Knowledge-Based Systems 85 (2015) 96–111

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# Random Balance: Ensembles of variable priors classifiers for imbalanced data

José F. Díez-Pastor [a], Juan J. Rodríguez [a,*], César García-Osorio [a], Ludmila I. Kuncheva [b]

[a] Lenguajes y Sistemas Informáticos, Escuela Politécnica Superior, Avda de Cantabria s/n, 09006 Burgos, Spain
[b] School of Computer Science, Bangor University Dean Street, Bangor, Gwynedd LL57 1UT, United Kingdom

ARTICLE INFO

*Article history:*
Received 4 January 2013
Received in revised form 2 March 2015
Accepted 22 April 2015
Available online 7 May 2015

*Keywords:*
Classifier ensembles
Imbalanced data sets
Bagging
AdaBoost
SMOTE
Undersampling

ABSTRACT

In Machine Learning, a data set is imbalanced when the class proportions are highly skewed. Imbalanced data sets arise routinely in many application domains and pose a challenge to traditional classifiers. We propose a new approach to building ensembles of classifiers for two-class imbalanced data sets, called Random Balance. Each member of the Random Balance ensemble is trained with data sampled from the training set and augmented by artificial instances obtained using SMOTE. The novelty in the approach is that the proportions of the classes for each ensemble member are chosen randomly. The intuition behind the method is that the proposed diversity heuristic will ensure that the ensemble contains classifiers that are specialized for different operating points on the ROC space, thereby leading to larger AUC compared to other ensembles of classifiers. Experiments have been carried out to test the Random Balance approach by itself, and also in combination with standard ensemble methods. As a result, we propose a new ensemble creation method called RB-Boost which combines Random Balance with AdaBoost.M2. This combination involves enforcing random class proportions in addition to instance re-weighting. Experiments with 86 imbalanced data sets from two well known repositories demonstrate the advantage of the Random Balance approach.

© 2015 Elsevier B.V. All rights reserved.

## 1. Introduction

The class-imbalance problem occurs when there are many more instances of some classes than others [1]. Imbalanced data sets are common in fields such as bioinformatics (translation initiation site (TIS) recognition in DNA sequences [2], gene recognition [3]), engineering (non-destructive testing in weld flaws detection through visual inspection [4]), finance (predicting credit card customer churn [5]), fraud detection [6] and many more.

Bespoke methods are needed for imbalanced classes for at least three reasons [7]. Firstly, standard classifiers are driven by accuracy so the minority class may be ignored. Secondly, standard classification methods operate under the assumption that the data sample is a faithful representation of the population of interest, which is not always the case with imbalanced problems. Finally, the classification methods for imbalanced problems should allow for errors coming from different classes to have different costs.

Galar et al. [8] systemize the wealth of recent techniques and approaches into four categories:

(a) *Algorithm level approaches.* This category contains variants of existing classifier learning algorithms biased towards learning more accurately the minority class. Examples include decision tree algorithms insensitive to the class sizes, like Hellinger Distance Decision Tree (HDDT) [9], Class Confidence Proportion Decision Tree (CCPDT) [10] and a SVM classifier with different penalty constants for different classes [11].
(b) *Data level approaches.* The main idea in this category is to pre-process the data so as to transform the imbalanced problem into a balanced one by manipulating the distribution of the classes. These algorithms are often used in combination with ensembles of classifiers. This category can be further subdivided into methods that increase the number of minority class examples: Oversampling [12], SMOTE [13], Borderline-SMOTE [14] and Safelevel-SMOTE [15] among others; and methods that reduce the size of the majority class, such as random undersampling, this approach has been used both with and without replacement [16]. These techniques can be jointly applied to increase the size of the minority class while simultaneously decreasing the majority class.

* Corresponding author.
*E-mail addresses:* jfdpastor@ubu.es (J.F. Díez-Pastor), jjrodriguez@ubu.es (J.J. Rodríguez), cgosorio@ubu.es (C. García-Osorio), l.i.kuncheva@bangor.ac.uk (L.I. Kuncheva).

http://dx.doi.org/10.1016/j.knosys.2015.04.022
0950-7051/© 2015 Elsevier B.V. All rights reserved.

(c) *Cost-sensitive learning.* While traditional algorithms aim at increasing the accuracy by giving equal weights to the examples of any class, cost-sensitive methods, such as cost-sensitive decision trees [17] or cost-sensitive neural networks [18], assign a different cost to each class. The best known methods in this category are the cost-sensitive versions of AdaBoost: AdaCost [19,20], AdaC1, AdaC2 and AdaC3 [21].

(d) *Ensemble learning.* Classifier ensembles have often offered solutions to challenging problems where standard classification methods have been insufficient. One approach for constructing ensembles for imbalanced data is based on using data level approaches: each base classifier is trained with a pre-processed data set. As data level approaches usually use random values, the pre-processed data sets and the corresponding classifiers will be different. Another strategy is based on combining conventional ensemble methods (i.e., not specific for imbalance) with data level approaches. Examples of this strategy are SMOTEBagging [22], SMOTEBoost [23] and RUSBoost [24]. It is also possible to have ensembles that combine classifiers obtained with different methods [25].

In general, according to [8], algorithm level and cost-sensitive approaches are more data-dependent, whereas data level and ensemble learning methods are more versatile.

Here we propose a new preprocessing technique that can be used to build ensembles, for two-class imbalanced learning tasks, based on a simple randomisation heuristic. The data for training an ensemble member is sampled from the training data using random class proportions. The classes are either undersampled or augmented with artificial examples to make up such a sample.

The rest of the paper is structured as follows. Section 2 presents the performance measures used in the experimental evaluation. Section 3 briefly overviews some of the most relevant methods in imbalanced learning, those used in the experimental study. Section 4 explains the proposed method. In Section 5 we provide a simulation example that tries to give some insight in why the method works. An experimental study is reported in Section 6, and finally, Section 7 contains our conclusions and several future research lines.

## 2. Measures of performance for imbalanced data

When working with binary classification problems instances can be labelled as positive ($p$) or negative ($n$). In binary imbalanced data sets usually the minority class is considered positive while the majority class is considered negative. For a prediction there are 4 possible outcomes: (a) True Positive: prediction is $p$ and the real label is $p$. (b) True Negative: prediction is $n$ and the real label is $n$. (c) False Positive: prediction is $p$ and the real label is $n$. (d) False Negative: prediction is $n$ and the real label is $p$. Given a test dataset, containing $P$ examples of the positive class and $N$ examples of the negative class, $TP$ is the number of True Positives, $FP$ is the number of False Positives, $TN$ is the number of True Negatives and $FN$ the number of False Negatives.

The True Positive Rate ($TPR$), also called Sensitivity or Recall, is defined as $TP/P$ and False Positive Rate ($FPR$) is defined as $FP/N$. The precision is defined as $TP/(TP+FP)$.

Commonly used measures of performance for imbalanced data are the Area Under the ROC (Receiver Operation Characteristic) curve [26], the F-Measure [27] and the Geometric Mean [28]. The F-Measure is defined as $2 \times \frac{precision \times recall}{precision + recall}$. The Geometric Mean is defined as $\sqrt{TP/P \times TN/N}$. The ROC Curve is a two-dimensional representation of classifier performance, it is created by plotting the $TPR$ against the $FPR$ for different decision thresholds. The Area Under the ROC curve (AUC) is a way to represent the performance of a binary classifier using a scalar.

## 3. Classification methods for imbalanced problems

In recent years, numerous techniques have been developed to deal with the problem of class-imbalance datasets. This section is a sort summary of the subset of methods tested in this article. The methods are organized using the same classification presented in the introduction:

- *Data level approaches.*
  - Random Undersampling. This technique will randomly drop some of the examples of the majority class. When it comes to sampling without replacement, an example of the minority class can appear only once in the sub-sampling; with replacement, the same example can appear multiple times.
  - Random oversampling [12] consists of adding exact copies of some minority class examples. With this technique overfitting is more common than in the prior technique.
  - SMOTE (Synthetic Minority Over-sampling Technique [13]) although this technique has "oversampling" in the name, it does not add copies of existing instances, but creates new artificial examples using the following procedure: a member of the minority class is selected and its $k$ nearest neighbours (from the minority class) are identified. One of them is randomly selected. Then, the new example added to the set is a random point in the line segment defined by the member and its neighbour. A value of $k = 5$ has been recommended and is the one used in this study. This method tries to avoid overfitting using a random procedure to create the new samples, but this can introduce noise or nonsensical samples.
- *Ensemble learning.* One of the keys for good performance of ensembles is the diversity, there are several ways to inject diversity into an ensemble, the most common is the use of sampling. In Bagging [29], each base classifier is obtained from a random sample of the training data. In AdaBoost [30] the resampling is based on a weighted distribution, the weights are modified depending on the correctness of the prediction for the example given by the previous classifier. Bagging and AdaBoost have been modified to deal with imbalanced datasets:
  - SMOTEBagging [22] combines Bagging with different amounts of SMOTE and Oversampling in each iteration, so that the data set is completely balanced and consists of three parts: (*i*) a sample with replacement of the majority class, keeping the original size; (*ii*) oversampling of the minority class; and (*iii*) SMOTE of the minority class. The Oversampling percentage varies in each iteration (ranging from 10% in the first iteration to 100% in the last.) The rest of the positive instances are generated by the SMOTE algorithm.
  - SMOTEBoost [23] and RUSBoost [24] are both modifications of AdaBoost.M2 [30], in each iteration, besides the instance reweighting done according to the algorithm Adaboost.M2, SMOTE or Random undersampling is applied to the training set of the base classifier. Boosting based ensembles tend to perform better than bagging based ensembles, however, in Boosting based ensembles, the base classifiers are trained in sequence which slows down the training, and they are more sensitive to noise. SMOTEBoost and RUSBoost are more robust to noise because they introduce a high degree of randomness by creating or deleting instances.
  - Although the most popular methods are modifications or variations of bagging or boosting, there are methods that do not perform resampling, oversampling or undersampling and, instead of that, they make partitions. One method

described in [31], which will be called "Partitioning" in this paper, is similar to undersampling based ensembles, it breaks the majority class into several disjoint partitions and constructs several models which use one partition from the majority class and the entire minority class.

– Most of the above methods, at the same time they increase accuracy in minority class, they decrease overall accuracy compared to traditional learning algorithms. Some approaches combine both types of classifiers, one trained with the original skewed data and other trained according one of the previous approaches in an attempt to cope with the imbalance. Reliability Based Classifier [32] trains two classifiers and then chooses between the output of the classifier trained on the original skewed distribution and the output of the classifier trained according to a learning method addressing the curse of imbalanced data. This decision is guided by a parameter whose value maximizes, on a validation set, the accuracy and a measure designed to evaluate the performance of a classifier in imbalanced classifiers, such as the geometric mean.

## 4. Random Balance and RB-Boost ensembles

This section presents the main contribution of the paper. In this section we present a new preprocessing technique called Random Balance, this technique can be used within an ensemble to increase the diversity and deal with imbalance. We also describe a new ensemble method for imbalanced learning called RB-Boost (Random Balance Boost) which is a Random Balance modification of AdaBoost.M2. We also explain the intuition behind the method in an aside subsection.

When dealing with imbalanced dataset the three common data-level approaches to balancing the classes are listed below[1]:

- The new data set is formed by taking the entire minority class and a random subsample from the majority class. The method has a parameter $N$ that is the desired percentage of instances that belongs to the majority class in the processed dataset. For example, consider a data set with 20 instances in the minority class and 480 instances in the majority class. For $N = 40$, the desired number of instances from the majority class is 30 so that the 20 instances of the minority class make up 40% of the data.
- The new data set is formed by adding to it $(M/100) \times size\ Minority$ synthetic instances of the minority class using the SMOTE method. The amount of artificial instances is expressed as a percentage $M$ of the size of the minority class, and is again a parameter of the algorithm. In the example above, if we choose $M = 200$, 40 examples from the minority class will be generated through SMOTE.
- Use both undersampling and oversampling through SMOTE to reach a desired new size of the data and proportions of the classes within.

The problem with these data-level approaches is that the optimal proportions depend on the data set and are hard to find, it is known that this proportions have a substantial influence on the performance of the classifier. The proposed method relies completely on randomness and repetition to try to overcome this problem.

### 4.1. Random Balance

While preprocessing techniques are commonly used to restore the balance of the class proportions to a given extent, Random Balance relies on a completely random ratio. This includes the case where the minority class is over-represented and the imbalance ratio is inverted.

An example of the sampling procedure can be seen in Fig. 1. Given a data set, a different data set of the same size is obtained for each member of ensemble where the imbalance ratio is chosen randomly. In this example, the initial proportions of both classes appears on the top. Classifiers $1, 2 \ldots, T$ are trained with variants of this data set where the ratio between classes varies randomly. In iteration 1, the imbalance ratio has been slightly reduced. In iteration 2, the ratio is reversed, the size of the previous minority class exceeds the size of the previous majority class. And in iteration 3, the minority class has become even smaller. All these cases are possible since the procedure is random.

The procedure is described in the pseudocode in Algorithm 1. The fundamental step is to randomly set the new size of the majority and minority classes (lines 6–7). Then SMOTE and Random Undersampling (resampling without replacement) are used to respectively increase or reduce the size of the classes to match the desired size (lines 8–11 or lines 12–15 as required.).

We call this generic ensemble method *Ensemble-RB*. Additionally, it can be combined with Bagging, resulting in what we call *Bagging-RB*.

Pre-processing strategies can have important drawbacks. Undersampling can throw out potentially useful data, while SMOTE increases the size of the dataset and hence the training time. Random-Balance maintains the size of the training set and because it is a process which is repeated several times, the problem of removing important examples is reduced.

**Algorithm 1.** Pseudocode for the Random Balance ensemble method.

RANDOM BALANCEX
**Require:** Set $S$ of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ where $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathbf{Y} = \{-1, +1\}$ ($+1$: positive or minority class, $-1$: negative or majority class), neighbours used in SMOTE, $k$
**Ensure:** New set $S'$ of examples with Random Balance
1: $totalSize \leftarrow |S|$
2: $S_N \leftarrow \{(\mathbf{x}_i, y_i) \in S | y_i = -1\}$
3: $S_P \leftarrow \{(\mathbf{x}_i, y_i) \in S | y_i = +1\}$
4: $majoritySize \leftarrow |S_N|$
5: $minoritySize \leftarrow |S_P|$
6: $newMajoritySize \leftarrow$ Random integer between 2 and $totalSize$-2
// *Resulting classes will have at least 2 instances*
7: $newMinoritySize \leftarrow totalSize - newMajoritySize$
8: **if** $newMajoritySize < majoritySize$ **then**
9: $S' \leftarrow S_P$
10: Take a random sample of size $newMajoritySize$ from $S_N$, add the sample to $S'$.
11: Create $newMinoritySize - minoritySize$ artificial examples from $S_P$ using SMOTE, add these examples to $S'$.
12: **else**
13: $S' \leftarrow S_N$
14: Take a random sample of size $newMinoritySize$ from $S_P$, add the sample to $S'$.
15: Create $newMajoritySize - majoritySize$ artificial examples from $S_N$ using SMOTE, add these examples to $S'$.
16: **end if**
17: **return** $S'$

[1] Note that although random undersampling and SMOTE are mentioned because they are the most used techniques, more sophisticated techniques could be used resulting in variants of the proposed method.

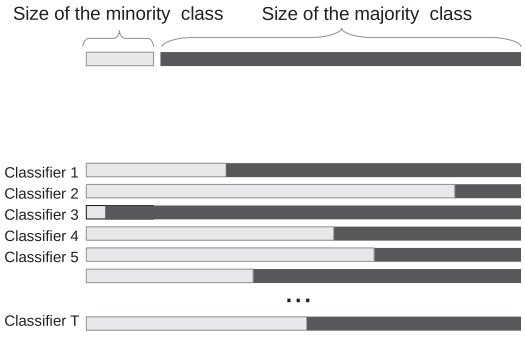



**Fig. 1.** Example of data sets used to train a Random Balance ensemble, note that the imbalance ratio is different for each dataset (even in favor of the minority class, for example, for the second classifier).

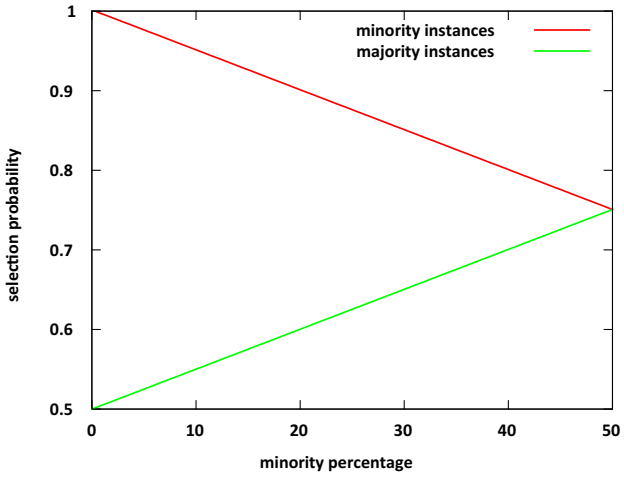



**Fig. 2.** Probabilities of including an instance in the transformed dataset, when the number of instances is $m = 1000$.

#### 4.1.1. Instance inclusion probability

The data sets generated in Random Balance have instances from the original training data and artificial instances. For Random Balance, the probability of including an instance is different for minority and majority instances. Given $p$ positive instances and $n$ negative instances with $m = n + p$ and assuming that $p \geqslant 2$ the probability of including an instance of the minority class is:

$$P_{mino} = \frac{1}{m-3}\left(\sum_{i=2}^{p-1}\frac{i}{p} + \sum_{i=p}^{m-2}1\right) = \frac{1}{m-3}\left(m - \frac{p+3}{2} - \frac{1}{p}\right)$$

In the generated data set, each class has at least two instances. Then, there are $n-3$ possible sizes of the minority class in the generated data sets (from 2 to $m-2$). The summation $\sum_{i=2}^{p-1}\frac{i}{p}$ is for the cases when the number of instances in the minority class is reduced (from $p$ instances we randomly take $i$ so the selection probability is $i/p$), while $\sum_{i=p}^{m-2}1$ is for the cases when the minority size is increased (the selection probability is 1).

Analogously, the probability of selecting an instance of the majority class will be:

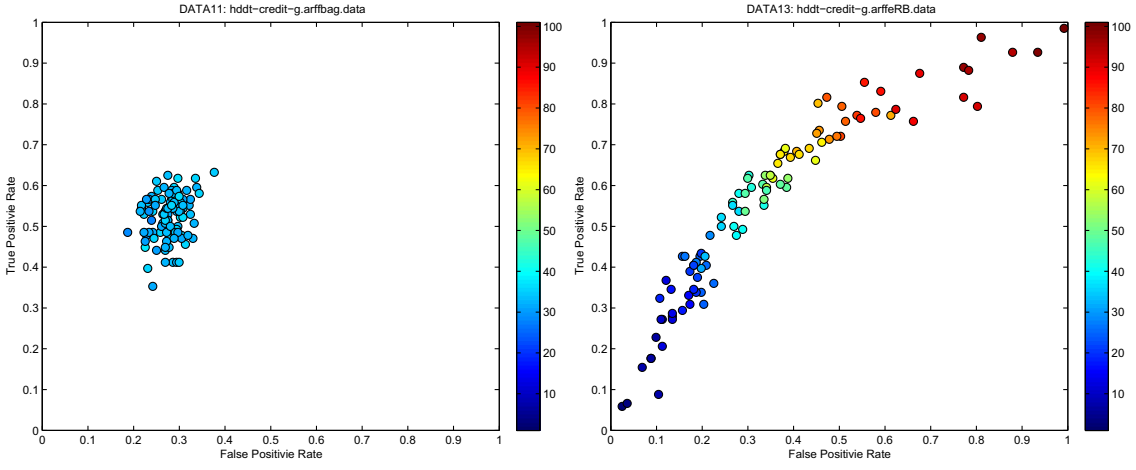
$$P_{majo} = \frac{1}{m-3}\left(m - \frac{n+3}{2} - \frac{1}{n}\right)$$

Fig. 2 shows the probabilities of selecting an instance in the generated data set as a function of the percentage of instances from the minority class, for a data set with 1000 instances. The probability of selecting an instance of the minority class decreases when the data set is more balanced.

It can be seen that if $p \leqslant n$ then $P_{majo} \leqslant P_{mino}, P_{mino} \geqslant 0.75$ and $P_{majo} \geqslant 0.5$. For a perfectly balanced data set, the probability of selecting an instance is a bit greater than 0.75 because there will be at least two instances of each class. The problem of discarding important instances of the majority class is ameliorated because the expected number of base classifiers that are trained with a given instance of the majority class is greater than 50%. Moreover, some of the instances included in the data set will also used to generate artificial instances.

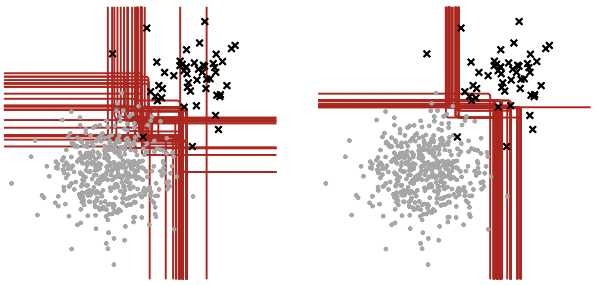
#### 4.1.2. Intuition behind the method

The ROC space is defined by *FPR* and *TPR* as $x$ and $y$ axes respectively because there is a trade-off between this two values. A classifier can be represented as a point in this space and all base classifiers in an ensemble can be represented as a cloud of points. Fig. 3a shows the cloud of points for a Bagging ensemble trained with the credit-g




(a) Bagging (b) Ensemble of Random Balance

**Fig. 3.** Base classifiers in the ROC Space (credit-g dataset). The color of each point represents the percentage of the instances than belong to the positive class in the dataset used for training that base classifier. Higher values (in red) represent that the imbalance ratio has been changed in favor of the minority class, values around 50 (in light blue/cyan) are for balanced cases, and lower values (in dark blue) the minority class has been made even smaller (originally credit-g dataset has 2.33 times more negative instances than positive). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Random Balance (b) Bagging

**Fig. 4.** An unbalanced data set and examples of the classification boundaries generated by two ensemble methods.

dataset, the color of each point represents the percentage of the instances than belong to the positive class in the dataset used for training that base classifier. It is easy to appreciate that all of the members of the ensemble are trained with samples which vary very slightly the proportion between classes. In contrast, Fig. 3b shows the cloud for an ensemble of Random Balance classifiers, the large variability in the ratio between classes in the datasets used to train each of the base classifiers, including cases in which the positive class becomes larger than the negative, makes the base classifiers of the ensemble spread out over the ROC space.

In the proposed method, the base classifiers are forced to learn different points on the ROC space and thereby expected to be more diverse and to improve the ensemble performance (see Fig. 3). Diversity is generally considered beneficial for ensemble methods, including the imbalanced case [33].

### 4.2. RB-Boost

There are several modification of AdaBoost.M2 for imbalanced problems. The best known of these methods is SMOTEBoost [23].

As in AdaBoost.M2, the examples of the training data have weights that are updated according to a pseudo-loss function. For each base classifier the weighted training data is augmented with artificial examples generated by SMOTE.

RUSBoost [24], as SMOTEBoost, is also an AdaBoost.M2 modification, but in this case instances of the majority class are removed using random undersampling in each iteration. No new weights are assigned; the weights of the remaining instances are normalized according to the new sum of weights of the data set. The rest of the procedure is as in AdaBoost.M2 and SMOTEBoost.

Both methods apply a preprocessing technique to the data and simultaneously alter the weights. Following this philosophy we propose *RB-Boost*, whose pseudocode is described in Algorithm 2. It is also a modification of AdaBoost.M2, in which line 3 is changed to generate a data set according to the procedure shown in Fig. 1. The number of instances removed by undersampling is equal to the number of instances introduced by SMOTE. The algorithm works as follows: for each of the $T$ rounds (lines 2–11) a data set $S'_t$ is generated according to the Random Balance procedure (line 3). Distribution $D'_t$ is updated, maintaining for each instance of the original data set its associated weight and assigning a uniform weight to the artificial examples (line 4). Then a weak learning algorithm is trained using $S'_t$ and $D'_t$ (line 5), this classifier will give a probability between 0 and 1 to each class.[2] The pseudo-loss $\varepsilon_t$ of the weak classifier $h_t$ is computed according to the formula presented in line 6. The distribution $D'_t$ is updated to make the weights associated with wrong classifications higher than the weights given to correct classifications (line 7–9). Finally, the different classifiers outputs are combined (line 11) taking into account their respective $\beta_t$ (obtained in line 7).

**Algorithm 2.** Pseudocode for the RB-Boost ensemble method.

RB-BOOST

**Require:** Set $S$ of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ where $\mathbf{x}_i \in \mathbf{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathbf{Y} = \{-1, +1\}$ ($+1$: positive or minority class, $-1$: negative or majority class),
Weak learner, *weakLearn*
Number of iterations, $T$
Number of neighbours used in SMOTE, $k$

**Ensure:** RB-Boost is built

// *Initialize distribution* $D_1$

1: $D_1(i) \leftarrow \frac{1}{m}$ **for** $i = 1, \ldots, m$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $S'_t \leftarrow$ **RandomBalance**$(S, k)$
4: $D'_t(i) \leftarrow D_t(j)$ **if** $S'_t(i) = S_t(j)$ **else** $\frac{1}{m}$, **for** $i = 1, \ldots, m$
// *If the example is from the sample it maintains its weight, if the example is artificial it has the initial weight.*
5: Using $S'_t$ and weights $D'_t$, train *weakLearn* $h_t : \mathbf{X} \times \mathbf{Y} \rightarrow [0, 1]$,
6: Compute the pseudo-loss of hypothesis $h_t$:

$$\varepsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(\mathbf{x}_i, y_i) + h_t(\mathbf{x}_i, y))$$

7: $\beta_t \leftarrow \varepsilon_t / (1 - \varepsilon_t)$
8: Update $D_t$:

$$D_{t+1}(i) \leftarrow D_t(i) \cdot \beta_t^{\frac{1}{2}(1 + h_t(\mathbf{x}_i, y_i) - h_t(\mathbf{x}_i, y))}$$

9: Normalize $D_{t+1}$: Let $Z_t \leftarrow \sum_i D_{t+1}(i)$

$$D_{t+1}(i) \leftarrow \frac{D_{t+1}(i)}{Z_t}$$

10: **end for**
11: **return** $h_f(\mathbf{x}) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t}\right) h_t(\mathbf{x}, y)$

## 5. A simulation experiment

To test-run the idea we carried out experiments with generated data. By contrasting the Random Balance with Bagging, we intend to gain more insight and support for our hypothesis that the Random Balance heuristic improves diversity in a way which leads to larger AUC.[3] We generated two 2-dimensional Gaussian classes centred at (0, 0) and (3, 3), both with identity covariance matrices. To simulate unbalanced classes, 450 points were sampled from the first class, and 50 points from the second class (10%). Each ensemble was composed of 50 decision tree classifiers.[4] The ensemble output was calculated as the average of the individual outputs. An example of the classification boundaries for the Random Balance ensemble and the Bagging ensemble is shown in Fig. 4. To evaluate the individual and ensemble accuracies as well as the AUC, we sampled a new data set from the same distribution and of the same size. The numerical results for this illustrative example are given in Table 1.

[2] In experiments, J48 classification tree with Laplace smoothing has been used as a weak classifier. The prediction returned by the classifier is the probability calculated taking into the instances that end in the leaf. With Laplace smoothing this is $(a_i + 1)/(A + c)$, where $a_i$ is the number of instances of class $i$ in the leaf, $A$ is the total number of instances in the leaf, and $c$ the number of classes.

[3] The varying parameter for the ROC curve is the threshold on the class membership probability estimated by the whole ensemble, not a particular base classifier.

[4] MATLAB's Statistic Toolbox was used for training the decision trees and estimating AUC.

**Table 1**
Comparison of Random Balance and bagging ensembles.

| Data sets | Ensemble | Individual error | Ensemble error | AUC |
|---|---|---|---|---|
| 1 Simulation (Fig. 4) | RB | 0.0272 | 0.0180 | 0.9979 |
| | Bagging | 0.0250 | 0.0220 | 0.9373 |
| 200 Simulations (average values) | RB | 0.0307 | 0.0162 | **0.9963** |
| | Bagging | **0.0192** | **0.0133** | 0.9917 |

It can be observed that the boundary lines for the Random Balance ensemble are more widely scattered compared to these for the Bagging ensemble, stepping well into the region of the majority class. Table 1 shows also the average results from 200 iterations, each iteration with freshly sampled training and testing data. The results indicate that: (*i*) individual errors of the decision trees for the ensemble-RB are larger than these for the Bagging ensemble, (*ii*) RB has a higher classification error than Bagging, and (*iii*) RB has a better AUC than Bagging. All differences were found to be statistically significant (two-tailed paired *t*-test, $p < 0.005$). This suggests that the better AUC produced by the ensemble-RB may come at the expense of slightly reduced classification accuracy. Since AUC is often viewed as the primary criterion for problems with unbalanced classes, the results of this simulation favor the ensemble-RB.

Kappa-error diagrams are often used for comparing classifier ensembles [34,35]. Consider a testing set with $N$ examples and the contingency table of two classifiers, $C_1$ and $C_2$.

| | $C_2$ correct | $C_2$ wrong |
|---|---|---|
| $C_1$ correct | a | b |
| $C_1$ wrong | c | d |

where the table entries are the number of examples jointly classified as indicated, and $a+b+c+d=N$. Diversity between the two classifiers is measured by $\kappa$ [36] as

$$\kappa = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)} \tag{1}$$

Kappa is plotted on the $x$-axis on the diagram. Smaller kappa indicates more diverse classifiers. The averaged individual error for the pair of classifiers is

$$e = \frac{1}{2}\left(\frac{c+d}{N}+\frac{b+d}{N}\right) = \frac{b+c+2d}{2N} \tag{2}$$

The error is plotted on the $y$-axis of the diagram. An ensemble with $L$ classifier generates a "cloud" of $L(L-1)/2$ points on the kappa-error diagram, one point for a pair of classifiers.

We calculated the centroid points of 200 RB and 200 Bagging ensemble clouds following the simulation protocol described above. Fig. 5 shows the kappa-error diagrams with the centroids, 200 in each subplot. The black points correspond to ensembles whose AUC is larger than the respective AUC of the rival ensemble. Out of the 200 ensembles, RB had larger AUC in 127 cases, which is seen as the larger proportion of black triangles in the left subplot compared to the proportion of black dots in the right subplot.

As expected, the ensemble-RB generates substantial diversity compared to Bagging, which is indicated by the stretch to the left of the set of points in the left subplot. The cloud of points is tilted, showing that the larger diversity is paid for by larger individual error. An interesting observation from the figure is that the black markers (triangles and dots) are spread uniformly along the point clouds, suggesting that there is no specific diversity–accuracy pattern which is symptomatic of better AUC.

## 6. Experimental setup and results

Two collections of data sets were used. The HDDT collection[5] contains the binary imbalanced data sets used in [37]. Table 2 shows the characteristics of the 20 data sets in this collection.

The KEEL collection[6] contains the binary imbalanced data sets from the repository of KEEL [38]. Table 3 shows the characteristics of the 66 data sets in this collection[7].

In both tables, the first column is the name of the data set, the second the number of examples, the third the number of attributes and the last is the imbalance ratio (the number of instances of the majority class for each instance of the minority class).

Many data sets in these two collections are available or are modifications of data sets in the UCI Repository [39].

Weka [40] was used for the experiments. The ensemble size was set to 100, for some methods this is not the exact size, but it is the maximum, since some method have a stopping criteria. J48 was chosen as the base classifier in all ensembles.[8] As recommended for imbalanced data [37], it was used without pruning and collapsing but with Laplace smoothing at the leaves. C4.5 with this options is called C4.4 [42].

The results were obtained with a $5 \times 2$-fold cross validation [43]. The data set is halved in two folds. One fold is used for training and the other for testing, and then the roles of the folds are reversed. This process is repeated five times. The results are the averages of these ten experiments. Cross validation was stratified: the class proportions was approximately preserved for each fold.

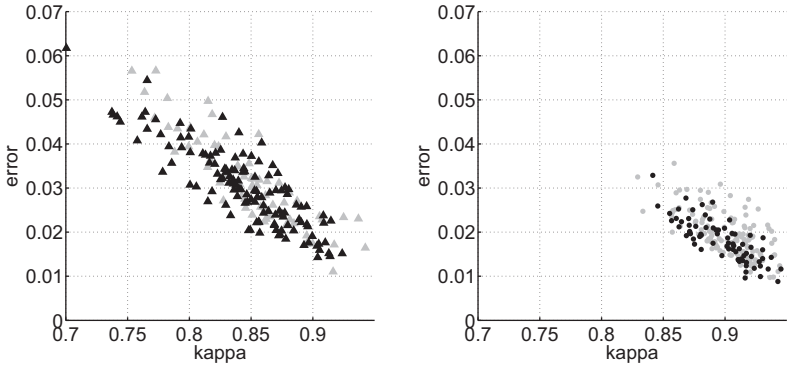
Given the large number of methods and variants tested, the comparisons are divided into families. Each family includes different types of classifier ensembles depending on the main diversity-generating strategy. We distinguished three such families: Data-preprocessing-only, Bagging and Boosting. The names, abbreviations and descriptions of the methods can be found in Tables 4–6.

The scores obtained by the proposed methods: E-RB, BAG-RB and RB-B are shown in Table 7, the reader is encouraged to consult the full table of results in Supplementary material. Some of the methods obtain low result in certain datasets. The reason is that some of the performance measures are a geometric mean (the G-mean) and a harmonic mean (the F-measure) so the results are biased towards the lower of the two values that are combined in the measure. With a classifier that always predict the majority class the accuracy will be very high (depending on the imbalance ratio), the AUC will be 0.5 if all the instances are given the same confidence; but for these two means the value will be 0.

We used the most common configurations of SMOTE where the number of synthetic instances was set to 100%, 200% and 500% of the minority class. In the variants called ESM and BAGSM, the minority class was oversampled to match the size of the majority class. For the undersampling ensembles, the size of the majority class was reduced to match the size of the minority class.

In addition, optimized versions of some the ensemble methods were tried. In the Data-preprocessing-only and the Bagging families we included three versions: optimizing the amount of SMOTE oversampling, optimizing the amount of Undersampling and optimizing both simultaneously. In all these variants we used a 5-fold internal cross-validation[9] and tested 10 different amounts

[5] Available at http://www.nd.edu/dial/hddt/.
[6] Available at http://sci2s.ugr.es/keel/imbalanced.php.
[7] Notice that several of the data sets come from data sets that were originally multiclass, the 66 datasets have been derived from 16 original sets.
[8] J48 is the Weka's re-implementation of C4.5 [41].
[9] That means that the training set is repeatedly divided into train and validation sets to find the optimal parameter value, and then the classifier is finally built using the complete training set.




(a) Random Balance (b) Bagging

**Fig. 5.** Kappa-error diagrams for the two ensemble methods. Black points indicate ensembles for which the AUC was larger than that for the rival method.

**Table 2**
Characteristics of the data sets from the HDDT collection.

| Data set | Examples | Attributes (numeric/nominal) | IR |
|---|---|---|---|
| boundary | 3505 | (0/175) | 27.50 |
| breast-y | 286 | (0/9) | 2.36 |
| cam | 18,916 | (0/132) | 19.08 |
| compustat | 13,657 | (20/0) | 25.26 |
| covtype | 38,500 | (10/0) | 13.02 |
| credit-g | 1000 | (7/13) | 2.33 |
| estate | 5322 | (12/0) | 7.37 |
| german-numer | 1000 | (24/0) | 2.33 |
| heart-v | 200 | (5/8) | 2.92 |
| hypo | 3163 | (7/18) | 19.95 |
| ism | 11,180 | (6/0) | 42.00 |
| letter | 20,000 | (16/0) | 24.35 |
| oil | 937 | (49/0) | 21.85 |
| optdigits | 5620 | (64/0) | 9.14 |
| page | 5473 | (10/0) | 8.77 |
| pendigits | 10,992 | (16/0) | 8.63 |
| phoneme | 5404 | (5/0) | 2.41 |
| PhosS | 11,411 | (480/0) | 17.62 |
| satimage | 6430 | (36/0) | 9.29 |
| segment | 2310 | (19/0) | 6.00 |

of SMOTE and Undersampling, which means that the version that optimizes both parameters simultaneously has evaluated 100 possible combinations. These amounts are expressed in terms of the difference between the majority and minority class sizes, as shown in Fig. 6, for SMOTE, in this case, a value of 0% means not to add any instance, a value of 100% means to create as many as necessary to match the size of the majority. For Undersampling a value of 0% means not to delete any instance, a value of 100% means remove instances to match the original size of the minority class. Once found, the parameters that maximize the AUC for a single decision tree are used for constructing the ensemble.

The Data-preprocessing-only also includes the Partitioning (or Random Splitting) method, described in [31]. In that work, the ensemble size was the Imbalanced Ratio, while in this work it is 100,[10] as for the other methods in this section, in order to make a fair comparison.

The Bagging family includes the Reliability-based Balancing (RbB) method [32]. The classifiers obtained with this method can be seen as a mini-ensemble of two classifiers, the first one using the original imbalanced class distribution (IC), the second one using a classifier with balanced data (BC). In order to have ensembles of 100 classifiers, two ensembles of 50 classifiers are combined. The first classifier is obtained with Bagging. For the second classifier two configurations are considered: Bagging with SMOTE and Bagging with Undersampling. RbB uses a threshold to determine which label return, when the reliability provided by IC is larger than the threshold, the final label corresponds to the label returned by IC, in the opposite case the label corresponds to the label returned by BC. This threshold is selected for each dataset, considering the values from 0.0 to 1.0 in steps of size 0.05, the threshold chosen is the one for which the sum of accuracy and geometric mean is maximized over a validation dataset.

The Data-preprocessing-only family includes the *Random Balance* ensemble (E-RB), while Bagging family includes the combination of Bagging and *Random Balance* (BAG-RB).

In the Boosting family, we have compared the most popular algorithms. For completeness, we included the standard boosting variants AdaBoost.M1 and MultiBoost. Both were tested with reweighting as well as with weighted resampling [44].

The main contenders in this family were the boosting variants especially designed for unbalanced data sets: SMOTEBoost, with three different rates of SMOTE, and RUSBoost.

The proposed method: *RB-Boost* was also added to the Boosting family.

For comparison between multiple algorithms for each family and multiple data sets we used average ranks [45]. For a given data set, the methods are sorted from best to worst. The best method receives rank 1, the second best receives rank 2, and so on. In case of a tie, average ranks are assigned. For instance, if two methods tie for the top rank, they both receive rank 1.5. Average ranks across all data sets are then obtained.

The first question is whether there are any significant differences between the ranks of the compared methods. The Friedman test and the subsequent version of Iman and Davenport [46] test were applied.

To detect pairwise differences between a designated method and the remaining methods, we used the Hochberg test [47], which was found to be more powerful than the Bonferroni–Dunn test [48,49].

Table 8a shows the results of the comparison of the algorithms of the Data-preprocessing-only family in the form of average ranking calculated from the area under the curve. The second column shows the average rank of each method. The Iman and Davenport test gives a $p$-value of 6.1904e−86, which means that it rejects the hypothesis that the compared algorithms are equivalent. The last column shows the adjusted Hochberg $p$-value between E-RB and the respective method of that row. An adjusted $p$-value less than 0.05 means that the two methods are

[10] To achieve this size, as many partitions as necessary are created. e.g. if the imbalance ratio is 5, the 100 classifiers are created using 20 times the partitioning technique.

**Table 3**
Characteristics of the data sets from the KEEL collection.

| data set | Examples | Attributes (numeric/nominal) | IR |
|---|---|---|---|
| abalone19 | 4174 | (7/1) | 129.44 |
| abalone9-18 | 731 | (7/1) | 16.40 |
| cleveland-0_vs_4 | 177 | (13/0) | 12.62 |
| ecoli-0-1-3-7_vs_2-6 | 281 | (7/0) | 39.14 |
| ecoli-0-1-4-6_vs_5 | 280 | (6/0) | 13.00 |
| ecoli-0-1-4-7_vs_2-3-5-6 | 336 | (7/0) | 10.59 |
| ecoli-0-1-4-7_vs_5-6 | 332 | (6/0) | 12.28 |
| ecoli-0-1_vs_2-3-5 | 244 | (7/0) | 9.17 |
| ecoli-0-1_vs_5 | 240 | (6/0) | 11.00 |
| ecoli-0-2-3-4_vs_5 | 202 | (7/0) | 9.10 |
| ecoli-0-2-6-7_vs_3-5 | 224 | (7/0) | 9.18 |
| ecoli-0-3-4-6_vs_5 | 205 | (7/0) | 9.25 |
| ecoli-0-3-4-7_vs_5-6 | 257 | (7/0) | 9.28 |
| ecoli-0-3-4_vs_5 | 200 | (7/0) | 9.00 |
| ecoli-0-4-6_vs_5 | 203 | (6/0) | 9.15 |
| ecoli-0-6-7_vs_3-5 | 222 | (7/0) | 9.09 |
| ecoli-0-6-7_vs_5 | 220 | (6/0) | 10.00 |
| ecoli-0_vs_1 | 220 | (7/0) | 1.86 |
| ecoli1 | 336 | (7/0) | 3.36 |
| ecoli2 | 336 | (7/0) | 5.46 |
| ecoli3 | 336 | (7/0) | 8.60 |
| ecoli4 | 336 | (7/0) | 15.80 |
| glass-0-1-2-3_vs_4-5-6 | 214 | (9/0) | 3.20 |
| glass-0-1-4-6_vs_2 | 205 | (9/0) | 11.06 |
| glass-0-1-5_vs_2 | 172 | (9/0) | 9.12 |
| glass-0-1-6_vs_2 | 192 | (9/0) | 10.29 |
| glass-0-1-6_vs_5 | 184 | (9/0) | 19.44 |
| glass-0-4_vs_5 | 92 | (9/0) | 9.22 |
| glass-0-6_vs_5 | 108 | (9/0) | 11.00 |
| glass0 | 214 | (9/0) | 2.06 |
| glass1 | 214 | (9/0) | 1.82 |
| glass2 | 214 | (9/0) | 11.59 |
| glass4 | 214 | (9/0) | 15.46 |
| glass5 | 214 | (9/0) | 22.78 |
| glass6 | 214 | (9/0) | 6.38 |
| haberman | 306 | (3/0) | 2.78 |
| iris0 | 150 | (4/0) | 2.00 |
| led7digit-0-2-4-5-6-7-8-9_vs_1 | 443 | (7/0) | 10.97 |
| new-thyroid1 | 215 | (5/0) | 5.14 |
| new-thyroid2 | 215 | (5/0) | 5.14 |
| page-blocks-1-3_vs_4 | 472 | (10/0) | 15.86 |
| page-blocks0 | 5472 | (10/0) | 8.79 |
| pima | 768 | (8/0) | 1.87 |
| segment0 | 2308 | (19/0) | 6.02 |
| shuttle-c0-vs-c4 | 1829 | (9/0) | 13.87 |
| shuttle-c2-vs-c4 | 129 | (9/0) | 20.50 |
| vehicle0 | 846 | (18/0) | 3.25 |
| vehicle1 | 846 | (18/0) | 2.90 |
| vehicle2 | 846 | (18/0) | 2.88 |
| vehicle3 | 846 | (18/0) | 2.99 |
| vowel0 | 988 | (13/0) | 9.98 |
| wisconsin | 683 | (9/0) | 1.86 |
| yeast-0-2-5-6_vs_3-7-8-9 | 1004 | (8/0) | 9.14 |
| yeast-0-2-5-7-9_vs_3-6-8 | 1004 | (8/0) | 9.14 |
| yeast-0-3-5-9_vs_7-8 | 506 | (8/0) | 9.12 |
| yeast-0-5-6-7-9_vs_4 | 528 | (8/0) | 9.35 |
| yeast-1-2-8-9_vs_7 | 947 | (8/0) | 30.57 |
| yeast-1-4-5-8_vs_7 | 693 | (8/0) | 22.10 |
| yeast-1_vs_7 | 459 | (7/0) | 14.30 |
| yeast-2_vs_4 | 514 | (8/0) | 9.08 |
| yeast-2_vs_8 | 482 | (8/0) | 23.10 |
| yeast1 | 1484 | (8/0) | 2.46 |
| yeast3 | 1484 | (8/0) | 8.10 |
| yeast4 | 1484 | (8/0) | 28.10 |
| yeast5 | 1484 | (8/0) | 32.73 |
| yeast6 | 1484 | (8/0) | 41.40 |

significantly different with a significance of $\alpha = 0.05$. The table shows that the Random Balance ensemble (E-RB) has a demonstrably better AUC than all the other ensembles in this family.

Table 8b shows the average ranks for the Bagging family calculated using the same measure. With $p$-value of 8.1240e−56, the Iman and Davenport test discards the hypothesis of equivalence between the algorithms. The combination of Bagging with the proposed method obtains the best ranking and also presents significant differences with the other methods.

Table 8c shows the average ranks for the Boosting family. With $p$-value of 1.0638e−37 the Iman and Davenport test discards the hypothesis of equivalence. The proposed algorithm *RB-Boost* takes the top spot for the AUC criterion, and there are significant differences with all other algorithms, except RUSBoost, which occupies the second position (adjusted Hochberg's $p$-value of 0.10634).

Table 9a shows the average ranks for the data-processing according to the F-Measure. In this case the Iman and Davenport test gives a $p$-value of 2.3794e−44, so the compared algorithms are not equivalent. The Random Balance ensemble gets the best ranking, but this time there are no statistically significant differences with the next three algorithms.

Table 9b shows the average ranks for the Bagging family according to the F-Measure. The Iman and Davenport test discards the hypothesis of equivalence between the algorithms with $p$-value of 1.2896e−23. The proposed method obtains the second highest ranking, but there are no significant differences from the first method.

Finally, Table 9c shows the average ranks for the Boosting family. With $p$-value of 6.912e−11 the Iman and Davenport test discards the hypothesis of equivalence between the algorithms. The proposed algorithm has the best place in the ranking with significant differences with all remaining algorithms in this family.

Fig. 7 shows scatter plots with the average ranks for the three families of methods. The best methods according to the AUC appear at the left, and the best methods according to the F-Measure appear at the bottom.

Similar patterns appear in the left and center plots:

- In the case of data processing family, ensembles which only use Random Undersampling (ERUS, ERUSR and EPart) obtain the three worst results for the F-Measure but according to the AUC criterion they are much better, only surpassed by E-RB.
- The ensembles that apply only SMOTE (ESM/100/200/500, EopS) are grouped into a cluster and methods that combine bagging and SMOTE (BAGSM/100/200/500, BAGopS) are grouped into another cluster.
- The proposed method appears far ahead of the other methods on the AUC criterion and is the best or the second best on the F-Measure criterion.

The right plot, showing the Boosting family, reveals that the methods are much closer to the diagonal line where the ranks for the AUC and the F-Measure are identical. The proposed method *RB-Boost* is located at a considerable distance from the other methods on both axes, which indicates its advantage.

Table 10 shows the rankings of the three families according to the geometric mean. The proposed methods get the best positions in the data processing and bagging families, in both cases significantly according to Hochberg's Test. But it gets the third position in the Boosting family ranking.

Although accuracy is not usually considered an adequate performance measure for imbalanced data, for the sake of completeness, Table 11 shows the average ranks for the considered ensemble methods according to this measure. As it could be expected, the methods that do not consider imbalance (i.e., Bagging, AdaBoost and MultiBoost) have the top ranks for their respective families. .

In this paper we have used several different measures to evaluate the performance of various methods. Some measures such as AUC, F-Measure and Geometric Mean are specific to unbalanced datasets, while accuracy is not specific to unbalanced. A combined average rank has been calculated to show the overall performance

**Table 4**
Algorithms used in the experimental study: data-processing family.

| Data-processing-only based ensembles | | |
|---|---|---|
| Abbr. | Method | Details |
| ESM100 | Ensemble, SMOTE = 100% | Amount of SMOTE in each iteration equal to 100% size of the minority class |
| ESM200 | Ensemble, SMOTE = 200% | Amount of SMOTE in each iteration equal to 200% size of the minority class |
| ESM500 | Ensemble, SMOTE = 500% | Amount of SMOTE in each iteration equal to 500% size of the minority class |
| ESM | Ensemble, SMOTE | SMOTE in each iteration until 50% of the data belongs to minority class |
| ERUS | Ensemble, RUS | Random Undersampling in each iteration until 50% of the data belongs to minority class |
| ERUSR | Ensemble, RUS with replacement | Random Undersampling with replacement in each iteration until 50% of the data belongs to minority class |
| EPart | Ensemble, Partitioning | Build balanced training sets by splitting the majority class into subsets. |
| EopS | Ensemble, optimized SMOTE | Amount of SMOTE selected by cross validation |
| EopU | Ensemble, optimized Undersampling | Amount of Random Undersampling selected by cross validation |
| EopB | Ensemble, optimized Both | Amounts of SMOTE and Undersampling selected by cross validation |
| E-RB | Ensemble, Random Balance | Random Balance in each iteration |

**Table 5**
Algorithms used in the experimental study: bagging family.

| Bagging based ensembles | | |
|---|---|---|
| Abbr. | Method | Details |
| SMBAG | SMOTEBagging | |
| BAG | Bagging | |
| BAGSM100 | Bagging, SMOTE = 100% | Amount of SMOTE in each iteration equal to 100% size of the minority class |
| BAGSM200 | Bagging, SMOTE = 200% | Amount of SMOTE in each iteration equal to 200% size of the minority class |
| BAGSM500 | Bagging, SMOTE = 500% | Amount of SMOTE in each iteration equal to 500% size of the minority class |
| BAGSM | Bagging, SMOTE | SMOTE in each iteration until 50% of the data belongs to minority class |
| BAGRUS | Bagging, RUS | Random Undersampling in each iteration until 50% of the data belongs to minority class |
| RbB:IC+BAGSM | Reliability-based Balancing with SMOTE | Miniensemble formed by Bagging and Bagging+SMOTE in each iteration until 50% of the data belongs to minority class |
| RbB:IC+BAGRUS | Reliability-based Balancing with UnderSampling | Miniensemble formed by Bagging and Bagging+Random Undersampling in each iteration until 50% of the data belongs to minority class |
| BAGopS | Bagging, optimized SMOTE | Amount of SMOTE selected by cross validation |
| BAGopU | Bagging, optimized Undersampling | Amount of Random Undersampling selected by cross validation |
| BAGopB | Bagging, optimized Both | Amounts of SMOTE and Undersampling selected by cross validation |
| BAG-RB | Bagging, Random Balance | Random Balance in each iteration |

**Table 6**
Algorithms used in the experimental study: Boosting family.

| Boosting based ensembles | | |
|---|---|---|
| Abbr. | Method | Details |
| AdaM1W | AdaBoost.M1 using reweighting | |
| AdaM1S | AdaBoost.M1 using resampling | |
| MultiW | MultiBoost using reweighting | Number of subcommittees = 10 |
| MultiS | MultiBoost using resampling | Number of subcommittees = 10 |
| SB100 | SMOTEBoost, SMOTE = 100% | Amount of SMOTE in each iteration equal to 100% size of the minority class |
| SB200 | SMOTEBoost, SMOTE = 200% | Amount of SMOTE in each iteration equal to 200% size of the minority class |
| SB500 | SMOTEBoost, SMOTE = 500% | Amount of SMOTE in each iteration equal to 500% size of the minority class |
| RUSB | RUSBoost | Random Undersampling in each iteration until 50% of the data belongs to minority class |
| RB-B | RB-Boost | Random Balance in each iteration |

of the four measures. This time the average rank for each method is the average of their average ranks for each measure. Table 12 shows the average ranks for the considered ensemble methods according to the combination of measures. In all families, the proposed method obtains the best position. In this case it is not appropriate to apply any test to detect equivalence between methods or pairwise differences because the values are not independent, for each dataset-algorithm pair there are several values (one per measure).

After comparing the methods within their own families, we performed a comparison between the methods that have achieved first place in their respective rankings.

Table 13 shows the average ranks for the best methods in each family, calculated for each different measure. With $p$-values of 8.113e−6 and 0.04933 the Iman and Davenport test discards the hypothesis of equivalence between the algorithms in AUC and F-Measure. By contrast a $p$-value of 0.91474, in the case of ranking calculated with the best methods according to their geometric means, indicates that there is not significant differences between methods, RUSBoost obtains the top position but it is equivalent to the next two methods.

In the ranking calculated from the AUC, the best position is for *Bagging-RB*, which shows significant differences with *Ensemble-RB*. In the ranking calculated with the F-Measure, the best position is

**Table 7**
Scores of the proposed methods according to de AUC, F-Measure and geometric mean.

| Dataset | AUC | | | F-Measure | | | Geometric mean | | |
|---|---|---|---|---|---|---|---|---|---|
| | E-RB | BAG-RB | RB-B | E-RB | BAG-RB | RB-B | E-RB | BAG-RB | RB-B |
| hddt_boundary | 0.6748 | 0.6945 | 0.7085 | 0.1421 | 0.1132 | 0.0388 | 0.3552 | 0.2730 | 0.1320 |
| hddt_breast-y | 0.6414 | 0.6460 | 0.6223 | 0.4417 | 0.4496 | 0.4023 | 0.5805 | 0.5872 | 0.5454 |
| hddt_cam | 0.7277 | 0.7631 | 0.7665 | 0.1922 | 0.1916 | 0.1356 | 0.3715 | 0.3610 | 0.2916 |
| hddt_compustat | 0.9072 | 0.9107 | 0.9320 | 0.3404 | 0.3632 | 0.4538 | 0.7924 | 0.7780 | 0.5965 |
| hddt_covtype | 0.9933 | 0.9934 | 0.9959 | 0.8517 | 0.8586 | 0.9055 | 0.9587 | 0.9555 | 0.9439 |
| hddt_credit-g | 0.7508 | 0.7695 | 0.7546 | 0.5536 | 0.5790 | 0.5173 | 0.6723 | 0.6941 | 0.6334 |
| hddt_estate | 0.6239 | 0.6239 | 0.6138 | 0.2425 | 0.2373 | 0.0813 | 0.5320 | 0.5266 | 0.2176 |
| hddt_german-numer | 0.7750 | 0.7856 | 0.7626 | 0.5819 | 0.6002 | 0.5334 | 0.6965 | 0.7134 | 0.6438 |
| hddt_heart-v | 0.6907 | 0.7067 | 0.7056 | 0.4250 | 0.4350 | 0.4107 | 0.5822 | 0.5871 | 0.5617 |
| hddt_hypo | 0.9911 | 0.9905 | 0.9925 | 0.8685 | 0.8793 | 0.8863 | 0.9610 | 0.9635 | 0.9432 |
| hddt_ism | 0.9394 | 0.9421 | 0.9130 | 0.5359 | 0.5660 | 0.6804 | 0.8860 | 0.8836 | 0.8308 |
| hddt_letter | 0.9990 | 0.9994 | 0.9999 | 0.9569 | 0.9618 | 0.9768 | 0.9747 | 0.9719 | 0.9779 |
| hddt_oil | 0.9128 | 0.9201 | 0.9281 | 0.4510 | 0.5254 | 0.5504 | 0.7642 | 0.7454 | 0.6679 |
| hddt_optdigits | 0.9986 | 0.9980 | 0.9999 | 0.9793 | 0.9811 | 0.9925 | 0.9901 | 0.9902 | 0.9937 |
| hddt_page | 0.9918 | 0.9918 | 0.9911 | 0.8498 | 0.8568 | 0.8792 | 0.9581 | 0.9569 | 0.9340 |
| hddt_pendigits | 0.9995 | 0.9996 | 1.0000 | 0.9725 | 0.9775 | 0.9892 | 0.9859 | 0.9869 | 0.9921 |
| hddt_phoneme | 0.9339 | 0.9379 | 0.9502 | 0.7837 | 0.7905 | 0.8149 | 0.8636 | 0.8663 | 0.8675 |
| hddt_PhosS | 0.7183 | 0.7502 | 0.7276 | 0.1753 | 0.1204 | 0.0045 | 0.3432 | 0.2598 | 0.0300 |
| hddt_satimage | 0.9513 | 0.9517 | 0.9620 | 0.6354 | 0.6427 | 0.6916 | 0.8513 | 0.8440 | 0.7929 |
| hddt_segment | 0.9991 | 0.9989 | 0.9999 | 0.9727 | 0.9753 | 0.9912 | 0.9873 | 0.9880 | 0.9932 |
| keel_abalone19 | 0.7427 | 0.7685 | 0.7154 | 0.0535 | 0.0607 | 0.0284 | 0.4729 | 0.3001 | 0.1099 |
| keel_abalone9-18 | 0.7919 | 0.8081 | 0.8070 | 0.3077 | 0.3487 | 0.3769 | 0.6512 | 0.6237 | 0.5716 |
| keel_cleveland-0_vs_4 | 0.9377 | 0.9539 | 0.9572 | 0.5551 | 0.6396 | 0.5681 | 0.7246 | 0.7622 | 0.6754 |
| keel_ecoli-0-1-3-7_vs_2-6 | 0.9278 | 0.9321 | 0.9204 | 0.6382 | 0.6226 | 0.5110 | 0.8256 | 0.7971 | 0.6880 |
| keel_ecoli-0-1-4-6_vs_5 | 0.9654 | 0.9637 | 0.9892 | 0.6883 | 0.7310 | 0.8031 | 0.8336 | 0.8442 | 0.8912 |
| keel_ecoli-0-1-4-7_vs_2-3 | 0.9308 | 0.9333 | 0.9325 | 0.6390 | 0.6721 | 0.6978 | 0.8355 | 0.8236 | 0.8243 |
| keel_ecoli-0-1-4-7_vs_5-6 | 0.9521 | 0.9603 | 0.9668 | 0.7227 | 0.7195 | 0.8016 | 0.8634 | 0.8359 | 0.8635 |
| keel_ecoli-0-1_vs_2-3-5 | 0.9480 | 0.9507 | 0.9495 | 0.6878 | 0.7247 | 0.7508 | 0.8726 | 0.8763 | 0.8544 |
| keel_ecoli-0-1_vs_5 | 0.9579 | 0.9709 | 0.9853 | 0.6755 | 0.7272 | 0.7602 | 0.8287 | 0.8547 | 0.8506 |
| keel_ecoli-0-2-3-4_vs_5 | 0.9690 | 0.9729 | 0.9833 | 0.6741 | 0.7135 | 0.7529 | 0.8717 | 0.8830 | 0.8746 |
| keel_ecoli-0-2-6-7_vs_3-5 | 0.9261 | 0.9285 | 0.9305 | 0.7111 | 0.7537 | 0.7589 | 0.8573 | 0.8636 | 0.8553 |
| keel_ecoli-0-3-4-6_vs_5 | 0.9568 | 0.9667 | 0.9789 | 0.7124 | 0.7425 | 0.7791 | 0.8683 | 0.8700 | 0.8841 |
| keel_ecoli-0-3-4-7_vs_5-6 | 0.9474 | 0.9497 | 0.9622 | 0.7236 | 0.7103 | 0.7983 | 0.8718 | 0.8405 | 0.8706 |
| keel_ecoli-0-3-4_vs_5 | 0.9619 | 0.9671 | 0.9815 | 0.7103 | 0.7475 | 0.7433 | 0.8441 | 0.8495 | 0.8495 |
| keel_ecoli-0-4-6_vs_5 | 0.9677 | 0.9721 | 0.9840 | 0.7450 | 0.7638 | 0.7453 | 0.8824 | 0.8804 | 0.8254 |
| keel_ecoli-0-6-7_vs_3-5 | 0.9213 | 0.9346 | 0.9237 | 0.6796 | 0.7124 | 0.6996 | 0.8402 | 0.8405 | 0.8059 |
| keel_ecoli-0-6-7_vs_5 | 0.9541 | 0.9610 | 0.9612 | 0.7534 | 0.7614 | 0.8079 | 0.9023 | 0.9034 | 0.8953 |
| keel_ecoli-0_vs_1 | 0.9954 | 0.9925 | 0.9909 | 0.9765 | 0.9728 | 0.9691 | 0.9814 | 0.9793 | 0.9771 |
| keel_ecoli1 | 0.9543 | 0.9573 | 0.9456 | 0.7876 | 0.7847 | 0.7650 | 0.8936 | 0.8886 | 0.8538 |
| keel_ecoli2 | 0.9429 | 0.9473 | 0.9639 | 0.7915 | 0.7910 | 0.8128 | 0.8825 | 0.8839 | 0.8694 |
| keel_ecoli3 | 0.9391 | 0.9379 | 0.9209 | 0.6214 | 0.6174 | 0.5567 | 0.8574 | 0.8519 | 0.7162 |
| keel_ecoli4 | 0.9630 | 0.9724 | 0.9855 | 0.6585 | 0.6947 | 0.7911 | 0.8196 | 0.8385 | 0.8860 |
| keel_glass-0-1-2-3_vs_4–5 | 0.9724 | 0.9747 | 0.9767 | 0.8412 | 0.8508 | 0.8363 | 0.9135 | 0.9183 | 0.8867 |
| keel_glass-0-1-4-6_vs_2 | 0.7662 | 0.7510 | 0.7737 | 0.2970 | 0.3398 | 0.2646 | 0.5575 | 0.5606 | 0.4106 |
| keel_glass-0-1-5_vs_2 | 0.7551 | 0.7466 | 0.7489 | 0.3464 | 0.2671 | 0.2688 | 0.6330 | 0.4768 | 0.4377 |
| keel_glass-0-1-6_vs_2 | 0.7335 | 0.7148 | 0.7582 | 0.2650 | 0.2425 | 0.1841 | 0.5334 | 0.4649 | 0.3315 |
| keel_glass-0-1-6_vs_5 | 0.9948 | 0.9938 | 0.9909 | 0.7913 | 0.7882 | 0.6867 | 0.9856 | 0.9756 | 0.8161 |
| keel_glass-0-4_vs_5 | 0.9957 | 0.9964 | 0.9956 | 0.9505 | 0.9505 | 0.8519 | 0.9939 | 0.9939 | 0.9185 |
| keel_glass-0-6_vs_5 | 0.9843 | 0.9837 | 0.9907 | 0.8946 | 0.8946 | 0.7988 | 0.9493 | 0.9493 | 0.8719 |
| keel_glass0 | 0.8593 | 0.8694 | 0.8833 | 0.7216 | 0.7206 | 0.7172 | 0.7976 | 0.7967 | 0.7868 |
| keel_glass1 | 0.8146 | 0.8264 | 0.8592 | 0.6354 | 0.6791 | 0.6997 | 0.7105 | 0.7466 | 0.7602 |
| keel_glass2 | 0.8214 | 0.8020 | 0.7502 | 0.2984 | 0.2500 | 0.2469 | 0.6008 | 0.5043 | 0.3848 |
| keel_glass4 | 0.9117 | 0.9322 | 0.9628 | 0.4748 | 0.5512 | 0.5128 | 0.7349 | 0.7836 | 0.6551 |
| keel_glass5 | 0.9922 | 0.9905 | 0.9864 | 0.7606 | 0.7571 | 0.6602 | 0.9759 | 0.9754 | 0.7761 |
| keel_glass6 | 0.9530 | 0.9602 | 0.9565 | 0.8239 | 0.8423 | 0.8551 | 0.9167 | 0.9235 | 0.9109 |
| keel_haberman | 0.7090 | 0.7130 | 0.6735 | 0.5002 | 0.4943 | 0.3409 | 0.6518 | 0.6454 | 0.4957 |
| keel_iris0 | 1.0000 | 1.0000 | 1.0000 | 0.9813 | 0.9813 | 0.9813 | 0.9816 | 0.9816 | 0.9816 |
| keel_led7digit-0-2-4-5-6- | 0.9577 | 0.9605 | 0.9653 | 0.7541 | 0.7779 | 0.7667 | 0.8925 | 0.8960 | 0.8714 |
| keel_new-thyroid1 | 0.9936 | 0.9949 | 0.9971 | 0.9077 | 0.9124 | 0.9270 | 0.9413 | 0.9494 | 0.9546 |
| keel_new-thyroid2 | 0.9950 | 0.9953 | 0.9983 | 0.8960 | 0.8993 | 0.9455 | 0.9482 | 0.9420 | 0.9637 |
| keel_page-blocks-1-3_vs_4 | 0.9997 | 0.9995 | 0.9998 | 0.9284 | 0.9271 | 0.9610 | 0.9700 | 0.9698 | 0.9837 |
| keel_page-blocks0 | 0.9913 | 0.9912 | 0.9904 | 0.8455 | 0.8530 | 0.8692 | 0.9519 | 0.9537 | 0.9302 |
| keel_pima | 0.8185 | 0.8214 | 0.8018 | 0.6654 | 0.6721 | 0.6225 | 0.7387 | 0.7451 | 0.7025 |
| keel_segment0 | 0.9983 | 0.9986 | 0.9999 | 0.9683 | 0.9700 | 0.9881 | 0.9847 | 0.9847 | 0.9919 |
| keel_shuttle-c0-vs-c4 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| keel_shuttle-c2-vs-c4 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| keel_vehicle0 | 0.9885 | 0.9896 | 0.9958 | 0.8803 | 0.8803 | 0.9335 | 0.9391 | 0.9394 | 0.9588 |
| keel_vehicle1 | 0.8452 | 0.8510 | 0.8511 | 0.6243 | 0.6197 | 0.5608 | 0.7643 | 0.7541 | 0.6830 |
| keel_vehicle2 | 0.9936 | 0.9945 | 0.9981 | 0.9329 | 0.9404 | 0.9665 | 0.9653 | 0.9665 | 0.9772 |
| keel_vehicle3 | 0.8478 | 0.8475 | 0.8458 | 0.6162 | 0.6140 | 0.5442 | 0.7626 | 0.7524 | 0.6647 |
| keel_vowel0 | 0.9965 | 0.9965 | 0.9997 | 0.8733 | 0.8787 | 0.9697 | 0.9623 | 0.9681 | 0.9817 |
| keel_wisconsin | 0.9921 | 0.9924 | 0.9931 | 0.9521 | 0.9501 | 0.9526 | 0.9661 | 0.9635 | 0.9646 |
| keel_yeast-0-2-5-6_vs_3-7 | 0.8449 | 0.8533 | 0.8427 | 0.5531 | 0.5957 | 0.5896 | 0.7745 | 0.7731 | 0.7195 |
| keel_yeast-0-2-5-7-9_vs_3 | 0.9483 | 0.9444 | 0.9436 | 0.7434 | 0.7775 | 0.8049 | 0.8956 | 0.9022 | 0.8799 |

(*continued on next page*)

**Table 7** (*continued*)

| Dataset | AUC | | | F-Measure | | | Geometric mean | | |
|---|---|---|---|---|---|---|---|---|---|
| | E-RB | BAG-RB | RB-B | E-RB | BAG-RB | RB-B | E-RB | BAG-RB | RB-B |
| keel_yeast-0-3-5-9_vs_7-8 | 0.7573 | 0.7638 | 0.7565 | 0.3717 | 0.3869 | 0.3635 | 0.6575 | 0.6694 | 0.5319 |
| keel_yeast-0-5-6-7-9_vs_4 | 0.8931 | 0.8963 | 0.8795 | 0.4982 | 0.5246 | 0.4876 | 0.7714 | 0.7433 | 0.6452 |
| keel_yeast-1-2-8-9_vs_7 | 0.7373 | 0.7592 | 0.7477 | 0.1868 | 0.1785 | 0.2663 | 0.6400 | 0.4947 | 0.4452 |
| keel_yeast-1-4-5-8_vs_7 | 0.6477 | 0.6617 | 0.6655 | 0.1644 | 0.1565 | 0.1135 | 0.5561 | 0.4222 | 0.2365 |
| keel_yeast-1_vs_7 | 0.8096 | 0.8184 | 0.8059 | 0.3310 | 0.3350 | 0.3824 | 0.6851 | 0.6455 | 0.5663 |
| keel_yeast-2_vs_4 | 0.9799 | 0.9799 | 0.9705 | 0.7149 | 0.7292 | 0.7514 | 0.9104 | 0.9070 | 0.8547 |
| keel_yeast-2_vs_8 | 0.8167 | 0.8204 | 0.8216 | 0.4098 | 0.5572 | 0.5942 | 0.7089 | 0.7286 | 0.7238 |
| keel_yeast1 | 0.7949 | 0.7992 | 0.7768 | 0.5920 | 0.6027 | 0.5309 | 0.7107 | 0.7212 | 0.6461 |
| keel_yeast3 | 0.9741 | 0.9745 | 0.9641 | 0.7788 | 0.7811 | 0.7649 | 0.9320 | 0.9294 | 0.8603 |
| keel_yeast4 | 0.9335 | 0.9381 | 0.9148 | 0.3336 | 0.3884 | 0.3790 | 0.8075 | 0.8055 | 0.5807 |
| keel_yeast5 | 0.9897 | 0.9901 | 0.9766 | 0.7311 | 0.7269 | 0.6899 | 0.9461 | 0.9391 | 0.8438 |
| keel_yeast6 | 0.9137 | 0.9168 | 0.8965 | 0.3685 | 0.4575 | 0.4997 | 0.7823 | 0.7869 | 0.6878 |

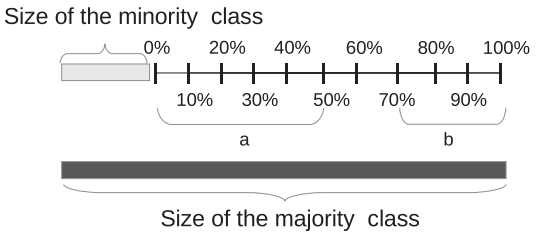



**Fig. 6.** The figure shows how to interpret the parameters used for SMOTE and Undersampling. These parameters can be thought of as a percentage of the difference of class sizes. For SMOTE, in this case, a value of 50% (*a* in the figure) indicates that the number of artificial instances to be created are the 50% of the number needed to match the size of the majority class. For Undersampling a value of 70% (*b* in the figure) indicates that the number of removed instances in the majority class will be 30% of the size of the difference.

for *RB-Boost*. In this case, despite the *p*-value given by the Iman and Davenport test, the post hoc Hochberg test found no significant differences between the methods at $\alpha = 0.05$; the *p*-value of Hochberg between the first ranking method and the last one is 0.05954. The method which obtains the best rank according to accuracy is Multiboost with resampling, but we emphasize that accuracy is not the best measure to evaluate classification methods in imbalanced dataset. And finally, the method that obtains the best average ranking considering all measures is one of the proposed methods: Random Balance Boost (*RB-B*).

### 6.1. Fusion rules

The outputs of the classifiers in an ensemble can be combined in several ways [50]. For Ensemble-RB and Bagging-RB, the outputs are combined using the simple average of probabilities. For RB-Boost, the outputs are combined using a weighted average (line 11 in Fig. 2), because it is the method used in AdaBoost.M2 and its variants for imbalance (RUSBoost, SMOTEBoost).

This section considers other combination methods for Ensemble-RB and Bagging-RB: majority voting and product of probabilities. Tables 14 and 15 show the average ranks for the considered fusion rules. Iman and Davenport Test discards the hypothesis of equivalence between the algorithms in all cases. Ensemble-RB and Bagging-RB show the same behavior: for AUC the order of fusion rules is average, product and majority voting, while for F-Measure the order is majority voting, average and product. When comparing the best method with the remaining methods, the adjusted *p*-values for Hochberg's procedure are small (<0.015). Hence, which fusion rule is used gives significant differences.

**Table 8**
Average ranks (AUC).

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) Data-processing family* | | |
| E-RB | 2.2674 | |
| ERUSR | 3.8256 | 0.0021 |
| EPart | 4.1337 | 4.4869e−004 |
| ERUS | 4.4767 | 3.7599e−005 |
| EopB | 5.5930 | 1.9442e−010 |
| ESM200 | 6.7442 | 4.3307e−018 |
| ESM500 | 7.0291 | 2.8543e−020 |
| ESM100 | 7.1861 | 1.6549e−021 |
| EopU | 7.6744 | 9.0344e−026 |
| ESM | 8.2674 | 1.6612e−031 |
| EopS | 8.8023 | 3.4537e−037 |
| *(b) Bagging family* | | |
| BAG-RB | 3.0930 | |
| BAGopB | 5.9302 | 1.7768e−006 |
| BAGSM500 | 6.0465 | 1.3180e−006 |
| BAGSM200 | 6.1456 | 8.2646e−007 |
| BAGSM100 | 6.3081 | 2.4709e−007 |
| BAGRUS | 6.3256 | 2.4709e−007 |
| BAGopS | 6.4826 | 6.8877e−008 |
| BAGSM | 6.5058 | 6.3800e−008 |
| BAGopU | 6.6977 | 1.0265e−008 |
| SMBAG | 8.3663 | 6.0674e−018 |
| BAG | 8.5233 | 6.0521e−019 |
| RbB:IC+BAGSM | 10.1919 | 6.8898e−032 |
| RbB:IC+BAGRUS | 10.3837 | 1.4618e−033 |
| *(c) Boosting family* | | |
| RB-B | 2.9884 | |
| RUSB | 3.6628 | 0.10634 |
| SB200 | 4.4419 | 0.00100 |
| SB500 | 4.5116 | 7.9490e−004 |
| SB100 | 4.8139 | 4.9417e−005 |
| MultiS | 4.9128 | 2.0338e−005 |
| AdaM1S | 6.0407 | 1.6196e−012 |
| MultiW | 6.1977 | 1.0754e−013 |
| AdaM1W | 7.4302 | 1.6253e−025 |

### 6.2. Base classifiers

Decision trees are usually used as base classifiers, since they are simple and fast to compute, and they are unstable (small variations in the training set can result in different trees and different predictions), which contributes to the diversity of the ensemble. This section considers the performance of the proposed ensemble methods with other two base classifiers: nearest neighbour (1-NN) and SVM with Gaussian kernel. Due to the high computational cost of SVM classifiers, the size of all ensembles used in the comparison of this section was set to 50.

**Table 9**
Average ranks (F-Measure).

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) Data-processing family* | | |
| E-RB | 4.0639 | |
| ESM200 | 4.3023 | 0.6374 |
| ESM500 | 4.3954 | 0.6374 |
| ESM100 | 4.7674 | 0.4928 |
| EopB | 5.5116 | 0.0168 |
| ESM | 5.7326 | 0.0049 |
| EopS | 5.9070 | 0.0016 |
| EopU | 6.7790 | 5.5677e−007 |
| ERUS | 7.9419 | 1.4066e−013 |
| EPart | 7.9535 | 1.3225e−013 |
| ERUSR | 8.6454 | 1.3278e−018 |
| *(b) Bagging family* | | |
| BAGSM500 | 5.3081 | |
| BAG-RB | 5.5930 | 0.6315 |
| BAGSM200 | 5.6686 | 0.6315 |
| BAGSM | 5.9651 | 0.6315 |
| BAGopS | 6.3023 | 0.3765 |
| BAGSM100 | 6.4302 | 0.2942 |
| RbB:IC+BAGSM | 6.7791 | 0.0796 |
| RbB:IC+BAGRUS | 7.1977 | 0.0103 |
| SMBAG | 7.2035 | 0.0103 |
| BAGopB | 7.2616 | 0.0090 |
| BAGopU | 8.5639 | 4.2025e−007 |
| BAG | 8.7849 | 5.2748e−008 |
| BAGRUS | 9.9419 | 7.2984e−014 |
| *(c) Boosting family* | | |
| RB-B | 3.3372 | |
| RUSB | 4.5639 | 0.00331 |
| AdaM1S | 4.6337 | 0.00331 |
| SB500 | 4.7326 | 0.00250 |
| MultiS | 4.9942 | 2.9049e−004 |
| SB200 | 5.2267 | 3.0287e−005 |
| SB100 | 5.6919 | 1.0318e−007 |
| AdaM1W | 5.7733 | 3.8116e−008 |
| MultiW | 6.0465 | 6.9931e−010 |

**Table 10**
Average ranks (geometric mean).

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) Data-processing family* | | |
| E-RB | 3.3779 | |
| ERUS | 3.7500 | 4.6192e−001 |
| EPart | 3.9767 | 4.6192e−001 |
| ERUSR | 4.3488 | 1.6470e−001 |
| EopB | 5.2442 | 8.9737e−004 |
| ESM | 6.4070 | 1.0563e−008 |
| ESM500 | 6.5116 | 3.4791e−009 |
| ESM200 | 7.8140 | 1.2430e−017 |
| EopU | 7.8372 | 9.4328e−018 |
| EopS | 8.2093 | 1.1410e−020 |
| ESM100 | 8.5233 | 2.6149e−023 |
| *(b) Bagging family* | | |
| BAG-RB | 3.2326 | |
| BAGRUS | 4.1686 | 1.1500e−001 |
| SMBAG | 4.7267 | 2.3746e−002 |
| BAGopB | 5.3837 | 8.7661e−004 |
| BAGSM | 5.8430 | 4.4210e−005 |
| BAGSM500 | 6.0640 | 9.3268e−006 |
| BAGSM200 | 7.6570 | 5.6084e−013 |
| BAGopU | 7.6686 | 5.6084e−013 |
| BAGopS | 7.7791 | 1.5418e−013 |
| RbB:IC+BAGRUS | 8.3081 | 1.1445e−016 |
| BAGSM100 | 8.9942 | 2.9739e−021 |
| RbB:IC+BAGSM | 9.3547 | 7.1062e−024 |
| BAG | 11.8198 | 2.6359e−046 |
| *(c) Boosting family* | | |
| RUSB | 2.0872 | |
| SB500 | 3.5233 | 5.8489e−004 |
| RB-B | 3.6686 | 3.0550e−004 |
| SB200 | 4.7384 | 6.5421e−010 |
| AdaM1S | 5.4709 | 2.1606e−015 |
| SB100 | 5.5756 | 3.3363e−016 |
| MultiS | 5.9477 | 1.4294e−019 |
| AdaM1W | 6.5872 | 3.1635e−026 |
| MultiW | 7.4012 | 3.4831e−036 |

Tables 16–18 show the average ranks for, respectively, Ensemble-RB, Bagging-RB and RB-Boost. These tables show, for AUC and F-Measure, the average ranks of the three considered base classifiers with the corresponding ensemble methods. Decision trees work better in these ensembles than the other two considered alternatives: in all the ranks, decision trees have the top position. The differences are larger for AUC than for F-Measure.

### 6.3. Ensemble size

In all the previous experiments the ensemble size was 100. This section considers the effect of the ensemble size in the performance. Fig. 8 shows the performance as a function of the ensemble size. The ensemble sizes vary from 5 to 100 in steps of size 5. The top two plots show the average value of the performance measure (AUC or F-Measure) across all the data sets, for Ensemble-RB, Bagging-RB and RB-Boost. As usual with ensembles, the performance improves with size, but the improvements are smaller as

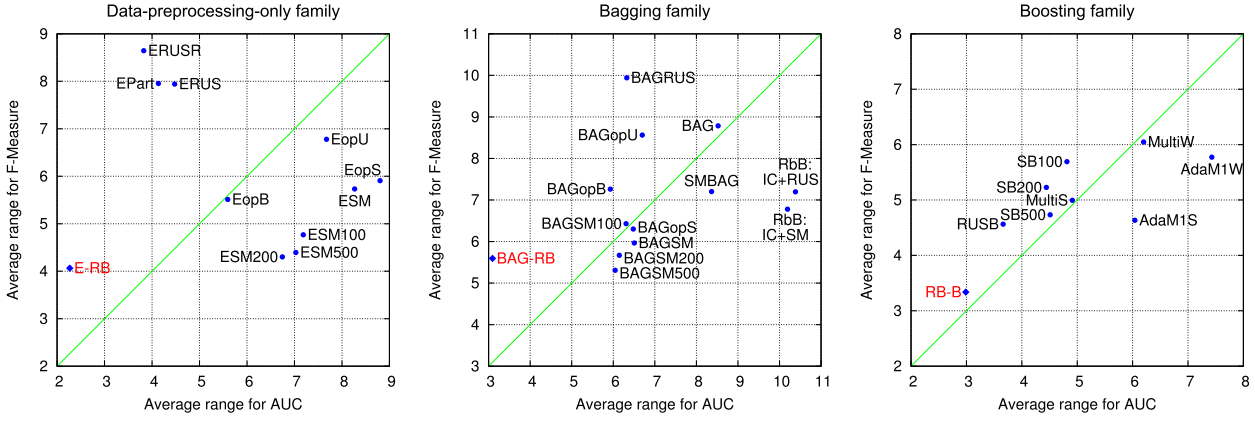



**Fig. 7.** Average ranks for the ensemble methods, according to te AUC and F-Measure.

**Table 11**
Average ranks for the considered ensemble methods, obtained from the accuracies.

| Algorithm | Rank |
|---|---|
| *Data-level family* | |
| ESM100 | 2.4826 |
| ESM200 | 3.4593 |
| EopS | 4.2326 |
| ESM500 | 4.7500 |
| E-RB | 5.0407 |
| ESM | 5.2267 |
| EopB | 6.0000 |
| EopU | 6.1570 |
| EPart | 9.1512 |
| RUS | 9.2209 |
| ERUSR | 10.2791 |
| *Bagging family* | |
| BAG | 3.5233 |
| BAGSM100 | 3.9942 |
| RbB:IC+BAGSM | 4.6221 |
| BAGSM200 | 5.1337 |
| BAGopS | 5.3198 |
| RbB:IC+BAGRUS | 6.3314 |
| BAGSM500 | 6.6337 |
| BAGSM | 6.9128 |
| BAG-RB | 8.3721 |
| BAGopU | 8.7151 |
| SMBAG | 9.4884 |
| BAGopB | 9.5116 |
| BAGRUS | 12.4419 |
| *(c) Boosting family* | |
| MultiS | 3.3314 |
| AdaS | 3.4826 |
| MultiW | 3.8430 |
| RB-B | 4.1337 |
| AdaW | 4.2093 |
| SB100 | 6.0174 |
| SB200 | 6.3023 |
| RUSB | 6.6337 |
| SB500 | 7.0465 |

**Table 12**
Average ranks (combined).

| Algorithm | Average rank |
|---|---|
| *(a) Data-processing family* | |
| E-RB | 3.6948 |
| ESM200 | 5.5828 |
| EopB | 5.5858 |
| ESM500 | 5.6744 |
| ESM100 | 5.7427 |
| EPart | 6.3009 |
| RUS | 6.3503 |
| ESM | 6.4113 |
| ERUSR | 6.7645 |
| EopS | 6.7820 |
| EopU | 7.1105 |
| *(b) Bagging family* | |
| BAG-RB | 5.0727 |
| BAGSM500 | 6.0131 |
| BAGSM200 | 6.1512 |
| BAGSM | 6.3067 |
| BAGSM100 | 6.4346 |
| BAGopS | 6.4695 |
| BAGopB | 7.0218 |
| SMBAG | 7.4462 |
| RbB:IC+BAGSM | 7.7384 |
| BAGopU | 7.9113 |
| RbB:IC+BAGRUS | 8.0552 |
| BAG | 8.1599 |
| BAGRUS | 8.2195 |
| *(c) Boosting family* | |
| RB-B | 3.5320 |
| RUSB | 4.2369 |
| MultiS | 4.7965 |
| AdaS | 4.9070 |
| SB500 | 4.9535 |
| SB200 | 5.1773 |
| SB100 | 5.5247 |
| MultiW | 5.8721 |
| AdaW | 6.0000 |

**Table 13**
Average ranks (best algorithms).

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) AUC* | | |
| BAG-RB | 1.76163 | |
| RB-B | 1.82558 | 0.67494 |
| E-RB | 2.41279 | 0.00004 |
| *(b) F-Measure* | | |
| RB-B | 1.88372 | |
| BAGSM500 | 1.90116 | 0.90894 |
| E-RB | 2.21512 | 0.05954 |
| *(c) G-Mean* | | |
| RUSB | 1.9651 | |
| E-RB | 2.0058 | 7.8957e−001 |
| BAG-RB | 2.0291 | 7.8957e−001 |
| Algorithm | Average rank | |
| *(d) Accuracy* | | |
| MultiS | 1.6395 | |
| BAG | 1.7209 | |
| ESM100 | 2.6395 | |
| *(e) Combined* | | |
| RB-B | 1.8488 | |
| BAG-RB | 1.8532 | |
| E-RB | 2.2980 | |

**Table 14**
Average ranks for ensemble-RB fusion rules.

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) AUC* | | |
| Average | 1.08721 | |
| Product | 1.95349 | 1.34e−8 |
| Majority voting | 2.9593 | 2.43e−34 |
| *(b) F-Measure* | | |
| Majority voting | 1.49419 | |
| Average | 1.94186 | 3.33e−003 |
| Product | 2.56395 | 4.60e−012 |

the size grows. The bottom two plots shows the performance but using average ranks instead of the mean across all the data sets. For each method, 20 sizes are considered $(5, 10, 15, \ldots, 100)$, and the average ranks are computed for them. The values are in the interval [1,20], and smaller values represent better performance. The average ranks are better as the ensemble size increases.

Fig. 9 shows six plots, each one compares a pair of methods across the considered ensemble sizes. Each plot shows, as a function of the ensemble size, the percentage of data sets with the best performance in terms of AUC (left plots) or F-Measure (right plots). The selected pairs are the two methods with best average ranks in each family (Tables 8 and 9).

**Table 15**
Average ranks for bagging-RB fusion rules.

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) AUC* | | |
| Average | 1.11047 | |
| Product | 1.91860 | 1.16e−007 |
| Majority Voting | 2.97093 | 6.23e−034 |
| *(b) F-Measure* | | |
| Majority Voting | 1.56395 | |
| Average | 1.94186 | 0.01321 |
| Product | 2.49419 | 2.12e−009 |

**Table 16**
Average ranks of base classifiers for ensemble-RB.

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) AUC* | | |
| J48 | 1.74419 | |
| 1-NN | 2.04070 | 0.0519 |
| SVM | 2.21512 | 0.0040 |
| *(b) F-Measure* | | |
| J48 | 1.63953 | |
| 1-NN | 1.88953 | 0.1011 |
| SVM | 2.47093 | 9.97e−008 |

**Table 17**
Average ranks of base classifiers for Bagging-RB.

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) AUC* | | |
| J48 | 1.72674 | |
| 1-NN | 2.05233 | 0.0328 |
| SVM | 2.22093 | 0.0024 |
| *(b) F-Measure* | | |
| J48 | 1.72674 | |
| 1-NN | 1.91279 | 0.2225 |
| SVM | 2.36047 | 6.49e−005 |

**Table 18**
Average ranks of base classifiers for RB-boost.

| Algorithm | Average rank | *p*-Hochberg |
|---|---|---|
| *(a) AUC* | | |
| J48 | 1.27326 | |
| SVM | 2.17442 | 3.44e-009 |
| 1-NN | 2.55233 | 9.94e-017 |
| *(b) F-Measure* | | |
| J48 | 1.89535 | |
| 1-NN | 2.01744 | 0.4234 |
| SVM | 2.08721 | 0.4167 |

According to the AUC, the methods based on RB have a percentage of victories around 70%. In the left center plot, when comparing Bagging-RB with BAGopB (Bagging with the amount of SMOTE and Undersampling selected by cross validation), the initial percentage is smaller but it increases to greater values with the ensemble size.

For the F-Measure (right plots), Ensemble-RB and Bagging-RB are not better than the corresponding pairs (ESM200 and BAG500), this was expected as in Tables 8 and 9 the differences for the considered pairs were not significant. When comparing RB-Boost with RUSBoost the difference is greater, although it decreases with the ensemble size.

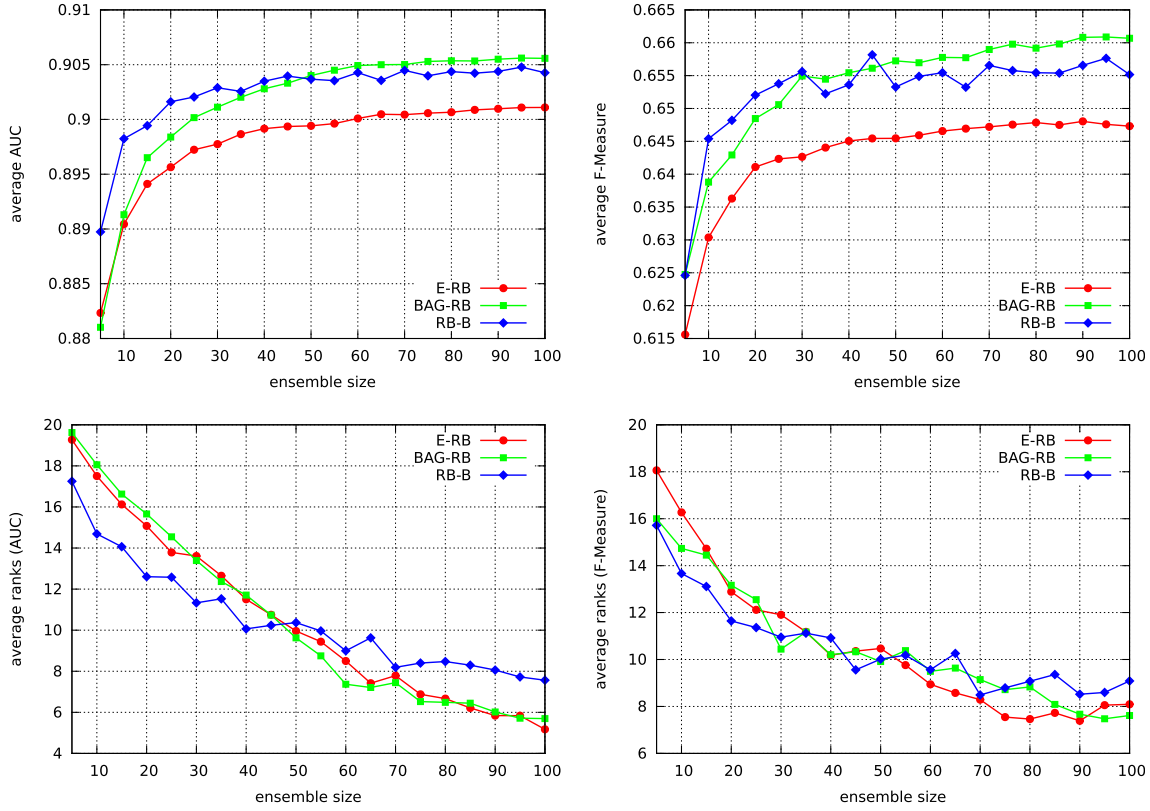



**Fig. 8.** Performance measures as a function of the ensemble size.

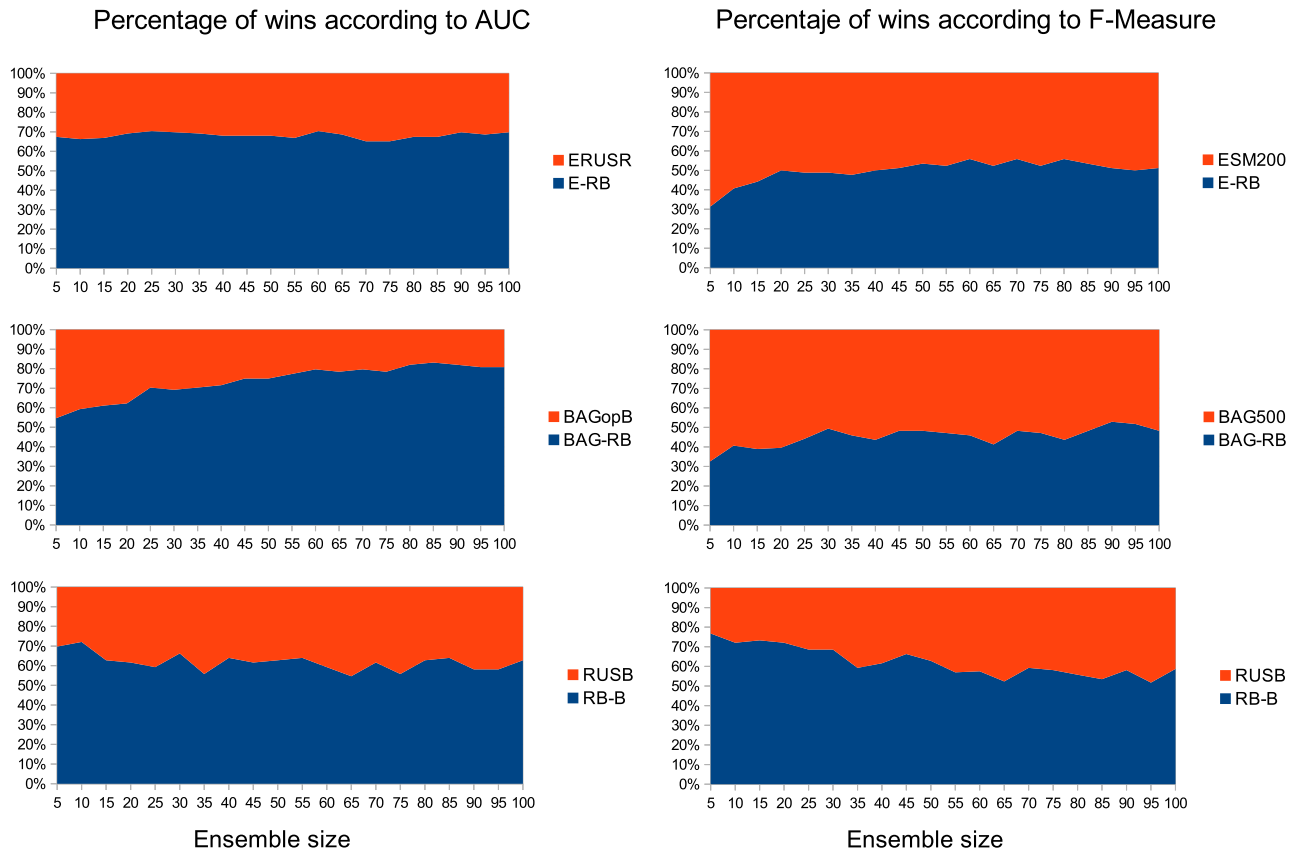



**Fig. 9.** Comparison of methods as a function of the ensemble size.

## 7. Conclusion

We propose a new preprocessing technique adequate to balance datasets within ensemble methods: Random Balance, based on the idea of varying randomly the proportions of the classes, and applied it to design a new ensemble method: RB-Boost. In addition to boosting the AUC, this intuitive heuristic bypasses the need to tune the sensitive class proportion parameter, common to most methods for imbalanced classification.

Despite their simplicity, the two proposed methods have proved competitive when compared with other state-of-the-art ensembles, including those specifically devised for imbalanced data classification.

There are several future research lines:

- Study the performance of Random Balance in presence of several data intrinsic characteristics which have been proven to have a strong influence on imbalanced classification [51,52]. Some of these problems are overlapping [53], noisy examples [54], small disjuncts [55] or borderline examples [56]. On several occasions these problems have been addressed with preprocessing techniques, these techniques could be combined using the same strategy that Random Balance uses to combine SMOTE and undersampling resulting in new methods. For example, the resampling strategy CBO [57] has been used successfully with small disjuncts; cleaning techniques such as ENN [58] or CNN [59] have been used with noisy datasets; and variants of SMOTE, such as, Safe-Level-SMOTE [15] or SPIDER [60] with borderline examples.
- Extend the ideas in this article to multiple-class unbalanced problems.
- Test other combinations of classifiers beyond the average or the weighted average.

## Acknowledgments

This work was partially supported by the Project TIN2011-24046 of the Spanish Ministry of Economy and Competitiveness. We wish to thank the developers of Weka [40], the KEEL Experimental Analysis Framework [38] and the donors of the different data sets.

## Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.knosys.2015.04.022.

## References

[1] N. Chawla, N. Japkowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, ACM SIGKDD Explor. Newslett. 6 (1) (2004) 1–6.
[2] N. García-Pedrajas, J. Pérez-Rodríguez, M.D. García-Pedrajas, D. Ortiz-Boyer, C. Fyfe, Class imbalance methods for translation initiation site recognition in dna sequences, Knowl.-Based Syst. 25 (1) (2012) 22–34.
[3] R. Batuwita, V. Palade, microPred: Effective classification of pre-miRNAs for human miRNA gene prediction, Bioinformatics 25 (8) (2009) 989–995.
[4] T.W. Liao, Classification of weld flaws with imbalanced class data, Expert Syst. Appl. 35 (3) (2008) 1041–1052, http://dx.doi.org/10.1016/j.eswa.2007.08.044.
[5] D. Anil Kumar, V. Ravi, Predicting credit card customer churn in banks using data mining, Int. J. Data Anal. Techn. Strat. 1 (1) (2008) 4–28.
[6] C. Phua, D. Alahakoon, V. Lee, Minority report in fraud detection: classification of skewed data, ACM SIGKDD Explor. Newslett. 6 (1) (2004) 50–59.
[7] S. Visa, A. Ralescu, Issues in mining imbalanced data sets – a review paper, in: Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference, 2005, pp. 67–73.
[8] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern., Part C: Appl. Rev. 42 (4) (2012) 463–484, http://dx.doi.org/10.1109/TSMCC.2011.2161285.

[9] D.A. Cieslak, N.V. Chawla, Learning decision trees for unbalanced data, in: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases – Part I, ECML PKDD '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 241–256, http://dx.doi.org/10.1007/978-3-540-87479-9_34.
[10] W. Liu, S. Chawla, D.A. Cieslak, N.V. Chawla, A robust decision tree algorithm for imbalanced data sets, in: Proceedings of the SIAM International Conference on Data Mining, SDM 2010, 2010, pp. 766–777.
[11] K. Veropoulos, C. Campbell, N. Cristianini, Controlling the sensitivity of support vector machines, in: Proceedings of the International Joint Conference on AI, 1999, pp. 55–60.
[12] G. Batista, R. Prati, M. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newslett. 6 (1) (2004) 20–29.
[13] N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[14] H. Han, W. Wang, B. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, in: 2005 International Conference on Intelligent Computing (ICIC05), Lecture Notes on Computer Science, vol. 3644, Springer-Verlag, 2005, pp. 878–887.
[15] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-smote: safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD09), Lecture Notes on Computer Science, vol. 5476, Springer-Verlag, 2009, pp. 475–482.
[16] R. Barandela, R. Valdovinos, J. Sánchez, New applications of ensembles of classifiers, Pattern Anal. Appl. 6 (3) (2003) 245–256.
[17] C. Ling, V. Sheng, Q. Yang, Test strategies for cost-sensitive decision trees, IEEE Trans. Knowl. Data Eng. 18 (8) (2006) 1055–1067.
[18] M. Kukar, I. Kononenko, Cost-sensitive learning with neural networks, in: Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98), Citeseer, 1998, pp. 445–449.
[19] W. Fan, S.J. Stolfo, J. Zhang, P.K. Chan, AdaCost: misclassification cost-sensitive boosting, in: Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, pp. 97–105.
[20] M. Joshi, V. Kumar, R. Agarwal, Evaluating boosting algorithms to classify rare classes: comparison and improvements, in: Proceedings IEEE International Conference on Data Mining, 2001, ICDM 2001, IEEE, 2001, pp. 257–264.
[21] Y. Sun, M. Kamel, A. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recogn. 40 (2007) 3358–3378.
[22] S. Wang, X. Yao, Diversity analysis on imbalanced data sets by using ensemble models, in: IEEE Symposium Series on Computational Intelligence and Data Mining (IEEE CIDM 2009), 2009, pp. 324–331.
[23] N. Chawla, A. Lazarevic, L. Hall, K. Bowyer, Smoteboost: improving prediction of the minority class in boosting, in: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2003), 2003, pp. 107–119.
[24] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, A. Napolitano, Rusboost: a hybrid approach to alleviating class imbalance, IEEE Trans. Syst. Man Cybern., Part A: Syst. Hum. 40 (1) (2010) 185–197.
[25] S.B. Kotsiantis, P.E. Pintelas, Mixture of expert agents for handling imbalanced data sets, Ann. Math. Comput. Teleinform. 1 (1) (2003) 46–55.
[26] T. Fawcett, An introduction to ROC analysis, Pattern Recogn. Lett. 27 (8) (2006) 861–874.
[27] C. Van Rijsbergen, Information Retrieval, Butterworth, 1979.
[28] M. Kubat, Matwin, Addressing the curse of imbalanced training sets: one-sided selection, in: Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 179–186.
[29] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[30] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3–6, 1996, 1996, pp. 148–156.
[31] M. Molinara, M. Ricamato, F. Tortorella, Facing imbalanced classes through aggregation of classifiers, in: 14th International Conference on Image Analysis and Processing, 2007, ICIAP 2007, IEEE, 2007, pp. 43–48.
[32] P. Soda, A multi-objective optimisation approach for class imbalance learning, Pattern Recogn. 44 (8) (2011) 1801–1810.
[33] S. Wang, X. Yao, Relationships between diversity of classification ensembles and single-class performance measures, IEEE Trans. Knowl. Data Eng. 25 (1) (2013) 206–219.
[34] D.D. Margineantu, T.G. Dietterich, Pruning adaptive boosting, in: Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), 1997, pp. 211–218.
[35] L. Kuncheva, A bound on kappa-error diagrams for analysis of classifier ensembles, IEEE Trans. Knowl. Data Eng. (99) (2011) 1, http://dx.doi.org/10.1109/TKDE.2011.234.
[36] J.L. Fleiss, Statistical Methods for Rates and Proportions, Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, John Wiley & Sons, 1981.
[37] D.A. Cieslak, T.R. Hoens, N.V. Chawla, W.P. Kegelmeyer, Hellinger distance decision trees are robust and skew-insensitive, Data Min. Knowl. Discov. 24 (1) (2012) 136–158, http://dx.doi.org/10.1007/s10618-011-0222-1.
[38] J. Alcala-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository and integration of algorithms and experimental analysis framework, J. Multple-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287.
[39] A. Frank, A. Asuncion, UCI machine learning repository, 2010. <http://archive.ics.uci.edu/ml>.
[40] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, SIGKDD Explor. Newsl. 11 (1) (2009) 10–18, http://dx.doi.org/10.1145/1656274.1656278.
[41] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kauffman, 1993.
[42] F. Provost, P. Domingos, Tree induction for probability-based ranking, Mach. Learn. 52 (3) (2003) 199–215.
[43] T. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput. 10 (7) (1998) 1895–1923.
[44] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[45] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[46] R. Iman, J. Davenport, Approximations of the critical region of the fbietkan statistic, Commun. Stat.-Theory Methods 9 (6) (1980) 571–595.
[47] Y. Hochberg, A sharper Bonferroni procedure for multiple tests of significance, Biometrika 75 (1988) 800–803.
[48] O. Dunn, Multiple comparisons among means, J. Am. Stat. Assoc. 56 (1961) 52–64.
[49] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, J. Heuristics 15 (6) (2009) 617–644.
[50] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley Interscience, 2004.
[51] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, Inform. Sci. 250 (0) (2013) 113–141, http://dx.doi.org/10.1016/j.ins.2013.07.007. <http://www.sciencedirect.com/science/article/pii/S0020025513005124>.
[52] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009), http://dx.doi.org/10.1109/TKDE.2008.239.
[53] J. Stefanowski, Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data, in: Emerging Paradigms in Machine Learning, Springer, 2013, pp. 277–306.
[54] C.E. Brodley, M.A. Friedl, Identifying mislabeled training data, J. Artif. Intell. Res. 11 (1999) 131–167.
[55] G.M. Weiss, The impact of small disjuncts on classifier learning, in: Data Mining, Springer, 2010, pp. 193–226.
[56] K. Napierała, J. Stefanowski, S. Wilk, Learning from imbalanced data in presence of noisy and borderline examples, in: Rough Sets and Current Trends in Computing, Springer, 2010, pp. 158–167.
[57] T. Jo, N. Japkowicz, Class imbalances versus small disjuncts, ACM SIGKDD Explor. Newslett. 6 (1) (2004) 40–49.
[58] D. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421.
[59] K. Gowda, G. Krishna, The condensed nearest neighbor rule using the concept of mutual nearest neighborhood (corresp.), IEEE Trans. Inform. Theory 25 (4) (1979) 488–490.
[60] J. Stefanowski, S. Wilk, Selective pre-processing of imbalanced data for improving classification performance, in: Data Warehousing and Knowledge Discovery, Springer, 2008, pp. 283–292.