A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches

Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, *Member, IEEE*, and Francisco Herrera, *Member, IEEE*

***Abstract*—Classifier learning with data-sets that suffer from imbalanced class distributions is a challenging problem in data mining community. This issue occurs when the number of examples that represent one class is much lower than the ones of the other classes. Its presence in many real-world applications has brought along a growth of attention from researchers. In machine learning, the ensemble of classifiers are known to increase the accuracy of single classifiers by combining several of them, but neither of these learning techniques alone solve the class imbalance problem, to deal with this issue the ensemble learning algorithms have to be designed specifically. In this paper, our aim is to review the state of the art on ensemble techniques in the framework of imbalanced data-sets, with focus on two-class problems. We propose a taxonomy for ensemble-based methods to address the class imbalance where each proposal can be categorized depending on the inner ensemble methodology in which it is based. In addition, we develop a thorough empirical comparison by the consideration of the most significant published approaches, within the families of the taxonomy proposed, to show whether any of them makes a difference. This comparison has shown the good behavior of the simplest approaches which combine random undersampling techniques with bagging or boosting ensembles. In addition, the positive synergy between sampling techniques and bagging has stood out. Furthermore, our results show empirically that ensemble-based algorithms are worthwhile since they outperform the mere use of preprocessing techniques before learning the classifier, therefore justifying the increase of complexity by means of a significant enhancement of the results.**

***Index Terms*—Bagging, boosting, class distribution, classification, ensembles, imbalanced data-sets, multiple classifier systems.**

I. Introduction

CLASS distribution, i.e., the proportion of instances belonging to each class in a data-set, plays a key role in classification. Imbalanced data-sets problem occurs when one class, usually the one that refers to the concept of interest (positive or minority class), is underrepresented in the data-set; in other words, the number of negative (majority) instances outnumbers the amount of positive class instances. Anyway, neither uniform distributions nor skewed distributions have to imply additional difficulties to the classifier learning task by themselves [1]–[3]. However, data-sets with skewed class distribution usually tend to suffer from class overlapping, small sample size or small disjuncts, which difficult classifier learning [4]–[7]. Furthermore, the evaluation criterion, which guides the learning procedure, can lead to ignore minority class examples (treating them as noise) and hence, the induced classifier might lose its classification ability in this scenario. As a usual example, let us consider a data-set whose imbalance ratio is 1:100 (i.e., for each example of the positive class, there are 100 negative class examples). A classifier that tries to maximize the accuracy of its classification rule, may obtain an accuracy of 99% just by the ignorance of the positive examples, with the classification of all instances as negatives.

In recent years, class imbalance problem has emerged as one of the challenges in data mining community [8]. This situation is significant since it is present in many real-world classification problems. For instance, some applications are known to suffer from this problem, fault diagnosis [9], [10], anomaly detection [11], [12], medical diagnosis [13], e-mail foldering [14], face recognition [15], or detection of oil spills [16], among others. On account of the importance of this issue, a large amount of techniques have been developed trying to address the problem. These proposals can be categorized into three groups, which depend on how they deal with class imbalance. The algorithm level (*internal*) approaches create or modify the algorithms that exist, to take into account the significance of positive examples [17]–[19]. Data level (*external*) techniques add a preprocessing step where the data distribution is rebalanced in order to decrease the effect of the skewed class distribution in the learning process [20]–[22]. Finally, cost-sensitive methods combine both algorithm and data level approaches to incorporate different misclassification costs for each class in the learning phase [23], [24].

In addition to these approaches, another group of techniques emerges when the use of ensembles of classifiers is considered. Ensembles [25], [26] are designed to increase the accuracy of a single classifier by training several different classifiers and combining their decisions to output a single class label. Ensemble methods are well known in machine learning and their

Manuscript received January 12, 2011; revised April 28, 2011 and June 7, 2011; accepted June 23, 2011. Date of publication August 8, 2011; date of current version June 13, 2012. This work was supported in part by the Spanish Ministry of Science and Technology under projects TIN2008-06681-C06-01 and TIN2010-15055. This paper was recommended by Associate Editor M. Last.

M. Galar, E. Barrenechea, and H. Bustince are with the Department of Automática y Computación, Universidad Pública de Navarra, 31006 Navarra, Spain (e-mail: mikel.galar@unavarra.es; edurne.barrenechea@unavarra.es).

A. Fernández is with the Department of Computer Science, University of Jaén, 23071 Jaén, Spain (e-mail: alberto.fernandez@ujaen.es).

F. Herrera is with the Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain (e-mail: herrera@decsai.ugr.es).

Digital Object Identifier 10.1109/TSMCC.2011.2161285

application range over a large number of problems [27]–[30]. In the literature, the term "ensemble methods" usually refers to those collection of classifiers that are minor variants of the same classifier, whereas "multiple classifier systems" is a broader category that also includes those combinations that consider the hybridization of different models [31], [32], which are not covered in this paper. When forming ensembles, creating diverse classifiers (but maintaining their consistency with the training set) is a key factor to make them accurate. Diversity in ensembles has a thorough theoretical background in regression problems (where it is studied in terms of bias-variance [33] and ambiguity [34] decomposition); however, in classification, the concept of diversity is still formally ill-defined [35]. Even though, diversity is necessary [36]–[38] and there exist several different ways to achieve it [39]. In this paper, we focus on data variation-based ensembles, which consist in the manipulation of the training examples in such a way that each classifier is trained with a different training set. AdaBoost [40], [41] and Bagging [42] are the most common ensemble learning algorithms among them, but there exist many variants and other different approaches [43].

Because of their accuracy-oriented design, ensemble learning algorithms that are directly applied to imbalanced data-sets do not solve the problem that underlay in the base classifier by themselves. However, their combination with other techniques to tackle the class imbalance problem have led to several proposals in the literature, with positive results. These hybrid approaches are in some sense algorithm level approaches (since they slightly modify the ensemble learning algorithm), but they do not need to change the base classifier, which is one of their advantages. The modification of the ensemble learning algorithm usually includes data level approaches to preprocess the data before learning each classifier [44]–[47]. However, other proposals consider the embedding of the cost-sensitive framework in the ensemble learning process [48]–[50].

In general, algorithm level and cost-sensitive approaches are more dependent on the problem, whereas data level and ensemble learning methods are more versatile since they can be used independently of the base classifier. Many works have been developed studying the suitability of data preprocessing techniques to deal with imbalanced data-sets [21], [51], [52]. Furthermore, there exist several comparisons between different external techniques in different frameworks [20], [53], [54]. On the other hand, with regard to ensemble learning methods, a large number of different approaches have been proposed in the literature, including but not limited to SMOTEBoost [44], RUSBoost [45], IIVotes [46], EasyEnsemble [47], or SMOTEBagging [55]. All of these methods seem to be adequate to deal with the class imbalance problem in concrete frameworks, but there are no exhaustive comparisons of their performance among them. In many cases, new proposals are compared with respect to a small number of methods and by the usage of limited sets of problems [44]–[47]. Moreover, there is a lack of a unification framework where they can be categorized.

Because of these reasons, our aim is to review the state of the art on ensemble techniques to address a two-class imbalanced data-sets problem and to propose a taxonomy that defines a general framework within each algorithm can be placed. We consider different families of algorithms depending on which ensemble learning algorithm they are based, and what type of techniques they used to deal with the imbalance problem. Over this taxonomy, we carry out a thorough empirical comparison of the performance of ensemble approaches with a twofold objective. The first one is to analyze which one offers the best behavior among them. The second one is to observe the suitability of increasing classifiers' complexity with the use of ensembles instead of the consideration of a unique stage of data preprocessing and training a single classifier.

We have designed the experimental framework in such a way that we can extract well-founded conclusions. We use a set of 44 two-class real-world problems, which suffer from the class imbalance problem, from the KEEL data-set repository [56], [57] (http://www.keel.es/dataset.php). We consider C4.5 [58] as base classifier for our experiments since it has been widely used in imbalanced domains [20], [59]–[61]; besides, most of the proposals we are studying were tested with C4.5 by their authors (e.g., [45], [50], [62]). We perform the comparison by the development of a hierarchical analysis of ensemble methods that is directed by nonparametric statistical tests as suggested in the literature [63]–[65]. To do so, according to the imbalance framework, we use the area under the ROC curve (AUC) [66], [67] as the evaluation criterion.

The rest of this paper is organized as follows. In Section II, we present the imbalanced data-sets problem that describes several techniques which have been combined with ensembles, and discussing the evaluation metrics. In Section III, we recall different ensemble learning algorithms, describe our new taxonomy, and review the state of the art on ensemble-based techniques for imbalanced data-sets. Next, Section IV introduces the experimental framework, that is, the algorithms that are included in the study with their corresponding parameters, the data-sets, and the statistical tests that we use along the experimental study. In Section V, we carry out the experimental analysis over the most significant algorithms of the taxonomy. Finally, in Section VI, we make our concluding remarks.

II. Introduction to Class Imbalance Problem in Classification

In this section, we first introduce the problem of imbalanced data-sets in classification. Then, we present how to evaluate the performance of the classifiers in imbalanced domains. Finally, we recall several techniques to address the class imbalance problem, specifically, the data level approaches that have been combined with ensemble learning algorithms in previous works.

Prior to the introduction of the problem of class imbalance, we should formally state the concept of supervised classification [68]. In machine learning, the aim of classification is to learn a system capable of the prediction of the unknown output class of a previously unseen instance with a good generalization ability. The learning task, i.e., the knowledge extraction, is carried out by a set of n input instances $x_1, \ldots, x_n$ characterized by i features $a_1, \ldots, a_i \in \mathbb{A}$, which includes numerical or nominal values, whose desired output class labels $y_j \in \mathbb{C} = \{c_1, \ldots, c_m\}$, in the case of supervised classification, are known before to the

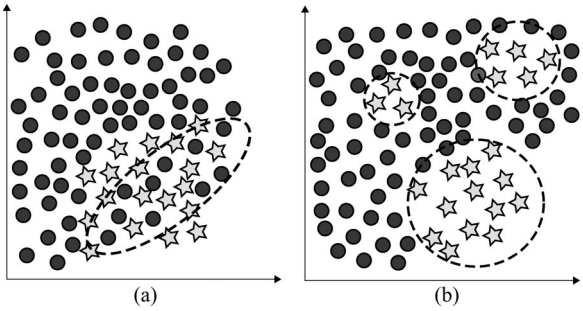


Fig. 1. Example of difficulties in imbalanced data-sets. (a) Class overlapping. (b) Small disjuncts.

learning stage. In such a way, the system that is generated by the learning algorithm is a mapping function that is defined over the patterns $\mathbb{A}^i \rightarrow \mathbb{C}$, and it is called classifier.

A. The Problem of Imbalanced Data-sets

In classification, a data-set is said to be imbalanced when the number of instances which represents one class is smaller than the ones from other classes. Furthermore, the class with the lowest number of instances is usually the class of interest from the point of view of the learning task [22]. This problem is of great interest because it turns up in many real-world classification problems, such as remote-sensing [69], pollution detection [70], risk management [71], fraud detection [72], and especially medical diagnosis [13], [24], [73]–[75].

In these cases, standard classifier learning algorithms have a bias toward the classes with greater number of instances, since rules that correctly predict those instances are positively weighted in favor of the accuracy metric, whereas specific rules that predict examples from the minority class are usually ignored (treating them as noise), because more general rules are preferred. In such a way, minority class instances are more often misclassified than those from the other classes. Anyway, skewed data distribution does not hinder the learning task by itself [1], [2], the issue is that usually a series of difficulties related to this problem turn up.

1) *Small sample size*: Generally imbalanced data-sets do not have enough minority class examples. In [6], the authors reported that the error rate caused by imbalanced class distribution decreases when the number of examples of the minority class is representative (fixing the ratio of imbalance). This way, patterns that are defined by positive instances can be better learned despite the uneven class distribution. However, this fact is usually unreachable in real-world problems.
2) *Overlapping* or *class separability* [see Fig. 1(a)]: When it occurs, discriminative rules are hard to induce. As a consequence, more general rules are induced that misclassify a low number of instances (minority class instances) [4]. If there is no overlapping between classes, any simple classifier could learn an appropriate classifier regardless of the class distribution.
3) *Small disjuncts* [see Fig. 1(b)]: The presence of small disjuncts in a data-set occurs when the concept represented by the minority class is formed of subconcepts [5]. Besides, small disjuncts are implicit in most of the problems. The existence of subconcepts also increases the complexity of the problem because the amount of instances among them is not usually balanced.

TABLE I
CONFUSION MATRIX FOR A TWO-CLASS PROBLEM

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

In this paper, we focus on two-class imbalanced data-sets, where there is a positive (minority) class, with the lowest number of instances, and a negative (majority) class, with the highest number of instances. We also consider the imbalance ratio (IR) [54], defined as the number of negative class examples that are divided by the number of positive class examples, to organize the different data-sets.

B. Performance Evaluation in Imbalanced Domains

The evaluation criterion is a key factor both in the assessment of the classification performance and guidence of the classifier modeling. In a two-class problem, the confusion matrix (shown in Table I) records the results of correctly and incorrectly recognized examples of each class.

Traditionally, the accuracy rate (1) has been the most commonly used empirical measure. However, in the framework of imbalanced data-sets, accuracy is no longer a proper measure, since it does not distinguish between the numbers of correctly classified examples of different classes. Hence, it may lead to erroneous conclusions, i.e., a classifier that achieves an accuracy of 90% in a data-set with an IR value of 9, is not accurate if it classifies all examples as negatives.

$$\text{Acc} = \frac{TP + TN}{TP + FN + FP + TN}. \tag{1}$$

For this reason, when working in imbalanced domains, there are more appropriate metrics to be considered instead of accuracy. Specifically, we can obtain four metrics from Table I to measure the classification performance of both, positive and negative, classes independently.

1) *True positive rate* $TP_{\text{rate}} = \frac{TP}{TP+FN}$ is the percentage of positive instances correctly classified.
2) *True negative rate* $TN_{\text{rate}} = \frac{TN}{FP+TN}$ is the percentage of negative instances correctly classified.
3) *False positive rate* $FP_{\text{rate}} = \frac{FP}{FP+TN}$ is the percentage of negative instances misclassified.
4) *False negative rate* $FN_{\text{rate}} = \frac{FN}{TP+FN}$ is the percentage of positive instances misclassified.

Clearly, since classification intends to achieve good quality results for both classes, none of these measures alone is adequate by itself. One way to combine these measures and produce an evaluation criterion is to use the receiver operating characteristic (ROC) graphic [66]. This graphic allows the visualization of the trade-off between the benefits (TP_{rate}) and costs (FP_{rate}); thus, it evidences that any classifier cannot increase the number

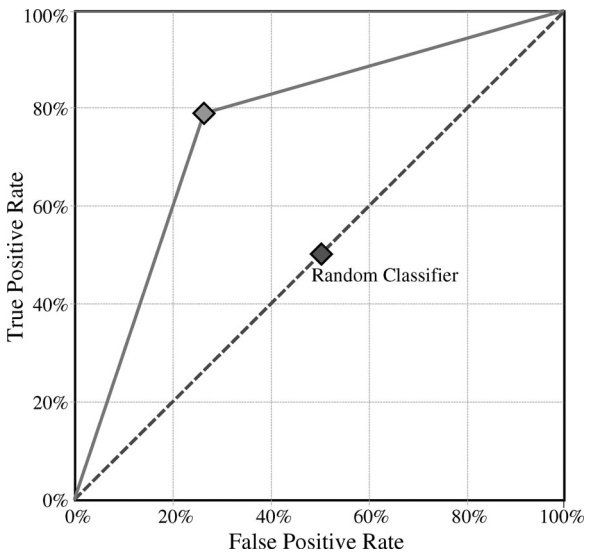


Fig. 2. Example of an ROC plot. Two classifiers' curves are depicted: the dashed line represents a random classifier, whereas the solid line is a classifier which is better than the random classifier.

of true positives without the increment of the false positives. The area under the ROC curve (AUC) [67] corresponds to the probability of correctly identifying which one of the two stimuli is noise and which one is signal plus noise. AUC provides a single measure of a classifier's performance for the evaluation that which model is better on average. Fig. 2 shows how to build the ROC space plotting on a two-dimensional chart, the TP_{rate} (Y-axis) against the FP_{rate} (X-axis). Points in $(0, 0)$ and $(1, 1)$ are trivial classifiers where the predicted class is always the negative and positive, respectively. On the contrary, $(0, 1)$ point represents the perfect classification. The AUC measure is computed just by obtaining the area of the graphic:

$$\text{AUC} = \frac{1 + TP_{\text{rate}} - FP_{\text{rate}}}{2}. \tag{2}$$

C. Dealing With the Class Imbalance Problem

On account of the importance of the imbalanced data-sets problem, a large amount of techniques have been developed to address this problem. As stated in the introduction, these approaches can be categorized into three groups, depending on how they deal with the problem.

1) Algorithm level approaches (also called *internal*) try to adapt existing classifier learning algorithms to bias the learning toward the minority class [76]–[78]. These methods require special knowledge of both the corresponding classifier and the application domain, comprehending why the classifier fails when the class distribution is uneven.
2) Data level (or *external*) approaches rebalance the class distribution by resampling the data space [20], [52], [53], [79]. This way, they avoid the modification of the learning algorithm by trying to decrease the effect caused by imbalance with a preprocessing step. Therefore, they are independent of the classifier used, and for this reason, usually more versatile.
3) Cost-sensitive learning framework falls between data and algorithm level approaches. It incorporates both data level transformations (by adding costs to instances) and algorithm level modifications (by modifying the learning process to accept costs) [23], [80], [81]. It biases the classifier toward the minority class the the assumption higher misclassification costs for this class and seeking to minimize the total cost errors of both classes. The major drawback of these approaches is the need to define misclassification costs, which are not usually available in the data-sets.

In this work, we study approaches that are based on ensemble techniques to deal with the class imbalance problem. Aside from those three categories, ensemble-based methods can be classified into a new category. These techniques usually consist in a combination between an ensemble learning algorithm and one of the techniques above, specifically, data level and cost-sensitive ones. By the addition of a data level approach to the ensemble learning algorithm, the new hybrid method usually preprocesses the data before training each classifier. On the other hand, cost-sensitive ensembles instead of modifying the base classifier in order to accept costs in the learning process guide the cost minimization via the ensemble learning algorithm. This way, the modification of the base learner is avoided, but the major drawback (i.e., costs definition) is still present.

D. Data Preprocessing Methods

As pointed out, preprocessing techniques can be easily embedded in ensemble learning algorithms. Hereafter, we recall several data preprocessing techniques that have been used together with ensembles, which we will analyze in the following sections.

In the specialized literature, we can find some papers about resampling techniques that study the effect of changing class distribution to deal with imbalanced data-sets, where it has been empirically proved that the application of a preprocessing step in order to balance the class distribution is usually a positive solution [20], [53]. The main advantage of these techniques, as previously pointed out, is that they are independent of the underlying classifier.

Resampling techniques can be categorized into three groups. Undersampling methods, which create a subset of the original data-set by eliminating instances (usually majority class instances); oversampling methods, which create a superset of the original data-set by replicating some instances or creating new instances from existing ones; and finally, hybrids methods that combine both sampling methods. Among these categories, there exist several different proposals; from this point, we only center our attention in those that have been used in combination with ensemble learning algorithms.

1) *Random undersampling*: It is a nonheuristic method that aims to balance class distribution through the random elimination of majority class examples. Its major drawback is that it can discard potentially useful data, which could be important for the induction process.
2) *Random oversampling*: In the same way as random undersampling, it tries to balance class distribution, but in

this case, randomly replicating minority class instances. Several authors agree that this method can increase the likelihood of occurring overfitting, since it makes exact copies of existing instances.

3) *Synthetic minority oversampling technique* (SMOTE) [21]: It is an oversampling method, whose main idea is to create new minority class examples by interpolating several minority class instances that lie together. SMOTE creates instances by randomly selecting one (or more depending on the oversampling ratio) of the k nearest neighbors (kNN) of a minority class instance and the generation of the new instance values from a random interpolation of both instances. Thus, the overfitting problem is avoided and causes the decision boundaries for the minority class to be spread further into the majority class space.
4) *Modified synthetic minority oversampling technique* (MSMOTE) [82]: It is a modified version of SMOTE. This algorithm divides the instances of the minority class into three groups, *safe*, *border* and *latent noise* instances by the calculation of the distances among all examples. When MSMOTE generates new examples, the strategy to select the nearest neighbors is changed with respect to SMOTE that depends on the group previously assigned to the instance. For safe instances, the algorithm randomly selects a data point from the kNN (same way as SMOTE); for border instances, it only selects the nearest neighbor; finally, for latent noise instances, it does nothing.
5) *Selective preprocessing of imbalanced data* (SPIDER) [52]: It combines local oversampling of the minority class with filtering difficult examples from the majority class. It consists in two phases, identification and preprocessing. The first one identifies which instances are flagged as noisy (misclassified) by kNN. The second phase depends on the option established (*weak, relabel,* or *strong*); when weak option is settled, it amplifies minority class instances; for relabel, it amplifies minority class examples and relabels majority class instances (i.e., changes class label); finally, using strong option, it strongly amplifies minority class instances. After carrying out these operations, the remaining noisy examples from the majority class are removed from the data-set.

III. State of the Art on Ensembles Techniques for Imbalanced Data-sets

In this section, we propose a new taxonomy for ensemble-based techniques to deal with imbalanced data-sets and we review the state of the art on these solutions. With this aim, we start recalling several classical learning algorithms for constructing sets of classifiers, whose classifiers properly complement each other, and then we get on with the ensemble-based solutions to address the class imbalance problem.

A. Learning Ensembles of Classifiers: Description and Representative Techniques

The main objective of ensemble methodology is to try to improve the performance of single classifiers by inducing several classifiers and combining them to obtain a new classifier that outperforms every one of them. Hence, the basic idea is to construct several classifiers from the original data and then aggregate their predictions when unknown instances are presented. This idea follows the human natural behavior that tends to seek several opinions before making any important decision. The main motivation for the combination of classifiers in redundant ensembles is to improve their generalization ability: each classifier is known to make errors, but since they are different (e.g., they have been trained on different data-sets or they have different behaviors over different part of the input space), misclassified examples are not necessarily the same [83]. Ensemble-based classifiers usually refer to the combination of classifiers that are minor variants of the same base classifier, which can be categorized in the broader concept of multiple classifier systems [25], [31], [32]. In this paper, we focus only on ensembles whose classifiers are constructed by manipulating the original data.

In the literature, the need of diverse classifiers to compose an ensemble is studied in terms of the statistical concepts of *bias-variance* decomposition [33], [84] and the related ambiguity [34] decomposition. The bias can be characterized as a measure of its ability to generalize correctly to a test set, whereas the variance can be similarly characterized as a measure of the extent to which the classifier's prediction is sensitive to the data on which it was trained. Hence, variance is associated with overfitting, the performance improvement in ensembles is usually due to a reduction in variance because the usual effect of ensemble averaging is to reduce the variance of a set of classifiers (some ensemble learning algorithms are also known to reduce bias [85]). On the other hand, ambiguity decomposition shows that, taking the combination of several predictors is better on average, over several patterns, than a method selecting one of the predictors at random. Anyway, these concepts are clearly stated in regression problems where the output is real-valued and the mean squared error is used as the loss function. However, in the context of classification, those terms are still ill-defined [35], [38], since different authors provide different assumptions [86]–[90] and there is no an agreement on their definition for generalized loss functions [91].

Nevertheless, despite not being theoretically clearly defined, diversity among classifiers is crucial (but alone is not enough) to form an ensemble, as shown by several authors [36]–[38]. Note also that, the measurement of the diversity and its relation to accuracy is not demonstrated [43], [92], but this is probably due to the measures of diversity rather than for not existing that relation. There are different ways to reach the required diversity, that is, different ensemble learning mechanisms. An important point is that the base classifiers should be *weak learners*; a classifier learning algorithm is said to be weak when low changes in data produce big changes in the induced model; this is why the most commonly used base classifiers are tree induction algorithms.

Considering a weak learning algorithm, different techniques can be used to construct an ensemble. The most widely used ensemble learning algorithms are AdaBoost [41] and Bagging [42] whose applications in several classification problems have led to significant improvements [27]. These methods provide a

Algorithm 1 Bagging

Input: S: Training set; T: Number of iterations; n: Bootstrap size; I: Weak learner

Output: Bagged classifier: $H(x) = sign\left(\sum_{t=1}^{T} h_t(x)\right)$ where $h_t \in [-1, 1]$ are the induced classifiers

1: **for** $t = 1$ to T **do**
2: $S_t \leftarrow$ RandomSampleReplacement(n,S)
3: $h_t \leftarrow$ I(S_t)
4: **end for**

way in which the classifiers are strategically generated to reach the diversity needed, by manipulating the training set before learning each classifier.

From this point, we briefly recall Bagging (including the modification called pasting small votes with importance sampling) and Boosting (AdaBoost and its variants AdaBoost.M1 and AdaBoost.M2) ensemble learning algorithms, which have been then integrated with previously explained preprocessing techniques in order to deal with the class imbalance problem.

1) *Bagging*: Breiman [42] introduced the concept of *bootstrap aggregating* to construct ensembles. It consists in training different classifiers with bootstrapped replicas of the original training data-set. That is, a new data-set is formed to train each classifier by randomly drawing (with replacement) instances from the original data-set (usually, maintaining the original data-set size). Hence, diversity is obtained with the resampling procedure by the usage of different data subsets. Finally, when an unknown instance is presented to each individual classifier, a majority or weighted vote is used to infer the class. Algorithm 1 shows the pseudocode for Bagging.

   Pasting small votes is a variation of Bagging originally designed for large data-sets [93]. Large data-sets are partitioned into smaller subsets, which are used to train different classifiers. There exist two variants, *Rvotes* that creates the data subsets at random and *Ivotes* that create consecutive data-sets based on the importance of the instances; important instances are those that improve diversity. The way used to create the data-sets consists in the usage of a balanced distribution of easy and difficult instances. Difficult instances are detected by *out-of-bag* classifiers [42], that is, an instance is considered difficult when it is misclassified by the ensemble classifier formed of those classifiers which did not use the instance to be trained. These difficult instances are always added to the next data subset, whereas easy instances have a low chance to be included. We show the pseudocode for Ivotes in Algorithm 2.

2) *Boosting*: Boosting (also known as ARCing, adaptive resampling and combining) was introduced by Schapire in 1990 [40]. Schapire proved that a weak learner (which is slightly better than random guessing) can be turned into a strong learner in the sense of *probably approximately correct* (PAC) learning framework. AdaBoost [41] is the most representative algorithm in this family, it was the first applicable approach of Boosting, and it has been appointed as one of the top ten data mining algorithms [94]. AdaBoost is known to reduce bias (besides from variance) [85], and similarly to support vector machines (SVMs) boosts the margins [95]. AdaBoost uses the whole data-set to train each classifier serially, but after each round, it gives more focus to difficult instances, with the goal of correctly classifying examples in the next iteration that were incorrectly classified during the current iteration. Hence, it gives more focus to examples that are harder to classify, the quantity of focus is measured by a weight, which initially is equal for all instances. After each iteration, the weights of misclassified instances are increased; on the contrary, the weights of correctly classified instances are decreased. Furthermore, another weight is assigned to each individual classifier depending on its overall accuracy which is then used in the test phase; more confidence is given to more accurate classifiers. Finally, when a new instance is submitted, each classifier gives a weighted vote, and the class label is selected by majority.

Algorithm 2 Ivotes

Input: S: Training set; T: Number of iterations; n: Bootstrap size; I: Weak learner

Output: Bagged classifier: $H(x) = sign\left(\sum_{t=1}^{T} h_t(x)\right)$ where $h_t \in [-1, 1]$ are the induced classifiers

1: $e_{new} \leftarrow 0.5$
2: **repeat**
3: $e_{old} \leftarrow e_{new}$
4: $S_t \leftarrow \emptyset$
5: **while** $size(S_t) < n$ **do** {Importance sampling}
6: $x \leftarrow$ RandomInstance(S)
7: **if** x misclassified by out-of-bag classifier **then**
8: $S_t \leftarrow S_t \cup \{x\}$
9: **else**
10: $S_t \leftarrow S_t \cup \{x\}$ with probability $\frac{e_{old}}{1-e_{old}}$
11: **end if**
12: **end while**
13: $h_t \leftarrow$ I(S_t)
14: $e_{new} \leftarrow$ error of out-of-bag classifier
15: **until** $e_{new} > e_{old}$

In this work, we will use the original two-class AdaBoost (Algorithm 3) and two of its very well-known modifications [41], [96] that have been employed in imbalanced domains: AdaBoost.M1 and AdaBoost.M2. The former is the first extension to multiclass classification with a different weight changing mechanism (Algorithm 4); the latter is the second extension to multiclass, in this case, making use of base classifiers' confidence rates (Algorithm 5). Note that neither of these algorithms by itself deal with the imbalance problem directly; both have to be changed or combined with another technique, since they focus their attention on difficult examples without differentiating their class. In an imbalanced data-set, majority class examples contribute more to the accuracy (they are more probably difficult examples); hence, rather than trying to improve the true positives, it is easier to improve the true negatives, also increasing the false negatives, which is not a desired characteristic.

Algorithm 3 AdaBoost

Input: Training set $S = \{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, N$; and $y_i \in \{-1, +1\}$; T: Number of iterations; I: Weak learner

Output: Boosted classifier: $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$ where h_t, α_t are the induced classifiers (with $h_t(x) \in \{-1, 1\}$) and their assigned weights, respectively

1: $D_1(i) \leftarrow 1/N$ for $i = 1, \ldots, N$
2: **for** $t = 1$ to T **do**
3: $h_t \leftarrow I(S, D_t)$
4: $\varepsilon_t \leftarrow \sum_{i, y_i \neq h_t(\mathbf{x}_i)} D_t(i)$
5: **if** $\varepsilon_t > 0.5$ **then**
6: $T \leftarrow t - 1$
7: **return**
8: **end if**
9: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
10: $D_{t+1}(i) = D_t(i) \cdot e^{(-\alpha_t h_t(\mathbf{x}_i) y_i)}$ for $i = 1, \ldots, N$
11: Normalize D_{t+1} to be a proper distribution
12: **end for**

Algorithm 4 AdaBoost.M1

Input: Training set $S = \{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, N$; and $y_i \in \mathbb{C}, \mathbb{C} = \{c_1, \ldots, c_m\}$; T: Number of iterations; I: Weak learner

Output: Boosted classifier:

$$H(x) = \arg\max_{y \in \mathbb{C}} \sum_{t=1}^{T} \ln\left(\frac{1}{\beta_t}\right) [h_t(x) = y] \text{ where } h_t, \beta_t$$

are the induced classifiers (with $h_t(x) \in \mathbb{C}$) and their assigned weights, respectively

1: $D_1(i) \leftarrow 1/N$ for $i = 1, \ldots, N$
2: **for** $t = 1$ to T **do**
3: $h_t \leftarrow I(S, D_t)$
4: $\varepsilon_t \leftarrow \sum_{i=1}^{N} D_t(i)[h_t(\mathbf{x}_i) \neq y_i]$
5: **if** $\varepsilon_t > 0.5$ **then**
6: $T \leftarrow t - 1$
7: **return**
8: **end if**
9: $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$
10: $D_{t+1}(i) = D_t(i) \cdot \beta_t^{1-[h_t(\mathbf{x}_i) \neq y_i]}$ for $i = 1, \ldots, N$
11: Normalize D_{t+1} to be a proper distribution
12: **end for**

Algorithm 5 AdaBoost.M2

Input: Training set $S = \{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, N$; and $y_i \in \mathbb{C}, \mathbb{C} = \{c_1, \ldots, c_m\}$; T: Number of iterations; I: Weak learner

Output: Boosted classifier:

$$H(x) = \arg\max_{y \in \mathbb{C}} \sum_{t=1}^{T} \ln\left(\frac{1}{\beta_t}\right) h_t(x, y)$$

where h_t, β_t (with $h_t(x, y) \in [0, 1]$) are the classifiers and their assigned weights, respectively

1: $D_1(i) \leftarrow 1/N$ for $i = 1, \ldots, N$
2: $w_{i,y}^1 \leftarrow D(i)/(m-1)$ for $i = 1, \ldots, N,\ y \in \mathbb{C} - \{y_i\}$
3: **for** $t = 1$ to T **do**
4: $W_i^t \leftarrow \sum_{y \neq y_i} w_{i,y}^t$
5: $q_t(i, y) \leftarrow \frac{w_{i,y}^t}{W_i^t}$ for $y \neq y_i$
6: $D_t(i) \leftarrow \frac{W_i^t}{\sum_{i=i}^{N} W_i^t}$
7: $h_t \leftarrow I(S, D_t)$
8: $\epsilon_t \leftarrow \frac{1}{2} \sum_{i=1}^{N} D_t(i) \left(1 - h_t(\mathbf{x}_i, y_i) + \sum_{i, y \neq y_i} q_t(i, y) h_t(\mathbf{x}_i, y)\right)$
9: $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
10: $w_{i,y}^{t+1} = w_{i,y}^t \cdot \beta_t^{\frac{1}{2}(1 + h_t(\mathbf{x}_i, y_i) - h_t(\mathbf{x}_i, y))}$ for $i = 1, \ldots, N,\ y \in \mathbb{C} - \{y_i\}$
11: **end for**

B. Addressing Class Imbalance Problem With Classifier Ensembles

As we have stated, in recent years, ensemble of classifiers have arisen as a possible solution to the class imbalance problem attracting great interest among researchers [45], [47], [50], [62]. In this section, our aim is to review the application of ensemble learning methods to deal with this problem, as well as to present a taxonomy where these techniques can be categorized. Furthermore, we have selected several significant approaches from each family of our taxonomy to develop an exhaustive experimental study that we will carry out in Section V.

To start with the description of the taxonomy, we show our proposal in Fig. 3, where we categorize the different approaches. Mainly, we distinguish four different families among ensemble approaches for imbalanced learning. On the one hand, cost-sensitive boosting approaches, which are similar to cost-sensitive methods, but where the costs minimization is guided by the boosting algorithm. On the other hand, we difference three more families that have a characteristic in common; all of them consist in embedding a data preprocessing technique in an ensemble learning algorithm. We categorize these three families depending on the ensemble learning algorithm they use. Therefore, we consider boosting- and bagging-based ensembles, and the last family is formed by hybrids ensembles. That is, ensemble methods that apart from combining an ensemble learning algorithm and a preprocessing technique, make use of both boosting and bagging, one inside the other, together with a preprocessing technique.

Next, we look over these families, reviewing the existing works and focusing in the most significant proposals that we use in the experimental analysis.

1) Cost-sensitive Boosting: AdaBoost is an accuracy-oriented algorithm, when the class distribution is uneven, this strategy biases the learning (the weights) toward the majority class, since it contributes more to the overall accuracy. For this reason, there have been different proposals that modify the weight update of AdaBoost (Algorithm 3, line 10 and, as a consequence, line 9). In such a way, examples from different classes are not equally treated. To reach this unequal treatment, cost-sensitive approaches keep the general learning framework of AdaBoost, but at the same time introduce cost items into the weight update formula. These proposals usually differ in the way that they modify the weight update rule, among this family AdaCost [48], CSB1, CSB2 [49], RareBoost [97], AdaC1, AdaC2, and AdaC3 [50] are the most representative approaches.

1) *AdaCost:* In this algorithm, the weight update is modified by adding a *cost adjustment function* φ. This function, for an instance with a higher cost factor increases its weight "more" if the instance is misclassified, but decreases its weight "less" otherwise. Being C_i the cost

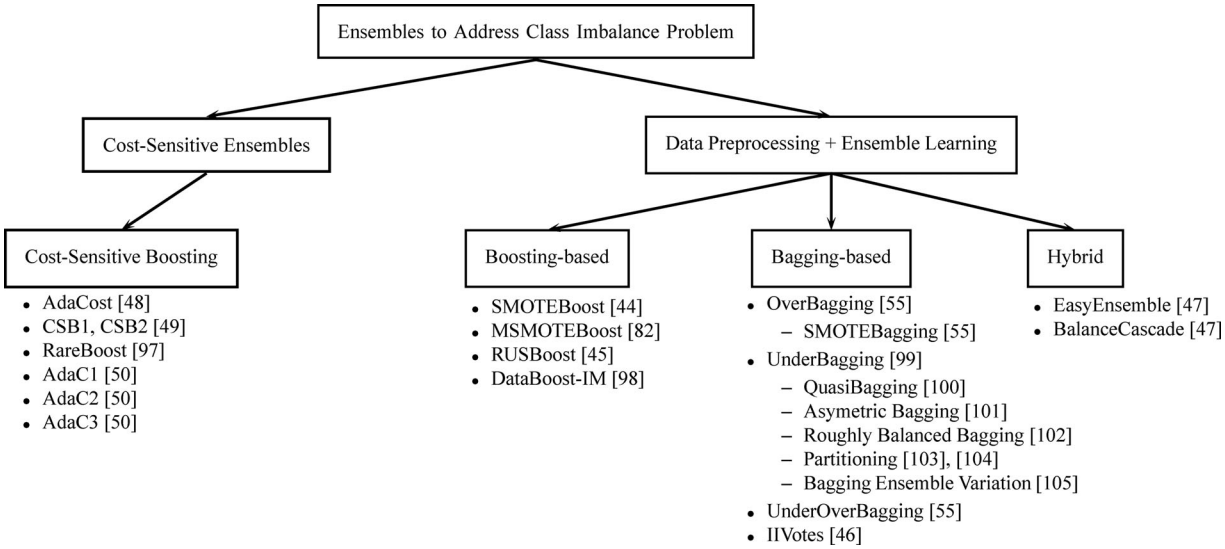


Fig. 3. Proposed taxonomy for ensembles to address the class imbalance problem.

of misclassifying the ith example, the authors provide their recommended function as $\varphi_+ = -0.5C_i + 0.5$ and $\varphi_- = 0.5C_i + 0.5$. The weighting function and the computation of α_t are replaced by the following formulas:

$$D_{t+1}(i) = D_t(i) \cdot e^{-\alpha_t y_i h_t(\mathbf{x}_i)\varphi_{\text{sign}(h_t(\mathbf{x}_i),y_i)}} \tag{3}$$

$$\alpha_t = \frac{1}{2}\ln\frac{1+\sum_i D_t(i)\cdot e^{-\alpha_t y_i h_t(\mathbf{x}_i)\varphi_{\text{sign}(h_t(\mathbf{x}_i),y_i)}}}{1-\sum_i D_t(i)\cdot e^{-\alpha_t y_i h_t(\mathbf{x}_i)\varphi_{\text{sign}(h_t(\mathbf{x}_i),y_i)}}} \tag{4}$$

2) *CSB:* Neither CSB1 nor CSB2 use an adjustment function. Moreover, these approaches only consider the costs in the weight update formula, that is, none of them changes the computation of α_t. CSB1 because it does not use α_t anymore ($\alpha_t = 1$) and CSB2 because it uses the same α_t computed by AdaBoost. In these cases, the weight update is replaced by

$$D_{t+1}(i) = D_t(i)C_{\text{sign}(h_t(\mathbf{x}_i),y_i)} \cdot e^{-\alpha_t y_i h_t(\mathbf{x}_i)} \tag{5}$$

where $C_+ = 1$ and $C_- = C_i \geq 1$ are the costs of misclassifying a positive and a negative example, respectively.

3) *RareBoost:* This modification of AdaBoost tries to tackle the class imbalance problem by simply changing α_t's computation (Algorithm 3, line 9) making use of the confusion matrix in each iteration. Moreover, they compute two different α_t values in each iteration. This way, false positives (FP_t is the weights' sum of FP in the tth iteration) are scaled in proportion to how well they are distinguished from true positives (TP_t), whereas false negatives (FN_t) are scaled in proportion to how well they are distinguished from true negatives (TN_t). On the one hand, $\alpha_t^p = TP_t/FP_t$ is computed for examples predicted as positives. On the other hand, $\alpha_t^n = TN_t/FN_t$ is computed for the ones predicted as negatives. Finally, the weight update is done separately by the usage of both factors depending on the predicted class of each instance. Note that, despite we have include RareBoost in cost-sensitive boosting family, it does not directly make use of costs, which can be an advantage, but it modifies AdaBoost algorithm in a similar way to the approaches in this family. Because of this fact, we have classified into this group. However, this algorithm has a handicap, TP_t and TN_t are reduced, and FP_t and FT_t are increased only if $TP_t > FP_t$ and $TN_t > FN_t$, that is equivalent to require an accuracy of the positive class greater than 50%:

$$TP_t/(TP_t + FP_t) > 0.5. \tag{6}$$

This constraint is not trivial when dealing with the class imbalance problem; moreover, it is a strong condition. Without satisfying this condition, the algorithm will collapse. Therefore, we will not include it in our empirical study.

4) *AdaC1:* This algorithm is one of the three modifications of AdaBoost proposed in [50]. The authors proposed different ways in which the costs can be embedded into the weight update formula (Algorithm 3, line 10). They derive different computations of α_t depending on where they introduce the costs. In this case, the cost factors are introduced within the exponent part of the formula:

$$D_{t+1}(i) = D_t(i) \cdot e^{-\alpha_t C_i h_t(\mathbf{x}_i) y_i} \tag{7}$$

where $C_i \in [0, +\infty)$. Hence, the computation of the classifiers' weight is done as follows:

$$\alpha_t = \tfrac{1}{2}\ln\frac{1+\sum_{i,y_i=h_t(\mathbf{x}_i)} C_i D_t(i) - \sum_{i,y_i\neq h_t(\mathbf{x}_i)} C_i D_t(i)}{1-\sum_{i,y_i=h_t(\mathbf{x}_i)} C_i D_t(i) + \sum_{i,y_i\neq h_t(\mathbf{x}_i)} C_i D_t(i)}. \tag{8}$$

Note that AdaCost is a variation of AdaC1 where there is a cost adjustment function instead of a cost item inside the exponent. Though, in the case of AdaCost, it does not reduce to the AdaBoost algorithm when both classes are equally weighted (contrary to AdaC1).

5) *AdaC2:* Likewise AdaC1, AdaC2 integrates the costs in the weight update formula. But the procedure is different;

the costs are introduced outside the exponent part:

$$D_{t+1}(i) = C_i D_t(i) \cdot e^{-\alpha_t h_t(\mathbf{x}_i) y_i}. \quad (9)$$

In consequence, α_t's computation is changed:

$$\alpha_t = \frac{1}{2} \ln \frac{\sum_{i, y_i = h_t(\mathbf{x}_i)} C_i D_t(i)}{\sum_{i, y_i \neq h_t(\mathbf{x}_i)} C_i D_t(i)}. \quad (10)$$

6) *AdaC3:* This modification considers the idea of AdaC1 and AdaC2 at the same time. The weight update formula is modified by introducing the costs both inside and outside the exponent part:

$$D_{t+1}(i) = C_i D_t(i) \cdot e^{-\alpha_t C_i h_t(\mathbf{x}_i) y_i}. \quad (11)$$

In this manner, over again α_t changes:

$$\alpha_t = \frac{1}{2} \ln \frac{\sum_i C_i D_t(i) + \sum_{i, y_i = h_t(\mathbf{x}_i)} C_i^2 D_t(i) - \sum_{i, y_i \neq h_t(\mathbf{x}_i)} C_i^2 D_t(i)}{\sum_i C_i D_t(i) - \sum_{i, y_i = h_t(\mathbf{x}_i)} C_i^2 D_t(i) + \sum_{i, y_i \neq h_t(\mathbf{x}_i)} C_i^2 D_t(i)}. \quad (12)$$

2) Boosting-based Ensembles: In this family, we have included the algorithms that embed techniques for data preprocessing into boosting algorithms. In such a manner, these methods alter and bias the weight distribution used to train the next classifier toward the minority class every iteration. Inside this family, we include SMOTEBoost [44], MSMOTEBoost [82], RUSBoost [45], and DataBoost-IM [98] algorithms.

a) *SMOTEBoost and MSMOTEBoost:* Both methods introduce synthetic instances just before Step 4 of AdaBoost.M2 (Algorithm 2), using the SMOTE and MSMOTE data preprocessing algorithms, respectively. The weights of the new instances are proportional to the total number of instances in the new data-set. Hence, their weights are always the same (in all iterations and for all new instances), whereas original data-set's instances weights are normalized in such a way that they form a distribution with the new instances. After training a classifier, the weights of the original data-set instances are updated; then another sampling phase is applied (again, modifying the weight distribution). The repetition of this process also brings along more diversity in the training data, which generally benefits the ensemble learning.

b) *RUSBoost:* In other respects, RUSBoost performs similarly to SMOTEBoost, but it removes instances from the majority class by random undersampling the data-set in each iteration. In this case, it is not necessary to assign new weights to the instances. It is enough with simply normalizing the weights of the remaining instances in the new data-set with respect to their total sum of weights. The rest of the procedure is the same as in SMOTEBoost.

c) *DataBoost-IM:* This approach is slightly different to the previous ones. Its initial idea is not different, it combines AdaBoost.M1 algorithm with a data generation strategy. Its major difference is that it first identifies hard examples (seeds) and then carries out a rebalance process, always for both classes. At the beginning, the N_s instances (as many as misclassified instances by the current classifier) with the largest weights are taken as seeds. Considering that $N_{\min}$ and N_{maj} are the number of instances of the minority and majority class, respectively; whereas N_{smin} and N_{smaj} are the number of seed instances of each class; $M_L = \min(N_{\text{maj}}/N_{\min}, N_{\text{smaj}})$ and $M_S = \min((N_{\text{maj}} \cdot M_L)/N_{\min}, N_{\text{smin}})$ minority and majority class instances are used as final seeds. Each seed produce N_{maj} or $N_{\min}$ new examples, depending on its class label. Nominal attributes' values are copied from the seed and the values of continuous attributes are randomly generated following a normal distribution with the mean and variance of class instances. Those instances are added to the original data-set with a weight proportional to the weight of the seed. Finally, the sums of weights of the instances belonging to each class are rebalanced, in such a way that both classes' sum is equal. The major drawback of this approach is its incapability to deal with highly imbalanced data-sets, because it generates an excessive amount of instances which are not manageable for the base classifier (i.e., $N_{\text{maj}} = 3000$ and $N_{\min} = 29$ with $Err = 15\%$, there will be 100 seed instances, where 71 have to be from the majority class and at least $71 \cdot 3000 = 213000$ new majority instances are generated in each iteration). For this reason, we will not analyze it in the experimental study.

3) Bagging-based Ensembles: Many approaches have been developed using bagging ensembles to deal with class imbalance problems due to its simplicity and good generalization ability. The hybridization of bagging and data preprocessing techniques is usually simpler than their integration in boosting. A bagging algorithm does not require to recompute any kind of weights; therefore, neither is necessary to adapt the weight update formula nor to change computations in the algorithm. In these methods, the key factor is the way to collect each bootstrap replica (Step 2 of Algorithm 1), that is, how the class imbalance problem is dealt to obtain a useful classifier in each iteration without forgetting the importance of the diversity.

We distinguish four main algorithms in this family, OverBagging [55], UnderBagging [99], UnderOverBagging [55], and IIVotes [46]. Note that, we have grouped several approaches into OverBagging and UnderBagging due to their similarity as we explain hereafter.

a) *OverBagging:* An easy way to overcome the class imbalance problem in each bag is to take into account the classes of the instances when they are randomly drawn from the original data-set. Hence, instead of performing a random sampling of the whole data-set, an oversampling process can be carried out before training each classifier (OverBagging). This procedure can be developed in at least two ways. Oversampling consists in increasing the number of minority class instances by their replication, all majority class instances can be included in the new bootstrap, but another option is to resample them trying to increase the diversity. Note that in OverBagging all instances will probably take part in at least one bag, but each bootstrapped replica will contain many more instances than the original data-set.

On the other hand, another different manner to oversample minority class instances can be carried out by the

usage of the SMOTE preprocessing algorithm. SMOTEBagging [55] differs from the use of random oversampling not only because the different preprocessing mechanism. The way it creates each bag is significantly different. As well as in OverBagging, in this method both classes contribute to each bag with N_{maj} instances. But, a SMOTE resampling rate ($a\%$) is set in each iteration (ranging from 10% in the first iteration to 100% in the last, always being multiple of 10) and this ratio defines the number of positive instances ($a\% \cdot N_{\mathrm{maj}}$) randomly resampled (with replacement) from the original data-set in each iteration. The rest of the positive instances are generated by the SMOTE algorithm. Besides, the set of negative instances is bootstrapped in each iteration in order to form a more diverse ensemble.

b) *UnderBagging:* On the contrary to OverBagging, UnderBagging procedure uses undersampling instead of oversampling. However, in the same manner as OverBagging, it can be developed in at least two ways. The undersampling procedure is usually only applied to the majority class; however, a resampling with replacement of the minority class can also be applied in order to obtain *a priori* more diverse ensembles. Point out that, in UnderBagging it is more probable to ignore some useful negative instances, but each bag has less instances than the original data-set (on the contrary to OverBagging).

On the one hand, the UnderBagging method has been used with different names, but maintaining the same functional structure, e.g., Asymmetric Bagging [101] and QuasiBagging [100]. On the other hand, roughly-balanced Bagging [102] is quite similar to UnderBagging, but it does not bootstrap a totally balanced bag. The number of positive examples is kept fixed (by the usage of all of them or resampling them), whereas the number of negative examples drawn in each iteration varies slightly following a negative binomial distribution (with $q = 0.5$ and $n = N_{\mathrm{min}}$). Partitioning [103], [104] (also called Bagging Ensemble Variation [105]) is another way to develop the undersampling, in this case, the instances of the majority class are divided into IR disjoint data-sets and each classifier is trained with one of those bootstraps (mixed with the minority class examples).

c) *UnderOverBagging:UnderBagging to OverBagging* follows a different methodology from OverBagging and UnderBagging, but similar to SMOTEBagging to create each bag. It makes use of both oversampling and undersampling techniques; a resampling rate ($a\%$) is set in each iteration (ranging from 10% to 100% always being multiple of 10); this ratio defines the number of instances taken from each class ($a\% \cdot N_{\mathrm{maj}}$ instances). Hence, the first classifiers are trained with a lower number of instances than the last ones. This way, the diversity is boosted.

d) *IIVotes:Imbalanced IVotes* is based on the same combination idea, but it integrates the SPIDER data preprocessing technique with IVotes (a preprocessing phase is applied in each iteration before Step 13 of Algorithm 2). This method has the advantage of not needing to define the number of bags, since the algorithm stops when the out-of-bag error estimation no longer decreases.

4) Hybrid Ensembles: The main difference of the algorithms in this category with respect to the previous ones is that they carry out a double ensemble learning, that is, they combine both bagging and boosting (also with a preprocessing technique). Both algorithms that use this hybridization were proposed in [47], and were referred to as exploratory undersampling techniques. EasyEnsemble and BalanceCascade use Bagging as the main ensemble learning method, but in spite of training a classifier for each new bag, they train each bag using AdaBoost. Hence, the final classifier is an ensemble of ensembles.

In the same manner as UnderBagging, each balanced bag is constructed by randomly undersampling instances from the majority class and by the usage of all the instances from the minority class. The difference between these methods is the way in which they treat the negative instances after each iteration, as explained in the following.

a) *EasyEnsemble:* This approach does not perform any operation with the instances from the original data-set after each AdaBoost iteration. Hence, all the classifiers can be trained in parallel. Note that, EasyEnsemble can be seen as an UnderBagging where the base learner is AdaBoost, if we fix the number of classifiers, EasyEnsemble will train less bags than UnderBagging, but more classifiers will be assigned to learn each single bag.

b) *BalanceCascade:* BalanceCascade works in a supervised manner, and therefore the classifiers have to be trained sequentially. In each bagging iteration after learning the AdaBoost classifier, the majority class examples that are correctly classified with higher confidences by the current trained classifiers are removed from the data-set, and they are not taken into account in further iterations.

IV. Experimental Framework

In this section, we present the framework used to carry out the experiments analyzed in Section V. First, we briefly describe the algorithms from the proposed taxonomy that we have included in the study and we show their set-up parameters in Subsection IV-A. Then, we provide details of the real-world imbalanced problems chosen to test the algorithms in Subsection IV-B. Finally, we present the statistical tests that we have applied to make a proper comparison of the classifiers' results in Subsection IV-C. We should recall that we are focusing on two-class problems.

A. Algorithms and Parameters

In first place, we need to define a baseline classifier which we use in all the ensembles. With this goal, we will use C4.5 decision tree generating algorithm [58]. Almost all the ensemble methodologies we are going to test were proposed in combination with C4.5. Furthermore, it has been widely used to deal with imbalanced data-sets [59]–[61], and C4.5 has also been included as one of the top-ten data-mining algorithms [94]. Because of these facts, we have chosen it as the most appropriate base learner. C4.5 learning algorithm constructs the decision

TABLE II
PARAMETER SPECIFICATION FOR C4.5

Parameters
Prune = True
Confidence level = 0.25
Minimum number of item-sets per leaf = 2
Confidence = Laplace Smoothing [107]

tree top-down by the usage of the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is the one used to make the decision. In Table II, we show the configuration parameters that we have used to run C4.5. We acknowledge that we could consider the use of a classification tree algorithm, such as Hellinger distance tree [106], that is specifically designed for the solution of imbalanced problems. However, in [106], the authors show that it often experiences a reduction in performance when sampling techniques are applied, which is the base of the majority of the studied techniques; moreover, being more robust (less weak) than C4.5 in the imbalance scenario, the diversity of the ensembles could be hindered.

Besides the ensemble-based methods that we consider, we include another nonensemble technique to be able to analyze whether the use of ensembles is beneficial, not only with respect to the original base classifier, but also to outperform the results of the classifier trained over preprocessed data-sets. To do so, before learning the decision trees, we use SMOTE preprocessing algorithm to rebalance the data-sets before the learning stage (see Section II-D). Previous works have shown the positive synergy of this combination leading to significant improvements [20], [53].

Regarding ensemble learning algorithms, on the one hand, we include classic ensembles (which are not specifically developed for imbalanced domains) such as Bagging, AdaBoost, AdaBoot.M1, and AdaBoost.M2. On the other hand, we include the algorithms that are designed to deal with skewed class distributions in the data-sets which, following the taxonomy proposed in Section III-B, are distinguished into four families: Cost-sensitive Boosting, Boosting-based, Bagging-based, and Hybrid ensembles.

Concerning the cost-sensitive boosting framework, a thorough empirical study was presented in [50]. To avoid the repetition of similar experiments, we will follow the results where AdaC2 algorithm stands out with respect to the others. Hence, we will empirically study this algorithm among the ones from this family in the experimental study.

Note that in our experiments we want to analyze which is the most robust method among ensemble approaches, that is, given a large variety of problems which one is more capable of assessing an overall good (better) performance in all the problems. Robustness concept also has an implicit meaning of generality, algorithms whose configuration parameters have to be tuned depending on the data-set are less robust, since changes in the data can easily worsen their results; hence, they have more difficulties to be adapted to new problems.

Recall from Section II-C that cost-sensitive approaches' weakness is the need of costs definition. These costs are not usually presented in classification data-sets, and on this account, they are usually set ad-hoc or found conducting a search in the space of possible costs. Therefore, in order to execute AdaC2, we set the costs depending on the IR of each data-set. In other words, we set up an adaptive cost strategy, where the cost of misclassifying a minority class instance is always $C_{\min} = 1$, whereas that of misclassifying a majority class instance is inversely proportional to the IR of the data-set ($C_{\mathrm{maj}} = 1/IR$).

The Boosting-based ensembles that are considered in our study are RUSBoost, SMOTEBoost and MSMOTEBoost. As we have explained, DataBoost-IM approach is not capable of dealing with some of the data-sets that are used in the study (more details in Subsection IV-B).

With respect to Bagging-based ensembles, we include from the OverBagging group, OverBagging (which uses random oversampling) and SMOTEBagging due to the great difference in their way to perform the oversampling to create each bag. In the same manner that we use MSMOTEBoost, in this case, we have also developed a MSMOTEBagging algorithm, whose unique difference with SMOTEBagging is the use of MSMOTE instead of SMOTE. Hence, we are able to analyze the suitability of their integration in both Boosting and Bagging. Among UnderBagging methods, we consider the random undersampling method to create each balanced bag. We discard rest of the approaches (e.g., roughly balanced bagging or partitioning) given their similarity; hence, we only develop the more general version. For UnderBagging and OverBagging, we incorporate their both possible variations (resampling of both classes in each bag and resampling of only one of them), in such a way that we can analyze their influence in the diversity of the ensemble. The set of Bagging-based ensembles ends with UnderOverBagging and the combination of SPIDER with IVotes, for IIVotes algorithm we have tested the three configurations of SPIDER.

Finally, we consider both hybrid approaches, EasyEnsemble, and BalanceCascade.

For the sake of clarity for the reader, Table III summarizes the whole list of algorithms grouped by families, we also show the abbreviations that we will use along the experimental study and a short description.

In our experiments, we want all methods to have the same opportunities to achieve their best results, but always without fine-tuning their parameters depending on the data-set. Generally, the higher the number of base classifiers, the better the results we achieve; however, this does not occur in every method (i.e., more classifiers without spreading diversity could worsen the results and they could also produce overfitting). Most of the reviewed approaches employ ten base classifiers by default, but others such as EasyEnsemble and BalanceCascade need more classifiers to make sense (since they train each bag with AdaBoost). In that case, the authors use a total of 40 classifiers (four bagging iterations and ten AdaBoost iterations per bag). On this account, we will first study which configuration is more appropriate for each ensemble method and then we will follow with the intrafamily and interfamily comparisons. Table IV shows the rest of the parameters required by the algorithms we

TABLE III
ALGORITHMS USED IN THE EXPERIMENTAL STUDY

Non-ensemble Classifiers		
Abbr.	*Method*	*Short Description*
C45	C4.5	Classic C4.5 decision tree learning algorithm
SMT	SMOTE + C4.5	C4.5 applied on data-sets previously preprocessed with SMOTE
Classic Ensembles		
Abbr.	*Method*	*Short Description*
ADAB	AdaBoost	Classic AdaBoost, without using confidences
M1	AdaBoost.M1	Multi-class AdaBoost, slightly different weight update
M2	AdaBoost.M2	Multi-class AdaBoost using confidence estimates
BAG	Bagging	Classic Bagging, resampling with replacement, bag size equal to orignal data-set size
Cost-sensitive Boosting Ensembles		
Abbr.	*Method*	*Short Description*
C2	AdaC2	AdaBoost with costs outside the exponent
Boosting-based Ensembles		
Abbr.	*Method*	*Short Description*
RUS	RUSBoost	AdaBoost.M2 with random undersampling in each iteration
SBO	SMOTEBoost	AdaBoost.M2 with SMOTE in each iteration
MBO	MSMOTEBoost	AdaBoost.M2 with MSMOTE in each iteration
Bagging-based Ensembles		
Abbr.	*Method*	*Short Description*
UB	UnderBagging	Bagging with undersampling of the majority class, data-set size doubles the number of positive instances
UB2	UnderBagging2	Bagging with resampling of both classes (balance), data-set size doubles the number of positive instances
OB	OverBagging	Bagging with oversampling of the minority class, data-set size doubles the number of negative instances
OB2	OverBagging2	Bagging with resampling of both classes (balance), data-set size doubles the number of negative instances
UOB	UnderOverBagging	Bagging where the number of instances of each bag varies (from undersampling to oversampling)
SBAB	SMOTEBagging	Bagging where each bag's SMOTE quantity varies
MBAG	MSMOTEBagging	Bagging where each bag's MSMOTE quantity varies
SPw	IIVotes weak	IVotes with SPIDER (weak) in each iteration
SPr	IIVotes relabel	IVotes with SPIDER (relabel) in each iteration
SPs	IIVotes strong	IVotes with SPIDER (strong) in each iteration
Hybrid Ensembles		
Abbr.	*Method*	*Short Description*
EASY	EasyEnsemble	UB2 but learning each bag with AdaBoost
BAL	BalanceCascade	Similar to EASY but removing examples from the majority class in each bagging iteration

have used in the experiments, which are the parameters recommended by their authors. All experiments have been developed using the KEEL[1] software [56], [57].

B. Data-sets

In the study, we have considered 44 binary data-sets from KEEL data-set repository [56], [57], which are publicly available on the corresponding web-page,[2] which includes general information about them. Multiclass data-sets were modified to obtain two-class imbalanced problems so that the union of one or more classes became the positive class and the union of one or more of the remaining classes was labeled as the negative class. This way, we have different IRs: from low imbalance to highly imbalanced data-sets. Table V summarizes the properties of the selected data-sets: for each data-set, the number of

[1]http://www.keel.es
[2]http://www.keel.es/dataset.php

TABLE IV
CONFIGURATION PARAMETERS FOR THE ALGORITHMS USED IN THE EXPERIMENTAL STUDY

Algorithm	*Parameters*
SMOTE	Number of Neighbors k = 5 Quantity = Balance Distance = Heterogeneous Value Difference Metric (HVDM)
C2	$C_{min} = 1$ $C_{maj} = 1/IR$
SBO MBO	Use SMOTE's configuration
SPIDER	Number of Neighbors k = 5
EASY BAL	Number of Bags = 4 AdaBoost Iterations = 10

TABLE V
SUMMARY DESCRIPTION OF THE IMBALANCED DATA-SETS USED IN THE EXPERIMENTAL STUDY

Data-set	#Ex.	#Atts.	Class (min;maj)	(%min;%maj)	IR
Glass1	214	9	(build-win-non_float-proc; remainder)	(35.51, 64.49)	1.82
Ecoli0vs1	220	7	(im; cp)	(35.00, 65.00)	1.86
Wisconsin	683	9	(malignant; benign)	(35.00, 65.00)	1.86
Pima	768	8	(tested-positive; tested-negative)	(34.84, 66.16)	1.90
Iris0	150	4	(Iris-Setosa; remainder)	(33.33, 66.67)	2.00
Glass0	214	9	(build-win-float-proc; remainder)	(32.71, 67.29)	2.06
Yeast1	1484	8	(nuc; remainder)	(28.91, 71.09)	2.46
Vehicle1	846	18	(Saab; remainder)	(28.37, 71.63)	2.52
Vehicle2	846	18	(Bus; remainder)	(28.37, 71.63)	2.52
Vehicle3	846	18	(Opel; remainder)	(28.37, 71.63)	2.52
Haberman	306	3	(Die; Survive)	(27.42, 73.58)	2.68
Glass0123vs456	214	9	(non-window glass; remainder)	(23.83, 76.17)	3.19
Vehicle0	846	18	(Van; remainder)	(23.64, 76.36)	3.23
Ecoli1	336	7	(im; remainder)	(22.92, 77.08)	3.36
New-thyroid2	215	5	(hypo; remainder)	(16.89, 83.11)	4.92
New-thyroid1	215	5	(hyper; remainder)	(16.28, 83.72)	5.14
Ecoli2	336	7	(pp; remainder)	(15.48, 84.52)	5.46
Segment0	2308	19	(brickface; remainder)	(14.26, 85.74)	6.01
Glass6	214	9	(headlamps; remainder)	(13.55, 86.45)	6.38
Yeast3	1484	8	(me3; remainder)	(10.98, 89.02)	8.11
Ecoli3	336	7	(imU; remainder)	(10.88, 89.12)	8.19
Page-blocks0	5472	10	(remainder; text)	(10.23, 89.77)	8.77
Yeast2vs4	514	8	(cyt; me2)	(9.92, 90.08)	9.08
Yeast05679vs4	528	8	(me2; mit,me3,exc,vac,erl)	(9.66, 90.34)	9.35
Vowel0	988	13	(hid; remainder)	(9.01, 90.99)	10.10
Glass016vs2	192	9	(ve-win-float-proc; build-win-float-proc, build-win-non_float-proc, headlamps)	(8.89, 91.11)	10.29
Glass2	214	9	(ve-win-float-proc; remainder)	(8.78, 91.22)	10.39
Ecoli4	336	7	(om; remainder)	(6.74, 93.26)	13.84
Yeast1vs7	459	8	(nuc; vac)	(6.72, 93.28)	13.87
Shuttle0vs4	1829	9	(Rad Flow; Bypass)	(6.72, 93.28)	13.87
Glass4	214	9	(containers; remainder)	(6.07, 93.93)	15.47
Page-blocks13vs2	472	10	(graphic; horiz.line,picture)	(5.93, 94.07)	15.85
Abalone9vs18	731	8	(18; 9)	(5.65, 94.25)	16.68
Glass016vs5	184	9	(tableware; build-win-float-proc, build-win-non_float-proc headlamps)	(4.89, 95.11)	19.44
Shuttle2vs4	129	9	(Fpv Open; Bypass)	(4.65, 95.35)	20.5
Yeast1458vs7	693	8	(vac; nuc,me2,me3,pox)	(4.33, 95.67)	22.10
Glass5	214	9	(tableware; remainder)	(4.20, 95.80)	22.81
Yeast2vs8	482	8	(pox; cyt)	(4.15, 95.85)	23.10
Yeast4	1484	8	(me2; remainder)	(3.43, 96.57)	28.41
Yeast1289vs7	947	8	(vac; nuc,cyt,pox,erl)	(3.17, 96.83)	30.56
Yeast5	1484	8	(me1; remainder)	(2.96, 97.04)	32.78
Ecoli0137vs26	281	7	(pp,imL; cp,im,imU,imS)	(2.49, 97.51)	39.15
Yeast6	1484	8	(exc; remainder)	(2.49, 97.51)	39.15
Abalone19	4174	8	(19; remainder)	(0.77, 99.23)	128.87

examples (#Ex.), number of attributes (#Atts.), class name of each class (minority and majority), the percentage of examples of each class and the IR. This table is ordered according to this last column in the ascending order.

We have obtained the AUC metric estimates by means of a 5-fold cross-validation. That is, the data-set was split into five folds, each one containing 20% of the patterns of the data-set. For each fold, the algorithm is trained with the examples contained in the remaining folds and then tested with the current fold. The data partitions used in this paper can be found in KEEL-dataset repository [57] so that any interested researcher can reproduce the experimental study.

C. Statistical Tests

In order to compare different algorithms and to show whether there exist significant differences among them, we have to give the comparison a statistical support [108]. To do so, we use nonparametric tests according to the recommendations made in [63]–[65], [108], where a set of proper nonparametric tests for statistical comparisons of classifiers is presented. We need to use nonparametric tests because the initial conditions that guarantee the reliability of the parametric tests may not be satisfied causing the statistical analysis to lose its credibility [63].

In this paper, we use two types of comparisons: pairwise (between a pair of algorithms) and multiple (among a group of algorithms).

1) *Pairwise comparisons:* we use Wilcoxon paired signed-rank test [109] to find out whether there exist significant differences between a pair of algorithms.
2) *Multiple comparisons:* we first use the Iman–Davenport test [110] to detect statistical differences among a group of results. Then, if we want to check out if a control algorithm (usually the best one) is significantly better than the rest ($1 \times n$ comparison), we use the Holm post-hoc test [111]. Whereas, when we want to find out which algorithms are distinctive among an $n \times n$ comparison, we use the Shaffer post-hoc test [112]. The post-hoc procedures allow us to know whether a hypothesis of comparison of means could be rejected at a specified level of significance α (i.e., there exist significant differences). Besides, we compute the *p*-value associated with each comparison, which represents the lowest level of significance of a hypothesis that results in a rejection. In this manner, we can also know how different two algorithms are.

These tests are suggested in different studies [63]–[65], where their use in the field of machine learning is highly recommended. Any interested reader can find additional information on the Website http://sci2s.ugr.es/sicidm/, together with the software for applying the statistical tests.

Complementing the statistical analysis, we also consider the average ranking of the algorithms in order to show at a first glance how good a method is with respect to the rest in the comparison. The rankings are computed by first assigning a rank position to each algorithm in every data-set, which consists in assigning the first rank in a data-set (value 1) to the best performing algorithm, the second rank (value 2) to the second best algorithm, and so forth. Finally, the average ranking of a method is computed by the mean value of its ranks among all data-sets.

V. Experimental Study

In this section, we carry out the empirical comparison of the algorithms that we have reviewed. Our aim is to answer several questions about the reviewed ensemble learning algorithms in the scenario of two-class imbalanced problems.

1) In first place, we want to analyze which one of the approaches is able to better handle a large amount of imbalanced data-sets with different IR, i.e., to show which one is the most robust method.
2) We also want to investigate their improvement with respect to classic ensembles and to look into the appropriateness of their use instead of applying a unique preprocessing step and training a single classifier. That is, whether the trade-off between complexity increment and performance enhancement is justified or not.

Given the amount of methods in the comparison, we cannot afford it directly. On this account, we develop a hierarchical analysis (a tournament among algorithms). This methodology allows us to obtain a better insight on the results by discarding those algorithms which are not the best in a comparison. We divided the study into three phases, all of them guided by the nonparametric tests presented in Section IV-C:

1) *Number of classifiers:* In the first phase, we analyze which configuration of how many classifiers is the best for the algorithms that are configurable to be executed with both 10 and 40 classifiers. As we explained in Section IV-A, this phase allows us to give all of them the same opportunities.
2) *Intra-family comparison:* The second phase consists in analyzing each family separately. We investigate which of their components has the best (or only a better) behavior. Those methods will be then considered to take part on the final phase (representatives of the families).
3) *Inter-family comparison:* In the last phase, we develop a comparison among the representatives of each family. In such a way, our objective is to analyze which algorithm stands out from all of them as well as to study the behavior of ensemble-based methods to address the class imbalance problem with respect to the rest of the approaches considered.

Following this methodology, at the end, we will be able to account for the questions that we have set out. We divide this section into three subsections according to each one of the goals of the study, and a final one (Subsection V-D) where we discuss and sum up the results obtained in this study.

Before starting with the analysis, we show the overall train and test AUC results ($\pm$ for standard deviation) in Table VI. The detailed test results of all methods and data-sets are presented in the Appendix.

A. Number of Classifiers

We start investigating the configuration of the number of classifiers. This parameter is configurable in all except nonensembles, hybrids, and IIVotes methods.

TABLE VI
MEAN AUC TRAIN AND TEST RESULTS FOR ALL THE ALGORITHMS IN THE EXPERIMENTAL STUDY (± FOR STANDARD DEVIATION)

Non-ensemble		
Algorithm	AUC_{Tr}	AUC_{Tst}
C45	.8733 ± .0354	.7929 ± .0687
SMT	.9682 ± .0099	.8257 ± .0631
Classic Ensembles		
Algorithm	AUC_{Tr}	AUC_{Tst}
ADAB1	.9920 ± .0045	.8016 ± .0613
ADAB4	.9920 ± .0045	.8040 ± .0610
M11	.9918 ± .0049	.8028 ± .0627
M14	.9918 ± .0050	.8022 ± .0619
M21	.9871 ± .0026	.8023 ± .0636
M24	.9862 ± .0019	.8019 ± .0611
BAG1	.8822 ± .0381	.7810 ± .0614
BAG4	.8924 ± .0432	.7927 ± .0632
Cost-sensitive Boosting		
Algorithm	AUC_{Tr}	AUC_{Tst}
C21	.9485 ± .0130	.8246 ± .0549
C24	.9468 ± .0281	.8278 ± .0594
Boosting-based Ensembles		
Algorithm	AUC_{Tr}	AUC_{Tst}
RUS1	.9433 ± .0027	.8555 ± .0526
RUS4	.9434 ± .0030	.8359 ± .0531
SBO1	.9896 ± .0268	.8450 ± .0541
SBO4	.9900 ± .0265	.8424 ± .0554
MBO1	.9279 ± .0244	.8204 ± .0702
MBO4	.9294 ± .0218	.8209 ± .0654
Bagging-based Ensembles		
Algorithm	AUC_{Tr}	AUC_{Tst}
UB1	.9130 ± .0153	.8518 ± .0486
UB4	.9187 ± .0140	.8637 ± .0463
UB21	.9225 ± .0152	.8511 ± .0553
UB24	.9341 ± .0135	.8584 ± .0507
OB1	.9835 ± .0072	.8162 ± .0663
OB4	.9850 ± .0071	.8158 ± .0639
OB21	.9820 ± .0075	.8309 ± .0633
OB24	.9846 ± .0064	.8330 ± .0623
UOB1	.9692 ± .0076	.8422 ± .0536
UOB4	.9904 ± .0049	.8340 ± .0639
SBAG1	.9591 ± .0094	.8509 ± .0601
SBAG4	.9618 ± .0098	.8528 ± .0585
MBAG1	.9161 ± .0169	.8323 ± .0604
MBAG4	.9184 ± .0177	.8365 ± .0638
SPw	.9767 ± .0072	.8302 ± .0560
SPr	.9720 ± .0074	.8336 ± .0633
SPs	.9793 ± .0074	.8194 ± .0649
Hybrid Ensembles		
Algorithm	AUC_{Tr}	AUC_{Tst}
EASY	.9041 ± .0206	.8433 ± .0478
BAL	.9036 ± .0207	.8371 ± .0537

Since we compare pairs of result sets, we use the Wilcoxon signed-rank test to find out whether there are significant differences between the usage of one or another configuration, and if not, to select the set-up which reaches the highest amount of ranks. This does not mean that the method is significantly better, but that it has an overall better behavior among all the data-sets, so we will use it in further comparisons.

Table VII shows the outputs of the Wilcoxon tests. We append a "1" to the algorithm abbreviation to refer that it uses ten classifiers and we do the same with a "4" whenever it uses 40 classifiers. We show the ranks for each method and whether the hypothesis is rejected with a significance value of $\alpha = 0.05$, but also the p-value which give us important information about the differences. The last column shows the configuration that we have selected for the next phase depending on the rejection of the hypothesis or if is not rejected, depending on the ranks.

TABLE VII
WILCOXON TESTS TO DECIDE THE NUMBER OF CLASSIFIERS

Comparison	R^+	R^-	Hypothesis($\alpha = 0.05$)	p-value	Selected
ADAB4 vs. ADAB1	616.5	373.5	Not Rejected	0.17791	ADAB4
M14 vs. M11	558.0	432.0	Not Rejected	0.58135	M14
M24 vs. M21	487.0	503.0	Not Rejected	0.94587	M21
BAG4 vs. BAG1	792.5	197.5	**Rejected for BAG4**	0.00063	BAG4
C24 vs. C21	680.5	309.5	Not Rejected	0.05341	C24
RUS4 vs. RUS1	301.0	689.0	**Rejected for RUS1**	0.01934	RUS1
SBO4 vs. SBO1	493.5	496.5	Not Rejected	0.80592	SBO1
MBO4 vs. MBO1	504.0	486.0	Not Rejected	0.93076	MBO4
UB4 vs. UB1	834.5	155.5	**Rejected for UB4**	0.00009	UB4
UB24 vs. UB21	753.5	236.5	**Rejected for UB24**	0.00404	UB24
OB4 vs. OB1	502.0	488.0	Not Rejected	0.95909	OB4
OB24 vs. OB21	601.0	389.0	Not Rejected	0.26104	OB24
UOB4 vs. UOB1	385.0	605.0	Not Rejected	0.13926	UOB1
SBAG4 vs. SBAG1	625.5	364.5	Not Rejected	0.11482	SBAG4
MBAG4 vs. MBAG1	643.5	346.5	Not Rejected	0.07690	MBAG4

R^+ corresponds to the execution with 40 and R^- to 10 classifiers.

Looking at Table VII, we observe that classic boosting methods have different behaviors; ADAB and M1 have better performance with 40 classifiers, whereas M2 is slightly better with 10. Classic bagging, as well as most of the bagging-based approaches (except UOB), has significantly better results using 40 base classifiers. The cost-sensitive boosting approach obtains a low p-value (close to 0.05) in favor of the configuration of 40 classifiers; hence, it benefits this strategy. With respect to boosting-based ensembles, RUS performance clearly outstands when only ten classifiers are used; on the other hand, the configuration of SBO and MBO is quite indifferent. As in the case of cost-sensitive boosting, for both SBAG and MBAG the p-value is quite low and the sum of ranks stresses the goodness of the selection of 40 classifiers in these ensemble algorithms. Bagging-based approaches that use random oversampling (OB, OB2, and UOB) have not got so big differences, but UOB is the unique that works globally better with the low number of classifiers.

B. Intrafamily Comparison

In this subsection, we develop the comparisons in order to select the best representatives of the families. When we only have a pair of algorithms in a family, we use the Wilcoxon signed-rank test; otherwise, we use the Iman–Davenport test and we follow with Holm post-hoc if it is necessary.

We divided this subsection into five parts, one for the analysis of each family. We have to recall that we do not analyze cost-sensitive Boosting approaches since we are only considering AdaC2 approach; hence, it will be their representative in the last phase. Therefore, first we get on with nonensemble and classic ensemble techniques and then, we go through the remaining three families of ensembles especially designed for imbalanced problems.

1) Nonensemble Techniques: Firstly, we execute the Wilcoxon test between the results of the two non-ensemble

TABLE VIII
WILCOXON TESTS FOR NONENSEMBLE METHODS

Comparison	R^+	R^-	Hypothesis($\alpha = 0.05$)	p-value	Selected
SMT vs. C45	798.5	191.5	**Rejected for SMT**	0.00039	SMT

R^+ are ranks in favor of SMT and R^- in favor of C45.

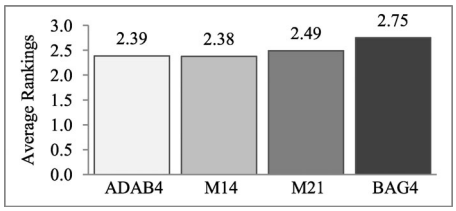


Fig. 4. Average rankings of classic ensembles.

techniques we are considering, C45 and SMT, that is, C4.5 decision tree alone and C4.5 trained over preprocessed data-sets (using SMOTE). The result of the test is shown in Table VIII.

We observe that the performance of C45 is affected by the presence of class imbalance. The Wilcoxon test shows, in concordance with previous studies [20], [53], that making use of SMOTE as a preprocessing technique significantly outperforms C4.5 algorithm alone. The overall performance of SMT is better, achieving higher ranks and rejecting the null hypotheses of equivalence with a *p*-value of 0.00039. For this reason, SMT will be the algorithm representing the family of nonensembles.

2) Classic Ensembles: Regarding classic ensembles, Boosting (AdaBoost, AdaBoost.M1 and AdaBoost.M2) and Bagging, we carry out Iman–Davenport test to find out whether they are statistically different in the imbalance framework. Fig. 4 shows the average rankings of the algorithms computed for the Iman–Davenport test.

We observe that the ranking of BAG4 is higher than the rest, which means that is the worst performer, whereas the rankings of Boosting algorithms are similar, which is understandable because of their common idea. However, the absolute differences of ranks are really low, this is confirmed by the Iman–Davenport test which obtains a *p*-value of 0.49681. Hence, we will select as representative of the family M14 for having the lowest average rank, but notice that in spite of selecting M14, there are not significant differences in this family.

3) Boosting-based Ensembles: This kind of ensembles includes RUSBoost, SMOTEBoost, and MSMOTEBoost approaches. We show the rankings computed to carry out the test in Fig. 5. In this case, Iman–Davenport test rejects the null hypothesis with a *p*-value of $2.97E-04$. Hence, we execute the Holm post-hoc test with RUS1 as control algorithm since it has the lowest ranking.

Holm test shows that RUS1 is significantly better than MBO4, whereas the same significance is not reached with respect to SBO1 (the results are shown in Table IX).

We want to better analyze the relation between RUS1 and SBO1, so we execute Wilcoxon test for this pair. The result is shown in Table X, RUS1 has a better overall behavior as expected, the *p*-value returned by the comparison is low, but despite this situation neither significant differences are attained.

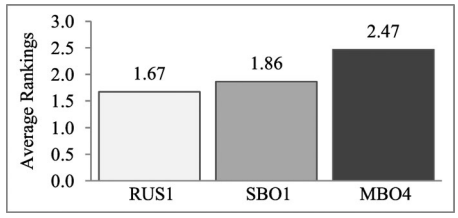


Fig. 5. Average rankings of boosting-based ensembles.

TABLE IX
HOLM TABLE FOR BOOSTING-BASED METHODS

i	Algorithm (Rank)	Z	p-value	Holm	Hypothesis ($\alpha = 0.05$)
2	MBO4 (2.47)	3.73101	0.00019	0.025	**Rejected for RUS1**
1	SBO1 (1.86)	0.90610	0.36488	0.05	Not Rejected

Control method: RUS1, Rank: 1.67.

TABLE X
WILCOXON TESTS TO SHOW DIFFERENCES BETWEEN SBO1 AND RUS1

Comparison	R^+	R^-	Hypothesis($\alpha = 0.05$)	p-value	Selected
SBO1 vs. RUS1	375.0	615.0	Not Rejected	0.1466	RUS1

R^+ are ranks for SBO1 and R^- for RUS1.

TABLE XI
WILCOXON TESTS FOR BAGGING-BASED ENSEMBLES REDUCTION

Comparison	R^+	R^-	Hypothesis($\alpha = 0.05$)	p-value	Selected
UB24 vs. UB4	458.5	531.5	Not Rejected	0.63516	UB4
OB24 vs. OB4	913.0	77.0	**Rejected for OB24**	1.06E-06	OB24
MBAG4 vs. SBAG4	285.0	705.0	**Rejected for SBAG4**	0.01065	SBAG4

RUS1 will represent this family in the next phase due to its better general performance.

4) Bagging-based Ensembles: Because of the number of Bagging-based approaches, we start making a preselection before to the comparison between the family members. On the one hand, we will make a reduction between similar approaches such as UB/UB2, OB/OB2, and SBAG/MBAG. On the other hand, we will select the best IIVotes ensemble comparing the three ways to develop the SPIDER preprocessing inside the IVotes iterations.

To get on with the first part, we use the Wilcoxon test to investigate which one of each pair of approaches is more adequate. The results of these tests are shown in Table XI. Between UnderBagging approaches, UB4 (which always uses all the minority class examples without resampling) obtains higher ranks. This result stresses that the diversity is no more exploited when minority class examples are also bootstrapped, this can be because not using all the minority class instances could make more difficult to learn the positive concept in some of the classifiers of the ensemble. In the case of OverBagging, the use of resampling of the majority class (OB2) clearly outperforms OB, this makes sense since the diversity of OB2 is *a priori* higher than the one of OB. In addition, between synthetic oversampling approaches, the original SMOTEBagging is significantly better than its modification with MSMOTE, which seems not to work as well as the original. Therefore, only UB4, OB24, and SBAG4 are selected for the next phase.

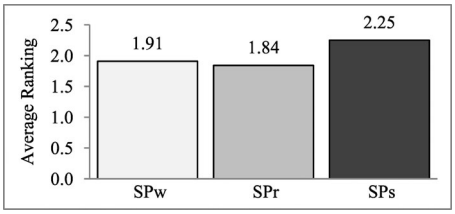


Fig. 6. Average rankings of IIVotes-based ensembles.

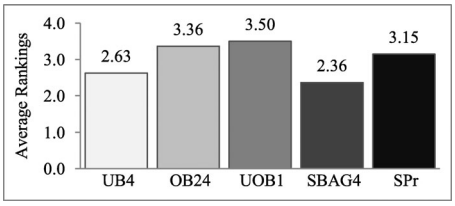


Fig. 7. Average rankings of bagging-based ensembles.

TABLE XII
HOLM TABLE FOR BEST BAGGING-BASED METHODS

i	Algorithm (Rank)	Z	p-value	Holm	Hypothesis ($\alpha = 0.05$)
4	UOB1 (3.5)	3.37100	0.00075	0.0125	**Rejected for SBAG4**
3	OB24 (3.36)	2.96648	0.00301	0.01667	**Rejected for SBAG4**
2	SPr (3.15)	2.32599	0.02002	0.025	**Rejected for SBAG4**
1	UB4 (2.63)	0.77533	0.43814	0.05	Not Rejected

Control method: SBAG4, Rank:2.36.

Regarding IIVotes methods, we start the multiple comparisons by executing the Iman–Davenport test which returns a p-value of 0.1208. Therefore, the hypothesis of equivalence is not rejected. However, as Fig. 6 shows, the rankings obtained by SPr are higher than the ones of the other two methods. Following these results, we will only take into account SPr in the following phase.

Once we have reduced the number of Bagging-based algorithms, we can develop the proper comparison among the remaining methods. The Iman–Davenport test executed for this group of algorithms returns a p-value of 0.00172, which means that there exist significant differences (in Fig. 7, we show the average rankings).

Hence, we apply the Holm post-hoc procedure to compare SBAG4 (the one with the best ranking) with the rest of the Bagging-based methods. Observing the results shown in Table XII, SBAG4 clearly outperforms the other methods (except for UB4) with significant differences.

Regarding UB4, and given its similar behavior to SBAG4 with respect to the rest, we will carry out a Wilcoxon test (Table XIII) in order to check whether there are any significant differences between them. From this test we conclude that, when both algorithms are confronted one versus the other, they are equivalent. On the contrary to the rankings computed among the group of algorithms, the ranks in this case are nearly the same. This occurs because SBAG4 has a good overall behavior among more data-sets, whereas UB4 stands out more in some of them and less in others. As a consequence, when they are put together with other methods, UB4 ranking decreases, whereas SBAG4 excels in spite of UB4 mean test result is slightly higher than

TABLE XIII
WILCOXON TESTS TO SHOW DIFFERENCES BETWEEN SBAG4 AND UB4

Comparison	R^+	R^-	Hypothesis($\alpha = 0.05$)	p-value
SBAG4 vs. UB4	492.5	497.5	Not Rejected	0.94224

R^+ are ranks for SBAG4 and R^- for UB4.

TABLE XIV
WILCOXON TESTS FOR NONENSEMBLE METHODS

Comparison	R^+	R^-	Hypothesis($\alpha = 0.05$)	p-value	Selected
BAL vs. EASY	418.5	571.5	Not Rejected	0.3356	EASY

R^+ are ranks for BAL and R^- for EASY.

TABLE XV
REPRESENTATIVE METHODS SELECTED FOR EACH FAMILY

Family	**Abbr.**	**Method**
Non-ensembles	SMT	SMOTE
Classic	M14	AdaBoost.M2 ($T = 40$)
Cost-sensitive	C24	AdaC2 ($T = 40$)
Boosting-based	RUS1	RUSBoost ($T = 10$)
Bagging-based	SBAG4	SMOTEBagging ($T = 40$)
Hybrids	EASY	EasyEnsemble

SBAG4. Knowing that both algorithms achieve similar performances, we will use as representative SBAG4 because its overall behavior when the comparison has included more methods has been better.

5) Hybrid Ensembles: This last family only has two methods; hence, we execute Wilcoxon signed-rank test to find out possible differences. Table XIV shows the result of the test, both methods are quite similar, but EASY attains higher ranks. This result is in accordance with previous studies [47], where the advantage of BAL is its efficiency when dealing with large data-sets without highly decreasing the performance with respect to EASY. Following the same methodology as in previous families, we will use EASY as representative.

C. Interfamily Comparison

We have selected a representative for every family, so now we can proceed with the global study of the performance. First, we recall the selected methods from the intrafamily comparison in Table XV.

We have summarized the results for the test partitions of these methods in Fig. 8 using the box plot as representation scheme. Box plots proved a most valuable tool in data reporting, since they allow the graphical representation of the performance of the algorithms, indicating important features such as the median, extreme values and spread of values about the median in the form of quartiles. We can observe that the RUS1 box is compact, as well as the SBAG4 box, both methods have similar results (superior to the rest), but the RUS1 median value is better. On the other hand, SMT seemsto be inferior to the other approaches with the exception of M14, which variance is the highest.

Starting with the comparison itself, we use the Iman–Davenport test to find out significant differences among these methods. The rankings computed to carry out the test are depicted in Fig. 9. The p-value returned by the test is very low ($1.27E-09$); hence, there exist differences among some of

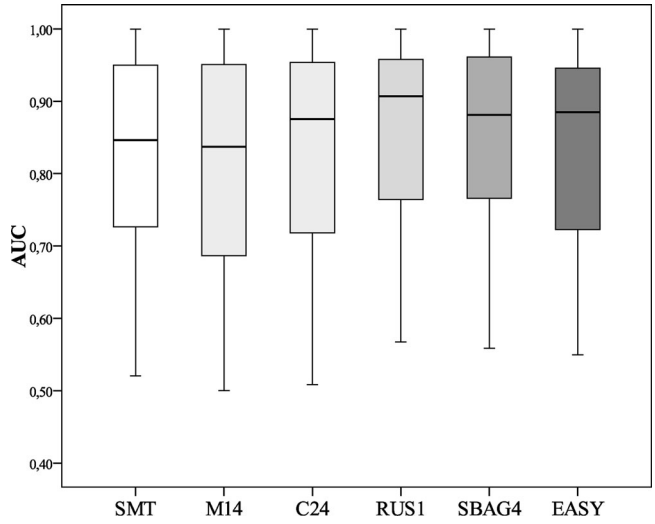


Fig. 8. Box-plot of AUC results of the families' representatives.

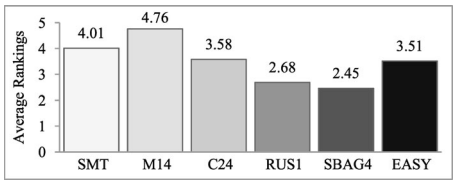


Fig. 9. Average rankings of the representatives of each family.

TABLE XVI
HOLM TABLE FOR BEST INTERFAMILY ANALYSIS

i	Algorithm (Rank)	Z	p-value	Holm	Hypothesis ($\alpha = 0.05$)
5	M14 (4.76)	5.78350	0.00000	0.01	**Rejected for SBAG4**
4	SMT (4.01)	3.90315	0.00009	0.0125	**Rejected for SBAG4**
3	C24 (3.58)	2.82052	0.00479	0.01667	**Rejected for SBAG4**
2	EASY (3.51)	2.64958	0.00806	0.025	**Rejected for SBAG4**
1	RUS1 (2.68)	0.56980	0.56881	0.05	Not Rejected

Control method : SBAG4, Rank :2.45.

TABLE XVII
WILCOXON TESTS TO SHOW DIFFERENCES BETWEEN SBAG4 AND RUS1

Comparison	R^+	R^-	Hypothesis($\alpha = 0.05$)	p-value
SBAG4 vs. RUS1	527.5	462.5	Not Rejected	0.71717

R^+ are ranks for SBAG4 and R^- for RUS1.

these algorithms and we continue with the Holm post-hoc test. The results of this test are shown in Table XVI.

The Holm test brings out the dominance of SBAG4 and RUS1 over the rest of the methods. SBAG4 significantly outperforms all algorithms except RUS1. We have two methods which behave similarly with respect to the rest, SBAG4 and RUS1; therefore, we will get them into a pairwise comparison via a Wilcoxon test. In such a way, our aim is to obtain a better insight on the behavior of this pair of methods (in Table XVII, we show the result). The Wilcoxon test neither indicates the existence of statistical differences; moreover, both algorithms are similar in terms of ranks, SBAG4 has an advantage and hence, apparently a better overall behavior, but we cannot support this fact with this test. Therefore, SBAG4 is the winner of the hierarchical analysis in terms of ranks, but it is closely followed by RUS1 and UB4

TABLE XVIII
SHAFFER TESTS FOR INTERFAMILY COMPARISON

	SMT	M14	C24	RUS1	SBAG4	EASY
SMT	×	=(0.24024)	=(1.0)	-(0.00858)	-(0.00095)	=(1.0)
M14	=(0.24024)	×	-(0.03047)	-(0.0)	-(0.0)	-(0.01725)
C24	=(1.0)	+(0.03047)	×	=(0.17082)	-(0.03356)	=(1.0)
RUS1	+(0.00858)	+(0.0)	=(0.17082)	×	=(1.0)	=(0.22527)
SBAG4	+(0.00095)	+(0.0)	+(0.03356)	=(1.0)	×	=(0.05641)
EASY	+(0.01725)	=(1.0)	=(1.0)	=(0.22527)	=(0.05641)	×

(as we have shown in Section V-B4). However, despite SBAG4 wins in terms of ranks, since there does not exist any statistical difference, we may also pay attention to the computational complexity of each algorithm in order to establish a preference. In this sense, RUS1 undoubtedly stands out with respect to both SBAG4 and UB4. RUS1 and UB4 classifiers' building time is lower than that of SBAG4's classifiers; this is due to the undersampling process they develop instead of the oversampling that is carried out by SBAG4, in such a way, the classifiers are trained with much less instances. Moreover, RUS1 only uses ten classifiers against the 40 classifiers that are used by SBAG4 and UB4, which apart from resulting in a less complex and more comprehensible ensemble, needs four times less time than UB4 to be constructed.

To end and complete the statistical study, we carry out another post-hoc test for the interfamily comparison in order to show the relation between all representatives, that is, a $n \times n$ comparison. To do so, we execute the Shaffer post-hoc test and we show the results in Table XVIII. In this table, a "+" symbol implies that the algorithm in the row is statistically better than the one in the column, whereas "−" implies the contrary; "=" means that the two algorithms that are compared have no significant differences. In brackets, the adjusted *p*-value that is associated with each comparison is shown. In this table, we can also observe the superiority of SBAG4 and RUS1 against the remaining algorithms and besides, the similarity (almost equivalence) between both approaches.

D. Discussion: Summary of the Results

In order to summarize the whole hierarchical analysis developed in this section, we include a scheme showing the global analysis in Fig. 10. Each algorithm is represented by a gray tone (color). For Wilcoxon tests, we show the ranks and the *p*-value returned; for Iman–Davenport tests, we show the rankings and whether the hypothesis has been rejected or not by the usage of the Holm post-hoc test. This way, the evolution of the analysis can be easily followed.

Summarizing the results of the hierarchical analysis, we point out the main conclusions that we have extracted from the experimental study afterwards:

1) The methods with the best (the most robust) behavior are SMOTEBagging, RUSBoost, and UnderBagging. Among them, in terms of ranks, SMOTEBagging stands out obtaining slightly better results. Anyway, this triple of algorithms outperforms statistically the others considered in this study, but they are statistically equivalent; for this reason, we should take the computational complexity into

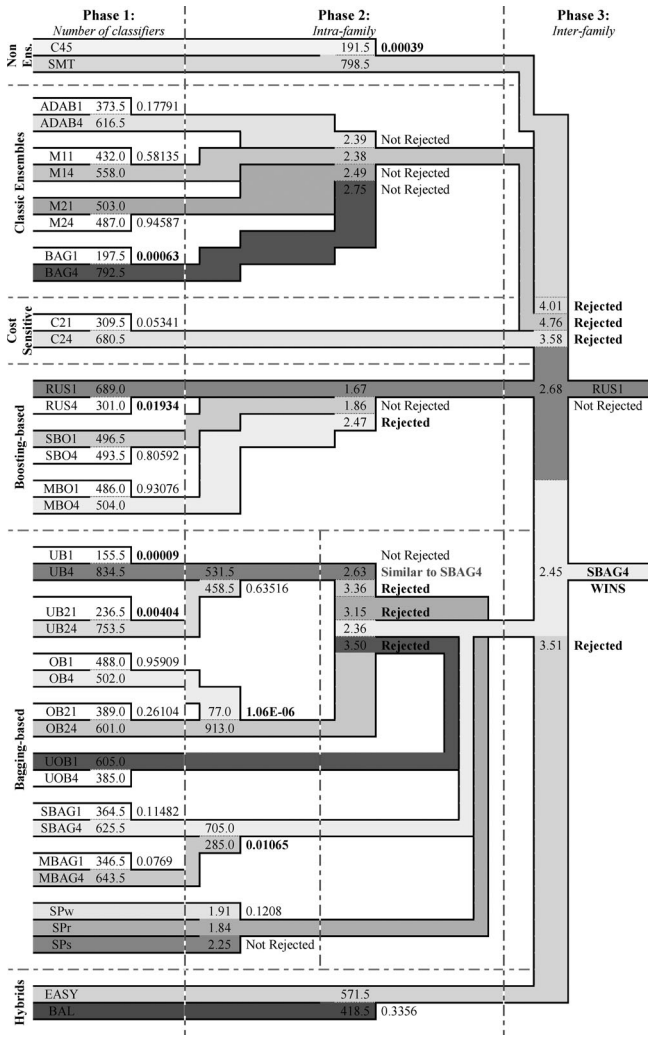


Fig. 10. Global analysis scheme. A gray tone (color) represents each algorithm. Rankings and *p*-values are shown for Wilcoxon tests whereas only rankings and the hypothesis results are shown for Iman–Davenport and Holm tests.

account, in such a manner, RUSBoost excels as the most appropriate ensemble method (Item 5 extends this issue).

2) More complex methods does not perform better than simpler ones. It must be pointed out that the performance of two of the simplest approaches (RUSBoost and UnderBagging), with the usage of a random and easy-to-develop strategy, achieve better results than many other approaches. The positive synergy between random undersampling and ensemble techniques has stood out looking at the experimental analysis. This sampling technique eliminates different majority class examples in each iteration; this way, the distribution of the class overlapping differs in all data-sets, and this causes the diversity to be boosted. In addition, in contrast with the mere use of an undersampling process before learning a nonensemble classifier, carrying it out in every iteration when constructing the ensemble allows the consideration of most of the important majority patterns that can be defined by concrete instances which using a unique classifier could be lost.

3) Bagging techniques are not only easy to develop, but also powerful when dealing with class imbalance if they are properly combined. Their hybridization with data preprocessing techniques has shown competitive results, the key

TABLE XIX
DETAILED TEST RESULTS TABLE OF NONENSEMBLE METHODS AND CLASSIC ENSEMBLE ALGORITHMS

		Non-ens.		Classic							
Data-set	IR	C45	SMT	ADAB1	ADAB4	M11	M14	M21	M24	BAG1	BAG4
Glass1	1.82	.7399	.7368	.7875	.7925	.7875	.8059	.7625	.7630	.7361	.7389
Ecoli0vs1	1.86	**.9832**	.9729	.9692	.9692	.9726	.9726	.9692	.9692	**.9832**	**.9832**
Wisconsin	1.86	.9454	.9532	.9666	.9645	.9656	.9656	.9529	.9666	.9668	.9613
Pima	1.90	.7012	.7245	.6886	.7108	.6863	.6968	.7263	.7189	.7155	.7205
Iris0	2.00	.9900	**.9900**	.9900	.9900	.9900	.9900	.9900	.9900	.9800	.9800
Glass0	2.06	.8167	.7752	.8267	.8408	.8090	.8267	.8050	.8262	.7802	.8159
Yeast1	2.46	.6642	.7090	.6462	.6642	.6468	.6681	.6663	.6743	.6930	.6989
Vehicle1	2.52	.6717	.7301	.6718	.7005	.7058	.6896	.6810	.6700	.6583	.6643
Vehicle2	2.52	.9561	.9498	.9708	.9723	.9708	.9754	.9761	.9801	.9547	.9654
Vehicle3	2.52	.6637	.7282	.6906	.7089	.6852	.6979	.6822	.6820	.6760	.6766
Haberman	2.68	.5757	.6163	.5815	.5815	.5815	.5815	.6086	.6022	.5662	.5514
Glass0123vs456	3.19	.9155	.9232	.8625	.8634	.8625	.8634	.9025	.8903	.8956	.9156
Vehicle0	3.23	.9296	.9188	.9511	.9703	.9503	.9703	.9556	.9589	.9519	.9594
Ecoli1	3.36	.8586	.9105	.8222	.8567	.8403	.8524	.8433	.8504	.8480	.8644
New-thyroid2	4.92	.9373	.9659	.9373	.9373	.9516	.9516	.9659	.9659	.9488	.9516
New-thyroid1	5.14	.9143	.9631	.9687	.9544	.9687	.9544	.9659	.9829	.9516	.9401
Ecoli2	5.46	.8641	.8811	.8632	.8523	.8632	.8523	.8841	.8823	.8884	.8705
Segment0	6.01	.9826	.9927	.9916	.9916	.9916	.9916	.9949	.9932	.9848	.9838
Glass6	6.38	.8132	.8842	.8725	.8752	.8725	.8752	.8698	.8659	.8632	.8838
Yeast3	8.11	.8597	.8905	.8249	.8224	.8023	.8226	.8437	.8315	.8504	.8589
Ecoli3	8.19	.7280	.8123	.7641	.7960	.7767	.7960	.7717	.8262	.7498	.7658
Page-blocks0	8.77	.9221	.9504	.9294	.9291	.9292	.9281	.9302	.9286	.9284	.9251
Yeast2vs4	9.08	.8307	.8588	.8518	.8128	.8518	.8228	.8853	.8516	.8241	.8330
Yeast05679vs4	9.35	.6802	.7602	.7067	.6838	.6967	.6838	.6795	.6927	.6868	.6868
Vowel0	10.10	.9706	.9505	.9706	.9706	.9706	.9706	.9594	.9822	.9433	.9594
Glass016vs2	10.29	.5938	.6062	.5355	.5355	.5355	.5355	.5660	.5688	.5526	.5305
Glass2	10.39	.7194	.6390	.6137	.5354	.6137	.5354	.5271	.5521	.5296	.5423
Ecoli4	13.84	.8437	.7794	.8421	.8671	.8421	.8671	.8655	.8171	.8452	.8437
Shuttle0vs4	13.87	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9994	.9997
Yeast1vs7	13.87	.6275	.7003	.6252	.6097	.6252	.6097	.5717	.5942	.5833	.5965
Glass4	15.47	.7542	.8867	.7875	.7875	.7875	.7875	.8733	.8450	.8208	.8233
Page-blocks13vs2	15.85	**.9978**	.9955	**.9978**	**.9978**	.9644	.9644	.9778	.9588	**.9978**	**.9978**
Abalone9vs18	16.68	.5983	.6283	.6336	.6276	.6336	.6283	.6093	.6262	.6040	.6055
Glass016vs5	19.44	.8943	.8129	.8943	.8943	.8943	.8943	.8943	.8943	.7914	.8914
Shuttle2vs4	20.50	.9500	.9917	.9500	.9500	.9500	.9500	**1**	**1**	.8500	.9500
Yeast1458vs7	22.10	.5000	.5367	.5288	.5477	.5114	.5318	.5462	.5144	.4992	.5000
Glass5	22.81	.8976	.8805	.9476	.9476	.9476	.9476	.8976	.8976	.8427	.8451
Yeast2vs8	23.10	.5250	**.8338**	.6989	.7489	.7239	.7489	.7478	.7239	.5250	.5250
Yeast4	28.41	.6135	.7121	.5798	.6043	.6205	.5940	.5918	.6032	.5949	.6247
Yeast1289vs7	30.56	.6156	.6832	.5956	.5962	.5962	.5790	.6112	.6295	.5312	.5489
Yeast5	32.78	.8833	.9337	.8483	.8712	.8483	.8712	.8493	.8597	.8622	.8951
Ecoli0137vs26	39.15	.7481	.8136	**.8481**	**.8481**	**.8481**	**.8481**	.6482	.6445	.6981	.7481
Yeast6	39.15	.7115	.8294	.7380	.6962	.7527	.6972	.7380	.7098	.7126	.7558
Abalone19	128.87	.5000	.5205	.4987	.4995	.4990	.4999	.5159	.4999	.5000	.5000
Mean		.7929	.8257	.8016	.8040	.8028	.8022	.8023	.8019	.7810	.7927

TABLE XX
DETAILED TEST RESULTS TABLE FOR COST-SENSITIVE BOOSTING, BOOSTING-BASED, AND HYBRID ENSEMBLES

		Cost-sen.		Boosting-based						Hybrids	
Data-set	IR	C21	C24	RUS1	RUS4	SBO1	SBO4	MBO1	MBO4	EASY	BAL
Glass1	1.82	.7866	.7867	.7632	.8116	.8206	.7932	.7259	.7400	.7222	.6971
Ecoli0vs1	1.86	.9692	.9692	.9691	.9726	.9691	.9692	.9692	.9692	.9796	.9762
Wisconsin	1.86	.9653	.9725	.9643	.9653	.9590	.9653	.9608	.9608	.9542	.9603
Pima	1.90	.7096	.7128	.7263	.7251	.7126	.7223	.7385	.7410	.7230	.6722
Iris0	2.00	.9900	.9900	.9900	.9900	.9900	.9900	.9900	.9900	.9900	.9900
Glass0	2.06	.8101	.8177	.8129	.8377	.8413	**.8622**	.7819	.7821	.7830	.7834
Yeast1	2.46	.6604	.6752	.7188	.6998	.7008	.6983	.7130	.7089	.7147	.6830
Vehicle1	2.52	.7531	**.8007**	.7469	.7207	.7486	.7501	.7313	.7306	.7406	.7580
Vehicle2	2.52	.9729	.9814	.9698	.9784	**.9851**	**.9851**	.9637	.9607	.9656	.9577
Vehicle3	2.52	.7345	.7699	.7653	.7655	.7458	.7450	.7699	.7723	.7591	.7611
Haberman	2.68	.5604	.5604	.6549	.6273	.6347	.6433	.6313	.6291	.6521	.5943
Glass0123vs456	3.19	.9033	.9233	.9302	.9264	.9223	.9223	.9163	.9163	.9025	.8650
Vehicle0	3.23	.9438	**.9768**	.9583	.9605	.9649	.9730	.9402	.9450	.9375	.9502
Ecoli1	3.36	.8763	.8928	.8829	.8894	.8604	.8571	.9017	.8922	.8844	.9021
New-thyroid2	4.92	.9575	.9575	.9377	.9774	.9778	.9833	.9460	.9433	.9187	.9266
New-thyroid1	5.14	.9464	.9464	.9575	.9687	.9889	.9889	.9659	.9659	.9353	.9381
Ecoli2	5.46	.8845	.8835	.8989	.9082	.9017	**.9088**	.8843	.8843	.8858	.8914
Segment0	6.01	.9826	.9826	.9927	.9921	**.9962**	.9962	.9899	.9899	.9836	.9836
Glass6	6.38	.8923	.8923	.9176	.9230	.8477	.8505	**.9365**	**.9365**	.8851	.8932
Yeast3	8.11	.9108	.8931	.9247	.9238	.8966	.8947	.8977	.8977	.9369	.9312
Ecoli3	8.19	.8478	.8224	.8563	.8794	.8634	.8558	.8449	.8449	.8917	.8910
Page-blocks0	8.77	.8816	.8154	.9484	.9450	.9376	.9407	.9394	.9394	.9502	.9464
Yeast2vs4	9.08	.9172	.9205	.9333	.9241	.8961	.8993	.8813	.8813	**.9411**	.9292
Yeast05679vs4	9.35	.7610	.7816	.8033	.7789	.7763	.7784	.7700	.7620	.7460	.7812
Vowel0	10.10	.9706	.9706	.9427	.9672	.9894	**.9917**	.9528	.9528	.9410	.9388
Glass016vs2	10.29	.5400	.5536	.6167	.5795	.5931	.6321	.5479	.5479	.6129	.6193
Glass2	10.39	.7099	.7308	**.7797**	.7217	.7740	.7433	.6566	.6842	.7132	.7154
Ecoli4	13.84	.9280	.9280	**.9418**	.8874	.9092	.8889	.8687	.8671	.8770	.7787
Shuttle0vs4	13.87	.9997	.9997	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Yeast1vs7	13.87	.7049	.7049	.7149	.6937	.6425	.6375	.6255	.6290	.7212	.6791
Glass4	15.47	.8706	.8706	.9151	.7967	.9226	.9226	.7083	.7083	.8505	.8381
Page-blocks13vs2	15.85	**.9978**	**.9978**	.9865	.9800	.9944	.9944	.9921	.9921	.9831	.9661
Abalone9vs18	16.68	.6954	.6954	.6933	.6793	**.7661**	.7136	.6840	.6855	.6999	.7161
Glass016vs5	19.44	.8800	.8800	**.9886**	.9386	.8829	.9386	.8829	.8829	.9429	.9429
Shuttle2vs4	20.50	.9500	.9500	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.9875	.9875
Yeast1458vs7	22.10	.5426	.5426	.5674	.4896	.5948	.5394	.5145	.5152	.5496	.5987
Glass5	22.81	.9732	.9732	.9427	.9402	.9805	.9805	.8902	.8854	.9488	.9488
Yeast2vs8	23.10	.6218	.6218	.7893	.7392	.7403	.7446	.7717	.7728	.7168	.7482
Yeast4	28.41	.7204	.7204	.8124	.7689	.7154	.6696	.7378	.7378	.8477	.8311
Yeast1289vs7	30.56	.6376	.6376	.7209	.6394	.6509	.6386	.5369	.5369	.7019	.6550
Yeast5	32.78	.8833	.8833	.9587	.9594	.9229	.9010	.9215	.9208	.9531	.9618
Ecoli0137vs26	39.15	.8154	.8154	.7935	.5463	.8372	.8391	.7445	.7445	.7318	.7427
Yeast6	39.15	.7163	.7163	.8233	.8244	.7872	.7893	.7757	.7757	.8492	.8468
Abalone19	128.87	.5085	.5085	.6306	.5372	.5387	.5272	.4981	.4981	.6956	.6565
Mean		.8246	.8278	.8555	.8359	.8450	.8424	.8204	.8209	.8433	.8371

TABLE XX1
DETAILED TEST RESULTS TABLE FOR BAGGING-BASED ALGORITHMS

		Bagging-based																
Data-set	IR	UB1	UB4	UB21	UB24	OB1	OB4	OB21	OB24	UOB1	UOB4	SBAG1	SBAG4	MBAG1	MBAG4	SPw	SPr	SPs
Glass1	1.82	.7483	.7372	.6910	.7519	.7638	.7576	.7789	.7796	.7520	.7742	.7840	.7279	.7551	.7321	.7596	.7662	.7764
Ecoli0vs1	1.86	.9693	.9796	.9693	.9796	.9798	.9798	.9796	.9796	.9796	.9796	.9761	**.9832**	**.9832**	**.9832**	.9726	.9587	.9656
Wisconsin	1.86	.9565	.9597	.9685	.9706	.9634	.9643	**.9780**	.9727	.9641	.9612	.9622	.9600	.9603	.9642	.9664	.9729	.9696
Pima	1.90	.7581	.7600	.7463	.7529	.7196	.7148	.7236	.7385	.7354	.7359	.7385	.7509	.7372	.7490	**.7632**	.7539	.7381
Iris0	2.00	.9900	.9900	.9800	.9800	.9800	.9800	.9800	.9800	.9800	.9800	.9800	.9800	.9800	.9800	.9900	.9900	.9900
Glass0	2.06	.8177	.8143	.8206	.8244	.8126	.8268	.8446	.8342	.7920	.8377	.8318	.8387	.7935	.8006	.8243	.8286	.8139
Yeast1	2.46	.7156	.7218	.7169	.7207	.7116	.7173	.7105	.7344	.7229	.7234	.7113	.7336	.7033	.7392	.7254	.7275	.7104
Vehicle1	2.52	.7648	.7873	.7434	.7605	.7144	.7162	.7509	.7567	.7354	.7244	.7505	.7692	.7307	.7612	.7229	.7173	.7113
Vehicle2	2.52	.9572	.9635	.9674	.9642	.9551	.9566	.9645	.9676	.9643	.9653	.9702	.9664	.9623	.9639	.9768	.9721	.9746
Vehicle3	2.52	.7638	**.8023**	.7694	.7843	.7027	.7059	.7691	.7385	.7417	.7222	.7380	.7629	.7337	.7866	.7480	.7632	.7535
Haberman	2.68	.6578	.6644	.6342	.6680	.6055	.6144	.6221	.6416	.6297	.5877	.6590	.6560	.6375	.6611	.6100	.6414	.6283
Glass0123vs456	3.19	.8939	.9039	.9431	.9169	.9316	.9347	.9172	.9264	.9010	.9116	.8992	**.9455**	.9062	.9099	.9223	.9123	.8923
Vehicle0	3.23	.9453	.9523	.9511	.9542	.9357	.9448	.9562	.9665	.9490	.9523	.9464	.9650	.9397	.9492	.9577	.9689	.9519
Ecoli1	3.36	.8978	.8996	.9107	.9021	.8662	.8646	.8746	.8875	**.9190**	.8862	.9002	.9002	.9170	.9112	.9012	.8706	.8722
New-thyroid2	4.92	.9468	.9579	.9325	.9381	.9317	.9317	.9317	.9317	.9290	.9544	.9607	.9607	.9631	.9516	.9857	.9687	.9687
New-thyroid1	5.14	.9552	.9635	.9635	.9690	.9544	.9401	.9774	.9631	.9575	.9774	.9774	.9746	.9603	.9774	.9488	.9687	.9687
Ecoli2	5.46	.8704	.8844	.8718	.8806	.8734	.8925	.8984	.8972	.8794	.8799	.8802	.8875	.8736	.8761	.9008	.8817	.8817
Segment0	6.01	.9851	.9876	.9846	.9858	.9919	.9919	.9932	.9934	.9927	.9914	.9939	.9944	.9914	.9911	.9934	.9949	.9954
Glass6	6.38	.8851	.9041	.9203	.9257	.8838	.8865	.9311	.9144	.9284	.9171	.9338	.9311	.9063	.9284	.9225	**.9365**	.8998
Yeast3	8.11	.9403	.9338	.9388	**.9444**	.8983	.9057	.8995	.9112	.9225	.9173	.9286	.9436	.9352	.9385	.8976	.8972	.8774
Ecoli3	8.19	.8824	**.9077**	.8857	.8941	.7448	.7399	.7904	.8096	.8123	.8113	.8694	.8848	.8535	.8694	.8491	.8618	.8474
Page-blocks0	8.77	.9516	.9580	.9569	**.9585**	.9370	.9379	.9406	.9443	.9500	.9528	.9509	.9531	.9484	.9536	.9467	.9491	.9504
Yeast2vs4	9.08	.9403	.9360	.9304	.9293	.8613	.8513	.8972	.9004	.9275	.8915	.8846	.8968	.8827	.9074	.8656	.8633	.8767
Yeast05679vs4	9.35	.7823	.7943	.7933	.8137	.7125	.7062	.7731	.7405	.7907	.7373	.8130	**.8182**	.8081	.7912	.7478	.7648	.7276
Vowel0	10.10	.9438	.9466	.9427	.9466	.9661	.9672	.9661	.9683	.9628	.9672	.9828	.9883	.9428	.9494	.9656	.9650	.9594
Glass016vs2	10.29	.6364	**.7538**	.6414	.6248	.6157	.6129	.6157	.6100	.6100	.6157	.6364	.5588	.6400	.6067	.5631	.6631	.5576
Glass2	10.39	.7584	.7685	.6519	.7059	.6918	.6585	.6843	.7118	.7074	.6969	.7788	.7787	.7023	.6947	.6029	.6036	.5502
Ecoli4	13.84	.8914	.8883	.8792	.8994	.8639	.8639	.8889	.8639	.9139	.8389	.9060	.9326	.8592	.8842	.8937	.8889	.8937
Shuttle0vs4	13.87	**1.000**	**1.000**	**1.000**	**1.000**	.9997	.9997	.9994	.9997	.9994	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997
Yeast1vs7	13.87	.7468	**.7857**	.7377	.7731	.6468	.6325	.6456	.6825	.6340	.6872	.6801	.6967	.6829	.7007	.6325	.5732	.6492
Glass4	15.47	.8530	.8456	.8702	.8705	.8967	.8942	**.9375**	.9275	.9200	.9275	.8767	.8742	.7750	.7750	.8300	.8108	.8183
Page-blocks13vs2	15.85	.9752	.9775	.9515	.9752	**.9978**	**.9978**	**.9978**	**.9978**	**.9978**	**.9978**	.9888	.9876	.9910	.9933	**.9978**	.9933	**.9978**
Abalone9vs18	16.68	.7101	.7189	.7182	.7100	.6095	.6124	.6310	.6561	.6828	.6553	.7532	.7451	.7158	.7055	.6669	.6655	.6500
Glass016vs5	19.44	.9429	.9429	.9429	.9429	.7886	.7857	.7857	.7857	.9329	.8914	.8771	.8657	.8443	.8443	.9414	.9414	.9443
Shuttle2vs4	20.50	.9875	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Yeast1458vs7	22.10	.5630	.6062	.6067	.6172	.5418	.5175	.5819	.5372	.5509	.5304	**.6268**	.6230	.4963	.4963	.5553	.5591	.5599
Glass5	22.81	.9488	.9488	.9488	.9488	.8927	.8927	.8878	.8878	**.9854**	.9402	.8756	.8780	.8878	.8878	.9476	.8976	.8976
Yeast2vs8	23.10	.7613	.7830	.7393	.7469	.7946	.7935	.7903	.8185	.8098	.8207	.7881	.7837	.7739	.7739	.6957	.7457	.6457
Yeast4	28.41	**.8601**	.8552	.8337	.8542	.6730	.6923	.7033	.6953	.7439	.7011	.7925	.7733	.7373	.7257	.6637	.7037	.6258
Yeast1289vs7	30.56	.6746	**.7335**	.7046	.6887	.5858	.6085	.5858	.6197	.6121	.6175	.6408	.6575	.5569	.5563	.6079	.6202	.6096
Yeast5	32.78	.9642	.9521	.9639	.9559	.8677	.8556	.9118	.9014	.9441	.9333	.9622	.9618	**.9656**	.9653	.9139	.9365	.9125
Ecoli0137vs26	39.15	.7263	.7445	.8063	.7809	.7409	.7409	.7409	.7409	.8318	.7355	.8281	.8281	.8463	.8445	.8445	.8445	.7463
Yeast6	39.15	.8641	.8685	.8577	**.8779**	.7865	.7869	.8032	.8039	.8259	.8155	.8354	.8358	.8151	.8155	.7934	.8060	.7802
Abalone19	128.87	.6950	**.7206**	.6619	.6797	.5203	.5195	.5208	.5358	.5353	.5513	.5710	.5716	.5271	.5125	.5149	.5290	.5155
Mean		.8518	**.8637**	.8511	.8584	.8162	.8158	.8309	.8330	.8422	.8340	.8509	.8528	.8323	.8365	.8302	.8336	.8194

issue of these methods resides in properly exploiting the diversity when each bootstrap replica is formed.

4) Clearly, the trade-off between complexity and performance of ensemble learning algorithms adapted to handle class imbalance is positive, since the results are significantly improved. They are more appropriate than the mere use of classic ensembles or data preprocessing techniques. In addition, extending the results of the last part of the experimental study, base classifier's results are outperformed.
5) Regarding the computational complexity, even though our analysis is mainly devoted to algorithms' performance, we should highlight that RUSBoost is competing against SMOTEBagging and UnderBagging with only ten classifiers (since it achieves better performance with less classifiers). The reader might also note that, RUSBoost's classifiers are much faster in building time, since less instances are used to construct each classifier (due to the undersampling process); besides, the ensemble is more comprehensible, containing only ten smaller trees. On the other hand, SMOTEBagging constructs larger trees (due to the oversampling mechanism). Likewise, UnderBagging is computationally harder than RUSBoost, in spite of obtaining comparable size trees, it uses four times more classifiers.

VI. CONCLUDING REMARKS

In this paper, the state of the art on ensemble methodologies to deal with class imbalance problem has been reviewed. This issue hinders the performance of standard classifier learning algorithms that assume relatively balanced class distributions, and classic ensemble learning algorithms are not an exception. In recent years, several methodologies integrating solutions to enhance the induced classifiers in the presence of class imbalance by the usage of ensemble learning algorithms have been presented. However, there was a lack of framework where each one of them could be classified; for this reason, a taxonomy where they can be placed has been presented. We divided these methods into four families depending on their base ensemble learning algorithm and the way in which they address the class imbalance problem.

Once that the new taxonomy has been presented, thorough study of the performance of these methods in a large number of real-world imbalanced problems has been performed, and these approaches with classic ensemble approaches and nonensemble approaches have been compared. We have performed this study developing a hierarchical analysis over the taxonomy proposed, which was guided by nonparametric statistical tests.

Finally, we have concluded that ensemble-based algorithms are worthwhile, improving the results that are obtained by the usage of data preprocessing techniques and training a single classifier. The use of more classifiers makes them more complex, but this growth is justified by the better results that can be assessed. We have to remark the good performance of approaches such as RUSBoost or UnderBagging, which despite being simple approaches, achieve higher performances than many other more complex algorithms. Moreover, we have shown the positive synergy between sampling techniques (e.g., undersampling or SMOTE) and Bagging ensemble learning algorithm. Particularly noteworthy is the performance of RUSBoost, which is the computationally least complex among the best performers.

APPENDIX

DETAILED RESULTS TABLE

In this appendix, we present the AUC test results for all the algorithms in all data-sets. Table XIX shows the results for nonensembles and classic ensembles. In Table XX we show the test results for cost-sensitive boosting, boosting-based and hybrid ensembles, whereas Table XXI shows the test results for bagging-based ones. The results are shown in ascending order of the IR. The last row in each table shows the average result of each algorithm. We stress with bold-face the best results among all algorithms in each data-set.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments and suggestions that contributed to the improvement of this work.

REFERENCES

[1] Y. Sun, A. C. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *Int. J. Pattern Recogn.*, vol. 23, no. 4, pp. 687–719, 2009.
[2] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[3] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data Mining and Knowledge Discovery Handbook*, 2010, pp. 875–886.
[4] V. García, R. Mollineda, and J. Sánchez, "On the k-nn performance in a challenging scenario of imbalance and overlapping," *Pattern Anal. App.*, vol. 11, pp. 269–280, 2008.
[5] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *J. Artif. Intell. Res.*, vol. 19, pp. 315–354, 2003.
[6] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, pp. 429–449, 2002.
[7] D. A. Cieslak and N. V. Chawla, "Start globally, optimize locally, predict globally: Improving performance on imbalanced data," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2009, pp. 143–152.
[8] Q. Yang and X. Wu, "10 challenging problems in data mining research," *Int. J. Inf. Tech. Decis.*, vol. 5, no. 4, pp. 597–604, 2006.
[9] Z. Yang, W. Tang, A. Shintemirov, and Q. Wu, "Association rule mining-based dissolved gas analysis for fault diagnosis of power transformers," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 6, pp. 597–610, 2009.
[10] Z.-B. Zhu and Z.-H. Song, "Fault diagnosis based on imbalance modified kernel fisher discriminant analysis," *Chem. Eng. Res. Des.*, vol. 88, no. 8, pp. 936–951, 2010.
[11] W. Khreich, E. Granger, A. Miri, and R. Sabourin, "Iterative boolean combination of classifiers in the roc space: An application to anomaly detection with hmms," *Pattern Recogn.*, vol. 43, no. 8, pp. 2732–2752, 2010.
[12] M. Tavallaee, N. Stakhanova, and A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev*, vol. 40, no. 5, pp. 516–524, Sep. 2010.
[13] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural Netw.*, vol. 21, no. 2–3, pp. 427–436, 2008.
[14] P. Bermejo, J. A. Gámez, and J. M. Puerta, "Improving the performance of naive bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2072–2080, 2011.
[15] Y.-H. Liu and Y.-T. Chen, "Total margin-based adaptive fuzzy support vector machines for multiview face recognition," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2005, vol. 2, pp. 1704–1711.
[16] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Mach. Learn.*, vol. 30, pp. 195–215, 1998.
[17] J. R. Quinlan, "Improved estimates for the accuracy of small disjuncts," *Mach. Learn.*, vol. 6, pp. 93–98, 1991.
[18] B. Zadrozny and C. Elkan, "Learning and making decisions when costs and probabilities are both unknown," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, New York, 2001, pp. 204–213.
[19] G. Wu and E. Chang, "KBA: kernel boundary alignment considering imbalanced data distribution," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 786–795, Jun. 2005.
[20] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Expl. Newslett.*, vol. 6, pp. 20–29, 2004.
[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.
[22] N. V. Chawla, N. Japkowicz, and A. Kolcz, Eds., *Special Issue Learning Imbalanced Datasets, SIGKDD Explor. Newsl.,* vol. 6, no. 1, 2004.
[23] N. Chawla, D. Cieslak, L. Hall, and A. Joshi, "Automatically countering imbalance and its empirical relationship to cost," *Data Min. Knowl. Discov.*, vol. 17, pp. 225–252, 2008.
[24] A. Freitas, A. Costa-Pereira, and P. Brazdil, "Cost-sensitive decision trees applied to medical data," in *Data Warehousing Knowl. Discov. (Lecture Notes Series in Computer Science)*, I. Song, J. Eder, and T. Nguyen, Eds., Berlin/Heidelberg, Germany: Springer, 2007, vol. 4654, pp. 303–312.
[25] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, 2006.
[26] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, pp. 1–39, 2010.
[27] N. C. Oza and K. Tumer, "Classifier ensembles: Select real-world applications," *Inf. Fusion*, vol. 9, no. 1, pp. 4–20, 2008.
[28] C. Silva, U. Lotric, B. Ribeiro, and A. Dobnikar, "Distributed text classification with an ensemble kernel-based learning approach," *IEEE Trans. Syst., Man, Cybern. C*, vol. 40, no. 3, pp. 287–297, May. 2010.
[29] Y. Yang and K. Chen, "Time series clustering via RPCL network ensemble with different representations," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 2, pp. 190–199, Mar. 2011.
[30] Y. Xu, X. Cao, and H. Qiao, "An efficient tree classifier ensemble-based approach for pedestrian detection," *IEEE Trans. Syst., Man, Cybern. B, Cybern*, vol. 41, no. 1, pp. 107–117, Feb. 2011.
[31] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 66–75, Jan. 1994.
[32] T. K. Ho, "Multiple classifier combination: Lessons and next steps," in *Hybrid Methods in Pattern Recognition*, Kandel and Bunke, Eds. Singapore: World Scientific, 2002, pp. 171–198.
[33] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *Proc. IEEE Int. Conf. Neural Netw.*, 1996, vol. 1, pp. 90–95.
[34] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1995, vol. 7, pp. 231–238.
[35] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: A survey and categorization," *Inf. Fusion*, vol. 6, no. 1, pp. 5–20, 2005 (diversity in multiple classifier systems).
[36] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connect. Sci.*, vol. 8, no. 3–4, pp. 385–404, 1996.
[37] X. Hu, "Using rough sets theory and database operations to construct a good ensemble of classifiers for data mining applications," in *Proc. IEEE Int. Conf. Data Mining*, 2001, pp. 233–240.
[38] L. I. Kuncheva, "Diversity in multiple classifier systems," *Inf. Fusion*, vol. 6, no. 1, pp. 3–4, 2005 (diversity in multiple classifier systems).
[39] L. Rokach, "Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography," *Comput. Stat. Data An.*, vol. 53, no. 12, pp. 4046–4072, 2009.
[40] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, pp. 197–227, 1990.
[41] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.
[42] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, pp. 123–140, 1996.
[43] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. New York: Wiley-Interscience, 2004.
[44] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proc. Knowl. Discov. Databases*, 2003, pp. 107–119.

[45] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[46] J. Błaszczyński, M. Deckert, J. Stefanowski, and S. Wilk, "Integrating selective pre-processing of imbalanced data with ivotes ensemble," in *Rough Sets and Current Trends in Computing* (Lecture Notes in Computer Science Series 6086), M. Szczuka, M. Kryszkiewicz, S. Ramanna, R. Jensen, and Q. Hu, Eds. Berlin/Heidelberg, Germany: Springer-Verlag, 2010, pp. 148–157.

[47] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern. B, Appl. Rev*, vol. 39, no. 2, pp. 539–550, 2009.

[48] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "Adacost: Misclassification cost-sensitive boosting," presented at the 6th Int. Conf. Mach. Learning, pp. 97–105, San Francisco, CA, 1999.

[49] K. M. Ting, "A comparative study of cost-sensitive boosting algorithms," in *Proc. 17th Int. Conf. Mach. Learning*, Stanford, CA, 2000, pp. 983–990.

[50] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recog.*, vol. 40, no. 12, pp. 3358–3378, 2007.

[51] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Comput. Intell.*, vol. 20, no. 1, pp. 18–36, 2004.

[52] J. Stefanowski and S. Wilk, "Selective pre-processing of imbalanced data for improving classification performance," in *Data Warehousing and Knowledge Discovery* (Lecture Notes in Computer Science Series 5182), I.-Y. Song, J. Eder, and T. Nguyen, Eds., 2008, pp. 283–292.

[53] A. Fernández, S. García, M. J. del Jesus, and F. Herrera, "A study of the behaviour of linguistic fuzzy-rule-based classification systems in the framework of imbalanced data-sets," *Fuzzy Sets Syst.*, vol. 159, no. 18, pp. 2378–2398, 2008.

[54] A. Orriols-Puig and E. Bernadó-Mansilla, "Evolutionary rule-based systems for imbalanced data sets," *Soft Comp.*, vol. 13, pp. 213–225, 2009.

[55] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *IEEE Symp. Comput. Intell. Data Mining*, 2009, pp. 324–331.

[56] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. Fernández, and F. Herrera, "KEEL: A software tool to assess evolutionary algorithms for data mining problems," *Soft Comp.*, vol. 13, no. 3, pp. 307–318, 2008.

[57] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Mult.-Valued Logic Soft Comput.*, vol. 17, no. 2–3, pp. 255–287, 2011.

[58] J. R. Quinlan, *C4.5: Programs for Machine Learning*, 1st ed. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[59] C.-T. Su and Y.-H. Hsiao, "An evaluation of the robustness of MTS for imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 10, pp. 1321–1332, Oct. 2007.

[60] D. Drown, T. Khoshgoftaar, and N. Seliya, "Evolutionary sampling and software quality modeling of high-assurance systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans.*, vol. 39, no. 5, pp. 1097–1107, Sep. 2009.

[61] S. García, A. Fernández, and F. Herrera, "Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems," *Appl. Soft Comput.*, vol. 9, no. 4, pp. 1304–1314, 2009.

[62] J. Van Hulse, T. Khoshgoftaar, and A. Napolitano, "An empirical comparison of repetitive undersampling techniques," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, 2009, pp. 29–34.

[63] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[64] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, pp. 2677–2694, 2008.

[65] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, pp. 2044–2064, 2010.

[66] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recog.*, vol. 30, no. 7, pp. 1145–1159, 1997.

[67] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, Mar. 2005.

[68] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[69] D. Williams, V. Myers, and M. Silvious, "Mine classification with imbalanced data," *IEEE Geosci. Remote Sens. Lett.*, vol. 6, no. 3, pp. 528–532, Jul. 2009.

[70] W.-Z. Lu and D. Wang, "Ground-level ozone prediction by support vector machine approach with a cost-sensitive classification scheme," *Sci. Total. Enviro.*, vol. 395, no. 2-3, pp. 109–116, 2008.

[71] Y.-M. Huang, C.-M. Hung, and H. C. Jiau, "Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem," *Nonlinear Anal. R. World Appl.*, vol. 7, no. 4, pp. 720–747, 2006.

[72] D. Cieslak, N. Chawla, and A. Striegel, "Combating imbalance in network intrusion datasets," in *IEEE Int. Conf. Granular Comput.*, 2006, pp. 732–737.

[73] K. Kiliç, Özge Uncu and I. B. Türksen, "Comparison of different strategies of utilizing fuzzy clustering in structure identification," *Inf. Sci.*, vol. 177, no. 23, pp. 5153–5162, 2007.

[74] M. E. Celebi, H. A. Kingravi, B. Uddin, H. Iyatomi, Y. A. Aslandogan, W. V. Stoecker, and R. H. Moss, "A methodological approach to the classification of dermoscopy images," *Comput. Med. Imag. Grap.*, vol. 31, no. 6, pp. 362–373, 2007.

[75] X. Peng and I. King, "Robust BMPM training based on second-order cone programming and its application in medical diagnosis," *Neural Netw.*, vol. 21, no. 2–3, pp. 450–457, 2008.

[76] B. Liu, Y. Ma, and C. Wong, "Improving an association rule based classifier," in *Principles of Data Mining and Knowledge Discovery* (Lecture Notes in Computer Science Series 1910), D. Zighed, J. Komorowski, and J. Zytkow, Eds., 2000, pp. 293–317.

[77] Y. Lin, Y. Lee, and G. Wahba, "Support vector machines for classification in nonstandard situations," *Mach. Learn.*, vol. 46, pp. 191–202, 2002.

[78] R. Barandela, J. S. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems," *Pattern Recog.*, vol. 36, no. 3, pp. 849–851, 2003.

[79] K. Napierała, J. Stefanowski, and S. Wilk, "Learning from Imbalanced data in presence of noisy and borderline examples," in *Rough Sets Curr. Trends Comput.*, 2010, pp. 158–167.

[80] C. Ling, V. Sheng, and Q. Yang, "Test strategies for cost-sensitive decision trees," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1055–1067, 2006.

[81] S. Zhang, L. Liu, X. Zhu, and C. Zhang, "A strategy for attributes selection in cost-sensitive decision trees induction," in *Proc. IEEE 8th Int. Conf. Comput. Inf. Technol. Workshops*, 2008, pp. 8–13.

[82] S. Hu, Y. Liang, L. Ma, and Y. He, "MSMOTE: Improving classification performance when training data is imbalanced," in *Proc. 2nd Int. Workshop Comput. Sci. Eng.*, 2009, vol. 2, pp. 13–17.

[83] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.

[84] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, pp. 1–58, 1992.

[85] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Ann. Statist.*, vol. 28, pp. 337–407, 1998.

[86] E. B. Kong and T. G. Dietterich, "Error-correcting output coding corrects bias and variance," in *Proc. 12th Int. Conf. Mach. Learning*, 1995, pp. 313–321.

[87] R. Kohavi and D. H. Wolpert, "Bias plus variance decomposition for zero-one loss functions," in *Proc. 13th Int. Conf. Mach. Learning*, 1996.

[88] L. Breiman, "Bias, variance, and arcing classifiers," University of California, Berkeley, CA, Tech. Rep. 460, 1996.

[89] R. Tibshirani, "Bias, variance and prediction error for classification rules," University of Toronto, Toronto, Canada, Dept. of Statistic, Tech. Rep. 9602, 1996.

[90] J. H. Friedman, "On bias, variance, 0/1-loss, and the curse-of-dimensionality," *Data Min. Knowl. Disc*, vol. 1, pp. 55–77, 1997.

[91] G. M. James, "Variance and bias for general loss functions," *Mach. Learning*, vol. 51, pp. 115–135, 2003.

[92] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learning*, vol. 51, pp. 181–207, 2003.

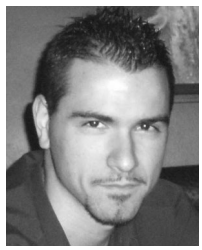
[93] L. Breiman, "Pasting small votes for classification in large databases and on-line," *Mach. Learn.*, vol. 36, pp. 85–103, 1999.

[94] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, pp. 1–37, 2007.

[95] C. Rudin, I. Daubechies, and R. E. Schapire, "The dynamics of AdaBoost: Cyclic behavior and convergence of margins," *J. Mach. Learn. Res.*, vol. 5, pp. 1557–1595, 2004.
[96] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, pp. 297–336, 1999.
[97] M. Joshi, V. Kumar, and R. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements," in *Proc. IEEE Int. Conf. Data Mining*, 2001, pp. 257–264.
[98] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: The DataBoost-IM approach," *SIGKDD Expl. Newsl.*, vol. 6, pp. 30–39, 2004.
[99] R. Barandela, R. M. Valdovinos, and J. S. Sánchez, "New applications of ensembles of classifiers," *Pattern Anal. App.*, vol. 6, pp. 245–256, 2003.
[100] E. Chang, B. Li, G. Wu, and K. Goh, "Statistical learning for effective visual information retrieval," in *Proc. Int. Conf. Image Process.*, 2003, vol. 3, no. 2, pp. 609–612.
[101] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, pp. 1088–1099, Jul. 2006.
[102] S. Hido, H. Kashima, and Y. Takahashi, "Roughly balanced bagging for imbalanced data," *Stat. Anal. Data Min.*, vol. 2, pp. 412–426, 2009.
[103] P. K. Chan and S. J. Stolfo, "Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection," in *Proc. 4th Int. Conf. Knowl. Discov. Data Mining (KDD-98)*, 1998, pp. 164–168.
[104] R. Yan, Y. Liu, R. Jin, and A. Hauptmann, "On predicting rare classes with SVM ensembles in scene classification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2003, vol. 3, pp. 21–24.
[105] C. Li, "Classifying imbalanced data using a bagging ensemble variation (BEV)," in *Proc. 45th Annual Southeast Regional Conference* (Association of Computing Machinery South East Series 45). New York: ACM, 2007, pp. 203–208.
[106] D. A. Cieslak and N. V. Chawla, "Learning decision trees for unbalanced data," in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Computer Science Series 5211), W. Daelemans, B. Goethals, and K. Morik, Eds., 2008, pp. 241–256.
[107] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Mach. Learn.*, vol. 52, pp. 199–215, 2003.
[108] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability," *Soft Comp.*, vol. 13, no. 10, pp. 959–977, 2009.
[109] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.
[110] D. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 2nd ed. London, U.K.: Chapman & Hall/CRC, 2006.
[111] S. Holm, "A simple sequentially rejective multiple test procedure," *Scand. J. Stat.*, vol. 6, pp. 65–70, 1979.
[112] J. P. Shaffer, "Modified sequentially rejective multiple test procedures," *J. Am. Stat. Assoc.*, vol. 81, no. 395, pp. 826–831, 1986.

Mikel Galar received the M.Sc. degree in computer sciences from the Public University of Navarra, Pamplona, Spain, in 2009. He is working toward the Ph.D. degree with the department of Automatics and Computation, Universidad Pública de Navarra, Navarra, Spain.

He is currently a Teaching Assistant in the Department of Automatics and Computation. His research interests include data-minig, classification, multi-classification, ensemble learning, evolutionary algorithms and fuzzy systems.



Alberto Fernández received the M.Sc. and Ph.D. degrees in computer science in 2005 and 2010, both from the University of Granada, Granada, Spain.

He is currently an Assistant Professor in the Department of Computer Science, University of Jaén, Jaén, Spain. His research interests include data mining, classification in imbalanced domains, fuzzy rule learning, evolutionary algorithms and multi-classification problems.

Edurne Barrenechea received the M.Sc. degree in computer science at the Pais Vasco University, San Sebastian, Spain, in 1990. She obtained the Ph.D. degree in computer science from Public University of Navarra, Navarra, Spain, in 2005, on the topic interval-valued fuzzy sets applied to image processing.

She an Assistant Lecturer at the Department of Automatics and Computation, Public University of Navarra. She worked in a private company (Bombas Itur) as an Analyst Programmer from 1990 to 2001, and then she joined the Public University of Navarra as an Associate Lecturer. Her publications comprise more than 20 papers in international journals and about 15 book chapters. Her research interests include fuzzy techniques for image processing, fuzzy sets theory, interval type-2 fuzzy sets theory and applications, decision making, and medical and industrial applications of soft computing techniques.

Dr. Barrenechea is a Member of the board of the European Society for Fuzzy Logic and Technology.

Humberto Bustince (M'08) received the Ph.D. degree in mathematics from Public University of Navarra, Navarra, Spain, in 1994.

He is a Full Professor at the Department of Automatics and Computation, Public University of Navarra. His research interests include fuzzy logic theory, extensions of fuzzy sets (type-2 fuzzy sets, interval-valued fuzzy sets, Atanassov's intuitionistic fuzzy sets), fuzzy measures, aggregation functions and fuzzy techniques for Image processing. He is author of more than 65 published original articles and involved in teaching artificial intelligence for students of computer sciences.

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

He is currently a Professor with the Department of Computer Science and Artificial Intelligence at the University of Granada. He acts as an Associate Editor of the journals: IEEE TRANSACTIONS ON FUZZY SYSTEMS, *Information Sciences*, *Mathware and Soft Computing*, *Advances in Fuzzy Systems*, *Advances in Computational Sciences and Technology*, and *International Journal of Applied Metaheuristics Computing*. He currently serves as an Area Editor of the *Journal Soft Computing* (area of genetic algorithms and genetic fuzzy systems), and he serves as member of several journal editorial boards, among others: *Fuzzy Sets and Systems*, *Applied Intelligence*, *Knowledge and Information Systems*, *Information Fusion*, *Evolutionary Intelligence*, *International Journal of Hybrid Intelligent Systems*, *Memetic Computation*. He has published more than 150 papers in international journals. He is the coauthor of the book "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases" (World Scientific, 2001). As edited activities, he has co-edited five international books and co-edited 20 special issues in international journals on different Soft Computing topics. His current research interests include computing with words and decision making, data mining, data preparation, instance selection, fuzzy-rule-based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms, and genetic algorithms.