University of Granada

Department of Computer Science
and Artificial Intelligence

PhD Program in Information and
Communication Technologies

PhD THESIS DISSERTATION

PREPROCESSING AND ENSEMBLE APPROACHES FOR SINGULAR PROBLEMS: MONOTONIC AND IMBALANCED CLASSIFICATION

PhD STUDENT

Sergio González Vázquez

PhD ADVISORS

Salvador García

Francisco Herrera

Granada, June 2020

Editor: Universidad de Granada. Tesis Doctorales
Autor: Sergio González Vázquez
ISBN: 978-84-1306-569-4
URI: http://hdl.handle.net/10481/63381

El doctorando / *The doctoral candidate* **Sergio González Vázquez** y los directores de la tesis / *and the thesis supervisors*: **Salvador García López** and **Francisco Herrera Triguero**

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.

Guarantee, by signing this doctoral thesis, that the work has been done by the doctoral candidate under the direction of the thesis supervisors and, as far as our knowledge reaches, in the performance of the work, the rights of the other authors to be cited (when their results or publications have been used) have been respected.

Granada, June 2020

The PhD student:	The PhD advisor:	The PhD advisor:
Sgd.: Sergio González	Sgd.: Salvador García	Sgd.: Francisco Herrera

This PhD thesis is supported by the Spanish National Research Project TIN2017-89517-P and by a FPU scholarship from the Spanish Ministry of Education FPU14/03079 holded by the PhD Student, S. González.

Cuando llegue la inspiración,
que me pille trabajando.

Pablo Picasso

Let everything happen to you:
beauty and terror.
Just keep going.
No feeling is final.

Rainer Maria Rilke

Agradecimientos

La culminación de esta tesis doctoral no hubiera sido posible sin la ayuda de mis directores, amigos y familia. Por eso, me gustaría dedicarles los siguientes agradecimientos.

He de comenzar agradeciendo a mis directores, Salva y Paco, por su trabajo y tiempo dedicado tanto a mi formación como a mi investigación durante estos años. Sin su supervisión, esta tesis no sería la misma. Me gustaría hacer una especial mención a Salva por su apoyo más allá de lo laboral.

También agradezco a mis amigos y compañeros del trabajo, con los cuales he podido compartir las dudas, frustraciones y alegrías de este periodo como doctorando. Además, han estado siempre ahí para poder desconectar y pasar buenos ratos fuera y dentro del CITIC.

Quiero extender dichos agradecimientos al grupo de Repsol. Con ellos, he experimentado otra manera de trabajar más colaborativa en ciencia de datos. Durante esos meses, me sentí bastante unido a los estudiantes del grupo Visbreaking. Aunque no todo fue ideal, el proyecto de Repsol me ha servido varias veces de motivación cuando el trabajo de investigación no me llenaba ni progresaba como deseaba. De este grupo, Juan Antonio ha sido la persona con la que más horas he pasado. Me alegro de haberlo conocido, trabajado con él, y supervisado sus primeros pasos en la investigación académica. Sin duda alguna, trabajar con él es una de las cosas que más echo de menos del proyecto de Repsol.

De mi entorno universitario, quiero agradecer especialmente a Jesús Maillo. Juntos, hemos compartido 10 años de estudios universitarios entre el grado, máster y doctorado. Hemos vivido multitud de experiencias juntos, ayudándonos mutuamente tanto en lo académico como en lo personal. Parece que, con la finalización de nuestras tesis, nuestros caminos se separan. Pero, sé que estaremos siempre ahí el uno para el otro. Le deseo lo mejor en esta nueva etapa fuera de la universidad.

Ajeno al mundo académico, agradezco a Ana Lucía y Beth. Con ellas, me siento más libre para expresar mis sentimientos y estado anímico. Quiero también dedicar un profundo agradecimiento a mis amigos de la infancia. Aunque últimamente nos veamos una vez al año en nuestras míticas cenas de navidad, siempre los siento muy cercanos y les tengo mucho cariño. En particular, quiero dedicar unas palabras a Juan Carlos, el cual es mi amigo desde los 6 años. Aún viviendo en sitios diferentes desde hace 10 años, no hemos perdido el contacto. Él siempre ha estado ahí para cualquier cosa que necesitara. También deseo agradecer a mis amistades a miles de kilómetros de distancia, que me han apoyado como los que más durante estos años.

Por último, doy mis gracias a mi familia, mi padre, mi hermano Ismael, y especialmente a mi madre. Ella me ha enseñado el esfuerzo y la exigencia hacia en trabajo, entre otros tantísimos valores. Tampoco puedo olvidar a mi hermano pequeño Gonzalo. Aunque con inevitables malas épocas, espero que crezca feliz y siendo una gran persona, como sé que es. También quiero agradecer al resto de mi familia que también me ha animado en todo momento.

Table of Contents

Chapter I

PhD dissertation

1 Introduction

The technological advances of recent years have enabled massive data generation and storage by companies, governments, and research institutes. As a result, there has been an increase in the interest of these organizations in extracting valuable knowledge from such data. Such knowledge can lead to new advances and a relevant competitive advantage. Therefore, data science has become the flagship of research, development, and innovation.

The demanding standards of data science, as well as the variety of new applications with different restrictions, have grown steadily. However, control over the quality of data has been neglected given its rapid and massive generation. This leads to greater inconsistency in such data.

In this context, the correct application of the Knowledge Discovery in Databases (KDD) process [1] has become more important. The KDD process is defined as the set of stages that make it possible to identify valuable patterns and relationships in the data [1, 2]. The stages of the KDD process are as follows:

- Problem specification: it is responsible for identifying the problem requirements and the objective of the discovery process.
- Data extraction: it selects data from the most relevant information sources for the problem, usually with the help of expert knowledge. The extracted data are grouped into a single data-set to be processed in subsequent stages.
- Data preprocessing: it intends to transform the data to be used by data mining techniques and to clean all possible impurities present in the data, such as noise, lack of information or redundant, and irrelevant data [3]. The final objective of data preprocessing is to obtain quality data, currently known as Smart Data [4, 5], for its use in subsequent stages.
- Data mining: it aims to extract patterns, relationships, and/or models from the processed data-sets [6]. The kind of knowledge to be extracted determines the category of the data mining problem and the group of feasible techniques. The selection of the best technique for each problem is a complex engineering process that requires the optimization and validation of the different techniques available.
- Interpretation and evaluation: the extracted knowledge is analyzed and described to be understandable and useful.

All these steps of the KDD process are essential. However, these do not usually require the same dedication and time. Some practitioners advocate the need for greater dedication to the data preprocessing stage due to its importance [3, 5]. Data mining algorithms usually assume the veracity and correctness of the data. For this reason, the inference of models or patterns with impure data can lead to misleading or erroneous results. Therefore, data quality must be a priority in the KDD process.

Data preprocessing [3] covers the tasks of data preparation and integration [7, 8], noise processing [9], missing value imputation [10, 11] and data reduction tasks such as feature selection [12], instance selection [13] and discretization [14]. The correct application of these tasks on the raw data will produce Smart Data [5], as is known in the literature to clean and useful quality data for data mining techniques.

Data mining [2, 6] is another core task of the KDD process, as it is responsible for inferring the patterns, relationships, or trends hidden in the data. Traditionally, it is divided into two areas according to the type of target knowledge to be learned:

- Supervised learning: it infers relationships/models between a target variable and input variables from known data to predict the value of the target variable for future unseen cases. Depending on the domain of the target variable, two categories are differentiated:
  - Classification [15]: the domain of the target variable is discrete with all its possible known classes or labels. The prediction of whether a patient is sick or healthy is an example of classification in the area of medicine.
  - Regression [16]: the values of the goal variable are continuous. An example of a regression problem is the estimation of the price of a product.
- Unsupervised learning: it aims to discover different relationships between instances or variables in the data set without a defined target variable. Two different families are distinguished:
  - Clustering [17]: it identifies groups of samples according to a measure of similarity. Clustering attempts to minimize differences between instances of the same group and to maximize distances between different groups.
  - Association [18]: it detects relevant relationships between different data attributes.

Most supervised learning research focuses on binary or multi-class classification and regression problems with a single target variable. However, there is a multitude of less known problems. In the literature, these are referred to as singular or nonstandard supervised learning problems [19].

Singular supervised learning problems are those that do not follow the usual scheme of supervised problems [20, 19]. Thus, standard data mining algorithms do not work correctly or directly, they cannot be applied to these problems.

According to the differences with classical supervised learning, singular problems can be categorized as follows:

- Problems with nonstandard structure: they do not obey the traditional structure of an input vector with a single target value. That is, these problems have multiple input vectors for each output or multiple target variables. Multi-instance learning [21] and multi-view learning [22] are examples of problems with multiple inputs. Multi-label learning [23] and multi-target regression [24] are examples of data with multiple outputs.

- Problems with partial or imprecise information: they have incorrect information or lack values in features, instances, or target variable. A well-known example is semi-supervised learning [25], where part of the instances lacks target variable values. Imbalanced classification [26] may be considered a problem with partial information due to the lack of representation in the minority classes. Classification with noisy data [9] has also been considered in this category [20]. Other more specific examples are one-class learning [27], zero-shot learning [28] or one-shot learning [29].
- Problems with prior knowledge constraints: they present relationships or restrictions coming from the expert knowledge of the applied problem. These constraints must be taken into consideration during the learning process. Ordinal regression [30] includes order relations between class labels. Monotonic classification [31] incorporates order constraints between the attributes and the class.

Singular supervised problems are mostly treated from two main approaches [19]: problem transformation and algorithm adaptation. The first approach makes use of preprocessing techniques to transform the original problem into one or more standard problems with less complexity. These simpler problems are solved with the usual data mining techniques. Their solutions are combined into a single solution for the original problem. The second group of solutions seeks to design new algorithms or adaptations of existing methods that consider the peculiarities of the data of the singular problem.

It should be noted that the singular problems are not mutually exclusive. It is common to find several singular scenarios in the same real application. The combination of these tends to significantly complicate the learning process. For example, the presence of class noise in imbalanced classification or monotonic classification may aggravate the accuracy of the less represented classes [32] or break the order constraints [33], respectively. Another example is the combination of multi-instance and multi-label scenarios [34]. However, few proposals study two or more singular problems at the same time.

This thesis focuses on two singular supervised problems: imbalanced classification [26] and classification with monotonic constraints [31].

Class imbalance problem [26] alludes to a significant difference in the number of representative cases of each class. That is, certain classes, commonly called minority classes, have many fewer instances than others, known as majority classes. Standard classifiers tend to lower the accuracy of minority classes due to such disparity in the number of examples. This situation is quite common in real problems, such as bank credit rating [35, 36] or business domain problems [37]. In these real-life scenarios, the misclassification of a minority class example usually entails a higher cost. Therefore, addressing this problem is extremely essential.

Therefore, new techniques have been designed to address the class imbalance problem with three different approaches [26]: algorithmic-based proposals [38, 39, 40], cost-sensitive learning [41, 42, 37], and data sampling [43]. Ensembles have been successfully combined with these previous methods obtaining a great performance in this problem [44].

Monotonic classification or classification with monotonic constraints is a singular classification problem, where there is an order relationship between the ordinal class variable and some ordinal or numerical input attributes [31]. Such monotonic constraints restrict the increase or decrease of the class label to the increase or decrease of the attributes. That is, the prediction of the class label should not decrease in the presence of higher input values while the rest remains the same.

Standard classification algorithms do not consider such constraints on the learning process. Thus, their predictions tend to violate monotonicity. Therefore, a multitude of monotonic classifiers have been developed based on decision trees [45, 46, 47], fuzzy classifiers [48, 49], neural networks

[50, 51], instance-based learning [52, 53, 54] and ensemble learning [55, 56]. However, many of these suffer from two common problems of monotonic classification: i) Some algorithms require purely monotonic data, which rarely occurs due to data inconsistencies, and ii) Some models are highly biased towards monotonic constraints and have very poor performance in terms of accuracy.

These constraints of prior knowledge are often required in real evaluation problems, such as credit risk modeling [57], housing pricing [58], and professor evaluation [59]. These are also common scenarios of imbalanced classification because those most valued classes tend to be less represented.

This thesis aims to propose new solutions based on robust classifiers and preprocessing techniques for these two singular supervised problems. Figure 1 summarizes all the different problems and techniques treated in the objectives of this thesis.

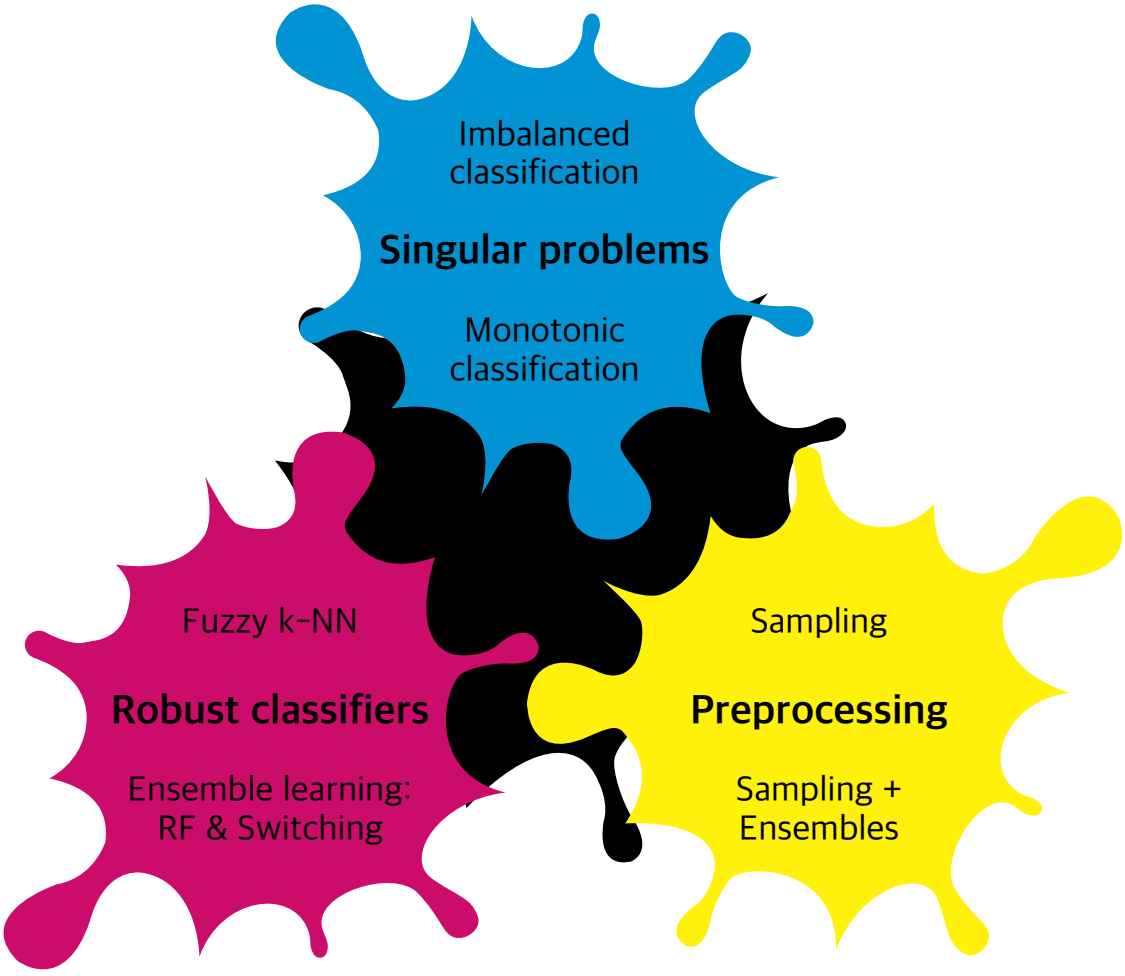


Figure 1: Objectives of the PhD thesis.

Ensembles are one of the most robust and successful models of data mining [60, 61]. They combine several base classifiers so that their merged prediction significantly improves the performance of a single model [62, 63]. This combination allows the correction of possible errors made by the base learners and promotes a great generalization without neglecting local knowledge [62, 64]. Incorporating diversity into the base learners of the ensemble is critical to achieving these precision and generalization benefits. A multitude of diversity promotion techniques are available, such as algorithmic process randomization [65, 66] or sampling (bootstrapping) [67, 68].

Random Forest [65] is one of the most accurate and popular algorithms [61, 69]. This decision tree ensemble [70] combines bootstrapping and the incorporation of randomness in the selection of the best tree cut to reach high levels of accuracy. Given its success, Random Forest with monotonic decision trees [45, 47] could be an ideal proposal to achieve the previously explained challenges of monotonic classification: high precision and monotonicity without purely monotonic data. This proposal is studied and developed later in this thesis.

The manipulation of class labels is another very interesting and little explored approach to promoting diversity in ensembles. Label switching algorithms [71, 72] modify the class values of randomly selected instances as a diversity promotion technique. Class Switching algorithm [72] tends to balance the percentages of samples of each class in slightly imbalanced scenarios. Therefore, a redesign of this model in combination with sampling techniques [44] is very interesting to mitigate the impact of class imbalance. In this thesis, an ensemble based on Class Switching according to the distance to the nearest enemy is proposed for the highly imbalanced classification.

Sampling techniques are preprocessing methods for transforming a class imbalance problem into a standard problem [3, 26]. These techniques are usually divided into under-sampling and over-sampling. Under-sampling reduces the number of samples from the majority classes to balance the data-set. In contrast, over-sampling increases the number of representatives of the minority classes. Both techniques have shown great results both independently [43] and in combination with ensembles [44]. However, their application in monotonic classification problems with class imbalance could deteriorate the monotonicity of the data and thus, affect the performance of the classifiers. This event is studied during the thesis, giving rise to sampling techniques adapted to imbalanced monotonic classification.

Algorithms based on k Nearest Neighbors (kNN) can be a great asset to both preprocessing techniques and robust classifiers. The kNN algorithm classifies based on the labels of the instances most similar to the given example. The use of kNN goes beyond classification. It has been used in many preprocessing techniques of all kinds, such as instance selection [73, 59], prototype generation [74], border detection [75, 76], sampling for the class imbalance problem [77, 26], noise filters [78], among others.

In this thesis, kNN is used transversely in multiple proposals. The distance to the nearest enemy guides the class change of border instances in the Switching-based ensemble. Some sampling techniques proposed for imbalanced monotonic classification are based on neighborhoods.

In addition, a new noise-robust classifier based on Fuzzy kNN with monotonic constraints is proposed. In the standard classification, Fuzzy kNN [79] has proven high performance and robustness to class noise [80], thanks to the prior extraction of class membership for crisp training samples. However, this mechanism does not take into account restrictions or monotonicity violations. Monotonic Fuzzy kNN proposal stands out for its high performance even in the presence of monotonic inconsistencies in the data.

In essence, the main objective of this thesis is to design new solutions for these problems, both independently and together, and considering other singular data situations, such as the presence of class noise. As previously mentioned, these proposals follow two different approximations similar to the traditional approaches for singular problems: robust classifiers and preprocessing based techniques. These approaches aim at specific issues of imbalanced and monotonic classification:

- Design of robust classifiers for the singular problems of imbalanced and monotonic classification. This objective includes:
  - High levels of accuracy as well as monotonicity with a Random Forest classifier for monotonic classification.

  - Robust ensemble learning based on Switching according to the Nearest Enemy Distance (SwitchingNED) for highly imbalanced problems.
  - Great robustness to monotonic noise for monotonic classification with a Fuzzy kNN proposal aware to monotonic constraints and violations.

- Development of preprocessing-based techniques in imbalanced and monotonic classification. Within this paradigm, the following goals are enclosed:
  - To enhance the performance of SwitchingNED through its combination with different sampling techniques for imbalanced classification.
  - To mitigate the impact of skewed class distributions in problems with monotonic constraints thanks to new sampling techniques for monotonic imbalanced classification.

Finally, the rest of the thesis is organized into two main parts: the PhD thesis and the associated publications. In this first part, Section 2 develops the preliminary knowledge and background necessary for this dissertation. In Section 3, the relevance of the thesis and its associated scientific works are justified in the context of the outlined problems. Section 4 describes the objectives established for the development of the thesis. Section 5 presents the methodology followed in the course of this work. In Section 6 and Section 7, the scientific studies of this PhD thesis and its results are summarized, respectively. Finally, Section 8 highlights the main conclusions drawn and the future lines of this work.

The second part of the thesis document includes the 4 publications associated with the development of this dissertation. The scientific papers are ordered according to their publication date:

- Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity.
- Class switching according to nearest enemy distance for learning from highly imbalanced data-sets.
- Chain based sampling for monotonic imbalanced classification.
- Fuzzy k-Nearest Neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise.

Introducción

Los avances tecnológicos de estos últimos años han permitido la masiva generación y almacenamiento de datos por parte de empresas, gobiernos y centros de investigación. Esto ha promovido el aumento del interés de estos organismos por extraer conocimiento valioso de dichos datos. Dicho conocimiento puede suponer nuevos avances y una relevante ventaja competitiva. Por tanto, la ciencia de datos se ha vuelto el buque insignia de la investigación, desarrollo e innovación.

Los estándares de exigencia sobre la ciencia de datos, así como la variedad de nuevas aplicaciones con diferentes restricciones, han crecido paulatinamente. Sin embargo, el control sobre la calidad de los datos ha podido descuidarse dada su rápida y masiva generación. Esto conlleva una mayor inconsistencia en dichos datos.

En este contexto, la correcta aplicación del proceso de descubrimiento de conocimiento en bases de datos (Knowledge Discovery in Databases - KDD) [1] ha cobrado una mayor importancia. El proceso KDD se define como el conjunto de etapas que posibilitan la identificación de valiosos patrones y relaciones en los datos [1, 2]. Las etapas del proceso KDD son las siguientes:

- Especificación del problema: es responsable de identificar los requisitos del problema y el objetivo del proceso de descubrimiento.
- Extracción de datos: selecciona los datos de las fuentes de información más relevantes para el problema, habitualmente con ayuda de conocimiento experto. Los datos extraídos se agrupan en un solo conjunto para ser procesado en siguientes etapas.
- Preprocesamiento de datos: pretende transformar los datos para poder ser utilizados por las técnicas de minería de datos y limpiar todas las posibles impurezas presentes en los mismo, tales como ruido, falta de información o datos redundantes e irrelevantes [3]. El objetivo final del preprocesamiento de datos es la obtención de datos de calidad, o actualmente conocidos como Smart Data [81, 5], para su uso en siguientes etapas.
- Minería de datos: tiene como objetivo la extracción de patrones, relaciones y/o modelos de un conjunto procesado de datos [6]. El tipo de conocimiento a extraer determina la categoría del problema de minería de datos y el grupo de técnicas viables para dicha categoría. La selección de la mejor técnica para cada problema es un proceso complejo de ingeniería que requiere de la optimización y validación de las diferentes técnicas disponibles.
- Interpretación e evaluación: el conocimiento extraído se analiza y se describe para ser comprensible y util.

Todos estos pasos del proceso KDD son esenciales. Sin embargo, habitualmente no todos requieren la misma dedicación ni tiempo. Algunos profesionales del sector defienden la necesidad de una mayor dedicación a la etapa de preprocesamiento de datos debido su importancia [3, 5]. Los algoritmos de minería de datos suelen asumir la veracidad y corrección de los datos. Debido a esto, la inferencia de modelos o patrones con datos impuros puede dar lugar a resultados engañosos o erróneos. Por tanto, la calidad de los datos debe ser prioritaria en el proceso KDD.

El preprocesamiento de datos [3] abarca las tareas de preparación e integración de datos [7, 8], tratamiento de ruido [9], imputación de valores faltantes [10, 11] y tareas de reducción de datos, como selección de características [12], selección de instancias [13] y discretización [14]. La correcta aplicación de estas tareas sobre los datos en crudo producirá Smart Data [5], tal como se conoce en la literatura a los datos de calidad limpios y útiles para las técnicas de minería de datos.

La minería de datos [2, 6] es otra tarea núcleo del proceso KDD, ya que es la encargada de inferir los patrones, relaciones o tendencias ocultos en los datos. Tradicionalmente, se divide en dos ramas según el tipo de conocimiento objetivo a aprender:

- Aprendizaje supervisado: infiere relaciones/modelos entre una variable objetivo y variables de entrada a partir de datos conocidos para predecir el valor de la variable objetivo para futuros casos no vistos. Según el dominio de la variable objetivo, se distinguen dos categorías:
  - Clasificación [15]: el dominio de la variable objetivo es discreto con todos sus posibles clases o etiquetas conocidas. La predicción de si un paciente está enfermo o sano es un ejemplo de clasificación en el area de la medicina.
  - Regresión [16]: los valores de la variable objetivo son continuos. Un ejemplo de problema de regresión es la estimación del precio de un producto.
- Aprendizaje no supervisado: tiene como objetivo descubrir diferentes relaciones entre instancias o variables del conjunto de datos sin una variable objetivo definida. Se distinguen dos familias diferentes:
  - Agrupamiento [17]: identifica grupos de muestras de acuerdo a una medida de similitud. Agrupamiento trata de minimizar las diferencias entre instancias del mismo grupo y de maximizar las distancias entre diferentes grupos.
  - Asociación [18]: detecta relaciones relevantes entre las diferentes variables de los datos.

La mayoría de las investigaciones de aprendizaje supervisado se centran en problemas de clasificación binaria o multiclase y regresión con una variable objetivo. Sin embargo, existe una multitud de problemas menos conocidos. En la literatura, estos reciben el nombre de problemas de aprendizaje supervisado singulares o no estándar [19].

Los problemas de aprendizaje supervisado singurales son aquellos que no siguen el esquema habitual de los problemas supervisados [20, 19]. Debido a esto, los algoritmos estándares de minería de datos no funcionan correctamente o directamente, no se pueden aplicar a estos problemas.

De acuerdo a las diferencias con el aprendizaje supervisado clásico, los problemas singulares pueden ser categorizados de la siguiente manera:

- Problemas con estructura no-estándar: no obedecen la estructura tradicional de un vector de entradas por un solo valor objetivo. Es decir, estos problemas poseen multiples vectores de entradas por cada salida o multiples variables objetivos. El aprendizaje multi-instancia [21] y el aprendizaje multi-vista [22] son ejemplos de problemas con multiples entradas. El aprendizaje multi-etiqueta [23] y la regresión multi-objetivo [24] son ejemplos de datos con multiples salidas.
- Problemas con información parcial o imprecisa: tienen información incorrecta o carecen de valores en características, instancias o variable objetivo. Un ejemplo bien conocido es el aprendizaje semi-supervisado [25], donde parte de las instancias carecen de valor para la variable objetivo. La clasificación desbalanceada [26] puede ser considerada un problema con información parcial por la falta de representación en las clases minoritarias. La clasificación con ruido de clase [9] también ha sido considerada en esta categoría [20]. Otros ejemplos más específicos son el aprendizaje one-class [27], el aprendizaje zero-shot [28] o el aprendizaje one-shot [29].
- Problemas con restricciones de conocimiento previo: presentan relaciones o limitaciones provenientes del conocimiento experto del problema aplicado. Estas restricciones deben de tenerse en

consideración durante el aprendizaje. La regresión ordinal [30] incluye relación de orden entre las etiquetas de clase. La clasificación monotónica [31] incorporan restricciones de orden entre los atributos y la clase.

Los problemas supervisados singulares se tratan mayormente desde dos enfoques principales [19]: transformación del problema y adaptación de algoritmos. El primer enfoque hace uso de técnicas de preprocesamiento para transformar el problema original en uno o más problemas estándares con menor complejidad. Estos problemas más simples se resuelven con las técnicas habituales de minería de datos. Sus soluciones son combinadas en una solo solución del problema original. El segundo grupo de soluciones busca diseñar nuevos algoritmos o adaptaciones de métodos existentes que consideren las peculiaridades de los datos del problema singular.

Cabe destacar que los problemas singulares no son excluyentes entre sí. Es común encontrarse diversos escenarios singulares en una misma aplicación real. La combinación de estos suele complicar significativamente el aprendizaje. Por ejemplo, la presencia de ruido de clase en clasificación desbalanceada o clasificación monotónica pueden agravar la precisión de las clases menos representadas [32] o romper la restricciones orden [33], respectivamente. Otro ejemplo es la combinación de los escenarios multi-instancia y multi-etiqueta [34]. Sin embargo, pocas propuestas estudian dos o más problemas singulares al mismo momento.

Esta tesis se centra en dos problemas supervisados singulares: clasificación desbalanceada [26] y la clasificación con restricciones monotónicas [31].

El problema del desequilibrio entre clases [26] alude a una diferencia significativa en el número de casos representativos de cada clase. Es decir, ciertas clases, comúnmente llamadas clases minoritarias, poseen muchas menos instancias que otras, conocidas como clases mayoritarias. Los clasificadores estándar tienden a perder precisión de las clases minoritarias debido a dicha disparidad en la cantidad de ejemplos. Esta situación es bastante común en problemas reales, tales como la clasificación de crédito bancario [35, 36] o problemas de dominio empresarial [37]. En estos escenarios reales, la errónea clasificación de un ejemplo de clases minoritarias habitualmente entraña un mayor coste. Por lo tanto, abordar este problema es sumamente esencial.

Por consiguiente, se han diseñado nuevas técnicas para hacer frente al problema del desequilibrio entre clases con tres enfoques diferentes [26]: propuestas basadas en algoritmos [38, 39, 40], aprendizaje sensible a costes [41, 42, 37], y muestreo de datos [43]. Los ensembles se han combinado con éxito con estos métodos anteriores obteniendo un gran rendimiento en este problema [44].

La clasificación monotónica o clasificación con restricciones monotónicas es un problema de clasificación singular, donde existe una relación de orden entre la variable ordinal de clase y algunos atributos de entrada ordinales o numéricos [31]. Dichas relaciones monotónicas restringen el incremento o decremento de la etiqueta de clase al incremento o decremento de los atributos. Es decir, la predicción de la etiqueta de clase no deben disminuir en presencia de mejores valores de entrada mientras que el resto se mantiene igual.

Los algoritmos de clasificación estándar no consideran dichas restricciones en el aprendizaje, por lo que sus predicciones tienden a violar la monotonía. Por ello, se han desarrollado multitud de clasificadores monotónicos basados en árboles de decisión [45, 46, 47], clasificadores difusos [48, 49], redes neuronales [50, 51], aprendizaje basado en instancias [52, 53, 54] y aprendizaje de ensemble [55, 56]. Sin embargo, muchos de estos padecen de dos problemas comunes en la clasificación monotónica: i) Algunos algoritmos requieren datos puramente monotónicos, lo cual rara vez ocurre debido a las inconsistencias en los datos y ii) Algunos modelos están altamente sesgados hacia las restricciones de monotonía y tienen un rendimiento muy pobre en términos de precisión.

Estas restricciones de conocimiento previo se requieren frecuentemente en problemas reales de evaluación, tales como modelado del riesgo de crédito [57], del precio de viviendas [58] y evaluación de profesorado [59]. Los cuales son también escenarios comunes de la clasificación desbalanceada, debido a que aquellas clases más valoradas suelen estar menos representadas.

Esta tesis pretende proponer nuevas soluciones basadas en clasificadores robustos y técnicas de preprocesamiento para estos dos singulares problemas supervisados. La Figura 2 resume todos los diferentes problemas y técnicas tratadas en los objetivos de esta tesis.

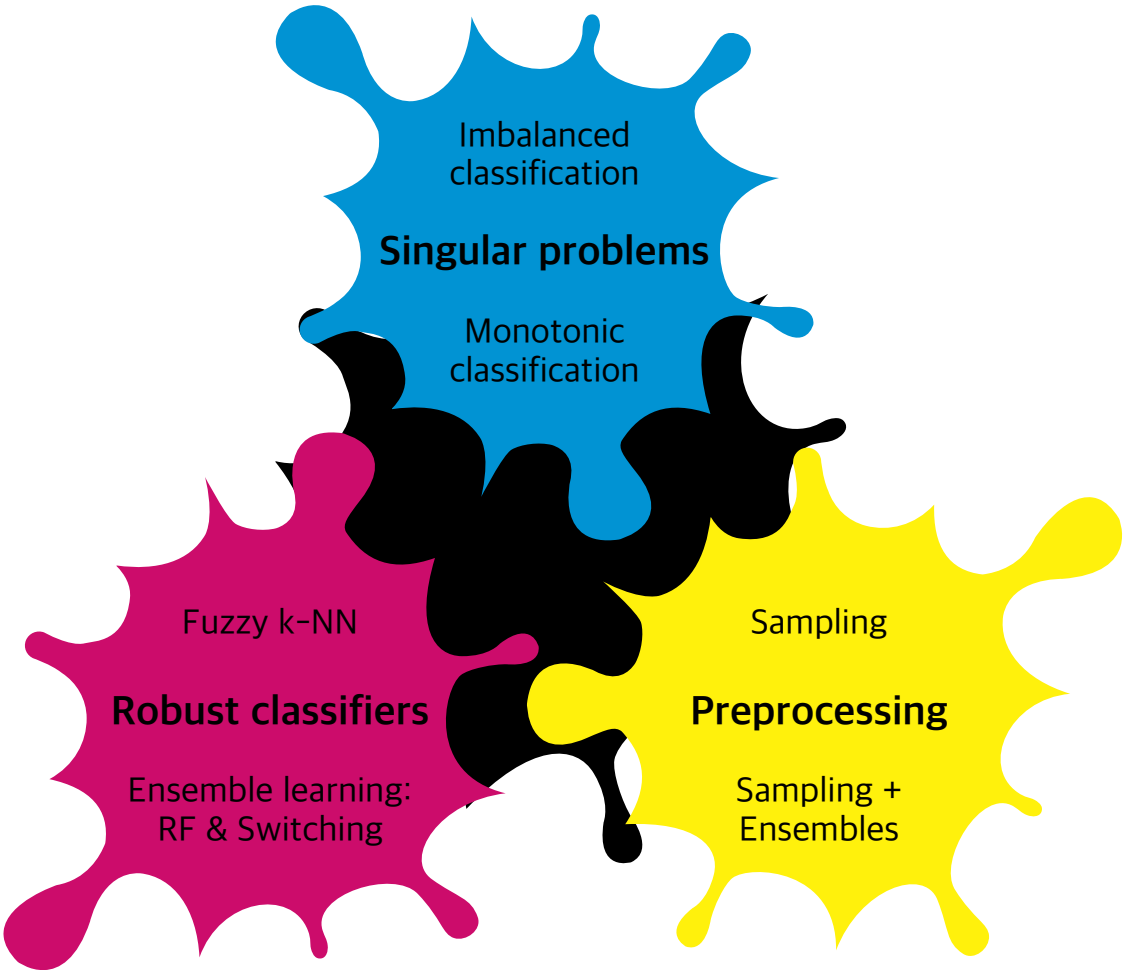


Figura 2: Objetivos de la tesis doctoral.

Los ensembles son uno de los modelos más robustos y exitosos de la minería de datos [60, 61]. Estos combinan varios clasificadores base para que su predicción fusionada mejore significativamente el rendimiento de un solo modelo [62, 63]. Dicha combinación permite la corrección de posibles errores cometidos por los modelos base y promueve una gran capacidad de generalización sin descuidar el conocimiento más local [62, 64].

La incorporación de diversidad en los modelos base del ensemble es fundamental para conseguir dichos beneficios de precisión y generalización. Existen una multitud de técnicas de promoción de la diversidad, tales como aleatorización de procesos algorítmicos [65, 66] o muestreo (bootstrapping) [67, 68].

Random Forest [65] es uno de los algoritmos más precisos y populares [61, 69]. Este ensemble

de árboles de decisión [70] combina bootstrapping y la incorporación de aleatoriedad en la selección del mejor corte en los árboles para alcanzar altas cotas de precisión. Dado su éxito, Random Forest con árboles de decisión monotónicos [45, 47] podría ser un propuesta ideal para conseguir los retos previamente explicados de la clasificación monotónica: alta precisión y monotonía sin datos puramente monotónicos. Dicha propuesta es estudiada y desarrollada más adelante en esta tesis.

La manipulación de las etiquetas de clase es otro enfoque muy interesante y poco explorado de la promoción de la diversidad en ensembles. Los algoritmos de cambio de etiquetas [71, 72] modifican los valores de clase de instancias seleccionadas al azar como una técnica de promoción de la diversidad. El algoritmo Switching [72] tiende a igualar los porcentajes de muestras de cada clase en escenarios ligeramente desequilibrados. Por lo que, un rediseño de este modelo en combinación con técnicas de muestreo [44] es muy interesante para paliar el impacto del desequilibrio de clase. En esta tesis, se propone un ensemble basado en cambio de clases de acuerdo a la distancia al enemigo más cercano para la clasificación altamente desequilibrada.

Las técnicas de muestreo son métodos de preprocesamiento para transformar un problema de desequibrio de clases en un problema estándar [3, 26]. Estas técnicas se suelen dividir en infra-muestreo o sobre-muestreo. El infra-muestreo reduce el número de muestras de las clases mayoritarias para equilibrar el conjunto de datos. Por el contrario, el sobre-muestreo aumenta el número de representantes de las clases minoritarias. Ambas técnicas han demostrado grandes resultados tanto de manera independiente [43] como combinadas con ensembles [44]. Sin embargo, su aplicación en problemas de clasificación monotónica con desequilibrio de clases podría deteriorar la monotonía de los datos y por tanto, afectar al rendimiento de los clasificadores. Este evento es estudiado en el transcurso de la tesis, dando lugar a técnicas de muestreo adaptadas a la clasificación monotónica desbalanceada.

Los algoritmos basados en los k Vecinos más Cercanos (kNN - k Nearest Neighbours) pueden ser una gran baza tanto para constituir técnicas de preprocesamiento como clasificadores robustos. El algoritmo kNN clasifica en base a las etiquetas de las instancias más similares al ejemplo dado. La utilidad de kNN transciende a la clasificación. Este ha sido usado en infinidad de técnicas de preprocesamiento de toda indole, tales como selección de instancias [73, 59], generación de prototipos [74], detección de fronteras [75, 76] y muestreo para el problema de desequilibrio de clases [77, 26], filtros de ruido [78], entre otros.

En esta tesis, se hace uso de kNN transversalmente en multiples propuestas. La distancia al enemigo más cercano guía el cambio de clase de instancias fronterizas en el ensemble basado en Switching. Algunas técnicas de muestreo propuestas para la clasificación monotónica desbalanceada están basadas en vecindarios.

Además, se propone un nuevo clasificador basado en Fuzzy kNN con restricciones monotónicas robusto al ruido monotónico. En la clasificación estándar, Fuzzy kNN [79] ha demostrado un alto rendimiento y robustez al ruido de clase [80], gracias a la previa extracción de la membresía de clase para las muestras de entrenamiento nítidas. Sin embargo, este mecanismo no considera las restricciones ni las violaciones de monotonicidad. La propuesta de Fuzzy kNN monotónico destaca por su alto rendimiento incluso en presencia de inconsistencias de monotonía en los datos.

En esencia, el objetivo principal de esta tesis es diseñar nuevas soluciones para dichos problemas, tanto de manera independiente como conjunta, y considerando otras situaciones singulares de datos, tales como la presencia de ruido. Como se ha mencionado anteriormente, estas propuestas siguen dos aproximaciones diferentes similares a los enfoques tradicionales para problemas singulares: clasificadores robustos y técnicas basadas en el preprocesamiento. Estas aproximaciones abordan cuestiones específicas de clasificación desequilibrada y la clasificación monotónica:

- Diseño de clasificadores robustos para los problemas singulares de clasificación desbalanceada y monotónica. Dicho objetivo incluye:
  - Altos niveles de precisión así como de monotonicidad con un clasificador Random Forest para la clasificación monotónica.
  - Robusto aprendizaje de ensembles basado en Switching según la distancia del enemigo más cercano (SwitchingNED) para problemas altamente desequilibrados.
  - Gran robustez al ruido monotónico para la clasificación monótona con una propuesta de Fuzzy kNN consciente de las restriciones y violaciones de monotonía.
- Desarrollo de técnicas basadas en preprocesamiento en clasificación desbalanceada y monotónica. Dentro de este paradigma, se incluyen las siguientes metas:
  - Mejorar el rendimiento de SwitchingNED mediante su combinación con diferentes técnicas de muestreo para la clasificación altamente desequilibrada.
  - Mitigar el impacto de las distribuciones desequilibradas de clases en los problemas de restricciones monótonas gracias a nuevas técnicas de muestreo para la clasificación monótona desequilibrada.

Finalmente, el resto de la tesis está organizado en dos partes principales: la tesis doctoral y las publicaciones asociadas. En esta primera parte, la Sección 2 desarrolla los conocimientos preliminares y los antecedes necesarios para esta tesis doctoral. En la Sección 3, se justifica la relevancia de la tesis y sus trabajos científicos asociados en el contexto de los problemas planteados. La Sección 4 describe los objetivos fijados para el desarrollo de la tesis. La Sección 5 presenta la metodología seguida en el transcurso de este trabajo. En la Sección 6 y la Sección 7, se resumen los estudios científicos de esta tesis doctoral y sus resultados, respectivamente. Finalmente, la Sección 8 remarca las principales conclusiones extraída y las líneas futuras de este trabajo.

La segunda parte de documento de tesis incluye las 4 publicaciones asociadas al desarrollo de este trabajo. Los trabajos científicos está ordenados de acuerdo a su fecha de publicación:

- Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity.
- Class switching according to nearest enemy distance for learning from highly imbalanced data-sets.
- Chain based sampling for monotonic imbalanced classification.
- Fuzzy k-Nearest Neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise.

2 Preliminaries

This section presents the background preliminaries of the singular supervised problems and the learning algorithms addressed in this doctoral thesis. Section 2.1 describes the class imbalance problem and some popular approaches. In Section 2.2, the classification with monotonicity constraints is explained in details. Section 2.3 introduces a general knowledge of ensemble learning, the most relevant ensemble approaches and diversity promotion mechanisms. Finally, a description of Fuzzy kNN algorithm is given in Section 2.4.

2.1 Class imbalance problem

The class imbalance problem occurs with the wide difference between class representations [26]. That is, one or more classes have much fewer instances than the remaining classes. These underrepresented classes are usually called minority or positive classes. The others are known as majority or negative classes.

The underlying issue with imbalanced data-sets is the impairment of minority class accuracy. Standard classifiers tend to misclassify minority class instances due to their generalization behavior. Furthermore, these minority classes are often the target in many real-life classification tasks [82, 36], which make these scenarios even more problematic.

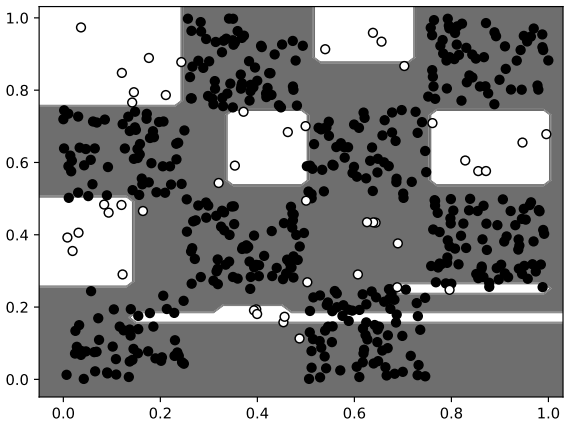
The high Imbalance Ratio (IR) is the main cause of performance loss of standard learning methods [83, 26]. This ratio represents the fraction between the number of negative instances and the number of positive examples. Scenarios with an IR over 9 are considered highly imbalanced and challenging for supervised learning [44, 26]. In addition, other commonly present factors considerably aggravate the imbalanced classification problem, such as lack of density, overlapping, noisy data and others.

The great problem complexity and its high presence in real applications have encouraged the design of many new proposals for the imbalanced classification. These approaches are categorized in three different groups [26]:

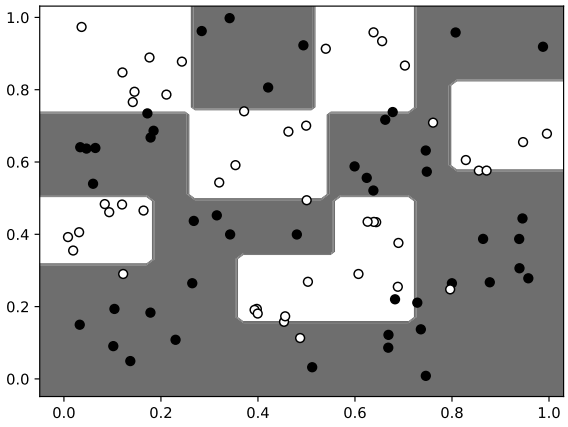
- Algorithm-level approaches: These solutions are adaptations of learning algorithms to handle imbalanced data-sets [38, 39, 40]. These proposals aim to modify the specific learning procedure of an existing classifier that produces the bias towards the most represented class. This requires a very deep knowledge of the classifier and its behavior. Therefore, these proposals are usually limited to a few classifier types.

- Cost-sensitive learning: These approaches take into account different costs for classification errors of each class [41, 42, 37]. A custom misclassification cost function different from the standard evaluation guides the learning process. These loss functions or error costs are usually determined by domain experts or other learning approaches. In imbalanced scenarios, cost-sensitive learning strongly penalizes the misclassification of minority class samples. Thus, learning is mainly focused on minority classes.

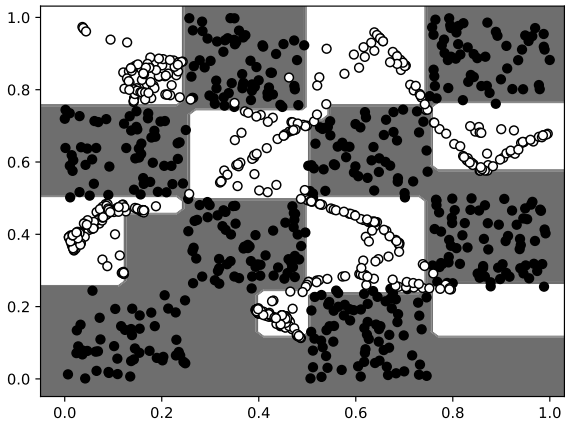
- Data level preprocessing methods: Data level techniques seek to balance class distributions through preprocessing [43]. These are relatively independent of the learning process. Moreover, they have the advantage of enabling existing standard classifiers. Figure 3 graphically exemplifies the main data level approaches. As seen in Figure 3a, the skewed class distribution causes the misclassification of many white-colored points from the minority class. Three main kinds of data level solutions are represented in Figure 3:



(a) Decision boundaries with imbalanced data.



(b) Decision boundaries with under-sampled data.



(c) Decision boundaries with over-sampled data

Figure 3: Examples of the impact of sampling techniques in decision tree learning.

- Under-sampling: This group of methods removes examples from the majority class to achieve a balanced data-sets. As shown in Figure 3b, many black dots have been eliminated and thus, the white decision boundaries have grown. The most basic under-sampling approach, known as Random Under-Sampling, randomly performs such sampling. Most sophisticated techniques attempt to retain the most relevant instances and eliminate the most dispensable or redundant ones [84, 85]. Under-sampling has the disadvantage of potentially eliminating valuable information about the problem.
- Over-sampling: As a counterpart, over-sampling techniques raise the number of instances of minority classes. Figure 3c shows better decision boundaries thanks to the generation of many more white examples using an over-sampling technique. The simplest over-sampling solution replicates some randomly selected instances from the minority class. This is called as Random Over-sampling. However, the state-of-the-art of over-sampling relies on the creation of synthetic data for the minority class [77, 86, 87]. SMOTE – Synthetic Minority Oversampling TEchnique [77] is the most popular approach. SMOTE produces synthetic data via linear interpolations of randomly chosen instances and their nearest neighbors of the same class. Over-sampling techniques may lead to noisy data.
- Hybrid techniques: These proposals combine under-sampling and over-sampling techniques to balance the class distributions. They seek to mitigate the potential disadvantages of the two main approaches [88].

Ensemble proposals for imbalanced classification may constitute an additional fourth category [26]. These approaches aim to improve the performance of standard cost-sensitive learning and data sampling methods. To do so, they combine different instances of a specific imbalanced classification method.

Cost-sensitive ensembles help to adjust the different misclassification costs during the learning process [41]. Data preprocessing ensembles incorporate different sampled data [44] in their iterations. Sampling techniques promote diversity in the ensemble, as well as balancing class distributions. Besides, ensemble schemes minimize the potential drawbacks of the sampling methods. For these reasons, ensembles are one of the most successful approaches for classification with imbalanced data-sets.

Another important aspect of the imbalanced classification is the evaluation of the different algorithms [26]. Standard evaluation metrics, such as accuracy, assume similar representations between classes. Thus, they fail to properly capture the misclassification of minority classes.

Therefore, practitioners employ other more suitable measures for imbalanced classification. The Geometric Mean or the Area Under the ROC Curve are commonly used in binary class imbalance scenarios [83, 43, 44]. For multi-class imbalanced classification, the averaged per-class accuracy or Macro Average Arithmetic is preferred [89].

2.2 Monotonic classification

Classification with monotonicity constraints is an extension of ordinal regression with additional order relations between the input features and the classes [31]. Both monotonic classification and ordinal regression are singular supervised problems with order constraints given by the problem prior knowledge [19]. In these problems, the output variable is ordinal without a precise magnitude between different class values. Monotonic classification restricts the increase or decrease of the class value to the increase or decrease of the input features.

These order constraints are sought by many real-life classification problems, such as credit rating [57], housing pricing [58], and professor evaluation[59]. To illustrate this, Table I.1 presents a small

Id.	Task #1	Task #2	Task #3	Class Rating
#1	High	High	High	Excellent
#2	High	Low	Low	Good
#3	Low	Medium	Low	Bad
#4	High	Low	Medium	Bad

Table I.1: Example of an employee evaluation problem where monotonic constraints are desired.

example of employee evaluation. This evaluation problem aims to rate some employees according to their performance in three different tasks. In Table I.1, employees from #1 to #3 are monotonically labeled. Employee #1 has a better evaluation than employees #2 and #3 because of his better performance in the tasks. And, employees #2 and #3 are not comparable since worker #2 is not strictly better in every task. However, the rating of employee #4 seems unfair. Worker #4 is rated worse than employee #2 even though the former performs the same or better than the later on all tasks. According to monotonic constraints, employee #4 should be labeled at least as good. The relation between employees #2 and #4 are monotonic violations. Classification with monotonicity constraints aims to avoid these violations in their predictions.

Formally, monotonic classification seeks to predict the class value $y \in \mathcal{Y} = \{l_1, l_2, \ldots, l_C\}$ for a given feature vector x. Class labels $\mathcal{Y}$ are ordinal with a specific order relation $\prec$ as $l_1 \prec l_2 \prec \ldots \prec l_C$. The feature values are also ordinal or numerical. The class predictions of monotonic classifiers are monotonically constrained by the order relations of the problem knowledge [31], that is, $x \succeq x' \rightarrow f(x) \geq f(x')$, where $x \succeq x' \Leftrightarrow \forall_j, x_j \geq x'_j$. The main goal is to build models whose predictions respect these constraints.

Standard classification algorithms are unaware of these monotonic constraints, which often lead to monotonic violations. Therefore, a multitude of proposals has recently been designed for the monotonic classification. These approaches are divided into two different groups [31]:

- Monotonic classifiers: These classifiers aim to satisfy the monotonicity constraints. There are different families of approaches such as instance-based learning [52, 53, 54], decision trees [45, 46, 47], neural networks [50, 51], fuzzy methods [48, 49], support vector machines [57, 48], and ensemble learning [55, 56]. Monotonic classifiers face two main challenges: i) learning with non-monotonic data, and ii) excellent performance in terms of both monotonicity and precision.

- Monotonic data preprocessing: Data inconsistencies such as noise impair learning with monotonic constraints. Preprocessing techniques must be aware of the monotonic constraints. Otherwise, they can also compromise the data monotonicity. These monotonic preprocessing proposals solve the common problems of poor data quality and facilitate the learning process of monotonic classifiers. Among these approaches, relabeling techniques [90, 91] and instances selection methods [59, 33] are noteworthy. These strategies pursue improving the monotonicity of a data-set by changing labels or filtering out non-monotonic examples.

The evaluation of monotonic classifiers is usually done considering two different perspectives of their performance: prediction capability and monotonicity.

Their prediction capability is commonly measured with standard accuracy [31]. Other common ordinal regression metrics are also used, such as Mean Absolute Error (MAE). MAE expresses the average of the differences between the real and predicted ratings.

There are several measures to evaluate monotonicity [31]. However, the majority of them are based on the same information: the number of monotonic violation pairs (NMP). Non-Monotonic Index (NMI) is the most popular metric to evaluate monotonicity. NMI is computed as the fraction of non-monotonic pairs (NMP) and the total number of sample pairs:

$$NMI = \frac{NMP}{N^2 - N}$$

2.3 Ensemble learning

Ensemble techniques combine several base learners to enhance the predictive performance of a single learner [62, 64, 70, 63]. These base methods are trained in a slightly different manner which leads to diverse predictions. The aggregation of such different predictions provides the ensemble with a high generalization without obviating the more local knowledge [62, 64]. Ensemble learning is considered one of the most powerful approaches in machine learning [60, 61]. Moreover, their performance has proven successful in a multitude of real-life problems [92, 93].

Ensemble algorithms are among the best methods for every supervised learning problem [62, 61], such as classification, regression, and ranking. These also stand out as excellent proposals for other learning problems, such as singular supervised tasks [31] and preprocessing [3, 5]. As previously mentioned, class imbalance classification is one of the singular problems where ensembles have proven to be very useful [44]. Besides, ensembles excel in each of the main preprocessing applications: noise filtering [94, 4], missing value imputation [95], data reduction [96, 97].

Diversity promotion is a key aspect of the prominent performance of ensembles. Many different mechanisms have been proposed to achieve diverse base estimators [63]. These methods are classified as data-level, algorithm-level, and hybrid techniques [62]. Data-level mechanisms introduce minor alterations to the training data to achieve different predictions from the same learning algorithm. Data sampling, such as bootstrapping [67], is one of the most popular data-level techniques. Algorithm-level approaches promote diversity through different parameter tuning, randomization of some algorithmic processes, or the use of different learning algorithms.

Ensemble pruning or selection is considered as an additional way to improve ensemble diversity [98, 99]. An oversized number of learners may impair the overall diversity and results in over-fitting. Ensemble pruning methods aim to select the best subset of estimators to improve the diversity and the performance of an ensemble.

Over the past few years, a vast amount of ensemble approaches have been designed with several combinations of diversity mechanisms, different prediction aggregations, and various base learners [62, 63]. Nevertheless, bagging [67] and boosting [100] schemes have been the most popular among the existing ensembles. Figure 4 presents the flowcharts of these ensemble techniques. Label Switching is also highlighted in Figure 4 due to its particular diversity promotion and its relevance to this doctoral thesis. These three ensemble schemes have the following characteristics:

- Bagging-based ensembles [67]: These algorithms train their different base learners independently of one another. Data-level mechanisms are usually their main source of diversity. Particularly, bagging-like ensembles commonly use data sampling with replacement, also known as bootstrapping. These features are represented in Figure 4a with slightly different training sets and separated estimators.

  Random Forest [65] is the most popular bagging ensemble. This ensemble uses weakly randomized decision trees to reach high levels of performance. Each tree is trained with bootstrapped data

and selects its next best cut from a subset of randomly chosen features. Random Forest is one of the most successful machine learning algorithms [61, 69].

- Boosting algorithms [100]: These ensembles have a sequential learning scheme. They learn from the classification mistakes made by previous learners. In future iterations, the learning importance increases for those misclassified training instances. As seen in Figure 4b, each base learner depends on the previous iterations, which set the weights of the training instances. These weights are represented with different sizes in Figure 4b.

  Some iconic examples of boosting ensembles are AdaBoost [68] or Gradient Boosting Machine [101].

- Label switching methods [71, 72]: These approaches manipulate the target attributes of some samples to promote diverse estimators. The negative impact of the introduced class noise disappears with the combination of a large number of base learners, as long as the modified instances are a considerably small fraction of the data-set. Figure 4c shows an example of a label switching scheme, where some red circles and blue squares switch their classes to blue and red, respectively.

  There are two different label switching approaches: Output Flipping [71] and Class Switching [72]. Output Flipping exchanges the classes of some instance pairs maintaining the original class distributions. On the contrary, Class Switching changes the labels of randomly selected examples. Thus, Class Switching achieves a better performance than Output Flipping and similar to Bagging in standard classification scenarios. However, label switching algorithms are not as explored as Bagging. This particular diversity promotion mechanism may have a great potential for other learning tasks.

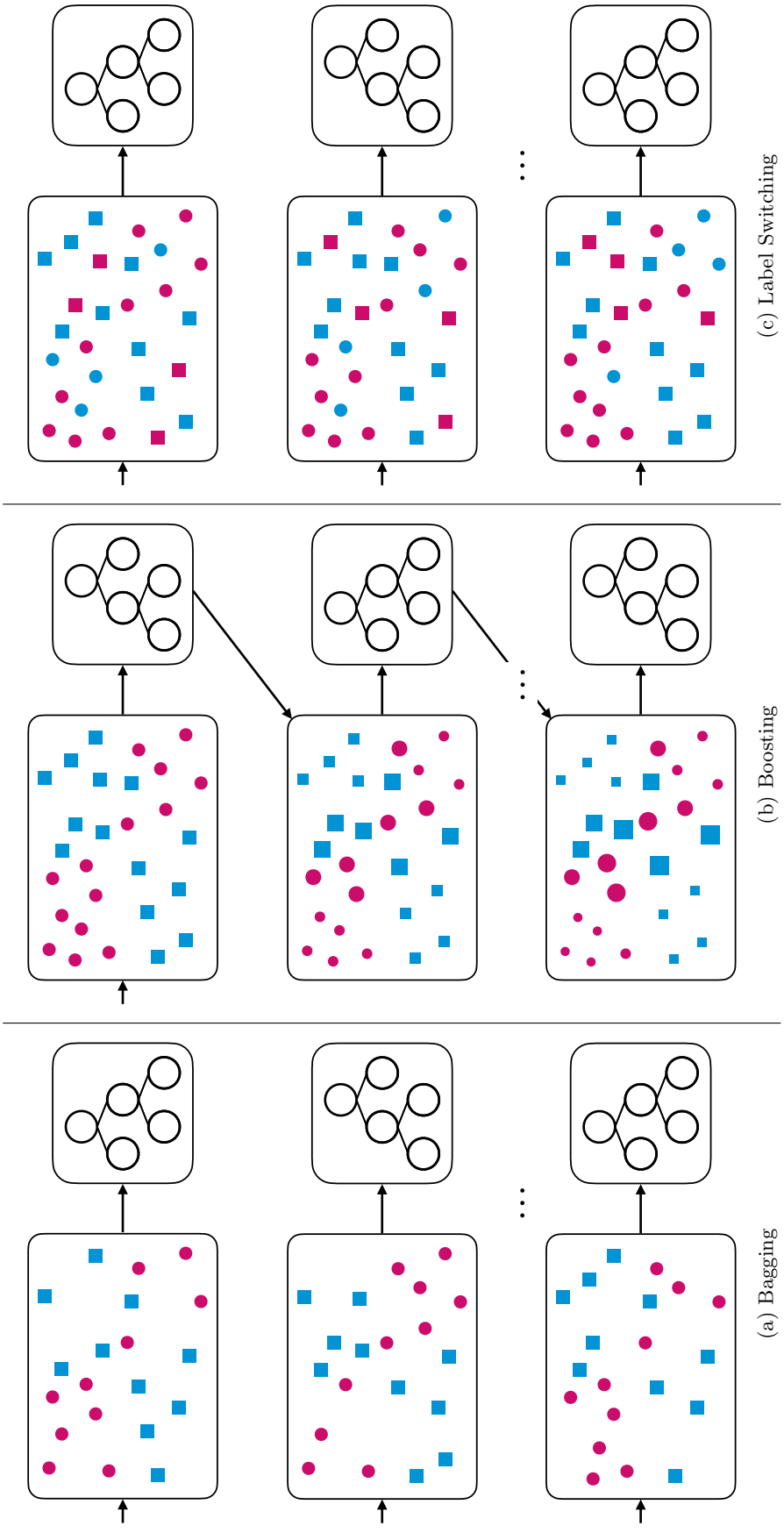


Figure 4: Ensemble schemes.

2.4 Fuzzy *k* Nearest Neighbors

The k Nearest Neighbors algorithm predicts the class label of a given sample x with the aggregation of the class values of the k most similar training instances to x. The similarity between examples is usually measured with distances, such as the Euclidean distance. In classification, the majority voting probably is the most common aggregation rule used by kNN algorithm.

Fuzzy classifiers model the possible uncertainty of belonging of different individuals to the existing classes. In Fuzzy Sets [102], such uncertainty is represented for each example x_i as a class membership vector $u_i = (u_{i1}, u_{i2}, \ldots, u_{iC})$, where C is the number of classes, $u_{il} \in [0, 1]$ and, $\sum_{l=1}^{c} u_{il} = 1$.

Fuzzy k Nearest Neighbors algorithms [80] embed fuzzy principles into this classic kNN algorithm (kNN), which enable the learning of fuzzy sets and fuzzy classification decisions. Many different approaches of Fuzzy kNN have been designed with diverse fuzzy mechanisms. And yet, the first Fuzzy k Nearest Neighbors algorithm (FkNN) [79] is one of the most outstanding proposals [80].

Similarly to classic kNN, the FkNN algorithm [79] classifies a given new instance x_i based on its K nearest training samples. However, the aggregation rule of FkNN combines the different memberships for each class l as follows:

$$u(x, l) = \frac{\sum_{j=1}^{K} u(x_j, l) * \frac{1}{||x - x_j||^{(m-1)}}}{\sum_{j=1}^{K} \frac{1}{||x - x_j||^{(m-1)}}} \tag{I.1}$$

As seen in Equation I.1, the membership $u(x_i, l) = u_{il}$ of individual x_i to label l is computed with the sum of the class memberships $u(x_j, l)$ of the neighbors x_j weighted by their distances to x_i. Neighbors with shorter distances have a greater weight in the aggregation of class memberships. The parameter m establishes the effect of the neighbor distances in the aggregation rule. With $m = 2$, the neighbor contribution to the class membership is linearly proportional to the inverse of its distance.

Once the class memberships are calculated, the crisp output value for the sample x_i is defined as the class l with the highest membership degree u_{il}.

Another major feature of FkNN comes from its application to standard classification data-sets. These data-sets do not include the class memberships of the training samples. Therefore, FkNN must transform the original crisp labels into class memberships to obtain a fuzzy set.

This transformation is carried out with an additional nearest neighbor procedure. The membership degree $u(x_i, l)$ of a training instance x_i to the class label l is calculated as the frequency of occurrences nn_l of the label l among the k nearest neighbors of x_i. The original class label of a sample x_i has a minimum membership value of 0.51. Equation I.2 expresses the explained calculation of the class memberships for a training sample x_i:

$$u(x_i, l) = \begin{cases} 0.51 + 0.49 * (nn_l/k), & \text{if} \quad y_i = l \\ 0.49 * (nn_l/k), & \text{otherwise} \end{cases} \tag{I.2}$$

This procedure is the main reason for its noise robustness. Noisy instances lose their negative impact as their class memberships are shared among the surrounding classes, and not the labeled noisy class.

3 Justification

As introduced in previous sections, singular supervised problems are a reality of data mining applications. However, these do not usually draw the same attention as standard problems and they are rarely addressed in conjunction with others.

Imbalanced classification is transversely common in multiple real classification problems. Although more specific, monotonic classification also has its interest in multiple fields of application. Due to the nature of problems with monotonic constraints, they often suffer from class imbalance and data inconsistencies that disrupt the monotonicity. These obstacles have to be treated considering these restrictions.

Therefore, it is necessary to design robust solutions to these two problems separately and combined. To this end, the following main fronts should be tackled:

- In imbalanced classification, ensemble algorithms in combination with sampling have been very successful [44]. While sampling techniques balance the training sets, the ensembles use them as diversity promotion and correct possible errors of these preprocessing techniques. These ensembles are mainly built with standard bagging and boosting schemes [44]. However, there are other ensemble schemes, that have not been explored and could be effective [63]. Class Switching is a very promising ensemble for this purpose, due to its good behavior in sightly imbalanced data-sets. This thesis explores a design based on Switching for the highly imbalanced classification.

- Predictive models of classification with monotonic constraints pursue two desired features: obtaining accurate and monotonic predictions and the tolerance to monotonic inconsistencies in the data. Ensembles are one of the most accurate methods in machine learning [61]. In particular, Random Forest [65] is probably the most popular algorithm among the different ensembles. Besides, decision trees have performed remarkably well in monotonic classification [45, 46, 47]. Therefore, Random Forest of monotonic decision trees could accomplish excellent levels of accuracy when addressing problems with monotonicity constraints. At the time of the exploration of this proposal, ensemble learning was little explored in monotonic classification, with only a few approaches [55] and none based on Random Forest.

- Tolerance to monotonic violations in the data is very desirable in monotonic classification, as the presence of inconsistencies such as class noise is highly common. Instance-based learning has shown great performance in monotonic classification [52, 53, 54]. However, these algorithms are very sensitive to monotonic violations in the training data. Fuzzy k Nearest Neighbors [79] has proven great robustness and excellent performance in standard classification [80]. Therefore, a proposal based on Fuzzy kNN with monotonic constraints could employ such robustness to mitigate the impact of monotonic violations on the training process.

- As stated before, skewed class distributions are widespread in monotonic classification problems. Sampling techniques are probably the most convenient solutions for imbalanced monotonic problems, due to the transformation into simpler monotonic problems and the possibility of enabling all existing monotonic classifiers. However, such techniques that do not consider monotonic constraints could exacerbate existing monotonic violations of the training set, or even introduce new ones. Thus, these preprocessing techniques need to be adapted to respect the data monotonicity. Furthermore, it may be worthwhile to reduce some monotonic violations simultaneously with sampling.

All these matters and the proposed solutions fall under the theme of this doctoral thesis: the design of preprocessing techniques and robust classifiers for the singular problems of imbalanced and monotonic classification.

4 Objectives

After the introduction of the fundamental principles of this thesis, its main objective is presented in detail. The central purpose of the PhD dissertation is to explore new solutions for certain singular supervised problems: imbalanced classification and monotonic classification. This research involves the tasks of analysis, design, implementation, and evaluation of novel approaches that address both problems independently or together with other singular scenarios.

Before addressing these tasks, a study of the current state of the class imbalance problem and monotonic classification is necessary. This theoretical study of the problems and existing approaches allows a better understanding of the situation and the open challenges of these research fields. The design of innovative proposals would be hard to accomplish without the knowledge foundations acquired by the conduct of this study.

In addition to the initial study of these singular problems, the global objective of the thesis is divided into two according to the followed approximations of the designed solutions. These objectives are described as follows:

1. **Design of robust classifiers for the singular supervised problems of imbalanced and monotonic classification with the following purposes:**

   1.0. An initial study of existing robust classifiers for standard and singular problems: this study aims to provide awareness of existing robust techniques that are suitable to adapt to the faced singular problems. In particular, ensemble-based proposals are studied in-depth, since they are among the most robust in standard classification.

   1.1. High levels of accuracy as well as monotonicity in monotonic classification: With this purpose, we design a Random Forest classifier of monotonic decision trees and an ensemble pruning technique guided by monotonicity.

   1.2. New ensemble learning for highly imbalanced problems: We develop a new ensemble based on a Switching scheme for highly imbalanced classification. The proposal improves the basis of the good performance of Class Switching with slightly imbalanced data. The switching scheme is redesigned to favor the learning of minority classes.

   1.3. Great robustness to monotonic noise for classification with monotonic constraints: With this goal, we design an algorithm based on Fuzzy kNN with monotonicity constraints. This new classifier is aware of monotonic violations in data of monotonic classification.

2. **Development of preprocessing-based techniques in imbalanced and monotonic classification:**

   2.1. Combination of the Switching based ensemble with sampling techniques for imbalanced classification: this combination aims to enhance the performance of the robust Switching based ensemble when facing highly imbalanced data. Sampling techniques can relieve the responsibility of class switching scheme to balance the class distribution.

   2.2. To mitigate the impact of skewed class distributions in problems with monotonic constraints: With this objective, we design sampling techniques for monotonic imbalanced classification. The most popular sampling techniques for class imbalance problems are redesigned to respect monotonicity during preprocessing and to enable all monotonic classifiers.

5 Methodology

Now, the methodology followed during the conduct of this thesis is introduced. This methodology is an adaption of the standard scientific method to both theoretical and empirical research works done in recent years. The following steps constitute the methodological development of the thesis:

1. **Problem characterization**: throughout the study of the current state of the singular supervised problems, their existing solutions and open challenges. The interactions of imbalanced and monotonic classification and other singular situations are also analyzed. Among the existing solutions, the robust classifiers, such as ensembles, are studied in standard and singular problems.

2. **Hypothesis formulation**: design of new preprocessing- and ensemble-based robust approaches for singular supervised problems: imbalanced and monotonic classification. These proposals address both problems independently or together with other singular scenarios.

3. **Experimentation**: gathering performance results of the implementation of the new approaches of these singular problems. The performance is measured in terms of the accuracy by default or specialized precision metrics for imbalanced classification, and monotonicity for monotonic classification.

4. **Hypothesis contrasting**: an empirical comparison of the gathered results with the performance of other state-of-the-art algorithms for imbalanced and monotonic classification. Through the analysis of the experimental comparison, the quality of the novel proposed algorithms is assessed concerning their problem-solving capabilities.

5. **Hypothesis validation or refutation**: confirmation or rejection of the stated hypothesis through the analysis of the results gathered in the different experiments. If the hypothesis is rejected, some modifications should be considered before iterating over the previous steps again.

6. **Scientific thesis**: extraction and acknowledgment of conclusions regarding the research progress. These conclusions along with the entire scientific process should be compiled into a thesis dissertation and journal publications.

6 Summary

In this section, the approaches proposed with the thesis are summarized. Each of the summaries includes a brief introduction of the problem context, a description of the proposed method, and an outline of the experimental study carried out. Their main results are described in the subsequent Section 7. These proposals were published as the following research papers in scientific journals:

- González, S., Herrera, F., & García, S. (2015). Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing*, 33(4), 367-388.
- González, S., García, S., Lázaro, M., Figueiras-Vidal, A. R., & Herrera, F. (2017). Class switching according to nearest enemy distance for learning from highly imbalanced data-sets. *Pattern Recognition*, 70, 12-24.
- González, S., García, S., Li, S. T., & Herrera, F. (2019). Chain based sampling for monotonic imbalanced classification. *Information Sciences*, 474, 187-204.
- González, S., García, S., Li, S. T., John, R., & Herrera, F. (Accepted, 2020). Fuzzy k-Nearest Neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise. *Neurocomputing*.

The rest of this section is organized as follows. First, Section 6.1 describes a monotonic Random Forest with an ensemble pruning mechanism for monotonic classification. In Section 6.2, an ensemble approach based on Class Switching is presented for highly imbalanced classification. Then, Section 6.3 summarizes the proposal of popular sampling techniques for monotonic imbalanced classification. Finally, Fuzzy kNN with monotonicity constraints is explained along with its robustness to monotonic noise in Section 6.4.

6.1 Random Forest for classification with monotonicity constraints

Classification with monotonicity constraints is common in real-life problems with ordinal data. Such problems require classifiers that respect the order relationships between the ordinal target variable and ordinal attributes [31]. That is, the presence of better input values should not lead to a worse class value.

Recently, multiple algorithms of different nature have been developed for monotonic classification [45, 46, 47]. Among these, decision trees are worth highlighting because of their good performance and explainable models [45, 46, 47]. However, monotonic predictive models tend to be biased towards the monotonicity constraints of possibly inconsistent training data, and thus their accuracy is severely impaired [103].

Therefore, the implementation of more robust models is interesting to achieve high levels of accuracy, while maintaining monotonicity. In this sense, ensembles and in particular, Random Forest [65], are some of the most robust and accurate methods of data mining [61].

The application of Random Forest is proposed to increase the accuracy in monotonic classification tasks (Objective 1.1.). This Random Forest approach is built with Monotonic Induction based C4.5 trees (MID-C4.5) as base learners [45]. MID-C4.5 combines information gain or entropy with the non-monotonic index as an impurity measure to guide tree growth. This measure of MID-C4.5 trees includes a parameter to determine the priority of monotonicity over information gain. Besides bootstrapping

and the random selection of the potential tree cuts, Monotonic Random Forest uses this parameter as a diversity promotion technique. That is, the forest trees are grown with different random values for this parameter. This results in more diverse trees with different priorities of accuracy and monotonicity. Finally, an ensemble pruning mechanism discards those trees with a lower non-monotonic index.

An empirical comparison is conducted against 5 representative algorithms of monotonic classification. The experimental framework includes 90 monotonic and non-monotonic data-sets and 3 evaluation metrics: accuracy, mean absolute error, and non-monotonic index. The obtained results are checked with non-parametric statistical tests.

The journal paper related to this proposal is:

González, S., Herrera, F., & García, S. (2015). Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing*, 33(4), 367-388.

6.2 Class switching for highly imbalanced classification

Class imbalance problem is globally common in many real classification tasks. Imbalanced classification refers to an accuracy decrease of weakly represented classes due to the significant difference in the number of instances with other classes. Moreover, this situation is even more harmful when these minority classes are often the target of the problems.

Ensemble learning has shown great potential for training with imbalanced data-sets [44, 26]. These approaches combine the robustness of standard ensembles algorithms with under- and over-sampling techniques [26]. There are other unexplored ensemble schemes differently from the standard bagging and boosting [63], which could be very effective for imbalanced classification (Objective 1.2.). Class Switching algorithm [72] changes the class labels of randomly selected instances to build diverse base learners of the ensemble. This ensemble scheme tends to balance the class distributions of the training data-set. Therefore, Switching has a bit better performance than bagging for slightly imbalanced data-sets.

A new Switching-based ensemble with Nearest Enemy Distance (SwitchingNED) is proposed for learning from highly imbalanced data-sets. The switching mechanism is redesigned to prioritize examples near to a different class. With this, SwitchingNED aims for smarter and better balance class distributions due to the growth of the minority class samples near to the boundaries. Furthermore, SwitchingNED intends to preserve the information of the under-represented class by switching only the majority class samples to the minority class.

The new NED-based switching procedure is implemented with random selections guided by a probability distribution. This distribution is calculated just once with the inverse of the NED of each majority sample. First, SwitchingNED gathers the nearest example of the minority class for each majority class instance. The switching probability of each majority class sample is computed as the inverse of the euclidean distance to its nearest minority instance. The extracted probabilities are normalized. These are employed in each iteration of the ensemble.

As done with standard ensembles, SwitchingNED can be combined with different sampling techniques. Thus, the responsibility for balancing the class distribution is shared between the switching mechanism and the sampling methods (Objective 2.1.).

Several different empirical comparisons are carried out concerning this new ensemble. First, SwitchingNED is compared against the original Class Switching algorithm to test the potential

improvement of the new switching mechanism. Then, the performance of SwitchingNED is analyzed in combination with 3 different sampling techniques. Finally, SwitchingNED is compared with 5 different combinations of sampling techniques and ensembles. All these experiments share the same experimental framework composed of 33 highly imbalanced data-sets, Area Under the Curve as evaluation metric and non-parametric statistical testing.

The journal publication associated with this approach is:

> González, S., García, S., Lázaro, M., Figueiras-Vidal, A. R., & Herrera, F. (2017). Class switching according to nearest enemy distance for learning from highly imbalanced data-sets. *Pattern Recognition*, 70, 12-24.

6.3 Sampling techniques for monotonic imbalanced classification

Monotonic classification [31] arises from real-life ordinal tasks [35, 36, 37], such as evaluation and rating problems. These tasks usually suffer from skewed class distributions because highly rated examples are much less frequent than those with an average rating. The imbalanced classification problem together with the strict enforcement of monotonicity constraints may highly deteriorate the accuracy of underrepresented classes.

Therefore, the imbalanced class distributions must be addressed in the scenarios with monotonic constraints. Sampling techniques [43, 26] can be one of the most appropriate solutions to monotonic imbalanced classification. Through training set preprocessing, these methods offer the advantage of enabling all available monotonic classifiers. However, sampling techniques may cause monotonic inconsistencies in the training data due to their lack of awareness of monotonicity.

This research work studies some solutions for the negative impact of standard sampling procedures on the monotonicity (Objective 2.2.). Chain based sampling techniques are proposed for monotonic imbalanced classification tasks. Chains are sets of comparable samples with useful monotonic information [52, 53]. These new sampling schemes employ knowledge of monotonic chains and monotonic violations to preserve, as much as possible, the data monotonicity.

Chain based sampling schemes follow several good practices concerning the relevance of samples towards data monotonicity. Given this criterion, three different types of instances are defined with their sampling policies as follows:

- Samples at the chain limit: are not upper- or lower-dominated by other samples of the same class. That is, they are the greatest or smallest sample in the chain. These examples establish the ordering border of each class. Thus, these should be preserved during under-sampling or reinforced with over-sampling.
- Instances inside chains: are fully dominated by other instances of the same class. Therefore, their removal may cause less loss of ordering information. During sampling, samples inside chains receive less priority than those at the chain limits.
- Monotonic violations: are inconsistencies that break the data monotonicity. If possible, sampling techniques should avoid or eliminate them. Otherwise, the preprocessing methods involve such instances proportionally to the number of violations they cause.

These good practices apply to the majority of under- and over-sampling techniques. In this study, the schema is implemented in five popular sampling approaches: Random Under-Sampling (RUS), Random Over-Sampling (ROS), Synthetic Minority Oversampling TEchnique (SMOTE) [77],

ADAptive SYNthetic sampling approach (ADASYN) [86] and Majority Weighted Minority Oversampling TEchnique (MWMOTE) [87].

The empirical study is divided into two different experiments. First, some standard and ordinal sampling techniques are analyzed in terms of their influence on the predictions of monotonic classifiers. The second experiment compares the performance of the proposed and standard sampling techniques in combination with different monotonic classifiers. The performance of 5 different monotonic models is measured as the multi-class accuracy and non-monotonic index on 8 well-known monotonic imbalanced data-sets. The outcomes are contrasted with several non-parametric statistical tests.

The research paper of this study is:

González, S., García, S., Li, S. T., & Herrera, F. (2019). Chain based sampling for monotonic imbalanced classification. *Information Sciences*, 474, 187-204.

6.4 Monotonic Fuzzy *k*NN: Moving towards the robustness of monotonic noise

Monotonic violations are common in the training data of practical monotonic classification applications, due to the existence of data inconsistencies such as class noise. This monotonic noise negatively affects the learning and predictions of many monotonic classifiers [33, 31]. Instance-based learning approaches [52, 53, 54] are a clear example of monotonic classifiers that are highly sensitive to monotonic violations. Therefore, classifiers highly tolerant to monotonic noise are very desirable for classification with monotonicity constraints.

Fuzzy k Nearest Neighbors (FkNN) [79] has shown excellent noise robustness and performance in standard classification [80]. Its tolerance mainly comes from the extraction of class memberships for each training instance according to their neighborhood classes. In this procedure, the class membership of a noisy sample is shared with nearby classes. Then, the wrong label loses some of its influence.

This research work proposes a FkNN classifier with monotonic constraints and great robustness against monotonic violations (Objective 1.3.). Monotonic Fuzzy k Nearest Neighbors (MonFkNN) aims for highly accurate and monotonic predictions without pure monotonic training set and performance versatility in monotonic classification. With these purposes, MonFkNN is designed with the following features:

- Mon$F$NN is fully aware of monotonic violations during training class membership extraction. First, the inconsistencies from repeated samples with different labels are combined into one class membership. Then, the class memberships for the remaining training instances are calculated with a monotonic nearest neighbor algorithm. For a given sample x, its neighbors are restricted to those ones with monotonic valid classes for x. The upper label of valid class set is defined as the lower class label among the training instances greater than x. The lower class is the highest label of the samples smaller than x.

- In the prediction phase, the aggregation of the previous monotonic class memberships is also monotonically restricted by a monotonic nearest neighbor rule or a contribution penalty.

- MonFkNN is a versatile classifier that satisfies different needs of monotonic classification. Through parameter tuning, MonFkNN can be configured in two different versions: a more monotonicity biased version (Pure Monotonic FkNN) and accuracy focused model (Approximate Monotonic FkNN).

The experiments of this study analyze the performance of MonFkNN in different scenarios. First, the two versions of MonFkNN are compared against the standard FkNN. Then, the performance of MonFkNN is contrasted with 7 different representative classifiers of monotonic classification. Finally, the impact of monotonic noise and the robustness of MonFkNN are analyzed. The performance is measured in terms of accuracy, mean absolute error, and non-monotonic index on 12 different commonly used data-sets. All conclusions are subjected to different non-parametric statistical tests.

The journal paper associated with this approach is:

González, S., García, S., Li, S. T., John, R., & Herrera, F. (Accepted, 2020). Fuzzy k-Nearest Neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise. *Neurocomputing*

7 Discussion of results

This section is devoted to analyze the main results of the research works submitted as part of this thesis. The result discussion is organized with the same subsections as the previous section.

7.1 Random Forest for classification with monotonicity constraints

In this research work, Random Forest is applied to increase the accuracy in classification problems with monotonicity constraints. To do so, the Monotonic Random Forest proposal is built with randomized monotonic induction C4.5 trees. Then, a monotonic based pruning mechanism selects the trees with the smallest non-monotonic index. Several empirical comparisons are carried out to test the performance of the Random Forest approach for monotonic classification.

The following conclusions are drawn given the experimental outcomes:

- The monotonic based ensemble pruning does not significantly influence the prediction capacities of Random Forest. However, the pruning mechanism strongly improves the monotonicity of the model as ratified by the statistical tests.
- In comparison with other classifiers, Monotonic Random Forest is overwhelmingly better in terms of accuracy and mean absolute error for both monotonic and non-monotonic data-sets. The main reason is the potential robustness and precision of ensemble schemes.
- Even though the maximum depth is not limited for the ensemble trees, Random Forest grows simpler decision trees than other approaches. This event is explained by the training randomization and the ensemble pruning procedure. The most monotonic trees usually have smaller numbers of leaves.
- In terms of monotonicity, the ensemble-based proposal performs better than almost all algorithms with exception of MID-C4.5. Statistical tests support similar good monotonic behavior between MID-RF and MID-C4.5.

To conclude, Random Forest succeeds to improve accuracy while maintaining high monotonicity in monotonic classification tasks. The ensemble approach achieves better performance in every precision measure than the compared methods. Random Forest also obtains great levels of monotonicity in its predictions. As a result, this contribution permits the fulfillment of Objective 1.1.

7.2 Class switching for highly imbalanced classification

A Switching based ensemble according to Nearest Enemy Distance is proposed for learning from highly imbalanced data. SwitchingNED limits the label changes from the majority class to the minority class. Its new procedure prioritizes samples near to the boundaries. With these techniques, the proposal aims for smarter balanced class distributions and improved learning in highly imbalance scenarios.

Several empirical studies analyze the performance of SwitchingNED compared with the standard Switching algorithm, in the combination with sampling techniques and against other ensemble algorithms.

SwitchingNED significantly improves the performance of the original Class Switching algorithm. Its NED-based mechanism allows more switched examples with an overall better behavior. This

improvement is widely confirmed by statistical tests. Besides, a graphical study shows greater and more precise decision areas for the SwitchingNED ensemble.

The combination of SwitchingNED with sampling techniques helps to balance the skewed class distributions. SwitchingNED combined with Random Under Sampling (RUS), Random Over Sampling (ROS), and SMOTE considerably outperforms the ensemble on itself. The strengths of these three approaches are clearly different. SwitchingNED with Random Under Sampling is substantially the best combination, as evidenced by the statistical tests.

SwitchingNED has a great performance compared to other representative ensembles for imbalance classification tasks. Under-sampling based SwitchingNED significantly outperforms almost all chosen ensembles. The SwitchingNED approach shows slightly better outcomes than EUSBoost without statistical support. However, the switching ensemble is computationally much more efficient.

Therefore, SwitchingNED is a great approach to address highly imbalanced classification problems. The new switching procedure alleviates the limitation of the switching rate of the original algorithm. This contribution opens news opportunities for label switching in different contexts, such as imbalance classification. SwitchingNED and its combinations with different sampling techniques fulfill Objective 1.2. and Objective 2.1. of this thesis, respectively.

7.3 Sampling techniques for monotonic imbalanced classification

This research study proposes chain based sampling techniques for monotonic imbalanced classification tasks. A new sampling scheme is designed with the awareness of monotonic constraints and violations to preserve the data monotonicity. In this work, five different well-know sampling techniques implement this monotonic sampling scheme: RUS, ROS, ADASYN, SMOTE, and MWMOTE. The empirical study includes two different experiments involving both standard and monotonic versions of these procedures.

The first experiment studies the effect of standard and ordinal sampling procedures on the average accuracy and monotonicity of monotonic classifiers. The outcomes of this experiment show that the standard and ordinal sampling methods improve the general precision of the classifiers. However, these techniques severely harm monotonicity.

In the second experiment, the five monotonic sampling techniques are compared with their standard versions. Both monotonic and standard methods achieve a similar level of improvement on the average accuracy improvements of the classifier predictions. But, the monotonic preprocessing procedures succeed to preserve better the monotonicity. Monotonic RUS and MWMOTE have a more remarkable difference with their standard counterparts.

Among the proposed monotonic techniques, monotonic ADASYN and ROS are the best on average in terms of average accuracy and monotonicity, respectively. Monotonic RUS has the worst performance in general. However, all of these techniques have their unique behaviors and different strengths. For example, monotonic RUS achieves good performance with slightly imbalanced data. Both monotonic ROS and RUS are interesting in the presence of repeated samples with different labels. Monotonic SMOTE has a balanced performance between improving precision and preserving monotonicity.

As a result, this research work connects the class imbalance problem and monotonic classification. The experimental study shows the negative influence of the existing sampling techniques on monotonicity. Chain based sampling proposals prove a similar improvement of the classifier precision while preserving better the data monotonicity. This meets the goals expressed in Objective 2.2. of this thesis.

7.4 Monotonic Fuzzy *k*NN: Moving towards the robustness of monotonic noise

As previously stated, a monotonic FkNN is proposed to mitigate the influence of monotonic noise and thus, achieve predictions with a good level of accuracy and monotonicity. MonFkNN is also designed to be flexible to the different requirements of monotonic problems. Three different experiments are carried out to prove its excellent performance.

The first experiment compares the performance of the two versions of MonFkNN against the standard FkNN. Both Pure and Approximate Monotonic versions of MonFkNN outperform on average the original FkNN algorithm in all tree metrics: accuracy, mean absolute error, and non-monotonic index. In particular, MonFkNN is overwhelmingly better than FkNN in terms of monotonicity. FkNN has a better accuracy performance only in 3 data-sets. But, there is no scenario where FkNN outperforms the monotonicity of MonFkNN. These wide differences are supported by Wilcoxon statistical tests.

In the second experimental study, MonFkNN is compared with 7 other monotonic classifiers of the state-of-the-art. MonFkNN results again the best algorithm on average in terms of predictive capacities and monotonicity. The proposed method achieves much better accuracy and mean absolute error than the other monotonic classifiers. According to monotonicity, our new approach is as good as the best models of the state-of-the-art. These outcomes are also contrasted by the Friedman rank test and the Bayesian Sign test.

Finally, the robustness of MonFkNN against large amounts of noise is analyzed in comparison with MkNN. The number of class noise is graphically shown to proportionally influence the number of monotonic violations. The performance of the two algorithms decreases significantly as the amount of class noise increases. However, MonFkNN maintains better performance than MkNN at all times. The performance loss in terms of monotonicity is significantly lower for the MonFkNN algorithm. In addition, MonFkNN manages to respect the class boundaries of Artiset data-set with 35% of noisy instances.

In conclusion, the proposal MonFkNN has significantly better precision while maintaining the best monotonicity of monotonic classifies. It also proves overwhelming robustness against a great number of noisy instances (Objective 1.3.).

8 Conclusions and future work

This section presents the conclusion remarks of this thesis and some future research lines related to the outlined problems and proposals.

8.1 Conclusion remarks

This thesis has presented in-depth some singular supervised learning problems. In particular, imbalanced classification and monotonic classification are the two main problems addressed. The main objective of this thesis is the design of new solutions to address these two problems independently and in conjunction with other singular scenarios. Within these solutions, two different perspectives stand out which are represented with two sub-objectives. The first one is the design of robust classifiers for singular supervised problems. The second approach is the design of preprocessing-based techniques for monotonic and imbalance classification problems.

Regarding the first objective, this thesis includes several novel proposals based on robust classifiers such as ensembles for the two outlined problems.

The first proposal is the application of Random Forest for classification with monotonic constraints. This proposal constitutes a Random Forest with C4.5 decision trees and an ensemble pruning mechanism both guided by the non-monotonicity index to solve a common issue of monotonic classifiers. This problem is low precision due to a bias towards the prior knowledge constraints. The results of this empirical study have shown that the Random Forest proposal achieves high levels of both accuracy and monotonicity in monotonic classification tasks.

The next contribution is the SwitchingNED ensemble for learning highly imbalanced data. This ensemble exchanges some examples from the majority class to the minority class in each iteration. This switching procedure is guided by the Nearest Enemy Distance to prioritize the growth of minority class boundaries. As empirical results show, these mechanisms reduce the limitation of the original Switching algorithm and allow a better and more intelligent class distribution. As a result, SwitchingNED shows great robustness to highly imbalanced data.

The design of a monotonic Fuzzy kNN algorithm is the latest robust classifier presented for singular problems. This proposal aims at another major dilemma of monotonic classification: obtaining accurate monotonic predictions in the presence of monotonic noise. MonFkNN includes a class membership extraction for the training set aware of the relationships and violations of monotonicity. This procedure facilitates to mitigate the negative influence of monotonic inconsistencies. Besides, the mechanism of membership aggregation has been designed to promote the flexibility of the algorithm. The experiments conducted with this proposal show one of the best performances in terms of precision and monotonicity. Furthermore, MonFkNN empirically reveals high robustness to large amounts of noise.

The second group of approaches is based on preprocessing techniques in monotonic and imbalanced classification problems.

The first proposal of this group retakes the SwitchingNED ensemble to be combined with popular sampling techniques. SwitchingNED incorporates three different sampling techniques to facilitate balanced class distributions. In the experimental studies, the three combinations demonstrate different behaviors. The approach based on random under-sampling shows better performance on average. Compared to other ensembles, the latter combination also exhibits the best behavior in highly imbalanced scenarios.

The second research work of this group proposes several sampling techniques for monotonic imbalanced classification. These implement some good practices into the sampling priority based on the relevance of the examples for the data monotonicity. During the experimental study, these new sampling procedures show a significant increase in the accuracy and better preservation of the monotonicity of the monotonic classifiers.

As a final conclusion, the presented proposals concede to widely fulfill the objectives of this thesis. These proposals are not only excellent approaches for the faced singular supervised problems. They also address specific aspects of those problems.

Conclusiones

Esta tesis ha presentado en profundidad algunos problemas de aprendizaje supervisado singular. En particular, la clasificación desequilibrada y la clasificación monotónica son los dos principales problemas tratados. El objetivo principal de esta tesis es el diseño de nuevas soluciones para afrontar estos dos problemas de manera independiente y en conjunto a otros escenarios singulares. Dentro de estas soluciones, destacan dos vertientes diferentes presentadas como subobjetivos. El primero es el diseño de clasificadores robustos para problemas supervisados singulares. Por otra parte, la segunda aproximación es el diseño de técnicas basadas en preprocesamiento en los problemas de clasificación monotónica y desequilibrada.

En cuanto al primer objetivo, esta tesis incluye varias propuestas novedosas basadas en clasificadores robustos tales como los ensembles para los dos problemas planteados.

La primera propuesta es la aplicación de Random Forest para la clasificación con restricciones monotónicas. Esta propuesta constituye un Random Forest con árboles de decision C4.5 y un mecanismo de poda de ensemble ambos guiados por el índice de no monotonicidad para solucionar un problema común de los clasificadores monotónicos. Dicho problema es la baja precisión debido a un sesgo hacia las restricciones de conocimiento previo. Los resultados de este estudio empírico han mostrado que la propuesta Random Forest consigue altas cotas tanto de precisión como monotonicidad en tareas de clasificación monotónica.

La siguiente contribución es el ensemble SwitchingNED para el aprendizaje de datos altamente desequilibrados. Este ensemble intercambia algunos ejemplos de la clase mayoritaria a la minoritaria en cada iteración. Dicho procedimiento de intercambio está guiado por la distancia del enemigo más cercano para priorizar el crecimiento de las fronteras de la clase minoritaria. Tal como muestran los resultados empíricos, estos mecanismos reduce la limitación del algoritmo original Switching y permiten una mejor y más inteligente distribución de clases. Como resultado, SwitchingNED muestra una gran robustez a datos altamente desequilibrados.

El diseño de un algoritmo monotónico Fuzzy kNN es el último clasificador robusto presentando para problemas singulares. Esta propuesta tiene como objetivo otro de los importantes dilemas de la clasificación monotónica: la obtención de predicciones monotónicas y precisas en presencia de ruido monotónico. MonFkNN incluye una extracción de membresías de clase para el conjunto de entrenamiento consciente de las relaciones y violaciones de monotonicidad. Este procedimiento permite paliar la influencia negativa de las inconsistencias monotónicas. Además, el mecanismo de agregación de membresías ha sido diseñado para promover la flexibilidad del algoritmo. Los experimentos realizados con dicha propuesta exhiben uno de los mejores rendimientos en términos de precisión y monotonicidad. Asimismo, MonFkNN empíricamente revela una alta robustez a grandes cantidades de ruido.

El segundo grupo de aproximaciones está basada en técnicas de preprocesamiento en problemas de clasificación monotónica y desequilibrada.

La primera propuesta de este grupo retoma el ensemble SwitchingNED para ser combinado con técnicas de muestreo populares. SwitchingNED incorpora tres técnicas de muestreo diferentes para facilitar distribuciones de clase equilibradas. En los estudios experimentales, las tres combinaciones demuestran comportamientos diferentes. La propuesta basada en infra-muestreo aleatorio manifiesta un mejor rendimiento en media. En comparación con otros ensembles, esta última combinación también exhibe el mejor comportamiento en escenarios altamente desequilibrados.

El segundo trabajo de investigación de este grupo propone varias técnicas de muestreo para la clasificación monotónica y desequilibrada. Estas implementan una serie de buenas prácticas de prioridad de muestreo basadas en la relevancia de los ejemplos para la monotonicidad de los datos. Durante el

estudio experimental, estos nuevos procedimientos de muestreo muestran un incremento significativo de la precisión y una mejor preservación de la monotonicidad de los clasificadores monotónicos.

Como conclusión final, las diversas propuestas presentadas permiten cumplir ampliamente los objetivos de esta tesis. Dichas propuestas no solo son excelentes enfoques para los problemas supervisados singulares planteados. Además, estas abordan aspectos concretos de dichos problemas.

8.2 Future work

The proposals and the outcomes presented in this thesis dissertation enable new open challenges and research lines. This section presents some future research works related to the previously drawn conclusions:

- Switching based diversity promotion mechanism for other classification scenarios: Switching algorithm has proven good performance similar to other ensembles in the standard classification [72]. In this thesis, the SwitchingNED ensemble has improved those results for the imbalance classification. Therefore, an interesting line of research would be the implementation of SwitchingNED to standard multi-class classification problems. One would expect a better performance than the original Switching algorithm and competitive outcomes against other ensembles.

  Furthermore, the diversity promotion mechanisms of SwitchingNED and Switching algorithms could be combined with other diversity techniques to improve the performance of an ensemble. In the ensemble learning literature, there are plenty of examples of ensembles with different diversity promotion procedures. For example, Random Forest [65], Random Patches [104] and Extra-Trees [66] combine different types of random subspace and bootstrapping/sub-sampling. Gradient Boosting Machines have also incorporated other diversity mechanisms such as random subspace, random sub-sampling, or dropouts [105, 106, 107]. Thus, the inclusion of Switching based diversity promotion in other ensembles, such as Random Forest or Boosting, is a very attractive future research work.

  Besides, the Class Switching ensemble could be applied to other singular classification problems, such as monotonic classification. Class switching ensembles have been already designed with a successful performance for ordinal regression [108]. Then, Class Switching seems a very promising approach for classification with monotonicity constraints.

- Ensemble learning for monotonic imbalance classification: As stated along this thesis, ensemble learning proves great performance for imbalance classification problems [44, 26]. The combination of ensembles and sampling techniques are one of the most successful approaches in class imbalance scenarios [44]. Chain based sampling scheme has enabled these preprocessing techniques for monotonic classification. And, ensemble learning with monotonic Random Forest has also shown good results in monotonic tasks. Ensembles with chain based sampling techniques are very promising techniques for monotonic imbalance classification.

  Moreover, decomposition ensembles could be very interesting for both standard monotonic and monotonic imbalance scenarios. The majority of monotonic data-sets are multi-class problems. In standard multi-class classification, decomposition ensembles divide a complex multi-class problem into several simpler binary problems [63]. In monotonic classification, decomposition ensembles could be adapted to simplify the monotonicity constraints in each binary monotonic classifier. Then, the ensemble fusion mechanism must ensure the monotonicity constraints between the classes. These multi-class decomposition techniques are also very popular for imbalanced multi-class problems [109, 26]. Monotonic decomposition ensembles may be useful for sampling data in class pairs. One-vs-One and other more advanced decomposition scheme have not been adapted yet to monotonic classification [31].

- Fuzzy techniques for monotonic classification: Fuzzy based algorithms have achieved very promising outcomes as shown in the thesis and other research studies [48, 49]. Besides, fuzzy systems could be also helpful to properly model different elements of classification with monotonic constraints.

For example, they may be useful when representing different degrees of constraints between labels and features. That is, some input features could be more relevant than others regarding monotonicity. This problem definition would be very interesting for monotonic classifiers to define the importance of different monotonic violations. Additionally, the real distances between the different categories of ordinal input variables and classes are usually unknown in monotonic classification. Fuzzy techniques may help to quantify the category differences.

- Monotonic constraints in other singular learning problems: As seen in this thesis, several singular problems can coexist in a single real-life learning task. Monotonicity constraints may be present in the problem knowledge of other singular learning paradigms, such as semi-supervised learning [25] and crowdsourced learning [110].

  Certain evaluation data-sets commonly used in the monotonic classification originate from surveys or multiple-person evaluations, i.e., these are crowdsourced data. Labeling by different individuals rather than by an expert often facilitates data acquisition but also impairs data quality and the learning process. Currently, these monotonic classification data sets are being treated as standard sets. Crowd machine learning techniques may help to better treat these sets and better accommodate monotonic constraints.

- Big Data and Smart Data in monotonic classification: Nowadays, the generation of new data is incredibly massive and fast, which leads to serious computing problems during the data science process. This is known as Big Data problems. Big Data technologies aim to alleviate the high computing and storage requirements with cloud-based and distributed computing systems, such as Apache Spark [111, 112]. Another common difficulty of Big Data problems is the lack of control over data quality [5]. Therefore, preprocessing techniques have to be used to produce valid and clean data, also known as Smart Data [81, 5].

  Many data mining and preprocessing approaches have been redesigned with Big Data technologies for standard supervised problems [113, 114, 5]. However, there are none Big Data proposals for singular problems such as monotonic classification. The design of monotonic classifiers and monotonic preprocessing mechanism with Big Data techniques are very interesting and needed, but also challenging because the monotonic constraints are a global knowledge of a whole data-set.

Chapter II

Publications

1 Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity

- González, S., Herrera, F., & García, S. (2015). Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing*, 33(4), 367-388.
  - Status: **Published**
  - Impact Factor (JCR 2015): **0.533**
  - Subject Category: **Computer Science, Theory & Methods**
  - Rank: **91/105**
  - Quartile: **Q4**

MONOTONIC RANDOM FOREST WITH AN ENSEMBLE PRUNING MECHANISM BASED ON THE DEGREE OF MONOTONICITY

Sergio González
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
sergiogvz@decsai.ugr.es

Salvador García
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
salvagl@decsai.ugr.es

Francisco Herrera
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
herrera@decsai.ugr.es

ABSTRACT

In classification problems, it is very common to have ordinal data in the variables, in both explanatory and class variables. When the class variable should increase according to a subset of explanatory variables, the problem must satisfy the monotonicity constraint. It is well known that standard classification tree algorithms, such as CART or C4.5, are not guaranteed to produce monotonic trees, even if the data set is completely monotone. Recently, some classifiers have been designed to handle these kinds of problems. In decision trees, growing and pruning mechanisms have been updated to improve the monotonicity of the trees. In this paper we study the suitability of using these mechanisms in the generation of Random Forests. For this, we propose a simple ensemble pruning mechanism based on the degree of monotonicity of the resulting trees. The performance of several decision trees are evaluated through experimental studies on monotonic data sets. We deduce that the trees produced by the Random Forest also hold the monotonicity restriction but achieve a slightly better predictive performance than standard algorithms.

Keywords Monotonic Classification · Decision Tree Induction · Random Forest · Ensemble Pruning.

1 Introduction

The classification of examples in ordered categories is a popular problem which has drawn attention in data mining practitioners over the last years. This problem has been given with different names, such as ordinal classification, ordinal regression or ranking labelling, but all they share a common property in the data: the output attribute or class is ordinal. Classification with monotonicity constraints, also known as monotonic classification [1], is an ordinal classification problem where monotonic restriction is clear: a higher value of an attribute in an example, fixing other values, should not decrease its class assignment.

The monotonicity of relations between the dependent and explanatory variables is a very common prior knowledge form in data classification [2]. To illustrate, while considering a credit card application [3], an income between \$1000 to \$2000 may be considered a medium value income in a data set. If a customer A has a medium income, a customer B has a low income (i.e. less than \$1000) and the rest of the input attributes remain the same, there is a relationship of partial order between A and B: $B < A$. Considering that the application estimates lending quantities as output class, it is quite obvious that the loan that the system should give to customer B cannot be greater than that given to customer A. If so, a monotonicity constraint is violated in the decision.

Decision trees [4, 5, 6] and rule induction [7] constitute two of the most promising techniques to tackle monotonic classification. One important reason is that they build legible models by humans which can be easily implemented in real applications whose monotonicity property has not to be contravened. Any approach for classification can be integrated into an ensemble-type classifier, thus empowering the achieved performance [8]. However, in particular cases such as limitations in time or memory, contradictions and redundancy; a classifier selection is needed within the ensemble. It is called ensemble pruning [9, 10] and its main objective is to obtain a subset of the ensemble that solves the same classification problem with better performance.

Bagging is a well-known form of ensembles [11]. Random Forests (RFs) [12] add an additional layer of randomness to bagging. In addition to constructing each tree with a different bootstrap sample of the data, it also changes the construction process, splitting each node using the best among a subset of predictors randomly chosen at that node. Their main virtues are the robustness against over-fitting achieved and being very user friendly.

The general goals pursued in this paper are:

- To promote the application of the RF approach in monotonic classification tasks, aiming at increasing the accuracy for this problem while the decision trees obtained are as weakly monotonic as possible.
- To introduce a monotonicity ordering-based pruning mechanism for RFs based on the non-monotonicity index.
- To compare our proposal with other monotonic decision tree learning models, obtained by the application of the monotonic principle explained in Monotonic Induction Trees (MID) [13] for ID3. Since ID3 is obsolete, we apply MID in C4.5 [5], CART [4] and RankTree [14]. Also, two representative algorithms from other groups are involved in the comparison: OLM [15] and OSDL [16].

The experimental evaluation includes a total of 90 data sets, 50 of them are monotonic by using the relabelling approach proposed in [17] and the other 40 are standard classification data sets. Furthermore, the empirical study has been checked using non-parametrical statistical testing [18, 19, 20].

This paper is organized as follows. In Section 2 we present some background concepts: the ordinal classification with monotonic constraints and decision tree approaches for monotonic classification. Section 3 is devoted to describing our proposal of RF and its adaptation to satisfy the monotonicity constraints. Section 4 describes the experimental framework and examines the results obtained in the empirical study, presenting a discussion and analysis. Finally, Section 5 concludes the paper.

2 Background

As we mentioned before, the paper is devoted to the application of the RF algorithm to monotonic classification. Next, we will present a brief review of the monotonic classification including the description of decision trees for this problem in Subsections 2.1 and 2.2.

2.1 Monotonic classification

Ordinal classification problems are those in which the class is neither numeric nor nominal. Instead, the class values are ordered. For instance, a worker can be described as "excellent", "good" or "bad", and a bond can be evaluated as "AAA", "AA", "A", "A-", etc. Similar to a numeric scale, an ordinal scale has an order, but it does not posses a precise notion of distance. Ordinal classification problems are important, since they are fairly common in our daily life. Employee selection and promotion, determining credit rating, bond rating, economic performance of countries, industries and firms, and insurance underwriting, are examples of ordinal problem-solving in business. Rating manuscripts, evaluating lecturers, student admissions, and scholarship decisions for students, are examples of ordinal decision-making in academic life. Ordinal problems have been investigated in scientific disciplines such as information retrieval, psychology, and statistics for many decades [21] .

A monotonic classifier is one that will not violate monotonicity constraints. Informally, the monotonic classification implies that the assigned class values are monotonically nondecreasing (in ordinal order) with the attribute values. More formally, let $(X_i, class(X_i))$ denote a set of examples with attribute vector X_i and a class $class(X_i)$, respectively. Let $X_i \geq X_j$ indicate that all the attribute values of X_i are greater than or equal to those of X_j in some ordinal order, then $X_i \geq X_j \implies class(X_i) \geq class(X_j)$. A data set $(X_i, class(X_i))$ is monotonic if and only if all the pairs of examples i, j are monotonic with respect to each other [13].

Some monotonic ordinal classifiers require monotonic data sets to successfully learn, although there are others that are capable of learning from non-monotonic data sets as well.

2.2 Decision Tree Approaches for Monotonic Classification

Many data mining algorithms have been adapted to be able to handle monotonicity constraints in several ways. There are two steps to follow when dealing with monotonic classification problems. The first one is to preprocess the data [22] in order to "monotonize" the data set [23], rejecting the examples that violate the monotonic restrictions; and the second one is to force learning only monotone classification functions. Proposals of this type are: classification trees and rule induction [24, 25, 26], neural networks [27, 28, 29], and instance-based learning [15, 16, 17].

A monotone extension of ID3 (MID) was proposed by Ben-David [13] using an additional impurity measure for splitting, the total ambiguity score. However, the resulting tree may not be monotone anymore even when starting from a monotone data set. MID defines the *total-ambiguity-score* as the sum of the entropy score of ID3 and the order-ambiguity-score. This last score is defined in terms of the non-monotonicity index of the tree, which computes the number of pair branches that are non-monotonic regarding the total possible non-monotonic pairs there may be.

Makino et al. [30] proposed a monotone (or positive) decision tree (P-DT) and a quasi-monotone (quasi-positive) decision tree (QP-DT) extension of ID3 in the two-class setting. They start from a monotone training set and demand, in the case of QP-DT, that monotonicity is (only) guaranteed on this training set, while in the case of P-DT the tree (or equivalently, the derived rule base) is required to be monotone. These methods have been nontrivially extended in [31, 24] to the multi-class problem and accommodating continuous attributes. In addition to the fact that these approaches start from a monotone training set, the main technique for guaranteeing (quasi-)monotonicity is by adding at each step, if necessary, new data generated from the data in the previous step.

A splitting criteria thought for monotonic classification has been proposed. For instance, in [25], the criterion aims are reducing the numbers of non-monotone pairs of points in the resulting branches. It chooses the split with the least number of inconsistencies/conflicts. Another way to achieve monotone classification models in a post-processing step is by pruning classification trees [32]. This method prunes the parent of the non-monotone leaf that provides the largest reduction in the number of non-monotonic leaves' pairs. Here, similar accuracy is reported, with increased comprehensibility.

One of the most popular decision trees for ordinal classification is RankTree [14]. It extended the Gini impurity used in CART to ordinal classification, called ranking impurity. Although RankTree enhances the capability of extracting ordinal information, it cannot guarantee the construction of a monotone tree, even if the training data are monotone.

As for explicit monotonic trees, we can find some representatives proposed in the literature. MDT [33] aimed to predict the implicit ordering in terms of pair comparison in the original classification. In [26] the authors propose a rank generalization of Shannon mutual information, namely rank mutual information, and underline that this measure is both sensitive to monotonicity and robust to noisy data. Then, this measure is used to build binary tree classifiers guaranteed to possess a weak form of monotonicity (rule monotonicity) in the case the starting data set is monotone consistent. They call this algorithm REMT and show that it behaves well compared to both monotone and non-monotone classifiers. An extension of the interval valued attributes decision tree to deal with monotonic classification is given in [34], which selects extended attributes by minimizing rank mutual information to generate a decision tree. Recently, in [35], the authors presented a binary tree classifier RDMT(H) parametrized by a discrimination measure H used for splitting and another three pre-pruning parameters. According to them, RDMT(H) guarantees a weak form of monotonicity on the resulting tree.

In this paper, we include the impurity measure proposed in MID [13] in two classical decision trees: CART and C4.5, altering these classifiers to be adapted to monotonic classification. Furthermore, we combine the RankTree proposal with the same impurity measure, also obtaining a monotonic version of RankTree. Thus, in summary, we will compare RFs with three algorithms: MID-C4.5, MID-CART and MID-RankTree.

3 Monotonic Random Forest

In this section, we explain our proposal to tackle monotonic classification. We will start by illustrating the different modifications added to the traditional RF. Then, we will continue with our main contribution, the monotonicity ordering-based pruning mechanism, which allow us to increase the accuracy while maintaining the monotonicity constraints. Finally, we will justify the necessity of this method and how it is applied.

The modifications introduced to the standard RF are mainly focused on the way the splitting is made for every tree, the promotion of the diversity by a new random factor and the aggregation of the results with the pruning mechanism proposed, maintaining the bootstrap sample method untouched.

First of all, we define the Non Monotonic Index (NMI) as the rate of number of violations of monotonicity divided by the total number of examples in a data set. Previously, we have introduced the MID based algorithms in the process of building the trees. With this change, we accomplish the initial objective of adapting the well-known ensemble to monotonic classification. We choose MID-C4.5 to build every random tree of the forest. This method selects the best attribute to perform the split using the $total\text{-}ambiguity$-score as a criterion. This measurement was defined by Ben-David in [13] as the sum of the E-score of the ID3 algorithm and the $order\text{-}ambiguity$-score weighted by the parameter R. The $order - ambiguity - score$ is computed, as shown in Equation 1, using the concept of the non-monotonicity index, which is the ratio between the actual number of

Algorithm 1 Monotonic RF algorithm.

function MONRF(D - dataset, $nTrees$ - number of random trees built, R_{limit} - importance factor for monotonic constrains, T - Threshold used in the pruning procedure, S - the predicted version of D)
  initialize: $S = \{\}$, $Trees[1..nTrees]$, $D_{bootstraps}[1..nTrees]$, $NMIs[1..nTrees]$
  for i in [1,$nTrees$] **do**
    $D_{bootstraps}[i] = Bootstrap\_Sampler(nTrees, D)$
    $rand = Random(1, R_{limit})$
    $Trees[i] = Build\_Tree(D_{bootstraps}[i], rand)$
    $NMIs[i] = Compute\_NMI(Trees[i])$
  end for
  $Trees = Sort(Trees, NMIs)$
  for i in [1,$\lceil nTrees * T \rceil$] **do**
    $\widehat{Trees} \leftarrow Trees[i]$
  end for
  for d in D **do**
    $S \leftarrow Predict\_Majority\_Voting(\widehat{Trees}, d)$
  end for
  return S
end function

non-monotonic branch pairs and the maximum number of pairs that could have been non-monotonic. In the MID-C4.5, the entropy of the ID3 is substituted by the gain information of the C4.5 decision tree.

$$A = \begin{cases} 0 & \text{if } NMI = 0 \\ -(\log_2 NMI)^{-1} & \text{otherwise} \end{cases} \quad (1)$$

The factor R was first introduced by Ben-David[13] as an importance factor of the *order-ambiguity-score* in the decision of the splitting with the calculation of the *total-ambiguity-score*. As higher as R was set, more relevant were the monotonicity constraints considered. We use this parameter as a way to further randomise and diversify the different trees built in the RF and at the same time, we force the tree building procedure to be dominated by the monotonicity considerations. In order to fulfill this, each tree is built from the beginning with a different factor R, picked as a random number from 1 to R_{limit}, set as a parameter shown in Algorithm 1.

Furthermore, we did not consider for our proposal the maximum depth imposed to all the random trees of the standard RF. We have decided this, due to the fact that monotonic decision tree classifiers already highly reduce the complexity of the built tree compared with the traditional ones.

Finally, we design a pruning threshold mechanism in the final combination of the different results to predict the class of each example. Instead of using all the decision trees built, to form the class through the majority vote of the predictions, we choose the best trees in term of monotonicity constraints within a certain threshold, latest lines of the Algorithm 1. With this objective, our Monotonic RF sorts the different trees built by the Non-Monotonic-Index in increasing order and the pruning method selects the first n trees, where n is the number of trees computed by product of the total number of trees built and the threshold T within the range (0,1]. We recommend to set it at 0.50, due to the results obtained in the next section.

It is highly important to understand the need for this pruning method and why without it, the result obtained would not be as good. By introducing randomness in the bootstrap selection of training data and in the attributes used to split the trees, we are obstructing, in a certain way, the possibility of obtaining real good monotonic trees, even though we can obtain better results in terms of accuracy. Therefore, it is necessary to find a way of balancing these two objectives with our pruning threshold proposal.

4 Experimental Framework, Results and Analysis

In this section, we present the experimental framework followed to compare and analyze the application of RFs to monotonic classification.

4.1 Experimental Methodology

The experimental methodology is described next by specifying some basic elements:

- Data Sets: A total number of 90 data sets are used in the study, 50 monotonic data sets and other 40 standard classification problems. Their main characteristics are described in Table 1. Most of the monotonic data sets are standard data sets used in the classification scope and extracted from the UCI and KEEL repositories [36, 37] which have been relabeled following the procedure used in the experimental design of [17]. We must point out that this relabeling process applied to the original data sets transforms them into monotonic data sets, changing their class distribution and complexity but maintaining the number of attributes and examples. Four classical monotonic data sets are also included [15]: *ERA*, *ESL*, *LEV*, *SWD*.

  Table 1 shows the name of the data sets and their number of instances, attributes and classes. The total number of instances of the original data set appears in parenthesis. Sometimes, this number differs because of the missing values, which have been ignored in this study.

- Algorithms to compare: We will compare RFs with three decision trees: MID-C4.5, MID-CART and MID-RankTree; and two classical algorithms in this field: OLM and OSDL. These algorithms were described in Subsection 2.2.

- Evaluation metrics: Several measures will be used in order to estimate the performance of the algorithms compared.
  - Accuracy (Acc): is the number of successful hits relative to the total number of classifications. It has been by far the most commonly used metric for assessing the performance of classifiers for years [38].
  - Mean Absolute Error (MAE): is calculated summing the absolute values of the errors and then dividing it by the number of classifications. It has been selected considering the studies of Gaudette and Japkowicz [39] and Japkowicz and Shah [40], which conclude that MAE is one of the best performance metrics in ordered classification.
  - Non Monotonic Index (NMI): explained above, is computed as the rate of number of violations of monotonicity divided by the total number of examples in a data set [13].
  - Number of Leaves/Branches (NL): is the number of possible paths from the root to each leaf of the tree. It corresponds with the number of rules which can be extracted from the decision tree [6].

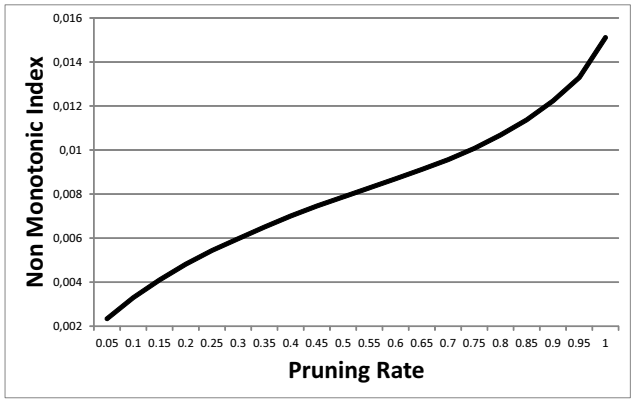
- Parameters configuration: Table 2 depicts the parameters used by the algorithms considered in the study. The choice of parameters has been done according to the standards and recommendations given by the authors in the original proposal papers. Due to the fact that the experimental evaluation comprises a high number of data sets, it is unreasonable to adjust each parameter individually for each data set. In fact, our purpose is just the opposite; we aim at comparing them in the most general scenario possible. We have not performed any tuning to adapt these parameters, because our objective is not to maximize the accuracy or any other performance metric, but to fairly compare the algorithms and their robustness in a common environment and upon different data sets. The run of the algorithms has been carried out following a ten fold cross validation schema (10-fcv) three times.

Table 1: Description of the 90 data sets used in the study. Super indexes: * denotes both monotonic and non-monotonic version, + denotes only monotonic version and # denotes only non-monotonic version

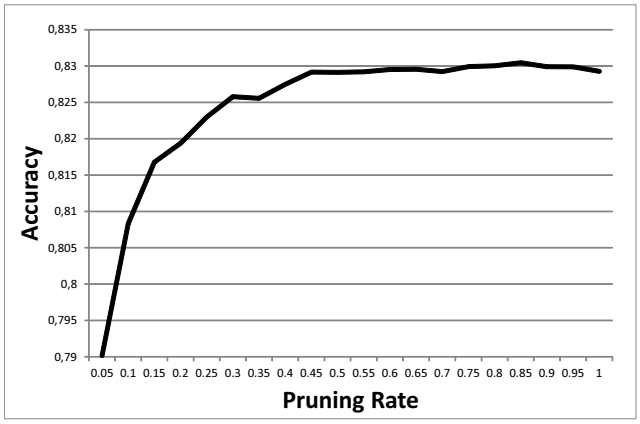
Data set	Ins.	At.	Cl.	Data set	Ins.	At.	Cl.
appendicitis*	106	7	2	led7Digit*	500	7	10
australian*	690	4	2	LEV+	1000	4	5
automobile*	150 (205)	25	6	lymphography*	148	18	4
auto-mpg+	392	7	4	machinecpu+	209	6	4
balance*	625	4	3	mammographic*	830 (961)	5	2
bands#	365 (539)	19	2	monk-2*	432	6	2
bostonhousing+	506	12	4	movement_libras+	360	90	15
breast*	277 (286)	9	2	newthyroid*	215	5	3
bupa+	345	6	2	phoneme#	5404	5	2
car*	1728	6	4	pima*	768	5	2
cleveland*	297 (303)	13	5	post-operative*	87 (90)	8	3
contraceptive*	1473	9	3	saheart*	462	9	2
crx*	653 (690)	15	2	segment+	2310	19	7
dermatology+	358 (366)	34	6	sonar+	208	60	2
ecoli*	336	7	8	spectfheart+	267	4	2
ERA+	1000	4	9	SWD+	1000	10	4
ESL+	488	4	9	tae*	151	5	3
flare*	1066	11	9	titanic*	2201	3	2
german#	1000	20	2	vehicle*	846	18	4
glass*	214	9	7	vowel*	990	13	11
haberman*	306	3	2	wdbc*	569	30	2
hayes-roth*	160	4	3	windorhousing+	546	11	2
heart*	270	13	2	wine*	178	13	2
hepatitis*	80 (155)	19	2	winequality-red#	1599	11	11
housevotes*	232(245)	16	2	wisconsin*	683 (699)	9	2
ionosphere*	351	33	2	yeast*	1484	10	8
iris*	150	4	3	zoo*	101	16	7

Table 2: Parameters considered for the algorithms compared.

Algorithm	Parameters
C4.5	itemsetsPerLeaf = 2; confidence = 0.25
CART	maxDepth = 90
RankTree	maxDepth = 90
MID	R = 1
Random Forest	nTrees = 100; R_{limit} = 100; Threshold = 0.50
	mAttributes = $\sqrt{TotalAttributes}$



(a) Average NMI of the Random Tree depending on the pruning rate.



(b) Average accuracy of the Random Tree depending on the pruning rate.

Figure 1: Effect of the pruning rate in the Random Forest

- Statistical procedures: Several hypothesis testing procedures are considered to determine the most relevant differences found between the methods. To address this, the use of nonparametric tests will be preferred to parametric ones, since the initial conditions that guarantee the reliability of the latter may not be satisfied, causing the statistical analysis to lose credibility. We use the Friedman ranks test [18, 19], to contrast the behaviour of each algorithm. It is used to highlight the existence of significant differences between methods. In a later stage, post-hoc procedures like Holm will find out which algorithms are distinctive among the $1 \times n$ comparisons performed [20].

4.2 Results

This section is dedicated to present the results gathered from the runs of the algorithms using the configuration described in the previous subsection.

First of all, we present the study that allows us to determine the best choice of the monotonicity pruning parameter value for the RF proposal. For this, we employ all the monotonic data sets specified in Table 1 running RF over them. The trees built from the ensemble are sorted by their NMI in increasing order. The pruning mechanism selects the trees using a threshold coming from 0.05 to 1. This represents the rate of trees that will belong to the ensemble. In this way, if the rate is 1 all the trees will belong to the ensemble and if the rate is 0.3, only 30% of the most monotonic trees will form the ensemble.

Figure 1 shows the effect of the pruning rate explained above in RF. The values represented for both graphics are associated with the average values of Acc and NMI of the 50 data sets. Observing Figure 1a, we can see that there is a turning point in the growth curve surrounding the rate value of 0.5. Simultaneously, in Figure 1b there is a limit in which the improvement registered in accuracy stops decreasing and this limit matches with the same turning point indicated previously: the rate of 0.5. Hence, it seems logical that this rate could be an interesting value to be adopted as the monotonicity pruning rate used for RFs.

Henceforth, we will consider 0.5 as the pruning rate used in RFs; that means that the most monotonic half of the trees belonging to the forest will be used in the classification stage and the other half will be discarded. The effect of this choice can be observed in Tables 3 and 4, where the results obtained by MID-RF are reported over monotonic and non-monotonic data sets respectively. Three configurations have been studied: (1) MID-RF with enable pruning mechanism, MID-RF without pruning and MID-RF with the R parameter fixed to 50 instead of using random numbers from 1 to R_{limit} for each tree.

Furthermore, under our recommended configuration of pruning and random choice of R parameter, we compare RF with the other three considered decision tree methods and two representative algorithms from other groups such as OLM [15] and OSDL [16]. Tables 5 and 6 exhibit the results obtained for the algorithms over monotonic data sets, in terms of average values of the three runs of 10-fcv.

In addition, Tables 7 and 8 present the results obtained for the algorithms over non monotonic data sets.

In order to support the results, we include a statistical analysis based on non parametric tests. The results obtained by the application of the Friedman test and the Holm post-hoc procedure are depicted in Table 9. This table shows the rankings obtained by using the Friedman test for each algorithm, measure and group (monotonic and non monotonic data sets). Furthermore, the Adjusted P-Value (APV) [20] computed by the Holm procedure is reported in the algorithms whose ranking is not the best in each group. This represents the p-value adjusted for the group associated with the comparison between the control algorithm (the one with the lowest ranking) and the algorithm in question. If this p-value is lower than 0.10, the differences between both algorithms are statistically significant.

4.3 Analysis

This section is devoted to studying the behaviour of the proposed ensemble compared to the other algorithms included in the study, with the ultimate objective of analysing the results and determining whether or not our proposal is valuable. Conclusions will be extracted in terms of the four different measurements that were used in the study.

From this study, we may stress the following conclusions:

- First of all, we focus our attention in the results reported in Tables 3 and 4, where the MID-RF algorithm is analyzed according to whether or not pruning is used and the effects on fixing the R parameter instead of using a random value for each decision tree. The statistical analysis conducted by the Friedman test indicates that there are no significant differences in accuracy and MAE for both monotonic and non-monotonic data sets (p-values equal to 0.835 and 0.869 for monotonic data sets, 0.219 and 0.294 for non-monotonic data sets, respectively). However, considering NMI, the pruning mechanism strongly influences in obtaining more monotonic models regarding to not to use it (p-value equals to 0.000 in both cases). Hence, NMI based pruning is useful to monotonize models without decreasing accuracy.

- The R parameter does not produce significant differences whether it is fixed or randomly chosen. Hence, it is safe to ensure that the R parameter has little influence in the results obtained when the pruning mechanism is enabled.

- With respect to the comparison with other learning algorithms, in terms of accuracy, the goodness of the Monotonic RF the with pruning threshold mechanism is clear. In all cases and, both for monotonic and non-monotonic data sets; the RF outperforms the other 5 algorithms by a significant difference, a fact that can be noticed in Tables 5, 7 and 9, where the p-value is smaller than 0.10. This result is obtained thanks to the potential of the RF scheme, which can accomplish really accurate and robust results, through the combination of diverse trees.

- With the same results, the superiority of RF in relation to the MAE over the other algorithms is overwhelming. This outcome was expected, when such a difference in terms of accuracy was obtained. It is worth noting how robust the results are as they barely change from the monotonic to the non-monotonic data.

- Furthermore, RF succeeds to obtain less complex trees, as can be seen with a smaller number of leaves for both types of data, in Tables 6 and 8. A remarkable fact keeping in mind that the maximum depth of

Table 3: Results reported over monotonic data sets of MID-RF with the parameter R fixed, with and without pruning.

	Accuracy			**Mean Absolute Error**			**Non Monotonic Index**		
	Pruning	**No Pruning**	**With R=50**	**Pruning**	**No Pruning**	**With R=50**	**Pruning**	**No Pruning**	**With R=50**
appendicitis	0.8667	**0.8818**	0.8636	0.1333	**0.1182**	0.1364	0.0352	0.1059	**0.0350**
australian	0.8261	0.8251	**0.8290**	0.1739	0.1749	**0.1710**	0.0120	0.0344	**0.0118**
auto-mpg	**0.6529**	0.6504	0.6513	0.4630	0.4654	0.4603	**0.0006**	0.0008	**0.0006**
automobile	0.8039	0.7957	**0.8164**	0.3069	0.3129	**0.2988**	**0.0003**	0.0008	**0.0003**
balance	0.9830	0.9793	**0.9835**	**0.0186**	0.0229	**0.0186**	0.0211	0.0396	**0.0210**
bostonhousing	0.6483	**0.6528**	0.6403	0.4958	**0.4882**	0.5118	**0.0004**	0.0005	**0.0004**
breast	**0.7597**	0.7537	**0.7597**	**0.2403**	0.2463	**0.2403**	**0.0001**	0.0002	**0.0001**
bupa	0.7981	0.7981	**0.7990**	0.2019	0.2019	**0.2010**	**0.0050**	0.0129	**0.0050**
car	0.8731	**0.8887**	0.8731	0.1609	**0.1449**	0.1609	**0.0001**	**0.0001**	**0.0001**
cleveland	0.5644	**0.5690**	0.5567	0.6893	**0.6747**	0.7228	**0.0009**	0.0010	**0.0009**
contraceptive	0.8185	0.8181	**0.8228**	0.2351	0.2358	**0.2263**	**0.0001**	**0.0001**	**0.0001**
crx	0.8290	**0.8321**	**0.8321**	0.1710	**0.1679**	**0.1679**	**0.0001**	**0.0001**	**0.0001**
dermatology	0.8633	0.8614	**0.8642**	**0.2810**	0.2857	0.2856	**0.0014**	0.0021	**0.0014**
ecoli	**0.6441**	0.6439	0.6421	1.0802	1.0999	**1.0740**	**0.0012**	0.0021	**0.0012**
ERA	**1.0000**	**1.0000**	**1.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0021**	0.0023	**0.0021**
ESL	0.9043	0.9043	**0.9050**	0.1107	**0.1100**	**0.1100**	**0.0009**	0.0011	**0.0009**
flare	**0.9025**	0.8981	**0.9025**	**0.1256**	0.1300	**0.1256**	**0.0002**	0.0005	**0.0002**
glass	0.7464	**0.7492**	0.7465	**0.4934**	0.5065	0.5150	**0.0026**	0.0035	**0.0026**
haberman	**0.9291**	0.9248	0.9280	**0.0709**	0.0752	0.0720	**0.0082**	0.0132	**0.0082**
hayes-roth	0.9042	**0.9125**	0.9083	0.1104	**0.0958**	**0.1063**	0.0074	0.0102	**0.0073**
heart	0.8235	**0.8296**	**0.8296**	0.1765	**0.1704**	**0.1704**	**0.0025**	0.0039	**0.0025**
hepatitis	0.8917	**0.8958**	0.8875	0.1083	**0.1042**	0.1125	0.0261	0.0546	**0.0260**
housevotes	0.9528	**0.9584**	0.9528	0.0472	**0.0416**	0.0472	**0.0073**	0.0152	**0.0073**
ionosphere	0.8832	0.8822	**0.8861**	0.1168	0.1178	**0.1139**	**0.0042**	0.0096	**0.0042**
iris	**0.9711**	0.9689	0.9644	**0.0289**	0.0311	0.0356	0.0487	0.0924	**0.0481**
led7digit	**0.8600**	0.8360	**0.8600**	**0.3800**	0.4607	**0.3800**	**0.0040**	0.0059	**0.0040**
LEV	**0.9993**	**0.9993**	**0.9993**	**0.0007**	**0.0007**	**0.0007**	**0.0009**	0.0010	**0.0009**
lymphography	0.7819	**0.7978**	0.7886	0.2314	**0.2133**	0.2225	**0.0013**	0.0019	**0.0013**
machinecpu	**0.6520**	0.6280	0.6504	**0.4741**	0.5062	0.4821	**0.0017**	0.0022	**0.0017**
mammographic	**0.9763**	0.9743	**0.9763**	**0.0237**	0.0257	**0.0237**	**0.0072**	0.0161	**0.0072**
monk-2	0.9807	0.9800	**0.9815**	0.0193	0.0200	**0.0185**	0.0184	0.0365	**0.0183**
movement_libras	0.6796	**0.6917**	0.6769	1.1602	**1.0880**	1.2278	**0.0005**	0.0006	**0.0005**
newthyroid	0.8621	0.8620	**0.8637**	0.1905	0.1906	**0.1873**	**0.0098**	0.0165	**0.0098**
pima	0.8702	0.8711	**0.8724**	0.1298	0.1289	**0.1276**	0.0014	0.0021	**0.0013**
post-operative	**0.6968**	**0.6968**	**0.6968**	0.3773	**0.3736**	0.3773	**0.0040**	0.0056	**0.0040**
saheart	0.7302	**0.7410**	0.7403	0.2698	**0.2590**	0.2597	**0.0020**	0.0043	**0.0020**
segment	**0.9759**	0.9749	0.9756	**0.0447**	0.0475	0.0468	**0.0004**	0.0006	**0.0004**
sonar	0.8042	**0.8123**	0.8090	0.1958	**0.1877**	0.1910	**0.0063**	0.0109	0.0064
spectfheart	0.8028	0.8065	**0.8076**	0.1972	0.1935	**0.1924**	0.0047	0.0115	**0.0046**
SWD	0.9993	**1.0000**	0.9990	0.0007	**0.0000**	0.0010	**0.0004**	**0.0004**	**0.0004**
tae	**0.8278**	0.8190	**0.8278**	**0.1921**	0.2051	**0.1921**	**0.0061**	0.0100	**0.0061**
titanic	**1.0000**	**1.0000**	**1.0000**	**0.0000**	**0.0000**	**0.0000**	0.1163	0.1812	**0.1161**
vehicle	**0.7409**	0.7378	0.7315	**0.4929**	0.5024	0.5158	**0.0002**	0.0003	**0.0002**
vowel	0.9525	**0.9552**	0.9481	0.1010	**0.0980**	0.1219	**0.0001**	**0.0001**	**0.0001**
wdbc	0.7183	0.7171	**0.7247**	0.2817	0.2829	**0.2753**	**0.0013**	0.0034	**0.0013**
windsorhousing	0.8932	**0.8944**	0.8889	0.1068	**0.1056**	0.1111	**0.0015**	0.0021	**0.0015**
wine	0.7926	**0.7981**	0.7944	0.2882	**0.2791**	0.2827	**0.0062**	0.0106	**0.0062**
wisconsin	0.9747	**0.9752**	0.9747	0.0253	**0.0248**	0.0253	**0.0075**	0.0112	**0.0075**
yeast	0.4095	**0.4165**	0.4059	1.7143	**1.7041**	1.7517	**0.0000**	**0.0000**	**0.0000**
zoo	**0.7427**	**0.7427**	**0.7427**	0.7000	**0.6900**	0.7000	**0.0090**	0.0161	**0.0090**

Table 4: Results reported over non-monotonic data sets of MID-RF with the parameter R fixed, with and without pruning.

	Accuracy			Mean Absolute Error			Non Monotonic Index		
	Pruning	No Pruning	With R=50	Pruning	No Pruning	With R=50	Pruning	No Pruning	With R=50
appendicitis	0.8791	**0.8821**	**0.8821**	0.1209	**0.1179**	**0.1179**	**0.0298**	0.1235	**0.0298**
australian	0.8725	**0.8758**	0.8734	0.1275	**0.1242**	0.1266	0.0007	0.0012	**0.0006**
automobile	0.8045	**0.8068**	0.7934	0.2650	**0.2609**	0.2790	**0.0003**	0.0008	**0.0003**
balance	0.8634	**0.8639**	0.8618	**0.1868**	0.1899	0.1904	**0.0006**	0.0007	**0.0006**
bands	**0.7137**	0.6903	0.7064	**0.2863**	0.3097	0.2936	**0.0016**	0.0102	0.0017
breast	0.7327	**0.7385**	0.7327	0.2673	**0.2615**	0.2673	**0.0001**	0.0002	**0.0001**
car	0.8609	**0.8764**	0.8609	0.1919	**0.1674**	0.1919	**0.0001**	**0.0001**	**0.0001**
cleveland	0.5745	0.5724	**0.5936**	0.6450	0.6641	**0.6184**	**0.0009**	0.0011	**0.0009**
contraceptive	0.5493	**0.5504**	0.5452	0.6618	**0.6569**	0.6684	**0.0001**	**0.0001**	**0.0001**
crx	0.8651	0.8675	**0.8677**	0.1349	0.1325	**0.1323**	**0.0001**	0.0002	**0.0001**
ecoli	0.8403	**0.8433**	0.8384	**0.5611**	0.5708	0.5667	**0.0026**	0.0057	0.0027
flare	0.7502	**0.7546**	0.7502	0.4964	**0.4874**	0.4964	**0.0002**	0.0003	**0.0002**
german	0.7420	**0.7480**	0.7423	0.2580	**0.2520**	0.2577	**0.0000**	**0.0000**	**0.0000**
glass	0.7613	**0.7719**	0.7600	0.4890	**0.4716**	0.5049	**0.0028**	0.0037	**0.0028**
haberman	0.7304	**0.7370**	0.7304	0.2696	**0.2630**	0.2696	0.0082	0.0576	**0.0078**
hayes-roth	0.7688	**0.7875**	0.7708	0.2479	**0.2250**	0.2438	**0.0055**	0.0081	**0.0055**
heart	**0.8346**	0.8284	0.8321	**0.1654**	0.1716	0.1679	**0.0025**	0.0039	**0.0025**
hepatitis	**0.8710**	0.8646	**0.8710**	**0.1290**	0.1354	**0.1290**	**0.0245**	0.0528	0.0247
housevotes	0.9472	**0.9654**	0.9472	0.0528	**0.0346**	0.0528	**0.0097**	0.0264	**0.0097**
ionosphere	0.9299	**0.9393**	0.9298	0.0701	**0.0607**	0.0702	**0.0057**	0.0165	0.0058
iris	0.9533	0.9533	**0.9556**	0.0467	0.0467	**0.0444**	**0.0406**	0.0843	0.0412
led7digit	**0.7127**	0.7087	**0.7127**	**1.2300**	1.2353	**1.2300**	**0.0033**	0.0050	**0.0033**
lymphography	**0.8309**	0.8230	0.8259	0.1812	**0.1810**	0.1844	**0.0016**	0.0026	**0.0016**
mammographic	0.8360	**0.8420**	0.8368	0.1640	**0.1580**	0.1632	**0.0027**	0.0058	**0.0027**
monk-2	0.9757	0.9742	**0.9765**	0.0243	0.0258	**0.0235**	0.0044	0.0236	**0.0043**
newthyroid	**0.9618**	**0.9618**	**0.9618**	0.0628	**0.0598**	0.0628	**0.0237**	0.0454	0.0239
phoneme	**0.8824**	0.8772	0.8816	**0.1176**	0.1228	0.1184	**0.0001**	0.0002	**0.0001**
pima	**0.7549**	0.7527	**0.7549**	**0.2451**	0.2473	**0.2451**	**0.0014**	0.0045	**0.0014**
post-operative	0.6972	**0.7051**	0.6972	0.3139	**0.3060**	0.3139	**0.0046**	0.0068	**0.0046**
saheart	**0.7071**	0.7064	0.7042	**0.2929**	0.2936	0.2958	**0.0031**	0.0144	**0.0031**
tae	**0.5904**	0.5792	0.5882	0.5311	0.5335	**0.5289**	**0.0045**	0.0123	**0.0045**
titanic	0.7862	**0.7865**	0.7862	0.2138	**0.2135**	0.2138	**0.0199**	0.0290	**0.0199**
vehicle	0.7530	0.7514	**0.7541**	0.4723	0.4754	**0.4625**	**0.0003**	0.0004	**0.0003**
vowel	0.9525	**0.9552**	0.9481	0.1010	**0.0980**	0.1219	**0.0001**	**0.0001**	**0.0001**
wdbc	0.9561	0.9555	**0.9596**	0.0439	0.0445	**0.0404**	**0.0036**	0.0114	0.0037
wine	0.9792	**0.9810**	0.9792	0.0208	**0.0190**	0.0208	0.0064	0.0294	**0.0062**
winequality-red	0.6825	**0.6831**	0.6827	**0.3509**	0.3513	0.3527	**0.0000**	0.0001	**0.0000**
wisconsin	0.9714	**0.9734**	0.9719	0.0286	**0.0266**	0.0281	**0.0056**	0.0088	**0.0056**
yeast	0.6144	**0.6220**	0.6106	0.7689	**0.7547**	0.7803	**0.0000**	**0.0000**	**0.0000**
zoo	**0.9583**	0.9541	**0.9583**	**0.0667**	0.0726	**0.0667**	**0.0144**	0.0277	**0.0144**

Table 5: Accuracy and MAE results reported over monotonic data sets.

	Accuracy						**Mean Absolute Error**					
	MID RF	**MID C4.5**	**MID CART**	**MID RankTree**	**OLM**	**OSDL**	**MID RF**	**MID C4.5**	**MID CART**	**MID RankTree**	**OLM**	**OSDL**
appendicitis	0.8667	0.8964	0.8864	**0.9064**	0.8109	0.6418	0.1333	0.1036	0.1136	**0.0936**	0.1891	0.3582
australian	0.8261	0.8029	0.8319	**0.8362**	0.7232	0.8319	0.1739	0.1971	0.1681	**0.1638**	0.2768	0.1681
auto-mpg	0.6529	0.6247	0.4489	0.6223	**0.6812**	0.3854	0.4630	0.4851	1.0335	0.5001	**0.3879**	0.8291
automobile	0.8039	**0.8250**	0.7304	0.7429	0.2333	0.3758	0.3069	**0.2688**	0.4696	0.4263	2.1046	0.8958
balance	0.9830	0.9777	0.9777	**0.9856**	0.9776	0.9777	0.0186	0.0239	0.0271	**0.0176**	0.0272	0.0271
bostonhousing	**0.6483**	0.5674	0.4982	0.5237	0.3003	0.2569	**0.4958**	0.6102	0.7504	0.6856	1.3045	1.0099
breast	0.7597	0.7337	0.6933	0.7440	**0.8409**	0.8015	0.2403	0.2663	0.3067	0.2560	**0.1591**	0.1985
bupa	0.7981	0.7508	0.7534	0.7879	**0.8375**	0.7625	0.2019	0.2492	0.2466	0.2121	**0.1625**	0.2375
car	0.8731	0.9433	0.8183	0.9386	**0.9705**	**0.9705**	0.1609	0.0666	0.2396	0.0735	**0.0324**	**0.0324**
cleveland	0.5644	0.4909	0.5284	0.5253	**0.5793**	0.5421	**0.6893**	0.8332	0.8014	0.8586	0.8311	0.7848
contraceptive	0.8185	0.7991	0.5601	0.7719	**0.8799**	0.8398	0.2351	0.2552	0.6449	0.2844	**0.1534**	0.1602
crx	**0.8290**	0.7903	0.7933	0.7839	0.6110	0.7058	**0.1710**	0.2097	0.2067	0.2161	0.3890	0.2942
dermatology	**0.8633**	0.8437	0.8408	0.7512	0.4499	0.1593	**0.2810**	0.3325	0.3465	0.5339	1.3821	1.6421
ecoli	**0.6441**	0.6074	0.5750	0.5779	0.6368	0.0652	1.0802	1.0549	1.5201	1.1250	**0.9467**	2.1998
ERA	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
ESL	0.9043	0.9159	0.6162	0.9344	0.9179	**0.9364**	0.1107	0.1026	0.5788	0.0738	0.0923	**0.0656**
flare	0.9025	0.9165	0.6380	0.9456	**0.9738**	0.9606	0.1256	0.1191	0.6479	0.0826	**0.0318**	0.0572
glass	**0.7464**	0.6773	0.6258	0.6883	0.3175	0.3223	**0.4934**	0.6929	0.7747	0.6747	1.7994	1.8000
haberman	0.9291	0.9177	0.9312	0.9537	0.9310	**0.9606**	0.0709	0.0823	0.0688	0.0463	0.0690	**0.0394**
hayes-roth	0.9042	0.9438	0.7688	0.8500	**0.9500**	0.9438	0.1104	**0.0563**	0.2500	0.1688	0.0750	**0.0563**
heart	**0.8235**	0.7926	0.7593	0.8111	0.6704	0.6259	**0.1765**	0.2074	0.2407	0.1889	0.3296	0.3741
hepatitis	0.8917	0.7750	0.8375	**0.9000**	0.2375	0.8000	0.1083	0.2250	0.1625	**0.1000**	0.7625	0.2000
housevotes	0.9528	**0.9741**	0.8750	0.9266	0.9047	0.9096	0.0472	**0.0259**	0.1250	0.0734	0.0953	0.0904
ionosphere	**0.8832**	0.8348	0.7863	0.7810	0.6580	0.7237	**0.1168**	0.1652	0.2137	0.2190	0.3420	0.2763
iris	0.9711	0.9667	0.9733	**0.9867**	0.9000	0.3733	0.0289	0.0333	0.0267	**0.0133**	0.1000	0.9067
led7digit	0.8600	0.9520	0.7880	0.9660	**0.9820**	0.9740	0.3800	0.1140	0.6780	0.0700	**0.0340**	**0.0340**
LEV	0.9993	**1.0000**	0.6990	**1.0000**	**1.0000**	**1.0000**	0.0007	**0.0000**	0.4450	**0.0000**	**0.0000**	**0.0000**
lymphography	**0.7819**	0.7705	0.6767	0.6900	0.7100	0.7029	**0.2314**	0.2567	0.3633	0.3714	0.3571	0.2971
machinecpu	**0.6520**	0.5638	0.4398	0.6369	0.6267	0.6362	0.4741	0.6086	0.7564	0.4726	0.5031	**0.4067**
mammographic	0.9763	0.9831	0.9735	**0.9904**	0.9892	0.9831	0.0237	0.0169	0.0265	**0.0096**	0.0108	0.0169
monk-2	0.9807	0.9746	0.9792	0.9769	0.9721	**0.9908**	0.0193	0.0254	0.0208	0.0232	0.0279	**0.0092**
movement_libras	**0.6796**	0.5194	0.5583	0.5333	0.3139	0.1194	**1.1602**	2.0306	1.7861	1.8528	4.3472	5.7000
newthyroid	**0.8621**	0.8279	0.8511	0.8329	0.6223	0.1818	**0.1905**	0.2186	0.1909	0.2139	0.5504	0.8413
pima	**0.8702**	0.8242	0.7837	0.8007	0.8151	0.6224	**0.1298**	0.1758	0.2163	0.1993	0.1849	0.3776
post-operative	0.6968	0.6333	0.4403	0.7403	**0.8292**	0.7569	0.3773	0.4806	0.7292	0.3153	**0.2042**	0.2431
saheart	**0.7302**	0.6627	0.6645	0.6624	0.6862	0.6839	**0.2698**	0.3373	0.3355	0.3376	0.3138	0.3161
segment	**0.9759**	0.9649	0.9632	0.9602	0.3061	0.1684	**0.0447**	0.0610	0.0671	0.0723	2.4022	2.8597
sonar	**0.8042**	0.7681	0.7250	0.7648	0.4662	0.5724	**0.1958**	0.2319	0.2750	0.2352	0.5338	0.4276
spectfheart	**0.8028**	0.7379	0.7412	0.7339	0.2095	0.8016	**0.1972**	0.2621	0.2588	0.2661	0.7905	0.1984
SWD	0.9993	**1.0000**	0.3820	**1.0000**	**1.0000**	**1.0000**	0.0007	**0.0000**	1.0240	**0.0000**	**0.0000**	**0.0000**
tae	0.8278	0.8483	0.5904	0.8417	0.8546	**0.8946**	0.1921	0.1650	0.5283	0.1913	0.1850	**0.1054**
titanic	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
vehicle	**0.7409**	0.6904	0.6384	0.6644	0.2530	0.2588	**0.4929**	0.5661	0.6475	0.6334	1.4917	0.9717
vowel	**0.9525**	0.7758	0.2182	0.7778	0.0909	0.0859	**0.1010**	0.6242	2.9313	0.5232	5.0000	4.8273
wdbc	**0.7183**	0.6768	0.6749	0.6713	0.6451	0.5607	**0.2817**	0.3232	0.3251	0.3287	0.3549	0.4393
windsorhousing	0.8932	0.8939	0.8738	0.8664	**0.9174**	0.8847	0.1068	0.1061	0.1262	0.1336	**0.0826**	0.1153
wine	**0.7926**	0.6794	0.7297	0.7578	0.3484	0.3314	**0.2882**	0.4219	0.3719	0.3154	0.9660	0.9667
wisconsin	**0.9747**	0.9591	0.9693	0.9591	0.8815	0.9547	**0.0253**	0.0409	0.0307	0.0409	0.1185	0.0453
yeast	0.4095	0.3659	0.2811	0.3639	**0.4596**	0.0836	1.7143	1.8605	3.1107	1.8111	**1.6300**	3.0862
zoo	0.7427	0.8127	0.4564	0.8127	**0.8409**	0.7727	0.7000	0.4727	1.8073	0.4336	**0.3682**	0.3745

Table 6: Number of leaves and Non Monotonic Index results reported over monotonic data sets.

	Number of leaves				Non Monotonic Index					
	MID RF	**MID C4.5**	**MID CART**	**MID RankTree**	**MID RF**	**MID C4.5**	**MID CART**	**MID RankTree**	**OLM**	**OSDL**
appendicitis	**5.2**	7.0	11.2	10.6	0.0352	0.0353	0.1592	0.1499	0.2180	**0.0000**
australian	9.7	**8.2**	90.0	90.1	0.0120	0.0131	**0.0008**	0.0034	0.2922	0.2794
auto-mpg	**43.3**	65.6	90.0	130.0	0.0006	0.0003	**0.0003**	0.2300	0.0464	0.0397
automobile	62.0	56.9	80.6	**35.8**	0.0003	0.0006	0.3764	0.4684	**0.0002**	0.0008
balance	**6.7**	9.6	90.0	17.1	0.0211	0.0106	0.0220	0.1946	0.0271	**0.0000**
bostonhousing	**55.2**	83.0	135.9	156.4	0.0004	0.0002	0.3375	0.3595	0.0011	**0.0000**
breast	95.3	118.6	90.0	**71.7**	0.0001	0.0001	**0.0000**	0.2498	0.0328	0.0328
bupa	**16.9**	23.3	77.2	57.5	0.0050	**0.0025**	0.1501	0.2218	0.1049	0.0803
car	128.3	125.4	**91.4**	180.0	**0.0000**	0.0001	0.0378	0.2624	0.0868	0.0868
cleveland	**36.1**	61.1	104.5	107.0	0.0009	**0.0003**	0.2528	0.2428	0.0116	0.0086
contraceptive	112.3	163.0	**94.6**	382.4	**0.0000**	**0.0000**	0.0280	0.2517	0.0794	0.0794
crx	122.9	154.5	**92.2**	103.2	**0.0000**	0.0001	0.0335	0.2762	0.0033	0.0031
dermatology	**25.9**	26.1	109.9	71.8	0.0014	**0.0013**	0.2647	0.3781	0.0020	0.0020
ecoli	**33.7**	50.7	115.4	116.3	0.0012	**0.0005**	0.2507	0.2853	0.1313	0.0024
ERA	31.0	28.5	**27.4**	36.0	**0.0021**	0.0026	0.2153	0.2633	0.1279	0.1279
ESL	**36.7**	39.9	112.1	85.4	0.0009	**0.0006**	0.3047	0.1189	0.6354	0.6354
flare	**87.0**	146.8	91.4	100.4	0.0002	0.0001	0.0623	0.3625	0.2779	0.2779
glass	**21.7**	27.8	49.7	54.0	0.0026	0.0016	0.3378	0.4485	0.0001	**0.0000**
haberman	**13.2**	17.4	39.2	30.0	0.0082	**0.0057**	0.1518	0.2181	0.2046	0.2046
hayes-roth	**12.9**	15.4	94.9	38.8	**0.0074**	0.0076	0.0571	0.3026	0.2187	0.2187
heart	**20.2**	30.8	55.9	44.3	0.0025	**0.0012**	0.1677	0.2009	0.0080	0.0083
hepatitis	**5.1**	7.9	9.9	10.7	0.0261	0.0187	0.1761	0.2527	**0.0009**	**0.0009**
housevotes	12.8	**6.8**	82.6	18.7	**0.0073**	0.0184	0.0766	0.3653	0.0509	0.0509
ionosphere	**16.0**	19.9	91.5	39.8	0.0042	0.0040	0.0498	0.2236	0.0038	**0.0035**
iris	5.1	**4.6**	8.1	7.7	0.0487	0.1114	0.2539	0.2552	0.2137	**0.0390**
led7digit	**19.1**	34.6	90.1	50.5	0.0040	**0.0016**	0.0233	0.2547	0.2513	0.2513
LEV	46.4	**36.4**	101.1	47.3	**0.0009**	0.0016	0.1472	0.2545	0.1925	0.1925
lymphography	31.2	**30.5**	91.2	39.3	**0.0013**	0.0015	0.0453	0.4516	0.0887	0.0887
machinecpu	**25.8**	35.7	93.6	79.9	0.0017	**0.0008**	0.0488	0.2028	0.0572	0.0572
mammographic	13.1	**11.8**	92.2	18.4	**0.0072**	0.0085	0.1064	0.1985	0.3230	0.3230
monk-2	**5.4**	12.6	91.8	20.8	0.0184	**0.0093**	0.0369	0.1589	0.0262	0.0262
movement_libras	**47.2**	66.1	104.2	128.4	0.0005	0.0003	0.2691	0.3227	0.0062	**0.0002**
newthyroid	**11.9**	12.6	39.2	29.0	0.0098	0.0086	0.2520	0.2378	0.0311	**0.0013**
pima	**32.4**	44.7	108.5	98.7	0.0014	**0.0007**	0.1469	0.2052	0.0554	0.0014
post-operative	**18.6**	27.9	91.0	33.0	0.0040	**0.0018**	0.0352	0.3957	0.1322	0.1322
saheart	**26.5**	47.7	82.1	94.5	0.0020	**0.0006**	0.1845	0.2179	0.0342	0.0385
segment	56.4	**47.8**	77.9	84.2	0.0004	0.0006	0.3421	0.4048	0.0001	**0.0000**
sonar	**13.0**	15.7	21.5	21.3	0.0063	0.0063	0.1842	0.2222	**0.0000**	**0.0000**
spectfheart	**14.3**	22.3	37.8	32.0	0.0047	0.0032	0.1830	0.3067	**0.0000**	**0.0000**
SWD	69.3	**53.5**	90.0	62.3	0.0004	0.0007	**0.0000**	0.2728	0.0979	0.0979
tae	**15.0**	16.3	93.4	36.9	0.0061	**0.0041**	0.0424	0.2397	0.1216	0.1216
titanic	**4.0**	4.0	4.0	4.0	0.1163	0.1667	0.1667	0.1667	**0.0282**	**0.0282**
vehicle	**71.3**	99.8	147.6	178.8	0.0002	0.0001	0.3101	0.2928	**0.0000**	**0.0000**
vowel	102.5	102.5	**94.9**	198.0	0.0001	0.0001	0.0590	0.3976	**0.0000**	**0.0000**
wdbc	**31.5**	53.1	84.1	86.6	0.0013	**0.0005**	0.2172	0.2267	0.0287	0.0057
windsorhousing	**24.6**	35.0	62.6	79.5	0.0015	**0.0007**	0.1852	0.1837	0.1918	0.1918
wine	**13.8**	18.8	30.6	34.8	0.0062	0.0040	0.2524	0.3337	0.0021	**0.0004**
wisconsin	**12.4**	16.8	57.0	27.0	0.0075	**0.0050**	0.0766	0.1854	0.3778	0.3778
yeast	231.6	349.3	**129.8**	802.5	**0.0000**	**0.0000**	0.2918	0.3101	0.0494	0.0023
zoo	**13.5**	16.6	90.0	25.2	0.0090	**0.0049**	0.0200	0.5680	0.0389	0.0389

Table 7: Accuracy and MAE results reported over non-monotonic data sets.

	Accuracy						**Mean Absolute Error**					
	MID RF	**MID C4.5**	**MID CART**	**MID RankTree**	**OLM**	**OSDL**	**MID RF**	**MID C4.5**	**MID CART**	**MID RankTree**	**OLM**	**OSDL**
appendicitis	**0.8791**	0.8218	0.7855	0.7655	0.8018	0.7727	**0.1209**	0.1782	0.2145	0.2345	0.1982	0.2273
australian	**0.8725**	0.8377	0.8493	0.8246	0.6971	0.7362	**0.1275**	0.1623	0.1507	0.1754	0.3029	0.2638
automobile	0.8045	**0.8097**	0.7086	0.7374	0.2195	0.3940	0.2650	**0.2641**	0.4765	0.3903	2.1534	0.8801
balance	**0.8634**	0.7903	0.5859	0.7728	0.4575	0.4991	**0.1868**	0.2545	0.7434	0.2560	1.0034	0.9234
bands	**0.7137**	0.6206	0.6148	0.6088	0.3702	0.6219	**0.2863**	0.3794	0.3852	0.3912	0.6298	0.3781
breast	**0.7327**	0.6395	0.7080	0.7112	0.6583	0.5464	**0.2673**	0.3605	0.2920	0.2888	0.3417	0.4536
car	0.8609	0.9335	0.8131	0.9294	0.9230	**0.9392**	0.1919	0.0868	0.2321	0.0932	0.1198	**0.0642**
cleveland	**0.5745**	0.5154	0.4852	0.5483	0.5625	0.5562	**0.6450**	0.7734	0.8341	0.6739	0.8408	0.6731
contraceptive	**0.5493**	0.4861	0.5065	0.4637	0.4324	0.3116	**0.6618**	0.7372	0.7113	0.8478	0.8473	0.7631
crx	**0.8651**	0.8327	0.8508	0.8056	0.5988	0.6333	**0.1349**	0.1673	0.1492	0.1944	0.4012	0.3667
ecoli	**0.8403**	0.8009	0.7443	0.7680	0.5715	0.0324	**0.5611**	0.7100	0.9428	0.8535	1.5379	2.3482
flare	**0.7502**	0.7336	0.6117	0.5994	0.5357	0.4944	**0.4964**	0.5111	0.6200	0.6689	0.8430	0.6894
german	**0.7420**	0.6940	0.6160	0.6750	0.7120	0.3730	**0.2580**	0.3060	0.3840	0.3250	0.2880	0.6270
glass	**0.7613**	0.6476	0.6811	0.6513	0.3244	0.3379	**0.4890**	0.7356	0.6567	0.7182	1.7816	1.7729
haberman	**0.7304**	0.7252	0.6988	0.6962	0.3496	0.7158	**0.2696**	0.2748	0.3012	0.3038	0.6504	0.2842
hayes-roth	0.7688	0.7938	0.5938	0.4563	0.5625	**0.8125**	0.2479	0.2063	0.4188	0.6688	0.4813	**0.1938**
heart	**0.8346**	0.7704	0.7296	0.7630	0.6667	0.6259	**0.1654**	0.2296	0.2704	0.2370	0.3333	0.3741
hepatitis	**0.8710**	0.7942	0.8257	0.8692	0.2497	0.8118	**0.1290**	0.2058	0.1743	0.1308	0.7503	0.1882
housevotes	0.9472	0.9525	**0.9620**	0.9507	0.8586	0.6843	0.0528	0.0475	**0.0380**	0.0493	0.1414	0.3157
ionosphere	**0.9299**	0.9060	0.8235	0.8491	0.6269	0.7407	**0.0701**	0.0940	0.1765	0.1509	0.3731	0.2593
iris	**0.9533**	0.9533	0.9533	0.9467	0.9333	0.3667	**0.0467**	0.0467	0.0467	0.0533	0.0667	0.9067
led7digit	0.7127	**0.7180**	0.2960	0.3320	0.3340	0.2240	1.2300	**1.1940**	2.3780	2.2300	2.6680	2.3860
lymphography	**0.8309**	0.7660	0.6991	0.7847	0.4866	0.5789	**0.1812**	0.2474	0.3009	0.2411	0.6176	0.4554
mammographic	**0.8360**	0.7991	0.8276	0.7888	0.8074	0.6291	**0.1640**	0.2009	0.1724	0.2112	0.1926	0.3709
monk-2	0.9757	**1.0000**	**1.0000**	**1.0000**	0.4734	0.5243	0.0243	**0.0000**	**0.0000**	**0.0000**	0.5266	0.4757
newthyroid	0.9618	0.9212	0.9491	**0.9673**	0.5636	0.1439	0.0628	0.1065	0.0786	**0.0468**	0.7662	0.8654
phoneme	**0.8824**	0.8699	0.7728	0.8605	0.7618	0.7060	**0.1176**	0.1301	0.2272	0.1395	0.2382	0.2940
pima	**0.7549**	0.7265	0.6993	0.6966	0.7110	0.6563	**0.2451**	0.2735	0.3007	0.3034	0.2890	0.3437
post-operative	**0.6972**	0.5528	0.6778	0.6653	0.6056	0.3417	**0.3139**	0.4472	0.3333	0.3458	0.4403	0.7750
saheart	0.7071	**0.7099**	0.6298	0.6514	0.6557	0.5066	0.2929	**0.2901**	0.3702	0.3486	0.3443	0.4934
tae	0.5904	**0.5971**	0.4317	0.4850	0.3508	0.3154	0.5311	**0.5154**	0.7138	0.6604	0.9871	0.7975
titanic	0.7862	**0.7883**	0.7669	0.7756	0.3249	0.4698	0.2138	**0.2117**	0.2331	0.2244	0.6751	0.5302
vehicle	**0.7530**	0.7505	0.6374	0.7352	0.2352	0.2447	0.4723	**0.4528**	0.6461	0.4905	1.5283	1.0260
vowel	**0.9525**	0.7758	0.2182	0.7778	0.0909	0.0859	**0.1010**	0.6242	2.9313	0.5232	5.0000	4.8273
wdbc	**0.9561**	0.9403	0.9226	0.9227	0.3392	0.3568	**0.0439**	0.0597	0.0774	0.0773	0.6608	0.6432
wine	**0.9792**	0.9271	0.8931	0.9157	0.3042	0.3261	**0.0208**	0.0729	0.1402	0.0843	0.9536	0.9487
winequality-red	**0.6825**	0.6091	0.5829	0.6110	0.2958	0.0119	**0.3509**	0.4747	0.4803	0.4553	1.7174	2.5341
wisconsin	**0.9714**	0.9401	0.9561	0.9501	0.8872	0.9593	**0.0286**	0.0599	0.0439	0.0499	0.1128	0.0407
yeast	**0.6144**	0.5115	0.4333	0.4892	0.2183	0.1820	**0.7689**	0.9953	1.1091	1.1448	1.9616	2.0214
zoo	**0.9583**	0.9381	0.7328	0.9425	0.8561	0.4547	**0.0667**	0.0903	0.3839	0.1500	0.6250	1.4744

Table 8: Number of leaves and Non Monotonic Index results reported over non-monotonic data sets.

	Number of leaves				Non Monotonic Index					
	MID RF	**MID C4.5**	**MID CART**	**MID RankTree**	**MID RF**	**MID C4.5**	**MID CART**	**MID RankTree**	**OLM**	**OSDL**
appendicitis	3.7	**3.4**	14.4	17.3	0.0298	0.1936	0.2715	0.2929	**0.0000**	0.0021
australian	**41.1**	61.4	90.8	94.2	0.0007	**0.0003**	0.0201	0.2786	0.0175	0.0187
automobile	60.6	60.2	86.7	**35.7**	0.0003	0.0007	0.3602	0.4770	**0.0002**	0.0009
balance	**41.7**	58.9	98.7	158.8	0.0006	**0.0002**	0.1286	0.3383	0.0053	0.0277
bands	**26.3**	46.7	93.8	74.0	0.0003	0.0007	0.1752	0.2396	0.0001	**0.0000**
breast	97.4	145.7	**90.0**	101.3	0.0001	0.0001	**0.0000**	0.0954	0.0279	0.0347
car	129.1	134.1	**90.3**	178.3	**0.0000**	0.0001	0.0291	0.2620	0.0837	0.0864
cleveland	**34.7**	58.8	108.0	104.2	0.0009	**0.0003**	0.2711	0.2520	0.0095	0.0086
contraceptive	138.5	261.9	103.8	**99.2**	**0.0000**	**0.0000**	0.1347	0.0940	0.0681	0.0342
crx	104.7	127.5	93.6	**83.9**	**0.0001**	**0.0001**	0.0756	0.3070	0.0024	0.0029
ecoli	**22.1**	27.3	81.0	73.1	0.0026	0.0015	0.2379	0.2989	0.1271	**0.0002**
flare	**88.4**	148.1	90.3	94.8	0.0002	**0.0000**	0.0308	0.1466	0.2120	0.2252
german	202.1	284.3	**93.3**	220.0	**0.0000**	**0.0000**	0.0441	0.2468	0.0005	0.0006
glass	**21.0**	28.7	47.1	52.0	0.0028	0.0016	0.3329	0.4635	**0.0000**	**0.0000**
haberman	10.3	**7.2**	94.4	98.9	0.0347	0.0365	0.0869	0.1257	0.1108	**0.0056**
hayes-roth	**15.0**	17.2	91.3	92.1	0.0055	**0.0044**	0.0223	0.0296	0.1267	0.2341
heart	**19.8**	32.0	54.5	45.3	0.0025	**0.0011**	0.1585	0.2171	0.0072	0.0086
hepatitis	**5.2**	8.5	9.7	10.6	0.0245	0.0138	0.1646	0.2658	**0.0010**	**0.0010**
housevotes	9.7	**5.8**	48.9	15.6	0.0097	0.0251	0.2304	0.3493	**0.0076**	0.0290
ionosphere	**12.5**	13.6	90.6	28.7	0.0057	0.0044	0.0449	0.2670	**0.0023**	0.0029
iris	5.4	**4.4**	10.4	11.3	0.0406	0.0524	0.2567	0.2501	0.2098	**0.0390**
led7digit	**20.4**	26.8	90.0	90.0	0.0033	**0.0007**	0.0218	0.0433	0.3411	0.3512
lymphography	**26.9**	26.9	91.0	27.9	**0.0016**	0.0020	0.0414	0.3394	0.0898	0.0798
mammographic	**17.7**	21.9	90.0	90.3	0.0027	0.0009	**0.0000**	0.0066	0.3156	0.2060
monk-2	14.4	**5.0**	7.2	7.0	0.0044	0.0600	0.4224	0.4286	**0.0000**	**0.0000**
newthyroid	**7.0**	8.1	15.8	12.5	0.0237	0.0248	0.2867	0.3075	0.0332	**0.0009**
phoneme	**112.8**	163.1	164.9	603.6	0.0001	0.0001	0.3036	0.2341	0.0126	**0.0000**
pima	31.7	**28.2**	96.0	151.0	0.0014	0.0021	0.0473	0.2126	0.0369	**0.0012**
post-operative	**14.7**	23.1	91.6	92.5	0.0046	**0.0019**	0.0209	0.0261	0.0617	0.0817
saheart	21.4	**16.1**	109.7	103.0	**0.0031**	0.0044	0.1677	0.2224	0.0208	0.0318
tae	**16.4**	30.4	90.0	97.7	0.0045	**0.0011**	0.0222	0.1399	0.0391	0.1220
titanic	**6.0**	6.1	90.0	90.0	0.0199	**0.0152**	0.0224	0.0236	0.2015	0.4673
vehicle	**66.6**	91.3	137.8	155.8	0.0003	0.0002	0.2873	0.2840	**0.0000**	**0.0000**
vowel	102.5	102.5	**94.9**	198.0	0.0001	0.0001	0.0590	0.3976	**0.0000**	**0.0000**
wdbc	**11.6**	14.1	24.6	21.3	0.0036	0.0052	0.2888	0.3359	0.0137	**0.0000**
wine	**6.8**	7.1	14.3	9.5	0.0064	0.0063	0.3470	0.5500	0.0005	**0.0001**
winequality-red	**160.2**	245.9	235.6	422.9	**0.0000**	**0.0000**	0.4761	0.2300	0.0027	**0.0000**
wisconsin	**14.2**	20.7	50.8	30.9	0.0056	**0.0034**	0.1459	0.1764	0.3621	0.3829
yeast	172.3	257.6	**150.0**	590.0	**0.0000**	**0.0000**	0.4240	0.4739	0.0365	0.0023
zoo	11.1	**10.8**	90.0	12.1	**0.0144**	0.0187	0.0220	0.2638	0.0220	0.0255

Table 9: Summary table for statistical inference outcome: Ranks and Adjusted P-Values

	Monotonic Data Sets				**Non Monotonic Data Sets**			
	Acc		**MAE**		**Acc**		**MAE**	
	Ranks	**APVs**	**Ranks**	**APVs**	**Ranks**	**APVs**	**Ranks**	**APVs**
MID-RF	**2.480**	–	**2.540**	–	**1.475**	–	**1.500**	–
MID-C4.5	3.360	0.037	3.310	0.079	2.625	0.006	2.550	0.012
MID-CART	4.310	0.000	4.320	0.000	3.600	0.000	3.500	0.000
MID-RankTree	3.200	0.054	3.230	0.079	3.375	0.000	3.425	0.000
OLM	3.650	0.005	3.800	0.003	4.900	0.000	5.175	0.000
OSDL	4.000	0.000	3.800	0.003	5.025	0.000	4.850	0.000
	NL		**NMI**		**NL**		**NMI**	
	Ranks	**APVs**	**Ranks**	**APVs**	**Ranks**	**APVs**	**Ranks**	**APVs**
MID-RF	**1.469**	–	2.300	0.423	**1.550**	–	2.287	0.811
MID-C4.5	2.041	0.028	**2.000**	–	2.125	0.046	**2.187**	–
MID-CART	2.345	0.000	4.130	0.000	3.025	0.000	4.437	0.000
MID-RankTree	3.255	0.000	5.600	0.000	3.300	0.000	5.500	0.000
OLM	–	–	3.780	0.000	–	–	3.375	0.005
OSDL	–	–	3.190	0.001	–	–	3.212	0.015

the standard RF was not used. This is all due to the variability caused by the pruning procedure, which allows the most monotonic and simple trees to be selected.

- Finally, referring to the NMI, Tables 6, 8 and 9 reflect better results for MID-C4.5. However, the difference between MID-RF and MID-C4.5 is not pointed out as significant in both monotonic and non-monotonic data sets. In conclusion Monotonic RF and MID-C4.5 have performed equally well.

To conclude, our approach performs better in almost every measure considered in the study than all the compared algorithms, except for NMI when comparing with MID-C4.5.

5 Concluding Remarks

The purpose of this paper is to present and to analyse a Random forest proposal for classification with monotonicity constraints. In order to be adapted to this problem, it includes the rate of monotonicity as a parameter to be randomised during the growth of the trees. After building of all the decision trees, an ensemble pruning mechanism based on the monotonicity index of each tree is used to select the subset of the most monotonic decision trees to constitute the forest. The results show that Random Forests are promising models to address this problem obtaining very accurate results involving trees with a low non monotonic index. We have compared it with other monotonic decision tree approaches and their effectiveness is clearly overwhelmed by the Random Forest approach.

As future work, we will examine the possibility of using other sophisticated pruning mechanisms, modifying them to use NMI for criterion.

Acknowledgements

This work is supported by the research project TIN2014-57251-P and by a research scholarship, given to the author Sergio Gonzalez by the University of Granada.

References

[1] A. Ben-David, L. Sterling, and Y. H. Pao. Learning, classification of monotonic ordinal concepts. *Computational Intelligence*, 5:45–49, 1989.

[2] W. Kotlowski and R. Slowinski. On nonparametric ordinal classification with monotonicity constraints. *IEEE Transactions on Knowledge Data Engineering*, 25(11):2576–2589, 2013.

[3] C.-C. Chen and S.-T. Li. Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16):7235–7247, 2014.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Chapman & Hall, 1984.

[5] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[6] L. Rokach and O. Maimon. *Data Mining with Decision Trees. Theory and Applications. 2nd Edition.* World Scientific, 2014.

[7] J. Furnkranz, D. Gamberger, and N. Lavrac. *Foundations of Rule Learning*. Springer, 2012.

[8] M. Wozniak, M. Graña, and E. Corchado. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3–17, 2014.

[9] Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338, 2006.

[10] G. Martínez-Muñoz, D Hernández-Lobato, and A. Suárez. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):245–259, 2009.

[11] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[12] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[13] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1):29–43, 1995.

[14] F. Xia, W. Zhang, F. Li, and Y. Yang. Ranking with decision tree. *Knowledge and Information Systems*, 17(3):381–395, 2008.

[15] A. Ben-David. Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications. *Decision Sciences*, 23:1357–1372, 1992.

[16] S. Lievens, B. De Baets, and K. Cao-Van. A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Annals of Operaional Research*, 163(1):115–142, 2008.

[17] W. Duivesteijn and A. Feelders. Nearest neighbour classification with monotonicity constraints. In *ECML/PKDD (1)*, pages 301–316, 2008.

[18] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Rechearch*, 7:1–30, 2006.

[19] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[20] S. García, A. Fernández, J. Luengo, and 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(10):2044–2064, 2010.

[21] A. Ben-David, L. Sterling, and T. Tran. Adding monotonicity to learning algorithms may impair their accuracy. *Expert Systems with Applications*, 36(3):6627–6634, 2009.

[22] S. García, J. Luengo, and F. Herrera. *Data Preprocessing in Data Mining*. Springer, 2015.

[23] R. Potharst, A. Ben-David, and M. C. van Wezel. Two algorithms for generating structured and unstructured monotone ordinal datasets. *Engineering Applications of Artificial Intelligence*, 22(4-5):491–496, 2009.

[24] R. Potharst and A. J. Feelders. Classification trees for problems with monotonicity constraints. *SIGKDD Explorations*, 4(1):1–10, 2002.

[25] K. Cao-Van and B. De Baets. Growing decision trees in an ordinal setting. *International Journal of Intelligent Systems*, 18(7):733–750, 2003.

[26] Q. Hu, X. Che, Lei Z., D. Zhang, M. Guo, and Daren Yu. Rank entropy-based decision trees for monotonic classification. *IEEE Transactions on Knowledge Data Engineering*, 24(11):2052–2064, 2012.

[27] I. Czarnowski and P. Jedrzejowicz. Designing RBF networks using the agent-based population learning algorithm. *New Generation Computing*, 32(3-4):331–351, 2014.

[28] H. Daniels and M. Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

[29] F. Fernández-Navarro, A. Riccardi, and S. Carloni. Ordinal neural networks without iterative tuning. *IEEE Transactions on Neural Networks and Learning Systems*, 25(11):2075–2085, 2014.

[30] K. Makino, T. Suda, H. Ono, and T. Ibaraki. Data analysis by positive decision trees. *IEICE Transactions on Information and Systems*, E82(D(1)):76–88, 1999.

[31] R. Potharst and J. C. Bioch. Decision trees for ordinal classification. *Intelligent Data Analysis*, 4(2):97–111, 2000.

[32] A. J. Feelders and M. Pardoel. Pruning for monotone classification trees. In *IDA*, volume 2810 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003.

[33] J. W. T. Lee, D. S. Yeung, and X. Wang. Monotonic decision tree for ordinal classification. In *IEEE International Conference on Systems, Man and Cybernetics.*, pages 2623–2628, 2003.

[34] H. Zhu, J. Zhai, S. Wang, and X. Wang. Monotonic decision tree for interval valued data. In *Machine Learning and Cybernetics - 13th International Conference*, pages 231–240, 2014.

[35] C. Marsala and D. Petturiti. Rank discrimination measures for enforcing monotonicity in decision tree induction. *Information Sciences*, 291:143–171, 2015.

[36] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[37] J. Alcala-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*, 17(2-3):255–287, 2011.

[38] R. C. Prati, G. E. A. P. A. Batista, and M. C. Monard. A survey on graphical methods for classification predictive performance evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 23(11):1601–1618, 2011.

[39] L. Gaudette and N. Japkowicz. Evaluation methods for ordinal classification. In *Canadian Conference on AI*, volume 5549 of *Lecture Notes in Computer Science*, pages 207–210, 2009.

[40] N. Japkowicz and M. Shah, editors. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

2 Class switching according to nearest enemy distance for learning from highly imbalanced data-sets

- González, S., García, S., Lázaro, M., Figueiras-Vidal, A. R., & Herrera, F. (2017). Class switching according to nearest enemy distance for learning from highly imbalanced data-sets. *Pattern Recognition*, 70, 12-24.
  - Status: **Published**
  - Impact Factor (JCR 2017): **3.965**
  - Subject Category: **Computer Science, Artificial Intelligence**
  - Rank: **16/132**
  - Quartile: **Q1**

CLASS SWITCHING ACCORDING TO NEAREST ENEMY DISTANCE FOR LEARNING FROM HIGHLY IMBALANCED DATA-SETS

Sergio González
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
sergiogvz@decsai.ugr.es

Salvador García
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
salvagl@decsai.ugr.es

Marcelino Lázaro
Department of Signal Theory and Communications
Carlos III University of Madrid, 28903 Getafe, Madrid
mlazaro@tsc.uc3m.es

Aníbal R. Figueiras-Vidal
Department of Signal Theory and Communications
Carlos III University of Madrid, 28903 Getafe, Madrid
arfv@tsc.uc3m.es

Francisco Herrera
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
herrera@decsai.ugr.es

ABSTRACT

The imbalanced data classification has been deeply studied by the machine learning practitioners over the years and it is one of the most challenging problems in the field. In many real-life situations, the under representation of a class in contrary to the rest commonly produces the tendency to ignore the minority class, this being normally the target of the problem. Consequently, many different techniques have been proposed. Among those, the ensemble approaches have resulted to be very reliable. New ways of generating ensembles have also been studied for standard classification. In particular, Class Switching, as a mechanism to produce training perturbed sets, has been proved to perform well in slightly imbalanced scenarios. In this paper, we analyze its potential to deal with highly imbalanced problems, fighting against its major limitations. We introduce a novel ensemble approach based on Switching with a new technique to select the switched examples based on Nearest Enemy Distance. We compare the resulting SwitchingNED with five distinctive ensemble-based approaches, with different combinations of sampling techniques. With a better performance, SwitchingNED is settled as one of best approaches on the field.

***K*eywords** Imbalanced classification · Ensembles · Preprocessing · Class Switching.

1 Introduction

The classification of datasets with skewed class distributions has drawn the attention of practitioners along the years of machine learning research. The class imbalance problem refers to datasets with a wide disparity in the number of instances for each class. In a binary imbalanced scenario, the minority class has much fewer examples than those of the majority class. These differences in the amount of samples cause a great loss of the minority class accuracy when applying standard classifiers. Real-life classification problems frequently suffer from this difficulty and misclassifying an example of the minority class usually entails greater costs. Therefore, the treatment of this problem is extremely important.

However, the skewed class distribution is not the only problem that has to be handled in order to obtain good behaviors [1, 2]. Those other problems are the overlapping between classes [3], the high impact of noise [4, 5], the identification of small disjuncts [6], the lack of density in the training data [7, 8] and the possible differences in the data distribution between the training and test sets (dataset shift) [9, 10]. Even though these are common problems in standard classification, they have a greater impact over the minority class in imbalanced environments.

Consequently, a large number of techniques have been proposed over these years to deal with this problem. Three groups can be defined: Algorithmic based approaches [11], cost-sensitive learning [12, 13], and sampling data based approaches [2]. Ensembles have been successfully combined with these earlier methods recalling the best performances towards the problem [14]. EUSBoost was proposed in [15], and it has been found as the best ensemble to deal with highly imbalanced scenarios.

A type of ensemble known as Output Flipping was proposed in [16], based on randomizing the output of the training set, maintaining the input data untouched, differently to Bootstrapping. Output Flipping exchanges the class labels between instances preserving the relative frequencies of the classes. This last characteristic limits its applicability to imbalanced datasets. Later on, this same concept was extended in [17] and named as Class Switching. This algorithm randomly selects the instances to be changed without maintaining the same number of representatives from each class. These approaches have been proved to perform similarly to Bagging. However, the particularities of the Switching algorithm tend to equal the number of representatives from all classes. Consequently, Switching performs better than other ensembles in slightly imbalanced scenarios.

Due to its effectiveness in slightly imbalanced scenarios, we explore the suitability of Switching for classification in highly imbalanced problems. We set the challenge of using the goodness of Switching approaches to successfully deal with highly imbalanced datasets, expecting to get performance advantages. In particular, we propose a novel Switching-based ensemble with a new technique to select the switched examples based on the Nearest Enemy Distance (NED). We call this approach SwitchingNED. Its new way of selecting the switched instances changes drastically the initial idea of the base approaches and solves their drawbacks for this kind of problems. Class Switching is applied to instances of the majority class according to their proximity to the minority class, measured by the NED. This allows a growth of the minority class population near to the classification boundaries, finding a better balanced representation between classes in highly imbalanced scenarios. As done with other ensembles [14], we combine SwitchingNED with different sampling methods, obtaining very promising results.

In order to test the proposed method under the given hypotheses, we perform several experiments which support that SwitchingNED successfully deals with highly imbalanced datasets and is one of the best approaches on the field. We have checked the improvement of our scheme against the basic Switching algorithm and its behavior in combination of preprocessing techniques. SwitchingNED is compared with five different representative approaches of each combination of sampling techniques and ensemble schemes [14]: EUSBoost [15], UnderBagging [18], SMOTEBoost [19], SMOTEBagging [20] and EasyEnsemble [21]. Among these methods, the best method of the literature is included, which is EUSBoost [15]. This experimental framework includes a

total of 33 highly imbalanced datasets, where the majority class is at least 9 times bigger than the minority one. Furthermore, the empirical study has been validated using non-parametric statistical testing [22].

This paper is organized as follows. In Section 2 we present the class imbalance problem, within its drawbacks, its measures and the possible solutions to it. Section 3 introduces the randomizing output approaches. Section 4 is devoted to introducing SwitchingNED with its novel technique of selecting the instances to be switched based on NED. Section 5 describes the experimental framework followed by the empirical results and compares them to those offered by the original Switching approach. In Section 6 the use of sampling techniques in SwitchingNED is presented and analyzed with different experiments. Section 7 closes the paper presenting its main conclusions.

2 The problem of imbalanced class distributions in classification

In this section, we first present the class imbalance problem in detail, specifying its characteristics, the difficulties that it creates for standard classifiers, and the particular metrics used in this problem. Afterwards, we briefly survey the different approaches to deal with this problem.

2.1 The imbalanced class problem

This problem appears when one of the classes, known as minority or positive class, has fewer representing examples than the other, majority or negative class. It is common in many real applications, such as biometric identification [23, 24] and bioinformatics [25, 26], that have brought a growth of attention by researchers. Furthermore, the minority class normally merits more interest from a learning point of view, implying a greater cost when it is not well classified [27].

The main concern with imbalanced datasets is that the standard classification learning methods are usually biased toward the majority class. Standard classifiers use commonly global performance measures, like the accuracy rate, to guide the learning process, which benefits the majority class. Even more, they are normally designed to discard very highly specialized classification rules, those needed to predict the positive class, in favour of more general ones. Consequently, there is a higher misclassification rate for the minority class.

This loss of performance is mainly caused by the differences in the amounts of samples between classes, increasing with the Imbalance Ratio (IR) [1] of the datasets. This ratio is computed as the number of the negative class examples divided by the number of positive class examples. The datasets with IR greater than 9 are considered as highly imbalanced [15]. There are several studies [1, 14] pointing out that there are other problems to consider in order to obtain good behaviors. The presence of small disjuncts, overlapping, lack of density or noisy have been proved to greatly reduce the accuracy of the minority class prediction [1].

The evaluation of the algorithms applied to this problem is of great relevance in order to properly guide their learning phase. Standard measures like accuracy cannot be used, because they assess the methods with the overall performance on the entire datasets and not the performance on each class independently. There are other more suitable measures for this environment, such as the Area Under the ROC Curve or Geometric Mean. The Area Under the ROC Curve (AUC) [28] is a widely used measure in imbalanced domains [1, 2, 14]. The Receiver Operating Characteristic (ROC) curve [29] represents graphically the trade-off between the percentage of well classified positive instances (TP_{rate}) and the percentage of incorrectly classified negative instances (FP_{rate}). In our experiments, we use the simplified expression of the AUC computed with the TP_{rate} and FP_{rate}[15].

2.2 Solutions to the class imbalance problem

As previously mentioned, the approaches for dealing with imbalanced dataset classifications could be divided in the three groups: Cost-sensitive learning solutions [13, 30], algorithmic approaches [11, 31] or data level methods [2, 19, 32].

The cost-sensitive learning considers the costs of the errors by minimizing a cost function that includes them. In imbalanced classification, these methods take into account the higher cost of misclassification of the minority class with respect to its alternative. These costs are defined by domain experts or learned with other approaches. The algorithmic approaches are modifications of base learning algorithms that achieve good performance with imbalanced datasets.

Finally, the data level methods focus on pre-processing the original dataset in order to obtain a more balanced one. These techniques allow to use standard learning algorithms after the preprocessing. The equal representation between classes is reached by generating more examples for the positive class (oversampling), removing examples from the negative class (undersampling) or both (hybrid methods). The most elementary techniques in this field are Random Under/Over- sampling, which randomly remove or replicate instances from the majority/minority class. Its first modality has the disadvantage of being able to eliminate real valuable data, while the oversampling can come out with over-fitting. Several sophisticated approaches have been proposed, such as SMOTE (Synthetic Minority Oversampling TEchnique)[33] and others based on complex heuristics like genetic algorithms [34].

Another category can be defined when ensemble classifiers are considered. Recently, these types of approaches have become more and more popular [14], due to their potential to enhance the results of traditional data preprocessing and cost-sensitive learning approaches by combining them. The cost-sensitive ensembles [12] have been less attractive, because the misclassification costs are not easy to define and the standard classification datasets usually do not include them. On the contrary, the data preprocessing ensembles are more general [32]. In this sub-group, Bagging- [18, 35], Boosting [19, 36]-, and Hybrid-based ensembles are combined with one or more of the traditional sampling techniques. This combination permits, while making the class representations equal, to increase the diversity of the ensemble and to reduce the impact of the sampling mistakes, such as removing relevant instances in the case of undersampling. Due to that fact, these methods achieve the best results for class imbalanced problems. An extensive empirical study of the ensembles solutions to this problem was presented in [14]. Among these solutions, it is worth to mention the superiority of EUSBoost [15] with respect to the other approaches in highly imbalanced scenarios.

3 Randomizing output and class-switching

In this section, we review the randomizing output and class-switching ensembles with enough depth to posteriorly settle down the new proposal. In [16], Breiman brought up a thrilling question about the possibility of obtaining a comparable performance to that of other ensembles by just altering the outputs of the problem, i.e. the class labels. The main idea for his approach, called Output Flipping, is to randomize the output of a small fraction of the dataset, producing different controlled noises that will be canceled when aggregated, leading to a better prediction accuracy. Breiman's Output Flipping has been proved in [17] to be very limited for imbalanced scenarios, with its flipping rate constrained by the minority class proportion. Therefore, we have not considered this proposal for our study. Posteriorly, a new ensemble based on randomizing outputs was proposed in [17], under the name of Switching, solving that issue in some degree. The Switching algorithm differs from Output Flipping in the mechanism used to change the class labels. In each iteration of the ensemble, a fixed fraction *fr*

of the instances is randomly selected. All these examples are switched to another randomly chosen class. This is represented with the following transition matrix:

$$\begin{aligned} P_{j\leftarrow i} &= fr/(K-1) \quad \text{for} \quad i \neq j \\ P_{i\leftarrow i} &= 1 - fr \end{aligned} \tag{1}$$

where $P_{j\leftarrow i}$ is the probability to change an instance from the class i to the class j, and K, the number of classes. This output switching mechanism does not ensure to maintain the original distribution of the data. Actually, the procedure tends to equilibrate the class distribution of imbalanced datasets with the rising of the parameter *fr*. The probability of randomly selecting an example of the majority class is higher than for positive examples.

Due to this fact, the constraint for the parameter *fr* becomes less restrictive. To fulfil the convergence of the ensemble, the majority of the examples for each class should not be switched, keeping the integrity of every class. This is generally satisfied if $P_{j\leftarrow i} < P_{i\leftarrow i}$ and, according to Eq. 1, the restriction for *fr* is:

$$fr < (K-1)/K \tag{2}$$

This permits a greater range of acceptable values for imbalanced sets than the flipping procedure does. This characteristic and the tendency to re-balance the class distribution turn this ensemble into a very promising initial approach to deal with the imbalanced data classification problem. However, this expression does not consider the maintenance of the integrity of the minority class in imbalanced problems due to flooding. Once the switching process is applied, there should be more real instances (i.e., not switched) than switched examples in each class. If this condition is not fulfilled, the models start to give more importance to the switched instances, deteriorating the results. The minority class is affected in a greater degree by this situation. Therefore, we do not recommend to exceed the value shown in Eq. 3 (this formulation and its development is explained in the A):

$$fr < \frac{K-1}{(P_m)^{-1} + K - 2} \tag{3}$$

where P_m is the proportion of the entire dataset samples belonging to minority classes. In the common binary scenario, the constraint is simplified to $fr < P_m$. Due to this fact, Switching only works well with slightly imbalanced sets. For highly imbalanced datasets, the problem of flooding the minority class with examples of the majority class remains unsolved. Therefore, a new design is needed to finally cope with that problem.

4 Class-Switching according to the Nearest Enemy Distance

In this section, we introduce SwitchingNED and explain how it is able to handle highly imbalanced data. We will separately detail in two subsections the different aspects of our Switching-based algorithm. In Subsection 4.1, we will start by focusing on the problems of Switching with imbalanced datasets and the first necessary step to solve them. Along Subsection 4.2, we present a novel way of selecting the instances to be changed to the minority class based on the Nearest Enemy Distance.

4.1 Switching-I: Switching for Imbalanced scenarios

For highly imbalanced scenarios, the Switching algorithm finds several difficulties. The convergence is compromised, needing a high switching rate and an excessive number of classifiers. The minority class is also flooded with majority class examples. Then, the classifiers could give more importance to the switched instances than to the real minority class instances while inferring the models. This could degenerate the majority

class accuracy, while still ignoring the minority class. Finally, with the increase of the switching rate, there is the possibility of loosing the few relevant minority instances towards other class, which makes impossible to infer a proper model for the minority class.

We maintain the minority instances untouched. The examples of the majority class are the only ones to be changed. This allows to retain all the relevant information of the minority class. This modification is an important step to facilitate the balancing process for slightly imbalanced datasets. We will refer to it as *Switching-I*.

Nevertheless, the flooding problem is still there for highly imbalanced datasets. We have noticed that it is not recommendable to excessively surpass the number of instances of the minority class with the switched samples. Exceeding a certain threshold, the real minority instances start to loose importance in the learning stage, deteriorating the ensemble models and the results. The convergence of the ensemble gets also harder to achieve, needing an excessive number of iterations. Therefore, we could set the restriction of $fr < P_{min}$, where *fr* is the fraction of samples to be changed to the minority class, and P_{min} the proportion of actual positive instances. This is the same restrictive threshold set of the Output Flipping algorithm, which has the problem of not allowing enough range of values for the parameter *fr* to reach the equal representation between classes.

4.2 SwitchingNED: Switching according to Nearest Enemy Distance

One way to mitigate the loss of importance of the real positive instances is to ensure that the switched negative examples are located as near as possible to these positive examples. This helps the classifiers to focus the attention on the real minority class subspace or its neighborhood, where the number of instances has suddenly grown.

We have designed a new way of selecting the instances to be switched, following the previous idea and avoiding the uniformly random way of the original Switching. The negative samples are chosen according to their proximity to the positive ones. Concretely, we define this proximity as the Nearest Enemy Distance (NED), the Euclidean distance to its closest positive sample. Farther a negative example is from the minority class, greater the distance to its nearest enemy is, and, consequently, it is less probable to switch it. We call this original Switching algorithm SwitchingNED. Its implementation is represented in Algorithm 1. In Lines 2-4, the switching probabilities ($Prob$) and number of switched examples are computed. In each iteration, the $fr * N$ negative instances are switched according to $Prob$ in dataset $\widehat{D}$, which is used to train a decision tree stored in $Trees$ (Lines 5-12). Finally, unknown examples are labeled with majority voting prediction of the trees in $Trees$.

SwitchingNED changes drastically how the instances are switched and moves away from the general concept of the Class Switching algorithm. The main idea behind the earlier randomizing output approaches was to introduce a controlled diversity to generate different perturbed training datasets, while dissipating it with the increasing of the ensemble, tending to a lower error. However, we want to use the switched examples to form a better class distribution, maintaining these changes few enough to avoid damaging the behaviour of the ensemble. Therefore, a totally random way of switching instances seems to be harmful for our purposes.

We implement this new switching technique with a probability distribution computed with the inverse of the NED of each majority instance. This distribution is used to choose the samples by roulette wheel selection in each iteration, as seen in Line 8. Algorithm 2 exhibits the computational process of the distribution. First, the nearest instance from the minority class is found for each sample of the majority class. The corresponding switching probability is set proportional to the inverse of the Euclidean distance to its nearest positive instance. These probabilities are normalized to sum one. This is a static process, computed once at the beginning of the algorithm, as shown in Line 3.

Summarizing, SwitchingNED is a novel and distinct ensemble that, as Bagging does with bootstrapping, uses the instance switching procedure to promote diversity along the different classifiers built. In addition, SwitchingNED has the property to properly learn from skewed data thanks to Nearest Enemy Distance.

Algorithm 1 SwitchingNED Algorithm.

function SWITCHINGNED(D - dataset, $nTrees$ - number of trees built, S - the predicted version of D, *fr* - switching rate, M - majority tag, m - minority tag)
  initialize: $S = \{\}$, $Trees[1..nTrees]$, $Prob[1..size(D)]$
  $Prob = NEDprob(D, M, m)$ ▷ Assign 0 to minority samples
  $\#Switched = fr * N$
  for i in [1,$nTrees$] **do**
    $\widehat{D} = D$
    for j in [1,$\#Switched$] **do**
      $t = RouletteSelection(Prob)$ ▷ Sample t from majority class
      $\widehat{D}_t^{class} = m$ ▷ Switched to minority class
    end for
    $Trees_i = BuildTree(\widehat{D})$
  end for
  for d in D **do**
    $\widehat{S_{i,d}} = Predict(Trees_i, d)$
  end for
  $S = PredictMajorityVoting(\widehat{S})$
  return S
end function

Algorithm 2 NED probabilities function.

function NEDPROB(D - dataset, M - majority tag, m - minority tag)
  initialize: $Prob[1..size(D)] = 0$, $Dist[1..size(D)]$
  for i in [1,$size(D)$] **do**
    if $D_i^{class} = M$ **then** ▷ Just for majority samples, else $Prob_i = 0$
      for j in [1,$size(D)$] **do**
        if $D_j^{class} = m$ **then**
          if $distance(D_i, D_j) < Dist_i$ **then**
            $Dist_i = distance(D_i, D_j)$
          end if
        end if
      end for
      $Prob_i = 1/Dist_i$
    end if
  end for
  $Normalize(Prob_i)$
  return $Prob_i$
end function

Figure 1 exemplifies SwitchingNED learning procedure from a given imbalanced dataset. As shown in Stage 1, SwitchingNED first computes the nearest enemies of all negative instances and, with the inverse of their distances, their probabilities of being switched. The resultant probabilities are sorted, normalized and prepared for the selection of $N * fr$ instances. In the exhibited array, the general positive, core and border negative samples are represented with their real switching probabilities and their states along the method. Positive instances are never selected, because their probabilities are set to 0. And border instances are more frequently switched than core ones.

In each iteration, a new modified training set is constructed by switching $N * fr$ negative instances to the minority class guided by the previous probability distribution. These sets are used to train several C4.5 trees which will infer different decision boundaries. This process is illustrated with the 3 scatter plots of the second step of Figure 1 (Stage 2). In those, one can be observe that certain groups of instances near to the borders are always switched. Additionally, big decision regions are inferred for the minority class. The erroneous ones do not overlap with each other, and as a result they will not matter when the predictions are aggregated.

Finally, the inferred C4.5 trees form an ensemble that decides the class predictions of unlabeled instances with majority voting scheme (Stage 3). The last illustration of Figure 1 manifests this scheme with the overlap of the different decision boundaries. The resultant model performs a nearly perfect classification of the data.

5 Experimental Framework, Results and Analysis

In this section, we present the experimental study carried out with all the class switching algorithms mentioned. In Subsection 5.1, we start by introducing the framework for this and posterior studies, pointing out the datasets, measures and general parameters used. Subsection 5.2 is dedicated to analyze the performance of SwitchingNED in comparison with the other Switching approaches. Finally, we present a graphical analysis of our method in Subsection 5.3.

5.1 Experimental Framework

The experimental framework has been completely extracted from [15], where a full study is carried out with the main state-of-the-art ensemble-based proposals. For our experiments, we use exactly the same 33 binary highly imbalanced datasets, i.e. with their IR greater than 9. This is a very diverse collection of datasets in terms of IR, going from barely reaching 9 till exceeding one hundred. The different datasets and their characteristics are presented in Table 1, ordered by their increasing IR. All these sets have been extracted from the UCI and KEEL repositories [37, 38].

They are multi-class standard classification datasets that have been adapted to conform binary imbalanced problems. This process mainly consists of labeling as positive class one or more small classes, and the rest as negative class.

Table 1 shows the name of the datasets, their number of instances (#Ex.), their number of attributes (#Att.), the percentage of examples of the minority (%min) and majority (%maj) classes, and their IR.

The execution of the algorithms has been carried out following a 5-fold stratified cross validation schema (5-fcv), with the same partitions used in [15] provided by the software KEEL [38]. These ensembles use randomness in some of their processes. Therefore, these executions have been repeated three times with different seeds to gather the average results.

As weak learners for these ensembles, decision tree C4.5 [39] has been used. Output switching algorithms have been presented with decision trees [16, 17]. This baseline classifier has been chosen for the most relevant ensembles in some studies of imbalanced scenarios [1, 14, 15].

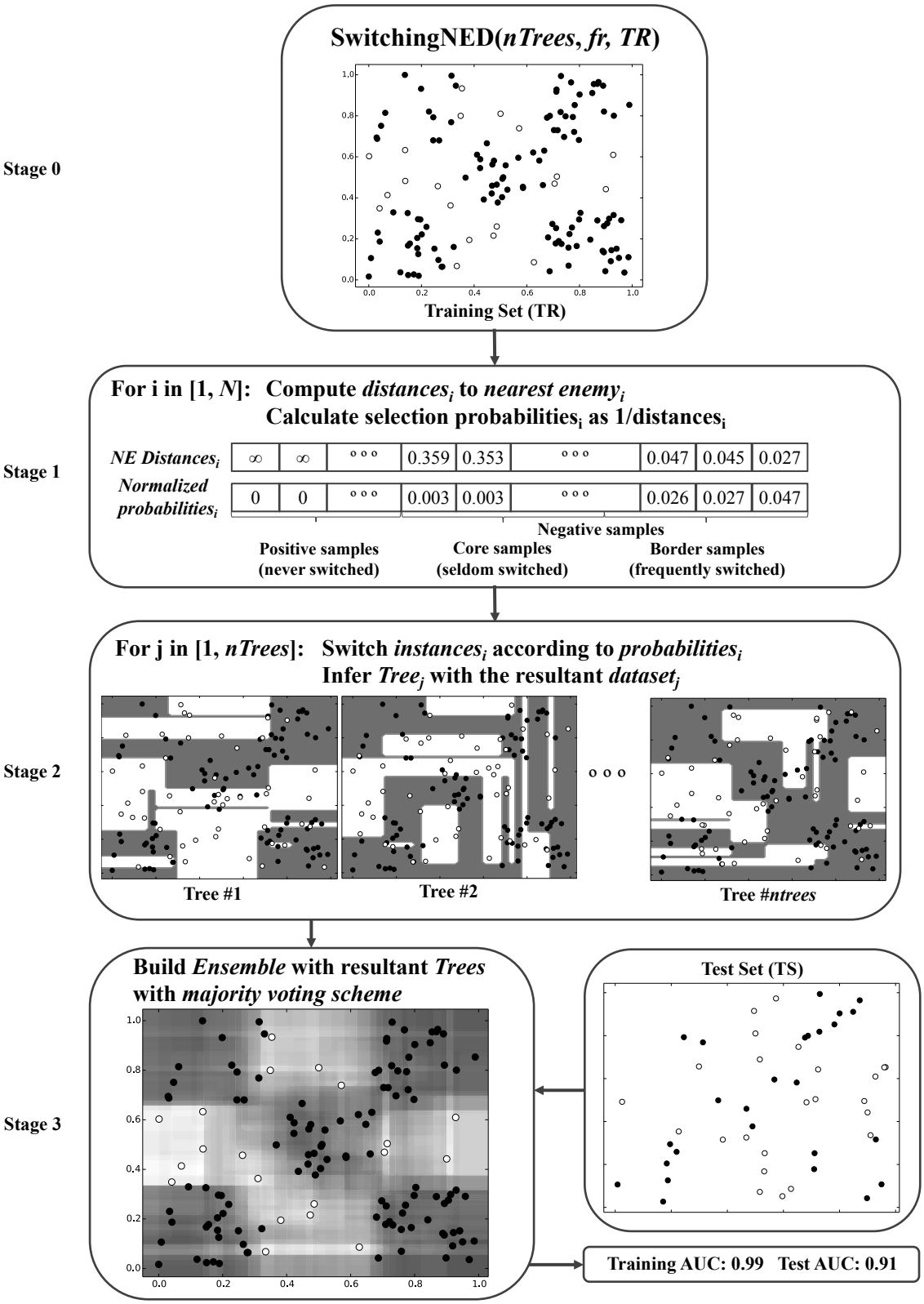


Figure 1: Learning procedure of SwitchingNED from an Imbalanced Chess3x3 Synthetic Data.

Table 1: Description of the 33 datasets used in the study.

No.	Datasets	#Ex.	#Atts.	(%min, %maj)	IR
1	*Glass04vs5*	92	9	(9.78, 90.22)	9.22
2	*Ecoli0346vs5*	205	7	(9.76, 90.24)	9.25
3	*Ecoli0347vs56*	257	7	(9.73, 90.27)	9.28
4	*Yeast05679vs4*	528	8	(9.66, 90.34)	9.35
5	*Ecoli067vs5*	220	6	(9.09, 90.91)	10.00
6	*Vowel0*	988	13	(9.01, 90.99)	10.10
7	*Glass016vs2*	192	9	(8.89, 91.11)	10.29
8	*Glass2*	214	9	(8.78, 91.22)	10.39
9	*Ecoli0147vs2356*	336	7	(8.63, 91.37)	10.59
10	*Led7Digit02456789vs1*	443	7	(8.35, 91.65)	10.97
11	*Ecoli01vs5*	108	9	(8.33, 91.67)	11.00
12	*Glass06vs5*	240	6	(8.33, 91.67)	11.00
13	*Glass0146vs2*	205	9	(8.29, 91.71)	11.06
14	*Ecoli0147vs56*	332	6	(7.53, 92.47)	12.28
15	*Cleveland0vs4*	177	13	(7.34, 92.66)	12.62
16	*Ecoli0146vs5*	280	6	(7.14, 92.86)	13.00
17	*Ecoli4*	336	7	(6.74, 93.26)	13.84
18	*Shuttle0vs4*	459	8	(6.72, 93.28)	13.87
19	*Yeast1vs7*	1829	9	(6.72, 93.28)	13.87
20	*Glass4*	214	9	(6.07, 93.93)	15.47
21	*Page-blocks13vs4*	472	10	(5.93, 94.07)	15.85
22	*Abalone918*	731	8	(5.65, 94.25)	16.68
23	*Glass016vs5*	184	9	(4.89, 95.11)	19.44
24	*Shuttle2vs4*	129	9	(4.65, 95.35)	20.50
25	*Yeast1458vs7*	693	8	(4.33, 95.67)	22.10
26	*Glass5*	214	9	(4.20, 95.80)	22.81
27	*Yeast2vs8*	482	8	(4.15, 95.85)	23.10
28	*Yeast4*	1484	8	(3.43, 96.57)	28.41
29	*Yeast1289vs7*	947	8	(3.17, 96.83)	30.56
30	*Yeast5*	1484	8	(2.96, 97.04)	32.78
31	*Ecoli0137vs26*	281	7	(2.49, 97.51)	39.15
32	*Yeast6*	1484	8	(2.49, 97.51)	39.15
33	*Abalone19*	4174	8	(0.77, 99.23)	128.87

Finally, as evaluation metric, we choose a widely used measure in imbalanced scenarios [1, 2, 14]: the above mentioned AUC [15]. We support the results with a non-parametric statistical analysis [22].

5.2 SwitchingNED for the class imbalance problem

The purpose of the following experiment is to test the performance of the new SwitchingNED with highly imbalanced datasets and to compare it with the original Switching and its modified version the Switching-I. The algorithms have been executed with different values for the parameter *fr* (0.1; 0.2; 0.3). However, we have fixed the number of classifiers to 40, a common value used in Bagging-based approaches. We wanted to seek an algorithm that fairly competes with the rest and not to exceed the number of trees where the rest of ensembles reach the over-fitting.

Table 2 contains the results for all datasets for each switching rate. These results show an overwhelming improvement of SwitchingNED over the original Switching and Switching-I. Even the small modification done with Switching-I results better than the Switching algorithm (see also Fig. 2) . Counting with the different *fr* configurations, the original Switching results better only in one dataset, being tied in 5. On the other hand, SwitchingNED beats both original Switching and the proposed modified version Switching-I in 20 different datasets. Additionally, all three average results are superior than the rest obtained by each different configurations of the other methods. These facts indicate a convincing superiority of the proposed SwitchingNED.

By prioritizing the exchange of instances near to the classification borders and just from the majority to the minority class, an equal representation for all classes of these highly imbalanced datasets can be achieved. This is supported by the Wilcoxon statistical test summarized in Table 3, which stresses this enhancement for different values of *fr*. Table 3 represents the hypothesis of equivalence for the pairwise comparison of the three algorithm. A rejected hypothesis means that there is a significant difference between the pair of algorithms.

Figure 2 exhibits the differences of average results obtained over the previous selected datasets by the three methods while *fr* grows. With a brief glance, the improvement of SwitchingNED over Switching is clearly visible. Looking deeply, two different behaviors can be observed. For Switching and Switching-I, the performance decreases with the growth of *fr* due to the loss of important information and the flooding problem. However, this declining tendency is broken by the SwitchingNED, because the impact of these two problems are mitigated when switching border samples. Better results are obtained for higher *fr* because it is easier to reach a balanced representation. Summarizing, SwitchingNED is, with any doubt, a better method to deal with imbalanced classification with strictly better performance and with a greater variety of *fr* values.

5.3 Graphical analysis of SwitchingNED

The purpose of this analysis is to improve the comprehension of the behavior of SwitchingNED and to further prove the superiority of SwitchingNED over the previous Switching, all with graphical case studies.

For our graphical case study, we chose a binary 2D synthetic data with 1000 examples forming a 5×5 chessboard. Then, we under-sampled the white class to only 12% of the instances, resulting a dataset with approx. 500 black and 60 white instances ($IR = 8.33$). Figure 3a and Figure 3b illustrate these balanced and imbalanced data with their classification boundaries inferred by C4.5 decision trees. Figure 3b exhibits the inability of the C4.5 to learn from the imbalanced data.

On the contrary, Figure 3c illustrates a better performance obtained with SwitchingNED. This figure was designed with the overlap of the classification boundaries inferred by 40 C4.5s within SwitchingNED ensemble. Its gradient map represents the probability of belonging to the minority class. With a quick look, the 5×5 grid is clearly perceived, which represents a better generalization of the data compared to Figure 3b. This is also supported by the AUC results obtained.

Table 2: AUC results for the original Switching, Switching-I and SwitchingNED. The best global results are stressed in bold-face.

	Switching			**Switching-I**			**SwitchingNED**		
	$\boldsymbol{fr} = 0.1$	$\boldsymbol{fr} = 0.2$	$\boldsymbol{fr} = 0.3$	$\boldsymbol{fr} = 0.1$	$\boldsymbol{fr} = 0.2$	$\boldsymbol{fr} = 0.3$	$\boldsymbol{fr} = 0.1$	$\boldsymbol{fr} = 0.2$	$\boldsymbol{fr} = 0.3$
Glass04vs5	**0.9941**	0.9775	0.5500	**0.9941**	**0.9941**	**0.9941**	**0.9941**	**0.9941**	**0.9941**
Ecoli0346vs5	0.8511	0.864	0.8083	0.8419	0.8577	**0.8743**	0.8401	0.8502	0.8586
Ecoli0347vs56	0.769	0.7809	0.6533	0.7557	0.7736	0.7995	0.7550	0.7876	**0.8529**
Yeast05679vs4	0.6741	0.6316	0.5063	0.6974	0.7186	0.7405	0.7003	0.7559	**0.7898**
Ecoli067vs5	0.8342	0.8425	0.6742	0.7925	0.8258	0.8425	0.7925	0.8258	**0.8767**
Vowel0	0.9661	0.9100	0.8467	0.9685	0.9674	0.9598	0.9713	**0.9813**	0.9749
Glass016vs2	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	**0.5613**
Glass2	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.4974	**0.6118**
Ecoli0147vs2356	0.7323	0.7378	0.5211	0.7156	0.7429	0.7418	0.7656	0.7823	**0.7904**
Led7Digit02456789vs1	0.8959	0.9001	0.8698	0.8955	**0.908**	0.9056	0.8955	0.8989	0.9034
Ecoli01vs5	0.8280	0.8129	0.6894	0.8682	0.8432	0.8773	0.8727	0.8879	**0.8955**
Glass06vs5	0.945	0.5167	0.5000	**0.995**	0.9283	0.7500	**0.995**	0.9617	0.865
Glass0146vs2	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5083	0.5083	**0.6327**
Ecoli0147vs56	0.7434	0.7445	0.5133	0.7367	0.7695	0.7751	0.7501	0.7901	**0.8196**
Cleveland0vs4	0.6384	0.6081	0.6409	0.6955	0.7924	0.8655	0.6843	0.8116	**0.8926**
Ecoli0146vs5	0.7692	0.7699	0.6833	0.7756	0.7929	0.7917	0.7846	0.8109	**0.8186**
Ecoli4	0.8275	0.8286	0.5667	0.8093	0.8416	0.8400	0.8176	0.8394	**0.8844**
Shuttle0vs4	0.9997	0.9997	**0.9999**	0.9997	0.9997	0.9997	0.9997	0.9997	0.9996
Yeast1vs7	0.5488	0.5000	0.5000	0.5806	0.5413	0.5211	0.5655	0.5329	**0.6155**
Glass4	0.6600	0.5000	0.5000	**0.7503**	0.5736	0.5483	0.7414	0.7236	0.6506
Page-blocks13vs4	0.9489	0.7889	0.5067	0.9911	0.9133	0.8700	0.9978	**0.9981**	0.9891
Abalone918	0.5516	0.5000	0.5000	**0.5755**	0.5358	0.5238	0.5649	0.5354	0.5194
Glass016vs5	0.7814	0.500	0.5000	0.8924	0.6314	0.5000	**0.8952**	0.6657	0.5500
Shuttle2vs4	**1.0000**	0.95	0.5833	**1.0000**	0.9833	**1.0000**	**1.0000**	**1.0000**	0.9793
Yeast1458vs7	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**
Glass5	0.6817	0.5000	0.5000	**0.9142**	0.6167	0.4992	0.8642	0.6476	0.5000
Yeast2vs8	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**
Yeast4	0.5578	0.5000	0.5000	0.5765	0.5720	0.6111	0.6060	0.6085	**0.6968**
Yeast1289vs7	0.5165	0.5000	0.5000	**0.5326**	0.5165	0.5000	0.5219	0.5000	0.5000
Yeast5	0.8767	0.7207	0.5193	0.8767	0.8712	0.9074	0.9122	0.9471	**0.9603**
Ecoli0137vs26	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.7000	0.6167	**0.749**
Yeast6	0.6560	0.5000	0.5000	0.7075	0.6978	0.7012	0.7547	0.7944	**0.8423**
Abalone19	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	0.4965
Avg:	0.7196	0.6632	0.5798	0.7406	0.7184	0.7103	0.7500	0.7440	**0.7597**

Table 3: Summary of the results obtained by a Wilcoxon test over Switching algorithms: Switching(1), Switching-I(2) and SwitchingNED(3)

$\boldsymbol{fr} = 0.1$				
Comparison	R^+	R^-	**Hypothesis** ($\alpha = 0.05$)	p-**Value**
(2)vs(1)	382.5	145.5	**Rejected**	0.02607
(3)vs(1)	465.0	96.0	**Rejected**	0.000948
(3)vs(2)	331.5	196.5	Not Rejected	0.201426
$\boldsymbol{fr} = 0.2$				
Comparison	R^+	R^-	**Hypothesis** ($\alpha = 0.05$)	p-**Value**
(2)vs(1)	506.0	55.0	**Rejected**	0.000054
(3)vs(1)	509.5	51.5	**Rejected**	0.000041
(3)vs(2)	442.5	85.5	**Rejected**	0.000817
$\boldsymbol{fr} = 0.3$				
Comparison	R^+	R^-	**Hypothesis** ($\alpha = 0.05$)	p-**Value**
(2)vs(1)	491.0	37.0	**Rejected**	0.000021
(3)vs(1)	545.0	16.0	**Rejected**	0.000002
(3)vs(2)	503.0	58.0	**Rejected**	0.000068

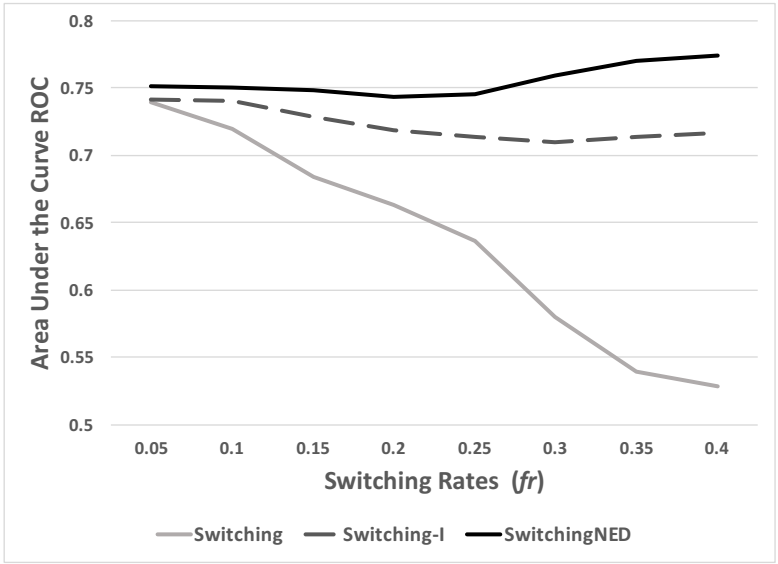
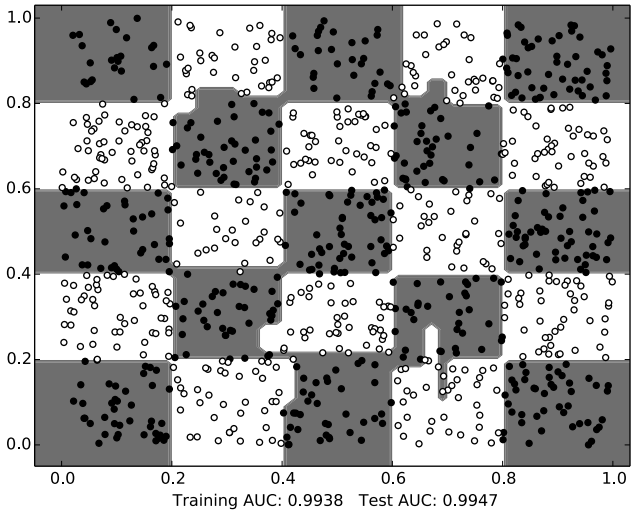
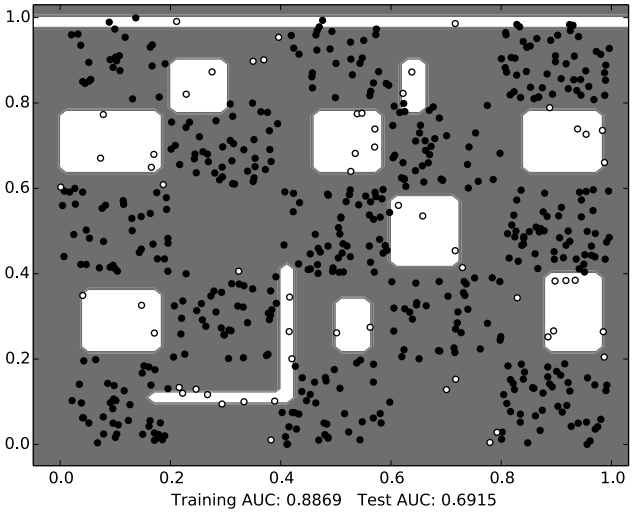


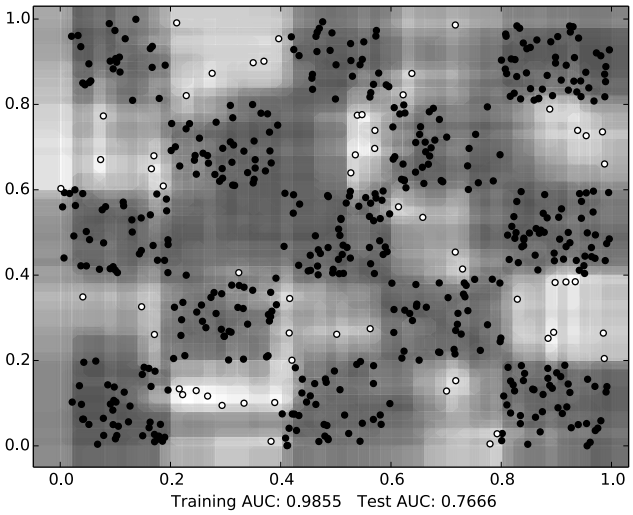
Figure 2: AUC results of Switching algorithms with different *fr* settings.



(a) C4.5 with Chess5×5 Synthetic Data.

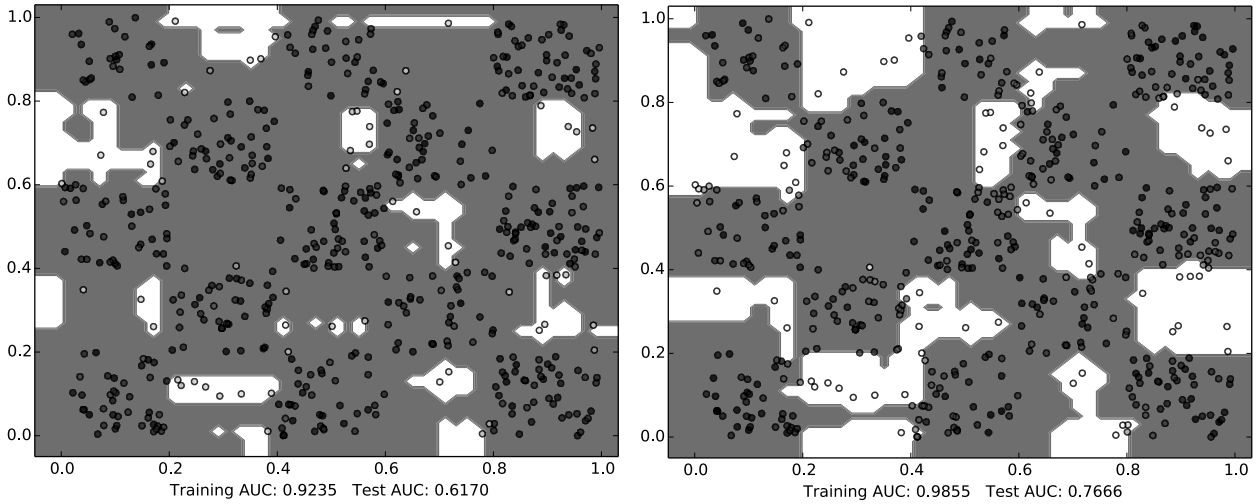


(b) C4.5 with Imbalanced Chess5×5 Synthetic Data.



(c) Boundaries aggregation of 40 SwitchingNED trees.

Figure 3: Decision surfaces over Chess5x5 Synthetic Data.



(a) Decision surface of Switching with 40 trees and $fr = 0.2$.

(b) Decision surface of SwitchingNED with 40 trees and $fr = 0.2$.

Figure 4: Behavior comparison of Switching and SwitchingNED.

Figure 4 exhibits the classification boundaries obtained from Switching and SwitchingNED with the same parameter settings. SwitchingNED (4b) achieves greater decision surfaces for the white class than Switching does (4a). Therefore, the prediction of the minority class is enhanced without compromising the specificity.

6 Preprocessing techniques in SwitchingNED

Traditionally, the ensemble approaches have been combined with preprocessing techniques to improve their performance in the imbalanced class problem [14]. In this section, we test the integration of these methods in SwitchingNED. Subsection 6.1. illustrates how the pre-processing method takes part in balancing the data, together with the exchange of class labels of the SwitchingNED and the importance of the parameter *fr* in this process. We will determinate the responsibility of each of these two methods during this procedure, and specify which will be prioritized when the inequality between classes is not so high. Later on, in Subsection 6.2, we analyse the results obtained by SwitchingNED with different preprocessing techniques. Finally, we test the best Sampling-SwitchingNED approach against the most successful method in the literature in Subsection 6.3.

6.1 Data Sampling for SwitchingNED

SwitchingNED permits to mitigate the flooding impact, thanks to switching mainly the instances near to the classification boundaries. However, this algorithm still has difficulties with very highly imbalanced datasets, where nearly half of the negative instances has to be switched to achieve an equal representation between classes. The final solution is to fuse the sample re-balancing of SwitchingNED with a standard preprocessing algorithm, as has been done with other ensembles [14] such as Bagging with Under-sampling [18].

By slightly balancing the classes with traditional sampling techniques before switching, fewer switched instances are needed to find an equal representation between classes, reducing the possibility of flooding and converging easily. On the other hand, using of Switching as an equalization method allows to reduce possible mistakes introduced by the preprocessing mechanisms, by reducing the amount of instances removed from the majority class (under-sampling) or the synthetic positive samples (over-sampling).

This is accomplished by over- or under-sampling until SwitchingNED with a fixed value of parameter *fr* can completely equalize the class proportions. i.e. if the original distribution is 20%-80% and the value of *fr* is 0.2, the preprocessing algorithm will change the proportions to 30%-70%, allowing Switching to completely balance the data when 20% of the instances are switched to the minority class. The ideal balance is considered at 1:1, when the majority and minority classes have exactly the same number of instances. Formally, SwitchingNED will work with most of the under-/over-sampling techniques, such as Random Under-Sampling, Random Over-Sampling, and SMOTE. Algorithm 3 illustrates this approach, where any sampling can be introduced.

Our algorithm gives priority to the SwitchingNED selection with respect to the sampling method. The NED distribution is computed independently at the beginning of the algorithm (Line 3), i.e., the probability distribution will not change with the distribution of the data after sampling. Furthermore, the instances are first selected by roulette wheel and switched, as indicated in Line 14. Then, sampling is executed over the remaining original instances (Line 17). If the value of *fr* is high enough to reach the equilibrium by itself, the preprocessing algorithm will not be triggered. Therefore, the execution of this preprocessing technique is subject to the following restriction:

$$fr < (P_M - P_m)/2 \tag{4}$$

where P_M and P_m are the proportion of the entire dataset samples belonging to the majority and minority classes, respectively. This is represented in Line 4.

Due to this fact, *fr* becomes a quite important parameter of the algorithm. It establishes not only the number of the instances to be switched, but also in which grade the preprocessing algorithm takes part in the balancing task, as shown in Lines 4-11. Greater the value of *fr* is, lesser is the relevance of the preprocessing procedure. The following mathematical expressions represent the quantity of instances that are switched and the amount of sampled ones with relation to the parameter *fr*:

$$\begin{aligned} &\#Switched = fr * N * P_m / P'_m \\ &\#Sampled = N * P_M - (N * P_m + \#Switched * 2) \\ &\quad \text{where} \quad P'_m = 0.5 - fr \end{aligned} \tag{5}$$

where P'_m is the proportion of the minority class after the preprocessing stage.

Thanks to this hybridization, the previously mentioned constraint seems to be less limiting, 0.25 being the maximum suitable value for *fr* to not surpass the positive instances with the switched examples. *fr* should be set by the user, depending of the necessities of the concrete problem. For general purposes, we recommend to set it at 0.1.

6.2 Evaluation of Sampling techniques for SwitchingNED

This experimental study has been carried out to ensure the best combination of the most popular sampling approaches (Random Under-Sampling, Random Over-Sampling and SMOTE) with SwitchingNED: USwitchingNED, OSwitchingNED and SMTSwitchingNED, respectively. The number of classifiers was fixed to 40. The number of neighbors considered by SMOTE was 5, and the distance used was the Heterogeneous Value Difference Metric (HVDM).

Table 4 collects the AUC results for the best configuration of the three explored. The value chosen for the parameter *fr* was 0.1, that recommended for general purposes.

Algorithm 3 SwitchingNED Algorithm with Sampling techniques.

function SWITCHING(D - dataset, $nTrees$ - number of trees built, S - the predicted version of D, *fr* - switching rate, M - majority tag, m - minority tag)
  initialize: $S = \{\}$, $Trees[1..nTrees]$, $Prob[1..size(D)]$
  $Prob = NEDprob(D, M, m)$
  if $fr < (P_M - P_m)/2$ **then**
    $newP_m = 0.5 - fr$
    $\#Switched = fr * N * P_m / newP_m$
    $\#Sampled = N * P_M - (N * P_m + \#Switched * 2)$
  else
    $\#Switched = fr * N$
    $\#Sampled = 0$ ▷ Preprocessing not triggered
  end if
  for i in [1,$nTrees$] **do**
    $\widehat{D} = D$
    for j in [1,$\#Switched$] **do** ▷ Switching Balancing stage
      $t = RouletteSelection(Prob)$
      $\widehat{D}_t^{class} = m$
    end for ▷ Preprocessing Balancing stage

    $\widehat{D} = Sampling(\widehat{D}, \#Sampled, M)$
    $Trees_i = BuildTree(\widehat{D})$
  end for
  for d in D **do**
    $\widehat{S_{i,d}} = Predict(Trees_i, d)$
  end for
  $S = PredictMajorityVoting(\widehat{S})$
  return S
end function

It can be observed in Table 4 that the results of USwitchingNED are overwhelmingly better than those obtained by Switching-I or SwitchingNED. This is due to the possibility of reaching the equal representation for all classes together with the more suitable way to switch the instances, avoiding any drawback of the two techniques that are applied.

In particular, the superiority of the USwitchingNED over the rest has to be stressed. As shown in Table 4, this approach obtained the best results in the majority of the datasets, with the exception of 12 sets. Its average results are also significantly higher than the two oversampling-based approaches.

The Wilcoxon statistical test points out the dominance of USwitchingNED over the rest. In Table 5, we observe how the ranks of USwitchingNED in each comparison are widely greater than the others, resulting in very low p-Values. This indicates that USwitchingNED is the best performing method.

Before comparing USwitchingNED with the State-of-the-art, the sensitivities of the ensemble to parameter settings have to be explored. USwitchingNED has two main parameters that are highly relevant and influential to its performance: the number of trees or iterations (nTrees) and the switching rate (*fr*).

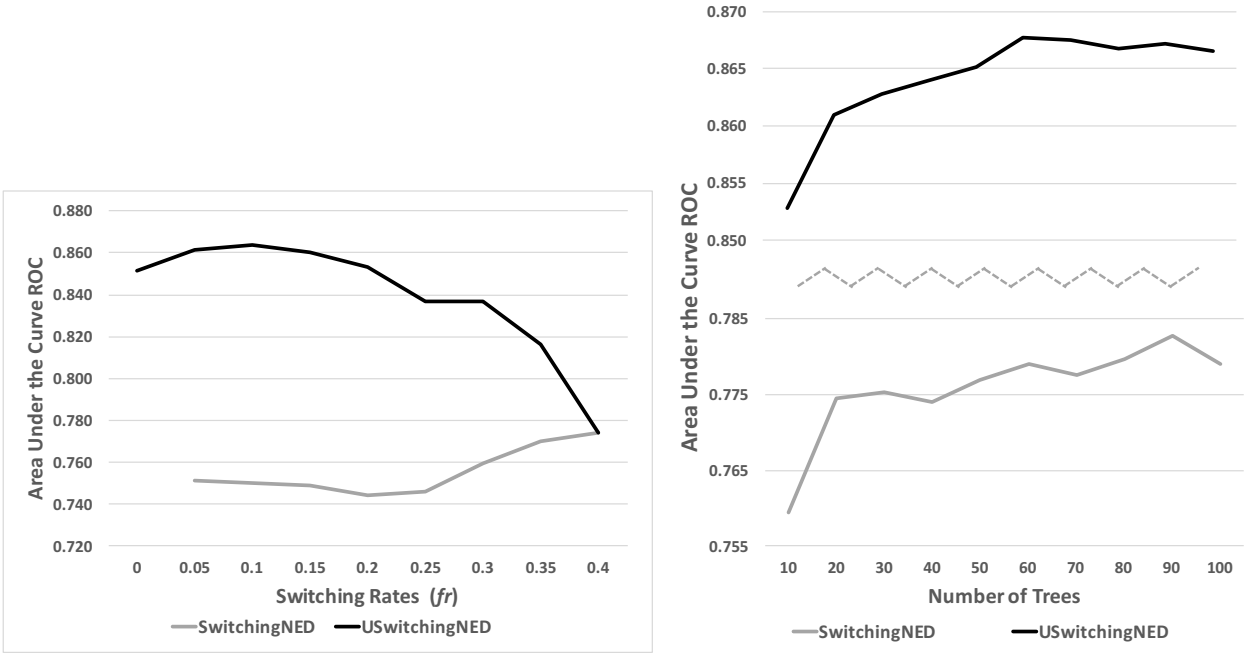
Figure 5 shows the performance of USwitchingNED and SwitchingNED in terms of the average of the AUC results of the 33 selected datasets. The performance is shown based on the number of trees and on different *fr*

Table 4: AUC results for the different SwitchingNED approaches.

	USwitchingNED	**SMTSwitchingNED**	**OSwitchingNED**
Glass04vs5	**0.9941**	**0.9941**	**0.9941**
Ecoli0346vs5	**0.9133**	0.9032	0.8939
Ecoli0347vs56	**0.9076**	0.9073	0.8694
Yeast05679vs4	0.8015	**0.8046**	0.7506
Ecoli067vs5	**0.9025**	0.8967	0.8758
Vowel0	0.9516	**0.9581**	0.9529
Glass016vs2	**0.7003**	0.6393	0.6138
Glass2	**0.7475**	0.7394	0.7109
Ecoli0147vs2356	0**.8872**	0.8716	0.8787
Led7Digit02456789vs1	0.8801	**0.9043**	0.8805
Ecoli01vs5	**0.9114**	0.8644	0.8598
Glass06vs5	**0.9950**	**0.9950**	**0.9950**
Glass0146vs2	0.7542	**0.7878**	0.7406
Ecoli0147vs56	**0.9096**	0.8889	0.8610
Cleveland0vs4	**0.8840**	0.7444	0.7975
Ecoli0146vs5	**0.9096**	0.8897	0.8519
Ecoli4	**0.9141**	0.8623	0.8357
Shuttle0vs4	**1.0000**	0.9997	0.9997
Yeast1vs7	**0.7782**	0.7592	0.6691
Glass4	**0.9001**	0.8958	0.8617
Page-blocks13vs4	0.9805	**0.9978**	**0.9978**
Abalone918	0.7189	**0.7294**	0.6368
Glass016vs5	**0.9714**	0.8943	0.8943
Shuttle2vs4	0.9918	**1.0000**	**1.0000**
Yeast1458vs7	0.5887	**0.5918**	0.5503
Glass5	0.9724	0.9451	**0.9878**
Yeast2vs8	0.7718	**0.7867**	0.7848
Yeast4	**0.8415**	0.7642	0.6887
Yeast1289vs7	**0.7020**	0.6479	0.6377
Yeast5	0.9569	**0.9740**	0.9439
Ecoli0137vs26	0.7935	**0.8354**	0.8348
Yeast6	**0.8808**	0.8144	0.8210
Abalone19	**0.6991**	0.5611	0.5339
Avg:	**0.8640**	0.8439	0.8244

Table 5: Summary of Wilcoxon test over SwitchingNED with different sampling techniques.

Comparison	R^+	R^-	**Hypothesis** ($\alpha = 0.05$)	p-**Value**
SMTSwitchingNED vs OSwitchingNED	503.5	91.5	**Rejected**	0.00022
USwitchingNED vs OSwitchingNED	527.0	68.0	**Rejected**	0.00003
USwitchingNED vs SMTSwitchingNED	431.5	163.5	**Rejected**	0.02109



(a) Average AUC depending on the switching rate (*fr*). (b) Average AUC depending on the number of trees ($nTrees$).

Figure 5: Performance of SwitchingNED and USwitchingNED with different parameter settings.

values, letting the other parameter fixed: $fr = 0.1$ for USwitchingNED and $fr = 0.4$ for SwitchingNED (see Sec. 5.2) and 40 trees for both.

According to number of iterations, our methods have an equivalent pattern to every other ensemble. They reach over-fitting when the number of trees exceeds a certain limit. As shown in Figure 5b, the limit of USwitchingNED is reached approx. at 60 trees. However, following the study [14, 15], we decided to use 40 trees to ensure a fair comparison.

Analyzing the impact of parameter *fr* for USwitchingNED (Fig. 5a), the best results are obtained with $fr = 0.1$. And we can certainly declare that the performance of USwitchingNED is highly robust when *fr* is selected from 0.05 to 0.2. With this experiment, we can conclude that the parameter setting was properly chosen.

It is worthy to mention that USwitchingNED with *fr* between 0.05 and 0.2 results significantly better than with $fr = 0.0$. USwitchingNED with $fr = 0.0$ has a similar behavior to UnderBagging [18] and USwitchingNED with $fr = 0.1$ is overwhelming superior as shown in Section 6.3. This fact proves that the combination of Under-sampling and Switching is worthy.

Contrary to SwitchingNED (Fig. 2), USwitchingNED performance declines when *fr* exceeds 0.2 (Fig. 5a). The interaction between switching re-balance procedure and under-sampling completely changes the ensemble behavior, therefore such differences were expected. One should notice that USwitchingNED is strictly better than SwitchingNED because the darker curve (USwitchingNED) is, in every moment, over the lighter one (SwitchingNED) in Fig. 5a

6.3 USwitchingNED vs State-of-the-art

Subsequently to identify our best approach based on Switching ensembles, we analyze the importance of USwitchingNED with respect to previously proposed ensembles in the literature. We have tested five different ensembles. Each one is a representative method of a distinct combination of Under-/Over- Sampling and

Table 6: Ensemble-based proposals considered.

Abbreviation	**Algorithm**	**Short description**
EUSB	EUSBoost [15]	AdaBoost.M2 with evolutionary under-sampling
UBAG	UnderBagging [18]	Bagging with random under-sampling
SBAG	SMOTEBagging [20]	Bagging with over-sampling with SMOTE
SBO	SMOTEBoost [19]	AdaBoost.M2 with over-sampling with SMOTE
EASY	EasyEnsemble [21]	Hybrid of under-sampling, Bagging and AdaBoost

Table 7: Parameters considered for the compared methods.

Algorithm	**Parameters**
C4.5	itemsetsPerLeaf=2, confidence=0.25, prune=True
EUSBoost	populationSize=50, #evaluations=10000, HUXprobability=0.25, fitness=EUSB$_Q$, distance=Euclidean, balancing=True, P=0.2

Bagging or Boosting. Table 6 summarizes the tested methods. In [15], EUSBoost has been proved to be the best performing ensemble for highly imbalanced classification.

In order to prepare the fairest comparison, we use exactly the same experimental framework of the EUSBoost paper [15], which was explained in Subsection 5.1. The parameters for the algorithms, represented in Table 7, was also extracted from that earlier study.

The number of trees generated by the ensembles have being chosen according to the convergence of each algorithm. Generally, Boosting-based ensembles reach their maximum potential earlier than Bagging approaches, as shown in [14]. Therefore, we contemplate 10 classifiers for Boosting-based ensembles, and 40 for USwitchingNED and similar Bagging approaches.

Table 8 gathers the obtained results in terms of AUC for the 33 selected datasets and the 6 tested methods. In it, the best results are stressed in bold-face. Even though the best results are distributed between the different approaches, USwitchingNED has a certain superiority over the rest. USwitchingNED is the method with the greater number of best results recalled, a total of 12 datasets. And additionally, it has the best performance on average.

The USwitchingNED superiority is also supported by the Wilcoxon statistical test in Table 9. In this test, the previous proposals are independently compared to USwitchingNED. The obtained ranks (R^+ and R^-) point USwitchingNED out as the best performing method, because the USwicthingNED rank (R^+) is always widely higher than the rank of other compared method (R^-). In order to reaffirm this hypothesis, the p-Value obtained must be smaller than 0.05. This is accomplished in every comparison, with exception of EUSBoost. Therefore, a deeper analysis of the comparison of USwitchingNED and EUSBoost must be done.

Table 10 collects the AUC results of USwitchingNED and EUSBoost, which stresses the superiority of the former over the latter. There are 18 sets where USwitchingNED obtains better results, other 12 sets where EUSBoost results are better, and 3 where they stay tied. On the average, USwitchingNED is still the best performer method.

USwitchingNED has a clear advantage over EUSBoost in terms of computational complexity. The EUSBoost training phase is computationally more expensive than training USwitchingNED. This is caused by the execution of the genetic algorithm EUS [34] in each iteration of Boosting.

Table 8: AUC results for the tested algorithms. The best results are stressed in bold-face.

	USwitchingNED	**EUSB**	**UBAG**	**SBAG**	**SBO**	**EASY**
Glass04vs5	**0.9941**	**0.9941**	**0.9941**	0.9819	0.9799	**0.9941**
Ecoli0346vs5	0.9133	0.8919	0.889	0.9173	**0.918**	0.8678
Ecoli0347vs56	**0.9076**	0.8842	0.868	0.8753	0.8918	0.8659
Yeast05679vs4	0.8015	0.7869	0.7991	**0.8093**	0.7838	0.7702
Ecoli067vs5	**0.9025**	0.8767	0.8908	0.8417	0.8567	0.8567
Vowel0	0.9516	0.9664	0.9461	**0.9826**	0.9822	0.9449
Glass016vs2	0.7003	0.7264	0.7193	0.6348	0.644	**0.7533**
Glass2	0.7475	0.7281	0.7756	**0.7951**	0.7502	0.7247
Ecoli0147vs2356	**0.8872**	0.8782	0.8427	0.8867	0.8732	0.8622
Led7Digit02456789vs1	0.8801	0.8755	0.8662	**0.8871**	0.7467	0.8734
Ecoli01vs5	**0.9114**	0.8848	0.8311	0.8515	0.847	0.891
Glass06vs5	**0.995**	**0.995**	0.9139	0.975	0.9664	0.847
Glass0146vs2	0.7542	**0.7768**	0.7695	0.7602	0.6872	0.7533
Ecoli0147vs56	**0.9096**	0.8941	0.8715	0.8474	0.8627	0.8411
Cleveland0vs4	**0.884**	0.8164	0.8141	0.7954	0.7752	0.8022
Ecoli0146vs5	0.9096	0.8994	0.8968	**0.916**	0.9135	0.8282
Ecoli4	0.9141	**0.9215**	0.8825	0.9143	0.8847	0.8782
Shuttle0vs4	**1.0**	**1.0**	**1.0**	0.9995	**1.0**	**1.0**
Yeast1vs7	**0.7782**	0.7644	0.7679	0.6797	0.7048	0.7167
Glass4	0.9001	0.9056	0.872	0.8742	**0.909**	0.8813
Page-blocks13vs4	0.9805	0.9869	0.9767	0.9891	**0.9951**	0.9711
Abalone918	0.7189	0.7119	0.7316	**0.7424**	0.7345	0.7223
Glass016vs5	0.9714	**0.9886**	0.9429	0.8714	0.9652	0.9524
Shuttle2vs4	0.9918	**1.0**	**1.0**	**1.0**	**1.0**	0.9905
Yeast1458vs7	0.5887	0.5948	**0.6238**	0.6137	0.5579	0.58
Glass5	0.9724	**0.9878**	0.9488	0.8947	0.9797	0.952
Yeast2vs8	0.7718	0.7636	0.7666	**0.7848**	0.7827	0.732
Yeast4	0.8415	0.8299	**0.8449**	0.776	0.6954	0.8317
Yeast1289vs7	0.702	0.7184	**0.7642**	0.6543	0.6559	0.6798
Yeast5	0.9569	0.9382	0.9571	**0.9705**	0.9128	0.9502
Ecoli0137vs26	0.7935	0.8087	0.736	0.8318	**0.836**	0.7281
Yeast6	**0.8808**	0.8601	0.865	0.8392	0.8109	0.8573
Abalone19	**0.6991**	0.673	0.684	0.5767	0.5103	0.698
Avg:	**0.8640**	0.8584	0.8500	0.8415	0.8307	0.8363

Table 9: Summary of Wilcoxon test of the comparison USwitchingNED versus the considered ensembles.

USwitchingNED vs	R^+	R^-	**Hypothesis** ($\alpha = 0.05$)	p-**Value**
EUSB	347.0	181.0	Not rejected	0.12404
UB	420.5	140.5	**Rejected**	0.01132
SB	395.0	166.0	**Rejected**	0.04054
SBO	437.0	91.0	**Rejected**	0.00077
EASY	526.5	34.5	**Rejected**	0.00001

Table 10: AUC results for USwitchingNED and EUSBoost

	EUSBoost	**USwitchingNED**
Glass04vs5	**0.9941**	**0.9941**
Ecoli0346vs5	0.8919	**0.9133**
Ecoli0347vs56	0.8842	**0.9076**
Yeast05679vs4	0.7869	**0.8015**
Ecoli067vs5	0.8767	**0.9025**
Vowel0	**0.9664**	0.9516
Glass016vs2	**0.7264**	0.7003
Glass2	0.7281	**0.7475**
Ecoli0147vs2356	0.8782	**0.8872**
Led7Digit02456789vs1	0.8755	**0.8801**
Ecoli01vs5	0.8848	**0.9114**
Glass06vs5	**0.9950**	**0.9950**
Glass0146vs2	**0.7768**	0.7542
Ecoli0147vs56	0.8941	**0.9096**
Cleveland0vs4	0.8164	**0.8840**
Ecoli0146vs5	0.8994	**0.9096**
Ecoli4	**0.9215**	0.9141
Shuttle0vs4	**1.0000**	**1.0000**
Yeast1vs7	0.7644	**0.7782**
Glass4	**0.9056**	0.9001
Page-blocks13vs4	**0.9869**	0.9805
Abalone918	0.7119	**0.7189**
Glass016vs5	**0.9886**	0.9714
Shuttle2vs4	**1.0000**	0.9918
Yeast1458vs7	**0.5948**	0.5887
Glass5	**0.9878**	0.9724
Yeast2vs8	0.7636	**0.7718**
Yeast4	0.8299	**0.8415**
Yeast1289vs7	**0.7184**	0.7020
Yeast5	0.9382	**0.9569**
Ecoli0137vs26	**0.8087**	0.7935
Yeast6	0.8601	**0.8808**
Abalone19	0.6730	**0.6991**
Avg:	0.8584	**0.8640**

The main time-consuming operation in EUS is the evaluation of the chromosomes using the Geometric Mean of an 1NN. The 1NN complexity is linear with respect to the number of instances and attributes, i.e., $\mathcal{O}(n_{ref} \cdot A)$, where n_{ref} is the number of the reference set (in this case, the undersampled one, which equals two times the minority class size, m) and A the number of attributes. The evaluation is done using a hold-one-out scheme, therefore, the 1NN is executed N times (the number of examples of the original dataset), resulting in a complexity of $\mathcal{O}(N \cdot 2m \cdot A)$. This process is repeated for every iteration and every chromosome, till reaching the maximum number of allowed evaluations (max_{eval}, 10000 in this experiments). Summarizing, the complexity of each EUSBoost iteration is approximately $\mathcal{O}(N \cdot 2m \cdot A \cdot max_{eval})$.

On the contrary, USwitchingNED lacks of this overhead cost in each iteration. The main costly function is the random selection of the instances by the Roulette Selection algorithm. If this selection is implemented with an ordered structure, it has a complexity of $\mathcal{O}(\log n)$, using a binary search. Even less, because the minority samples are excluded. The probability distribution used is computed once at the beginning, consuming just an 1NN process outside the main loop of the ensemble. The computational complexity of this 1NN and the sorting by QuickSort of the weight distribution is $\mathcal{O}(M \cdot m \cdot A + M \cdot \log M)$, where M is the number of negative examples and m the number of positives. None of them are more expensive than the over-cost imposed by EUS.

7 Concluding Remarks

We have proposed a novel ensemble approach based on Switching ensembles to deal with highly imbalanced datasets. SwitchingNED represents an improvement due to its novel way of guiding the switching process based on the Nearest Enemy Distance, which changes entirely the concept of Class Switching algorithms. This technique permits to utilize the switched instances as a way of balancing the class distribution by ensuring that they are near to the classification boundary, reducing the induced noise and maintaining the minority class information. SwitchingNED mitigates the limitations of the switching approaches for highly imbalanced scenarios, freeing its switching fraction parameter, and resulting very promising in terms of performance.

Subsequently, we have explored three different approaches based on these previous improvements and the most popular sampling techniques: Random Under/Over-Sampling and SMOTE. These techniques help SwitchingNED to balance the class distribution of really high imbalanced datasets, and they are triggered just when are needed. After performing an experimental study, the Under-Sampling SwitchingNED technique (USwitchingNED) turned out to be the most promising from the three ensembles. Finally, a comparison versus a representative set of the combinations of preprocessing and ensembles was carried out. The quality of USwitchingNED was clarified by the empirical results, settling it as one of best approaches on the field.

Preprocessing-based ensembles have been one of essential manners to address the imbalanced class problem. SwitchingNED, as a new ensemble, opens new opportunities and possibilities up in the context of imbalanced data. Other ensemble schemes within the scope of class switching can be explored to find better rebalanced class distributions. For example, boosting scheme could be applied in conjunction with NED probabilities to estimate the best instances to be switched. Additionally, the most recent models of artificial instances generation would be interesting to be explored with a switching-based setting.

A Formulation on the constraint of parameter *fr*

To avoid the flooding problem, each class should have a majority of real instances, not switched, That is, the number of instances of a class minus the examples switched of this class must be greater than the switched

examples from other classes. Assuming that the class distributions are maintained in the uniformly random selection of instances to switch, the resulting expression is the following:

$$n - fr*n > \frac{(N-n)*fr}{K-1} \tag{6}$$

where n is the number of instances of the minority class, N is the total number of examples of the dataset, K is the number of classes, and *fr* denotes the switching rate. This formula is expressed with the proportion of minority class, because it is the most vulnerable to the flooding problem and proportionally it will receive much more instances than the rest. The left side of the expression represents the number of unchanged instances from the minority class, while the right side is the number of instances switched from other classes to the minority class. After performing some simplifications (Eq. 7), the final expression shown in Eq. 8 limits the parameter *fr* to consider the integrity of the minority class.

$$\begin{aligned}
&n*(1-fr)(K-1) > fr*(N-n) \\
&(1-fr)(K-1) > fr*(\frac{N}{n}-1) \quad \frac{N}{n} = \frac{1}{P_m} \\
&k-1-fr*k+fr > \frac{fr}{P_m} - fr \\
&k-1-fr*k > \frac{fr}{P_m} - 2fr \\
&k-1 > \frac{fr}{P_m} - 2fr + fr*K \\
&k-1 > fr*(\frac{1}{P_m} + K - 2)
\end{aligned} \tag{7}$$

$$fr < \frac{K-1}{(P_m)^{-1} + K - 2} \tag{8}$$

Acknowledgements

This work is supported by the research project TIN2014-57251-P, by a first research scholarship given to the author Sergio Gonzalez by the University of Granada and a second scholarship (FPU) given to the same author by the Spanish Ministry of Education, Culture and Sports.

References

[1] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113 – 141, 2013.

[2] Ronaldo C. Prati, Gustavo E. A. P. A. Batista, and Diego F Silva. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*, 45(1):247–270, 2015.

[3] Vicente García, Ramón Alberto Mollineda, and José Salvador Sánchez. On the k-nn performance in a challenging scenario of imbalance and overlapping. *Pattern Analysis and Applications*, 11(3-4):269–280, 2008.

[4] Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[5] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Andres Folleco. An empirical study of the classification performance of learners on imbalanced and noisy software quality data. *Information Sciences*, 259:571–595, 2014.

[6] Taeho Jo and Nathalie Japkowicz. Class imbalances versus small disjuncts. *SIGKDD Explor. Newsl.*, 6(1):40–49, 2004.

[7] Sarunas J Raudys and Anil K. Jain. Small sample size effects in statistical pattern recognition: recommendations for practitioners. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 3:252–264, 1991.

[8] Gary M Weiss and Foster Provost. Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, pages 315–354, 2003.

[9] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227 – 244, 2000.

[10] Victoria López, Alberto Fernández, and Francisco Herrera. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257:1 – 13, 2014.

[11] E. Ramentol, S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, and F. Herrera. Ifrowann: Imbalanced fuzzy-rough ordered weighted average nearest neighbor classification. *IEEE Transactions on Fuzzy Systems*, 23(5):1622–1637, 2015.

[12] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[13] Susan Lomax and Sunil Vadera. A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys*, 45(2):16:1–16:35, 2013.

[14] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, 2012.

[15] Mikel Galar, Alberto Fernández, Edurne Barrenechea, and Francisco Herrera. EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 46(12):3460–3471, 2013.

[16] Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

[17] Gonzalo Martínez-Muñoz and Alberto Suárez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38(10):1483–1494, 2005.

[18] Ricardo Barandela, Rosa Maria Valdovinos, and José Salvador Sánchez. New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6(3):245–256, 2003.

[19] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer, 2003.

[20] Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *IEEE Symposium on Computational Intelligence and Data Mining. (CIDM'09)*, pages 324–331, 2009.

[21] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(2):539–550, 2009.

[22] S. García, A. Fernández, J. Luengo, and 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(10):2044–2064, 2010.

[23] Miguel De la Torre, Eric Granger, Robert Sabourin, and Dmitry O. Gorodnichy. Adaptive skew-sensitive ensembles for face recognition in video surveillance. *Pattern Recognition*, 48(11):3385 – 3406, 2015.

[24] Hunny Mehrotra, Richa Singh, Mayank Vatsa, and Banshidhar Majhi. Incremental granular relevance vector machine: A case study in multimodal biometrics. *Pattern Recognition*, 56:63 – 76, 2016.

[25] Isaac Triguero, Sara del Río, Victoria López, Jaume Bacardit, José M. Benítez, and Francisco Herrera. ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem. *Knowledge-Based Systems*, 87:69 – 79, 2015.

[26] Alberto Freitas. Building cost-sensitive decision trees for medical applications. *AI Communications*, 24(3):285–287, 2011.

[27] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: Special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[28] Jin Huang and Charles X Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 2005.

[29] Andrew P Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[30] Nitesh V Chawla, David A Cieslak, Lawrence O Hall, and Ajay Joshi. Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery*, 17(2):225–252, 2008.

[31] Shounak Datta and Swagatam Das. Near-bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs. *Neural Networks*, 70:39 – 52, 2015.

[32] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(1):185–197, 2010.

[33] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, pages 321–357, 2002.

[34] Salvador García and Francisco Herrera. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary computation*, 17(3):275–306, 2009.

[35] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[36] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[37] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[38] J. Alcala-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*, 17(2-3):255–287, 2011.

[39] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

3 Chain based sampling for monotonic imbalanced classification

- González, S., García, S., Li, S. T., & Herrera, F. (2019). Chain based sampling for monotonic imbalanced classification. *Information Sciences*, 474, 187-204.
  - Status: **Published**
  - Impact Factor (JCR 2018): **5.524**
  - Subject Category: **Computer Science, Information Systems**
  - Rank: **9/155**
  - Quartile: **Q1**

CHAIN BASED SAMPLING FOR MONOTONIC IMBALANCED CLASSIFICATION

Sergio González
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
sergiogvz@decsai.ugr.es

Salvador García
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
salvagl@decsai.ugr.es

Sheng-Tun Li
Department of Industrial and Information Management
Institute of Information Management
Center for Innovative FinTech Business Models
National Cheng Kung University, Tainan 701, Taiwan
stli@mail.ncku.edu.tw

Francisco Herrera
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
Faculty of Computing and Information Technology
King Abdulaziz University, Jeddah, Saudi Arabia
herrera@decsai.ugr.es

ABSTRACT

Classification with monotonic constraints arises from some ordinal real-life problems. In these real-life problems, it is common to find a big difference in the number of instances representing middle-ranked classes and the top classes, because the former usually represents the average or the normality, while the latter are the exceptional and uncommon. This is known as class imbalance problem, and it deteriorates the learning of those under-represented classes. However, the traditional solutions cannot be applied to applications that require monotonic restrictions to be asserted. Since these were not designed to consider monotonic constraints, they compromise the monotonicity of the data-sets and the performance of the monotonic classifiers. In this paper, we propose a set of new sampling techniques to mitigate the imbalanced class distribution and, at the same time, maintain the monotonicity of the data-sets. These methods perform the sampling inside monotonic chains, sets of comparable instances, in order to preserve them and, as a result, the monotonicity. Five different approaches are redesigned based on famous under- and over-sampling techniques and their standard and ordinal versions are compared with outstanding results.

***K*eywords** Monotonic Classification · Imbalanced classification · Sampling techniques · Preprocessing · Data mining.

1 Introduction

Ranking and evaluation of assets or even individuals are intrinsic characteristics of human nature. Hence, the presence of ordinal variables is common in tons of real-life data-sets. Credit rating [1, 2], house ranking [3] and employee evaluation [4, 5] are good examples of their presence in present-day applications.

These problems aim to determine the most valuable items according to their virtues, i. e. classification into ordinal labels according to ordinal attributes. Additionally, these applications usually require a monotonic restriction between the inputs and the class. That is, the class prediction of an individual should not decrease with a better value for a certain variable, fixing the remainder. Otherwise, an unfair evaluation of the individuals can be made. These classification problems with prior knowledge of the order relations between attributes and the class are known as classification with monotonicity constraints or monotonic classification [6]. Failure to respect these constraints are referred to as violations of monotonicity and must be avoided in the class decision of new samples.

When dealing with monotonic classification problems [6], we look for those examples that belong to the most remarkable class, with a higher value. It is reasonable to have fewer samples of really good and remarkable individuals than those considered normal or average. For example, in the evaluation of future employees of a company, there will probably be fewer ”excellent candidates” than ”average candidates.”

This difference in the number of representatives between classes has proven to cause a great loss of prediction accuracy in the minority classes [7, 8]. This issue is known as a class imbalance problem or imbalanced classification. Multiple real-life applications present this problem, even those from non-standard classifications, such as Monotonic Classification [6, 4, 2] or Multi-task learning [9, 5]. The majority of the monotonic problems considered in the literature suffers due to this issue. Therefore, the imbalance class distribution must be approached in the scope of monotonic classification.

Traditionally data level approaches [8] have been well accepted because they allow the use of a standard classifier after balancing the skewed training sets by under-/over- sampling. However, these techniques also have their own drawbacks. When using under-sampling, there is the risk of losing relevant information from the treated class. On the other hand, over-sampling can introduce noisy instances.

These approaches are not designed for monotonic classification [6] and do not take monotonic constraints into consideration. Due to this lack of awareness of monotonicity [6], these preprocessing techniques can severely deteriorate the monotonicity of the data-sets and reduce the performance of the classifiers. For example, the possible noisy instances generated by over-sampling could mean a greater damage in monotonic classification, because they may increase the number of monotonicity violations in the data-sets. The under-sampling techniques could remove important instances that determine the limit of the classes in term of monotonicity.

Therefore, new sampling approaches must be designed considering the monotonicity constraints. We propose new sampling techniques based on monotonic chains. In monotonic classification, a chain [10] is a set of comparable instances, that is, they can be sorted. These are very important assets of the classification carried out relevant methods such as KNN [11] and OSDL [10, 12], because they determine the possible classes without monotonic violations for new instances. Our techniques perform the sampling using these chains and preserving, as much as possible, the monotonicity of the data-sets. Additionally, these methods take monotonic noise into consideration, in order to avoid instances that violate monotonicity during the sampling process. These differences with the traditional methods reduce the deterioration of monotonicity of sampled data-sets and maintain the improvement of the accuracy for minority classes.

To do so, we have put together a new scheme for applying both under- and over-sampling to monotonic imbalanced data-sets. This scheme consists of several good practices, related to the influence of monotonic violations and chains on sampling, that can be extended to the almost all the sampling techniques in the literature. This scheme has been implemented in five famous under- and over-sampling approaches of the State-of-the-Art of

imbalanced classification: Random Under-Sampling (RUS), Random Over-Sampling (ROS), Synthetic Minority Oversampling TEchnique (SMOTE) [13], ADAptive SYNthetic sampling approach (ADASYN) [14] and Majority Weighted Minority Oversampling TEchnique (MWMOTE) [15].

Throughout this paper, two different empirical studies are carried out with exactly the same experimental framework. The first experiments test the selected sampling techniques in their standard and ordinal versions using 8 monotonic imbalanced sets which are very common in the literature. The majority are multi-class and can be considered highly imbalanced problems. The original and sampled data-sets are classified by five well-known classifiers. Two evaluation metrics are used: Macro Average Arithmetic (MAvA) [16] evaluates the prediction capability in multi-class imbalanced scenarios and Non-Monotonic Index (NMI) [6] determines the monotonicity of data-sets and predictions. The obtained results show empirically the deterioration of the monotonicity degree in data-sets caused by standard and ordinal sampling approaches.

Then, a second experimental study is performed following the same framework to analyze the behavior of new monotonic sampling techniques. The different predictions obtained are compared in terms of multi-class accuracy and monotonicity. The comparison shows the capacity of monotonicity preservation of the monotonic sampling techniques over the standard ones. The outcomes are corroborated by the use of non-parametric statistical tests: Friedman ranking test [17, 18] and Bayesian Sign test [19].

This paper is organized as follows. In Section 2, we present the problems approached and their solutions: classification with monotonic constraints and class imbalance problem. Section 3 is devoted to setting up the bases to adapt sampling approaches to monotonic scenarios and explain in detail the chain based sampling techniques. In Section 4, the experimental framework used in the different empirical studies is presented. Section 5 recalls two experimental studies: an analysis on the impact of standard and ordinal sampling on monotonic classification and a comparison of the results achieved by monotonic sampling. Finally, in Section 6, the main conclusions of this study are given.

2 Background

In this section, we introduce the background knowledge of the problems addressed in this paper.

2.1 Monotonic Classification

Monotonic classification, just as ordinal regression and/or classification, aims to predict an ordinal class label y for new sample x with ordinal attributes with the help of a labeled set, i.e. $f : x \rightarrow y$. In both problems, the classes $\mathcal{Y}$ are categories $\mathcal{Y} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_C\}$ with a problem imposed arrangement $\mathcal{L}_1 \prec \mathcal{L}_2 \prec \ldots \prec \mathcal{L}_C$. However, there is a big difference between both problems. Ordinal classification just focuses on minimizing the errors of predicted and real labels. Monotonic classification imposes monotonicity constraints between the input variables and predicted labels, that is, every instance x' dominated by x should have a lower or the same assigned class $f(x')$ than x class label $f(x)$. Formally, $x \succeq x' \rightarrow f(x) \geq f(x')$ [20], where $x \succeq x' \leftrightarrow x_j \geq x'_j, j = 1, \ldots, A$.

Recently this problem has drawn the attention of data mining practitioners, who have designed monotonic classifiers based on distinct models, such as, instance-based learning [21, 11, 10, 22], rules-based methods and decision trees [6, 23, 24, 25], support vector machines [1, 26], neural networks [27, 28, 29] and ensemble learning [30, 31, 32]. These monotonic classifiers avoid monotonicity violations in their predictions. And they can be pure, when their decisions are always monotonic, or partial, if they minimize their violations as much as possible. Some of these monotonic classifiers have to learn from monotonic data-sets in order to properly predict new samples. Only if all the pairs of example i, j of a training set D are monotonic, the data-set D is considered monotonic [21], i.e. $x_i \succeq x_j \rightarrow y_i \geq y_j, \forall x_i, x_j \in D$.

Even though these constraints are defined in the learning and prediction phases of the classification process, they have to be taken into consideration in preprocessing phases. Otherwise the monotonicity of the involved training set can be severely compromised. Few studies have been undertaken in this field [33, 4, 34]. For example, imbalanced classification in scenarios with monotonic constraints has not been explored, although it has been for ordinal classification [35, 36].

2.2 The class imbalance problem

The class imbalance problem refers to a severe loss of classification accuracy of certain classes due to their under-representation in the training data-set [7]. That is, some classes have a lot less instances than others, which affects their identification by standard classifiers. These under-represented classes are known as minority or positives, whilst the rest are referred to as majority or negative classes. In many real-life applications, the most important classes are usually the most imbalanced. Therefore, the misclassification of these classes entails greater costs [37].

The inaccurate prediction of the minority classes mainly results from the generalization of the standard classifiers used to infer models and avoid over-fitting. The use of global performance measures, such as accuracy rate, also negatively affects the learning phase. Therefore, the difference in the number of representatives of each class has a great influence on this issue. However, there are other issues that aggravate this problem [7, 38], such as noise, overlapping, lack of density and small disjuncts.

Many data scientists have shown great interest in the field and several approaches have been designed to deal with it. These can be categorized into the following three [7]:

- Data level approaches [8, 13]: the affected data-sets are rebalanced by sampling. The equal representation between classes is reached by generating more examples for the minority class (over-sampling) [13], removing examples from the majority class (under-sampling) or both (hybrid methods).
- The cost-sensitive learning [39, 40] considers the costs of the errors of the misclassification of the minority class into function minimization.
- The algorithmic approaches [41, 42] are modifications of base learning algorithms in order to achieve better performance with imbalanced data-sets.

Data level approaches are the most popular solutions to the problem, since they enable the standard classification methods. Random Over- and Under-Sampling are the most basic procedures that randomly duplicate or remove examples from the minority or majority classes. Other more complex methods have been proposed, such as SMOTE (Synthetic Minority Oversampling TEchnique) [13], ADASYN (ADAptive SYNthetic sampling approach) [14] or MWMOTE (Majority Weighted Minority Oversampling TEchnique) [15]. These five are remarkable approaches among the State-of-the-Art methods.

SMOTE [13] generates synthetic instances x_g through the linear interpolation of randomly selected instances x and their nearest neighbors x_n from the same minority class (see Eq. 1). Similarly, ADASYN [14] also generates new instances through this interpolation, however, the selection of the examples are not uniformly random. It prioritizes instances near to the borders of the class according to a distribution of weights. These weights are computed per instance as the ratio of neighbors with a different class label of the class of the evaluated instance.

Linear interpolation is carried out using the following formula:

$$x_g = x + (x_n - x) * \alpha \tag{1}$$

where α is a random number chosen in the range [0,1].

MWMOTE [15] is based on clustering to define the group of instances to be interpolated. First, MWMOTE identifies the minority instances at the borders, i.e. hard to learn instances, and removes the possible noisy instances. Then, sample weights are assigned in relation to their closeness to a border and the low density of their clusters. Finally, these instances are randomly selected according to their weights and interpolated with another sample of the same cluster.

Additionally, standard global metrics negatively influence the performance and they have to be replaced by more suitable options: Area Under the ROC Curve [43] or Geometric Mean are commonly used in binary imbalance problems [7, 8]. Macro Average Arithmetic (MAvA) [16] has been frequently chosen for multi-class scenarios and is computed as followed:

$$MAvA = \frac{1}{m}\sum_{i=1}^{m} ACC_i \tag{2}$$

where ACC_i is the partial accuracy rate for the class i independently.

3 Chain based Sampling in Monotonic Imbalanced Classification

The intention of this work is to design proper ways of sampling monotonic data-sets without loosing their monotonicity. To do so, we propose a new sampling schema based on chains. The main difference is the maintenance of the essential parts of the chains of each class. That is, the upper- and lower- instances of each class in each chain [10].

A monotonic chain is a set S of comparable instances, i.e. those which can be sorted. The upper-instance $\uparrow x_S$ of the chain S is the one that dominates all the other samples of the chain. The lower-instance $\downarrow x_S$ of the chain is dominated by the rest. The upper- and lower- instances of each class for a whole chain determine the upper and lower bounds of the range of possible labels without monotonic violations for new instances which are in that chain [10, 11]. Formally, the upper- and lower- instances are defined as the following:

$$\uparrow x_S = x \in S : \forall x_i \in S \; x \succeq x_i \tag{3}$$

$$\downarrow x_S = x \in S : \forall x_i \in S \; x_i \succeq x \tag{4}$$

Since sampling techniques usually work independently for each class, chains can be segmented in subsets of a certain class. Then, the instances at the limits of these subsets are the ones that must remain untouched.

In contrast to traditional methods, monotonic sampling also takes monotonic noisy instances into consideration, avoiding them so as not to greatly affect the decisions during preprocessing. Even more, our techniques mitigate their impacts when possible.

To achieve these objectives, the monotonic sampling techniques classify the instances of each class in three different groups of relevance. This classification is used to bias the selection of the examples involved in the sampling, promoting instances at the limits and inside chains and avoiding monotonic noise. The three groups of representatives are detailed in the following:

- At the limits of chains: As said before, these are the most important instances of the chains and the classes, because they determine the ordering limit of each class. Therefore, the sampling techniques prioritize them over the rest. This type of example is identified when there is no comparable example of the same class that is greater or smaller, i.e. upper- or lower- limit, respectively. As explained above, chains are treated independently for each class.

Algorithm 1 Example in pseudo-code of the identification of the type of an example x.

function SAMPLETYPE(x - evaluated instance, y - instance x class label, D - data-set)
  if $\exists(x_i, y_i) \in$D $\quad | \quad x_i \succeq x \wedge y > y_i \quad \vee \quad x \succeq x_i \wedge y_i > y$ **then**
    Set x as violation of monotonicity
  else ▷ Get set of instances with same class as y
    $S_y \leftarrow \{\forall(x_i, y_i) \in D \quad | \quad y_i = y\}$
    if $\nexists x_i \in S_y \quad | \quad x_i \succeq x \quad \vee \quad x \succeq x_i$ **then**
      Set x as a instance at the limit of chain
    else
      Set x as a instance inside a chain
    end if
  end if
end function

- Inside chains: These representatives give consistency to the chains. The more of this type there is in a chain, the stronger and more relevant the chain is. When possible, these examples must be conserved during sampling. These are the counterparts of the previous type, so they are identified by finding greater and smaller comparable samples.

- Violation of monotonicity: These are monotonic discrepancies, that is, instances that are not monotonically consistent with others. They must be avoided, because they are one of the main reasons for monotonic deterioration of the models. They are identified by comparison with comparable instances of other classes. Sampling techniques avoid them when possible. Tolerance to these examples is computed as indirectly proportional to the number of violations of the instances. When a instance has a high number of violations, there will be only a small possibility of being involved in the sampling.

Algorithm 1 exemplifies with pseudo-code and formal expressions how the types of an instance x are determined.

With these ideas, we settle the bases to extend the use of chains in nearly every sampling technique, both under-sampling and over-sampling approaches.

3.1 Under-Sampling for Monotonic Imbalanced Classification

For monotonic versions of under-sampling methods, it is highly important to preserve the upper- and lower-limits of the chains. As many as possible of these instances will be prioritized and maintained in the final sampled set until equal balance is reached.

Additionally, under-sampling methods could help to mitigate the impact of monotonic violations of the data-set by purposely sampling the monotonic noisy instances. This feature is unique in under-sampling methods, since over-sampling lacks any similar characteristics.

Depending on the original under-sampling technique, this knowledge can be introduced in different ways. For weight-based sampling, a monotonic weight can be given to each instance according the three previously mentioned types of instances. The weights given to instances at the limits of chains are recommended to be higher than the ones given to the samples inside chains. Non monotonic instances receive a weight of 0 ensuring their exclusion, but only if there are enough samples of the other two types to achieve class balance. Otherwise, these weights may be indirectly proportional to the number of monotonic violations caused by the instances. These monotonic weights can be included as a factor of the original weighting. See the following subsection for more details.

However, we have extended the famous Random Under-Sampling. The original method randomly selects a subset of the majority classes to rebalance the data-set. RUS does not use any weights or probabilities in its process of selection. It merely chooses the samples in a uniformly random way. Therefore, the monotonic RUS approach (mRUS) is designed as a hierarchical selection according to the mentioned groups of relevance instead of being purely random. This hierarchical selection has the following steps:

- *Step 1:* If the number of at-the-limits samples is big enough to fulfill the rebalance of the data-sets, a subset of this group is uniformly and randomly selected. Otherwise, every instance is selected and Step 2 will be executed.
- *Step 2:* If the number of instances inside the chains is enough to rebalance the data-sets within the already selected instances, the same selection procedure is applied to this type of instances. Otherwise, all are selected and Step 3 will continue with the rebalance.
- *Step 3:* Only if needed, monotonic noisy instances will be used to complete the balanced data-set. The selection is not uniform; it is guided by a probability distribution. The probability to select a certain non monotonic sample is computed as the inverse of the number of monotonic violations. So, instances with more discrepancies have less chance of being selected, while instances with just one inconsistency have the highest probability. This final selection has been implemented following a roulette wheel scheme, as seen in Line 26. This procedure computes a cumulative probability distribution and generates a random number between 0 and 1. The first cumulative probability greater than this number indicates the selected instance.

These steps are exemplified in the following algorithm 2.

3.2 Over-Sampling for Monotonic Imbalanced Classification

Over-Sampling techniques are more likely to be weight-based methods. Therefore, their monotonic extensions could easily include a monotonic weighting according to the three categories of the previously defined instances. These weights can then be transformed in a probability distribution to guide the selection of the over-sampled instances. The hierarchical approach used for under-sampling lacks common sense for over-sampling techniques, since they cannot easily implement it.

As previously mentioned, monotonic noise or non monotonic instances should be excluded from the sampling decisions. However, they cannot be removed from the final data-set, as in under-sampling. In order to reduce their impact as much as possible, the probability of them being selected is equal to zero. Then, they will not be replicated. But if the sampled class is entirely composed of instances with monotonic violations, they cannot be avoided. Then, they will be selected according to the inverse of their number of violations within the range (0,1].

Since instances at the limit of chains seize more valuable information, they should have a significantly higher chance than those inside chains. At least, the probability of the former instances should double that of the latter. Finally, these weights are normalized to form a probability distribution. Algorithm 3 shows the implementation of this weight scheme in pseudo-code.

According to this weighting approach, we have chosen four well-known over-sampling techniques to design a monotonic version of them. We have detailed the highlights of each of these methods in the following:

- **ROS:** The original Random Over-Sampling technique randomly replicates instances of the minority classes, which could greatly deteriorate the monotonicity of the data-sets if non monotonic instances are frequently selected. The monotonic version of ROS (mROS) does not perform a uniformly random selection anymore. It selects the instances following a roulette wheel selection scheme as in mRUS,

Algorithm 2 Monotonic Random Under-Sampling

1: **function** MRUS(N - number of instances to select, S_y - sampled set of class y, D - data-set)
2: **initialize:** $S_{violations} \leftarrow \emptyset$, $S_{atLimits} \leftarrow \emptyset$, $S_{inChain} \leftarrow \emptyset$, $Prob \leftarrow \emptyset$, $S_{res} \leftarrow \emptyset$
3: **for** $(x_i, y_i) \in S_y$ **do**
4: **if** $sampleType(x_i, y_i, D) = monotonicViolation$ **then**
5: $Prob_i = 1.0/numViolations(x_i, y_i, D)$
6: $S_{violation} \leftarrow (x_i, y_i)$
7: **else**
8: **if** $sampleType(x_i, y_i, D) = atLimits$ **then**
9: $S_{atLimits} \leftarrow (x_i, y_i)$
10: **else**
11: $S_{inChain} \leftarrow (x_i, y_i)$
12: **end if**
13: **end if**
14: **end for**
15: **if** $Size(S_{atLimits}) > N$ **then** ▷ Step 1
16: $S_{res} \leftarrow uniformlyRandomSelection(S_{atLimits})$
17: **else** ▷ Step 2
18: $S_{res} \leftarrow S_{atLimits}$
19: **if** $Size(S_{inChain}) > (N - Size(S_{atLimits}))$ **then**
20: $S_{res} \leftarrow uniformlyRandomSelection(S_{inChain})$
21: **else** ▷ Step 3
22: $S_{res} \leftarrow S_{inChain}$
23: $Normalize(Prob)$
24: $\#Samples = N - Size(S_{atLimits}) - Size(S_{atChains})$
25: **for** $j \in [1, \#Samples]$ **do**
26: $S_{res} \leftarrow rouletteSelection(S_{violations}, Prob)$
27: **end for**
28: **end if**
29: **end if**
30: **return** S_{res}
31: **end function**

but with the previously explained probability distribution computed by Algorithm 3. An explanation in pseudo-code of mROS can be found in Algorithm 4.

mROS is not so effective for the methods that transform replicas into class membership, such as OSDL [10] or some relabeling techniques [44, 45]. These methods intend to mitigate the impact of replicas with different classes by fusing them into probabilities. For data-sets with these inconsistencies, mROS is very effective. On the other hand, mROS achieves nothing. Therefore, we have designed a different method of replication (Line 7). The replications will have a small variation in their values, to avoid attaining exactly the same instances. This is done using jittering, that is, by introducing a small Gaussian noise to the feature values of the selected instance.

In order to ensure that the derived sample is as close as possible to the selected one, the standard deviation σ of the Gaussian noise generator must be sufficiently small, so that $\sigma = 0.001$. Therefore, this small variation of each feature is randomly generated following a Gaussian distribution with a mean μ equal to original feature value and a σ of 0.001 (Line 8). These two versions of mROS are

Algorithm 3 Monotonic weights function

function MONOTONICWEIGHTS(S_y - data-set of class,D - data-set)
  initialize: $Prob[1..size(S_y)] = 0$, $S_{violations} \leftarrow \emptyset$
  for $(x_i, y_i) \in S_y$ **do**
    if $sampleType(x_i, y_i, D) = monotonicViolation$ **then**
      $Prob_i = 1.0/numViolations(x_i, y_i, D)$
      $S_{violations} \leftarrow i$
    else
      if $sampleType(x_i, y_i, D) = atLimits$ **then**
        $Prob_i = 4.0$
      else
        $Prob_i = 2.0$
      end if
    end if
  end for
  if $Size(S_{violations}) \quad != \quad Size(S_y)$ **then**
    for $i \in S_{violations}$ **do**
      $Prob_i = 0$
    end for
  end if
  $Normalize(Prob)$
  return $Prob$
end function

Algorithm 4 Monotonic Random Over-Sampling

function MROS(rep - option of replication (std or 1-NN), N - number of instances to replicate, S_y - sampled set of class y, D - data-set)
  initialize: $Prob \leftarrow monotonicWeights(S_y, D)$, $S_{res} \leftarrow S_y$
  for $i \in [1, N]$ **do**
    $(x_i, y_i) = rouletteSelection(S_y, Prob)$
    if $rep = true$ **then**
      $S_{res} \leftarrow (x_i, y_i)$
    else
      $x_g = gaussianRandom(\mu = 0, \sigma = 0.001) + x_i$
      $S_{res} \leftarrow (x_g, y_i)$
    end if
  end for
  return S_{res}
end function

implemented as a parameter of the same method, so the best result is always chosen according to the features of the classification method.

- **SMOTE [13]:** This method generates synthetic instances through interpolation of randomly selected instances and their nearest neighbors. Its adaption must avoid the selection of monotonic noisy instances, because their interpolations will probably be noisy. Algorithm 5 represents the method Monotonic SMOTE (mSMOTE).

Algorithm 5 Monotonic SMOTE

1: **function** MSMOTE(k - nearest neighbors, $interpolationRatio = 0.5$, N - number of instances to replicate, S_y - sampled set of class y, D - data-set)
2: **initialize:** $Prob \leftarrow monotonicWeights(S_y, D)$, $S_{res} \leftarrow S_y$
3: **for** $i \in [1, N]$ **do**
4: $(x_i, y_i) = rouletteSelection(S_y,\ Prob)$
5: $S_{nn} \leftarrow kNN(k, S_y).getNeighbors(x_i, y_i)$
6: $Prob_{nn} \leftarrow monotonicWeights(S_{nn}, D)$
7: $(x_{nn}, y_{nn}) = rouletteSelection(S_{nn},\ Prob_n n)$
8: $\alpha = random(0, 1)$
9: $x_g = x_i + interpolationRatio * (x_{nn} - x_i) * \alpha$
10: $S_{res} \leftarrow (x_g, y_i)$
11: **end for**
12: **return** S_{res}
13: **end function**

mSMOTE follows exactly the same selection procedure as mROS. When a sample is selected (Line 4), its k nearest neighbors are computed and just one is chosen to perform the interpolation. As shown in Line 6, the nearest neighbors are selected with the same probabilities scheme mentioned before. That is, neighbors at the limits of chains are prioritized over those that are inside a chain.

For mSMOTE, the interpolations are limited and ensured to be computed with a maximum distance of a percentage of the total distance to the selected nearest neighbor. This percentage can be considered to be a parameter of the method, *interpolationRatio*. This parameter can be set from 0, meaning no interpolation is performed and thus it works as a mROS with repetitions, to the value of 1, as the standard linear interpolation. Values in this range modulate "room for error" during the interpolation process. If we want to be conservative and try to preserve monotonicity as much as possible, we will try to generate synthetic instances near the one selected, i.e. setting a low value for the parameter. We recommend setting it to 0.5, as a good trade off between being conservative and covering more of the problem space. The expression in Line 9 reflects the modification of the standard expression for linear interpolation.

- **ADASYN [14]:** The original ADASYN already includes a weighting schema to prioritize samples at the borders of the minority class. The weight of a instance is obtained as the ratio of nearest neighbors with a different class value. The instances are selected according to these weights, then, the linear interpolation with their neighbors are performed as SMOTE.

  Monotonic ADASYN (mADASYN) is similar to mSMOTE. However, since it already computes weights for each instance, a combination of both schemes has to be defined. This is done by the product of both normalized weights, then they are normalized again. The following expression shows this transformation:

$$Prob_i = \frac{\Delta_i}{k} * monotonicWeights(x_i, D) \tag{5}$$

  where Δ_i is the number of neighbors with a different class than x_i in the set of k nearest neighbors and $monotonicWeight(x_i, D)$ is the monotonic weight of x_i defined in Algorithm 3 and normalized in the range (0,1]

- **MWMOTE [15]:** The original MWMOTE is a very different approach compared to SMOTE and ADASYN. It uses clustering as well as several nearest neighbor rules to define the instances S_{imin} from

the minority class that are really hard to learn. To do so, the set S_{minf} is obtained with the minority instances which are not considered to be noise, that is, instances that have at least one minority instance among their k_1 nearest neighbors. Then, S_{bmaj} is computed as the union of all k_2 nearest enemies, instances from other classes, of the instances in S_{minf}. Finally, S_{imin} is the result of all k_3 nearest neighbors belonging to the minority class of S_{bmaj}. Instances in S_{imin} are the only samples that can be selected according to weight distribution S_w.

The weight S_w for given sample x_i is computed with the sum of the information weights I_w contributed by all the instances in S_{bmaj}:

$$S_w(x_i) = \sum_{x_j \in S_{bmaj}} I_w(x_j, x_i) \tag{6}$$

The information weight is composed of a closeness factor $C_f(x_j, x_i)$ and a density factor $D_f(x_j, x_i)$. The former determines the closeness to the decision boundaries, while the latter quantifies the density of the cluster to which the instance x_i belongs.

$$I_w(x_j, x_i) = C_f(x_j, x_i) * D_f(x_j, x_i) \tag{7}$$

Monotonic MWMOTE (mMWMOTE) maintains this algorithmic structure untouched. Similarly as mADASYN, the monotonic weight distribution is merged with the original distribution S_w. This is done by multiplying both factors:

$$Prob_i = S_w(x_i) * monotonicWeights(x_i, D) \tag{8}$$

As a reminder, the selection of the neighbor or corresponding cluster instance to be interpolated with the previously selected sample x_i is also guided using the monotonic weighting in the monotonic methods SMOTE, ADASYN and MWMOTE.

4 Experimental framework

This section introduces the experimental framework followed in all the experiments conducted in this paper. In order to study the viability of sampling in monotonic environments, we select five well-known sampling techniques from the-State-of-the-Art: RUS, ROS, SMOTE [13], ADASYN [14] and MWMOTE [15]. These were explained in previous sections.

They were chosen following the study [36] that develops a version of these methods for ordinal regression. Then, we can also test them within monotonic scenarios. Ordinal versions of ROS and SMOTE do not change much: they are just applied to all the classes to balance them. For this reason, we have only included one version of RUS, ROS and SMOTE. However, ADASYN and MWMOTE ordinal extensions include a factor to consider the ordinal rank difference between the evaluated instances and their neighbors or closest samples. For ADASYN, this factor is introduced in the weights calculation, while for MWMOTE, it appears in the closeness factor.

Additionally, in the experiments we have included the new over-sampling technique developed in [36]. CWOSOrd is a cluster-based oversampling technique that is focused on the more complex and smaller clusters, where it generates more synthetic samples. The ordering relationship and the distances to other classes samples are used to compute a probability distribution to guide the random selection of samples for synthetic generation. For more specific details, we refer the readers to original paper [36]. Table 1 recalls the sampling techniques used and their parameters, including our proposals.

Table 1: Summary of the sampling techniques and their suggested parameters.

Sampling techniques	**Parameters**
RUS & ROS	no parameters
SMOTE & ADASYN & ADASYNOrd	$k = 5$, $distance = Euclidean$
MWMOTE & MWMOTEOrd	$k_1 = 5, k_2 = 3, k_3 = 3, C_p = 3,$ $C_f = 5, C_{max} = 2, distance = Euclidean$
CWOSOrd	$NN = 5, NS = 5, \alpha = 1,$ $C_{thres} = 2, distance = Euclidean$
mRUS & mROS	no parameters
mSMOTE & mADASYN & mMWMOTE	Same recommended parameters, $interpolationRatio = 0.5$

Table 2: Description of the data-sets used.

Data-set	**Ins.**	**At.**	**Cl.**	**At. Directions**	**%Ins. per Class**
balance	625	4	3	{-, -, +, +}	46.1/**7.8**/46.1
car	1728	6	4	All direct	70.1/22.2/**4.0/3.8**
ERA	1000	4	9	All direct	**9.1/14.2**/18.1/17.2/15.8/**11.8/8.8/3.1/1.9**
ESL	488	4	9	All direct	**0.2/2.5/7.7**/20.5/23.9/27.6/**12.8/3.9/0.9**
LEV	1000	4	5	All direct	**9.2**/28.0/40.3/19.7/**2.8**
SWD	1000	10	4	All direct	**3.1**/35.2/39.9/21.8
windsorhousing	546	11	2	All direct	76.6/**23.4**
wisconsin	699	9	2	All direct	65.3/**34.7**

These methods are applied to 8 different monotonic data-sets that have been selected as they are commonly used in the literature of classification with monotonic constraints. . Table 2 shows the characteristics of each data-set: number of instances (Ins.), attributes (At.) and classes (Cl.), monotone direction between each attribute and the class, and the percentage of representation of each class label in the data-set. As can be seen in the table, the majority of these data-sets are multi-class and highly imbalanced problems. However, the classification cannot be carried out by class decomposition schemes [46, 47], such as One-vs-One [48, 49] or One-vs-All [50]. Since these algorithms perform the classification in decomposed binary independent problems, the order and monotonic relations between classes are distorted.

Therefore, the sampling techniques are applied to the classes depending on whether they are majority or minority classes for under- or over-sampling, respectively. The sampling is stopped when equal balance is reached. The minority classes are represented with bold-face font in the column %Ins. per Class at Table 2, the rest are considered majority classes.

Then, the resulting data-sets are classified by 5 different multi-class monotonic classifiers. Table 3 recalls the monotonic classifiers used during these experiments and their parameters. All this process is carried out following a 10-fold stratified cross validation scheme (10-fcv) with the partitions found in the software KEEL [51].

Table 3: Parameters considered for the algorithms compared.

Algorithm	Parameters
MkNN [11]	k = 5, distance = euclidean, neighborsType = inRange
OSDL [10]	balanced = No, classificationType = median, lowerBound = 0, upperBound = 1 tuneInterpolationParameter = No, weighted = No, interpolationStepSize = 10, interpolationParameter = 0.5
OLM [21]	modeResolution = conservative modeClassification = conservative
C4.5-MID [6]	R = 1, confidence = 0.25, items per leaf = 2
MonMLP [27]	default parameters, hidden1 = 8 iter.max = 1000, monotone = all att

As performance measures, we have selected MAvA and Non-Monotonic Index (NMI). The above mentioned MAvA evaluates the prediction capability of the classifiers for multi-class imbalance problems. MAvA results should improve after applying sampling. Non-Monotonic Index is used to determine the impact of the sampling techniques on the monotonicity of the monotonic classifier predictions. Non-Monotonic Index (NMI) [6] measures the ratio of instance pairs NMP that violate monotonicity, with respect to the total number of pairs of instances in a predicted set. To compute it, the sets of test predictions resulting from the 10-fcv classification are merged into only one set. Then, the NMI is computed over this set following the expression:

$$NMI = \frac{NMP}{N^2 - N} \tag{9}$$

where N stands for the total number of instances.

In order to corroborate the different outcomes of the experiments, we use non-parametric statistical tests: the well-known Friedman rank test [17, 18] and Bayesian Sign test [19].

Bayesian Sign test is a pairwise Bayesian non-parametric sign test based on the Dirichlet Process [19]. This test computes a distribution with the difference of the results obtained by the two compared algorithms (A vs B). Then, a decision is made according to the position of the majority of the distributions in one of the three regions: left (superiority of method B), rope (statistical equivalence) and right (superiority of method A)[19].

We have used the R package, named rNPBST[52], to perform the different tests and to present the graphics in the following section.

5 Experimental studies

This section is devoted to experimentally showing the influence on monotonicity of standard, ordinal and monotonic chain based sampling techniques. First, the performance results of several standard and ordinal samplings are analyzed, paying special attention to their impact on the monotonicity of the prediction of different monotonic classifiers. The following subsection presents the results obtained by our new monotonic sampling scheme. An extensive comparison is made with the original results of the classifiers and the standard sampling.

5.1 On the viability of standard sampling techniques in monotonic imbalanced scenarios

In order to test the feasibility of sampling for monotonic imbalanced data-sets, we must first assure that the prediction capability has increased after applying them. Table 4 recalls the average MAvA results achieved by each classifier with and without the resulting set of each preprocessing method. The table cells colored in

Table 4: MAvA average results obtained by the selected classifiers with standard and ordinal sampling.

	Original	**RUS**	**ROS**	**SMOTE**	**ADASYN**	**ADASYNOrd**	**MWMOTE**	**MWMOTEOrd**	**CWOSOrd**
MkNN	0.5727	0.5827	0.5742	0.6106	*0.6143*	0.5828	*0.6128*	0.6105	0.6137
OSDL	0.5909	0.4587	0.4502	0.4351	0.4344	*0.4373*	0.4459	*0.4472*	0.4429
OLM	0.5468	0.5273	0.5656	0.5762	*0.5771*	*0.5771*	0.5810	*0.5815*	0.5746
MID	0.5434	0.5693	0.6272	0.6064	*0.6104*	0.5955	0.5929	*0.5987*	0.5912
MonMLP	0.5582	0.6034	0.6300	0.5902	0.5865	*0.5924*	0.5689	*0.5771*	0.5709

Table 5: Friedman ranking for MAvA results obtained by the selected classifiers with standard and ordinal sampling.

	MAvA				
Rank	**MkNN**	**OLM**	**MID**	**OSDL**	**MonMLP**
1	ADASYN (3.50)	MWMOTEOrd (3.31)	ROS (2.50)	Original (1.50)	SMOTE (3.75)
2	MWMOTE (3.75)	MWMOTE (3.93)	ADASYN (3.56)	RUS (3.62)	ADASYN (3.93)
3	CWOSOrd (3.75)	ADASYNOrd (4.06)	SMOTE (3.62)	SMOTE (4.93)	ADASYNOrd (4.18)
4	SMOTE (4.62)	SMOTE (4.37)	MWMOTEOrd (4.06)	ADASYNOrd (5.00)	ROS (4.37)
5	MWMOTEOrd (4.75)	ADASYN (4.50)	MWMOTE (5.56)	ROS (5.12)	MWMOTEOrd (4.68)
6	RUS (5.25)	CWOSOrd (4.56)	ADASYNOrd (5.81)	ADASYN (5.31)	RUS (5.25)
7	ADASYNOrd (5.37)	ROS (5.62)	Original (6.00)	MWMOTEOrd (5.5)	MWMOTE (5.31)
8	ROS (7.00)	Original (7.12)	CWOSOrd (6.25)	MWMOTE (6.81)	CWOSOrd (6.50)
9	Original (7.00)	RUS (7.50)	RUS (7.62)	CWOSOrd (7.18)	Original (7.00)

gray indicate an improvement compared to the original results obtained by the classifier. Italics font is used to highlight the best performance for each sampling group; in this case, the comparison of the standard and ordinal version of ADASYN and MWMOTE.

As shown in Table 4, nearly all results achieved with balanced sets outperform the classification with skewed data-sets. Even though it was expected, these improvements are remarkably significant, especially the one obtained with the combination of ADASYN+MkNN, ROS+MID and ROS+MonMLP. However, OSDL always achieves better results with the original data-sets. With this outcome, we can deduce that sampling techniques will not work properly for OSDL. Probably, this is related to its particular way of classification based on stochastic dominance and probability distribution functions extracted from repeated instances with different class labels[10]. Therefore, we exclude OSDL from the rest of the experiments.

In addition, Table 5 shows a Friedman ranking per selected classifier of the sampling techniques sorted by their performance in terms of MAvA. Here, the MAvA advantage of sampling can be also appreciated, since the original results are usually ranked last, with the exception of OSDL.

There is not one sampling method which is superior to the rest. For some classifiers, certain sampling methods are well ranked, while for others, they are not. Their good or bad performance strongly depends on the base learner. For ordinal sampling, there is no significant improvement over their standard versions. ADASYN is ranked higher more times than ADASYNOrd, while, for MWMOTE, the ordinal version is usually ranked better. CWOSOrd is not well ranked for any of the selected classifiers.

A better prediction ability thanks to a preprocessing technique is not enough to prove its viability for classification with monotonic constraints. Its impact on monotonicity must be studied. Table 6 gathers the average NMI retrieved from the selected classifiers with the different sampling methods. As in the previous Table 4, the results in gray improve those recalled purely by the classifier. With exception of SMOTE and ADASYN for OLM, all NMI measured with sampling are much higher. Therefore, these sampling techniques really deteriorate the monotonicity of the data-sets and, as consequence, the monotonicity of the monotonic learner.

Table 6: NMI average results obtained by the selected classifiers with standard and ordinal sampling.

	Original	RUS	ROS	SMOTE	ADASYN	ADASYNOrd	MWMOTE	MWMOTEOrd	CWOSOrd
MkNN	0.0011	0.0034	0.0014	0.0020	*0.0024*	0.0053	*0.0020*	0.0022	0.0026
OSDL	0.0009	0.0020	0.0164	0.0052	0.0051	*0.0048*	*0.0044*	0.0057	0.0069
OLM	0.0018	0.0028	0.0021	0.0017	0.0016	*0.0015*	*0.0019*	*0.0019*	0.0020
MID	0.0030	0.0090	0.0050	0.0038	*0.0045*	0.0051	*0.0042*	0.0044	0.0037
MonMLP	0.0005	0.0021	0.0013	0.0013	*0.0015*	0.0016	*0.0036*	0.0037	0.0076

Table 7: Friedman ranking for NMI results obtained by the selected classifiers with standard and ordinal sampling.

	NMI				
Rank	**M*k*NN**	**OLM**	**MID**	**OSDL**	**MonMLP**
1	Original (2.43)	ADASYNOrd (3.31)	Original (3.37)	Original (3.18)	Original (2.50)
2	ROS (3.81)	ADASYN (3.68)	MWMOTEOrd (4.00)	RUS (4.43)	ADASYN (4.37)
3	ADASYN (4.68)	ROS (4.68)	SMOTE (4.12)	SMOTE (4.81)	ADASYNOrd (4.62)
4	SMOTE (4.87)	SMOTE (4.75)	ADASYN (4.56)	MWMOTE (4.81)	SMOTE (4.87)
5	MWMOTE (5.12)	Original (5.12)	CWOSOrd (4.87)	ADASYN (5.31)	MWMOTE (4.87)
6	CWOSOrd (5.25)	RUS (5.5)	MWMOTE (5.12)	ADASYNOrd (5.37)	ROS (5.00)
7	MWMOTEOrd (5.50)	MWMOTEOrd (5.62)	ADASYNOrd (5.68)	ROS (5.62)	MWMOTEOrd (5.37)
8	ADASYNOrd (5.93)	CWOSOrd (6.06)	ROS (5.87)	MWMOTEOrd (5.62)	CWOSOrd (6.00)
9	RUS (7.38)	MWMOTE (6.25)	RUS (7.37)	CWOSOrd (5.81)	RUS (7.37)

This fact can be also observed in the Friedman ranking presented in the Table 7 that composed of all the NMI results. The original classifiers are nearly always ranked first with a small ranking value compared to the second best.

Additionally, thanks this study of monotonicity, ordinal sampling methods become impractical, since ADASYN is still usually better than ADASYNOrd, MWMOTE is now ranked higher than its ordinal version in most cases and once again, CWOSOrd ranks last in the Friedman ranking.

With this first experimental study, we can conclude that standard sampling can improve the prediction of the classifiers, but they impair the monotonicity to a large extent. And the existing ordinal sampling methods do not solve the problem.

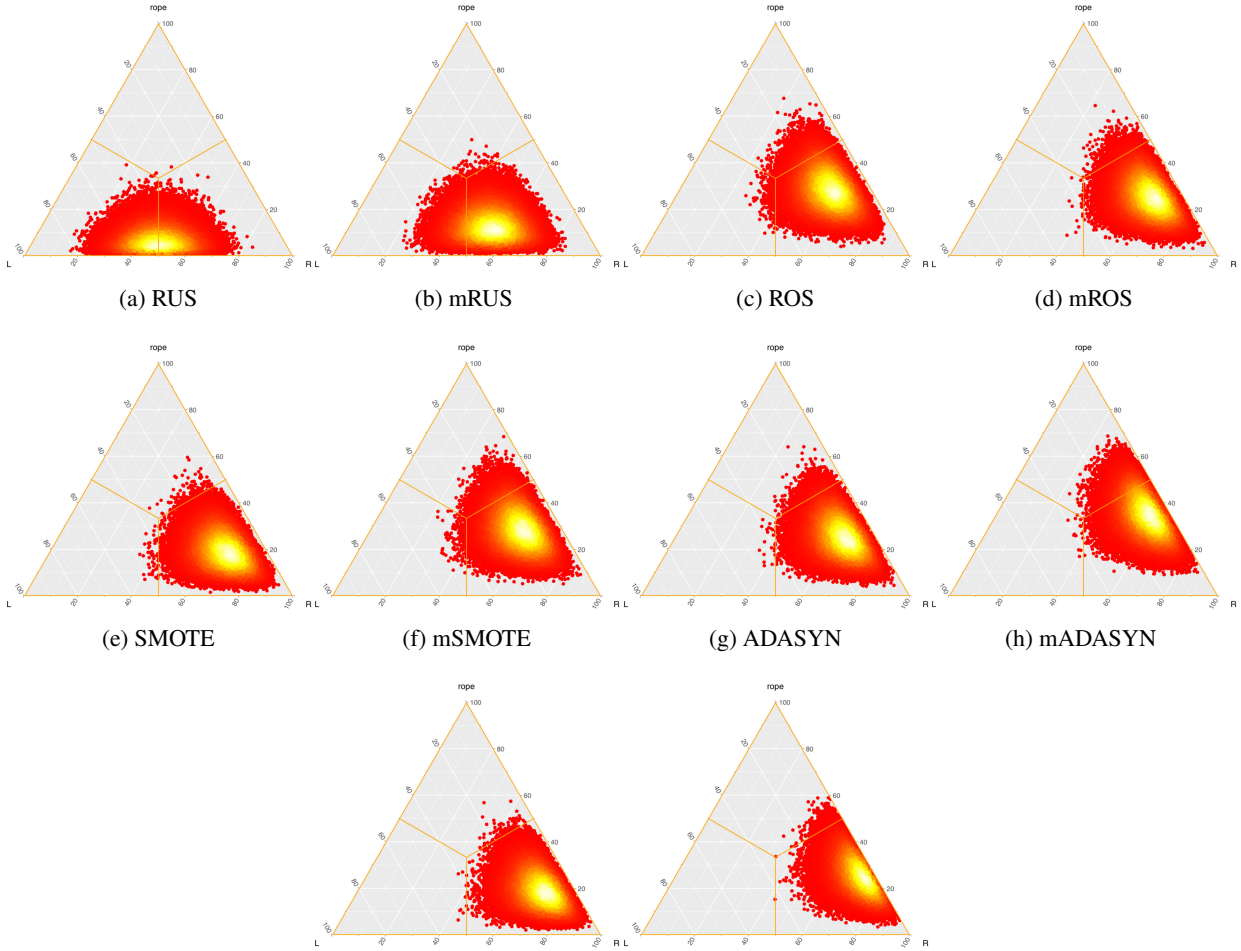
The complete results achieved by these methods in terms of MAvA and NMI can be found in the following experimental study, for the standard sampling methods, and in A. They have been removed from this section for the sake of clarity.

5.2 Standard and Monotonic Sampling: Results and Analysis

Throughout this empirical study, we analyze the behavior of our chain based sampling techniques. Their performance in terms of MAvA and NMI are compared with the classifiers with and without the standard sampling methods. Given the bad performance shown in the previous experimental study using OSDL with sampled data-sets in comparison to the one with original sets, OSDL has been excluded from this experiment.

Table 8 recalls the MAvA achieved per data-set with the different configurations of base learners and standard or monotonic preprocessing. The best results between the standard and monotonic versions of each sampling are highlighted in italics. The gray cells represent an improvement when compared to the classifiers without sampling.

All the classifiers improve their performance in terms of MAvA by using sampling, both standard and monotonic, for most of the data-sets. The best sampling methods on average are the standard ROS sampling and



(a) RUS (b) mRUS (c) ROS (d) mROS

(e) SMOTE (f) mSMOTE (g) ADASYN (h) mADASYN

(i) MWMOTE (j) mMWMOTE

Figure 1: Bayesian Sign Test heatmap for the sampling MAvA results vs the original MAvA results.

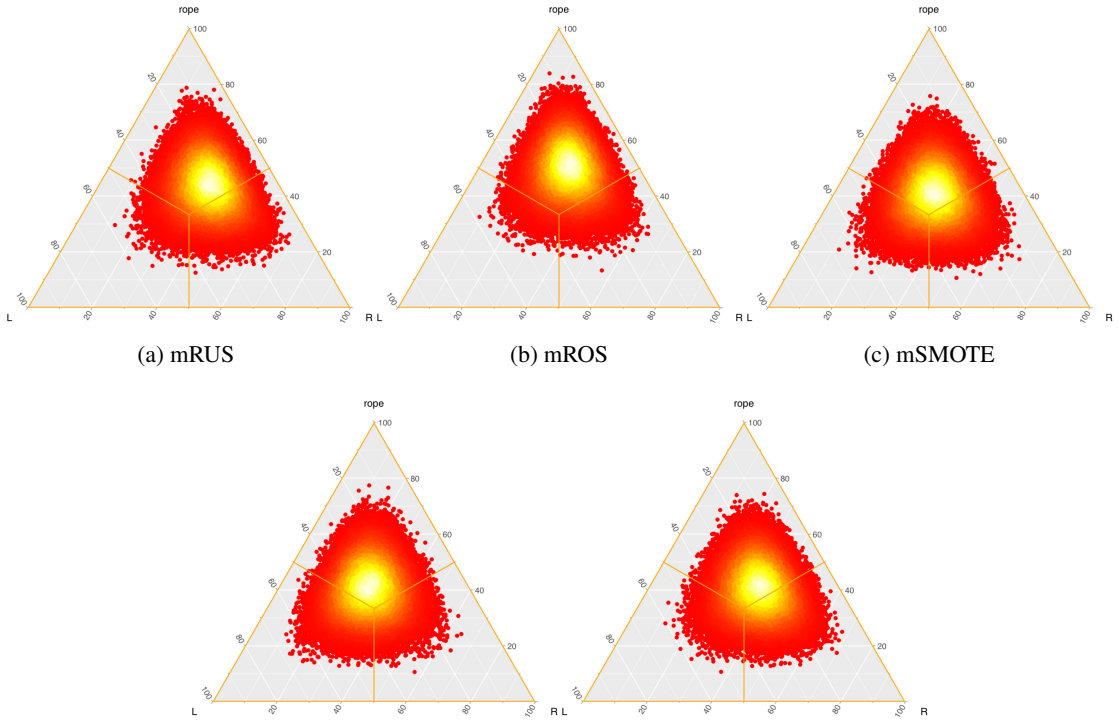
Monotonic ADASYN (mADASYN). RUS and mRUS are the methods which show the least improvement, probably because the under-sampling performed to balance the class distribution is too aggressive. The monotonic version is slightly better. At first sight, it is difficult to guess which is the best set of methods in terms of predictability: standard or monotonic sampling.

Bayesian Sign test has been used to thoroughly analyze the results and to clearly present the significance of the differences between the evaluated methods with the help of heatmap plots. Figure 1 represents the probability distributions of the differences between the sampling methods and the original results of the classifiers. As we can observe, the majority of the points in nearly all the plots are on the right side of the triangle. This means that the results obtained with sampling are significantly better than those obtained without sampling. With the exception of RUS (Fig. 1a) and mRUS (Fig. 1a), the test assigns an almost 0 probability in favor of the original classifiers. Monotonic RUS (Fig. 1a) is still valued, since most of the distribution is in the right region.

Figure 2 graphically compares the MAvA results of the monotonic versus standard sampling techniques. With a large part of distributions in the rope regions, there is no significant difference in terms of predicative capability between these two sampling groups. Monotonic RUS, ROS and MWMOTE (Figures 2a, 2b and 2e, respectively) are slightly better than their standard versions, since their distributions have shifted a bit to the right.

Table 8: MAvA results obtained by the selected classifiers with standard and monotonic sampling.

		Original	RUS	mRUS	ROS	mROS	SMOTE	mSMOTE	ADASYN	mADASYN	MWMOTE	mMWMOTE
MkNN	*balance*	0.6343	*0.7241*	0.6575	*0.6100*	*0.6100*	*0.6436*	0.5982	*0.6376*	0.5974	0.6637	*0.6969*
	car	0.9134	0.8824	*0.9759*	0.9553	*0.9673*	*0.9696*	0.9673	*0.9709*	0.9677	*0.9660*	0.9620
	ERA	0.1357	*0.1480*	0.1450	*0.1338*	*0.1338*	0.2501	*0.2878*	0.2794	*0.2949*	*0.2845*	0.2717
	ESL	0.4674	*0.3986*	0.3573	0.4621	*0.4834*	0.4660	*0.4764*	0.4711	*0.4754*	*0.4945*	0.4843
	LEV	0.3924	0.4182	*0.4372*	0.3834	*0.4783*	0.4682	*0.5097*	0.4737	*0.5089*	0.4467	*0.4680*
	SWD	0.4004	0.4692	*0.4908*	*0.4111*	*0.4111*	0.4608	*0.4703*	*0.4616*	0.4605	*0.4034*	0.3989
	windsorhousing	0.6755	*0.6467*	0.6318	*0.6755*	*0.6755*	0.6562	*0.6755*	0.6477	*0.6755*	0.6726	*0.6755*
	wisconsin	0.9625	*0.9745*	0.9716	0.9625	*0.9662*	*0.9699*	*0.9669*	*0.9721*	0.9709	*0.9709*	0.9706
	Avg:	0.5727	0.5827	*0.5834*	0.5742	*0.5907*	0.6106	*0.6190*	0.6143	*0.6189*	0.6128	*0.6160*
OLM	*balance*	0.6019	0.5243	*0.5961*	*0.6019*	*0.6019*	*0.6268*	0.6019	*0.6200*	0.6019	*0.6632*	0.6480
	car	0.8704	0.8424	*0.9231*	*0.9028*	0.8849	*0.8885*	0.8740	*0.8885*	0.8740	*0.8885*	0.8801
	ERA	0.2423	0.2276	*0.2289*	*0.2674*	0.2665	*0.2962*	0.2900	*0.3012*	0.2954	0.2716	*0.2825*
	ESL	0.3981	0.3242	*0.3588*	0.4070	*0.4250*	0.3974	*0.4358*	0.4019	*0.4342*	*0.4225*	0.4176
	LEV	0.4044	0.3801	*0.4188*	0.3729	*0.4225*	0.4165	*0.4530*	0.4075	*0.4560*	0.4155	*0.4315*
	SWD	0.4834	0.4024	*0.5004*	*0.4707*	0.4692	*0.4834*	*0.4834*	*0.4834*	*0.4834*	0.4830	*0.4834*
	windsorhousing	0.5320	*0.6358*	*0.6358*	*0.6352*	0.5599	*0.6444*	0.5442	*0.6324*	0.5410	*0.6487*	0.5553
	wisconsin	0.8418	*0.8814*	*0.8814*	*0.8668*	0.8446	*0.8567*	0.8432	*0.8821*	0.8425	*0.8551*	0.8467
	Avg:	0.5468	0.5273	*0.5679*	*0.5656*	0.5593	*0.5762*	0.5657	*0.5771*	0.5660	*0.5810*	0.5681
MID	*balance*	0.5705	*0.5531*	*0.5531*	0.5941	*0.6499*	*0.5849*	0.5848	0.5674	*0.5826*	0.5755	*0.5849*
	car	0.4603	*0.8191*	0.7840	*0.8810*	0.6499	0.8374	*0.8808*	*0.8786*	0.8640	0.8222	*0.8588*
	ERA	0.3182	*0.2907*	0.2872	0.3040	*0.3131*	*0.3059*	0.3026	*0.2926*	0.2916	0.2867	*0.3102*
	ESL	0.4254	*0.3213*	0.2767	0.4384	*0.4644*	0.4411	*0.4432*	*0.4608*	0.4404	0.4372	*0.4633*
	LEV	0.4731	*0.4592*	0.4517	*0.5732*	0.5641	*0.5367*	0.5048	*0.5101*	0.5073	*0.5157*	0.4958
	SWD	0.4470	0.4676	*0.4916*	0.5542	*0.5592*	*0.5192*	0.4565	*0.4993*	0.4940	0.4542	*0.4888*
	windsorhousing	0.7000	*0.6912*	0.6684	*0.7165*	0.6873	0.6642	*0.6764*	*0.7064*	0.7056	0.6868	*0.7128*
	wisconsin	0.9524	*0.9522*	0.9509	*0.9566*	0.9548	0.9619	*0.9628*	*0.9678*	0.9577	*0.9651*	0.9607
	Avg:	0.5434	*0.5693*	0.5580	*0.6272*	0.6053	*0.6064*	0.6015	*0.6104*	0.6054	0.5929	*0.6094*
MonMLP	*balance*	0.9051	0.8523	*0.8799*	0.9114	*0.9138*	0.8837	*0.9057*	0.8918	*0.9080*	0.8291	*0.8591*
	car	0.6487	*0.7788*	0.7260	0.7326	*0.7769*	*0.7754*	0.7532	0.7564	*0.8088*	0.6941	*0.7552*
	ERA	0.1723	0.1957	*0.2017*	0.2887	*0.2894*	*0.3084*	0.3027	*0.3108*	0.2913	*0.2886*	0.2795
	ESL	0.5370	*0.4503*	0.4435	*0.4941*	0.1111	*0.1111*	*0.1111*	*0.1111*	*0.1111*	*0.1325*	0.1111
	LEV	0.4839	*0.5077*	0.5023	*0.5259*	0.5000	*0.5561*	0.4697	*0.5367*	0.5018	*0.5451*	0.5450
	SWD	0.3497	0.4247	*0.4433*	0.4740	*0.4974*	0.4679	*0.4875*	0.4842	*0.4877*	*0.4415*	0.4014
	windsorhousing	0.5566	*0.6637*	0.6609	0.6544	*0.6581*	*0.6562*	0.6478	0.6359	*0.6478*	*0.6595*	0.6463
	wisconsin	0.8118	0.9537	*0.9564*	*0.9588*	0.9548	*0.9628*	0.9570	0.9648	*0.9688*	0.9611	*0.9665*
	Avg:	0.5582	*0.6034*	0.6017	*0.6300*	0.5877	*0.5902*	0.5794	0.5865	*0.5907*	0.5689	*0.5705*
	Complete Avg:	0.5553	0.5707	*0.5778*	*0.5993*	0.5857	*0.5959*	0.5914	*0.5971*	0.5953	0.5889	*0.5910*



(a) mRUS (b) mROS (c) mSMOTE

(d) mADASYN (e) mMWMOTE

Figure 2: Bayesian Sign Test heatmap for the monotonic sampling MAvA results vs the standard sampling MAvA results.

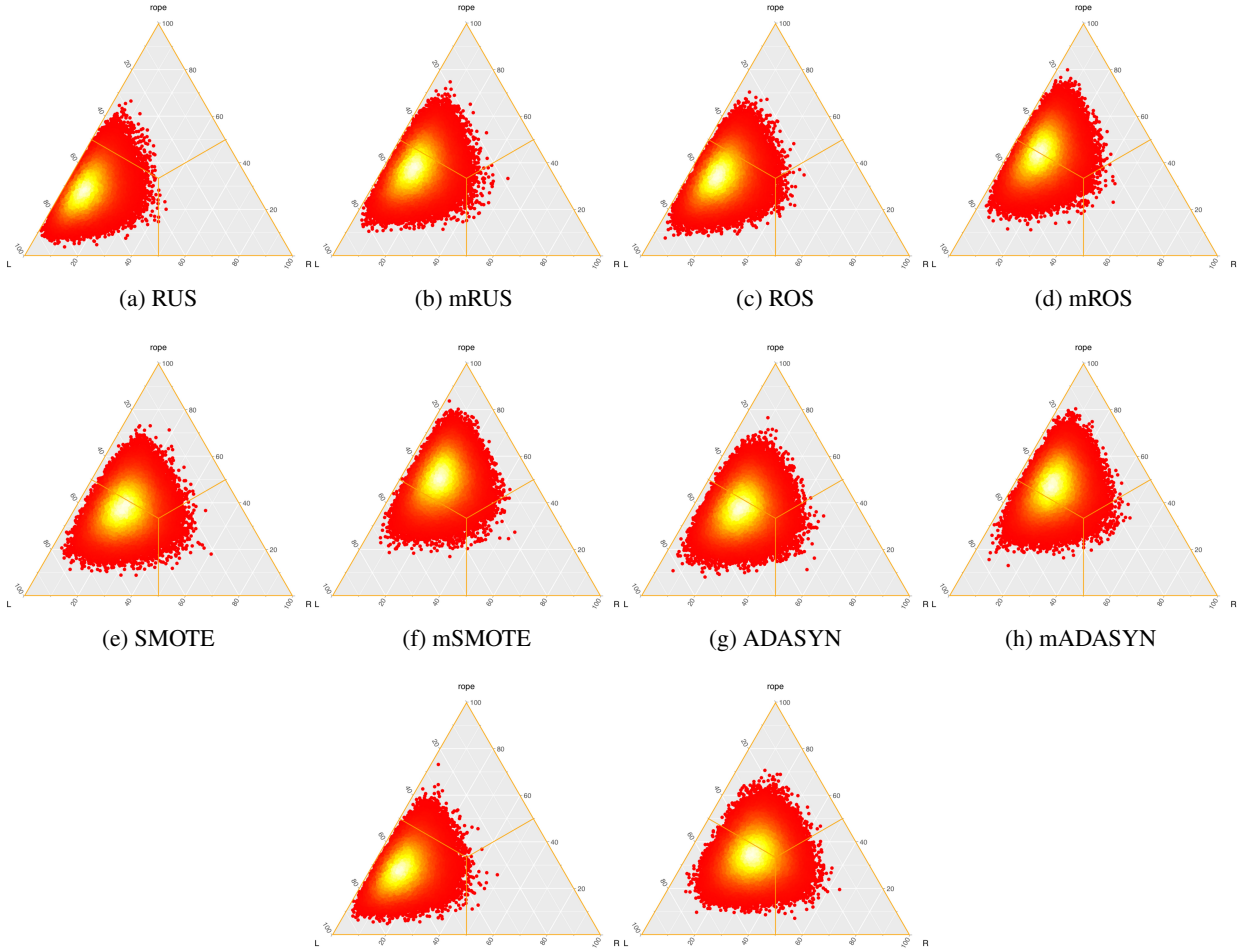
Table 9 gathers the NMI results reached using the different standard and monotonic sampling methods for every classifier per data-set. Cells in gray indicate the preservation of monotonicity. Even though there are not a huge amount of cells in gray, monotonic sampling results are quite similar to the original results when they do not achieve an improvement. It is worth mentioning that monotonicity is preserved on average for OLM and MID for nearly all monotonic sampling, except mROS and mRUS, respectively. On average, every monotonic sampling outperforms its counterpart for each classifier and the mean of all the results. The majority of best results per data-set and type of sampling, marked in italics, are located in the monotonic columns. These results represent a better preservation of monotonicity using the monotonic sampling compared to the traditional methods.

Figure 3 represents the Bayesian Sign test results for the difference of each of the sampling techniques and the NMI results obtained without them. At first glance, there is a change in the tendency of the distributions of the standard and the monotonic methods. Even though all of them have shifted to the left, towards the original results of the classifiers, the distributions of the monotonic sampling techniques are largely located in the rope region, i.e. the practical equivalence region. Monotonic RUS is excluded from this good behavior. Additionally, for monotonic MWMOTE (Fig. 3j), a good amount of points are located in the right section, indicating some monotonic improvements as compared to the original results.

Figure 4 shows the result of the Bayesian Sign test for the NMI comparison of monotonic sampling vs standard sampling methods. As we can observe, all the distributions are heavily shifted to the right, emphasizing the superiority of monotonic sampling over standard ones using NMI as measure. For the monotonic RUS and MWMOTE comparisons (Fig. 4a and 4e), the majority of the distribution is located in the right part. For the rest, the distributions are more or less shared by the rope and right sections. However, for monotonic ROS (Fig.

Table 9: NMI results obtained by the selected classifiers with standard and monotonic sampling.

		Original	RUS	mRUS	ROS	mROS	SMOTE	mSMOTE	ADASYN	mADASYN	MWMOTE	mMWMOTE
MkKNN	*balance*	0.0001	0.0006	*0.0003*	*0.0003*	*0.0003*	*0.0004*	0.0005	*0.0004*	0.0005	0.0004	*0.0003*
	car	0.0000	0.0004	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	ERA	0.0056	*0.0081*	0.0084	*0.0071*	*0.0071*	0.0113	*0.0112*	0.0146	*0.0114*	0.0118	*0.0113*
	ESL	0.0012	0.0072	*0.0030*	0.0014	*0.0012*	0.0013	*0.0012*	0.0015	*0.0011*	0.0011	*0.0010*
	LEV	0.0010	0.0066	*0.0044*	0.0016	*0.0012*	0.0019	*0.0012*	0.0018	*0.0012*	0.0019	*0.0016*
	SWD	0.0005	0.0043	*0.0026*	*0.0008*	*0.0008*	*0.0007*	0.0011	*0.0007*	0.0011	0.0011	*0.0009*
	windsorhousing	0.0000	0.0002	*0.0000*	*0.0000*	*0.0000*	0.0001	*0.0000*	0.0001	*0.0000*	*0.0000*	*0.0000*
	wisconsin	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	Avg:	0.0011	0.0034	*0.0023*	0.0014	*0.0013*	0.0020	*0.0019*	0.0024	*0.0019*	0.0020	*0.0019*
OLM	*balance*	0.0000	0.0001	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	0.0003	*0.0000*
	car	0.0000	0.0001	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	ERA	0.0063	*0.0040*	0.0041	0.0100	*0.0081*	0.0056	*0.0052*	0.0058	*0.0055*	0.0064	*0.0054*
	ESL	0.0025	0.0084	*0.0025*	*0.0023*	0.0032	*0.0023*	0.0025	*0.0023*	0.0025	*0.0018*	0.0027
	LEV	0.0043	0.0038	*0.0025*	*0.0028*	0.0043	*0.0039*	0.0043	*0.0028*	0.0044	*0.0051*	*0.0051*
	SWD	0.0015	0.0062	*0.0046*	*0.0016*	*0.0016*	*0.0015*	*0.0015*	*0.0015*	*0.0015*	0.0016	*0.0015*
	windsorhousing	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	0.0002	*0.0000*	0.0001	*0.0000*	0.0002	*0.0000*
	wisconsin	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	Avg:	0.0018	0.0028	*0.0017*	*0.0021*	*0.0021*	0.0017	*0.0017*	*0.0016*	0.0017	0.0019	*0.0018*
MID	*balance*	0.0017	*0.0056*	*0.0056*	0.0015	*0.0012*	*0.0012*	0.0017	0.0016	*0.0014*	0.0025	*0.0014*
	car	0.0034	*0.0050*	0.0060	0.0014	*0.0012*	0.0013	*0.0009*	0.0012	*0.0011*	0.0014	*0.0013*
	ERA	0.0086	*0.0064*	0.0071	0.0115	*0.0085*	*0.0087*	0.0090	0.0101	*0.0087*	*0.0106*	0.0109
	ESL	0.0023	0.0205	*0.0146*	0.0064	*0.0031*	0.0041	*0.0021*	0.0038	*0.0033*	0.0044	*0.0025*
	LEV	0.0024	*0.0116*	0.0139	0.0037	*0.0021*	*0.0027*	0.0028	0.0031	*0.0023*	0.0025	*0.0022*
	SWD	0.0020	0.0089	0.0107	0.0026	*0.0025*	0.0027	*0.0025*	0.0029	*0.0027*	*0.0024*	0.0025
	windsorhousing	0.0035	0.0135	*0.0014*	0.0127	*0.0040*	0.0094	*0.0024*	0.0135	*0.0043*	0.0094	*0.0034*
	wisconsin	0.0001	*0.0001*	*0.0001*	*0.0000*	*0.0000*	*0.0001*	*0.0001*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	Avg:	0.0030	0.0090	*0.0074*	0.0050	*0.0028*	0.0038	*0.0027*	0.0045	*0.0030*	0.0042	*0.0030*
MonMLP	*balance*	0.0000	*0.0001*	*0.0001*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	0.0003	*0.0000*
	car	0.0001	*0.0006*	0.0010	0.0006	*0.0003*	*0.0003*	*0.0003*	*0.0003*	0.0009	*0.0002*	0.0005
	ERA	0.0026	*0.0050*	0.0054	0.0048	*0.0043*	0.0063	*0.0056*	0.0069	*0.0055*	*0.0056*	*0.0056*
	ESL	0.0003	0.0030	*0.0025*	0.0016	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	0.0196	*0.0000*
	LEV	0.0008	*0.0041*	0.0049	0.0027	*0.0018*	0.0023	*0.0018*	0.0023	*0.0015*	0.0020	*0.0016*
	SWD	0.0004	0.0044	*0.0024*	*0.0011*	0.0012	*0.0013*	0.0019	0.0021	*0.0015*	*0.0011*	0.0028
	windsorhousing	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	wisconsin	0.0000	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
	Avg:	0.0005	0.0021	*0.0020*	0.0013	*0.0010*	0.0013	*0.0012*	0.0015	*0.0012*	0.0036	*0.0013*
	Complete Avg:	0.0016	0.0043	*0.0034*	0.0025	*0.0018*	0.0022	*0.0019*	0.0025	*0.0020*	0.0029	*0.0020*



(a) RUS (b) mRUS (c) ROS (d) mROS

(e) SMOTE (f) mSMOTE (g) ADASYN (h) mADASYN

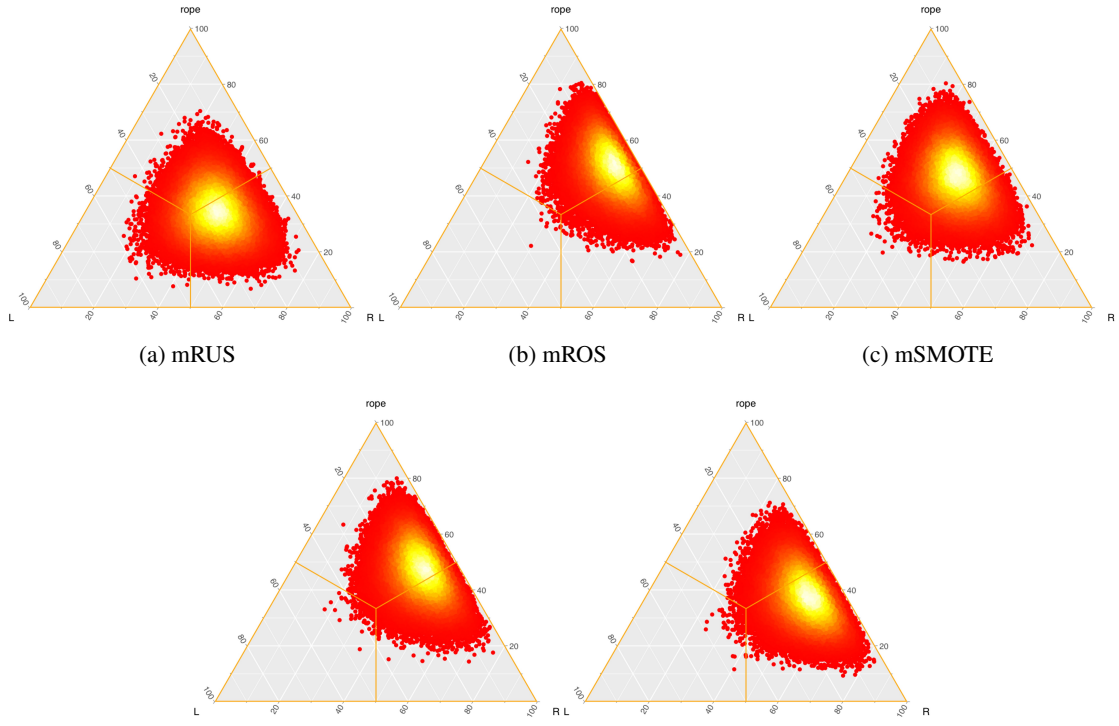
(i) MWMOTE (j) mMWMOTE

Figure 3: Bayesian Sign Test heatmap for the sampling NMI results vs the original NMI results.

4b), ADASYN (Fig. 4d) and MWMOTE (Fig. 4e), there are barely no points on the left side, therefore their superiority is significant.

Recapitulating, the five monotonic sampling techniques based on monotonic chains increase the predictability of the classifier and, at the same time, to a large extent, improve the standard sampling approaches in terms of monotonicity. mRUS and mMWMOTE show a marked improvement as compared to the standard versions.

Among the monotonic sampling methods analyzed, it is difficult to choose the best technique, since the best results strongly depend on the characteristics of the problem and classifier used. On average, mADASYN results in the best techniques in terms of MAvA, while mROS achieves the best monotonic results. mRUS is the worst on average, but it obtains fairly good results for those datasets where there is a good amount of representatives of the minority classes, such as *car* and *wisconsin*, especially when combined with OLM and MonMLP. mROS is very interesting for problems with repeated samples of different classes. Comparing the methods based on linear interpolation, mSMOTE is the best at maintaining monotonicity and a good trade-off between predictability improvement and monotonicity maintenance.



(a) mRUS (b) mROS (c) mSMOTE

(d) mADASYN (e) mMWMOTE

Figure 4: Bayesian Sign Test heatmap for the monotonic sampling NMI results vs the standard sampling NMI results.

6 Concluding Remarks

The imbalanced class problem is a real issue for many real-life applications and commonly used data-sets in classification with monotonic constraints. However, as shown empirically in this paper, traditional solutions are not valid for those scenarios in which monotonic constraints are assumed, since these methods heavily degrade the monotonicity of the data-sets.

In this paper, a new sampling scheme based on monotonic chains was designed to consider the constraints of monotonicity and to be able to be applied to monotonic data-sets. The main objective was to improve the accuracy of those minority classes, while preserving the monotonicity of the models. We have developed this scheme within 5 different well-known sampling techniques, one under-sampling and four over-sampling algorithms. These new methods have been empirically tested with their standard versions, statistically resulting in the same results in terms of prediction enhancement and in much better preservation of the monotonicity of the predictions. In some cases, they have even improved the monotonicity of the models compared to the original without sampling. Among the chain based sampling methods, mROS is the best in terms of monotonicity and mSMOTE is the best in predictability improvement and monotonicity preservation.

A Results tables for standard and ordinal sampling

Table 10: MAvA results obtained by the selected classifiers with standard and ordinal sampling.

		Original	RUS	ROS	SMOTE	ADASYN	ADASYNOrd	MWMOTE	MWMOTEOrd	CWOSOrd
MKNN	*balance*	0.6343	0.7241	0.6100	0.6436	0.6376	0.6376	0.6637	0.6637	0.6657
	car	0.9134	0.8824	0.9553	0.9696	0.9709	0.9709	0.9660	0.9636	0.9645
	ERA	0.1357	0.1480	0.1338	0.2501	0.2794	0.2588	0.2845	0.2769	0.2911
	ESL	0.4674	0.3986	0.4621	0.4660	0.4711	0.3228	0.4945	0.4915	0.5123
	LEV	0.3924	0.4182	0.3834	0.4682	0.4737	0.3925	0.4467	0.4414	0.4491
	SWD	0.4004	0.4692	0.4111	0.4608	0.4616	0.4599	0.4034	0.4031	0.4044
	windsorhousing	0.6755	0.6467	0.6755	0.6562	0.6477	0.6477	0.6726	0.6726	0.6526
	wisconsin	0.9625	0.9745	0.9625	0.9699	0.9721	0.9721	0.9709	0.9709	0.9702
	Avg:	0.5727	0.5827	0.5742	0.6106	0.6143	0.5828	0.6128	0.6105	0.6137
OLM	*balance*	0.6019	0.5243	0.6019	0.6268	0.6200	0.6200	0.6632	0.6632	0.6577
	car	0.8704	0.8424	0.9028	0.8885	0.8885	0.8885	0.8885	0.8909	0.8885
	ERA	0.2423	0.2276	0.2674	0.2962	0.3012	0.2983	0.2716	0.2794	0.2550
	ESL	0.3981	0.3242	0.4070	0.3974	0.4019	0.4069	0.4225	0.4203	0.4253
	LEV	0.4044	0.3801	0.3729	0.4165	0.4075	0.4054	0.4155	0.4109	0.4077
	SWD	0.4834	0.4024	0.4707	0.4834	0.4834	0.4834	0.4830	0.4834	0.4834
	windsorhousing	0.5320	0.6358	0.6352	0.6444	0.6324	0.6324	0.6487	0.6487	0.6293
	wisconsin	0.8418	0.8814	0.8668	0.8567	0.8821	0.8821	0.8551	0.8551	0.8502
	Avg:	0.5468	0.5273	0.5656	0.5762	0.5771	0.5771	0.5810	0.5815	0.5746
MID	*balance*	0.5705	0.5531	0.5941	0.5849	0.5674	0.5674	0.5755	0.5755	0.5621
	car	0.4603	0.8191	0.8810	0.8374	0.8786	0.8683	0.8222	0.8422	0.8723
	ERA	0.3182	0.2907	0.3040	0.3059	0.2926	0.2997	0.2867	0.3044	0.2929
	ESL	0.4254	0.3213	0.4384	0.4411	0.4608	0.4638	0.4372	0.4430	0.4179
	LEV	0.4731	0.4592	0.5732	0.5367	0.5101	0.5226	0.5157	0.4981	0.4947
	SWD	0.4470	0.4676	0.5542	0.5192	0.4993	0.4441	0.4542	0.4689	0.4566
	windsorhousing	0.7000	0.6912	0.7165	0.6642	0.7064	0.6489	0.6868	0.6966	0.6741
	wisconsin	0.9524	0.9522	0.9566	0.9619	0.9678	0.9490	0.9651	0.9607	0.9593
	Avg:	0.5434	0.5693	0.6272	0.6064	0.6104	0.5955	0.5929	0.5987	0.5912
OSDL	*balance*	0.5159	0.5130	0.5299	0.3980	0.3965	0.3965	0.5106	0.5106	0.5181
	car	0.9179	0.4976	0.4594	0.4648	0.4638	0.4621	0.4562	0.4615	0.4496
	ERA	0.2525	0.2005	0.1990	0.1988	0.2000	0.1987	0.1878	0.1923	0.1967
	ESL	0.4934	0.1470	0.1429	0.1618	0.1612	0.1618	0.1560	0.1560	0.1560
	LEV	0.4936	0.3532	0.3597	0.3538	0.3545	0.3791	0.3594	0.3606	0.3318
	SWD	0.4588	0.4515	0.4497	0.4344	0.4359	0.4368	0.4334	0.4334	0.4311
	windsorhousing	0.6334	0.5854	0.5400	0.5480	0.5424	0.5424	0.5424	0.5424	0.5392
	wisconsin	0.9617	0.9211	0.9211	0.9211	0.9211	0.9211	0.9211	0.9211	0.9211
	Avg:	0.5909	0.4587	0.4502	0.4351	0.4344	0.4373	0.4459	0.4472	0.4429
MonMLP	*balance*	0.9051	0.8523	0.9114	0.8837	0.8918	0.8918	0.8291	0.8291	0.8258
	car	0.6487	0.7788	0.7326	0.7754	0.7564	0.8133	0.6941	0.7508	0.7552
	ERA	0.1723	0.1957	0.2887	0.3084	0.3108	0.2927	0.2886	0.3023	0.2867
	ESL	0.5370	0.4503	0.4941	0.1111	0.1111	0.1111	0.1325	0.1305	0.2985
	LEV	0.4839	0.5077	0.5259	0.5561	0.5367	0.5029	0.5451	0.5403	0.5370
	SWD	0.3497	0.4247	0.4740	0.4679	0.4842	0.5270	0.4415	0.4431	0.4419
	windsorhousing	0.5566	0.6637	0.6544	0.6562	0.6359	0.6359	0.6595	0.6595	0.5000
	wisconsin	0.8118	0.9537	0.9588	0.9628	0.9648	0.9648	0.9611	0.9611	0.9221
	Avg:	0.5582	0.6034	0.6300	0.5902	0.5865	0.5924	0.5689	0.5771	0.5709
	Complete Avg:	0.5624	0.5620	0.5694	0.5637	0.5645	0.5570	0.5603	0.5630	0.5587

Table 11: NMI results obtained by the selected classifiers with standard and ordinal sampling.

		Original	RUS	ROS	SMOTE	ADASYN	ADASYNOrd	MWMOTE	MWMOTEOrd	CWOSOrd
MKNN	*balance*	0.0001	0.0006	0.0003	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
	car	0.0000	0.0004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	ERA	0.0056	0.0081	0.0071	0.0113	0.0146	0.0172	0.0118	0.0130	0.0106
	ESL	0.0012	0.0072	0.0014	0.0013	0.0015	0.0125	0.0011	0.0011	0.0012
	LEV	0.0010	0.0066	0.0016	0.0019	0.0018	0.0116	0.0019	0.0019	0.0024
	SWD	0.0005	0.0043	0.0008	0.0007	0.0007	0.0007	0.0011	0.0011	0.0010
	windsorhousing	0.0000	0.0002	0.0000	0.0001	0.0001	0.0001	0.0000	0.0000	0.0053
	wisconsin	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Avg:	0.0011	0.0034	0.0014	0.0020	0.0024	0.0053	0.0020	0.0022	0.0026
OLM	*balance*	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0003	0.0003	0.0003
	car	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	ERA	0.0063	0.0040	0.0100	0.0056	0.0058	0.0051	0.0064	0.0057	0.0059
	ESL	0.0025	0.0084	0.0023	0.0023	0.0023	0.0023	0.0018	0.0018	0.0023
	LEV	0.0043	0.0038	0.0028	0.0039	0.0028	0.0027	0.0051	0.0053	0.0053
	SWD	0.0015	0.0062	0.0016	0.0015	0.0015	0.0015	0.0016	0.0015	0.0015
	windsorhousing	0.0000	0.0000	0.0000	0.0002	0.0001	0.0001	0.0002	0.0002	0.0004
	wisconsin	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Avg:	0.0018	0.0028	0.0021	0.0017	0.0016	0.0015	0.0019	0.0019	0.0020
MID	*balance*	0.0017	0.0056	0.0015	0.0012	0.0016	0.0016	0.0025	0.0025	0.0028
	car	0.0034	0.0050	0.0014	0.0013	0.0012	0.0008	0.0014	0.0012	0.0014
	ERA	0.0086	0.0064	0.0115	0.0087	0.0101	0.0103	0.0106	0.0102	0.0103
	ESL	0.0023	0.0205	0.0064	0.0041	0.0038	0.0030	0.0044	0.0033	0.0038
	LEV	0.0024	0.0116	0.0037	0.0027	0.0031	0.0031	0.0025	0.0025	0.0030
	SWD	0.0020	0.0089	0.0026	0.0027	0.0029	0.0027	0.0024	0.0020	0.0022
	windsorhousing	0.0035	0.0135	0.0127	0.0094	0.0135	0.0188	0.0094	0.0138	0.0057
	wisconsin	0.0001	0.0001	0.0000	0.0001	0.0000	0.0002	0.0000	0.0000	0.0001
	Avg:	0.0030	0.0090	0.0050	0.0038	0.0045	0.0051	0.0042	0.0044	0.0037
OSDL	*balance*	0.0006	0.0016	0.0000	0.0058	0.0060	0.0060	0.0022	0.0022	0.0015
	car	0.0000	0.0019	0.0025	0.0034	0.0034	0.0033	0.0057	0.0052	0.0035
	ERA	0.0049	0.0009	0.0301	0.0310	0.0299	0.0274	0.0251	0.0302	0.0326
	ESL	0.0006	0.0000	0.0963	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	LEV	0.0004	0.0051	0.0010	0.0005	0.0006	0.0009	0.0011	0.0068	0.0163
	SWD	0.0009	0.0030	0.0009	0.0009	0.0009	0.0009	0.0008	0.0008	0.0008
	windsorhousing	0.0036	0.0000	0.0002	0.0001	0.0002	0.0002	0.0002	0.0002	0.0002
	wisconsin	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Avg:	0.0009	0.0020	0.0164	0.0052	0.0051	0.0048	0.0044	0.0057	0.0069
MonMLP	*balance*	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0003	0.0003	0.0002
	car	0.0001	0.0006	0.0006	0.0003	0.0003	0.0002	0.0002	0.0002	0.0001
	ERA	0.0026	0.0050	0.0048	0.0063	0.0069	0.0069	0.0056	0.0071	0.0111
	ESL	0.0003	0.0030	0.0016	0.0000	0.0000	0.0000	0.0196	0.0190	0.0426
	LEV	0.0008	0.0041	0.0027	0.0023	0.0023	0.0035	0.0020	0.0019	0.0035
	SWD	0.0004	0.0044	0.0011	0.0013	0.0021	0.0022	0.0011	0.0011	0.0008
	windsorhousing	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	wisconsin	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0022
	Avg:	0.0005	0.0021	0.0013	0.0013	0.0015	0.0016	0.0036	0.0037	0.0076
	Complete Avg:	0.0015	0.0040	0.0052	0.0028	0.0030	0.0037	0.0032	0.0036	0.0045

Acknowledgements

This work was supported by the Spanish National Research Project TIN2017-89517-P and the Project BigDaP-TOOLS - Ayudas Fundación BBVA a Equipos de Investigación Científica 2016 and by a research scholarship (FPU) given to the author Sergio González by the Spanish Ministry of Education, Culture and Sports. Additionally, this paper is the result of a collaboration with the Prof. Sheng-Tun Li during the international stay made by Sergio González at National Cheng Kung University. This stay was partially supported by the 2017

Summer Program in Taiwan for Spanish Graduate Students held by the Ministry of Science and Technology of Taiwan (R.O.C.).

References

[1] Chih-Chuan Chen and Sheng-Tun Li. Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16):7235–7247, 2014.

[2] Jie Sun, Jie Lang, Hamido Fujita, and Hui Li. Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates. *Information Sciences*, 425:76–91, 2018.

[3] Marina Velikova and Hennie Daniels. Decision trees for monotone price models. *Computational Management Science*, 1(3):231–244, Oct 2004.

[4] Jose-Ramon Cano, Naif R Aljohani, Rabeeh Ayaz Abbasi, Jalal S Alowidbi, and Salvador Garcia. Prototype selection to improve monotonic nearest neighbor. *Engineering Applications of Artificial Intelligence*, 60:128–135, 2017.

[5] Ye Liu, Luming Zhang, Liqiang Nie, Yan Yan, and David S Rosenblum. Fortune teller: Predicting your career path. In *AAAI*, volume 2016, pages 201–207, 2016.

[6] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1):29–43, 1995.

[7] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113 – 141, 2013.

[8] Ronaldo C. Prati, Gustavo E. A. P. A. Batista, and Diego F Silva. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*, 45(1):247–270, 2015.

[9] Ye Liu, Liqiang Nie, Lei Han, Luming Zhang, and David S Rosenblum. Action2activity: Recognizing complex activities from sensor data. In *IJCAI*, volume 2015, pages 1617–1623, 2015.

[10] Stijn Lievens, Bernard De Baets, and Kim Cao-Van. A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Annals of Operations Research*, 163(1):115–142, 2008.

[11] W. Duivesteijn and A. Feelders. Nearest neighbour classification with monotonicity constraints. In *ECML/PKDD (1)*, pages 301–316, 2008.

[12] Stijn Lievens and Bernard De Baets. Supervised ranking in the weka environment. *Information Sciences*, 180(24):4763–4771, 2010.

[13] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, pages 321–357, 2002.

[14] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE, 2008.

[15] Sukarna Barua, Md Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2014.

[16] Juan Pablo Sánchez-Crisostomo, Roberto Alejo, Erika López-González, Rosa María Valdovinos, and J. Horacio Pacheco-Sánchez. Empirical analysis of assessments metrics for multi-class imbalance learning on the back-propagation context. In *Advances in Swarm Intelligence: 5th International Conference, ICSI 2014*, pages 17–23. Springer, 2014.

[17] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937.

[18] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[19] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[20] W. Kotłowski and R. Słowiński. On nonparametric ordinal classification with monotonicity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2576–2589, 2013.

[21] A. Ben-David. Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications. *Decision Sciences*, 23:1357–1372, 1992.

[22] Javier García, A. M. AlBar, N. R. Aljohani, J.-R. Cano, and S. García. Hyperrectangles selection for monotonic classification by using evolutionary algorithms. *International Journal of Computational Intelligence Systems*, 9(1):184–202, 2016.

[23] Christophe Marsala and Davide Petturiti. Rank discrimination measures for enforcing monotonicity in decision tree induction. *Information Sciences*, 291:143–171, 2015.

[24] Jesús Alcalá-Fdez, Rafael Alcalá, Sergio González, Yusuke Nojima, and Salvador García. Evolutionary fuzzy rule-based methods for monotonic classification. *IEEE Transactions on Fuzzy Systems*, 25(6):1376–1390, 2017.

[25] Shenglei Pei and Qinghua Hu. Partially monotonic decision trees. *Information Sciences*, 424:104–117, 2018.

[26] Sheng-Tun Li and Chih-Chuan Chen. A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge. *IEEE Transactions on Fuzzy Systems*, 23(5):1713–1727, 2015.

[27] Bernhard Lang. Monotonic multi-layer perceptron networks as universal approximators. In *International Conference on Artificial Neural Networks*, pages 31–37. Springer, 2005.

[28] Francisco Fernández-Navarro, Annalisa Riccardi, and Sante Carloni. Ordinal neural networks without iterative tuning. *IEEE Transactions on Neural Network and Learning Systems*, 25(11):2075–2085, 2014.

[29] Hong Zhu, Eric CC Tsang, Xi-Zhao Wang, and Rana Aamir Raza Ashfaq. Monotonic classification extreme learning machine. *Neurocomputing*, 225:205–213, 2017.

[30] K. Dembczyński, W. Kotłowski, and R. Słowiński. Learning rule ensembles for ordinal classification with monotonicity constraints. *Fundamenta Informaticae*, 94(2):163–178, 2009.

[31] Yuhua Qian, Hang Xu, Jiye Liang, Bing Liu, and Jieting Wang. Fusing monotonic decision trees. *IEEE Transactions on Knowledge Data Engineering*, 27(10):2717–2728, 2015.

[32] Sergio González, Francisco Herrera, and Salvador García. Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing*, 33(4):367–388, 2015.

[33] Qinghua Hu, Weiwei Pan, Yanping Song, and Daren Yu. Large-margin feature selection for monotonic classification. *Knowledge-Based Systems*, 31:8 – 18, 2012.

[34] J.-R. Cano and S. García. Training set selection for monotonic ordinal classification. *Data & Knowledge Engineering*, 112:94 – 105, 2017.

[35] María Pérez-Ortiz, Pedro Antonio Gutierrez, César Hervás-Martínez, and Xin Yao. Graph-based approaches for over-sampling in the context of ordinal regression. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1233–1245, 2015.

[36] Iman Nekooeimehr and Susana K. Lai-Yuen. Cluster-based weighted oversampling for ordinal regression (cwos-ord). *Neurocomputing*, 218:51 – 60, 2016.

[37] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: Special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[38] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, 2012.

[39] Nitesh V Chawla, David A Cieslak, Lawrence O Hall, and Ajay Joshi. Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery*, 17(2):225–252, 2008.

[40] Fenglian Li, Xueying Zhang, Xiqian Zhang, Chunlei Du, Yue Xu, and Yu-Chu Tian. Cost-sensitive and hybrid-attribute measure multi-decision tree over imbalanced data sets. *Information Sciences*, 422:242–256, 2018.

[41] E. Ramentol, S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, and F. Herrera. Ifrowann: Imbalanced fuzzy-rough ordered weighted average nearest neighbor classification. *IEEE Transactions on Fuzzy Systems*, 23(5):1622–1637, 2015.

[42] Shounak Datta and Swagatam Das. Near-bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs. *Neural Networks*, 70:39 – 52, 2015.

[43] Jin Huang and Charles X Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 2005.

[44] Rob Potharst, Arie Ben-David, and Michiel C. van Wezel. Two algorithms for generating structured and unstructured monotone ordinal data sets. *Engineering Applications of Artificial Intelligence*, 22(4-5):491–96, 2009.

[45] Ad Feelders. Monotone relabeling in ordinal classification. In *ICDM*, pages 803–808. IEEE Computer Society, 2010.

[46] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 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(8):1761 – 1776, 2011.

[47] Alberto Fernández, Victoria López, Mikel Galar, María José del Jesus, and Francisco Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 42:97 – 110, 2013.

[48] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. NMC: nearest matrix classification–a new combination model for pruning one-vs-one ensembles by transforming the aggregation problem. *Information Fusion*, 36:26–51, 2017.

[49] Zhong-Liang Zhang, Xing-Gang Luo, Sergio González, Salvador García, and Francisco Herrera. DRCW-ASEG: One-versus-one distance-based relative competence weighting with adaptive synthetic example generation for multi-class imbalanced datasets. *Neurocomputing*, 285:176–187, 2018.

[50] Loïc Cerf, Dominique Gay, Nazha Selmaoui-Folcher, Bruno Crémilleux, and Jean-François Boulicaut. Parameter-free classification in multi-class imbalanced data sets. *Data & Knowledge Engineering*, 87:109–129, 2013.

[51] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. Keel 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, 2017.

[52] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. rNPBST: An R package covering non-parametric and bayesian statistical tests. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 281–292. Springer, 2017.

4 Fuzzy *k*-Nearest Neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise

- González, S., García, S., Li, S. T., John, R., & Herrera, F. (2020). Fuzzy k-Nearest Neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise. *Neurocomputing*.
  - Status: **Accepted**
  - Impact Factor (JCR 2018): **4.072**
  - Subject Category: **Computer Science, Artificial Intelligence**
  - Rank: **28/134**
  - Quartile: **Q1**

FUZZY k-NEAREST NEIGHBORS WITH MONOTONICITY CONSTRAINTS: MOVING TOWARDS THE ROBUSTNESS OF MONOTONIC NOISE

Sergio González
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
sergiogvz@decsai.ugr.es

Salvador García
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
salvagl@decsai.ugr.es

Sheng-Tun Li
Department of Industrial and Information Management
Institute of Information Management
Center for Innovative FinTech Business Models
National Cheng Kung University, Tainan 701, Taiwan
stli@mail.ncku.edu.tw

Robert John
ASAP Research Group School of Computer Science
University of Nottingham NG8 1BB, Nottingham, UK
robert.john@nottingham.ac.uk

Francisco Herrera
Department of Computer Science
and Artificial Intelligence
University of Granada, Granada, Spain 18071
Faculty of Computing and Information Technology
King Abdulaziz University, Jeddah, Saudi Arabia
herrera@decsai.ugr.es

ABSTRACT

This paper proposes a new model based on Fuzzy k-Nearest Neighbors for classification with monotonic constraints, Monotonic Fuzzy k-NN (MonFkNN). Real-life data-sets often do not comply with monotonic constraints due to class noise. MonFkNN incorporates a new calculation of fuzzy memberships, which increases robustness against monotonic noise without the need for relabeling. Our proposal has been designed to be adaptable to the different needs of the problem being tackled. In several experimental studies, we show significant improvements in accuracy while matching the best degree of monotonicity obtained by comparable methods. We also show that MonFkNN empirically achieves improved performance compared with Monotonic k-NN in the presence of large amounts of class noise.

***K*eywords** Fuzzy k-NN · monotonic constraints · ordinal classification · ordinal regression · class noise.

1 Introduction

Monotonic constraints are prior-knowledge of some ordinal classification or regression problems about the order relationships between attributes and class labels [1]. Consider the example of house pricing. The following constraints are applied: A bigger house in the same neighborhood is constrained by higher prices as compared to smaller houses with the same features. That is, the classifier decisions should not decrease in the presence of better features while the rest remains the same. These prior constraints are required by many real-life evaluation problems, such as credit risk modeling [2] and lecturer evaluation [3]. These problems are known as Classification with Monotonic Constraints or Monotonic Classification [4].

These learning tasks have additional objectives besides accurate models, such as the monotonic consistency of predictions and minimization of the misclassification costs. The latter is also relevant since the errors between ordered classes do not hold the same importance. More metrics must be used during the learning and validation of the models. However, these other objectives may impair accuracy [5]. Hence, a fair balance must be sought between the different needs of each problem.

Standard classifiers are discouraged for monotonic classification since they do not contemplate these constraints and their predictions violate the monotonicity required by certain applications. A classic example of these non-monotonic models is the standard decision tree [4]. Standard k-Nearest Neighbors algorithm also does not take these restrictions into account, which may lead to further harm as a result of their presence in preprocessing techniques [6].

In recent years, new algorithms have been designed to minimize the number of monotonic violations in their predictions [4, 7, 1], i.e. the number of pairs of instances that break monotonicity [4]. To do so, some approaches focus their entire learning mechanism just on monotonicity. This strategy usually achieves completely monotonic models, but it could lead to wrong generalizations being made that are different to the knowledge of the problem. Others infer monotonic relations from the training set while maximizing their accuracy. These models have been adapted from different families of classifiers [1], such as decision trees [4, 8, 9], support vector machines [2], fuzzy model based classifiers [10, 11], neural networks [12, 13] and ensemble learning [14, 15, 7].

Instance-based learning has proven to be a good approach for monotonic classification [16, 17, 18, 19]. However, some of these methods, such as Monotonic k-Nearest Neighbors [17] (MkNN), need to learn from a fully monotonic set to ensure monotonic predictions. This is rarely the case in real-life scenarios, where class noise and discrepancies are common. Therefore, data preprocessing [20, 21, 22, 6] and relabeling strategies [23, 24] must be used to remove non-monotonic samples or to change their class labels in order to force a monotonic set.

In standard classification, Fuzzy k-Nearest Neighbors [25] is a very solid method with high performance, thanks to its high robustness to class noise [26]. This class noise robustness mainly lies in the extraction of the class memberships for the crisp training samples by nearest neighbor rule. In this process, the class memberships of noisy instances are shared with surrounding classes and the incorrectly assigned class looses its influence. However, these mechanisms do not consider monotonic constraints and Fuzzy k-NN cannot deal with monotonic violations or monotonic noise in the training set.

In this paper, a new model designed on the basis of Fuzzy k-NN with notions of MkNN is proposed to take monotonic constraints into account, and is called Monotonic Fuzzy k-Nearest Neighbors (MonFkNN). MonFkNN has been designed with three desired features:

(i) Robustness against monotonic violations.

(ii) Monotonic predictions without a pure monotonic training set.

(iii) Flexibility in its configurations covering different needs of performance.

With these objectives in mind, MonFkNN has been designed with new mechanisms to manage monotonicity constraints and the monotonic violations in the training set. The main contributions of the MonFkNN design are:

(i) The initial robustness of Fuzzy k-NN has been redesigned to mitigate the influence of monotonic violations. Firstly, the violations due to sample replicas with different classes are joined to form one class membership. Then, our approach incorporates a strictly monotonic nearest neighbor rule to the calculation of the memberships of the training examples.

(ii) These monotonically constrained memberships and their medians are used in the prediction phase. The class memberships aggregation of MonFkNN is also monotonically constrained by the nearest neighbor extraction or a penalty to the contribution of non-monotonic instances.

(iii) MonFkNN was built as a flexible classifier that covers different necessities of monotonicity and accuracy by tuning its parameters. It can be configured with a rigidly monotonic or standard k-NN rule if monotonicity or precision is preferred in the predictions, respectively.

All these mechanisms reinforce the robustness of our proposal against monotonic noise without the need for relabeling. We understand monotonic noise as being the actual noise that can alter the class labels and, as a result, change the monotonic constraints among the samples in the data. Their parameters make our proposal adaptable to the different objectives of monotonic classification. We distinguish two different parameter configurations: a pure monotonic version in which monotonicity is prioritized, and an approximate configuration that focuses more on the prediction accuracy.

We have performed several empirical studies to verify the desired features of MonFkNN. First, different behaviors of its two configurations are empirically analyzed and compared to the original FkNN. Then, our proposal is compared with 7 methods from the state-of-the-art, exhibiting substantial improvements in accuracy and maintaining the best degree of monotonicity. Finally, the robustness of our method against monotonic noise, i.e. monotonic violations, is shown in contrast to MkNN. In this last experiment, MonFkNN performs considerably better than Monotonic k-NN in scenarios with large amounts of class noise. The experimental framework used consists of 12 data-sets commonly used in monotonic classification, 7 monotonic classifiers and 3 metrics covering different aspects of performance: Accuracy, Mean Absolute Error and Non-Monotonic Index. All results are additionally validated with the non-parametric statistical Wilcoxon and Friedman rank [27, 28] and Bayesian Sign tests [29].

The paper is organized as follows. In Section 2, we present the problem of classification with monotonic constraints and the methods related to our proposal: MkNN and Fuzzy k-NN. Section 3 is dedicated to explaining our model MonFkNN in detail and its algorithmic differences as compared to FkNN. The experimental framework used in the different empirical studies is presented in Section 4. In Section 5, the previously mentioned empirical studies are carried out and analyzed. Finally, the main conclusions of this study are stated in Section 6.

2 Preliminaries

In this section, we introduce the preliminaries needed: Classification with monotonicity constraints, Monotonic k-Nearest Neighbors and the original Fuzzy k-Nearest Neighbors.

2.1 Monotonic Classification

Monotonic classification [1] is an ordinal regression problem with monotonic constraints relating to the order of the variables and the class labels. Ordinal regression and/or classification can be seen as a nonstandard classification problem [30], which attempts to minimize the difference between the predicted labels and the real

labels. Classification with monotonic constraints is also considered to be a nonstandard supervised learning problem [30].

Formally, monotonic classification aims to predict the class label y from input vector x with Q number of features, where $y \in \mathcal{Y} = \{l_1, l_2, \ldots, l_c\}$ and x represents an individual of our classification problem. The categories $\mathcal{Y}$ are arranged in an order relation $\prec$ as $l_1 \prec l_2 \prec \ldots \prec l_c$. And, as the main property of monotonic classification, the attributes and class predictions are monotonically constrained by the problem prior-knowledge, i.e. $x \succeq x' \rightarrow f(x) \geq f(x')$ [31], where $x \succeq x'$ implies $\forall_{j=1,\ldots,Q}, x_j \geq x'_j$, that is, x dominates x'. Therefore, the main objective is to build classifiers that do not violate these constraints, otherwise known as monotonic classifiers.

Two different types of monotonic classifiers can be distinguished: approximate monotonic models, which minimizes the number of monotonic violations in their decisions and pure monotonic classifiers, whose predictions are always monotonic concerning the training and future examples. The latter is hard to achieve, particularly in real-life applications where the training data-sets are rarely purely monotonic. To be considered monotonic, all of the pairs of instances in a data-set must be monotonic [16]: $x_i \succeq x_j \ \rightarrow y_i \geq y_j, \forall_{i,j}$.

2.2 Monotonic *k*-Nearest Neighbors

MkNN [17] modifies the standard nearest neighbor rule of the well-known lazy learning method to avoid monotonic violations in its predictions. To do so, MkNN computes for each new example x_i the range $r_i = [y_{min}, y_{max}]$ of valid class labels, which satisfies the monotonic constraints. The lower-bound y_{min} of r_i is computed as the highest class label of all instances in the training set $\mathcal{D}$ below the example x_i. Analogously the upper-bound y_{max} is the minimum class label of the instances in $\mathcal{D}$ that are higher than x_i (see Eq. 1).

$$r_i = \begin{cases} y_{min} = \max\{y \mid (x,y) \in \mathcal{D} \wedge x_i \succeq x\} \\ y_{max} = \min\{y \mid (x,y) \in \mathcal{D} \wedge x \succeq x_i\} \end{cases} \tag{1}$$

Two different MkNN variants can be distinguished depending on how the neighbors are extracted for a new instance x_i. The *InRange* variant considers the k nearest examples x_j with their class labels y_j in the range $[y_{min}, y_{max}]$. The *OutRange* version extracts first the k nearest neighbors x_j and then, those neighbors outside of the range r_i are filtered out from the decision. If all of them are removed, a random label in r_i is chosen. As in the standard k-NN method, the majority class among the k neighbors is used as the predicted label.

MkNN is one of the methods that require monotonic data-sets to work properly [17]. Since, with monotonicity violations, the range r_i could not be correctly computed, a relabeling technique should be used to transform the non-monotonic training data into monotonic data. These techniques intend to identify and remove the monotonicity violations by making the fewest possible changes with minimum class difference. [17, 23, 24].

2.3 Fuzzy *k*-Nearest Neighbors

Fuzzy Sets [32] express the uncertainty of the example memberships to each class label. The memberships of the example x_i are represented as a degree of each class belonging $u_i = (u_{i1}, u_{i2}, \ldots, u_{ic})$, where $u_{il} \in [0,1]$ and $\sum_{l=1}^{c} u_{il} = 1$. Nowadays, development in fuzzy sets and classifiers is still an ongoing process [33].

Fuzzy k-Nearest Neighbors algorithms [26] incorporate fuzzy concepts into the classical k-NN decision rule to learn from fuzzy sets and produce fuzzy classification rules. Recently, different approaches have been proposed based on distinct fuzzy set extensions. However, the original Fuzzy k-NN [25] (FkNN) is still one of the best approaches [26]. Recent approaches provide for the optimization of parameters in FkNN [34].

For a given new instance x_i, Fuzzy k-NN [25] extracts its K nearest neighbors in the same manner as the standard k-NN. Then, its memberships for each class l are computed with the following expression:

$$u(x,l) = \frac{\sum_{j=1}^{K} u(x_j,l) * \frac{1}{||x-x_j||^{(m-1)}}}{\sum_{j=1}^{K} \frac{1}{||x-x_j||^{(m-1)}}} \tag{2}$$

As shown in Eq. 2, the membership $u(x_i,l) = u_{il}$ of sample x_i to class l is assigned with the product of the class membership $u(x_j,l)$ of the neighbors x_j and the inverse of their distances to x_i. The latter serves as a weight that biases towards the memberships of nearer samples. The parameter m determines the degree of influence of the neighbor distances. The recommended value $m = 2$ [25] makes the contributions of the neighboring samples reciprocal to their distances. A crisp class label for the example x_i can be decided as being the label l with the greatest membership degree u_{il}.

Facing a labeled training set, Fuzzy k-NN [25] brings it into a fuzzy set with sample memberships using the nearest neighbor rule. For each training sample x_i, k nearest neighbors are extracted using the leave-one-out scheme. Then memberships $u(x_i,l)$ for each class l are computed according to Eq. 3 with the number of neighbors nn_l found for each class l. This transformation has proven useful against noisy samples as the memberships lose influence as they are spread to the surrounding classes (not the assigned class).

$$u(x_i,l) = \begin{cases} 0.51 + 0.49 * (nn_l/k)\,, & \text{if} \quad y_i = l \\ 0.49 * (nn_l/k)\,, & \text{otherwise} \end{cases} \tag{3}$$

3 Monotonic Fuzzy *k*-Nearest Neighbors

In this section, we explain our approach in detail – MonFkNN and all its mechanisms that consider monotonicity constraints. In Subsection 3.1, we explain how MonFkNN gives a final class from class memberships in a more proper manner according to monotonicity. Subsection 3.2 is dedicated to the extraction of the class memberships from the training set and redesigned to reduce the impact of monotonic noise without the need for monotonic relabeling. In Subsection 3.3, the class membership aggregation built-in MonFkNN is explained and related to the robustness and flexibility of the classifier using its parameters. Finally, we discuss the algorithmic differences between our proposal and the original FkNN in Subsection 3.4.

3.1 From class memberships to the final class label

Since FkNN works with class memberships, a mechanism that respects monotonicity is needed to get a final class from a vector whose elements sum up to the value of one. The class with the greatest membership is the most common decision in multiple classifiers. The original Fuzzy k-NN gives their crisp predictions as the class label with the highest membership.

However, this might not be appropriate for scenarios with monotonic constraints. For example, let $x_i \leq x_j$ and their class memberships $u_i = (0.2, 0.2, 0.4, 0.2, 0.0)$ and $u_j = (0.0, 0.4, 0.3, 0.2, 0.1)$, then their final classes chosen with the highest membership break the monotonicity: $argmax(u_i) = l_3 > l_2 = argmax(u_j)$. Even though, the instance x_j has more weight values assigned to the higher labels than instance x_i. In fact, u_j weakly dominates u_i according to the *first degree stochastic dominance relation* (FSD) [35] since the x_i cumulative distribution function $U_i = (0.2, 0.4, 0.8, 1.0, 1.0)$ is greater, element by element, than $U_j = (0.0, 0.4, 0.7, 0.9, 1.0)$, that is, $u_i \preceq_{FSD} u_j \iff (\forall l \in \mathcal{Y})(U_i(l) \geq U_j(l))$. To make FSD applicable,

class membership vectors are normalized to sum up to the value of one and treated as probability mass functions. Therefore, a cumulative distribution function U can be computed for given normalized class memberships, where FSD is defined. This transformation can be done thanks to the order relation between classes in monotonic classification. FSD is useful for defining monotonicity constraints in probabilistic classifications [18, 36], with the expression $x_i \leq x_j \implies u_i \preceq_{FSD} u_j$.

Therefore, the function that transfers a membership vector to a class label must satisfy $u_i \preceq_{FSD} u_j \implies y_i \leq y_j$. Centrality measures, such as mean and median, have proven to be good solutions [35, 18]. Particularly, the median is applicable to ordinal problems. Following the traditional definition of median as the 50th percentile, the median is computed as the range $[l_m, l_M]$:

$$\begin{aligned} l_m &= \min\{l \in \mathcal{Y} \mid U\{X \leq l\} \geq 1/2\} \\ l_M &= \max\{l \in \mathcal{Y} \mid U\{X \geq l\} \geq 1/2\} \end{aligned} \tag{4}$$

where l is a class label of possible labels $\mathcal{Y}$, $U\{X \leq l\}$ is the cumulative membership/probability of belonging to a class smaller or equal to l and $U\{X \geq l\}$ is the analogous definition for a class greater or equal to l.

Going back to the previous example, the classes for x_i and x_j chosen by the median does not break monotonicity: $med(u_i) = med(u_j) = 3$. For $l_m \neq l_M$, any class label l which $l_m < l < l_M$ must have a membership $u(l) = 0$ and $U(l_m) = U(l_M) = 1/2$. For example, instance x_t with class memberships $u_t = (0.2, 0.3, 0, 0.3, 0.2)$ could be assigned to the classes $med(u_t) = [2, 4] = 3$.

3.2 Class memberships robust to monotonic noise

In this subsection, the class membership calculation redesigned to monotonic classification is explained. The objective of this first stage is to fix or reduce the influence of non-monotonic examples in the classification. Our method uses the robustness of the traditional Fuzzy k-NN within the knowledge of the monotonic relations between the neighbors. Algorithm 1 summarizes the procedure of obtaining robust noise class memberships for the training set.

First, we have to deal with the simplest monotonic violations, that is, instances with the same input values and different classes (Lines 2-13 of Algorithm 1). These mislabels frequently appear in traditional data-sets [16] of classification with monotonic constraints as these sets are rankings or evaluations made by different experts.

Therefore, MonFkNN first substitutes the replicas of any example x with one feature vector x and its memberships $u(x)$. The membership $u(x, l)$ of the instance x to the class l is computed with the frequency of duplicated examples x_j in the training set $\mathcal{D}$ belonging to class l ($y_j = l$), as shown in the following expression:

$$u(x, l) = \frac{|\{x_j \in \mathcal{D} | x_j = x \wedge y_j = l\}|}{|\{x_j \in \mathcal{D} | x_j = x\}|} \tag{5}$$

The class label of an instance x after the elimination of its replicas is obtained by the median of the resulting memberships, as shown in Line 13 of Algorithm 1. However, this vector will be used in the classification function with the membership aggregation as stated in the next subsection.

Then, MonFkNN estimates the memberships of the remaining instances, which corresponds to Lines 13 - 24 of Algorithm 1. This estimation is made using the information of the nearest neighbors of each instance. However, these nearest neighbors are extracted with a monotonic nearest neighbor rule (MkNN) instead of a traditional rule as we aim for memberships that respect monotonic constraints as much as possible. Algorithm 2 exemplifies the extraction of these monotonically constrained neighbors for a given instance x as in MkNN.

Algorithm 1 Training class memberships extraction

function TRAINCLASSMEMBERSHIPS($\{\mathcal{D}, y\}$ - Training data-set, k - Nearest neighbors considered, *RCr* - Real Class relevance)
  for $x_i \in \mathcal{D}$ **do**
    for $l \in \mathcal{Y}$ **do**
      if x_i duplicated-in $\mathcal{D}$ **then**
        Compute $u(x_i, l)$ with **expression 5**
      else
        $u(x_i, l) = \begin{cases} 1 & y_i = l \\ 0 & \end{cases}$
      end if
    end for
  end for
  $\mathcal{D}' = removeDuplicates(\mathcal{D})$
  for $x_i \in \mathcal{D}'$ **do**
    $y'_i = med(u_i)$ ▷ See **expression 4**
  end for
  for $x_i \in \mathcal{D}'$ **do**
    if x_i not-duplicated-in $\mathcal{D}$ **then**
      Compute range r_i with $(\mathcal{D}', y')$ and **expression 1**
      ▷ See **Algorithm 2**
      $nn =$ neighborsAsMkNN$(x_i, r_i, k, inRange, \mathcal{D}', y')$
      for $l \in \mathcal{Y}$ **do**
        Compute $u(x_i, l)$ with **expression 6**
      end for
    end if
  end for
  output: $(\mathcal{D}', u)$
end function

Algorithm 2 Monotonic nearest neighbor rule

function NEIGHBORSASMkNN(x - tested sample, r - range of valid classes, k - considered neighbors, $typeRange$ - inRange or outRange, $\{\mathcal{D}, y\}$ - Training data-set)
  initialize: $nn = \{\}$
  for $x_i \in \mathcal{D}$ **do**
    if $typeRange == outRange$ **or** $y_i \in r$ **then**
      if $Size(nn) < k$ **then**
        Insert x_i in nn
      else
        $x_{max} = \arg\max_{x_j \in nn} ||x - x_j||$
        if $||x - x_i|| < ||x - x_{max}||$ **then**
          Replace x_{max} by x_i in nn
        end if
      end if
    end if
  end for
  output: nn
end function

In this case, Algorithm 2 is configured as an *inRange* variant as pointed out in Line 16 of Algorithm 1. That is, the nearest neighbors of an example x_i are constrained to a range $r_i = [y_{min}, y_{max}]$ of possible classes (Line 17), which preserves the monotonicity of the data-set.

Once the nearest neighbors for each example x_i are obtained, the information of the neighbor classes is fused into x_i class memberships (Line 20). For an instance x_i, the membership $u(x_i, l)$ to class l is computed with the following expression:

$$u(x_i, l) = \begin{cases} \textit{RCr} + (nn_l/k) * (1 - \textit{RCr}) & \text{if} \quad y_i = l \\ (nn_l/k) * (1 - \textit{RCr}) \end{cases} \tag{6}$$

where nn_l is the number of nearest neighbors of the class l, k the total number of neighbors extracted for instance x_i and y_i is the original class label of the example x_i. *RCr* is a new parameter called *"Real Class relevance"*.

Apart from the use of the monotonic nearest neighbor rule, the inclusion of *RCr* is another main difference between our approach MonFkNN and the original Fuzzy k-NN. *RCr* can be seen as the minimum membership assigned to original class y_i of the instance x_i, in case there are no neighbors labeled with y_i. In FkNN, *RCr* corresponds to the value of 0.51, that is, every instance maintains its real class, even those noisy examples surrounded by other classes. By being a parameter, our method lets the user control the treatment of monotonic noise.

There are some values for *RCr* in the range $[0, 1]$ that have very interesting and distinct behaviors. In the case of a really noisy data-set where no labels can be trusted, *RCr* could be set to 0. This leaves all the responsibility to the calculation of the range of valid classes r_i and the nearest neighbors. In the presence of instances with the same input values and different classes, the user could choose only to treat them with $\textit{RCr} = 1$. Finally, if practitioners want to consider the originally labeled instances, we recommend assigning *RCr* to 0.5. This value ensures that the actual class is within the set of medians. In contrast to Fuzzy k-NN and its 0.51, if all neighbors belong to a same single class that is different to the current class, our method forces to choose in between these two classes. Usually, this last value ($\textit{RCr} = 0.5$) is a good trade-off, mainly stable and with better performance.

During this process, the impact of monotonic inconsistencies will be either reduced or fixed. The inconsistencies of instances with the same input vectors and different classes are completely fixed by being substituted by only a sample and class memberships with the information of their different classes. The mislabeled samples, i.e. noisy or non-monotonic examples, will have less influence towards their noisy class as they will be surrounded by more appropriated classes and their class memberships will be shared into classes in which they fit monotonically. This is the first mechanism of our method to alleviate the presence of monotonic violations, without the need for relabeling.

3.3 Flexible membership aggregation

After estimating the class memberships of every training instance, our algorithm is ready to predict new examples. This last phase has been designed to cover different needs of monotonic scenarios. In addition to the control of noise treatment, greater flexibility has been sought, allowing users to choose between more accurate or pure monotonic predictions.

Algorithm 3 represents in pseudo-code the whole prediction procedure of our proposal MonFkNN. Particularly, the prediction of a new instance x_i is detailed after having previously computed the monotonically-constrained class memberships of the training set as the previous Algorithm 1 is referred in Line 2.

As shown in Line 6, MonFkNN embeds another MkNN (Algorithm 2) to obtain the neighbors used in the membership aggregation and final prediction. This MkNN also has two versions, *inRange* and *outRange* versions. They are, however, substantially different when compared to original variants.

Algorithm 3 MonFkNN: Prediction stage

1: **function** MONFkNN(x_i - sample to predict, $\{\mathcal{D}, y\}$ - training data-set, k - neighbors considered for training class memberships, *RCr* - Real Class relevance, K - neighbors considered for prediction, $typeRange$ - inRange or outRange, *pOR′* - out-of-range penalty)
2: $(\mathcal{D}', u') =$ TrainClassMemberships$(\mathcal{D}, y, k,$ *RCr*$)$
3: Obtain medians y' of each sample in $\mathcal{D}'$ with u' and **expression 4**
4: Compute range r_i with **expression 1** and $(\mathcal{D}', y')$
5: ▷ See **Algorithm 2**
6: $nn =$ neighborsAsMkNN$(x_i, r_i, K, typeRange, \mathcal{D}', y')$
7: **for** $x_j \in nn$ **do**
8: **if** $typeRange == inRange$ **or** $y_j \in r_x$ **then**
9: $pOR_j = 1$
10: **else** ▷ Neighbors nn out of range r_x are penalized with *pOR′*
11: $pOR_j = pOR'$
12: **end if**
13: **end for**
14: Compute class memberships u_i of x_i with **expression 7**
15: **output:** $med(u_i)$
16: **end function**

The *inRange* alternative is based on the same idea of the original MkNN, where the neighbors of an example must belong to a set of monotonically valid classes. However, this range of classes is obtained using the medians acquired from the class memberships of the training instances constrained by monotonicity, as seen in Line 3 and Line 4. This breakthrough improves our method by increasing monotonic noise robustness. Firstly, an *inRange* nearest neighbor rule removes monotonic inconsistencies in the known data-set as previously shown in Algorithm 1. Then, the second MkNN uses this fixed training set $(\mathcal{D}', y')$ to give monotonic predictions as seen in Algorithm 3.

The *outRange* version of our method is completely different from the previous *outRange* rule. It has been designed with the intention of prioritizing to some extent the predictive ability of the classifier over monotonicity. With this purpose in mind, our method considers any example as a valid neighbor regardless of its class label. In contrast to the original model, no filtering or removal of neighbors outside the valid range is performed. However, their relevance in the membership aggregation can be reduced if needed, thanks to a penalty factor introduced in the aggregation expression.

Then, for a new example x, its nearest neighbors are obtained according to the chosen variant. Their memberships are aggregated with the original FkNN formula with the addition of the penalty factor for the *outRange* version. The following expression shows how this parameter is integrated:

$$u(x,l) = \frac{\sum_{j=1}^{K} u(x_j,l) * \frac{pOR_j}{||x - x_j||^{(m-1)}}}{\sum_{j=1}^{K} \frac{pOR_j}{||x - x_j||^{(m-1)}}} \tag{7}$$

As previously, the membership $u(x,l)$ of the new sample x to the class label l is the result of the sum of the class memberships $u(x_j,l)$ of the neighbors x_j inversely weighted with their distance to x. In the *outRange* version of our method, there is another weighting factor in the contribution to the final memberships, the parameter referred to as *"penalty of outRange"* (*pOR*). The factor pOR_j is applicable only if the class y_j of the neighbor

FkNN	**MonFkNN**
No special treatment of duplicates.	Duplicates are reduced to a single instance.
Standard nearest neighbor rules.	Monotonic nearest neighbor rules.
Standard training membership extraction.	Monotonically constrained class memberships.
Conservation of original classes in the training class membership extraction.	Loss of influence of original class towards monotonicity with *RCr* $<= 0.5$.
Value 0.51 in Eq. 3	Parameter *RCr* in Eq. 6
Standard class membership aggregation.	Monotonically constrained membership aggregation.
No penalty to any neighbors in Eq. 2	*pOR* Penalty to out-of-range neighbors in Eq. 7.
Final class as highest membership	Final class as median of class memberships

Table 1: Summary of algorithmic differences between standard FkNN and MonFkNN.

x_j is not in the valid class range r_x of x as exemplified in Lines 7 to 13 . It can be configured with continuous values from 0 to 1. When it is assigned to 1, no penalty is applied. The value 0 means a full penalty, that is, neighbors with invalid classes will not participate in the membership aggregation. For all practical purposes, this last behavior is equivalent to the *outRange* MkNN. We recommend using 0.5 since it is a good balance between reducing their relevance and considering them in the decision.

Finally, the class prediction of the new example x is the median of the resulting normalized class memberships.

As presented, MonFkNN has been developed to be robust to monotonic noise and versatile in many scenarios. The two versions *inRange* and *outRange* with the parameter *pOR* and the previously mentioned *RCr* help to tune the algorithm according to the necessities of different kinds of problems.

Among the possibilities that offer these parameters, we have named two configurations with very distinctive behaviors: Pure Monotonic (MonFkNN-PM or PM) and Approximate Monotonic (MonFkNN-AM or AM) Fuzzy k-NN. The Pure Monotonic configuration corresponds to a value of 0.5 for the *RCr* parameter and the use of *inRange* rule to obtain the memberships of new instances. This approach aims to give predictions with the minimum violations of monotonicity. In every part of the algorithm, it prioritizes monotonicity over very accurate predictions.

MonFkNN-AM prioritizes the predictive ability and relaxes the monotonic constraints. The memberships of the training set are obtained by the treatment of samples with the same feature values and different classes. Those unique examples will have a membership of 1 to the actual class and 0 for the rest. This behavior is achieved with *RCr* $= 1$. Then, as we are looking for more accurate predictions, all instances can be considered to be valid neighbors and to contribute to the final aggregation. Those instances with invalid class labels, however, will contribute with only half of their class memberships (*pOR* $= 0.5$).

Our proposal MonFkNN is available at the GitHub Repository[1].

3.4 Differences between standard F*k*NN and MonF*k*NN: Theoretical discussion

Standard FkNN and MonFkNN have a similar mathematical formulation. In other words, the expressions used by MonFkNN in the training class membership extraction (Eq. 6) and in the membership aggregation (Eq. 7) are the same as those used by FkNN (Eq. 3 and Eq. 2), for *RCr* $= 0.51$ and *pOR* $= 1$. The global behavior of our method is however still completely different to the standard FkNN, due to significant algorithmic differences. Table 1 summarizes the main differences between standard FkNN and our proposal MonFkNN.

Each of the differences mentioned in Table 1 is described and explained below:

[1] `https://github.com/sergiogvz/MonFkNN`

- The data-set used to compute the training class memberships is modified before applying the neighborhood rule. The inconsistencies of duplicates are eliminated and reduced to a single instance. The classes of the resultant instances are assigned to the median calculated with the frequency of the appearance of duplicates for each class. This procedure could not even be considered in standard classification, where there is no ordering relationship between classes.

- The neighborhood considered for each training instance is constrained to the monotonicity of the data-set. Then, their resultant class memberships are also monotonically constrained. These adaptations completely modify the neighbors contributing in Eq. 3 and the whole procedure. In addition, the value of 0.51 for *RCr* is discouraged in MonFkNN in favor of 0.5 due to its contribution to the medians of the samples, above-mentioned in Section 3.2.

- The original FkNN and MonFkNN also share the same membership aggregation, i.e. their expressions (Eq. 3 and Eq. 6) are the same for *InRange* and *outRange* (with *pOR* $= 1$) versions of MonFkNN. However, their behavior and their predictions are completely different, due to the differences in the nearest neighbor rule, in the training set and class memberships used in the aggregation procedure. As previously explained, the training class memberships extraction of MonFkNN modifies the training set fixing some monotonic inconsistencies. Duplicates are removed and some training samples might change their classes to preserve the monotonicity of the data-set.

- In MonFkNN, the classes of the training samples determine the monotonically valid classes of the unlabeled instances. Thus, training samples with classes not valid for an instance x will be discarded from the neighborhood (*inRange* version) or penalized with the parameter *pOR* (*outRange* version). The configuration *outRange* version with *pOR* $= 1$ is also discouraged since the final purpose of MonFkNN is to take monotonic constraints into consideration, at least to some extent.

- These mechanics acquire different neighbors to those drawn by FkNN for the same test sample, that is, different class memberships and prediction. Finally, the median as the final class of the class membership vector already implies a significant change in the behavior of the method.

These differences between our proposal and the traditional FkNN are clearly supported by the experiments carried out in Section 5.1.

4 Experimental framework

This section is devoted to introducing the experimental framework used in the different empirical studies of the paper. In our experiments, we have included 12 data-sets of a good variety of problems presenting real monotonic constraints. The data-sets can be seen in Table 2, where the number of instances, attributes and classes are detailed for each data-set in the column Ins., At. and Cl., respectively. The column At. Directions indicates the monotonic direction of the relationship between each attribute and the class: direct (+) or inverse monotony (-). This information is extracted from the description of the problems involving the data-sets. The column Comparable Pairs shows the percentage of pairs of comparable samples over the total number of pairs. Two instances x_i and x_j are comparable if their inputs have an order relation, i.e. $x_i \succeq x_j$ or $x_i \preceq x_j$. On average, one-third of the total number of pairs of these data-sets are comparable and potential violations of monotonicity in the classification process. This quite large amount cannot be neglected.

These data-sets are chosen as the most frequently used in the monotonic classification literature. The classical monotonic set *ERA*, *ESL*, *LEV* and *SWD* [16] are also considered in the study. Additionally, the data-set *artiset* is employed for a comparative study on monotonic noise robustness of MonFkNN (see Subsection 5.4). *Artiset* is an artificial data-set with two attributes (x_1, x_2) and nCl number of classes. For attributes $x_1, x_2 \in [0, 1]$, the class is computed as the truncation of the outcome of the following formula:

Table 2: Description of the 12 data-sets used.

Data-set	Ins.	At.	Cl.	At. Directions	Comparable Pairs
artiset	1000	2	10	All direct directions	49.79%
balance	625	4	3	{-, -, +, +}	25.64%
bostonhousing4cl	506	13	4	{-, +, -, +, -, +, -, +, -, -, -, +, -}	14.85%
car	1728	6	4	All direct directions	14.36%
ERA	1000	4	9	All direct directions	16.77%
ESL	488	4	9	All direct directions	70.65%
LEV	1000	4	5	All direct directions	24.08%
machineCPU	209	6	4	{-, +, +, +, +, +}	49.53%
qualitative_bankruptcy	250	6	2	All inverse directions	43.77%
SWD	1000	10	4	All direct directions	12.62%
windsorhousing	546	11	2	All direct directions	27.07%
wisconsin	683	9	2	All direct directions	58.04%

$$f(x_1, x_2) = (x_1 + \frac{x_2^2 - x_1^2}{2}) * nCl$$

A 10-fold cross-validation scheme (10-fcv) is carried out to run the different classifiers over these sets. Their partitions have been extracted from the KEEL repository [37].

The classifiers involved in the empirical comparisons are:

- Monotonic k-NN (MkNN) [17]
- Ordinal Stochastic Dominance Learning (OSDL) [18]
- Ordinal Learning Module (OLM) [16]
- Monotonic Multi-Layer Perceptron network (MonMLP) [38]
- C4.5 decision tree for monotonic induction (MID) [4]
- Rank Discrimination Measure Tree (RDMT) [8]
- Partially Monotonic Decision Tree (PMDT) [9]

Table 3 details the parameters chosen according to the recommendations found in the original papers. As a requirement of MkNN, a relabeling technique [24] is applied to training data-sets before fitting MkNN. On the contrary, the rest of the algorithms, including MonFkNN, do not need this relabeling procedure. Therefore, all the results shown for MkNN are obtained with relabeled training sets, while other methods are trained with the original training data-sets.

In order to evaluate the classifiers' proficiency, we have employed three measures of different aspects of their performance: predictive capability, error cost and monotonicity. Standard accuracy is used to evaluate the predictive capability of the models. Mean Absolute Error (MAE) is computed as the average differences of the true instance ranks and the predicted ranks. To evaluate monotonicity, Non-Monotonic Index (NMI) [1] measures the ratio of pairs of samples (NMP) that break monotonicity among the total of pairs, with N being the number of samples in the data-set:

$$NMI = \frac{NMP}{N^2 - N}$$

Table 3: Parameters considered for the algorithms compared.

Algorithm	Parameters
MkNN [17]	k = 5, distance = euclidean, neighborsType = inRange
OSDL [18]	balanced = No, classificationType = median, lowerBound = 0, upperBound = 1 tuneInterpolationParameter = No, weighted = No, interpolationStepSize = 10, interpolationParameter = 0.5
OLM [16]	modeResolution = conservative modeClassification = conservative
MonMLP [38]	default parameters, hidden1 = 8 iter.max = 1000, monotonic = all att
MID [4]	R = 1, confidence = 0.25, items per leaf = 2
RDMT [8]	H = Pessimistic rank discrimination measure, measureThreshold = 0, items per leaf = 2
PMDT [9]	threshold θ = 0, items per leaf = 2
FkNN [25]	k = 5, K = 9, distance = euclidean
MonFkNN	k = 5, K = 9, distance = euclidean
Pure Monotonic	*RCr* = 0.5, neighborsType = inRange
Approximate Monotonic	*RCr* $= 1$, neighborsType = outRange, *pOR* $= 0.5$

These measures are computed over a set merged from the test predictions of 10-fcv sets for each data-set and classifier. Finally, the Wilcoxon statistical test, Friedman rank test [27, 28] with Holm post-hoc procedure [39] and Bayesian Sign test [29] are used to validate the results of the empirical comparisons. In the Bayesian Sign test, a distribution of the differences of the results achieved by methods A and B is computed thanks to the Dirichlet Process. This distribution is shown in a graphical space divided into 3 regions: left, rope and right. The location of the majority of distribution in these sectors indicates the final decision of the pairwise Bayesian non-parametric sign test: superiority of algorithm B (left sector), statistical equivalence (rope sector) and superiority of algorithm A (right sector). For the accuracy and MAE results, we have set the inferior and superior limit of the rope region to -0.01 and 0.01, respectively. However, we have adjusted the limits to -0.0001 and 0.0001 for NMI since NMI values tend to be significantly smaller due to the big difference between the numbers of comparable instance pairs and all possible pairs. The R package rNPBST [40] has been used to extract the graphical representations of the Bayesian Sign tests analyzed in the following empirical studies.

5 Results and analysis

This section presents the results of the empirical studies and their analyses. First, the two configurations of MonFkNN are compared in Subsection 5.1, showing their different strengths. Then, our proposal is compared to methods from the state-of-the-art in terms of prediction capability and monotonicity in Subsection 5.3 and Subsection 5.3, respectively. In Subsection 5.4, the last experiment tests the noise robustness of MonFkNN in contrast to MkNN.

5.1 Evaluation of Monotonic Fuzzy *k*-NN approaches. Pure Monotonic vs Approximate Monotonic

A comparison between the Pure and Approximate Monotonic version of MonFkNN stresses the different behaviors and aspects of their performance. Additionally, the performance differences between the original FkNN and MonFkNN are analyzed. Table 4 shows the results of FkNN and the two configurations of our

Table 4: Results for the Pure and Approximate Monotonic Fuzzy k-NN

	Accuracy			**MAE**			**NMI**		
		MonFkNN			**MonFkNN**			**MonFkNN**	
	FkNN	**PM**	**AM**	**FkNN**	**PM**	**AM**	**FkNN**	**PM**	**AM**
artiset	0.9339	0.9309	**0.9349**	0.0661	0.0691	**0.0651**	**0.0000**	**0.0000**	**0.0000**
balance	0.8896	**0.9307**	0.9008	0.1424	**0.0853**	0.1168	**0.0000**	**0.0000**	0.0001
bostonhousing4cl	**0.7174**	0.6561	0.7134	**0.3241**	0.3972	0.3261	0.0004	**0.0000**	0.0001
car	0.9311	0.9740	**0.9834**	0.0793	0.0295	**0.0195**	0.0002	**0.0000**	**0.0000**
ERA	0.1730	0.2420	**0.2430**	1.6660	**1.2813**	1.2993	0.0141	**0.0052**	**0.0052**
ESL	0.6783	0.7036	**0.7131**	0.3484	0.3149	**0.3053**	0.0014	0.0004	**0.0003**
LEV	0.6020	**0.6377**	0.6110	0.4330	**0.3927**	0.4223	0.0021	**0.0004**	0.0009
machineCPU	0.6699	**0.7033**	0.6699	0.3589	**0.3158**	0.3493	0.0058	**0.0002**	0.0017
qualitative_bankruptcy	**0.9960**	**0.9960**	**0.9960**	**0.0040**	**0.0040**	**0.0040**	**0.0000**	**0.0000**	**0.0000**
SWD	0.5350	0.5807	**0.5833**	0.5180	**0.4370**	0.4380	0.0027	0.0007	**0.0003**
windsorhousing	**0.7857**	0.7576	0.7839	**0.2143**	0.2424	0.2161	0.0062	**0.0005**	0.0051
wisconsin	**0.9678**	0.9653	0.9663	**0.0322**	0.0347	0.0337	**0.0000**	**0.0000**	**0.0000**
Avg:	0.7400	0.7565	**0.7583**	0.3489	0.3003	**0.2996**	0.0027	**0.0006**	0.0012

proposal MonFkNN in terms of Accuracy, MAE and NMI. Bold-face font indicates the best results obtained for each data-set and metric.

In Table 4, the differences between both approaches (PM and AM) can be seen clearly. Just as they were designed, MonFkNN-AM has better accuracy on average, while MonFkNN-PM achieves monotonically reliable predictions. Both have good, stable results in terms of MAE, with AM coming out slightly on top.

AM configuration obtains the most accurate predictions for more than 50% of the benchmark used. On the other hand, the PM model achieves better results according to monotonicity in 10 of the 12 data-sets used, with large differences in *Windsorhousing* and *MachineCPU* problems. When compared with FkNN, MonFkNN greatly improves the performance of the original algorithm. Both versions of MonFkNN (PM and AM) are better on average for each of the three different measures. Particularly, there is an overwhelmingly large difference between FkNN and MonFkNN-PM in terms of monotonicity. FkNN is better only for 3 data-sets when taking just accuracy and MAE into consideration. However, it does not outperform the monotonic predictions of MonFkNN.

This improvement is also reflected in the Wilcoxon statistical test applied to the results achieved using these methods. Table 5 presents the hypothesis of equivalence of the Wilcoxon test for $\alpha = 0.1$ on the pairwise comparison of FkNN (1) and our two proposals (MonFkNN-PM (2) and MonFkNN-AM (3)). As shown in Table 5, MonFkNN-AM is statistically better than FkNN in terms of accuracy and MAE with p-Values under 0.1. Considering monotonicity, MonFkNN-PM and -AM statistically outperform FkNN with very low p-Values. Overall, MonFkNN is clearly superior to FkNN in scenarios with monotonic constraints.

The reasons for these differences in results are clear and mainly due to their algorithmic differences. MonFkNN has learning procedures with notions in the order relation of classes and the monotonic constraints between input and output, which explain an overall better performance in terms of MAE and NMI. Additionally, MonFkNN has a greater awareness and treatment of noisy data, which helps obtain better accuracy.

Since monotonicity is usually prioritized in classification with monotonic constraints, we will use MonFkNN-PM in the following empirical studies.

Table 5: Wilcoxon test applied to the results obtained by Fuzzy k-NN algorithms: FkNN (1), MonFkNN-PM (2) and MonFkNN-AM (3)

Comparison	R^+	R^-	**Hypothesis** ($\alpha = 0.1$)	p-**Value**
Accuracy:				
(2) vs. (1)	49.0	17.0	Not Rejected	0.1748
(3) vs. (1)	61.5	16.5	**Rejected**	0.0847
MAE:				
(2) vs. (1)	51.0	15.0	Not Rejected	0.1230
(3) vs. (1)	57.0	9.0	**Rejected**	0.0322
NMI:				
(2) vs. (1)	76.5	1.50	**Rejected**	0.0012
(3) vs. (1)	72.5	5.50	**Rejected**	0.0059

Table 6: Results in terms of Accuracy achieved by the tested algorithms

	MonFkNN-PM	**MkNN**	**OSDL**	**OLM**	**MonMLP**	**MID**	**RDMT**	**PMDT**
artiset	0.9309	0.9199	0.1952	0.7948	**0.9463**	0.7237	0.8749	0.8539
balance	**0.9307**	0.8624	0.6352	0.8320	0.9131	0.7808	0.7216	0.7792
bostonhousing4cl	0.6561	0.6126	0.2787	0.5277	0.3979	0.6739	0.6304	**0.6739**
car	**0.9740**	0.9711	0.9549	0.9543	0.8474	0.8027	0.7297	0.9682
ERA	0.2420	0.1990	0.2320	0.1690	0.2380	**0.2760**	0.2390	0.2430
ESL	0.7036	0.6332	0.6721	0.5738	**0.7234**	0.6414	0.5635	0.6598
LEV	0.6377	0.4630	**0.6400**	0.4250	0.6167	0.6070	0.5210	0.6370
machineCPU	**0.7033**	0.6890	0.2919	0.6746	0.6730	0.6220	0.6555	0.6507
qualitative_bankruptcy	**0.9960**	**0.9960**	0.9160	0.9800	0.6427	0.9840	0.9840	0.9920
SWD	0.5807	0.5200	**0.5840**	0.4160	0.5063	0.5540	0.5180	0.5830
windsorhousing	0.7576	0.5861	0.4927	0.7564	0.7790	**0.8205**	0.8022	0.7564
wisconsin	**0.9653**	0.9649	0.9590	0.8873	0.8604	0.9517	0.9502	0.9561
Avg:	**0.7565**	0.7014	0.5710	0.6659	0.6787	0.7031	0.6825	0.7294

5.2 Comparison with the State-of-the-Art: Prediction capabilities

Here we evaluate the performance of our approach in comparison to methods from the state-of-the-art of monotonic classification. In this comparison, we look for a balance between accurate and monotonic predictions. Therefore, we compare the results obtained in terms of the selected metrics independently. Then, we draw our conclusions and check if our approach behaves well in the different aspects of classification with monotonic constraints.

First, we evaluate the prediction capability of our method. Table 6 gathers the accuracy results for the different data-sets obtained by the tested algorithms. With these outcomes, MonFkNN-PM performs overwhelmingly better than the rest in terms of accuracy. Our approach achieves the most accurate predictions on average with a wide margin. Additionally, it obtains the best results for 5 data-sets, with particularly remarkable cases, such as *balance*. PMDT is the second best method in terms of accuracy and it is the only method that come close to the performance of MonFkNN-PM. However, it obtains the overall best results for one data-set only (*bostonhousing*).

Table 7: Holm test applied to the Accuracy results among the tested algorithms

Control Method: MonFkNN-PM (2.04)				
i	**Algorithm (Rank)**	**Z**	p**-Value**	**Hypothesis** ($\alpha = 0.05$)
7	OLM (6.13)	4.083	0.00004	**Rejected**
6	OSDL (5.42)	3.375	0.00073	**Rejected**
5	RDMT (5.38)	3.333	0.00085	**Rejected**
4	MonMLP (4.67)	2.625	0.00866	**Rejected**
3	MID (4.42)	2.375	0.01754	**Rejected**
2	MkNN (4.21)	2.167	0.03026	**Rejected**
1	PMDT (3.75)	1.708	0.08757	Not Rejected

As mentioned before, we have used the Friedman rank test and the Bayesian Sign test to corroborate the significance of the differences of our approach and the selected methods. Table 7 includes the outcome of the Friedman rank and Holm tests in relation to the obtained Accuracy results. MonFkNN-PM is ranked first with a high ranking value compared to others. All the hypotheses of equivalence are rejected with small p-values with the exception of PMDT, which would be rejected for $\alpha = 0.1$. The distance between the ranks of MonFkNN-PM and PMDT is still quite large.

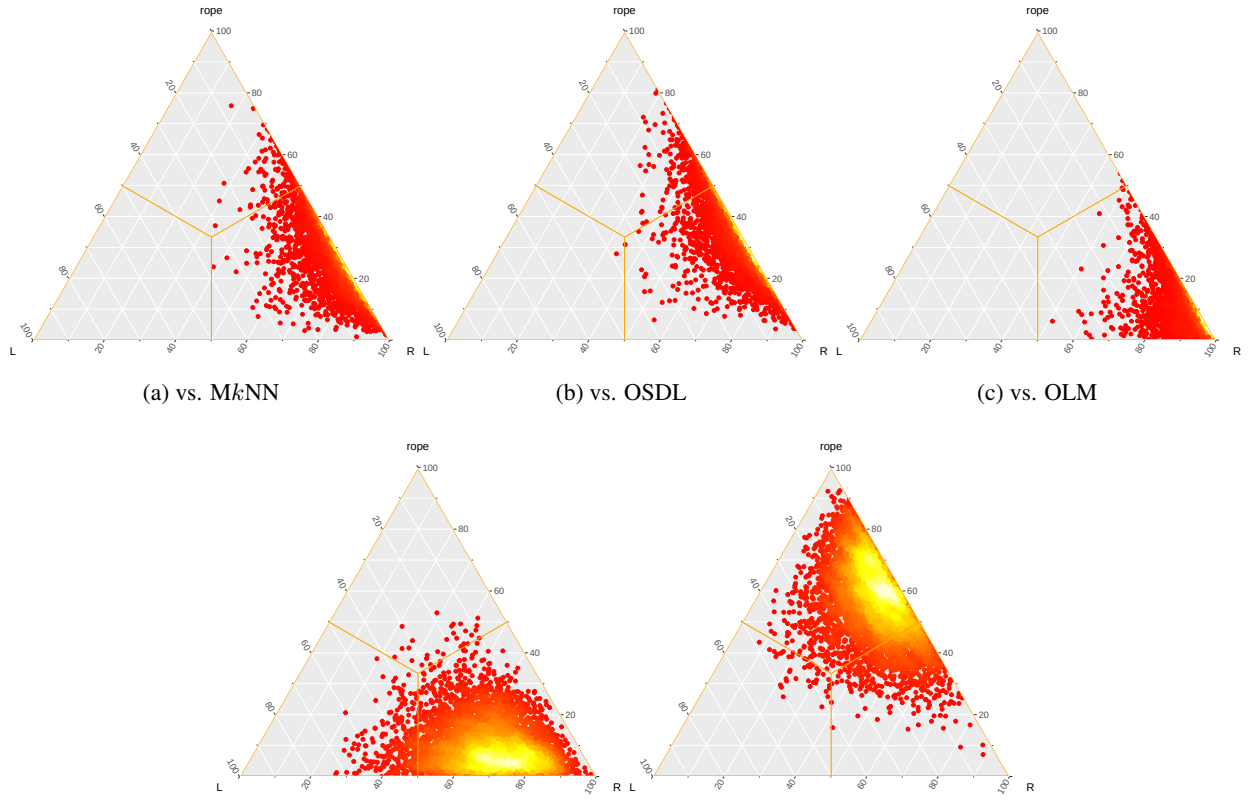
Figure 1 graphically represents the difference between MonFkNN-PM and other methods and its statistical significance in terms of accuracy. In order to save space and avoid plotting 7 heat-maps for each metric, we have only included PMDT, as it is the best and most recent algorithm among the monotonic decision trees [9]. As mentioned before, the position of the majority of the distribution in these maps determines the decision of the test: the right sector means the statistical superiority of MonFkNN-PM over the compared method, the rope sector is the statistical equivalency and the left side indicates the superiority of the other algorithm.

These heat-maps clearly indicate the significant superiority of MonFkNN-PM over all methods except PMDT as the computed distributions are always located in the right region. The most significant outcome is the comparison with OLM (Figure 1c), even though it does not obtain the worst results. For MkNN (Figure 1a) and OSDL (Figure 1b), there are a few cases where their performances are statically equivalent to MonFkNN-PM. On the contrary, MonMLP is significantly more accurate in a few data-sets, although the MonFkNN-PM is clearly superior (Figure 1d). Considering the comparison with PMDT (Figure 1d), the majority of the distribution is located in the statistical equivalence. However, it is still shifted to the right with a large number of points, indicating a better performance for MonFkNN-PM. Almost none support the performance of PMDT.

Error costs could be essential for monotonic ranking problems. Table 8 shows the error in the form of MAE made by the evaluated classifiers. As was the case in accuracy performance, MonFkNN-PM clearly performs better than the rest, with the smallest error on average and for 4 of the data-sets. It also achieves similar results in problems where other algorithms come out on top, such as *LEV* or *wisconsin*.

Table 9 shows the ranking of the methods and p-values obtained with the post hoc test for the MAE comparison. As in the accuracy tests, our proposal is once again ranked as the best method with a solid statistical significance as compared to almost all algorithms. PMDT still achieves similar results to MonFkNN-PM with a p-value that does not reject the hypothesis for $\alpha = 0.05$, but does for $\alpha = 0.1$. In this case, the p-value of PMDT is smaller and its rank difference with our proposal is larger than that obtained in terms of accuracy.

Figure 2 shows the Bayesian Sign test on pairwise comparison with our method according to MAE. As shown by the distributions in the right part of the majority of the figures, MonFkNN-PM is definitely better when considering error costs. This is more statistically significant as compared to OLM (Figure 2c), where nearly the entire distribution is in the right region. MonFkNN, MkNN and OSDL share some good results, but these last



(a) vs. MkNN (b) vs. OSDL (c) vs. OLM

(d) vs. MonMLP (e) vs. PMDT

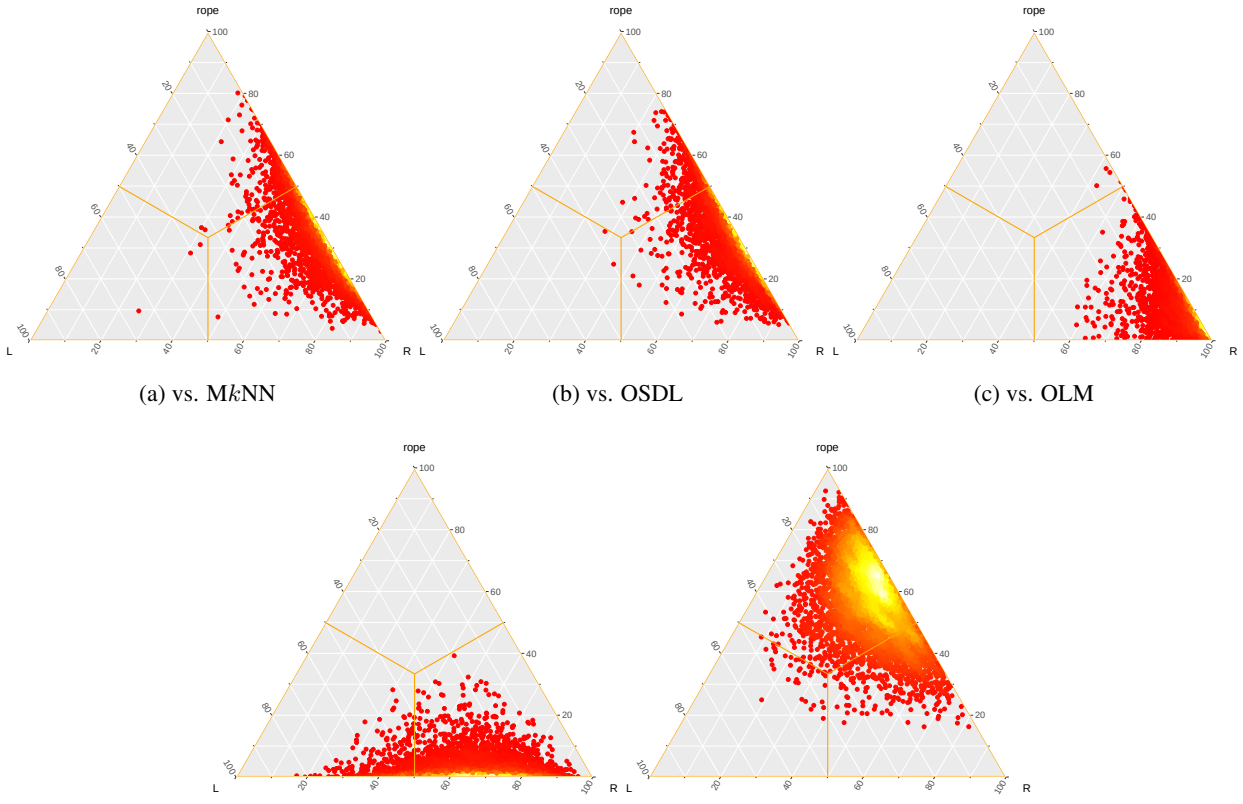
Figure 1: Bayesian Sign Test heat-map for MonFkNN-PM vs. the rest in terms of accuracy.

Table 8: Results in terms of MAE achieved by the tested algorithms

	MonFkNN-PM	**MkNN**	**OSDL**	**OLM**	**MonMLP**	**MID**	**RDMT**	**PMDT**
artiset	0.0691	0.0771	1.6897	0.2082	**0.0537**	0.3123	0.1251	0.1471
balance	**0.0853**	0.1504	0.4912	0.1920	0.0992	0.3360	0.3840	0.2560
bostonhousing4cl	0.3972	0.4901	0.9368	0.5988	0.7655	0.3893	0.4249	**0.3676**
car	**0.0295**	0.0359	0.0475	0.0538	0.1599	0.2506	0.3079	0.0365
ERA	1.2813	1.4270	1.2850	2.1500	**1.2317**	1.2970	1.3060	1.2870
ESL	0.3149	0.3791	0.3607	0.4734	**0.2910**	0.3934	0.4918	0.3750
LEV	0.3927	0.5740	**0.3920**	0.6680	0.4170	0.4290	0.5430	0.3940
machineCPU	**0.3158**	0.3301	0.9043	0.3589	0.3413	0.4211	0.3589	0.3732
qualitative_bankruptcy	**0.0040**	**0.0040**	0.0840	0.0200	0.3573	0.0160	0.0160	0.0080
SWD	0.4370	0.4840	0.4370	0.7630	0.5167	0.4750	0.4990	**0.4340**
windsorhousing	0.2424	0.4304	0.5073	0.2436	0.2210	**0.1795**	0.1978	0.2436
wisconsin	0.0347	**0.0337**	0.0410	0.1127	0.1396	0.0483	0.0498	0.0439
Avg:	**0.3003**	0.3680	0.5980	0.4869	0.3828	0.3790	0.3920	0.3305

Table 9: Holm test applied to the MAE results among the tested algorithms

Control Method: MonFkNN-PM (2.00)				
i	**Algorithm (Rank)**	**Z**	**p-Value**	**Hypothesis** ($\alpha = 0.05$)
7	OLM (6.17)	4.167	0.00003	**Rejected**
6	RDMT (5.54)	3.542	0.00040	**Rejected**
5	OSDL (5.29)	3.292	0.00099	**Rejected**
4	MID (4.96)	2.958	0.00309	**Rejected**
3	MonMLP (4.25)	2.250	0.02445	**Rejected**
2	MkNN (4.04)	2.042	0.04119	**Rejected**
1	PMDT (3.75)	1.750	0.08011	Not Rejected



(a) vs. MkNN (b) vs. OSDL (c) vs. OLM

(d) vs. MonMLP (e) vs. PMDT

Figure 2: Bayesian Sign Test heat-map for MonFkNN-PM vs. the rest in terms of MAE.

two are not statistically better than the former in any circumstance as seen in Figure 2a and Figure 2b. As we have also seen in the accuracy comparison, Figure 2d points out the statistical superiority of MonFkNN-PM over MonMLP, but the latter has a better MAE in some cases. Given Figure 2e, MonFkNN-PM and PMDT can be considered to be statistically the same in terms of error costs. However, MonFkNN-PM performs better statistically than PMDT in an important part of the benchmark, as a fragment of the distribution is located on the right side and almost none are found on the left.

Table 10: Results in terms of NMI achieved by the tested algorithms

	MonFkNN-PM	**MkNN**	**OSDL**	**OLM**	**MonMLP**	**MID**	**RDMT**	**PMDT**
artiset	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0039	**0.0000**	0.0001
balance	**0.0000**	0.0001	0.0006	**0.0000**	**0.0000**	0.0017	0.0029	0.0010
bostonhousing4cl	**0.0000**	**0.0000**	**0.0000**	0.0003	0.0007	0.0022	0.0010	0.0010
car	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0001	0.0046	0.0002	**0.0000**
ERA	0.0052	0.0056	0.0049	0.0063	**0.0026**	0.0082	0.0085	0.0058
ESL	0.0004	0.0012	0.0006	0.0025	**0.0003**	0.0021	0.0066	0.0032
LEV	**0.0004**	0.0010	**0.0004**	0.0043	0.0008	0.0018	0.0086	0.0006
machineCPU	0.0002	**0.0000**	**0.0000**	0.0014	0.0001	0.0037	0.0047	0.0028
qualitative_bankruptcy	**0.0000**	**0.0000**	0.0003	**0.0000**	0.0079	0.0002	**0.0000**	**0.0000**
SWD	0.0007	0.0005	0.0009	0.0015	0.0004	0.0020	**0.0000**	0.0010
windsorhousing	0.0005	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0030	0.0002	0.0059
wisconsin	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0001	**0.0000**
Avg:	**0.0006**	0.0007	**0.0006**	0.0014	0.0011	0.0028	0.0027	0.0018

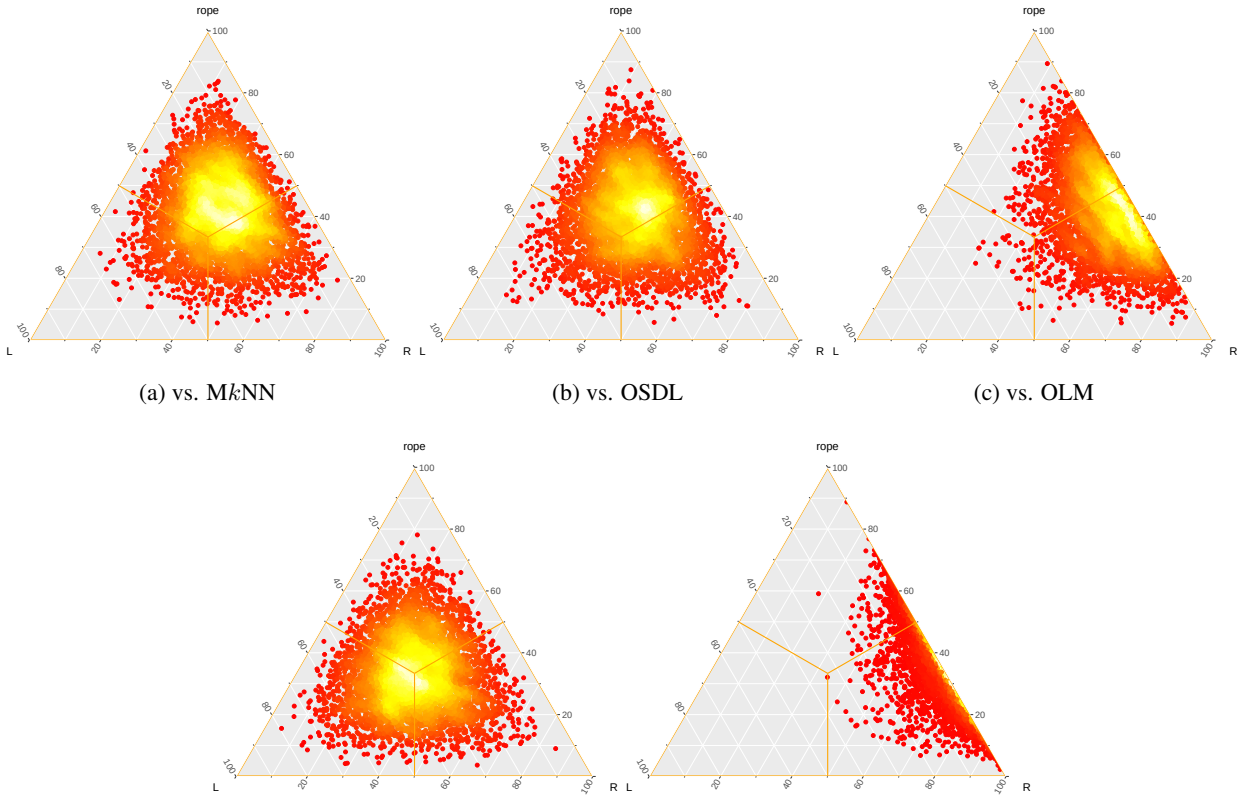
Table 11: Holm test applied to the NMI results among the tested algorithms

Control Method: MonFkNN-PM (2.9583)				
i	**Algorithm (Rank)**	**Z**	**p-Value**	**Hypothesis ($\alpha = 0.05$)**
7	MID (7.00)	4.042	0.00005	**Rejected**
6	RDMT (6.33)	3.375	0.00074	**Rejected**
5	PMDT (5.75)	2.792	0.00524	**Rejected**
4	OLM (4.13)	1.167	0.24335	Not Rejected
3	MonMLP (3.63)	0.667	0.50499	Not Rejected
2	MkNN (3.13)	0.167	0.86763	Not Rejected
1	OSDL (3.08)	0.125	0.90052	Not Rejected

5.3 Comparison with the State-of-the-Art: Monotonicity

Now we will analyze the performance according to the monotonicity of our proposal compared to methods chosen from the state-of-the-art. Table 10 shows the NMI results achieved by the selected models. In this case, the competition is close. Monotonic decision trees (MID, RDMT, and PMDT) clearly obtain less monotonic predictions. MID has the worst behavior considering only monotonicity and PMDT is the most monotonic decision tree classifier. OLM and MonMLP are slightly better than PMDT, but they still do not come close to the best methods. MonFkNN-PM, MkNN, and OSDL perform similarly. MonFkNN-PM and OSDL are slightly better on average. It is worth mentioning the existence of simpler data-sets, such as *artiset* and *wisconsin*, in relation to monotonicity as almost every algorithm accomplishes the same good results. The best results for the more complex sets are shared by the different methods.

Table 11 summarizes the comparison according to monotonicity with the Friedman statistical test results. In this case, MonFkNN-PM is barely selected as the control method. For half of the benchmark (OSDL, MkNN, MonMLP and OLM), the hypotheses of equivalence are not rejected for $\alpha = 0.05$. On the contrary, all monotonic decision trees are statistically worse than MonFkNN-PM by a wide margin. The best monotonic decision tree (PMDT) does not reach good performance in terms of monotonicity of the best algorithms. This is probably due to the greedy construction of monotonic constraints into the tree.



(a) vs. MkNN (b) vs. OSDL (c) vs. OLM

(d) vs. MonMLP (e) vs. PMDT

Figure 3: Bayesian Sign Test heat-map for MonFkNN-PM vs. the rest in terms of NMI.

In Figure 3, the statistical comparisons of the NMI results are represented with Bayesian Sign Test heat-maps. These plots show similar conclusions extracted from the previous table with NMI results. MonFkNN is significantly superior to PMDT (Figure 3e). In Figure 3c, the right-shifted distribution points out that MonFkNN-PM is better than OLM. Although they share a part of the distribution in the rope section, OLM has too few individuals in its left section (Figure 3c). When compared with MkNN (Figure 3a), OSDL (Figure 3b) and MonMLP (Figure 3d), big parts of the distributions are located in all the decision sectors. Even though their distributions are slightly shifted to the right (Figure 3a and Figure 3b), the core of the distributions are found in the rope. Then, we can roughly assume statistical equivalence.

In summary, MonFkNN-PM obtains significantly better results in terms of accuracy and error cost than almost all of the considered methods. Our approach also achieves the most monotonic predictions alongside OSDL. MonFkNN-PM is slightly and non-statistically better than PMDT in terms of accuracy and error costs, but the former overwhelmingly outperforms PMDT considering monotonicity. Therefore, MonFkNN-PM is an overall better method.

The main reason behind the remarkable performance of MonFkNN is its capability of not sacrificing any objective of monotonic classification. Usually, some classifiers, such as OSDL, sacrifice accurate predictions in order to accomplish monotonic models. The results of OSDL for *artiset* and *bostonhousing* and the outcome of MkNN for *balance* are good examples of this statement. On the other hand, other methods, such as monotonic decision trees and particularly PMDT, achieve accurate predictions but break the monotonic constraints in their predictions more frequently. However, the MonFkNN procedure of training class membership extraction is designed to mitigate the influence of non-monotonic noisy data, without the need to aggressively modify the

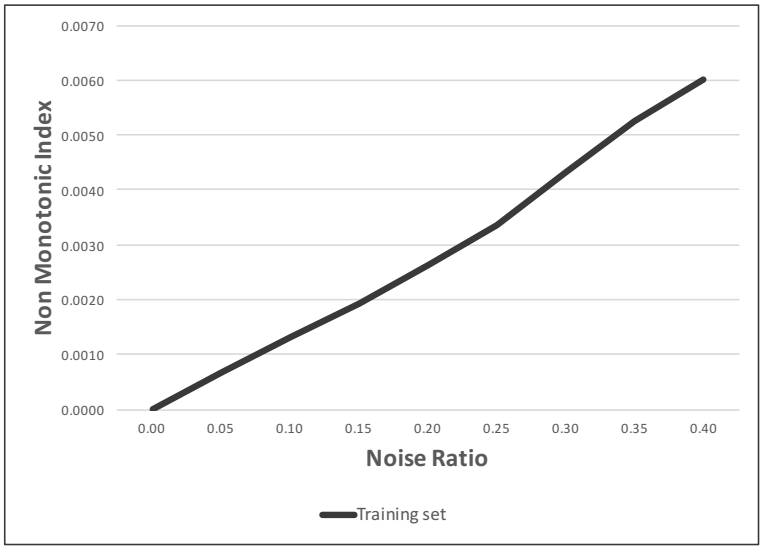


Figure 4: Impact of the addition of class noise in Artiset on monotonic violations measured by NMI.

training data as done by relabeling in MkNN. The MonFkNN prediction stage offers the flexibility of choice for most accurate or monotonic predictions. Additionally, MonFkNN includes technologies that are more appropriate for ordinal and monotonic classification, such as median as a final class.

5.4 On the robustness of Monotonic Fuzzy k-NN to monotonic noise

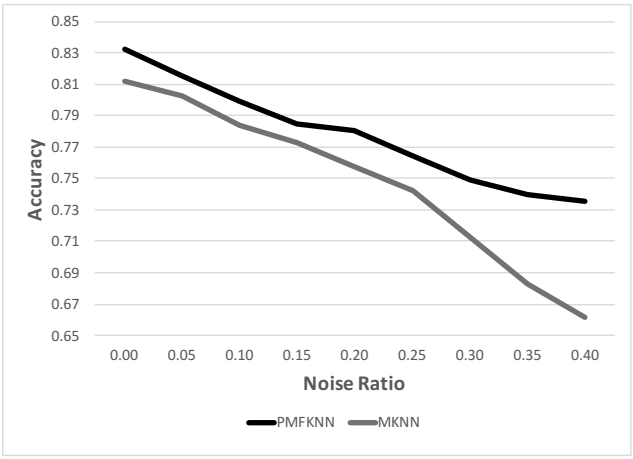
With this last empirical study, we aim to test the robustness of MonFkNN-PM to the presence of monotonic violations or noise in the training sets as compared to MkNN. Thus, we have introduced different amounts of noisy instances in the training partitions of the artificial data-set *Artiset*. Then, the performance of MonFkNN-PM and MkNN is measured and compared in terms of accuracy, MAE and NMI while the noise ratio increases.

In order to increase the impact of class noise, we have randomly under-sampled every training set to 25% of their instances. Then, a subset of randomly selected instances is converted to noise by changing their class labels. This label modification is done according to the adjacent classes of the implicated instance. Specifically, a large number of neighbors are computed for the future noisy example x_i. 15 nearest neighbors were the value used in this experiment. Next, the neighbors with the same class as x_i are removed and a new class is randomly obtained in relation to the presence ratio of other classes in its filtered neighbors. This ensures a certain degree of proximity between the changed sample and its new class.

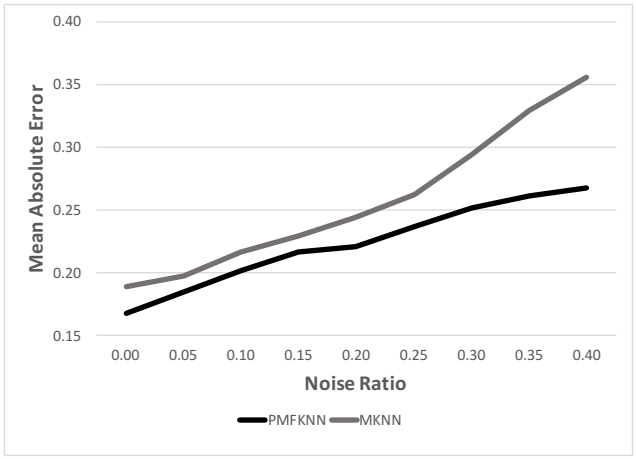
This process is executed following the same cross-validation scheme mentioned earlier. Since the noise generation has a random component, the experiment was repeated three times with different seeds, averaging the obtained results. After the noise generation and before the execution of MkNN, a relabeling technique [24] was applied to the resultant data-sets.

Figure 4 shows the impact of increasing noise on the number of monotonic violations in Artiset training sets. This effect is measured by the Non-Monotonic Index (NMI) over the resulting training samples. As previously mentioned, class noise significantly aggravates the monotonicity of the data-sets. The increase in NMI is directly proportional to the increase in noise as clearly shown in Figure 4.

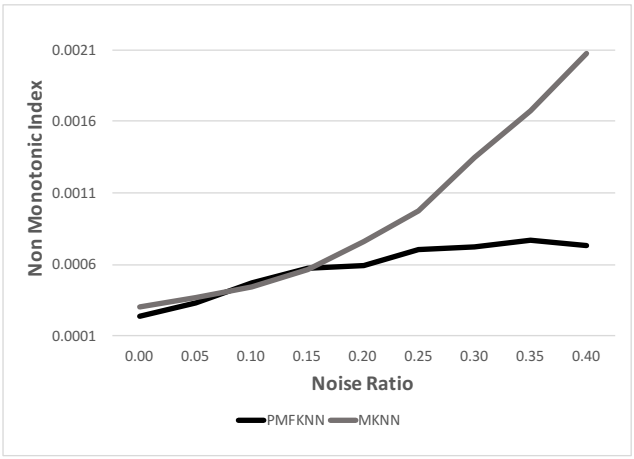
Figure 5 shows the performance of MonFkNN-PM and MkNN (darker and lighter lines, respectively) on the basis of precision (5a), MAE (5b) and NMI (5c), with the progression of noise. As expected, while the amount of noise grows, the performance of both methods get worse, that is, their accuracy decreases and errors and non-monotonic predictions increases. However, there are some big differences between classifiers.



(a) Noise effect in terms of Accuracy.



(b) Noise effect in terms of MAE.



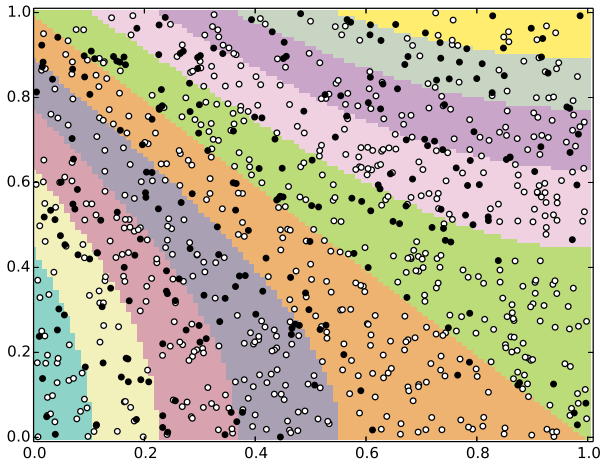
(c) Noise effect in terms of NMI.

Figure 5: Comparison of MonFkNN-PM and MkNN performance on Artiset data-set with the different amounts of noisy samples.

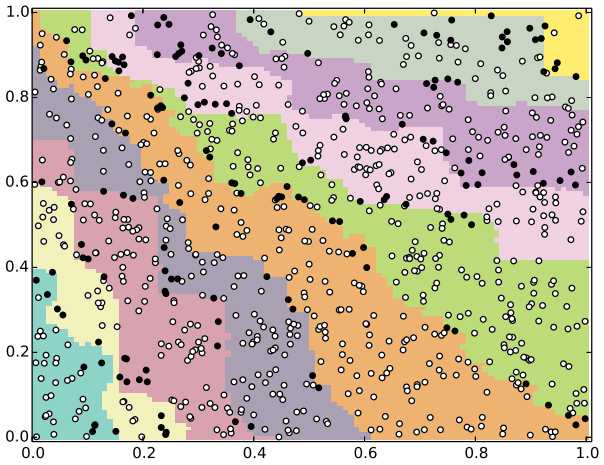
Firstly, the behavior of MonFkNN-PM facing noise is clearly better than that of MkNN in every tested aspect. The black lines are always located above the lighter ones in Figure 5a, which indicates greater accuracy, and under them in Figures 5b and 5c, meaning better MAE and NMI for MonFkNN-PM. Usually, the distance between both methods is large, with the exception of the NMI results obtained for the smallest values of noise. In addition, while the noise ratio increases, their differences also increase.

The slope of deterioration of MonFkNN-PM performance remains stable, even being reduced in some cases, while the MkNN slope becomes steeper as the amount of noise increases. This last event can be clearly seen when the noise ratio reaches the 25% of the instances, where the decline of MkNN is magnified, especially in terms of monotonicity (Figure 5c). On the other hand, the NMI results of MonFkNN-PM seem to increase at a slower rate by that point. This exhibits the great robustness of MonFkNN-PM to monotonic violations.

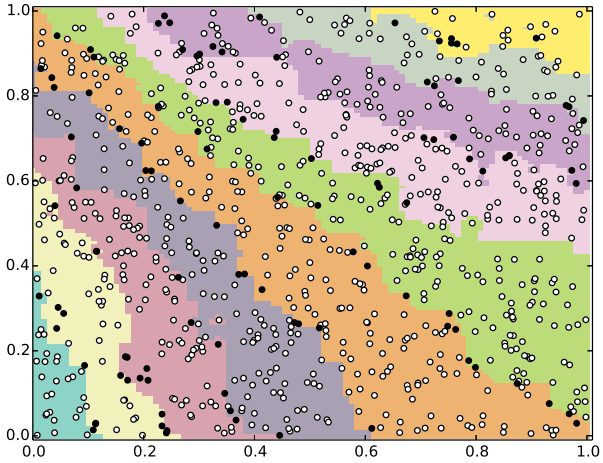
Next, the behavior of both methods in relation to noise are analyzed using a graphical example. Figure 6 is a graphical representation of the predictions and classification boundaries inferred by MkNN and MonFkNN-PM for *Artiset* with 35% noise. Figure 6a represents the perfect class surfaces defined by *Artiset* generation expression (see Section 4) and the training samples. In Figure 6a, black points represent the noise artificially



(a) Exact decision surface for Artiset. Black points represent noise.



(b) Decision surfaces inferred by MkNN.



(c) Decision surfaces inferred by MonFkNN-PM.

Figure 6: Classification boundaries inferred by MkNN and MonFkNN-PM from the plotted Artiset with 35% noisy instances. Black points represent the instances wrongly classified by the decision surfaces shown.

introduced into the data-set. In Figures 6b and 6c, the black examples are wrongly classified instances, while the right predictions are colored in white.

The first clear difference between the MkNN and MonFkNN-PM performances shown in Figures 6b and 6c is the amount of black dots. MonFkNN has far fewer classification mistakes than MkNN. Additionally, MonFkNN-PM is better at conserving the right regions for the classes, while MkNN can lose nearly all the entire sections of some of them. The regions in lighter and brighter yellow are shrunk by MkNN in favor of their adjacent classes.

With these experiments, MonFkNN has shown strong robustness to monotonic noise preserving the decision boundaries as precisely as possible, and hence, has performed well in terms of precision, error costs and monotonicity. This robustness is the result of all the procedures included in MonFkNN, but it may also be mainly due to the reduction of the impact of non-monotonic noise during the extraction of the class memberships of the training instances.

6 Conclusion

In this paper, we proposed a Fuzzy k-Nearest Neighbors model for classification with monotonic constraints. The final class label obtained from membership functions has been revised to respect these constraints. MonFkNN has been designed with different mechanisms to reduce the influence of monotonic violations. As a demonstration of its flexibility, two different model configurations with different behaviors have been presented.

Over the course of the experimental analyses, the great potential of both proposed versions, namely Pure and Approximate Monotonic Fuzzy k-NN, has been shown in relation to monotonicity and accuracy, respectively. Compared to other methods, MonFkNNN is significantly better in terms of accuracy and error cost, matching the best NMI results. In addition, it has shown its robustness to large amounts of noise while preserving its good performance.

Future proposals should be robust to monotonic noise in order to obtain accurate and monotonic predictions. MonFkNN, as an example, opens possibilities to other fuzzy approaches since they are also potentially reliable against noise. Additionally, fuzzy techniques may be useful when defining different levels of constraints between input and output attributes. That is, some attributes may be more important than others regarding monotonicity. This problem representation may be very useful for monotonic classifiers.

Acknowledgements

This work was supported by the Spanish Ministry of Economy and Competitiveness under Grant TIN2017-89517-P and a research scholarship (FPU) given to Sergio González by the Spanish Ministry of Education, Culture and Sports.

References

[1] José-Ramón Cano, Pedro Antonio Gutiérrez, Bartosz Krawczyk, Michał Woźniak, and Salvador García. Monotonic classification: an overview on algorithms, performance measures and data sets. *Neurocomputing*, 341:168–182, 2019.

[2] Chih-Chuan Chen and Sheng-Tun Li. Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16):7235–7247, 2014.

[3] Jose-Ramon Cano, Naif R Aljohani, Rabeeh Ayaz Abbasi, Jalal S Alowidbi, and Salvador Garcia. Prototype selection to improve monotonic nearest neighbor. *Engineering Applications of Artificial Intelligence*, 60:128–135, 2017.

[4] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1):29–43, 1995.

[5] Arie Ben-David, Leon Sterling, and TriDat Tran. Adding monotonicity to learning algorithms may impair their accuracy. *Expert Systems with Applications*, 36(3):6627–6634, 2009.

[6] Sergio González, Salvador García, Sheng-Tun Li, and Francisco Herrera. Chain based sampling for monotonic imbalanced classification. *Information Sciences*, 474:187–204, 2019.

[7] Sergio González, Francisco Herrera, and Salvador García. Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing*, 33(4):367–388, 2015.

[8] Christophe Marsala and Davide Petturiti. Rank discrimination measures for enforcing monotonicity in decision tree induction. *Information Sciences*, 291:143–171, 2015.

[9] Shenglei Pei and Qinghua Hu. Partially monotonic decision trees. *Information Sciences*, 424:104–117, 2018.

[10] Jesús Alcalá-Fdez, Rafael Alcalá, Sergio González, Yusuke Nojima, and Salvador García. Evolutionary fuzzy rule-based methods for monotonic classification. *IEEE Transactions on Fuzzy Systems*, 25(6):1376–1390, 2017.

[11] Sheng-Tun Li and Chih-Chuan Chen. A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge. *IEEE Transactions on Fuzzy Systems*, 23(5):1713–1727, 2015.

[12] Francisco Fernández-Navarro, Annalisa Riccardi, and Sante Carloni. Ordinal neural networks without iterative tuning. *IEEE Transactions on Neural Network and Learning Systems*, 25(11):2075–2085, 2014.

[13] Hong Zhu, Eric CC Tsang, Xi-Zhao Wang, and Rana Aamir Raza Ashfaq. Monotonic classification extreme learning machine. *Neurocomputing*, 225:205–213, 2017.

[14] K. Dembczyński, W. Kotłowski, and R. Słowiński. Learning rule ensembles for ordinal classification with monotonicity constraints. *Fundamenta Informaticae*, 94(2):163–178, 2009.

[15] Yuhua Qian, Hang Xu, Jiye Liang, Bing Liu, and Jieting Wang. Fusing monotonic decision trees. *IEEE Transactions on Knowledge Data Engineering*, 27(10):2717–2728, 2015.

[16] A. Ben-David. Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications. *Decision Sciences*, 23:1357–1372, 1992.

[17] W. Duivesteijn and A. Feelders. Nearest neighbour classification with monotonicity constraints. In *ECML/PKDD (1)*, pages 301–316, 2008.

[18] Stijn Lievens, Bernard De Baets, and Kim Cao-Van. A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Annals of Operations Research*, 163(1):115–142, 2008.

[19] Javier García, A. M. AlBar, N. R. Aljohani, J.-R. Cano, and S. García. Hyperrectangles selection for monotonic classification by using evolutionary algorithms. *International Journal of Computational Intelligence Systems*, 9(1):184–202, 2016.

[20] S. García, J. Luengo, and F. Herrera. *Data Preprocessing in Data Mining*. Springer, 2015.

[21] Weiwei Pan, Qinghua Hu, Yanping Song, and Daren Yu. Feature selection for monotonic classification via maximizing monotonic dependency. *International Journal of Computational Intelligence Systems*, 7(3):543–555, 2014.

[22] J.-R. Cano and S. García. Training set selection for monotonic ordinal classification. *Data & Knowledge Engineering*, 112:94 – 105, 2017.

[23] Rob Potharst, Arie Ben-David, and Michiel C. van Wezel. Two algorithms for generating structured and unstructured monotone ordinal data sets. *Engineering Applications of Artificial Intelligence*, 22(4-5):491–96, 2009.

[24] Ad Feelders. Monotone relabeling in ordinal classification. In *ICDM*, pages 803–808. IEEE Computer Society, 2010.

[25] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(4):580–585, 1985.

[26] Joaquín Derrac, Salvador García, and Francisco Herrera. Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects. *Information Sciences*, 260:98–119, 2014.

[27] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[28] S. García, A. Fernández, J. Luengo, and 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(10):2044–2064, 2010.

[29] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[30] David Charte, Francisco Charte, Salvador García, and Francisco Herrera. A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. *Progress in Artificial Intelligence*, pages 1–14, 2019.

[31] W. Kotłowski and R. Słowiński. On nonparametric ordinal classification with monotonicity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2576–2589, 2013.

[32] Lotfi A Zadeh. Fuzzy sets. In *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*, pages 394–432. World Scientific, 1996.

[33] Igor Škrjanc, Jose Iglesias, Araceli Sanchis, Daniel Leite, Edwin Lughofer, and Fernando Gomide. Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey. *Information Sciences*, 2019.

[34] Nimagna Biswas, Saurajit Chakraborty, Sankha Subhra Mullick, and Swagatam Das. A parameter independent fuzzy weighted k-nearest neighbor classifier. *Pattern Recognition Letters*, 101:80–87, 2018.

[35] Haim Levy. *Stochastic dominance: Investment decision making under uncertainty*. Springer, 2015.

[36] Stijn Lievens and Bernard De Baets. Supervised ranking in the weka environment. *Information Sciences*, 180(24):4763–4771, 2010.

[37] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. Keel 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, 2017.

[38] Bernhard Lang. Monotonic multi-layer perceptron networks as universal approximators. In *International Conference on Artificial Neural Networks*, pages 31–37. Springer, 2005.

[39] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[40] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. rNPBST: An R package covering non-parametric and bayesian statistical tests. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 281–292. Springer, 2017.

Bibliography

[1] Gregory Piateski and William Frawley. *Knowledge discovery in databases*. MIT press, 1991.

[2] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[3] Salvador García, Julián Luengo, and Francisco Herrera. *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated, 2015.

[4] Diego García-Gil, Julián Luengo, Salvador García, and Francisco Herrera. Enabling smart data: noise filtering in big data classification. *Information Sciences*, 479:135–152, 2019.

[5] Julián Luengo, Diego García-Gil, Sergio Ramírez-Gallego, Salvador García, and Francisco Herrera. *Big Data Preprocessing: Enabling Smart Data*. Springer Nature, 2020.

[6] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher Pal. Data mining: Practical machine learning tools and techniques. *Morgan Kaufmann*, 2016.

[7] Dorian Pyle. *Data preparation for data mining*. morgan kaufmann, 1999.

[8] Shichao Zhang, Chengqi Zhang, and Qiang Yang. Data preparation for data mining. *Applied artificial intelligence*, 17(5-6):375–381, 2003.

[9] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210, 2004.

[10] Julián Luengo, Salvador García, and Francisco Herrera. 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, 2012.

[11] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[12] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

[13] Huan Liu and Hiroshi Motoda. *Instance selection and construction for data mining*, volume 608. Springer Science & Business Media, 2013.

[14] Sergio Ramírez-Gallego, Salvador García, Héctor Mouriño-Talín, David Martínez-Rego, Verónica Bolón-Canedo, Amparo Alonso-Betanzos, José Manuel Benítez, and Francisco Herrera. Data discretization: taxonomy and big data challenge. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(1):5–21, 2016.

[15] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[16] Norman R Draper and Harry Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, 1998.

[17] Boris Mirkin. *Clustering: a data recovery approach*. CRC Press, 2012.

[18] Krzysztof J Cios, Roman W Swiniarski, Witold Pedrycz, and Lukasz A Kurgan. Unsupervised learning: association rules. In *Data Mining*, pages 289–306. Springer, 2007.

[19] David Charte, Francisco Charte, Salvador García, and Francisco Herrera. A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. *Progress in Artificial Intelligence*, 8(1):1–14, 2019.

[20] Jerónimo Hernández-González, Iñaki Inza, and Jose A Lozano. Weak supervision and other non-standard classification problems: a taxonomy. *Pattern Recognition Letters*, 69:49–55, 2016.

[21] Francisco Herrera, Sebastián Ventura, Rafael Bello, Chris Cornelis, Amelia Zafra, Dánel Sánchez-Tarragó, and Sarah Vluymans. *Multiple instance learning*. Springer, 2016.

[22] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[23] Francisco Herrera, Francisco Charte, Antonio J Rivera, and María J Del Jesus. *Multilabel classification*. Springer, 2016.

[24] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.

[25] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. *Semi-supervised learning*. The MIT Press, 2010.

[26] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*. Springer, 2018.

[27] Mary M Moya and Don R Hush. Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, 9(3):463–474, 1996.

[28] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.

[29] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[30] Pedro Antonio Gutierrez, Maria Perez-Ortiz, Javier Sanchez-Monedero, Francisco Fernandez-Navarro, and Cesar Hervas-Martinez. Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):127–146, 2015.

[31] José-Ramón Cano, Pedro Antonio Gutiérrez, Bartosz Krawczyk, Michał Woźniak, and Salvador García. Monotonic classification: an overview on algorithms, performance measures and data sets. *Neurocomputing*, 341:168–182, 2019.

[32] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Andres Folleco. An empirical study of the classification performance of learners on imbalanced and noisy software quality data. *Information Sciences*, 259:571–595, 2014.

[33] José-Ramón Cano, Julián Luengo, and Salvador García. Label noise filtering techniques to improve monotonic classification. *Neurocomputing*, 353:83–95, 2019.

[34] Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320, 2012.

[35] Jie Sun, Jie Lang, Hamido Fujita, and Hui Li. Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates. *Information Sciences*, 425:76–91, 2018.

[36] Kai Lei, Yuexiang Xie, Shangru Zhong, Jingchao Dai, Min Yang, and Ying Shen. Generative adversarial fusion network for class imbalance credit scoring. *Neural Computing and Applications*, pages 1–12, 2019.

[37] Man Leung Wong, Kruy Seng, and Pak Kan Wong. Cost-sensitive ensemble of stacked denoising autoencoders for class imbalance problems in business domain. *Expert Systems with Applications*, 141:112918, 2020.

[38] David A Cieslak, T Ryan Hoens, Nitesh V Chawla, and W Philip Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012.

[39] L Gonzalez-Abril, Haydemar Nuñez, Cecilio Angulo, and F Velasco. Gsvm: An svm for handling imbalanced accuracy between classes inbi-classification problems. *Applied Soft Computing*, 17:23–31, 2014.

[40] E. Ramentol, S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, and F. Herrera. Ifrowann: Imbalanced fuzzy-rough ordered weighted average nearest neighbor classification. *IEEE Transactions on Fuzzy Systems*, 23(5):1622–1637, 2015.

[41] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[42] Susan Lomax and Sunil Vadera. A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys*, 45(2):16:1–16:35, 2013.

[43] Ronaldo C. Prati, Gustavo E. A. P. A. Batista, and Diego F Silva. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*, 45(1):247–270, 2015.

[44] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, 2012.

[45] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1):29–43, 1995.

[46] Christophe Marsala and Davide Petturiti. Rank discrimination measures for enforcing monotonicity in decision tree induction. *Information Sciences*, 291:143–171, 2015.

[47] Shenglei Pei and Qinghua Hu. Partially monotonic decision trees. *Information Sciences*, 424:104–117, 2018.

[48] Sheng-Tun Li and Chih-Chuan Chen. A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge. *IEEE Transactions on Fuzzy Systems*, 23(5):1713–1727, 2015.

[49] Jesús Alcalá-Fdez, Rafael Alcalá, Sergio González, Yusuke Nojima, and Salvador García. Evolutionary fuzzy rule-based methods for monotonic classification. *IEEE Transactions on Fuzzy Systems*, 25(6):1376–1390, 2017.

[50] Francisco Fernández-Navarro, Annalisa Riccardi, and Sante Carloni. Ordinal neural networks without iterative tuning. *IEEE Transactions on Neural Network and Learning Systems*, 25(11):2075–2085, 2014.

[51] Hong Zhu, Eric CC Tsang, Xi-Zhao Wang, and Rana Aamir Raza Ashfaq. Monotonic classification extreme learning machine. *Neurocomputing*, 225:205–213, 2017.

[52] W. Duivesteijn and A. Feelders. Nearest neighbour classification with monotonicity constraints. In *ECML/PKDD (1)*, pages 301–316, 2008.

[53] Stijn Lievens, Bernard De Baets, and Kim Cao-Van. A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Annals of Operations Research*, 163(1):115–142, 2008.

[54] Javier García, A. M. AlBar, N. R. Aljohani, J.-R. Cano, and S. García. Hyperrectangles selection for monotonic classification by using evolutionary algorithms. *International Journal of Computational Intelligence Systems*, 9(1):184–202, 2016.

[55] K. Dembczyński, W. Kotłowski, and R. Słowiński. Learning rule ensembles for ordinal classification with monotonicity constraints. *Fundamenta Informaticae*, 94(2):163–178, 2009.

[56] Yuhua Qian, Hang Xu, Jiye Liang, Bing Liu, and Jieting Wang. Fusing monotonic decision trees. *IEEE Transactions on Knowledge Data Engineering*, 27(10):2717–2728, 2015.

[57] Chih-Chuan Chen and Sheng-Tun Li. Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16):7235–7247, 2014.

[58] Marina Velikova and Hennie Daniels. Decision trees for monotone price models. *Computational Management Science*, 1(3):231–244, Oct 2004.

[59] Jose-Ramon Cano, Naif R Aljohani, Rabeeh Ayaz Abbasi, Jalal S Alowidbi, and Salvador Garcia. Prototype selection to improve monotonic nearest neighbor. *Engineering Applications of Artificial Intelligence*, 60:128–135, 2017.

[60] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[61] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[62] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[63] Lior Rokach. *Ensemble Learning: Pattern Classification Using Ensemble Methods.* World Scientific, 2019.

[64] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications.* Springer, 2012.

[65] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[66] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[67] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[68] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[69] Abraham J Wyner, Matthew Olson, Justin Bleich, and David Mease. Explaining the success of adaboost and random forests as interpolating classifiers. *The Journal of Machine Learning Research*, 18(1):1558–1590, 2017.

[70] Lior Rokach. Decision forest: Twenty years of research. *Information Fusion*, 27:111–125, 2016.

[71] Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

[72] Gonzalo Martínez-Muñoz and Alberto Suárez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38(10):1483–1494, 2005.

[73] S. García, J. Derrac, J.R. Cano, and F. Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):417–435, 2012.

[74] I. Triguero, J. Derrac, S. García, and F. Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 42(1):86–100, 2012.

[75] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[76] Jianjun Zhang, Ting Wang, Wing WY Ng, Shuai Zhang, and Chris D Nugent. Undersampling near decision boundary for imbalance problems. In *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 1–8. IEEE, 2019.

[77] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[78] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[79] J. M. Keller, M. R. Gray, and J. A. Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(4):580–585, July 1985.

[80] Joaquin Derrac, Salvador García, and Francisco Herrera. Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects. *Information Sciences*, 260:98 – 119, 2014.

[81] Isaac Triguero, Diego García-Gil, Jesús Maillo, Julián Luengo, Salvador García, and Francisco Herrera. Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(2):e1289, 2019.

[82] Isaac Triguero, Sara del Río, Victoria López, Jaume Bacardit, José M. Benítez, and Francisco Herrera. ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem. *Knowledge-Based Systems*, 87:69 – 79, 2015.

[83] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113 – 141, 2013.

[84] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 63–66. Springer, 2001.

[85] Mikel Galar, Alberto Fernández, Edurne Barrenechea, and Francisco Herrera. EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 46(12):3460–3471, 2013.

[86] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE, 2008.

[87] Sukarna Barua, Md Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2014.

[88] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

[89] Zhong-Liang Zhang, Xing-Gang Luo, Sergio González, Salvador García, and Francisco Herrera. DRCW-ASEG: One-versus-one distance-based relative competence weighting with adaptive synthetic example generation for multi-class imbalanced datasets. *Neurocomputing*, 285:176–187, 2018.

[90] Rob Potharst, Arie Ben-David, and Michiel C. van Wezel. Two algorithms for generating structured and unstructured monotone ordinal data sets. *Engineering Applications of Artificial Intelligence*, 22(4-5):491–96, 2009.

[91] Ad Feelders. Monotone relabeling in ordinal classification. In *ICDM*, pages 803–808. IEEE Computer Society, 2010.

[92] Derek T. Ahneman, Jesús G. Estrada, Shishi Lin, Spencer D. Dreher, and Abigail G. Doyle. Predicting reaction performance in c–n cross-coupling using machine learning. *Science*, 360(6385):186–190, 2018.

[93] Kanggeun Lee, Hyoung-oh Jeong, Semin Lee, and Won-Ki Jeong. Cpem: Accurate cancer type classification based on somatic alterations using an ensemble of a random forest and a deep neural network. *Scientific reports*, 9(1):1–9, 2019.

[94] Julián Luengo, Seong-O Shim, Saleh Alshomrani, Abdulrahman Altalhi, and Francisco Herrera. Cnc-nos: Class noise cleaning by ensemble filtering and noise scoring. *Knowledge-Based Systems*, 140:27–49, 2018.

[95] Xiaoling Lu, Jiesheng Si, Lanfeng Pan, and Yanyun Zhao. Imputation of missing data using ensemble algorithms. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 2, pages 1312–1315. IEEE, 2011.

[96] Nicolás García-Pedrajas and Aida De Haro-García. Boosting instance selection algorithms. *Knowledge-Based Systems*, 67:342–360, 2014.

[97] Verónica Bolón-Canedo and Amparo Alonso-Betanzos. Ensembles for feature selection: A review and future trends. *Information Fusion*, 52:1–12, 2019.

[98] Grigorios Tsoumakas, Ioannis Partalas, and Ioannis Vlahavas. A taxonomy and short review of ensemble selection. In *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*, pages 1–6, 2008.

[99] Rafael MO Cruz, Robert Sabourin, and George DC Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2018.

[100] Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*, 2013.

[101] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[102] Lotfi A Zadeh. Fuzzy sets. In *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*, pages 394–432. World Scientific, 1996.

[103] A. Ben-David, L. Sterling, and T. Tran. Adding monotonicity to learning algorithms may impair their accuracy. *Expert Systems with Applications*, 36(3):6627–6634, 2009.

[104] Gilles Louppe and Pierre Geurts. Ensembles on random patches. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 346–361. Springer, 2012.

[105] Jerome H. Friedman. Stochastic gradient boosting. *Computional Statistics & Data Analysis*, 38(4):367–378, 2002.

[106] Korlakai Vinayak Rashmi and Ran Gilad-Bachrach. DART: Dropouts meet multiple additive regression trees. In *AISTATS*, pages 489–497, 2015.

[107] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[108] Pedro Antonio Gutiérrez, María Pérez-Ortiz, and Alberto Suárez. Class switching ensembles for ordinal regression. In *International Work-Conference on Artificial Neural Networks*, pages 408–419. Springer, 2017.

[109] Alberto FernáNdez, Victoria LóPez, Mikel Galar, MaríA José Del Jesus, and Francisco Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-based systems*, 42:97–110, 2013.

[110] Jennifer Wortman Vaughan. Making better use of the crowd: How crowdsourcing can advance machine learning research. *The Journal of Machine Learning Research*, 18(1):7026–7071, 2017.

[111] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 1–14, 2012.

[112] Diego García-Gil, Sergio Ramírez-Gallego, Salvador García, and Francisco Herrera. A comparison on scalability for batch big data processing on apache spark and apache flink. *Big Data Analytics*, 2(1):1, 2017.

[113] X. Wu, X. Zhu, G. Wu, and W. Ding. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107, Jan 2014.

[114] Sergio Ramírez-Gallego, Alberto Fernández, Salvador García, Min Chen, and Francisco Herrera. Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce. *Information Fusion*, 42:51–61, 2018.