels.
  3. **Results of BagC4.5.** BagC4.5 is equivalent up to a noise level of 25% and worse at the rest of the noise levels.

From these performance and robustness results, the following points can be concluded:

1. The uniform scheme is the most disruptive class noise for the majority of the classifiers (single classifiers, MCS3-*k* and MCS5) but for BagC4.5, which is more affected by the pairwise noise. However, from a medium noise level (15–25%) onwards, the pairwise class noise becomes more disruptive for 3-NN and 5-NN (and consequently for MCS3-3, MCS3-5 and also for MCS5). This fact clearly indicates that the behavior of each single classification algorithm trained with noisy data influences that of the corresponding MCS into which it is incorporated (in the case of the MCSs built with

heterogeneous classifiers). In addition, the sampling mechanism used to select the training examples can also affect the results (in the case of BagC4.5). The reasons of the behavior of each type of MCS (built with heterogeneous classifiers and with bagging) with both kinds of class noise are given in the following points:

- The higher disruptiveness of the uniform class noise in MCSs built with heterogeneous classifiers can be attributed to two main reasons: (i) this type of noise affects all the output domain, that is, all the classes, to the same extent, whereas the pairwise scheme only affects the two majority classes; (ii) a noise level $x\%$ with the uniform scheme implies that exactly $x\%$ of the examples in the datasets contain noise, whereas with the pairwise scheme, the number of noisy examples for the same noise level $x\%$ depends on the number of examples of the majority class N_{maj}; as a consequence, the global noise level in the whole dataset is usually lower–more specifically, the number of noisy examples can be computed as $(x \cdot N_{maj})/100$.
  1-NN is very sensitive to the number of noisy examples and it is therefore highly affected by the uniform class noise. Nevertheless, the pairwise class noise affects 3-NN and 5-NN more than the uniform class noise when the noise level is high enough–despite the former generating a lower number of noisy examples. This fact can be attributed to the fact that the uniform class noise generates more examples, but their presence in the output domain is more dispersed (since it affects all the classes). 3-NN and 5-NN take into account more than one neighbor in their prediction, hence, they can correctly predict the class of the examples that are close to those noisy training examples, since the rest of the neighbors are likely to be correctly labeled. However, with the pairwise scheme, it is more likely for these noisy examples to be close to each other, since all of them belonged to the majority class. Therefore, most of the neighbors used in the predictions of 3-NN and 5-NN in these noisy areas are incorrectly labeled, since these methods are more sensitive to this type of noise.
- BagC4.5 is more affected by the pairwise noise than by the uniform noise, whereas C4.5 behaves oppositely. This can be attributed to the diversity that both noise schemes produce in BagC4.5. In this case, the diversity comes from the examples in each sample used to train a classifier, since the baseline classifier is always the same (C4.5). The uniform noise introduces more diversity than the pairwise noise (since the noise affects all the classes instead of only two classes) and BagC4.5 take advantage of this higher diversity. This good behavior of BagC4.5 with the uniform class noise has been also mentioned in previous works [7].

2. The performance results show that:
   - Uniform class noise. Each MCS3-k generally outperforms its single classifier components. MCS3-k is better than SVM and k-NN regardless of the value of k, whereas it only performs statistically better than C4.5 at the lowest noise levels – even though the noise level where MCS3-k is better increases together with the value of k, obtaining significant differences at all the noise levels with $k = 5$. MCS5 is better than SVM, 1-NN and 3-NN but only better than C4.5 and 5-NN at the lowest noise levels. BagC4.5 always outperforms C4.5.
   - Pairwise class noise. MCS3-k improves SVM up to a 40% noise level, it is better than C4.5 at the lowest noise levels – these noise levels are lower than those of the uniform class noise – and also outperforms k-NN. Therefore, the behavior of MCS3-k with respect to their individual components is better in the uniform scheme than in the pairwise one. MCS5 is better than SVM and C4.5 at the lowest noise levels and better than k-NN regardless the value of k. BagC4.5 outperforms C4.5 if the noise levels is not very high.
3. The robustness results show that the higher the value of k, the greater the robustness of the MCS3-k. This fact is particularly notable with the most disruptive noise scheme, i.e., the uniform class noise scheme. Moreover:
   - Uniform class noise. MCS3-k are generally more robust, even though they are never more robust than all their components. The same occurs with MCS5; it is even less robust compared to 5-NN. BagC4.5 is more robust than C4.5.
   - Pairwise class noise. MCS3-k are more robust against uniform class noise than against pairwise noise. They are more robust than their single classifiers at lower noise levels than in the case of the uniform noise (in the case of SVM) or are indeed less robust statistically or equivalent in all the noise levels (in the case of C4.5) – even though MCS3-k is generally more robust than k-NN. In this case, MCS5 is sometimes less robust than its single classifiers and BagC4.5 is equal (at lowest noise levels) or less robust than C4.5.

5.2. Second scenario: Datasets with attribute noise

Table 4 shows the performance and robustness results of each classification algorithm at each noise level on datasets with attribute noise.

The results in this table are summarized hereafter:

- **Performance results with uniform attribute noise**.
  1. **Results of MCS3-*k***.
     - **MCS3-*k* vs. SVM**. MCS3-k statistically outperforms SVM regardless of the value of k. The p-values at the different noise levels decrease when k increases.
     - **MCS3-*k* vs. C4.5**. MCS3-1 performs well at the lowest noise levels (up to 10%), and is equivalent to C4.5 at the rest of the noise levels. Statistical differences are found in favor of MCS3-3 and MCS3-5 up to 15%, and are equivalent to C4.5 at the rest of the noise levels (even though MCS3-5 obtains lower p-values than MCS3-3).

Table 4
Performance and robustness results on datasets with attribute noise.

x%	Results										*p*-values													
	Single methods					Multi-classifiers					MCS3-1 vs.			MCS3-3 vs.			MCS3-5 vs.			MCS5 vs.					BagC4.5 vs.
	SVM	C4.5	1-NN	3-NN	5-NN	MCS3-1	MCS3-3	MCS3-5	MCS5	BagC4.5	SVM	C4.5	1-NN	SVM	C4.5	3-NN	SVM	C4.5	5-NN	SVM	C4.5	1-NN	3-NN	5-NN	C4.5
Performance																									
Uniform attribute noise																									
0%	83.25	82.96	81.42	82.32	82.32	85.42	85.48	85.52	83.74	85.18	5.2E−03	1.8E−03	7.1E−04	1.3E−02	5.0E−04	3.1E−03	1.0E−02	2.8E−04	5.0E−03	4.4E−01	1.9E−02	1.4E−04	9.7E−06	2.2E−03	1.5E−07
5%	82.83	82.62	80.17	81.20	81.38	84.57	84.84	84.77	82.96	84.89	2.8E−03	1.2E−02	2.4E−05	6.7E−04	1.7E−03	8.7E−05	6.4E−04	1.7E−03	9.4E−04	9.8E−02	5.8E−02	5.2E−06	6.6E−07	1.4E−04	1.5E−07
10%	81.78	81.58	78.52	79.78	80.25	83.33	83.64	83.80	81.69	84.36	3.9E−03	8.3E−02	8.1E−06	1.3E−03	8.1E−03	1.4E−04	7.4E−04	1.8E−03	5.8E−04	2.1E−01	7.8E−02	2.0E−06	1.5E−06	1.0E−03	2.3E−07
15%	80.25	80.64	77.22	78.66	79.25	81.81	82.18	82.31	80.61	83.51	5.2E−03	2.8E−01	9.2E−06	3.9E−03	5.8E−02	2.4E−04	2.6E−03	3.8E−02	1.0E−03	1.4E−01	2.4E−01	2.2E−06	2.0E−06	7.4E−04	6.5E−08
20%	78.75	79.98	75.73	77.52	78.40	80.64	81.14	81.42	79.68	82.92	4.4E−03	6.2E−01	5.2E−06	2.1E−03	1.9E−01	7.8E−05	1.5E−03	1.3E−01	3.9E−04	1.3E−01	4.0E−01	4.0E−07	4.6E−07	2.5E−03	4.3E−07
25%	77.58	78.92	74.26	76.20	77.39	79.37	79.85	80.22	78.53	82.03	2.8E−03	9.7E−01	3.8E−06	2.1E−03	5.1E−01	1.3E−04	8.2E−04	1.6E−01	5.5E−04	1.2E−01	9.6E−01	3.0E−07	5.0E−07	5.9E−03	1.7E−07
30%	76.09	77.64	72.58	74.84	76.04	77.97	78.42	78.75	77.14	80.99	2.1E−03	9.4E−01*	1.0E−05	1.1E−03	6.9E−01	2.7E−04	3.7E−04	2.3E−01	4.8E−03	5.8E−02	9.1E−01*	4.6E−07	1.5E−06	6.6E−03	1.7E−07
35%	74.11	76.33	71.11	73.68	74.95	76.26	76.91	77.23	75.96	79.94	5.2E−04	7.8E−01*	1.2E−05	4.7E−04	7.8E−01	2.6E−04	1.2E−04	3.8E−01	7.5E−03	2.1E−02	9.4E−01*	3.5E−07	4.3E−07	5.3E−03	4.3E−07
40%	72.75	75.19	69.58	72.22	73.60	74.84	75.57	76.03	74.58	78.76	9.9E−03	7.6E−01*	1.3E−06	2.3E−03	7.7E−01	1.6E−04	1.0E−03	4.0E−01	2.8E−03	6.8E−02	7.4E−01*	1.5E−07	8.1E−07	4.6E−03	1.3E−06
45%	70.51	73.38	68.10	70.85	72.38	72.93	73.70	74.27	73.03	77.38	1.9E−03	6.3E−01*	3.7E−05	9.4E−04	9.7E−01	1.2E−03	2.4E−04	4.5E−01	1.4E−02	3.2E−02	7.0E−01*	2.0E−06	4.0E−06	4.2E−02	2.0E−06
50%	69.46	72.12	66.59	69.16	70.87	71.36	72.14	72.85	71.57	76.18	1.2E−02	3.1E−01*	2.7E−05	2.4E−03	6.8E−01*	5.2E−04	6.1E−04	7.9E−01	7.8E−03	5.6E−02	5.5E−01*	6.1E−07	3.3E−06	5.1E−02	1.8E−06
Gauss ian attribute noise																									
0%	83.25	82.96	81.42	82.32	82.32	85.42	85.48	85.52	83.74	85.18	5.2E−03	1.8E−03	7.1E−04	1.3E−02	5.0E−04	3.1E−03	1.0E−02	2.8E−04	5.0E−03	4.4E−01	1.9E−02	1.4E−04	9.7E−06	2.2E−03	1.5E−07
5%	83.40	82.70	80.79	81.87	81.89	85.12	85.27	85.29	83.45	85.27	2.4E−03	3.0E−03	4.0E−05	2.3E−03	4.5E−04	6.4E−04	3.5E−03	3.3E−04	3.0E−03	2.4E−01	1.9E−02	7.6E−06	3.5E−06	6.4E−04	1.2E−07
10%	82.83	82.15	80.08	81.25	81.32	84.49	84.62	84.55	82.89	84.71	3.4E−03	4.8E−03	4.0E−05	1.6E−03	1.7E−03	7.8E−04	2.5E−03	1.9E−03	2.5E−03	2.4E−01	2.1E−02	1.5E−05	1.7E−06	3.5E−04	2.4E−07
15%	82.35	81.74	79.30	80.55	80.93	84.03	84.24	84.35	82.43	84.51	1.2E−02	5.4E−03	1.0E−05	7.8E−03	7.7E−04	4.0E−05	1.9E−03	5.4E−04	1.1E−03	1.9E−01	2.2E−02	8.1E−06	6.6E−07	1.6E−03	6.1E−08
20%	81.62	81.16	78.52	79.95	80.46	83.33	83.52	83.65	81.83	84.00	4.4E−03	2.1E−02	1.0E−05	2.7E−03	5.9E−03	1.9E−04	5.2E−04	2.6E−03	1.2E−03	1.4E−01	7.6E−02	8.1E−06	8.7E−07	2.4E−03	5.4E−07
25%	81.14	80.60	77.70	79.42	80.03	82.62	82.92	83.16	81.41	83.44	3.4E−03	4.3E−02	4.0E−05	8.6E−04	4.8E−03	5.8E−04	3.5E−04	2.0E−03	3.7E−03	1.1E−01	6.0E−02	1.4E−06	9.3E−07	4.0E−03	2.0E−07
30%	80.48	80.25	76.74	78.42	79.20	81.85	82.15	82.37	80.56	83.08	1.1E−02	2.0E−01	1.0E−05	2.7E−03	6.4E−02	1.6E−04	9.9E−04	1.9E−02	1.4E−03	9.8E−02	1.9E−01	2.0E−06	2.8E−07	1.2E−03	3.5E−07
35%	79.76	79.25	75.85	77.68	78.46	81.03	81.32	81.54	79.75	82.27	2.6E−02	1.2E−01	2.0E−05	1.2E−02	3.0E−02	7.8E−04	7.8E−03	5.4E−03	2.5E−03	3.1E−01	1.5E−01	1.6E−06	1.9E−06	2.1E−03	1.1E−07
40%	78.88	78.84	74.82	77.11	78.03	80.34	80.87	81.04	79.32	81.87	1.6E−02	4.0E−01	1.0E−05	4.0E−03	6.8E−02	2.2E−04	3.2E−03	1.9E−02	2.4E−03	2.1E−01	3.6E−01	3.3E−07	8.1E−07	5.6E−03	7.3E−07
45%	78.27	77.80	74.01	76.02	76.99	79.45	79.71	79.98	78.25	81.04	3.5E−02	2.6E−01	0.0E + 00	1.3E−02	1.3E−01	5.5E−04	6.6E−03	3.7E−02	1.9E−03	4.3E−01	3.1E−01	8.1E−07	3.1E−06	7.2E−03	2.4E−06
50%	77.26	77.01	73.31	75.59	76.50	78.49	79.08	79.21	77.86	80.39	2.2E−02	3.4E−01	3.0E−05	3.9E−03	7.6E−02	3.0E−04	2.9E−03	3.0E−02	5.4E−03	2.9E−01	2.7E−01	8.1E−07	1.8E−06	1.1E−03	2.4E−06
Robustness																									
Uniform attribute noise																									
5%	0.16	0.39	1.38	1.28	0.97	0.95	0.71	0.85	0.94	0.34	2.8E−01	1.3E−01*	2.6E−01	8.1E−02	3.7E−01*	4.9E−02	2.4E−01	1.7E−01*	2.7E−02	2.6E−01	6.0E−02*	1.2E−02	2.4E−03	4.7E−02	5.2E−01
10%	1.38	1.68	3.63	3.29	2.52	2.40	2.11	1.98	2.57	0.91	6.7E−01*	6.6E−02*	1.3E−02	5.5E−01	3.0E−01*	7.5E−03	4.1E−01	4.6E−01*	4.8E−02	8.4E−01*	1.5E−01*	7.8E−04	4.5E−04	5.6E−01	7.8E−03
15%	3.28	2.84	5.19	4.29	3.39	4.23	3.86	3.76	3.63	1.95	8.6E−01*	1.8E−02*	7.8E−02	5.9E−01	5.5E−02*	8.3E−02	5.3E−01	3.7E−02*	3.3E−01	6.5E−01	1.8E−01*	6.7E−04	7.2E−03	5.5E−01	7.8E−03
20%	5.06	3.60	6.78	5.53	4.35	5.54	5.02	4.80	4.68	2.63	6.1E−01	5.2E−03*	1.3E−02	4.3E−01	3.7E−02*	2.7E−02	3.2E−01	3.2E−02*	1.1E−01	3.5E−01	8.5E−02*	2.6E−05	3.7E−03	8.2E−01	1.2E−02
25%	6.45	4.98	8.47	7.04	5.41	7.09	6.61	6.23	6.10	3.73	9.8E−01	5.9E−03*	9.5E−03	7.6E−01	2.2E−02*	1.9E−02	2.9E−01	5.5E−02*	1.1E−01	5.5E−01	3.2E−02*	3.1E−05	6.9E−03	7.3E−01*	4.6E−03
30%	8.35	6.62	10.73	8.67	7.17	8.85	8.36	8.06	7.84	5.00	7.8E−01	2.6E−03*	1.2E−02	6.7E−01	6.9E−03*	6.0E−02	2.9E−01	2.2E−02*	2.6E−01	5.0E−01	4.0E−02*	4.6E−05	5.4E−03	1.0E + 00	5.2E−04
35%	10.62	8.23	12.32	9.93	8.13	10.79	10.12	9.82	9.17	6.24	3.1E−01	4.4E−03*	2.0E−02	2.3E−01	2.8E−02*	4.1E−02	1.0E−01	4.1E−02*	4.4E−01	1.7E−01	8.5E−02*	3.5E−05	5.9E−03	8.6E−01*	1.4E−04
40%	12.13	9.60	14.29	11.64	9.71	12.44	11.67	11.23	10.74	7.57	7.8E−01*	1.1E−02*	2.0E−03	8.1E−01	4.8E−02*	2.0E−02	5.8E−01	9.3E−02*	1.3E−01	6.6E−01	7.4E−02*	2.4E−06	4.8E−03	9.0E−01	1.2E−03
45%	14.92	11.87	16.04	13.28	10.91	14.73	13.98	13.38	12.75	9.31	6.9E−01	1.1E−02*	2.5E−02	5.5E−01	6.8E−02*	9.3E−02	2.2E−01	1.3E−01*	3.5E−01	4.2E−01	1.6E−01*	2.4E−04	9.9E−03	6.6E−01*	3.2E−04
50%	16.28	13.46	17.70	15.17	12.48	16.68	15.85	15.09	14.40	10.75	8.5E−01*	3.4E−03*	2.0E−02	6.9E−01	4.0E−02*	3.0E−02	2.9E−01	1.3E−01*	2.9E−01	4.6E−01	1.0E−01*	1.6E−04	3.4E−03	7.5E−01*	3.3E−04
Gaussian attribute noise																									
5%	−0.46	0.33	0.80	0.50	0.58	0.31	0.19	0.25	0.32	−0.16	7.8E−02	1.4E−01*	3.2E−01	6.8E−02	9.6E−01	3.3E−01	1.2E−01	6.0E−01*	5.4E−01	1.6E−01	6.4E−01*	8.1E−02	1.2E−01	1.2E−01	8.9E−02
10%	0.25	0.94	1.60	1.28	1.20	1.03	0.98	1.16	0.92	0.53	1.7E−01	2.4E−01*	2.6E−01	2.5E−01	4.8E−01*	9.0E−02	7.7E−01	2.3E−01*	4.1E−01	2.2E−01	8.4E−01*	3.5E−02	2.2E−02	6.4E−02	1.7E−01
15%	0.75	1.48	2.51	2.09	1.84	1.54	1.39	1.32	1.48	0.77	9.9E−01*	1.3E−01*	5.3E−02	5.5E−01	6.0E−01*	1.6E−02	4.1E−01	7.5E−01*	2.2E−01	6.2E−01	6.5E−01*	8.2E−04	5.9E−03	5.3E−01	3.8E−02
20%	1.74	2.10	3.36	2.74	2.03	2.36	2.21	2.14	2.07	1.35	3.3E−01	1.2E−01*	1.0E−01	8.3E−02	1.4E−01*	2.6E−02	9.3E−02	3.3E−01*	3.1E−01	2.8E−01	2.0E−01*	8.8E−03	1.8E−03	8.3E−01	2.0E−01
25%	2.32	2.89	4.25	3.27	2.67	3.20	2.95	2.73	2.56	2.03	6.2E−01	1.7E−01*	5.8E−02	3.9E−01	3.1E−01*	8.5E−02	2.1E−01	6.2E−01*	5.0E−01	2.6E−01	7.0E−01*	3.9E−04	8.1E−03	6.6E−01	4.2E−02
30%	3.14	3.25	5.64	4.70	3.80	4.14	3.86	3.65	3.76	2.44	7.3E−01*	1.3E−02*	7.2E−02	5.4E−01	3.8E−02*	2.6E−02	2.6E−01	1.1E−01*	2.5E−01	4.0E−01	1.4E−01*	3.2E−03	1.3E−03	7.3E−01	5.0E−02
35%	4.11	4.52	6.67	5.60	4.48	5.09	4.82	4.63	4.62	3.38	7.1E−01*	1.4E−01*	4.5E−02	8.7E−01	3.8E−01*	5.6E−02	5.4E−01	5.8E−01*	4.6E−01	7.7E−01	4.9E−01*	2.1E−04	1.5E−03	9.8E−01*	6.0E−03
40%	5.23	5.02	7.91	6.17	4.96	5.93	5.38	5.24	5.11	3.89	9.4E−01	1.5E−02*	2.6E−03	4.0E−01	2.4E−01*	7.2E−03	3.8E−01	2.0E−01*	5.3E−01	4.0E−01	3.1E−01*	6.7E−06	1.0E−03	6.8E−01	4.2E−02
45%	6.02	6.30	8.96	7.54	6.39	7.03	6.80	6.53	6.38	4.88	8.3E−01*	1.8E−01*	1.4E−02	9.1E−01*	3.5E−01*	3.6E−02	9.4E−01*	5.5E−01*	7.2E−01	7.9E−01	4.5E−01*	4.1E−05	1.8E−03	6.3E−01	2.7E−02
50%	7.28	7.29	9.51	7.75	6.68	8.14	7.50	7.43	6.68	5.60	9.8E−01	6.0E−02*	5.1E−02	4.4E−01	3.9E−01*	3.0E−02	5.4E−01	2.6E−01*	5.9E−01	3.4E−01	4.8E−01*	9.2E−05	1.6E−03	5.8E−01	6.6E−03

   - **MCS3-*k* vs. *k*-NN**. MCS3-*k* is statistically better than *k*-NN regardless of the value of *k*. The *p*-values increase together with *k*.

2. **Results of MCS5.** MCS5 outperforms *k*-NN for the different values of *k* (the *p*-values increase together with *k*). MCS5 is equivalent to SVM below of the noise level 30% and better from this level. MCS5 is better than C4.5 at the lowest noise levels (up to 10%), equivalent at intermediate noise levels (15–25%) and worse at the rest of the noise levels.
3. **Results of BagC4.5.** BagC4.5 is better than C4.5 at all the noise levels.

- **Performance results with Gaussian attribute noise**.
  1. **Results of MCS3-*k*.**
     - **MCS3-*k* vs. SVM**. MCS3-*k* is significantly better than SVM regardless of the *k* value. The *p*-values at the different noise levels decrease as *k* increases.
     - **MCS3-*k* vs. C4.5**. MCS3-*k* performs well up to 25% ($k = 1$) and at all noise levels for $k = 3$ and $k = 5$. The *p*-values at the different noise levels decrease as *k* increases.
     - **MCS3-*k* vs. *k*-NN**. MCS3-*k* performs statistically better than *k*-NN regardless of the value of *k*. The *p*-values at the different noise levels increase with greater values of *k*.

  2. **Results of MCS5.** MCS5 outperforms *k*-NN for the different values of *k* (the *p*-values increase together with *k*). MCS5 works well below 30% compared with C4.5 and both are equivalent onwards. MCS5 is also equivalent to SVM at all the noise levels.
  3. **Results of BagC4.5.** BagC4.5 is better than C4.5 at all the noise levels.

The robustness results obtained are summarized below:

- **Robustness results with uniform attribute noise**.
  1. **Results of MCS3-*k*.**
     - **MCS3-*k* vs. SVM**. MCS3-*k* is equivalent to SVM regardless of the value of *k*. MCS3-*k* generally obtains more ranks, except for some cases with $k = 1$. The *p*-values at the different noise levels decrease as *k* increases.
     - **MCS3-*k* vs. C4.5**. C4.5 obtains more ranks, outperforming MCS3-*k*. This superiority is not significant at the lowest noise levels (up to 5–10%). The *p*-values at the different noise levels increase with *k*.
     - **MCS3-*k* vs. *k*-NN**. MCS3-*k* obtains more ranks than *k*-NN. Statistical differences are found with with $k = 1$ and $k = 3$. MCS3-5 outperforms 5-NN at the lowest noise levels (5% and 10%) and is equivalent at the rest. The *p*-values at the different noise levels increase with the value of *k*.

  2. **Results of MCS5.** MCS5 is equivalent to SVM, 5-NN and C4.5 (it is even worse at some isolated noise levels). However, it is better than 1-NN and 3-NN.
  3. **Results of BagC4.5.** Generally, BagC4.5 obtains more ranks than C4.5.
- **Robustness results with Gaussian attribute noise**.
  1. **Results of MCS3-*k*.**
     - **MCS3-*k* vs. SVM**. No remarkable differences are found between MCS3-*k* and SVM. For $k = 1$ and $k = 3$, MCS3-*k* is only better at 5%. In most of the cases MCS3-*k* obtains more ranks than SVM, although some exceptions are found with $k = 1$.
     - **MCS3-*k* vs. C4.5**. C4.5 obtains more ranks, although both methods are statistically equivalent. Moreover, C4.5 is better at some noise levels with $k = 1$.
     - **MCS3-*k* vs. *k*-NN**. MCS3-*k* obtains more ranks than *k*-NN, although statistical differences are only found with $k = 1$ and $k = 3$. The *p*-values with different noise levels increase together with *k*.

  2. **Results of MCS5.** MCS5 is equivalent to SVM, C4.5 and 5-NN and better than 1-NN and 3-NN at most of the noise levels.
  3. **Results of BagC4.5.** Generally, BagC4.5 is better than C4.5 except at some isolated noise levels.

From these performance and robustness results, the following remarks can be made:

1. The results on datasets with uniform attribute noise are much worse than those on datasets with Gaussian noise for all the classifiers, including MCSs. Hence, the most disruptive attribute noise is the uniform scheme.
2. The results of performance on datasets with attribute show:
   - Uniform attribute noise. This is the most disruptive noise scheme; MCS3-*k* outperforms SVM and *k*-NN. However, with respect to C4.5, MCS3-*k* is significantly better at the lowest noise levels (up to 10–15%), and is equivalent at the rest of the noise levels. MCS5 is better than *k*-NN regardless the value of *k* and it is better than SVM only from an intermediate noise level (30%). The results of MCS5 compared to C4.5 depends on the noise level: they are better if the noise level is lower. BagC4.5 outperforms C4.5.

- Gaussian attribute noise. This is the least disruptive noise scheme; MCS3-*k* outperforms its individual classifiers in most of the cases, particularly if the noise level is relatively low. Thus, MCS3-*k* is generally the best method with $k = 3$ and $k = 5$. In the case of $k = 1$, MCS3-1 is only better than 1-NN and SVM, and better than C4.5 at the lowest noise levels (up to 25%). MCS5 is better than *k*-NN regardless the value of *k*, equivalent to SVM and it is better or equivalent to C4.5 (depending if the noise level is low or high, respectively). BagC4.5 also outstands over C4.5 in this case.

3. The robustness results show, as in the case of the robustness results with class noise, that the higher the value of *k* is, the greater robustness the MCS3-*k* has. This fact is particularly notable with the most disruptive noise scheme (the uniform one) but also when the noise level becomes high in both schemes, uniform and Gaussian. Moreover, the following points must be stressed:
   - Uniform attribute noise. The robustness of the MCS3-*k* does not outperform that of its individual classifiers, as it is statistically equivalent to SVM and sometimes worse than C4.5. Regarding *k*-NN, MCS3-*k* performs better than 1-NN and 3-NN, but is equivalent to 5-NN. The same occurs with MCS5, since it is equivalent to SVM, 5-NN and C4.5 and, at some noise levels, even worse than C4.5. The exception is with BagC4.5, which is more robust than C4.5.
   - Gaussian attribute noise. The robustness results are better than those of the uniform noise. The main difference in this case is that MCS3-*k* and MCS5 are not statistically worse than C4.5. BagC4.5 is also more robust than C4.5, except at some isolated noise levels.

6. Comparing different decisions combination methods

In this section a series of decisions combination methods (W-MAJ, NB and BKS) are compared with respect to the MAJ scheme used in the previous section. The performance and robustness of the MCSs built with heterogeneous classifiers (MCS3-*k* and MCS5) using these combination methods are studied. The results of each one of these three techniques (W-MAJ, NB and BKS) are compared to those of the MAJ scheme using Wilcoxon test. Table 5 shows the performance and robustness results of the decisions combination methods on datasets with class noise.

From this table, it can be observed that:

1. **Performance results:**
   (a) **Uniform class noise.**
   - Attending to the average results, MAJ is generally the best method, followed by NB; W-MAJ and BKS are placed to a certain distance of the previous ones.
   - The *p*-values obtained show that MAJ is better than W-MAJ and NB (from a noise level of 5–10% approximately). At the lowest noise levels W-MAJ and NB are even better than MAJ. In addition, MAJ is better than BKS.

   (b) **Pairwise class noise.**
   - The average results show that, in MCS3-*k*, MAJ is the best method, followed by NB and W-MAJ (with equaled results at some noise levels). The worst method is BKS. In MCS5, the situation is different: the best methods are W-MAJ and NB, followed by BKS and MAJ to a certain distance.
   - Attending to the *p*-values, W-MAJ and NB are usually better than MAJ in MCS3-*k* at extreme noise levels (0–5% and 45–50%). In MCS5, W-MAJ and NB are clearly better than MAJ, but MAJ is better than BKS (except at the highest noise levels–from a noise level of 40% onwards).

2. **Robustness results:**
   (a) **Uniform class noise.**
   - W-MAJ and BKS share the best average results, followed by NB and MAJ.
   - However, the *p*-values show that MAJ is more robust than the rest of the methods.

   (b) **Pairwise class noise.**
   - The average results show that, in MCS3-*k*, W-MAJ and NB are the best methods, followed by BKS and MAJ in the last position. In MCS5, NB and BKS are the first methods in average results, followed by W-MAJ and MAJ (except at the highest noise levels where MAJ is better).
   - The *p*-values show that MAJ is more robust up to an intermediate noise level (25–30%). Then, the rest of the methods are equivalent or even more robust (45–50% of noise level approximately).

Table 6 shows the performance and robustness results of the decisions combination methods on datasets with attribute noise.

From this table, it can be observed that:

1. **Performance results.**
   (a) **Uniform attribute noise.**

Table 5
Performance and robustness results on datasets with class noise: comparison of decisions combination methods.

x%	Results																p-values											
	Majority				Weighted				Bayes				BKS				Weighted				Bayes				BKS			
	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5
Performance																												
Uniform class noise																												
0%	85.42	85.48	85.52	83.74	84.92	85.04	85.04	85.83	85.03	85.22	85.29	86.10	84.25	84.39	84.39	84.72	9.2E−01°	8.0E−01°	5.0E−01	3.0E−02°	9.3E−01°	9.2E−01°	9.5E−01	1.2E−02°	2.4E−02	3.4E−02	2.6E−02	7.8E−01
5%	84.11	84.86	85.03	83.15	82.27	83.05	83.17	84.61	82.36	82.61	82.73	84.57	81.54	81.48	81.51	82.26	1.5E−01	4.9E−02	2.1E−01	2.5E−01°	6.2E−02	4.9E−03	1.0E−02	2.9E−01°	5.4E−04	1.9E−05	5.9E−06	2.8E−02
10%	83.00	84.25	84.57	82.60	80.47	80.83	80.94	83.15	80.84	81.25	81.29	83.14	79.93	80.04	80.09	81.02	1.2E−02	4.2E−04	6.4E−05	6.0E−01	2.0E−02	9.0E−05	1.8E−04	9.1E−01	3.1E−05	5.5E−07	2.5E−07	5.6E−03
15%	81.60	83.21	83.63	81.74	79.02	79.43	79.44	80.98	79.23	79.51	79.67	81.28	78.37	78.37	78.31	79.60	5.7E−03	2.1E−04	6.6E−05	1.3E−01	3.5E−03	2.9E−05	2.2E−05	2.9E−01	2.3E−05	6.1E−07	1.3E−07	2.0E−03
20%	80.09	81.98	82.69	80.55	77.02	77.45	77.68	78.03	77.40	77.70	77.86	79.33	76.47	76.60	76.57	77.91	1.1E−03	4.2E−05	1.3E−05	6.4E−02	9.2E−04	5.5E−06	3.1E−06	9.6E−02	1.4E−06	9.8E−08	8.2E−08	1.2E−03
25%	78.53	80.87	81.78	79.50	74.86	75.73	76.12	75.42	75.54	75.93	76.18	77.57	74.71	74.94	74.97	76.53	3.4E−04	2.2E−05	1.3E−05	1.4E−02	7.1E−04	6.3E−06	9.2E−07	3.2E−02	5.2E−06	8.8E−08	4.8E−08	1.0E−03
30%	77.10	79.69	80.75	78.18	72.68	73.54	74.35	73.56	74.17	74.60	74.80	76.06	73.18	73.37	73.36	75.01	1.5E−04	2.0E−06	4.4E−06	5.1E−03	2.4E−04	1.3E−06	3.7E−07	2.3E−02	2.4E−06	8.2E−08	4.2E−08	9.4E−04
35%	75.11	77.87	79.16	76.44	70.91	71.37	71.92	71.39	72.37	72.80	73.15	74.29	71.46	71.66	71.62	73.15	9.6E−05	2.2E−06	1.9E−06	3.5E−03	3.5E−04	1.9E−06	9.0E−07	1.9E−02	3.8E−06	5.6E−08	3.6E−08	4.7E−04
40%	73.20	76.04	77.56	74.72	68.95	69.37	69.94	68.77	70.46	70.97	71.37	72.31	69.45	69.59	69.65	71.08	9.0E−05	5.4E−06	3.8E−06	3.4E−04	8.3E−04	7.5E−06	3.4E−06	1.8E−02	6.7E−06	4.3E−07	1.7E−07	3.2E−04
45%	70.68	73.61	75.27	72.16	66.57	67.30	67.75	66.19	68.45	68.91	69.30	70.08	67.35	67.68	67.63	69.01	1.1E−04	5.8E−06	2.1E−06	4.5E−04	2.2E−03	2.0E−05	2.6E−06	1.2E−02	1.2E−05	8.7E−07	1.7E−07	1.1E−03
50%	67.64	70.82	72.70	69.43	63.33	64.33	64.81	63.50	65.99	66.30	66.76	67.40	64.68	65.01	65.07	66.19	1.7E−03	2.6E−06	1.7E−06	2.2E−04	5.6E−02	2.5E−05	5.0E−06	3.3E−02	2.1E−04	8.1E−07	2.6E−07	1.7E−03
Pairwise class noise																												
0%	85.42	85.48	85.52	83.74	84.92	85.04	85.04	85.83	85.03	85.22	85.29	86.10	84.25	84.39	84.39	84.72	9.2E−01°	8.0E−01°	5.0E−01	3.0E−02°	9.3E−01°	9.2E−01°	9.5E−01	1.2E−02°	2.4E−02	3.4E−02	2.6E−02	7.8E−01
5%	84.75	85.19	85.28	83.54	83.50	84.12	84.18	85.19	83.43	83.63	83.66	85.01	82.74	82.76	82.78	83.27	1.2E−01	1.1E−01	1.3E−01	2.3E−01°	5.5E−03	7.1E−04	1.9E−03	6.4E−01°	8.7E−05	1.1E−05	2.3E−05	1.9E−02
10%	83.95	84.61	84.86	82.99	81.83	82.11	82.24	84.43	82.33	82.42	82.52	83.59	81.51	81.61	81.60	82.17	2.5E−02	3.1E−02	1.7E−02	2.6E−01°	1.9E−03	2.7E−04	1.8E−04	3.2E−01	3.3E−05	2.9E−06	1.1E−06	6.1E−03
15%	83.09	83.89	84.29	82.23	80.81	81.08	81.28	83.36	81.19	81.36	81.31	82.41	80.37	80.51	80.53	81.04	1.5E−02	3.1E−02	9.3E−03	2.9E−01°	1.4E−03	2.2E−04	6.9E−05	1.6E−01	1.3E−05	2.1E−06	4.6E−07	9.1E−03
20%	82.21	83.06	83.50	80.99	80.18	80.49	80.59	82.33	80.49	80.67	80.63	81.43	79.71	79.84	79.87	80.38	3.3E−02	2.8E−02	4.9E−03	2.1E−01°	2.1E−03	7.1E−04	1.2E−04	3.7E−01	6.9E−05	9.2E−06	2.5E−06	4.7E−02
25%	80.81	81.64	82.14	79.54	78.83	79.12	79.20	80.25	78.96	79.17	79.22	79.80	78.33	78.47	78.57	79.06	1.0E−02	9.8E−03	6.5E−03	2.8E−01°	5.0E−03	1.7E−03	1.1E−03	2.8E−01	2.1E−04	3.7E−05	1.4E−05	9.8E−02
30%	79.52	80.24	80.76	77.48	77.64	77.90	77.99	78.62	77.90	78.03	78.09	78.29	77.25	77.36	77.50	77.84	8.4E−02	5.6E−02	1.2E−02	7.4E−02°	4.0E−02	9.1E−03	3.0E−03	9.1E−01	2.2E−03	1.8E−04	6.2E−05	7.7E−01
35%	77.65	78.13	78.66	75.20	76.09	76.42	76.58	76.98	76.42	76.56	76.51	76.73	75.71	75.81	75.86	76.17	4.6E−01	2.8E−01	1.1E−01	6.5E−03°	2.2E−01	1.3E−01	4.1E−02	2.9E−01°	2.1E−02	1.0E−02	2.5E−03	5.6E−01°
40%	75.25	75.61	75.93	72.21	74.09	74.23	74.27	74.87	74.50	74.58	74.55	75.03	73.96	74.05	74.09	74.27	7.4E−01	3.2E−01	2.4E−01	1.0E−03°	4.9E−01	2.6E−01	1.4E−01	6.1E−03°	1.4E−01	5.1E−02	2.9E−02	8.5E−02°
45%	71.82	71.97	72.01	68.85	71.65	71.54	71.44	71.75	71.61	71.77	71.85	72.60	71.26	71.35	71.35	71.31	5.8E−01°	7.6E−01°	9.5E−01°	8.3E−05°	9.6E−01°	9.9E−01	9.1E−01	6.2E−05°	4.6E−01	4.5E−01	4.3E−01	4.7E−02°
50%	64.46	64.23	64.00	63.67	65.77	65.74	65.71	66.15	67.08	67.26	67.32	68.96	66.67	66.63	66.67	66.64	1.2E−02°	8.1E−03°	4.7E−03°	3.4E−03°	1.7E−04°	4.1E−05°	1.6E−05°	2.6E−07°	2.1E−03°	1.6E−03°	4.1E−04°	6.9E−03°
Robustness																												
Uniform class noise																												
5%	1.59	0.73	0.61	0.76	3.18	2.50	2.35	1.62	3.27	3.18	3.10	1.85	3.34	3.49	3.44	2.83	2.7E−02	9.8E−03	8.5E−02	5.8E−02	3.0E−04	1.5E−06	1.1E−05	1.5E−04	1.6E−04	4.3E−07	3.8E−07	7.3E−05
10%	2.91	1.47	1.12	1.43	5.22	4.99	4.88	3.42	5.04	4.77	4.78	3.46	5.23	5.18	5.08	4.26	4.4E−03	7.6E−04	1.6E−04	1.1E−03	8.8E−05	9.4E−06	1.7E−06	8.6E−04	4.9E−05	1.5E−06	4.0E−07	5.2E−05
15%	4.59	2.72	2.31	2.35	6.98	6.75	6.75	6.14	7.00	6.88	6.76	5.70	7.11	7.21	7.24	5.99	3.0E−02	2.1E−04	2.0E−04	1.2E−03	7.5E−04	1.9E−06	7.9E−07	2.3E−04	8.2E−04	7.6E−06	1.1E−06	1.4E−04
20%	6.40	4.24	3.47	3.85	9.37	9.10	8.87	9.67	9.15	9.01	8.91	7.99	9.42	9.35	9.37	8.06	3.0E−03	1.4E−04	1.9E−04	4.4E−04	1.7E−04	1.2E−06	2.4E−06	2.0E−04	4.4E−05	8.7E−07	1.4E−06	1.8E−05
25%	8.26	5.54	4.52	5.10	11.94	11.17	10.71	12.66	11.32	11.09	10.86	10.02	11.51	11.32	11.25	9.64	3.1E−03	8.0E−05	1.3E−04	3.7E−04	2.5E−04	3.1E−06	2.3E−07	1.5E−04	8.7E−05	9.9E−07	2.6E−07	3.7E−05
30%	9.95	6.92	5.78	6.72	14.63	13.85	12.87	14.93	12.93	12.64	12.48	11.79	13.29	13.15	13.13	11.37	1.7E−03	6.2E−06	6.1E−06	1.8E−04	4.4E−04	4.7E−07	3.4E−07	1.3E−04	4.6E−05	7.6E−07	2.6E−07	1.0E−04
35%	12.33	9.10	7.72	8.67	16.68	16.42	15.75	17.44	15.04	14.75	14.41	13.85	15.32	15.16	15.20	13.58	3.5E−03	1.2E−05	1.8E−05	8.2E−05	1.5E−03	2.6E−06	1.0E−06	2.6E−04	2.7E−04	2.4E−06	1.3E−06	3.5E−04
40%	14.54	11.20	9.55	10.56	18.99	18.77	18.07	20.46	17.25	16.86	16.49	16.12	17.70	17.58	17.51	15.95	2.6E−03	2.3E−05	1.1E−05	3.0E−05	3.5E−03	8.5E−06	2.9E−06	3.2E−04	2.2E−04	1.0E−05	4.3E−06	1.3E−04
45%	17.56	14.10	12.30	13.86	21.89	21.26	20.76	23.53	19.66	19.34	18.97	18.76	20.23	19.90	19.96	18.43	2.9E−03	1.7E−05	5.4E−06	7.3E−05	1.7E−02	4.1E−05	4.4E−06	8.6E−04	7.4E−04	2.3E−05	5.2E−06	9.4E−04
50%	21.08	17.41	15.35	16.81	25.63	24.67	24.07	26.38	22.47	22.33	21.87	21.80	23.29	23.00	22.91	21.68	1.4E−02	2.3E−05	4.0E−06	1.5E−05	1.7E−01	8.3E−05	1.2E−05	1.6E−03	9.1E−03	2.2E−05	6.7E−06	1.7E−03
Pairwise class noise																												
5%	0.79	0.34	0.28	0.17	1.65	1.17	1.09	0.83	1.83	1.83	1.87	1.27	1.74	1.88	1.84	1.67	1.2E−02	2.3E−02	6.4E−02	6.9E−03	2.2E−04	7.3E−05	6.5E−05	1.7E−05	6.7E−04	3.5E−05	2.1E−04	1.4E−04
10%	1.73	1.05	0.81	0.76	3.56	3.44	3.31	1.75	3.13	3.25	3.20	2.91	3.18	3.22	3.24	2.94	5.5E−03	5.2E−03	1.6E−02	1.5E−02	2.4E−04	1.3E−05	5.9E−06	2.4E−05	2.8E−04	9.2E−05	1.8E−05	2.1E−04
15%	2.81	1.95	1.53	1.68	4.72	4.61	4.40	3.00	4.39	4.45	4.58	4.22	4.51	4.50	4.46	4.27	2.7E−02	8.0E−03	5.8E−03	5.8E−02	7.1E−04	3.1E−05	2.5E−06	4.6E−05	4.3E−04	3.5E−05	1.1E−05	3.5E−04
20%	3.86	2.97	2.51	3.08	5.46	5.32	5.22	4.19	5.23	5.28	5.38	5.35	5.28	5.26	5.20	4.98	1.2E−01	4.5E−02	1.2E−02	5.8E−01	6.9E−03	6.7E−04	4.1E−05	9.0E−04	1.3E−02	1.3E−03	3.3E−04	2.5E−02
25%	5.55	4.68	4.16	4.69	7.06	6.95	6.87	6.58	7.01	7.01	7.02	7.23	6.87	6.86	6.73	6.52	2.3E−01	5.9E−02	3.3E−02	8.2E−01	1.9E−02	8.1E−03	1.1E−03	4.0E−03	5.3E−02	7.2E−03	3.9E−03	1.4E−01
30%	7.11	6.35	5.81	7.16	8.45	8.40	8.32	8.47	8.25	8.35	8.35	8.95	8.16	8.16	7.99	8.01	4.0E−01	4.1E−01	6.5E−02	5.1E−01°	8.8E−02	2.1E−02	6.6E−03	1.5E−01	2.2E−01	4.5E−02	2.1E−02	7.3E−01
35%	9.30	8.88	8.32	9.64	10.21	10.09	9.93	10.34	9.94	10.01	10.14	10.76	9.94	9.96	9.90	9.92	6.3E−01	8.4E−01	6.0E−01	1.0E−01°	2.0E−01	2.4E−01	1.0E−01	9.6E−01°	2.2E−01	2.1E−01	1.2E−01	5.9E−01°
40%	12.08	11.83	11.52	13.17	12.51	12.60	12.57	12.75	12.14	12.29	12.40	12.67	11.96	11.99	11.93	12.20	7.1E−01°	7.9E−01	7.7E−01	3.5E−02°	5.0E−01	4.1E−01	2.7E−01	1.2E−01°	7.7E−01	5.8E−01	4.8E−01	1.6E−01°
45%	16.05	15.96	16.01	16.92	15.33	15.64	15.78	16.20	15.45	15.46	15.42	15.37	15.10	15.12	15.11	15.59	3.6E−01°	5.4E−01°	6.5E−01°	2.5E−02°	7.2E−01°	8.1E−01°	7.4E−01°	1.0E−02°	6.6E−01°	8.0E−01°	7.2E−01°	1.3E−01°
50%	24.13	24.54	24.90	22.70	21.93	22.08	22.13	22.39	20.55	20.50	20.50	19.38	20.34	20.48	20.43	20.82	5.4E−03°	7.8E−03°	5.1E−03°	2.5E−01°	4.3E−05°	2.1E−05°	5.9E−06°	3.7E−04°	2.3E−05°	4.4E−05°	1.0E−05°	4.8E−02°

Table 6
Performance and robustness results on datasets with attribute noise: comparison of decisions combination methods.

x%	Results																p-values											
	Majority				Weighted				Bayes				BKS				Weighted				Bayes				BKS			
	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5	MCS3-1	MCS3-3	MCS3-5	MCS5
Performance																												
Uniform attribute noise																												
0%	85.42	85.48	85.52	83.74	84.92	85.04	85.04	85.83	85.03	85.22	85.29	86.10	84.25	84.39	84.39	84.72	9.2E−01*	8.0E−01*	5.0E−01	3.0E−02*	9.3E−01*	9.2E−01*	9.5E−01	1.2E−02*	2.4E−02	3.4E−02	2.6E−02	7.8E−01
5%	84.57	84.84	84.77	82.96	83.58	83.78	83.80	84.69	84.25	84.46	84.38	85.13	83.56	83.39	83.29	83.87	9.8E−02	6.8E−02	2.9E−02	1.1E−01*	7.4E−01	5.7E−01	7.7E−01	2.5E−02*	1.9E−02	6.4E−03	3.4E−03	2.9E−01
10%	83.33	83.64	83.80	81.69	82.51	82.63	82.59	82.86	83.10	83.25	83.27	84.09	82.28	82.17	82.17	82.65	1.1E−01	1.8E−01	5.5E−02	3.1E−01*	8.2E−01	8.1E−01	5.2E−01	1.3E−02*	1.2E−02	3.5E−03	5.9E−03	6.2E−01
15%	81.81	82.18	82.31	80.61	80.69	80.87	80.93	81.04	81.42	81.61	81.64	82.45	80.89	80.72	80.63	81.14	5.2E−02	1.0E−01	5.6E−02	8.0E−01	2.6E−01	1.7E−01	1.5E−01	9.0E−02*	3.2E−02	5.5E−03	7.7E−03	3.5E−01
20%	80.64	81.14	81.42	79.68	78.73	79.18	79.34	79.22	79.82	79.98	80.01	81.03	79.31	79.20	79.16	79.80	5.8E−03	6.4E−03	5.1E−03	6.6E−01	1.2E−01	5.3E−02	3.9E−02	2.9E−01*	2.2E−03	1.8E−03	3.0E−03	2.9E−01
25%	79.37	79.85	80.22	78.53	77.18	77.64	77.98	77.99	78.46	78.66	78.74	79.69	78.07	78.00	78.02	78.60	2.2E−03	3.2E−03	1.8E−03	3.6E−01	8.4E−02	4.6E−02	1.8E−02	3.1E−01*	2.3E−03	9.4E−04	5.0E−04	3.2E−01
30%	77.97	78.42	78.75	77.14	75.23	75.50	75.75	76.53	76.84	76.87	77.05	78.07	76.54	76.39	76.40	76.98	4.8E−04	2.9E−04	9.2E−05	4.0E−01	3.6E−02	1.5E−02	8.7E−03	6.5E−01*	1.6E−03	4.3E−04	1.1E−04	1.7E−01
35%	76.26	76.91	77.23	75.96	73.44	73.57	73.69	74.66	74.78	74.90	74.99	76.16	74.61	74.60	74.53	75.29	9.0E−05	8.5E−05	1.2E−05	1.6E−01	4.8E−03	4.4E−03	1.5E−03	5.4E−01	3.7E−04	5.8E−04	6.2E−05	5.1E−02
40%	74.84	75.57	76.03	74.58	72.34	72.60	72.76	72.92	73.24	73.37	73.44	74.43	73.02	72.99	73.18	73.73	5.5E−03	1.1E−03	4.5E−04	1.4E−01	3.8E−02	2.0E−02	6.5E−03	6.8E−01	5.6E−03	2.3E−03	7.4E−04	3.1E−01
45%	72.93	73.70	74.27	73.03	70.26	70.64	70.78	70.90	70.99	71.10	71.20	72.05	70.87	70.81	70.92	71.63	8.3E−04	5.4E−04	1.4E−04	5.5E−02	5.2E−03	5.2E−03	1.6E−03	4.4E−01	1.3E−03	4.4E−04	8.2E−05	1.6E−01
50%	71.36	72.14	72.85	71.57	69.22	69.32	69.45	69.61	69.65	69.82	69.83	70.54	69.47	69.51	69.69	70.37	5.2E−03	1.3E−03	2.1E−04	1.1E−01	1.2E−02	5.2E−03	8.6E−04	1.8E−01	3.0E−03	5.0E−04	6.2E−05	1.4E−01
Gaussian attribute noise																												
0%	85.42	85.48	85.52	83.74	84.92	85.04	85.04	85.83	85.03	85.22	85.29	86.10	84.25	84.39	84.39	84.72	9.2E−01*	8.0E−01*	5.0E−01	3.0E−02*	9.3E−01*	9.2E−01*	9.5E−01	1.2E−02*	2.4E−02	3.4E−02	2.6E−02	7.8E−01
5%	85.12	85.27	85.29	83.45	84.26	84.26	84.28	85.29	85.02	85.03	84.99	85.70	84.15	84.06	84.01	84.37	3.4E−01	1.3E−01	1.3E−01	5.1E−02*	8.0E−01*	7.8E−01*	8.8E−01*	2.2E−02*	2.3E−02	3.4E−02	2.3E−02	6.5E−01
10%	84.49	84.62	84.55	82.89	83.47	83.47	83.40	84.01	84.30	84.33	84.34	85.12	83.57	83.39	83.23	83.76	1.1E−01	4.9E−02	7.2E−02	3.5E−01*	9.9E−01*	9.1E−01	9.1E−01	1.4E−02*	1.2E−02	1.5E−02	1.0E−02	4.8E−01
15%	84.03	84.24	84.35	82.43	83.03	83.07	83.05	83.20	83.59	83.71	83.83	84.68	83.00	82.89	82.82	83.26	1.2E−01	1.3E−01	5.4E−02	4.6E−01*	7.4E−01	6.1E−01	8.4E−01*	1.9E−02*	2.1E−02	2.0E−02	5.6E−03	3.9E−01
20%	83.33	83.52	83.65	81.83	82.04	82.13	82.16	81.86	82.83	82.93	82.98	83.94	82.23	82.13	82.10	82.45	5.0E−02	4.8E−02	1.1E−02	8.7E−01	5.1E−01	4.5E−01	4.3E−01	2.6E−02*	1.0E−02	3.0E−03	2.4E−03	2.9E−01
25%	82.62	82.92	83.16	81.41	81.07	81.28	81.53	81.32	82.18	82.44	82.54	83.40	81.78	81.77	81.72	82.20	2.2E−02	5.1E−03	4.7E−03	4.6E−01	3.2E−01	2.3E−01	2.0E−01	8.9E−02*	1.5E−02	3.7E−03	2.1E−03	3.7E−01
30%	81.85	82.15	82.37	80.56	79.84	80.03	80.27	80.59	81.44	81.57	81.70	82.58	80.96	80.88	80.89	81.39	3.3E−02	8.6E−03	6.5E−03	4.7E−01	5.0E−01	2.7E−01	3.2E−01	1.1E−01*	2.6E−02	1.5E−02	1.6E−02	4.4E−01
35%	81.03	81.32	81.54	79.75	79.12	79.28	79.58	79.84	80.58	80.80	80.84	81.61	80.22	80.21	80.22	80.62	3.5E−02	1.9E−02	1.9E−02	5.8E−01	2.8E−01	6.3E−01	4.9E−01	1.7E−01*	3.4E−02	2.6E−02	3.5E−02	7.2E−01
40%	80.34	80.87	81.04	79.32	78.37	78.58	78.70	79.00	79.64	79.97	80.01	80.83	79.27	79.28	79.19	79.88	2.9E−02	2.1E−02	1.0E−02	7.5E−01	3.2E−01	1.8E−01	2.0E−01	4.8E−01*	3.7E−02	6.1E−03	6.6E−03	5.3E−01
45%	79.45	79.71	79.98	78.25	77.66	77.88	78.04	78.09	78.92	79.12	79.10	79.89	78.55	78.58	78.46	78.92	1.3E−01	7.0E−02	2.7E−02	8.9E−01	3.5E−01	3.9E−01	2.5E−01	1.9E−01*	4.0E−02	3.0E−02	6.9E−03	6.9E−01
50%	78.49	79.08	79.21	77.86	76.73	77.07	77.23	77.37	77.97	78.12	78.23	79.03	77.59	77.64	77.60	78.15	5.9E−02	1.9E−02	2.7E−02	9.1E−01	2.0E−01	6.1E−02	9.7E−02	6.6E−01*	2.2E−02	4.0E−03	3.9E−03	5.6E−01
Robustness																												
Uniform attribute noise																												
5%	0.95	0.71	0.85	0.94	1.44	1.47	1.47	1.39	0.84	0.85	1.07	1.12	0.69	1.10	1.21	0.89	1.1E−01	3.2E−02	2.4E−01	3.2E−01	7.6E−01	3.7E−01	9.5E−01	5.9E−01*	6.5E−01	1.7E−01	3.2E−01	4.9E−01
10%	2.40	2.11	1.98	2.57	2.64	2.78	2.86	3.69	2.16	2.28	2.36	2.31	2.19	2.52	2.48	2.35	4.1E−01	4.0E−01	2.6E−01	2.0E−01	9.5E−01*	7.6E−01	5.3E−01	1.6E−01*	9.6E−01	2.3E−01	3.1E−01	8.0E−01
15%	4.23	3.86	3.76	3.63	4.82	4.90	4.86	5.88	4.17	4.23	4.31	4.30	3.89	4.25	4.36	4.16	1.4E−01	2.1E−01	2.1E−01	4.2E−02	6.6E−01	4.4E−01	3.0E−01	5.0E−01	5.5E−01*	3.3E−01	3.1E−01	4.0E−01
20%	5.54	5.02	4.80	4.68	7.17	6.87	6.70	8.03	6.03	6.10	6.19	5.90	5.68	5.99	6.06	5.65	4.2E−02	2.3E−02	2.9E−02	1.2E−02	3.1E−01	1.3E−01	8.6E−02	4.6E−01	2.8E−01	8.5E−02	8.5E−02	3.1E−01
25%	7.09	6.61	6.23	6.10	8.99	8.69	8.28	9.43	7.58	7.64	7.63	7.44	7.16	7.42	7.37	7.08	9.3E−02	4.9E−02	2.8E−02	1.5E−02	2.6E−01	1.1E−01	3.8E−02	3.5E−01	7.6E−01	1.8E−01	1.1E−01	5.2E−01
30%	8.85	8.36	8.06	7.84	11.52	11.42	11.12	11.19	9.58	9.81	9.69	9.38	9.06	9.42	9.41	9.10	1.7E−02	8.0E−03	2.6E−03	2.0E−02	2.8E−01	7.4E−02	2.7E−02	3.7E−01	6.6E−01	1.7E−01	4.4E−02	4.6E−01
35%	10.79	10.12	9.82	9.17	13.50	13.59	13.45	13.24	11.89	11.98	11.98	11.50	11.26	11.41	11.50	11.00	6.8E−03	1.3E−03	5.4E−04	8.4E−03	1.3E−01	4.0E−02	8.7E−03	5.3E−02	3.8E−01	1.4E−01	4.2E−02	2.6E−01
40%	12.44	11.67	11.23	10.74	14.73	14.63	14.44	15.14	13.59	13.68	13.68	13.40	13.07	13.26	13.04	12.78	2.3E−01	6.0E−02	3.7E−02	2.0E−02	4.0E−01	2.0E−01	1.0E−01	1.7E−01	6.5E−01	4.7E−01	3.0E−01	8.8E−01
45%	14.73	13.98	13.38	12.75	17.30	17.05	16.88	17.53	16.31	16.42	16.37	16.23	15.71	15.91	15.79	15.31	4.3E−02	2.0E−02	1.0E−02	6.4E−03	1.7E−01	6.0E−02	3.4E−02	7.2E−02	4.0E−01	2.4E−01	9.8E−02	4.4E−01
50%	16.68	15.85	15.09	14.40	18.56	18.63	18.45	19.04	17.97	17.97	18.06	18.03	17.45	17.55	17.33	16.88	1.7E−01	3.2E−02	1.0E−02	1.2E−02	2.3E−01	8.1E−02	7.8E−03	3.0E−02	4.9E−01	1.8E−01	3.4E−02	3.9E−01
Gaussian attribute noise																												
5%	0.31	0.19	0.25	0.32	0.64	0.90	0.90	0.69	−0.04	0.22	0.38	0.47	0.02	0.33	0.38	0.31	1.9E−01	6.5E−02	3.0E−02	1.6E−01	5.8E−01	4.6E−01	7.0E−01	6.9E−01*	9.0E−01	2.1E−01	4.4E−01	7.7E−01
10%	1.03	0.98	1.16	0.92	1.63	1.86	1.97	2.34	0.81	1.09	1.17	1.16	0.69	1.12	1.32	1.05	8.6E−02	8.2E−02	3.5E−01	4.5E−02	5.8E−01	6.0E−01	8.9E−01*	6.0E−01*	9.8E−01	3.9E−01	4.6E−01	5.1E−01
15%	1.54	1.39	1.32	1.48	2.05	2.26	2.31	3.31	1.59	1.76	1.70	1.62	1.33	1.62	1.73	1.58	1.3E−01	7.5E−02	1.3E−01	3.2E−02	6.3E−01	3.4E−01	3.5E−01	8.2E−01*	7.7E−01*	5.6E−01	3.5E−01	6.2E−01
20%	2.36	2.21	2.14	2.07	3.29	3.41	3.41	4.93	2.54	2.69	2.75	2.48	2.33	2.58	2.65	2.58	1.8E−02	2.0E−02	2.9E−02	5.7E−03	3.0E−01	1.2E−01	1.8E−01	8.0E−01*	4.4E−01	2.2E−01	1.4E−01	4.3E−01
25%	3.20	2.95	2.73	2.56	4.53	4.51	4.20	5.58	3.31	3.27	3.24	3.14	2.84	3.00	3.09	2.83	4.4E−02	1.1E−02	1.5E−02	1.8E−03	5.6E−01	4.0E−01	2.3E−01	4.8E−01	6.2E−01*	6.0E−01	2.5E−01	4.4E−01
30%	4.14	3.86	3.65	3.76	6.07	6.05	5.76	6.38	4.17	4.28	4.23	4.04	3.82	4.07	4.06	3.81	1.7E−01	3.6E−02	2.7E−02	7.8E−03	7.2E−01	4.2E−01	3.3E−01	3.9E−01	5.0E−01*	6.8E−01	4.9E−01	7.6E−01
35%	5.09	4.82	4.63	4.62	7.00	7.02	6.61	7.25	5.25	5.23	5.28	5.20	4.77	4.94	4.92	4.76	7.4E−02	3.7E−02	6.7E−02	2.3E−02	8.0E−01	4.9E−01	4.1E−01	5.7E−01	4.6E−01*	8.3E−01*	8.9E−01	9.6E−01*
40%	5.93	5.38	5.24	5.11	7.95	7.90	7.74	8.27	6.44	6.28	6.32	6.18	5.97	6.13	6.22	5.69	7.4E−02	5.0E−02	2.1E−02	3.4E−02	5.0E−01	2.1E−01	1.8E−01	2.5E−01	7.1E−01*	6.2E−01	2.5E−01	7.5E−01
45%	7.03	6.80	6.53	6.38	8.82	8.73	8.51	9.32	7.31	7.29	7.41	7.28	6.90	6.98	7.15	6.91	2.1E−01	1.3E−01	1.8E−01	5.5E−02	8.9E−01	4.9E−01	3.2E−01	4.4E−01	8.3E−01*	9.3E−01	4.8E−01	6.0E−01
50%	8.14	7.50	7.43	6.68	9.94	9.68	9.47	10.10	8.43	8.47	8.42	8.23	8.04	8.09	8.17	7.78	9.2E−02	4.6E−02	5.2E−02	5.8E−02	6.1E−01	1.9E−01	2.2E−01	1.6E−01	7.7E−01*	5.8E−01	4.8E−01	2.9E−01

- The average results show that, in MCS3-*k*, MAJ and NB are the best methods, whereas BKS and W-MAJ are the worst ones. In MCS5, NB is the best method; BKS and MAJ are the following methods and W-MAJ is placed at the last positions at many noise levels.
- The *p*-values show that MAJ is better than W-MAJ from 20% onwards. It is better than NB for MCS3-3 and MCS3-5 from a noise level 20–25% and for MCS5 at the highest noise levels (45–50%), but this does not occur with MCS3-1. MAJ is neither better than BKS (only at some intermediate-high noise levels).

(b) **Gaussian attribute noise.**
- In MCS3-*k*, MAJ and NB are the best methods in average results, followed by BKS and W-MAJ. In MCS5, the ranking is composed by NB, and BKS, MAJ and W-MAJ (these last three obtain the lowest average results at most of the noise levels).
- The *p*-values show that MAJ is not better than NB nor BKS. It is generally better than W-MAJ, even though they are equivalent at some isolated noise levels.

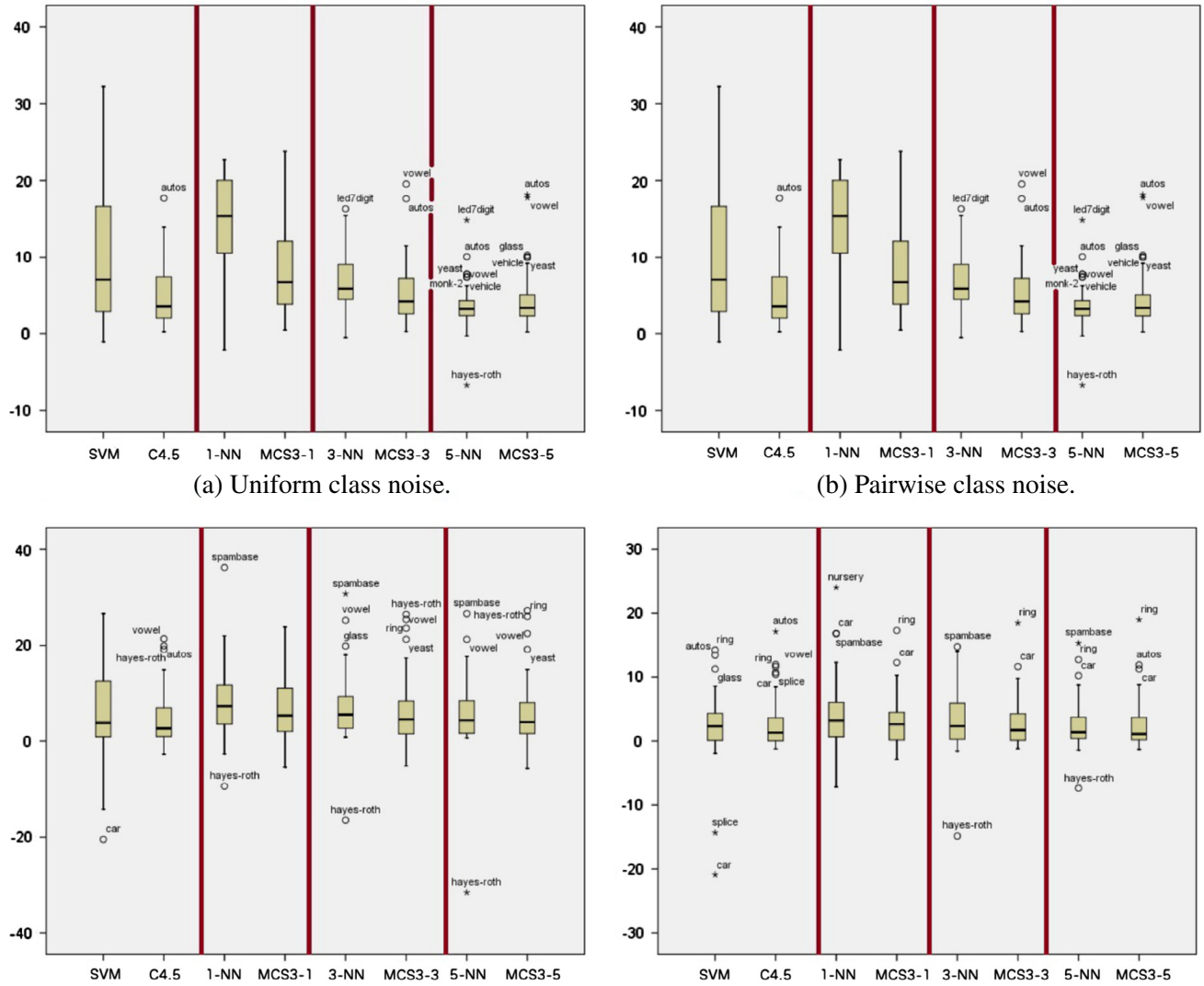
2. **Robustness results.**
(a) **Uniform attribute noise.**
- The average results show that W-MAJ is the best method, followed by NB, BKS and MAJ.
- Attending to the *p*-values, we observe that, in general, MAJ is better than W-MAJ (from 20% onwards), even though they are equivalent at some high noise levels considering MCS3-1. MAJ is also better than NB for medium–high noise levels (except for MCS3-1, where both are equivalent). BKS is equivalent to MAJ, except for MCS3-5 and MCS3-5 where MAJ is better than BKS from 20% onwards.

(b) **Gaussian attribute noise.**
- W-MAJ obtains the best average results. It is followed by NB, and the worst combination methods are BKS and MAJ.
- The *p*-values show that MAJ is usually better than W-MAJ (except at some isolated noise level where both are equivalent). In general, MAJ is equivalent to NB and BKS, even though NB and BKS are better than MAJ for some MCSs (NB is better for MCS5 and BKS is better for MCS3-1).

7. On the usage of Multiple Classifier Systems with noisy data: Lessons learned

In this paper a thorough empirical study has been performed in order to find the conditions under which the MCSs studied (the three MCS-*k*, the MCS5 and BagC4.5) behave well with noisy data, as well as the terms of this improvement. From the results shown in the previous section and their corresponding analysis, several lessons can be learned:

1. **The behavior of the studied MCSs built with heterogeneous classifiers depend on the behavior of their individual classifiers with noisy data.** This situation can be clearly observed in the obtained results where, for example, a higher value of *k*-and therefore a better performance and robustness of *k*-NN-implies that the corresponding MCS3-*k* attains a better performance and robustness. This higher robustness is particularly notable with the most disruptive noise schemes (uniform class and attribute noise) and the highest noise levels. However, BagC4.5 gain robustness against several noise types (such as the uniform class or attribute noises) which C4.5 is weaker with.
2. **The study of behavior of each single classifier considered to build an MCS in noisy environments is an important issue.** For example, SVM obtains the highest performance without noise but is the most affected when class noise is considered. Hence, it is recommended to execute all the available classifiers to check which are more suitable for the type and level of noise, finally choosing those performing better in order to build the MCS.
3. **Different types of noise have a very different disruptiveness and, therefore, they do not affect the performance in the same way.** For the classifiers studied, the uniform class noise is generally more disruptive than the pairwise class noise-except in the case of BagC4.5 and for the medium–highest noise levels with those classifiers involving *k*-NN with $k = 3,5$. We must not forget that uniform class noise can be more disruptive because it affects more examples than the pairwise class noise. The uniform attribute noise is more disruptive than the Gaussian attribute noise. Moreover, the uniform class noise is more disruptive than the uniform attribute noise.
4. **On the behavior of the studied MCSs with data with class noise.** The performance of the studied MCSs compared to that of their individual classifiers is better with the uniform class noise (the more disruptive scheme) than with the pairwise noise, since the differences between MCSs and their base classifiers are more accentuated with the uniform noise. The robustness results show a similar conclusion, with the MCSs generally more robust with the uniform class noise and obtaining in this case lower *p*-values in the comparisons with respect to their individual classifiers.
5. **On the behavior of the studied MCSs with data with attribute noise.** The studied MCSs perform better than their individual classifiers with the less disruptive attribute noise scheme (Gaussian) and slightly worse with the more disruptive one (uniform). Similar conclusions are drawn from the robustness results. The exception is BagC4.5, which perform clearly better than C4.5 with both schemes.



(a) Uniform class noise.

(b) Pairwise class noise.

(c) Uniform attribute noise.

(d) Gaussian attribute noise.

Fig. 1. Box-plots representing the distribution of the RLA results with a 25% noise level for MCS3-k and its individual classifiers. The RLA metric (Eq. (2)) is used to estimate the robustness of a classifier at a noise level comparing the accuracy of the classifier at that noise level with respect to that of the classifier trained without additional noise.

6. **Different types of noise affect the studied MCSs differently.** The studied MCSs built with heterogeneous classifiers work worse with attribute noise than with class noise. With attribute noise, MCSs significantly improve the results of the more robust single method, that is, C4.5, if the noise is not very disruptive (Gaussian) or if the noise level is low enough. However, it is important to note that BagC4.5 works well with both types of noise except with the pairwise class noise, in which it does not perform as accurate as with other types of noise.
7. **Even though the studied MCSs built with heterogeneous classifiers may lead to an improvement in the performance results working with noisy data, they will never be statistically more robust than the most robust of its individual classifiers.** The improvement of performance will depend on many factors, as mentioned above, but it is likely to occur if the proper base classifiers are selected. However, an improvement in the robustness of the methods seems to be difficult. In fact, the robustness of the MCS built can be thought of as the average of the robustness of each of its single classifiers. Fig. 1 shows the distribution of the robustness results of each single classification algorithm and each MCS3-*k* for each noise scheme at the medium noise level considered in this paper; that is, 25%. As it can be appreciated in this figure, the distribution of the robustness values of each MCS3-*k* is always between the distribution of the least and most robust single classifiers–even though this fact is more observable in the robustness results with class noise. However, from the results obtained we observe that using BagC4.5 the robustness of its only single classifier C4.5 can be improved.
8. **The majority vote scheme is a powerful alternative against other decision combination methods for the MCS studied when working with noisy data.** Even though MAJ is outperformed with some types of noise and at some noise levels, it performs relatively well for most of the noise types. Since the type and level of noise is usually unknown in real-world data, its usage can be recommended.

The main reason for the better performance of MCSs with noisy data can be attributed to the increment of the capability of generalization when an MCS is built. Each single classifier is known to make errors, but since they are different, that is, they have different behaviors over different parts of the domain, misclassified examples are not necessarily the same [21], and the same is true for noisy instances. This fact enables MCSs to achieve a better generalization from the examples of a problem and leads to better avoiding the overfitting of noisy data and, therefore, to obtain more accurate solutions. Some classifiers can tolerate the noise better or are more sensitive than others, depending on the different parts of the input/output space. Combining several classifiers, preferably noise-tolerant in most of the input/output space, may lead them to complement each other, obtaining a higher noise-robustness, or at least better generalization capabilities. Thus, if a set of noisy examples does not affect the learning of a concrete set of classifiers, their predictions will not be hindered.

8. Concluding remarks

This paper analyzes how well several MCSs behave with noisy data by means the realization of a huge experimental study. In order to do this, both the performance and the noise-robustness of different MCSs have been compared with respect to their single classification algorithms. A large number of noisy datasets have been created considering different types, schemes and levels of noise to perform these comparisons.

The results obtained have shown that the MCSs studied do not always significantly improve the performance of their single classification algorithms when dealing with noisy data, although they do in the majority of cases (if the individual components are not heavily affected by noise). The improvement depends on many factors, such as the type and level of noise. Moreover, the performance of the MCSs built with heterogeneous classifiers depends on the performance of their single classifiers, so it is recommended that one studies the behavior of each single classifier before building the MCS. Generally, the MCSs studied are more suitable for class noise than for attribute noise. Particularly, they perform better with the most disruptive class noise scheme (uniform one) and with the least disruptive attribute noise scheme (Gaussian one). However, BagC4.5 works well with both types of noise, with the exception of the pairwise class noise, where its results are not as accurate as with other types of noise.

The robustness results show that the studied MCS built with heterogeneous classifiers will not be more robust than the most robust among their single classification algorithms. In fact, the robustness can always be shown as an average of the robustnesses of the individual methods. The higher the robustnesses of the individual classifiers are, the higher the robustness of the MCS is. Nevertheless, BagC4.5 is again an exception: it becomes more robust than its single classifier C4.5.

The study of several decisions combination methods show that the majority vote scheme is a simple yet powerful alternative to other techniques when working with noisy data.

Acknowledgments

Supported by the Spanish Ministry of Science and Technology under Projects TIN2011-28488 and TIN2010-15055, and also by the Regional Project P10-TIC-6858. José A. Sáez holds an FPU scholarship from the Spanish Ministry of Education and Science.

References

[1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2011) 255–287.
[2] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing-A Fusion of Foundations, Methodologies and Applications 13 (2009) 307–318.
[3] C. Catal, O. Alan, K. Balkan, Class noise detection based on software metrics and ROC curves, Information Sciences 181 (2011) 4867–4877.
[4] W.W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann Publishers, 1995, pp. 115–123.
[5] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.
[6] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[7] T. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Machine Learning 40 (2000) 139–157.
[8] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., John Wiley, New York, 2001.
[9] J. Fűrnkranz, Noise-tolerant windowing, Journal of Artificial Intelligence Research 8 (1997) 129–164.
[10] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Information Sciences 180 (2010) 2044–2064.
[11] S. García, F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.
[12] R.M. Haralick, The table look-up rule, Communications in Statistics-Theory and Methods A5 (1976) 1163–1191.
[13] M.A. Hernández, S.J. Stolfo, Real-world data is dirty: data cleansing and the merge/purge problem, Data Mining and Knowledge Discovery 2 (1998) 9–37.
[14] T.K. Ho, Multiple classifier combination: lessons and next steps, in: A. Kandel, E. Bunke (Eds.), Hybrid Methods in Pattern Recognition, World Scientific, 2002, pp. 171–198.
[15] T.K. Ho, J.J. Hull, S.N. Srihari, Decision combination in multiple classifier systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (1994) 66–75.
[16] J.L. Hodges, E.L. Lehmann, Ranks methods for combination of independent experiments in analysis of variance, Annals of Mathematical Statistics 33 (1962) 482–497.

[17] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.
[18] Y.S. Huang, C.Y. Suen, A method of combining multiple experts for the recognition of unconstrained handwritten numerals, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (1995) 90–93.
[19] P.J. Huber, Robust Statistics, John Wiley and Sons, New York, 1981.
[20] T. Khoshgoftaar, J. Van Hulse, A. Napolitano, Comparing boosting and bagging techniques with noisy and imbalanced data, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 41 (2011) 552–568.
[21] J. Kittler, M. Hatef, R. Duin, J. Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 226–239.
[22] I. Kononenko, M. Kukar, Machine Learning and Data Mining: Introduction to Principles and Algorithms, Horwood Publishing Limited, 2007.
[23] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, 2004.
[24] L.I. Kuncheva, Diversity in multiple classifier systems, Information Fusion 6 (2005) 3–4.
[25] R. Maclin, D. Opitz, An empirical evaluation of bagging and boosting, in: Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence, pp. 546–551.
[26] E. Mandler, J. Schuermann, Combining the classification results of independent classifiers based on the Dempster/Shafer theory of evidence, in: E.S. Gelsema, L.N. Kanal (Eds.), Pattern Recognition and Artificial Intelligence, North-Holland, Amsterdam, 1988, pp. 381–393.
[27] U. Maulik, D. Chakraborty, A robust multiple classifier system for pixel classification of remote sensing images, Fundamenta Informaticae 101 (2010) 286–304.
[28] V.D. Mazurov, A.I. Krivonogov, V.S. Kazantsev, Solving of optimization and identification problems by the committee methods, Pattern Recognition 20 (1987) 371–378.
[29] G.J. McLachlan, Discriminant Analysis and Statistical Pattern Recognition, John Wiley and Sons, 2004.
[30] R.K. Nath, Fingerprint recognition using multiple classifier system, Fractals 15 (2007) 273–278.
[31] D. Nettleton, A. Orriols-Puig, A. Fornells, A study of the effect of different types of noise on the precision of supervised learning techniques, Artificial Intelligence Review 33 (2010) 275–306.
[32] R. Polikar, Ensemble based systems in decision making, IEEE Circuits and Systems Magazine 6 (2006) 21–45.
[33] B. Qian, K. Rasheed, Foreign exchange market prediction with multiple classifiers, Journal of Forecasting 29 (2010) 271–284.
[34] J.R. Quinlan, Induction of decision trees, in: Machine Learning, pp. 81–106.
[35] J.R. Quinlan, The effect of noise on concept learning, in: Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann Publishers, 1986, pp. 149–166.
[36] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.
[37] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, A. Folleco, An empirical study of the classification performance of learners on imbalanced and noisy software quality data, Information Sciences (2011), http://dx.doi.org/10.1016/j.ins.2010.12.016.
[38] L. Shapley, B. Grofman, Optimizing group judgmental accuracy in the presence of interdependencies, Public Choice 43 (1984) 329–343.
[39] C.M. Teng, Correcting noisy data, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999, pp. 239–248.
[40] C.M. Teng, Polishing blemishes: issues in data correction, IEEE Intelligent Systems 19 (2004) 34–39.
[41] D.M. Titterington, G.D. Murray, L.S. Murray, D.J. Spiegelhalter, A.M. Skene, J.D.F. Habbema, G.J. Gelpke, Comparison of discriminant techniques applied to a complex data set of head injured patients, Journal of the Royal Statistical Society, Series A (General) 144 (1981) 145–175.
[42] R.Y. Wang, V.C. Storey, C.P. Firth, A framework for analysis of data quality research, IEEE Transactions on Knowledge and Data Engineering 7 (1995) 623–640.
[43] K.D. Wemecke, A coupling procedure for the discrimination of mixed data, Biometrics 48 (1992) 497–506.
[44] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, Journal of Artificial Intelligence Research 6 (1997) 1–34.
[45] M. Woźniak, M. Graña, E. Corchado, A survey of multiple classifier systems as hybrid systems, Information Fusion (2013). http://dx.doi.org/10.1016/j.inffus.2013.04.006.
[46] X. Wu, Knowledge Acquisition from Databases, Ablex Publishing Corp., Norwood, NJ, USA, 1996.
[47] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems 14 (2007) 1–37.
[48] X. Wu, X. Zhu, Mining with noise knowledge: error-aware data mining, IEEE Transactions on Systems, Man, and Cybernetics 38 (2008) 917–932.
[49] L. Xu, A. Krzyzak, C.Y. Suen, Methods of combining multiple classifiers and their applications to handwriting recognition, IEEE Transactions on Systems, Man, and Cybernetics 22 (1992) 418–435.
[50] Y. Zhang, X. Wu, Integrating induction and deduction for noisy data mining, Information Sciences 180 (2010) 2663–2673.
[51] S. Zhong, T.M. Khoshgoftaar, N. Seliya, Analyzing software measurement data with clustering techniques, IEEE Intelligent Systems 19 (2004) 20–27.
[52] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study, Artificial Intelligence Review 22 (2004) 177–210.
[53] X. Zhu, X. Wu, Q. Chen, Eliminating class noise in large datasets, in: Proceeding of the Twentieth International Conference on Machine Learning, pp. 920–927.
[54] X. Zhu, X. Wu, Y. Yang, Error detection and impact-sensitive instance ranking in noisy datasets, in: Proceedings of the Nineteenth National Conference on Artificial Intelligence, AAAI Press, 2004, pp. 378–383.

1.2 Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition

Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems* 38(1): 179–206

- Status: **Published**.
- Impact Factor (JCR 2014): Not available.
- Current Impact Factor of the Journal (JCR 2013): 2.639
- Subject Category: Computer Science, Artificial Intelligence. Ranking 21 / 121 (**Q1**).
- Subject Category: Computer Science, Information Systems. Ranking 15 / 135 (**Q1**).

Knowl Inf Syst (2014) 38:179–206
DOI 10.1007/s10115-012-0570-1

REGULAR PAPER

Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition

José A. Sáez · Mikel Galar · Julián Luengo · Francisco Herrera

Received: 7 February 2012 / Revised: 12 August 2012 / Accepted: 6 October 2012 /
Published online: 6 November 2012

Abstract The presence of noise in data is a common problem that produces several negative consequences in classification problems. In multi-class problems, these consequences are aggravated in terms of accuracy, building time, and complexity of the classifiers. In these cases, an interesting approach to reduce the effect of noise is to decompose the problem into several binary subproblems, reducing the complexity and, consequently, dividing the effects caused by noise into each of these subproblems. This paper analyzes the usage of decomposition strategies, and more specifically the One-vs-One scheme, to deal with noisy multi-class datasets. In order to investigate whether the decomposition is able to reduce the effect of noise or not, a large number of datasets are created introducing different levels and types of noise, as suggested in the literature. Several well-known classification algorithms, with or without decomposition, are trained on them in order to check when decomposition is advantageous. The results obtained show that methods using the One-vs-One strategy lead to better performances and more robust classifiers when dealing with noisy data, especially with the most disruptive noise schemes.

Keywords Noisy data · Class noise · Attribute noise · One-vs-One · Decomposition strategies · Ensembles · Classification

J. A. Sáez ( ) · F. Herrera
Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, 18071 Granada, Spain
e-mail: smja@decsai.ugr.es

F. Herrera
e-mail: herrera@decsai.ugr.es

M. Galar
Department of Automática y Computación, Universidad Pública de Navarra, 31006 Pamplona, Spain
e-mail: mikel.galar@unavarra.es

J. Luengo
Department of Civil Engineering, LSI, University of Burgos, 09006 Burgos, Spain
e-mail: jluengo@ubu.es

1 Introduction

Any classification problem [14,49] consists of m training examples, characterized by n attributes A_i, $i = 1, \ldots, n$ that are either numerical or categorical, with $\mathbb{D}_i$ their corresponding domains. Thus, an example $\mathbf{x}$ is represented as an n-dimensional attribute vector

$$\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{D} = \mathbb{D}_1 \times \cdots \times \mathbb{D}_n.$$

Each of these examples is labeled with one out of the M possible classes $\mathbb{L} = \{\lambda_1, \ldots, \lambda_M\}$. Many current real-world classification problems, such as the classification of cancer tissues [5] or the recognition of business documents [36], must distinguish between more than two classes, that is, $M > 2$. These problems are formally known as multi-class classification problems.

Classification algorithms aim to extract the implicit knowledge from previously known labeled examples of the problem creating a model, called a classifier, which should be capable of predicting the class for new unobserved examples. For this reason, the classification accuracy of a classifier is directly influenced by the quality of the training data used. Data quality depends on several components [50], for example, the source and the input procedure, inherently subject to errors. Real-world datasets usually contain corrupted data that may hinder the interpretations, decisions, and therefore, the classifiers built from that data.

Usually, the more classes in a problem, the more complex it is. In multi-class learning, the generated classifier must be able to separate the data into more than a pair of classes, which increases the chances of incorrect classifications (in a two-class balanced problem, the probability of a correct random classification is $1/2$, whereas in a multi-class problem it is $1/M$). Furthermore, in problems affected by noise, the boundaries, the separability of the classes, and therefore, the prediction capabilities of the classifiers may be severely hindered.

Given the loss of accuracy produced by noise, the need of techniques to deal with it has been proved in previous works [9,23,56]. In the specialized literature, two ways have been proposed in order to mitigate the effects produced by noise:

1. Adaptation of the algorithms to properly handle the noise [11,42]. These methods are known as *robust learners* and they are characterized by being less influenced by noisy data.
2. Preprocessing of the datasets aiming to remove or correct the noisy examples [8,18].

However, even though both techniques can provide good results, drawbacks exist. The former depends on the classification algorithm, and therefore, the same result is not directly extensible to other learning algorithms, since the benefit comes from the adaptation itself. Moreover, this approach requires to change an existing method, which neither is always possible nor easy to develop. However, the latter requires the usage of a previous preprocessing step, which is usually time-consuming. Furthermore, these methods are only designed to detect an specific type of noise and hence, the resulting data might not be perfect [53]. For these reasons, it is important to investigate other mechanisms, which could lead to decrease the effects caused by noise without neither needing to adapt each specific algorithm nor having to make assumptions about the type and level of noise present in the data.

When dealing with multi-class problems, several works [6,27] have demonstrated that decomposing the original problem into several binary subproblems is an easy, yet accurate way to reduce their complexity. These techniques are referred to as binary decomposition strategies [32]. The most studied schemes in the literature are: *One-vs-One* (OVO) [27], which trains a classifier to distinguish between each pair of classes, and *One-vs-All* (OVA) [6], which trains a classifier to distinguish each class from all other classes. Both strategies

can be encoded within the error-correcting output codes framework [4,13]. However, none of these works provide any theoretical nor empirical results supporting the common assumption that supposes a better behavior against noise of decomposition techniques (than not using decomposition). Neither they show what type of noise is better handled by decomposition techniques.

On this account, this paper analyzes the usage of the OVO strategy, which generally outstands over OVA [15,16,24,43,45], and checks its suitability with noisy training data. It should be mentioned that, in real situations, the existence of noise in the datasets is usually unknown—therefore, neither the type nor the quantity of noise in the dataset can be known or supposed a priori. Hence, tools which are able to manage the presence of noise in the datasets, despite its type or quantity (or unexistence), are of great interest. If the OVO strategy (which is a simple yet effective methology when clean[1] datasets are considered) is also able to properly (better than the baseline non-OVO version) handle the noise, its usage could be recommended in spite of the presence of noise and without taking into account its type. Furthermore, this strategy can be used with any of the existing classifiers which are able to deal with two-class problems. Therefore, the problems of algorithm level modifications and preprocessing techniques could be avoided, and if desired, they could also be combined.

In order to carry out the analysis, a thorough empirical study will be developed considering several well-known learning algorithms having a very different behavior with noisy data: two rule-based systems, which are considered robust to noise—C4.5 [42] and *Repeated Incremental Pruning to Produce Error Reduction* (RIPPER) [11]—and an instance-based learning method, which is considered very noise-sensitive—*k-Nearest Neighbors* (*k*-NN) [35]—will be studied. Even though the theoretical robustness of these methods has been previously studied [29,40,41], there is a lack of empirical studies analyzing the real behavior of such methods when dealing with noisy data, particularly if the OVO decomposition is used. Considering two different noise categories, class and attribute noise, 800 datasets will be created [56]. Several noise schemes presented in the literature will be used to introduce these types of noise [46,56–58] and a large number of noise levels—from 5 to 50 %, by increments of 5 %—will be also studied. The differences between the OVO and non-OVO (baseline) classifiers will be analyzed through an analysis of both the accuracy and the robustness achieved on these datasets. The results obtained will be contrasted using the proper statistical tests, as recommended in the specialized literature [12].

The experimental framework stated will allow us to extract a series of conclusions on the effect of noise in multi-class problems and the usage of OVO in this scenario. We will concrete the types of noise—class (random/pair) or attribute noise(random/Gaussian)—that are more detrimental for the classifier performance and those where OVO provides a higher advantage. We will also determine in which extent OVO helps robust and noise-sensitive learners to deal with noisy data and the reasons of the increase of the robustness of such methods when using OVO. All these conclusions will be presented in Sect. 7.

A web page with all the complementary material associated with this paper is available at http://sci2s.ugr.es/ovo_noise, including the basic information of this paper, all the datasets created, and the complete results obtained for each classification algorithm.

The rest of this paper is organized as follows. Section 2 presents an introduction to classification with noisy data. Section 3 is devoted to the motivations for the usage of binary decomposition strategies in multi-class classification problems and recalls the OVO decomposition scheme. Next, Sect. 4 describes the experimental framework. Section 5 includes the analysis

[1] We refer to clean and noise-free datasets to the original datasets without additional induced noise, despite they might have noise, but it is not quantifiable, and hence it cannot be used to evaluate the robustness of the methods against noise.

of the results obtained by the classifiers on data with class noise, whereas Sect. 6 focuses on attribute noise. Section 7 provides a summary including the conclusions that can be extracted from the analysis of the results. Finally, Sect. 8 presents the concluding remarks.

2 Classification with noisy data

Real-world data are never perfect and often suffers from corruptions that may hinder the analysis of the data and their interpretations, that is, the models extracted and hence the decisions made on their basis. In classification, noise can negatively affect the system performance in terms of classification accuracy, building time, size, and interpretability of the classifier [55,56]. The presence of noise in the data may affect the intrinsic characteristics of a classification problem, since these corruptions could introduce new properties in the problem domain. For example, noise can lead to the creation of small clusters of examples of a particular class in areas of the domain corresponding to another class, or it can cause the disappearance of examples located in key areas within a specific class. The boundaries of the classes and the overlapping between them are also factors that can be affected as a consequence of noise. All these alterations difficult the knowledge extraction from the data and spoil the models obtained using that noisy data when they are compared to the models learned from clean data, which represent the real implicit knowledge of the problem [56].

The quality of a dataset is determined by a large number of components [50]. Among them, the class labels and the attribute values are two sources influencing the quality of a classification dataset. The quality of the class labels refers to whether the class of each example is correctly assigned; the quality of the attributes refers to the capability to characterize the examples for classification purposes. Two types of noise in a given dataset can be distinguished based on these two information sources [52]:

1. **Class noise** (or labeling errors). It occurs when an example is incorrectly labeled. Class noise can be attributed to several causes, such as subjectivity during the labeling process, data entry errors, or inadequacy of the information used to label each example. Two types of class noise can be distinguished:
   - *Contradictory examples.* There are duplicate examples in the dataset having different class labels [21].
   - *Misclassifications.* Examples are labeled with other class label different from the real one [57].
2. **Attribute noise**. It refers to corruptions in the values of one or more attributes. Examples of attribute noise are: erroneous attribute values, missing or unknown attribute values, and incomplete attributes or "do not care" values.

In this paper, class noise refers to misclassifications, whereas attribute noise refers to the erroneous attribute values, because they are the most common in real-world data [56]. Furthermore, erroneous attribute values, unlike other types of attribute noise, such as missing values [33] (which are easily detectable), have been less studied in the literature.

Treating class and attribute noise as corruptions of the class labels and attribute values, respectively, has been also considered in other works in the literature [37,56]; for instance, in [56], the authors reached a series of interesting conclusions, showing that attribute noise is more harmful than class noise or that eliminating or correcting examples in datasets with class and attribute noise, respectively, may improve classifier performance. They also showed that attribute noise is more harmful in those attributes highly correlated with the class labels.

In [37], the authors checked the robustness of methods from different paradigms, such as probabilistic classifiers, decision trees, instance-based learners or support vector machines, studying the possible causes of their behaviors.

However, most of the works found in the literature are only focused on class noise. In [7], the problem of multi-class classification in the presence of labeling errors was studied. The authors proposed a generative multi-class classifier to learn with labeling errors, which extends the multi-class quadratic normal discriminant analysis by a model of the mislabeling process. They demonstrated the benefits of this approach in terms of parameter recovery as well as improved classification performance. In [22], the problems caused by labeling errors occurring far from the decision boundaries in multi-class Gaussian process classifiers were studied. The authors proposed a robust multi-class Gaussian process classifier, introducing binary latent variables that indicate when an example is mislabeled. Similarly, the effect of mislabeled samples appearing in gene expression profiles was studied in [54]. A detection method for these samples was proposed, which take advantage of the measuring effect of data perturbations based on the support vector machine regression model. They also proposed three algorithms based on this index to detect mislabeled samples. An important common characteristic of these works, also considered in this paper, is that the suitability of the proposals was evaluated on both real-world and synthetic or noisy-modified real-world datasets, where the noise can be somehow quantified.

In order to model class and attribute noise, we consider four different synthetic noise schemes found in the literature, in such a way that we can simulate the behavior of the classifiers in the presence of noise:

1. **Class noise** usually occurs on the boundaries of the classes, where the examples have similar characteristics—although it might occur on any other areas of the domain. In this paper, class noise is introduced using a *random class noise scheme* [46] (randomly corrupting the class labels of the examples) and a *pairwise class noise scheme* (labeling examples of the majority class with the second majority class) [56,57]. Considering these two schemes, the similarities between any pair of classes and only between the two majority classes are simulated, respectively.
2. **Attribute noise** can proceed from several sources, such as transmission constraints, faults in sensor devices, irregularities in sampling, and transcription errors [47]. The erroneous attribute values can be totally unpredictable, that is, random, or they can imply a low variation with respect to the correct value. We use a *random attribute noise scheme* [56,58] and a *Gaussian attribute noise scheme* [44] in order to simulate each one of the possibilities, respectively. We introduce attribute noise in accordance with the hypothesis that interactions between attributes are weak [56]. As a result, the noise introduced into each attribute has a low correlation with the noise introduced into the rest of the attributes.

Robustness is the capability of an algorithm to build models that are insensitive to data corruptions and suffer less from the impact of noise [25]. Thus, a classification algorithm is said to be more robust than other one if the former builds classifiers which are less influenced by noise than the latter, that is, more robust. In order to analyze the degree of robustness of the classifiers in the presence of noise, we will compare the performance of the classifiers learned with the original (without induced noise) dataset with the performance of the classifiers learned using the noisy dataset. Therefore, those classifiers learned from noisy datasets being more similar (in terms of results) to the noise-free classifiers will be the most robust ones.

3 Addressing multi-class classification problems by decomposition

Multi-class classification problems are frequent in real-world classification tasks. Examples of such problems are the classification of micro-arrays [30], electroencephalogram signals [19] or texts [31], and audio streams [1]. These problems are more general and complex than the special case of two classes, that is, binary classification problems.

In the literature, multi-class classifier learning has been overcome in two different ways [32]: (1) adapting the internal operations of the learning algorithm and (2) decomposing the multi-class problem into a set of easier to solve binary subproblems. The former embeds the management of the multiple classes in the algorithm, whereas the latter aims to reduce the complexity of the original problem by decomposing it into simpler binary subproblems. In such a way, any binary classifier learning algorithm can be used as base learner, without needing to adapt its learning procedure. The first alternative may be a very complex issue [38]; therefore, it is common to use the decomposition alternative when the algorithms are not easily adaptable, that is, suppor vector machines [48], but also when adaptations exist, since its benefits have been proved [16].

3.1 Decomposition strategies for multi-class problems

Several motivations for the usage of binary decomposition strategies in multi-class classification problems can be found in the literature [15,16,24,43]:

- The separation of the classes becomes easier (less complex), since less classes are considered in each subproblem [15,34]. For example, in [28], the classes in a digit recognition problem were shown to be linearly separable when considered in pairs. A simpler alternative than learning a unique nonlinear classifier to separate all classes simultaneously.
- Classification algorithms, whose extension to multi-class problems is not easy, can address multi-class problems using decomposition techniques [15].
- In [39], the advantages of the usage of decomposition was pointed out when the classification errors for different classes have distinct costs. The binarization allows the binary classifiers generated to impose preferences for some of the classes.
- Decomposition allows one to easily parallelize the classifier learning, since the binary subproblems are independent and can be solved with different processors.

Dividing a problem into several new subproblems, which are then independently solved, implies the need of a second phase where the outputs of each problem need to be aggregated. Therefore, decomposition includes two steps:

1. *Problem division*. The problem is decomposed into several binary subproblems which are solved by independent binary classifiers, called *base classifiers* [15]. Different decomposition strategies can be found in the literature [32]. The most common one is OVO [27].
2. *Combination of the outputs* [16]. The different outputs of the binary classifiers must be aggregated in order to output the final class prediction. In [16], an exhaustive study comparing different methods to combine the outputs of the base classifiers in the OVO and OVA strategies is developed. Among these combination methods, the weighted voting [26] and the approaches in the framework of probability estimates [51] are highlighted.

This paper focuses the OVO decomposition strategy due to the several advantages shown in the literature with respect to OVA [15,16,24,43]:

– OVO creates simpler borders between classes than OVA.
– OVO generally obtains a higher classification accuracy and a shorter training time than OVA because the new subproblems are easier and smaller.
– OVA has more of a tendency to create imbalanced datasets which can be counterproductive [17,45].
– The application of the OVO strategy is widely extended and most of the software tools considering binarization techniques use it as default [3,10,20].

3.2 One-vs-One decomposition scheme

The OVO decomposition strategy consists of dividing a classification problem with M classes into $M(M-1)/2$ binary subproblems. A classifier is trained for each new subproblem only considering the examples from the training data corresponding to the pair of classes (λ_i, λ_j) with $i < j$ considered.

When a new instance is going to be classified, it is presented to all the binary classifiers. This way, each classifier discriminating between classes λ_i and λ_j provides a confidence degree $r_{ij} \in [0, 1]$ in favor of the former class (and hence, r_{ji} is computed by $1 - r_{ij}$). These outputs are represented by a score matrix R:

$$R = \begin{pmatrix} - & r_{12} & \cdots & r_{1M} \\ r_{21} & - & \cdots & r_{2M} \\ \vdots & & & \vdots \\ r_{M1} & r_{M2} & \cdots & - \end{pmatrix} \tag{1}$$

The final output is derived from the score matrix by different aggregation models. The most used and simplest combination, also considered in the experiments of this paper, is the application of a voting strategy:

$$\text{Class} = \arg\max_{i=1,\ldots,M} \sum_{1 \le j \ne i \le M} s_{ij} \tag{2}$$

where s_{ij} is 1 if $r_{ij} > r_{ji}$ and 0 otherwise. Therefore, the class with the largest number of votes will be predicted. This strategy has proved to be competitive with different classifiers obtaining similar results in comparison with more complex strategies [16].

4 Experimental framework

First, the base datasets used in the experiments are described (Sect. 4.1). Afterward, how the noise is induced into them is explained (Sect. 4.2). The algorithms used as base classifiers and their parameters are presented in Sect. 4.3. Finally, the methodology for the analysis of the results is explained in Sect. 4.4.

4.1 Base datasets

The experimentation is based on twenty real-world multi-class classification problems from the KEEL-dataset repository[2] [2]. Table 1 shows the datasets sorted by the number of classes (#CLA). Moreover, for each dataset, the number of examples (#EXA) and the number of attributes (#ATT), along with the number of real, integer and nominal attributes (R/I/N) are

[2] http://www.keel.es/datasets.php.

Table 1 Summary description of the classification datasets

Dataset	#CLA	#EXA	#ATT (R/I/N)	Dataset	#CLA	#EXA	#ATT (R/I/N)
Balance	3	625	4 (4/0/0)	Flare	6	1,066	11 (0/0/11)
Contraceptive	3	1,473	9 (0/9/0)	Glass	7	214	9 (9/0/0)
Iris	3	150	4 (4/0/0)	Satimage	7	643	36 (0/36/0)
Newthyroid	3	215	5 (4/1/0)	Segment	7	2,310	19 (19/0/0)
Splice	3	319	60 (0/0/60)	Shuttle	7	2,175	9 (0/9/0)
Thyroid	3	720	21 (6/15/0)	Ecoli	8	336	7 (7/0/0)
Vehicle	4	846	18 (0/18/0)	Led7digit	10	500	7 (7/0/0)
Nursery	5	1,269	8 (0/0/8)	Penbased	10	1,099	16 (0/16/0)
Page-blocks	5	547	10 (4/6/0)	Yeast	10	1,484	8 (8/0/0)
Automobile	6	150	25 (15/0/10)	Vowel	11	990	13 (10/3/0)

presented. Some of the largest datasets (*nursery*, *page-blocks*, *penbased*, *satimage*, *splice*, and *led7digit*) were stratified at 10 % in order to reduce the computational time required for training, given the large amount of executions carried out. For datasets containing missing values (such as *automobile* or *dermatology*), instances with missing values were removed from the dataset before the partitioning.

4.2 Introducing noise into datasets

In the datasets presented in the previous section, the initial amount and type of noise present on them are unknown. Therefore, no assumptions about the base noise type and level can be made. For this reason, these datasets are considered to be noise-free, in the sense that no new noise has been induced. In order to control the amount of noise in each dataset and to check how it affects the classifiers, noise is introduced into each dataset in a supervised manner. Four different noise schemes, which are proposed in the specialized literature, are used in order to introduce a noise level of x % into each dataset. The following procedures are followed in order to induce the different noise schemes:

1. **Introduction of class noise.**
   - **Random class noise** [46]. It supposes that exactly x % of the examples are corrupted. The class labels of these examples are randomly changed by other one out of the M classes.
   - **Pairwise class noise** [56,57] . Being X the majority class and Y the second majority class, an example with the label X has a probability of $x/100$ of being incorrectly labeled as Y.
2. **Introduction of attribute noise.**
   - **Random attribute noise** [56,58]. x % of the values of each attribute in the dataset are corrupted. To corrupt an attribute A_i, approximately x % of the examples in the data set are chosen, and their A_i value is assigned a random value from $\mathbb{D}_i$. A uniform distribution is used either for numerical or nominal attributes.
   - **Gaussian attribute noise** [44]. This scheme is similar to the random attribute noise, but in this case, the A_i values are corrupted adding a random value to them following a Gaussian distribution of $mean = 0$ and $standard\ deviation = (max - min)/5$,

being *max* and *min* the upper and lower limits of $\mathbb{D}_i$, respectively. Nominal attributes are treated as in the case of the random attribute noise.

In order to create a noisy dataset from an original noise-free dataset, the noise is introduced into the training partitions as follows:

1. A unique identifier, that is, an index, is assigned to each example of the full original dataset.
2. A level of noise x %, of either class noise (random or pairwise) or attribute noise (random or Gaussian), is introduced into a copy of the full original dataset. Each example maintains its identifier in this noisy copy.
3. Both datasets, the original one and the noisy copy, are partitioned into fivefolds. Each one of the folds in the original dataset must have examples with the same identifiers that the corresponding fold in the noisy copy.
4. The training partitions are built from the noisy copy (using 4 of the fivefolds), whereas the test partitions are formed of instances from the original dataset (using the fold with examples whose identifiers have not been considered in the training set).

Introducing noise into the training partitions while keeping the test partitions noise-free, as performed in other works in the literature [56], allows one to observe how noisy data affect the training process, observing how the test results are degraded depending on the type and level of noise introduced. Furthermore, the robustness of the methods can be better studied since the effects of noise are isolated in the training process.

The accuracy estimation of the classifiers in a dataset is obtained by means of 5 runs of a stratified fivefolds cross-validation (5-fcv). Hence, a total of 25 runs per dataset, noise type, and level are averaged. Each fold has a larger number of examples considering 5 partitions than considering a higher number of partitions, for example, 10, which is desirable in multi-class problems where some of the classes might be not represented in the test sets. Therefore, the performance of each classifier built is evaluated with a larger number of examples in each test set of the 5-fcv. This fact lets that little modifications in the classifier due to the effect of noise on training sets to be shown better in test sets because we consider a larger number of examples. Furthermore, performing 5 runs of each 5-fcv, the final results obtained are stabilized.

A large collection of new noisy datasets are created from the aforementioned 20 base datasets. Both types of noise are independently considered: class and attribute noise. For each type of noise, the noise levels ranging from $x = 0\,\%$ (base datasets) to $x = 50\,\%$, by increments of 5 %, are studied. Therefore, 200 noisy datasets are created for each of the four noise schemes. The total number of datasets in the experimentation is 820. Hence, considering the 5×5fcv of the 820 datasets, 20,500 executions are carried out for each classifier (which are repeated for the OVO and non-OVO versions). All these datasets are available on the web page associated with this paper.

4.3 Algorithms and parameters

The choice of the learning algorithms—C4.5 [42], RIPPER [11], and k-NN [35]—has been made on the basis of their good behavior in a large number of real-world problems and their different characteristics against noise. They have been also considered in previous works focused on noisy data [37,56]. Moreover, notice that all these learning methods are capable of handling multiple classes inherently, which is needed in order to be comparable against the usage of the OVO strategy.

Table 2 Setup of the parameters for the classification algorithms

Rule-based learning		Instance-based learning
C4.5	RIPPER	k-NN
Confidence: $c = 0.25$	Folds: $f = 3$	Neighbors: $k = 3, 5$
Min. instances per leaf: $i = 2$	Optimizations: $r = 2$	Distance: HVDM
Pruned tree		

C4.5 and RIPPER are considered robust learners tolerant to noisy data. Both use pruning strategies to reduce the chances of classifiers to be affected by noisy instances from the training data [40,41]. However, when the noise level is relatively high, even these robust learners may obtain a poor performance. Regarding k-NN, it is known to be more sensitive to noise than other learning algorithms. Furthermore, the value of k determines a higher or lower sensitivity to noise [29], since a larger value of k usually implies a lower influence on the prediction of the closest potential noisy examples. In this manner, this paper studies the effect of noise on the performance of robust and noise-sensitive learners, and more specifically focusing on multi-class problems, it compares their baseline results with respect to the usage of the OVO strategy. Hence, we check whether the advantages usually attributed to OVO are maintained in the presence of noise or not; in such a way, we provide an in-depth study of these cases (Sects. 5, 6) followed by a thorough explanation of the results (Sect. 7).

The classification algorithms have been executed using the default parameters recommended by the authors, which are shown in Table 2.

4.4 Methodology of analysis

In order to check the suitability of methods using OVO when dealing with noisy data, the results of C4.5, RIPPER, 3-NN, and 5-NN with and without decomposition are compared one another using three distinct properties:

1. The performance of the classification algorithms on the test sets for each level of induced noise defined as its accuracy rate. For the sake of brevity, only averaged results are shown (the rest can be found on the web page associated with this paper), but it must be taken into account that our conclusions are based on the proper statistical analysis, which considers all the results (not averaged).
2. The *relative loss of accuracy* (RLA) (Eq. 3) is used to measure the percentage of variation of the accuracy of the classifiers in a concrete noise level with respect to the original case with no additional noise:

$$\mathrm{RLA}_{x\,\%} = \frac{\mathrm{Acc}_{0\,\%} - \mathrm{Acc}_{x\,\%}}{\mathrm{Acc}_{0\,\%}}, \tag{3}$$

   where $\mathrm{RLA}_{x\,\%}$ is the relative loss of accuracy at a noise level $x\,\%$, $\mathrm{Acc}_{0\,\%}$ is the test accuracy in the original case, that is, with $0\,\%$ of induced noise, and $\mathrm{Acc}_{x\,\%}$ is the test accuracy with a noise level $x\,\%$.
3. Box-plots are used to easily analyze the distribution of the RLA values. The values of the median and the interquartile range, along with its size, can provide a good approximation about the robustness of the methods over all the datasets. Thus, a method with a lower median and a lower and more compact interquartile range will be always preferable, since

its behavior with new noisy datasets is more similar in accuracy to that obtained with the original dataset.

In order to properly analyze the performance and RLA results, the Wilcoxon's signed rank statistical test is used, as suggested in the literature [12]. This is a nonparametric pairwise test that aims to detect significant differences between two sample means, that is, the behavior of the two algorithms involved in each comparison. For each type and noise level, the OVO and non-OVO versions will be compared using Wilcoxon's test and the p values associated with these comparisons will be obtained. The p value represents the lowest level of significance of a hypothesis that results in a rejection and it allows one to know whether two algorithms are significantly different and the degree of their difference. We will consider a difference to be significant if the p value obtained is lower than 0.1—even though p values slightly higher than 0.1 might be showing important differences. We study both, performance and robustness, because the conclusions reached with one of these metrics necessary not imply the same conclusions with the other one.

5 Analysis of the OVO strategy with class noise

In this section, the performance and robustness of the classification algorithms using the OVO decomposition with respect to its baseline results when dealing with data suffering from class noise are analyzed. Section 5.1 is devoted to the study of the random class noise scheme, whereas Sect. 5.2 analyzes the pairwise class noise scheme. The results obtained for each single dataset can be found on the web page associated with this paper.

5.1 Random class noise scheme

Table 3 shows the test accuracy and RLA results for each classification algorithm at each noise level along with the associated p-values between the OVO and the non-OVO version from the Wilcoxon's test. The few exceptions where the baseline classifiers obtain more ranks than the OVO version in the Wilcoxon's test are indicated with a star next to the p value.

From these results, the following points can be highlighted:

- The test accuracy of the methods using OVO is higher at all the noise levels. Moreover, the low p values show that this advantage in favor of OVO is significant.
- The RLA values of the methods using OVO are lower than those of the baseline methods at all noise levels. These differences are also statistically significant as reflected by the low p values. Only at some very low noise levels—5 and 10 % for C4.5 and 5 % for 5-NN—the results between the OVO and the non-OVO version are statistically equivalent, but notice that the OVO decomposition does not hinder the results, simply the loss is not lower.

Figure 1 shows the distribution of the RLA results of each algorithm at each noise level on datasets with random class noise. For all the classification algorithms, these graphics show that the medians of the RLA results of the OVO approach are much lower with respect to those of non-OVO. Moreover, the interquartile range is generally lower and more compact for OVO. Therefore, when noise randomly affects the class labels, the suitability of the OVO decomposition is proved to be advantageous. The binary decomposition of the problem provides better predictions. Hence, OVO is more robust against this type of class noise, obtaining a greater performance and a lower RLA result. This may be attributed to the

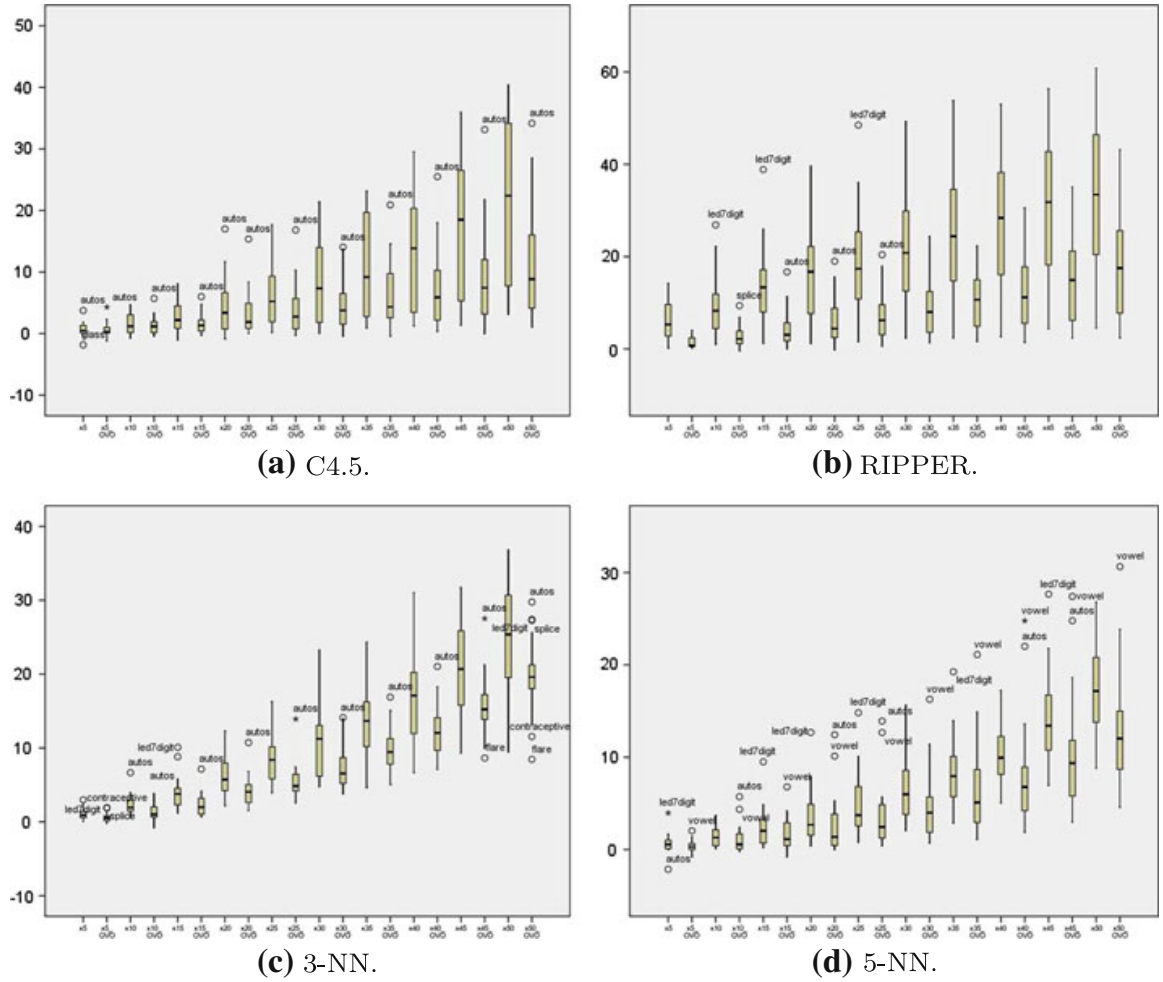
Table 3 Test accuracy, RLA results, and p values on datasets with random class noise

	C4.5		RIPPER		3-NN		5-NN	
	Base	OVO	Base	OVO	Base	OVO	Base	OVO
Test accuracy								
Results (%)								
0	81.66	82.70	77.92	82.15	81.79	83.39	82.10	83.45
5	81.13	82.14	73.51	80.82	81.00	82.93	81.65	83.14
10	80.50	81.71	71.30	79.86	80.00	82.29	81.01	82.56
15	79.37	81.39	68.52	78.41	78.76	81.49	80.39	82.13
20	78.13	80.27	66.71	77.35	76.91	80.01	79.55	81.36
25	76.96	79.54	64.26	76.25	75.15	78.99	78.43	80.58
30	75.22	78.87	62.91	74.98	73.24	77.39	77.21	79.82
35	73.35	77.89	60.48	73.40	70.82	75.30	75.48	78.28
40	71.10	76.88	58.32	72.12	68.18	73.08	73.82	76.83
45	67.96	75.74	56.18	69.98	64.90	70.27	70.86	74.93
50	64.18	73.71	53.79	67.56	61.66	66.98	68.04	72.73
p values (%)								
0	–		–		–		–	
5	0.0206		0.0001		0.0003		0.0033	
10	0.0124		0.0001		0.0001		0.0036	
15	0.0008		0.0001		0.0001		0.0137	
20	0.0028		0.0001		0.0004		0.0017	
25	0.0010		0.0001		0.0002		0.0032	
30	0.0002		0.0001		0.0003		0.0013	
35	0.0003		0.0001		0.0008		0.0028	
40	0.0002		0.0001		0.0005		0.0111	
45	0.0003		0.0001		0.0019		0.0019	
50	0.0001		0.0001		0.0028		0.0008	
RLA value								
Results (%)								
0	–		–		–		–	
5	0.66	0.73	6.04	1.59	1.07	0.57	0.63	0.38
10	1.56	1.28	9.35	2.79	2.27	1.33	1.46	1.12
15	3.01	1.65	13.13	4.56	3.96	2.34	2.37	1.67
20	4.63	3.15	15.44	5.86	6.20	4.15	3.48	2.67
25	6.12	3.98	18.89	7.23	8.37	5.33	4.87	3.61
30	8.36	4.90	20.74	8.82	10.78	7.27	6.40	4.53
35	10.77	6.11	23.97	10.76	13.48	9.74	8.46	6.38
40	13.46	7.41	26.72	12.35	16.79	12.35	10.30	8.11
45	17.30	8.86	29.70	15.05	20.74	15.79	14.12	10.48
50	21.87	11.29	32.74	18.10	24.51	19.64	17.47	13.12
p values (%)								
0	–		–		–		–	
5	0.7369*		0.0003		0.0040		0.6012	

Table 3 continued

	C4.5		RIPPER		3-NN		5-NN	
	Base	OVO	Base	OVO	Base	OVO	Base	OVO
10	0.5257		0.0001		0.0017		0.1354	
15	0.0025		0.0001		0.0012		0.0731	
20	0.0304		0.0001		0.0004		0.0479	
25	0.0017		0.0001		0.0005		0.0522	
30	0.0006		0.0001		0.0005		0.0124	
35	0.0003		0.0001		0.0025		0.0276	
40	0.0001		0.0001		0.0012		0.0333	
45	0.0002		0.0001		0.0022		0.0057	
50	0.0001		0.0001		0.0036		0.0015	

Those cases where the baseline classifiers obtain more ranks than the OVO version in the Wilcoxon's test are indicated with a *



(a) C4.5. **(b)** RIPPER. **(c)** 3-NN. **(d)** 5-NN.

Fig. 1 Box-plots representing the distribution of the RLA results on datasets with random class noise

division of the mislabeled examples, hindering in this way only some classifiers. As these classifiers are only a part of the global system, they do not affect as much as in the original case.

5.2 Pairwise class noise scheme

The pairwise class noise results are shown in Table 4. The test accuracy and RLA of each classification algorithm at each noise level are presented. The associated p values between the OVO and non-OVO version of each algorithm are also shown. The following points can be concluded:

- The test accuracy of the methods using OVO is statistically better—as shown by the p values—than those of non-OVO at almost all noise levels. C4.5 and RIPPER at a noise level of 50 % are exceptions: with C4.5, both OVO and non-OVO, are statistically equivalent; with RIPPER, the non-OVO version is statistically better. In the last part of this subsection, we try to obtain an explanation to these results.
- Attending to the RLA results:
  - C4.5 with OVO is only statistically better at intermediate noise levels 15–20 %. Both methods are statistically equivalent in the rest of noise levels—except at the maximum noise level, 50 %, where non-OVO is statistically better. But having equivalent RLA, OVO performs statistically better in most of the cases.
  - Both versions of RIPPER, with and without OVO, are statistically equivalent at all noise levels—except at the maximum noise level 50 % where non-OVO is statistically better.
  - 3-NN and 5-NN with OVO present better RLA results. The lower p-values are generally obtained from 20 to 25 % of noise. Other levels of noise are also remarkable with 5-NN, as 5 %.

Figure 2 shows the distribution of RLA results with pairwise class noise. These graphics show similar conclusions to those obtained from the analysis of the RLA results and the corresponding p-values:

- At the lowest noise levels, C4.5 and RIPPER using OVO are slightly better than non-OVO (attending to their medians and interquartile ranges). However, from 30 % on (C4.5) and from 25 % on (RIPPER), the methods using OVO are more detrimental than those not using it.
- 3-NN and 5-NN with and without OVO are much more similar, but OVO is better at some noise levels.

These results show that OVO achieves more accurate predictions when dealing with this type of noise; however, it is not so advantageous with C4.5 or RIPPER as with k-NN in terms of robustness when noise only affects one class. For example, the behavior of RIPPER with this noise scheme can be related to the hierarchical way in which the rules are learned: it starts learning rules of the class with the lowest number of examples and continues learning those classes having more examples. When introducing this type of noise, RIPPER might change its training order, but the remaining part of the majority class can still be properly learned, since it has now more priority. Moreover, the original second majority class, now with noisy examples, will probably be the last one to be learned and it would depend on how the rest of the classes have been learned. Decomposing the problem with OVO, a considerable number of classifiers will have a notable quantity of noise—those of the majority and the second majority classes and hence, the tendency to predict the original majority class decreases—when the noise level is high, it strongly affects the accuracy, since the majority has more influence on it.

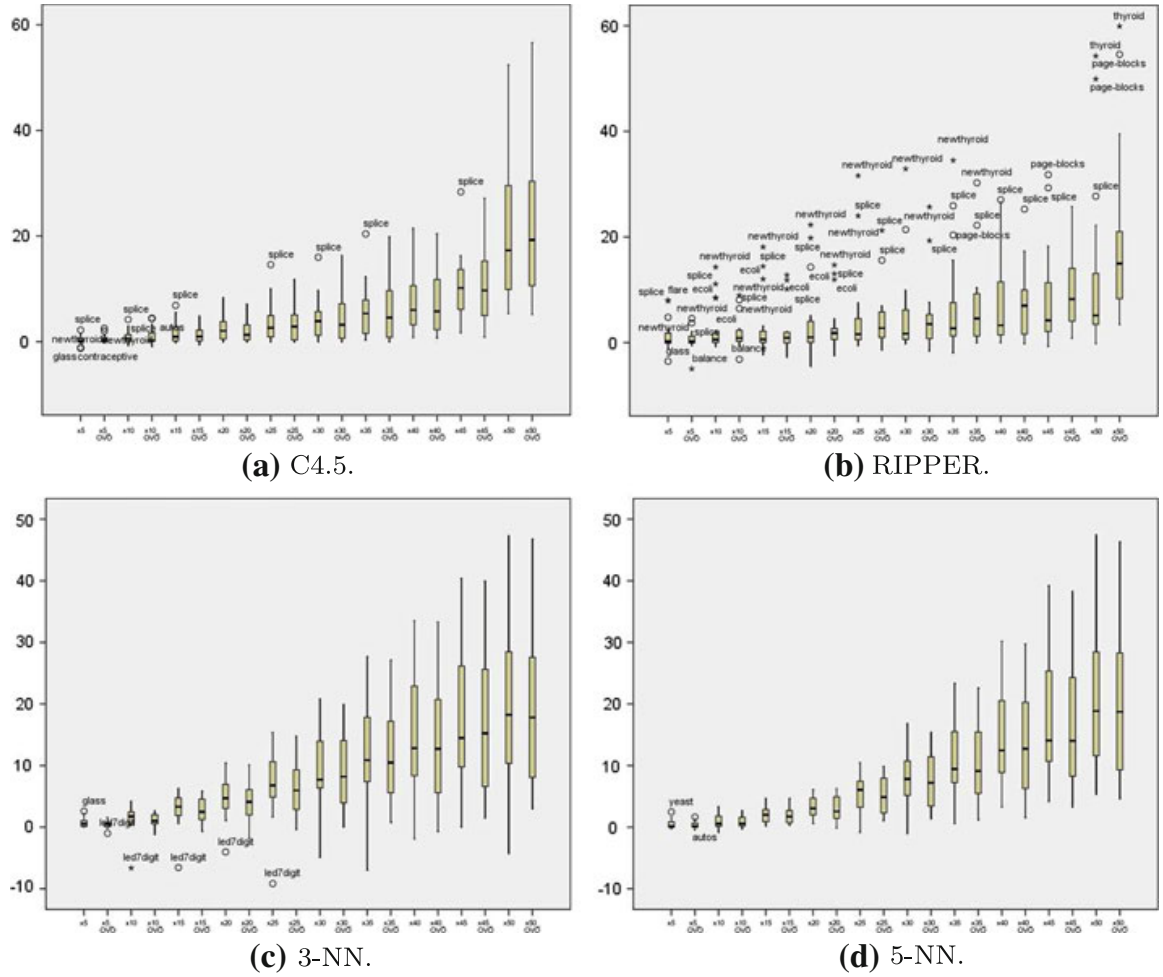
Table 4 Test accuracy, RLA results, and p values on datasets with pairwise class noise

	C4.5		RIPPER		3-NN		5-NN	
	Base	OVO	Base	OVO	Base	OVO	Base	OVO
Test accuracy								
Results (%)								
0	81.66	82.70	77.92	82.15	81.79	83.39	82.10	83.45
5	81.40	82.24	76.94	81.71	81.24	83.02	81.78	83.19
10	80.94	81.86	75.94	80.71	80.65	82.36	81.42	82.82
15	80.43	81.71	75.64	80.32	79.25	80.97	80.49	81.94
20	79.82	81.03	74.77	79.62	77.75	79.65	79.41	81.01
25	78.96	80.28	73.89	78.67	75.88	77.71	77.55	79.08
30	78.49	79.26	73.38	78.05	73.53	75.47	75.29	76.81
35	77.41	78.28	71.89	76.42	71.24	73.18	72.92	74.50
40	76.17	76.91	71.60	76.19	68.77	70.89	69.89	71.65
45	73.45	74.26	70.73	74.04	66.55	68.48	67.13	68.83
50	63.63	63.52	67.11	65.78	64.11	65.86	64.02	65.52
p values (%)								
0	0.0070		0.0002		0.0522		0.0930	
5	0.025		0.0001		0.0152		0.0228	
10	0.0033		0.0003		0.0019		0.0137	
15	0.0022		0.0003		0.0036		0.0090	
20	0.0017		0.0002		0.0005		0.0022	
25	0.0008		0.0005		0.0012		0.0015	
30	0.0090		0.0001		0.0045		0.0100	
35	0.0366		0.0013		0.0002		0.0032	
40	0.0276		0.0003		0.0015		0.0040	
45	0.0333		0.0333		0.0022		0.0072	
50	0.5016*		0.0930*		0.0008		0.0057	
RLA value								
Results (%)								
0	–		–		–		–	
5	0.30	0.56	1.17	0.48	0.74	0.44	0.48	0.34
10	0.91	1.01	2.38	1.72	1.25	1.22	0.89	0.82
15	1.62	1.25	2.58	2.17	2.88	2.81	2.05	1.88
20	2.39	2.13	3.52	3.03	4.69	4.33	3.29	2.93
25	3.52	3.13	4.58	4.22	6.75	6.58	5.48	5.22
30	4.16	4.49	5.13	4.84	9.57	9.09	8.05	7.81
35	5.50	5.69	6.77	6.82	12.12	11.76	10.76	10.41
40	7.01	7.38	7.25	7.12	15.03	14.29	14.37	13.68
45	10.38	10.65	8.28	9.70	17.57	17.01	17.53	16.86
50	21.03	22.28	12.22	18.75	20.23	20.00	21.06	20.67
p values (%)								
0	–		–		–		–	
5	0.2043*		0.2790		0.2954		0.1354	

Table 4 continued

	C4.5		RIPPER		3-NN		5-NN	
	Base	OVO	Base	OVO	Base	OVO	Base	OVO
10	0.8721*		0.3317		0.3507		0.3507	
15	0.0304		0.3507		0.2043		0.5503	
20	0.0859		0.4781		0.2959		0.0674	
25	0.3317		0.9702		0.1005		0.0620	
30	0.6813		0.6542		0.2471		0.3507	
35	0.7652		0.6542*		0.0674		0.1913	
40	0.6274		0.6274*		0.1169		0.0793	
45	0.7369		0.1169*		0.1259		0.0522	
50	0.0400*		0.0001*		0.0731		0.0620	

Those cases where the baseline classifiers obtain more ranks than the OVO version in the Wilcoxon's test are indicated with a *

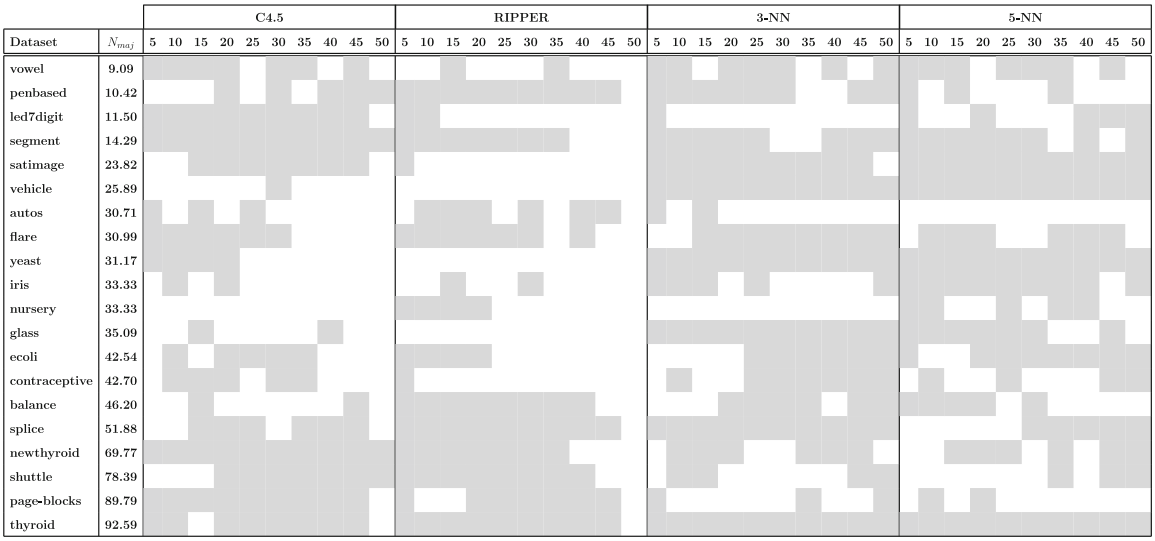


(a) C4.5. **(b)** RIPPER. **(c)** 3-NN. **(d)** 5-NN.

Fig. 2 Box-plots representing the distribution of the RLA results on datasets with pairwise class noise

In contrast with the rest of noise schemes, with this noise scheme, all the datasets have different real percentages of noisy examples at the same noise level of x %. This is because each dataset has a different number of examples of the majority class, and thus a noise level of x % does not affect all the datasets in the same way. In this case, the percentage of

Table 5 Noise levels—remarked with gray—where the RLA results of OVO are better that those of the non-OVO version



		C4.5										RIPPER										3-NN										5-NN									
Dataset	N_{maj}	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50
vowel	9.09																																								
penbased	10.42																																								
led7digit	11.50																																								
segment	14.29																																								
satimage	23.82																																								
vehicle	25.89																																								
autos	30.71																																								
flare	30.99																																								
yeast	31.17																																								
iris	33.33																																								
nursery	33.33																																								
glass	35.09																																								
ecoli	42.54																																								
contraceptive	42.70																																								
balance	46.20																																								
splice	51.88																																								
newthyroid	69.77																																								
shuttle	78.39																																								
page-blocks	89.79																																								
thyroid	92.59																																								

noisy examples with a noise level of $x\,\%$ is computed as $(x \cdot N_{\text{maj}})/100$, where N_{maj} is the percentage of examples of the majority class.

Therefore, in order to deepen in the analysis of the robustness results with this noise scheme, the RLA results on each single dataset must be studied. With this aim, Table 5 shows for each dataset, remarked in gray, those noise levels where the OVO version obtains better RLA results than the non-OVO version. The datasets appearing in this table are sorted by their percentage of examples of the majority class (N_{maj}). Noise affects adjacent datasets similarly—since close datasets have similar percentages of noisy examples at the same noise level of $x\,\%$. Thus, those datasets and noise levels where OVO works better are easily identified.

From this table, several conclusions can be drawn:

- C4.5 and RIPPER with OVO have better RLA results on those dataset having a large number of examples of the majority class.
- 3-NN and 5-NN with OVO have better RLA results, without being affected by the number of examples of the majority class.

Therefore, following the previous analysis, we can state that when the noise only affects one class, OVO clearly provides advantages in terms of the accuracy of the classifiers built. However, methods using OVO are affected by noise in a similar way to those that do not use OVO, that is, both have similar robustness, although OVO outstands at some noise levels. The robustness of C4.5 and RIPPER with OVO is remarkable with those datasets having a large number of majority class examples, that is, with those dataset where the noise percentage is the highest. The robustness of k-NN methods is not so dependent on the number of examples of the majority class, and the methods with OVO have better RLA results, which are distributed more homogeneously.

6 Analysis of the OVO strategy with attribute noise

In this section, the performance and robustness of the classification algorithms using OVO in comparison with its non-OVO version when dealing with data with attribute noise are

analyzed. Section 6.1 is devoted to the study of the random attribute noise scheme, whereas Sect. 6.2 analyzes the Gaussian attribute noise scheme.

6.1 Random attribute noise scheme

The test accuracy, RLA results, and p values of each classification algorithm at each noise level are shown in Table 6. From these results, the following points can be highlighted:

- The test accuracy of the methods using OVO is always statistically better at all the noise levels.
- The RLA values of the methods using OVO are lower than those of the baseline methods at all noise levels—except in the case of C4.5 with a 5 % of noise level. Regarding the p values, a clear tendency is observed, the p-value decreases when the noise level increases with all the algorithms.
- With most of the methods—C4.5, RIPPER, and 5-NN—the p values of the RLA results at the lowest noise levels (up to 20–25 %) show that the robustness of OVO and non-OVO methods is statistically equivalent. From that point on, the OVO versions statistically outperform the non-OVO ones. The case of 3-NN is even more favorable to OVO, since only a high p value is found at the lowest noise level, i.e., 5 %.

Figure 3 shows the box-plots of RLA results. For all the classification algorithms and noise levels, these graphics show that the medians of the RLA results of OVO are much lower with respect those of non-OVO. Moreover, the interquartile range is also generally lower and more compact for the methods using OVO.

Therefore, the usage of OVO is clearly advantageous in terms of accuracy and robustness when noise affects the attributes in a random and uniform way. This behavior is particularly notable with the highest noise levels, where the effects of noise are expected to be more detrimental.

6.2 Gaussian attribute noise scheme

In Table 7, the test accuracy and RLA results of each classification algorithm at each noise level, along with the associated p value between the OVO and non-OVO version of each algorithm, are shown. From these results, the following conclusions can be pointed out:

- The test accuracy of the methods using OVO is better at all the noise levels. Moreover, the low p values show that this advantage in favor of OVO is statistically significant.
- Regarding the RLA results, the p values show a clear decreasing tendency when the noise level increases with all the algorithms. In the case of C4.5, OVO is statistically better from a 35 % of noise level on, and in 3-NN from 20 % on. RIPPER and 5-NN are statistically equivalent at all noise levels—although 5-NN with OVO obtains higher Wilcoxon's ranks.

It is important to note that in some cases, particularly in the comparisons involving RIPPER, some RLA results show that OVO is better than the non-OVO version in average but the latter obtains more ranks in the statistical test—even though these differences are not significant. This is due to the extreme results of some individual datasets, such as *led7digit* or *flare*, in which the RLA results of the non-OVO version are much worse than those of the OVO version. Anyway, we should notice that average results themselves are not meaningful and the corresponding nonparametric statistical analysis must be carried out in order to extract meaningful conclusions, which reflects the real differences between algorithms.

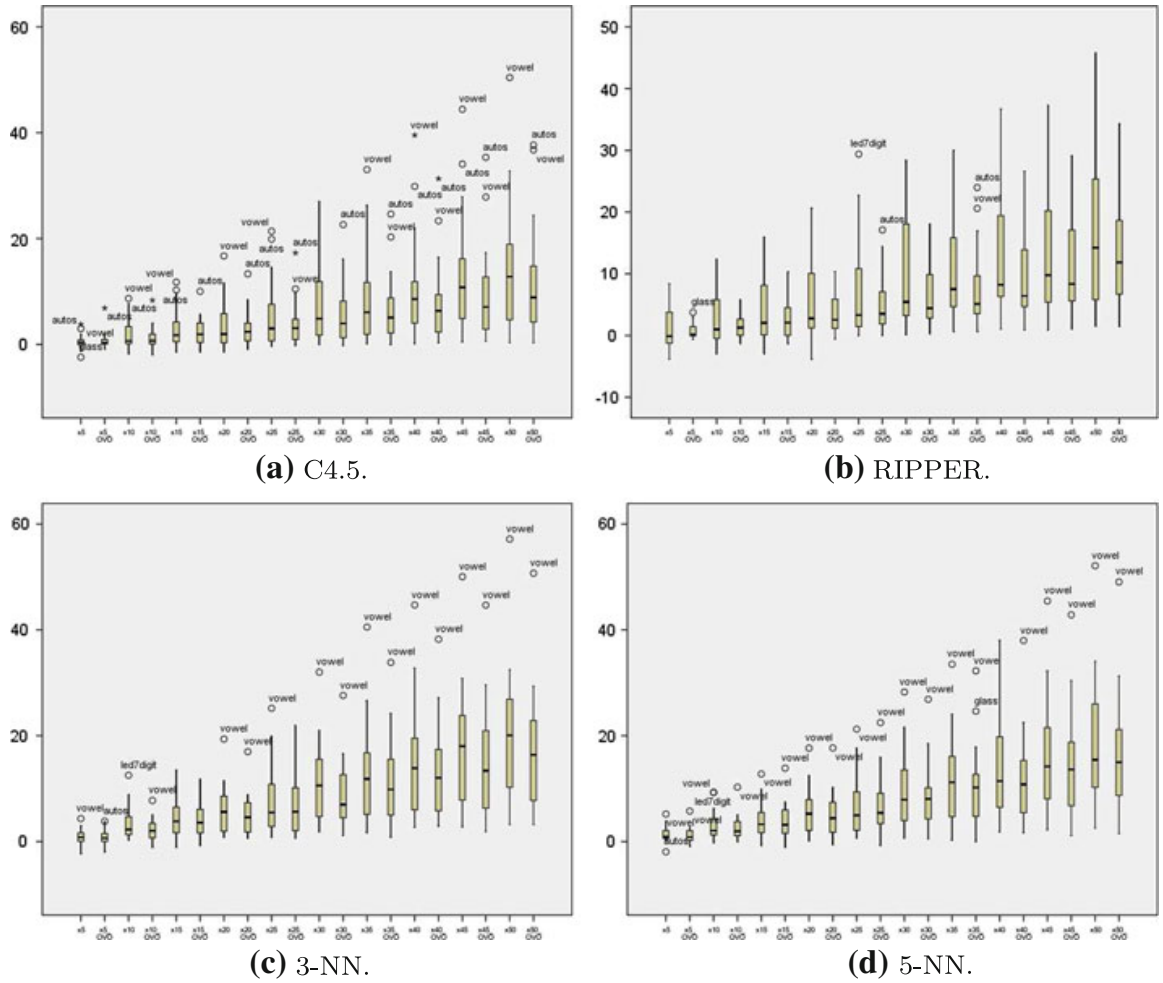
Table 6 Test accuracy, RLA results, and p values on datasets with random attribute noise

	C4.5		RIPPER		3-NN		5-NN	
	Base	OVO	Base	OVO	Base	OVO	Base	OVO
Test accuracy								
Results (%)								
0	81.66	82.70	77.92	82.15	81.79	83.39	82.10	83.45
5	81.26	82.10	77.18	81.50	80.90	82.53	81.01	82.45
10	80.31	81.65	76.08	80.85	79.23	81.52	79.81	81.34
15	79.39	80.83	74.83	80.03	78.31	80.22	78.97	80.31
20	78.71	80.27	73.95	79.15	76.99	79.20	77.63	79.38
25	77.54	79.64	72.77	78.11	75.36	77.71	76.58	77.96
30	76.01	78.25	71.25	77.06	73.37	76.05	74.68	76.46
35	74.55	77.42	70.05	76.15	71.62	74.28	73.05	75.01
40	73.58	76.19	68.66	74.56	69.62	72.66	71.29	73.65
45	71.79	75.21	67.64	73.35	67.56	70.56	69.26	71.53
50	70.49	73.51	65.50	71.66	65.88	69.15	67.72	70.07
p values (%)								
0	0.0070		0.0002		0.0522		0.0930	
5	0.0400		0.0012		0.0038		0.0100	
10	0.0169		0.0003		0.0004		0.0910	
15	0.0169		0.0001		0.0022		0.0707	
20	0.0057		0.0003		0.0008		0.0015	
25	0.0007		0.0005		0.0017		0.0080	
30	0.0043		0.0001		0.0005		0.0112	
35	0.0004		0.0001		0.0019		0.0032	
40	0.0032		0.0001		0.0005		0.0006	
45	0.0009		0.0002		0.0008		0.0012	
50	0.0036		0.0007		0.0001		0.0011	
RLA value								
Results (%)								
0	–		–		–		–	
5	0.50	0.74	0.97	0.80	0.98	0.94	1.39	1.20
10	1.82	1.32	2.56	1.62	3.45	2.23	3.03	2.62
15	3.02	2.41	4.24	2.69	4.31	3.82	3.97	3.87
20	3.88	3.11	5.46	3.77	5.85	4.99	5.72	5.03
25	5.48	3.87	7.10	5.13	7.96	6.87	6.94	6.80
30	7.54	5.77	9.20	6.42	10.71	9.01	9.57	8.76
35	9.38	6.70	10.89	7.57	12.65	11.12	11.57	10.49
40	10.64	8.25	12.81	9.64	15.22	12.98	13.73	12.08
45	13.04	9.60	14.13	11.24	17.91	15.72	16.34	14.85
50	14.74	11.74	17.21	13.33	19.96	17.40	18.14	16.55
p values (%)								
0	–		–		–		–	
5	0.4115*		0.5016*		0.8813*		0.5755	

Table 6 continued

	C4.5		RIPPER		3-NN		5-NN	
	Base	OVO	Base	OVO	Base	OVO	Base	OVO
10	0.4781		0.5755		0.1560		1.0000*	
15	0.2471		0.3507		0.2627		0.9108	
20	0.2471		0.1454		0.0930		0.1354	
25	0.0930		0.2322		0.0620		0.5257	
30	0.0304		0.0438		0.0124		0.1672	
35	0.0015		0.0064		0.0333		0.0731	
40	0.0569		0.0036		0.0100		0.0111	
45	0.0137		0.0251		0.0064		0.0152	
50	0.0152		0.0064		0.0008		0.0228	

Those cases where the baseline classifiers obtain more ranks than the OVO version in the Wilcoxon's test are indicated with a *



(a) C4.5. **(b)** RIPPER. **(c)** 3-NN. **(d)** 5-NN.

Fig. 3 Box-plots representing the distribution of the RLA results on datasets with random attribute noise

Figure 4 shows the distribution of the RLA results by means of box-plots. As in the case of the random attribute noise, these graphics show that the medians and interquartile ranges of the RLA results of OVO are much lower with respect to those of non-OVO.

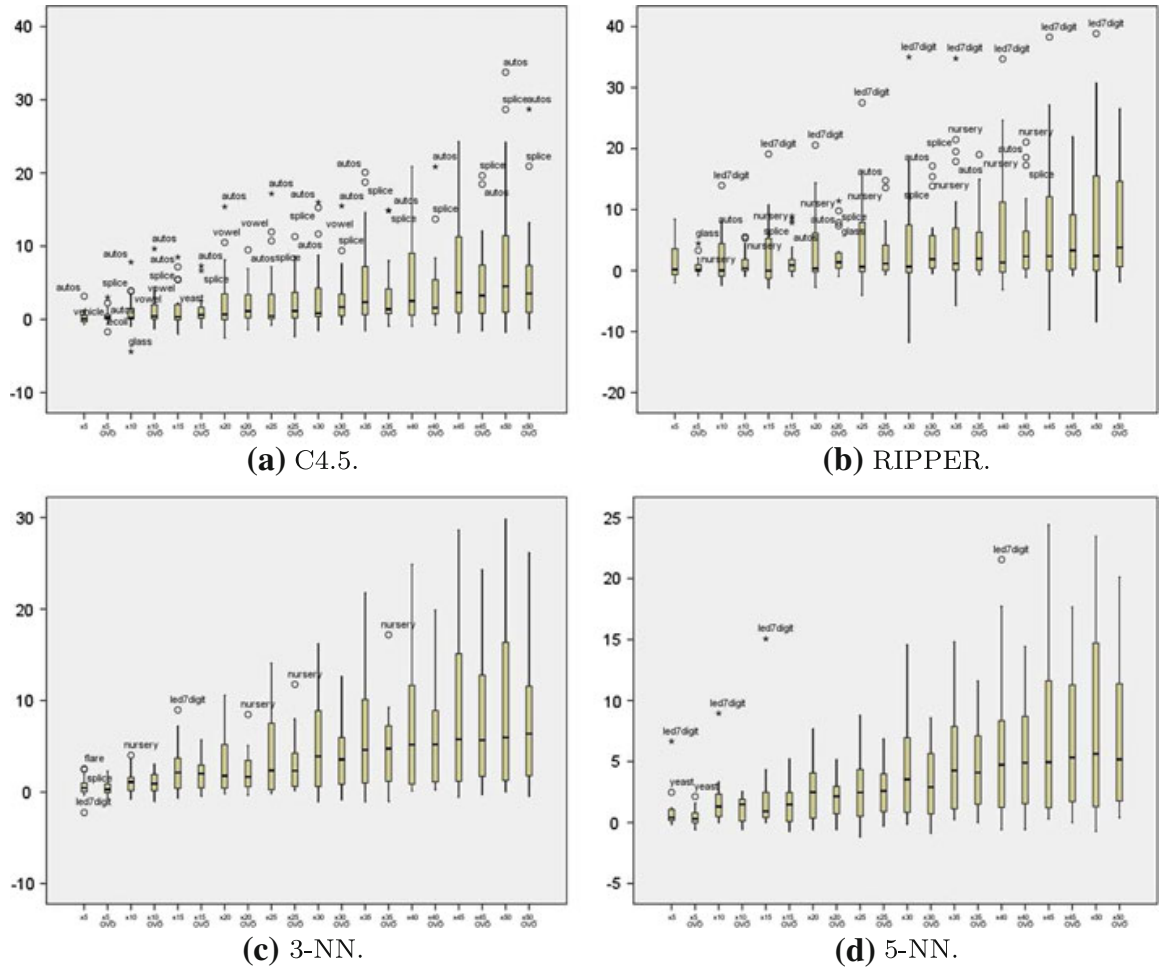
Table 7 Test accuracy, RLA results and p values on datasets with Gaussian attribute noise

	C4.5		RIPPER		3-NN		5-NN	
	Base	OVO	Base	OVO	Base	OVO	Base	OVO
Test accuracy								
Results (%)								
0	81.66	82.70	77.92	82.15	81.79	83.39	82.10	83.45
5	81.46	82.33	76.82	81.64	81.29	83.02	81.52	83.09
10	80.93	81.67	76.53	81.12	80.88	82.54	80.91	82.52
15	80.51	81.69	76.16	80.64	79.87	81.72	80.61	82.21
20	79.77	81.11	75.35	80.06	79.56	81.41	80.16	81.74
25	79.31	80.98	74.69	79.72	78.87	80.90	79.85	81.21
30	79.03	80.40	74.46	78.93	77.98	80.29	78.84	80.77
35	77.95	79.94	73.85	78.76	77.05	79.35	78.12	79.75
40	77.36	79.51	72.94	78.10	76.27	78.60	77.53	79.11
45	76.38	78.64	72.37	77.34	75.11	77.48	76.58	78.25
50	75.29	78.03	71.57	76.27	74.90	77.10	76.02	77.72
p values (%)								
0	0.0070		0.0002		0.0522		0.0930	
5	0.0442		0.0003		0.0033		0.0050	
10	0.1262		0.0004		0.0089		0.0064	
15	0.0152		0.0003		0.0033		0.0169	
20	0.0048		0.0002		0.0033		0.0036	
25	0.0019		0.0002		0.0100		0.0187	
30	0.0051		0.0003		0.0008		0.0025	
35	0.0007		0.0002		0.0036		0.0040	
40	0.0019		0.0003		0.0025		0.1262	
45	0.0004		0.0004		0.0017		0.0364	
50	0.0004		0.0008		0.0019		0.0251	
RLA value								
Results (%)								
0	–		–		–		–	
5	0.28	0.47	1.53	0.62	0.57	0.44	0.88	0.46
10	0.92	1.27	1.90	1.26	1.11	0.98	1.68	1.13
15	1.53	1.25	2.38	1.84	2.57	2.02	2.10	1.56
20	2.40	1.99	3.49	2.59	2.74	2.32	2.51	2.09
25	3.05	2.13	4.44	2.97	3.73	2.98	2.91	2.75
30	3.42	2.91	4.66	3.95	5.00	3.74	4.34	3.32
35	4.86	3.43	5.52	4.18	6.27	4.87	5.19	4.61
40	5.67	4.03	6.74	4.96	7.23	5.85	6.03	5.38
45	6.95	5.14	7.46	5.99	8.79	7.26	7.25	6.51
50	8.37	5.87	8.50	7.35	8.80	7.64	7.82	7.16
p values (%)								
0	–		–		–		–	
5	0.3144*		0.5503		0.6274		0.4781	

Table 7 continued

	C4.5		RIPPER		3-NN		5-NN	
	Base	OVO	Base	OVO	Base	OVO	Base	OVO
10	0.0766*		0.8519*		0.9702		0.4115	
15	0.5755		0.3507*		0.5016		0.4115	
20	0.8405		0.9108*		0.1913		0.3905	
25	0.3547		0.9702*		0.2627		0.9108	
30	0.6542		0.2627*		0.1005		0.2627	
35	0.1354		1.0000*		0.1169		0.3507	
40	0.1169		0.3905		0.0620		0.9405	
45	0.0930		0.9405*		0.0859		0.2471	
50	0.0090		0.6542*		0.0731		0.2180	

Those cases where the baseline classifiers obtain more ranks than the OVO version in the Wilcoxon's test are indicated with a *



(a) C4.5. **(b)** RIPPER. **(c)** 3-NN. **(d)** 5-NN.

Fig. 4 Box-plots representing the distribution of the RLA results on datasets with Gaussian attribute noise

Hence, the OVO approach is also suitable considering the accuracy achieved with this type of attribute noise. The robustness results are similar between the OVO and non-OVO versions with RIPPER and 5-NN. However, in C4.5 and 3-NN, there are statistical differences in favor of OVO at the highest noise levels. The box-plots show that methods using OVO

have a better and more homogeneous behavior with all the datasets, that is, methods using OVO have a more similar behavior with noisy problems, whereas the robustness results are much more unpredictable with the non-OVO methods. Hence, the behavior of the non-OVO methods is much better with some particular datasets, whereas with others is much worse. Nevertheless, regardless of the dataset considered, OVO is more stable with respect to its robustness results.

7 OVO Decomposition and Noise Schemes: Lessons Learned

Attending to the accuracy and robustness results analyzed in the previous sections, several conclusions can be extracted about the degree of disruptiveness of the different types of noise:

1. **Class Noise.** The random class noise scheme is much more disruptive than the pairwise class noise scheme.
2. **Attribute Noise.** The random attribute noise scheme is more disruptive than the Gaussian attribute noise scheme.
3. **Class vs. Attribute Noise**. The random class noise is clearly more disruptive than the random attribute noise. The ranking of disruptiveness follows with the pairwise class noise and the Gaussian attribute noise.

Regarding the behavior of the methods using the OVO decomposition when dealing with the different noise types, the following points can be pointed out:

1. **OVO & Class Noise.** The methods using OVO have better classification accuracies at the different noise levels. The robustness of the methods using OVO is more notable with the random class noise scheme, although it also outstands with the pairwise class noise scheme on those datasets with the highest percentages of noisy examples.
2. **OVO & Attribute Noise.** The usage of the OVO approach produces better accuracies with both attribute noise schemes. The robustness results of OVO are remarkable with the random attribute noise scheme, where the differences are larger due to its higher disruptiveness.
3. **OVO & Homogeneity of the Robustness Results.** The box-plots representing the distribution of RLA results show that methods using OVO are expected to have a more similar behavior with problems suffering from noise, being generally more robust than methods not using OVO.

The following remarks can be made about how the methods with a different noise tolerance benefit from the usage of OVO:

1. **OVO & Robust Learners.** In spite of being robust learners, the performance of C4.5 and RIPPER with the more disruptive noise schemes—the random class noise scheme and the random attribute noise scheme—is much more deteriorated as the noise level increases if they do not use OVO. Therefore, the good behavior of both methods with OVO considering standard datasets [16] remains with noisy datasets. Indeed, their differences with respect to the baseline classifiers are increased.
2. **OVO & Noise-sensitive Learners.** k-NN methods also benefits from the usage of OVO. The differences of robustness between the OVO and non-OVO version, although they are generally in favor of OVO, are not so accentuated as in the case of the robust learners. With the less disruptive noise schemes—the pairwise class noise and the Gaussian attribute noise—RLA results of OVO and non-OVO are affected more similarly than in the case of the random noise schemes, where the differences increases along with the noise level.

Therefore, the methods using OVO obtain better performances than baseline methods when noise is present in the data. Methods using OVO also generally create more robust classifiers than baseline classifiers when the noise level increases, and particularly, with the more disruptive noise schemes—random class noise and random attribute noise. We can conclude that these results may be supported by the following hypotheses:

1. **Distribution of the noisy examples in the subproblems.** When decomposing the problem into several binary subproblems with OVO, the complexity of the original problem decreases. As a consequence, noisy examples are divided into each subproblem, which also decreases the effect of noise in each binary classifier, thereby having a lower influence in the final performance.
2. **Increase of the separability of the classes.** The decomposition increases the separability of the classes, since only one boundary must be established. The corruptions of noise in these regions is less notable and the classifiers are less influenced.
3. **Collecting information from different classifiers.** The aggregation of the outputs from the base classifiers produces more robust classifiers, since some fails can be corrected. Besides, if a noisy example does not belong to one of both classes involved in the learning of a classifier, the classifier will not be affected by that example, and its predictions will not be hindered.

8 Concluding remarks

This paper analyzes the suitability of the usage of the OVO decomposition when dealing with noisy training datasets in multi-class problems. A large number of noisy datasets have been created considering different types, schemes, and levels of noise, as proposed in the literature. The C4.5 and RIPPER robust learners and the noise-sensitive k-NN method have been evaluated on these datasets, with and without the usage of OVO.

The results obtained have shown that the OVO decomposition improves the baseline classifiers in terms of accuracy when data are corrupted by noise in all the noise schemes studied. The robustness results are particularly notable with the more disruptive noise schemes—the random class noise scheme and the random attribute noise scheme—where a larger amount of noisy examples and with higher corruptions are available, which produces greater differences (with statistical significance).

Three hypotheses have been introduced aiming to explain the better performance and robustness of the methods using OVO when dealing with noisy data: (1) the distribution of the noisy examples in the subproblems, (2) the increase of the separability of the classes, and (3) the possibility of collecting information from different classifiers.

As final remark, we must emphasize that one usually does not know the type and level of noise present in the data of the problem that is going to be addressed. Decomposing a problem suffering from noise with OVO has shown a better accuracy, higher robustness, and homogeneity with all the classification algorithms tested. For this reason, the usage of the OVO decomposition strategy in noisy environments can be recommended as an easy to applicate, yet powerful tool to overcome the negative effects of noise in multi-class problems.

In future works, the synergy between OVO in combination with noise preprocessing techniques will be studied in order to check its suitability to deal with noisy data. Moreover, the behavior of OVO in different noisy frameworks must be studied, that is, where both the training and the test sets are affected by noise.

Acknowledgments Supported by the Spanish Ministry of Science and Technology under Project TIN2011-28488, and also by the Regional Projects P10-TIC-06858 and P11-TIC-9704. José A. Sáez holds an FPU scholarship from the Spanish Ministry of Education and Science.

References

1. Aggarwal CC (2009) On classification and segmentation of massive audio data streams. Knowl Inf Syst 20(2):137–156
2. Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Multiple Valued Logic Soft Comput 17(2–3):255–287
3. Alcalá-Fdez J, Sánchez L, García S, del Jesus M, Ventura S, Garrell J, Otero J, Romero C, Bacardit J, Rivas V, Fernández J, Herrera F (2009) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput Fusion Found Methodol Appl 13:307–318
4. Allwein EL, Schapire RE, Singer Y, Kaelbling P (2000) Reducing multiclass to binary: a unifying approach for margin classifiers. J Mach Learn Res 1:113–141
5. Anand A, Suganthan PN (2009) Multiclass cancer classification by support vector machines with class-wise optimized genes and probability estimates. J Theor Biol 259(3):533–540
6. Anand R, Mehrotra K, Mohan CK, Ranka S (1995) Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Netw 6(1):117–124
7. Bootkrajang J, Kabán A (2011) Multi-class classification in the presence of labelling errors. In: European symposium on artificial neural networks 2011 (ESANN 2011), pp 345–350
8. Brodley CE, Friedl MA (1999) Identifying mislabeled training data. J Artif Intell Res 11:131–167
9. Cao J, Kwong S, Wang R (2012) A noise-detection based AdaBoost algorithm for mislabeled data. Pattern Recognit 45(12):4451–4465
10. Chang CC, Lin CJ (2011) LIBSVM: A library for support vector machines. ACM Trans Intell Syst Technol 2:27:1–27:27
11. Cohen WW (1995) Fast effective rule induction. In: Proceedings of the twelfth international conference on machine learning. Morgan Kaufmann Publishers, pp 115–123
12. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
13. Dietterich TG, Bakiri G (1995) Solving multiclass learning problems via error-correcting output codes. J Artifl Intell Res 2:263–286
14. Duda RO, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley, New York
15. Furnkranz J (2002) Round Robin classification
16. Galar M, Fernández A, Barrenechea E, Bustince H, Herrera F (2011) An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes. Pattern Recognit 44:1761–1776
17. Galar M, Fernández A, Barrenechea E, Bustince H, Herrera F (2012) A review on ensembles for the class imbalance problem: bagging, boosting, and hybrid-based approaches. IEEE Trans Syst Man Cybern Part C Appl Rev 42(4):463–484
18. Gamberger D, Boskovic R, Lavrac N, Groselj C (1999) Experiments with noise filtering in a medical domain. In: Proceedings of the sixteenth international conference on machine learning. Morgan Kaufmann Publishers, pp 143–151
19. Guler I, Ubeyli ED (2007) Multiclass support vector machines for EEG-signals classification. IEEE Trans Inf Technol Biomed 11(2):117–126
20. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor 11:10–18
21. Hernández MA, Stolfo SJ (1998) Real-world data is dirty: data cleansing and the merge/purge problem. Data Min Knowl Discov 2:9–37
22. Hernández-Lobato D, Hernández-Lobato JM, Dupont P (2011) Robust multi-class Gaussian process classification. In: Annual conference on neural information processing systems (NIPS 2011), pp 280–288
23. Hido S, Tsuboi Y, Kashima H, Sugiyama M, Kanamori T (2011) Statistical outlier detection using direct density ratio estimation. Knowl Inf Syst 26(2):309–336
24. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Netw 13(2):415–425
25. Huber PJ (1981) Robust statistics. Wiley, New York
26. Hüllermeier E, Vanderlooy S (2010) Combining predictions in pairwise classification: an optimal adaptive voting strategy and its relation to weighted voting. Pattern Recognit 43(1):128–142

27. Knerr S, Personnaz L, Dreyfus G (1990) A stepwise procedure for building and training a neural network. In: Fogelman Soulié F, Hérault J (eds) Neurocomputing: algorithms, architectures and applications. Springer, Berlin, pp 41–50
28. Knerr S, Personnaz L, Dreyfus G, Member S (1992) Handwritten digit recognition by neural networks with single-layer training
29. Kononenko I, Kukar M (2007) Machine learning and data mining: introduction to principles and algorithms. Horwood Publishing Limited, New York
30. Liu KH, Xu CG (2009) A genetic programming-based approach to the classification of multiclass microarray datasets. Bioinformatics 25(3):331–337
31. Liu L, Liang Q (2011) A high-performing comprehensive learning algorithm for text classification without pre-labeled training set. Knowl Inf Syst 29(3):727–738
32. Lorena A, de Carvalho A, Gama J (2008) A review on the combination of binary classifiers in multiclass problems. Artif Intell Rev 30:19–37
33. Luengo J, García S, Herrera F (2012) On the choice of the best imputation methods for missing values considering three groups of classification methods. Knowl Inf Syst 32(1):77–108
34. Mayoraz E, Moreira M (1996) On the decomposition of polychotomies into dichotomies
35. McLachlan GJ (2004) Discriminant analysis and statistical pattern recognition. Wiley, New York
36. Ménard PA, Ratté S (2011) Classifier-based acronym extraction for business documents. Knowl Inf Syst 29(2):305–334
37. Nettleton D, Orriols-Puig A, Fornells A (2010) A study of the effect of different types of noise on the precision of supervised learning techniques. Artif Intell Rev 33(4):275–306
38. Passerini A, Pontil M, Frasconi P (2004) New results on error correcting output codes of kernel machines. IEEE Trans Neural Netw 15:45–54
39. Pimenta E, Gama J (2005) A study on error correcting output codes. In: Portuguese conference on artificial intelligence EPIA 2005, pp 218–223
40. Quinlan JR (1986) Induction of decision trees. In: Machine learning, pp 81–106
41. Quinlan JR (1986) The effect of noise on concept learning. In: Machine learning: an artificial intelligence approach, chap. 6. Morgan Kaufmann Publishers, pp 149–166
42. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Francisco
43. Rifkin R, Klautau A (2004) In defense of one-vs-all classification. J Mach Learn Res 5:101–141
44. da Silva I, Adeodato P (2011) PCA and gaussian noise in MLP neural network training improve generalization in problems with small and unbalanced data sets. In: Neural networks (IJCNN), the 2011 international joint conference on, pp 2664–2669
45. Sun Y, Wong AKC, Kamel MS (2009) Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell 687–719
46. Teng CM (1999) Correcting noisy data. In: Proceedings of the sixteenth international conference on machine learning. Morgan Kaufmann Publishers, San Francisco, pp 239–248
47. Teng CM (2004) Polishing blemishes: Issues in data correction. IEEE Intell Syst 19:34–39
48. Vapnik V (1998) Statistical learning theory. Wiley, New York
49. Verikas A, Guzaitis J, Gelzinis A, Bacauskiene M (2011) A general framework for designing a fuzzy rule-based classifier. Knowl Inf Syst 29(1):203–221
50. Wang RY, Storey VC, Firth CP (1995) A framework for analysis of data quality research. IEEE Trans Knowl Data Eng 7(4):623–640
51. Wu TF, Lin CJ, Weng RC (2004) Probability estimates for multi-class classification by pairwise coupling. JMach Learn Res 5:975–1005
52. Wu X (1996) Knowledge acquisition from databases. Ablex Publishing Corp, Norwood
53. Wu X, Zhu X (2008) Mining with noise knowledge: error-aware data mining. IEEE Trans Syst Man Cybern Part A Syst Humans 38(4):917–932
54. Zhang C, Wu C, Blanzieri E, Zhou Y, Wang Y, Du W, Liang Y (2009) Methods for labeling error detection in microarrays based on the effect of data perturbation on the regression model. Bioinformatics 25(20):2708–2714
55. Zhong S, Khoshgoftaar TM, Seliya N (2004) Analyzing software measurement data with clustering techniques. IEEE Intell Syst 19(2):20–27
56. Zhu X, Wu X (2004) Class noise vs. attribute noise: a quantitative study. Artif Intell Rev 22:177–210
57. Zhu X, Wu X, Chen Q (2003) Eliminating class noise in large datasets. In: Proceeding of the twentieth international conference on machine learning, pp 920–927
58. Zhu X, Wu X, Yang Y (2004) Error detection and impact-sensitive instance ranking in noisy datasets. In: Proceedings of the nineteenth national conference on artificial intelligence. AAAI Press, pp 378–383

Author Biographies

José A. Sáez received his M.Sc. in Computer Science from the University of Granada, Granada, Spain, in 2009. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence in the University of Granada. His main research interests include noisy data in classification, discretization methods and imbalanced learning.

Mikel Galar received the M.Sc. and Ph.D. degrees in Computer Science in 2005 and 2010, both from the Public University of Navarra, Pamplona, Spain. He is currently a teaching assistant in the Department of Automatics and Computation at the Public University of Navarra. His research interests are data-minig, classification, multi-classification, ensemble learning, evolutionary algorithms and fuzzy systems.

Julián Luengo received the M.S. degree in Computer Science and the Ph.D. degree from the University of Granada, Granada, Spain, in 2006 and 2011, respectively. His research interests include machine learning and data mining, data preparation in knowledge discovery and data mining, missing values, data complexity and fuzzy systems.

Francisco Herrera received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has had more than 200 papers published in international journals. He is coauthor of the book "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases" (World Scientific, 2001). He currently acts as Editor in Chief of the international journal "Progress in Artificial Intelligence" (Springer) and serves as Area Editor of the Journal Soft Computing (area of evolutionary and bioinspired algorithms) and International Journal of Computational Intelligence Systems (area of information systems). He acts as Associated Editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Advances in Fuzzy Systems, and International Journal of Applied Metaheuristics Computing; and he serves as member of several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Knowledge and Information Systems, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, Swarm and Evolutionary Computation. He received the following honors and awards: ECCAI Fellow 2009, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", and International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010). His current research interests include computing with words and decision making, data mining, bibliometrics, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.

2. Feature weighting schemes to treat with attribute noise in nearest neighbor classifiers

The publications associated to this part are:

2.1 Statistical computation of feature weighting schemes through data estimation for nearest neighbor classifiers

Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Statistical computation of feature weighting schemes through data estimation for nearest neighbor classifiers. *Pattern Recognition* 47(12): 3941–3948

- Status: **Published**.
- Impact Factor (JCR 2014): Not available.
- Current Impact Factor of the Journal (JCR 2013): 2.584
- Subject Category: Computer Science, Artificial Intelligence. Ranking 22 / 121 (**Q1**).
- Subject Category: Engineering, Electrical & Electronic. Ranking 40 / 247 (**Q1**).

Pattern Recognition 47 (2014) 3941–3948

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Statistical computation of feature weighting schemes through data estimation for nearest neighbor classifiers

José A. Sáez [a,*], Joaquín Derrac [b], Julián Luengo [c], Francisco Herrera [a]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, Granada 18071, Spain
[b] School of Computer Science & Informatics, Cardiff University, Cardiff CF24 3AA, United Kingdom
[c] Department of Civil Engineering, LSI, University of Burgos, Burgos 09006, Spain

ARTICLE INFO

Article history:
Received 21 March 2013
Received in revised form
14 May 2014
Accepted 14 June 2014
Available online 24 June 2014

Keywords:
Feature weighting
Imputation methods
Nearest neighbor
Classification

ABSTRACT

The Nearest Neighbor rule is one of the most successful classifiers in machine learning. However, it is very sensitive to noisy, redundant and irrelevant features, which may cause its performance to deteriorate. Feature weighting methods try to overcome this problem by incorporating weights into the similarity function to increase or reduce the importance of each feature, according to how they behave in the classification task. This paper proposes a new feature weighting classifier, in which the computation of the weights is based on a novel idea combining imputation methods – used to estimate a new distribution of values for each feature based on the rest of the data – and the Kolmogorov–Smirnov nonparametric statistical test to measure the changes between the original and imputed distribution of values. This proposal is compared with classic and recent feature weighting methods. The experimental results show that our feature weighting scheme is very resilient to the choice of imputation method and is an effective way of improving the performance of the Nearest Neighbor classifier, outperforming the rest of the classifiers considered in the comparisons.

1. Introduction

The Nearest Neighbor (NN) classifier [1] is one of the most widely used methods in classification tasks due to its simplicity and good behavior in many real-world domains [2]. It is a nonparametric classifier which simply uses the full training data set to establish a classification rule, based on the most similar or nearest training instance to the query example.

The most frequently used similarity function for the NN classifier in the instance-based learning area is Euclidean distance [3]. However, redundant, irrelevant and highly correlated features may lead to erroneous similarities between the examples obtained and, therefore, to a deterioration in performance [4]. One way of overcoming this problem lies in modifying the similarity function, that is, the way in which the distances are computed. With this objective, weighting schemes can be applied in order to improve the similarity function, by introducing a weight for each of the features. High weights are assigned to those features that are helpful to classification and low weights are assigned to harmful or redundant features.

Feature Weighting methods [5] are able to enhance the NN classifier following the above procedure. By contrast to Feature Selection [6–9], the usage of weighting schemes provides the classifiers with a way of considering features *partially*, giving them some degree of importance in the classification task. This is usually preferred since weak, yet useful features may still be considered, instead of forcing the methods to either accept or completely ignore them. Many approaches using Feature Weighting have been proposed in the literature, some of which have focused on the NN classifier [10–12].

This paper proposes a novel approach for weighting features, based on the usage of imputation methods [13,14]. These are commonly employed to estimate those feature values in a data set that are unknown, formally known as missing values (MV) [15], using the rest of the data available. Therefore, imputation methods enable us to estimate a new distribution of the original data set, in which the distribution of each feature is conditioned to the rest of the features or all the data. These conditioned distributions of each feature can be compared with the original ones in order to detect the relevance of each feature, depending on the accuracy of the estimation for that feature performed by the imputation method.

The Kolmogorov–Smirnov statistic [16] may then be used to evaluate the differences between the original distribution of the features and that of the imputed ones. It is thus possible to measure how well the values of each feature can be predicted

* Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
E-mail addresses: smja@decsai.ugr.es (J.A. Sáez), jderrac@decsai.ugr.es (J. Derrac), jluengo@ubu.es (J. Luengo), herrera@decsai.ugr.es (F. Herrera).

http://dx.doi.org/10.1016/j.patcog.2014.06.012
0031-3203/

using the rest of the data. This enables us to give more importance to those features with high changes between their original and estimated value distributions – these features keep most of the structural information of the data and are not easily predictable using the rest of the data, which reduces the effect of those features that are easily predictable, and which are therefore likely to be redundant.

The study is completed with an experimentation in which our proposal is compared with several classic and recent proposals of feature weighting, considering 25 supervised classification problems taken from the Keel-Dataset repository [17]. A web page with material complementary to this paper is available at http://sci2s.ugr.es/fw-imputation including the data sets used and the performance results of each classifier.

The rest of this paper is organized as follows. Section 2 introduces imputation and feature weighting methods. In Section 3 we describe our proposal. In Section 4 we present the experimental framework, and in Section 5 we analyze the results obtained. Finally, in Section 6 we enumerate some concluding remarks.

2. Preliminaries

This section introduces our proposal's main topics: imputation in Section 2.1 and feature weighting in Section 2.2.

2.1. Imputation methods for the estimation of values

Many real-world problems contain missing values as a result of, for example, manual data entry procedures or equipment errors. This poses a severe problem for machine learning applications, since most classifiers cannot work directly with incomplete data sets. Furthermore, MVs may cause different problems in a classification task [13]: (i) loss of efficiency, (ii) complications in handling and analyzing the data and (iii) bias resulting from differences between missing and complete data. Therefore, a preprocessing stage in which the data are prepared and cleaned is usually required [18].

Imputation methods [14,19] aim to predict a value for each MV. In most cases, the features of a data set are not independent of each other. Thus, through the identification of relationships among features, MVs can be determined. An advantage of this approach is that the MV treatment is independent of the learning algorithm used. Hence, the user is able to select the most appropriate imputation depending on the learning approach considered [13].

One of the simplest imputation methods is based on the NN rule: *k-NN Imputation* (KNNI). C4.5 or CN2 usually benefit from its usage [19]. Other approaches try to improve or complement its performance over various domains, for example, in [20] a *Support Vector Machine* (SVM) was used to fill in MVs (SVMI).

Other works are mostly focused on studying the behavior of several imputation methods in a specific scenario. For example, in [21], the authors induced MVs in several data sets. The prediction value – that is, the similarity of the imputed value to the originally removed one – of several imputation methods, such as *Regularized Expectation-Maximization* [22] or *Concept Most Common* (CMC) [23], and the accuracy obtained by several classifiers were studied. From the results, the authors stated that better prediction results do not imply better classification results. A similar approach was adopted in [14], in which the behavior of classifiers belonging to different paradigms, such as decision trees or instance-based learning methods, was studied over data sets with different levels of MVs.

All the aforementioned works have shown that imputation methods work properly when estimating missing values from the rest of the available data. They are therefore also suitable for use in our proposal.

2.2. Feature weighting in nearest neighbor classification

Data preparation [18,24] provides a number of ways to improve the performance of the NN classifier, such as Prototype Selection [25] or Feature Selection [6–9]. A different, yet powerful approach is Feature Weighting [5].

Feature Weighting methods can be included as a part of another type of more general methods: those based on adaptive distance measures [26–29]. These techniques try to learn distance metrics from the labeled examples of a problem in order to improve the classification performance. A reference work within this topic is, for example, that of Weinberger and Saul [26], in which the Mahanalobis distance metric is learned for k-nearest neighbor classification by semidefinite programming. The metric is trained in order that the k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. On the other hand, the approach of [29] proposes a framework in which the metrics are parameterized by pairs of identical convolutional neural nets. Other works [27,28] consider schemes for locally adaptive distance metrics that change across the input space to overcome the bias problem of NN when working in high dimensions. In [27] a local linear discriminant analysis is used to compute neighborhoods, whereas in [28] a technique that computes a locally flexible metric by means of support vector machines is proposed.

The main objective of Feature Weighting methods is to reduce the sensitivity of the NN rule to redundant, irrelevant or noisy features. This is achieved by modifying its similarity function [4] with the inclusion of weights. These weights can be regarded as a measure of how useful a feature is with respect to the final classification task. The higher a weight is, the more influence the associated feature will have in the decision rule used to compute the classification of a given example. Therefore, an adequate scheme of weights could be used to highlight the best features of the domain of the problem, diminishing the impact of redundant, irrelevant and noisy ones. Thus, the accuracy of the classifier could be greatly improved if a proper selection of weights is made.

In the case of the NN classifier, most of the techniques developed to include Feature Weighting schemes have been focused on incorporating the weights in the distance measure, mainly to Euclidean distance (see Eq. (1), where X and Y are two instances and M is the number of features that describes them). In spite of its simplicity, the usage of Euclidean distance has been preferred in many research approaches, since it is easy to optimize and shows a good discriminative power in most classification tasks. In fact, it is the most commonly used similarity measure in the instance based learning field [3].

$$d(X,Y)=\sqrt{\sum_{i=0}^{M}(x_i-y_i)^2} \tag{1}$$

Feature Weighting methods often extend this definition through the inclusion of weights associated with each feature (W_i, usually $W_i \in [0,1]$). These modify the way in which the distance measure is computed (Eq. (2)), increasing the relevance (the squared difference between feature's values) of those features with greater weights associated with them (near to 1.0).

$$d_w(X,Y)=\sqrt{\sum_{i=0}^{M}W_i\cdot(x_i-y_i)^2} \tag{2}$$

The application of this technique to the NN classifier has been widely addressed. To the best of our knowledge, the most complete study undertaken to this end can be found in [5], in

which a review of several Feature Weighting methods for Lazy Learning algorithms [30] is presented (with most of them applied to improve the performance of the NN rule). In this review, Feature Weighting techniques are categorized by several dimensions, regarding the weight learning bias, the weight space (binary or continuous), the representation of features employed, their generality and their degree of employment of domain specific knowledge.

A wide range of classical Feature Weighting techniques are available in the literature, both classical (see [5] for a complete review) and recent [10,12]. The most well known compose the family of *Relief-based* algorithms.

The Relief algorithm [31] (which was originally a Feature Selection method [6]) has been widely studied and modified, producing several interesting variations of the original approach. Some of them [32,11] are based on ReliefF [33], which is the first adaptation of Relief as a Feature Weighting approach.

In addition to these approaches, Feature Weighting methods are also very useful when considered as a part of larger supervised learning schemes. In these approaches, Feature Weighting can be regarded as an improved version of Feature Selection (in fact, Feature Selection is a binary version of Feature Weighting, defining a weight of 1 if a feature is selected, or 0 if it is discarded). Again, if the weights scheme is properly chosen, Feature Weighting can play a decisive role in enhancing the performance of the NN classifier in these techniques [34].

3. A weighting algorithm based on feature differences after values imputation

This section describes the weighting method proposed, which is based on three main steps (see Fig. 1):

1. *Imputation of the data set* (Section 3.1): In this phase, an imputation method is used to build a new estimated data set DS' from the original one DS.
2. *Computation of weights* (Section 3.2): The distribution of the values of each feature f_i of DS and the corresponding estimated feature f'_i of DS' are compared using the Kolmogorov–Smirnov statistical test. This enables the extraction of the D^i_n statistic for each feature f_i.
3. *Construction of the classifier* (Section 3.3): Once the D^i_n statistic is computed for each feature i, the NN classifier is used, incorporating a modified version of Euclidean distance. This version is based on a weighting scheme derived from the D^i_n statistics.

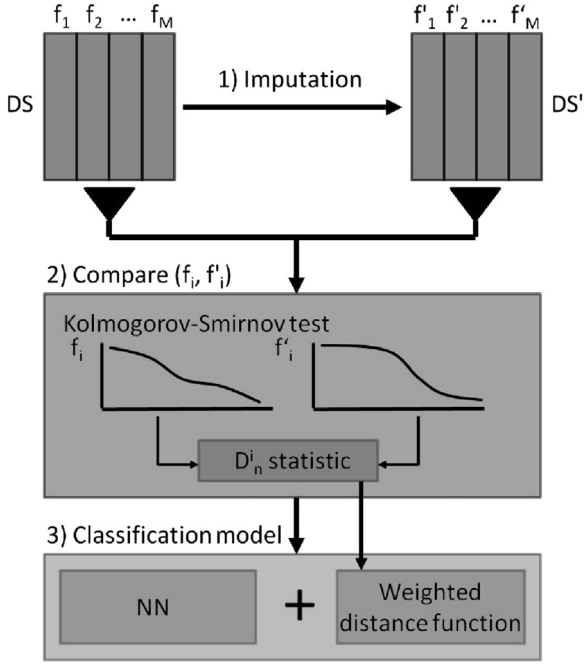


Fig. 1. Feature weighting method proposed.

The following sections describe each of these steps in depth. Section 3.1 is devoted to the imputation phase, whereas Section 3.2 describes the computation of the weights. Finally, Section 3.3 characterizes the classification model.

3.1. Imputation of the data set

The first step consists of creating a whole new estimated data set DS' from the original one DS. In order to do this, an imputation method is used (in this paper we will consider KNNI [19], CMC [23] and SVMI [20], although other imputation methods may be chosen). If the original data set DS is composed of the features $f_1, f_2, \ldots, f_M$, the imputed data set DS' will be formed by the features $f'_1, f'_2, \ldots, f'_M$ whose values are obtained by the imputation method.

The procedure to obtain DS' from DS is represented in Algorithm 1. This is based on assuming iteratively that each feature value of each example of the data set DS, that is, $e(f_i)$, is missing (lines 2–5). Then, the imputation method IM is used to predict a new value for that feature value (line 6). The new data set DS' is obtained by repeating this process for each feature value, until the whole data set has been processed. Carrying out this process, it is possible to estimate a distribution of values for each feature, which is conditioned to the rest of the features or the totality of the data. The new data set DS' will contain these conditioned distributions for each feature. This will allow us to check those features that are more difficult to predict with the rest of the features/data and contain the structural information of the data set, making them more important to the classification task.

Algorithm 1. Pseudocode of the first step of the method: imputation of the dataset.

Input: original dataset DS, imputation method IM.
Output: estimated dataset DS'.

```
1  Set DS' = ∅;
2  for each example e ∈ DS do
3  |  e' = null;
4  |  for each feature f_i do
5  |  |  Suppose e(f_i) as missing;
6  |  |  e'(f'_i) ⟵ Estimate the value for e(f_i) using IM over DS;
7  |  end
8  |  DS' ⟵ DS' ∪ {e'}
9  end
```

3.2. Computation of weights using the Kolmogorov–Smirnov test

The next step consists of measuring which features are most changed after the application of the imputation method. Given the nature of the imputation techniques, some features are expected to remain unchanged (or to present only small changes in their values' distribution) whereas other features may present a higher level of disruption when their imputed values are compared with the original ones. The Kolmogorov–Smirnov test [16] provides a way of measuring these changes. This test works by computing a

statistic D_n, which can be regarded as a measure of how different two samples are.

The test is a nonparametric procedure for testing the equality of two continuous, one-dimensional probability distributions. It quantifies a distance between the empirical distribution functions of two samples. The null distribution of its statistic, D_n, is computed under the null hypothesis that the samples are drawn from the same distribution.

The main advantage of using the D_n statistic (computed in the Kolmogorov–Smirnov test) instead of other simpler statistics such as the variance is that, for our purpose, which consists of measuring the similarity of two given distributions, shape measures used to compare the two distributions are more appropriate than other types of measures (such as dispersion measures in the case of the variance). Thus, when comparing two distributions, the changes in the variance do not provide enough information on how similar the two distributions are. Variances are only a measure of how the values of an attribute are concentrated around the mean, and is just one of the many factors that may be changed by distribution. However, the D_n statistic contains the structural information that describes how the distribution has changed. This can be done by identifying where the higher or lower concentrations of values are (in the lowest values of the distribution, in the highest values, if there are several intervals with a higher concentration of values, etc.). Thus, the D_n statistic is therefore much more representative than a simple comparison between the variances of the two distributions.

On the other hand, two samples of values with the same variance do not necessarily imply that both follow the same distribution (the same shape), or even that they have similar distributions. A simple example in which the variance does not work properly can be seen in regard to the property that makes the variance invariant to changes in the origin. Suppose two attributes: A (real distribution of values of an attribute) and A' (the distribution with the estimated values of that attribute). Assume that $A' = A + C$, where C is a constant. Then, $variance(A) = variance(A')$. The two samples have the same variance, even though they obviously come from two different distributions and this fact is not detected using the variance. This problem is avoided if the D_n statistic is employed.

Given two samples, X and Y, and their empirical distribution functions F_X and F_Y

$$F_X(x) = \frac{1}{n}\sum_{i=1}^{n} I_{X_i \le x}, \quad F_Y(x) = \frac{1}{n}\sum_{i=1}^{n} I_{Y_i \le x} \tag{3}$$

(where $I_{X_i \le x}$ is the indicator function, equal to 1 if $X_i \le x$ and equal to 0 otherwise) the Kolmogorov–Smirnov statistic is

$$D_n = \sup_x |F_X - F_Y| \tag{4}$$

Table 1 shows two toy samples (where two distributions of values $X = \{X_1, \ldots, X_n\}$ and $Y = \{Y_1, \ldots, Y_n\}$ with $n = 5$ are given), whereas Table 2 shows an example of the computation of the Kolmogorov–Smirnov statistic from them.

In the approach of this paper, the D_n statistic provides a valuable way of estimating the degree of change undergone by a feature through the imputation process. By computing the D_n statistic associated with the differences between both samples of the feature (original and imputed), it is possible to measure the greater degree of difference between the expected distribution of both samples. Hence, the greater D_n value obtained, the more different the imputed version of the feature distribution will be (when compared with the original one).

Table 1
Two toy samples of size $n = 5$.

$X = \{X_1 = 0.01, X_2 = 0.11, X_3 = 0.12, X_4 = 0.22, X_5 = 0.85\}$
$Y = \{Y_1 = 0.09, Y_2 = 0.41, Y_3 = 0.65, Y_4 = 0.73, Y_5 = 0.91\}$

Table 2
Example of the computation of the Kolmogorov–Smirnov statistic.

x	F_X	F_Y	$\lvert F_X - F_Y \rvert$	$\sup_x \lvert F_X - F_Y \rvert$
0	0	0	0	0
0.01	0.2	0	0.2	0.2
0.09	0.2	0.2	0	0.2
0.11	0.2	0.2	0.2	0.2
0.12	0.6	0.2	0.4	0.4
0.22	0.8	0.2	0.6	0.6
0.41	0.8	0.4	0.4	0.6
0.65	0.8	0.6	0.2	0.6
0.73	0.8	0.8	0	0.6
0.85	1.0	0.8	0.2	0.6
0.91	1.0	1.0	0	0.6
D_n	0.6			

The D_n statistic can be easily transformed into a weight. Since $D_n \in [0, 1]$, features with a lower value of D_n (near to 0.0) it will have little influence on the computation of the similarity function of the NN rule, whereas features with a higher value of D_n (near to 1.0) will be the most influential when computing the distance between two examples. Defining the statistical D_n^i for the feature i as

$$D_n^i = \text{Kolmogorov–Smirnov}(f_i, f_i') \quad \forall i, f_i \in \mathcal{A}, f_i' \in \mathcal{A}' \tag{5}$$

(where $\mathcal{A}$ denotes the set of features of the original data set DS and $\mathcal{A}'$ denotes the set of features imputed in DS'), then the weights $W_i \in [0, 1]$ computed for a feature $f_i \in \mathcal{A}$ are

$$W_i = D_n^i \Big/ \sum_{j=1}^{M} D_n^j \tag{6}$$

3.3. Final classification model

The final classifier considers NN with the weighted Euclidean distance (Eq. (2)) and the weights computed throughout the Kolmogorov–Smirnov statistic (Eq. (6)).

Considering weights computed from the D_n statistic, we aim to highlight the effect that changing features have on the computation of the distance. These features, with a larger associated D_n value, will be those poorly estimated by the imputation method (whose sample distribution differs greatly if the original and imputed versions are compared). They are preferred since they keep most of the structural information of the data, and are the key features describing the data set (they cannot be properly estimated using the rest of the data).

By contrast, features with a small D_n value will be those whose sample distribution has not been changed after the application of the imputation method. Since these features are easily estimated when the rest of the data is available (the imputation method can recover their values properly), they are not preferred in the final computation of the distance, and thus a lower weight is assigned to them.

4. Experimental framework

This section presents the framework of the experimental study conducted. The imputation methods considered in the previous section are presented in Section 4.1, whereas Section 4.2 is devoted to the feature weighting methods used. Section 4.3

describes the data sets employed. Finally, Section 4.4 describes the methodology followed to analyze the results.

4.1. Imputation methods

The proposal described in this paper allows us to include any standard imputation method. For the sake of generality, we have chosen to test the behavior using three different imputation techniques, well-known representatives of the field [13,19]:

1. *KNNI* [19]: Based on the *k*-NN algorithm, every time an MV is found in a current example, KNNI computes the *k* nearest neighbors and their average value is imputed. KNNI also uses the Euclidean distance as a similarity function.
2. *CMC* [23]: This method replaces the MVs by the average of all the values of the corresponding feature considering only the examples with the same class as the example to be imputed.
3. *SVMI* [20]: This is an SVM regression-based algorithm developed to fill in MVs. It works by firstly selecting the examples in which there are no missing feature values. In the next step, the method sets one of the input features, some of the values of which are missing, as the decision feature, and the decision feature as the input feature. Finally, an SVM for regression is used to predict the new decision feature.

The parameter setup used for their execution is presented in Table 3. Each imputation method considered will lead to a different feature weighting classifier. Throughout the study, we will denote them as FW-KNNI, FW-CMC and FW-SVMI.

4.2. Feature weighting methods for NN

In order to check the performance of the approach proposed, the following feature weighting algorithms for nearest neighbor classification as comparison methods have been chosen:

1. *NN* [1]: The NN rule is used as a baseline limit of performance which most of the methods should supersede.
2. *CW* [10]: A gradient descent based algorithm developed with the aim of minimizing a performance index that is an approximation of the leave one out error over the training set. In this approach, weights are obtained for each combination of feature and class, that is, the set of weights is different depending on the class of each training example.
3. *MI* [5]: Mutual Information (MI) between features can be used successfully as a weighting factor for NN based algorithms. This method was marked as the best preset FW method in [5].
4. *ReliefF* [33]: The first Relief-based method adapted to perform the FW process. By contrast to the original Relief method, weights computed in ReliefF are not binarized to 0,1. Instead, they are used as final weights for the NN classifier. This method was noted as the best performance-based FW method in [5].
5. *IRelief* [11]: A multiclass, iterative extension of Relief. The objective function of the iterative process aims at reducing the distances between each example and its nearest hit (nearest training example of the same class) and increasing the distances between each example and its nearest enemy (nearest training example of another class).

Table 3
Parameter specification for the imputation methods.

Algorithm	Ref.	Parameters
KNNI	[19]	*k* value: 10
CMC	[23]	It has no parameters to be fixed
SVMI	[20]	Kernel type: RBF, C: 1.0, RBF-γ: 1.0

Table 4
Parameter specification for the classifiers of the study.

Algorithm	Ref.	Parameters
NN	[1]	It has no parameters to be fixed
CW	[10]	β: Best in [0.125, 128], μ: Best in [0.001, 0.1], ϵ: 0.001, Iterations: 1000
MI	[5]	It has no parameters to be fixed
ReliefF	[33]	*K* value: Best in [1, 20]
IRelief	[11]	Maximum iterations: 100, ϵ: 0.00001, σ: Best in [0.001, 1000]

Table 5
Data sets employed in the experimentation.

Data set	#EXA	#FEA	#CLA	Data set	#EXA	#FEA	#CLA
banana	5300	2	2	pima	768	8	2
bands	365	19	2	satimage	6435	36	7
bupa	345	6	2	sonar	208	60	2
dermatology	358	34	6	tae	151	5	3
ecoli	336	7	8	texture	5500	40	11
heart	270	13	2	vowel	990	13	11
hepatitis	80	19	2	wdbc	569	30	2
ionosphere	351	33	2	wine	178	13	3
iris	150	4	3	wq-red	1599	11	11
led7digit	500	7	10	wq-white	4898	11	11
mov-libras	360	90	15	wisconsin	683	9	2
newthyroid	215	5	3	yeast	1484	8	10
phoneme	5404	5	2				

Table 4 summarizes the parameter setup used for the feature weighting methods in the experimental study, which was used in the reference in which the methods were originally described.

4.3. Data sets

The experimentation considers 25 data sets from the KEEL-Dataset repository [17]. They are described in Table 5, where #EXA refers to the number of examples, #FEA to the number of numeric features and #CLA to the number of classes.

For data sets containing missing values (such as *bands* or *dermatology*), the examples with missing values were removed from the data sets before their usage and thus all the attribute values of the data sets considered are known. In this way, the percentage of missing values of each data set does not influence the results or conclusions obtained and it does not harm the methods that are not specially designed to deal with them. Therefore, the only missing values considered in this paper are those assumed during the execution of Algorithm 1 in order to build the new estimated distribution of values.

4.4. Methodology of analysis

The performance estimation of each classifier on each data set is obtained by means of 3 runs of a 10-fold *distribution optimally balanced stratified cross-validation* (DOB-SCV) [35], averaging its test accuracy results. The usage of this partitioning reduces the negative effects of both prior probability and covariate shifts [36] when classifier performance is estimated with cross-validation schemes. The results with the standard cross-validation can be found on the web page of this paper.

Statistical comparisons of the data sets considered will be also performed. Wilcoxon's test [37] will be applied to study the differences among the proposals of this paper and also between

Table 6
Test accuracy results.

Data set	CW	MI	ReliefF	IRelief	NN	FW-KNNI	FW-CMC	FW-SVMI
banana	61.59	59.51	87.7	**88.00**	87.87	87.91	87.79	87.60
bands	**72.31**	45.27	65.99	36.97	72.04	72.04	69.02	69.85
bupa	62.36	42.02	59.09	55.35	62.36	**64.11**	64.09	63.50
dermatology	95.18	**97.19**	96.9	93.22	94.9	94.62	95.75	95.21
ecoli	80.09	78.88	70.69	76.88	80.09	79.77	80.09	**80.67**
heart	76.3	**82.96**	78.89	78.89	74.81	73.7	75.19	75.19
hepatitis	82.94	81.27	80.26	**85.65**	82.94	85.62	81.83	82.94
ionosphere	85.96	87.4	90.26	**91.11**	87.11	88.55	87.09	87.67
iris	**96.00**	83.33	95.33	94.67	95.33	95.33	**96.00**	94.00
led7digit	44.88	51.55	51.6	51.62	44.88	**52.45**	51.55	52.03
mov-libras	82.81	69.85	25.6	84.1	82.81	85.73	**85.95**	85.51
newthyroid	97.19	94.35	**98.59**	95.32	97.19	96.26	97.19	96.71
phoneme	90.43	76.85	68.24	72.63	90.41	91.04	91.06	**91.08**
pima	70.45	69.13	63.02	66.28	70.45	**70.97**	70.71	70.19
satimage	**90.94**	90.46	90.89	90.64	90.88	90.71	90.71	**90.94**
sonar	86.52	73.13	84.99	**87.52**	86.52	86.97	86.55	86.02
tae	42.07	31.81	28.63	**69.42**	42.07	65.55	65.55	65.55
texture	**99.15**	98	99.09	98.8	**99.15**	**99.15**	99.07	**99.15**
vowel	99.39	80.51	98.89	99.19	99.39	**99.49**	99.39	99.39
wdbc	95.97	**96.14**	93.46	95.26	95.96	94.91	95.8	95.97
wine	95.58	97.84	**98.36**	97.25	95.58	95.58	96.14	96.69
wq-red	53.72	48.96	66	63.76	53.66	**66.19**	65.74	65.55
wq-white	50.19	54.06	51.1	20.89	50.17	66.67	**67.36**	67.04
wisconsin	95.61	96.49	**96.78**	96.49	96.04	96.04	95.75	96.05
yeast	51.68	37.95	43.03	49.3	51.55	53.71	**54.05**	53.57
Average	78.37	73	75.34	77.57	79.37	**82.12**	81.98	81.92
Best result (out of 25)	4	3	3	5	1	6	4	4

each of these proposals and NN using the Euclidean distance. Regarding the comparison among feature weighting methods, the results of the Friedman Aligned test [38] and the Finner procedure [39] will be computed. Comparisons with other tests, such as the Holm test [40], may be found on the web page of this paper. More information about these statistical procedures can be found at http://sci2s.ugr.es/sicidm/.

5. Analysis of results

This section presents the analysis of the results obtained. Table 6 shows the test accuracy obtained by each classifier on each data set. The best results for each data set are highlighted in bold. From this table, several remarks can be made:

- The method obtaining the best results in most single data sets is FW-KNNI (in 6 of the 25 data sets). It is followed by IRelief (5 data sets), FW-CMC, FW-SVMI and CW (4 data sets), MI and ReliefF (3 data sets) and NN (1 data set).
- Even though IReliefF or CW obtain the best results in a certain number of data sets – 5 and 4 respectively –, they show a variable performance for different problems. For instance, in data sets such as *banana* and *tae*, CW's results are very far from the results obtained by the best performing methods in these data sets. The same occurs for IRelief – in *bands*, *phoneme* and *wq-white* – whereas this issue is not very remarkable with regard to any of the other proposals of this paper. This fact shows that our methods are generally more robust than those of the rest of the algorithms included in the comparison.
- Regardless of the imputation method selected, our approaches usually obtain results close to those of the best performing method in each data set. Moreover, all of them obtain better accuracy on average than the comparison methods over the 25 problems.

Table 7
Wilcoxon's comparison of the proposed methods.

	FW-KNNI			FW-CMC			FW-SVMI		
Method	R+	R−	*p*-value	R+	R−	*p*-value	R+	R−	*p*-value
FW-KNNI	–	–	–	159	166	1.0000	191.5	133.5	0.4273
FW-CMC	166	159	0.9140	–	–	–	154	146	0.8970
FW-SVMI	133.5	191.5	1.0000	146	154	1.0000	–	–	–

To add depth to the analysis of the results, several statistical comparisons are performed below, studying the differences among the proposals of this paper, their comparison with NN and also with the rest of the feature weighting methods.

Comparison among the feature weighting methods based on imputation: The results of the three proposals of this paper (FW-KNNI, FW-CMC and FW-SVMI) shown in Table 6 are quite similar. In order to study whether there are statistical differences among them, Wilcoxon's test has been performed – see Table 7. In this table, the classifier of each row is established as the control method for the statistical test and its ranks (R+), the ranks in favor of the method of the column (R−) and the *p*-value associated are shown. From the high *p*-values obtained in these comparisons, one can conclude that statistical differences among the three proposals do not exist. This fact shows the robustness of the proposal independent of the imputation method chosen. Therefore, the good behavior of the approach is due to the strategy for obtaining the weights, which combines imputation methods and the Kolmogorov–Smirnov test; the concrete imputation method employed does not influence the results so much.

Comparison with NN: Table 8 shows the results of applying Wilcoxon's test to each of the proposals performed and NN. As the table shows, every proposal is statistically better than NN due to the low *p*-values obtained – all are lower than 0.05. This shows

Table 8
Wilcoxon's comparison of the proposed methods with NN.

Methods	R+	R−	p-value
FW-KNNI vs NN	229	71	0.0229
FW-CMC vs NN	224.5	75.5	0.0327
FW-SVMI vs NN	228.5	71.5	0.0239

Table 9
Statistical comparison among feature weighting methods.

	FW-KNNI		FW-CMC		FW-SVMI	
Method	Rank	p_{Finn}	Rank	p_{Finn}	Rank	p_{Finn}
Imputation	42.90	–	43.78	–	43.74	–
CW	60.36	0.0884	60.08	0.1117	59.94	0.1139
MI	84.30	0.0002	83.64	0.0004	83.90	0.0004
ReliefF	65.24	0.0576	65.26	0.0708	64.82	0.0778
IRelief	62.20	0.0787	62.24	0.0943	62.60	0.0866
p_{FA}	0.0003		0.0003		0.0003	

that the application of our approach to feature weighting improves the performance of the NN classifier significantly, regardless of the specific imputation method chosen.

Comparison among feature weighting methods: Table 9 presents the statistical comparison performed for each proposal (FW-KNNI, FW-CMC and FW-SVMI). Each proposal is independently compared with the rest of the feature weighting methods since we have already confirmed that there are no significant differences among our three approaches (see Table 7). The ranks obtained by the Friedman Aligned procedure (Rank column), which represent the effectiveness associated with each algorithm, and the p-value related to the significance of the differences found by this test (p_{FA} row) are shown. The p_{Finn} column shows the adjusted p-value computed by the Finner test.

Looking at Table 9, we can observe that:

- The average ranks obtained by our proposals are the best (the lowest) and they are notably differentiated from the ranks of the rest of the methods.
- These are followed by CW, IRelief and ReliefF with very close ranks among them. MI obtains the highest rank.
- The p-values of the Friedman Aligned test are very low in every case, meaning that the differences found among the methods are very significant.
- The p-values obtained with the Finner procedure when comparing FW-KNNI, FW-CMC and FW-SVMI with the comparison algorithms are very low. The differences found are always significant (lower than 0.1), except in the case of FW-CMC and FW-SVMI with CW, in which the p-value obtained is still very low.

From the results of Tables 6– 9, it is possible to conclude that the proposals presented in this paper perform better than the rest of the feature weighting methods considered. They are also able to improve the performance of the NN classifier. Even though they do not obtain the best results in a large number of single data sets, the statistical tests illustrate the improvement of performance achieved by our approaches, showing a great robustness and a good behavior in most of the data sets. The comparison among our three proposals does not show statistical differences, suggesting that the strategy for obtaining the weights performs accurately independent of the concrete imputation method employed.

6. Conclusions

In this paper we have proposed a new scheme for feature weighting developed to improve the performance of the NN classifier, in which the weights are computed by combining imputation methods and the Kolmogorov–Smirnov statistic. From the experimental results it is possible to conclude that our feature weighting scheme is not very sensitive to the selection of the imputation method, since the results obtained in every case are quite similar regardless of the specific imputation technique chosen, and statistical differences among them have not been found.

The results obtained show that all our approaches enhance the performance of NN to a greater degree than the rest of the feature weighting methods analyzed. They also show a robust behavior in several domains, in contrast to the rest of the classifiers, which demonstrate a variable performance when different data sets are considered. The statistical analysis performed confirms our conclusions. The results with standard cross-validation provide similar conclusions to those shown here (see the results at http://sci2s.ugr.es/fw-imputation).

Conflict of interest

None declared.

Acknowledgments

Supported by the Projects TIN2011-28488, P10-TIC-06858 and P11-TIC-9704. J.A. Sáez holds an FPU grant from the Spanish Ministry of Education and Science (reference AP2009-2930).

References

[1] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1967) 21–27.
[2] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2007) 1–37.
[3] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1991) 37–66.
[4] Y. Chen, E.K. Garcia, M.R. Gupta, A. Rahimi, L. Cazzanti, Similarity-based classification: concepts and algorithms, J. Mach. Learn. Res. 10 (2009) 747–776.
[5] D. Wettschereck, D.W. Aha, T. Mohri, A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, Artif. Intell. Rev. 11 (1997) 273–314.
[6] H. Liu, H. Motoda (Eds.), Computational methods of Feature Selection, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC, London, 2007.
[7] B. Xue, M. Zhang, W. Browne, Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms, Appl. Soft Comput. J. 18 (2014) 261–276.
[8] J. Kersten, Simultaneous feature selection and Gaussian mixture model estimation for supervised classification problems, Pattern Recognit. 47 (8) (2014) 2582–2595.
[9] D. Liu, H. Qian, G. Dai, Z. Zhang, An iterative SVM approach to feature selection and classification in high-dimensional datasets, Pattern Recognit. 46 (9) (2013) 2531–2537.
[10] R. Paredes, E. Vidal, Learning weighted metrics to minimize nearest-neighbor classification error, IEEE Trans. Pattern Anal. Mach. Intell. 28 (7) (2006) 1100–1110.
[11] Y. Sun, Iterative relief for feature weighting: algorithms, theories, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 29 (6) (2007) 1035–1051.
[12] F. Fernández, P. Isasi, Local feature weighting in nearest prototype classification, IEEE Trans. Neural Netw. 19 (1) (2008) 40–53.
[13] J. Luengo, S. García, F. Herrera, On the choice of the best imputation methods for missing values considering three groups of classification methods, Knowl. Inf. Syst. 32 (1) (2012) 77–108.
[14] A. Farhangfar, L. Kurgan, J. Dy, Impact of imputation of missing values on classification error for discrete data, Pattern Recognit. 41 (12) (2008) 3692–3705.
[15] M. Juhola, J. Laurikkala, Missing values: how many can they be to preserve classification reliability? Artif. Intell. Rev. 40 (3) (2013) 231–245.

[16] N.V. Smirnov, Estimate of deviation between empirical distribution functions in two independent samples (in Russian), Bull. Mosc. Univ. 2 (1939) 3–16.
[17] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2–3).
[18] D. Pyle, Data Preparation for Data Mining, Morgan Kaufmann, Los Altos, 1999.
[19] G.E.A.P.A. Batista, M.C. Monard, An analysis of four missing data treatment methods for supervised learning, Appl. Artif. Intell. 17 (5–6) (2003) 519–533.
[20] H. Feng, G. Chen, C. Yin, B. Yang, Y. Chen, A SVM regression based approach to filling in missing values, in: Lecture Notes in Computer Science, vol. 3683, Springer, 2005.
[21] E.R. Hruschka Jr., E.R. Hruschka, N.F.F. Ebecken, Bayesian networks for imputation in classification problems, J. Intell. Inf. Syst. 29 (3) (2007) 231–252.
[22] T. Schneider, Analysis of incomplete climate data: estimation of mean values and covariance matrices and imputation of missing values, J. Clim. 14 (5) (2001) 853–871.
[23] J.W. Grzymala-Busse, L.K. Goodwin, W.J. Grzymala-Busse, X. Zheng, Handling missing attribute values in preterm Birth data sets, in: Lecture Notes in Computer Science, vol. 3642, Springer, 2005.
[24] J. Lara, D. Lizcano, M. Martínez, J. Pazos, Data preparation for KDD through automatic reasoning based on description logic, Inf. Syst. 44 (2014) 54–72.
[25] S. García, J. Derrac, J.R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435.
[26] K. Weinberger, L. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (2009) 207–244.
[27] T. Hastie, R. Tibshirani, Discriminant adaptive nearest neighbor classification, IEEE Trans. Pattern Anal. Mach. Intell. 18 (6) (1996) 607–616.
[28] C. Domeniconi, D. Gunopulos, J. Peng, Large margin nearest neighbor classifiers, IEEE Trans. Neural Netw. 16 (4) (2005) 899–909.
[29] S. Chopra, R. Hadsell, Y. LeCun, Learning a similiarty metric discriminatively with application to face verification, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR-05), San Diego, CA, vol. 1, 2005, pp. 539–546.
[30] D.W. Aha (Ed.), Lazy Learning, Springer, New York, 1997.
[31] K. Kira, L.A. Rendell, A practical approach to feature selection, in: Proceedings of the Ninth International Conference on Machine Learning, Aberdeen, Scotland, Morgan Kaufmann, 1992, pp. 249–256.
[32] M.R. Sikonja, I. Kononenko, Theoretical and empirical analysis of Relief and ReliefF, Mach. Learn. 53 (1–2) (2003) 23–69.
[33] I. Kononenko, Estimating attributes: analysis and extensions of RELIEF, in: Proceedings of the 1994 European Conference on Machine Learning, Catania, Italy, Springer Verlag, 1994, pp. 171–182.
[34] J. Derrac, I. Triguero, S. García, F. Herrera, Integrating instance selection, instance weighting and feature weighting for nearest neighbor classifiers by co-evolutionary algorithms, IEEE Trans. Syst. Man, Cybern. Part B: Cybern. 42 (5) (2012) 1383–1397.
[35] J.G. Moreno-Torres, J.A. Sáez, F. Herrera, Study on the impact of partition-induced dataset shift on k-fold cross-validation, IEEE Trans. Neural Netw. Learn. Syst. 23 (8) (2012) 1304–1312.
[36] J.G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N.V. Chawla, F. Herrera, A unifying view on dataset shift in classification, Pattern Recognit. 45 (1) (2012) 521–530.
[37] F. Wilcoxon, Individual comparisons by ranking methods, Biometr. Bull. 1 (6) (1945) 80–83.
[38] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (2010) 2044–2064.
[39] H. Finner, On a monotonicity problem in step-down multiple test procedures, J. Am. Stat. Assoc. 88 (1993) 920–923.
[40] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (1979) 65–70.

José A. Sáez received his M.Sc. in Computer Science from the University of Granada (Granada, Spain) in 2009. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence in the University of Granada. His main research interests include noisy data in classification, discretization methods and imbalanced learning.

Joaquín Derrac received the M.Sc. and Ph.D. in computer science from the University of Granada, Granada, Spain, in 2008 and 2013, respectively. His research interests include data mining, data reduction, nearest neighbor classifiers, statistical inference and evolutionary algorithms.

Julián Luengo received the M.S. in computer science and the Ph.D. degree from the University of Granada, Granada, Spain, in 2006 and 2011, respectively. His research interests include machine learning and data mining, data preparation in knowledge discovery and data mining, missing values, data complexity and fuzzy systems.

Francisco Herrera received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain.

He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has been the supervisor of 30 Ph.D. students. He has published more than 260 papers in international journals. He is coauthor of the book "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases" (World Scientific, 2001).

He currently acts as Editor in Chief of the international journals "Information Fusion" (Elsevier) and "Progress in Artificial Intelligence (Springer). He acts as an area editor of the International Journal of Computational Intelligence Systems and associated editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Knowledge and Information Systems, Advances in Fuzzy Systems, and International Journal of Applied Metaheuristics Computing; and he serves as a member of several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Information Fusion, Knowledge-Based Systems, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, and Swarm and Evolutionary Computation.

He received the following honors and awards: ECCAI Fellow 2009, IFSA Fellow 2013, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 Paper Award (bestowed in 2011), 2011 Lotfi A. Zadeh Prize Best paper Award of the International Fuzzy Systems Association, and 2013 AEPIA Award to a scientific career in Artificial Intelligence (September 2013).

His current research interests include computing with words and decision making, bibliometrics, data mining, data preparation, instance selection and generation, imperfect data, fuzzy rule based systems, genetic fuzzy systems, imbalanced classification, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms, biometrics, cloud computing and big data.

2.2 Improving the behavior of the nearest neighbor classifier against noisy data with feature weighting schemes

Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Improving the behavior of the nearest neighbor classifier against noisy data with feature weighting schemes. In *Hybrid Artificial Intelligence Systems*, volumen 8480 of *Lecture Notes in Computer Science*, pp. 597–606. Springer International Publishing

- Status: **Published**.

Improving the Behavior of the Nearest Neighbor Classifier against Noisy Data with Feature Weighting Schemes

José A. Sáez[1], Joaquín Derrac[2], Julián Luengo[3], and Francisco Herrera[1]

[1] Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, Granada, Spain, 18071
{smja,herrera}@decsai.ugr.es
[2] School of Computer Science & Informatics, Cardiff University, Cardiff CF24 3AA, United Kingdom
jderrac@decsai.ugr.es
[3] Department of Civil Engineering, LSI, University of Burgos, Burgos, Spain, 09006
jluengo@ubu.es

Abstract. The Nearest Neighbor rule is one of the most successful classifiers in machine learning but it is very sensitive to noisy data, which may cause its performance to deteriorate. This contribution proposes a new feature weighting classifier that tries to reduce the influence of noisy features. The computation of the weights is based on combining imputation methods and non-parametrical statistical tests. The results obtained show that our proposal can improve the performance of the Nearest Neighbor classifier dealing with different types of noisy data.

Keywords: noisy data, feature weighting, classification.

1 Introduction

The Nearest Neighbor (NN) classifier [4] uses the full training dataset to establish a classification rule, based on the most similar or nearest training instance to the query example. The most frequently used similarity function for the NN classifier is Euclidean distance [1] (see Equation 1, where X and Y are two instances and M is the number of features that describes them).

$$d(X,Y) = \sqrt{\sum_{i=0}^{M}(x_i - y_i)^2} \tag{1}$$

However, features containing enough noise may lead to erroneous similarities between the examples obtained and, therefore, to a deterioration in the performance of NN, which is known to be very sensitive to noisy data [10]. One way of overcoming this problem lies in modifying the similarity function, that is, the way in which the distances are computed. With this objective, Feature

M. Polycarpou et al. (Eds.): HAIS 2014, LNAI 8480, pp. 597–606, 2014.

Weighting methods [12], [9] try to improve the similarity function, by introducing a weight for each of the features (W_i, usually $W_i \in [0,1]$). These methods, which are mostly based in the Euclidean distance, modify the way in which the distance measure is computed (Equation 2), increasing the relevance of those features with greater weights associated with them (near to 1.0).

$$d_w(X,Y) = \sqrt{\sum_{i=0}^{M} W_i \cdot (x_i - y_i)^2} \tag{2}$$

These weights W_i can be regarded as a measure of how useful a feature is with respect to the final classification task. The higher a weight is, the more influence the associated feature will have in the decision rule used to compute the classification of a given example. Therefore, an adequate scheme of weights could be used to diminish the worst features of the domain of the problem, which could be those containing the more harmful amount of noise to the classification task. Thus, the accuracy of the classifier could be greatly improved if a proper selection of weights is made.

This contribution proposes a novel approach for weighting features, based on the usage of imputation methods [3], [6], [5]. These are commonly employed to estimate those feature values in a dataset that are unknown, formally known as missing values (MV), using the rest of the data available. Therefore, imputation methods enable us to estimate a new distribution of the original dataset, in which the distribution of each feature is conditioned to the rest of the features or all the data. These conditioned distributions of each feature can be compared with the original ones in order to detect the relevance of each feature, depending on the accuracy of the estimation for that feature performed by the imputation method.

The Kolmogorov-Smirnov statistic [11] may then be used to evaluate the differences between the original distribution of the features and that of the imputed ones. It is thus possible to measure how well the values of each feature can be predicted using the rest of the data. This enables us to give less importance to those features with high changes between their original and estimated value distributions - these features that contain too much noise or the more harmful noise and therefore are not easily predictable using the rest of the data, which increases the effect of those features that are easily predictable, and which have therefore likely a less amount of noise.

The study is completed with an experimentation in which our proposal is compared with the NN classifier, considering 25 supervised classification problems taken from the Keel-Dataset repository [2], into which we will introduce different types and levels of noise.

The rest of this contribution is organized as follows. In Section 2 we describe our proposal. In Section 3 we present the experimental framework, and in Section 4 we analyze the results obtained. Finally, in Section 5 we enumerate some concluding remarks.

2 A Weighting Scheme to Reduce the Effect of Noisy Data

This section describes the weighting method proposed, which is based on three main steps, described in the following subsections. Section 2.1 is devoted to the first step (called *the imputation phase*), whereas Section 2.2 describes the second step (*the computation of the weights*). Finally, Section 2.3 characterizes the third step (*the classification model*).

2.1 Imputation of the Dataset

The first step consists of creating a whole new estimated dataset DS' from the original one DS. In order to do this, an imputation method is used. In this contribution we will consider the following imputation methods (although other imputation methods may be chosen):

1. **KNNI** [3]. Based on the k-NN algorithm, every time an MV is found in a current example, KNNI computes the k ($k = 10$ in our experimentation) nearest neighbors and their average value is imputed. KNNI also uses the Euclidean distance as a similarity function.
2. **CMC** [6]. This method replaces the MVs by the average of all the values of the corresponding feature considering only the examples with the same class as the example to be imputed.
3. **SVMI** [5]. This is an SVM regression-based algorithm developed to fill in MVs. It works by firstly selecting the examples in which there are no missing feature values. In the next step, the method sets one of the input features, some of the values of which are missing, as the decision feature, and the decision feature as the input feature. Finally, an SVM for regression is used to predict the new decision feature.

If the original dataset DS is composed of the features $f_1, f_2, \ldots, f_M$, the imputed dataset DS' will be formed by the features $f'_1, f'_2, \ldots, f'_M$ whose values are obtained by the imputation method.

The procedure to obtain DS' from DS is based on assuming iteratively that each feature value of each example of the dataset DS, that is, $e(f_i)$, is missing. Then, the imputation method IM is used to predict a new value for that feature value. The new dataset DS' is obtained by repeating this process for each feature value, until the whole dataset has been processed. Carrying out this process, it is possible to estimate a distribution of values for each feature, which is conditioned to the rest of the features or the totality of the data. The new dataset DS' will contain these conditioned distributions for each feature.

2.2 Computation of Weights Using the Kolmogorov-Smirnov Test

The next step consists of measuring which features are most changed after the application of the imputation method. Given the nature of the imputation techniques, some features are expected to remain unchanged (or to present only

small changes in their values' distribution) whereas other features may present a higher level of disruption when their imputed values are compared with the original ones. Thus, those features that are more difficult to predict with the rest of the features/data will contain the more harmful noise and therefore we will try to make them less important to the classification task. The Kolmogorov-Smirnov test [11] provides a way of measuring these changes. This test works by computing a statistic D_n, which can be regarded as a measure of how different two samples are.

The test is a nonparametric procedure for testing the equality of two continuous, one-dimensional probability distributions. It quantifies a distance between the empirical distribution functions of two samples. The null distribution of its statistic, D_n, is computed under the null hypothesis that the samples are drawn from the same distribution.

Given two samples, X and Y, and their empirical distribution functions F_X and F_Y

$$F_X(x) = \frac{1}{n}\sum_{i=1}^{n} I_{X_i \leq x}, \qquad F_Y(x) = \frac{1}{n}\sum_{i=1}^{n} I_{Y_i \leq x} \tag{3}$$

(where $I_{X_i \leq x}$ is the indicator function, equal to 1 if $X_i \leq x$ and equal to 0 otherwise) the Kolmogorov-Smirnov statistic is

$$D_n = \sup_x |F_X - F_Y| \tag{4}$$

In the approach of this contribution, the D_n statistic provides a valuable way of estimating the degree of change undergone by a feature through the imputation process. By computing the D_n statistic associated with the differences between both samples of the feature (original and imputed), it is possible to measure the greater degree of difference between the expected distribution of both samples. Hence, the greater D_n value obtained, the more different the imputed version of the feature distribution will be (when compared with the original one).

The D_n statistic can be easily transformed into a weight. Since $D_n \in [0, 1]$, features with a lower value of D_n (near to 0.0) it will have little influence on the computation of the similarity function of the NN rule, whereas features with a higher value of D_n (near to 1.0) will be the most influential when computing the distance between two examples. Defining the statistical D_n^i for the feature i as

$$D_n^i = \text{Kolmogorov-Smirnov}(e_{f_i}, e_{f_i'}) \tag{5}$$

(where e_{f_i} and $e_{f_i'}$ are the empirical distributions of the features $f_i \in \mathcal{A}$ and $f_i' \in \mathcal{A}'$ respectively, and $\mathcal{A}$ denotes the set of features of the original dataset DS and $\mathcal{A}'$ denotes the set of features imputed in DS'), then the weights $W_i \in [0, 1]$ computed for a feature $f_i \in \mathcal{A}$ are

$$W_i = (1 - D_n^i)/(\sum_{j=1}^{M} 1 - D_n^j) \tag{6}$$

Therefore, the Kolmogorov-Smirnov test is applied to measure the degree of difference between each attribute f_i and its estimated version f_i'; then, this difference is used to build the weight for the attribute f_i (see Equation 6).

2.3 Final Classification Model

The final classifier considers NN with the weighted Euclidean distance (Equation 2) and the weights computed throughout the Kolmogorov-Smirnov statistic (Equation 6). Since we will consider three different imputation methods (KNNI, CMC and SVMI), three different feature weighting classifiers will be created. Throughout the study, we will denote them as FW-KNNI, FW-CMC and FW-SVMI according to the imputation method used.

Considering weights computed from the D_n statistic, we aim to reduce the effect that changing features have on the computation of the distance. These features, with a larger associated D_n value, will be those easily estimated by the imputation method (whose sample distribution differs poorly if the original and imputed versions are compared). They are preferred since they will contain a less harmful noise, and are the key features describing the dataset.

By contrast, features with a small D_n value will be those whose sample distribution has been greatly changed after the application of the imputation method. Since these features are not easily estimated when the rest of the data is available (the imputation method cannot recover their values properly), they are not preferred in the final computation of the distance, and thus a lower weight is assigned to them.

3 Experimental Framework

Section 3.1 describes the base datasets employed and Section 3.2 shows the noise introduction processes. Finally, Section 3.3 describes the methodology followed to analyze the results.

3.1 Base Datasets

The experimentation considers 25 real-world datasets from the KEEL-Dataset repository [2]. They are described in Table 3, where #EXA refers to the number of examples, #FEA to the number of numeric features and #CLA to the number of classes. For datasets containing missing values (such as *bands* or *dermatology*), the examples with missing values were removed from the datasets before their usage and thus all the attribute values of the datasets considered are known. In this way, the percentage of missing values of each dataset does not influence the results or conclusions obtained. Therefore, the only missing values considered in this contribution are those assumed during the execution of our proposal.

Table 1. Datasets employed in the experimentation

dataset	#EXA	#FEA	#CLA	dataset	#EXA	#FEA	#CLA
banana	5300	2	2	pima	768	8	2
bands	365	19	2	satimage	6435	36	7
bupa	345	6	2	sonar	208	60	2
dermatology	358	34	6	tae	151	5	3
ecoli	336	7	8	texture	5500	40	11
heart	270	13	2	vowel	990	13	11
hepatitis	80	19	2	wdbc	569	30	2
ionosphere	351	33	2	wine	178	13	3
iris	150	4	3	wq-red	1599	11	11
led7digit	500	7	10	wq-white	4898	11	11
mov-libras	360	90	15	wisconsin	683	9	2
newthyroid	215	5	3	yeast	1484	8	10
phoneme	5404	5	2				

3.2 Introducing Noise into Datasets

In order to control the amount of noise in each dataset and to check how it affects the classifiers, noise is introduced into each dataset in a supervised manner. Two different noise schemes, which are proposed in the specialized literature [14], are used in order to introduce a noise level of $x\%$ into each dataset.

- **Random Class Noise**. It supposes that exactly $x\%$ of the examples are corrupted. The class labels of these examples are randomly changed by other one out of the M classes.
- **Random Attribute Noise**. $x\%$ of the values of each attribute in the dataset are corrupted. To corrupt an attribute A_i, approximately $x\%$ of the examples in the dataset are chosen, and their A_i value is assigned a random value from $\mathbb{D}_i$. A uniform distribution is used either for numerical or nominal attributes.

A collection of new noisy datasets are created from the aforementioned 25 base real-world datasets. Both types of noise are independently considered: class and attribute noise. For each type of noise, the noise levels $x = 10\%$ and $x = 30\%$ are studied. Thus, the results of our proposal will be compared with those of NN considering three different scenarios: with the 25 unaltered real-world datasets, with the 25 datasets with a 10% of noise level and with the 25 datasets with a 30% of noise level.

3.3 Methodology of Analysis

The performance estimation of each classifier on each dataset is obtained by means of 3 runs of a 10-fold *distribution optimally balanced stratified cross-validation* (DOB-SCV) [7], averaging its test accuracy results. The usage of this

partitioning reduces the negative effects of both prior probability and covariate shifts [8] when classifier performance is estimated with cross-validation schemes.

For the sake of brevity, only the averaged performance results are shown for each classification algorithms at each type and level of induced noise, but it must be taken into account that our conclusions are based on the proper statistical analysis, which considers all the results (not averaged). Thus, in order to properly analyze the results obtained, Wilcoxons signed rank statistical test [13] is used, as suggested in the literature. This is a non-parametric pairwise test that aims to detect significant differences between two sample means; that is, between the behavior of the two algorithms involved in each comparison (which is usually viewed as the the averaged test performance results for each dataset). For each type and noise level, our proposal and NN using the Euclidean distance will be compared using Wilcoxons test and the p-values associated with these comparisons will be obtained. The p-value represents the lowest level of significance of a hypothesis that results in a rejection and it allows one to know whether two algorithms are significantly different and the degree of this difference.

4 Analysis of Results

This section presents the analysis of the results obtained. Each table of results is divided into two different parts. On the left hand of the table the average accuracy results are found, whereas on the right hand of the table the associated Wilconson's test p-values resulting of the comparison of each one of our proposals with the NN method are shown.

Table 2 shows the test accuracy obtained by each classifier on base and class noise datasets.

Table 2. Results on base and class noise datasets and associated p-values

	Accuracy			p-values		
Method	Base	$x = 10\%$	$x = 30\%$	Base	$x = 10\%$	$x = 30\%$
NN	79.37	74.96	65.46	-	-	-
FW-CMC	81.98	77.38	67.46	0.1107	0.1107	0.0787
FW-KNNI	81.97	77.36	67.41	0.1447	0.0827	0.2699
FW-SVMI	81.94	77.30	67.19	0.0626	0.0647	0.4352

From this table, several remarks can be made:

– The performance results of each one of our proposals is better than those of the NN method with the base datasets and also with the class noise datasets (approximately, higher than a 2% in all the cases).

– As the table shows, every proposal obtains low p-values when they are compared with NN: with the base datasets and both levels of class noise in the case of FW-CMC and with the base datasets and the noise level $x = 10\%$ in the case of the methods FW-KNNI and FW-SVMI. Some of these comparisons are also significant at a level of significance 0.1. This shows that the application of our approach to feature weighting improves the performance of the NN classifier with datasets suffering from class noise (sometimes significantly), regardless of the specific imputation method chosen.

On the other hand, Table 3 shows the test accuracy obtained by each classifier on base and attribute noise datasets. The following points are observed from this table:

– Our methods also outperforms the performance of NN with the datasets with different levels of attribute noise (generally they are a 2% better with the base datasets, a 1% better with the noise level $x = 10\%$ and a 0.5% with the noise level $x = 30\%$).
– The Wilcoxon's test p-values are also low, showing and advantage of our three proposals, even though in the case of FW-SVMI against NN with the noise level of $x = 30\%$ the p-value obtained is slightly higher. However, very low p-values are obtained with the two noise levels for the methods FW-CMC and FW-KNNI; they are indeed significant considering a significance level of 0.1.

Table 3. Results on base and attribute noise datasets and associated p-values

	Accuracy			p-values		
Method	Base	$x = 10\%$	$x = 30\%$	Base	$x = 10\%$	$x = 30\%$
NN	79.37	71.69	58.40	-	-	-
FW-CMC	81.98	72.62	59.17	0.1107	0.0067	0.0002
FW-KNNI	81.97	72.58	58.87	0.1447	0.0483	0.0246
FW-SVMI	81.94	72.44	58.61	0.0626	0.1318	0.2414

From the results of Tables 2-3, it is possible to conclude that the proposals presented in this contribution are able to improve the performance of the NN classifier dealing with noisy data, and in some cases, in a significant way.

5 Conclusions

In this contribution we have proposed a new scheme for feature weighting developed to improve the performance of the NN classifier in presence of noisy data, in which the weights are computed by combining imputation methods and the

Kolmogorov-Smirnov statistic. We have assigned a lower weight to that features that were more affected by the presence of noise (those features whose original and imputed distribution of values were more different). In this way, we have reduced the importance of these features that contain the more harmful noise and therefore are not easily predictable using the rest of the data and increased the importance of of those features that are easily predictable, and which have therefore likely a less amount of noise.

The results obtained show that all our approaches enhance the performance of NN in the presence of noise. The statistical analysis performed confirms our conclusions, even though in some cases the differences found are not statistically significant.

Acknowledgment. Supported by the Projects TIN2011-28488, P10-TIC-06858 and P11-TIC-9704. J. A. Sáez holds an FPU grant from the Spanish Ministry of Education and Science.

References

1. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning 6, 37–66 (1991)
2. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing 17(2-3) (2011)
3. Batista, G.E.A.P.A., Monard, M.C.: An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence 17(5-6), 519–533 (2003)
4. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13, 21–27 (1967)
5. Khosla, R., Howlett, R.J., Jain, L.C. (eds.): KES 2005. LNCS (LNAI), vol. 3683. Springer, Heidelberg (2005)
6. Ślęzak, D., Yao, J., Peters, J.F., Ziarko, W.P., Hu, X. (eds.): RSFDGrC 2005. LNCS (LNAI), vol. 3642. Springer, Heidelberg (2005)
7. Moreno-Torres, J.G., Sáez, J.A., Herrera, F.: Study on the Impact of Partition-Induced Dataset Shift on k-fold Cross-Validation. IEEE Transactions on Neural Networks and Learning Systems 23(8), 1304–1312 (2012)
8. Moreno-Torres, J.G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition 45(1), 521–530 (2012)
9. Paredes, R., Vidal, E.: Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(7), 1100–1110 (2006)
10. Sáez, J., Luengo, J., Herrera, F.: Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. Pattern Recognition 46(1), 355–364 (2013)

11. Smirnov, N.V.: Estimate of deviation between empirical distribution functions in two independent samples. Bulletin of Moscow University 2, 3–16 (1939) (in Russian)
12. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review 11, 273–314 (1997)
13. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83 (1945)
14. Zhu, X., Wu, X.: Class Noise vs. Attribute Noise: A Quantitative Study. Artificial Intelligence Review 22, 177–210 (2004)

3. Data level approaches and proposals to deal with noisy data

The journal papers associated to this part are:

3.1 Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification

Sáez J. A., Luengo J., and Herrera F. (2013) Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition* 46(1): 355–364

- Status: **Published**.
- Impact Factor (JCR 2013): 2.584
- Subject Category: Computer Science, Artificial Intelligence. Ranking 22 / 121 (**Q1**).
- Subject Category: Engineering, Electrical & Electronic. Ranking 40 / 247 (**Q1**).

Pattern Recognition 46 (2013) 355–364

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification

José A. Sáez [a,*], Julián Luengo [b], Francisco Herrera [a]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, Granada 18071, Spain
[b] Department of Civil Engineering, LSI, University of Burgos, Burgos 09006, Spain

ARTICLE INFO

Article history:
Received 8 March 2012
Received in revised form
6 July 2012
Accepted 14 July 2012
Available online 23 July 2012

Keywords:
Classification
Noisy data
Noise filtering
Data complexity measures
Nearest neighbor

ABSTRACT

Classifier performance, particularly of instance-based learners such as *k*-nearest neighbors, is affected by the presence of noisy data. Noise filters are traditionally employed to remove these corrupted data and improve the classification performance. However, their efficacy depends on the properties of the data, which can be analyzed by what are known as data complexity measures. This paper studies the relation between the complexity metrics of a dataset and the efficacy of several noise filters to improve the performance of the nearest neighbor classifier. A methodology is proposed to extract a rule set based on data complexity measures that enables one to predict in advance whether the use of noise filters will be statistically profitable. The results obtained show that noise filtering efficacy is to a great extent dependent on the characteristics of the data analyzed by the measures. The validation process carried out shows that the final rule set provided is fairly accurate in predicting the efficacy of noise filters before their application and it produces an improvement with respect to the indiscriminate usage of noise filters.

1. Introduction

Real-world data is commonly affected by noise [1,2]. The building time, complexity and, particularly, the performance of the model, are usually deteriorated by noise in classification problems [3–5]. Several learners, e.g., C4.5 [6], are designed taking these problems into account and incorporate mechanisms to reduce the negative effects of noise. However, many other methods ignore these issues. Among them, instance-based learners, such as *k*-nearest neighbors (*k*-NN) [7–9], are known to be very sensitive to noisy data [10,11].

In order to improve the classification performance of noise-sensitive methods when dealing with noisy data, noise filters [12–14] are commonly applied. Their aim is to remove potentially noisy examples before building the classifier. However, both correct examples and examples containing valuable information can also be removed. This fact implies that these techniques do not always provide an improvement in performance. As indicated by Wu and Zhu [1], the success of these methods depends on several circumstances, such as the kind and nature of the data errors, the quantity of noise removed or the capabilities of the classifier to deal with the loss of useful information related to the filtering. Therefore, the efficacy of noise filters, i.e., whether their usage causes an improvement in classifier performance, depends on the noise-robustness and the generalization capabilities of the classifier used, but it also strongly depends on the characteristics of the data.

Data complexity measures [15] are a recent proposal to represent characteristics of the data which are considered difficult in classification tasks, e.g., the overlapping among classes, their separability or the linearity of the decision boundaries.

This paper proposes the computation of these data complexity measures to predict in advance when the usage of a noise filter will statistically improve the results of a noise-sensitive learner: the nearest neighbor classifier (1-NN). This prediction can help, for example, to determine an appropriate noise filter for a concrete noisy dataset – that filter providing a significant advantage in terms of the results – or to design new noise filters which select more or less aggressive filtering strategies considering the characteristics of the data. Choosing a noise-sensitive learner facilitates the checking of when a filter removes the appropriate noisy examples in contrast to a robust learner—the performance of classifiers built by the former is more sensitive to noisy examples retained in the dataset after the filtering process. In addition, this paper has the following objectives:

1. To analyze the relation between the characteristics of the data and the efficacy of several noise filters.
2. To find a reduced set of the most appropriate data complexity measures for predicting the noise filtering efficacy.

* Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
E-mail addresses: smja@decsai.ugr.es, tschigorine@gmail.com (J.A. Sáez), jluengo@ubu.es (J. Luengo), herrera@decsai.ugr.es (F. Herrera).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.07.009

3. Even though each noise filter may depend on concrete characteristics of the data to work correctly, it would be interesting to identify common characteristics of the data under which most of the noise filters work properly.
4. To provide a set of interpretable rules which a practitioner can use to determine whether to use a noise filter with a classification dataset.

A web page with the complementary material of this paper is available at http://sci2s.ugr.es/filtering-efficacy. It includes the details of the experimentation, the datasets used, the performance results of the noise filters and the distribution of the data complexity metrics of the datasets.

The rest of this paper is organized as follows. Section 2 presents data complexity measures. Section 3 introduces the noise filters and enumerates those considered in this paper. Section 4 describes the method employed to extract the rules predicting the noise filtering efficacy. Section 5 shows the experimental study performed and the analysis of results. Finally, Section 6 enumerates some concluding remarks.

2. Data complexity measures

In this section, first a brief review of recent studies on data complexity metrics is presented (Section 2.1). Then, the measures of overlapping (Section 2.2), the measures of separability of classes (Section 2.3) and the measures of geometry (Section 2.4) used in this paper are described.

2.1. Recent studies on data complexity

There are some methods used in classification, either learner or preprocessing techniques, which work well with concrete datasets, while other techniques work better with different ones. This is due to the fact that each classification dataset has particular characteristics that define it. Issues such as the generality of the data, the inter-relationships among the variables and other factors are key for the results of such methods. An emergent field proposes the usage of a set of data complexity measures to quantify these particular sources of the problem on which the behavior of classification methods usually depends [15].

A seminal work on data complexity is [16], in which some complexity measures for binary classification problems are proposed, gathering metrics of three types: overlaps in feature values from different classes; separability of classes; and measures of geometry, topology and density of manifolds. Extensions can also be found in the literature, such as in the work of Singh [17], which offers a review of data complexity measures and proposes two new ones.

From these works, different authors attempt to address different data mining problems using these measures. For example, Baumgartner and Somorjai [18] define specialized measures for regularized linear classifiers. Other authors try to explain the behavior of learning algorithms using these measures, optimizing the decision tree creation in the binarization of datasets [19] or to analyze fuzzy-UCS and the model obtained when applied to data streams [20]. The data complexity measures have been referred to other related fields, such as gene expression analysis in Bioinformatics [21,22].

The research efforts in data complexity are currently focused on two fronts. The first aims to establish suitable problems for a given classification algorithm, using only the data characteristics, and thus determining their domains of competence. In this line of research recent publications, e.g., the works of Luengo and Herrera [23] and Bernadó-Mansilla and Ho [24], provide a first insight into the determination of an individual classifier's domains of competence. Parallel to this, Sánchez et al. [25] study the effect of data complexity on the nearest neighbor classifier. The relationships between the domains of competence of similar classifiers were analyzed by Luengo and Herrera [26], indicating that related classifiers benefit from common sources of complexity of the data.

Data complexity measures are increasingly used in order to characterize when a preprocessing stage will be beneficial to a subsequent classification algorithm in many challenging domains. García et al. [27] firstly analyzed the behavior of the evolutionary prototype selection strategy using one complexity measure based on overlapping. Further developments resulted in a characterization of when the preprocessing in imbalanced datasets is beneficial [28]. The data complexity measures can also be used online in the data preparation step. An example of this is the work of Dong [29], in which a feature selection algorithm based on complexity measures is proposed.

This paper follows the second research line. It aims to characterize when a filtering process is beneficial using the information provided by the data complexity measures. Noise will affect the geometry of the dataset, and thus the values of the data complexity metrics. It can be expected that such metrics will enable one to know in advance whether noise filters will be useful for the given dataset.

In this study, 11 of the metrics proposed by Ho and Basu [16] will be analyzed. In the following subsections, these measures, classified by their family, are briefly presented. For a deeper description of their characteristics, the reader may consult [16].

2.2. Measures of class overlapping

These measures focus on the effectiveness of a single feature dimension in separating the classes, or the composite effects of a number of dimensions. They examine the range and spread of values in the dataset within each class and check for overlapping among different classes.

- F1—*maximum Fisher's discriminant ratio*: This is the value of Fisher's discriminant ratio of the attribute that enables one to better discriminate between the two classes, computed as

$$\mathrm{F1} = \max_{i=1,\ldots,d} \frac{(\mu_{i,1}-\mu_{i,2})^2}{\sigma_{i,1}^2+\sigma_{i,2}^2} \tag{1}$$

  where d is the number of attributes, and $\mu_{i,j}$ and $\sigma_{i,j}^2$ are the mean and variance of the attribute i in the class j, respectively.
- F2—*volume of the overlapping region*: This measures the amount of overlapping of the bounding boxes of the two classes. Let $\max(f_i,C_j)$ and $\min(f_i,C_j)$ be the maximum and minimum values of the feature f_i in the set of examples of class C_j, let minmax_i be the minimum of $\max(f_i,C_j),(j=1,2)$ and maxmin_i be the maximum of $\min(f_i,C_j),(j=1,2)$ of the feature f_i. Then, the measure is defined as

$$\mathrm{F2} = \prod_{i=1\ldots d} \frac{\mathrm{minmax}_i-\mathrm{maxmin}_i}{\max(f_i,C_1\cup C_2)-\min(f_i,C_1\cup C_2)} \tag{2}$$

- F3—*maximum feature efficiency*: This is the maximum fraction of points distinguishable with only one feature after removing unambiguous points falling outside of the overlapping region in this feature [30].

2.3. Measures of separability of classes

These give indirect characterizations of class separability. They assume that a class is made up of single or multiple manifolds

that form the support of the probability distribution of the given class. The shape, position and interconnectedness of these manifolds give hints of how well the two classes are separated, but they do not describe separability by design.

- L1—*minimized sum of error distance by linear programming*: This is the value of the objective function that tries to minimize a linear classifier obtained by the linear programming formulation proposed by Smith [31]. The method minimizes the sum of distances of error points to the separating hyperplane. The measure is normalized by the number of points in the problem and also by the length of the diagonal of the hyper-rectangular region enclosing all training points in the feature space.
- L2—*error rate of linear classifier by linear programming*: This measure is the error rate of the linear classifier defined for L1, measured with the training set.
- N1—*rate of points connected to the opposite class by a minimum spanning tree*: N1 is computed using a minimum spanning tree [32], which connects all the points to their nearest neighbors. Then the number of points connected to the opposite class by an edge of this tree are counted. These are considered to be the points lying next to the class boundary. N1 is the fraction of such points over all points in the dataset.
- N2—*ratio of average intra/inter class nearest neighbor distance*: This is computed as

$$N2 = \frac{\sum_{i=0}^{m} intra(x_i)}{\sum_{i=0}^{m} inter(x_i)} \quad (3)$$

where m is the number of examples in the dataset, $intra(x_i)$ the distance to its nearest neighbor within the class, and $inter(x_i)$ the distance to the nearest neighbor of any other class. This metric compares the within-class spread with the distances to the nearest neighbors of other classes. Low values of this metric suggest that the examples of the same class lie close in the feature space, whereas large values indicate that the examples of the same class are dispersed.
- N3—*error rate of the 1-NN classifier*: This is the error rate of a nearest neighbor classifier estimated by the leave-one-out method. This measure denotes how close the examples of different classes are. Low values of this metric indicate that there is a large gap in the class boundary.

2.4. Measures of geometry, topology, and density of manifolds

These measures evaluate to what extent two classes are separable by examining the existence and shape of the class boundary. The contributions of individual feature dimensions are combined and summarized in a single score, usually a distance metric, rather than evaluated separately.

- L3—*nonlinearity of a linear classifier by linear programming*: Hoekstra and Duin [33] propose a measure for the nonlinearity of a classifier with respect to a given dataset. Given a training set, the method first creates a test set by linear interpolation (with random coefficients) between randomly drawn pairs of points from the same class. Then, the error rate of the classifier (trained by the given training set) on this test set is measured.
- N4—*nonlinearity of the 1-NN classifier*: The error is calculated for a nearest neighbor classifier. This measure is for the alignment of the nearest neighbor boundary with the shape of the gap or overlap between the convex hulls of the classes.
- T1—*ratio of the number of hyperspheres, given by* ϵ-neighborhoods, *by the total number of points*: The local clustering properties of a point set can be described by an ϵ-neighborhood pretopology [34]. Instance space can be covered by ϵ-neighborhoods by means of hyperspheres (the procedure to compute them can be found in [16]). A list of such hyperspheres needed to cover the two classes is a composite description of the shape of the classes. The number and size of the hyperspheres indicate how much the points tend to be clustered in hyperspheres or distributed in thinner structures. In a problem where each point is closer to points of the other class than points of its own, each hypersphere is retained and is of a low size. T1 is the normalized count of the retained hyperspheres by the total number of points.

3. Corrupted data treatment by noise filters

Noise filters are preprocessing mechanisms designed to detect and eliminate noisy examples in the training set. The result of noise elimination in preprocessing is a reduced and improved training set which is then used as an input to a machine learning algorithm.

There are several of these filters based on using the distance between examples to determine their similarity and create neighborhoods. These neighborhoods are used to detect suspicious examples which can then be eliminated. The Edited Nearest Neighbor [12] or the Prototype Selection based on Relative Neighborhood Graphs [35] are some examples of methods that can be found within this group of noise filters.

Another group of noise filters creates classifiers over several subsets of the training data in order to detect noisy examples. Brodley and Friedl [13] trained multiple classifiers built by different learning algorithms, such as k-NN [7], C4.5 [6] and a Linear Discriminant Analysis [36], from a corrupted dataset and then used them to identify mislabeled data, which are characterized as the examples that are incorrectly classified by the multiple classifiers. Similar techniques have been widely developed considering the building of several classifiers with the same learning algorithm [37,38]. Instead of using multiple classifiers learned from the same training set, Gamberger et al. [37] suggest a Classification Filter (CF) approach, in which the training set is partitioned into n subsets, then a set of classifiers is trained from the union of any $n-1$ subsets; those classifiers are used to classify the examples in the excluded subset, eliminating the examples that are incorrectly classified.

The noise filters analyzed in this paper are shown in Table 1. They have been chosen due to their good behavior with many real-world problems.

4. Obtaining rules to predict the noise filtering efficacy

In order to provide a rule set based on the characteristics of the data which enables one to predict whether the usage of noise filters will be statistically beneficial, the methodology shown in

Table 1
Noise filters employed in the experimentation.

Filter	Reference	Abbreviation
Classification filter	[37]	CF
Cross-validated committees filter	[38]	CVCF
Ensemble filter	[13]	EF
Edited nearest neighbor with estimation of probabilities threshold	[39]	ENNTh
Edited nearest neighbor	[12]	ENN
Iterative-partitioning filter	[40]	IPF
Nearest centroid neighborhood edition	[41]	NCNEdit
Prototype selection based on relative neighborhood graphs	[35]	RNG

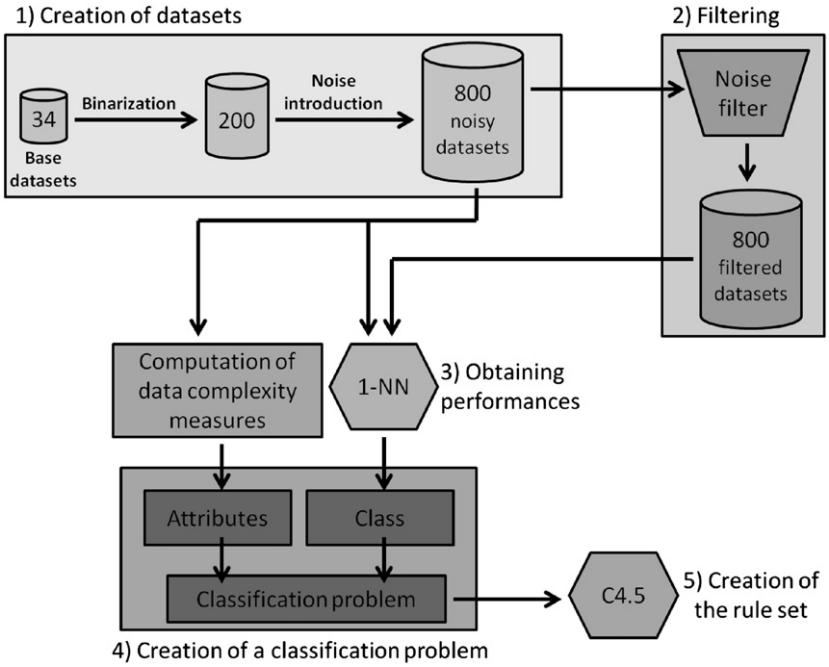


Fig. 1. Methodology to obtain the rule set predicting the noise filtering efficacy.

Table 2
Base datasets and their number of instances (#INS), attributes (#ATT) and classes (#CLA). (R/I/N) refers to the number of real, integer and nominal attributes.

Dataset	#INS	#ATT (R/I/N)	#CLA	Dataset	#INS	#ATT (R/I/N)	#CLA
australian	690	14 (3/5/6)	2	led7digit	500	7 (7/0/0)	10
balance	625	4 (4/0/0)	3	mammographic	830	5 (0/5/0)	2
banana	5300	2 (2/0/0)	2	monk-2	432	6 (0/6/0)	2
bands	365	19 (13/6/0)	2	mushroom	5644	22 (0/0/22)	2
bupa	345	6 (1/5/0)	2	pima	768	8 (8/0/0)	2
car	1728	6 (0/0/6)	4	ring	7400	20 (20/0/0)	2
chess	3196	36 (0/0/36)	2	saheart	462	9 (5/3/1)	2
contraceptive	1473	9 (0/9/0)	3	sonar	208	60 (60/0/0)	2
crx	653	15 (3/3/9)	2	spambase	4597	57 (57/0/0)	2
ecoli	336	7 (7/0/0)	8	tae	151	5 (0/5/0)	3
flare	1066	11 (0/0/11)	6	tic-tac-toe	958	9 (0/0/9)	2
glass	214	9 (9/0/0)	7	titanic	2201	3 (3/0/0)	2
hayes-roth	160	4 (0/4/0)	3	twonorm	7400	20 (20/0/0)	2
heart	270	13 (1/12/0)	2	wdbc	569	30 (30/0/0)	2
housevotes	232	16 (0/0/16)	2	wine	178	13 (13/0/0)	3
ionosphere	351	33 (32/1/0)	2	wisconsin	683	9 (0/9/0)	2
iris	150	4 (4/0/0)	3	yeast	1484	8 (8/0/0)	10

Fig. 1 has been designed. The complete process[1] is described as follows.

1. 800 different classification datasets are built as follows (these are common to all noise filters):
   - The 34 datasets shown in Table 2 have been selected from the KEEL-dataset repository[2] [42].
   - 200 binary datasets – with more than 100 examples in each one – are built from these 34 datasets. Multi-class datasets are used to create other binary datasets by means of the selection and/or combination of their classes. Only problems with two classes are considered as the data complexity measures are only well defined to work on binary problems. The amount of examples of the two classes has been taken into account in order to create the datasets; they are intended to be as similar as possible. Let *IR* be the fraction between the number of examples of the majority and the minority class—formally known as *imbalanced ratio* [43]. In order to control the size of both classes, only datasets with a low imbalanced ratio were created, specifically with $1 \leq IR \leq 2.25$. Therefore, the size of both classes is sufficiently similar. This prevents filtering methods from deleting all the examples from the minority class, which can occur if a high imbalanced ratio is present in the data since the filtering methods used do not take into account the class imbalance and may consider these examples to be noise.
   - Finally, in order to study the behavior of the noise filters in several circumstances, several noise levels x (0%, 5%, 10% and 15%) are introduced into these 200 datasets, resulting in 800 datasets. Noise is introduced in the same way as in [3], a reference paper in the framework of noisy data in classification. Each attribute A_i is corrupted separately: x% of the examples are chosen and the A_i value of each of these

[1] The datasets used in this procedure and the performance results of 1-NN – with and without the usage of noise filters – can be found on the web page of this paper.
[2] http://www.keel.es/datasets.php.

examples is assigned a random value of the domain of that attribute following a uniform distribution. One must take into account that these 200 datasets may contain noise, so the real noise level after the noise introduction process may be higher.

2. These 800 datasets are filtered with a noise filter, leading to 800 new filtered datasets.
3. The test performance of 1-NN [7,44,45] on each of the 800 datasets, both with and without the application of the noise filter, is computed. The estimation of the classifier performance is obtained by means of three runs of a 10-fold cross-validation and their results are averaged. The AUC metric [46] is used due it being commonly employed when working with binary datasets and the fact that it is less sensitive to class imbalance. The performance estimation is used to check which datasets are improved in their performance by 1-NN when using the noise filter.
4. A classification problem is created with each example being one of the datasets built and in which:
   - The attributes are the 11 data complexity metrics for each dataset. The distribution of the values of each data complexity measure can be found on the web page with complementary material for this paper.
   - The class label represents whether the usage of the noise filter implies a statistically significant improvement of the test performance. Wilcoxon's statistical test [47] – with a significance level of $\alpha = 0.1$ – is applied to compare the performance results of the 3×10 test folds with and without the usage of the noise filter. Depending on whether the usage of the noise filter is statistically better than the lack of filtering, each example is labeled as *positive* or *negative*, respectively.
5. Finally, similar to the method of Orriols-Puig and Casillas [20], the C4.5 algorithm [6] is used to build a decision tree on the aforementioned classification problem, which can be transformed into a rule set. The performance estimation of this rule set is obtained using a 10-fold cross-validation. By means of the analysis of the decision trees built by C4.5, it is possible to check which are the most important data complexity metrics to predict the noise filtering efficacy, i.e., those in the top levels of the tree and appearing more times, and their performance examining the test results.

5. Experimental study

The experimentation is organized in five different parts, each one in a different subsection and with a different objective:

1. *To check to what extent the noise filtering efficacy can be predicted using data complexity measures* (Section 5.1). In order to do this, the procedure described in Section 4 is followed with each noise filter. Thus, a rule set based on all the data complexity measures is learned to predict the efficacy of each noise filter. Its performance, which is estimated using a 10-fold cross-validation, gives a measure of the relation existing between the data complexity metrics and the noise filtering efficacy—a higher performance will imply a stronger relation.
2. *To provide a reduced set of data complexity metrics that best determine whether to use a noise filter and do not cause the prediction capability to deteriorate* (Section 5.2). The decision trees built in the above step by C4.5 are analyzed, studying two elements:
   - The *order*, from 1 to 11, in which the first node corresponding with each data complexity metric appears in the decision tree, starting from the root. This order is averaged over the 10 folds.
   - The *percentage* of nodes of each data complexity metric in the decision tree, averaged over the 10 folds.

   This analysis will provide the better discriminating metrics and those appearing more times in the decision trees—they are not necessarily placed in the top positions of the tree but are still important to discriminate between the two classes. In this way, the rule sets obtained in the above step are simplified and thus become more interpretable.
3. *To find common characteristics of the data on which the efficacy of all noise filters depends* (Section 5.3). Each noise filter may depend on concrete values of the data complexity metrics, i.e., on concrete characteristics of the data, to work properly. However, it is interesting to investigate whether there are common characteristics of the data under which all noise filters work properly. To do this, the rule set learned with each noise filter will be applied to predict the efficacy of the rest of the noise filters. The rule set achieving the highest performance predicting the efficacy of the different noise filters will have rules more similar to the rest of noise filters, i.e., the rules will cover similar areas of the domain.
4. *To provide the rule set which works best predicting the noise filtering efficacy of all the noise filters* (Section 5.4). The study of the above point will provide the rule set which best represents the characteristics under which the majority of the noise filters work well. The behavior of these rules with each noise filter will be analyzed in this section, paying attention to the *coverage* of each rule—the percentage of examples covered, and its *accuracy*—the percentage of correct classifications among the examples covered.
5. *To perform an additional validation of the chosen rule set* (Section 5.5). Even though the behavior of each rule set is validated using a 10-fold cross-validation in each of the above steps, a new validation phase with new datasets is performed in this section. These datasets are used to check if the chosen rule set is really more advantageous than the indiscriminate application of the noise filters to all the datasets.

5.1. Data complexity measures and noise filtering efficacy

The procedure described in Section 4 has been followed with each one of the noise filters. Table 3 shows the performance results of the rule sets obtained with C4.5 on the training and test sets for each noise filter when predicting the noise filtering efficacy, i.e., when discriminating between the aforementioned *positive* and *negative* classes.

The training performance is very high for all the noise filters – it is close to the maximum achievable performance – and there are no differences between the eight noise filters. The test performance results, although not at the same level as

Table 3
Performance results of C4.5 predicting the noise filtering efficacy (11 data complexity measures used).

Noise filter	Training	Test
CF	0.9979	0.8446
CVCF	0.9966	0.8353
EF	0.9948	0.8176
ENNTh	0.9958	0.8307
ENN	0.9963	0.8300
IPF	0.9973	0.8670
NCNEdit	0.9945	0.8063
RNG	0.9969	0.8369
Mean	**0.9963**	**0.8335**

Table 4
Averaged order of the data complexity measures in the decision trees.

Metric	CF	CVCF	EF	ENNTh	ENN	IPF	NCNEdit	RNG	Mean
F1	3.70	4.80	5.90	8.60	6.40	4.50	6.00	8.20	**6.01**
F2	1.40	1.00	1.00	1.00	1.00	1.00	1.50	1.00	**1.11**
F3	2.50	3.40	10.10	5.80	4.10	3.30	7.20	4.50	**5.11**
N1	10.50	9.90	9.10	10.30	8.40	7.10	8.10	8.50	8.99
N2	6.20	2.00	3.30	8.00	2.30	3.00	4.60	2.70	**4.01**
N3	8.80	8.50	7.80	11.00	7.00	9.50	7.90	8.70	8.65
N4	7.40	9.70	9.90	7.20	11.00	10.50	8.20	5.60	8.69
L1	9.20	10.00	7.90	9.70	11.00	6.00	9.40	9.60	9.10
L2	8.10	6.80	9.30	8.40	10.30	10.00	11.00	10.50	9.30
L3	7.80	8.70	4.60	8.60	11.00	5.90	7.80	8.40	7.85
T1	6.70	6.80	5.20	3.50	4.80	11.00	6.30	4.50	**6.10**

the training results, are also noteworthy. All of them have more than 0.8 success, with the averaged test performance of all the noise filters higher than 0.83. These results show that noise filtering efficacy can be predicted with a good performance by means of data complexity measures. Therefore, a clear relation can be seen between both concepts, i.e., data complexity metrics and filtering efficacy.

5.2. Metrics that best predict the noise filtering efficacy

In order to find the subset of data complexity measures that enables the best decision to be made of whether a noise filter should be used, the decision trees built by C4.5 in the previous section are analyzed. Table 4 shows the averaged *order* of each data complexity measure in which it appears in the decision trees built for each noise filter.

These results show that the three best measures are generally F2, N2 and F3:

- F2 is the first measure for all noise filters.
- N2 is placed in six of the eight noise filters as the second metric.
- F3 is placed between the second and third positions in another six of the eight noise filters.

The following two measures in importance are T1 and F1:

- T1 appears in seven of the eight noise filters between the second and fifth positions.
- F1 appears in six of the eight noise filters between the third and fifth positions.

The rest of the measures have a lower discriminative power, due their positions being worse. Averaged results for all noise filters also support these conclusions. Therefore, the aforementioned measures (F2, N2, F3, T1 and F1) are the most important for all the noise filters, even though the concrete order can vary slightly from some filters to others.

From these results, the measures of overlapping among the classes (F1, F2 and F3) are the group of metrics that most influence predictions of the filtering efficacy. The filtering efficacy is particularly dependent on the volume of the overlapping region (F2) and, to a lesser degree, on the rest of the overlapping metrics (F3 and F1) which, using different methods, compute the discriminative power of the attributes. The dispersion of the examples within each class (N2) and the shape of the classes and the complexity of the decision boundaries (T1) must also be taken into account to predict the filtering efficacy. In short, all these metrics provide information about the shape of the classes and the overlapping among them, which may be key factors in the success of any noise filtering technique.

Since the efficacy of the noise filters has been studied over the results of the 1-NN classifier, one could expect a greater influence of measures based on 1-NN, such as N3 and N4. These measures are based on the error rate of the 1-NN classifier –the former is computed on the training set whereas the latter is computed on an artificial test set. It is important to point out that 1-NN is very sensitive to the closeness of only one example to others belonging to a different class [16,25] and a similar error rate may be due to multiple situations where the filtering may be beneficial or not, for example:

1. Existence of isolated noisy examples.
2. A large overlapping between the classes.
3. Closeness between the classes (although overlapping does not exist).

A noise filtering method is likely to be beneficial in the first scenario because isolated noisy examples are likely to be identified and removed, improving the final performance of the classifier. However, the situation is not so clear in the other two scenarios: the filtering may delete important parts of the domain and disturb the boundaries of the classes or, on the contrary, it may clean up the overlapping region and create more regular class boundaries [1,48]. Therefore, the multiple causes on which the error rate of 1-NN depends imply that measures based on it, such as N3 and N4, are not always good indicators of the noise filtering efficacy.

Table 5 shows the *percentage* of nodes referring to each data complexity measure in the decision trees for each of the noise filters. These results provide similar conclusions to those of the *order* results, with the most representative measures again being F1, F2, F3, N2 and T1, while the rest of the measures have lower percentages.

The *order* and *percentage* results show that the measures F1, F2, F3, N2 and T1 are the most discriminative and have a higher number of nodes in the decision trees. It is aimed to attain a reduced set, from among these five metrics, that enables filtering efficiency to be predicted without a loss in accuracy with respect to all the measures. In order to avoid the study of all the existing combinations of the five metrics, the following experimentation is mainly focused on the measures F2, N2 and F3, the most discriminative ones—since the *order* results can be considered more important than the *percentage* results. The incorporation into this set of T1, F1 or both is also studied. The prediction capability of the measure F2 alone, since is the most discriminative one, is also shown. All these results are presented in Table 6.

The training results of these combinations do not change with respect to the usage of all the metrics. However, the test performance results improve in many cases the results of using all the metrics, particularly in the cases of F2–N2–F3–T1–F1 and

Table 5
Percentage of the number of nodes of each data complexity measure in the decision trees.

Metric	CF	CVCF	EF	ENNTh	ENN	IPF	NCNEdit	RNG	Mean
F1	22.45	21.24	14.94	8.47	17.07	20.66	18.67	5.71	**16.15**
F2	15.31	11.50	14.94	18.64	12.20	9.09	14.67	9.52	**13.23**
F3	16.33	23.01	2.30	13.56	20.73	23.14	12.00	18.10	**16.15**
N1	2.04	2.65	3.45	1.69	8.54	8.26	5.33	3.81	4.47
N2	8.16	11.50	17.24	6.78	19.51	9.92	16.00	19.05	**13.52**
N3	5.10	5.31	5.75	0.00	6.10	3.31	6.67	4.76	4.62
N4	8.16	3.54	2.30	13.56	0.00	0.83	6.67	12.38	5.93
L1	3.06	2.65	8.05	3.39	0.00	7.44	2.67	5.71	4.12
L2	6.12	7.08	5.75	8.47	1.22	2.48	0.00	0.95	4.01
L3	5.10	3.54	16.09	6.78	0.00	14.88	9.33	4.76	7.56
T1	8.16	7.96	9.20	18.64	14.63	0.00	8.00	15.24	**10.23**

Table 6
Performance results of C4.5 predicting the noise filtering efficacy (measures used: F2, N2, F3, T1 and F1).

Noise filter	F2		F2–N2–F3–T1–F1		F2–N2–F3–F1		F2–N2–F3–T1		F2–N2–F3	
	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test
CF	0.9991	0.7766	0.9975	0.8848	0.9986	0.8623	0.9983	0.8949	0.9972	0.8713
CVCF	1.0000	0.5198	0.9997	0.8102	0.9983	0.7943	0.9994	0.8165	0.9977	0.8152
EF	1.0000	0.7579	0.9993	0.8102	0.9991	0.8101	0.9997	0.8297	0.9997	0.8421
ENNTh	1.0000	0.8419	0.9996	0.8309	0.9996	0.8281	0.9907	0.8052	0.9992	0.8302
ENN	1.0000	0.7361	0.9928	0.8942	0.9935	0.8662	0.9966	0.8948	0.9967	0.7946
IPF	1.0000	0.7393	0.9975	0.8378	0.9989	0.8119	0.9986	0.8019	0.9985	0.7725
NCNEdit	0.9981	0.8024	0.9977	0.8164	0.9982	0.8231	0.9983	0.8436	0.9912	0.8136
RNG	0.9993	0.7311	0.9967	0.8456	0.9983	0.8086	0.9989	0.8358	0.9980	0.7754
Mean	0.9996	0.7381	0.9976	0.8413	0.9981	0.8256	0.9976	0.8403	0.9973	0.8144

Table 7
Ranks computed by Wilcoxon's test $R+/R-$, representing the ranks obtained by the combination of the row and the column, respectively. *All* refers to the usage of all the complexity metrics.

Metrics	F2–N2–F3	F2–N2–F3–T1	F2–N2–F3–F1	F2–N2–F3–F1–T1	All
F2–N2–F3	–	6/30	12/24	8/28	11/25
F2–N2–F3–T1	30/6	–	30/6	19/17	20/16
F2–N2–F3–F1	24/12	6/30	–	3/33	13/23
F2–N2–F3–F1–T1	28/8	17/19	33/3	–	23/13
All	25/11	16/20	23/13	13/23	–

F2–N2–F3–T1. This is because the unnecessary measures to predict the filtering efficacy which can introduce a bias into the datasets have been removed. However, the usage of the measure F2 alone to predict the noise filtering efficacy with a good performance can be discarded, since its results are not good enough compared with the cases where more than one measure is considered. This fact reflects that the usage of single measures does not provide enough information to achieve a good filtering efficacy prediction result. Therefore, it is necessary to combine several measures which examine different aspects of the data.

In order to determine which combination of measures is chosen as the most suitable one, Wilcoxon's statistical test is performed, comparing the test results of Tables 3 and 6 of each noise filter. Table 7 shows the ranks obtained by each combination of metrics.

From these results, the combinations of metrics F2–N2–F3–T1 and F2–N2–F3–T1–F1 are noteworthy. Removing some data complexity metrics improves the performance with respect to all the metrics. However, it is necessary to retain a minimum number of metrics representing as much information as possible. Note that these two sets contain measures of three different types: overlapping, separability of classes and geometry of the dataset. Therefore, even though the differences are not significant in all cases, the combination with more ranks and a lower number of measures, i.e., F2–N2–F3–T1, can be considered the most appropriate and will be chosen for a deeper study.

5.3. Common characteristics of the data on which the efficacy of the noise filters depends

From the results shown in Table 6, the rules learned with any noise filter can be used to accurately predict filtering efficacy because they obtain good test performance results. However, these rules should be used to predict the behavior of the filter from which they have been learned.

It would be interesting to provide a single rule set, better adapting the behavior of all the noise filters. In order to do this, the rules learned to predict the behavior of one filter will be tested to predict the behavior of the rest of the noise filters (see Table 8). From these results, the prediction performance of the rules learned for the RNG filter is clearly the more general, since they are applicable to the rest of the noise filters obtaining the best prediction results—see the last column with an average of 0.8786. Therefore, this rule set has rules that are more similar to the rest of the noise filters and thus, it represents better the common characteristics on which the efficacy of all noise filters depends.

5.4. Analysis of the chosen rule set

The rule set chosen to predict the filtering efficacy of all the noise filters is shown in Table 9. The analysis of such rules is shown in Table 10, where the *coverage* (Cov) and the *accuracy* (Acc) of each rule is shown.

These results show that the rules with the highest coverage in predicting the behavior of all noise filters are R6, R5 and R10. Moreover, the rules predicting the *positive* examples have a very high accuracy rate, close to 100%. The rule R5 has the highest coverage among the rules predicting the *negative* class, although its accuracy is a bit lower than that of the rules R6 and R10. This could be due to the fact that the datasets in which the application of a noise filter implies a disadvantage are more widely dispersed in the search space and, that being so, creating general rules is more complex. The rest of the rules have a lower coverage, although their accuracy is generally high, so they are more specific rules.

The rules R6 and R10 are characterized by having a value of F2 higher than 0.43. Moreover, the rule R6 requires a value of T1 lower than 0.9854, i.e., a large part of the domain of the metric T1. However, as reflected in the experimentation in [16] and also on the web page with complementary material for this paper, a large number of datasets have a T1 value of around 1. The incorporation, therefore, of the measure T1 into the rules and the multiple values between 0.9 and 1 of this metric in the antecedents should not be surprising.

By contrast, the rule R5 has a value of F2 lower than 0.43. Other metrics are also included in this rule, such as N2 with a value higher than 0.41 and F3 with a value higher than 0.1.

Table 8
Performance results of the rules learned with the method in the column predicting the efficacy of the noise filter in the row.

Noise filter	CF	CVCF	EF	ENN	ENNTh	IPF	NCNEdit	RNG
CF	–	0.8848	0.8631	0.9049	0.8114	**0.9230**	0.8590	0.9172
CVCF	0.8030	–	0.7656	0.8884	0.7373	0.9024	0.7747	**0.9115**
EF	0.8756	0.8044	–	0.8597	0.8540	0.8425	0.8824	**0.8901**
ENN	0.7795	0.8588	0.7804	–	0.7512	0.8161	0.7804	**0.8865**
ENNTh	0.7900	0.7681	0.8176	0.8083	–	0.8114	**0.8362**	0.8267
IPF	0.8455	**0.9092**	0.7922	0.8680	0.7164	–	0.7915	0.8694
NCNEdit	0.8313	0.7644	0.8462	0.7897	0.8120	0.8333	–	**0.8487**
RNG	0.7959	0.7988	0.8069	**0.8251**	0.7538	0.8128	0.8130	–
Mean	0.8173	0.8269	0.8103	0.8491	0.7766	0.8488	0.8196	**0.8786**

Table 9
Rule set chosen to predict the noise filtering efficacy.

Rule	F2	N2	T1	F3	Filter
R1	≤ 0.439587	≤ 0.264200	≤ 0.995100		Positive
R2	≤ 0.439587	≤ 0.264200	> 0.995100		Negative
R3	≤ 0.439587	(0.2642, 0.419400]			Negative
R4	≤ 0.439587	> 0.419400		≤ 0.101900	Positive
R5	≤ 0.439587	> 0.419400		> 0.101900	Negative
R6	> 0.439587		≤ 0.985400		Positive
R7	> 0.439587	≤ 0.298600	(0.985400, 0.994900]		Positive
R8	> 0.439587	(0.298600, 0.344700]	(0.985400, 0.994900]		Negative
R9	> 0.439587	≤ 0.344700	> 0.994900		Negative
R10	> 0.439587	(0.344700, 0.836984]	(0.985400, 0.996005]		Positive
R11	> 0.439587	(0.344700, 0.515300]	> 0.996005	≤ 0.294916	Negative
R12	> 0.439587	(0.515300, 0.836984]	> 0.996005	≤ 0.294916	Positive
R13	> 0.439587	(0.344700, 0.836984]	> 0.996005	> 0.294916	Negative
R14	> 0.439587	> 0.836984	> 0.985400	≤ 0.011076	Negative
R15	> 0.439587	> 0.836984	> 0.985400	> 0.011076	Positive

Table 10
Analysis of the behavior of the chosen rule set, which comes from the RNG filter, with all the noise filters.

Rule	CF		CVCF		EF		ENN		ENNTh		IPF		NCNEdit	
	Cov	Acc	Cov	Acc	Cov	Acc	Cov	Acc	Cov	Acc	Cov	Acc	Cov	Acc
R1	4.05	100.00	6.47	100.00	3.24	100.00	4.46	100.00	3.72	66.67	5.58	100.00	3.50	85.71
R2	1.21	33.33	1.29	33.33	0.46	0.00	1.34	33.33	1.24	100.00	1.20	33.33	1.00	50.00
R3	1.21	33.33	1.72	25.00	0.46	100.00	1.79	75.00	2.48	100.00	1.59	25.00	2.50	100.00
R4	3.64	100.00	3.02	100.00	2.31	100.00	1.34	100.00	1.65	25.00	3.19	100.00	2.00	75.00
R5	12.96	75.00	8.19	57.89	11.11	62.50	18.30	75.61	23.14	85.71	11.95	60.00	21.00	80.95
R6	38.06	98.94	42.67	100.00	42.13	98.90	38.84	97.70	35.95	94.25	41.04	98.06	34.00	97.06
R7	3.24	100.00	3.02	100.00	2.78	100.00	2.68	100.00	2.48	100.00	1.99	100.00	2.00	100.00
R8	0.40	0.00	0.86	0.00	0.46	0.00	0.89	0.00	0.83	0.00	1.20	0.00	1.00	0.00
R9	4.05	10.00	3.45	25.00	3.70	0.00	5.80	69.23	4.55	18.18	3.19	25.00	4.00	12.50
R10	14.98	100.00	14.66	100.00	15.28	100.00	12.95	96.55	12.40	93.33	14.74	100.00	11.00	90.91
R11	0.81	50.00	0.86	50.00	0.93	0.00	1.34	100.00	2.07	40.00	1.20	33.33	0.50	0.00
R12	6.48	100.00	7.76	94.44	7.87	94.12	4.46	90.00	4.13	90.00	6.37	93.75	8.00	93.75
R13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
R14	3.64	55.56	2.16	20.00	3.70	50.00	2.68	50.00	2.89	42.86	2.79	42.86	4.50	66.67
R15	4.45	100.00	3.88	100.00	5.56	100.00	2.68	83.33	1.65	75.00	2.79	100.00	4.00	87.50

From the analysis of these three rules, which are the most representative, it can be concluded that a high value of F2 generally leads to a statistical improvement in the results of the nearest neighbor classifier if a noise filter is used. If the classification problem is rather simple, with a lower value of F2, the application of a noise filter is generally not necessary. The high values of the measure N2 in the rule R5 reflects the fact that the examples of the same class are dispersed. Thus, when dealing with complex problems with high degrees of overlapping, filtering can improve the classification performance. However, if the problem is rather simple, with low degrees of overlapping, and moreover the examples of the same class are dispersed, e.g., if there are many clusters with low overlapping among them, noise

Table 11
Base datasets used for the validation phase.

Dataset	#INS	#ATT (R/I/N)	#CLA
abalone	4174	8 (7/0/1)	28
breast	277	9 (0/0/9)	2
dermatology	358	34 (0/34/0)	6
german	1000	20 (0/7/13)	2
page-blocks	5472	10 (4/6/0)	5
phoneme	5404	5 (5/0/0)	2
satimage	6435	36 (0/36/0)	7
segment	2310	19 (19/0/0)	7
vehicle	846	18 (0/18/0)	4
vowel	990	13 (10/3/0)	11

Table 12
Ranks obtained applying the final rule set ($R+$) and the indiscriminate usage of the filter ($R-$).

Dataset	CF	CVCF	EF	ENN	ENNTh	IPF	NCNEdit	RNG
$R+$	32 132.5	26 865.5	28 103.5	33 297.5	37 238.0	30 871.5	31 497.5	30 718.5
$R-$	13 017.5	17 984.5	16 746.5	11 552.5	7612.0	14 278.5	13 352.5	14 431.5
p-**Value**	0.000001	0.002265	0.000089	0.000001	0.000001	0.000001	0.000001	0.000001

filtering is not usually necessary—since the filtering may remove any of those clusters and be detrimental to the test performance.

5.5. Validation of the chosen rule set

In order to validate the usefulness of the rule set provided in the previous section to discern when to apply a noise filter to a concrete dataset, an additional experimentation has been prepared considering the 10 datasets shown in Table 11. From these datasets, another 300 binary ones have been created in the same way as explained in Section 4, but increasing the noise levels up to 25%.

For each noise filter, the test performance of 1-NN is computed for these datasets in two different cases:

1. Indiscriminately applying the noise filter to each training dataset.
2. Applying the noise filter to a training dataset only if the rule set of Section 5.4 so indicates. Concretely, the rule set indicates that noise filters must be applied in a 56% of the cases.

Then, the test results of both cases are compared using Wilcoxon's test. Table 12 shows the ranks obtained by case 1 ($R-$) and case 2 ($R+$) along with the corresponding p-values.

The results of this table show that, with some noise filters such as ENNTh and ENN, the advantage of using the rule set is more accentuated, whereas with others, such as CVCF and EF, this difference is less remarkable. However, very low p-values have been obtained in all the comparisons, which implies that the usage of the rule set to predict when to apply filtering is clearly positive with all the noise filters considered. Therefore, the conclusions obtained in the previous sections are maintained in this validation phase, even though a wider range of noise levels have been considered in the latter.

6. Concluding remarks

This paper has studied to what extent noise filtering efficacy can be predicted using data complexity measures when the nearest neighbor classifier is employed. A methodology to extract a rule set based on data complexity measures to predict in advance when a noise filter will statistically improve the results has been provided.

The results obtained have shown that there is a notable relation between the characteristics of the data and the efficacy of several noise filters, as the rule sets have good prediction performances. The most influential metrics are F2, N2, F3 and T1. Moreover, a single rule set has been proposed and tested to predict the noise filtering efficacy of all the noise filters, providing a good prediction performance. This shows that the conditions under which a noise filter works well are similar for other noise filters.

The analysis of the rule set provided shows that, generally, noise filtering statistically improves the classifier performance of the nearest neighbor classifier when dealing with problems with a high value of overlapping among the classes. However, if the problem has several clusters with a low overlapping among them, noise filtering is generally unnecessary and can indeed cause the classification performance to deteriorate.

This paper has focused on the prediction of noise filtering efficacy with the nearest neighbor classifier due it being perhaps the most noise-sensitive learner and then, the true filtering efficacy was checked. In future works, how noise filtering efficacy can be predicted for other classification algorithms with different noise-tolerance will be studied.

Acknowledgment

Supported by the Spanish Ministry of Science and Technology under Projects TIN2011-28488 and TIN2010-15055, and also by Regional Project P10-TIC-6858. J.A. Sáez holds an FPU Scholarship from the Spanish Ministry of Education and Science.

References

[1] X. Wu, X. Zhu, Mining with noise knowledge: error-aware data mining, IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans 38 (4) (2008) 917–932.
[2] D. Liu, Y. Yamashita, H. Ogawa, Pattern recognition in the presence of noise, Pattern Recognition 28 (7) (1995) 989–995.
[3] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study, Artificial Intelligence Review 22 (2004) 177–210.
[4] Y. Li, L.F.A. Wessels, D. de Ridder, M.J.T. Reinders, Classification in the presence of class noise using a probabilistic kernel Fisher method, Pattern Recognition 40 (12) (2007) 3349–3357.
[5] R. Kumar, V.K. Jayaraman, B.D. Kulkarni, An SVM classifier incorporating simultaneous noise reduction and feature selection: illustrative case examples, Pattern Recognition 38 (1) (2005) 41–49.
[6] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.
[7] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1967) 21–27.
[8] J. Toyama, M. Kudo, H. Imai, Probably correct k-nearest neighbor search in high dimensions, Pattern Recognition 43 (4) (2010) 1361–1372.
[9] Y. Liaw, M. Leou, C. Wu, Fast exact k nearest neighbors search using an orthogonal search tree, Pattern Recognition 43 (6) (2010) 2351–2358.
[10] I. Kononenko, M. Kukar, Machine Learning and Data Mining: Introduction to Principles and Algorithms, Horwood Publishing Limited, 2007.
[11] Y. Wu, K. Ianakiev, V. Govindaraju, Improved k-nearest neighbor classification, Pattern Recognition 35 (10) (2002) 2311–2318.
[12] D. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems, Man and Cybernetics 2 (3) (1972) 408–421.
[13] C. Brodley, M. Friedl, Identifying mislabeled training data, Journal of Artificial Intelligence Research 11 (1999) 131–167.
[14] X. Zhu, X. Wu, Q. Chen, Eliminating class noise in large datasets, in: Proceeding of the 20th International Conference on Machine Learning, 2003, pp. 920–927.
[15] M. Basu, T. Ho, Data Complexity in Pattern Recognition, Springer, Berlin, 2006.
[16] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 289–300.
[17] S. Singh, Multiresolution estimates of classification complexity, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (12) (2003) 1534–1539.
[18] R. Baumgartner, R.L. Somorjai, Data complexity assessment in undersampled classification, Pattern Recognition Letters 27 (2006) 1383–1389.
[19] A.C. Lorena, A.C.P.L.F. de Carvalho, Building binary-tree-based multiclass classifiers using separability measures, Neurocomputing 73 (2010) 2837–2845.
[20] A. Orriols-Puig, J. Casillas, Fuzzy knowledge representation study for incremental learning in data streams and classification problems, Soft Computing 15 (12) (2011) 2389–2414.

[21] A.C. Lorena, I.G. Costa, N. Spolar, M.C.P. de Souto, Analysis of complexity indices for classification problems: cancer gene expression data, Neurocomputing 75 (1) (2012) 33–42.
[22] O. Okun, H. Priisalu, Dataset complexity in gene expression based cancer classification using ensembles of k-nearest neighbors, Artificial Intelligence in Medicine 45 (2009) 151–162.
[23] J. Luengo, F. Herrera, Domains of competence of fuzzy rule based classification systems with data complexity measures: a case of study using a fuzzy hybrid genetic based machine learning method, Fuzzy Sets and Systems 161 (2010) 3–19.
[24] E. Bernadó-Mansilla, T.K. Ho, Domain of competence of XCS classifier system in complexity measurement space, IEEE Transactions on Evolutionary Computation 9 (1) (2005) 82–104.
[25] J.S. Sánchez, R.A. Mollineda, J.M. Sotoca, An analysis of how training data complexity affects the nearest neighbor classifiers, Pattern Analysis and Applications 10 (3) (2007) 189–201.
[26] J. Luengo, F. Herrera, Shared domains of competence of approximate learning models using measures of separability of classes, Information Sciences 185 (2012) 43–65.
[27] S. García, J.R. Cano, E. Bernadó-Mansilla, F. Herrera, Diagnose Effective Evolutionary Prototype Selection Using an Overlapping Measure, 2009.
[28] J. Luengo, A. Fernández, S. García, F. Herrera, Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling, Soft Computing—A Fusion of Foundations, Methodologies and Applications 15 (2011) 1909–1936.
[29] R.K.M. Dong, Feature subset selection using a new definition of classificability, Pattern Recognition Letters 24 (2003) 1215–1225.
[30] T.K. Ho, H.S. Baird, Pattern classification with compact distribution maps, Computer Vision and Image Understanding 70 (1) (1998) 101–110.
[31] F.W. Smith, Pattern classifier design by linear programming, IEEE Transactions on Computers 17 (4) (1968) 367–372.
[32] S.P. Smith, A.K. Jain, A test to determine the multivariate normality of a data set, IEEE Transactions on Pattern Analysis and Machine Intelligence 10 (5) (1988) 757–761.
[33] A. Hoekstra, R.P.W. Duin, On the nonlinearity of pattern classifiers, in: 13th International Conference on Pattern Recognition, 1996, pp. 271–275.
[34] F. Lebourgeois, H. Emptoz, Pretopological approach for supervised learning, in: 13th International Conference on Patter Recognition, 1996, pp. 256–260.
[35] J. Sánchez, F. Pla, F. Ferri, Prototype selection for the nearest neighbor rule through proximity graphs, Pattern Recognition Letters 18 (1997) 507–513.
[36] G. McLachlan, Discriminant Analysis and Statistical Pattern Recognition, John Wiley and Sons, 2004.
[37] D. Gamberger, N. Lavrac, C. Groselj, Experiments with noise filtering in a medical domain, in: 16th International Conference on Machine Learning (ICML99), 1999, pp. 143–151.
[38] S. Verbaeten, A. Assche, Ensemble methods for noise elimination in classification problems, in: 4th International Workshop on Multiple Classifier Systems (MCS 2003), Lecture Notes on Computer Science, vol. 2709, Springer, 2003, pp. 317–325.
[39] F. Vazquez, J. Sánchez, F. Pla, A stochastic approach to Wilsonś editing algorithm, in: 2nd Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA05), Lecture Notes on Computer Science, vol. 3523, Springer, 2005, pp. 35–42.
[40] T. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise filtering techniques, Journal of Computer Science and Technology 22 (2007) 387–396.
[41] J. Sánchez, R. Barandela, A. Márques, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recognition Letters 24 (2003) 1015–1022.
[42] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2–3) (2011) 255–287.
[43] A. Orriols-Puig, E. Bernadó-Mansilla, Evolutionary rule-based systems for imbalanced data sets, Soft Computing 13 (3) (2009) 213–225.
[44] N.A. Samsudin, A.P. Bradley, Nearest neighbour group-based classification, Pattern Recognition 43 (10) (2010) 3458–3467.
[45] I. Triguero, S. García, F. Herrera, Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification, Pattern Recognition 44 (4) (2011) 901–916.
[46] J. Huang, C.X. Ling, Using AUC and accuracy in evaluating learning algorithms, IEEE Transactions on Knowledge and Data Engineering 17 (3) (2005) 299–310.
[47] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[48] J.S. Sánchez, R. Barandela, A.I. Marqués, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recognition Letters 24 (7) (2003) 1015–1022.

José A. Sáez received his M.Sc. in Computer Science from the University of Granada, Granada, Spain, in 2009. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence in the University of Granada. His main research interests include noisy data in classification, discretization methods and imbalanced learning.

Julián Luengo received the M.S. degree in Computer Science and the Ph.D. degree from the University of Granada, Granada, Spain, in 2006 and 2011, respectively. His research interests include machine learning and data mining, data preparation in knowledge discovery and data mining, missing values, data complexity and fuzzy systems.

Francisco Herrera received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain.

He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has had more than 200 papers published in international journals. He is coauthor of the book ''Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases'' (World Scientific, 2001).

He currently acts as Editor in Chief of the international journal ''Progress in Artificial Intelligence'' (Springer) and serves as Area Editor of the Journal Soft Computing (area of evolutionary and bioinspired algorithms) and International Journal of Computational Intelligence Systems (area of information systems). He acts as Associated Editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Advances in Fuzzy Systems, and International Journal of Applied Metaheuristics Computing; and he serves as member of several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Knowledge and Information Systems, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, Swarm and Evolutionary Computation.

He received the following honors and awards: ECCAI Fellow 2009, 2010 Spanish National Award on Computer Science ARITMEL to the ''Spanish Engineer on Computer Science'', and International Cajastur ''Mamdani'' Prize for Soft Computing (Fourth Edition, 2010).

His current research interests include computing with words and decision making, data mining, bibliometrics, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.

3.2 INFFC: An iterative noise filter based on the fusion of classifiers with noise sensitiveness control

Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) INFFC: An iterative noise filter based on the fusion of classifiers with noise sensitiveness control (submitted)

- Status: **Submitted**.

INFFC: An iterative noise filter based on the fusion of classifiers with noise sensitiveness control

José A. Sáez[a,*], Mikel Galar[b], Julián Luengo[c], Francisco Herrera[a]

[a]*Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, Granada, Spain, 18071*
[b]*Department of Automática y Computación, Universidad Pública de Navarra, Pamplona, Spain, 31006*
[c]*Department of Civil Engineering, LSI, University of Burgos, Burgos, Spain, 09006*

Abstract

In classification, noise may deteriorate the system performance and increase the complexity of the models built. In order to mitigate its consequences, several approaches have been proposed in the literature. Among them, noise filtering, which removes noisy examples from the training data, is one of the most used. This paper proposes a new noise filtering method that combines several filtering strategies in order to increase the accuracy of the classification algorithms used later. The filtering is based on the fusion of the predictions of several classifiers used to detect the presence of noise. We translate the idea behind multiple classifier systems, where the information gathered from different models is combined, to noise filtering. In this way, we consider the combination of classifiers instead of using only one to detect noise. Additionally, the proposed method follows an iterative noise filtering scheme that allows us to avoid the usage of detected noisy examples in each new iteration of the filtering process. Finally, we introduce a noisy score to control the filtering sensitiveness, in such a way that the amount of noisy examples removed in each iteration can be adapted to the necessities of the practitioner. The first two strategies (use of multiple classifiers and iterative filtering) are used to improve the filtering accuracy, whereas the last one (the noisy score) controls the conservativeness of the filter removing potentially noisy examples. The validity of the proposed method is studied in an exhaustive experimental study. We compare the new filtering method against several state-of-the-art methods to deal with noisy datasets and study their efficacy in three classifiers with different sensitiveness to noise.

Keywords:

Ensembles, Fusion of Classifiers, Noisy Data, Class Noise, Noise Filters, Classification.

*Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
Email addresses: smja@decsai.ugr.es (José A. Sáez), mikel.galar@unavarra.es (Mikel Galar), jluengo@ubu.es (Julián Luengo), herrera@decsai.ugr.es (Francisco Herrera)

1. Introduction

Data collection and preparation processes are commonly subjected to errors in Data Mining applications [1, 2]. For this reason, real-world datasets usually contain imperfections or *noise* [3, 4, 5]. In classification, a model is built from labeled examples, which should be capable of reliably predicting the class for new previously unobserved examples. Obviously, if the data used to train this model (formally known as a classifier) are corrupted, both the learning phase and the model obtained will be negatively affected. The former will require more time to find a solution but also more examples in order to be able to obtain an accurate classifier. As a consequence, the final model will probably be less accurate due to the presence of noise, and it will be more complex, since non-real patterns may be modeled.

Two different types of noise can be found in classification datasets: attribute and class noise [3]. Class noise is the most disruptive type of noise since incorrectly labeled examples have a high impact when building classifiers, whose performance is often reduced [3, 6]. On this account, many works in the literature, including this paper, focus on its treatment [7, 8, 9, 10]. Among these works, two types of approaches have been proposed to deal with class noise [3]:

1. *Algorithm level approaches* [11, 12]. The methods in this category comprise the adaptations of existing algorithms to properly handle the noise or being less influenced by its presence.
2. *Data level approaches* [7, 8]. These methods consist of preprocessing the datasets aiming at getting rid of the noisy examples as a previous step.

Algorithm level approaches are not often an available choice since they depend on the particular adaptation of each classification algorithm, and therefore they are not directly extensible to other learning algorithms. Otherwise, data level approaches are independent of the classifier used and allow one to preprocess the datasets beforehand in order to use them to train different classifiers (hence, the computation time needed to prepare the data is only required once). Thus, the latter type of techniques is usually the most popular choice.

Among data level approaches, noise filters, which remove noisy examples from the training data, are widely used due to their benefits in the learning in terms of classification accuracy and complexity reduction of the models built [7, 13]. Even though several noise filtering schemes are proposed in the literature [7, 14, 15, 16], their study focuses our attention on three main paradigms:

- *Ensemble-based filtering* [7]. There are studies where some authors proposed the usage of ensembles for filtering. The main advantage of these approaches is based on the hypothesis

that collecting predictions from different classifiers could provide a better class noise detection than collecting information from a single classifier.

- *Iterative filtering* [17]. The strength of this type of filters is using an iterative elimination of noisy examples under the idea that the examples removed in one iteration do not influence the noise detection in subsequent ones.

- *Metric-based filtering* [14, 9]. These noise filters are based on the computation of measures over the training data and usually allow the practitioner to control the conservativeness level of the filtering in such a way that only examples whose estimated noise level exceed a prefixed threshold are removed.

On this account, this paper proposes a novel noise filtering technique combining these three noise filtering paradigms: the usage of ensembles for filtering, the iterative filtering and the computation of noise measures. The proposal of this paper removes noisy examples in multiple iterations considering filtering techniques that employ systems based on the *Fusion of Classifiers* (FC), also known as *Multiple Classifier Systems* (MCSs) [18, 19, 20]. This type of systems have already shown a good behavior with noisy data in the field of classification [21, 6]. Besides, our filtering proposal uses a noise score computed over each potentially noisy example in order to determine which of them are finally eliminated in each iteration. In this way, we take advantage of the three different paradigms. The proposed method is called *Iterative Noise Filter based on the Fusion of Classifiers* (INFFC).

A thorough empirical study will be developed comparing several representative noise filters with our proposal. All of them will be used to preprocess 25 real-world datasets, into which different class noise levels will be introduced (from 5% to 30%, by increments of 5%). The filtered datasets will be then used to create classifiers with three learning methods of a different and well-known behavior against noise: a learner considered robust to noise as C4.5 [11] is, a *Support Vector Machine* (SVM) [22], considered accurate but being noise sensitive and the Nearest Neighbor rule [23] which is also considered very noise sensitive. Their test accuracy over the datasets preprocessed with our proposal and the other existing filters will be compared using the appropriate statistical tests [24] in order to check the significance of the differences found. Full results and details of the experimentations are available in the webpage associated to this paper at `http://sci2s.ugr.es/INFFC`.

The rest of this paper is organized as follows. Section 2 presents an introduction to classification with noisy data. In Section 3 we introduce the details of the noise filter proposed. In Section 4 we describe the experimental framework, whereas in Section 5 we analyze the results obtained. Finally, in Section 6 we enumerate some concluding remarks.

2. Classification with noisy data

This section first introduces the problem of noisy data in the classification framework in Section 2.1. Next, previous works on noise filters are briefly reviewed in Section 2.2, paying special attention to those filters on which our proposal is based.

2.1. Noise in classification problems

Noise is anything that obscures the relationship between the attributes of an example and its class [25]. For example, noise may be present as errors in the source and input of the data affecting the quality of the dataset [26]. In classification, this quality is mainly influenced by two information sources, that is, the class labeling and the attributes' value sampling. Based on these two information sources, two types of noise are traditionally distinguished in the literature [3]: attribute noise and class noise.

Attribute noise affects the values of one or more attributes of examples in a dataset. It can proceed from several sources, such as transmission constraints, faults in sensor devices and transcription errors [27]. *Class noise* (or labeling errors) is produced when the examples are labeled with the wrong classes. Class noise is recognized to be more harmful than attribute noise to classifier performance mainly due to the fact that whereas the importance of each feature for learning may be different, labels always have a large impact on learning [3, 21, 6]. For this reason, this paper focuses on class noise, aiming at removing those wrongly labeled examples from the datasets.

Class noise can be attributed to several causes [25]. One of them is the inadequacy of the information used to label each example, for example, when a amnesic patient imprecisely answers the questions of the doctor [28]. Data entry errors and the subjectivity during the labeling process also can produce class noise; for example, in medical applications a variability in the labeling by several experts may exist [29].

Among the effects of class noise in the system performance, the most frequently reported consequence is the decrement of classification accuracy [6]. Class noise can also affect the complexity of the classifier built in terms of size and interpretability (for example, in [7], it is shown how the size of decision trees increases when class noise is present).

Errors in real-world datasets are therefore common and techniques that eliminate noise or reduce its impact are need [3]. Two main alternatives have been proposed in the literature to deal with noisy data:

- *Algorithm level approaches.* Also known as *robust learners*, these are techniques characterized by being less influenced by noisy data. Examples of a robust learner are C4.5

[11] or RIPPER [12]. These classifiers have been adapted to properly handle the noise. Thus, for example, C4.5 uses pruning strategies to reduce the chances that the trees are overfitting due to noise in the training data [30].

- *Data level approaches.* The most well-known type of methods within this group is that of *noise filters* [7, 17]. They identify noisy instances which can be eliminated from the training data. These methods are used with many learners that are sensitive to noisy data and require data preprocessing to address the problem, even though robust learners also benefit from their usage. The separation of noise detection and learning phase has the advantage of avoiding the usage of noisy instances in the classifier building process [14]. Our proposal is included into this group of methods.

2.2. Noise filters

Preprocessing the dataset aiming to clean the noisy examples is one of the most common approaches when training data are affected by noise [7]. Noise filters are designed to eliminate noisy examples in the training set, which is then used as an input to classifiers [7, 17, 5]. They are particularly oriented to remove class noise, since the elimination of such examples has shown to be advantageous [8]. However, the elimination of instances with attribute noise seems to be counterproductive [30, 3], since they still contain valuable information in other attributes which can help to build a more accurate classifier.

Even though several noise filtering schemes are proposed in the literature [7, 14, 15, 16], the following three groups have interesting bases as their foundaments:

- *Ensemble-based filtering* [7]. They are based on using several classifiers to detect the noisy examples, which are those mislabeled by a part of these classifiers.
- *Iterative filtering* [17]. They are based on removing the noisy examples iteratively. In this way, they try to avoid the fact that noisy examples may influence the detection of noise in other examples.
- *Metric-based filtering* [14]. Their main characteristic is that they allow one to easily control the conservativeness level of the filtering.

Even though metric-based filtering methods has the advantage of controlling the conservativeness level of the filtering, they are generally simple approaches and do not perform as well as other more advanced types of noise filters in many cases. For this reason, many other approaches have been proposed in the literature [9, 10, 8, 7].

There are filtering methods that are based on the fact that the k-NN classifier [23] is sensitive to noisy data [9, 10], particularly when k is low [31]. Other type of noise filters uses classifiers to detect the noisy examples, which are those that are misclassified. The *Classification Filter* (CF) [8] performs a partitioning of the training set into n subsets, then a set of classifiers is trained from the union of any $n-1$ subsets; those classifiers are afterwards used to classify the examples in the excluded subset, eliminating the incorrectly classified examples. This filter has the risk of removing too many instances due to the usage of a single classifier. In order to solve this problem, ensembles of classifiers are used to identify mislabeled instances; the proposals of [7, 17] are two of the most representative and well known methods within this field (described hereafter).

The *Ensemble Filter* (EF) [7] uses a set of three learning algorithms (C4.5 [11], 1-NN [23] and LDA [32]) to remove the potentially noisy instances. The training data is classified using an n-fold cross-validation with each classification algorithm and the noisy examples are identified using a voting scheme. Two voting schemes are proposed: consensus (which removes an example if it is misclassified by all the classifiers) and majority (which removes an example if it is misclassified by more than a half of the classifiers).

The *Iterative-Partitioning Filter* (IPF) [17] proposes a similar technique, but removes the noisy data iteratively using several classifiers built with the same learning algorithm. IPF removes noisy examples in multiple iterations until the quantity of examples eliminated is under a threshold. In each iteration, the current training dataset is split into n equal sized subsets and the C4.5 classifier is built over each of these n subsets to evaluate the whole training set. Then, the incorrectly labeled examples are removed from it (according to one of the two aforementioned voting schemes) and a new iteration is started.

Both methods, EF and IPF, claimed two important postulates in the field of the filtering techniques:

1. Brodley and Friedl (EF) [7] stated that if some examples have been mislabeled and it is assumed that the label errors are independent of the particular classifier learned from the data, collecting predictions from different classifiers could provide a better estimation of mislabeled examples than collecting information from a single classifier.
2. Khoshgoftaar and Rebours (IPF) [17] claimed that a iterative elimination of noisy examples implies that the examples removed in one iteration do not influence the detection in subsequent ones, resulting in a more accurate noise filtering.

However, the methods belonging to these approaches have some drawbacks. Since EF, which is an ensemble-based filter, does not follow an iterative elimination of the noisy examples,

the classifiers are built from wrong data, and therefore their noise detection may be biased. Otherwise, the IPF iterative noise filter only considers one classification algorithm to build the filter, and thus it does not benefit from collecting information from different models built with different classification algorithms.

In this work, we aim to follow these postulates at the same time, combining both types of mechanisms to develop a new filtering method. In addition, we also take into account different measures of noise in each potentially noise example in order to decide whether it should be removed or not. In this way, we take advantage of the strengths of each type of filtering method to create a new noise filter. The complete proposal is explained in the next section.

3. The Iterative Noise Filter based on the Fusion of Classifiers

Inspired by the ideas that motivated the design of the EF and IPF filters, the proposal of this paper removes noisy examples in multiple iterations considering filtering techniques based on the usage of multiple classifiers [18, 21, 33]. For this reason, we have called our proposal *Iterative Noise Filter based on the Fusion of Classifiers*. Figure 1 shows an scheme of the filtering method proposed. Three steps are carried out in each iteration. First, a preliminary filtering is performed with a FC-based filter. Then, another FC-based filter is built from the examples that are not identified as noisy in the preliminary filtering to detect the noisy examples in the full set of instances in the current iteration. Finally, noisy examples are only removed if they exceed a noise score metric.

The building of the FC-based filter is described in Section 3.1. Then, the three main steps carried out at each iteration are described in separate sections:

1. **Preliminary filtering** (Section 3.2). This first step removes a part of the existing noise in the current iteration to reduce its influence in posterior steps. More specifically, noise examples identified with high confidence are expected to be removed in this step.
2. **Noise-free filtering** (Section 3.3). A new filtering, which is built from the partially clean data from the previous step, is applied over all the training examples in the current iteration resulting into two sets of examples: a clean and a noisy set. This filtering is expected to be more accurate than the previous one since the noise filters are built from cleaner data.
3. **Final removal of noise** (Section 3.4). A noise score is computed over each potentially noisy example from the noisy set obtained in the previous step in order to determine which of them are finally eliminated.

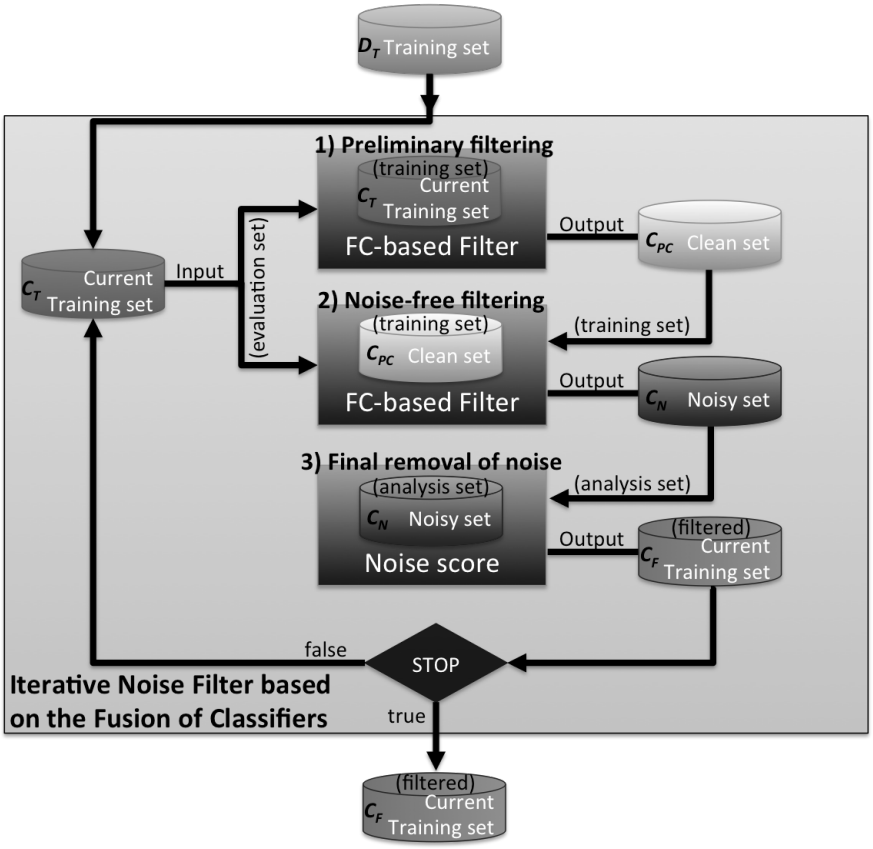


Figure 1: Filtering method proposed: Iterative Noise Filter based on the Fusion of Classifiers.

We introduce these three steps in each iteration aiming at removing noisy examples with maximum reliability, that is, those which are noisy examples with great confidence. In this way, examples that may be noisy or not, are leaved in the training set for posterior processing. Ensuring that only the examples that are most likely to be noise are removed implies that it will be less probable to delete noise-free examples, which would harm the learning process.

Finally, the iterative process stops when, for a number of consecutive iterations k, the number of identified noisy examples in each of these iterations k is less than a percentage p of the size of the original training dataset (in this paper, these parameters are set to $k = 3$ iterations and $p = 1\%$).

Table 1 describes the notation used in the following sections to describe the different sets of examples considered in our filter.

3.1. Noise filter based on the fusion of classifiers

The filtering technique used in the steps 2 and 3 in each iteration of the method proposed in this paper is based on the combination of different classifiers. This filtering strategy was

Table 1: Sets of examples used in the proposed noise filter.

Set	Description
D_T	Initial training set
C_T	Training set at the start of the iteration
C_{PC} and C_{PN}	Sets of clean and noisy examples from C_T provided by the preliminary filtering
C_C and C_N	Sets of clean and noisy examples from C_T provided by the noise-free filtering
C_F	Filtered C_T

previously used in the EF noise filter [7]. However, there are two main differences of our filtering technique with respect to EF:

1. *Noise evaluation strategy.* The authors of EF proposed the usage of a k-fold cross-validation to label each example as correct or noisy by each classifier, whereas in our proposal all the training examples are considered to create only one model with each classifier to label the examples. Thus, the time complexity of the method is reduced.
2. *Classifiers used to build the filter.* The authors of EF propose to use C4.5 [11], 1-NN [23] and LDA [32], whereas we propose to change 1-NN by 3-NN and LDA by Logistic regression (LOG) [32], respectively. These changes are mainly motivated by the noise evaluation strategy used. Since our FC-based filter only builds one model to label each example as clean or noisy, it needs to include classifiers that behave better with noisy data than the very noise-sensitive 1-NN and LDA methods used in EF. Because of this, we increment the k value of the k-NN method, increasing its robustness against noise [31]. On the other hand, LOG is an statistical classifier, such as LDA, but is recognized to behave better in several domains. In this way, we improve the global behavior of the filter when detecting noisy examples since 3-NN and LOG could be considered better than LDA and 1-NN when dealing with noisy data.

In the following, the classifiers used by our FC-based filter are briefly described along with their respective noise-tolerance:

- **C4.5 decision tree generator** [11]. C4.5 constructs a decision tree in a top-down way, using the normalized information gain (difference in entropy) that results from choosing an attribute to split the data. It is considered a robust learner, which uses pruning strategies to reduce the chances of classifiers being affected by noisy examples [30].
- ***k*-Nearest Neighbors** [23]. This finds a group of k examples in the training set that

are closest to the test pattern. The predicted class label is based on the predominance of a particular class in this neighborhood. The value of k determines a higher or lower sensitivity of k-NN to noise [31].

- **Logistic regression** [32]. This method is a discriminative probabilistic classification model based on the logistic function. Due to its probabilistic nature, one can expect that it behaves relatively well with low noise levels but it performs worse when the noise level increases.

After obtaining the prediction of each one of the aforementioned classifiers over the set of examples to evaluate, those examples incorrectly classified by the majority of the classifiers, that is, 2 of the 3 classifiers, are labeled as noisy. It is important to note that we can also use the consensus scheme to determine which examples are noisy, but in this paper we will use the majority scheme in all the ensemble-based filters, making the results comparable. Furthermore, remember that in our proposal we use the same data to train each classifier and evaluate them to determine the examples to be tagged as noisy.

3.2. Preliminary filtering

Data that we want to filter C_T (note that $C_T = D_T$ at the first iteration) are, logically, likely to contain noisy examples. Therefore, filtering based on these noisy data may be misleading since the filtering models built are affected by the noisy examples. Thus, these data are not reliable enough to decide the final elimination of the noisy examples. On this account, we first perform a preliminary filtering of the dataset C_T at each iteration in order to remove most of the potentially noisy examples. Afterwards, we consider a second filtering step, the *noise-free filtering*, in which the filter is trained only with the examples considered as noise-free (C_{PC}) at this stage, and thus its noise identification is expected to be more reliable.

The filtering consists of the creation of a system based on the fusion of classifiers considering the three aforementioned classifiers (C4.5, 3-NN and LOG) from C_T (the training set at the start of the iteration). This FC-based filter is used to evaluate the examples of the same set C_T. The noisy examples C_{PN} identified by the filter are removed from C_T, resulting in the training data C_{PC}.

Please, note that the noise identification of this first filtering is based on data which may contain noise. Thus, the noise identified in this step may be erroneous and noisy data will be probably considered as clean data and vice versa; this is the reason why we call this step *preliminary filtering*.

3.3. Noise-free filtering

The filtered dataset provided by the preliminary filtering (C_{PC}) is a cleaner version than the training set at the start of the iteration C_T. Therefore, the FC-based filter built from these cleaner data C_{PC} is expected to perform a more accurate identification of the noise in C_T, since the models built for filtering are not affected by the most disruptive noisy examples previously detected and removed from C_T.

Therefore, in the second step of each iteration, a new FC-based filter is created considering C_{PC}, that is, C_T after the preliminary filtering (without the noisy examples identified in the first step). This filter is evaluated on the examples of the whole set C_T (all the training examples in the current iteration). This results in two different sets of examples: $C_C \subseteq C_T$, which consists of those examples considered as clean by the filter and $C_N \subseteq C_T$, which is the set of examples considered as potentially noisy (note that $C_C \cap C_N = \emptyset$ and $C_C \cup C_N = C_T$).

Note that the noise identification carried out by the FC-based filter in this step is based on the filtered C_T, so it is expected to be more accurate in identifying noise that the filter in the previous step; this is the reason why we call this step *noise-free filtering*.

3.4. Final removal of noise: the noise score

This last step controls the noise sensitiveness of the filter moderating the amount of noisy examples removed. By means of this last step, we try to ensure that only true noisy examples are removed. Hence, questionable examples are analyzed in posterior iterations, since the elimination of non-noisy examples may carry a significant decrease in accuracy and therefore it is important to be sure that they are truly noise.

The noisy examples identified in the second step of the iteration C_N are those considered to analyze with the noise score. They are ordered according to this noise score, from those which are more probably noise to those which are less probably noise or that may be indeed clean examples wrongly identified as noisy by the filter. Finally, the examples that exceed a threshold set by the user are eliminated (the effects of different values of the threshold on the filtering will be studied in Section 5.6). In order to define the noise score, we have made the following assumptions:

Assuption 1. *The class label of some training examples may be erroneous.*

Any dataset is susceptible of containing noise [3]. Since our proposal is particularly designed to deal with datasets with class noise, one cannot blindly trust on the class of all the examples.

Assuption 2. *Noisy examples detected by any filter may be incorrect.*

From the above premise, decisions obtained from noisy data may be also incorrect. In our proposal, this assumption is related to the decisions made by the noise filter in which the examples that are noisy are presented. Therefore, the set of noisy examples detected in the second step at each iteration, that is, the set of noisy examples to analyze with the noise score, need not be correct. Thus, examples labeled as noise might be clean and vice versa.

Assuption 3. *Examples in noisy clusters are less reliable.*

The information obtained from a cluster of noisy examples, that is, an agglomeration of noisy examples, is less reliable. Our proposal will be tested with several percentages of examples with class noise in the training set, more specifically up to 30% (see Section 4.1). Therefore, it is very likely that clusters of noisy examples will be created, particularly for the higher noise levels, in which a higher quantity of examples are corrupted. In this scenario, an example that was labeled as noise by the filter may be clean (clean examples within the cluster may be labeled as noise), and vice versa. The same occurs with the class labels: it is clear that in a cluster of noisy examples, most of them would have the class label incorrectly assigned. Therefore, information coming from these clusters should be taken cautiously (with less confidence).

Assuption 4. *The presence of examples with different class labels in the neighborhood of an example may indicate that it is a noisy example.*

The more examples in the neighborhood (k nearest neighbors) of an example e have their class label different to that of the example e, the more likely for this example e to be noisy is. It will be even more likely for e to be noisy if, in addition, its nearest neighbors have been labeled as clean examples by the noise filter.

Assuption 5. *The presence of examples with the same class label in the neighborhood of an example may indicate that it is a clean example.*

The more examples in the neighborhood (k nearest neighbors) of an example e have the same class label to that of the example e, the more likely for the example e to be clean is. It will be even more likely for e to be clean if, in addition, its nearest neighbors have been labeled as clean examples by the noise filter.

On account of these assumptions, we will consider the following information (provided by the examples from C_T) in order to set the confidence of an example $e \in C_N$ labeled as noisy to be noisy:

1. **Detection results of the noise-free FC-based filter** (related to Assuptions 1 and 2): each example is labeled as *clean* or *noisy* by the FC-based filter constructed in the second step of the method.

2. **Information of each training example as belonging to the neighborhood of other noisy examples** (related to Assuption 3): the times that one example is among the k nearest neighbors of other examples labeled as noisy in C_N (denoted as $t(e)$). This value provides an idea of how involved is an example in noisy areas (clusters with noise). If the value is high, it means that this example is among the nearest neighbors of many other examples that may have noise.
3. **Information of the neighborhood of each example *e*** (related to Assuptions 4 and 5): the classes of e and the examples close to e, that is, its k nearest neighbors ($k = 5$ is considered in this paper).

Based on Assumption 3, we have defined the function *confidence*(e) (see Equation 1). It checks whether the example e is close to other noisy examples. Figure 2 shows the plot of the function *confidence* with respect to $t(e)$, the number of times that e is present in the neighborhood of other noisy examples. As it can be observed in this figure, the function *confidence* returns values in the interval $(0, 1]$. The value of *confidence*(e) is higher if e is not in the neighborhood of other noisy examples, whereas it is lower if e is in the neighborhood of several noisy examples. Hence, if *confidence*$(e) = 1$ (when e is not among the nearest neighbors of any noisy example), the information that this example provides (such as its class label and its label *clean* or *noise* given by the FC-based filter) is very reliable. However, if *confidence*$(e) \approx 0$ (when e is among the nearest neighbors of many noisy examples), the information that it provides must not be taken into account.

$$\textit{confidence}(e) = \frac{1}{\sqrt{1 + t(e)^2}} \tag{1}$$

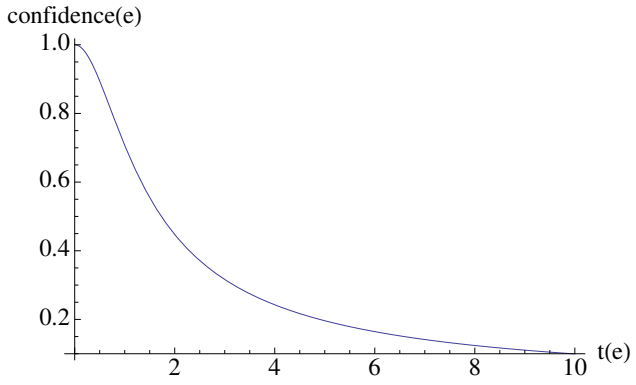


Figure 2: Graphical representation of the function *confidence*(e).

Similarly, based on Assumptions 3-5, we have defined the function *neighborhood*(e) (Equation 2). Its aim is to analyze the neighborhood of a given example e (its k nearest neighbors)

to determine the degree of e being in a noisy cluster.

$$neighborhood(e) = \frac{\sum_{i=1}^{k} clean(e_i) \cdot confidence(e_i) \cdot differentClasses(e, e_i)}{k} \tag{2}$$

This function computes an average value of the k nearest neighbors considering their classes (function $differentClasses(e, e_i)$), the degree of the cleanness of each neighbor of e (function $clean(e_i)$) and how can we rely on each neighbor (function $confidence(e_i)$).

The function $differentClasses(e_1, e_2)$, defined in Equation 3, takes into account Assumptions 4 (different classes increase the noise score) and 5 (coincident classes reduce the noise score). In such away, if the example e and its neighbor e_i have different class labels the value of $neighborhood(e)$ increases, whereas if they have the same class label the value of $neighborhood(e)$ is reduced.

$$differentClasses(e_1, e_2) = \begin{cases} 1, & if\ class(e_1) \neq class(e_2) \\ -1, & if\ class(e_1) = class(e_2) \end{cases} \tag{3}$$

Furthermore, Assumptions 4 and 5 state that *clean* examples must have a higher weight than examples with *noise* in the computation of the noise score. Thus, the function $clean(e_i)$ is defined based on the consideration that the number of noisy examples surrounding a given example e_i (that is, $n(e_i)$) is an indicator of the cleanness of that example - see Table 2 in order to check the difference between the functions $t(e)$ and $n(e)$. Thus, clean examples surrounded by many clean examples have an higher degree of cleanness than clean examples surrounded by noisy examples, since one must not trust in the information provided from areas with many noisy examples (Assumption 3). The same occurs with the noisy examples: if a noisy example is surrounded by many other noisy examples, one can say that this example has a lower degree of noisiness than other noisy example surrounded by clean examples, which is placed in a more reliable area. For these reasons, the function $clean(e_i)$ is defined as follows:

$$clean(e_i) = \frac{k + isnoise(e_i) \cdot (n(e_i) - k)}{2k}, \qquad isnoise(e_i) = \begin{cases} 1, & if e_i\ is\ noise \\ -1, & if e_i\ is\ clean \end{cases} \tag{4}$$

The function $isnoise(e_i)$, used by $clean(e_i)$, simply returns 1 if e_i is a *noisy* example and -1 if it is *clean*. Recall that the belonging of each example to the *clean* and *noise* sets is determined by the FC-based filter used in the second step of the filtering.

Thus, the function $clean(e)$ (see Figure 3) provides a value of the cleanness of the example e (not only if it is *clean* or *noise* as it is performed by the FC-based filter). The motivation for its usage is that some examples are expected to have a higher degree of confidence to be *clean* than other ones, and the same occurs with the noisy examples. It considers the fact that a

Table 2: Difference between the functions $t(e)$ and $n(e)$.

Function	Description
$t(e)$	Number of times that e is among the k nearest neighbors of other noisy examples in C_N
$n(e)$	Number of noisy examples in C_N among the k nearest neighbors of the example e

clean example is more reliable than an example with *noise* (Assumptions 4 and 5). As Figure 3 shows, this function returns values in the interval $[0, 1]$, being 0 the value corresponding to an example with a maximum degree of being noise and 1 the value corresponding to an example having a maximum degree of being clean. Concretely, the interval $[0, 0.5]$ is that of the values of noisy examples (0 for a noisy example in a cluster of clean examples and 0.5 for a noisy example in a cluster of noisy examples), whereas the interval $[0.5, 1]$ is that of the values of clean examples (0.5 for a clean example in a cluster of noisy ones and 1 for a clean example in a clean cluster). Thus, a higher value of cleanness is assigned to clean examples (considered by the FC-based filter) than to noisy examples.

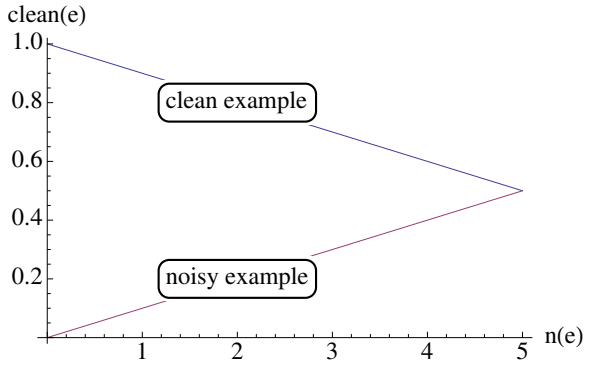


Figure 3: Graphical representation of the function *clean*(e). The value of *clean*(e) depends on the label of e (*clean* or *noisy*) provided by the noise-free filtering (the second step in each iteration).

Finally, the computation of the noise score *NS* for an example $e \in C_N$ is mainly based on the analysis of its neighborhood, represented by *neighborhood*(e), and this value is weighted by the reliability of the own example e, represented by *confidence*(e). Thus, both functions are combined to define the noise score $NS(e)$ as follows:

$$NS(e) = \textit{confidence}(e) \cdot \textit{neighborhood}(e) \tag{5}$$

As we have previously commented, the function *confidence*(e) is defined in the interval

$(0, 1]$, whereas the function *neighborhood*(e) is defined in $[-1, 1]$. Therefore, *NS* is defined in the interval $[-1, 1]$, being higher if the example e is more likely to have noise. The sign of the result provided by the function *neighborhood*(e) defines if the example e is in fact *clean* (negative values) or *noisy* (positive values), whereas its absolute value defines the degree of confidence of this choice (being -1 the value corresponding if the example e is totally clean and 1 if the example e is probably noise). A value $NS(e) = 0$ implies that there is not reliable information about if the example e is clean or noise. On the other hand, the function *confidence*(e) is another factor that establishes how representative the result provided by *neighborhood*(e) is, based on the degree of membership of e to noisy clusters.

After calculating the noise score for each potential noise example in C_N, those examples with a noise score higher than a threshold set by the user are removed. In the experimentation carried out in this paper, this threshold is by default fixed to 0. Thus, those examples in which there is any indication that they are noisy because their $NS(e) > 0$ are deleted. However, we also study the behavior of proposed filter with other different values of the threshold in Section 5, showing that it is robust with respect to this value.

4. Experimental framework

This section presents the details of the experimental study carried out in order to check the validity of the proposed noise filter. First, Section 4.1 describes the datasets used. Then, Section 4.2 shows the parameter setup for the classification algorithms used in the FC-based filter. Section 4.3 presents the noise filters compared with our proposal. Finally, Section 4.4 describes the methodology followed to analyze the results.

4.1. Datasets

The experimentation is based on the 25 datasets from the KEEL-Dataset repository [34] shown in Table 3, where #EX refers to the number of examples, #AT to the number of attributes and #CL to the number of classes. Some of the largest datasets (*penbased*, *satimage*, *shuttle* and *splice*) were stratified at 10% in order to reduce the computational time required for training, given the large amount of executions carried out. Examples containing missing values are removed from the datasets before their usage.

In order to control the amount of noise in each dataset, different noise levels $x\%$ are introduced into each training dataset in a supervised manner following an *uniform class noise scheme* [35]: $x\%$ of the examples are corrupted randomly replacing the class labels of these examples by other ones from the set of classes. We will consider the noise levels ranging from

Table 3: Base datasets used in the experimentation.

Dataset	#EX	#AT	#CL	Dataset	#EX	#AT	#CL
automobile	159	25(15/10)	6	monk	432	6(6/0)	2
balance	625	4(4/0)	3	new-thyroid	215	5(5/0)	3
banana	5300	2(2/0)	2	penbased	1099	16(16/0)	10
car	1728	6(0/6)	4	pima	768	8(8/0)	2
cleveland	297	13(13/0)	5	satimage	643	36(36/0)	7
contraceptive	1473	9(9/0)	3	shuttle	2175	9(9/0)	7
dermatology	358	33(1/32)	6	splice	319	60(0/60)	3
ecoli	336	7(7/0)	8	twonorm	7400	20(20/0)	2
flare	1066	11(0/11)	6	vehicle	846	18(18/0)	4
german	1000	20(13/7)	2	wdbc	569	30(30/0)	2
glass	214	9(9/0)	7	yeast	1484	8(8/0)	10
ionosphere	351	33(33/0)	2	zoo	101	16(0/16)	7
iris	150	4(4/0)	3				

$x = 0\%$ (base datasets) to $x = 30\%$, by increments of 5%. As a consequence, 150 noisy datasets with class noise are created from the aforementioned 25 base datasets (a total of 175 datasets). All these datasets are available on the webpage associated with this paper.

In order to create a noisy dataset from the original one, the noise is introduced into the training partitions as follows:

1. A level of noise $x\%$ of class noise is introduced into a copy of the full original dataset.
2. Both datasets, the original one and the noisy copy, are partitioned into 5 equivalent folds, that is, with the same examples in each one.
3. The training partitions are built from the noisy copy, whereas the test partitions are formed from examples from the base dataset, that is, the noise free dataset.

The accuracy estimation of the classifiers in a dataset is obtained by means of 5 runs of a stratified 5-fold cross-validation. Hence, a total of 25 runs per dataset and noise level are averaged. The aforementioned 175 datasets will be preprocessed with our approach an other 7 noise filters resulting in 1400 new preprocessed datasets. The preprocessing with all the 8 noise filters of the 5x5 folds for each one of the 175 unprocessed datasets implies a total of 35000 executions, and the running of each one of the three classification algorithms (C4.5, SVM and k-NN) of the 5x5fcv of the 1575 datasets (175 unprocessed and 1400 processed) results in 118125 additional executions, from which the results obtained are analyzed in this paper. Furthermore, we have performed additional experiments with several thresholds for our proposal, which increases the number of experiments carried out.

4.2. Parameter setup of the FC-based filter

The parameter setup for the three classification algorithms used by our FC-based filter is presented in Table 4:

Table 4: Parameter specification for the classification algorithms used in the FC-based filter.

Classifier	Ref.	Parameters
C4.5	[11]	Confidence: 0.25, minimal instances per leaf: 2, prune after the tree building
k-NN	[23]	$k = 3$, Euclidean distance
LOG	[32]	Ridge value in the log-likelihood: 10^{-8}

4.3. Noise filtering methods

The noise filters used for the comparison have been chosen due they apply different filtering strategies and are well-known representatives of the field. They are briefly described in the following:

1. *Edited Nearest Neighbor* (ENN) [9]. This algorithm removes those examples which class does not agree with that of the majority of its k nearest neighbors.
2. *All k-Nearest Neighbors* (AllKNN) [16]. This applies the k-NN rule k times varying the number of neighbors considered between 1 to k. Those examples misclassified by k-NN are removed from the training set when all the values of k have been considered.
3. *Classification Filter* (CF) [8]. CF splits the training set into n subsets. A set of classifiers is trained from the union of any $n-1$ subsets. The examples misclassified in the excluded subset are then eliminated from the training set.
4. *Multiedit* (ME) [15]. This splits the training data into n parts. k-NN classifies the examples from the part x considering the part *(x+1) mod n* as training set and the misclassified examples are removed. This process is repeated until no examples are eliminated.
5. *Nearest Centroid Neighbor Edition* (NCNE) [10]. This is a slight modification of ENN, which consists of discarding from the training set every example misclassified by the k nearest centroid neighbors (k-NCN) rule.
6. *Ensemble Filter* (EF) [7]. EF classifies the training data using an n-fold cross-validation with several classification algorithms. Then, the noisy examples are identified using a voting scheme (consensus or majority) and removed from the training data.

7. *Iterative-Partitioning Filter* (IPF) [17]. IPF removes noisy examples in multiple iterations. In each iteration, the training data is split into n subsets and C4.5 is built over each of these subsets to evaluate all the examples. Then, the examples misclassified are removed (using the consensus or majority scheme) and a new iteration is started.

The parameter setup for all the noise filters is shown in Table 5. Even though almost all of the parameters are the default ones recommended by the authors of such filters, we have set the majority scheme and $n = 3$ partitions for ensemble-based filters in order to establish a fair comparison among this type of filters.

Table 5: Parameter specification for the noise filters.

Filter	Ref.	Abbreviation	Parameters
AllKNN	[16]	AllKNN	k value: 3, Distance: Euclidean
Classification Filter	[8]	CF	Classifier: C4.5, n: 3
Edited Nearest Neighbor	[9]	ENN	k value: 3, Distance: Euclidean
Multiedit	[15]	ME	k value: 1, Subblocks: 3, Distance: Euclidean
Nearest Centroid Neighborhood Edition	[10]	NCNE	k value: 3
Ensemble Filter	[7]	EF	Voting scheme: majority, n: 3
Iterative-Partitioning Filter	[17]	IPF	Voting scheme: majority, n: 3
Iterative Noise Filter based on the Fusion of Classifiers	-	INFFC	Voting scheme: majority, n: 3

Note that EF and IPF are filtering approaches based on the use of ensembles (*ensemble-based filters*), whereas the rest of the methods are based on single classifiers and/or measures (*non-ensemble filters*). Therefore, we will consider these two groups in the comparisons carried out in the following section.

4.4. Methodology of analysis

The effect of the aforementioned filters along with our proposal will be analyzed comparing the performance obtained for each dataset with three different classifiers: C4.5 [11], SVM [22] and k-NN [23] with $k = 1$. As we have already mentioned, the performance estimation is obtained by means of 5 runs of a 5-fold *stratified cross-validation*, averaging the test accuracy results. Given the large amount of results obtained, for the sake of brevity only averaged results are shown in the paper (the detailed results can be found on the web page associated with this paper), but it must be taken into account that our conclusions are based on the proper statistical analysis, which considers all the results (not averaged). In addition, the number of datasets preprocessed with each filter in which each classifier obtains the best result is shown.

The performance of our approach is studied with each classifier (C4.5, SVM and 1-NN) using statistical comparisons in three different scenarios:

1. Comparison between INFFC and not applying preprocessing (Section 5.2). By means of this comparison we try to check if the application of noise filtering techniques implies an advantage with respect to no preprocessing (*None*).
2. Comparison among INFFC and the *non-ensemble* filters (Section 5.3). Its motivation is to check the behavior of our proposal against classic state-of-the-art filters belonging to other different paradigms from ensemble-based methods. These include the noise filters AllKNN, CF, ENN, ME and NCNE.
3. Comparison among INFFC and the *ensemble-based* filters (Section 5.4). We also compare our proposal, which is based on ensembles of classifiers, with other related noise filters that also use multiple classifiers (EF and IPF).

We have separately studied the differences between our proposal and the *non-ensemble* and *ensemble* methods for two main reasons. First, the separation is motivated by the different nature of the methods of both groups. Second, performing a multiple statistical comparison usually requires a much higher quantity of datasets to detect significant differences when the number of comparison methods increases. Multiple statistical comparisons are then limited by the number of datasets and an unified comparison can only be performed if a much higher quantity of datasets than the one considered in this paper is available for study.

Wilcoxon's test [36] will be applied to study the differences between the proposal of this paper and no using preprocessing. The p-values associated with the comparison of the results of the two methods involved over all the datasets will be obtained. The p-value represents the lowest level of significance of a hypothesis that results in a rejection and it allows one to know whether two algorithms are significantly different and the degree of their difference. We will consider a difference to be significant if the p-value obtained is lower than 0.1 - even though p-values slightly higher than 0.1 might be showing important differences.

Regarding the comparison between our approach and the other noise filters (either *non-ensemble* or *ensemble-based* methods), the Friedman Aligned-ranks test [24, 37] will be used. We will use this test to compute the set of ranks that represent the effectiveness associated with each algorithm and the p-value related to the significance of the differences found by this test. In addition, the adjusted p-value with Holm test [38] will be computed. More information about these tests and other statistical procedures can be found at `http://sci2s.ugr.es/sicidm/`.

Other point to analyze includes an analysis of which examples are removed from the data set belong to those corrupted by the noise introduction scheme and which to the non-corrupted set of examples. Consider that $D_T = D_N \sqcup D_O$, being D_N the set of examples whose class labels have been corrupted by the noise scheme and D_O the set of original non-corrupted examples.

On the other hand, each noise filter removes a set of examples $D_R \subseteq D_T$. Based on these sets of examples, we have defined two different metrics in order to check the capabilities of elimination of each noise filter (see Table 6).

Finally, we will also study the effect of the noise threshold in the performance of our approach, by comparing the results of INFFC considering several thresholds by means of the usage of the Friedman Aligned procedure and the Holm test.

Table 6: Measures computed.

Metric description	Expression
Eliminations among the corrupted examples	$100\frac{\lvert D_N \cap D_R \rvert}{\lvert D_N \rvert}$
Eliminations among the non-corrupted examples	$100\frac{\lvert D_O \cap D_R \rvert}{\lvert D_O \rvert}$

5. Analysis of results

This section presents the analysis of the results obtained. First, performance results are presented and analyzed in Section 5.1. To add depth to the analysis of the results, several statistical comparisons are performed, studying the differences among the proposal of this paper and not preprocessing (Section 5.2), its comparison with the other *non-ensemble methods* (Section 5.3) and its comparison with the other *ensemble-based methods* (Section 5.4). The analysis of the examples removed by each noise filter is shown in Section 5.5, whereas the study on the behavior of INFFC with different thresholds for the noise score is shown in Section 5.6.

5.1. Accuracy results and number of datasets with best result

Table 7 shows the test accuracy obtained by each classifier when using each one of the 8 filters considered and without preprocessing (*None* column). This table also shows the number of datasets on which each filter provides the best result with each one of the three classifiers and noise level. The best results at each noise level are highlighted in boldface. From this table, several remarks can be made:

1. **Test accuracy results:**

   - For all the classifiers (C4.5, SVM and 1-NN), INFFC is the best method at all the noise levels, and also without additional noise.
   - Considering C4.5, the results of IPF and EF are also remarkable (even though they are lower than those of INFFC). In general, the worst filters are ME and AllKNN.

Table 7: Test accuracy of each classifier (C4.5, SVM and 1-NN) and number of datasets on which each filter provides the best result (best results are remarked in bold).

	Test accuracy							Best (out of 25)						
Method	0%	5%	10%	15%	20%	25%	30%	0%	5%	10%	15%	20%	25%	30%
C4.5														
AllKNN	79.20	78.87	78.48	78.00	77.39	77.36	76.35	1	0	0	1	1	1	1
CF	80.43	80.21	79.83	79.37	78.87	78.63	78.01	0	0	0	0	1	0	0
ENN	80.09	79.87	79.75	79.06	78.76	78.33	77.65	0	2	2	0	3	5	1
EF	80.41	80.22	79.83	79.56	79.33	79.01	78.46	2	5	6	5	5	3	3
IPF	81.18	80.79	80.56	79.92	79.32	79.27	79.03	6	3	2	3	1	5	1
ME	77.88	77.60	76.79	76.60	75.52	75.32	74.45	3	1	0	0	0	2	0
NCNE	80.58	80.23	79.98	79.18	78.65	78.43	77.41	1	1	2	3	1	1	0
INFFC	**81.77**	**81.57**	**81.21**	**80.97**	**80.44**	**80.07**	**79.99**	**8**	**9**	**9**	**12**	**12**	**8**	**17**
None	81.32	80.89	80.35	79.46	78.24	77.21	76.16	7	5	5	2	2	0	2
SVM														
AllKNN	77.89	77.50	77.48	76.76	75.86	75.14	74.20	3	3	4	2	1	1	1
CF	79.53	79.00	78.79	78.28	77.75	77.40	76.75	0	1	0	2	0	1	1
ENN	78.60	78.48	78.24	77.91	77.25	76.87	75.83	1	1	1	1	1	1	0
EF	79.72	79.33	79.02	78.73	78.20	78.24	77.56	3	4	3	4	4	2	4
IPF	79.76	79.42	79.28	78.88	78.47	78.29	77.84	2	1	1	0	1	4	4
ME	77.25	76.87	76.21	75.49	74.90	74.25	73.25	0	0	0	1	0	2	0
NCNE	78.92	78.56	78.01	77.33	76.15	75.21	74.11	3	3	1	3	1	0	1
INFFC	**80.71**	**80.43**	**80.17**	**79.89**	**79.67**	**79.41**	**78.97**	6	**9**	**12**	**10**	**13**	**10**	**11**
None	79.98	77.80	76.06	74.71	72.87	71.05	69.77	**8**	4	3	4	4	4	3
1-NN														
AllKNN	79.09	78.80	78.38	77.75	77.31	76.65	76.24	6	5	5	2	1	0	1
CF	79.99	79.76	79.56	79.07	79.01	78.18	77.73	1	0	2	1	2	2	0
ENN	79.58	79.32	78.98	78.56	78.18	77.43	76.92	0	1	0	1	1	0	0
EF	79.95	79.61	79.32	79.05	78.84	78.43	78.27	2	4	2	5	6	3	6
IPF	80.29	79.95	79.70	79.40	79.22	78.74	78.52	2	4	4	4	3	4	4
ME	77.18	76.88	76.17	75.85	75.37	74.81	74.01	0	0	0	1	1	4	1
NCNE	80.01	79.70	79.22	78.46	77.87	77.29	76.22	4	3	5	2	2	1	0
INFFC	**80.75**	**80.37**	**80.10**	**79.89**	**79.65**	**79.25**	**79.17**	4	**8**	**8**	**10**	**8**	**12**	**13**
None	79.15	76.43	74.01	71.67	69.49	67.04	64.07	**7**	2	0	0	1	0	0

The behavior of *None* should be mentioned: at the lowest noise levels (up to 10%), it obtains good test accuracy results; at intermediate noise levels, from 15% to 20%, it obtains medium results compared with the rest of the noise filters; finally, when the noise level is higher than 25% it obtains the worst results, as expected.

- The results of SVM are similar to those of C4.5. The two best filters after INFFC are IPF and EF. The worst filters are again ME and AllKNN (up to 5% of noise) and from 10% onwards, *None* is clearly the worst method. However, *None* obtains

very good results without noise.

- Regarding to 1-NN, INFFC is usually followed by IPF, even though CF (up to 20%) and EF (from 20% onwards) are obtain remarkable results. The worst results are obtained usually by ME and *None*.
- Even though filters may be counterproductive without excessive noise since they are more likely to remove clean examples, one must realize that they only imply a low loss of accuracy without noise (0% of noise level). In any case, we must known that this observation is based on average results. In order to reach meaningful conclusions we should use statistical test as it is recommended in the specialized literature [24], and it is performed in the following sections.

2. **Number of datasets with best result:**

   - INFFC generally obtains the best results in the higher number of datasets with independence of the classifier considered. There are two exceptions: for the noise sensitive classifiers, SVM and 1-NN, *None* highlights over the rest of filtering techniques. These exceptions can be attributed to the fact that, without any noise, SVM and 1-NN are able to build accurate classifiers. Thus, in absence of noise, the more information (number of examples) they have, the more exact the models constructed may be (so they do not need the application of filtering techniques in this case). However, their behavior obviously go worse when the noise level increases, since they are considered very noise-sensitive techniques.
   - For C4.5, IPF and EF (and *None* at the lowest noise levels, up to 10%) also obtains good results.
   - For SVM, IPF only obtains good results at the highest noise levels, from 25% onwards. The results of EF, AllKNN (up to 10%) or *None* must be also considered, since they are also good compared with the rest of noise filters.
   - For 1-NN, IPF and EF also highlight at many noise levels, whereas AllKNN and NCNE only obtains good results at the lowest noise levels (up to 10%).

5.2. Comparison between INFFC and not preprocessing

The results of INFFC with each one of the classifiers (C4.5, SVM and 1-NN) are compared with those of the datasets without preprocessing (*None*). In order to study whether there are statistical differences among them, Wilcoxon's test has been performed - see Table 8. In this table, INFFC and *None* using different classifiers (C4.5, 1-NN and SVM) are compared

using the Wilcoxon's test and the p-value associated are shown. From the very low p-values obtained in these comparisons, one can conclude that there exist statistical differences for all the classifiers at all the noise levels between the usage of our filter and *None* (considering a significance level of $\alpha = 0.1$). For C4.5 and SVM without noise, even though these differences are not significant, very low p-value are obtained. This fact shows the great advantage of applying filtering techniques with respect to no preprocessing, particularly when the noise level increases.

Table 8: p-values of the Wilcoxon's test after comparing the proposed filter versus no preprocessing using each one of the three classifiers: C4.5, SVM and 1-NN.

Method	0%	5%	10%	15%	20%	25%	30%
C4.5	0.14854	0.02831	0.02735	0.00172	0.00010	0.00004	0.00006
SVM	0.17024	0.00189	0.00081	0.00019	0.00017	0.00017	0.00010
1-NN	0.01702	0.00002	0.00001	0.00001	0.00001	0.00001	0.00001

5.3. Comparison among INFFC and the non-ensemble filters

Table 9 presents the statistical comparison performed among INFFC and the *non-ensemble* filters using each one of the three classifiers considered (C4.5, SVM and 1-NN). The results using these filters are statistically compared; Table 9 shows for each filter a results in the form *ranks/p-value*, where *ranks* represent the ranks obtained by the Friedman Aligned procedure and *p-value* is the adjusted p-value computed by the Holm test. This table also shows the p-value associated with the significance of the differences found by the Friedman Aligned test. Looking at Table 9, we can observe that:

1. *Results of C4.5.* INFFC is significantly better than all the other filters at all the noise levels considering a significance level of $\alpha = 0.1$, although there are a exception: the comparison versus NCNE at the two lowest noise levels (0% and 5%), even thought the p-values are relatively low (close to 0.2 and 0.1, respectively).
2. *Results of SVM.* INFFC is statistically better than the rest of the filters, even though with NCNE and CF without noise these p-values are slightly higher than 0.1.
3. *Results of 1-NN.* INFFC is statistically better than AllKNN and ME at all the noise levels. It also is statistically better than ENN and NCNE from 15% onwards (and it obtains a very low p-value close to 0.1 versus ENN with 10% of noise level). Finally, INFFC is statistically better than CF at the highest noise levels, from 25% onwards, although relatively low p-values are obtained with 15% and 20% (close to 0.17 and 0.14,

respectively). Thus, in the worst cases, at the lowest noise levels, INFFC shows a better behavior, but without statistical differences.

Table 9: Results for each *non-ensemble* filter and noise level in the form *ranks/p-value*, where *ranks* represent the ranks obtained by the Friedman Aligned test and *p-value* is the adjusted p-value computed by the Holm test. Those cases where the null hypothesis is not rejected ($\alpha = 0.1$) are indicated with a star (*).

Method	0%	5%	10%	15%	20%	25%	30%
C4.5							
AllKNN	98.46/0.00007	109.56/0.00000	109.54/0.00000	109.37/0.00000	108.15/0.00000	102.79/0.00000	109.31/0.00000
CF	80.50/0.01292	72.42/0.04831	77.42/0.00481	81.94/0.00046	77.13/0.00166	70.42/0.03724	72.27/0.00101
ENN	71.25/0.06861	71.29/0.04831	66.23/0.04736	62.10/0.02753	62.73/0.02124	66.96/0.03724	68.85/0.00135
ME	114.88/0.00000	114.06/0.00000	117.08/0.00000	113.90/0.00000	120.25/0.00000	117.85/0.00000	117.35/0.00000
NCNE	61.17/0.18944*	61.40/0.12673*	62.85/0.04736	69.21/0.01115	68.87/0.01043	72.15/0.03724	74.56/0.00075
INFFC	44.73	42.27	37.88	34.48	33.87	40.83	28.67
SVM							
AllKNN	95.54/0.00020	97.92/0.00008	89.94/0.00289	91.65/0.00033	98.73/0.00000	98.31/0.00000	90.46/0.00000
CF	67.13/0.12529*	72.88/0.04587	69.31/0.08286	72.02/0.03531	66.69/0.01048	67.69/0.00590	66.38/0.00276
ENN	85.00/0.00393	78.15/0.02111	74.00/0.06993	69.77/0.03531	68.17/0.01048	68.13/0.00590	73.27/0.00079
ME	110.54/0.00000	107.62/0.00000	107.98/0.00001	111.69/0.00000	106.98/0.00000	105.50/0.00000	105.73/0.00000
NCNE	68.06/0.12529*	70.04/0.04587	82.19/0.01720	83.58/0.00295	97.23/0.00000	100.48/0.00000	106.27/0.00000
INFFC	44.73	44.38	47.58	42.29	33.19	30.88	28.88
1-NN							
AllKNN	91.08/0.01439	94.65/0.00418	90.85/0.00738	97.77/0.00012	103.52/0.00000	98.23/0.00000	90.31/0.00001
CF	69.06/0.49687*	64.71/0.74839*	62.56/0.39177*	62.38/0.17781*	56.13/0.14441*	59.19/0.05091	60.25/0.02636
ENN	75.54/0.28394*	75.98/0.22132*	77.65/0.11785*	73.10/0.05527	74.31/0.00723	85.27/0.00012	83.44/0.00009
ME	118.90/0.00000	119.02/0.00000	118.69/0.00000	116.12/0.00000	114.19/0.00000	107.38/0.00000	104.46/0.00000
NCNE	61.83/0.56389*	63.06/0.74839*	69.42/0.32044*	76.13/0.04347	85.00/0.00050	86.19/0.00012	100.12/0.00000
INFFC	54.6	53.58	51.83	45.5	37.85	34.73	32.42

5.4. Comparison among INFFC and the ensemble-based filters

Table 10 presents the statistical comparison performed among INFFC and the *ensemble-based* filters using each one of the three classifiers considered (C4.5, SVM and 1-NN). Looking at Table 10, we can observe that, for C4.5 and SVM, INFFC is clearly better than EF and IPF at all the noise levels. With 1- NN, our proposal is statistically better than the other two *ensemble-based* filters from 20% onwards, considering a significance level $\alpha = 0.1$ (at the other noise levels, very low p-values are obtained, all close to 0.1 except for 10% in which the p-value is higher.

Table 10: Results for each *ensemble-based* filter and noise level in the form *ranks/p-value*, where *ranks* represent the ranks obtained by the Friedman Aligned test and *p-value* is the adjusted p-value computed by the Holm test. Those cases where the null hypothesis is not rejected ($\alpha = 0.1$) are indicated with a star (*).

Method	0%	5%	10%	15%	20%	25%	30%
C4.5							
EF	48.78/0.00032	45.60/0.00106	46.28/0.00185	44.26/0.00078	40.92/0.00702	42.58/0.01139	46.16/0.00017
IPF	39.70/0.02143	44.16/0.00123	41.86/0.00944	46.20/0.00047	48.78/0.00014	44.44/0.00924	45.88/0.00017
INFFC	25.52	24.24	25.86	23.54	24.3	26.98	21.96
SVM							
EF	44.04/0.00330	43.80/0.00153	44.16/0.00737	44.22/0.00753	47.76/0.00007	46.02/0.00060	46.24/0.00105
IPF	44.68/0.00330	45.94/0.00087	43.58/0.00737	43.42/0.00753	44.04/0.00040	44.24/0.00088	42.90/0.00343
INFFC	25.28	24.26	26.26	26.36	22.2	23.74	24.86
1-NN							
EF	42.90/0.10551*	41.54/0.13166*	39.22/0.33988*	42.12/0.11285*	40.64/0.09284	44.60/0.00761	39.96/0.11043*
IPF	40.14/0.13644*	41.90/0.13166*	41.62/0.33988*	41.52/0.11285*	43.08/0.07571	42.64/0.00999	43.92/0.05036
INFFC	30.96	30.56	33.16	30.36	30.28	26.76	30.12

5.5. Number of examples removed by each filter

Table 11 shows the results for the two metrics described in Table 6, referring to percentages of examples removed by each noise filter at each noise level in the sets of corrupted examples and non-corrupted examples. For sake of generality, only average results for each metric at each noise level are shown. The complete results for each data set at each noise level can be found in the webpage associated with this paper. The analysis of this table of results leads to following observations:

1. **Examples removed among the corrupted examples.** Generally, the number of examples removed among the corrupted examples (with noise) is maintained for all the filters at the different noise levels. However, this capability of noise detection is reduced for most of the noise filters when the noise level increases, except for the the INFFC, IPF and ME filters, which are less affected for the increasing of the noise level.

   Even though INFFC is the filter that detects less noisy examples. This fact is more clearly observed at the lowest noise levels (for example, between AllKNN and INFFC a high difference can be observed). However, this difference is usually decreased when noise level increases. This may be due to the fact that, with low noise levels, the amount of noisy examples is small and the detection of some of these examples may notably alter

this percentage, whereas more examples identified as noisy are need to notably alter this percentage at higher noise levels.

Therefore, even though INFFC detects less noisy examples, it obtains good noise elimination results (around 87% at all the noise levels). It can be considered to be comparable to the rest of the noise filters, particularly when the noise level increases (for example, with CF, ENN, IPF or NCNE).

2. **Examples removed among the original non-corrupted examples.** The number of examples removed among non-corrupt examples (those in which noise has not been introduced) changes in a higher degree for the majority of the noise filters when the noise level increases. Thus, for example, AllKNN drastically removes more examples when the noise level increases (from a 31.82% at 5% of noise level to 46.82 at 30% of noise level). Something similar occurs with the rest of the filters, with the exception of INFFC and IPF, which are able to better maintain the number of examples removed in this set. INFFC is the method that less examples removes among the non-corrupted examples, with great differences with respect to the rest of the noise filters considered. For example, it obtains 12.92% versus 17.31% of IPF (the second method that removes less examples) and versus 46.82% of AllKNN (the method that removes the largest number of examples).

As a conclusion, it can be said that our proposal is able to maintain the level of elimination in both the corrupted and non-corrupted sets of examples regardless of the noise level. Even though it removes less examples among the corrupted examples, it maintains a similar elimination level to other noise filters, particularly when the noise level increases. However, almost all the filters behaves worse considering the non-corrupted set of examples. They eliminate large amounts of possible clean examples compared with INFFC. Thus our method is able to maintain a good balance between the examples that one must delete and those that must not. Moreover, observing the results obtained along this study, the importance of maintaining as many non-corrupted examples as possible can be highlighted, since statistically better results are obtained even though less noisy examples are removed. Hence, a good balance between the elimination of noisy examples and the correct detection of noise-free example should be maintained.

5.6. Influence of the threshold in INFFC

In addition to the comparison with respect to the state-of-the-art filtering methods, we have studied the effect of the value of the threshold in our proposal. In order to perform this analysis, we have computed the number of examples for each value of the noise score on which this metric has been computed, considering all the datasets used in this paper. These results

Table 11: Percentages of examples removed by each noise filter at each noise level in the sets of corrupted examples and non-corrupted examples (the best results are remarked in bold).

Set	Corrupted examples						Non-corrupted examples					
Noise level	5%	10%	15%	20%	25%	30%	5%	10%	15%	20%	25%	30%
AllKNN	**95.67**	**94.77**	**94.47**	**94.57**	**93.85**	**93.58**	31.81	34.86	37.87	40.73	43.65	46.82
CF	90.98	90.63	90.26	90.12	89.51	88.90	20.38	21.02	22.31	23.33	24.40	26.07
ENN	92.36	90.81	90.23	90.00	88.89	88.28	20.39	21.42	22.60	23.92	25.65	27.48
EF	93.18	92.65	92.32	92.08	91.65	91.63	19.21	19.92	21.02	21.95	22.77	24.46
IPF	90.51	89.72	89.68	89.55	89.09	89.16	15.42	15.56	16.17	16.62	16.87	17.31
ME	92.98	92.73	92.58	92.54	92.39	92.70	30.54	32.51	34.69	36.56	38.64	40.97
NCNE	91.84	90.17	89.52	88.12	87.78	86.24	20.71	22.58	24.45	26.66	28.77	31.19
INFFC	87.86	86.75	86.86	86.73	87.17	87.03	**11.58**	**11.84**	**12.00**	**12.34**	**12.78**	**12.92**

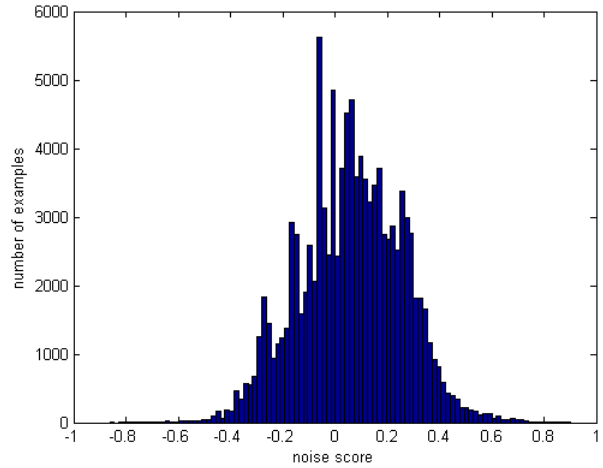
are represented in Figure 4 for four noise levels, that are 0%, 10%, 20% and 30% (the graphics for rest of noise levels can be found in the webpage associated to this paper).

Attending to Figure 4, two points must be remarked:

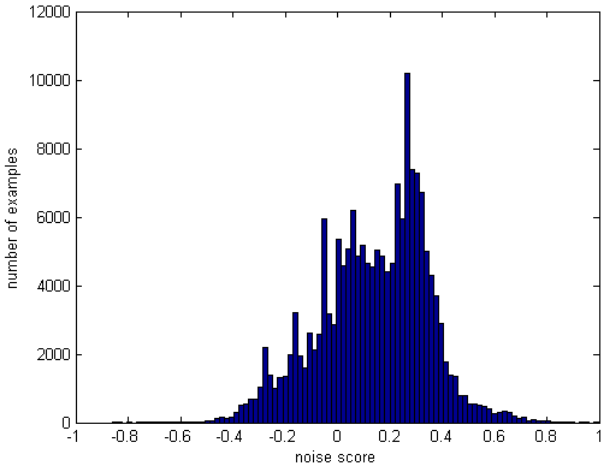
- The distribution of the noise score displaces to the right (to higher values) when the noise level increases. This is due to the fact that, increasing the number of noisy examples in a dataset produces a higher appearance of examples that are more easily identified as noisy by the noise metric (even though they are not necessarily noisy since they may be in noisy clusters), and therefore their noise score is higher.

- Most of the values of the noise score are in the interval (-0.5, 0.8). On this account, we have chosen this interval of values to perform an study of the behavior of the proposed filter considering several values of the threshold, in order to check how it affects to the results obtained.

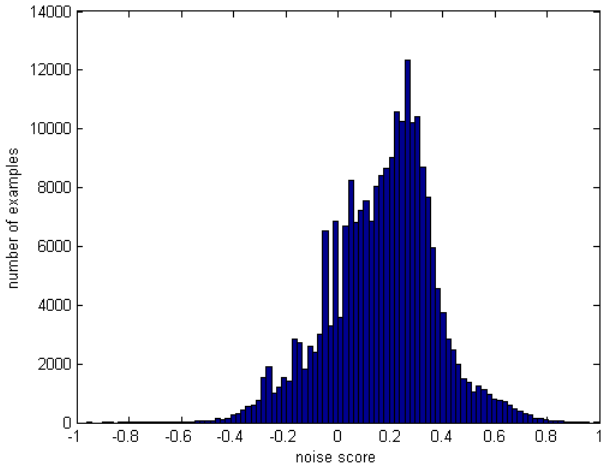
Table 12 shows for each of the thresholds considered in INFFC (from 0-5 to 0.8, by increments of 0.1) the results in the form *ranks/p-value*, where *ranks* represent the ranks obtained by the Friedman Aligned procedure and *p-value* is the adjusted p-value computed by the Holm test. In order to perform this comparison we have considered the results of the C4.5 classifier, although the results with SVM and 1-NN (found in the webpage of the paper) provide similar results to those shown here. The results of Table 12 show that the thresholds selected as the those with the best Friedman Aligned rankings are 0, 0.1 and 0.2 at most of the noise levels. In general, no statistical differences are found between these thresholds and consider lower values, even though they are generally statistically better that higher values of the threshold. This fact shows that removing all the potentially noisy examples in each iteration is not always the



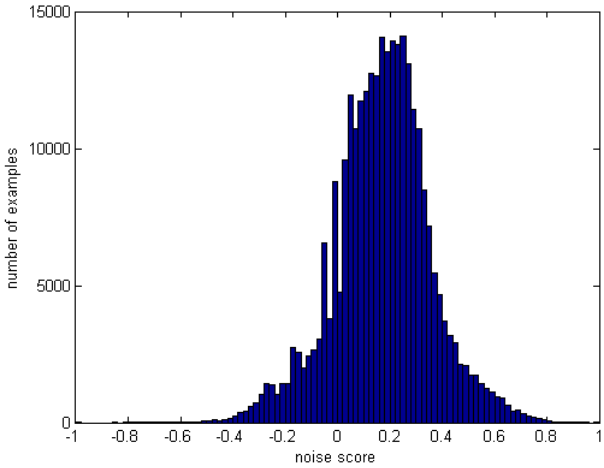
(a) 0% of class noise.



(b) 10% of class noise.



(c) 20% of class noise.



(d) 30% of class noise.

Figure 4: Distribution of the noise score for different noise levels. Each figure represents the number of examples for each value of the noise score on which this metric has been computed, considering all the datasets used in this paper.

best option. Thus, in many cases, it is better to adopt more conservative strategies to detect and remove the noisy examples (even though this advantage may not be statistically better, as shown in our experimentation). However, removing very low quantities of noisy examples in each iteration seems to be a worse alternative, since statistical differences are found comparing the highest thresholds with the lower one set as control methods by the Friedman Aligned-ranks test.

Table 12: Results for comparison among INFFC considering different thresholds in the form *ranks/p-value*, where *ranks* represent the ranks obtained by the Friedman Aligned-ranks test and *p-value* is the adjusted p-value computed by the Holm test. Those cases where the null hypothesis is not rejected ($\alpha = 0.1$) are indicated with a star (*).

Threshold	0%	5%	10%	15%	20%	25%	30%
-0.5	160.90/1.00000*	169.80/0.41766*	155.68/1.00000*	144.08/0.99949*	134.06/1.00000*	119.68/1.00000*	110.46/1.00000*
-0.4	159.38/1.00000*	167.74/0.43093*	156.20/1.00000*	143.88/0.99949*	132.62/1.00000*	118.50/1.00000*	110.80/1.00000*
-0.3	158.92/1.00000*	163.52/0.43225*	158.00/1.00000*	141.60/0.99949*	124.26/1.00000*	115.98/1.00000*	108.54/1.00000*
-0.2	167.56/1.00000*	155.38/0.60225*	158.36/1.00000*	136.18/0.99949*	120.24/1.00000*	118.64/1.00000*	109.16/1.00000*
-0.1	172.50/1.00000*	147.38/0.74059*	150.86/1.00000*	127.82/1.00000*	116.84/1.00000*	112.54/1.00000*	112.76/1.00000*
0	159.08/1.00000*	137.98/0.81359*	141.06/1.00000*	120.82/1.00000*	112.76/1.00000*	115.18/1.00000*	104.36
0.1	156.10/1.00000*	114.24	141.66/1.00000*	104.58/1.00000*	108.48	103.18	110.22/1.00000*
0.2	146.1	117.40/0.91208*	124.88	102.14	115.66/1.00000*	113.00/1.00000*	129.10/1.00000*
0.3	159.70/1.00000*	165.72/0.43225*	160.02/1.00000*	192.36/0.01295	171.92/0.21311*	188.38/0.02328	195.10/0.01217
0.4	182.54/1.00000*	204.50/0.01450	200.64/0.07303	230.98/0.00006	224.54/0.00045	244.80/0.00001	253.30/0.00000
0.5	204.02/0.42982*	226.40/0.00089	219.26/0.00974	239.64/0.00002	263.28/0.00000	264.26/0.00000	270.58/0.00000
0.6	209.42/0.31528*	229.60/0.00070	228.14/0.00339	256.98/0.00000	274.98/0.00000	279.40/0.00000	278.90/0.00000
0.7	210.20/0.31528*	229.84/0.00070	231.14/0.00266	258.86/0.00000	278.82/0.00000	281.88/0.00000	282.24/0.00000
0.8	210.58/0.31528*	227.50/0.00083	231.10/0.00266	257.08/0.00000	278.54/0.00000	281.58/0.00000	281.48/0.00000

As final remark, one must realize that all the reported results and the statistical comparisons performed show the suitability of our filtering approach dealing with class noise datasets. The combination of the iterative ensemble-based filter and the noise score to control the noise sensitiveness is shown as a good alternative to other existing noise filters found in the literature. The threshold of the proposal must be carefully fixed in order to obtain the best possible results. Very high values of the threshold seems to be inferior to choosing lower values - we recommended that this threshold is fixed close to 0, according to our experimentation.

6. Concluding remarks

This paper proposes a new noise filtering method combining three different noise filtering paradigms: the usage of ensembles for filtering, the iterative filtering and the computation of noise measures. Thus, we propose an iterative noise filtering method based on the fusion of the predictions of several classifiers. We also introduce a noisy score to control the filtering sensitiveness and remove more or less noisy examples according to the practitioner's necessities. We have compared our proposal against other well-known filters found in the literature over a large collection of real-world datasets with different levels of class noise.

The most remarkable idea behind the analysis of results is that INFFC maintains a similar elimination level of noisy examples than that of other noise filters, particularly when the noise level increases. However, almost all the filters studied in this paper eliminate higher amounts

of clean examples compared with INFFC. Thus our method is able to maintain a good balance between the examples that one must delete (noisy examples) and those that must not (clean examples). Furthermore, from the experimental results we can conclude that our noise filter enhance the performance of the rest of the noise filters and no preprocessing. The statistical analysis performed supports our conclusions.

The analysis on the impact of the threshold fixed in the noise score shows that removing all the potentially noisy examples in each iteration is not always the best option, but neither removing very low quantities of noisy examples. Hence, a balance must be found in order to obtain the desired results. For this reason, we recommend a threshold close to 0, whose optimal value can be adapted depending on the problem.

Acknowledgment

Supported by the Projects TIN2011-28488, P10-TIC-06858 and P11-TIC-7765. J. A. Sáez holds an FPU grant from the Spanish Ministry of Education and Science.

References

[1] X. Wu, X. Zhu, Mining With Noise Knowledge: Error-Aware Data Mining, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 38 (4) (2008) 917–932.

[2] S. García, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Vol. 72 of Intelligent Systems Reference Library, Springer, 2015.

[3] X. Zhu, X. Wu, Class Noise vs. Attribute Noise: A Quantitative Study, Artificial Intelligence Review 22 (2004) 177–210.

[4] B. Frenay, M. Verleysen, Classification in the presence of label noise: A survey, IEEE Transactions on Neural Networks and Learning Systems 25 (5) (2014) 845–869.

[5] J. A. Sáez, J. Luengo, F. Herrera, Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification, Pattern Recognition 46 (1) (2013) 355–364.

[6] J. A. Sáez, M. Galar, J. Luengo, F. Herrera, Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition, Knowledge and Information Systems 38 (1) (2014) 179–206.

[7] C. E. Brodley, M. A. Friedl, Identifying Mislabeled Training Data, Journal of Artificial Intelligence Research 11 (1999) 131–167.

[8] D. Gamberger, R. Boskovic, N. Lavrac, C. Groselj, Experiments With Noise Filtering in a Medical Domain, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, 1999, pp. 143–151.

[9] D. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems and Man and Cybernetics 2 (3) (1972) 408–421.

[10] J. Sánchez, R. Barandela, A. Márques, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recognition Letters 24 (2003) 1015–1022.

[11] J. R. Quinlan, C4.5: programs for machine learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.

[12] W. W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann Publishers, 1995, pp. 115–123.

[13] D. Gamberger, N. Lavrac, S. Dzeroski, Noise elimination in inductive concept learning: A case study in medical diagnosis, in: In Proc. of the 7th International Workshop on Algorithmic Learning Theory, Springer, 1996, pp. 199–212.

[14] D. Gamberger, N. Lavrac, S. Dzeroski, Noise Detection and Elimination in Data Preprocessing: experiments in medical domains, Applied Artificial Intelligence 14 (2000) 205–223.

[15] P. Devijver, On the editing rate of the MULTIEDIT algorithm, Pattern Recognition Letters 4 (1) (1986) 9–12.

[16] I. Tomek, An Experiment With The Edited Nearest-Neighbor Rule, IEEE Transactions on Systems and Man and Cybernetics 6 (6) (1976) 448–452.

[17] T. M. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise filtering techniques, Journal of Computer Science and Technology 22 (2007) 387–396.

[18] T. K. Ho, J. J. Hull, S. N. Srihari, Decision Combination in Multiple Classifier Systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (1) (1994) 66–75.

[19] M. Woźniak, M. Graña, E. Corchado, A Survey of Multiple Classifier Systems as Hybrid Systems, Information Fusion 16 (2014) 3–17.

[20] L. I. Kuncheva, J. J. Rodríguez, A weighted voting framework for classifiers ensembles, Knowledge and Information Systems 38 (2) (2014) 259–275.

[21] J. A. Sáez, M. Galar, J. Luengo, F. Herrera, Tackling the problem of classification with noisy data using Multiple Classifier Systems: Analysis of the performance and robustness, Information Sciences 247 (2013) 1–20.

[22] V. Vapnik, Statistical Learning Theory, Wiley, New York, USA, 1998.

[23] G. J. Mclachlan, Discriminant Analysis and Statistical Pattern Recognition (Wiley Series in Probability and Statistics), Wiley-Interscience, 2004.

[24] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (2010) 2044–2064.

[25] R. J. Hickey, Noise modelling and evaluating learning from examples, Artificial Intelligence 82 (1-2) (1996) 157–179.

[26] R. Y. Wang, V. C. Storey, C. P. Firth, A Framework for Analysis of Data Quality Research, IEEE Transactions on Knowledge and Data Engineering 7 (4) (1995) 623–640.

[27] C. M. Teng, Polishing blemishes: Issues in data correction, IEEE Intelligent Systems 19 (2004) 34–39.

[28] A. P. Dawid, A. M. Skene, Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm, Applied Statistics 28 (1) (1979) 20–28.

[29] A. Malossini, E. Blanzieri, R. T. Ng, Detecting potential labeling errors in microarrays by data perturbation, Bioinformatics 22 (17) (2006) 2114–2121.

[30] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106.

[31] I. Kononenko, M. Kukar, Machine Learning and Data Mining: Introduction to Principles and Algorithms, Horwood Publishing Limited, 2007.

[32] S. le Cessie, J. van Houwelingen, Ridge estimators in logistic regression, Applied Statistics 41 (1) (1992)

191–201.

[33] J. Maudes, J. J. Rodríguez, C. García-Osorio, N. García-Pedrajas, Random feature weights for decision tree ensemble construction, Information Fusion 13 (1) (2012) 20–30.

[34] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2–3) (2011) 255–287.

[35] C.-M. Teng, Correcting Noisy Data, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999, pp. 239–248.

[36] J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 7 (2006) 1–30.

[37] S. García, F. Herrera, An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.

[38] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.

3.3 SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering

Sáez J. A., Luengo J., Stefanowski J., and Herrera F. (2014) SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, doi: 10.1016/j.ins.2014.08.051. *Information Sciences* (in press)

- Status: **Accepted**.
- Impact Factor (JCR 2014): Not available.
- Current Impact Factor of the Journal (JCR 2013): 3.893
- Subject Category: Computer Science, Information Systems. Ranking 8 / 135 (**Q1**).

SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering

José A. Sáez[a,*], Julián Luengo[b], Jerzy Stefanowski[c], Francisco Herrera[a]

[a] *Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, Granada, Spain, 18071*
[b] *Department of Civil Engineering, LSI, University of Burgos, Burgos, Spain, 09006*
[c] *Institute of Computing Science, Poznań University of Technology, ul. Piotrowo 2, 60-965 Poznań, Poland.*

Abstract

Classification datasets often have an unequal class distribution among their examples. This problem is known as imbalanced classification. The *Synthetic Minority Over-sampling Technique* (SMOTE) is one of the most well-know data pre-processing methods to cope with it and to balance the different number of examples of each class. However, as recent works claim, class imbalance is not a problem in itself and performance degradation is also associated with other factors related to the distribution of the data. One of these is the presence of noisy and borderline examples, the latter lying in the areas surrounding class boundaries. Certain intrinsic limitations of SMOTE can aggravate the problem produced by these types of examples and current generalizations of SMOTE are not correctly adapted to their treatment.

This paper proposes the extension of SMOTE through a new element, an iterative ensemble-based noise filter called *Iterative-Partitioning Filter* (IPF), which can overcome the problems produced by noisy and borderline examples in imbalanced datasets. This extension results in SMOTE-IPF. The properties of this proposal are discussed in a comprehensive experimental study. It is compared against a basic SMOTE and its most well-known generalizations. The experiments are carried out both on a set of synthetic datasets with different levels of noise and shapes of borderline examples as well as real-world datasets. Furthermore, the impact of introducing additional different types and levels of noise into these real-world data is studied. The results show that the new proposal performs better than existing SMOTE generalizations for all these different scenarios. The analysis of these results also helps to identify the characteristics of IPF which differentiate it from other filtering approaches.

Keywords:

Imbalanced Classification, Borderline Examples, Noisy Data, Noise Filters, SMOTE.

*Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
Email addresses: smja@decsai.ugr.es (José A. Sáez), jluengo@ubu.es (Julián Luengo), jerzy.stefanowski@cs.put.poznan.pl (Jerzy Stefanowski), herrera@decsai.ugr.es (Francisco Herrera)

1. Introduction

Several real-world classification problems in fields such as text categorization [49], medicine [52], bankruptcy prediction [38] and intrusion detection [31], are characterized by a highly imbalanced distribution of examples among the classes. In these problems, one class (known as the minority or positive class) contains a much smaller number of examples than the other classes (the majority or negative classes). The minority class is often the most interesting from the application point of view [11, 5]. Class imbalance constitutes a difficulty for most learning algorithms which assume an approximately balanced class distribution and are biased toward the learning and recognition of the majority class. As a result, minority class examples usually tend to be misclassified.

The problem of learning from imbalanced data has been intensively researched in the last decade and several methods have been proposed to address it - for a review see, e.g., [24]. Re-sampling methods [9, 8, 33, 47] are a classifier-independent type of techniques that modify the data distribution taking into account local characteristics of examples to change the balance between classes. There are numerous works discussing their advantages [4, 10]. Among these methods, the *Synthetic Minority Over-sampling Technique* (SMOTE) [9] is one of the most well-known; it generates new artificial minority class examples by interpolating among several minority class examples that lie together.

However, some researchers have shown that the class imbalance ratio is not a problem itself. Even though the observation of a low classification performance in some concrete imbalanced problems may be influenced by the validation scheme used to estimate this performance of the classifiers [35], the classification performance degradation is usually linked to other factors related to data distributions [28], [22], [40]. Among them, in [40] the influence of noisy and borderline examples on classification performance in imbalanced datasets is experimentally studied. Borderline examples are defined as examples located either very close to the decision boundary between minority and majority classes or located in the area surrounding class boundaries where classes overlap. The authors of [33, 40] refer to noisy examples as those from one class located deep inside the region of the other class. Furthermore, this paper, considers noisy examples in the wider sense of [57, 43], in which they are treated as examples corrupted either in the attribute values or the class label.

Even though SMOTE achieves a better distribution of the number of examples in each class, when used in isolation it may obtain results that are not as good as they could be or it may even be counterproductive in many cases. This is because SMOTE presents several drawbacks related to its *blind* oversampling, whereby the creation of new positive (minority) examples only takes into account the closeness among positive examples and the number of examples of each class, whereas other characteristics of the data are ignored - such as the distribution of examples from the majority classes. These drawbacks, which can further aggravate the difficulties produced for noisy and borderline examples in

the learning process, include: (i) the creation of too many examples around unnecessary positive examples which do not facilitate the learning of the minority class, (ii) the introduction of noisy positive examples in areas belonging to the majority class and (iii) the disruption of the boundaries between the classes and, therefore, an increase in the overlapping between them. In order to overcome these problems, two different approaches are followed in the literature:

1. Modifications of SMOTE (hereafter called *change-direction* methods). These guide the creation of positive examples performed by SMOTE towards specific parts of the input space, taking into account specific characteristics of the data. Within this group, the *Safe-Levels-SMOTE* (SL-SMOTE) [8], the *Borderline-SMOTE* (B1-SMOTE and B2-SMOTE) [23] or LN-SMOTE [37] methods are found, which try to create positive examples close to areas with a high concentration of positive examples or only inside the boundaries of the positive class.
2. Extensions of SMOTE by integrating it with additional techniques (these extensions will be referred to as *filtering-based* methods since SMOTE is integrated with either special *cleaning* or filtering methods). In the standard classification tasks, noise filters are often used in order to detect and eliminate noisy examples from training datasets and also to clean up and to create more regular class boundaries [55, 53]. Experimental studies, such as [4], confirm the usefulness of integrating such filters - e.g., *Edited Nearest Neighbor Rule* (ENN) or *Tomek Links* (TL) [53] - as a post-processing step after using SMOTE.

The ability to deal with imbalanced datasets with noisy and borderline examples of methods belonging to both approaches will be studied in the experimental section, even though this paper also proposes a new extension of SMOTE. Existing extensions of SMOTE are very simple because they are based on using a single learning algorithm or simple measures such as, e.g., *k-Nearest Neighbors* (*k*-NN) [39] paradigm inside ENN [55] - used in SMOTE-ENN.

Some works highlight the good behavior of ensembles for classification in noisy environments, showing that the combined use of several classifiers is a good alternative in these scenarios as opposed to the employment of single classifiers [42], [43]. In the same way, some authors also propose the usage of ensembles for filtering [7], [17], [18], [54]. However, all these works only consider the point of view of the standard classification and the overall classification accuracy. Thus, ensembles are used for filtering in [7] considering that some examples have been mislabeled and the label errors are independent of particular classifiers learned from the data. In this scenario, the authors claim that collecting predictions from different classifiers could provide a better estimation of mislabeled examples rather than collecting information from a single classifier only. According to our best knowledge, these ensemble-based filters have not yet been used in the context of learning from imbalanced data. Analyzing such filters focuses our attention on the *Iterative-Partitioning Filter* (IPF) [32]. Its characteristics differentiate it from most

of the filters, making it particularly suitable to overcome the problems produced by noisy and borderline examples specific to the dataset plus those additional ones that SMOTE may introduce.

The main aim of this paper is to propose and examine a new extension of SMOTE, in which the IPF noise filter is applied in post-processing resulting in SMOTE-IPF - its implementation can be found in KEEL[1] [2]. Its suitability for handling noisy and borderline examples in imbalanced data will be a particular focus of evaluation as these are one of the main sources of difficulties for learning algorithms. Differences between this approach and other re-sampling methods also based on generalizations of SMOTE will be discussed and studied. One cannot treat this proposal as a simple combination of two methods, as we want to study more deeply the conditions of its appropriate use in dealing with different types of noise in imbalanced data which have not been considered yet. We discuss its properties in comparison to other previous, related generalizations of SMOTE.

The other contribution of this paper is to provide a comprehensive experimental comparison of SMOTE-IPF with these generalizations. Moreover, different data factors will be considered in these parts of this experimental study. A first part will be carried out with special synthetic datasets containing different shapes of the minority class example boundaries and levels of borderline examples, as considered in related studies [22, 28, 29, 40]. Additionally, a set of real-world datasets which are also known to be affected by noisy and borderline examples will be considered. All of these were used in [40] and are available in the KEEL-dataset repository [1]. Yet another contribution of this paper will be to introduce additional class or attribute noise into these real-world datasets and to study its impact on compared SMOTE generalizations. After preprocessing these datasets, the performances of the classifiers built with C4.5 [41] will be evaluated and they will also be contrasted using the proper statistical tests as recommended in the specialized literature [14, 19, 25]. The characteristics of IPF which differentiate it from other filters and a discussion on the strengths and weaknesses of IPF in dealing with imbalanced datasets with noisy and borderline examples will be analyzed in Section 6.

In addition, experiments with many other classification algorithms on the preprocessed datasets will be carried out in order to show the behavior of the preprocessing techniques with different classifiers. These are *k*-NN [39], a *Support Vector Machine* (SVM) [13, 51], *Repeated Incremental Pruning to Produce Error Reduction* (RIPPER) [12] and PART [16]. Due to length restrictions, their results are only included on the web-page associated with this paper, available at `http://sci2s.ugr.es/noisebor-imbalanced`. This web-page also includes the basic information of this paper, the datasets used and the parameter setup for all the classification algorithms.

The rest of this paper is organized as follows. Section 2 presents the imbalanced dataset problem. Section 3 is devoted to the motivations behind our extension of SMOTE. Next, Section 4 describes

[1] www.keel.es

the experimental framework. Section 5 includes the analysis of the experimental results, and Section 6 outlines the results and the suitability of IPF for the problem treated. Finally, in Section 7, some concluding remarks are presented.

2. Classification for imbalanced datasets

In this section, first the problem of imbalanced datasets is introduced in Section 2.1. Some additional problems related to class imbalance that may harm classifier performance are described in Section 2.2.

2.1. The problem of imbalanced datasets

The main difficulty of imbalanced datasets is that a standard classifier might ignore the importance of the minority class because its representation inside the dataset is not strong enough and the classifier is biased toward the majority class or, in other words, it is oriented to achieve a good total classification accuracy. Consequently, the examples that belong to the minority class are misclassified more often than those belonging to the majority class [27].

This type of data may be categorized depending on its imbalance ratio (*IR*) [15], which is defined as the relation between the majority class and minority class examples, by the expression

$$IR = \frac{N^-}{N^+} \tag{1}$$

where N^- and N^+ are the number of examples belonging to the majority and minority classes, respectively. Thus, a dataset is imbalanced when $IR > 1$.

A large number of approaches have been previously proposed to deal with the class imbalance problem. These approaches can be mainly categorized in two groups [3]:

1. *Algorithmic level approaches*. This group of methods tries to change search techniques or the classification decision strategies to impose bias toward the minority class or to improve the prediction performance by adjusting weights for each class [27].
2. *Data level approaches*. This group of methods preprocess the dataset modifying the data distribution to change the balance between classes considering local characteristics of examples [4].

Furthermore, cost-sensitive learning solutions incorporating both the data and algorithmic level approaches assume higher misclassification costs with samples in the minority class and seek to minimize the high cost errors [50].

There are some data level approaches particularly adapted to the usage of a concrete classifier. For example, the authors of [36] propose an evolutionary framework that uses an instance generation technique that modifies the original training set based on the performance of a specific classifier on the

minority class. However, this is not the most common scenario and the great advantage of data level approaches is that they are more versatile, since their use is independent of the classifier selected. Furthermore, one may preprocess all datasets beforehand in order to use them to train different classifiers. In this manner, the computation time needed to prepare the data is only required once. For these reasons, the proposal made in this paper belongs to this group of methods. In addition, re-sampling approaches can be categorized into two sub-categories: under-sampling [53, 55], which consists of reducing the data by eliminating examples belonging to the majority class with the objective of balancing the number of examples of each class; and over-sampling [9, 8], which aims to replicate or generate new positive examples in order to gain importance, improving the importance of this class.

In order to estimate the quality of classifiers built from imbalanced data, several measures have been proposed in the literature [24]. This is because the most widely used empirical measure, *total accuracy*, does not distinguish between the number of correct labels of different classes, which in the ambit of imbalanced problems may lead to erroneous conclusions. This paper considers the usage of the *Area Under the ROC Curve* (AUC) measure [6], which provides a single-number summary for the performance of learning algorithms and it is recommended in many other works in the literature [4, 15].

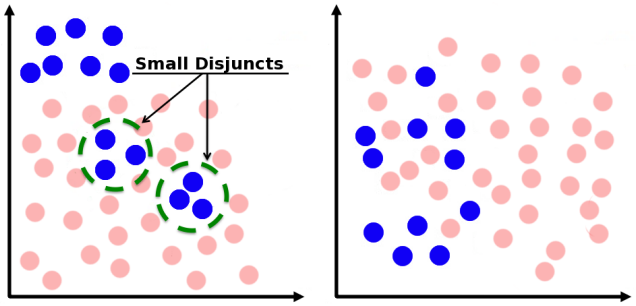
2.2. Other factors characterizing imbalanced data

The imbalance ratio between classes is a problem that may hinder the performance of classifiers. However, it is not the only source of difficulty for classifiers; recent works have indicated other relevant issues related to the degradation of performance:

- *Presence of small disjuncts* [28, 29] (Figure 1a). The minority class can be decomposed into many sub-clusters with very few examples in each one, surrounded by majority class examples. This is a source of difficulty for most learning algorithms in detecting enough of these sub-concepts.

- *Overlapping between classes* [22, 21] (Figure 1b). There are often some examples from different classes with very similar characteristics, in particular if they are located in the regions around decision boundaries between classes. These examples refer to overlapping regions of classes.

Closely related to the overlapping between classes, in [40] another interesting problem in imbalanced domains is pointed out: the higher or lower presence of examples located in the area surrounding class boundaries, which are called borderline examples. Researchers have found that misclassification often occurs near class boundaries where overlapping usually occurs as well and it is hard to find a feasible solution for it [20]. The authors in [40] showed that classifier performance degradation was strongly affected by the quantity of borderline examples and that the presence of other noisy examples located farther outside the overlapping region also made the task of re-sampling methods very difficult.



(a) Small disjuncts (b) Overlapping

Figure 1: Examples of the imbalance between classes: (a) small disjuncts and (b) overlapping between classes.

This paper focuses on studying the influence of noisy and borderline examples in generalizations of SMOTE, considering the synthetic datasets and also the real-world ones used in [40] along with new noisy datasets built from the latter. These datasets will be described in Section 4. To clarify terminology, one must distinguish (inspired by [40, 33]) between safe, borderline and noisy examples (see Figure 2):

- *Safe examples* are placed in relatively homogeneous areas with respect to the class label.
- *Borderline examples* are located in the area surrounding class boundaries, where either the minority and majority classes overlap or these examples are very close to the difficult shape of the boundary - in this case, these examples are also difficult as a small amount of the attribute noise can move them to the wrong side of the decision boundary [33].
- *Noisy examples* are individuals from one class occurring in the safe areas of the other class. According to [33] they could be treated as examples affected by class label noise. Notice that the term *noisy examples* will be further used in this paper in the wider sense of [57], in which noisy examples are corrupted either in their attribute values or the class label.

The examples belonging to the last two groups often do not contribute to a correct class prediction [30]. Therefore, one could ask whether removing them (all or the most difficult misclassification parts) could improve classification performance. Thus, this paper proposes the use of noise filters to achieve this goal, because they are amply used with good results in classification. We are particularly interested in ensemble filters as they are the most careful when deciding whether an example should be viewed as noise and removed.

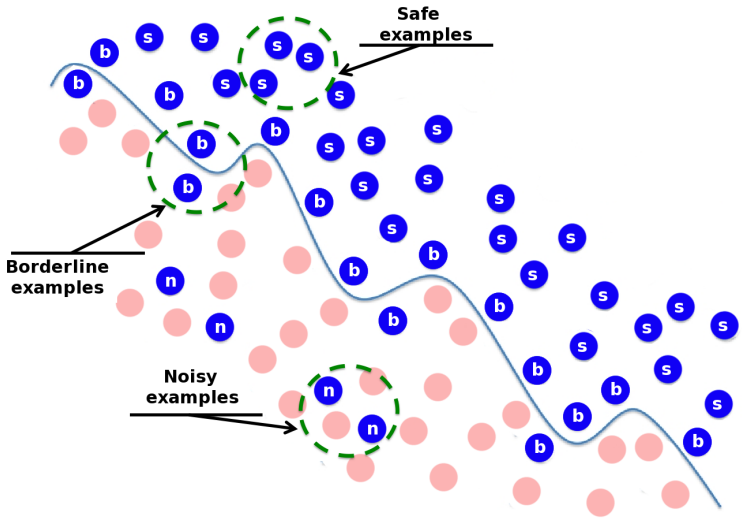


Figure 2: The three types of examples considered in this paper: safe examples (labeled as *s*), borderline examples (labeled as *b*) and noisy examples (labeled as *n*). The continuous line shows the decision boundary between the two classes.

3. SMOTE along with noise filters to tackle noisy and borderline examples

In this section, first, the main details of the proposed extension of SMOTE are given in Section 3.1. Next, its two implicated parts are described in depth: the SMOTE algorithm in Section 3.2 and noise filters in Section 3.3.

3.1. Combining re-sampling and noise filters

As has been mentioned previously, SMOTE is one of the most well-known and widely used re-sampling techniques, however it may still have problems with some distributions of data. Two different generalizations of SMOTE have been proposed in the literature in order to improve final classification performance: *change-direction* and *filtering-based* methods - for the evaluation and discussion of their limitations see also [37]. The former, however, may present several important drawbacks when imbalanced datasets are suffering from noisy and borderline examples:

1. Noisy and borderline examples could be removed from the data to improve the final performance [30] but *change-direction* methods do not allow this option since their only function is to create new positive examples.
2. The creation of new positive examples, although directed towards specific parts of the input space, may be erroneous because it is based on data with noisy examples. This fact shows, once again, the need to introduce a cleaning phase after the creation of examples.

Methods based on extending SMOTE with additional cleaning seem to be more appropriate to deal with imbalanced problems with noisy and borderline examples. These methods follow two postulates:

1. Class distribution has to be transformed, e.g., balanced in some degree, in order to support the learning of classifiers [4].
2. The most difficult noisy and borderline examples should be removed from the training data since they are often the most difficult for learning [30, 18].

The first task can be coped with by means of using re-sampling techniques. The SMOTE algorithm [9] helps to balance the class distribution, avoiding the overfitting problem that might be produced by using other techniques such as simple *Random Over-sampling* [4]. Furthermore, SMOTE can fill the interior of minority class sub-parts with synthetic minority class examples. This is a positive fact since in imbalanced datasets with a considerable quantity of borderline examples, minority class clusters are usually defined by those with an emptier interior. Nevertheless, class clusters may not be well defined in cases where some majority class examples might be invading the minority class space. The opposite can also be true, since interpolating minority class examples can expand the minority class clusters, introducing artificial minority class examples too deeply into the majority class space. This additional minority noise is also caused by the blind over-generalization of SMOTE-based techniques of looking for nearest neighbors from the minority class only. Both situations could introduce additional noise into datasets.

The aforementioned problems are already well known. They have been tackled by combining SMOTE with an additional step of under-sampling, e.g., with the ENN filtering [55], which aims to remove mislabeled data from the training data after the usage of SMOTE. However, these methods do not perform this task as well as they should in all cases.

The second task requires specific and more powerful methods designed to eliminate mislabeled examples when datasets have a considerable number of such examples. A group of methods that address this problem is ensemble-based noise filters [18, 7, 54]. This paper proposes the extension of SMOTE with one of these filters: the IPF filter [32], which will be responsible for removing noisy examples originally present in the dataset and also those created by SMOTE. Besides this, IPF cleans up class boundaries, making them more regular and facilitating in this way the posterior learning phase [18].

These two techniques (SMOTE and IPF) must be applied to the imbalanced dataset in the correct order in order to obtain reasonable final results: SMOTE in the first place and then the IPF noise filter. This is due to IPF being designed to deal with standard classification datasets. Its application over an imbalanced dataset before the usage of SMOTE (which balances the distribution of classes) may carry the risk of removing all the examples from the minority class, which may be seen as noisy examples because they are underrepresented in the dataset. In short, as a summary and justification of this approach,

it is claimed that:

1. The SMOTE algorithm fulfills a dual function: it balances the class distribution and it helps to fill in the interior of sub-parts of the minority class.
2. The IPF filter removes the noisy examples originally present in the dataset and also those created by SMOTE. Besides this, IPF cleans up the boundaries of the classes, making them more regular.

Note that the scheme proposed (SMOTE-IPF) enables one to replace SMOTE with any of its modifications, in particular, the change-direction generalizations. However, in this paper we prefer to use the basic version of SMOTE because it is consistent with earlier research on the usage of filtering techniques and facilitates the comparison with them. The usage of IPF may also disturb the effects of the earlier modifications of examples by the change-direction methods. Moreover, some of these methods strongly focus on the over-sampling of concrete regions of the original data. For example, *Borderline-SMOTE* may over-strength the boundary zone, which will be problematic for studying data with many noisy and borderline examples.

3.2. The synthetic minority over-sampling technique

SMOTE [9], introduced by Chawla and co-authors, is now one of the most popular over-sampling methods. In this approach, the positive class is over-sampled by taking each minority class example and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. In order to find these neighbors in the space of numerical and nominal attributes, the HVDM metric is applied [56]. Depending on the amount of oversampling required, neighbors from the k nearest neighbors are randomly chosen. Analyzing the current literature on the usage of SMOTE, one can notice that $k = 5$ neighbors is usually chosen. This procedure of building a local neighborhood is also often applied in other resampling methods, such as SPIDER. Although one could ask a more general question by tuning a particular k value depending on the given data characteristics, we decided to use one value $k = 5$ to be more consistent with other related works on SMOTE and its generalizations since they were compared using the same data sets as we have chosen for our study. Taking the same motivations to be consistent with related works, we tune the oversampling amount to balance both classes to 50%.

Synthetic examples are generated in the following way. Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

3.3. Noise filters

Noise filters are preprocessing mechanisms designed to detect and eliminate noisy examples in the training set [55, 7, 32, 44]. The result of noise elimination in preprocessing is a reduced training set which is then used as an input to a machine learning algorithm.

Some of these filters are based on the computation of different measures over the training data. For instance, the method proposed in [18] is based on the observation that the elimination of noisy examples reduces the *Complexity of the Least Complex Correct Hypothesis* value of the training set.

In addition, there are many other noise filters based on the usage of ensembles. In [7], multiple classifiers belonging to different learning paradigms were built and trained from a possibly corrupted dataset and then used to identify mislabeled data, which is characterized as the examples that are incorrectly classified by the multiple classifiers. Similar techniques have been widely developed considering the building of several classifiers with the same learning algorithm [17, 54]. Instead of using multiple classifiers learned from the same training set, in [17] a *Classification Filter* (CF) approach is suggested, in which the training set is partitioned into n subsets, then a set of classifiers is trained from the union of any $n - 1$ subsets; those classifiers are used to classify the examples in the excluded subset, eliminating the examples that are incorrectly classified.

From our knowledge and preliminary, earlier experiments performed with different ensemble-based filters, e.g., the *Ensemble Filter* (EF) [7], the *Cross-Validated Committees Filter* (CVCF) [54], CF and others, the notable good behavior of the *Iterative-Partitioning Filter* [32] when detecting noisy examples must be pointed out. IPF has characteristics which differentiate it from most of the noise filters and may provide the reasons why it performs better than them - they will be analyzed and discussed in Section 6.

IPF removes noisy examples in multiple iterations until a stopping criterion is reached. The iterative process stops when, for a number of consecutive iterations k, the number of identified noisy examples in each of these iterations is less than a percentage p of the size of the original training dataset. Initially, the method starts with a set of noisy examples $A = \emptyset$. The basic steps of each iteration as follows:

1. Split the current training dataset E into n equal sized subsets.
2. Build a classifier with the C4.5 algorithm over each of these n subsets and use them to evaluate the whole current training dataset E.
3. Add to A the noisy examples identified in E using a voting scheme (consensus or majority).
4. Remove the noisy examples: $E \leftarrow E \setminus A$.

Two voting schemes can be used to identify noisy examples: consensus and majority. The former removes an example if it is misclassified by all the classifiers, whereas the latter removes an example if it is misclassified by more than half of the classifiers.

The parameter setup for the implementation of IPF used in this work has been determined experimentally in order to better fit it to the characteristics of imbalanced datasets with noisy and borderline examples once they have been preprocessed with SMOTE. More precisely, the majority scheme is used to identify the noisy examples, $n = 9$ partitions with random examples in each one are created and $k = 3$ iterations for the stop criterion and $p = 1\%$ of removed examples are considered. This parameter setup is based on a study of the influence of each parameter on the results - the justification of each parameter value and its influence is given in Section 6.3.

4. Experimental framework

In this section, the details of the experimental study developed in this paper are presented. First, in Section 4.1, we describe how the synthetic imbalanced datasets with borderline examples were built. Then, the real-world datasets and the noise introduction processes are presented in Section 4.2. In Section 4.3 the preprocessing techniques considered in this work are briefly described. Finally, in Section 4.4, the methodology of the analysis carried out is described.

4.1. Synthetic imbalanced datasets with borderline examples

This paper uses the family of synthetic datasets used in prior research on the role of borderline examples [40]. These data were created following other experimental studies of small disjuncts [28, 29] and overlapping between classes [21]. However, these studies were focused on studying single factors and very simple (lines and rectangles) shapes of decision boundaries. Thus, the authors of [40] created more complex data affected by many factors, including borderline examples. Many datasets with different configurations were generated by special software and evaluated; for more details see [46]. In this paper we consider these configurations of datasets which were the basis for the previous analysis of the role of borderline examples for different basic classifiers, such as C4.5, and re-sampling methods [40]. These datasets are briefly characterized below:

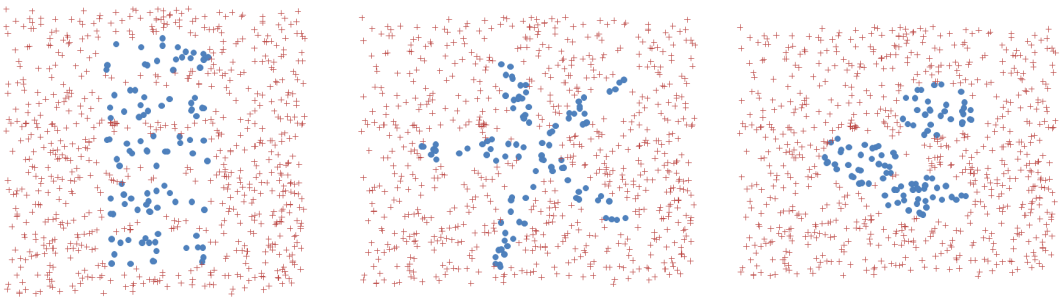
1. *Number of classes and attributes*. This work focuses on binary classification problems (the minority versus the majority class) with examples randomly and uniformly distributed in the two-dimensional real-value space.
2. *Number of examples and imbalance ratios*. Multiple datasets with two different numbers of examples and imbalance ratios are considered: datasets with 600 examples and $IR = 5$ and datasets with 800 examples and $IR = 7$. The values of the parameters resulted from the assumption of having at least 20 examples for the subpart of the decomposed minority class. Smaller cardinalities led to unstable results [46].

3. *Shapes of the minority class*. Three different shapes of the minority class surrounded uniformly by the majority class are taken into account:

   - *Sub-cluster* (Figure 3a): the examples from the minority class are located inside rectangles following related works on small disjuncts [28].
   - *Clover* (Figure 3b): represents a more difficult, non-linear setting, in which the minority class resembles a flower with elliptic petals.
   - *Paw* (Figure 3c): the minority class is decomposed into 3 elliptic subregions of varying cardinalities, in which two subregions are located close to each other, and the smaller subregion is separated.

   The minority class is decomposed into 5 parts except for paw data with 3 subregions. Paw could better represent real-world data than clover. Moreover, both clover and paw should be more difficult to learn than the simple circles that were considered in some related works.

4. *Disturbance ratios* (*DR*). The impact of disturbing the borders of sub-regions in the minority class will is studied. This is carried out by increasing the ratio of borderline examples, i.e., the disturbance ratio, from the minority class subregions. 5 levels of *DR* are considered: 0%, 30%, 50%, 60% and 70%. The width of the borderline overlapping areas is comparable to the width of the safe parts of sub-regions. The aforementioned DR values result from earlier studies [46] which showed that smaller values (less than 20%) did not affect the performance of the considered classifier very much. Moreover, higher values will better discriminate between the usefulness of different preprocessing methods [40].



(a) Sub-cluster. (b) Clover. (c) Paw.

Figure 3: Shapes of the minority class.

In such a way, 30 synthetic datasets with the above mentioned properties have been managed, all of which are available on the web-page associated with this paper.

4.2. Real-world datasets

The choice of the real-world datasets was based on the work on imbalanced classification with noisy and borderline examples presented in [40]. These are available at the KEEL-dataset repository[2] [1] or have been provided by the authors of [45] in the case of the *acl* dataset. Multi-class datasets are modified to obtain two-class imbalanced problems, defining the joint of one or more classes as positive and the remainder as negative. Table 1 shows the characteristics of these datasets. For each one, the number of examples (#Exa), attributes (#Att), the name of the minority class (Min), the number of examples of the minority class (#Min), of the majority class (#Maj) and the imbalance ratio (IR) are shown.

Table 1: Characteristics of the real-world datasets.

Dataset	**#Exa**	**#Att**	**Min**	**#Min**	**#Maj**	**IR**
acl	140	6	with knee injury	40	100	2.5
breast	286	9	recurrence-events	85	201	2.36
bupa	345	6	sick	145	200	1.38
cleveland	303	13	positive	35	268	7.66
ecoli	336	7	imU	35	301	8.60
haberman	306	3	died	81	225	2.78
hepatitis	155	19	die	32	123	3.84
newthyroid	215	5	hyper	35	180	5.14
pima	768	8	positive	268	500	1.87

These data are also characterized by a large number of difficult examples, which could be recognized as borderline or noisy ones. In [37] a simple analysis of a local neighborhood has been made with k-NN classifiers according to rules applied in *Borderline-SMOTE* [23]. So, an example was treated as noisy if all its neighbors belong to opposite classes, as a borderline example if it was misclassified by the majority of the opposite classes and safe if it was correctly classified. Nearly all of these datasets were difficult with respect to the minority class as they contained much less safe examples than others. In particular, in the *cleveland* dataset the minority class contains 35 examples with 22 noisy and 13 borderline ones. A similar situation occurs for *breast cancer*. Some other datasets, e.g., *haberman* or *ecoli*, contain more borderline examples than noisy ones (e.g. *haberman* has 51 and 20 respectively plus 10 safe ones). As the number of noisy examples is quite high compared to the size of the minority class these data sets are chosen to study their impact. The exception is *newthyroid*, which seems to contain easier to understand data as its number of safe examples is higher than the others.

Furthermore, we decided that these real-world datasets could be transformed into more complex and even more difficult versions with an artificially increased noise level. In such a way, two different

[2] `http://www.keel.es/datasets.php`

noise levels will be introduced into them: $x = 20\%$ and $x = 40\%$. The introduction of noise consists of separately corrupting either the class labels or the attribute values of some examples belonging to each dataset. These corruptions respectively belong to the noise categories formally known as class noise and attribute noise and were amply used and studied in [57], a reference paper in the framework of noisy data in classification. The same two noise introduction schemes proposed by the authors of that paper are used in this work, so that:

- Class noise is introduced into the datasets following a pairwise scheme in which a majority class example has a probability of $x/100$ to be incorrectly labeled as belonging to the minority class.
- Attribute noise is introduced into the original datasets corrupting each attribute separately. To corrupt each attribute A_i, approximately $x\%$ of the examples in the dataset are chosen and the value of A_i of each of these examples is assigned a random value between the minimum and maximum of the domain of that attribute following a uniform distribution (if A_i is numerical), or choosing a random value of the domain (if A_i is nominal).

The performance estimation of each classifier for each of these real-world datasets, and also for the synthetic ones, is obtained by means of 5 runs of a stratified 5-fold cross-validation and their results are averaged. Dividing the dataset into 5 folds is considered in order to dispose of a sufficient quantity of minority class examples in the test partitions. In this way, test partition examples are more representative of the underlying knowledge and meaningful performance results can be obtained.

4.3. Re-sampling techniques for comparison

Different re-sampling techniques to adjust the class distribution in the training data based on generalizations of SMOTE are studied in this paper. The usefulness of a new SMOTE extension is studied in a comprehensive comparative study with other, related versions of SMOTE, in particular the best known representations of *change-direction* and *filtering-based* methods. Table 2 shows the SMOTE-based methods considered in this study - a wider description of such methods is found on the web-page associated with this paper.

Table 2: Re-sampling techniques considered.

Method	**Reference**	**Method**	**Reference**
SMOTE	[9]	SL-SMOTE	[8]
SMOTE-TL	[4]	B1-SMOTE	[23]
SMOTE-ENN	[4]	B2-SMOTE	[23]

Note that SMOTE-TL and SMOTE-ENN are approaches based on extending SMOTE with an additional filtering (*filtering-based* methods), whereas SL-SMOTE, B1-SMOTE and B2-SMOTE are approaches based on directing the creation of the positive examples (*change-direction* methods).

4.4. Analysis methodology

In order to check the suitability of the proposed extension of SMOTE versus the other re-sampling techniques when dealing with imbalanced datasets with noisy and borderline examples, the experiments are divided into three differentiated parts depending on the type of datasets considered in each one: synthetic, real-world and noisy modified real-world datasets.

The effect of the aforementioned preprocessing techniques will be analyzed comparing the AUC for each dataset obtained with C4.5 [41], which has been used in many other works in imbalanced classification, in particular concerning SMOTE [47, 48]. Furthermore, it is known to be more sensitive to different factors of imbalanced data than, e.g., SVM [13] and is often used inside the ensembles. The standard parameters along with a post-pruning have been considered for the executions.

For each of the three types of datasets, the AUC results obtained by C4.5 for our approach against (i) not applying preprocessing and applying SMOTE alone, (ii) the other *filtering-based* methods (SMOTE-ENN and SMOTE-TL) and (iii) the *change-direction* methods (B1-SMOTE, B2-SMOTE and SL-SMOTE) will be separately compared. We have separately studied the differences between our proposal and the filtering-based and change-direction methods for two main reasons. First, the separation is motivated by the different nature of the methods of both groups that share common characteristics, which allow us to independently obtain conclusions with each kind of method. On the other hand, performing a multiple statistical comparison usually requires a much higher quantity of datasets to detect significant differences when the number of comparison methods increases. Multiple statistical comparisons are then limited by the number of datasets, and the comparison grouping the two types of methods (filtering-based and change-direction) can only be performed if a much higher quantity of datasets than the one considered in this paper is available for study.

Additionally, statistical comparisons in each of these cases will be also performed. Wilcoxon's signed ranks statistical test [14] will be applied to compare SMOTE-IPF with no preprocessing and the usage of SMOTE alone. This is a non-parametric pairwise test that aims to detect significant differences between two sample means; that is, the behavior of the two algorithms involved in each comparison. The results of the two methods involved in the comparison over all the datasets will be compared using Wilcoxon's test and the p-values associated with these comparisons will be obtained. The p-value represents the lowest level of significance of a hypothesis that results in a rejection and it allows one to know both whether two algorithms are significantly different and the degree of their difference.

Regarding the comparison between our approach and the other re-sampling techniques (either *filtering-based* or *change-direction* methods), the aligned Friedman's procedure [19] will be used. This is an advanced nonparametric test for performing multiple comparisons, which improves the classic Friedman test. The Friedman test is based on sets of ranks, one set for each data set; and the performances of the algorithms analyzed are ranked separately for each data set. Such a ranking scheme allows for intra-set comparisons only, since inter-set comparisons are not meaningful. When the number of algorithms for comparison is small, this may pose a disadvantage. In such cases, comparability among data sets is desirable and we can employ the method of aligned ranks [26]. Because of this, we will use the aligned Friedman's test to compute the set of ranks that represent the effectiveness associated with each algorithm and the p-value related to the significance of the differences found by this test. In addition, the adjusted p-value with Hochberg's test [25] will be computed. More information about these tests and other statistical procedures can be found at `http://sci2s.ugr.es/sicidm/`.

We will consider a difference to be significant if the p-value obtained is lower than 0.1 [14], [19] - even though p-values slightly higher than 0.1 might be also showing important differences.

5. Evaluation of re-sampling methods with noisy and borderline examples

In this section, the performance of C4.5 using the different preprocessing techniques over the imbalanced datasets with noisy and borderline examples is analyzed. In Section 5.1, the results considering synthetic datasets are analyzed, whereas Section 5.2 and Section 5.3 are respectively devoted to analyzing the results on the real-world datasets and the noisy modified real-world ones.

5.1. Results on synthetic datasets

Table 3 presents the AUC results obtained by C4.5 on each synthetic dataset when preprocessing with each re-sampling approach considered in this paper. The column denoted by *None* corresponds to the case in which no re-sampling is performed prior to C4.5. The best case for each dataset is highlighted in bold. From these results, the following main points should be stressed:

- Increasing *DR*, fixing a shape of the minority class and an *IR*, strongly deteriorates the performance of C4.5 without preprocessing.
- Preprocessing improves the performance with respect to the case without preprocessing in nearly all the datasets. The improvements in the results for each single dataset reflect this fact.
- SMOTE-IPF obtains better results than the rest of the re-sampling methods in 11 of the 30 datasets considered and obtains results close to the best performances in the rest of the cases.

- Without preprocessing, linear rectangle shapes (sub-cluster datasets) are easier to learn than non-linear ones (clover or paw), since the former obtain higher performances. However, with most of the preprocessing techniques the non-linear paw datasets even outperform the linear sub-cluster datasets at the same *DR* level.

- The highest improvements of SMOTE-IPF are obtained in the learning of non-linear datasets, since 8 of the 11 overall best performance results are obtained for these types of datasets, which are the most difficult ones.

Table 4 collects the results of applying Wilcoxon's signed ranks statistical test between SMOTE-IPF versus *None* and SMOTE. As the p-values ($p_{Wilcoxon}$) and the sums of ranks (R^+ and R^-) reflect, the application of SMOTE-IPF produces an improvement in the results obtained with respect to not preprocessing or preprocessing only with SMOTE with these synthetic imbalanced datasets with borderline examples.

Regarding the comparison between re-sampling techniques considering the synthetic datasets, Table 5 presents the ranks of the aligned Friedman's procedure (Rank column) for each group of techniques (*filtering-based* methods and *change-direction* ones). In the case of the Friedman's aligned rank test, the method with the best average ranking among all the datasets is considered to be the best. Please note that SMOTE-IPF is established in all the cases as the control algorithm because it has obtained the best aligned Friedmans rank, indicating the high performance of the approach.

The p-value related to the significance of the differences found by the aligned Friedman's test ($p_{AlignedFriedman}$ row) is also shown. In addition, the $p_{Hochberg}$ column shows the adjusted p-value with Hochberg's test. Post-hoc tests indicate those methods that the control algorithm outperforms significantly.

Looking at Table 5, one can make some observations:

- The average rank for SMOTE-IPF obtained by the aligned Friedman's test is the best, i.e. the lowest, in all the considered cases and it is notably differentiated from the ranks of the rest of the methods. This occurs with both *filtering-based* and *change-direction* methods.

- Comparing ranks for *filtering-based* methods, SMOTE-IPF is followed by SMOTE-TL. SMOTE-ENN obtains the highest rank notably differentiated from the rest.

- Among the *change-direction* methods, SL-SMOTE outperforms the ranks of B1-SMOTE and B2-SMOTE, which are quite similar.

- The p-values of the aligned Friedman's test are very low in all the scenarios, which shows a great significance in the differences found.

Table 3: AUC results obtained by C4.5 on synthetic datasets with borderline examples.

Dataset	None	SMOTE	SMOTE-ENN	SMOTE-TL	SL-SMOTE	B1-SMOTE	B2-SMOTE	SMOTE-IPF
sub-cluster $_{IR=5,\ DR=0}$	94.10	93.00	90.00	91.30	91.50	**94.20**	93.50	92.70
sub-cluster $_{IR=5,\ DR=30}$	64.80	81.90	81.80	81.30	82.20	79.20	80.00	**83.40**
sub-cluster $_{IR=5,\ DR=50}$	51.30	80.80	77.80	**81.00**	80.40	77.30	77.50	77.80
sub-cluster $_{IR=5,\ DR=60}$	52.00	78.60	77.00	77.80	78.10	77.00	73.10	**79.70**
sub-cluster $_{IR=5,\ DR=70}$	50.00	77.50	77.00	79.20	**82.30**	78.50	78.00	80.10
sub-cluster $_{IR=7,\ DR=0}$	87.50	94.79	93.07	92.29	90.79	95.42	**95.64**	95.21
sub-cluster $_{IR=7,\ DR=30}$	76.86	81.57	79.64	77.64	81.71	80.29	79.71	**83.00**
sub-cluster $_{IR=7,\ DR=50}$	53.86	78.29	75.29	78.50	**81.29**	75.79	74.79	78.57
sub-cluster $_{IR=7,\ DR=60}$	50.00	80.21	75.71	78.57	**81.36**	73.93	73.86	79.79
sub-cluster $_{IR=7,\ DR=70}$	50.00	**82.50**	78.21	81.07	81.21	76.43	73.14	80.93
clover $_{IR=5,\ DR=0}$	70.50	83.50	**86.80**	87.30	85.70	86.40	**86.80**	85.50
clover $_{IR=5,\ DR=30}$	54.30	83.80	83.40	84.20	81.10	81.20	81.90	**85.30**
clover $_{IR=5,\ DR=50}$	51.60	83.40	81.00	**84.40**	83.00	79.90	83.10	82.40
clover $_{IR=5,\ DR=60}$	56.50	80.80	80.50	81.50	78.70	76.90	78.70	**82.20**
clover $_{IR=5,\ DR=70}$	50.00	77.20	76.10	**79.40**	77.10	74.10	78.70	79.00
clover $_{IR=7,\ DR=0}$	70.71	87.93	87.79	89.36	88.21	87.14	88.93	**91.64**
clover $_{IR=7,\ DR=30}$	53.21	83.64	83.14	81.00	84.36	79.29	81.29	**85.71**
clover $_{IR=7,\ DR=50}$	50.00	81.21	**82.86**	82.00	78.71	78.93	73.36	81.93
clover $_{IR=7,\ DR=60}$	50.00	78.79	77.71	**82.21**	77.21	77.79	71.71	82.14
clover $_{IR=7,\ DR=70}$	51.57	78.29	76.07	78.86	78.29	75.57	72.79	**80.21**
paw $_{IR=5,\ DR=0}$	91.00	**96.30**	94.90	94.90	92.80	93.10	93.30	94.50
paw $_{IR=5,\ DR=30}$	70.00	84.40	86.40	84.20	84.50	84.80	**87.30**	84.90
paw $_{IR=5,\ DR=50}$	67.90	84.60	83.30	83.90	**85.00**	81.70	82.80	84.90
paw $_{IR=5,\ DR=60}$	54.10	81.00	81.30	81.90	82.30	76.30	78.90	**83.30**
paw $_{IR=5,\ DR=70}$	57.70	82.80	83.50	**84.40**	82.70	80.60	80.40	83.40
paw $_{IR=7,\ DR=0}$	68.29	93.43	93.93	93.43	92.93	**95.21**	93.79	94.29
paw $_{IR=7,\ DR=30}$	56.71	84.29	84.93	84.21	83.50	84.36	**85.43**	85.29
paw $_{IR=7,\ DR=50}$	50.00	85.00	84.79	84.86	84.29	78.14	79.71	**85.79**
paw $_{IR=7,\ DR=60}$	53.00	**83.36**	80.71	82.57	83.00	72.71	75.14	82.14
paw $_{IR=7,\ DR=70}$	50.00	82.57	80.57	79.93	84.14	78.64	77.71	**85.71**

- The adjusted p-values by Hochberg's test are very low in all comparisons.

From the results of Tables 3, 4 and 5, one can conclude that SMOTE-IPF performs better than other SMOTE versions when dealing with the synthetic imbalanced datasets built with borderline examples, particularly in those with non-linear shapes of the minority class. All the statistical tests also clearly

Table 4: Wilcoxon's test results for the comparison of SMOTE-IPF (R^+) versus *None* and SMOTE (R^-) on synthetic datasets considering the AUC results obtained by C4.5.

Methods	$\mathbf{R^+}$	$\mathbf{R^-}$	$\mathbf{p_{Wilcoxon}}$
SMOTE-IPF vs. None	464.0	1.0	< 0.0001
SMOTE-IPF vs. SMOTE	366.0	99.0	0.0050

Table 5: Multiple statistical comparison with synthetic datasets.

	Algorithm	**Rank**	$\mathbf{p_{Hochberg}}$
Filtering	SMOTE-IPF	26.73	-
	SMOTE-ENN	64.93	< 0.000001
	SMOTE-TL	44.83	0.007289
	$\mathbf{p_{AlignedFriedman}}$	0.000014363186	
Change-Dir.	SMOTE-IPF	30.30	-
	SL-SMOTE	47.85	0.050698
	B1-SMOTE	83.15	< 0.000001
	B2-SMOTE	80.70	< 0.000001
	$\mathbf{p_{AlignedFriedman}}$	0.00002425464	

show the statistical significance of this better performance of SMOTE-IPF.

5.2. Results on real-world datasets

Table 6 presents the AUC results obtained by C4.5 on each real-world dataset with noisy and borderline examples when preprocessing with each re-sampling approach. With these types of datasets, the application of preprocessing techniques does not always obtain the expected results. For instance, some of the selected preprocessing methods, such as SMOTE-ENN, SMOTE-TL and B2-SMOTE, do not outperform the performance of *None* on *acl* and *bupa* datasets. *acl* could be considered as an easier dataset for basic methods (this has also been observed in [40]) while *bupa* could be characterized by other complex data factors apart from the presence of noisy and borderline examples.

Wilcoxon's test between SMOTE-IPF and *None* and the aligned Friedman's and Hochberg's multiple comparison tests comparing SMOTE-IPF along with the rest of re-sampling techniques are respectively found in Table 7 and Table 8.

From the AUC results of Table 6 and the statistical comparisons performed in Table 7 and Table 8, one should note that:

- The Wilcoxon's test between SMOTE-IPF and *None*, and between SMOTE-IPF and SMOTE

Table 6: AUC results obtained by C4.5 on real-world datasets with noisy and borderline examples.

Dataset	None	SMOTE	SMOTE-ENN	SMOTE-TL	SL-SMOTE	B1-SMOTE	B2-SMOTE	SMOTE-IPF
acl	88.75	86.75	86.75	88.00	85.25	**89.00**	88.00	88.50
breast	61.73	60.56	63.70	62.01	**64.72**	63.31	63.58	64.40
bupa	64.40	66.88	61.46	60.18	66.84	**68.60**	63.61	67.53
cleveland	52.58	54.85	57.22	**64.33**	60.07	54.75	56.66	62.82
ecoli	72.46	82.16	**89.97**	82.33	84.52	79.55	79.37	86.55
haberman	57.57	65.41	64.68	62.03	**67.07**	61.40	60.23	66.76
hepatitis	67.66	71.38	71.90	71.15	68.53	66.39	62.70	**72.25**
newthyroid	90.87	96.35	94.64	94.37	90.95	93.45	**97.18**	96.63
pima	70.12	71.29	71.40	69.48	**73.97**	70.94	73.77	73.58

Table 7: Wilcoxon's test results for the comparison of SMOTE-IPF (R^+) versus *None* and SMOTE (R^-) on real-world datasets considering the AUC results obtained by C4.5.

Methods	$\mathbf{R^+}$	$\mathbf{R^-}$	$\mathbf{p_{Wilcoxon}}$
SMOTE-IPF vs. None	44.0	1.0	0.0078
SMOTE-IPF vs. SMOTE	45.0	0.0	0.0039

Table 8: Multiple statistical comparison with real-world datasets.

	Algorithm	**Rank**	$\mathbf{p_{Hochberg}}$
Filtering	SMOTE-IPF	6.89	-
	SMOTE-ENN	16.00	0.014890
	SMOTE-TL	19.11	0.002178
	$\mathbf{p_{AlignedFriedman}}$	0.037679463832	
Change-Dir.	SMOTE-IPF	9.33	-
	SL-SMOTE	16.11	0.172352
	B1-SMOTE	23.67	0.007804
	B2-SMOTE	24.89	0.005208
	$\mathbf{p_{AlignedFriedman}}$	0.064566807525	

again clearly shows an improvement in the results obtained with respect to not preprocessing or applying SMOTE alone.

- SMOTE-IPF only obtains the best results on one single dataset (*hepatitis*) considering all the preprocessing methods. However, the Friedman's rank of SMOTE-IPF is clearly the best result

compared with the rest of re-sampling techniques if the performance of all the datasets is summarized. This shows the great robustness of SMOTE-IPF when applied to the real-world datasets.

- The p-values of the aligned Friedman's test are very low in all the scenarios, which shows a great significance in the differences found.

- The adjusted p-values by Hochberg's test are also very low in all comparisons, particularly with the *filtering-based* methods, even though with SL-SMOTE the p-value obtained is a little higher. Therefore, the suitability of SMOTE-IPF to address imbalanced real-world datasets with noisy and borderline examples is statistically shown.

Comparing the results of the synthetic datasets in Table 5 with respect to those of the real-world ones in Table 8, one observes that some of the methods that obtain the better Friedman's ranks on synthetic datasets, such as SMOTE-TL among the *filtering-based* methods, obtain less notable results on real-world ones. The latter data are perhaps more complex than the former and they may require more elaborate techniques. SMOTE-IPF remains the best method considering its aligned Friedman's rank and the Hochberg's test p-values with both synthetic and real-world datasets.

5.3. Results on noisy modified real-world datasets

This section is devoted to the analysis of the AUC results obtained by C4.5 on the noisy modified real-world datasets (see Table 9). From this table, one can observe that:

- Outperforming the results of *None* by the re-sampling methods is often more difficult when dealing with more noisy real-world datasets than with real-world datasets without additional noise. For instance, the performance of *None* on the $newthyroid_{x=20\%}$ and $pima_{x=40\%}$ datasets with class noise are the best results with respect to considering the use of preprocessing.

- The observation of the best results in each single dataset show that SMOTE-IPF is the best method in 8 of 18 class noise datasets, whereas it is the best in 9 of 18 attribute noise datasets. SMOTE-ENN and B2-SMOTE are also notable, with each obtaining 3 of 18 of the best results in class noise datasets and 2 of 18 in attribute noise datasets.

Table 10 shows the results of comparing the application of Wilcoxon's test to SMOTE-IPF with *None* and SMOTE, whereas Table 11 shows the aligned Friedman's test and the Hochberg's test results comparing SMOTE-IPF with respect to the rest of re-sampling techniques.

From these tables, one can make some observations:

- The need to apply advanced preprocessing techniques is shown by the low p-values obtained in Table 10, even though a slightly higher p-value (0.1551) is obtained comparing SMOTE-IPF and *None* with 20% of class noise.

Table 9: AUC results obtained by C4.5 on noisy modified real-world datasets with both class noise and attribute noise.

	Dataset	None	SMOTE	SMOTE-ENN	SMOTE-TL	SL-SMOTE	B1-SMOTE	B2-SMOTE	SMOTE-IPF
Class Noise	acl$_{x=20\%}$	86.50	87.25	81.75	82.00	85.50	83.75	81.75	**88.25**
	breast$_{x=20\%}$	63.32	63.44	62.96	59.78	64.44	61.31	**67.12**	64.15
	bupa$_{x=20\%}$	60.75	63.35	56.05	63.47	62.89	61.94	**64.81**	61.46
	cleveland$_{x=20\%}$	66.07	61.47	50.00	**70.93**	60.33	59.79	58.83	63.51
	ecoli$_{x=20\%}$	73.58	75.86	**83.55**	75.45	78.38	75.64	79.37	78.42
	haberman$_{x=20\%}$	62.18	62.61	60.95	53.97	**62.65**	59.82	61.03	62.25
	hepatitis$_{x=20\%}$	70.37	62.09	66.90	71.68	69.87	62.10	60.89	**77.58**
	newthyroid$_{x=20\%}$	**92.06**	87.98	90.44	79.40	89.92	84.17	89.72	88.21
	pima$_{x=20\%}$	68.99	72.70	71.63	66.09	67.28	69.69	70.09	**73.08**
	acl$_{x=40\%}$	68.50	71.00	74.00	62.50	66.75	66.75	73.50	**74.75**
	breast$_{x=40\%}$	56.41	57.26	**61.87**	53.65	55.54	57.28	59.52	55.86
	bupa$_{x=40\%}$	55.03	52.10	55.29	50.68	53.42	55.53	**56.59**	56.02
	cleveland$_{x=40\%}$	57.02	58.97	61.41	59.57	52.65	59.19	59.94	**62.98**
	ecoli$_{x=40\%}$	60.76	55.65	69.61	53.36	56.48	61.10	61.89	**71.12**
	haberman$_{x=40\%}$	50.00	50.00	53.16	51.19	50.00	50.00	50.00	**61.92**
	hepatitis$_{x=40\%}$	50.00	53.60	56.49	58.57	61.79	56.67	50.00	**65.54**
	newthyroid$_{x=40\%}$	50.00	51.03	**57.58**	53.53	51.03	56.59	51.03	53.29
	pima$_{x=40\%}$	**62.63**	60.75	54.19	53.40	61.72	61.56	62.16	61.85
Attribute Noise	acl$_{x=20\%}$	73.25	68.25	73.00	68.25	70.75	72.75	**74.00**	70.00
	breast$_{x=20\%}$	54.56	**64.03**	63.40	55.47	60.77	61.01	59.46	62.81
	bupa$_{x=20\%}$	53.10	51.47	56.23	51.31	55.35	52.36	55.23	**58.41**
	cleveland$_{x=20\%}$	53.33	57.30	56.54	58.62	57.34	54.28	54.66	**63.89**
	ecoli$_{x=20\%}$	59.93	71.70	64.81	62.65	**74.03**	65.61	67.48	72.89
	haberman$_{x=20\%}$	55.68	59.09	**63.25**	57.42	58.07	60.42	59.41	62.81
	hepatitis$_{x=20\%}$	78.26	65.10	72.69	70.99	75.58	74.52	**85.23**	78.74
	newthyroid$_{x=20\%}$	80.60	87.26	83.61	83.02	85.32	86.90	84.05	**87.58**
	pima$_{x=20\%}$	66.71	65.85	67.64	63.72	63.99	63.17	64.80	**69.99**
	acl$_{x=40\%}$	83.25	79.50	79.75	80.75	84.00	82.50	79.50	**86.25**
	breast$_{x=40\%}$	52.00	55.52	**60.14**	50.99	56.51	57.28	53.08	56.53
	bupa$_{x=40\%}$	57.40	61.28	58.98	54.97	59.88	**61.66**	57.54	57.93
	cleveland$_{x=40\%}$	59.99	50.00	58.74	62.73	64.61	50.94	61.52	**65.69**
	ecoli$_{x=40\%}$	65.81	70.78	74.13	**74.56**	72.14	70.46	67.61	73.02
	haberman$_{x=40\%}$	50.18	62.88	63.23	61.21	61.30	62.33	58.60	**66.22**
	hepatitis$_{x=40\%}$	76.61	63.34	65.08	61.39	70.78	69.29	70.06	**77.34**
	newthyroid$_{x=40\%}$	83.77	89.52	88.13	90.16	87.58	88.93	90.04	**91.55**
	pima$_{x=40\%}$	62.87	63.84	65.48	65.42	**68.47**	64.26	63.99	66.91

- SMOTE-IPF is established with both types of noise (class and attribute noise), both noise levels

Table 10: Wilcoxon's test results for the comparison of SMOTE-IPF (R^+) versus *None* and SMOTE (R^-) on real-world datasets with both class noise and attribute noise considering the AUC results obtained by C4.5.

	Class Noise - 20%			**Class Noise - 40%**			**Attribute Noise - 20%**			**Attribute Noise - 40%**		
Methods	$\mathbf{R^+}$	$\mathbf{R^-}$	$\mathbf{p_{Wilcoxon}}$	$\mathbf{R^+}$	$\mathbf{R^-}$	$\mathbf{p_{Wilcoxon}}$	$\mathbf{R^+}$	$\mathbf{R^-}$	$\mathbf{p_{Wilcoxon}}$	$\mathbf{R^+}$	$\mathbf{R^-}$	$\mathbf{p_{Wilcoxon}}$
SMOTE-IPF vs. None	34.0	11.0	0.1551	42.0	3.0	0.0195	43.0	2.0	0.0117	45.0	0.0	0.0039
SMOTE-IPF vs. SMOTE	37.0	8.0	0.0977	43.0	2.0	0.0117	42.0	3.0	0.0195	39.0	6.0	0.0546

Table 11: Multiple statistical comparison with real-world datasets with both class noise and attribute noise.

	Class Noise - 20%		**Class Noise - 40%**		**Attribute Noise - 20%**		**Attribute Noise - 40%**	
Algorithm	**Rank**	$\mathbf{p_{Hochberg}}$	**Rank**	$\mathbf{p_{Hochberg}}$	**Rank**	$\mathbf{p_{Hochberg}}$	**Rank**	$\mathbf{p_{Hochberg}}$
SMOTE-IPF	9.22	-	7.56	-	6.11	-	7.44	-
SMOTE-ENN	15.56	0.090521	12.89	0.154044	13.67	0.043455	16.00	0.022221
SMOTE-TL	17.22	0.065019	21.56	0.000366	22.22	0.000033	18.56	0.005964
$\mathbf{p_{AlignedFriedman}}$	0.03280041552		0.040217689132		0.043181723863		0.037237166607	
SMOTE-IPF	11.00	-	9.11	-	9.11	-	8.89	-
SL-SMOTE	15.56	0.359013	26.11	0.001859	20.33	0.025274	16.00	0.152201
B1-SMOTE	28.44	0.001332	20.39	0.046325	24.33	0.006531	21.33	0.024445
B2-SMOTE	19.00	0.214458	18.39	0.061755	20.22	0.025274	27.78	0.000428
$\mathbf{p_{AlignedFriedman}}$	0.065317774636		0.067460983239		0.06469719834		0.069232529882	

(20% and 40%) and with both types of techniques (*filtering-based* and *change-direction* methods) as the control algorithm because it obtains the best aligned Friedman's rank.

- The p-values of the aligned Friedman's test are low with both class and attribute noise. However, the most significant differences are found when comparing against the *filtering-based* techniques rather than against the *change-direction* ones.

- The adjusted p-values by Hochberg's test are generally low in all comparisons, with the exception of 20% of class noise in the *change-direction* group of methods, in which the highest p-values are obtained. The p-values are lower in the case of the attribute noise datasets than in the class noise datasets for a noise level of 20%. However, increasing the noise level up to 40% makes the differences found more significant independently of the type of noise.

Therefore, SMOTE-IPF also performs well in this scenario with these datasets, with additional noise induced as the AUC results and statistical comparisons show. The better performance and statistical significance of the results of SMOTE-IPF with the attribute noise datasets must be pointed out. This

property seems to be important due to the fact that attribute noise is much more common than class noise in real-world datasets, as indicated in [57].

5.4. Analysis of the noise filters in imbalanced datasets preprocessed with SMOTE

Since the only difference among all the existing *filtering-based* methods is the type of noise filter used, in this section the results of the filtering process are analyzed. Four additional criteria will be studied based on the percentage of examples removed by the noise filters after preprocessing with SMOTE. We analyze the percentage of examples filtered with respect to the set of: (i) all the examples (*%Total*), (ii) the synthetic positive examples created by SMOTE (*%Synt*), (iii) the original positive examples (*%Pos*) and (iv) the original negative examples (*%Neg*). For the sake of simplicity, only ENN and IPF will be compared and also only the noise level (20%) will be studied, even though the results of the other filtering method (TL) and the other noise level (40%) are found in the web-page associated with this paper.

Table 12 shows the percentage of examples removed in synthetic datasets with borderline examples. The analysis of these results leads to the following observations:

- ENN usually removes more examples (*%Total*) in *sub-cluster* and *paw* datasets, whereas IPF removes more examples only in *clover* datasets. One must note that *clover* datasets have more non-linear shapes of the minority class than the rest of the types of datasets.
- The *%Synt* results show similar conclusions to those of *%Total* results. It is important to point out that ENN does not remove any examples in *clover* datasets.
- ENN usually removes large quantities of examples of the original positive examples (*%Pos*), whereas IPF removes very low quantities independently of the type of dataset.
- IPF usually removes more examples of the negative class (*%Neg*) in *sub-cluster* and *paw* datasets, whereas it removes less examples in *clover* datasets.

Table 13 shows the percentage of examples removed in real-world datasets with noisy and borderline examples. These results show the following points:

1. **Real-world datasets.** IPF removes more synthetic positive examples (*%Synt*) and more original negative examples (*%Neg*) in almost all the datasets, where as it always removes less original positive examples (*%Pos*).
2. **Noisy modified real-world datasets.** IPF removes more synthetic examples (*%Synt*) in almost all the datasets and it removes less original positive examples (*%Pos*) with both types of noise. However, there is an important difference in the percentage of negative examples removed *%Neg*

Table 12: Percentages of examples removed by ENN and IPF after preprocessing the synthetic datasets with SMOTE.

	ENN	IPF	ENN	IPF	ENN	IPF	ENN	IPF
Dataset	**%Total**		**%Synt**		**%Pos**		**%Neg**	
sub-cluster $_{IR=5,\ DR=0}$	**14.30**	7.65	**9.00**	6.25	**19.00**	0.00	**17.60**	10.30
sub-cluster $_{IR=5,\ DR=30}$	**17.83**	14.65	**10.19**	9.44	**41.50**	3.00	19.20	**21.15**
sub-cluster $_{IR=5,\ DR=50}$	**20.88**	17.00	**11.44**	6.69	**50.00**	1.75	22.60	**28.30**
sub-cluster $_{IR=5,\ DR=60}$	**18.23**	16.72	**9.31**	8.63	**49.50**	1.50	19.10	**26.25**
sub-cluster $_{IR=5,\ DR=70}$	**19.90**	17.93	**10.69**	7.50	**49.00**	1.25	21.45	**29.60**
sub-cluster $_{IR=7,\ DR=0}$	**12.73**	4.64	6.96	**7.08**	**22.75**	0.00	**16.25**	3.21
sub-cluster $_{IR=7,\ DR=30}$	**15.68**	14.23	**8.67**	5.42	**45.50**	2.00	17.43	**23.54**
sub-cluster $_{IR=7,\ DR=50}$	**17.27**	15.52	**9.08**	6.83	**54.50**	2.50	18.96	**24.82**
sub-cluster $_{IR=7,\ DR=60}$	16.11	**17.45**	**9.25**	5.42	**54.50**	0.25	16.50	**30.21**
sub-cluster $_{IR=7,\ DR=70}$	16.89	**18.29**	**7.92**	5.42	**57.75**	0.25	18.75	**31.89**
clover $_{IR=5,\ DR=0}$	3.05	**8.80**	0.00	**2.19**	**13.00**	1.25	3.50	**15.60**
clover $_{IR=5,\ DR=30}$	6.75	**11.83**	0.00	**3.94**	**37.25**	0.50	6.05	**20.40**
clover $_{IR=5,\ DR=50}$	9.05	**13.75**	0.00	**3.81**	**53.00**	1.50	7.50	**24.15**
clover $_{IR=5,\ DR=60}$	10.18	**13.10**	0.00	**2.88**	**60.50**	0.25	8.25	**23.85**
clover $_{IR=5,\ DR=70}$	10.87	**14.93**	0.00	**3.69**	**65.50**	0.75	8.65	**26.75**
clover $_{IR=7,\ DR=0}$	2.64	**6.05**	0.00	**1.54**	**19.75**	1.00	2.46	**10.64**
clover $_{IR=7,\ DR=30}$	4.77	**10.84**	0.00	**3.04**	**44.25**	0.00	3.21	**19.07**
clover $_{IR=7,\ DR=50}$	6.98	**11.75**	0.00	**2.75**	**63.50**	1.50	4.89	**20.93**
clover $_{IR=7,\ DR=60}$	8.13	**12.00**	0.00	**2.62**	**71.25**	0.50	6.07	**21.68**
clover $_{IR=7,\ DR=70}$	8.09	**12.70**	0.00	**2.88**	**75.25**	0.50	5.43	**22.86**
paw $_{IR=5,\ DR=0}$	**7.75**	4.08	**2.06**	1.25	**7.50**	0.25	**12.35**	7.10
paw $_{IR=5,\ DR=30}$	**14.48**	9.25	**7.50**	1.94	**20.75**	1.25	**18.80**	16.70
paw $_{IR=5,\ DR=50}$	**17.20**	9.58	**9.00**	2.75	**23.50**	0.25	**22.50**	16.90
paw $_{IR=5,\ DR=60}$	**16.25**	10.85	**5.06**	2.31	**28.00**	1.25	**22.85**	19.60
paw $_{IR=5,\ DR=70}$	**17.38**	10.47	**7.56**	1.69	**30.75**	1.00	**22.55**	19.40
paw $_{IR=7,\ DR=0}$	**7.45**	4.02	**2.75**	1.50	**7.75**	0.25	**11.43**	6.71
paw $_{IR=7,\ DR=30}$	**13.21**	8.34	**5.71**	2.54	**22.25**	0.50	**18.36**	14.43
paw $_{IR=7,\ DR=50}$	**14.98**	8.98	**6.38**	2.92	**26.00**	0.75	**20.79**	15.36
paw $_{IR=7,\ DR=60}$	**14.46**	10.93	**4.63**	2.42	**32.25**	1.00	**20.36**	19.64
paw $_{IR=7,\ DR=70}$	**15.50**	10.68	**5.79**	2.17	**36.50**	0.50	**20.82**	19.43

in both types of noise: IPF generally removes less negative examples than ENN with class noise datasets and more examples with attribute noise datasets.

The percentages of the total number of examples removed *%Total* do not provide significant results

in any of the aforementioned cases. Furthermore, it is important to remark that, in many datasets, ENN does not remove any synthetic examples.

Table 13: Percentages of examples removed by ENN and IPF after preprocessing the real-world and noisy modified real-world datasets with SMOTE.

		ENN	IPF	ENN	IPF	ENN	IPF	ENN	IPF
	Dataset	**%Total**		**%Synt**		**%Pos**		**%Neg**	
Real-world	acl	7.50	**15.00**	0.42	**22.50**	**26.88**	7.50	4.00	**13.50**
	breast	**37.06**	29.97	**38.50**	30.72	**40.42**	31.48	**34.82**	28.91
	bupa	**33.81**	23.50	0.00	**22.73**	**46.55**	27.07	**33.88**	21.13
	cleveland	8.92	**9.26**	0.00	**3.42**	**87.86**	29.29	6.11	**11.65**
	ecoli	5.73	**7.85**	0.00	**4.51**	**48.57**	2.14	5.82	**11.46**
	haberman	20.44	**27.17**	13.02	**24.98**	**52.46**	21.32	13.67	**30.67**
	hepatitis	10.19	**17.18**	0.00	**18.01**	**49.67**	1.67	10.73	**19.51**
	newthyroid	1.46	**4.44**	0.00	**8.28**	**9.29**	1.43	1.11	**1.94**
	pima	**21.10**	20.67	0.65	**17.89**	**42.82**	19.31	18.95	**22.70**
Class Noise	$acl_{x=20\%}$	**22.58**	20.88	4.22	**32.05**	**38.82**	30.63	**16.50**	10.70
	$breast_{x=20\%}$	**42.62**	35.42	34.52	**38.25**	**47.09**	46.70	**40.98**	25.88
	$bupa_{x=20\%}$	**41.74**	33.69	0.00	**24.17**	**48.10**	39.56	**38.96**	28.64
	$cleveland_{x=20\%}$	**24.54**	22.04	0.00	**12.45**	**75.96**	51.47	**19.14**	16.24
	$ecoli_{x=20\%}$	19.09	**25.37**	0.00	**26.48**	**60.74**	29.16	15.88	**23.22**
	$haberman_{x=20\%}$	**35.39**	31.90	14.22	**28.94**	**50.20**	33.52	**32.03**	31.70
	$hepatitis_{x=20\%}$	19.77	**31.22**	0.00	**34.18**	**51.80**	30.74	15.91	**30.01**
	$newthyroid_{x=20\%}$	16.69	**21.93**	0.00	**34.52**	**49.28**	36.22	**10.10**	8.57
	$pima_{x=20\%}$	**33.83**	26.77	2.52	**25.85**	**40.32**	26.40	**31.97**	27.21
Attribute Noise	$acl_{x=20\%}$	18.00	**19.88**	2.50	**23.75**	**54.37**	13.75	12.75	**20.00**
	$breast_{x=20\%}$	**35.59**	31.09	**37.81**	34.75	31.16	**34.88**	**36.10**	27.37
	$bupa_{x=20\%}$	**40.44**	32.25	0.00	**23.64**	**58.79**	26.38	38.25	**38.88**
	$cleveland_{x=20\%}$	8.25	**10.49**	0.00	**4.73**	**85.71**	32.14	5.06	**12.59**
	$ecoli_{x=20\%}$	8.72	**11.09**	0.94	**4.98**	**52.14**	3.57	10.55	**17.36**
	$haberman_{x=20\%}$	21.56	**26.94**	10.77	**25.52**	**52.77**	18.83	17.22	**30.78**
	$hepatitis_{x=20\%}$	11.05	**21.48**	0.00	**24.21**	**60.67**	4.00	10.20	**22.71**
	$newthyroid_{x=20\%}$	5.00	**9.58**	0.00	**14.66**	**40.00**	5.00	2.22	**6.39**
	$pima_{x=20\%}$	**29.23**	25.95	6.25	**23.06**	**52.42**	23.32	27.45	**28.70**

From the results shown in this section, it can be observed that an important characteristic of the filtering performed by IPF is that it usually removes less original positive examples than ENN. One must not forget that, although noisy examples exist in the original dataset, the minority class is usually under-represented and the classifier should mainly reflect the properties of these original examples. Thus, being more conservative with regard to removing too many examples from the minority class

seems to be an advantage.

IPF removes more synthetic positive examples created by SMOTE than ENN in real-world and noisy modified real-world datasets, along with the more complex non-linear synthetic datasets (*clover*). These types of datasets are the most complex considered in this paper and it is therefore more likely that the synthetic examples introduced by SMOTE will be noisy.

With noisy real-world datasets - which are more likely to suffer from attribute noise than from class noise [57] - and noisy modified real-world datasets with attribute noise, the number of original negative examples removed by IPF is larger than with other noise filters, whereas with class noise datasets the quantity of negative examples removed by IPF is generally fewer. This is due to real-world and attribute noise datasets having larger quantities of negative examples suffering from noise than class noise datasets. It seems to be logical to delete more negative examples in these types of datasets than in class noise datasets and this is indeed the behavior of IPF. Therefore, this fact leads one to think that IPF performs a more accurate filtering than other noise filters in these scenarios.

6. SMOTE-IPF: suitability of the approach, strengths and weaknesses

This section summarizes the main conclusions obtained in the experimental section. Section 6.1 outlines the results obtained with the different types of datasets. Then, Section 6.2 describes the characteristics of IPF that make it suitable for this type of problem when preprocessed with SMOTE. Section 6.3 analyzes the main drawback of SMOTE-IPF, its parametrization. Finally, Section 6.4 establishes a hypothesis in order to explain its good behavior in the different scenarios considered.

6.1. Results obtained with the different types of datasets

The experimental results shows that SMOTE-IPF statistically outperforms the rest of the methods with all the types of datasets considered: synthetic datasets with borderline examples, real-world datasets with noisy and borderline examples and noisy modified real-world datasets. It is particularly suitable for the most complex types of problems: the non-linear synthetic datasets and the noisy modified real-world problems. Within the latter group, it particularly stands out with attribute noise datasets, which is the most common type of noise [57] and, in this case, it is also the most disruptive one due it affecting both classes, whereas class noise only affects the majority class. The increase in the noise level makes the differences in favor of SMOTE-IPF still more remarkable.

6.2. Characteristics of IPF and suitability for problems preprocessed with SMOTE

One important fact that makes IPF suitable for imbalanced datasets with noisy and borderline examples preprocessed with SMOTE is its *iterative elimination of noisy examples.* This fact implies that the

examples removed in one iteration do not influence detection in subsequent ones, resulting in a more accurate noise filtering.

Furthermore, the *ensemble nature* of IPF enables it to collect predictions from different classifiers which may provide a better estimation of difficult noisy examples, as apposed to collecting information from a single classifier only [7]. IPF also enables the *creation of more different classifiers* - using random partitions, for example - than other ensemble-based filters due to it providing more freedom when creating the partitions from which these classifiers are built. Creating diversity among the classifiers built is a key factor when ensembles of classifiers are used [34]. Finally, unlike other ensemble-based filters such as EF, which uses different classification algorithms to build the classifiers, IPF *only requires one classification algorithm* and is thus simpler.

6.3. On the parametrization of IPF

The choice of the different parameters of IPF can be seen as its main drawback, since there are numerous parameters and the behavior of the filter is quite dependent on their values. From our many experiments we can draw several conclusions regarding the influence of the different parameters on the performance results.

We have confirmed that using the majority scheme leads to better results than the consensus scheme since the number of noisy and borderline examples is large enough in comparison with the quantity of safe examples. The consensus scheme is very strict in removing examples and does not enable one to remove enough examples to change the performance significantly.

Regarding the number of partitions, a larger number usually implies better noise detection (and also a higher preprocessing time) since the voting scheme depends on more information. It is recommended that this number be odd, in order to avoid ties in the votes of the classifiers. Considering $n = 9$ partitions leads to a good balance between computational cost and performance.

The way to build the partitions enables one to control the diversity among classifiers. We have tested different strategies to create the partitions, such as stratified cross-validation - e.g. EF or CVCF - or random partitions. Random partitions were considered because they lead to better performance results since, as was expected, the partitions and, therefore the classifiers built, are more different.

The rest of the parameters allow a wider range of possibilities obtaining similar performances. Standard parameters recommended by the authors of IPF also work well with our SMOTE-preprocessed imbalanced datasets, so they are fixed to $k = 3$ iterations for the stop criterion and $p = 1\%$ for the percentage of removed examples.

6.4. Hypothesis to explain the good behavior of IPF with respect to other filters

The properties of IPF seem to be well adapted to the removal of noisy and borderline examples, implying an advantage over other noise filters. Most of the noise filters combined with SMOTE, such

as ENN or TL, have a noise identification based on distances among examples to their nearest neighbors, taking into account their classes. This issue, although overlooked in the literature, may be an important drawback: since SMOTE is based on the distance to the nearest neighbors to create a new positive example, such synthetic examples introduced by SMOTE, although noisy, are highly likely to not be identified by filters based on distances to the nearest neighbors. Using a noise identification method based on more complex rules, such as IPF, enables one to group larger quantities of examples with similar characteristics, although exceptions exist that will be considered to be noise, avoiding the aforementioned problem and detecting noisy examples easily.

7. Concluding remarks

This paper has focused on the presence of noisy and borderline examples, which is an important and contemporary research issue for learning classifiers from imbalanced data. It has been proposed to extend SMOTE with a new element, the IPF noise filter, to control the noise introduced by the balancing between classes produced by SMOTE and to make the class boundaries more regular. The suitability of the approach in this scenario has been analyzed.

Synthetic imbalanced datasets with different shapes of the minority class, imbalance ratios and levels of borderline examples have been considered. A set of real-world imbalanced datasets with different quantities of noisy and borderline examples and other factors have been also considered. Additional noise has been introduced into the latter considering two noise schemes: a class noise scheme and an attribute noise scheme. All these datasets have been preprocessed with our proposal and several resampling techniques that can be found in the literature. Finally, the C4.5 algorithm has been tested over these preprocessed datasets.

The values of the AUC measure have shown that our proposal has a notably better performance when dealing with imbalanced datasets with noisy and borderline examples with both synthetic and real-world datasets. SMOTE-IPF also outperforms the rest of the methods with the real-world datasets with additional noise. Our proposal especially outperforms the rest of the methods with the more complex to learn datasets in each group of datasets: the non-linear synthetic datasets and the attribute noise real-world datasets. These observations are supported by statistical tests.

The experiments performed with others classifiers (k-NN, SVM, RIPPER and PART) have provided similar results and conclusions on the superiority of SMOTE-IPF to those shown for C4.5. Although statistical differences are less significant using SVM and RIPPER with attribute noise datasets, SMOTE-IPF obtains the better performances and the aligned Friedman's ranks. The especially good results of SMOTE-IPF on the synthetic datasets using RIPPER and on the class noise datasets using k-NN must also be pointed out. These results have been better than those of C4.5 shown here.

One must consider that the ensemble-nature of IPF, which constitutes a robust and accurate way of detecting mislabeled examples, the iterative noise detection and elimination processes carried out and the possibility of controlling the diversity between classifiers are the key points of IPF which finally produce a more accurate filtering process. All these factors help SMOTE-IPF to obtain better performances than other re-sampling techniques in the scenarios considered.

Acknowledgment

Supported by the National Project TIN2011-28488, and also by the Regional Projects P10-TIC-06858 and P11-TIC-9704. José A. Sáez holds an FPU scholarship from the Spanish Ministry of Education and Science.

References

[1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2011) 255–287.

[2] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing - A Fusion of Foundations, Methodologies and Applications 13 (2009) 307–318.

[3] R. Barandela, J.S. Sánchez, V. García, E. Rangel, Strategies for learning in class imbalance problems, Pattern Recognition 36 (2003) 849–851.

[4] G. Batista, R. Prati, M. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explorations Newsletter 6 (2004) 20–29.

[5] U. Bhowan, M. Johnston, M. Zhang, Developing new fitness functions in genetic programming for classification with unbalanced data, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42 (2012) 406–421.

[6] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (1997) 1145–1159.

[7] C.E. Brodley, M.A. Friedl, Identifying Mislabeled Training Data, Journal of Artificial Intelligence Research 11 (1999) 131–167.

[8] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem, in: Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 475–482.

[9] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[10] N.V. Chawla, D.A. Cieslak, L.O. Hall, A. Joshi, Automatically countering imbalance and its empirical relationship to cost, Data Mining and Knowledge Discovery 17 (2008) 225–252.

[11] N.V. Chawla, N. Japkowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, SIGKDD Explorations 6 (2004) 1–6.

[12] W.W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann Publishers, 1995, pp. 115–123.

[13] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.

[14] J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 7 (2006) 1–30.

[15] A. Fernández, M.J. del Jesus, F. Herrera, On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets, Information Sciences 180 (2010) 1268–1291.

[16] E. Frank, I.H. Witten, Generating accurate rule sets without global optimization, in: ICML98: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, pp. 144–151.

[17] D. Gamberger, R. Boskovic, N. Lavrac, C. Groselj, Experiments With Noise Filtering in a Medical Domain, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, 1999, pp. 143–151.

[18] D. Gamberger, N. Lavrac, S. Dzeroski, Noise Detection and Elimination in Data Preprocessing: experiments in medical domains, Applied Artificial Intelligence 14 (2000) 205–223.

[19] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (2010) 2044–2064.

[20] V. García, R. Alejo, J. Sánchez, J. Sotoca, R. Mollineda, Combined effects of class imbalance and class overlap on instance-based classification, in: E. Corchado, H. Yin, V. Botti, C. Fyfe (Eds.), Intelligent Data Engineering and Automated Learning IDEAL 2006, volume 4224 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2006, pp. 371–378.

[21] V. García, R. Mollineda, J. Sánchez, On the k-NN performance in a challenging scenario of imbalance and overlapping, Pattern Analysis Applications 11 (2008) 269–280.

[22] V. García, J. Sánchez, R. Mollineda, An empirical study of the behavior of classifiers on imbalanced and overlapped data sets, in: Rueda, L., Mery, D., Kittler, J. (eds.) CIARP 2007. LNCS, volume 4756, Springer, Heidelberg, 2007, pp. 397–406.

[23] H. Han, W.Y. Wang, B.H. Mao, Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning, in: Proceedings of the 2005 International Conference on Advances in Intelligent Computing - Volume Part I, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 878–887.

[24] H. He, E. Garcia, Learning from imbalanced data, IEEE Transactions on Data and Knowledge Engineering 21 (2009) 1263–1284.

[25] Y. Hochberg, A sharper Bonferroni procedure for multiple tests of significance, Biometrika 75 (1988) 800–803.

[26] J. Hodges, E. Lehmann, Ranks methods for combination of independent experiments in analysis of variance, Annals of Mathematical Statistics 33 (1962) 482–497.

[27] K. Huang, H. Yang, I. King, M.R. Lyu, Imbalanced learning with a biased minimax probability machine, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36 (2006) 913–923.

[28] N. Japkowicz, Class imbalance: Are we focusing on the right issue?, in: II Workshop on Learning from Imbalanced Data Sets, ICML, Morgan Kaufmann Publishers Inc., 2003, pp. 17–23.

[29] T. Jo, N. Japkowicz, Class Imbalances versus small disjuncts, SIGKDD Explorations 6 (2004) 40–49.

[30] K.L. Kermanidis, The effect of borderline examples on language learning, Journal of Experimental & Theoretical Artificial Intelligence 21 (2009) 19–42.

[31] K.C. Khor, C.Y. Ting, S. Phon-Amnuaisuk, A cascaded classifier approach for improving detection rates on rare attack categories in network intrusion detection, Applied Intelligence 36 (2012) 320–329.

[32] T.M. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise filtering techniques, Journal of Computer Science and Technology 22 (2007) 387–396.

[33] M. Kubat, S. Matwin, Addressing the Curse of Imbalanced Training Sets: One-Sided Selection, in: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 179–186.

[34] L.I. Kuncheva, Diversity in multiple classifier systems, Information Fusion 6 (2005) 3–4.

[35] V. López, A. Fernández, F. Herrera, On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed, Information Sciences 257 (2014) 1–13.

[36] V. López, I. Triguero, C.J. Carmona, S. García, F. Herrera, Addressing imbalanced classification with instance generation techniques: IPADE-ID, Neurocomputing 126 (2014) 15–28.

[37] T. Maciejewski, J. Stefanowski, Local Neighbourhood Extension of SMOTE for Mining Imbalanced Data, in: Proceedings of IEEE Symposium on Computational Intelligence and Data Mining, SSCI IEEE, IEEE Press, 2011, pp. 104–111.

[38] R. Mathiasi Horta, B. Pires De Lima, C. Borges, A semi-deterministic ensemble strategy for imbalanced datasets (SDEID) applied to bankruptcy prediction, WIT Transactions on Information and Communication Technologies 40 (2008) 205–213.

[39] G.J. McLachlan, Discriminant Analysis and Statistical Pattern Recognition, John Wiley and Sons, 2004.

[40] K. Napierala, J. Stefanowski, S. Wilk, Learning from imbalanced data in presence of noisy and borderline examples, in: Rough Sets and Current Trends in Computing, volume 6086 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2010, pp. 158–167.

[41] J.R. Quinlan, C4.5: programs for machine learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.

[42] J.A. Sáez, M. Galar, J. Luengo, F. Herrera, Tackling the problem of classification with noisy data using Multiple Classifier Systems: Analysis of the performance and robustness, Information Sciences 247 (2013) 1–20.

[43] J.A. Sáez, M. Galar, J. Luengo, F. Herrera, Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition, Knowledge and Information Systems 38 (2014) 179–206.

[44] J.A. Sáez, J. Luengo, F. Herrera, Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification, Pattern Recognition 46 (2013) 355–364.

[45] K. Slowiński, J. Stefanowski, D. Siwiński, Application of rule induction and rough sets to verification of magnetic resonance diagnosis, Fundamenta Informaticae 53 (2002) 345–363.

[46] J. Stefanowski, Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data, in: S. Ramanna, L.C. Jain, R.J. Howlett (Eds.), Emerging Paradigms in Machine Learning, volume 13 of *Smart Innovation, Systems and Technologies*, Springer Berlin Heidelberg, 2013, pp. 277–306.

[47] J. Stefanowski, S. Wilk, Selective Pre-processing of Imbalanced Data for Improving Classification Performance, in: I.Y. Song, J. Eder, T. Nguyen (Eds.), Data Warehousing and Knowledge Discovery, volume 5182 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2008, pp. 283–292.

[48] C.T. Su, Y.H. Hsiao, An evaluation of the robustness of MTS for imbalanced data, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 1321–1332.

[49] A. Sun, E.P. Lim, Y. Liu, On strategies for imbalanced text classification using SVM: A comparative study, Decision Support Systems 48 (2009) 191–201.

[50] Y. Sun, M.S. Kamel, A.K. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognition 40 (2007) 3358–3378.

[51] Y. Tang, Y.. Zhang, N.V. Chawla, SVMs modeling for highly imbalanced classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39 (2009) 281–288.

[52] F.B. Tek, A.G. Dempster, I. Kale, Parasite detection and identification for automated thin blood film malaria diagnosis, Computer Vision and Image Understanding 114 (2010) 21–32.

[53] I. Tomek, Two Modifications of CNN, IEEE Transactions on Systems, Man and Communications 6 (1976) 769–772.

[54] S. Verbaeten, A.V. Assche, Ensemble methods for noise elimination in classification problems, in: Fourth International Workshop on Multiple Classifier Systems, Springer, 2003, pp. 317–325.

[55] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems, Man and Cybernetics 2 (1972) 408–421.

[56] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, Journal of Artificial Intelligence Research 6 (1997)

1–34.

[57] X. Zhu, X. Wu, Class Noise vs. Attribute Noise: A Quantitative Study, Artificial Intelligence Review 22 (2004) 177–210.

3.4 Class noise reparation by an aggregated noise filter ensemble voting algorithm

Sáez J. A., Luengo J., Shim S., and Herrera F. (2014) Class Noise Reparation by an Aggregated Noise Filter Ensemble Voting Algorithm (submitted)

- Status: **Submitted**.

Class Noise Reparation by an Aggregated Noise Filter Ensemble Voting Algorithm

José A. Sáez[a,*], Julián Luengo[b], Seong-O Shim[c], Francisco Herrera[a]

[a] *Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, Granada, Spain, 18071*
[b] *Department of Civil Engineering, LSI, University of Burgos, Burgos, Spain, 09006*
[c] *Faculty of Computing and Information Technology - North Jeddah, King Abdulaziz University, 21589, Jeddah, Saudi Arabia*

Abstract

Obtaining data in the real world is subject to imperfections. In the data collecting process, a common consequence of these imperfections is the appearance of noise. In classification, noisy data may deteriorate the performance of a classifier depending on the learning method sensitiveness to data corruptions. A particular disruptive type of noise in classification occurs when noise affects example class labeling, as it may severely mislead the model building.

Several strategies have emerged to deal with class noise in classification. Among the most popular is that of filtering. However, instance filtering can be harmful as it may eliminate more examples than necessary or produce loss of information. For this reason, we advance a new proposal based on an ensemble of noise filters with the goal to not only to filter the instances but correct, if possible, those that are mislabeled. Using the label that every base filter considers correct for a given instance, a voting scheme is applied to decide whether the instance label is filtered, relabeled or maintained. In order to avoid the negative influence of overlapped examples in class boundaries, we also study the use of ENN to clear such boundaries before creating the ensemble of filters.

Keywords:
Noisy Data, Class Noise, Noise Filters, Data Reparation, Data Polishing, Classification.

1. Introduction

The aim of classification algorithms is to extract implicit knowledge from previously known labeled examples of a problem by creating a model by induction, called a classifier, that is

*Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
Email addresses: smja@decsai.ugr.es (José A. Sáez), jluengo@ubu.es (Julián Luengo), seongo@gmail.com (Seong-O Shim), herrera@decsai.ugr.es (Francisco Herrera)

capable of predicting the class for new unlabeled examples. For this reason, the classification accuracy of a classifier is directly influenced by the quality of the labeled data used for its training. Data quality depends on several components [31] such as, for example, the source and the input procedure. As a consequence, data gathered from real-world problems are inherently subject to errors. Thus, limitations or disturbances of measurement instruments usually generate the occurrence of noise in the data [36], [23].

Noise is an unavoidable problem and the origin of errors that affect data collection and data preparation processes in Data Mining applications [34, 35]. Classifiers with built-in noisy frameworks will heavily depend on the quality of the training data and also on the robustness to noise of the classifier itself to obtain a good accuracy. Hence classification problems containing noise are complex and accurate solutions are often difficult to achieve, particularly if the classifier is noise-sensitive. Noise can be observed both in the input attributes or the class labels, and as a consequence two types of noise are usually distinguished in the literature: attribute and class noise [37].

Class noise (also known as *label noise*) is widely denoted as the most harmful type of noise to the classifier performance [37], due to incorrectly labeled examples severely biasing the learning method. These result in inaccurate models. Class noise usually appears mainly due two causes: (i) contradictory examples [14] – examples with identical input attribute values having different class labels –, and (ii) misclassifications [37], also known as mislabeled examples – examples that are incorrectly labeled. Detecting contradictory examples is easier than identifying misclassifications [37]. For this reason most of the literature, including this work, is focused on the study of misclassifications and, thus, the term *class noise* usually referred to this type of noise [28, 25]. The performance of classifiers built with mislabeled examples will depend on the amount of data with incorrect class labels, as well as the capacity of the methods to deal with class noise.

In order to reduce the effects produced by noise, two main approaches have been proposed in the specialized literature:

1. The use of *robust learners* methods, distinguished by being less affected by noisy examples. These classifiers have been developed or adapted to appropriately deal with noise [22, 4].
2. *Preprocessing* the datasets to remove or correct the noisy examples [3, 8] is the most common approach if no robust learner is available or eligible.

Adapting a learning method to make it robust against noise is not simple as it can be difficult and time consuming. It is also not often an available choice, although the use of multiple classifier systems have proved to be helpful to this respect [24]. As a consequence,

preprocessing to clean the data is usually the most popular choice to remove instances and has no effect in the model inference process [9].

The use of ensembles for filtering in the class noise framework has been studied for some decades [3], [30], [16], [27]. Brodley and Friedl [3] introduced this use claiming that if some examples have been mislabeled and it is assumed that the label errors are independent of particular classifiers inferred from the data, collecting predictions from different classifiers could provide a better estimation of mislabeled examples than collecting information from a single classifier. Based on this idea, Verbaeten and Assche proposed the *Cross-Validated Committees Filter* (CVCF) [30]. This filtering method used a cross-validation scheme to split the full training data and buildt classifiers in each training subset, which were later used to identify the potentially noisy examples. Khoshgoftaar and Rebours proposed an iterative noise filter called the *Iterative- Partitioning Filter* (IPF) [16], which removed noisy examples in multiple iterations until an stopping criterion was fulfilled. All these aimed to improve the accuracy of the classifiers to be induced from the data. One of the most recent research on noise identification based on ensembles is the work of Sluban et al. [27]. These authors proposed to create a ranking of noisy instances according to the predictions undertaken by several different noise detection algorithms, and later supervised by a domain expert. However, the main aim of the work of Sluban et al. is not the filtering of the noisy data but to achieve a noise and outlier identification to achieve an improved understanding of the data.

As indicated by Frénay and Verleysen [7], removing mislabeled instances is more efficient than repairing and relabeling them [19, 5]. However this removing may be excessive [17] and be aggravated in imbalanced classification where the minority class instances are prone to be easily misclassified [15]. In any case, maintaining the noisy instances as Brodley and Friedl [3] suggest, is even worse than overfiltering (or *overcleansing* [7]). Thus a balance between the excessive filtering and the conservation of every instance must be found. Very few proposals try to tackle this balance, so the problem remains mainly open [7].

In this paper we aim to provide a novel preprocessing technique capable of repairing the class noisy instances when possible as well act as a filter when the class label for the instance is undoubtedly noisy and cannot be safely relabeled. Relabeling noisy instances is performed by the use of an ensemble of specialized class noise filters. By aggregating the information they provide for each instance it is possible to repair the class label in many cases and discard fewer instances than that of the individual filters. The dual behavior of the method –label repairing and noise filtering– is intended to achieve the desired balance between overcleansing and maintaining questionable instances.

In order to verify the validity of the proposal, a thorough empirical study will be developed

using the *k-Nearest Neighbors* (k-NN) [18] classifier, which is considered very noise-sensitive, a learner that is robust to noise as is C4.5 [22], as well as a SVM classifier known to be precise but also for being noise sensitive [29]. The hypothesis of this proposal improving the performance of these classifiers with noisy data will be checked in detail and the conditions under which the proposal works well with noisy data will be analyzed. In order to reach meaningful conclusions based on the characteristics of the noise, a large collection of real-world datasets will be considered. We will introduce class noise into 25 base datasets to create a total of 175 datasets. It should be mentioned that, in real situations, the quantification of noise is for the most part impossible and therefore neither the initial type nor the quantity of noise in the dataset can be known or supposed *a priori*. For that reason, several class noise levels, from 5% to 30% with increments of 5%, will be introduced, since these are usually unknown in real-world data. The results obtained will be contrasted using Wilcoxon's statistical Test [13, 11] in order to identify the significance of the differences. We will also determine the ability of our proposal to identify the noisy inducted examples and how it is capable of correctly relabeling them by comparing them with the noise free dataset. Full results and details of the experimentations are available in the webpage associated with this paper `http://sci2s.ugr.es/class-noise-correction`.

The rest of this paper is organized as follows. Section 2 presents an introduction to classification with noisy data. Section 3 introduces the proposal that is able to repair class noise. Section 4 presents the experimental framework, and Section 5 analyzes the results obtained. Finally, Section 6 enumerates a series of concluding remarks.

2. Classification with noisy data

This section presents the concept of noisy data in the field of classification in Subsection 2.1. This is followed by Subsection 2.2 that presents noise filters in general, along with a description of the noise filters used in this paper.

2.1. Introduction to noisy data

Real-world data is far from being perfect and accurate. Errors made by measurement instruments, corruptions and inaccuracies may harm interpretation of data, the design of models and the making of decisions. In the particular scenario of this paper, noise can negatively affect the classifier performance in several aspects as building time, size and interpretability of the model obtained [36, 37].

The effects of how noise affects the characteristics of a classification problem can be observed by analyzing its spatial characteristics. Noise may create small clusters of instances of a particular class in the instance space corresponding to another class, displace or remove instances located in key areas within a concrete class or disrupt the boundaries of the classes resulting in an increased boundaries overlap. As an ultimate consequence, all these alterations result in a corruption of the knowledge extracted from the problem, spoiling the classifiers built from that noisy data especially when compared to the same classifiers built from equivalent clean data that more accurately builds a model from the problem [37].

In order to avoid or diminish the effects of noise, it is essential to identify the components that can be affected by noise and establish what extent they are affected. As described by Wang et al. [31], the quality of any dataset is determined by a large number of components. Among them, the class labels and the attribute values are two sources influencing the quality of a classification dataset [33]. The quality of the class labels represents whether the class of each instance is correctly assigned and the quality of the attributes indicates how well they characterize instances for classification purposes. Based on these two information sources, we can distinguish two types of noise in a given dataset [37, 33]:

1. *Class noise*. Also known as labeling errors or *label noise*, it occurs when an instance belongs to the incorrect class. Class noise can be attributed to several causes, including subjectivity during the labeling process, data entry errors, or inadequacy of the information used to label each instance. There are two possible types of class noise:
    - Contradictory instances: the same instances appear more than once and are labeled with different classes. A possible scenario for this type of class noise can occur when records processed within different systems bearing different data formats are merged [14] or by a faulty attribute selection/creation.
    - Misclassifications: are instances labeled with the wrong classes. In this paper we use this type of error for class noise due to its frequency in real-world data, since it generally occurs at the boundaries of the classes where different classes have similar attribute values.
2. *Attribute noise*. This is used to refer to corruptions in the values of one or more attribute of instances in a dataset. Examples of attribute noise include: erroneous attribute values, missing or unknown attribute values, and incomplete attributes or "do not care" values. In our work we treat as attribute noise the erroneous attribute values. This is due to the fact that, unlike other types of attribute noise such as missing values, this type has been little studied in the literature and is very common in real-world data [37].

Several methods have been studied in the literature to deal with noisy data and to obtain higher classification accuracies on test data:

- *Robust learners* [22]. These are techniques characterized by being less influenced by noisy data. An example of a robust learner is the C4.5 algorithm [22]. C4.5 uses pruning strategies to reduce the chances that the trees are overfitting to noise in the training data [21]. However, if the noise level is relatively high, even a robust learner may have a poor performance. By including C4.5 in our experiments, we have attempted to verify that the effect of noise on the performance of the classifiers built by robust learners may be lower by using our proposal.

- *Noise filters* [3, 16, 30]. They identify instances which can be eliminated from the training data. These are used with many learners sensitive to noisy data and require data preprocessing to address the problem. We propose to avoid the elimination of instances by means of aggregating the decisions of different filters and, if possible, relabel instead of deleting them.

- *Data polishing methods* [28]. Their aim is to correct noisy instances prior to training a learner. This option is only viable, due to the problem of time consumption, when datasets are small. Several works [37, 28] claim that complete or partial noise correction in training data, with test data still containing noise, improves test performance results in comparison with no preprocessing. Our proposal initially tries to repair the instance label when several filters agree on a label different to the current shown by the instance.

2.2. Noise filters

Noise filters are preprocessing mechanisms designed to detect and eliminate noisy examples in the training set [3, 16, 23]. The result of noise elimination in preprocessing is a reduced training set which is then used as an input to a machine learning algorithm. The separation of noise detection and learning has the advantage that noisy instances do not influence the classifier design [10].

Noise filters are generally oriented to detect and eliminate instances with class noise from the training data. Elimination of such instances has been shown to be advantageous [8]. However, the elimination of instances with attribute noise seems counterproductive [21, 37] since instances with attribute noise still contain valuable information in other attributes which can help to build the classifier.

Some of these filters are based on the computation of different measures on the training data. For instance, the method proposed by Gamberger et al. [10] is based on the notion

that the elimination of noisy examples reduces the *Complexity of the Least Complex Correct Hypothesis* value of the training set.

In addition, there are many other noise filters based on the use of ensembles. Brodley and Friedl [3] propose the use of multiple classifiers belonging to different learning paradigms and trained from a possibly corrupted dataset to identify mislabeled data, which are characterized as the examples that are incorrectly classified by the multiple classifiers. Similar techniques have been widely developed related to the building of several classifiers with the same learning algorithm [8, 30]. Instead of using multiple classifiers gleaned from the same training set, Gamberger et al. [8] suggest a *Classification Filter* (CF) approach, in which the training set is partitioned into n subsets, then a set of classifiers is trained from the union of any $n-1$ subsets; those classifiers are used to classify the examples in the excluded subset, eliminating the examples that are incorrectly classified.

For our proposal, we employ three noise filters designed to deal with mislabeled instances: the *Prototype Selection based on Relative Neighborhood Graphs*, the *Cross-Validated Committees Filter* and the *Iterative-Partitioning Filter*:

1. Prototype Selection based on Relative Neighborhood Graphs (**RNG**) [26]. This builds a proximity undirected graph $G = (V, E)$, in which each vertex in V corresponds to an example from the training set D_T. Two vertices are connected by an edge $(x_i, x_j) \in E$ if and only if x_i and x_j satisfy a neighborhood relation (see Equation 1, where d is the distance function). In this case, we state that these instances are *graph neighbors*. RNG discards those instances misclassified by their majority of graph neighbors.

$$(x_i, x_j) \in E \Leftrightarrow d(x_i, x_j) \leq max(d(x_i, x_k), d(x_j, x_k)), \forall x_k \in D_T, k \neq i, j. \quad (1)$$

2. Cross-Validated Committees Filter (**CVCF**) [30]. CVCF is mainly based on performing a Γ-fold cross-validation to split the full training data and on building classifiers using decision trees in each training subset. The training dataset D_T is split into Γ equal sized subsets and, for each of these Γ parts, C4.5 is trained with the other $\Gamma - 1$ parts. The examples in the training set D_T which are incorrectly labeled (following a voting scheme) among the Γ classifiers are removed. Two voting schemes can be used to identify the noisy examples: consensus and majority. The first removes an example if it is misclassified by all the classifiers, whereas the second removes an example if it is misclassified by more than half of the classifiers. Consensus filters are characterized by being conservative in rejecting good data at the expense of retaining bad data. Majority filters are better at detecting bad data at the expense of rejecting good data.

3. Iterative-Partitioning Filter (**IPF**) [16]. IPF removes noisy examples in multiple iterations until the quantity of examples eliminated in K_{IPF} successive iterations is low enough (it is less than a percentage P_{IPF} of the size of D_T). In each iteration, the current training dataset T_{IPF} is split into Γ equal sized subsets and the C4.5 classifier is built over each of these Γ subsets to evaluate T_{IPF}. Then, the examples incorrectly labeled of T_{IPF} are removed (according to one of the two aforementioned voting schemes) and a new iteration is begun.

3. Class noise reparation by an aggregated noise filters ensemble voting algorithm

Filtering is a successful approach but it is prone to produce some drawbacks. The elimination of examples in the dataset may be harmful, especially when the data is costly to obtain or incorrectly labeled as noise. A typical case of this occurs in imbalanced classification, where the minority class examples may be incorrectly recognized as noise. Excessive filtering can also produce smaller classifier models due to a smaller training dataset and this simplicity can actually yield an inaccurate model [20]. Filtering may also be dependant on the classification algorithm to be applied afterwards, as some robust learners are less sensitive to noise as we have mentioned and thus may benefit from a less aggressive filtering over the instances. Data polishing methods, on the other hand, is a perfect solution in these cases, keeping more instances in the dataset and allowing robust learners to benefit from a larger amount of data. However, the complete repair of the dataset is not always possible.

An intermediate approach is where the instances are repaired, i.e. relabeled if affected by class noise, when it is possible to establish that they are noisy and the class label they belong to with a high confidence degree. When the instance cannot be repaired due to the impossibility of determining the true class label, a safe filtering can be applied.

In this section we describe a preprocessing method that obeys the aforementioned way of working, focusing in relabeling and repairing the instances when possible. This is denoted *Class Noise Corrector* (CNC). In order to repair the instances we will use the filtering algorithms introduced in Section 2.2. Using several different noise filters enables the proposal to obtain, with respect to correct ones, a more founded discrimination over the noisy instances. It is essential to note that the filters are based on classifiers, so a class prediction is ultimately made by each noise filter. In ensemble class noise filters, when the base classifiers do not agree with the current class label of the treated example, they are considered noise and dropped from the dataset.

If several class noise filters are gathered together, their outputs for each instance can be

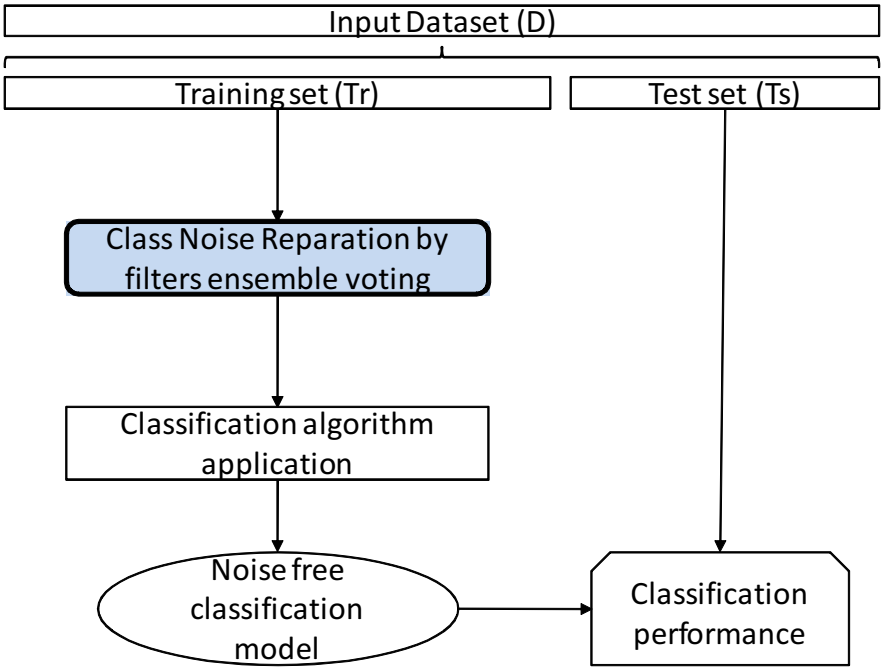


Figure 1: Main schema of the ensemble based class noise reparation filter CNC

compared. A simple way to aggregate the filter's output is the majority voting scheme:

1. When the label predicted by the noise filters is the same or the most voted and it matches the current example label, the label remains unchanged.
2. When the label predicted by all the noise filters is the same and it does not match the current example label, the label is changed to the predicted one. This case can be considered as a polishing or correction of the data.
3. When situations 1 and 2 do not hold, the example is considered noisy and it is erased from the dataset. This approach can potentially erase correct examples, but this is preferable to leaving noisy examples in the dataset [3].

In the experimentation carried out in this paper we will treat with our proposal (correcting or removing) those examples whose label predicted by the majority of the noise filters, that is, when two of the three noise filters do not match the current example label.

Figure 1 depicts the process, where several class noise filters can be used using a voting scheme over their outputs. Using the examples from the training set that is being considered, the first step is to apply the proposal in order to correct the noisy examples (marked in blue). This proposed technique is based in the decision yielded by several well-known filters, forming a filtering ensemble. Using an ensemble is motivated as the robustness of the decision taken

will be higher, taking advantage of the different approaches used to identify a noisy example. Once the Training data (Tr) is corrected, any suitable standard classification algorithm can be safely applied. It is expected that the generalization obtained by the model generated will be better than using the Tr data without applying the proposal. In order to correctly evaluate such an improvement in the generalization and accuracy abilities of the model, the remaining test data (Ts) is used. The noise-free model generated should be less sensible to overfitting, and thus obtain better results in noisy environments.

The following is an outline of the proposal:

1. First a copy of the training set Tr is cleaned using ENN [32]. By this procedure, the frontiers between the classes' hulls are clearly defined before "training" the filters. Our aim is to avoid borderline examples that negatively influence the filters construction.
2. Using this cleaned copy of Tr, a group of noise filtering methods are applied (RNG, CVCF and IPF). Thanks to the previous cleaning carried out in step 1, the filters thus created will rely on examples of each class which clearly belong to it.
3. Once all the filters are created, an aggregation over the filters output is performed:
   (a) If the majority of the filters indicate that the class label for the example is correct, the example is left as is.
   (b) If the majority of the filters indicate that the class label for the example is **other** than the current one, the example is relabeled with the consensual label.
   (c) If a consensus between the filters' output for the analyzed example is not reached, the example is considered as irreparable and it is deleted.

Graphically, this process can be summarized in Figure 2. As can be appreciated, the ENN filtered version of the training set Tr is used only to train the base classifiers contained in the noise filters. All the instances from Tr will be evaluated by the noise filter ensemble, no matter if they where taken out by ENN or not. Provided the use of three filters, at least two coincident votes are necessary to consider consensus between the filters' output.

The proposal has been implemented in KEEL, a Java-based open source Data Mining platform. All of the base filters' source code is currently available.

4. Experimental framework

This section presents the details of the experimental study carried out in order to check the behavior of our proposal. First, Section 4.1 describes the datasets used. Then, Section 4.2 presents the noise filters included in our proposal. Finally, Section 4.3 describes the methodology followed to analyze the results.

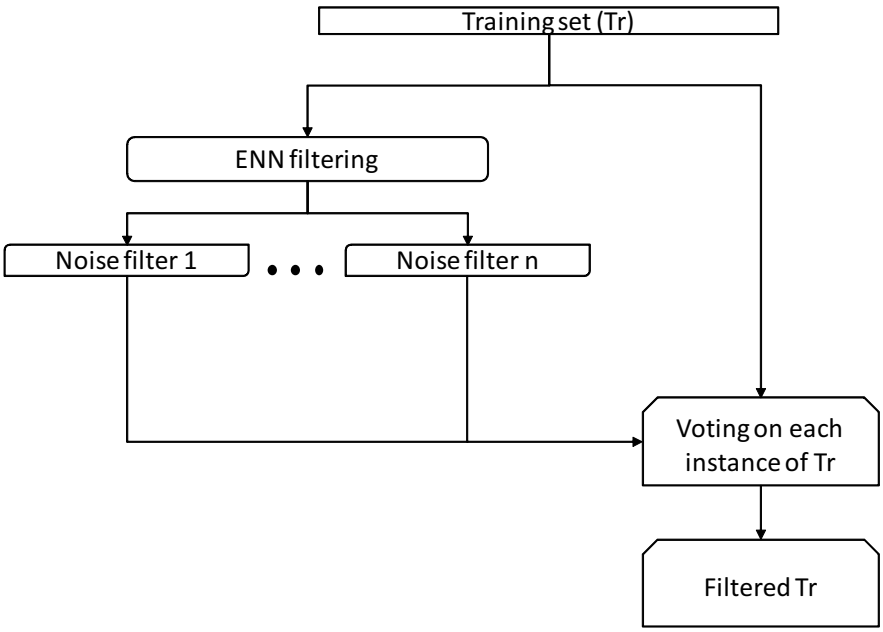


Figure 2: Detail of the CNC repairing process over the instances of the complete training set Tr.

4.1. Datasets

The experimentation is based on 25 datasets from the KEEL-Dataset repository [1]. They are described in Table 1, where #EXA refers to the number of examples, #ATT to the number of attributes and #CLA to the number of classes. Some of the largest datasets (nursery, shuttle and splice) were stratified at 10% in order to reduce the computational time required for training, given the large amount of executions carried out. Examples containing missing values are removed from the datasets before use.

In order to control the amount of noise in each dataset and check how it affects the classifiers, noise is introduced into each dataset. We introduce a class noise level $L\% = 100 * (\#noisyexamples)/\#examples$ into each complete dataset following an *uniform class noise scheme* [28] in which the $L\%$ of the examples are corrupted. The class labels of these examples are randomly replaced by another one from the C classes. In our experimentation, we will consider the noise levels ranging from $L = 0\%$ (base datasets) to $L = 30\%$, by increments of 5%. As a consequence, 150 noisy datasets with class noise are created from the aforementioned 25 base datasets (a total of 175 datasets). All these datasets are available on the webpage associated with this paper.

The accuracy estimation of the classifiers in a dataset is obtained by means of 5 runs of a stratified 5-fold cross-validation. Hence, a total of 25 runs per dataset and noise level are

Table 1: Base datasets used in the experimentation.

Dataset	#EXA	#ATT	#CLA	Dataset	#EXA	#ATT	#CLA
balance	625	4	3	newthyroid	215	5	3
banana	5300	2	2	nursery	1296	8	5
cleveland	297	13	5	pima	768	8	2
contraceptive	1473	9	3	segment	2310	19	7
dermatology	358	34	6	shuttle	2175	9	7
ecoli	336	7	8	sonar	208	60	2
german	1000	20	2	splice	319	60	3
glass	214	9	7	vehicle	846	18	4
heart	270	13	2	wdbc	569	30	2
ionosphere	351	33	2	wine	178	13	3
iris	150	4	3	yeast	1484	8	10
lymph	148	18	4	zoo	101	16	7
monk-2	432	6	2				

averaged out. The noisy instances will be present only in the training partitions, as any instance in the test partitions with induced noise is replaced by its original version. It is necessary to note that we ensure the the corrupted instances shared by two or more training partitions are the same, but the test partitions used to compute the performance measures are not affected.

In each run, the training partitions are preprocessed using our proposal and also the base filters, resulting in a preprocessed training partition for each filter. Then three classification algorithms (k-NN, SVM and C4.5) are applied to build a model using every training partition, and to obtain the test performance over the associated test partition.

4.2. Noise filtering methods

The proposal described in this paper allows us to include in it any standard noise filter. For the sake of generality, we have chosen to test the behavior using three different noise filtering techniques that are well-known representatives of the field [7]: RNG [26], CVCF [30] and IPF [16].

The parameter setup used for their execution is presented in Table 2. Even though almost all of the parameters are the default examples recommended by the authors of such filters, we have chosen to differentiate the behavior of the filters by choosing a different voting scheme for CVCF and IPF. On the one hand, CVCF is used with a consensus scheme (making it more restrictive) and IPF is used with the majority scheme (so that it detects more noisy examples). We are interested in having filters that behave differently against the noise. This differentiation is achieved not only by having different filters but also by different voting schemes.

Table 2: Parameter specification for the noise filters.

Algorithm	Ref.	Parameters
RNG	[26]	Distance function: Euclidean
CVCF	[30]	Voting scheme: consensus, Γ: 4
IPF	[16]	Voting scheme: majority, Γ: 5, K_{IPF}: 3, P_{IPF}: 1%

4.3. Methodology of analysis

In order to check how well our proposal behaves dealing with class noise, the analysis performed in the next section is divided into five different parts:

1. To study the advantages in classification performance of preprocessing with our approach versus no-preprocessing (Section 5.1).
2. To study of the behavior of the class noise corrector from the point of view of the data: analysis of the examples corrected and removed (Section 5.2).
3. To compare CNC against each one of the noise filters that compose it (Section 5.3).
4. To study the advantage provided by using an ensemble of filters within CNC against consider CNC only with one filtering method (Section 5.4).
5. To study how the usage of the ENN filter affects the behavior of our approach (Section 5.5).

Regarding to the first point, the performance estimation of each classifier (k-NN, SVM and C4.5) on each dataset is obtained, averaging its test accuracy results. In order to properly analyze these performance results, Wilcoxon's Signed Rank Statistical Test is used [6]. The usage of non-parametrical statistical tests is recommended in these types of comparisons since the conditions to apply parametrical tests are not usually fulfilled [6], [12]. For each noise level, the results of our proposal and the classifier without preprocessing are compared by means of Wilcoxon's Test and the p-values associated with these comparisons are obtained.

The second point to study includes an analysis of which examples are corrected and which are removed from the dataset, and which belong to those corrupted by the noise introduction scheme and which to the non-corrupted set. Consider $D_T = D_N \cup D_O$, being D_N the set of examples whose class labels have been corrupted by the noise scheme and D_O the set of original non-corrupted examples. On the other hand, our proposal acts (modifying the class label or removing the example) over a set of examples $M = \{M_C \cup M_E\} \subseteq D_T$, with M_C being the set of the examples corrected by our method and M_E the set of examples eliminated. We also define

the set $M_{CC} \subseteq M_C$ as the set of examples which are corrected and their label is equal to that of the original set without induced corruptions. In order to check the behavior of the proposed method, we compute 5 different metrics related with its capabilities of the correction/elimination (see Table 3). Moreover, we compare such measures over all the datasets among the different noise levels using the Friedman Aligned Test [12]. In this way, for each metric we will be able to establish an order between the different noise levels, checking in which noise level the measure is higher or lower.

Table 3: Measures computed.

Metric	Description	Expression
c^{D_N}	Corrections among the corrupted examples	$100\frac{\lvert D_N \cap M_C \rvert}{\lvert D_N \rvert}$
cc^{D_N}	Correct corrections among the corrected examples	$100\frac{\lvert D_N \cap M_{CC} \rvert}{\lvert D_N \cap M_C \rvert}$
e^{D_N}	Eliminations among the corrupted examples	$100\frac{\lvert D_N \cap M_E \rvert}{\lvert D_N \rvert}$
c^{D_O}	Corrections among the non-corrupted examples	$100\frac{\lvert D_O \cap M_C \rvert}{\lvert D_O \rvert}$
e^{D_O}	Eliminations among the non-corrupted examples	$100\frac{\lvert D_O \cap M_E \rvert}{\lvert D_O \rvert}$

The third point analyzes how CNC behaves toward each one of the filters that compose it (RNG, CVCF and IPF). In these comparisons we take into consideration the test accuracy of CNC and the three aforementioned filters, and Wilcoxon's test p-values associated to each one of the comparisons.

The rest of the points (fourth and fifth) aim to deepen the question of how the different characteristics of CNC affect the results obtained. In concrete, what is the advantage of using an ensemble of filters within CNC with respect to considering a single filter within CNC (fourth point) and the study of the usage of ENN in CNC (fifth point). In the latter, we study the classical accuracy used in the previous analysis, in addition to another evaluation traditionally employed in imbalanced classification: the *Area Under the ROC Curve* (AUC) measure [2]. The use of such metric employed in imbalanced classification provides an idea on how the removal of examples performed by each noise filter is affecting the examples of each class, since noise filters tend to eliminate the examples of classes with fewer instances, a fact that is not desirable.

5. Analysis of results

This section presents the analysis of the results. Section 5.1 is devoted to the analysis of the performance results of our proposal versus not preprocessing. Section 5.2 describes the percentages of examples corrected and removed by our proposal. Then, Section 5.3 compares

CNC against each one of the noise filters that compose it. Section 5.4 shows the advantage provided by using an ensemble of filters within CNC against considering CNC only with one filtering method. Finally, Section 5.5 shows how the ENN filter affects the behavior of our approach.

5.1. Performance results of CNC versus not preporcessing

The performance results of NN, SVM and C4.5 using our approach (*CNC* rows) and without preprocessing (*None* rows) are presented in Table 4. For the sake of brevity, only average results are presented, even though the complete results for each dataset are accessible in the webpage of this paper. Along with these average results, this table also shows the number of datasets (out of 25) in which the CNC method is the best (*best* rows) and Wilcoxon's test p-value associated with the comparison between CNC and *None* (*p-value* rows). The best performance result for each noise level and those p-values showing significant differences in the comparisons (considering a significance level $\alpha = 0.1$) are highlighted in bold.

Table 4: Test accuracy results without preprocessing (*None*) and with the CNC preprocessing. For each noise level, the best accuracy results are noted in bold. *p-value* rows show the p-values obtained by Wilcoxon's Signed Ranks Test when comparing CNC vs *None* (those cases where *None* obtains more ranks in the comparison are indicated with an asterisk *). *best* rows reveal the number of datasets (out of 25) in which the CNC method is the best.

NN	0%	5%	10%	15%	20%	25%	30%
CNC	**80.09**	**79.83**	**79.69**	**79.28**	**79.08**	**78.58**	**78.27**
None	78.89	76.35	74.18	71.82	69.58	67.30	64.47
p-value	**6.67E-02**	**1.01E-05**	**1.19E-07**	**1.19E-07**	**1.19E-07**	**5.96E-08**	**5.96E-08**
best	15	22	24	24	24	25	25
SVM	0%	5%	10%	15%	20%	25%	30%
CNC	**81.94**	**81.77**	**81.52**	**81.24**	**80.78**	**80.48**	**79.84**
None	81.24	78.76	77.20	76.01	74.32	72.65	71.33
p-value	3.17E-01	**2.03E-03**	**4.89E-04**	**1.62E-04**	**1.83E-05**	**1.51E-05**	**1.23E-05**
best	14	21	19	22	21	22	21
C4.5	0%	5%	10%	15%	20%	25%	30%
CNC	81.03	**80.79**	**80.69**	**79.97**	**79.85**	**79.64**	**79.08**
None	**81.10**	80.71	80.30	78.96	78.07	77.14	76.05
p-value	*5.68E-01	6.19E-01	1.82E-01	**6.73E-03**	**2.50E-04**	**5.39E-05**	**1.40E-04**
best	12	16	18	21	22	23	21

From this table, several remarks can be made:

1. **Performance results of NN**. The average results show that preprocessing by means of our proposal is better than not preprocessing at every noise level. The differences increase along with the noise level. Comparing the limit cases (datasets without noise and with a 30% of class noise), the average results only vary 1.82% with CNC. Otherwise, in the case of no-preprocessing, the difference is more accentuated at each noise level, reaching the maximum difference when the noise level is the highest (30%), that is, 14.42%. This fact shows the importance of CNC in these cases, helping to the maintenance of the initial performance without induced noise when class noise is considered.

   The number of datasets in which each alternative (preprocessing or not) is better only shows a slight advantage in favor of CNC when no induced noise is considered (15 versus 10). For the rest of the noise levels, CNC is better than no preprocessing for almost all the datasets (being the number of datasets in which CNC is better always higher than 22 and, in some cases, the improvement is indeed for all the datasets, such as in the cases of 25% and 30% of noise level). The Wilcoxon Test p-values show that significant differences are always found in favor of CNC due to the obtention of low p-values, showing again the validity of our proposal.
2. **Performance results of SVM**. These results are similar to those obtained with the NN classifier. Thus, average results are always better for the preprocess than those with no preprocess, and the number of datasets with best results is also in favor of our proposal. The p-values also show statistical differences for all the noise levels, with the exception of 0%, in which no statistical differences are found in the comparison of CNC with respect to the absence of preprocessing.
3. **Performance results of C4.5**. In this case, the advantage of CNC is observed from 15% of class noise onwards. Even though at lower noise levels (5% and 10%) CNC is better in terms of accuracy and number of datasets with the best results. From this noise level (15%), the average, the number of datasets with best results and the Wilcoxon Test p-values (which show statistical differences), show a clear advantage of CNC. For the low noise levels both methods show a similar behavior and no statistical differences are found.

These results reveal that our proposal is valid for methods that are sensitive to noise (such as NN or SVM), and that, for robust methods (C4.5), it only implies an improvement if the noise level is relatively high (from 15% onwards).

Even though no significant differences have been observed between the two alternatives (CNC and *None*) without induced noise for SVM and for the lowest noise levels for C4.5 (below 10%), this fact can result due to the application of CNC not being noticeable enough in some

cases because of the very low quantities of class noise (or non-existent). However, when the noise level is higher (from 5% onwards for SVM and from 15% onwards for C4.5), the behavior or the proposal improves notably contrary to not preprocessing.

5.2. Analysis of the examples corrected and removed by CNC

Table 5 shows the results for the five metrics described in Table 3. For sake of simplicity, only the average results for each metric at each noise level are shown. The complete results of each metric for each dataset at each noise level can be found in the webpage associated with this paper. Furthermore, the rankings obtained by the Friedman Aligned procedure are also shown in order to help to sort each measure by the noise levels in an incremental way.

Table 5: Average results and rankings obtained with the Friedman Aligned for each metric at each noise level.

	Measure					Rankings				
Noise	c^{D_N}	cc^{D_N}	e^{D_N}	c^{D_O}	e^{D_O}	c^{D_N}	cc^{D_N}	e^{D_N}	c^{D_O}	e^{D_O}
0%	-	-	-	5.53	6.89	-	-	-	68.24	139.80
5%	71.64	92.28	16.88	5.47	7.23	21.20	77.12	126.16	72.16	129.00
10%	66.77	93.30	19.42	5.41	7.42	50.72	63.42	101.52	78.88	120.24
15%	65.80	93.11	20.49	5.33	7.88	59.28	67.98	90.72	85.84	90.20
20%	63.15	93.68	22.79	5.22	8.33	89.92	67.20	61.88	100.64	62.96
25%	61.47	92.90	23.55	5.20	8.67	107.92	82.48	48.00	100.72	40.76
30%	59.50	92.37	25.80	5.13	9.27	123.96	94.80	24.72	109.52	33.04

The analysis of these results leads to following observations:

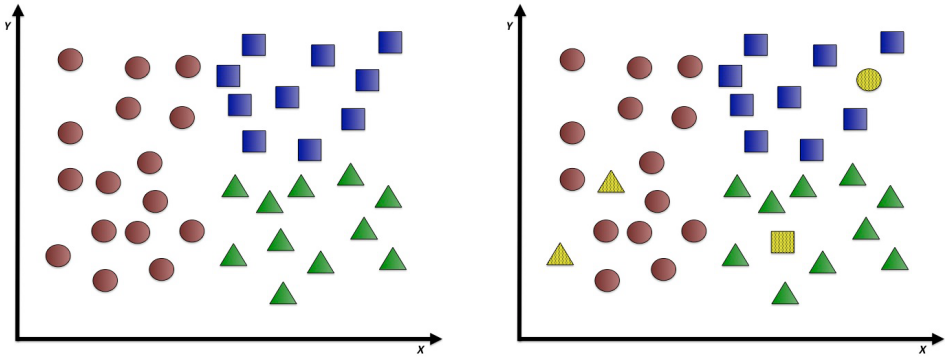
1. **Modifications of the proposal affecting the corrupted examples (c^{D_N}, cc^{D_N} and e^{D_N}).** The average results for all the datasets show that the corrections (c^{D_N}) diminish, whereas the eliminations (e^{D_N}) increment along with the noise level. Furthermore, the corrections c^{D_N} represent a high percentage (always higher than a 59%) and the eliminations e^{D_N} are low values (lower than a 26%). On the other hand, the correct corrections percentage among the corrected examples is also very high (higher than a 92%).
2. **Modifications of the proposal affecting the original examples (c^{D_O} and e^{D_O}).** The corrections (c^{D_O}) diminish, whereas the eliminations (e^{D_O}) increment along with the noise level. Moreover, the corrections are always lower than a 5% (and they therefore imply a number relatively low) and the eliminations are also low (lower than a 10%).

It is important to note that, the rankings obtained with the Friedman Aligned Test to sort the distribution of each metric among the different noise levels show similar results to those of the average results. Even though for some noise levels the positions may be changed, the tendency described in the two previous points is followed.

To conclude, the results obtained in this section show that, in general, the percentage of examples corrected diminish along with the increase of the noise level. Nevertheless, the eliminations increase along with the noise level. A possible explanation of these results could be simply based on the increment of density of corrupted examples in a concrete area of the domain as the noise increases, that is, the creation of clusters of corrupted examples of either the same or different class. These clusters are more likely to be created when the noise level is higher and they can affect both the other corrupted or the original examples, implying a variation with the noise level of each one of the five metrics analyzed above. Thus, clusters with a considerable quantity of corrupted examples may affect in a higher degree other examples around them than isolated examples, which can be easily identified and treated. Furthermore, if the corrupted examples are grouped (with the same or different class labels), the correction is also more difficult to perform, and if it is done, it is probably less effective.

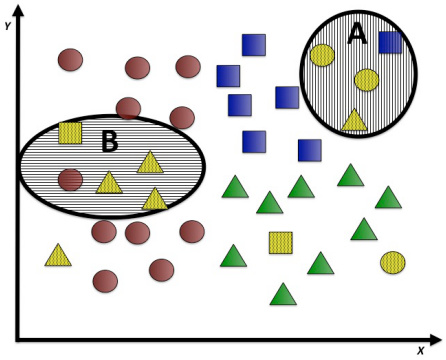
In order to better explain this idea, we have built the bi-dimensional synthetic datasets illustrated in Figure 3. Figure 3(a) represent the original dataset without induced noise, whereas Figures 3(b) and 3(c) represent the same dataset with a respectively low and high quantity of examples with class noise. In these datasets we have examples belonging to three different classes: $\bigcirc$, $\square$ and $\triangle$. Analyzing these figures, we can observe the following:

1. In the case where a low class noise level is considered (Figure 3(b)), corrupted examples are likely to be isolated. Therefore, they will be recognized as noisy more easily by this approach and their correct class can be also easily identified because they are probably rounded by examples of only one class.
2. In the case where a high class noise level is considered, it is more likely that, for a concrete area of the domain, there is a notable concentration of corrupted examples (see areas A and B in Figure 3(c)). The following situations could be produced:
    - In the area B, where some three examples of class $\triangle$ have been introduced, it is likely that these examples be considered correct by our approach and their class label not be modified. On the other hand, in the area A, the new corrupted example with class $\triangle$ could belong to the class $\bigcirc$ and be, therefore, wrongly corrected. These examples could explain the reduction of the metric representing the corrections of the proposal c^{D_N} when the noise level increases.



(a) Original non-corrupted data.

(b) Low level of class noise.



(c) High level of class noise.

Figure 3: A synthetic dataset with (a) non-corrupted examples, (b) a low quantity of examples with class noise and (c) a high quantity of examples with class noise and two clusters of corrupted data (A and B).

- When the noise level increases, the corrupted examples affect more their original non-corrupted surrounding examples and therefore it is more likely that these original examples will be identified as noisy by the proposal (it seems that this original example is noisy because is near several noisy examples). In some cases, this fact may imply a correction by CNC among the original examples. This is represented in the Figure 3(c), in the area A. The correct and non-corrupted example with class □ is likely to be affected by a correction of our proposal since it is surrounded by several corrupted examples of the class ◯, and then a consensus among all the noise filters of CNC may be produced.
- In particular, when several corrupted examples are grouped in a cluster and they possess different class labels (or they are placed close to the boundaries of the classes)

and affect other corrupted or original examples, it is likely that this example be removed by our approach, since it is more difficult to find a consensus among the decisions of the filters to correct its class. This is represented in Figure 3(c), in area B. The correct and non-corrupted examples with class □ and ◯ are likely to be affected by a elimination of our proposal since they are relatively isolated and surrounded by several examples of different classes.

5.3. Comparison of CNC versus RNG, CVCF and IPF

Table 6 shows the accuracy results and the Wilcoxon Test p-values of CNC compared to those of each of the filters that compose it: RNG, CVCF and IPF. The p-values highlighted in bold show significant differences in favor of CNC (with $\alpha = 0.1$), while the p-values with an asterisk '*' indicate that the single filter has obtained more ranks than CNC in the Wilcoxon Test.

Table 6: Test accuracy results and the Wilcoxon Test p-values of the comparison of CNC versus RNG, CVCF and IPF filters. The cases where the filter obtains more ranks are indicated with an asterisk *, and those showing significant differences in favor of CVC are in bold.

Method	Accuracy							p-values						
NN	0%	5%	10%	15%	20%	25%	30%	0%	5%	10%	15%	20%	25%	30%
CNC	80.09	79.83	**79.69**	**79.28**	**79.08**	**78.58**	**78.27**	1.00E+00	*7.98E-01	3.31E-01	4.76E-01	**9.57E-02**	**3.78E-03**	**2.75E-02**
RNG	**80.34**	**80.07**	79.59	79.26	78.74	77.71	77.56							
CNC	**80.09**	**79.83**	**79.69**	**79.28**	**79.08**	**78.58**	**78.27**	1.73E-01	**1.60E-02**	**1.03E-04**	**1.01E-05**	**8.34E-07**	**5.96E-07**	**5.96E-08**
CVCF	79.54	78.69	77.78	76.38	75.23	73.59	71.62							
CNC	80.09	79.83	79.69	79.28	79.08	**78.58**	**78.27**	*1.99E-01	*4.59E-01	*6.68E-01	*3.82E-01	*9.68E-01	1.65E-01	1.82E-01
IPF	**80.45**	**80.08**	**79.85**	**79.62**	**79.17**	78.34	78.03							
SVM	0%	5%	10%	15%	20%	25%	30%	0%	5%	10%	15%	20%	25%	30%
CNC	**81.94**	**81.77**	**81.52**	**81.24**	**80.78**	**80.48**	**79.84**	1.92E-01	2.09E-01	**2.29E-02**	**2.75E-02**	**1.15E-02**	**5.56E-04**	**7.42E-04**
RNG	81.40	81.20	80.80	80.36	79.65	79.04	78.40							
CNC	**81.94**	**81.77**	**81.52**	**81.24**	**80.78**	**80.48**	**79.84**	2.64E-01	**5.51E-02**	**2.19E-02**	**3.09E-03**	**1.62E-04**	**1.01E-05**	**3.19E-05**
CVCF	81.64	80.88	80.24	78.89	77.64	76.52	74.93							
CNC	81.94	81.77	**81.52**	**81.24**	**80.78**	**80.48**	**79.84**	*7.16E-01	8.40E-01	9.89E-01	6.77E-01	2.53E-01	2.11E-01	3.00E-01
IPF	**82.14**	**81.80**	**81.52**	81.16	80.47	80.29	79.70							
C4.5	0%	5%	10%	15%	20%	25%	30%	0%	5%	10%	15%	20%	25%	30%
CNC	**81.03**	**80.79**	**80.69**	**79.97**	**79.85**	**79.64**	**79.08**	4.20E-01	2.53E-01	**7.10E-02**	3.97E-01	1.07E-01	**8.82E-03**	**6.26E-02**
RNG	80.77	80.41	80.29	79.78	79.35	78.99	78.63							
CNC	81.03	**80.79**	**80.69**	**79.97**	**79.85**	**79.64**	**79.08**	*4.75E-01	*9.68E-01	6.67E-01	**9.57E-02**	**9.12E-04**	**6.37E-05**	**4.30E-04**
CVCF	**81.18**	**80.79**	80.48	79.53	78.21	77.37	76.24							
CNC	81.03	80.79	80.69	79.97	79.85	79.64	**79.08**	*6.13E-03	*7.37E-03	*4.83E-02	*1.14E-01	*1.07E-01	*9.46E-01	9.54E-01
IPF	**81.69**	**81.57**	**81.29**	**80.64**	**80.33**	**79.68**	78.91							

From this table, it is possible to deduce that CNC is always better in terms of accuracy than

the CVCF filter (except for C4.5 with no noise). Furthermore, CNC obtains more ranks than CVCF with the noise sensitive learners (NN and SVM) at all the noise levels, and from 10% onwards in the case of the C4.5 robust learner. The p-values also show that these differences are significant in most of the cases. In the case of NN and SVM these differences are significant from 5% onwards, whereas for C4.5 the differences are significant from 15% onwards.

CNC outperforms RNG for SVM and for C4.5 at all the noise levels and NN from 10% onwards. In these cases, statistical differences with the Wilcoxon Test are only observed from 20% with NN, 10% with SVM and the noise levels of 10%, 25% and 30% with C4.5.

Finally, considering the comparison with IPF, the accuracy results show that CNC is only better from 25% onwards in the case of NN, 10% with SVM and only in the last noise level with C4.5. However, the Wilcoxon Test shows no statistical differences in favor of any of the methods of comparison in any of these cases.

To conclude, we can claim that CNC is able to overcome both the CVCF filter at almost all the noise levels and the RNG filter when the noise level is high enough (with p-values below 0.1). However, no statistical differences have been found in the comparison of CNC and IPF with any of the classifiers studied (NN, SVM and C4.5).

It is important to note that IPF and CVCF stand out (since higher performances and higher p-values are obtained in the comparisons) when C4.5 is used than when any of the other two classifiers (NN or SVM) is used. This fact may be due to the filtering performed by CVCF and IPF using C.45 as internal classifier and it can imply an advantage to these filters when C4.5 is later used as a classifier.

Note that performing a correction is a more complex task than removing noise. The filtering of noisy examples only requires their identification, whereas the correction requires this stage as well as an additional phase of correction, in which one of all the possible classes of the problem must be chosen (obviously, the more classes a problem has, the more complex it is to chose the correct class of a noisy example). This fact can explain the results obtained in the less favorable case (versus IPF), where no statistical differences are found that favor on any of the methods.

5.4. Using ensembles versus single filters within CNC

Table 7 shows the accuracy results of the comparison of CNC built with the ensemble of the three filters (RNG, CVCF and IPF) discussed in Section 2.2 versus CNC considering each of the filters separately. Also shown are the p-values obtained with the Wilcoxon Test.

This table shows that the use of an ensemble of filters is, in general, statistically beneficial compared to using each separate filter. The single exception is the case of using CVCF within CNC with the methods SVM and C4.5, in which no statistical differences are found at almost

all the noise levels. Here only statistical differences are found at 30% of noise level for SVM in favor of CNC and at the noise levels 0% and 15% for C4.5 in favor of CNC_{CVCF}.

These results reveal that, in most of the cases studied (including the exceptions we mentioned), using an ensemble of filters within CNC, rather than using a single filter within CNC, may improve the capability to detect noisy examples.

Table 7: Test accuracy results and the Wilcoxon Test p-values comparing CNC built with the ensemble of the three filters RNG, CVCF and IPF (*CNC* rows) versus CNC considering each one of the filters separately (CNC_{RNG}, CNC_{CVCF} and CNC_{IPF} rows). The cases where CNC with a single filter obtains more ranks in the comparison are indicated with an asterisk *, and those with significant differences in favor of CVC are in bold.

Method	Accuracy							p-values						
KNN	0%	5%	10%	15%	20%	25%	30%	0%	5%	10%	15%	20%	25%	30%
CNC	80.09	79.83	79.69	79.28	79.08	78.58	78.27	**5.96E-07**	**1.13E-06**	**1.79E-07**	**1.79E-07**	**1.79E-07**	**1.79E-07**	**1.79E-07**
CNC_{RNG}	77.46	77.04	76.60	75.78	74.75	73.75	72.59							
CNC	80.09	79.83	79.69	79.28	79.08	78.58	78.27	**9.57E-02**	**2.03E-02**	**6.73E-03**	**5.56E-04**	**2.03E-03**	**5.33E-05**	**3.29E-04**
CNC_{CVCF}	79.92	79.49	79.36	78.71	78.30	77.76	77.30							
CNC	80.09	79.83	79.69	79.28	79.08	78.58	78.27	**6.56E-06**	**3.19E-05**	**3.28E-06**	**3.28E-06**	**1.49E-06**	**1.97E-06**	**5.25E-06**
CNC_{IPF}	78.10	77.76	77.34	76.81	76.36	76.13	75.21							
SVM	0%	5%	10%	15%	20%	25%	30%	0%	5%	10%	15%	20%	25%	30%
CNC	81.94	81.77	81.52	81.24	80.78	80.48	79.84	**8.08E-04**	**3.76E-04**	**1.62E-04**	**2.03E-03**	**3.76E-04**	**4.54E-05**	**8.17E-06**
CNC_{RNG}	80.09	79.47	79.01	78.19	76.70	76.18	74.89							
CNC	81.94	81.77	81.52	81.24	80.78	80.48	79.84	*9.89E-01	3.00E-01	5.81E-01	3.67E-01	2.11E-01	2.11E-01	**7.10E-02**
CNC_{CVCF}	82.15	81.80	81.66	81.17	80.62	80.21	79.42							
CNC	81.94	81.77	81.52	81.24	80.78	80.48	79.84	**3.18E-02**	**1.73E-02**	**5.07E-03**	**1.30E-03**	**1.63E-03**	**3.19E-05**	**1.88E-04**
CNC_{IPF}	80.35	80.00	79.68	79.31	78.94	78.10	77.44							
C4.5	0%	5%	10%	15%	20%	25%	30%	0%	5%	10%	15%	20%	25%	30%
CNC	81.03	80.79	80.69	79.97	79.85	79.64	79.08	**1.13E-06**	**8.17E-06**	**5.96E-08**	**5.96E-07**	**1.19E-07**	**5.96E-08**	**2.98E-07**
CNC_{RNG}	78.25	77.97	77.30	76.71	75.78	75.09	74.33							
CNC	81.03	80.79	80.69	79.97	79.85	79.64	79.08	*4.51E-02	*3.26E-01	*3.00E-01	*7.55E-02	*3.67E-01	*4.12E-01	*4.76E-01
CNC_{CVCF}	81.30	81.00	80.88	80.29	80.10	79.70	79.23							
CNC	81.03	80.79	80.69	79.97	79.85	79.64	79.08	**3.81E-05**	**3.28E-06**	**4.42E-05**	**6.37E-05**	**2.56E-06**	**1.01E-05**	**3.19E-05**
CNC_{IPF}	80.16	79.79	79.48	78.61	78.51	78.10	77.35							

5.5. Analysis on the use of ENN within CNC

Table 8 shows the p-values associated with the comparison of CNC versus each one of the methods of each row (*None*, RNG, CVCF and IPF), with and without the use of ENN within CNC. We have studied the results of two different evaluation measures: the classical accuracy (ACC) and the *Area Under the ROC Curve* (AUC) measure [2], which is usually employed with

imbalanced classification problems where the number of examples in each class is not equal. Due to the large number of results, only the p-values and the method obtaining more ranks in the Wilcoxon Test are shown, since in this case we are not interested in the concrete value of the performance of each individual case, but rather whether there is a improvement or decline in the comparisons. The complete results of accuracy and AUC are available at this paper's website.

Table 8: The Wilcoxon Test p-values associated with the comparison of CNC (with and without ENN) versus *None*, RNG, CVCF and IPF, considering the accuracy and the AUC.

Noise	0%	5%	10%	15%	20%	25%	30%	0%	5%	10%	15%	20%	25%	30%
ACC	**Considering ENN**							**Not considering ENN**						
vs	**NN**							**NN**						
None	**6.67E-02**	**1.01E-05**	**1.19E-07**	**1.19E-07**	**1.19E-07**	**5.96E-08**	**5.96E-08**	**5.85E-03**	**1.19E-07**	**5.96E-08**	**1.19E-07**	**5.96E-08**	**5.96E-08**	**5.96E-08**
RNG	1.00E+00	*7.98E-01	3.31E-01	4.76E-01	**9.57E-02**	**3.78E-03**	**2.75E-02**	9.89E-01	8.40E-01	2.41E-01	5.81E-01	6.57E-01	2.11E-01	4.43E-01
CVCF	1.73E-01	**1.60E-02**	**1.03E-04**	**1.01E-05**	**8.34E-07**	**5.96E-07**	**5.96E-08**	**2.94E-03**	**1.51E-05**	**5.96E-08**	**5.96E-08**	**5.96E-08**	**5.96E-08**	**5.96E-08**
IPF	*1.99E-01	*4.59E-01	*6.68E-01	*3.82E-01	*9.68E-01	1.65E-01	1.82E-01	7.57E-01	9.43E-01	*9.54E-01	*3.00E-01	*8.08E-04	*7.87E-02	*7.19E-03
vs	**SVM**							**SVM**						
None	2.76E-01	**1.03E-03**	**1.62E-04**	**4.54E-05**	**1.51E-05**	**5.25E-06**	**8.17E-06**	1.92E-01	**1.83E-05**	**5.25E-06**	**2.66E-05**	**5.25E-06**	**4.17E-06**	**6.56E-06**
RNG	1.92E-01	2.09E-01	**2.29E-02**	**2.75E-02**	**1.15E-02**	**5.56E-04**	**7.42E-04**	**5.16E-02**	1.41E-01	**2.55E-02**	1.14E-01	**5.88E-02**	**2.75E-02**	**8.82E-03**
CVCF	2.64E-01	**5.51E-02**	**2.19E-02**	**3.09E-03**	**1.62E-04**	**1.01E-05**	**3.19E-05**	**5.69E-02**	**2.78E-03**	**4.60E-03**	**1.03E-04**	**1.51E-05**	**2.21E-05**	**1.51E-05**
IPF	*7.16E-01	8.40E-01	9.89E-01	6.77E-01	2.53E-01	2.11E-01	3.00E-01	2.16E-01	1.07E-01	4.75E-01	3.31E-01	7.10E-01	*5.68E-01	*6.96E-01
vs	**C4.5**							**C4.5**						
None	*7.53E-01	6.07E-01	2.21E-01	**2.96E-02**	**3.76E-04**	**5.39E-05**	**1.40E-04**	**5.69E-02**	**9.63E-04**	**9.08E-05**	**2.56E-06**	**1.79E-07**	**5.96E-07**	**1.49E-06**
RNG	4.20E-01	2.53E-01	**7.10E-02**	3.97E-01	1.07E-01	**8.82E-03**	**6.26E-02**	**2.50E-04**	**2.44E-05**	**7.19E-03**	**3.19E-05**	**3.29E-04**	**4.60E-03**	2.76E-01
CVCF	*4.75E-01	*9.68E-01	6.67E-01	**9.57E-02**	**9.12E-04**	**6.37E-05**	**4.30E-04**	1.52E-01	**2.05E-04**	**3.42E-03**	**2.56E-06**	**3.28E-06**	**1.13E-06**	**3.28E-06**
IPF	*6.13E-03	*7.37E-03	*4.83E-02	*1.14E-01	*1.07E-01	*9.46E-01	9.54E-01	8.98E-01	6.58E-01	1.52E-01	**5.16E-02**	1.28E-01	1.69E-01	5.11E-01
AUC	**Considering ENN**							**Not considering ENN**						
vs	**NN**							**NN**						
None	*2.42E-01	6.57E-01	**3.93E-02**	**2.36E-02**	**4.89E-04**	**1.88E-04**	**1.23E-05**	7.67E-01	**2.36E-02**	**3.29E-04**	**7.50E-05**	**2.98E-07**	**1.79E-07**	**5.96E-08**
RNG	*1.20E-01	*3.93E-02	*5.27E-01	*2.31E-01	*4.76E-01	4.59E-01	7.78E-01	1.56E-01	1.14E-01	1.43E-01	1.73E-01	2.11E-01	**4.51E-02**	**8.02E-02**
CVCF	*2.31E-01	*3.53E-01	8.19E-01	4.12E-01	1.73E-01	**9.57E-02**	**3.18E-02**	*9.79E-01	2.11E-01	**6.73E-03**	**9.12E-04**	**4.30E-04**	**1.20E-04**	**2.66E-05**
IPF	*4.91E-02	*1.36E-02	*8.02E-02	*1.07E-01	*1.07E-01	*7.98E-01	*5.63E-01	2.64E-01	3.91E-01	7.26E-01	*6.57E-01	*1.14E-01	*7.06E-01	*3.31E-01
vs	**SVM**							**SVM**						
None	*8.02E-02	*6.96E-01	4.76E-01	1.20E-01	**1.47E-02**	**3.78E-03**	**1.87E-02**	*3.46E-01	1.34E-01	**1.36E-02**	**1.82E-03**	**1.88E-04**	**3.76E-04**	**1.30E-03**
RNG	*5.10E-01	*5.81E-01	7.16E-01	3.97E-01	3.39E-01	2.42E-01	2.11E-01	**7.55E-02**	**6.67E-02**	**6.73E-03**	**2.96E-02**	**2.55E-02**	**4.22E-02**	**1.63E-03**
CVCF	*8.51E-02	*2.88E-01	*3.67E-01	9.46E-01	5.27E-01	**8.02E-02**	**7.10E-02**	*3.33E-01	4.76E-01	3.97E-01	1.65E-01	**1.47E-02**	**1.05E-02**	**2.03E-03**
IPF	*8.51E-02	*5.88E-02	*1.41E-01	*3.26E-01	8.40E-01	8.61E-01	*8.40E-01	**6.91E-02**	**6.52E-03**	2.53E-01	**4.51E-02**	**9.57E-02**	**9.57E-02**	**9.57E-02**
vs	**C4.5**							**C4.5**						
None	*7.42E-04	*1.36E-02	*8.51E-02	*1.56E-01	*7.37E-01	5.63E-01	5.45E-01	9.32E-01	7.53E-01	**7.38E-02**	**3.78E-03**	**2.51E-03**	**5.56E-04**	**1.87E-02**
RNG	8.64E-01	3.97E-01	2.88E-01	*9.46E-01	9.25E-01	1.48E-01	4.27E-01	**1.51E-05**	**2.38E-07**	**4.54E-05**	**1.01E-05**	**1.97E-06**	**8.08E-04**	**2.19E-02**
CVCF	*6.50E-04	*8.08E-04	*2.36E-02	*4.22E-02	*3.26E-01	*8.61E-01	5.81E-01	1.00E+00	7.32E-01	3.39E-01	**4.51E-02**	**2.55E-02**	**5.51E-02**	**1.60E-02**
IPF	*4.60E-03	*3.29E-04	*1.30E-03	*7.15E-04	*6.50E-04	*8.02E-02	*2.11E-01	5.77E-01	8.93E-01	2.53E-01	**5.16E-02**	4.75E-01	9.25E-01	5.11E-01

The accuracy results (ACC) shown in Table 8 considering the use of ENN are the same as those discussed in the previous sections. They reveal that the CNC method generally works well compared to *None* and the other filters (with the exception of IPF, and especially with C4.5 at the lowest noise levels). The results of CNC improve for C4.5 when ENN is not applied, are comparable for SVM and are decreased for NN (particularly versus IPF).

This fact may be because ENN is based on the k-NN decision rule to remove the noisy examples, which can be particularly beneficial if this classifier is subsequently considered. By contrast, using a robust algorithm such as C4.5, the cleansing performed by ENN is not always necessary (C4.5 uses a post-pruning process to build the model).

However, regarding to the AUC metric, the affects of the application of ENN can be clearly appreciated. Considering the use of ENN, CNC works apparently badly, particularly in the case of C4.5, which is a robust method that maybe does not require a cleaning step and penalizes our method. However, it is clear that not considering ENN is positive for CNC using all classifiers, since much better results are obtained when it is compared to *None* and the rest of the filters.

This may mean that the cleaning step of ENN within CNC is possibly detrimental in cases where the classes are not balanced (ENN or any filtering method, in general). Not applying ENN is such cases implies that the AUC metric obtains good results for CNC. Therefore, it can be proposed that the application of the ENN step is optional, depending on the specific characteristics of the problem confronted.

6. Conclusions

This paper has focused on the presence of class noise examples which is an important research issue while learning classifiers, particularly noise-sensitive ones. A class noise correction and filtering method has been proposed based on a ensemble of three filters (RNG, CVCF and IPF) whose main objective is the correction of class noise in the data, even though it also removes some examples where the correction is not reliable. The suitability of this approach in this scenario has been analyzed using a large number of noisy datasets created considering different noise levels. The k-NN, SVM and C4.5 classifiers have been evaluated on these datasets, with and without the use of the preprocessing with our approach.

The results obtained have shown that the use of our preprocessing improves the performance results without preprocessing when the data are corrupted by noise (for all the noise level for the noise-sensitive methods k-NN and SVM and for intermediate-high noise levels for the robust method C4.5). When no noise is introduced into the data, in general, no statistical differences have been found between CNC and not preprocessing. All these observations have been supported by applying appropriate statistical tests.

We also have analyzed the validity of our proposal in terms of the number of examples corrected and removed, showing the efficacy of the correction process and low rates of removed examples, which is desirable in many domains (for example, in imbalanced domains).

We have also compared CNC versus each one of the filters that compose it (RNG, CVCF

and IPF), reaching the conclusion that CNC is able to overcome CVCF at almost all the noise levels, RNG when the noise level is enough high and no statistical differences have been found in the comparison with IPF. Furthermore, we have checked the usefulness of using an ensemble of noise filters within CNC, which has shown with some exceptions, a good behavior against the consideration of CNC with only one noise filter. Finally, we have shown that the elimination of the step of ENN within CNC may improve its AUC results compared to the rest of noise filters. This may be related to the elimination of the examples belonging to minority classes by part of ENN and the rest of the noise filters.

In future research the behavior of other filters in the ensemble could be studied as well as the adequacy of this proposal for other types of noise presented in real-world data.

Acknowledgments

This work was supported by the Projects TIN2011-28488, P10-TIC-06858 and P11-TIC-7765. J. A. Sáez holds an FPU grant from the Spanish Ministry of Education and Science. This paper was funded by the King Abdulaziz University, under grant No. (3-611-1434-HiCi). The authors kindly acknowledge technical and financial support of the KAU.

References

[1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL Data-Mining Software Tool: Data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2-3) (2011) 255–287.

[2] A. P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (7) (1997) 1145–1159.

[3] C. E. Brodley, M. A. Friedl, Identifying Mislabeled Training Data, Journal of Artificial Intelligence Research 11 (1999) 131–167.

[4] W. W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann Publishers, 1995, pp. 115–123.

[5] S. Cuendet, D. Z. Hakkani-Tr, E. Shriberg, Automatic labeling inconsistencies detection and correction for sentence unit segmentation in conversational speech., in: A. Popescu-Belis, S. Renals, H. Bourlard (eds.), MLMI, vol. 4892 of Lecture Notes in Computer Science, Springer, 2007, pp. 144–155.

[6] J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 7 (2006) 1–30.

[7] B. Frenay, M. Verleysen, Classification in the presence of label noise: a survey, IEEE Transactions on Neural Networks and Learning Systems, in press, doi: 10.1109/TNNLS.2013.2292894.

[8] D. Gamberger, R. Boskovic, N. Lavrac, C. Groselj, Experiments With Noise Filtering in a Medical Domain, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, 1999, pp. 143–151.

[9] D. Gamberger, N. Lavrac, S. Dzeroski, Noise elimination in inductive concept learning: A case study in medical diagnosis, in: In Proc. of the 7th International Workshop on Algorithmic Learning Theory, Springer, 1996, pp. 199–212.

[10] D. Gamberger, N. Lavrac, S. Dzeroski, Noise Detection and Elimination in Data Preprocessing: experiments in medical domains, Applied Artificial Intelligence 14 (2000) 205–223.

[11] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (2010) 2044–2064.

[12] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (2010) 2044–2064.

[13] S. García, F. Herrera, An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.

[14] M. A. Hernández, S. J. Stolfo, Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem, Data Mining and Knowledge Discovery 2 (1998) 9–37.

[15] J. V. Hulse, T. M. Khoshgoftaar, Knowledge discovery from imbalanced and noisy data., Data Knowledge & Engineering 68 (12) (2009) 1513–1542.

[16] T. M. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise filtering techniques, Journal of Computer Science and Technology 22 (2007) 387–396.

[17] J. Koplowitz, T. A. Brown, On the relation of performance to editing in nearest neighbor rules., Pattern Recognition 13 (3) (1981) 251–255.

[18] G. J. McLachlan, Discriminant Analysis and Statistical Pattern Recognition, John Wiley and Sons (2004).

[19] A. L. B. Miranda, L. P. F. Garcia, A. C. P. L. F. Carvalho, A. C. Lorena, Use of classification algorithms in noise detection and elimination., in: E. Corchado, X. Wu, E. Oja, . Herrero, B. Baruque (eds.), HAIS, vol. 5572 of Lecture Notes in Computer Science, Springer, 2009, pp. 417–424.

[20] T. Oates, D. Jensen, The effects of training set size on decision tree complexity., in: D. H. Fisher (ed.), ICML, Morgan Kaufmann, 1997, pp. 254–262.

[21] J. R. Quinlan, Induction of Decision Trees, in: Machine Learning, 1986, pp. 81–106.

[22] J. R. Quinlan, C4.5: programs for machine learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.

[23] J. Sáez, J. Luengo, F. Herrera, Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification, Pattern Recognition 46 (1) (2013) 355–364.

[24] J. A. Sáez, M. Galar, J. Luengo, F. Herrera, Tackling the problem of classification with noisy data using multiple classifier systems: Analysis of the performance and robustness., Information Sciences. 247 (2013) 1–20.

[25] J. A. Sáez, M. Galar, J. Luengo, F. Herrera, Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition, Knowledge and Information Systems 38 (1) (2014) 179–206.

[26] J. Sánchez, F. Pla, F. Ferri, Prototype selection for the nearest neighbor rule through proximity graphs, Pattern Recognition Letters 18 (1997) 507–513.

[27] B. Sluban, D. Gamberger, N. Lavra, Ensemble-based noise detection: noise ranking and visual performance evaluation, Data Mining and Knowledge Discovery 28 (2) (2014) 265–303.

[28] C.-M. Teng, Correcting Noisy Data, in: Proceedings of the Sixteenth International Conference on Machine

Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999, pp. 239–248.

[29] V. Vapnik, Statistical Learning Theory, Wiley, New York, USA, 1998.

[30] S. Verbaeten, A. V. Assche, Ensemble methods for noise elimination in classification problems, in: Fourth International Workshop on Multiple Classifier Systems, Springer, 2003, pp. 317–325.

[31] R. Y. Wang, V. C. Storey, C. P. Firth, A Framework for Analysis of Data Quality Research, IEEE Transactions on Knowledge and Data Engineering 7 (4) (1995) 623–640.

[32] D. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems and Man and Cybernetics 2 (3) (1972) 408–421.

[33] X. Wu, Knowledge acquisition from databases, Ablex Publishing Corp., Norwood, NJ, USA, 1996.

[34] X. Wu, X. Zhu, Mining with noise knowledge: Error-aware data mining, IEEE Transactions on Systems, Man, and Cybernetics 38 (2008) 917–932.

[35] Y. Zhang, X. Wu, Integrating induction and deduction for noisy data mining, Information Sciences 180 (14) (2010) 2663–2673.

[36] S. Zhong, T. M. Khoshgoftaar, N. Seliya, Analyzing Software Measurement Data with Clustering Techniques, IEEE Intelligent Systems 19 (2) (2004) 20–27.

[37] X. Zhu, X. Wu, Class Noise vs. Attribute Noise: A Quantitative Study, Artificial Intelligence Review 22 (2004) 177–210.

4. Evaluation measures of the behavior of classifiers with noisy data

The publications associated to this part are:

4.1 Fuzzy Rule Based Classification Systems versus Crisp Robust Learners Trained in Presence of Class Noise's Effects: a Case of Study

Sáez J. A., Luengo J., and Herrera F. (2011) Fuzzy Rule Based Classification Systems versus Crisp Robust Learners Trained in Presence of Class Noise's Effects: a Case of Study. In *11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), Córdoba (Spain)*, pp. 1229–1234

- Status: **Published**.

Fuzzy Rule Based Classification Systems versus Crisp Robust Learners Trained in Presence of Class Noise's Effects: a Case of Study

José A. Sáez, Julián Luengo, Francisco Herrera
Department of Computer Science and Artificial Intelligence
University of Granada, CITIC-UGR
Granada, Spain, 18071
e-mail: {smja, julianlm, herrera}@decsai.ugr.es

***Abstract*—The presence of noise is common in any real-world dataset and may adversely affect the accuracy, construction time and complexity of the classifiers in this context. Traditionally, many algorithms have incorporated mechanisms to deal with noisy problems and reduce noise's effects on performance; they are called robust learners. The C4.5 crisp algorithm is a well-known example of this group of methods. On the other hand, models built by Fuzzy Rule Based Classification Systems are widely recognized for their robustness to imperfect data, but also for their interpretability.**

The aim of this contribution is to analyze the good behavior and robustness of Fuzzy Rule Based Classification Systems when noise is present in the examples' class labels, especially versus robust learners. In order to accomplish this study, a large number of datasets are created by introducing different levels of noise into the class labels in the training sets. We compare a Fuzzy Rule Based Classification System, the *Fuzzy Unordered Rule Induction Algorithm*, with respect to the C4.5 classic robust learner which is considered tolerant to noise. From the results obtained it is possible to observe that Fuzzy Rule Based Classification Systems have a good tolerance, in comparison to the C4.5 algorithm, to class noise.

***Keywords*-Noisy Data; Class Noise; Fuzzy Rule Based Systems; Robust Learners; Classification.**

I. Introduction

Fuzzy Rule Based Classification Systems (FRBCSs) [1], [2] are widely used due to their ability to build a linguistic model interpretable to the users with the possibility of mixing different information such as that proceeding from expert knowledge and information from mathematical models or empirical measures. Among the applications of FRBCSs we can find proposals in a variety of fields, including standard classification [3], [4], detection of intrusions [5] or medical applications [6].

One goal of classification algorithms is to form a generalization from a set of labeled training instances so that classification accuracy for previously unobserved instances is maximized. Hence the accuracy of the model created by any induction-based learning algorithm is determined by the quality of training data upon which this model is built. Data quality is determined by several components [7], among which are the source of that data and the input of the data, inherently subject to error. Thus, real-world datasets rarely lack these types of error and they usually have corruptions that can affect the interpretations, decisions taken and the models created from the data.

Therefore, the maximum achievable accuracy depends not only on the quality of the data, but also on the appropriateness of the chosen learning algorithm for the data. Knowing what kind of classification algorithms are more suitable when working with noisy data is a challenging question.

In this work we will analyze the suitability of FRBCSs, specifically we will focus on the *Fuzzy Unordered Rule Induction Algorithm* (FURIA) [4], when dealing with noise in examples' class labels and we will compare it to the C4.5 crisp algorithm [8] which is considered tolerant to noise and can be translated as a rule set. When training a classifier with problems with noise, the capability of this classifier to avoid the overfitting of the new characteristics introduced by the noisy examples is a key question [9]. Due to the inherent characteristics of fuzzy rules and the inference process of the FRBCSs that differ from those of the classic crisp systems, models obtained by FRBCSs are expected to absorb noise and work better than crisp interval rules used by robust learners such as C4.5. These characteristics enable the creation of a better generalization from the instances of the problem, since they better avoid the overfitting of noisy data and, therefore, obtain more robust and accurate models.

In order to carry out this comparison, we will consider 19 datasets from the KEEL-dataset repository [10]. Four different levels of noise are taken into account in the experimentation: 5%, 10%, 15% and 20%. Thus, 76 new synthetic datasets are created with class noise in the training sets. As we will consider two different types of class noise, the number of datasets created is doubled, for an experimentation with a total of 171 datasets. We will obtain the test accuracy of the models created with all the classification algorithms and we will use the Wilcoxon's statistical test [11] in order to check the significance of the differences found. We will propose a measure to quantify the degradation of the test accuracy of the models with the introduction of noise with respect to the original obtained without noise. We will also check the number of rules of each model in order to see how the size of the models is

978-1-4577-1675-1

affected by the noise.

The remainder of this paper is organized as follows. Section II presents an introduction to classification with noisy data. Next, Section III describes the FRBCS used in our work. Section IV shows the details of the experimental framework, which summarizes the datasets used, the validation scheme and the process to build the noisy datasets, along with the parameters used by the classification algorithms, and the scheme of comparisons. Section V includes the analysis and the experimental results obtained by the FURIA algorithm versus the C4.5 robust learner. Next, in Section VI we analyze the causes of the good behavior of FRBCSs when dealing with class noise. Finally, in Section VII we make some concluding remarks.

II. Classification with Noisy Data

Real-world data is never perfect and often suffers from corruptions that may harm interpretations of the data, models created and decisions made. In classification, noise can negatively affect the system performance in terms of classification accuracy, time in building, size and interpretability of the classifier built [12].

The quality of any dataset is determined by a large number of components as described in [7]. Some of these are the source of the data and the input of the data, which are inherently subject to error.

Class labels and attributes are two information sources which can influence the quality of a classification dataset. The quality of the class labels represents whether the class of each instance is correctly assigned; and the quality of the attributes indicates how well the attributes characterize instances for classification purposes.

Based on these two information sources which define the quality of a classification dataset we can distinguish two types of noise in a given dataset [13]: class noise and attribute noise.

1) Class noise or labeling errors occur when an instance belongs to the incorrect class. Class noise can be attributed to several causes, including subjectivity during the labeling process, data entry errors, or inadequacy of the information used to label each object. There are two possible types of class noise:
   - Contradictory examples: the same examples appear more than once and are labeled with different classes [14].
   - Misclassifications: instances are labeled with the wrong classes [15].
2) Attribute noise is used to refer to corruptions in the values of one or more attribute of instances in the dataset. Examples of attribute noise include: erroneous attribute values, missing or unknown attribute values, and incomplete attributes or "do not care" values.

The two most common approaches to noisy data in the literature are robust learners and noise preprocessing techniques:

- Robust learners are characterized by being less influenced by noisy data. An example of a robust learner is the C4.5 algorithm [8]. C4.5 uses pruning strategies to reduce the chances of trees being built with noise in the training data [16]. However, when the noise level becomes relatively high, even a robust learner may obtain a poor performance.
- Noise preprocessing techniques try to remove the negative impact of noise in the datasets prior to creating a model over the original data. Among these techniques, the most well-known methods are noise filtering ones. Their objective is to identify noisy instances which can be eliminated from the training data [17], [18].

In this contribution, we study mislabeled data as noise because it is very common in real-world data [12], [15]. These errors can be produced in situations where different classes have similar symptoms, as generally happens on the class boundaries. Furthermore, we compare the behavior of the FRBCS considered in our work with the well-known C4.5 robust learner. We want to verify that the effect of class noise on the accuracy and size of the models created by the FURIA algorithm is lower than on the models built by the C4.5 robust learner.

III. Fuzzy Rule Based Classification Systems

This section describes the basis of the fuzzy model that we have used in our study. First we introduce the basic notation that we will use later to describe the FRBCS. Next we describe the FURIA method in Subsection III-A.

Any classification problem consists of w training patterns $x_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, 2, \ldots, w$, labeled with one of M possible classes $\mathbb{L} = \{\lambda_1, \ldots, \lambda_M\}$, where x_{pi} is the i-th attribute value ($i = 1, 2, \ldots, n$) of the p-th training pattern. In this paper, we use fuzzy rules with a single class and a rule weight associated to this class in the consequent [19]:

$$\begin{aligned}\text{Rule } R_j :&\texttt{IF } x_1 \texttt{ is } A_j^1 \texttt{ AND } \ldots \texttt{ AND } x_n \texttt{ is } A_j^n \\ &\texttt{THEN CLASS} = C_j \texttt{ WITH } RW_j\end{aligned} \tag{1}$$

where R_j is the label of the j-th rule, $x = (x_1, \ldots, x_n)$ is an n-dimensional pattern vector, A_j^i is an antecedent fuzzy set, C_j is a class label and RW_j is the rule weight [20].

A. Fuzzy Unordered Rule Induction Algorithm

FURIA [4] builds upon the RIPPER interval rule induction algorithm [21]. The model built by FURIA uses fuzzy rules of the form given in Equation (1) where A_j^k is a fuzzy set $I^F = (\phi^{s,L}, \phi^{c,L}, \phi^{c,U}, \phi^{s,U})$ with a trapezoidal

membership function

$$I^F(v) = \begin{cases} 1, & \text{if} \quad \phi^{c,L} \leq v \leq \phi^{c,U} \\ \frac{v-\phi^{s,L}}{\phi^{c,L}-\phi^{s,L}}, & \text{if} \quad \phi^{s,L} \leq v \leq \phi^{c,L} \\ \frac{\phi^{s,U}-v}{\phi^{s,U}-\phi^{c,U}}, & \text{if} \quad \phi^{c,U} \leq v \leq \phi^{s,U} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

and $C_j \in \mathbb{L} = \{\lambda_1, \ldots, \lambda_M\}$ is a class label. The rule weight RW_j of the rule R_j is computed as

$$RW_j = \frac{2\frac{|D_T^{(c)}|}{|D_T|} + \sum_{x \in D_T^{(c)}} \mu_{R_j}(x)}{2 + \sum_{x \in D_T} \mu_{R_j}(x)} \tag{3}$$

where D_T denotes the training set instances, $D_T^{(c)}$ denotes the subset of training instances with the label λ_c and $\mu_{R_j}(x) = \prod_{i=1\ldots n} I_i^F(x_i)$

To assign an output to a new example, suppose that fuzzy rules $R_1, \ldots, R_k$ have been learned for class λ_c. For a new query instance x, the support of this class is defined by

$$s_c(x) = \sum_{j=1}^{k} \mu_{R_j}(x) RW_j \tag{4}$$

The class predicted by FURIA is the one with maximal support. In the case of a tie, a decision in favor of the class with the highest frequency is made. When the query is not covered by any rule, a rule stretching method is proposed based on modifying the rules in a local way so as to make them applicable to the query. In order to do this it is checked the order in which the antecedents appear in the rule, and all premises from the first one that do not match the new instance are eliminated.

FURIA builds the fuzzy rule base by means of these two steps:

1) Learn a rule set for every single class λ_c of the problem, using a one-versus-all decomposition. In order to do this, the RIPPER algorithm is used, which consists of two fundamental steps described in [21]: the building and the optimization phase.
2) Obtain the fuzzy rules by means of fuzzifying the final rules from the above step. Each rule is fuzzified retaining the same structure as the original rule and replacing original intervals in the antecedent with fuzzy intervals. To fuzzify an interval, it is required to compute the four parameters needed for the trapezoidal fuzzy set from the original interval (complete procedure is described in [4]).

IV. Experimental Framework

In this section, we first describe the original datasets our experimentation is based on in Subsection IV-A. Then, in Subsection IV-B, the noise introduction process over the above mentioned original datasets and the class noise levels in order to create the final datasets are presented. Section IV-C indicates the parameters for the classification algorithms used for this work. Finally, Section IV-D establishes the comparison methodology carried out between the FRBCS and the robust learner considered.

A. Original Datasets

The experimentation has been based on 19 datasets taken from the KEEL-dataset repository[1] [10]. Table I summarizes the properties of the originally selected datasets. For each dataset, the number of instances (#Ins), the number of numeric attributes (#Att) along with the number of real and integer attributes (R/I) and the number of classes (#Cla) are presented.

Table I
Original Datasets Used from the KEEL-Dataset Repository

Dataset	#Ins	#Att (R/I)	#Cla	Dataset	#Ins	#Att (R/I)	#Cla
contraceptive	1,473	9 (0/9)	3	satimage	6,435	36 (0/36)	7
ecoli	336	7 (7/0)	8	segment	2,310	19 (19/0)	7
glass	214	9 (9/0)	7	sonar	208	60 (60/0)	2
heart	270	13 (1/12)	2	spambase	4,597	57 (57/0)	2
ionosphere	351	33 (32/1)	2	thyroid	7,200	21 (6/15)	3
iris	150	4 (4/0)	3	twonorm	7,400	20 (20/0)	2
page-blocks	5,472	10 (4/6)	5	wdbc	569	30 (30/0)	2
penbased	10,992	16 (0/16)	10	wine	178	13 (13/0)	3
pima	768	8 (8/0)	2	yeast	1,484	8 (8/0)	10
ring	7,400	20 (20/0)	2				

The accuracy estimation of each classifier is obtained by means of 5 runs of a stratified 5-fold cross-validation. The dataset is divided into 5 partition sets with equal numbers of examples and maintaining the proportion between classes in each fold. Each partition set is used as a test for the model learned from the four remaining partitions. This procedure is repeated 5 times. We use 5 partitions since if each partition has a large number of examples the noise's effects will be more notable, facilitating their analysis.

B. Process for Inducing Noise in Datasets

The initial amount of noise present in the previous datasets is unknown so we cannot make any assumptions about this base noise level. Therefore, as we want to control the level of noise in the existing data, we use a manual mechanism to add noise to each dataset.

From the 19 original datasets from the KEEL-dataset repository we have created new noisy datasets considering the introduction of class noise in the training sets. We have taken into account four levels of noise: $x = 5\%$, $x = 10\%$, $x = 15\%$ and $x = 20\%$. Introducing noise only in training sets and testing the models built over clean test sets will let us to check how classifier's generalization capability is affect by the noise's effect.

In order to introduce a level of class noise $x\%$ in a dataset, we use two different schemes:

- *Pairwise class noise scheme*. Class noise is introduced into the datasets following the pairwise scheme used

[1] http://www.keel.es/datasets.php

in [15]: given a pair of classes (X, Y), with X the majority class and Y the second majority class, and a noise level $x\%$, an instance with the label X has a probability of $x\%$ of being incorrectly labeled as Y.

- *Random class noise scheme*. We have also used a more general class noise scheme than that described above. In this scheme, a level of noise of $x\%$ supposes that exactly $x\%$ of the examples are corrupted. The class labels of these examples are aleatory changed by different ones within the domain of the class.

In order to create a noisy synthetic dataset from the original one, the noise is introduced consistently by means of the following steps:

1) A level of noise $x\%$ of a concrete type of class noise is introduced into a copy of the full original dataset.
2) Both datasets, the original one and the noisy copy, are partitioned into 5 equivalent folds, i.e. the examples within each fold of the noisy copy are the same as those within the corresponding fold of original dataset.
3) We use a 5-fold cross-validation scheme for new synthetic datasets. The datasets are created by building the training sets with the noisy copy and the test sets with the original copy.

In this manner, we have created 76 datasets with the pairwise class noise scheme and 76 with the random class noise scheme. The total number of datasets of the experimentation is therefore 171.

C. Parameters Configuration

The classification algorithms have been executed with the KEEL tool[2] [22] using the best parameters on average as shown in Table II.

Table II
PARAMETERS CONFIGURATION FOR CLASSIFICATION ALGORITHMS

FURIA	C4.5
• Number of folds: $f = 3$ • Num. of optimizations: $k = 2$ • Min. instances per premise: $i = 2$	• Confidence level: $c = 0.25$ • Min. instances per leaf: $i = 2$ • Prune after the tree building

D. Comparison methodology

In order to check which kind of algorithms, FRBCSs or robust crisp methods, are more tolerant when dealing with class noise, we compare the FURIA fuzzy method with the C4.5 crisp robust learner. We perform this comparison training the methods with noisy data, and testing the models are over clean data. In order to be able to carry out this study we use three distinct methods:

1) The mean accuracy provided by the classification algorithms over the test sets for each level of induced noise, defined as its performance averaged across all classification problems. Over the test accuracy results, we also use the Wilcoxon's signed ranks statistical test [11] with a level of significance of $\alpha = 0.05$. For each level of noise, we compare an FRBCS versus a crisp method using the Wilcoxon's test and we obtain the p-values associated with these comparisons.
2) We use the *relative loss of accuracy* (*RLA*) (Equation 5) to observe the form in which the accuracy of the model is affected when increasing the levels of noise with respect to the case with no noise:

$$RLA_{x\%} = \frac{Acc_{0\%} - Acc_{x\%}}{Acc_{0\%}} \tag{5}$$

where $RLA_{x\%}$ is the relative loss of accuracy at a level of noise $x\%$, $Acc_{0\%}$ is the mean accuracy in test in the original case, that is, with 0% of induced noise, and $Acc_{x\%}$ is the mean accuracy in test with a level of noise $x\%$.
3) We also use the *relative increase of rules* (*RIR*) (Equation 6) since another aspect that can be affected by the noise is the model's size [12], [18] and therefore, the number of rules can be related to the robustness of the model learned:

$$RIR_{x\%} = \frac{Rules_{x\%} - Rules_{0\%}}{Rules_{0\%}} \tag{6}$$

where $RIR_{x\%}$ is the relative increase of rules at a level of noise $x\%$, $Rules_{0\%}$ is the mean number of rules of the model learned from the training set with no additional noise, and $Rules_{x\%}$ is the mean number of rules of the model learned from the training set with a level of noise $x\%$.

V. CLASS NOISE'S EFFECT ON CLASSIFIERS' PERFORMANCE

In this section we focus on the analysis of the behavior of the FURIA fuzzy method versus the C4.5 algorithm when training with noisy data and the models are tested over clean test sets.

Table III shows the results of both schemes of class noise considered. The first part of the table shows the mean accuracy in test at each level of induced noise. Along with these results, the second part of the table shows the Wilcoxon's test p-values.

The mean accuracy in test of FURIA is always better than that of C4.5 for each level of induced noise in both noise schemes. This clearly shows the better performance of the FRBCS when training with data with class noise. From the associated p-values (considering a level of significance of $\alpha = 0.05$) we can say that there are significant differences in the results. This occurs with both class noise schemes for all levels of noise. However, it highlights the better behavior of FURIA with the random class noise scheme with respect to C4.5, due to the latter's test accuracy being more affected

[2]www.keel.es

Table III
RESULTS ON DATASETS WITH CLASS NOISE: TEST ACCURACY AND RELATED P-VALUES

		Mean accuracy in test					p-values for class noise				
	Noise %	**0%**	**5%**	**10%**	**15%**	**20%**	**0%**	**5%**	**10%**	**15%**	**20%**
Pairwise	FURIA	85.81	85.37	84.74	84.23	83.10	1.1444E-5	1.9074E-5	9.652E-4	1.6404E-4	2.022E-3
	C4.5	83.93	83.66	82.81	82.25	81.41					
Random	FURIA	85.81	85.17	84.54	84.06	83.66	1.1444E-5	7.63E-6	3.356E-4	3.814E-5	1.1444E-5
	C4.5	83.93	82.97	82.38	81.69	80.28					

Table IV
RESULTS ON DATASETS WITH CLASS NOISE: RELATIVE LOSS OF ACCURACY IN TEST AND RELATIVE INCREASE OF RULES IN TRAINING

		Relative loss of accuracy				Relative increase of rules			
	Noise %	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**
Pairwise	FURIA	0.0051	0.0127	0.0191	0.0331	-0.03	-0.07	-0.08	-0.10
	C4.5	0.0035	0.0141	0.0212	0.0323	0.05	0.11	0.16	0.18
Random	FURIA	0.0073	0.0152	0.0207	0.0253	-0.01	-0.04	-0.07	-0.08
	C4.5	0.0124	0.0204	0.0285	0.0466	0.10	0.14	0.27	0.30

than that of the former one than in the case of the pairwise class noise scheme. The p-values also reflect this fact, since lower p-values are generally obtained for the random class noise scheme.

In order to obtain an approximation of the greater or lower robustness of the considered methods against class noise, Table IV shows the averages of the results of relative loss of accuracy in test of each classification algorithm and the relative increase of rules for each level of induced noise and both class noise schemes.

As is shown in Table IV, the *RLA* is lower for FURIA than for C4.5 at all considered levels of noise for both class noise schemes. However, with 5% and 20% of the pairwise class noise scheme this does not occur, although these values are very close. This again shows the greater robustness of FURIA when dealing with mislabeled data.

Regarding the *RIR*, for both class noise schemes, the results obtained by the FURIA fuzzy method must be highlighted. These results are indeed reduced with respect to the case with no noise when higher levels of class noise are introduced in the datasets. The number of rules of the FURIA algorithm is on average much better than that of the C4.5 algorithms. FURIA's rule stretching method can influence in this fact. We may conclude that the FURIA algorithm has greater robustness against class noise with respect to C4.5.

VI. REASONS OF FRBCSS' BETTER PERFORMANCE WITH DATA WITH CLASS NOISE

In this section we perform the analysis of the reasons why FRBCSs present greater robustness than crisp robust methods when dealing with data with class noise. This better behavior is due to FRBCSs having a series of properties that make them different from most of the crisp systems when dealing with class noise. Some of these properties, the most general, that we can emphasize are:

1) *The use of fuzzy sets in the antecedents of the rules, instead of crisp intervals*. This lets, for instance, to give more or less importance to the class of an example predicted by a rule, according to whether this example falls in one area or another of the membership function of the antecedents of this rule. Noisy examples can fall in areas with a lower value of the membership function of the antecedents of the rule while belonging to a different class to that predicted by the rule. Thus, these noisy examples will be influenced to a lower degree by the prediction of this rule.
2) *The assignment of a weight to each fuzzy rule*. This, along with the fuzzy sets in the antecedents, enables an overlapping between fuzzy rules. If several rules cover an example, the rule weights (and the membership function in the fuzzy sets) will let to determine the most appropriate rule that covers this example. The possible overlapping between rules is a very important fact when dealing with noisy data, because it causes the rules to be less affected by the noise that corrupts other rules.
3) *The aggregation of the fuzzy rules' predictions in order to predict the final class of an example*. This is a natural, robust way to deal with noise because the prediction is not only determined by the action of a single rule, but it is determined by the intervention of all or a part of the rules of the model. It is possible to use this thanks to the two above mentioned characteristics.

These properties cause the FRBCSs to be less affected when class noise is induced in datasets. Therefore, these systems achieve a lower overfitting of noise, leading to an increase in the accuracy of labeling test examples.

VII. Concluding Remarks

In this contribution we have analyzed the advantages of FRBCSs when dealing with data with class noise. The good performance and tolerance of the FURIA fuzzy method compared with the C4.5 crisp robust learner when class noise is present has been highlighted.

We have considered two different kinds of class noise: the pairwise class noise scheme and the random class noise scheme. Based on them, we have created 76 datasets with the former one and 76 datasets with the latter, by introducing noise in the training partitions and the models have been evaluated over clean test sets.

The results obtained have indicated that FRBCSs have better test accuracy and a better robustness in terms of the model's size when training with data with class noise than classic crisp robust learners.

Acknowledgment

Supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01. J. A. Sáez holds an FPU scholarship from the Spanish Ministry of Education and Science.

References

[1] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining (Advanced Information Processing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2004.

[2] L. Kuncheva, *Fuzzy classifier design*. Springer-Verlag, 2000.

[3] J. C. Hühn and E. Hüllermeier, "FR3: a fuzzy rule learner for inducing reliable classifiers," *IEEE Transactions on Fuzzy Systems*, vol. 17, pp. 138–149, 2009.

[4] ——, "FURIA: An Algorithm for Unordered Fuzzy Rule Induction," *Data Mining and Knowledge Discovery*, vol. 19, no. 3, pp. 293–319, 2009.

[5] C.-H. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection," *Pattern Recognition*, vol. 40, pp. 2373–2391, 2007.

[6] G. Schaefer, M. Zviek, and T. Nakashima, "Thermography based breast cancer analysis using statistical features and fuzzy classification," *Pattern Recognition*, vol. 42, no. 6, pp. 1133–1137, 2009.

[7] R. Y. Wang, V. C. Storey, and C. P. Firth, "A Framework for Analysis of Data Quality Research," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 4, pp. 623–640, 1995.

[8] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1993.

[9] C.-M. Teng, "Correcting Noisy Data," in *Proceedings of the Sixteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1999, pp. 239–248.

[10] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework," *Journal of Multiple-Valued Logic and Soft Computing, in press*, 2010.

[11] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[12] X. Zhu and X. Wu, "Class Noise vs. Attribute Noise: A Quantitative Study," *Artificial Intelligence Review*, vol. 22, pp. 177–210, 2004.

[13] X. Wu, *Knowledge acquisition from databases*. Norwood, NJ, USA: Ablex Publishing Corp., 1996.

[14] M. A. Hernández and S. J. Stolfo, "Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem," *Data Mining and Knowledge Discovery*, vol. 2, pp. 9–37, 1998.

[15] X. Zhu, X. Wu, and Q. Chen, "Eliminating class noise in large datasets," in *Proceeding of the Twentieth International Conference on Machine Learning*, 2003, pp. 920–927.

[16] J. R. Quinlan, "The Effect of Noise on Concept Learning," in *Machine learning: an artificial intelligence approach*. Morgan Kaufmann Publishers, 1986, ch. 6, pp. 149–166.

[17] D. Gamberger, R. Boskovic, N. Lavrac, and C. Groselj, "Experiments With Noise Filtering in a Medical Domain," in *Proceedings of the Sixteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers, 1999, pp. 143–151.

[18] C. E. Brodley and M. A. Friedl, "Identifying Mislabeled Training Data," *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[19] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets and Systems*, vol. 52, pp. 21–32, 1992.

[20] H. Ishibuchi and T. Nakashima, "Effect of rule weights in fuzzy rule-based classification systems," *IEEE Transactions on Fuzzy Systems*, vol. 9, pp. 506–515, 2001.

[21] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann Publishers, 1995, pp. 115–123.

[22] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández, and F. Herrera, "KEEL: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, pp. 307–318, 2009.

4.2 Evaluating the classifier behavior with noisy data considering performance and robustness: the Equalized Loss of Accuracy measure

Sáez J. A., Luengo J., and Herrera F. (2014) Evaluating the classifier behavior with noisy data considering performance and robustness: the Equalized Loss of Accuracy measure (submitted)

- Status: **Submitted**.

Evaluating the classifier behavior with noisy data considering performance and robustness: the Equalized Loss of Accuracy measure

José A. Sáez[a,*], Julián Luengo[b], Francisco Herrera[a]

[a] *Department of Computer Science and Artificial Intelligence, University of Granada, CITIC-UGR, Granada, Spain, 18071*
[b] *Department of Civil Engineering, LSI, University of Burgos, Burgos, Spain, 09006*

Abstract

Noise is common in any real-world data set and may adversely affect classifiers built under the effect of such type of disturbance. Some of these classifiers are widely recognized for their good performance when dealing with imperfect data. However, the noise robustness of the classifiers is an important issue in noisy environments and it must be carefully studied. Both performance and robustness are two independent concepts that are usually considered separately, but the conclusions reached with one of these metrics do not necessarily imply the same conclusions with the other. Therefore, involving both concepts seems to be crucial in order determine the expected behavior of the classifiers against noise. This paper proposes a new measure to establish the expected behavior of a classifier with noisy data trying to minimize the problems of considering performance and robustness individually: the Equalized Loss of Accuracy (ELA). The advantages of ELA against other robustness metrics are studied and all of them are also compared in a case of study which considers the results of several classifiers with a different noise tolerance over numerous data sets. Both the analysis of the distinct measures and the empirical results show that ELA is able to overcome some of the problems that the rest of the robustness metrics could produce, being useful to represent the behavior of the classifiers against noise.

Keywords:
Noisy Data, Behavior against Noise, Robustness Measures, Classification.

*Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
Email addresses: smja@decsai.ugr.es (José A. Sáez), jluengo@ubu.es (Julián Luengo), herrera@decsai.ugr.es (Francisco Herrera)

1. Introduction

It is widely know that classifier performance is influenced by the quality of the training data upon which this classifier is built [20]. Since real-world data sets rarely are clean of corruptions [18], these can therefore affect the decisions taken by the classifiers built from these data [22]. However, the maximum achievable performance depends not only on the quality of the data, but also on the appropriateness of the chosen classification algorithm for the data.

Knowing what kind of classification algorithms are most suitable when working with noisy data is a challenging proposition [22, 15, 10]. Ideally, since the systems must be adapted to the data they treat, if the data that we train are characterized by their inaccuracy, then systems that create classifiers capable of handling some degree of imprecission are needed [20]. One may wonder how to know which systems are more suitable or are better adapted to deal with these noisy data. Even though some classifiers have been related to this capability of working with imperfect data, this fact is usually based on only checking the accuracy of those and other classifiers over a concrete collection of data sets, with independence of the type and noise level present in the data. This analysis procedure has a double disadvantage in noisy environments. First of all, the study of the performance alone does not provide enough information on the classifier behavior affected by the noise [9, 8, 14]. Moreover, a study with a controlled (probably artificial) noise level for each data set is also necessary to reach meaningful conclusions when evaluating the classifier behavior against noise [22].

This paper proposes a new single score to perform an analysis of the classifier behavior with noisy data trying to solve the aforementioned problems. This will be done from a double point of view focusing on the classic performance assessment of the methods but also on their robustness [9, 8, 14], an important issue in noisy environments that must be carefully studied. We understand as performance the accuracy of a classifier predicting the class of a new example, whereas the noise robustness has been defined as the classifier accuracy loss rate [9, 8], which is produced by presence of noise in the data, with respect to the case without noise. Since performance and robustness are different concepts, the conclusions that they provide may also be different, yet this comparative analysis remains disregarded in the literature.

Even though the robustness of the methods is important dealing with noisy data, there are a lack of proposals of robustness-based measures in the literature and the few existing ones also present several drawbacks. This paper will analyze the existing robustness measures in the classification framework focusing on their advantages and disadvantages. We will motivate the necessity of combining the robustness and performance concepts to obtain an unified conclusion on the expected behavior of the methods with noisy data. We will propose a new

behavior-against-noise measure to characterize the behavior of a method with noisy data, the Equalized Loss of Accuracy (ELA) measure, which tries to minimize the problems of considering performance and robustness measures individually.

In order to complete our analysis, we will perform an experimental evaluation of the behavior and representativeness of the different measures, considering several classifiers with a known behavior against noise (concretely, the C4.5 decision tree generator [13] and a Support Vector Machine [6]). The behavior of such classifiers described by using ELA will be tested using 32 data sets from the KEEL-dataset repository [2], over which we will introduce a 10% of noise level into the class labels in a controlled way [22]. All these data sets and other complementary material associated with this paper, such as the performance and robustness-based metrics results, are available at the web-page `http://sci2s.ugr.es/ela_noise`.

The rest of this paper is organized as follows. Section 2 presents an introduction to noisy data and robustness in classification. Next, Section 3 describes the new proposed measure ELA. Section 4 shows the details of the experimental framework including the noise introduction process, the parameter setup for the algorithms and the comparison methodology. Section 5 includes the analysis of the experimental results obtained with the different robustness-based metrics. Finally, in Section 6 we point out some concluding remarks.

2. Classification with noisy data

This section presents an introduction to noisy data in the field of classification, found in Section 2.1. Then, the concept of robustness in classification is explained in Section 2.2.

2.1. Introduction to noisy data

The quality of any data set is determined by a large number of components as described in [18]. Two of these are the source of the data and the input of the data, which are inherently subject to error. Thus, real-world data is rarely perfect it is often affected by corruptions that hinder the models built as well as the interpretations and decisions made from them. In the particular framework of classification, the most notable effect of noise is that it negatively affects the system performance in terms of classification accuracy, time in building, size and interpretability of the model obtained [21, 22]. In the literature there are two types of noise distinguished [19]:

1. *Class noise* [4, 1]. Also known as labeling errors, they occur when an instance belongs to the incorrect class due to, for example, data entry errors or inadequacy of the information used to label each instance.

2. *Attribute noise* [22, 17]. This is used to refer to corruptions in the attribute values of instances in a data set. Examples of attribute noise include: erroneous attribute values, missing or unknown attribute values, and incomplete attributes or "do not care" values.

In this paper we consider the most common type of class noise, which is also the most disruptive; this is known as *misclassifications* and refers to those examples incorrectly labeled with a wrong class label [22].

Therefore, since errors in real-world data sets are common, actions must be taken to mitigate their consequences [19]. Several methods have been studied in the literature to deal with noisy data [22]. They follow two main postulates: (i) the adaptation of the algorithms to properly handle the noise [13], [5] and; (ii) the preprocessing of the data sets aiming to remove or correct the noisy examples [3]. The former are also known as *robust learners* and they are characterized by being less influenced by noisy data. An example of a robust learner is the C4.5 algorithm [13] considered in the experimental case of study of this paper, which uses pruning strategies to reduce the chances that the trees are overfitting to noise in the training data [12]. However, if the noise level is relatively high, even a robust learner may have a poor performance.

2.2. Robustness measures

Noise hinders the knowledge extraction from the data and spoils the models obtained using these noisy data when they are compared to the models learned from clean data from the same problem [22]. In this sense, robustness [7] is the capability of an algorithm to build models that are insensitive to data corruptions and suffer less from the impact of noise; that is, the more robust an algorithm is, the more similar the models built from clean and noisy data are. Thus, a classification algorithm is said to be more robust than another if the former builds classifiers which are less influenced by noise than the latter. Robustness is considered very important when dealing with noisy data, because it allows one to expect *a priori* the amount of variation of the learning method's performance against noise with respect to the noiseless performance in those cases where the characteristics of noise are unknown. It is important to note that a higher robustness of a classifier does not imply a good behavior of that classifier with noisy data, since a good behavior implies a high robustness but also a high performance without noise.

In the literature, the measures that are used to analyze the degree of robustness of the classifiers in the presence of noise compare the performance of the classifiers learned with the original (without controlled noise) data set with the performance of the classifiers learned using a noisy version of that data set. Therefore, those classifiers learned from noisy data sets that are more similar (in terms of results) to the noise-free classifiers will be the most robust ones.

To the best of our knowledge, the robustness-based measures found in the literature are the two following:

1. The robustness measure proposed in [9] considers the performance of Bayesian Decision rule as a reference, which is considered as the classifier providing the minimal risk when the training data are not corrupted. Concretely, the next expression is used:

$$BRM_{x\%} = \frac{E_{x\%} - E}{E}, \tag{1}$$

where $E_{x\%}$ is the risk (that we will understand as the classification error rate in our case) of the classifier at a noise level $x\%$ and E is the risk of the Bayesian Decision rule without noise. This classifier is a theoretical decision rule, that is, it is not learned from the data, which depends on the data generating process. Its error rate is by definition the minimum expected error that can be achieved by any decision rule.

2. The *Relative Loss of Accuracy* (RLA) is the robustness measure employed in [14] and was defined as:

$$RLA_{x\%} = \frac{A_{0\%} - A_{x\%}}{A_{0\%}}, \tag{2}$$

where $A_{0\%}$ is the accuracy of the classifier with a noise level 0%, and $A_{x\%}$ is the accuracy of the classifier with a noise level $x\%$. RLA evaluates the robustness as the loss of accuracy with respect to the case without noise $A_{0\%}$, weighted by this value $A_{0\%}$. This measure has two clear advantages: (i) it is simple and interpretable and (ii) to the same values of loss $A_{0\%} - A_{x\%}$, the methods having a higher value of accuracy without noise $A_{0\%}$ will have a lower RLA value.

In the next section, some shortcomings of these measures are explained.

3. The Equalized Loss of Accuracy measure

This section discusses the problems of the existing robustness-based measures as indicators of the behavior of a classifier with noisy data (Section 3.1) and the necessity of combining the robustness and the performance of the classifier (Section 3.2). Finally, we present the ELA measure as our proposal (Section 3.3).

3.1. Problems of the existing robustness measures

Even though the robustness-based measures presented in the above section let us to evaluate the higher or lower robustness of the classifiers, they have a series of important disadvantages

which do not make their usage recommendable. In [8] two main points that a robust algorithm must satisfied are established:

1. It must have a good initial accuracy $A_0\%$.
2. When the noise level is increased, it must suffer a low loss if the noise level is low and the performance must not be drastically deteriorated when the noise level is high.

The measure proposed in [9] (Equation (1)) considers these two points, even though it has a clear disadvantage: it is based on a theoretical model. Thus, the quality of the performance without noise is determined with respect to that of the optimal Bayesian classifier for the data. This optimal performance can be rarely computed since it is typically characterized by probability distributions on the input/output space, which are intrinsic to the data set and are rarely known. The estimation of these distributions from the data generally requires the choice of a known probability distribution that could not properly represent the characteristics of the data. For this reason this measure is not feasible for practical cases, where mixed data and computation time bound the obtention of the Bayesian classifier.

On the other hand, the RLA measure (Equation (2)) has several points considered as drawbacks:

- From the two points implied in the definition of a robust algorithm given above, point 1, that is, the necessity to have a good initial performance $A_0\%$, has a very low influence in the RLA equation; being point 2 the main aspect computed by it.

- Classifiers obtaining a poor generalization from the training data without noise, that is, those in which $A_{0\%}$ is low, are usually affected in a lower degree by the presence of noise (their RLA value is therefore lower) than classifiers with a high $A_{0\%}$. In these cases, the lower loss of accuracy is not due to the better capability of the algorithm to get adapted to noise but for their inability to successfully model the data and for creating too general models that are little affected by noise.

- The RLA values do not represent the behavior against noise. For example, consider a random classifier in a balanced data set with $A_{0\%} = 50\%$. This classifier may maintain an accuracy $A_{x\%} = 50\%$ for different noise levels $x\%$ implying a robustness $RLA_{x\%} = 0$. On the other hand, a classifier with a higher starting accuracy suffering from a very low loss of accuracy when noise level increases has higher RLA values always $RLA_{x\%} > 0$, and then it is less robust, even though its behavior with noisy data is better.

- The RLA measure presents problems if $A_{x\%}$ is higher than the base accuracy $A_{0\%}$, obtaining negative numbers of RLA. This fact is more frequent with classifiers with a low

base accuracy $A_{0\%}$, whereas it is more rare with a good classifier with a high base accuracy $A_{0\%}$. These low negative values are interpreted as an excellent robustness, but they denote a very bad working of the classifier without noise.

Apart from the aforementioned problems, the main drawback of the RLA measure is that it assumes that both methods have the same robustness ($RLA_{0\%} = 0$) in the case without controlled noise $x = 0\%$. However, the information of the robustness without controlled noise must be also taken into account. If we are interested in analyzing only a single classifier, the RLA measure may suffice, but it fails when comparing two different methods when their performance without noise are different as important information is being ignored. RLA analyzes the robustness in the classic sense of variation with respect to the case without noise and thus the problem of knowing which methods will behave better with noisy data is considered partially. Therefore, it seems necessary to somehow combine the robustness in the sense of performance variation (as RLA makes) with the behavior without noise as (it is performed in [9]), but determining the quality of that initial accuracy without depending of the results of any external nor theoretical classifier.

3.2. Combining performance and robustness

As we as commented above, focusing only in the robustness, such as RLA makes it, in order to determine the behavior of several methods against noise is a partial way to address the problem. Thus, it is also important to properly consider the performance of these methods without noise. For example, consider a classifier C_1 that is only slightly affected by the noise and another classifier C_2 that is affected by the noise in a higher degree. If we ignore the initial accuracy $A_{0\%}$ of both methods without noise, the following two cases can be produced:

- If both methods C_1 and C_2 obtain two high and similar performances without noise, we would probably choose C_1 as the more robust method as it probably outperforms the method C_2 so far when we deal with new noisy data sets.
- If the performance of C_1 is significantly lower than that of C_2 without noise, then the method C_2 could be expected to be more accurate than the method C_1 when noise appears thanks to C_2's initial good behavior in spite of having a higher degradation of performance.

If we consider the usage of RLA, the second case would be incorrectly described. Furthermore, with the RLA measure, bad classifiers will have less probability to deteriorate their results to the same scale that a good classifier (they could indeed improve their results in an extreme case) when noise is introduced. Finally, in order to better understand the importance

of the initial performance (A_0), consider how the RLA measure is defined for the limits of the initial performance A_0. For a classifier C_1 with initial performance $A_0 = 1$, the only possible variation when introducing noise is that the classifier to be hindered as $A_x \leq A_0$. However, for another classifier C_2 with an very low initial accuracy $A_0 \approx 0$ the opposite may occur, being probably $A_x \geq A_0$. Therefore, we will obtain that $RLA(C_1) \leq RLA(C_2)$. That would mean that the worst imaginable classifier behaves better with noise than the almost perfect classifier.

3.3. The ELA measure

In order to overcome the problems mentioned in the above sections, we propose a correction of the RLA measure inspired in the measure proposed in [9]. The new measure is:

$$ELA_{x\%} = \frac{100 - A_{x\%}}{A_{0\%}}, \tag{3}$$

Using a pessimistic approach comparing to the perfect classifier instead of the optimal theoretical Bayesian classifier, it is possible to derive the expression mentioned as:

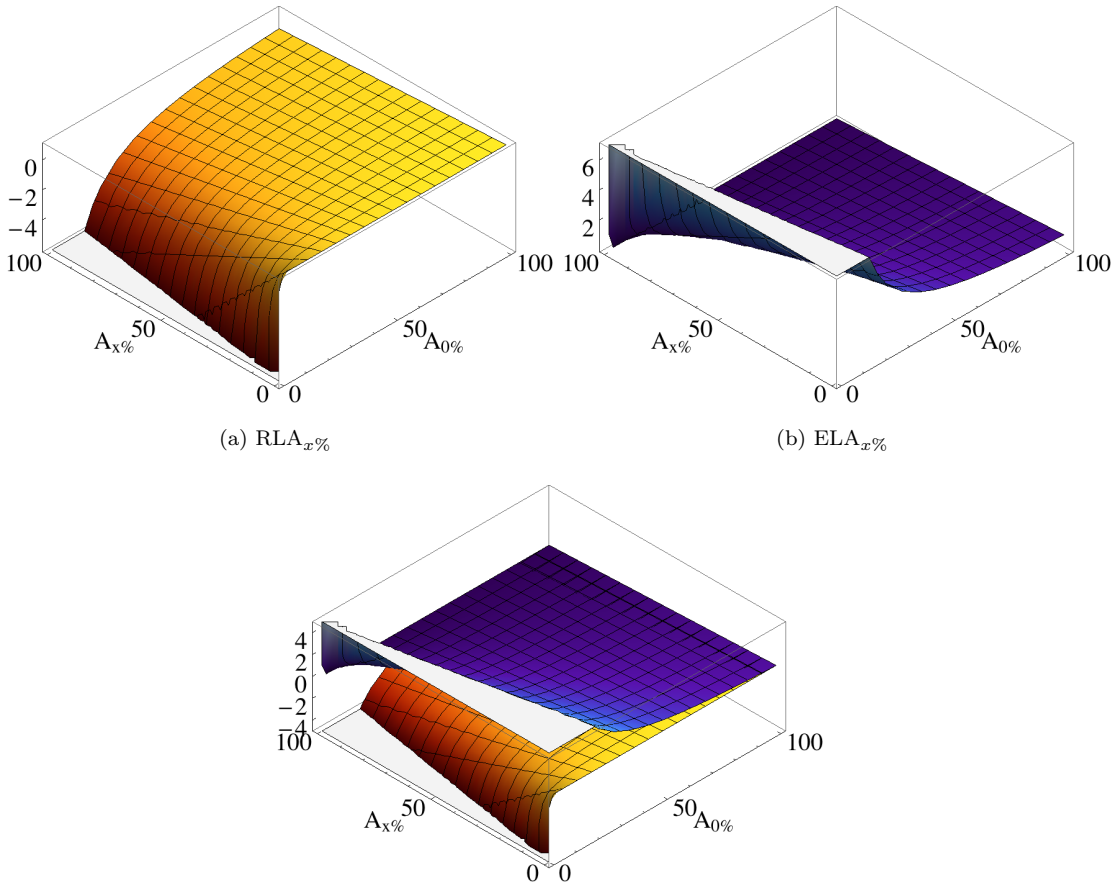
$$ELA_{x\%} = \frac{100 - A_x}{A_0} = \frac{100 - A_x + A_0 - A_0}{A_0} = \frac{A_0 - A_x}{A_0} + \frac{100 - A_0}{A_0} = RLA_{x\%} + f(A_0) \tag{4}$$

Therefore, ELA combines the robustness computed by RLA and a factor depending on the initial accuracy A_0 ($f(A_0)$ in Equation 4). Please, note that this factor $f(A_0)$ is precisely $ELA_{0\%}$, that is, $f(A_0) = ELA_{0\%} = (100 - A_0)/A_0$. Therefore, if we define $ELA_{x\%}$ as a measure of *behavior with noise* at a given noise level $x\%$, then $ELA_{x\%}$ is based on:

- the robustness of the method, that is, the loss of performance at a controlled noise level $x\%$ ($RLA_{x\%}$).
- the *behavior with noise* for the clean data, that is, without controlled noise ($ELA_{0\%}$).

Figure 1 shows a graphical 3-dimensional representation of the RLA and ELA measures, in which several similarities and differences among these two metrics can be appreciated. For example, both metrics have similar values when $A_{0\%}$ is high and diverge along with the decrement of $A_{0\%}$ (even though in this case both RLA and ELA have higher values when $A_{x\%}$ is lower and lower values when $A_{x\%}$ is higher). This divergence is produced thanks to the correction obtained by considering the intial accuracy in the measure and is essential to overcome the limitations of RLA.

The ELA measure changes the initial reference $A_{0\%}$ of the RLA measure by a constant value. As expressed by the BRM measure, the constant value should be the best attainable



(a) $\mathrm{RLA}_{x\%}$

(b) $\mathrm{ELA}_{x\%}$

(c) $\mathrm{RLA}_{x\%}$ vs $\mathrm{ELA}_{x\%}$

Figure 1: Representations of the RLA and ELA metrics.

accuracy value, and as proposes BRM the optimal Bayesian Decision Rule should be used to obtain a theoretical best accuracy value based on the joint underlying distributions of the data set. However, and as we have previously stated, this optimal value is rarely known. For this reason we choose an upper bound to this unknown in practice optimal Bayesian classifier's accuracy instead. The safer and most pessimistic value used for A_0 is fixed to 100% considering it as the accuracy of a perfect classifier. In this way, the loss of accuracy respect to the perfect classifier is weighted by the base accuracy $A_{0\%}$. As a result when taking into account the same loss of accuracy $100 - A_{x\%}$, the classifier with better value of base accuracy $A_{0\%}$ is considered to have a better behavior against noise.

This measure used to evaluate the behavior of a classifier with noisy data overcomes some problems of the RLA measure:

1. It takes into account the noiseless performance $A_{0\%}$ when considering which classifier is

more appropriate with noisy data. This fact makes ELA more suitable to compare the behavior against noise between 2 different classifiers. We must take into account that a benchmark data set might contain implicit and not controlled noise with a noise level $x = 0\%$.

2. A classifier with a low base accuracy $A_{0\%}$ that is not deteriorated at higher noise levels $A_{x\%}$ is not better than another better classifier suffering from a low loss of accuracy when the noise level is increased.

Table 1 shows four simple examples that extend the aforementioned ideas about RLA and ELA. These examples describe the possible problems caused by RLA and how they are solved by the ELA measure. In these examples, we consider a classifier C_1 with a good initial performance without noise and a classifier C_2 with a worse performance without noise. These are the scenarios studied:

Example 1. *C_1 and C_2 are equally hindered when the noise is introduced.*

In this case, C_1 is always more robust than C_2 with both measures (RLA and ELA). To the same amount of loss of accuracy, both ELA and RLA give more importance to the method with a higher performance (C_1). However, ELA makes more remarkable the difference between C_1 and C_2 since it takes into account the initial situation (Initial ACC) and the loss of accuracy (Final ACC). This information shows that is much probably that C_1 behaves well with noisy data considering ELA, whereas with RLA the differences would be much lower.

Furthermore, it is important to note that, when the performance of both classifiers without and with noise do not vary, RLA gives the same importance to both of them, but ELA clearly establishes C_1 as that classifier with the best behavior with noisy data (and this fact fits more to the desirable answer than that provided by RLA).

Example 2. *C_1 and C_2 are hindered when the noise is introduced but both have the same RLA values.*

In this case, even though the classifier C_2 is equally robust than C_1, C_2 obviously is not better than C_1 and ELA is able to reflect this issue. This situation (to a same RLA value equal to 0) also occurs if the classifiers do not alter their performance when the noise is introduced: from InitialACC = 100 to FinalACC = 100, then ELA(C_1) = 0, whereas from InitialACC = 50 to FinalACC = 50, ELA(C_2) = 1, so C_1 will behave better with noisy data than C_2.

Example 3. *C_1 and C_2 are benefited when the noise is introduced.*

In this rare case, both methods gain the same amount of performance when the noise is introduced. RLA shows that method C_2, which has a very poor performance, is more robust than the method C_1. ELA solves this problem and shows that the classifier C_1 has a better behavior with noisy data that the classifier C_2.

Example 4. *C_1 is slightly affected by the noise whereas C_2 is slightly benefited by the noise.*

This case clearly shows that RLA does not take into account the initial accuracy. Again, the classifier C_2 is more robust with RLA, whereas it clearly behaves worse with the ELA measure.

Table 1: Four different illustrative toy examples comparing RLA versus ELA.

	Classifier C_1				Classifier C_2			
	Initial ACC	Final ACC	RLA	ELA	Initial ACC	Final ACC	RLA	ELA
Example 1	100	100	0	0	50	50	0	1
	100	96	0.04	0.04	50	46	0.08	1.08
	100	92	0.08	0.08	50	42	0.16	1.16
Example 2	100	100	0	0	50	50	0	1
	100	96	0.04	0.04	50	48	0.04	1.04
	100	92	0.08	0.08	50	46	0.08	1.08
Example 3	80	80	0	0.25	30	30	0	2.33
	80	84	-0.05	0.2	30	34	-0.13	2.2
	80	88	-0.1	0.15	30	38	-0.27	2.07
Example 4	100	100	0	0	50	50	0	1
	100	99.996	0.00004	0.00004	50	50.004	-0.00008	0.99992
	100	99.992	0.00008	0.00008	50	50.008	-0.00016	0.99984

4. Experimental framework

In this section, we present the details of the experimentation developed in this paper. We first show how to build the noisy data sets in Section 4.1. Then, Section 4.2 indicates the classification methods used and their parameters. Finally, Section 4.3 establishes the analysis methodology carried out.

4.1. Data sets

The experimentation has been based on 32 data sets taken from the KEEL-dataset repository[1] [2]. Table 2 summarizes the properties of the originally selected data sets. For each data

[1] `http://www.keel.es/datasets.php`

set, the number of instances (#Ins), the number of numeric attributes (#Att) and the number of classes (#Cla) are presented.

Table 2: Base data sets used in the experimentation.

Data set	#EX	#AT	#CL	Data set	#EX	#AT	#CL
automobile	159	25(15/10)	6	magic	19020	10(10/0)	2
balance	625	4(4/0)	3	monk	432	6(6/0)	2
banana	5300	2(2/0)	2	new-thyroid	215	5(5/0)	3
car	1728	6(0/6)	4	phoneme	5404	5(5/0)	2
cleveland	297	13(13/0)	5	pima	768	8(8/0)	2
contraceptive	1473	9(9/0)	3	ring	7400	20(20/0)	2
dermatology	358	33(1/32)	6	segment	2310	19(19/0)	7
ecoli	336	7(7/0)	8	sonar	208	60(60/0)	2
flare	1066	11(0/11)	6	spambase	4597	57(57/0)	2
german	1000	20(13/7)	2	twonorm	7400	20(20/0)	2
glass	214	9(9/0)	7	vehicle	846	18(18/0)	4
hayes-roth	160	4(4/0)	3	vowel	990	13(13/0)	11
heart	270	13(13/0)	2	wdbc	569	30(30/0)	2
ionosphere	351	33(33/0)	2	wine	178	13(13/0)	3
iris	150	4(4/0)	3	yeast	1484	8(8/0)	10
lymphography	148	18(3/15)	4	zoo	101	16(0/16)	7

In order to control the noise level in the existing data, we use a manual mechanism to add noise into each training data set. Thus, we have considered the introduction of class noise following the scheme proposed in [22, 16]. This scheme, also known as *random class noise scheme*, introduces a noise level $x\%$ into a data set by randomly changing the class labels of exactly $x\%$ of the examples by other one out of the other classes.

The accuracy estimation of each classifier is obtained by means of 5 runs of a stratified 5-fold cross-validation. The data set is divided into 5 partition sets with equal numbers of instances and maintaining the proportion between classes in each fold. Each partition set is used as a test for the classifier learned from the four remaining partitions. This procedure is repeated 5 times.

4.2. Parameters

Two learning algorithms have been chosen to be used in this paper: C4.5 [13] and SVM [6]. This choice is based on their good behavior in a large number of real-world problems; moreover, they were selected because these methods have a highly differentiated and well known noise-robustness. In the following, their noise-tolerance is described along with the parameter configuration used for the experimentation:

- **C4.5 decision tree generator** [13]. C4.5 is considered a robust learner, which uses pruning strategies to reduce the chances of classifiers being affected by noisy examples [12]. The parameter setup for C4.5 used in this paper is the following: confidence level (0.25), minimal instances per leaf (2) and prune after the tree building.

- **Support Vector Machine** [6]. Since SVM relies on the support vectors (that are training examples lying near the separating hyperplane) to derive the decision model, this can be easily altered including or excluding a single noisy example [11]. Thus, SVM should *a priori* be more noise-sensitive than C4.5. The parameter setup for SVM used in this paper is the following: type of Kernel (*Puk* with $\sigma = 1$, $\omega = 1$), cost ($C = 100$), tolerance (0.001) and parameter for the round-off error ($\epsilon = 10^{-12}$).

4.3. Methodology of analysis

The experimental analysis of the capabilities of the ELA measure will be based on a complete case of study which involves the two aforementioned classification algorithms with a different noise tolerance: the noise-robust algorithm C4.5 and the noise-sensitive method SVM. These methods will be tested over the 32 base data sets without noise, that is $x = 0\%$, and another 32 noisy data sets with the noise level $x = 10\%$, which will be created with the *random class noise scheme*. All the data sets created can be found on the web-page associated with this paper.

The classification accuracy of C4.5 and SVM will be computed on the 64 data sets (without and with noise), along with their corresponding ELA and RLA results for the noise level 10%. Please note that it is not our intention to establish the most robust method between C4.5 and SVM, but to provide an ample and varied test bed where the two methods' behavior will help us to show the benefits of ELA measure against RLA. Because of this, our analysis will be based on studying the similarities and differences between the evaluations of ELA and RLA on the behavior with noise of each classification algorithm with each data set.

5. Benefits of ELA against other robustness metrics: a case of study

In this section we focus on the analysis of the behavior of the classifiers to study (C4.5 and SVM) when training with noisy data considering the usage of the ELA and RLA measures. As we cannot know the probability distribution of the benchmark data sets, we cannot use BRM as a comparison measure. Table 3 shows the performance results of C4.5 and SVM for all the data sets considered in this paper (at the noise levels 0% and 10% as indicated in Section 4.3), and their ELA and RLA results. From this case of study, several observations can be appreciated, which be grouped into two main parts: global results (including the *average* and *best* rows in

Table 3) and the individual results for each data set. These remarks on the results presented in this table will be focused on the similarities and differences between RLA and ELA, attending to the problems of RLA and how ELA can solve them.

Table 3: Performance results for C4.5 and SVM at 0% and 10% of class noise level, and their ELA and RLA results at 10% for all the data sets considered. Best results are remarked in bold.

Measure	Performance				ELA		RLA	
Noise level	0%		10%		10%		10%	
Data set	C4.5	SVM	C4.5	SVM	C4.5	SVM	C4.5	SVM
autos	**77.10**	69.29	**73.56**	64.50	**0.3429**	0.5123	**0.0459**	0.0691
balance	77.73	**89.09**	78.27	**81.50**	0.2796	**0.2077**	**-0.0069**	0.0852
banana	88.98	**90.28**	88.96	**90.27**	0.1241	**0.1078**	0.0002	**0.0001**
car	**91.33**	64.83	**90.20**	60.68	**0.1073**	0.6065	**0.0124**	0.0640
cleveland	**51.58**	45.85	**52.26**	41.47	**0.9256**	1.2766	**-0.0132**	0.0955
contraceptive	**52.14**	47.56	**50.52**	46.80	**0.9490**	1.1186	0.0311	**0.0160**
dermatology	93.91	**96.92**	93.02	**96.59**	0.0743	**0.0352**	0.0095	**0.0034**
ecoli	**79.05**	78.11	**79.29**	67.81	**0.2620**	0.4121	**-0.0030**	0.1319
flare	**73.86**	70.34	**74.15**	71.82	**0.3500**	0.4006	-0.0039	**-0.0210**
german	**71.54**	66.44	**71.02**	66.42	**0.4051**	0.5054	0.0073	**0.0003**
glass	66.07	**71.40**	64.46	**65.22**	0.5379	**0.4871**	**0.0244**	0.0866
hayes-roth	**81.67**	77.87	**82.87**	74.55	**0.2097**	0.3268	**-0.0147**	0.0426
heart	77.11	**78.52**	76.22	**77.93**	0.3084	**0.2811**	0.0115	**0.0075**
ionosphere	89.34	**91.91**	**87.52**	81.72	**0.1397**	0.1989	**0.0204**	0.1109
iris	**95.07**	94.53	**94.13**	87.20	**0.0617**	0.1354	**0.0099**	0.0775
lymphography	76.88	**80.82**	77.31	**80.96**	0.2951	**0.2356**	**-0.0056**	-0.0017
magic	85.10	**87.18**	84.69	**86.57**	0.1799	**0.1540**	**0.0048**	0.0070
monk-2	**100.00**	96.25	**100.00**	91.62	**0.0000**	0.0871	**0.0000**	0.0481
newthyroid	92.84	**95.81**	91.53	**92.65**	0.0912	**0.0767**	**0.0141**	0.0330
phoneme	85.88	**87.18**	84.76	**86.66**	0.1775	**0.1530**	0.0130	**0.0060**
pima	**73.99**	69.74	**73.15**	67.37	**0.3629**	0.4679	**0.0114**	0.0340
ring	90.06	**97.09**	88.72	**91.54**	0.1252	**0.0871**	**0.0149**	0.0572
segment	96.35	**97.28**	**95.19**	90.23	**0.0499**	0.1004	**0.0120**	0.0725
sonar	72.50	**86.63**	73.36	**85.94**	0.3674	**0.1623**	**-0.0119**	0.0080
spambase	92.57	**93.57**	**91.32**	91.20	**0.0938**	0.0940	**0.0135**	0.0253
twonorm	84.82	**97.35**	83.86	**96.31**	0.1903	**0.0379**	0.0113	**0.0107**
vehicle	71.04	**80.69**	68.91	**74.75**	0.4376	**0.3129**	**0.0300**	0.0736
vowel	78.59	**99.33**	74.91	**87.11**	0.3193	**0.1298**	**0.0468**	0.1230
wdbc	93.64	**94.41**	**92.79**	91.07	**0.0770**	0.0946	**0.0091**	0.0354
wine	92.23	**97.30**	89.75	**95.95**	0.1111	**0.0416**	0.0269	**0.0139**
yeast	55.54	**57.44**	53.34	**54.39**	0.8401	**0.7940**	**0.0396**	0.0531
zoo	**92.29**	80.64	**91.70**	73.08	**0.0899**	0.3338	**0.0064**	0.0938
average	81.28	**82.24**	**80.37**	78.50	**0.2777**	0.3117	**0.0115**	0.0457
best	12	**20**	**16**	**16**	**16**	**16**	**23**	9

Analysis of the global results. This part of the analysis compares the average results for both C4.5 and SVM across all the data sets, by using the performance and those average results of the ELA and RLA metrics, and the number of data sets where each classifier is the best.

Regarding the performance results, SVM has a better performance without noise than C4.5, obtaining an average performance of 82.24 versus a 81.28 of C4.5. Furthermore, SVM is also better in more data sets than C4.5, concretely in 20 of the 32 data sets. The situation reverses when noise is considered, and C4.5 obtains a better average performance (80.37 versus 78.50 of SVM) and the same number of data sets in which each classifier is the best (16 data sets in total). Note that these results without and with noise are consistent with the expected behavior of both classifiers.

Since the results of SVM for many of the data sets drop in an higher degree than those of C4.5, their average RLA value is therefore higher than that of C4.5 (0.0457 versus 0.0115). Thus, SVM is clearly less robust considering this metric. The average of the ELA measure also offers the same final result, showing to C4.5 as the method that globally behaves better with noisy data. These average results of both ELA and RLA are again in concordance with the expected behavior of the two implied classifiers, since we knew that C4.5 could probably have a better behavior with noise than SVM due to the punning mechanism.

However, as the accuracy of SVM is notably better than that of C4.5 without noise, the number of data sets in which each method is the best considering ELA is not so clear in favor of C4.5 like that of RLA. This fact is due to ELA considers the initial performance without noise, but it is not taken into account by RLA, that only considers the percentage variation of the performances with and without noise.

These global results highlights that ELA does not only use the loss of performance to evaluate the behavior with noisy data as RLA makes, but also the performance without noise, that must be considered to obtain a good evaluation metric of the behavior against noise as we have previously commented in Section 3.

Analysis of the individual results for each data set. Even though the aforementioned average results give an idea of how ELA and RLA work, it is interesting to observe their results in each single data set to better understand the differences and coincidences between both metrics depending on the behavior of the classifiers. In order to properly analyze these results, we categorize them into two different groups - a group devoted to those data sets in which the evaluation of ELA and RLA agrees and another for those data sets in which this evaluation is different:

1. **ELA and RLA predict the same classifier with the best behavior.** In this case, it is important to note that, even though both metrics provide the same final result, the difference in these values for each classifier could be very different depending on the

drop in performance with noise but also on the performance without noise (when ELA is considered). We can differentiate three scenarios within this case:

- *One of the classifiers is better than the other one with and without noise and both of them are deteriorated from the effect of noise.* There are some data sets (such as *autos* and *wine*) in which both methods have a remarkable and similar loss of performance, whereas in other data (such as *banana*, *dermatology*, *heart*, *phoneme* and *twonorm*) this loss of performance is very low. In all these data sets, SVM has always a better performance than C4.5. However, there are also other data sets, in which C4.5 has the best performance and its loss of accuracy is also lower than that of SVM, see for example *car*, *iris*, *pima*, *zoo* and *monk-2*. In these data sets the method with the best performance is chosen by ELA and RLA as that behaving best with noise.
- *One of the classifiers improves and the other deteriorates its accuracy without noise when noise is considered.* All these data sets are characterized by C4.5 being the method that behaves best with noisy data. For example, in the *cleveland* data set, both classifiers have a low accuracy without noise (being SVM worse without noise and deteriorating its accuracy, whereas C4.5 improves in presence of noise). The same situation occurs with *ecoli* and *hayes-roth*, even though the initial performance is higher than with *cleveland*.
- *One of the classifiers has a better performance without noise, but it suffers a very high drop in performance with noise and finally it has a worse performance in the noisy data set.* SVM is usually this classifier with a better performance without noise but it is more affected in the noisy version of the data set (see, for example, the *ionosphere*, *segment*, *spambase* and *wdbc* data sets).

2. **ELA and RLA predict a different classifier with the best behavior.** This case is perhaps more interesting than that of above since each one of the metrics give more importance to a different classifier with noisy data. Thus, we can clearly check the differences between ELA and RLA. We differentiate two scenarios within this case:
   - *One or both classifiers improve in presence of noise.* Note that all the data sets under these circumstances are characterized by ELA giving a higher importance to SVM (the method with the best performance without noise), whereas RLA highlights C4.5 (the method that usually has a lower performance without noise but having a higher improvement when noise is considered). For example, with *balance* and *sonar*, even though C4.5 slightly improves with noisy data whereas SVM has a higher drop

in performance, the latter is notably better than C4.5 without and with noise in terms of performance. With other data sets, such as *flare* and *lymphography*, both classifiers improve their results when noise is considered. In these cases, even though C4.5 experiments an higher improvement in performance than SVM (and thus RLA gives it more importance), the latter has remarkable better results without noise, and this fact is also considered by ELA.

- *Both classifiers deteriorate their performance when considering noise.* Some data sets (such as *vehicle*, *ring*, *newtiroid*, *vowel*. *magic* or *yeast*) are characterized by SVM being more deteriorated than C4.5 by the noise, but also having a higher performance without noise. Thus, RLA highlights C4.5 in this cases, whereas ELA also considered the importance of the higher performance of SVM estimating that it behaves better with noisy data. The opposite fact occurs with the *german* data set, where C4.5 and SVM interchange their behaviors. With the *glass* data set, although SVM is more affected than C4.5 by the presence of noise, it obtains a notable higher accuracy without noise. Thus, ELA establishes SVM as the method with the best behavior with noisy data since it valorizes more than RLA the initial accuracy (although the ELA value of SVM is very similar to that of C4.5), whereas RLA establishes C4.5 as the best method based on the higher drop in performance of SVM.

  Another case is that of the *contraceptive* data set, where the two classifiers obtain a very low accuracy without noise (SVM is indeed worse than C4.5, having a performance lower than 50%). This particular case shows that RLA favors those algorithms with lower classification performances without noise when the loss of performance of the classifiers are comparable (although not equal), whereas ELA does not harm so much the methods with higher performances.

Individual results for each data set again emphasize that RLA is only based on the percentage drop in performance, giving no chance, for example, to those methods that experiment an higher drop but having a very high performance without noise (being competitive enough when the noise is considered). In contrast, ELA takes into account both factors, making its usage overcoming some of the problems of the RLA measure.

6. Concluding Remarks

Performance and robustness are two independent concepts that imply different conclusions. Considering both concepts together seems to be crucial in order determine the expected behavior

of the classifiers against noise. Therefore, a new measure is proposed to know the expected behavior of a classifier with noisy data, the ELA measure, which tries to overcome some of the problems of the existing robustness-based metrics.

In order to check the suitability of our proposal, we have analyzed the existing robustness measures pointing out their main drawbacks and how ELA can solve them. We have provided a variety of practical examples supporting our analysis. In order to complete this analysis, we have experimentally compared the ELA and RLA measures, showing that the evaluation of the ELA and RLA metrics agree in some cases, but in other cases the behavior of the RLA is not so desirable since it is only based on the percentage variation of the performance without and with noise. The results obtained show that ELA is able to overcome some of the problems that RLA produces, being useful to represent the behavior of the classifiers against noise.

Acknowledgment

Supported by the Projects TIN2011-28488, P10-TIC-06858 and P11-TIC-7765. J. A. Sáez holds an FPU grant from the Spanish Ministry of Education and Science.

References

[1] J. Abellán, A. Masegosa, Bagging schemes on the presence of class noise in classification, Expert Systems with Applications 39 (8) (2012) 6827–6837.

[2] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2–3) (2011) 255–287.

[3] C. E. Brodley, M. A. Friedl, Identifying Mislabeled Training Data, Journal of Artificial Intelligence Research 11 (1999) 131–167.

[4] J. Cao, S. Kwong, R. Wang, A noise-detection based adaboost algorithm for mislabeled data, Pattern Recognition 45 (12) (2012) 4451–4465.

[5] W. W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann Publishers, 1995, pp. 115–123.

[6] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.

[7] P. J. Huber, Robust Statistics, John Wiley and Sons, New York, 1981.

[8] Y. Kharin, Robustness in statistical pattern recognition, Mathematics and Its Applications 380.

[9] Y. Kharin, E. Zhuk, Robustness in statistical pattern recognition under contaminations of training samples., in: Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 2 - Conference B: Computer Vision and Image Processing, 1994, pp. 504–506.

[10] H.-X. Li, J.-L. Yang, G. Zhang, B. Fan, Probabilistic support vector machines for classification of noise affected data, Information Sciences 221 (2013) 60–71.

[11] D. Nettleton, A. Orriols-Puig, A. Fornells, A Study of the Effect of Different Types of Noise on the Precision of Supervised Learning Techniques, Artificial Intelligence Review 33 (2010) 275–306.

[12] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106.

[13] J. R. Quinlan, C4.5: programs for machine learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.

[14] J. A. Sáez, J. Luengo, F. Herrera, Fuzzy rule based classification systems versus crisp robust learners trained in presence of class noise's effects: a case of study, in: 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), 2011, pp. 1229–1234.

[15] T. Santhanam, S. Radhika, Probabilistic neural network-a better solution for noise classification, Journal of Theoretical and Applied Information Technology 27 (1) (2011) 39–42.

[16] C.-M. Teng, Correcting Noisy Data, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999, pp. 239–248.

[17] C. M. Teng, Polishing blemishes: Issues in data correction, IEEE Intelligent Systems 19 (2004) 34–39.

[18] R. Y. Wang, V. C. Storey, C. P. Firth, A Framework for Analysis of Data Quality Research, IEEE Transactions on Knowledge and Data Engineering 7 (4) (1995) 623–640.

[19] X. Wu, Knowledge acquisition from databases, Ablex Publishing Corp., Norwood, NJ, USA, 1996.

[20] X. Wu, X. Zhu, Mining With Noise Knowledge: Error-Aware Data Mining, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 38 (4) (2008) 917–932.

[21] S. Zhong, T. M. Khoshgoftaar, N. Seliya, Analyzing Software Measurement Data with Clustering Techniques, IEEE Intelligent Systems 19 (2) (2004) 20–27.

[22] X. Zhu, X. Wu, Class Noise vs. Attribute Noise: A Quantitative Study, Artificial Intelligence Review 22 (2004) 177–210.

Bibliography

[AFFL+11] Alcalá-Fdez J., Fernández A., Luengo J., Derrac J., García S., Sánchez L., and Herrera F. (2011) Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing* 17(2-3).

[Aha97] Aha D. W. (Ed.) (1997) *Lazy Learning.* Springer.

[AIS93] Agrawal R., Imieliński T., and Swami A. (1993) Mining association rules between sets of items in large databases. *SIGMOD Rec.* 22(2): 207–216.

[AKA91] Aha D. W., Kibler D., and Albert M. K. (1991) Instance-based learning algorithms. *Machine Learning* 6: 37–66.

[Alp10] Alpaydin E. (2010) *Introduction to Machine Learning.* The MIT Press, 2nd edition.

[AMMR95] Anand R., Mehrotra K., Mohan C. K., and Ranka S. (1995) Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks* 6(1): 117–124.

[BBBE11] Bizer C., Boncz P. A., Brodie M. L., and Erling O. (2011) The meaningful use of big data: four perspectives - four challenges. *SIGMOD Record* 40(4): 56–60.

[BF99] Brodley C. and Friedl M. (1999) Identifying mislabeled training data. *Journal of Artificial Intelligence Research* 11: 131–167.

[BH06] Basu M. and Ho T. (2006) *Data Complexity in Pattern Recognition.* Springer, Berlin.

[BHS09] Bell G., Hey T., and Szalay A. (2009) Beyond the data deluge. *Science* 323: 1297–1298.

[BM03] Batista G. E. A. P. A. and Monard M. C. (2003) An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence* 17(5–6): 519–533.

[Bra07] Bramer M. (2007) *Principles of Data Mining (Undergraduate Topics in Computer Science).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[CBHK02] Chawla N. V., Bowyer K. W., Hall L. O., and Kegelmeyer W. P. (2002) SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16: 321–357.

[CGG+09] Chen Y., Garcia E. K., Gupta M. R., Rahimi A., and Cazzanti L. (2009) Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research* 10: 747–776.

[CH67] Cover T. and Hart P. (1967) Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13: 21–27.

[CM98] Cherkassky V. S. and Mulier F. (1998) *Learning from Data: Concepts, Theory, and Methods*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition.

[Coh95] Cohen W. W. (1995) Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 115–123. Morgan Kaufmann Publishers.

[CV95] Cortes C. and Vapnik V. (1995) Support vector networks. *Machine Learning* 20: 273–297.

[Dem06] Demšar J. (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7: 1–30.

[DHS00] Duda R., Hart P., and Stork D. (2000) *Pattern Classification*. Wiley-Interscience, John Wiley & Sons, Southern Gate, Chichester, West Sussex, England, 2nd edition.

[DTGH12] Derrac J., Triguero I., García S., and Herrera F. (2012) Integrating instance selection, instance weighting and feature weighting for nearest neighbor classifiers by coevolutionary algorithms. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* 42(5): 1383–1397.

[FI08] Fernández F. and Isasi P. (2008) Local feature weighting in nearest prototype classification. *IEEE Transactions on Neural Networks* 19(1): 40–53.

[FKD08] Farhangfar A., Kurgan L., and Dy J. (2008) Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition* 41(12): 3692–3705.

[Fri97] Friedman J. H. (1997) Data mining and statistics: What's the connection? In *Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics*.

[Fur02] Furnkranz J. (2002) Round Robin Classification.

[GBLG99] Gamberger D., Boskovic R., Lavrac N., and Groselj C. (1999) Experiments With Noise Filtering in a Medical Domain. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 143–151. Morgan Kaufmann Publishers.

[GDCH12] García S., Derrac J., Cano J. R., and Herrera F. (2012) Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(3): 417–435.

[GFB+11] Galar M., Fernández A., Barrenechea E., Bustince H., and Herrera F. (2011) An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition* 44: 1761–1776.

[GFLH10] García S., Fernández A., Luengo J., and Herrera F. (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences* 180: 2044–2064.

[GLD00] Gamberger D., Lavrac N., and Dzeroski S. (2000) Noise Detection and Elimination in Data Preprocessing: experiments in medical domains. *Applied Artificial Intelligence* 14: 205–223.

[Har75] Hartigan J. A. (1975) *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition.

[HH09] Hüehn J. C. and Hüellermeier E. (2009) FURIA: An Algorithm for Unordered Fuzzy Rule Induction. *Data Mining and Knowledge Discovery* 19(3): 293–319.

[HHS94] Ho T. K., Hull J. J., and Srihari S. N. (1994) Decision Combination in Multiple Classifier Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(1): 66–75.

[HS95] Huang Y. S. and Suen C. Y. (1995) A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17: 90–93.

[Hub81] Huber P. J. (1981) *Robust Statistics*. John Wiley and Sons, New York.

[HYKL06] Huang K., Yang H., King I., and Lyu M. R. (2006) Imbalanced learning with a biased minimax probability machine. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 36(4): 913–923.

[INN04] Ishibuchi H., Nakashima T., and Nii M. (2004) *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining (Advanced Information Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[JL13] Juhola M. and Laurikkala J. (2013) Missing values: How many can they be to preserve classification reliability? *Artificial Intelligence Review* 40(3): 231–245.

[KB81] Koplowitz J. and Brown T. A. (1981) On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition* 13(3): 251–255.

[Ker14] Kersten J. (2014) Simultaneous feature selection and Gaussian mixture model estimation for supervised classification problems. *Pattern Recognition* 47(8): 2582–2595.

[Kha96] Kharin Y. (1996) Robustness in statistical pattern recognition. *Mathematics and Its Applications* 380.

[KHDM98] Kittler J., Hatef M., Duin R., and Matas J. (1998) On combining classifiers. *ieee transactions on pattern analysis and machine intelligence* 20(3): 226–239.

[KK07] Kononenko I. and Kukar M. (2007) Machine Learning and Data Mining: Introduction to Principles and Algorithms. Horwood Publishing Limited.

[Kon94] Kononenko I. (1994) Estimating attributes: Analysis and extensions of RELIEF. In *Proceedings of the 1994 European Conference on Machine Learning, Catania, Italy*, pp. 171–182. Springer Verlag.

[KPD90] Knerr S., Personnaz L., and Dreyfus G. (1990) Single-Layer Learning Revisited: A Stepwise Procedure for Building and Training a Neural Network. In Fogelman Soulié F. and Hérault J. (Eds.) *Neurocomputing: Algorithms, Architectures and Applications*, pp. 41–50. Springer-Verlag.

[KR92] Kira K. and Rendell L. A. (1992) A practical approach to feature selection. In *Proceedings of the Ninth International Conference on Machine Learning, Aberdeen, Scotland*, pp. 249–256. Morgan Kaufmann.

[KR07] Khoshgoftaar T. M. and Rebours P. (2007) Improving software quality prediction by noise filtering techniques. *Journal of Computer Science and Technology* 22: 387–396.

[Kun04] Kuncheva L. (2004) *Combining Pattern Classifiers: Methods and Algorithms*. Wiley.

[KZ94] Kharin Y. and Zhuk E. (1994) Robustness in statistical pattern recognition under contaminations of training samples. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 2 - Conference B: Computer Vision and Image Processing*, pp. 504–506.

[LdCG08] Lorena A., de Carvalho A., and Gama J. (2008) A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review* 30: 19–37.

[LGH12] Luengo J., García S., and Herrera F. (2012) On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and Information Systems* 32(1): 77–108.

[LM07] Liu H. and Motoda H. (Eds.) (2007) *Computational Methods of Feature Selection*. Chapman & Hall/Crc Data Mining and Knowledge Discovery Series. Chapman & Hall/Crc.

[LQDZ13] Liu D., Qian H., Dai G., and Zhang Z. (2013) An iterative SVM approach to feature selection and classification in high-dimensional datasets. *Pattern Recognition* 46(9): 2531–2537.

[McL04] McLachlan G. J. (2004) Discriminant Analysis and Statistical Pattern Recognition, John Wiley and Sons.

[MKK87] Mazurov V. D., Krivonogov A. I., and Kazantsev V. S. (1987) Solving of optimization and identification problems by the committee methods. *Pattern Recognition* 20: 371–378.

[MM96] Mayoraz E. and Moreira M. (1996) On the decomposition of polychotomies into dichotomies.

[PV06] Paredes R. and Vidal E. (2006) Learning weighted metrics to minimize nearest-neighbor classification error. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(7): 1100–1110.

[Pyl99] Pyle D. (1999) *Data preparation for data mining*. Morgan Kaufmann, Los Altos.

[Qui93] Quinlan J. R. (1993) *C4.5: programs for machine learning*. Morgan Kaufmann Publishers, San Francisco, CA, USA.

[RK04] Rifkin R. and Klautau A. (2004) In defense of one-vs-all classification. *Journal of Machine Learning Research* 5: 101–141.

[SDLH14a] Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Statistical computation of feature weighting schemes through data estimation for nearest neighbor classifiers. *Pattern Recognition* 47(12): 3941–3948.

[SDLH14b] Sáez J. A., Derrac J., Luengo J., and Herrera F. (2014) Improving the behavior of the nearest neighbor classifier against noisy data with feature weighting schemes. In *Hybrid Artificial Intelligence Systems*, volumen 8480 of *Lecture Notes in Computer Science*, pp. 597–606. Springer International Publishing.

[SG84] Shapley L. and Grofman B. (1984) Optimizing group judgmental accuracy in the presence of interdependencies. *Public Choice* 43: 329–343.

[SGLH13] Sáez J. A., Galar M., Luengo J., and Herrera F. (2013) Tackling the Problem of Classification with Noisy Data using Multiple Classifier Systems: Analysis of the Performance and Robustness. *Information Sciences* 247: 1–20.

[SGLH14a] Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems* 38(1): 179–206.

[SGLH14b] Sáez J. A., Galar M., Luengo J., and Herrera F. (2014) INFFC: An iterative noise filter based on the fusion of classifiers with noise sensitiveness control (submitted).

[SK03] Sikonja M. R. and Kononenko I. (2003) Theoretical and empirical analysis of Relief and ReliefF. *Machine Learning* 53(1–2): 23–69.

[SLH11] Sáez J. A., Luengo J., and Herrera F. (2011) Fuzzy Rule Based Classification Systems versus Crisp Robust Learners Trained in Presence of Class Noise's Effects: a Case of Study. In *11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), Córdoba (Spain)*, pp. 1229–1234.

[SLH13] Sáez J. A., Luengo J., and Herrera F. (2013) Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition* 46(1): 355–364.

[SLH14] Sáez J. A., Luengo J., and Herrera F. (2014) Evaluating the classifier behavior with noisy data considering performance and robustness: the Equalized Loss of Accuracy measure (submitted).

[SLSH14a] Sáez J. A., Luengo J., Shim S., and Herrera F. (2014) Class Noise Reparation by an Aggregated Noise Filter Ensemble Voting Algorithm (submitted).

[SLSH14b] Sáez J. A., Luengo J., Stefanowski J., and Herrera F. (2014) SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a resampling method with filtering, doi: 10.1016/j.ins.2014.08.051. *Information Sciences* (in press).

[Smi39] Smirnov N. V. (1939) Estimate of deviation between empirical distribution functions in two independent samples (in russian). *Bulletin of Moscow University* 2: 3–16.

[Sun07] Sun Y. (2007) Iterative relief for feature weighting: Algorithms, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(6): 1035–1051.

[Ten99] Teng C.-M. (1999) Correcting Noisy Data. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 239–248. Morgan Kaufmann Publishers, San Francisco, CA, USA.

[Ten04] Teng C. M. (2004) Polishing blemishes: Issues in data correction. *IEEE Intelligent Systems* 19: 34–39.

[TK07] Tsoumakas G. and Katakis I. (2007) Multi-label classification: An overview. *Internation Journal of Data Warehousing and Mining* 2007: 1–13.

[TMM+81] Titterington D. M., Murray G. D., Murray L. S., Spiegelhalter D. J., Skene A. M., Habbema J. D. F., and Gelpke G. J. (1981) Comparison of discriminant techniques applied to a complex data set of head injured patients. *Journal of the Royal Statistical Society, Series A (General)* 144: 145–175.

[Tom76] Tomek I. (1976) An Experiment With The Edited Nearest-Neighbor Rule. *IEEE Transactions on Systems and Man and Cybernetics* 6(6): 448–452.

[TSK05] Tan P., Steinbach M., and Kumar V. (2005) *Introduction to Data Mining.* Addison-Wesley Longman Publishing Co., Inc., 1st edition.

[Vap98] Vapnik V. (1998) *Statistical Learning Theory.* Wiley, New York, USA.

[WAM97] Wettschereck D., Aha D. W., and Mohri T. (1997) A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review* 11: 273–314.

[Wil72] Wilson D. (1972) Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems and Man and Cybernetics* 2(3): 408–421.

[WKRQ+07] Wu X., Kumar V., Ross Quinlan J., Ghosh J., Yang Q., Motoda H., McLachlan G. J., Ng A., Liu B., Yu P. S., Zhou Z.-H., Steinbach M., Hand D. J., and Steinberg D. (2007) Top 10 algorithms in data mining. *Knowledge and Information Systems* 14(1): 1–37.

[WSF95] Wang R. Y., Storey V. C., and Firth C. P. (1995) A Framework for Analysis of Data Quality Research. *IEEE Transactions on Knowledge and Data Engineering* 7(4): 623–640.

[Wu96] Wu X. (1996) *Knowledge acquisition from databases.* Ablex Publishing Corp., Norwood, NJ, USA.

[WZ08] Wu X. and Zhu X. (2008) Mining With Noise Knowledge: Error-Aware Data Mining. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 38(4): 917–932.

[XZB14] Xue B., Zhang M., and Browne W. (2014) Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Applied Soft Computing Journal* 18: 261–276.

[YW06] Yang Q. and Wu X. (2006) 10 challenging problems in data mining research. *International Journal of Information Technology and Decision Making* 5(4): 597–604.

[ZKS04] Zhong S., Khoshgoftaar T. M., and Seliya N. (2004) Analyzing Software Measurement Data with Clustering Techniques. *IEEE Intelligent Systems* 19(2): 20–27.

[ZW04] Zhu X. and Wu X. (2004) Class Noise vs. Attribute Noise: A Quantitative Study. *Artificial Intelligence Review* 22: 177–210.

[ZWC03] Zhu X., Wu X., and Chen Q. (2003) Eliminating class noise in large datasets. In *Proceeding of the Twentieth International Conference on Machine Learning*, pp. 920–927.

[ZWY04] Zhu X., Wu X., and Yang Y. (2004) Error detection and impact-sensitive instance ranking in noisy datasets. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 378–383. AAAI Press.

[ZYY+14] Zhu F., Ye N., Yu W., Xu S., and Li G. (2014) Boundary detection and sample reduction for one-class support vector machines. *Neurocomputing* 123: 166 – 173.