Journal Pre-proof

FW-SMOTE: a feature-weighted oversampling approach for imbalanced classification

Sebastián Maldonado, Carla Vairetti, Alberto Fernandez, Francisco Herrera

PII: S0031-3203(21)00687-7
DOI: https://doi.org/10.1016/j.patcog.2021.108511
Reference: PR 108511

To appear in: *Pattern Recognition*

Received date: 21 September 2020
Revised date: 17 December 2021
Accepted date: 22 December 2021

Please cite this article as: Sebastián Maldonado, Carla Vairetti, Alberto Fernandez, Francisco Herrera, FW-SMOTE: a feature-weighted oversampling approach for imbalanced classification, *Pattern Recognition* (2021), doi: https://doi.org/10.1016/j.patcog.2021.108511

Highlights

- We claim that SMOTE has a weakness when facing high-dimensional problems
- We propose a general version of the SMOTE strategy using OWA operators.
- The proposal includes a feature weighting process that considers relevancy/redundancy.
- This new component leads to a better definition of the neighborhood of minority samples.
- Experiments carried out on 42 datasets show the virtues of our method.

FW-SMOTE: a feature-weighted oversampling approach for imbalanced classification

Sebastián Maldonado[a,d], Carla Vairetti[b,d,*], Alberto Fernandez[c], Francisco Herrera[c,e]

[a] *Department of Management Control and Information Systems, School of Economics and Business, University of Chile, Santiago, Chile. (e-mail: sebastianm@fen.uchile.cl).*
[b] *Universidad de los Andes, Chile, Facultad de Ingeniería y Ciencias Aplicadas. (e-mail: cvairetti@uandes.cl).*
[c] *Andalusian Research Institute in Data Science and Computational Intelligence, University of Granada, 18071 Granada, Spain. (e-mail: {alberto,herrera}@decsai.ugr.es).*
[d] *Instituto Sistemas Complejos de Ingeniería (ISCI), Chile.*
[e] *Faculty of Computing and Information Technology, King Abdulaziz University Jeddah, Saudi Arabia.*

Abstract

The Synthetic Minority Over-sampling Technique (SMOTE) is a well-known resampling strategy that has been successfully used for dealing with the class-imbalance problem, one of the most challenging pattern recognition tasks in the last two decades. In this work, we claim that SMOTE has an important issue when defining the neighborhood in order to create new minority samples: the use of the Euclidean distance may not be suitable in high-dimensional settings. Our hypothesis is that the use of a weighted metric that does not assume that all features are equally important could improve performance in the presence of noisy/redundant variables. In this line, we present a novel SMOTE-like method that uses the weighted Minkowski distance for defining the neighborhood for each example of the minority class. This methodology leads to a better definition of the neighborhood since it prioritizes those features that are more relevant for the classification task. A complementary advantage of the proposal is performing feature selection since attributes can be discarded when their corresponding weights are below a given threshold. Our experiments on 42 class-imbalance datasets show the virtues of the proposed SMOTE variant, achieving the best predictive performance when compared with the traditional SMOTE approach and other recent variants on low- and high-dimensional settings, handling issues such as class overlap and hubness adequately without increasing the complexity of the method.

Keywords: Data resampling, SMOTE, OWA operators, Feature selection, Imbalanced data classification.

*Corresponding author

1. Introduction

Imbalanced classification is a topic of high interest for nowadays applications. This issue may cause a bias during the learning process so that minority class instances are not well-identified [1, 2].

One of the most straightforward approaches to overcome the class-imbalance issue is via preprocessing techniques, such as data resampling [3]. Resampling is beneficial to rebalance the classes, to clean the borderline areas, and to enlarge the minority class regions [4]. The Synthetic Minority Oversampling TEchnique (SMOTE) has become the "de facto" standard for imbalanced classification via resampling [3, 5]. SMOTE generates new minority class examples in the neighboring areas of the original ones by means of interpolation.

We claim in this paper that SMOTE has a weakness when facing high-dimensional problems. Generally, the classical Euclidean distance metric is chosen for computing the neighbors for each minority class instance. However, this metric may converge to the same value for all instances in high-dimensional settings, becoming almost uniformly distant from each other. In the particular case of SMOTE, this "curse of the dimensionality" may lead to an ineffective interpolation process [6].

The hypothesis of this study is that a weighted distance measure can alleviate the "curse of the dimensionality" in the k-nearest neighbors step of SMOTE. Furthermore, this metric can be also useful in the presence of noisy/redundant variables, even in low-dimensional settings. The Euclidean distance assumes that all variables are equally important, which is seldom correct in most machine learning tasks. A general metric that is able to weight the features according to their importance can be very useful at defining a proper neighborhood for the minority samples, leading to a better predictive performance of the base classifier [6].

An adequate definition of the neighborhood also strengthens the boundaries of the positive region, mitigating the hubness issue [7]. This phenomenon occurs when few points in the minority class account for most of the observed neighbor occurrences due to the skewness in the distribution. This issue is faced usually in high-dimensional settings [7].

In this work we propose a general version of the SMOTE strategy that includes a feature weighting process that takes into account relevancy/redundancy. This new component is intended to lead to a better definition of the neighborhood of minority samples, conferring flexibility to the process of generating synthetic examples. The new methodology is named after FW-SMOTE, which stands for *Feature Weighted-SMOTE*.

During the k-nearest neighbors step, the proposed method is designed to discard those variables whose weights are below a given threshold. This is done for reducing the negative effect of the curse of the dimensionality in the construction of the neighborhood. Although the variables *are not removed for the classification task*, FW-SMOTE has the potential to perform feature selection by utilizing the feature weighting strategy to select the input variables for classification without increasing the complexity of the method. We explore this

approach in the present study, achieving best predictive results compared to alternative feature selection strategies that deal with the class imbalance problem.

The feature weighting process is done by including the Induced Ordered Weighted Average (IOWA) operator [8] in the definition of the distance metric. This results in a variation of the weighted Minkowski distance, called induced Minkowski OWA distance (IMOWAD) [9]. The weights for the attributes are defined using a fast feature ranking method, such as the Fisher Score [10], as an input. The goal is to prioritize those features that are more relevant for the classification task in the k-nearest neighbors step. These modifications are simple statistical operations that do not increase the complexity of the problem significantly.

We developed a comprehensive experimental study carried out on 42 low- and high-dimensional datasets. The goal of this analysis is three-fold: First, we show that FW-SMOTE performs much better than other SMOTE variants in terms of predictive performance. Next, we incorporate feature selection in the learning process, showing the virtues of our strategy at dealing with redundant/irrelevant variables. Finally, a stability analysis is performed, illustrating the robustness of FW-SMOTE in terms of predictive performance under different parameter settings. Our study is complemented with theoretical discussions on aspects such as complexity and learning in an embedding space.

In summary, this research fills an important gap in the SMOTE literature when noisy/redundant attributes are present. FW-SMOTE is the first resampling technique that considers a weighted scheme for the definition of the neighborhood in the SMOTE algorithm, to the best of our knowledge, being also the first one that considers OWA operators.

The remainder of this paper is structured as follows: In Section 2 the class-imbalance problem and SMOTE variations are presented. The core of this research work where FW-SMOTE is formalized in Section 3. Next, we carry out an empirically analysis to confirm the good behavior of FW-SMOTE on various low and high-dimensional datasets in Section 4. Finally, the main conclusions of this study are presented in Section 5.

2. The class-imbalanced problem

When it comes to addressing classification for imbalanced problems, there are different types of approaches that can be considered. Specifically, we may highlight preprocessing techniques, cost-sensitive learning, one-class classification (OCC), and feature selection [11].

Among these solutions, the use of resampling-based preprocessing to balance class distribution has undoubtedly been the most widely used of them all [5]. The advantages of data resampling are clear, since it allows different proposals to be applied to the same or several classifiers, with the aim of identifying the one approach that best adapts to the input data. There are two types of preprocessing techniques. On the one hand, there are oversampling methods

that replicate instances of the minority class. And on the other hand, there are undersampling methods that eliminate examples of the majority class. Each approach has its specific capabilities. For example, oversampling allows maintaining the original information of the problem, strengthening the borderline areas for the clusters of the minority class, while undersampling allows an implicit cleaning of possible noisy data and helps in the treatment of class overlap [6].

A disadvantage related to resampling is that it may either remove relevant information (undersampling) or introduce new artificially-generated data, potentially biasing the results (oversampling). To avoid these issues, algorithmic-level solutions have been proposed, such as cost-sensitive learning and one-class classification. These approaches are trained with class-imbalanced data directly [12].

On the one hand, cost-sensitive techniques include an estimation of the misclassification costs, adapting existing machine learning techniques to favor the minority class. Support Vector Machine (SVM) is a well-known classification method that offers great flexibility. Therefore, several cost-sensitive extensions have been designed with excellent predictive results [13, 14].

On the other hand, OCC approaches addresses the class-imbalance problem by constructing a *description* of the target class. This is done by training the model without the information of the labels, as in a clustering algorithm for outlier detection [12]. Similar to cost-sensitive techniques, several OCC variants have been developed considering SVM as the baseline classifier, including the well-known Support Vector Data Description (SVDD) [12].

From the myriad of different solutions in this context, in this research work we focus on SMOTE [5], due to its good properties and widespread usage. To do so, first we provide a gentle introduction to the preprocessing in general, and to the working procedure of SMOTE in particular (Section 2.1). Then, we enumerate several extensions of SMOTE that have been proposed in the literature to overcome some of its initial drawbacks (Section 2.2). Finally, feature selection is discussed as a potential solution for dealing with the class-imbalance issue in high-dimensional settings (Section 2.3).

2.1. Resampling and the SMOTE preprocessing algorithm

The baseline oversampling algorithm is random oversampling. It works simply by replicating examples of the majority class, which basically implies a higher weight in case of misclassification; but it may cause overfitting [15]. In 2002, N.V. Chawla proposed a novel approach as an alternative to the former standard method [16]. The idea was to assist the classifier to improve its generalization by creating new minority instances. This technique was named SMOTE. The basis of its procedure was to carry out an interpolation among neighboring minority class instances. As such, it was able to increase the number of minority class instances by introducing new minority class examples in their cluster areas.

Given a sample $\mathbf{x}_i$ from the minority class, and N randomly chosen samples from its neighborhood $\mathbf{x}_i^p$, with $p = 1, \ldots, N$, a new synthetic sample $\mathbf{x}_i^{*p}$ is obtained with the following expression:

$$\mathbf{x}_i^{*p} := \mathbf{x}_i + u\left(\mathbf{x}_i^p - \mathbf{x}_i\right), \tag{1}$$

where u is a randomly generated number between 0 and 1. This method has the advantages of being fast to compute and successful at providing balanced and accurate classification performance.

2.2. Extensions for SMOTE preprocessing

Many different extensions of this original approach have been proposed with the objective of improving some of the capabilities of SMOTE [16]. One common alternative is to compute which the best candidates are to be oversampled in the data before the process of synthetic example generation starts. The idea behind this type of strategy is being able to address the problems of noise and/or overlapping within the training dataset [17].

Based on the idea of data complexity, ADASYN [18] selects the amount of oversampling for each minority example dynamically by estimating its intrinsic difficulty, which is based on the ratio of examples belonging to the majority class in the neighborhood.

A similar approach is the Adaptive Neighbor SMOTE (AN-S.) [19]. In this case, it uses the oversampling process with a different K-parameter for each minority instance. The selected value is based on the density of surrounding same-class instances.

There are also some techniques that focus on the selection of instances which are closer to the boundary areas, such as Borderline-SMOTE (BL-S.), a clearly representative approach [20]. This algorithm draws on the premise that the examples far from the borderline may not provide a strong contribution to the classification ability of the model. To obtain the borderline areas, the algorithm works by considering the ratio between the majority and minority examples within the neighborhood of each instance to be oversampled.

A method inspired by BL-S. is Density Based SMOTE (DB-S.) [21]. Its working procedure is based on a density-based approach of clustering called DBSCAN, and it generates the synthetic instances by computing the shortest path from each minority instance to a pseudo-centroid of a minority-class cluster. Like BL-S., it operates in an overlapping region, but the main difference is that it seeks to maintain both the minority and majority class predictions.

Other types of techniques are based on identifying whether or not to generate synthetic examples by examining the actual number of minority class examples belonging to the complete neighborhood. Such is the case with SafeLevel-SMOTE (SL-S.) [22]. Specifically, this technique assigns each minority example a so-called "safe level" value, obtained as the ratio of the number of minority examples within its neighborhood area. Then, the interpolation function is modified using a gap that is dependent on the former safe level ratio of each minority instance.

A recent adaptation of the former SL-S. technique, named Relocating Safe-level SMOTE (RSL-S.) [23], is dedicated to ensuring that new synthetic positive instances are truly far from the surrounding areas of the negative examples. To do this, the instance generation procedure is repeated while the distance to the nearest majority class instance is lower than the distance to the original minority class.

Similar to the previously described techniques, MWMOTE [24] seeks to ensure that new minority instances are generated within a "correct" class cluster. To do so, it applies a weighting mechanism based on the distance to the nearest majority class point. Then, synthetic examples are generated from the weighted informative minority class instance using a clustering approach.

Another resampling approach, called Radial-Based Oversampling (RBO), was proposed recently by Koziarski et al. [25]. The main goal of RBO is to find adequate regions of interest for generating synthetic examples from the minority class by using radial basis functions.

To the best of our knowledge on SMOTE and all its extensions, the definition of the neighborhood is computed while giving all the variables the same importance. However, in a high-dimensional context, this may imply a strong bias caused by two issues. The first is that the higher the number of variables, the higher the convergence to a similar distance value for all examples. Secondly, it is not able to cope with noise and redundancy properly [26]. In these cases, a smart computation of the neighborhood instances is mandatory to guarantee the generation of useful synthetic data.

2.3. Feature selection for class-imbalanced datasets

When addressing a classification problem, the success of any ML algorithm depends greatly on the inner characteristics of the dataset [27]. Some issues that could hinder the classification ability are as the imbalanced problem (uneven class distribution), the curse of dimensionality (a high number of input attributes), or the class overlapping (same a priori class probability within a small cluster) [28].

A common solution for this scenario is applying feature selection methods. As its name suggests, the goal is to reduce the full feature set to a strong subset with the same (or possibly better) discerning capacity. Irrelevant, noisy, and redundant information can be discarded, yielding better predictive performance since it reduces the risk of overfitting [12]. To carry out this procedure, there are three types of schemes. One is using filtering methods which compute the score or significance of each attribute in order to rank the input variables. The second one is using wrapper approaches that use an auxiliary classification method for establishing the best cooperating variables in an iterative procedure. Finally, embedded methods perform feature selection in the training process, and are specific to a given machine learning method [12].

Filter methods have been discussed in the literature in combination with data resampling techniques or cost-sensitive classification approaches [29, 30]. Some filter methods, such as the Fisher score, mutual information, or the correlation score (also known as Correlation-based Feature Selection, or CFS), do not

require adaptations in order to be suitable for class-imbalance classification [6]. Alternatively, well-known filter methods such as Relief have been adapted for the task of filtering out irrelevant attributes taking the class-imbalance problem into consideration [29, 30].

There are some recent studies that propose wrapper strategies for simultaneous feature selection and class-imbalance classification. Since the exhaustive search for an optimal feature subset is a complex combinatorial problem, genetic algorithms and other metaheuristics have been proposed for this task [31]. For example, Chen et al. [32]combined neighborhood rough set theory with the Particle Swarm Optimization (PSO) metaheuristic for this task. Neighborhood rough set theory was also considered in [33] for online feature selection based on k-nearest neighbors.

Some embedded feature selection methods related to the class-imbalance problem have been proposed, especially in relation with the SVM method [12, 14]. For example, Zhang et al. [34] proposed the Border-Resampling Feature Elimination (SVM-BRFE), in which misclassified minority samples from the boundary constructed by the SVM method are oversampled for a better assessment of the feature relevance. Alternatively, a backward iterative process based on balanced accuracy was proposed in [35], resulting in the BFE-SVM method.

In summary, several feature selection strategies have been designed for dealing with the class-imbalanced problem. However, most state of the art approaches follow two independent processes for feature elimination and data resampling, or evaluate different combinations via metaheuristics. We propose a novel scheme, in which a feature ranking step is introduced in the SMOTE strategy.

3. The FW-SMOTE method: using feature weighting and selection generating artificial instances

The main idea of the proposed oversampling technique is to generalize the classic SMOTE approach using aggregation operators. FW-SMOTE deals with two important issues: the curse of dimensionality that affects distance measures in case of high-dimensional datasets, and the fact that although SMOTE weights all attributes equally for defining its neighborhood, most datasets contain redundant or irrelevant covariates which can be weighted down to favor relevant attributes.

FW-SMOTE replaces of the Euclidean distance used in SMOTE oversampling by the IMOWAD distance, which is a very flexible norm that allows a weighting process for the attributes via the IOWA operator. Notice that the traditional SMOTE approach is a special case of our proposal, in which the Euclidean norm is used and all attributes are equally weighted.

In spite of its simplicity and good general performance, the SMOTE approach has a noticeable drawback. The Euclidean distance used for computing the k nearest neighbors of a minority sample assumes that all variables are equally relevant; an assumption that seldom holds for most applications, and,

in particular, in imbalanced domains [6]. A feature weighting process that takes into account relevancy/redundancy into account could lead to a better definition of the neighborhood. A high-dimensional setting worsens the problem since the Euclidean distance tends to converge to the same value for all instances, according to the curse of the dimensionality.

FW-SMOTE is not only presented as a weighting strategy, but also as a resampling method that has the ability of performing variable selection. The weights provide an additional insight in the sense that attributes can be discarded from the whole learning process and not only from the definition of the neighborhood. A threshold ϵ is defined in order to remove variables based on their relevancy according to the feature ranking technique. In our experimental analysis we explore both alternatives: FW-SMOTE as a weighted resampling technique and its extension as a feature selection method.

In Section 3.1, we introduce the concept of OWA operators and present the IOWA-based distance measure used in FW-SMOTE. Next, the feature ranking techniques that induce the order of the IOWA operator are discussed in Section 3.2. Then, the FW-SMOTE algorithm is formalized in Section 3.3. This section concludes with a theoretical analysis regarding the contribution of FW-SMOTE and its relation with other pattern recognition techniques. This analysis is presented in Section 3.4.

3.1. The weighted distance measure for FW-SMOTE

The ordered weighted averaging (OWA) operator was introduced by R.R. Yager to provide a method for aggregating several inputs that lie between the maximum and minimum values [36]. It is widely used when aggregating the data according to the attitudinal character of the decision-maker. The classic OWA operator is given by:

$$OWA(a_1, a_2, \ldots, a_n) = \sum_{j=1}^{n} w_j b_j \tag{2}$$

where b_j is the j_{th} largest value of the input vector $(a_1, a_2, \ldots, a_n)$, and $w_j \in [0, 1]$ are the weights, with $\sum_{j=1}^{n} w_j = 1$.

The classic OWA operator assumes that the value of the elements to be aggregated is relevant for defining the weights. This is not true in our case since we want to induce order using an external variable, so the rankings are obtained by filter methods. Therefore, we use the induced OWA (IOWA) operator [8], which is a generalized version of the classic OWA operator. Given input tuples of the form $(\langle u_1, a_1\rangle, \langle u_2, a_2\rangle, \ldots, \langle u_n, a_n\rangle)$, where the a values are the objects to be aggregated and the u the order-inducing vector, the IOWA operator has the following form:

$$IOWA(\langle u_1, a_1\rangle, \ldots, \langle u_n, a_n\rangle) = \sum_{j=1}^{n} w_j b_j \tag{3}$$

where b_j is the a value of the IOWA tuple that has the j_{th} largest u value. Similar to those of the classic OWA operator, $w_j \epsilon [0,1]$ are the weights, with $\sum_{j=1}^{n} w_j = 1$.

The use of aggregation operators such as IOWA has become popular due to their ability to provide a more general process for reordering the information [37]. The flexibility provided by these operators can be extremely useful for designing machine learning schemes that are robust in the presence of problems such as noisy data and/or outliers [8].

FW-SMOTE considers a variant of the weighted Minkowski distance for the definition of the neighborhood. The weights for the distance measure are suggested to be obtained via an IOWA operator. Given two observations $\mathbf{x}_i \in \Re^n$ and $\mathbf{x}_{i'} \in \Re^n$, and the weight vector $\mathbf{w} \in \Re^n$, the weighted Minkowski distance [9] follows:

$$d(\mathbf{x}_i, \mathbf{x}_{i'}, \mathbf{w}) = \left(\sum_{j=1}^{n} w_j |x_{i,j} - x_{i',j}|^p \right)^{1/p}, \tag{4}$$

for $p \geq 1$. Well-known options for this parameter are $p \in \{1, 2, \infty\}$, i.e., the Manhattan, Euclidean, and Chebyshev distances, respectively. The weights in Eq. (4) can be obtained via an IOWA operator, leading to the induced Minkowski OWA distance (IMOWAD). Following the notation in Eq. (3), this distance function is formalized as follows:

Definition 1. *The IMOWAD distance is a mapping* $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *such that:*

$$IMOWAD(\langle u_1, x_{i,1}, x_{i',1} \rangle, \ldots, \langle u_n, x_{i,n}, x_{i',n} \rangle) = \left(\sum_{j=1}^{n} w_j b_j^p \right)^{1/p}, \tag{5}$$

where b_j *is the* $|x_{i,j} - x_{i',j}|$ *value of the 3-tuple with the* j_{th} *largest of the* u *vector (the order-inducing variable). Similar to the IOWA operator, it holds for the weight vector that* $\sum_{j=1}^{n} w_j = 1$ *and* $w_j \in [0,1]$ *[9].*

Then, the distance between two samples of the minority class $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ is given by a weighted sum of the differences between corresponding elements of these two vectors. The weights are assigned to each dimension according to how relevant the variables are based on a feature ranking technique. This ranking results in the order-inducing vector u.

Notice that the IOWA operator and the IMOWAD functions are different concepts. The IOWA operator is a very general aggregation function that confers flexibility to operators such as the weighted average. In contrast, the IMOWAD function is a distance measure that uses the IOWA operator, resulting in a general version of the weighted Minkowski distance.

3.2. The feature ranking methods for FW-SMOTE

Since the IMOWAD operator requires an order-inducing variable $\mathbf{u}$ for obtaining the weights of the IOWA operator, a feature ranking is performed using

a filter method. As it was previously discussed in Section 2.3, there are several alternatives for this step. Specifically, there are methods that rank attributes according to relevance (i.e. correlation between covariates and the target variable), such as Mutual Information (MI) and the Fisher Score (FS); to redundancy (i.e. correlations among covariates), such as the Correlation Score (CFS); or to both relevance and redundancy, such as Eigenvector Centrality (EC) and the Minimum Redundancy Maximum Relevance (MRMR) method.

Since this research study is focused on the preprocessing step, only filtering techniques are presented. From them, we selected four different scoring functions. The first two are based on assessing the dependency between the covariates and the label vector; the next one computes the redundancy degree among the variables; and, finally, the fourth one represents a synergy among all the previous ones. Below, the four filtering techniques are enumerated:

1. *Fisher Score* [10]: It computes the absolute difference between the means of the two classes, normalized with a joint standard deviation, as follows:

$$FS(j) = \frac{|\mu_{j1} - \mu_{j2}|}{\sigma_{j1}^2 + \sigma_{j2}^2}, \tag{6}$$

   where μ_{jl} is the mean value for the j-th attribute and class l, $l = 1, 2$, while σ_{jl} is its respective standard deviation.

2. *Mutual Information* [38]: It computes the amount of information on one attribute that can be gained by observing another one, as follows:

$$MI(j) = \sum_{y \in \mathbf{y}} \sum_{x \in \mathbf{x}_j} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right), \tag{7}$$

   where x and y are the various levels of attribute $\mathbf{x}_j$ and the target vector $\mathbf{y}$, respectively; whereas $p(x)$ and $p(y)$ are their marginal probability distributions, with $p(x, y)$ being their joint distribution. As can be seen, this approach assumes that the covariates are nominal variables, unlike the Fisher Score. Mutual Information, however, can be used with numerical variables after binning them [39].

3. *Correlation Score* [6]: It computes the Pearson correlation $\rho_{j,j'}$ for each pair of attributes j and j', and subsequently computes the lowest absolute correlation, as follows:

$$CFS(j) = \min_{j'} |\rho_{j,j'}|. \tag{8}$$

4. *Eigenvector Centrality* [40]: It combines the Fisher Score, the Mutual Information, and the covariates' standard deviation to construct an adjacency matrix A. Then, feature importance is assessed by computing the eigenvector related to the largest eigenvalue of A. The edges of A can be seen as the influence of two attributes that are used together for classification based on the metrics that were mentioned previously.

3.3. The FW-SMOTE algorithm

Having defined our generalized distance metric and the feature ranking methods used as an input for this metric, we now describe an algorithm that conducts data oversampling on the basis of this distance measure. The FW-SMOTE algorithm is formalized in Algorithm 1.

Algorithm 1: FW-SMOTE method for data oversampling

Input: Training tuples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$; Minority class sample set $\mathcal{M}$; Amount of oversampling N; Number of nearest neighbors k; Number of selected attributes r or threshold ϵ; Minkowski distance parameter p; IOWA quantifier parameter α.

Output: Oversampled set of elements of the minority class $\mathcal{M}^*$.

1. $\mathcal{M}^* \leftarrow \mathcal{M}$.
2. $\mathbf{u} \leftarrow$ Feature ranking method $FR(\{(\mathbf{x}_i, y_i)\}_{i=1}^{m})$.
3. $\mathbf{w} \leftarrow$ RIM quantifier $RIM(\mathbf{u}, \alpha)$.
4. $\mathcal{S} \leftarrow$ Take the r largest values of $\mathbf{w}$, or $\{j \in \mathcal{X} | w_j > \epsilon\}$.
5. **for** $i \in \mathcal{M}$
6. **for** $i' \in \mathcal{M},\ i \neq i'$
7. $IMOWAD(\mathbf{x}_i, \mathbf{x}_{i'}, \mathbf{w}) = \left(\sum_{j \in \mathcal{S}} w_j b_j^p\right)^{1/p}$.
8. **end for**
9. $\mathcal{T} \leftarrow \arg\min_{\mathcal{T}} \sum_{i' \in \mathcal{T}} IMOWAD(\mathbf{x}_i, \mathbf{x}_{i'}, \mathbf{w}),\ \mathcal{T} \subseteq \mathcal{M} \setminus \{i\},\ |\mathcal{T}| = k$.
10. **for** $k \leftarrow 1$ **to** N
11. $\mathbf{x}_k \leftarrow$ Select a random sample from $\mathcal{T}$.
12. $\mathbf{x}_k^* \leftarrow \mathbf{x}_i + v(\mathbf{x}_k - \mathbf{x}_i)$.
13. $\mathcal{M}^* \leftarrow (\mathcal{M}^*, \mathbf{x}_k^*)$.
14. $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\mathbf{x}_k\}$.
15. **end for**
16. **end for**

Assuming a two-class problem with objects $\mathbf{x}_i \in \Re^n$, $i = 1, \ldots, m$, and their respective outputs $y_i \in \{-1, +1\}$, FW-SMOTE first computes the k nearest neighbors for each element that belongs to the set of minority class samples $\mathcal{M}$. For this step, which is done using the Euclidean norm in the classic SMOTE

method and most of its variations, we propose using the IMOWAD operator presented in Eq. (5).

In Step 1, the oversampled minority class set $\mathcal{M}^*$ is initialized as the original set of minority samples in $\mathcal{M}$. Next, the order-inducing variable $\mathbf{u}$ required by IMOWAD distance is computed by means of the feature ranking chosen for this task (see Step 2 of Algorithm 1).

In Step 3, the weights for the IMOWAD distance are obtained from the order-inducing variable $\mathbf{u}$. For this step, we explore four variants of the Regular Increasing Monotone (RIM) quantifier, where α is an input parameter. This strategy, also known as fuzzy linguistic quantifiers, are arguably the best-known approach for obtaining the OWA weighting vectors [41]. The quantifiers used in this study are the following:

- Basic Regular Increasing Monotone (RIM) quantifier:

$$w_j = \left(\frac{j}{n}\right)^{\alpha} - \left(\frac{j-1}{n}\right)^{\alpha} \quad \forall\, j. \tag{9}$$

- Quadratic RIM quantifier [42]:

$$w_j = \left(\frac{1}{1-\alpha\left(\frac{j}{n}\right)^{0.5}}\right) - \left(\frac{1}{1-\alpha\left(\frac{j-1}{n}\right)^{0.5}}\right) \quad \forall\, j. \tag{10}$$

- Exponential RIM quantifier:

$$w_j = e^{-\alpha\left(\frac{j}{n}\right)} - e^{-\alpha\left(\frac{j-1}{n}\right)} \quad \forall\, j. \tag{11}$$

- Trigonometric RIM quantifier:

$$w_j = \arcsin\left(\alpha\left(\frac{j}{n}\right)\right) - \arcsin\left(\alpha\left(\frac{j-1}{n}\right)\right) \quad \forall\, j. \tag{12}$$

Step 4 performs feature selection within the SMOTE strategy, defining a subset $\mathcal{S} \subseteq \mathcal{X}$ of relevant variables, in which $\mathcal{X}$ represents the full set of variables. This subset is obtained from the weights computed in the previous step.

The feature selection step is suggested in order to alleviate the curse of dimensionality when facing high-dimensional datasets. For this step, the literature on feature selection offers two alternatives. First, a predefined threshold ϵ can be defined, and those attributes whose weights are below ϵ can be discarded. Alternatively, a target number of ranked variables $r \leq n$ can be defined. In this case, those attributes that are ranked among the $n-r$ variables with the lowest weights are discarded (see Step 4 of Algorithm 1).

As mentioned in the introduction, FW-SMOTE is not designed as a feature selection method for the classification task. However, it can be used for this purpose, taking advantage of the feature ranking method. In other words, the

use of the set $\mathcal{S}$ for classification instead of using the full set of attributes is up to the modeler.

Steps 5 to 15 of Algorithm 1 are the core of the FW-SMOTE method. For each minority sample in $\mathcal{M}$ (Step 5), its neighborhood of size k is computed using the IMOWAD function (Step 6 to 9). Set $\mathcal{T}$ includes the k nearest neighbors of the target sample i.

Similar to the SMOTE algorithm, FW-SMOTE selects the $N < k$ neighbors randomly once the neighborhood is defined. The values for k and N must be defined beforehand. Step 11 selects a random sample $\mathbf{x}_k$ from $\mathcal{T}$, while Step 12 creates a synthetic examples $\mathbf{x}_k^*$ via interpolation with the target sample. For this step, the algorithm requires a random number υ between 0 and 1.

In Step 13, the synthetic example $\mathbf{x}_k^*$ is appended to $\mathcal{M}^*$. Next, the randomly selected sample is $\mathbf{x}_k$ is excluded from $\mathcal{T}$ (see step 14 of Algorithm 1). This process is repeated N times for each of the minority examples.

The output of the algorithm is then the oversampled minority class $\mathcal{M}^*$ that includes all the original samples in $\mathcal{M}$ and the synthetic examples.

Our approach is a generalized version of the SMOTE algorithm since the IMOWAD operator becomes the traditional Euclidean norm when $w_j = 1/n$ for all j, $r = n$ (no feature selection), and $p = 2$.

3.4. Theoretical analysis and contribution

Our main contribution is the proposed weighting strategy for the definition of the neighborhood in the SMOTE algorithm. It is important to notice that the proposed metric (the IMOWAD distance) was not considered previously in the context of machine learning. Furthermore, the feature ranking methods have not been considered in previous studies with the purpose of developing a weighted distance metric. These elements make FW-SMOTE a non-trivial oversampling strategy with a lot of potential since it generalizes the original SMOTE algorithm.

The introduction of a weighting process in the construction of distance functions showed good empirical results in unsupervised learning, as suggested in [43]. In this study, the authors report an important improvement in the description and interpretability of data from molecular dynamics (MD) simulations. This is a very relevant high-dimensional task designed to describe biomolecules observed in time. Although this approach considers a completely different weighting strategy and it is designed for a different purpose, it supports our hypothesis that a weighted scheme can improve distance-based learning strategies in high-dimensional settings.

Our proposal is essentially similar to SMOTE in terms of complexity. FW-SMOTE is based on fast feature ranking methods, such as Fisher Score, and involves additional steps which are very fast arithmetic operations: computing the RIM function in order to obtain the weight vector, sorting the output of the feature ranking method, and finally the inclusion of the weights in the distance function. The sole exception to this is the use of Eigenvector Centrality as feature ranking method, which is more time-consuming than the alternatives

since it requires the other feature ranking methods as input, and it performs the computation of eigenvalues for the adjacency matrix.

The following remarks relate the proposed approach with the use of oversampling in combination with feature extraction methods:

Remark 1. *There is an interesting relation between FW-SMOTE and the application of SMOTE oversampling on an embedding space. Feature extraction strategies such as Locally Linear Embedding (LLE) have some mathematical similarities with FW-SMOTE: it constructs a k-NN graph and defines weights to multiply the input variables. However, this is done with a completely different purpose: to map the data into a low-dimensional manifold, where the patterns become more distinguishable [44].*

Remark 2. *FW-SMOTE is flexible enough to be able to resemble (at least partially) a feature extraction method. In particular, the correlation score (Eq.* (8)*) weighs down attributes that are uncorrelated with others in the data matrix* $\boldsymbol{X}$*. Alternatively, Eigenvector Centrality integrates different strategies that can be linked with feature extraction, such as the computation of eigenvectors. Nevertheless, we believe that there are important differences between FW-SMOTE and studies that apply SMOTE on the embedding space:*

(a) Casting the original space into a lower-dimensional feature space is important in many applications, such as computer vision or text analytics. However, in many others, such as in business analytics, this step seldom brings an improvement in prediction. In tabular datasets, the correlations between variables are usually not strong and the predictive performance is linked to the relationship between each one of the original variables and the labels.

(b) Our weighting strategy is directly linked to the original variables, allowing the possibility of performing feature selection. In contrast, feature extraction methods perform combinations of the inputs, making it impossible to perform feature selection in most cases.

(c) The correlation-based weighting approach (FW-SMOTE with correlation score as feature ranking method) showed a worse performance in comparison to relevance-based feature ranking techniques (Fisher Score or Mutual Information). This confirms that the virtue of FW-SMOTE relies on the proper definition of the neighborhood by assessing the individual contribution of the original features, at least in the benchmark datasets considered in this study.

(d) Feature extraction techniques are usually more time-consuming than the feature ranking techniques recommended in this study. Therefore, the overall complexity of an approach that performs SMOTE on the embedding space would be larger than the one of FW-SMOTE.

(e) There are ad-hoc embedding strategies depending on the task at hand. It is not evident which feature extraction strategy works best in a given application. These issues are task-dependent, while FW-SMOTE can be applied in any domain.

We strongly believe that performing SMOTE in a feature embedding space has many advantages, as shown in [44], and it would perform better than FW-SMOTE in some domains. However, it represents a different strategy for addressing the class-imbalance problem. In order to support this claim, we evaluate empirically the use of Principal Component Analysis (PCA) for feature extraction in combination with SMOTE oversampling, as suggested in [45], on low and high-dimensional datasets.

4. Experimental Results

We applied the proposed FW-SMOTE and alternative oversampling approaches to well-known benchmark datasets with a wide range of sample sizes and imbalance ratios. In Section 4.1 we present a description of all the datasets and the experimental setting, while Section 4.2 provides a summary of the performance obtained for all techniques. Next, the influence of the different parameters related to our approach is analyzed in Section 4.3. Finally, the adaptation of the FW-SMOTE method as a feature selection strategy is discussed in Section 4.4, comparing it with alternative feature selection approaches that deal with the class-imbalance problem.

4.1. Experimental Framework and Available Datasets

Of the 42 datasets used for benchmarking, 12 are high-dimensional microarray datasets reported in Maldonado et al. [6], while the remaining 30 are low-dimensional applications from the UCI (`http://archive.ics.uci.edu/`) and KEEL (`http://sci2s.ugr.es/keel`) data repositories.

Regarding the relevant information of the various datasets, Table 1 summarizes the relevant metadata, such as Imbalance Ratio (IR), Number of Attributes (#Att), Number of Samples (#Samples), and Percentage of Samples in each class (min.,maj.) for all datasets.

The following alternative approaches were considered in the experimental analysis: the classic SMOTE [16]; the SMOTE variations Borderline-SMOTE [20], SL-SMOTE, [22], ADASYN [18], Adaptive Neighbor SMOTE [19], Density Based SMOTE [21], MWMOTE [24], Relocating Safe-level SMOTE [23], the RBO technique [25], and PCA in combination with standard SMOTE [45]. For the latter method, the number of extracted components was decided in such a way that at least 90% of the variance was explained with the selected features, as suggested in [45].

A major challenge in class-imbalance classification is the choice of the performance measure [46, 47]. Ten-fold cross-validation was conducted using AUC (Area Under the Curve) as the performance metric to perform model validation [47]. We also included a metric based on the binary confusion matrix that is

Table 1: Relevant metadata for all datasets.

ID	Dataset	IR	#Att	#Samples	%class(min.,maj.)
Low-dimensional datasets					
ldd1	abalone7	9.7	8	4177	(9.3, 90.7)
ldd2	ecoli	8.6	7	336	(10.4, 89.0)
ldd3	ecoli-0-1 vs 5	11	6	240	(8.3, 91.7)
ldd4	ecoli-0-1-4-6 vs 5	13	6	280	(7.1, 92.9)
ldd5	ecoli-0-1-4-7 vs 5-6	12.28	6	332	(7.5, 92.5)
ldd6	ecoli-0-2-3-4 vs 5	9.1	7	202	(9.9, 90.1)
ldd7	ecoli-0-2-6-7 vs 3-5	9.18	7	224	(9.8, 90.2)
ldd8	ecoli-0-3-4 vs 5	9	7	200	(10.0, 90.0)
ldd9	ecoli-0-3-4-6 vs 5	9.25	7	205	(9.8, 90.2)
ldd10	ecoli-0-3-4-7 vs 5-6	9.28	7	257	(9.7, 90.3)
ldd11	ecoli-0-4-6 vs 5	9.15	6	203	(9.9, 90.1)
ldd12	ecoli-0-6-7 vs 3-5	9.09	7	222	(9.9, 90.1)
ldd13	ecoli-0-6-7 vs 5	10	6	220	(9.1, 90.9)
ldd14	ecoli4	13.84	7	336	(6.7, 93.3)
ldd15	glass-0-1-6 vs 2	10.29	9	192	(8.9, 91.1)
ldd16	glass-0-4 vs 5	9.22	9	92	(9.8, 90.2)
ldd17	glass-0-6 vs 5	11	9	108	(91.7, 8.3)
ldd18	glass4	15.47	9	214	(6.1, 93.9)
ldd19	image1	6	19	2310	(14.3, 85.7)
ldd20	page-blocks-1-3 vs 4	15.85	10	472	(5.9, 94.1)
ldd21	shuttle-c0-vs-c4	13.87	9	1829	(6.7, 93.3)
ldd22	solar	19.4	10	1389	(4.9.95.1)
ldd23	yeast-0-2-5-7-9 vs 3-6-8	9.14	8	1004	(9.9, 90.1)
ldd24	yeast-0-5-6-7-9 vs 4	9.35	8	528	(9.7, 90.3)
ldd25	yeast-1 vs 7	13.87	7	459	(6.7, 93.3)
ldd26	yeast-1-4-5-8 vs 7	22.1	8	693	(4.3, 95.7)
ldd27	yeast-2 vs 4	9.08	8	514	(9.9, 90.1)
ldd28	yeast3	8.1	8	1484	(11.0, 89.0)
ldd29	yeast4	28.1	8	1484	(3.4, 96.6)
ldd30	yeast5	32.78	8	1484	(3.0, 97.0)
High-dimensional datasets					
hdd1	bhat1	9.15	3312	203	(9.9,90.1)
hdd2	bhat2	9.15	3312	203	(8.4,91.6)
hdd3	bhat3	9.15	3312	203	(10.3,89.7)
hdd4	bhat4	9.15	3312	203	(3.0,97.0)
hdd5	bullinger	11.25	17,404	98	(8.2, 91.8)
hdd6	car1	14.8	9182	174	(6.3, 93.7)
hdd7	car2	14.8	9182	174	(6.9, 93.1)
hdd8	car3	14.8	9182	174	(4.0, 96.0)
hdd9	car4	14.8	9182	174	(3.4, 96.6)
hdd10	glioma	6.14	4434	50	(14.0,86.7)
hdd11	lung	4.85	12,533	181	(17.1,82.9)
hdd12	srbct	6.55	2308	83	(13.2, 86.8)

able to provide a balance between the two class accuracies, as suggested in [46]. Therefore, the G-mean, computed as the geometric mean of the sensitivity and the specificity [46], is also reported for completeness.

The following classification techniques are used after the data resampling method is applied: k-Nearest Neighbors (k-NN), Logistic Regression (LR), and linear Support Vector Machine (SVM). While logistic regression does not require parameter setting in its formulations, the values for k and C must be defined for k-NN and SVM, respectively. We used $k = 5$ and $C = 1$ for these

methods, since they are suggested in the literature as good default values for these methods [6]. For all the oversampling methods, the number of neighbors was set to $k = 5$, as suggested in the original paper by Chawla et al. [16]. We selected $N = 1$ and $N = 3$ objects from these five neighbors (100% and 300% oversampling, respectively). For the proposed method, the following values for r (the cardinality of the subset of selected features), p (the Minkowski distance parameter), and α (the OWA quantifier parameter) were explored: $r \in \{\frac{n}{2}, \frac{3n}{4}, n\}$, $p \in \{1, 2, \infty\}$ (Manhattan, Euclidean, and Chebyshev norms, respectively), and $\alpha \in \{0.4, 0.6\}$. Different combinations of these parameters were explored using grid search.

The parameter tuning procedure was performed within the training set for the various parameters included in our proposal. The only "parameter" choice that was made based on the average test performance was the amount of oversampling N. For all methods, we selected the best performance between these two alternatives. The comparison between all models is fair since the test set remains unseen until the final performance assessment for all models.

Regarding model implementation, our proposal was developed in Matlab, and the codes are available at `http://github.com/cvairetti`. The SMOTE, BL-SMOTE, and SL-SMOTE variants were implemented in Matlab by Iman Nekooeimehr (`http://github.com/Nekooeimehr`), while the remaining SMOTE variants were developed in R using the *imbalance* package by Ignacio Cordón [48] (`http://github.com/ncordon`).

4.2. Performance analysis

Tables 2 and 3 summarize the results obtained for oversampling methods using k-NN, SVM, and LR as classification models, and for both low- and high-dimensional datasets, respectively. For each oversampling technique (presented in ascending order of average rank) and classification method, these tables include the following information:

- The average rank computed by the Friedman test with Iman-Davenport correction. This test was used to assess whether or not all ranks are statistically similar [49]. This is a common approach for assessing classification performance among various supervised learning methods. For each classification method, the average rank is computed based on the AUC value on all the datasets for each oversampling approach.

- The average AUC x100 and G-mean x100 with their corresponding standard deviations.

- The p-values obtained by the Holm test and the outcome of the test. This test was suggested in [49] to use in case the Friedman test is rejected, which is our case. This test compares the pairwise performance between each oversampling approach and the one with the best rank. The outcome is 'reject' when this p-value falls below a threshold $\beta/(j-1)$, with $\beta = 5\%$ and $j = 2, \ldots, 11$ being the overall ranking for a given oversampling method. This outcome implies that the corresponding SMOTE variant is

outperformed by the one with the best rank. In this case, the p-value is highlighted with an asterisk.

- The number of wins/ties/loses (W/T/L) for each method in comparison with the one with the highest rank. This analysis is presented for both performance measures (AUC and G-mean).

The values for the Friedman F tests with Iman-Davenport correction for the low dimensional datasets with the AUC measure are 13.28, 7.99, and 9.11 for the k-NN, SVM, and LR methods, respectively, suggesting that the null hypothesis of equal ranks can be rejected for all methods with p-values below 0.01. For the high-dimensional datasets, the values for this test are 6.26, 6.75, and 5.82 for the k-NN, SVM, and LR methods, respectively, suggesting that the null hypothesis of equal ranks can also be rejected for the k-NN, SVM, and LR approaches with p-values below 0.01.

Table 2: Holm's post-hoc test for pairwise comparisons. Low-dimensional datasets.

Method	Ranking	AUC	W/T/L (AUC)	p-value (Holm test)	G-mean	W/T/L (G-mean)
k-nearest neighbors						
FW-SMOTE	1.600	88.2±9	-	-	83±19.5	-
SMOTE	5.300	85.5±11.1	0/2/28	< 0.001*	80.3±20	1/6/23
AN-S.	5.700	84.2±12.1	1/4/25	< 0.001*	77.9±22.7	1/5/24
MWMOTE	5.867	84.7±12.4	0/4/26	< 0.001*	79.3±22.3	1/4/25
BL-S.	5.883	85.6±10.5	0/1/29	< 0.001*	79.1±21.5	0/2/28
RBO	6.000	85.5±10.8	0/2/28	< 0.001*	82.8±15.4	2/2/26
SL-S.	6.050	84.5±11.6	0/5/25	< 0.001*	77.1±24	0/4/26
RSL-S.	6.483	83.7±13.5	0/7/23	< 0.001*	75.7±25.5	0/6/24
ADASYN	6.483	83.8±12.4	0/4/26	< 0.001*	77.4±23.4	2/2/26
DB-S.	7.533	83.1±12.7	0/4/26	< 0.001*	76.1±23.8	1/3/26
PCA-S.	9.100	72.4±11.8	0/0/30	< 0.001*	62.2±57.5	0/0/30
Support Vector Machine						
FW-SMOTE	2.600	84.6±13.4	-	-	78.5±25.3	-
SL-S.	5.400	83.2±13	2/2/26	0.001*	75.5±26.9	2/4/24
SMOTE	5.533	82.7±13.9	1/3/26	0.001*	75.3±27.8	1/3/26
AN-S.	5.733	82.7±12.8	1/3/26	< 0.001*	77.3±22.6	3/2/25
BL-S.	6.033	82.2±14.5	2/1/27	< 0.001*	74.5±30.1	3/1/26
MWMOTE	6.050	82.5±13.1	1/2/27	< 0.001*	76.7±24	2/3/25
RSL-S.	6.050	82.4±12.5	1/2/27	< 0.001*	76.6±22.3	2/2/26
RBO	6.183	82.4±13.6	0/2/28	< 0.001*	74.8±27.4	0/3/27
DB-S.	6.667	82.1±13.4	2/2/26	< 0.001*	75.7±24.4	1/2/27
ADASYN	6.833	81.6±13.1	2/1/27	< 0.001*	75.2±23.6	3/1/26
PCA-S.	8.917	63.8±16.7	1/0/29	< 0.001*	36.2±42.5	1/1/28
Logistic regression						
FW-SMOTE	3.267	85.4±10.2	-	-	82.2±17.5	-
SMOTE	5.083	84.8±9.8	2/3/25	0.034*	80±19.2	1/8/21
MWMOTE	5.267	83.5±11.8	4/3/23	0.020*	77.7±21.5	1/4/25
SL-S.	5.550	84.4±10	1/3/26	0.008*	80.6±15.8	2/5/23
BL-S.	5.617	84.2±10	4/2/24	0.006*	79.8±17.7	4/3/23
AN-S.	5.917	83.3±11.7	0/5/25	0.002*	78±20.9	1/4/25
RSL-S.	6.033	82.7±12.5	2/5/23	0.001*	75.7±23.7	1/5/24
RBO	6.083	83.3±11.6	3/4/23	0.001*	77.9±20.9	0/4/26
DB-S.	6.433	82.8±11.8	2/4/24	< 0.001*	76.7±21.5	1/4/25
ADASYN	7.250	81.9±11.8	1/3/26	< 0.001*	75.8±21.1	2/5/23
PCA-S.	9.500	64.4±15.2	0/1/29	< 0.001*	42.9±47.2	0/1/29

Table 3: Holm's post-hoc test for pairwise comparisons. High-dimensional datasets.

Method	Ranking	AUC	W/D/L	APV (Holm test)	G-mean	W/D/L
k-nearest neighbors						
FW-SMOTE	2.708	92.7±10.5	-	-	95.7±23.9	-
AN-S.	4.792	91.3±10.2	1/3/8	0.124	93.6±26.2	2/1/9
RSL-S.	5.083	91.3±10.3	1/2/9	0.079	92.8±28.5	1/1/10
ADASYN	5.292	90.9±10.1	2/1/9	0.056	92.2±26.5	2/2/8
DB-S.	5.375	90.9±10.6	0/2/10	0.049	92.4±27.2	1/1/10
MWMOTE	5.458	91.1±10.8	0/2/10	0.042	93.6±25.8	1/1/10
SL-S.	6.417	90.4±10.9	0/1/11	0.006*	92.4±26.9	0/2/10
SMOTE	6.625	90.5±10.8	0/2/10	0.004*	91.8±23.8	0/2/10
RBO	7.042	89.4±9.8	0/2/10	0.001*	90.9±20.2	1/1/10
BL-S.	7.125	90.5±10.8	0/1/11	0.001*	92.5±25	0/1/11
PCA-S.	10.083	65.4±17.5	1/0/11	< 0.001*	19.9±22.6	1/0/11
Support Vector Machine						
BL-S.	5.458	88±13.1	0/12/0	-	87.2±31.9	0/8/4
FW-SMOTE	5.458	88±13.1	0/12/0	-	90±28.4	0/8/4
RBO	5.458	88±13.1	0/12/0	-	87.2±30	0/8/4
SL-S.	5.458	88±13.1	0/12/0	-	87.2±29.6	0/8/4
SMOTE	5.458	88±13.1	0/12/0	-	87.2±30.3	0/8/4
DB-S.	5.750	87.4±12.8	0/11/1	0.829	87.2±27.6	0/8/4
MWMOTE	5.750	87.4±12.8	0/11/1	0.829	87.2±27.1	0/8/4
ADASYN	5.917	87.4±12.8	0/11/1	0.735	86.4±26.6	0/7/5
RSL-S.	5.917	87.4±12.8	0/11/1	0.735	87.2±26	0/8/4
AN-S.	6.042	87.3±12.8	0/11/1	0.667	87.2±26	0/8/4
PCA-S.	9.333	66.7±18.8	2/0/10	0.004*	38.1±37.0	1/0/11
Logistic regression						
FW-SMOTE	1.083	91.3±11.3	-	-	92.9±22.3	-
MWMOTE	4.958	83.7±12.7	0/0/12	0.004*	75.4±27.3	0/0/12
DB-S.	5.125	83.2±15.6	0/0/12	0.003*	73.8±28	0/0/12
SMOTE	5.625	82.4±13.8	0/0/12	0.001*	77.9±25	0/0/12
AN-S.	6.250	81.6±14.9	0/0/12	< 0.001*	75.2±27.4	0/0/12
RBO	6.250	81.7±16	0/0/12	< 0.001*	68.6±28.1	0/0/12
SL-S.	6.375	82±16	0/0/12	< 0.001*	74.8±24.7	0/0/12
ADASYN	7.125	80.9±14.6	0/0/12	< 0.001*	70.7±26.3	0/0/12
RSL-S.	7.125	79.5±14.7	0/0/12	< 0.001*	70±28.6	0/0/12
BL-S.	7.583	80.6±16	0/0/12	< 0.001*	74.7±24.9	0/0/12
PCA-S.	8.500	66.4±17.1	1/0/11	< 0.001*	32.2±32.9	1/0/11

It can be observed in tables 2 and 3 that the proposed FW-SMOTE clearly outperforms the other SMOTE variations on both low and high-dimensional datasets. The only exception is the SVM case on high-dimensional datasets, in which most oversampling methods are equally good. This can be due to the small number of minority samples in the microarray datasets, leading to relatively similar sets of oversampled objects. Furthermore, the use of unregularized classification approaches on high-dimensional datasets may explain the small differences in methods such as SVM.

Regarding the use of a feature extraction method such as PCA in combination with SMOTE oversampling, we observe in tables 2 and 3 that this approach has the worse average rank and average predictive performance in terms of AUC and G-mean. Furthermore, it is statistically outperformed by the proposed FW-SMOTE in all six cases. We can conclude that the use of a feature extraction method in combination with SMOTE is not recommended in tabular datasets with a low degree of redundancy, being able to achieve best performance only

on a few occasions.

The mean AUC and G-mean values confirm the superiority of the proposed method in terms of average predictive performance. Best average performance is achieved with the k-NN classifier, however, our approach for feature selection and data resampling lead to competitive results with logistic regression, which has the additional virtue of being interpretable.

The detailed results for each method and dataset are reported in Appendix A, presented as supplementary material. Tables A.1 to A.3 consider AUC as performance measure, while tables A.4 to A.6 use G-mean.

4.3. Influence of the FW-SMOTE parameters

As mentioned above, the proposal is a very general version of SMOTE oversampling, which allows several alternatives for feature weighting and selection thanks to the use of OWA operators. Table 4 presents the percentage of times a given parameter leads to the largest AUC with the FW-SMOTE strategy and the four classification approaches. We note that four feature ranking strategies were explored (Fisher Score, Mutual information, Eigenvector Centrality, and Correlation Score), three alternatives for the number of ranked features ($r \in \{\frac{n}{2}, \frac{3n}{4}, n\}$), three variants for the Minkowski distance ($p \in \{1, 2, \infty\}$, i.e. Manhattan, Euclidean, and Chebyshev norms, respectively), four OWA quantifiers (Basic RIM, Quadratic RIM, Exponential RIM, and Trigonometric RIM), and two values for the OWA quantifier parameter ($\alpha \in \{0.4, 0.6\}$). Notice that when two FW-SMOTE variants achieve the same performance, the first one is considered to be the one with the best performance in the ordering provided in Table 4.

According to Table 4, the first alternatives are usually the ones that achieved the best results. Fisher Score is the recommended feature selection strategy, while the basic RIM quantifier is the suggested OWA approach. Notice that the best performance is seldom achieved with $r = n$, suggesting that the feature selection step is very important for the algorithm, and confirming the results obtained in [6]. Finally, the Chebyshev norm is preferred among the various alternatives for p. Theoretically speaking, the Chebyshev provides a better definition of distance in high-dimensional settings when compared with the Manhattan and Euclidean norms [6], and this result confirms this point. Nevertheless, there is a large number of ties in terms of performance for all FW-SMOTE variants. On the one hand, it suggests that the results are very stable, and the influence of the various parameters is not strong. On the other hand, this analysis must be used with caution in the sense that most variants achieve a similar performance.

Notice that the results obtained with the proposal were obtained via grid search based on the parameter set reported in Section 4.1. Table 4 is constructed by taking into account the best parameter configuration with this procedure for each dataset and classification method. The percentage of times a given parameter lead to the largest AUC (the values in Table 4) were computed using this information. In other words, only the optimal parameter configuration is used in Table 4.

Table 4: Percentage of times a given parameter performs best.

	k-NN	SVM	LR
Feature ranking method			
Fisher	46.7	64.4	44.5
Mutual Inf.	16.7	11.1	21.1
Eig. Centrality	16.7	7.8	16.7
Correlation	20.0	16.7	17.8
Number of ranked features			
$r = n/2$	58.9	67.8	50.0
$r = 3n/4$	22.2	15.6	28.9
$r = n$	18.9	16.7	21.1
Minkowski distance parameter			
$p = \infty$	55.6	55.6	57.8
$p = 1$	21.1	30.0	21.1
$p = 2$	23.3	14.4	21.1
OWA quantifier			
Basic RIM	57.8	74.4	63.3
Quadratic	15.6	10.0	20.0
Exponential	12.2	7.8	7.8
Trigonometric	14.4	7.8	8.9
Quantifier parameter			
$\alpha = 0.4$	71.1	83.3	76.7
$\alpha = 0.6$	28.9	16.7	23.3

4.4. Feature selection results

As it was previously pointed out, FW-SMOTE is designed to define a better neighborhood for the oversampling process via feature weighting. In case a variable has a very low weight, it will be ignored in the definition of the neighborhood. However, the method does not remove it from the classification task as it is designed to be an intelligent oversampling comparable to any other SMOTE variant. Nevertheless, we can utilize the feature weighting strategy and allow the complete exclusion of the covariates that receive a low weight in the IOWA function. Table 5 reports a new set of experiments in which feature selection is incorporated in the learning process.

Table 5: Holm's post-hoc test for pairwise comparisons. Feature selection.

Method	Ranking	AUC	W/T/L	APV (Holm test)	G-mean	W/T/L
k-nearest neighbors						
FW-SMOTE	2.833	96.8±3.6	-	-	96.8±3.7	-
Fisher+SMOTE	5.344	90.3±7.1	3/0/45	0.002*	89.7±7.8	3/0/45
Relief+SMOTE	5.927	89.6±7.3	0/0/48	< 0.001*	89.2±7.9	0/0/48
Fisher+BL-S.	6.240	92.8±7.2	6/9/33	< 0.001*	92.3±7.9	5/9/34
Fisher+SL-S.	6.240	92.8±7.2	6/9/33	< 0.001*	92.3±7.9	5/9/34
Relief+BL-S.	6.313	94.1±5.6	4/12/32	< 0.001*	94±5.8	6/10/32
Relief+SL-S.	7.115	90±7	1/0/47	< 0.001*	89.6±7.6	1/0/47
CFS+BL-S.	7.333	91.2±7.6	2/0/46	< 0.001*	90.8±8.4	2/0/46
CFS+SL-S.	8.281	92.9±6	6/4/38	< 0.001*	92.5±6.4	7/3/38
CFS+SMOTE	8.448	94.2±5.4	7/5/36	< 0.001*	94±5.7	7/5/36
BFE+SL-S.	8.635	91.8±14.9	6/3/39	< 0.001*	91.4±15.1	6/3/39
BFE+SMOTE	8.917	90.2±7.4	1/0/47	< 0.001*	89.6±8.4	1/0/47
BFE+BL-S.	9.375	90.5±7	0/0/48	< 0.001*	90.1±7.7	0/0/48
Support Vector Machine						
FW-SMOTE	4.375	93.7±6.9	-	-	93.3±7.5	-
Fisher+BL-S.	5.000	93.1±7.1	2/27/19	0.432	92.6±7.7	2/27/19
Fisher+SL-S.	5.000	93.1±7.1	3/26/19	0.432	92.6±7.7	2/27/19
Fisher+SMOTE	5.042	88.4±10.8	0/12/36	0.402	87±12.8	0/12/36
Relief+SL-S.	6.417	88.4±11	0/12/36	0.01*	87±12.9	0/12/36
Relief+SMOTE	6.938	88.2±11.1	0/12/36	0.001*	86.8±13	0/12/36
Relief+BL-S.	7.448	93±7	0/26/22	< 0.001*	92.5±7.7	0/26/22
CFS+SL-S.	7.948	91.5±10.6	1/20/27	< 0.001*	90.5±13.4	1/20/27
CFS+BL-S.	8.219	90.2±9.1	4/10/34	< 0.001*	89.2±10.8	4/10/34
CFS+SMOTE	8.333	91.2±10.2	1/17/30	< 0.001*	90.2±12.6	2/16/30
BFE+SL-S.	8.656	89.4±16	1/16/31	< 0.001*	88.5±17	1/16/31
BFE+SMOTE	8.708	89.8±9.4	4/10/34	< 0.001*	88.7±11.2	4/10/34
BFE+BL-S.	8.917	89.6±10.9	1/13/34	< 0.001*	88.2±13.9	1/13/34
Logistic regression						
FW-SMOTE	2.677	91.1±9.1	-	-	90.8±9.4	-
Relief+BL-S.	5.615	84.9±12.6	4/3/41	< 0.001*	84.1±13.4	4/3/41
Fisher+BL-S.	5.625	85±13.9	4/7/37	< 0.001*	84.1±15.3	4/7/37
Fisher+SL-S.	5.625	85±13.9	4/7/37	< 0.001*	84.1±15.3	4/7/37
Relief+SL-S.	5.906	79.3±13.3	3/1/44	< 0.001*	78.1±14.5	3/1/44
Relief+SMOTE	6.021	79±13.6	2/1/45	< 0.001*	77.9±14.9	2/1/45
Fisher+SMOTE	6.042	78.6±14.2	1/1/46	< 0.001*	77.3±15.9	1/1/46
BFE+BL-S.	8.656	79.3±12.3	0/2/46	< 0.001*	77.8±13	0/2/46
CFS+BL-S.	8.667	80±12.4	2/0/46	< 0.001*	79.1±13.2	2/0/46
BFE+SMOTE	8.938	78.8±12.6	0/0/48	< 0.001*	77.6±14.2	0/0/48
CFS+SMOTE	8.938	84.4±12.5	6/2/40	< 0.001*	83.5±13.7	6/2/40
BFE+SL-S.	8.990	84.1±17.4	4/4/40	< 0.001*	83.3±18	4/4/40
CFS+SL-S.	9.302	84.5±12.4	5/4/39	< 0.001*	83.5±13.3	5/5/38

For the twelve high-dimensional datasets, we select the top n^* attributes with the filter strategies proposed for FW-SMOTE (FS, MI, EC, or CFS) while performing the oversampling approach, with $n^* \in \{50, 100, 250, 1000\}$. For each feature selection+oversampling technique and for each classifier (k-NN, SVM, and LR), Table 5 reports the average rank, the average AUC x100 and G-mean x100

with their corresponding standard deviations, the number of wins/ties/loses, the p-values of the Holm test.

As alternative approaches we consider the filter methods Fisher Score, CFS, Relief, and BFE-SVM (only as a measure for feature ranking), in combination with the resampling techniques SMOTE, Borderline SMOTE (BL-S.), and SafeLevel-SMOTE (SL-S.). Relief and BFE-SVM were selected because they are relevant measures discussed in the feature selection literature for dealing with the class-imbalance problem (see Section 2.3).

The values for the Friedman tests with Iman-Davenport correction for these experiments are 13.63, 15.57, and 18.38 for the k-NN, SVM, and LR approaches, respectively. It can be concluded that the hypothesis of equal ranks can be rejected for all classifiers with p-values below 0.01.

Similar to tables 2 and 3, it can be observed that the proposed FW-SMOTE with the feature selection step outperforms the alternative approaches, being the top-ranked strategy with the three classifiers. Furthermore, feature selection improves the average performance of the resampling techniques, including our proposal. This is particularly noticeable for SVM classification. We conclude that our proposal is not only extremely useful as an oversampling technique, but also as an integrated solution for dealing with the class-imbalance problem in high-dimensional settings.

5. Conclusions

In this work, we have proposed a novel oversampling approach, called FW-SMOTE, designed to deal with imbalanced classification under the presence of irrelevant/redundant variables. The proposal uses the weighted Minkowski distance for identifying the k nearest objects in the minority class. This leads to a general and efficient SMOTE variant that is able to up-weigh relevant features for a better definition of the neighborhood.

Our experimental analysis, carried out on 42 imbalanced datasets, shows the effectiveness of this new proposal, which outperforms several state-of-the-art SMOTE-based variants under different classification algorithms. Furthermore, our strategy is also able to perform feature selection and data resampling simultaneously thanks to the feature weighting step. Experiments on high-dimensional datasets prove that FW-SMOTE outperformed two-step strategies for independent feature selection and oversampling.

We must discuss about a potential FW-SMOTE limitation regarding its large number of tuning parameters, which opens many possible combinations of parameter configurations. Exploring a wide range of values for the various parameters can be a time-consuming process since it requires successive model estimations on the resampled data. Computationally speaking, the model training step is usually more expensive than data resampling. It is important to notice that our approach is faster than most alternatives: FW-SMOTE does not consider the majority class, resulting in a relatively similar complexity in relation to SMOTE, while being faster than alternatives that consider the majority samples.

From our grid search study, we observe very stable results for most parameter configurations. We can conclude from this analysis that a default configuration can ease the model validation process when facing large datasets. Therefore, we recommend a default setting based on the best-performing parameters found with our thorough experimentation. In particular, we suggest using Fisher Score as feature ranking method because of its superior performance and efficiency.

As future work, we intend to extend this proposal to Big Data problems. In a Big Data setting, both SMOTE and our variant can be extremely time consuming or even intractable. Some solutions have been proposed, such as the use of an approximation of the distance function [50]. We believe that this line of research is extremely important for the future of pattern recognition and machine learning in general. In order to address this challenge, we plan to design and develop a smart and scalable feature ranking approach that, in synergy with a distributed oversampling solution, will be able to cope with Big Data applications. In addition, we plan to design hybrid undersampling-oversampling approaches to get the best of both worlds.

Another important avenue for research is deep learning (DL). The advent of artificial intelligence and AI has created new opportunities in domains such as medical diagnosis, computer vision, and natural language processing [51, 52]. Furthermore, deep learning tasks can also face the class imbalance problem [51, 52]. Although our FW-SMOTE method is designed to weigh to original input variables, it can be adapted to introduce weights on a feature space, making it suitable for learning machines such as DL architectures.

Acknowledgements

This research was partially funded by ANID, FONDECYT project 1200221 and 12200007, and by PIA/BASAL AFB180003. It has been also partially supported by the Spanish Ministry of Science and Technology under project PID2020-119478GB-I00, including European Regional Development Funds, and the Andalusian regional project P18-TP-5035. The authors are grateful to the anonymous reviewers who contributed to improving the quality of the original paper.

References

[1] K. Oksuz, B. C. Cam, S. Kalkan, E. Akbas, Imbalance problems in object detection: A review, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2021) 3388–3415.

[2] A. N. Tarekegn, M. Giacobini, K. Michalak, A review of methods for imbalanced multi-label classification, Pattern Recognition 118 (2021) 107965.

[3] X. Liang, A. Jiang, T. Li, Y. Xue, G. Wang, Lr-smote—an improved unbalanced data set oversampling based on k-means and svm, Knowledge-Based Systems 196 (2020) 105845.

[4] F. Thabtah, S. Hammoud, F. Kamalov, A. Gonsalves, Data imbalance in classification: Experimental evaluation, Information Sciences 513 (2020) 429–441.

[5] A. Fernandez, S. Garcia, F. Herrera, N. V. Chawla, SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary, Journal of Artificial Intelligence Research 61 (2018) 863–905.

[6] S. Maldonado, J. López, C. Vairetti, An alternative smote oversampling strategy for high-dimensional datasets, Applied Soft Computing 76 (2019) 380–389.

[7] Q. Wu, Y. Lin, T. Zhu, Y. Zhang, Hiboost: A hubness-aware ensemble learning algorithm for high-dimensional imbalanced data classification, Journal of Intelligent & Fuzzy Systems (2020) 1–12.

[8] S. Maldonado, J. M. Merigo, J. Miranda, Iowa-svm: A density-based weighting strategy for svm classification via owa operators, IEEE Transactions on Fuzzy Systems 28 (2020) 2143 – 2150.

[9] J. M. Merigó, M. Casanovas, A new minkowski distance based on induced aggregation operators, International Journal of Computational Intelligence Systems 4 (2011) 123–133.

[10] M. Li, H. Wang, L. Yang, Y. Liang, Z. Shang, H. Wan, Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction, Expert Systems with Applications 150 (2020) 113277.

[11] A. Fernandez, S. Garcia, M. Galar, R. C. Prati, B. Krawczyk, F. Herrera, Learning from Imbalanced Data Sets, Springer, 2018.

[12] S. Maldonado, J. López, Dealing with high-dimensional class-imbalanced datasets: embedded feature selection for SVM classification, Applied Soft Computing 67 (2018) 94–105.

[13] C. Jimenez-Castano, A. Alvarez-Meza, A. Orozco-Gutierrez, Enhanced automatic twin support vector machine for imbalanced data classification, Pattern Recognition 107 (2020) 107442.

[14] B. Richhariya, M. Tanveer, A reduced universum twin support vector machine for class imbalance learning, Pattern Recognition 102 (2020) 107150.

[15] I. Nekooeimehr, S. Lai-Yuen, Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets, Expert Systems With Applications 46 (2016) 405–416.

[16] N. V. Chawla, L. Hall, K. Bowyer, W. Kegelmeyer, SMOTE: synthetic minority oversampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[17] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, Information Sciences 250 (2013) 113–141.

[18] H. He, Y. Bai, E. Garcia, S. Li, ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: Proceedings of the IEEE International Joint Conference on Computational Intelligence IJCNN 2008, pp. 1322–1328.

[19] W. Siriseriwan, K. Sinapiromsaran, Adaptive neighbor synthetic minority oversampling technique under 1NN outcast handling, Songklanakarin Journal of Science and Technology 39 (2017) 565–576.

[20] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-SMOTE: A new oversampling method in imbalanced data sets learning, in: S.-V. B. Heidelberg (Ed.), International Conference on Intelligent Computing ICIC 2005: Advances in Intelligent Computing. Lecture Notes in Computer Science, volume 3644, pp. 878–887.

[21] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, DBSMOTE: Density-based synthetic minority over-sampling TEchnique, Applied Intelligence 36 (2012) 664–684.

[22] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: S.-V. B. Heidelberg (Ed.), PAKDD 2009: Advances in Knowledge Discovery and Data Mining. Lecture Notes in Computer Science., volume 5476, pp. 475–482.

[23] W. Siriseriwan, K. Sinapiromsaran, The effective redistribution for imbalance dataset : Relocating safe-level smote with minority outcast handling, Chiang Mai Journal of Science 43 (2016) 234 – 246.

[24] S. Barua, M. Islam, X. Yao, K. Murase, MWMOTE - majority weighted minority oversampling technique for imbalanced data set learning, IEEE Transactions on Knowledge and Data Engineering 26 (2014) 405–425.

[25] M. Koziarski, B. Krawczyk, M. Woźniak, Radial-based oversampling for noisy imbalanced data classification, Neurocomputing 343 (2019) 19–33.

[26] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, Morgan Kaufmann, Waltham, MA, USA, 3. edition, 2011.

[27] S. Das, S. Datta, B. B. Chaudhuri, Handling data irregularities in classification: Foundations, trends, and future challenges, Pattern Recognition 81 (2018) 674–693.

[28] A. Fernández, C. J. Carmona, M. J. del Jesús, F. Herrera, A pareto-based ensemble with feature and instance selection for learning from multi-class imbalanced datasets, Int. J. Neural Syst. 27 (2017) 1–21.

[29] Y. He, J. Zhou, Y. Lin, T. Zhu, A class imbalance-aware relief algorithm for the classification of tumors using microarray gene expression data, Computational Biology and Chemistry 80 (2019) 121–127.

[30] S. A. Shahee, U. Ananthakumar, An effective distance based feature selection approach for imbalanced data, Applied Intelligence 50 (2020) 717–745.

[31] F. Viegas, L. Rocha, M. Gonçalves, F. Mourão, G. Sá, T. Salles, G. Andrade, I. Sandin, A genetic programming approach for feature selection in highly dimensional skewed data, Neurocomputing 273 (2018) 554–569.

[32] H. Chen, T. Li, X. Fan, C. Luo, Feature selection for imbalanced data based on neighborhood rough sets, Information Sciences 483 (2019) 1–20.

[33] P. Zhou, X. Hu, P. Li, X. Wu, Online feature selection for high-dimensional class-imbalanced data, Knowledge-Based Systems 136 (2017) 187–199.

[34] C. Zhang, Y. Zhou, J. Guo, G. Wang, X. Wang, Research on classification method of high-dimensional class-imbalanced datasets based on svm, International Journal of Machine Learning and Cybernetics 10 (2019) 1765–1778.

[35] S. Maldonado, R. Weber, F. Famili, Feature selection for high-dimensional class-imbalanced data sets using support vector machines, Information Sciences 286 (2014) 228–246.

[36] W. Li, P. Yi, D. Zhang, Quantile-induced vector-based heavy owa operator and the application in dynamic decision making, International Journal of Intelligent Systems 35 (2020) 250–266.

[37] C. Gong, Y. Su, W. Liu, Y. Hu, Y. Zhou, The distance induced owa operator with application to multi-criteria group decision making, International Journal of Fuzzy Systems 22 (2020) 1624–1634.

[38] X.-f. Song, Y. Zhang, D.-w. Gong, X.-y. Sun, Feature selection using bare-bones particle swarm optimization with mutual information, Pattern Recognition 112 (2021) 107804.

[39] J. Vergara, P. A. Estévez, A review of feature selection methods based on mutual information, Neural Computing and Applications 24 (2014) 175–186.

[40] G. Roffo, S. Melzi, New Frontiers in Mining Complex Patterns, Fifth International workshop, nfMCP2016. Lecture Notes in Computer Science, Springer, pp. 19–35.

[41] P. Luuka, O. Kurama, Similarity classifier with ordered weighted averaging operators, Expert Systems with Applications 40 (2013) 995–1002.

[42] R. Ribeiro, R. Alberto, N. Pereira, Generalized mixture operators using weighting functions: A comparative study with WA and OWA, European Journal of Operational Research 145 (2003) 329–342.

[43] N. Blöchliger, A. Caflisch, A. Vitalis, Weighted distance functions improve analysis of high-dimensional data: Application to molecular dynamics simulations, Journal of Chemical Theory and Computation 11 (2015) 5481–5492.

[44] J. Wang, M. Xu, H. Wang, J. Zhang, Classification of imbalanced data by using the smote algorithm and locally linear embedding, in: 2006 8th international Conference on Signal Processing, volume 3.

[45] M. Naseriparsa, M. Mansour Riahi Kashani, Combination of pca with smote resampling to boost the prediction rate in lung cancer dataset, International Journal of Computer Applications 77 (2013) 33–38.

[46] A. Luque, A. Carrasco, A. Martín, A. de las Heras, The impact of class imbalance in classification performance metrics based on the binary confusion matrix, Pattern Recognition 91 (2019) 216–231.

[47] S. S. Mullick, S. Datta, S. G. Dhekane, S. Das, Appropriateness of performance indices for imbalanced data classification: An analysis, Pattern Recognition 102 (2020) 107197.

[48] I. Cordón, S. García, A. Fernández, F. Herrera, Imbalance: Oversampling algorithms for imbalanced classification in r, Knowl.-Based Syst. 161 (2018) 329–341.

[49] J. Demšar, Statistical comparisons of classifiers over multiple data set, Journal of Machine Learning Research 7 (2006) 1–30.

[50] J. Maillo, S. García, J. Luengo, F. Herrera, I. Triguero, Fast and scalable approaches to accelerate the fuzzy k-nearest neighbors classifier for big data, IEEE Transactions on Fuzzy Systems 28 (2020) 874–886.

[51] C. Huang, Y. Li, C. C. Loy, X. Tang, Deep imbalanced learning for face recognition and attribute prediction, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (2020) 2781–2794.

[52] S. Suh, P. Lukowicz, Y. O. Lee, Discriminative feature generation for classification of imbalanced data, Pattern Recognition 122 (2022) 108302.

Appendix A. Detailed Results without feature selection

Tables A.1 to A.3 present the results for all the datasets and oversampling methods using k-NN, SVM, and LR as classification models, respectively. Each table reports the average AUC of the best parameter configuration obtained for each oversampling method. The best performance is emphasized in bold type. The average AUC computed from all low-dimensional ($\overline{X}_{LDD}$) and high-dimensional ($\overline{X}_{HDD}$) datasets is also reported for each method.

Table A.1: Performance summary using k-nearest neighbors for binary classification. AUC measure.

k-NN	FW-SMOTE	SMT	BL-S.	SL-S.	A.SYN	AN-S.	DB-S.	MWT	RSL-S.	RBO	PCA-S.
Low-dimensional datasets											
ldd1	**72.9**	69.1	68.4	68.3	69.7	72.0	69.2	70.9	68.3	65.6	65.9
ldd2	**92.3**	92.0	92.0	89.5	89.5	**92.3**	**92.3**	91.6	**92.3**	92.0	84.7
ldd3	**92.3**	92.1	91.9	**92.3**	89.8	89.8	91.9	92.1	**92.3**	91.9	56.8
ldd4	**91.4**	89.2	90.0	89.7	87.1	85.1	85.4	88.9	83.6	89.3	58.8
ldd5	**91.9**	91.7	91.1	**91.9**	91.1	91.7	89.4	**91.9**	**91.9**	91.4	49.9
ldd6	**90.0**	88.0	86.8	84.8	**90.0**	85.3	84.6	84.8	85.3	88.0	76.7
ldd7	**92.2**	**92.2**	91.1	**92.2**	89.7	**92.2**	**92.2**	91.7	**92.2**	**92.2**	64.4
ldd8	**92.2**	92.0	91.7	**92.2**	**92.2**	92.0	**92.2**	91.4	**92.2**	91.4	76.9
ldd9	**91.2**	90.2	90.3	90.8	87.4	90.8	88.5	90.8	90.8	90.6	72.5
ldd10	**92.2**	89.7	91.7	91.9	91.9	**92.2**	89.7	91.7	91.9	91.9	62.5
ldd11	**89.1**	86.3	88.1	85.1	**89.1**	86.3	85.1	85.8	85.3	87.3	73.9
ldd12	89.8	88.5	88.5	89.8	86.3	**91.3**	86.0	88.8	89.5	88.8	64.8
ldd13	**88.1**	86.6	86.2	84.3	87.3	85.1	86.5	85.6	85.6	82.8	70.3
ldd14	**92.3**	91.9	89.4	92.0	89.4	92.2	92.0	89.5	89.5	91.9	89.5
ldd15	**78.5**	72.6	72.3	73.4	50.5	53.3	52.2	52.7	48.0	71.5	54.8
ldd16	**90.0**	83.8	83.8	88.8	88.8	88.8	83.8	88.8	88.8	88.8	71.3
ldd17	**89.0**	79.5	84.5	79.5	80.0	79.5	79.5	84.0	79.0	84.0	78.5
ldd18	**95.5**	95.3	95.3	83.0	88.0	83.3	85.5	**95.5**	**95.5**	95.3	76.0
ldd19	**99.7**	99.4	99.3	99.4	99.3	99.4	98.8	99.2	99.2	99.3	78.4
ldd20	**98.1**	97.8	94.9	93.0	93.1	93.9	94.0	92.3	94.9	94.2	85.2
ldd21	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	61.8
ldd22	**66.3**	60.3	60.3	59.4	62.0	60.3	56.4	60.0	56.7	55.6	56.2
ldd23	**95.8**	90.6	88.4	89.9	89.4	89.1	89.7	89.3	90.1	89.9	89.3
ldd24	**85.8**	82.9	80.8	79.3	81.8	80.7	81.8	82.7	77.7	76.9	79.8
ldd25	**70.1**	61.5	63.2	59.1	57.5	57.5	57.8	61.7	56.3	68.0	68.1
ldd26	**65.9**	59.3	62.6	52.6	53.2	58.2	54.8	54.2	55.7	64.2	60.9
ldd27	**91.8**	89.6	89.6	87.3	88.1	88.6	86.0	89.4	86.0	88.4	89.6
ldd28	**90.3**	88.4	87.5	88.1	88.8	88.7	87.1	89.0	87.9	87.9	89.0
ldd29	**76.9**	70.5	73.3	72.1	74.9	73.7	69.0	72.6	72.0	73.0	72.4
ldd30	**95.8**	94.5	95.0	94.6	87.6	92.1	92.3	**95.8**	92.2	94.5	94.5
$\overline{X}_{LDD}$	88.2	85.5	85.6	84.5	83.8	84.2	83.1	84.7	83.7	85.5	72.4
High-dimensional datasets											
hdd1	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	81.5
hdd2	88.7	88.7	88.7	88.7	87.9	**88.9**	88.7	88.7	**88.9**	86.7	47.5
hdd3	**84.1**	79.1	83.6	83.6	83.3	83.9	78.9	83.0	**84.1**	78.5	65.8
hdd4	**100**	99.7	99.7	99.7	99.7	99.7	**100**	**100**	99.7	**100**	75.3
hdd5	81.7	80.6	81.7	81.7	85.0	81.1	83.9	77.2	81.1	84.4	**97.2**
hdd6	**99.7**	97.2	97.8	98.4	91.0	**99.7**	99.1	99.1	99.1	94.7	51.1
hdd7	**95.0**	86.9	87.2	91.0	89.7	88.3	88.6	94.3	90.4	90.7	53.2
hdd8	**100**	92.9	92.6	85.7	92.9	92.9	92.9	92.9	92.9	92.9	48.0
hdd9	**99.7**	98.8	98.8	98.8	99.1	98.8	99.4	98.8	98.8	90.2	97.2
hdd10	**65.0**	63.8	61.3	61.3	63.8	**65.0**	63.8	63.8	63.8	63.8	52.0
hdd11	**100**	**100**	98.0	99.3	99.7	98.3	97.7	97.3	98.0	93.8	60.4
hdd12	98.7	98.0	97.3	97.3	**99.4**	98.7	98.0	98.8	98.7	97.3	55.7
$\overline{X}_{HDD}$	92.7	90.5	90.5	90.4	90.9	91.3	90.9	91.1	91.3	89.4	65.4

Table A.2: Performance summary using linear Support Vector Machine for binary classification. AUC measure.

SVM	FW-SMOTE	SMT	BL-S.	SL-S.	A.SYN	AN-S.	DB-S.	MWT	RSL-S.	RBO	PCA-S.
Low-dimensional datasets											
ldd1	74.8	74.2	75.1	73.5	**75.4**	73.9	73.6	72.9	73.3	73.4	74.0
ldd2	**91.4**	85.5	90.5	83.9	83.4	90.9	90.9	88.4	83.6	85.7	90.1
ldd3	88.7	86.3	**90.8**	89.0	86.2	86.2	86.2	86.3	86.3	86.5	50.0
ldd4	**90.5**	90.2	88.2	90.2	88.2	85.2	85.9	82.2	84.9	89.7	50.0
ldd5	**91.9**	91.7	88.3	91.9	91.7	89.4	89.2	91.7	91.7	91.4	50.0
ldd6	**86.8**	85.3	83.0	82.1	83.0	83.5	82.3	82.3	82.6	82.8	50.0
ldd7	91.1	91.1	88.1	91.1	85.8	88.6	**91.4**	88.6	89.2	89.7	79.9
ldd8	**91.2**	86.7	89.3	88.7	90.9	89.2	89.2	88.7	89.2	89.2	50.0
ldd9	**91.0**	88.3	90.6	87.7	89.9	90.4	87.3	90.4	87.5	90.2	50.0
ldd10	**91.1**	88.9	89.5	86.1	86.8	89.2	86.7	85.9	86.4	88.6	50.0
ldd11	**87.6**	87.3	86.8	85.1	86.8	86.8	84.8	86.3	85.1	86.8	50.0
ldd12	**89.5**	89.3	89.0	89.0	88.5	89.3	89.3	89.3	89.3	89.0	76.3
ldd13	**90.3**	89.9	88.6	90.1	89.1	88.4	89.6	89.6	89.1	89.9	79.3
ldd14	92.5	92.0	94.2	92.0	91.9	92.2	92.0	**94.5**	92.0	92.0	92.0
ldd15	50.0	50.0	50.0	50.0	49.7	49.7	49.4	49.4	**50.8**	50.0	50.0
ldd16	**85.0**	**85.0**	83.8	84.4	84.4	84.4	**85.0**	**85.0**	84.4	84.4	54.4
ldd17	**90.0**	83.5	85.0	89.0	84.5	84.5	84.5	89.5	89.0	85.0	45.0
ldd18	**95.5**	94.5	89.8	95.0	77.8	94.5	90.3	89.5	89.5	89.5	55.5
ldd19	99.2	99.2	99.4	99.2	**99.7**	99.2	99.2	99.2	99.2	99.2	58.4
ldd20	**86.3**	85.4	82.9	83.6	82.2	72.3	70.3	72.8	66.3	78.7	54.3
ldd21	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	50
ldd22	61.2	**61.3**	58.8	**61.3**	60.3	60.2	57.9	61.2	60.5	61.2	50.0
ldd23	**93.1**	90.2	88.6	90.2	89.6	90.1	89.2	89.0	90.3	90.2	89.8
ldd24	**80.7**	79.5	78.2	80.1	76.2	80.7	79.4	80.0	79.9	79.8	79.7
ldd25	51.7	50.0	50.0	**58.2**	53.2	53.2	53.0	53.0	54.7	51.7	51.7
ldd26	51.7	50.0	50.0	50.0	54.4	**59.2**	55.4	56.0	58.9	50.0	50.0
ldd27	**88.9**	**88.9**	83.7	87.8	85.1	**88.9**	88.8	88.3	**88.9**	**88.9**	88.9
ldd28	89.1	88.6	89.2	88.7	89.4	88.6	**89.5**	89.2	88.6	88.5	88.9
ldd29	**73.7**	57.3	51.0	66.4	56.5	63.7	60.2	63.5	68.2	59.5	62.2
ldd30	93.1	90.5	93.9	93.0	86.9	88.2	91.6	93.1	91.8	90.6	**94.0**
$\overline{X}_{LDD}$	84.6	82.7	82.2	83.2	81.6	82.7	82.1	82.5	82.4	82.4	63.8
High-dimensional datasets											
hdd1	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	84
hdd2	**85.3**	**85.3**	**85.3**	**85.3**	**85.3**	**85.3**	**85.3**	**85.3**	**85.3**	**85.3**	50.0
hdd3	79.4	79.4	79.4	79.4	79.4	79.4	79.4	79.4	79.4	79.4	**86.0**
hdd4	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	67
hdd5	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	97
hdd6	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	50.0
hdd7	**83.3**	**83.3**	**83.3**	**83.3**	**83.3**	**83.3**	**83.3**	**83.3**	**83.3**	**83.3**	67.9
hdd8	**85.7**	**85.7**	**85.7**	**85.7**	**85.7**	**85.7**	**85.7**	**85.7**	**85.7**	**85.7**	50.0
hdd9	83.3	83.3	83.3	83.3	83.3	83.3	83.3	83.3	83.3	83.3	**97.2**
hdd10	**52.5**	**52.5**	**52.5**	**52.5**	**52.5**	**52.5**	**52.5**	**52.5**	**52.5**	**52.5**	50.0
hdd11	**96.7**	**96.7**	**96.7**	**96.7**	89.1	88.4	89.6	89.6	89.1	**96.7**	50.0
hdd12	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	50
$\overline{X}_{HDD}$	88.0	88.0	88.0	88.0	87.4	87.3	87.4	87.4	87.4	88.0	66.7

Similar to tables A.1 to A.3, tables A.4 to A.6 present the results using G-mean as performance metric. The best performance is emphasized in bold type. The average G-mean computed from all low-dimensional ($\overline{X}_{LDD}$) and high-dimensional ($\overline{X}_{HDD}$) datasets is also reported for each method.

Table A.3: Performance summary using logistic regression for binary classification. AUC measure.

LR	FW-SMOTE	SMT	BL-S.	SL-S.	A.SYN	AN-S.	DB-S.	MWT	RSL-S.	RBO	PCA-S.
Low-dimensional datasets											
ldd1	**74.0**	73.6	73.7	73.3	73.4	73.8	71.3	73.3	72.8	73.2	67.4
ldd2	**91.6**	91.1	91.1	89.1	86.4	**91.6**	**91.6**	91.1	88.9	86.1	89.1
ldd3	88.8	88.7	**90.6**	88.7	86.2	86.2	86.3	88.8	86.3	86.3	50.0
ldd4	**90.3**	89.0	89.7	90.2	86.6	82.4	83.1	84.7	81.2	87.7	50.0
ldd5	**89.5**	87.8	87.3	88.1	88.1	85.6	88.1	87.8	88.1	**89.8**	50.0
ldd6	**88.0**	86.3	86.3	86.3	85.3	86.0	86.0	86.5	86.5	84.6	49.2
ldd7	88.3	88.3	88.9	88.3	82.8	88.3	88.3	**90.3**	88.3	**90.3**	74.0
ldd8	90.7	88.7	90.1	88.7	**91.4**	88.7	88.7	88.7	88.7	88.4	48.1
ldd9	**90.8**	90.6	87.1	89.1	88.4	90.8	90.4	90.4	90.8	88.9	48.4
ldd10	**91.1**	88.9	90.9	88.9	87.6	88.9	88.9	88.9	88.9	88.4	51.2
ldd11	**89.0**	88.3	88.8	89.0	88.5	88.8	88.8	88.3	89.0	86.3	49.6
ldd12	88.8	88.8	88.5	**89.0**	88.5	88.5	**89.0**	**89.0**	**89.0**	88.8	71.8
ldd13	87.4	**90.6**	89.3	89.3	88.4	85.8	89.6	89.3	86.3	88.9	75.0
ldd14	**91.5**	**91.5**	91.4	91.4	**91.5**	**91.5**	**91.5**	**91.5**	**91.5**	**91.5**	**91.5**
ldd15	77.1	74.6	**80.4**	72.4	56.9	54.2	56.7	56.7	52.2	50.7	50.0
ldd16	82.5	82.5	82.5	82.5	82.5	82.5	82.5	82.5	82.5	**82.6**	55.0
ldd17	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	50.0
ldd18	85.0	87.3	75.0	79.8	75.0	85.0	82.0	**87.5**	**87.5**	77.0	67.3
ldd19	**99.5**	99.2	99.1	99.1	99.2	99.0	99.1	99.3	99.2	**99.5**	53.4
ldd20	**97.6**	88.9	79.1	91.6	78.9	93.6	88.7	87.3	91.8	91.0	63.2
ldd21	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	50.0
ldd22	**64.7**	63.3	62.3	62.3	60.5	62.7	57.4	60.4	61.2	62.2	54.9
ldd23	**90.6**	89.7	88.3	90.3	88.7	90.1	88.6	89.4	90.4	90.2	89.7
ldd24	79.4	**80.3**	79.5	79.7	69.8	77.3	74.8	74.2	72.3	78.1	79.3
ldd25	**74.3**	68.1	70.7	69.7	56.3	56.3	56.3	54.9	53.2	72.9	69.1
ldd26	54.0	53.8	52.5	53.5	54.9	56.2	**56.7**	55.8	53.1	52.4	51.6
ldd27	**89.3**	87.6	82.6	85.3	85.6	87.8	84.3	88.3	86.0	88.1	85.5
ldd28	89.0	88.2	**89.5**	88.1	89.2	88.6	88.4	88.6	88.0	88.8	88.0
ldd29	63.8	73.5	71.7	72.7	71.5	72.8	72.5	**75.3**	72.7	69.1	73.9
ldd30	87.2	86.1	**90.7**	86.1	84.5	86.1	85.1	87.1	86.2	88.0	87.1
$\overline{X}_{LDD}$	85.4	84.8	84.2	84.4	81.9	83.3	82.8	83.5	82.7	83.3	64.4
High-dimensional datasets											
hdd1	**99.2**	95.8	94.8	93.4	93.4	92.6	96.5	95.3	96.7	89.8	74.6
hdd2	**85.4**	78.2	77.7	77.7	72.8	78.7	76.1	80.0	80.8	80.8	50.0
hdd3	**83.4**	74.0	65.7	69.4	64.8	72.5	72.4	71.7	60.2	67.0	82.4
hdd4	**99.2**	93.1	91.4	90.6	96.0	92.5	94.0	97.8	92.5	97.3	65.8
hdd5	95.0	84.2	92.2	92.2	85.3	86.4	92.5	88.6	85.3	89.7	**97.2**
hdd6	**99.4**	89.4	86.9	94.4	90.7	91.6	97.0	88.2	86.9	93.8	58.8
hdd7	**84.3**	69.4	75.9	78.1	77.2	76.5	73.0	76.5	73.3	80.1	65.7
hdd8	**98.2**	83.9	75.6	80.0	84.5	76.2	74.7	83.3	68.1	83.3	50.0
hdd9	**100**	91.1	98.5	98.5	90.8	98.5	98.5	98.8	91.1	97.9	97.2
hdd10	**60.0**	46.3	37.5	37.5	42.5	40.0	43.8	51.3	45.0	36.3	50.0
hdd11	**91.0**	84.1	80.1	77.8	79.7	81.1	82.3	81.7	81.8	80.3	54.4
hdd12	**100**	99.4	91.4	94.3	92.9	92.9	97.9	91.4	92.1	84.3	50.0
$\overline{X}_{HDD}$	91.3	82.4	80.6	82.0	80.9	81.6	83.2	83.7	79.5	81.7	66.4

Table A.4: Performance summary using k-nearest neighbors for binary classification. G-mean measure.

k-NN	FW-SMOTE	SMT	BL-S.	SL-S.	A.SYN	AN-S.	DB-S.	MWT	RSL-S.	RBO	PCA-S.
Low-dimensional datasets											
ldd1	**71.6**	68.3	67.0	64.5	68.4	70.9	66.6	69.3	65.3	62.4	62.9
ldd2	**87.5**	83.5	85.3	84.1	86.5	83.7	84.8	84.9	84.2	81.6	83.7
ldd3	**91.1**	90.7	90.4	90.8	87.9	**91.1**	90.9	90.4	**91.1**	90.8	30.4
ldd4	**91.0**	**91.0**	**91.0**	90.8	88.1	**91.0**	90.7	90.8	**91.0**	90.6	39.4
ldd5	**90.5**	88.0	87.6	87.5	86.1	79.7	79.9	88.0	77.9	87.7	11.3
ldd6	**90.7**	**90.7**	90.0	**90.7**	90.0	90.5	83.7	**90.7**	**90.7**	90.3	62.1
ldd7	86.7	80.1	80.1	78.7	**88.9**	79.1	78.3	78.5	78.9	86.5	49.7
ldd8	**90.9**	**90.9**	90.1	90.6	88.0	**90.9**	**90.9**	90.4	**90.9**	**90.9**	69.7
ldd9	**90.9**	**90.9**	90.3	90.9	90.9	90.8	**90.9**	90.2	**90.9**	90.2	66.1
ldd10	90.1	**91.2**	89.3	89.7	86.0	89.7	87.3	89.7	89.7	89.5	51.8
ldd11	**90.9**	**90.9**	90.7	**90.9**	90.6	**90.9**	88.0	90.5	90.7	90.6	67.2
ldd12	81.5	78.9	81.1	78.4	**87.7**	80.8	78.9	80.0	79.1	81.5	46.6
ldd13	88.0	87.6	87.4	87.8	80.1	**90.0**	79.8	87.0	87.8	87.2	57.3
ldd14	**87.0**	79.8	79.8	86.8	79.7	86.8	86.6	79.7	79.7	86.2	79.7
ldd15	**72.3**	68.4	65.9	62.5	6.7	9.4	8.8	9.4	0.0	61.4	27.4
ldd16	**98.8**	87.5	87.5	98.6	98.6	98.6	87.5	98.6	98.6	98.6	63.0
ldd17	0.0	0.0	0.0	0.0	77.8	77.8	**87.7**	**87.7**	77.2	**87.7**	76.6
ldd18	**95.2**	94.7	94.9	68.5	78.7	68.7	75.5	**95.2**	68.5	94.9	61.8
ldd19	**99.7**	99.4	99.3	99.4	99.3	99.4	98.8	99.2	99.2	99.3	78.1
ldd20	**97.8**	92.2	94.3	92.3	92.6	93.5	93.6	91.9	94.5	93.9	83.2
ldd21	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	54.8
ldd22	**62.4**	51.4	38.5	38.2	50.5	45.3	36.8	44.9	32.8	34.0	39.0
ldd23	**89.8**	88.9	88.2	89.4	89.0	88.5	89.2	88.8	89.5	89.4	88.9
ldd24	**84.7**	82.6	78.1	77.2	80.3	80.1	80.8	81.3	76.7	75.3	77.3
ldd25	**58.7**	51.3	48.6	33.2	25.2	27.1	25.2	37.8	23.0	53.5	53.5
ldd26	51.2	49.0	38.9	19.1	39.5	46.6	40.7	44.3	40.9	**51.3**	46.5
ldd27	**92.0**	88.7	89.1	85.9	87.4	87.8	83.7	88.7	84.2	87.4	88.6
ldd28	**90.3**	88.6	87.4	88.2	88.5	88.3	86.5	88.7	87.4	87.4	88.8
ldd29	**71.7**	70.9	66.9	66.7	21.9	27.6	18.9	28.7	17.8	69.3	67.6
ldd30	**95.6**	94.4	94.6	91.4	86.5	91.4	91.7	**95.6**	91.5	94.2	94.2
$\overline{X}_{LDD}$	83.0	80.3	79.1	77.1	77.4	77.9	76.1	79.3	75.7	82.8	62.2
High-dimensional datasets											
hdd1	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	**99.7**	81.4
hdd2	98.5	98.5	98.5	98.5	97.6	**98.8**	98.5	98.5	**98.8**	96.0	27.9
hdd3	**96.0**	76.5	86.6	86.6	87.0	87.6	76.5	86.6	87.9	76.0	65.2
hdd4	99.7	99.7	99.7	99.7	89.7	99.7	**100**	**100**	99.7	**100**	74
hdd5	76.7	74.8	76.7	76.7	83.1	76.3	74.8	74.8	76.3	80.0	**97.2**
hdd6	98.1	89.7	89.7	89.7	85.9	**99.7**	99.0	99.0	99.0	89.7	45.2
hdd7	**89.6**	85.9	83.1	83.1	85.8	83.1	83.1	88.7	76.3	79.8	44.1
hdd8	**100**	92.7	92.3	88.7	90.4	92.7	92.7	92.7	92.7	92.7	38.3
hdd9	**98.8**	96.3	96.3	96.3	97.8	96.3	98.2	96.3	96.3	96.3	97.2
hdd10	**92.8**	90.4	92.3	92.3	90.4	92.3	90.4	90.4	90.4	90.4	28.2
hdd11	**99.7**	**99.7**	98.2	**99.7**	**99.7**	98.2	97.5	97.1	97.8	93.2	51.3
hdd12	98.6	98.2	97.1	97.1	**99.4**	98.6	98.0	98.7	98.6	97.2	39.7
$\overline{X}_{HDD}$	95.7	91.8	92.5	92.4	92.2	93.6	92.4	93.6	92.8	90.9	57.5

Table A.5: Performance summary using linear Support Vector Machine for binary classification. G-mean measure.

SVM	FW-SMOTE	SMT	BL-S.	SL-S.	A.SYN	AN-S.	DB-S.	MWT	RSL-S.	RBO	PCA-S.
Low-dimensional datasets											
ldd1	74.4	73.9	67.2	73.9	**75.2**	73.6	73.3	72.5	73.0	73.0	73.6
ldd2	**89.5**	87.6	88.4	88.5	88.5	87.7	89.0	89.0	88.4	89.3	89.5
ldd3	**90.1**	86.9	89.4	77.1	76.9	89.9	77.1	86.9	77.1	79.7	0.0
ldd4	87.0	87.2	**88.9**	77.4	84.2	84.4	84.2	84.4	84.3	84.6	0.0
ldd5	**89.1**	88.2	88.7	88.8	87.2	83.9	84.3	72.4	83.1	88.4	0.0
ldd6	**90.6**	90.4	89.6	**90.6**	90.5	87.7	87.5	90.5	90.5	90.2	0.0
ldd7	**80.2**	73.2	72.4	71.8	72.5	73.1	71.5	71.8	71.8	72.5	79.2
ldd8	90.1	86.8	87.1	**90.5**	83.9	87.0	90.2	87.1	87.5	88.4	0.0
ldd9	90.2	87.5	**90.4**	86.9	89.6	87.5	87.5	87.0	87.5	87.5	0.0
ldd10	**89.9**	87.7	89.4	89.4	89.1	89.4	85.4	89.4	85.6	89.3	0.0
ldd11	87.7	87.7	**88.3**	84.8	85.5	87.7	84.8	84.2	84.6	87.3	0.0
ldd12	**81.9**	81.6	80.6	78.7	81.1	81.1	78.7	80.6	78.9	81.1	71.7
ldd13	**87.9**	87.4	87.4	**87.9**	87.0	87.7	87.6	87.6	87.7	87.5	78.2
ldd14	87.1	86.6	93.4	86.6	86.4	86.7	86.6	**93.7**	86.6	86.7	86.6
ldd15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**6.9**	0.0	0.0
ldd16	**88.9**	**88.9**	88.2	88.2	88.2	88.2	**88.9**	**88.9**	88.2	88.2	30.4
ldd17	**99.5**	88.2	88.9	99.4	88.3	88.3	99.4	99.4	98.9	88.9	0.0
ldd18	**95.2**	84.9	94.5	75.4	62.2	94.2	85.2	84.9	85.4	84.7	16.6
ldd19	99.2	99.1	99.4	99.2	**99.7**	99.1	99.2	99.1	99.2	99.1	42.9
ldd20	**87.1**	80.8	86.8	86.3	76.5	68.2	63.0	69.9	58.7	74.9	28.1
ldd21	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	0.0
ldd22	**46.1**	**46.1**	42.3	**46.1**	44.4	44.3	39.3	**46.1**	45.7	**46.1**	0.0
ldd23	**91.2**	89.6	88.2	89.8	89.3	89.7	88.6	88.5	89.9	89.8	89.4
ldd24	78.1	77.4	78.5	77.6	75.8	**80.1**	79.0	79.5	79.3	77.5	77.3
ldd25	5.8	0.0	5.8	5.8	11.5	**23.0**	11.5	11.5	17.3	5.8	5.8
ldd26	36.4	0.0	0.0	0.0	36.4	**42.5**	34.8	36.4	**42.5**	0.0	0.0
ldd27	**88.1**	88.0	78.0	87.1	83.1	**88.1**	88.0	87.5	**88.1**	**88.1**	**88.1**
ldd28	88.7	88.9	88.3	88.6	**89.3**	88.2	89.3	88.9	88.2	88.1	88.6
ldd29	42.5	42.1	4.5	**57.6**	49.1	49.3	46.6	50.6	52.6	36.2	46.5
ldd30	91.4	92.7	89.7	90.0	85.3	87.2	91.2	92.7	91.3	89.8	**93.7**
$\overline{X}_{LDD}$	78.5	75.3	74.5	75.5	75.2	77.3	75.7	76.7	76.6	74.8	36.2
High-dimensional datasets											
hdd1	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	83.7
hdd2	**94.1**	**94.1**	**94.1**	**94.1**	**94.1**	**94.1**	**94.1**	**94.1**	**94.1**	**94.1**	0.0
hdd3	**94.6**	**77.1**	**77.1**	**77.1**	**77.1**	**77.1**	**77.1**	**77.1**	**77.1**	**77.1**	84.9
hdd4	**100**	**100**	**100**	**100**	90.0	**100**	**100**	**100**	**100**	**100**	59.2
hdd5	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	97.2
hdd6	**90.5**	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	23.6
hdd7	**81.6**	77.9	77.9	77.9	77.9	77.9	77.9	77.9	77.9	77.9	64.6
hdd8	**84.5**	**84.5**	**84.5**	**84.5**	**84.5**	**84.5**	**84.5**	**84.5**	**84.5**	**84.5**	0.0
hdd9	81.6	81.6	81.6	81.6	81.6	81.6	81.6	81.6	81.6	81.6	**97.2**
hdd10	**55.2**	**55.2**	**55.2**	**55.2**	**55.2**	**55.2**	**55.2**	**55.2**	**55.2**	**55.2**	0.0
hdd11	**98.4**	**96.3**	**96.3**	**96.3**	**96.3**	**96.3**	**96.3**	**96.3**	**96.3**	**96.3**	0.0
hdd12	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	0.0
$\overline{X}_{HDD}$	90.0	87.2	87.2	87.2	86.4	87.2	87.2	87.2	87.2	87.2	42.5

Table A.6: Performance summary using logistic regression for binary classification. G-mean measure.

LR	FW-SMOTE	SMT	BL-S.	SL-S.	A.SYN	AN-S.	DB-S.	MWT	RSL-S.	RBO	PCA-S.
Low-dimensional datasets											
ldd1	**73.2**	73.2	73.0	72.1	72.5	72.9	70.0	72.1	71.7	72.3	64.6
ldd2	86.8	86.6	87.3	87.0	87.8	84.3	**88.9**	88.6	85.6	88.2	88.4
ldd3	90.2	90.2	90.0	87.5	80.2	**90.4**	87.5	90.0	87.2	80.0	0.0
ldd4	87.0	87.0	**88.7**	84.3	84.2	84.4	84.3	87.2	84.3	84.3	0.0
ldd5	**89.0**	88.9	88.1	88.9	85.7	77.1	77.5	79.5	75.6	85.9	0.0
ldd6	**89.6**	89.0	86.1	86.7	86.7	83.7	86.7	86.5	86.7	88.7	13.1
ldd7	**86.8**	80.5	80.3	80.7	79.3	80.3	80.0	80.7	80.5	78.9	66.0
ldd8	**89.3**	86.7	88.0	86.7	80.5	86.7	86.7	89.0	86.7	89.0	13.1
ldd9	89.7	86.7	89.4	87.0	**90.1**	87.0	87.0	87.0	87.0	86.7	13.2
ldd10	**89.9**	89.4	86.7	89.5	87.4	89.7	89.4	89.4	89.7	87.2	28.9
ldd11	**90.1**	**90.1**	88.8	87.5	86.2	87.5	87.5	87.5	87.5	87.0	19.6
ldd12	**83.5**	**83.5**	82.0	**83.5**	82.9	83.2	83.2	82.7	**83.5**	80.6	62.5
ldd13	87.2	87.2	87.2	**87.5**	87.1	87.1	87.4	87.4	**87.5**	87.2	69.4
ldd14	**86.2**	**86.2**	85.8	**86.2**	**86.2**	**86.2**	**86.2**	**86.2**	**86.2**	85.7	**86.2**
ldd15	56.6	50.5	**57.0**	49.5	16.5	13.7	16.5	16.5	7.1	13.5	0.0
ldd16	**86.0**	**86.0**	**86.0**	**86.0**	**86.0**	**86.0**	**86.0**	**86.0**	**86.0**	**86.0**	30.4
ldd17	**97.0**	**97.0**	**97.0**	**97.0**	**97.0**	**97.0**	**97.0**	**97.0**	**97.0**	**97.0**	11.1
ldd18	79.7	69.5	70.2	79.3	59.7	79.7	72.3	**82.5**	60.2	62.3	43.3
ldd19	**99.5**	98.8	99.1	98.8	99.2	99.0	99.1	99.3	99.2	**99.5**	52.4
ldd20	**97.5**	89.9	73.7	90.9	71.7	93.2	87.3	81.2	90.7	89.7	40.3
ldd21	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	**99.6**	0.0
ldd22	**62.1**	34.2	41.1	52.1	41.3	53.3	40.6	43.1	46.0	47.0	28.1
ldd23	**90.6**	**90.6**	87.7	89.8	88.6	89.7	87.7	88.9	89.9	89.9	89.3
ldd24	76.8	75.4	77.0	**77.1**	68.9	76.6	73.7	72.9	71.1	75.2	76.7
ldd25	66.2	**69.7**	66.3	60.4	15.5	17.8	15.5	13.8	8.1	60.8	60.0
ldd26	5.8	5.7	11.4	22.7	**46.6**	41.8	43.9	42.2	36.1	11.2	5.8
ldd27	**87.5**	84.4	81.4	82.6	83.9	85.9	81.7	86.6	83.9	86.4	83.4
ldd28	88.5	88.2	**89.0**	87.9	**89.0**	88.2	88.0	88.2	87.5	88.4	87.6
ldd29	**69.5**	68.7	65.7	68.9	52.5	52.6	46.4	52.6	43.3	61.4	68.0
ldd30	85.8	85.8	**91.5**	80.4	82.5	84.8	83.6	85.8	84.8	86.8	85.8
$\overline{X}_{LDD}$	82.2	80.0	79.8	80.6	75.8	78.0	76.7	77.7	75.7	77.9	42.9
High-dimensional datasets											
hdd1	**99.7**	96.4	94.8	92.0	89.3	92.0	96.4	94.8	96.6	87.9	73.8
hdd2	**90.5**	83.9	83.9	79.6	79.6	86.8	83.9	88.1	89.3	89.2	0.0
hdd3	**88.9**	74.8	68.9	70.0	60.2	74.8	70.0	68.8	51.5	68.6	82.3
hdd4	**99.7**	93.4	96.3	91.9	86.0	91.9	93.4	97.8	91.8	97.2	60.5
hdd5	95.5	28.6	7.4	11.0	11.0	5.7	6.1	6.1	7.4	6.1	**97.2**
hdd6	**98.8**	88.7	86.5	86.5	85.6	86.5	96.9	83.1	76.8	88.7	57.3
hdd7	**84.3**	40.9	55.2	78.4	76.8	79.7	74.8	55.2	42.7	59.4	61.0
hdd8	**97.6**	91.9	79.3	86.5	76.2	80.5	48.1	87.7	76.2	42.7	0.0
hdd9	**99.7**	91.9	96.9	96.9	88.5	96.9	96.9	98.8	91.9	98.8	97.2
hdd10	**72.9**	60.2	55.2	28.6	28.6	36.8	40.9	55.2	48.1	28.6	0.0
hdd11	**87.8**	85.9	80.5	80.5	78.4	79.3	80.5	80.5	79.9	79.7	36.8
hdd12	**99.3**	98.2	91.9	96.3	87.7	91.9	97.8	88.5	87.7	76.2	0.0
$\overline{X}_{HDD}$	92.9	77.9	74.7	74.8	70.7	75.2	73.8	75.4	70.0	68.6	47.2

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

	**Sebastián Maldonado** received his B.S. and M.S. degrees from the University of Chile, in 2007, and his Ph.D. degree from the University of Chile, in 2011. He is currently Full Professor at the Department of Management Control and Information Systems, School of Economics and Business, University of Chile. His research interests include statistical learning, data mining and business analytics. Sebastián Maldonado has published more than 70 scientific contributions in the last ten years.
	**Carla Vairetti** received her B.S. degree in Computer Science in 2000 from the University Nacional de La Plata, Argentina. She also received an M.S. degree in Sciences in 2013 from the Pontificia Universidad Católica, Chile, and the Ph.D. degree in Engineering Sciences in 2016 from the University of Trento, Italy. Currently, she is Assistant Professor of Universidad de Los Andes. Her research interests include classification in imbalanced domains, multiclassification problems with ensembles and decomposition techniques, data science in big data applications, computational Intelligence (including machine and deep learning) and data science.
	**Alberto Fernández** received the M.Sc. and Ph.D. degrees in computer science from the University of Granada, Granada, Spain, in 2005 and 2010, respectively. He is currently an Assistant Professor with the Department of Computer Science and Artificial Intelligence, University of Granada, Spain. He has published more than 100 papers in highly rated JCR journals and international conferences. In 2013, 2014, and 2017 Dr. Fernández received the University of Granada Prize for Scientific Excellence Works in the field of Engineering. He has also been awarded in 2011 with the Lofti A. Zadeh Best Paper prize (IFSA Association). He has been recently selected as a Highly Cited Researcher http://highlycited.com (in the field of Computer Science, 2017 Clarivate Analytics). He is member of editorial board of several JCR journals such as Applied Intelligence and Plos-One, and also acted as guest editor for special issues in Cognitive Computation and Big Data Analytics. He is also a member of the Spanish Association of Artificial Intelligence (AEPIA). His research interests include classification in imbalanced domains, fuzzy rule learning, evolutionary algorithms, multiclassification problems with ensembles and decomposition techniques, data science in big data applications and the field of Bioinformatics. He has been involved in several projects to apply these subjects of study into industry, healthcare, and economics, among others.

Francisco Herrera (SM'15) received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada and Director of the Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI). He's an academician in the Royal Academy of Engineering (Spain).

He has been the supervisor of 51 Ph.D. students. He has published more than 500 journal papers, receiving more than 86000 citations (Scholar Google, H-index 144). He has been nominated as a Highly Cited Researcher (in the fields of Computer Science and Engineering, respectively, 2014 to present, Clarivate Analytics). He currently acts as Editor in Chief of the international journal "Information Fusion" (Elsevier). He acts as editorial member of a dozen of journals.

He received the several honors and awards, among others: ECCAI Fellow 2009, IFSA Fellow 2013, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 and 2012 Papers, 2011 Lotfi A. Zadeh Prize Best paper Award (IFSA Association), 2013 AEPIA Award to a scientific career in Artificial Intelligence, 2014 XV Andalucía Research Prize Maimónides, 2017 Andalucía Medal (by the regional government of Andalucía), 2018 “Granada: Science and Innovation City” award.

His current research interests include among others, Computational Intelligence (including fuzzy modeling, computing with words, evolutionary algorithms and deep learning), information fusion and decision making, and data science (including data preprocessing, prediction, non-standard classification problems, and big data).