Expert Systems with Applications 39 (2012) 6585–6608

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics

Victoria López [a,*], Alberto Fernández [b], Jose G. Moreno-Torres [a], Francisco Herrera [a]

[a] *Dept. of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071 Granada, Spain*
[b] *Dept. of Computer Science, University of Jaén, 23071 Jaén, Spain*

ARTICLE INFO

Keywords:
Classification
Imbalanced datasets
Preprocessing
Cost-sensitive learning
Class overlap
Dataset shift

ABSTRACT

Class imbalance is among the most persistent complications which may confront the traditional supervised learning task in real-world applications. The problem occurs, in the binary case, when the number of instances in one class significantly outnumbers the number of instances in the other class. This situation is a handicap when trying to identify the minority class, as the learning algorithms are not usually adapted to such characteristics.

The approaches to deal with the problem of imbalanced datasets fall into two major categories: data sampling and algorithmic modification. Cost-sensitive learning solutions incorporating both the data and algorithm level approaches assume higher misclassification costs with samples in the minority class and seek to minimize high cost errors. Nevertheless, there is not a full exhaustive comparison between those models which can help us to determine the most appropriate one under different scenarios.

The main objective of this work is to analyze the performance of data level proposals against algorithm level proposals focusing in cost-sensitive models and versus a hybrid procedure that combines those two approaches. We will show, by means of a statistical comparative analysis, that we cannot highlight an unique approach among the rest. This will lead to a discussion about the data intrinsic characteristics of the imbalanced classification problem which will help to follow new paths that can lead to the improvement of current models mainly focusing on class overlap and dataset shift in imbalanced classification.

1. Introduction

One major problem facing data mining is the class imbalance problem (He & Garcia, 2009; Sun, Wong, & Kamel, 2009). It appears in many applications, and is defined as the case where there exists a significant difference between the class prior rates, that is, the probability a particular example belongs to a particular class. The class imbalance is dominant in a high number of real problems including, but not limited to, telecommunications, WWW, finances, ecology, biology, medicine and so on. It must also be stressed that the positive or minority class is usually the one that has the highest interest from the learning point of view and it also implies a great cost when it is not well classified (Elkan, 2001).

A wide number of approaches have been proposed to the imbalanced learning problem that fall largely into two major categories. The first one is data sampling in which the training instances are modified in such a way as to produce a balanced data distribution that allow classifiers to perform in a similar manner to standard classification (Batista, Prati, & Monard, 2004; Chawla, Bowyer, Hall, & Kegelmeyer, 2002). The second one is through algorithmic modification to make base learning methods more attuned to class imbalance issues (Zadrozny & Elkan, 2001). Cost-sensitive learning solutions incorporating both the data and algorithm level approaches assume higher misclassification costs with samples in the rare class and seek to minimize the high cost errors (Ling, Yang, Wang, & Zhang, 2004; Zadrozny, Langford, & Abe, 2003).

Works in imbalanced classification usually focus on the development of new algorithms along one of the categories previously mentioned. However, there is not a study that exhaustively compares solutions from one category to another making difficult the selection of one kind of algorithm when classifying. The aim of this paper is to develop a thorough experimental study to analyze the possible differences between preprocessing techniques and cost-sensitive learning for addressing classification with imbalanced data. In addition, we also present in the comparison a hybrid procedure that combines those two approaches to check whether there is a synergy between them.

In order to analyze the oversampling and undersampling methodologies against cost-sensitive learning approaches, we will use

[*] Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
E-mail addresses: vlopez@decsai.ugr.es (V. López), alberto.fernandez@ujaen.es (A. Fernández), jose.garcia.mt@decsai.ugr.es (J.G. Moreno-Torres), herrera@decsai.ugr.es (F. Herrera).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.12.043

the "Synthetic Minority Over-sampling Technique" (SMOTE) (Chawla et al., 2002) and its variant with the Wilson's Edited Nearest Neighbor (ENN) rule (Wilson, 1972) as they have shown to obtain a very robust behaviour among many different situations (Batista et al., 2004; Fernández, García, del Jesus, & Herrera, 2008). As cost-sensitive methods we study several modifications to well-known classification methods such as C4.5 (Quinlan, 1993), Support Vector Machines (SVMs) (Vapnik, 1998), k-Nearest Neighbor classifier (k-NN) (Cover & Hart, 1967) or Fuzzy Hybrid Genetics-Based Machine Learning (FH-GBML) rule generation algorithm (Ishibuchi & Yamamoto, 2005). The combination of these approaches is carried out through a wrapper classifier (Chawla, Cieslak, Hall, & Joshi, 2008) that uses the aforementioned cost-sensitive techniques with the preprocessing technique obtaining the adequate parameters to perform altogether.

In this work, we focus on imbalanced binary classification problems, having selected a benchmark of 66 problems from KEEL dataset repository[1] (Alcalá-Fdez et al., 2011). We perform our experimental study focusing on the precision of the models using the Area Under the ROC curve (AUC) (Huang & Ling, 2005). This study is carried out using nonparametric tests to check whether there exist significant differences among the obtained results (Demšar, 2006; García & Herrera, 2008).

On the other hand, after comparing these techniques we also want to find what is the source where the difficulties for imbalanced classification emerge. Many other studies on the behavior of several standard classifiers in imbalance domains have shown that significant loss of performance is mainly due to skew of class distributions. However, several investigations also suggest that there are other factors that contribute to such performance degradation, for example, size of the dataset, class imbalance level, small disjuncts, density, and overlap complexity (Japkowicz & Stephen, 2002; Prati & Batista, 2004; Weiss & Provost, 2003). This work focuses on the analysis of two of the most pressing open problems related to data intrinsic characteristics: overlap and dataset shift.

This paper is organized as follows: first, Section 2 presents the problem of imbalanced datasets and the metric we have employed in this context whereas Section 3 describes some ways to tackle the problem: the preprocessing methods used, cost-sensitive classification and a wrapper approach to combine both. Next, Section 4 describes the algorithms we have used in this study, selected benchmark datasets and the configuration of the methods. In Section 5 an analysis of preprocessing techniques versus cost-sensitive learning approaches can be found. Section 6 is devoted to discuss the imbalanced classification problem characteristics that make that problem difficult, analysing the open problems related to data intrinsic characteristics, class overlap and dataset shift. The conclusions of this work can be found in Section 7. Additionally, we include an appendix with the complete tables of results from the experimental study.

2. Imbalanced datasets in classification

In this section, we first introduce the problem of imbalanced datasets and then we present the evaluation metrics for this type of classification problem which differ from usual measures in classification.

2.1. The problem of imbalanced datasets

In some classification problems, the number of instances of every class can be very different. Specifically when facing a dataset with only two classes, the imbalance problem occurs when one class is represented by a large number of examples, while the other is represented by only a few (Chawla, Japkowicz, & Kotcz, 2004).

The problem of imbalanced datasets is extremely significant (Yang & Wu, 2006) because it is implicit in most real world applications, such as very high resolution airborne imagery (Chen, Fang, Huo, & Li, 2011), e-mail foldering (Bermejo, Gámez, & Puerta, 2011) or micro seismic hazards in coal mines (Sikora, 2011), just citing some of them. It is important to point out that the minority class usually represents the concept of interest, for example patients with illnesses in a medical diagnosis problem; whereas the other class represents the counterpart of that concept (healthy patients).

Usually, standard classifier algorithms have a bias towards the majority class, since the rules that predict the higher number of examples are positively weighted during the learning process in favour of the accuracy metric. Consequently, the instances that belong to the minority class are misclassified more often than those belonging to the majority class. Another important issue related to this type of problem is the presence of small disjuncts in the dataset (Weiss & Provost, 2003) and the difficulty most learning algorithms have in detecting those regions. Furthermore, the main handicap in imbalanced datasets is the overlapping between the examples of the positive and the negative class (García, Mollineda, & Sánchez, 2008). These facts are depicted in Fig. 1(a) and (b) respectively.

2.2. Evaluation in imbalanced domains

The measures of the quality of classification are built from a confusion matrix (shown in Table 1) which records correctly and incorrectly recognized examples for each class.

The most commonly used empirical measure, accuracy (1), does not distinguish between the number of correct labels of different classes, which in the framework of imbalanced problems may lead to erroneous conclusions. For example a classifier that obtains an accuracy of 90% in a dataset with a degree of imbalance 9:1, might not be accurate if it does not cover correctly any minority class instance.

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

Because of this, instead of using accuracy, more correct metrics are considered. Specifically, from Table 1 it is possible to obtain four metrics of performance that measure the classification quality for the positive and negative classes independently:

- **True positive rate** $TP_{rate} = \frac{TP}{TP+FN}$ is the percentage of positive cases correctly classified as belonging to the positive class.
- **True negative rate** $TN_{rate} = \frac{TN}{FP+TN}$ is the percentage of negative cases correctly classified as belonging to the negative class.
- **False positive rate** $FP_{rate} = \frac{FP}{FP+TN}$ is the percentage of negative cases misclassified as belonging to the positive class.
- **False negative rate** $FN_{rate} = \frac{FN}{TP+FN}$ is the percentage of positive cases misclassified as belonging to the negative class.

One appropriate metric that could be used to measure the performance of classification over imbalanced datasets is the Receiver Operating Characteristic (ROC) curve (Bradley, 1997). In this curve, the tradeoff between the benefits (TP_{rate}) and costs (FP_{rate}) can be visualized, and acknowledges the fact that the capacity of any classifier cannot increase the number of true positives without also increasing the false positives. The Area Under the ROC Curve (AUC) (Huang & Ling, 2005) corresponds to the probability of correctly identifying which of the two stimuli is noise and which is signal plus noise. AUC provides a single-number summary for the performance of learning algorithms.

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

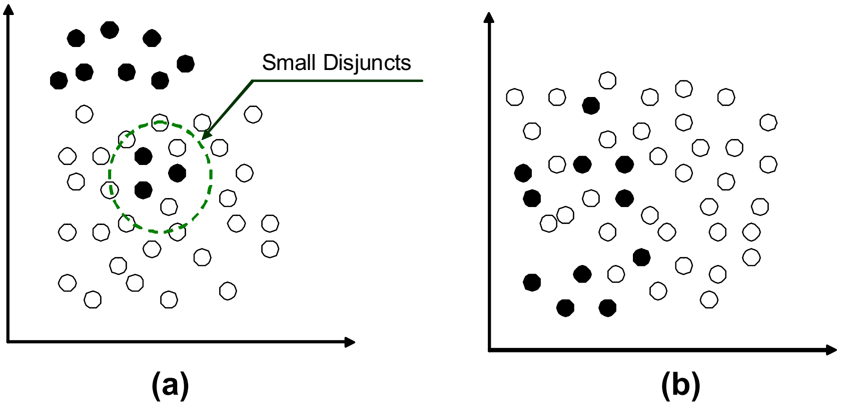

Fig. 1. Example of the imbalance between classes: (a) small disjuncts and (b) overlapping between classes.

Table 1
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)

The way to build the ROC space is to plot on a two-dimensional chart the true positive rate (*Y* axis) against the false positive rate (*X* axis) as shown in Fig. 2. The points (0,0) and (1,1) are trivial classifiers in which the output class is always predicted as negative and positive respectively, while the point (0,1) represents perfect classification. To compute the AUC we just need to obtain the area under the curve as:

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

3. Solutions to the class imbalance problem

A large number of approaches have been previously proposed to deal with the class-imbalance problem. These approaches can be categorized in two groups: the internal approaches that create new algorithms or modify existing ones to take the class-imbalance problem into consideration (Barandela, Sánchez, García, & Rangel, 2003; Ducange, Lazzerini, & Marcelloni, 2010; Wu & Chang, 2005; Xu, Chow, & Taylor, 2007) and external approaches that preprocess the data in order to diminish the effect of their class imbalance (Batista et al., 2004; Estabrooks, Jo, & Japkowicz, 2004). Furthermore, cost-sensitive learning solutions incorporating both the data and algorithmic level approaches assume higher misclassification costs with samples in the minority class and seek to minimize the high cost errors (Domingos, 1999; Sun, Kamel, Wong, & Wang, 2007; Zhou & Liu, 2006).

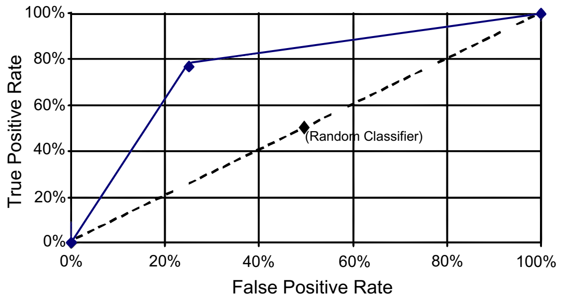

Fig. 2. Example of an ROC plot. Two classifiers are represented: the solid line is a good performing classifier whereas the dashed line represents a random classifier.

In this section, we first introduce the SMOTE and SMOTE + ENN methods in Section 3.1. Then, cost-sensitive learning is described in Section 3.2. Finally, Section 3.3 presents a framework to automatically detect a threshold for preprocessing using an underlying algorithm, in this case, a cost-sensitive approach.

3.1. Preprocessing imbalanced datasets. The SMOTE and SMOTE + ENN algorithms

As mentioned before, applying a preprocessing step in order to balance the class distribution is an effective solution to the imbalanced dataset problem (Batista et al., 2004). Specifically, in this work we have chosen an oversampling method which is a well-known reference in the area: the SMOTE algorithm (Chawla et al., 2002) and a variant called SMOTE + ENN (Batista et al., 2004) as they have been shown to present a very robust behavior among many different situations (Batista et al., 2004; Fernández et al., 2008).

In this approach, the positive class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. This process is illustrated in Fig. 3, where x_i is the selected point, x_{i1} to x_{i4} are some selected nearest neighbors and r_1 to r_4 the synthetic data points created by the randomized interpolation.

Synthetic samples are generated in the following way: take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point

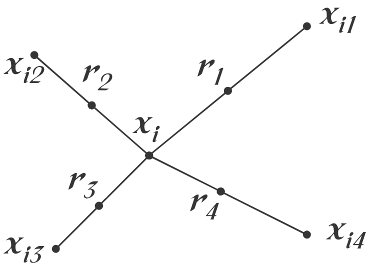

Fig. 3. An illustration of how to create the synthetic data points in the SMOTE algorithm.

along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general. An example is detailed in Fig. 4.

In short, its main feature is to form new minority class examples by interpolating between several minority class examples that lie together. Thus, the overfitting problem is avoided and causes the decision boundaries for the minority class to spread further into the majority class space.

Nevertheless, class clusters may be not well defined in cases where some majority class examples invade the minority class space. The opposite can also be true, since interpolating minority class examples can expand the minority class clusters, introducing artificial minority class examples too deeply into the majority class space. Inducing a classifier in such a situation can lead to overfitting. For this reason we also consider a hybrid approach in this work, "SMOTE + ENN", where the Wilson's ENN rule (Wilson, 1972) is used after the SMOTE application to remove from the training set any example misclassified by its three nearest neighbors.

3.2. Cost-sensitive learning

Cost-sensitive learning takes into account the variable cost of a misclassification of the different classes (Domingos, 1999; Zadrozny et al., 2003). A cost matrix codifies the penalties of classifying examples of one class as a different one. Let $C(i,j)$ be the cost of predicting an instance of class i as class j; with this notation $C(+,-)$ is the cost of misclassifying an instance of the positive class as if it was negative and $C(-,+)$ is the cost of the opposite case.

When dealing with imbalanced problems it is usually more important to recognize the positive instances rather than the negative ones. Therefore, the cost of misclassifying a positive instance is higher than the cost of misclassifying a negative one ($C(+,-) > C(-,+)$). As a classical example, the reader may refer to a diagnosis problem in which it is often less dangerous to obtain a false positive than a false negative.

Three main general approaches have been proposed to deal with cost-sensitive problems:

1. Methods based on modifying the training data. The most popular technique lies in resampling the original class distribution of the training dataset according to the cost decision matrix by means of undersampling/oversampling, modifying decision thresholds or assigning instance weights. These modifications have shown to be effective and can also be applied to any cost insensitive learning algorithm (Zadrozny et al., 2003; Zhou & Liu, 2006).
2. Methods that change the learning process in order to build a cost-sensitive classifier, for example, in the context of decision tree induction, the tree-building strategies are adapted to minimize the misclassification costs. The cost information is used to: (1) choose the best attribute to split the data Ling et al. (2004) and Riddle, Segal, and Etzioni (1994); and (2) determine whether a subtree should be pruned Bradford, Kunz, Kohavi, Brunk, and Brodley (1998).

```
Consider a sample (6,4) and let (4,3) be its nearest neighbor.
(6,4) is the sample for which k-nearest neighbors
are being identified and (4,3) is one of its k-nearest neighbors.
Let: f1_1 = 6 f2_1 = 4, f2_1 - f1_1 = -2
f1_2 = 4 f2_2 = 3, f2_2 - f1_2 = -1
The new samples will be generated as
(f1',f2') = (6,4) + rand(0-1) * (-2,-1)
rand(0-1) generates a random number between 0 and 1.
```

Fig. 4. Example of the SMOTE application.

3. Methods based on the Bayes decision theory that assign instances to the class with minimum expected cost. For example, a typical decision tree for a binary classification problem assigns the class label of a leaf node depending on the majority class of the training samples that reach the node. A cost-sensitive algorithm assigns the class label to the node that minimizes the classification cost Domingos (1999) and Zadrozny and Elkan (2001).

Cost-sensitive learning supposes that there is a cost matrix available for the different type of errors. However, given a dataset, this matrix is not usually given Sun et al. (2007, 2009).

3.3. Hybridization. Automatically countering imbalance

The different solutions used to deal with the imbalanced problem have been presented in the previous subsections. So the question now is "Can we use both techniques together and achieve better results?"

Cost-sensitive learning algorithms associate high misclassification costs to positive instances which bias the search towards the positive class. If the cost associated to positive instances is too high or if the specific cost-sensitive algorithm is easily biased towards the positive class, we can observe that the decision region generated by the algorithm is far away from those instances. Therefore, we need to bias those algorithms in a way that pushes the boundary towards the positive instances, but still classifies correctly both classes. If the positive instances are sparse, a case that is likely to occur in imbalanced datasets, then the boundary may not have the proper shape.

On the other hand, preprocessing methods try to balance class distributions to let the standard classifier algorithms accomplish similar results to their performance in a balanced data scenario. There is a diversity of preprocessing methods with a behavior focused on generating new samples, removing some of the samples or carrying out both operations jointly. Nevertheless, these methods can fail due to the loss of information produced when we delete samples that define our decision boundaries or when we create examples that introduce noise to the classifier.

Regarding cost-sensitive learning classifiers, a way to avoid biasing towards positive instances without modifying the algorithm itself lies in the creation of a few positive instances or the deletion of some negative examples. This causes a more balanced data distribution which means that the misclassification costs associated to positive instances will also be lower thus making the search process less biased. In addition, since we are using a cost-sensitive classifier we do not need to apply a preprocessing procedure to balance the distribution because cost-sensitive learners are able to learn in imbalanced conditions, therefore, the resampling stage is quicker than using only a preprocessing approach and the whole learning process is sped up, especially when the base classifier efficiency deeply depends on the number of instances.

We can find some works related to this idea. For example, Akbani, Kwek, and Japkowicz (2004) propose a solution with support vector machines where they integrate a cost-sensitive support vector machine (Veropoulos, Campbell, & Cristianini, 1999) with the SMOTE technique of oversampling the minority instances (Chawla et al., 2002). With this behavior they manage to push the boundary away from the positive instances (cost-sensitive learning) and make the boundary better defined (because of the denser positive instance distribution).

Due to the previous facts we aim to develop a procedure to integrate the cost-sensitive learning and preprocessing approaches into one. Chawla et al. (2008) propose a wrapper paradigm that discovers the amount of resampling needed for a dataset based on optimizing evaluation functions which can include the cost associated to the classification. This wrapper infrastructure applies cross-validation to first discover the best amounts of undersampling and oversampling, applies the preprocessing algorithms with the amounts estimated and finally runs the algorithm used over the preprocessed dataset.

Obviously, searching the entire space of undersampling and SMOTE combinations can quickly become intractable, so the search procedure must be fine-tuned. This strategy removes the "excess" examples of the majority classes, which reduces the size of the training dataset. This also makes learning time more manageable. SMOTE is used to add synthetic examples of the minority classes and increase the generalization performance of the classifier over the minority classes. Fig. 5 shows the algorithm procedure.

The estimation is done over a training and a test set. The training data is split into five partitions for an internal five-fold cross-validation. The wrapper applies this independent validation stage to each fold to discover the appropriate percentages of sampling for a given method and classifier combination. Once these percentages are discovered, the classifier is re-learned on the original training fold using the discovered percentages and tested on the corresponding testing fold.

The undersampling estimation starts with no undersampling for all majority classes and obtains baseline results on the training data. Then it traverses through the search space of undersampling percentages in decrements of *Sample Decrement*, in a greedy iterative fashion, to increase performance over the minority classes without sacrificing performance on the majority class.

The oversampling algorithm evaluates different amounts of SMOTE at steps of *Sample Increment* (percentage of the number of examples from the minority class that will be generated in each step). This is a greedy search, and at each step the new performance estimates become the new baseline. That is, the initial baseline is the performance obtained via the Wrapper Undersample. If SMOTE = *Sample Increment* improves the performance over that baseline by some margin *Increment Min*, then the performance achieved at SMOTE = *Sample Increment* becomes the new baseline. The amount of SMOTE is then incremented by *Sample Increment*, and another evaluation is performed to check if the performance increase at new SMOTE amount is at least greater than *Increment Min*. This process repeats, greedily, until no performance gains are observed.

However, there is an important caveat to the search to avoid being trapped in a local maximum. If the average does not improve by *Increment Min* we have to verify that we have not settled on a local maximum. In order to do so, we look ahead some more steps at increasing amounts of SMOTE. If the look-ahead does not result in an improvement in performance, then the amount of SMOTE is reset to the value discovered prior to the look-ahead. This is done to allow SMOTE to introduce additional examples with the aim of improving performance. However, if the addition of examples does not help, then we go back to using the lesser amount of SMOTE discovered prior to the look-ahead.

We can use different measures to evaluate the performance of the classifier to estimate the sampling parameters. Since we are using cost-sensitive learning algorithms as base classifiers a logical evaluation criteria is the cost itself. Cost is calculated as shown in Eq. (3) when we assume $C(+|+) = C(-|-) = 0$ (as it is usual in imbalanced classification).

$$cost = FNrate \cdot C(-|+) + FPrate \cdot C(+|-) \tag{3}$$

4. Experimental framework

In this section, we first introduce the algorithms which are included in the study (Section 4.1). Next, we provide details of the imbalanced problems chosen for the experimentation and the configuration parameters of the methods (Sections 4.2 and 4.3). Finally, we present the statistical tests applied to compare the results obtained with the different classifiers (Section 4.4).

4.1. Algorithms selected for the study

This section presents the description of the state of the art algorithms of four different classification paradigms selected for our study. For each paradigm we outline the base classifier commonly used in general classification problems and the cost-sensitive learning version associated to that classifier.

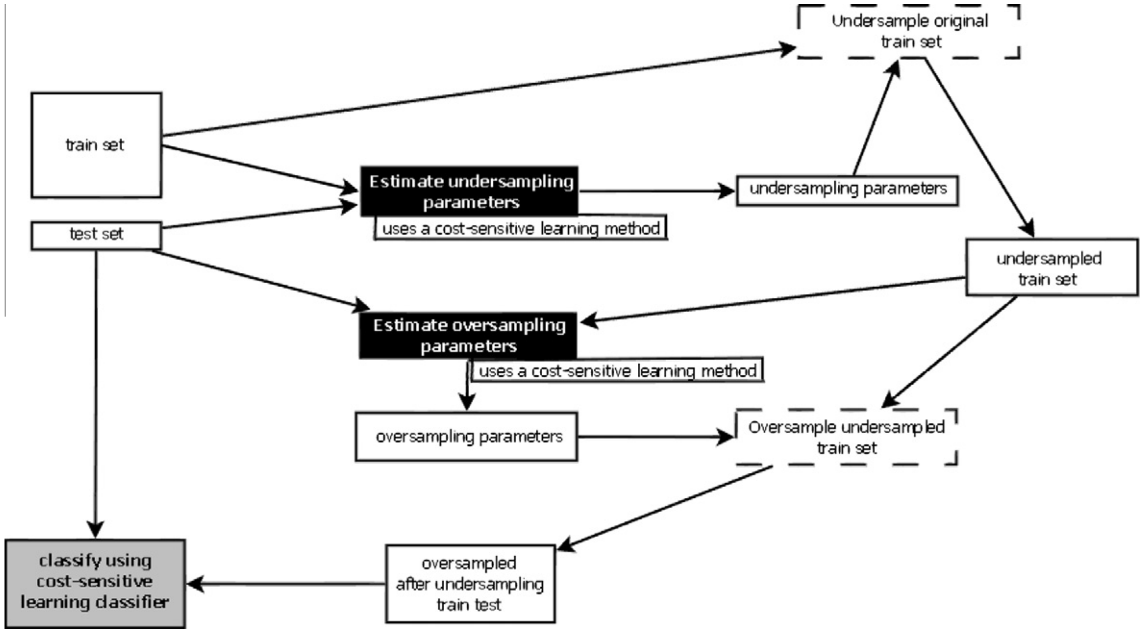
Undersample original train set
train set
Estimate undersampling parameters
undersampling parameters
uses a cost-sensitive learning method
test set
undersampled train set
Estimate oversampling parameters
uses a cost-sensitive learning method
oversampling parameters
Oversample undersampled train set
classify using cost-sensitive learning classifier
oversampled after undersampling train test

Fig. 5. Illustration on the wrapper undersample SMOTE algorithm. Dashed lines means resampling actions, black boxes represent the parameters estimation and the final result is in grey.

4.1.1. Decision trees

Decision trees use simple knowledge representation to classify examples into a finite number of classes. In a typical setting, the tree nodes represent the attributes, the edges represent the possible values for a particular attribute, and the leaves are assigned with class labels. Classifying a test sample is straightforward once a decision tree has been constructed. An object is classified by following paths from the root node through the tree to a leaf, taking the edges corresponding to the values of attributes.

C4.5 decision tree. C4.5 (Quinlan, 1993) is a decision tree generating algorithm. It induces classification rules in the form of decision trees from a set of given examples. The decision tree is constructed top-down using 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 chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

Cost-sensitive C4.5 decision tree. The cost-sensitive C4.5 decision tree (C4.5CS) (Ting, 2002) is a method to induce cost-sensitive trees that seeks to minimize the number of high cost errors and, as a consequence of that, leads to minimization of the total misclassification costs in most cases.

The method changes the class distribution such that the tree induced is in favor of the class with high weight/cost and is less likely to commit errors with high cost. Specifically, the computation of the split criteria for C4.5 (normalized information gain) is modified to take into account the *a priori* probabilities according to the number of samples for each class.

C4.5CS modifies the weight of an instance proportional to the cost of misclassifying the class to which the instance belonged, leaving the sum of all training instance weights still equal to N. Let $C(j)$ be the cost of misclassifying a class j instance; the weight of a class j instance can be computed as

$$w(j) = C(j)\frac{N}{\sum_i C(i)N_i} \tag{4}$$

such that the sum of all instance weights is $\sum_j w(j)N_j = N$.

The standard greedy divide-and-conquer procedure for inducing minimum error trees can then be used without modification, except that $W_j(t)$ is used instead of $N_j(t)$ in the computation of the test selection criterion in the tree growing process and the error estimation in the pruning process. That $W_j(t)$ is the result of weighting the initial number of instances from a class with the weight computed in Eq. (4): $W_j(t) = w(j) \cdot N_j(t)$ Thus, both processes are affected due to this change.

This modification effectively converts the standard tree induction procedure that seeks to minimize *the number of errors*, regardless of cost, to a procedure that seeks to minimize *the number of errors with high weight or cost*. To classify a new instance, C4.5CS predicts the class which has the maximum weight at a leaf, as in C4.5.

C4.5CS also introduces another optional modification that alters the usual classification process after creating the decision tree. Instead of classifying using the minimum error criteria, it is advisable to classify using the expected misclassification cost in the last part of the classification procedure. The expected misclassification cost for predicting class i with respect to the instance x is given by

$$EC_i(x) \propto \sum_j W_j(t(x))cost(i,j) \tag{5}$$

where $t(x)$ is the leaf of the tree that instance x falls into and $W_j(t)$ is the total weight of class j training instances in node t.

To classify a new instance x using a minimum error tree with the minimum expected cost criterion, $EC_i(x)$ is computed for every class. The instance x is assigned to class i with the smallest value for $EC_i(x)$; that is, $EC_i(x) < EC_{i'}(x)$ for all $i' \neq i$.

4.1.2. Support vector machines

SVMs are one of the binary classifiers based on maximum margin strategy introduced by Vapnik and Lerner (1963). Originally, SVMs were designed for linear two-class classification with margin, where margin means the minimal distance from the separating hyperplane to the closest data points. SVMs seek an optimal separating hyperplane, where the margin is maximal. The solution is based only on those data points at the margin. These points are called as support vectors. The linear SVMs have been extended to nonlinear examples when the nonlinear separated problem is transformed into a high dimensional feature space using a set of nonlinear basis functions. However, the SVMs are not necessary to implement this transformation to determine the separating hyperplane in the possibly high dimensional feature space. Instead, a kernel representation can be used, where the solution is written as a weighted sum of the values of a certain kernel function evaluated at the support vectors.

Soft margin SVM. In 1995, Cortes and Vapnik suggested a modified maximum margin idea that allows for mislabeled examples (Cortes & Vapnik, 1995; Vapnik, 1998). If there exists no hyperplane that can split the "yes" and "no" examples, the Soft Margin method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. The method introduces slack variables, ξ_i, which measure the degree of misclassification of the datum x_i.

Cost-sensitive SVM. The cost-sensitive SVM (SVMCS) (Veropoulos et al., 1999) is a modification of the soft-margin support vector machine. We need to bias SVM in a way that will push the boundary away from the positive instances using different error costs for the positive (C^+) and negative (C^-) classes. Specifically, the change implies a new optimization function

$$\min_{\mathbf{w},\xi,b}\max_{\alpha,\beta}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C^+\sum_{\{i|y_i=+1\}}^{n_+}\xi_i + C^-\sum_{\{j|y_j=-1\}}^{n_-}\xi_j - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x_i} - b) - 1 + \xi_i] - \sum_{i=1}^{n}\beta_i\xi_i\right\} \tag{6}$$

The constraints on α_i then become:

$$0 \leqslant \alpha_i \leqslant C^+ \quad \text{if } y_i = +1 \tag{7}$$

and

$$0 \leqslant \alpha_i \leqslant C^- \quad \text{if } y_i = -1 \tag{8}$$

Furthermore, $\xi_i > 0$ only when $\alpha_i = C$. Therefore non-zero errors on positive support vectors will have larger α_i while non-zero errors on negative support vectors will have smaller α_i. The net effect is that the boundary is pushed more towards the negative instances.

4.1.3. Fuzzy rule based classification system learning methods

A fuzzy rule based classification system (FRBCS) has two main components: the inference system and the knowledge base. In a linguistic FRBCS, the knowledge base is composed of a rule base, constituted by a set of fuzzy rules, and the data base that stores the membership functions of the fuzzy partitions associated to the input variables.

In this work we use fuzzy rules of the following form for our FRBCSs:

$$\begin{aligned}\text{Rule } R_j :\ & \text{If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \\ & \text{then Class} = C_j \text{ with } RW_j\end{aligned} \tag{9}$$

where R_j is the label of the jth rule, $x = (x_1,\ldots,x_n)$ is an n-dimensional pattern vector, A_{ji} is an antecedent fuzzy set, C_j is a class label, and RW_j is the rule weight (Ishibuchi & Nakashima, 2001). We use triangular membership functions as fuzzy partitions associated to the input variables. To compute the rule weight, many alternatives have been proposed, although we have considered as a good choice the use of the heuristic method known as the Penalized Certainty Factor (PCF) Ishibuchi and Yamamoto (2005):

$$\mathrm{PCF}_j = \frac{\sum_{x_p \in C_j} \mu_{A_j}(x_p) - \sum_{x_p \notin C_j} \mu_{A_j}(x_p)}{\sum_{p=1}^{m} \mu_{A_j}(x_p)} \tag{10}$$

where x_p is the pth example of the training set, C_j is the consequent class of rule j and $\mu_{A_j}(x_p)$ is the membership degree of the example with the antecedents of the rule.

Fuzzy hybrid genetic based machine learning rule generation algorithm. In order to build the rule base, we have chosen the FH-GBML algorithm (Ishibuchi, Yamamoto, & Nakashima, 2005), a proposal that presents a good behaviour in both standard and imbalanced classification (Fernández, del Jesús, & Herrera, 2010; Luengo, Fernández, García, & Herrera, 2011).

The FH-GBML method consists of a Pittsburgh approach where each rule set is handled as an individual. It also contains a Genetic Cooperative-Competitive Learning (GCCL) approach (an individual represents a unique rule), which is used as a kind of heuristic mutation for partially modifying each rule set. This method uses standard fuzzy rules with rule weights (Ishibuchi & Yamamoto, 2005) where each input variable x_i is represented by a linguistic term or label. The system defines 14 possible linguistic terms for each attribute as well as a special "do not care" as an additional linguistic term.

In the learning process, N_{pop} rule sets are created by randomly selecting N_{rule} training patterns. Then, a fuzzy rule from each of the selected training patterns is generated by probabilistically choosing an antecedent fuzzy set from the 14 candidates $\left(P_{donot\ care}(B_k) = \frac{\mu_{B_k}(x_{pi})}{\sum_{j=1}^{14} \mu_{B_j}(x_{pi})}\right)$ and each antecedent fuzzy set of the generated fuzzy rule is replaced with don't care using a pre-specified probability $P_{do\ not\ care}$.

N_{pop} −1 rule sets are generated by selection, crossover and mutation in the same manner as the Pittsburgh-style algorithm. Next, with a pre-specified probability, a single iteration of the Genetic Cooperative–Competitive-style algorithm is applied to each of the generated rule sets.

Finally, the best rule set is added to the current population in the newly generated $(N_{pop} - 1)$ rule sets to form the next population and, if the stopping condition is not satisfied, the genetic process is repeated again. Classification is performed following the fuzzy reasoning method of the winning rule.

Cost-sensitive fuzzy hybrid genetic based machine learning rule generation algorithm. The FH-GBML-CS (Fuzzy Hybrid Genetics-Based Machine Learning Cost-Sensitive) algorithm (López, Fernández, & Herrera, 2010) is a modification of the FH-GBML original algorithm. The main goal of FH-GBML-CS is to obtain a FRBCS that is able to consider the different costs associated to misclassification of some of its samples during the building process of the RB. To achieve that purpose an algorithmic level solution is used, modifying the original behaviour of the FH-GBML algorithm in some of its steps:

- *Adaptation of the fitness function of the Pittsburgh approach*. Instead of using the number of correctly classified training examples FH-GBML-CS tries to minimize the misclassification cost: $FN_{rate} \cdot C(-,+) + FP_{rate} \cdot C(+,-)$.
- *Modifications in the computation of the rule weight*. The PCF heuristic has been adapted to cost-sensitive learning building the Cost-Sensitive Penalized Certainty Factor (CS-PCF) which is used in FH-GBML-CS to compute the rule weight:

$$\mathrm{CS} - \mathrm{PCF}_j = \frac{\sum_{x_p \in C_j} \mu_{A_j}(x_p) \times \mathrm{Cs}_p}{\sum_{p=1}^{m} \mu_{A_j}(x_p) \times \mathrm{Cs}_p} - \frac{\sum_{x_p \notin C_j} \mu_{A_j}(x_p) \times \mathrm{Cs}_p}{\sum_{p=1}^{m} \mu_{A_j}(x_p) \times \mathrm{Cs}_p} \tag{11}$$

where Cs_p is the misclassification cost of an example from class p.
- *Different class label choice for the rule*. Instead of selecting the class considering only the highest compatibility the class with the highest *compatibility* · *cost* is chosen.

4.1.4. Lazy learning

Lazy learning is a set of methods in which generalization beyond the training data is delayed until a query is made to the system, as opposed to in eager learning, where the system tries to generalize the training data before receiving queries.

The main advantage gained in employing a lazy learning method is that the target function will be approximated locally, such as in the k-NN algorithm. Because the target function is approximated locally for each query to the system, lazy learning systems can simultaneously solve multiple problems and deal successfully with changes in the problem domain.

K-nearest neighbor algorithm. k-NN (Cover & Hart, 1967) finds a group of k instances in the training set that are closest to the test pattern. Given a test sample, the algorithm computes the distance (or similarity) between the test sample and all of the training samples to determine its k-nearest neighbors. The class of the test sample is decided by the most abundant class within the k-nearest samples.

Cost-sensitive k-nearest neighbor algorithm. Cost-sensitive k-NN algorithm (Hand & Vinciotti, 2003) is a cost-sensitive learning version of k-NN based on Bayes risk theory to assign each sample to its lowest risk class.

Let the cost of misclassifying a class i case be c_i. Now, if points at x are assigned to class 1, the loss at x is $c_0p(0|x)$. Similarly, if points at x are assigned to class 0, the loss at x is $c_1p(1|x)$. The minimum loss at x is thus achieved by assigning points at x to class 1 if $c_0p(0|x) < c_1p(1|x)$ and to class 0 otherwise. This is equivalent to the condition

$$p(1|x) > c_0/(c_0 + c_1) \tag{12}$$

Without loss of generality we will rescale the costs so that $(c_0 + c_1) = 1$, so that the classification rule becomes "Assign points at x to class 1 when $p(1|x) > c_0$ and to class 0 otherwise".

Nearest neighbor methods estimate the $p(i|x)$ by the proportion of class i points amongst the k nearest neighbors to the point x to be classified. This requires a choice of a distance metric and a choice of the parameter k.

To sum up, the cost-sensitive k-NN classification rule assigns a point with measurement vector x to class 1 if $k_1/k > c_0$, and otherwise to class 0, where k_1 is the number of class 1 points amongst the k design set points closest to x.

4.1.5. Summary of the different schemes selected for the experimental study

In this work, we test several combinations of preprocessing and cost-sensitive learning with the classification algorithms from each paradigm described in this section. Specifically, the schemes used can be arranged into three categories:

1. Oversampling approaches to balance the data distribution before applying the algorithm which were described in Section 3.1.
2. Cost-sensitive learning methods which take into consideration costs. The methods used are specific versions that come from the original non-balanced algorithms. These algorithm versions have been described in this section.
3. Application of the hybrid methodology that combines cost-sensitive learning and preprocessing: a methodology to automatically countering imbalance using cost was described in Section 3.3. We use different combinations of algorithms to evaluate the performance of the methodology.

Table 2
Acronyms used to designate the different algorithm variations used in the experimental study.

Acronym	Version description
None	The original classifier that names the algorithm family
SMOTE	The original classifier that names the algorithm family applied to a dataset preprocessed with the SMOTE algorithm
SENN	The original classifier that names the algorithm family applied to a dataset preprocessed with the SMOTE + ENN algorithm
CS	The cost-sensitive version of the original classifier from the corresponding algorithm family which was explained in the previous section
Wr_SMOTE	Version of the Wrapper routine described in the previous section that uses as main algorithm the cost-sensitive version of the algorithm family and only performs the oversampling step with the SMOTE algorithm
Wr_US	Version of the Wrapper routine described in the previous section that uses as main algorithm the cost-sensitive version of the algorithm family, performs the undersampling step with a random undersampling algorithm and the oversampling step with the SMOTE algorithm
Wr_SENN	Version of the Wrapper routine described in the previous section that uses as main algorithm the cost-sensitive version of the algorithm family and only performs the oversampling step with the SMOTE + ENN algorithm

For the sake of clarity, Table 2 indicates a list of acronyms used to identify the different algorithm versions for each paradigm and a brief description of each one of them.

4.2. Datasets and data partitions

In order to analyze the preprocessing approach against the cost-sensitive learning strategy, we have selected 66 datasets from the KEEL dataset repository[2] (Alcalá-Fdez et al., 2011).

In the specialized literature, researchers usually manage all imbalanced datasets as a whole (Barandela et al., 2003; Batista et al., 2004; Chen, Chen, Hsu, & Zeng, 2008). In this work we sort the different datasets according to their degree of imbalance using the imbalance ratio (IR) (Orriols-Puig & Bernadó-Mansilla, 2009), which is defined as the ratio of the number of instances of the majority class and the minority class.

The datasets are summarized in Table 3, where we denote the number of examples (#Ex.), number of attributes (#Atts.), class name of each class (positive and negative), class distribution and IR.

To develop the different experiments we consider a *5-folder cross-validation model*, i.e., five random partitions of data with a 20% and the combination of 4 of them (80%) as training and the remaining one as test. For each dataset we consider the average results of the five partitions. The datasets used in this study use the partitions provided by the repository in the imbalanced classification dataset section.[3]

4.3. Parameters

The configuration parameters for the base classifiers are shown in Table 4. The selected values are common for all the versions of the algorithm in the same family to maintain an experimental scenario on equal terms. On the other hand, the parameters for the preprocessing methods used in this study are presented in Table 5. Finally, Table 6 points out the parameters for the implementation of the wrapper routine. All these values were selected according to the recommendation of the corresponding authors of each algorithm, which is the default parameters' setting included in the KEEL software (Alcalá-Fdez et al., 2008).

The only ad-hoc parameter value is the k parameter of nearest neighbors. We have set that value to $k = 3$ instead of $k = 1$ which is the usual approach because the cost-sensitive k-NN used in this study achieves an identical performance for 1-NN and 1-NNCS.

Furthermore, we have to identify the misclassification costs associated to the positive and negative class for the cost-sensitive learning versions. If we misclassify a positive sample as a negative one the associated misclassification cost is the IR of the dataset ($C(+,-) = IR$) whereas if we misclassify a negative sample as a positive one the associated cost is 1 ($C(-,+) = 1$). The cost of classifying correctly is 0 ($C(+,+) = C(-,-) = 0$) because guessing the correct class should not penalize the built model.

Although we acknowledge that the tuning of the parameters for each method on each particular problem could lead to better results, we chose to maintain a baseline performance of each method as the basis for comparison. Since the experimental study is focused in the performance of methods from the same family, our hypothesis is that methods that win on average on all problems would also win if a better setting was used. Furthermore, in a framework where no method is tuned, winner methods tend to correspond to the most robust learners, which is also a desirable characteristic.

4.4. Statistical tests for performance comparison

Statistical analysis needs to be carried out in order to find significant differences among the results obtained by the studied methods (García, Fernández, Luengo, & Herrera, 2009). We consider the use of non-parametric tests, according to the recommendations made in Demšar (2006), García and Herrera (2008), García et al. (2009), García, Fernández, Luengo, and Herrera (2010) where a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers is presented. These tests are used due to the fact that the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, causing the statistical analysis to lose credibility (Demšar, 2006).

Since the study is split in parts comparing a group of algorithms, we use statistical tests for multiple comparisons. Specifically, we use the Iman–Davenport test (Sheskin, 2006) to detect statistical differences among a group of results and the Shaffer post-hoc test (Shaffer, 1986) in order to find out which algorithms are distinctive among an $n \times n$ comparison.

The post-hoc procedure allows us to know whether a hypothesis of comparison of means could be rejected at a specified level of significance α, which we set to 95% in our study. However, it is very interesting to 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 know whether two algorithms perform significantly differently and to what degree.

Furthermore, we consider the average ranking of the algorithms in order to show graphically how good a method is with respect to its partners. This ranking is obtained by assigning a position to each algorithm depending on its performance for each dataset. The algorithm which achieves the best accuracy in a specific dataset will have the first ranking (value 1); then, the algorithm with the second best accuracy is assigned rank 2, and so forth. This task is carried out for all datasets and finally an average ranking is computed as the mean value of all rankings.

These tests are suggested in the studies presented by Demšar (2006), García and Herrera (2008), and García et al. (2009), where their use in the field of machine learning is strongly recommended.

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

[3] http://www.keel.es/imbalanced.php.

Table 3
Summary of imbalanced datasets.

Datasets	# Ex.	# Atts.	Class (−,+)	%Class (−;+)	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
Ecoli034vs5	200	7	(p, imL, imU; om)	(10.00, 90.00)	9.00
Yeast2vs4	514	8	(cyt; me2)	(9.92, 90.08)	9.08
Ecoli067vs35	222	7	(cp, omL, pp; imL, om)	(9.91, 90.09)	9.09
Ecoli0234vs5	202	7	(cp, imS, imL, imU; om)	(9.90, 90.10)	9.10
Glass015vs2	172	9	(build-win-non_float-proc, tableware, build-win-float-proc; ve-win-float-proc)	(9.88, 90.12)	9.12
Yeast0359vs78	506	8	(mit, me1, me3, erl; vac, pox)	(9.88, 90.12)	9.12
Yeast02579vs368	1004	8	(mit, cyt, me3, vac, erl; me1, exc, pox)	(9.86, 90.14)	9.14
Yeast0256vs3789	1004	8	(mit, cyt, me3, exc; me1, vac, pox, erl)	(9.86, 90.14)	9.14
Ecoli046vs5	203	6	(cp, imU, omL; om)	(9.85, 90.15)	9.15
Ecoli01vs235	244	7	(cp, im; imS, imL, om)	(9.83, 90.17)	9.17
Ecoli0267vs35	224	7	(cp, imS, omL, pp; imL, om)	(9.82, 90.18)	9.18
Glass04vs5	92	9	(build-win-float-proc,containers; tableware)	(9.78, 90.22)	9.22
Ecoli0346vs5	205	7	(cp, imL, imU, omL; om)	(9.76, 90.24)	9.25
Ecoli0347vs56	257	7	(cp, imL, imU, pp; om, omL)	(9.73, 90.27)	9.28
Yeast05679vs4	528	8	(me2; mit, me3, exc, vac, erl)	(9.66, 90.34)	9.35
Ecoli067vs5	220	6	(cp, omL, pp; om)	(9.09, 90.91)	10.00
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
Ecoli0147vs2356	336	7	(cp, im, imU, pp; imS, imL, om, omL)	(8.63, 91.37)	10.59
Led7digit02456789vs1	443	7	(0, 2, 4, 5, 6, 7, 8, 9; 1)	(8.35, 91.65)	10.97
Glass06vs5	108	9	(build-win-float-proc,headlamps; tableware)	(8.33, 91.67)	11.00
Ecoli01vs5	240	6	(cp, im; om)	(8.33, 91.67)	11.00
Glass0146vs2	205	9	(build-win-float-proc, containers, headlamps, build-win-non_float-proc; ve-win-float-proc)	(8.29, 91.71)	11.06
Ecoli0147vs56	332	6	(cp, im, imU, pp; om, omL)	(7.53, 92.47)	12.28
Cleveland0vs4	177	13	(0; 4)	(7.34, 92.66)	12.62
Ecoli0146vs5	280	6	(cp, im, imU, omL; om)	(7.14, 92.86)	13.00
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

5. Experimental study

In this section, we will perform an analysis to determine the performance of the different alternatives used for imbalanced classification. Our aim is to analyze three different issues:

1. The improvement obtained by preprocessing datasets and cost-sensitive learning over the original algorithm.
2. The possible differences between the rebalancing techniques versus cost-sensitive learning and in which cases.

Table 4
Parameter specification for the algorithms family employed in the experimentation.

Algorithm family	Parameters

C4.5	Pruned = true
	Confidence = 0.25
	Minimum number of item-sets per leaf = 2

SVM	Kernel type = polynomial C = 100.0 Tolerance of termination criterion = 0.001 Degree (for kernel function) = 1 Gamma (for kernel function) = 0.01 coef0 (for kernel function) = 0.0 Use the shrinking heuristics = true
FH-GBML	Conjunction operator = product t-norm Rule weight = PCF (FH-GBML and FH-GBML + preprocessing) and PCF-SC (FH-GBML-CS) Fuzzy reasoning method = winning rule Number of fuzzy rules = 5 · d (max. 50 rules) Number of rule sets = 200 Crossover probability = 0.9 Mutation probability = 1/d Number of replaced rules = all rules except the best-one (Pittsburgh-part, elitist approach) Number of rules/5 (GCCL-part) Total number of generations = 1.000 Do not care probability = 0.5 Probability of the application of the GCCL iteration = 0.5
k-NN	k = 3 Distance = Heterogeneous value difference metric (HVDM)

Table 5
Parameter specification for the preprocessing algorithms used in this study.

Preprocessing Algorithm	Parameters

SMOTE	kSMOTE = 5
	Balancing = 1:1
	distanceFunction = HVDM

SMOTE_ENN	kSMOTE = 5
	kENN = 3
	Balancing = 1:1
	distanceFunction = HVDM

Table 6
Parameter specification for the wrapper routine.

Parameter	Value
Sample decrement	10%
Sample increment	100%
Increment min	5%
Look-ahead steps	2

3. Whether a hybrid methodology that combines a preprocessing approach and a cost-sensitive learning algorithm supposes a positive synergy and enables the achievement of more accurate results.

The study is divided into different paradigms to check whether the conclusions achieved for one paradigm can be extrapolated to the others.

5.1. Study of decision trees versions: C4.5

Table 7 shows the average results in training and test together with the corresponding standard deviation for the seven versions of the C4.5 algorithm used in the study: the base classifier, the base classifier used over the preprocessed datasets, the cost-sensitive version of the algorithm and the hybrid versions of it. We stress in **boldface** the best results achieved for the prediction ability of the different techniques. The complete table of results for all datasets is shown in the appendix of this work.

Table 7
Average table of results using the AUC measure for the C4.5 variety of algorithms.

Algorithm	AUC_{tr}	AUC_{tst}
C4.5	0.8774 ± 0.0392	0.7902 ± 0.0804
C4.5 SMOTE	0.9606 ± 0.0142	0.8324 ± 0.0728
C4.5 SENN	0.9471 ± 0.0154	**0.8390 ± 0.0772**
C4.5CS	0.9679 ± 0.0103	0.8294 ± 0.0758
C4.5 Wr_SMOTE	0.9679 ± 0.0103	0.8296 ± 0.0763
C4.5 Wr_US	0.9635 ± 0.0139	0.8245 ± 0.0760
C4.5 Wr_SENN	0.9083 ± 0.0377	0.8145 ± 0.0712

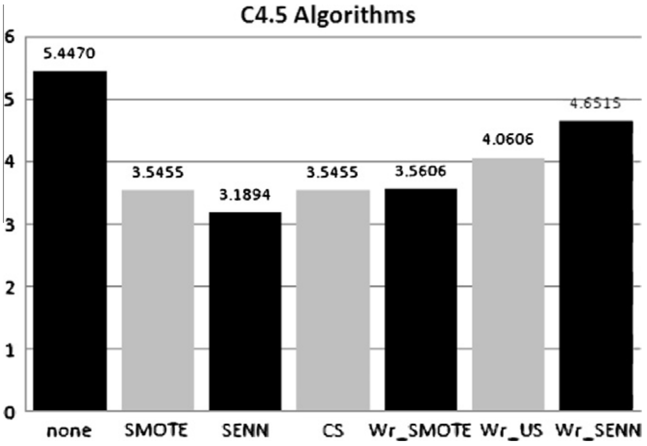

Fig. 6. Average rankings using the AUC measure for the C4.5 variety of algorithms.

From this table of results it can be observed that the highest average value corresponds to preprocessing approaches closely followed by the cost-sensitive learning approach and one version of the wrapper routine. This suggests the goodness of the preprocessing and cost-sensitive learning approaches.

In order to compare the results, a multiple comparison test is used to find the performance relationship between the different versions studied. The results of the statistical analysis of the C4.5 family are as follows. For the sake of a visual comparison, Fig. 6 shows the average ranking obtained through Friedman's test (Friedman, 1937) for these approaches. Under the AUC measure, the Iman–Davenport test detects significant differences among the algorithms, since the p-value returned (1.88673E−10) is lower than our α-value (0.05). The differences found are analyzed with a Shaffer test, shown in Table 8. 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 compared have no significant differences. In brackets, the adjusted p-value associated to each comparison is shown.

Observing the results from Tables 7 and 8, we conclude that the standard C4.5 approach is outperformed by most of the methodologies that deal with imbalanced data. The base version is different from every other version except the hybrid version that uses only an oversampling step with SMOTE + ENN. Thus, we can state that the imbalanced classification approaches (preprocessing and cost-sensitive learning) improve the base classifier.

Comparing the results when applying preprocessing we can see that the performance of these methods is not statistically different for any of its versions. In addition, the performance of those preprocessing methods is also not different to the cost-sensitive

Table 8
Shaffer test for the C4.5 variety of algorithms using the AUC measure.

C4.5	None	SMOTE	SENN	CS	Wr_SMOTE	Wr_US	Wr_SENN
None	×	−(6.404E−6)	−(4.058E−8)	−(6.404E−6)	−(7.904E−6)	−(.00341)	=(.37846)
SMOTE	+(6.404E−6)	×	=(1.0)	=(1.0)	=(1.0)	=(1.0)	+(.04903)
SENN	+(4.058E−8)	=(1.0)	×	=(1.0)	=(1.0)	=(.22569)	+(.00152)
CS	+(6.404E−6)	=(1.0)	=(1.0)	×	=(1.0)	=(1.0)	+(.04903)
Wr_SMOTE	+(7.904E−6)	=(1.0)	=(1.0)	=(1.0)	×	=(1.0)	+(.04903)
Wr_US	+(.00341)	=(1.0)	=(.22569)	=(1.0)	=(1.0)	×	=(1.0)
Wr_SENN	=(.37846)	−(.04903)	−(.00152)	−(.04903)	−(.04903)	=(1.0)	×

Table 9
Average table of results using the AUC measure for the SVM variety of algorithms.

Algorithm	AUC_{tr}	AUC_{tst}
SVM	0.7563 ± 0.0198	0.7341 ± 0.0530
SVM SMOTE	0.8806 ± 0.0140	0.8514 ± 0.0568
SVM SENN	0.8826 ± 0.0146	**0.8517 ± 0.0557**
SVMCS	0.7869 ± 0.0281	0.7651 ± 0.0621
SVM Wr_SMOTE	0.6981 ± 0.0283	0.6820 ± 0.0521
SVM Wr_US	0.7077 ± 0.0315	0.6895 ± 0.0619
SVM Wr_SENN	0.7656 ± 0.0303	0.7461 ± 0.0662

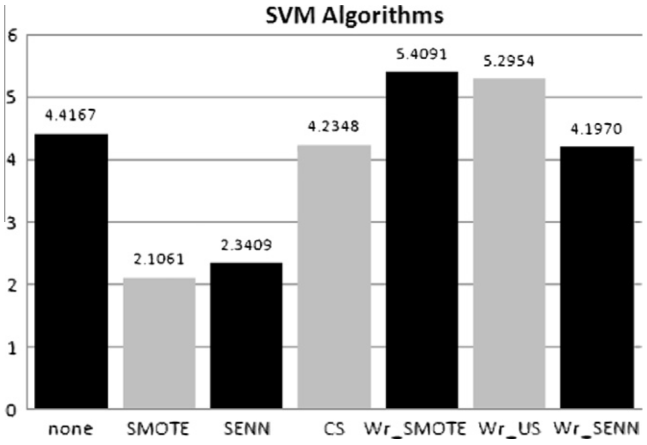

Fig. 7. Average rankings using the AUC measure for the SVM variety of algorithms.

Table 11
Average table of results using the AUC measure for the FH-GBML variety of algorithms.

Algorithm	AUC_{tr}	AUC_{tst}
FH-GBML	0.8352 ± 0.0226	0.7692 ± 0.0756
FH-GBML SMOTE	0.9181 ± 0.0130	0.8364 ± 0.0733
FH-GBML SENN	0.9127 ± 0.0131	0.8350 ± 0.0736
FH-GBMLCS	0.9328 ± 0.0076	**0.8373 ± 0.0773**
FH-GBML Wr_SMOTE	0.9330 ± 0.0075	0.8244 ± 0.0830
FH-GBML Wr_US	0.9304 ± 0.0095	0.8322 ± 0.0834
FH-GBML Wr_SENN	0.8866 ± 0.0306	0.8168 ± 0.0901

learning version of C4.5. This means that in decision trees both preprocessing and cost-sensitive learning are good approaches to deal with the problem.

Focusing on the hybridization of cost-sensitive learning and preprocessing by using a wrapper routine, it can be seen that there are significant differences both between the different hybrid versions and with the other alternatives. The hybrid version that uses only an oversampling step with SMOTE + ENN is outperformed by all the other versions except the base version. The rest of the hybrid versions are not statistically different from the performance of usual approaches for imbalanced classification. Therefore, we cannot state that the hybridization in decision trees produces a positive synergy between the two techniques.

5.2. Study of support vector machines versions

In this part of the study, we follow the same scheme that was previously carried out. The average results are shown in Table 9 and, as in the former case, the complete table of results can be found in Appendix A of this work.

According to the results presented in Table 9, we may conclude that the preprocessing approaches perform better than the remaining proposals. We first check for significant differences using an Iman–Davenport test, which obtains a *p*-value (5.25259E−36) below our level of significance and near to zero. The associated statistical study is developed in Table 10, where we show the p-values computed by a Shaffer test with which we compare every SVM version using the AUC measure. In Fig. 7 the average ranking obtained through Friedman's test for these versions displayed, in which we can observe that the best rankings correspond to preprocessing approaches whereas worst rankings coincide with the hybrid approaches.

Table 10 shows that the original SVM is outperformed by the two preprocessing versions whereas there are not significant differences to the rest of versions. This means that the preprocessing approach improves the base classifier, however, the cost-sensitive learning proposal for SVMs is not competitive enough to be able to state that there are statistical differences. The hybridizations also cannot exceed the base classifier.

Comparing the results of preprocessing datasets we can see that the performance of these methods is not statistically different for

Table 10
Shaffer test for the SVM variety of algorithms using the AUC measure.

SVM	None	SMOTE	SENN	CS	Wr_SMOTE	Wr_US	Wr_SENN
None	×	−(1.364E−8)	−(4.749E−7)	=(1.0)	=(.05819)	=(.11667)	=(1.0)
SMOTE	+(1.364E−8)	×	=(1.0)	+(2.409E−7)	+(3.329E−17)	+(4.454E−16)	+(4.042E−7)
SENN	+(4.749E−7)	=(1.0)	×	+(6.167E−6)	+(6.421E−15)	+(7.094E−14)	+(9.585E−6)
CS	=(1.0)	−(2.409E−7)	−(6.167E−6)	×	+(.01792)	+(.03837)	=(1.0)
Wr_SMOTE	=(.05819)	−(3.329E−17)	−(6.421E−15)	−(.01792)	×	=(1.0)	−(.01394)
Wr_US	=(.11667)	−(4.454E−16)	−(7.094E−14)	−(.03837)	=(1.0)	×	−(.03139)
Wr_SENN	=(1.0)	−(4.042E−7)	−(9.585E−6)	=(1.0)	+(.01394)	+(.03139)	×

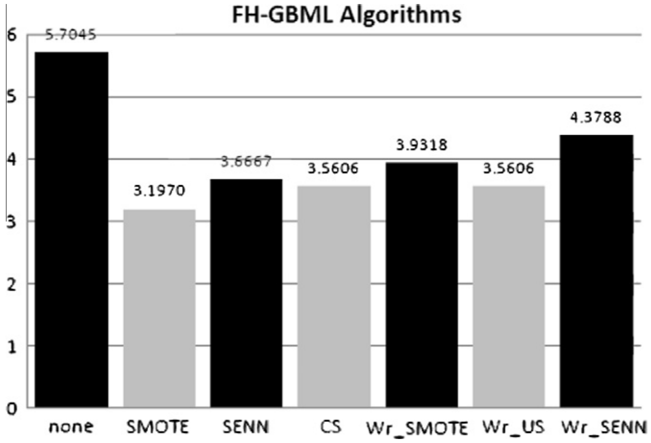

Fig. 8. Average rankings using the AUC measure for the FH-GBML variety of algorithms.

any of its versions. Nevertheless, there are significant differences between the preprocessing versions and the cost-sensitive learning version for SVMs. Furthermore, the preprocessing versions outperform statistically any other version.

If we just look at the hybridization of cost-sensitive learning and preprocessing by using a wrapper routine to check if the hybridization contributes to improve the cost-sensitive learning performance we find that there are significative differences between the different hybrid versions and between the other alternatives. The hybrid version that uses only an oversampling step with SMOTE + ENN outperforms the other hybrid versions whereas it has no significant differences with the cost-sensitive learning version. The rest of hybrids versions are not statistically different, however, they are also outperformed by the cost-sensitive version. In this paradigm, we cannot say that the hybridization produces a positive synergy between the two techniques because some of the hybrid versions are even outperformed by the cost-sensitive learning proposal.

5.3. Study of fuzzy rule based systems versions: FH-GBML

Table 11 shows the average results in training and test together with the corresponding standard deviation for the seven versions of the FH-GBML algorithm. The complete table of results for all datasets is also shown in Appendix A of this work together with the results of the previous experiments.

According to the average values shown in this table the best methods in this case are the preprocessing approaches and the cost-sensitive learning. To carry out the statistical study we first check for significant differences among the algorithms using an Iman–Davenport test. The *p*-value (8.20497E−12) is lower than our level of confidence $\alpha = 0.05$ and near to zero. Thus, we can conclude that significant differences do exist, proceeding with a Shaffer test. The ranks of the algorithms are presented in Fig. 8, and the results of the multiple comparison test performed on all algorithms are shown in Table 12.

Table 13
Average table of results using the AUC measure for the k-NN variety of algorithms.

Algorithm	AUC_{tr}	AUC_{tst}
3-NN	0.7697 ± 0.0555	0.7752 ± 0.0916
3-NN SMOTE	0.8880 ± 0.0495	0.8212 ± 0.0836
3-NN SENN	0.8743 ± 0.0434	0.8166 ± 0.0733
3-NNCS	0.8229 ± 0.0567	0.8295 ± 0.0854
3-NN Wr_SMOTE	0.8594 ± 0.0253	**0.8596 ± 0.0626**
3-NN Wr_US	0.8564 ± 0.0283	0.8561 ± 0.0655
3-NN Wr_SENN	0.8849 ± 0.0316	0.8509 ± 0.0664

At first glance we can check the tendency that we have seen in the previous studies: the base classifier is significantly different from other versions in the experimental study. Particularly, the base FH-GBML classifier is outperformed by the other versions, which means that the techniques used in imbalanced classification are useful and achieve better results than not performing special strategies to improve the results.

If we focus now on the performance of preprocessing methods we can observe that the oversampling versions are not statistically different. If we examine the preprocessing versions versus the cost-sensitive learning proposal we can see that they also do not differ statistically. With this information we can state that preprocessing and cost-sensitive learning are a good option to deal with the imbalanced classification problem.

Finally, we look at the hybridization versions from cost-sensitive learning and preprocessing. We find that between the different hybrid versions there are not statistical differences. If we compare the hybrid versions against the other versions of the study we can appreciate a difference between one of the hybrid versions and the cost-sensitive learning algorithm. Specifically, the cost-sensitive version surpasses the hybrid version that uses only an oversampling step with SMOTE + ENN. From this study, we cannot find a synergy in the hybridization.

5.4. Study of lazy learning versions: k-NN

Similar to the studies of other paradigms, we show in Table 13 the average results in training and test for the different versions of the study. We also refer the reader to the appendix for the complete table of results.

According to the average values shown in this table the best methods in this case seem to be the hybridizations of the preprocessing approaches with cost-sensitive learning. To carry out the statistical study we first check for significant differences among the algorithms using an Iman–Davenport test. The *p*-value (2.71648E−22) is lower than our level of confidence $\alpha = 0.05$ and near to zero. Thus, we can conclude that significant differences do exist, proceeding with a Shaffer test. The ranks of the algorithms are presented in Fig. 9, and the results of the multiple comparison test performed on all algorithms are shown in Table 14.

Table 12
Shaffer test for the FH-GBML variety of algorithms using the AUC measure.

FH-GBML	None	SMOTE	SENN	CS	Wr_SMOTE	Wr_US	Wr_SENN
None	×	−(5.439E−10)	−(1.078E−6)	−(2.379E−7)	−(4.128E−5)	−(2.379E−7)	−(.00676)
SMOTE	+(5.439E−10)	×	=(.64093)	=(1.0)	=(.41406)	=(1.0)	=(1.0)
SENN	+(1.078E−6)	=(.64093)	×	=(1.0)	=(1.0)	=(1.0)	=(.60824)
CS	+(2.379E−7)	=(1.0)	=(1.0)	×	=(1.0)	=(1.0)	+(.02511)
Wr_SMOTE	+(4.128E−5)	=(.41406)	=(1.0)	=(1.0)	×	=(1.0)	=(1.0)
Wr_US	+(2.379E−7)	=(1.0)	=(1.0)	=(1.0)	=(1.0)	×	=(.41406)
Wr_SENN	+(.00676)	=(1.0)	=(.60824)	−(.02511)	=(1.0)	=(.41406)	×

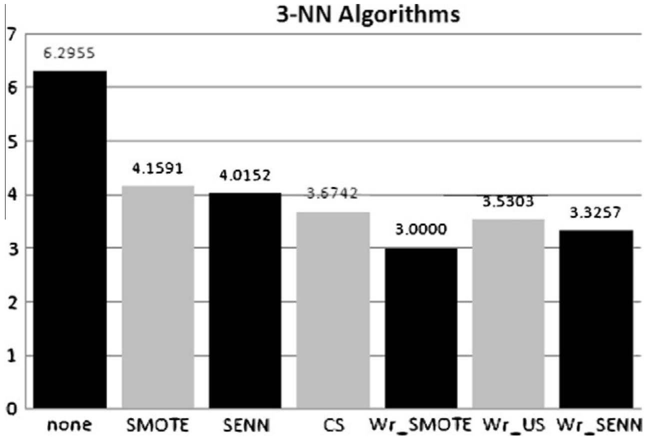

Fig. 9. Average rankings using the AUC measure for the k-NN variety of algorithms.

In this last part of experiments we confirm the tendency we pointed out after the previous statistical procedures: the base classifier is statistically different from the other versions used in the study. Using 3-NN the base classifier is outperformed by all the other algorithms in the analysis. This fact proves again that the approaches used in imbalanced classification are a need to solve these problems.

Moving to the comparison between preprocessing methods we can see that they do not differ statistically. If we broadened the comparison and we include the cost-sensitive learning proposal we still can see that there are no statistical differences. With these facts we can say that preprocessing and cost-sensitive learning are strong alternatives to solve the imbalanced classification problem.

The hybridization of cost-sensitive learning and preprocessing for 3-NN seems promising according to the average values. However, the Shaffer test does not indicate us the presence of statistical differences between the different hybrid versions. When we extend the comparison to the preprocessing and cost-sensitive learning versions we can find a difference between the base classifier combined with the SMOTE dataset and the hybrid version that uses only an oversampling step with SMOTE + ENN. Surprisingly, the difference in this case is in favor of the hybrid version. Due to these facts, for 3-NN we cannot say that there is no synergy between preprocessing and cost-sensitive learning; however, this improvement is so small that gets outshined by its bad results in the other paradigms.

5.5. General analysis on the suitability of preprocessing vs. cost-sensitive learning

As summary of the experimental study, and to unify the different conclusions extracted through the analysis of the results from the different selected paradigms, in this subsection we discuss the results we can discern attending to the three different issues we wanted to deal with: the first one devoted to demonstrate the goodness of both approaches for enhancing the performance of standard learning algorithms on this scenario, the second one for contrasting the behaviour of both preprocessing and cost-sensitive between them and the third part where a hybrid approach combining the two approaches is studied.

Before addressing the general conclusions we want to emphasize an idea about the generalization of these experiments: we cannot extrapolate the behaviour of a version from one paradigm to another. This fact has been critical in the hybrids models where a hybrid version was put at the same level of the base classifier in a paradigm whereas the same hybrid version outperformed a preprocessing approach in another paradigm.

Focusing now on the questions of the study, regarding the first issue, it is straightforward that classification performance is degraded in an imbalance scenario having a bias towards the majority class examples and that the use of the aforementioned techniques allow us to obtain a better discrimination of the examples of both classes resulting in an overall good classification for all concepts of the problem (positive and negative classes).

The second part of the study has reflected that the two employed solutions are quite similar between them and it was not possible to highlight one of them as the most adequate one for no one of the different type of algorithms (paradigms) selected for this study. Therefore, the question on which approach is preferable for addressing classification with imbalanced datasets is still unresolved.

Finally, the last approach differs from our expectations on a positive synergy. In most cases, the preliminary versions of hybridization techniques do not show a good behaviour in contrast to standard preprocessing and cost-sensitive learning. Nevertheless, some work on the combination of preprocessing and cost-sensitive learning can still be addressed with more specific methods that enhance the behaviour of these approaches.

6. Analyzing the limitations of both preprocessing and cost-sensitive learning in imbalanced classification. Open problems related to data intrinsic characteristics

According to the conclusions extracted in the previous section, we should focus on the nature of the problem itself in order to detect why both type of techniques obtain a comparable quality of solutions and how to address the imbalance problem in a more reasonable way. In this section we look at two data intrinsic characteristics issues, class overlapping and dataset shift, and their influence on imbalanced classification.

In Section 6.1 we will discuss some results about the influence of the imbalance ratio over the classification process and its relationship with the class overlap regions. Then, in Section 6.2 we will talk about the class overlapping problem and how it increases the difficulty to solve imbalanced classification problems. Finally, Section 6.3 will present the dataset shift problem and its relationship to imbalanced datasets classification.

6.1. On the influence of the imbalance ratio and its relationship with the class overlap regions

As we have stated previously, in real world machine learning applications, it has often been reported that the class imbalance

Table 14
Shaffer test for the k-NN variety of algorithms using the AUC measure.

3-NN	None	SMOTE	SENN	CS	Wr_SMOTE	Wr_US	Wr_SENN
None	×	−(2.142E−7)	−(2.260E−8)	−(5.690E−11)	−(3.981E−17)	−(3.679E−12)	−(5.711E−14)
SMOTE	+(2.142E−7)	×	=(1.0)	=(1.0)	−(.03081)	=(.80278)	=(.34698)
SENN	+(2.260E−8)	=(1.0)	×	=(1.0)	=(.09722)	=(1.0)	=(.80119)
CS	+(5.690E−11)	=(1.0)	=(1.0)	×	=(.94508)	=(1.0)	=(1.0)
Wr_SMOTE	+(3.981E−17)	+(.03081)	=(.09722)	=(.94508)	×	=(1.0)	=(1.0)
Wr_US	+(3.679E−12)	=(.80278)	=(1.0)	=(1.0)	=(1.0)	×	=(1.0)
Wr_SENN	+(5.711E−14)	=(.34698)	=(.80119)	=(1.0)	=(1.0)	=(1.0)	×

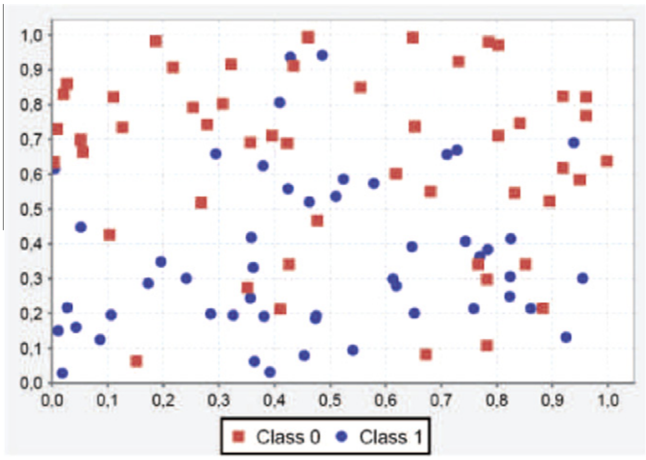

Fig. 10. *F*1 = 0.6994.

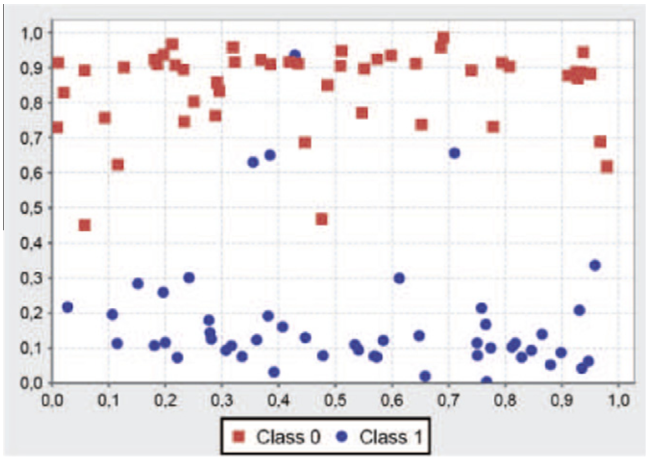

Fig. 11. *F*1 = 9.69.

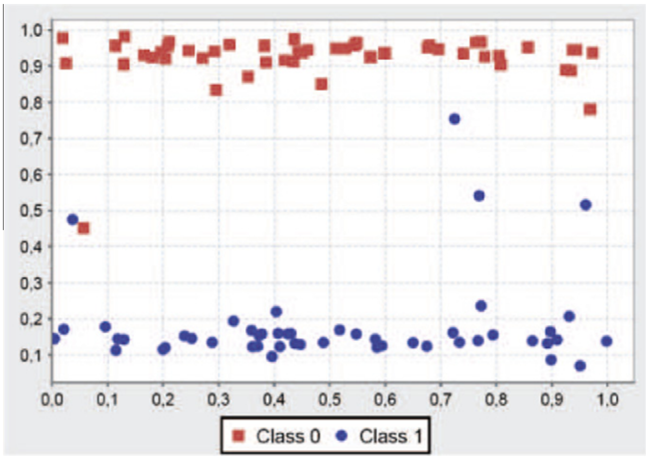

Fig. 12. *F*1 = 26.16.

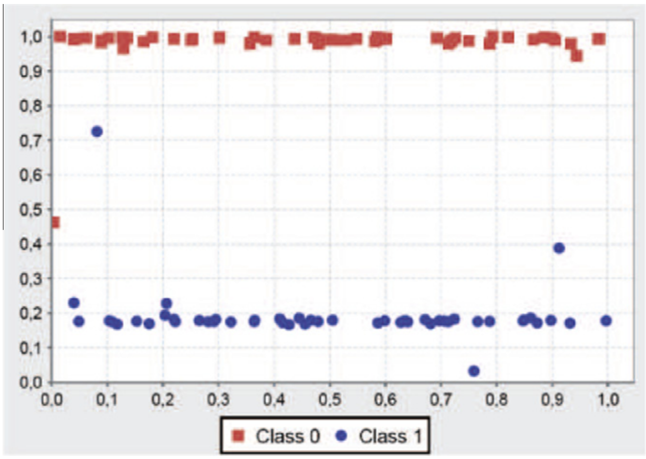

Fig. 13. *F*1 = 48.65.

hinders the performance of some standard classifiers. However, the relationship between class imbalance and learning algorithms is not clear yet, and a good understanding of how each one affects the other is lacking. Japkowicz and Stephen (2002) state that "Linearly separable domains are not sensitive to any amount of imbalance. As a matter of fact, as the degree of concept complexity increases, so does the system's sensitivity to imbalance". Thus, it does not seem fair to directly correlate class imbalance to the loss of performance of learning algorithms. Rather, it is quite possible that beyond class imbalances yield certain conditions that hamper classifiers induction.

As a direct result, there is a need to check whether class imbalances are truly to blame for the loss of performance of learning systems or whether the class imbalances are not a problem by themselves. Prati and Batista (2004) develop a study where they try to find an explanation for this performance decrease. Their experiments suggest that the problem is not solely caused by class imbalanced, but it is also related to the degree of data overlapping among the classes. They propose several experiments with synthetic datasets varying the IR and the overlap existing between the two classes. From them, it is deduced that it is not the class probabilities the main responsible for the hinder in the classification performance, but instead the degree of overlapping between the classes. This class overlapping may have a role even more important to concept induction than class imbalance. Thus, dealing with class imbalances will not always help classifiers performance improvement.

García et al. (2008) also develop a study focusing on the relationship between the IR and the overlap class regions. They studied the performance of several algorithms in different situations of imbalance and overlap focusing in the k-NN algorithm. For their study, they also use a set of synthetic datasets to check the relationship of these problems in several different situations. On the one hand, they try to find the relation when the IR in the overlap region is similar to the overall IR whereas on the other hand, they search for the relation when the IR in the overlap region is inverse to the overall one (the positive class is locally denser than the negative class in the overlap region). This first situation concludes that the increase in overlapping of a homogeneous imbalance affects more the (overall) minority class. Furthermore, the more local schemes tend to be better at classifying the minority class whereas models based on a more global learning are more robust at classifying the majority class. The second situation produces results where the accuracy on positive class is improved whereas negative class produces almost-stable accuracy curves. This example reveals that when the overlapped data is not balanced, the IR in overlapping can be more important than the overlapping size. In addition, classifiers based on more global learning attain greater TP rates whereas more local learning models obtain better TN rates than the former. This complementarity between global and local classifiers suggest a direction for future works on learning from imbalance data which will be discussed in Section 6.2.

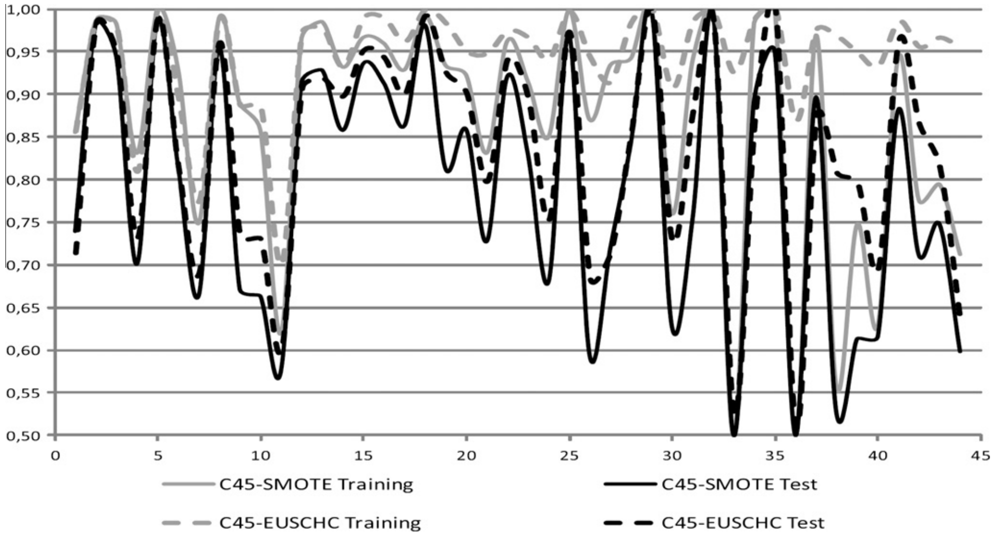

Fig. 14. Performance in training and test for the C4.5 decision tree with SMOTE sorted using the IR.

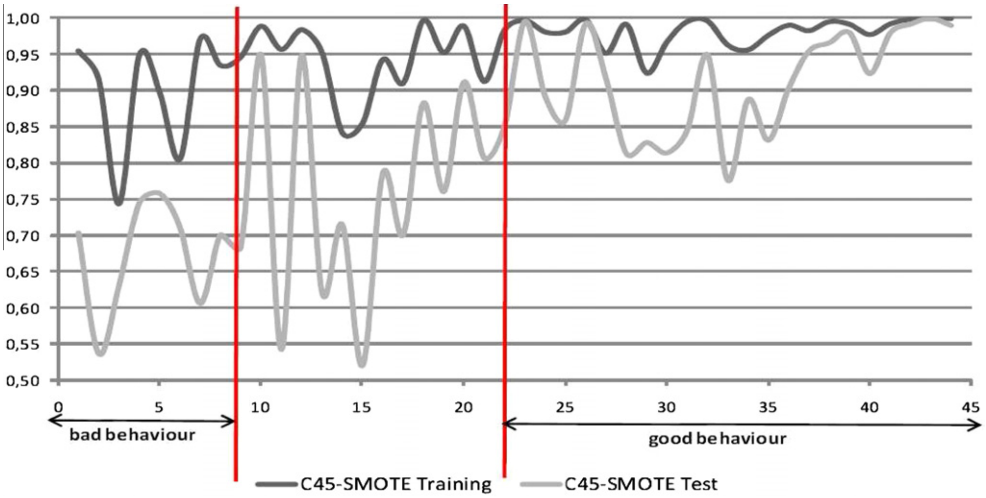

Fig. 15. Performance in training and test for the C4.5 decision tree with SMOTE sorted using the F1 data complexity metric.

Similarly, Denil and Trappenberg (2010) examine the effects of overlap and imbalance on the complexity of the learned model and demonstrate that overlap is a far more serious factor than imbalance in this respect. They demonstrate that these two problems acting in concert cause difficulties that are more severe than one would expect by examining their effects in isolation. In order to do so, they also use synthetic datasets for classifying with a SVM where they vary the IR, the overlap between classes and the IR and overlap jointly. Their results show that when the training set size is small, high levels of imbalance cause a dramatic drop in classifier performance, explained by the presence of small disjuncts. Overlapping classes cause a consistent drop in performance regardless of the size of the training set. However, with overlap and imbalance combined the classifier performance is degraded significantly beyond what the model predicts.

On the other hand, there exist recent works which have shown empirically with real world datasets that the quality of the results has not a clear relationship with the IR. Specifically, in Luengo et al. (2011) the authors try to characterize this datasets using complexity measures, which capture different aspects or sources of complexity which are considered complicated to the classification task. Specifically, they use the so called metric *F*1 or *maximum Fisher's discriminant ratio* (Ho & Basu, 2002) which measures the overlap of individual feature values. This data complexity metric, for one feature dimension, is defined as:

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{13}$$

where μ_1, μ_2, σ_1 y σ_2 are the means and variances of the two classes, respectively, in that feature dimension. We compute f for each feature and take the maximum as measure F1. For a multidimensional problem, not all features have to contribute to class discrimination. The problem is easy as long as there exists one discriminating feature. Therefore, we can just take the maximum f over all feature dimensions in discussing class separability. Small values indicate that the classes have a high overlapping degree. Figs. 10–13 show illustrative examples of artificially generated data with two variables in the range [0.0; 1.0] and two classes as example, similar to those used in the studies from García et al. (2008) or Denil and Trappenberg (2010).

In Luengo et al. (2011) the authors depicted the performance of the different datasets ordered according to the imbalanced ratio and the F1 measure in order to search for some regions of interesting good or bad behaviour. In Fig. 14 we can observe that the good and bad results of both learning methods with respect to the preprocessing are not related with the IR value, nor the improvements achieved with preprocessing steps. However, if the datasets are ordered using the F1 measure depicted in Fig. 15 both good and bad behavior intervals can be extracted, indicated by vertical lines. Therefore, the IR is not good enough to predict a classifier behavior and we need to focus on other problems to achieve better performance.

6.2. Addressing the significance of overlapping for imbalanced datasets

According to the studies previously presented, we observe the necessity to focus our efforts on the research for solutions in the imbalanced scenario towards the problem of overlapping between classes, without discarding in any case the issue of data distribution.

Our aim, given the current studies on the topic, is to address the overlapping problem integrating measures to deal with imbalance, opening many ways for future work. Therefore, following the approaches for imbalanced classification we can find several paths to improve the performance.

If we look at approaches at the **algorithm-level** we try to find algorithms that can show complementarity between global and local classifiers as suggested by García et al. (2008). A recently emerging solution to class imbalance is through the use of "information granulation". This high level procedure takes a less literal interpretation of data: instead of viewing a training sample as a set of example points, this type of classification tries to first establish higher level concepts via the construction of information granules. Kaburlasos (2007) propose a method that uses Fuzzy ART (Carpenter, Grossberg, & Rosen, 1991) to select a level of granularity. Based on these results, data is represented and a traditional learner is used. Fuzzy ART at its core is a clustering (unsupervised) system and this approach may be viewed as an additional feature transformation phase prior to classification. Chen et al. (2008) apply a similar framework, although k-means clustering is used to determine information granules instead of Fuzzy Art.

Regarding FRBCSs, Fernández, del Jesus, and Herrera (2009) proposed to make use of a Hierarchical FRBCS, which consists in the application of a thicker granularity in order to generate the initial rule base, and to reinforce those problem subspaces that are specially difficult by means of the application of rules with a higher granularity. Also, in Gama (2004) the author uses a framework of decision trees which allows to, for those leaves which have difficulties to discriminate between examples of different classes, to apply a strong classifier (for example an SVM or any other technique) in order to obtain a better separability in this specific area of the problem, rather than just using a standard heuristic.

Therefore, a very positive approach at the algorithm-level could consist in working with different granular levels, in a way that more general submodels of knowledge could cover the largest part of the problem space, whereas in more difficult areas, that is, boundary zones with a high degree of overlapping, we could use more specific discrimination functions in different paradigms of learning algorithms.

If we now turn a look at preprocessing approaches at the **data-level** we have in mind a double objective: try to find algorithms that can balance the data distribution whereas trying to avoid overlap as much as possible.

In oversampling techniques, and specially for the SMOTE algorithm, the problem of over generalization is largely attributed to the way in which it creates synthetic samples. Specifically, SMOTE generates the same number of synthetic data samples for each original minority example and does so without consideration to neighboring examples, which increases the occurrence of overlapping between classes (Wang & Japkowicz, 2004). To this end, various adaptive sampling methods have been proposed to overcome this limitation; some representative works include the Borderline-SMOTE (Han, Wang, & Mao, 2005), Adaptive Synthetic Sampling (He, Bai, Garcia, & Li, 2008) and the Safe-Level-SMOTE (Bunkhumpornpat, Sinapiromsaran, & Lursinsap, 2009) algorithms. In Fig. 16 we can observe the generation of new instances using an over-sampling algorithm. It defines three kind of instances according to its neighbors: "Safe" instances that can be used to generate synthetic samples, "Danger" instances that can be used to generate synthetic samples but can introduce overlap between the two classes and "Noise" instances that should not be used to generate instances as they are considered wrongly labelled instances.

Also, some combination of preprocessing of instances with data cleaning techniques could lead to diminish the overlapping that is introduced from sampling methods. Some representative work in this area includes the one-sided selection method Kubat and Matwin (1997), the condensed nearest neighbor rule and Tomek Links integration method Batista et al. (2004), the neighborhood cleaning rule Laurikkala (2001) based on the edited nearest neighbor (ENN) rule which removes examples that differ from two of its three nearest neighbors, and the integrations of SMOTE with ENN and SMOTE with Tomek links Batista et al. (2004) (Fig. 17).

In this manner, applying new ways of informed preprocessing techniques in order to identify and weight significant samples and discard noisy examples in the boundary areas could be an interesting topic for future work for both relaxing overlapping

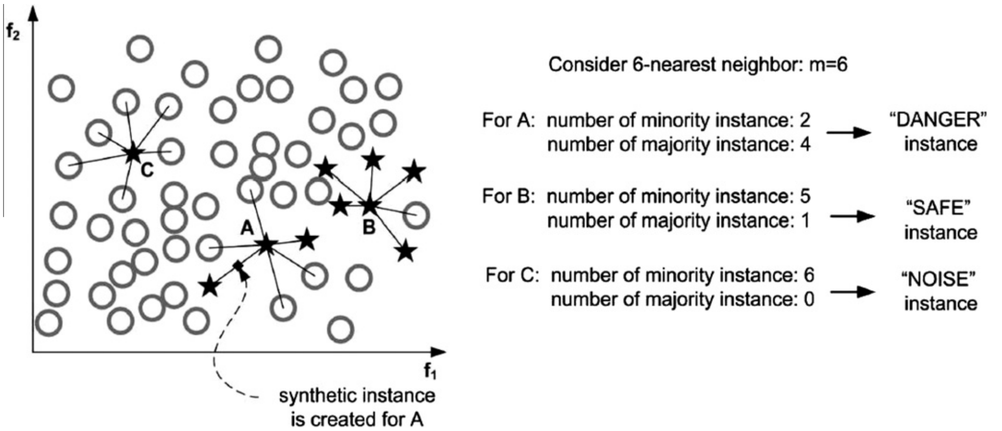
f2
f1
C
A
B
synthetic instance
is created for A
Consider 6-nearest neighbor: m=6
For A: number of minority instance: 2
number of majority instance: 4
"DANGER"
instance
For B: number of minority instance: 5
number of majority instance: 1
"SAFE"
instance
For C: number of minority instance: 6
number of majority instance: 0
"NOISE"
instance

Fig. 16. Data creation based on Borderline instance.

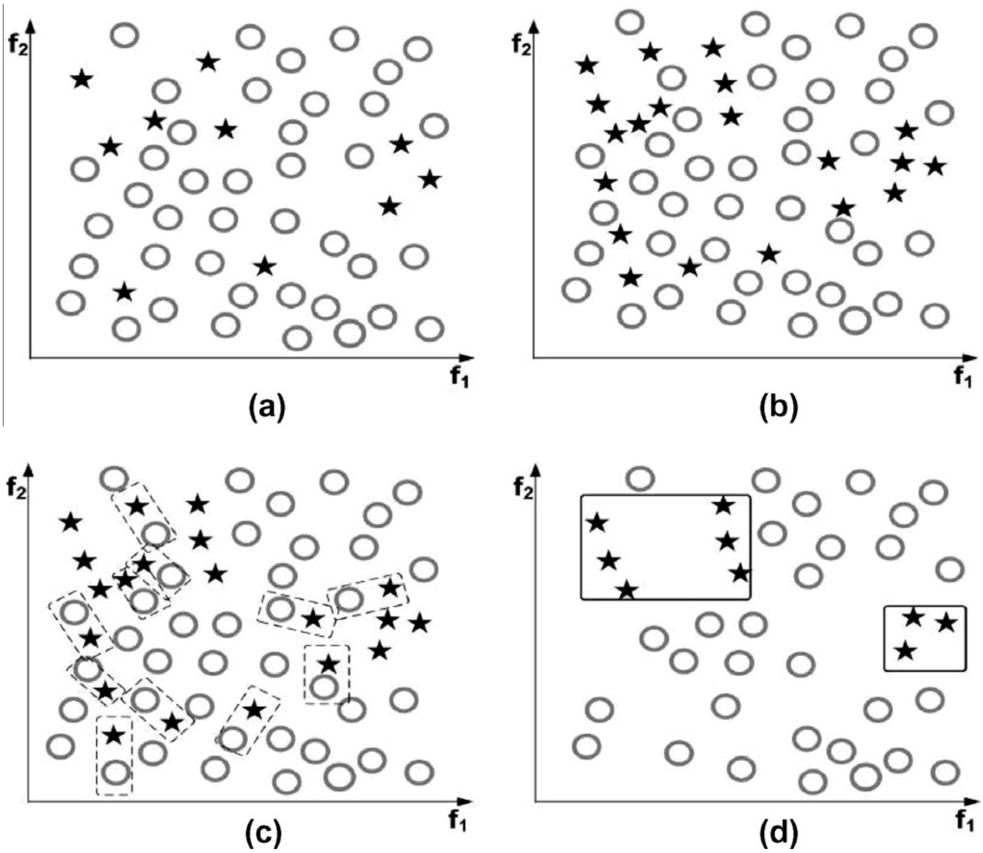

Fig. 17. (a) Original dataset distribution, (b) post-SMOTE dataset, (c) the identified Tomek links, and (d) the dataset after removing Tomek links.

and rebalancing the data distribution for avoiding the bias towards the majority class examples during the learning stage.

Still in the preprocessing approach, Martín-Félez and Mollineda (2010) propose an approach that combines preprocessing and feature selection (specifically in this order), in a way that preprocessing deals with data distribution and small disjuncts ("IR part") and feature selection somehow reduces the degree of overlapping ("F1 part"). In a more general way, the idea behind this approach consists on overcoming different sources of data complexity such as class overlap, irrelevant and redundant features, noisy samples, class imbalance, low ratios of the sample size to dimensionality and so on using different approaches used to solve each complexity.

Also, Villar, Fernández, Sánchez, and Herrera (2009, 2010) use a FRBCS that performs an *a priori* learning of the data base to obtain the optimal number of variables and granularity level for the fuzzy partitions in an imbalance scenario. The authors combine preprocessing (SMOTE in this case) with the former technique with very good results in performance (in contrast with C4.5) with a reduction of about the 65% of the variables for high imbalanced problems.

In summary, in order to reduce the original overlapping of a problem, we may apply a feature selection process in order to remove those instances which do not introduce any relevant information but makes hard to obtain discrimination functions for a given dataset.

6.3. Dataset shift in imbalanced classification

The problem of dataset shift (Alaiz-Rodríguez & Japkowicz, 2008; Shimodaira, 2000; Quiñonero Candela, Sugiyama, Schwaighofer, & Lawrence, 2009) is defined as the case where training and test data follow different distributions. This is a common problem that can affect all kind of classification problems, and it often appears due to sample selection bias issues. A mild degree of dataset shift is present in most real-world problems, but general classifiers are often capable of handling it without a severe performance loss.

However, the dataset shift issue is specially relevant when dealing with imbalanced classification, because in highly imbalanced domains, the minority class is particularly sensitive to singular classification errors, due to the typically low number of examples it presents (Moreno-Torres & Herrera, 2010). In the most extreme cases, a single misclassified example of the minority class can create a significant drop in performance.

Fig. 18 presents an example of dataset shift in imbalanced classification for clarity. Note how, in the test set, some of the minority class examples are in an area where there was none in the training set, creating a situation that is likely to produce low classifier performance.

Since dataset shift is a highly relevant issue in imbalanced classification, it is easy to see why it would be an interesting perspective to focus on in future research regarding the topic.

There are two different potential approaches in the study of the effect and solution of dataset shift in imbalanced domains. The first one focuses on intrinsic dataset shift, that is, the data of interest includes some degree of shift that is producing a relevant drop in performance. In this case, we need to:

- Develop techniques to discover and measure the presence of dataset shift following the suggestions made in (Cieslak & Chawla, 2009; Wang et al., 2003; Yang, Wu, & Zhu, 2008); but adapting them to focus on the minority class. To do so, either a partially labeled test set will be needed (in order to properly identify the minority class examples in the test set), or a new strategy will have to be developed.

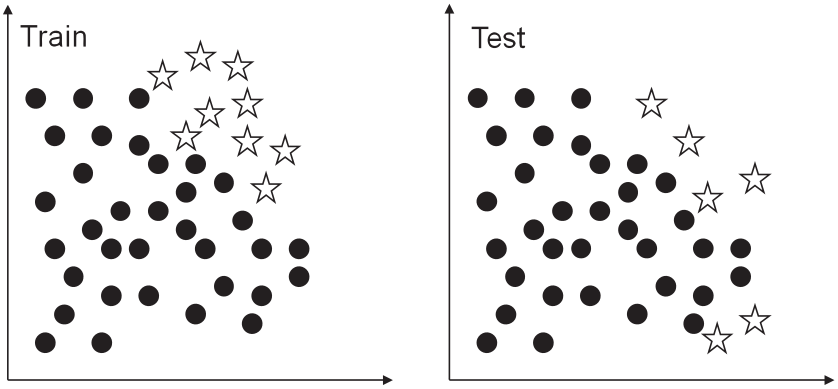

Fig. 18. Example of the impact of dataset shift in imbalanced domains.

- Design algorithms that are capable of working under dataset shift conditions. These could be either preprocessing techniques (Moreno-Torres, Llorà, Goldberg, & Bhargava, in press) or algorithms that are designed to have the capability to adapt and deal with dataset shift without the need for a preprocessing step (Alaiz-Rodríguez, Guerrero-Curieses, & Cid-Sueiro, 2009; Bickel, Brückner, & Scheffer, 2009; Globerson, Teo, Smola, & Roweis, 2009; Gretton et al., 2009; Sugiyama, Krauledat, & Müller, 2007). In both cases, we are not aware of any proposals in the literature that focus on the problem of imbalanced classification in the presence of dataset shift.

The second branch in terms of dataset shift in imbalanced classification is related to induced dataset shift. Most current state of the art research is validated through stratified cross-validation techniques, which are another potential source of shift in the machine learning process. A more suitable validation technique needs to be developed in order to avoid introducing dataset shift issues artificially.

7. Concluding remarks

In this work we have analyzed the preprocessing performance in the framework of imbalanced datasets against other approaches in this problem such as cost-sensitive learning. We have considered two oversampling methods: SMOTE and SMOTE + ENN, a cost-sensitive version and a hybrid approach that tries to integrate both approaches together.

We have observed that the approaches used to address the imbalanced problem improve the overall performance in all the paradigms used in the study, which was the expected behaviour.

The comparison between preprocessing techniques against cost-sensitive learning hints that there are no differences among the different preprocessing techniques. The statistical study carried out let us say that both preprocessing and cost-sensitive learning are good and equivalent approaches to address the imbalance problem.

The preliminary versions of hybridization techniques are truly competitive with the standard methodologies only in some cases, which determines more work needs to be done in addressing this approach.

Finally, we develop a discussion about how to go above preprocessing and cost-sensitive learning limits. We try to analyze the problem according to the results and we focus on the open problems to improve the algorithms. Specifically, we have emphasized that the IR is important but there are still other issues like the class overlapping and dataset shift problems that arise in some cases and can prove detrimental in terms of classification performance. Since overcoming these problems is the key to the improvement of the algorithms' performance, future work should be oriented to analyze the existing overlap to create accurate algorithms that can improve in imbalanced classification and to use dataset shift repairing techniques to fill the gap between data distributions.

Acknowledgments

This work has been supported by the Spanish Ministry of Education and Science under Project TIN2008-06681-C06-01. V. López and J.G. Moreno-Torres hold FPU scholarships from Spanish Ministry of Education.

Appendix A. Detailed results for the experimental study

In this appendix we present the complete results tables for all the algorithms used in this work. Thus, the reader can observe the full training and test results, in order to compare the performance of each approach. In Table A.15 we show the

Table A.15
Complete table of results using the AUC measure for the C4.5 variety of algorithms.

C4.5	None		SMOTE		SENN		CS		Wr_SMOTE		Wr_US		Wr_SENN	
Dataset	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst
Glass1	0.8561	**0.7399**	0.9234	0.7368	0.8690	0.6921	0.9069	0.7160	0.9069	0.7160	0.8831	0.6682	0.8595	0.7367
Ecoli0vs1	0.9870	**0.9832**	0.9926	0.9729	0.9870	**0.9832**	0.9870	**0.9832**	0.9870	**0.9832**	0.9800	**0.9832**	0.9870	**0.9832**
Wisconsin	0.9840	0.9454	0.9826	0.9532	0.9776	0.9576	0.9780	**0.9636**	0.9780	**0.9636**	0.9768	0.9555	0.9755	0.9524
Pima	0.8317	0.7012	0.8179	0.7245	0.8012	**0.7403**	0.8571	0.7125	0.8571	0.7125	0.8621	0.7311	0.8203	0.7077
Iris0	1.0000	**0.9900**	1.0000	**0.9900**	1.0000	**0.9900**	1.0000	**0.9900**	1.0000	**0.9900**	1.0000	**0.9900**	1.0000	**0.9900**
Glass0	0.9306	0.8167	0.9459	0.7752	0.8897	0.7994	0.9205	**0.8212**	0.9205	**0.8212**	0.9100	0.8042	0.8636	0.7999
Yeast1	0.7494	0.6642	0.8085	**0.7090**	0.7829	0.6954	0.7855	0.6779	0.7855	0.6779	0.7806	0.6767	0.8023	0.6945
Vehicle1	0.8898	0.6717	0.9503	0.7301	0.8817	**0.7542**	0.9362	0.7013	0.9362	0.7013	0.9276	0.7130	0.8173	0.6719

Table A.15 (*continued*)

C4.5	None		SMOTE		SENN		CS		Wr_SMOTE		Wr_US		Wr_SENN	
Dataset	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst
Vehicle2	0.9905	**0.9561**	0.9905	0.9498	0.9848	0.9419	0.9866	0.9434	0.9866	0.9434	0.9850	0.9450	0.9811	0.9419
Vehicle3	0.8588	0.6637	0.9508	0.7282	0.8914	**0.7409**	0.9221	0.7283	0.9221	0.7283	0.9275	0.7010	0.8569	0.6791
Haberman	0.6204	0.5757	0.7124	**0.6163**	0.7383	0.5884	0.6380	0.5752	0.6380	0.5752	0.5879	0.5476	0.6417	0.5396
Glass0123vs456	0.9671	0.9155	0.9868	0.9232	0.9635	**0.9240**	0.9806	0.8777	0.9806	0.8777	0.9783	0.8931	0.9513	0.9108
Vehicle0	0.9863	0.9296	0.9878	0.9188	0.9724	0.9072	0.9861	0.9289	0.9861	0.9289	0.9799	**0.9373**	0.9752	0.9178
Ecoli1	0.9329	0.8586	0.9507	0.9105	0.9335	0.8926	0.9457	**0.9114**	0.9457	**0.9114**	0.9394	0.9017	0.9368	0.9065
New-thyroid2	0.9679	0.9373	0.9922	0.9659	0.9817	0.9774	0.9903	**0.9802**	0.9903	**0.9802**	0.9868	0.9437	0.9744	0.9063
New-thyroid1	0.9607	0.9143	0.9879	0.9631	0.9944	**0.9889**	0.9903	0.9746	0.9903	0.9746	0.9882	0.9746	0.9774	0.9405
Ecoli2	0.9297	0.8641	0.9738	0.8811	0.9716	**0.8976**	0.9594	0.8905	0.9594	0.8905	0.9515	0.8641	0.9473	0.8580
Segment0	0.9932	0.9826	0.9986	**0.9927**	0.9989	0.9916	0.9988	0.9919	0.9988	0.9919	0.9967	0.9894	0.9940	0.9876
Glass6	0.9347	0.8132	0.9872	0.8842	0.9851	0.9203	0.9865	0.8896	0.9865	0.8896	0.9878	0.8923	0.9369	**0.9365**
Yeast3	0.9237	0.8597	0.9607	0.8905	0.9617	**0.9230**	0.9784	0.9117	0.9784	0.9117	0.9796	0.9096	0.9587	0.9176
Ecoli3	0.8320	0.7280	0.9671	0.8123	0.9371	**0.8705**	0.9585	0.8326	0.9585	0.8326	0.9605	0.8452	0.9133	0.8694
Page-blocks0	0.9637	0.9221	0.9848	**0.9504**	0.9797	0.9427	0.9903	0.9458	0.9903	0.9458	0.9894	0.9435	0.9614	0.9284
Ecoli034vs5	0.9188	0.8389	0.9854	0.9000	0.9764	0.8806	0.9938	**0.9250**	0.9938	**0.9250**	0.9896	0.8972	0.9694	0.9111
Yeast2vs4	0.9158	0.8307	0.9814	0.8588	0.9746	**0.9042**	0.9797	0.8866	0.9797	0.8866	0.9768	0.8955	0.9323	0.8291
Ecoli067vs35	0.8789	0.8250	0.9781	0.8500	0.9775	0.8125	0.9875	0.8825	0.9875	0.8825	0.9869	0.8775	0.9201	**0.8875**
Ecoli0234vs5	0.9313	0.8307	0.9897	**0.8974**	0.9828	0.8947	0.9966	0.8334	0.9966	0.8334	0.9835	0.7946	0.9730	0.8835
Glass015vs2	0.8910	0.5011	0.9766	0.6772	0.9083	**0.7957**	0.9790	0.6003	0.9790	0.6003	0.9758	0.5938	0.8727	0.5508
Yeast0359vs78	0.7028	0.5868	0.9490	**0.7047**	0.9217	0.7024	0.9715	0.6765	0.9715	0.6765	0.9556	0.6721	0.8362	0.6641
Yeast02579vs368	0.8809	0.8432	0.9767	**0.9143**	0.9576	0.9138	0.9874	0.8996	0.9874	0.8996	0.9855	0.8896	0.9533	0.9102
Yeast0256vs3789	0.7563	0.6606	0.9330	**0.7951**	0.9179	0.7817	0.9743	0.7846	0.9743	0.7846	0.9435	0.7403	0.8906	0.7648
Ecoli046vs5	0.9368	0.8418	0.9870	0.8701	0.9836	**0.8869**	0.9911	0.8310	0.9911	0.8310	0.9884	0.8174	0.9543	0.7978
Ecoli01vs235	0.9097	0.7136	0.9656	**0.8377**	0.9650	0.8332	0.9739	0.7641	0.9739	0.7641	0.9727	0.7664	0.9263	0.7532
Ecoli0267vs35	0.8788	0.7752	0.9796	0.8155	0.9827	0.8179	0.9889	0.8527	0.9889	0.8527	0.9852	**0.8653**	0.9067	0.8577
Glass04vs5	0.9940	**0.9941**	0.9910	0.9816	0.9910	0.9754	0.9940	**0.9941**	0.9940	**0.9941**	0.9940	**0.9941**	0.9940	**0.9941**
Ecoli0346vs5	0.9118	0.8615	0.9892	**0.8980**	0.9885	**0.8980**	0.9905	0.8507	0.9905	0.8507	0.9905	0.8534	0.9579	0.7730
Ecoli0347vs56	0.8600	0.7757	0.9778	**0.8568**	0.9568	0.8546	0.9892	0.7586	0.9898	0.7764	0.9806	0.7985	0.9384	0.8100
Yeast05679vs4	0.8508	0.6802	0.9526	0.7602	0.9199	0.7802	0.9741	0.7243	0.9741	0.7243	0.9691	0.7480	0.9134	**0.7804**
Ecoli067vs5	0.9363	0.7675	0.9875	0.8475	0.9744	0.8450	0.9888	**0.8825**	0.9888	**0.8825**	0.9869	0.8775	0.9081	0.8600
Vowel0	0.9999	**0.9706**	0.9971	0.9505	0.9943	0.9455	0.9925	0.9422	0.9925	0.9422	0.9928	0.9311	0.9928	0.9322
Glass016vs2	0.8710	0.5938	0.9716	0.6062	0.9375	**0.6388**	0.9829	0.6155	0.9829	0.6155	0.9807	0.5793	0.8529	0.5788
Glass2	0.9350	0.7194	0.9700	0.6390	0.9280	**0.7457**	0.9734	0.6416	0.9734	0.6416	0.9639	0.6715	0.8669	0.6501
Ecoli0147vs2356	0.8578	0.8051	0.9789	0.8277	0.9565	0.8228	0.9882	0.8772	0.9882	0.8772	0.9866	**0.8788**	0.9112	0.7673
Led7digit02456789vs1	0.9022	0.8788	0.9225	**0.8908**	0.9249	0.8379	0.9203	0.8436	0.9203	0.8436	0.9178	0.8387	0.9042	0.8616
Glass06vs5	0.9950	**0.9950**	0.9912	0.9147	0.9912	0.9647	0.9950	**0.9950**	0.9950	**0.9950**	0.9637	0.9579	0.9950	**0.9950**
Ecoli01vs5	0.9114	0.8159	0.9886	0.7977	0.9830	0.8250	0.9778	0.8182	0.9778	0.8182	0.9858	**0.8318**	0.9392	0.8136
Glass0146vs2	0.7879	0.6616	0.9676	**0.7842**	0.9042	0.7095	0.9847	0.6797	0.9847	0.6797	0.9708	0.6421	0.7930	0.6102
Ecoli0147vs56	0.8842	0.8318	0.9798	**0.8592**	0.9610	0.8424	0.9756	0.8539	0.9756	0.8539	0.9813	0.8371	0.9468	0.7774
Cleveland0vs4	0.8648	0.6878	0.9939	**0.7908**	0.9816	0.7605	0.9886	0.6893	0.9906	0.6823	0.9914	0.6885	0.9086	0.6795
Ecoli0146vs5	0.9178	0.7885	0.9870	**0.8981**	0.9851	**0.8981**	0.9808	0.8385	0.9808	0.8385	0.9837	0.8135	0.9572	0.8212
Ecoli4	0.9430	0.8437	0.9703	0.7794	0.9827	**0.9044**	0.9680	0.8636	0.9680	0.8636	0.9684	0.8636	0.9505	0.8386
Yeast1vs7	0.7608	0.6275	0.9351	0.7003	0.9097	**0.7371**	0.9741	0.6139	0.9741	0.6139	0.9671	0.6794	0.8530	0.6627
Shuttle0vs4	1.0000	0.9997	0.9999	0.9997	0.9999	0.9997	1.0000	0.9997	1.0000	0.9997	1.0000	**1.0000**	1.0000	**1.0000**
Glass4	0.9403	0.7542	0.9901	**0.8867**	0.9670	0.8650	0.9104	0.8431	0.9104	0.8431	0.9340	0.8298	0.8861	0.7831
Page-blocks13vs2	0.9989	0.9978	0.9975	0.9955	0.9975	0.9910	0.9989	**0.9789**	0.9989	**0.9789**	0.9977	0.9978	0.9791	0.9498
Abalone9vs18	0.6907	0.5859	0.9142	0.6283	0.9058	**0.7193**	0.9864	0.6655	0.9864	0.6655	0.9849	0.6369	0.8515	0.7150
Glass016vs5	0.9843	0.8943	0.9921	0.8129	0.9864	0.8629	0.9914	**0.9886**	0.9914	**0.9886**	0.9914	**0.9886**	0.9914	**0.9886**
Shuttle2vs4	1.0000	0.9500	0.9990	0.9917	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**
Yeast1458vs7	0.5000	0.5000	0.9158	0.5367	0.8770	0.5563	0.9640	0.5540	0.9640	0.5540	0.9625	**0.5464**	0.7788	0.4943
Glass5	0.9702	0.8976	0.9921	0.8805	0.9705	0.7756	0.9976	0.9427	0.9976	0.9427	0.9872	**0.9854**	0.8624	0.8439
Yeast2vs8	0.5563	0.5250	0.9071	0.8338	0.8960	0.8197	0.9927	**0.8652**	0.9927	0.8652	0.9913	0.8359	0.8541	0.7978
Yeast4	0.7482	0.6135	0.9071	0.7121	0.9007	0.7257	0.9722	0.7222	0.9722	0.7222	0.9700	0.6999	0.8872	**0.7400**
Yeast1289vs7	0.6290	0.6156	0.9465	0.6832	0.9414	0.6332	0.9752	0.6769	0.9752	0.6769	0.9748	**0.6973**	0.7073	0.6107
Yeast5	0.9453	0.8833	0.9777	0.9337	0.9820	0.9406	0.9929	0.9330	0.9929	0.9330	0.9928	0.9326	0.9743	**0.9434**
Ecoli0137vs26	0.7953	0.7481	0.9678	0.8136	0.9660	0.8136	0.9804	0.8281	0.9804	0.8281	0.9594	0.7954	0.8907	**0.8445**
Yeast6	0.7762	0.7115	0.9326	**0.8294**	0.9314	0.8270	0.9883	0.8082	0.9883	0.8082	0.9864	0.8099	0.8165	0.7311
Abalone19	0.5000	0.5000	0.8550	0.5205	0.8890	0.5166	0.9839	**0.5701**	0.9839	**0.5701**	0.9835	0.5543	0.6211	0.5231
Average	0.8774	0.7902	0.9606	0.8324	0.9471	**0.8390**	0.9679	0.8294	0.9679	0.8296	0.9635	0.8245	0.9083	0.8145

Table A.16
Complete table of results using the AUC measure for the SVM variety of algorithms.

SVM	None		SMOTE		SENN		CS		Wr_SMOTE		Wr_US		Wr_SENN	
Dataset	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst
Glass1	0.5155	0.4963	0.6613	0.6179	0.6780	0.6391	0.6624	**0.6264**	0.5000	0.5000	0.5097	0.5000	0.6229	0.5682
Ecoli0vs1	0.9675	0.9671	0.9844	**0.9796**	0.9811	0.9770	0.9675	0.9671	0.9844	**0.9796**	0.9810	0.9731	0.9828	**0.9796**
Wisconsin	0.9728	0.9666	0.9770	0.9727	0.9794	0.9691	0.9724	0.9719	0.9653	0.9552	0.9726	0.9626	0.9777	**0.9737**
Pima	0.7334	0.7194	0.7523	0.7348	0.7520	0.7300	0.7378	0.7289	0.6985	0.6916	0.6960	0.7116	0.7452	**0.7449**
Iris0	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**

(*continued on next page*)

Table A.16 (*continued*)

SVM	None		SMOTE		SENN		CS		Wr_SMOTE		Wr_US		Wr_SENN	
Dataset	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst
Glass0	0.7070	0.6914	0.7716	0.7377	0.7755	0.7243	0.5215	0.5074	0.7778	0.7517	0.7778	0.7517	0.7411	**0.7520**
Yeast1	0.5771	0.5732	0.7108	**0.7100**	0.7096	0.7067	0.6675	0.6749	0.5000	0.5000	0.5012	0.5000	0.6750	0.6897
Vehicle1	0.7510	0.7202	0.8001	0.7742	0.8184	**0.8055**	0.7930	0.7546	0.6401	0.6180	0.6147	0.6076	0.7125	0.6882
Vehicle2	0.9693	0.9527	0.9722	**0.9601**	0.9711	0.9578	0.9734	0.9571	0.9223	0.9068	0.9371	0.9070	0.9023	0.8891
Vehicle3	0.7290	0.7134	0.7805	0.7613	0.8101	0.7881	0.8072	**0.7904**	0.4789	0.4871	0.5612	0.5753	0.6339	0.6306
Haberman	0.5223	0.5036	0.6287	**0.6344**	0.6621	0.6332	0.5225	0.5382	0.5000	0.5000	0.5000	0.5000	0.5217	0.4996
Glass0123vs456	0.9151	0.9043	0.9351	**0.9050**	0.9426	0.8987	0.8572	0.8445	0.8572	0.8445	0.8672	0.8445	0.9425	0.8987
Vehicle0	0.9780	0.9490	0.9778	0.9632	0.9778	0.9611	0.9781	0.9493	0.9798	0.9620	0.9805	**0.9653**	0.9610	0.9470
Ecoli1	0.8331	0.8192	0.9082	**0.9062**	0.9006	0.9024	0.9084	**0.9062**	0.6430	0.6367	0.6523	0.6535	0.8776	0.8659
New-thyroid2	0.9972	0.9829	0.9965	**0.9917**	0.9917	0.9889	0.9972	0.9829	0.9750	0.9687	0.9802	0.9603	0.9680	0.9659
New-thyroid1	0.9972	0.9829	0.9965	**0.9944**	0.9944	0.9861	0.9943	0.9687	0.9786	0.9516	0.9901	0.9829	0.9701	0.9603
Ecoli2	0.7675	0.7351	0.9073	**0.9067**	0.9065	0.9050	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.8916	0.8805
Segment0	0.9954	0.9927	0.9987	0.9955	0.9985	**0.9965**	0.9990	**0.9965**	0.9947	0.9932	0.9946	0.9932	0.9944	0.9922
Glass6	0.9379	**0.9198**	0.9497	0.9063	0.9554	0.9009	0.8882	0.8725	0.8882	0.8725	0.8964	0.8919	0.9281	0.9032
Yeast3	0.6305	0.6299	0.9056	0.8917	0.9114	**0.9061**	0.9057	0.8951	0.5000	0.5000	0.5000	0.5000	0.5200	0.5154
Ecoli3	0.5000	0.5000	0.9037	**0.8984**	0.8964	0.8818	0.8222	0.7925	0.5000	0.5000	0.5855	0.5614	0.7267	0.6976
Page-blocks0	0.8287	0.8218	0.9251	0.9258	0.9292	**0.9273**	0.9248	0.9254	0.5001	0.5004	0.4976	0.4769	0.5738	0.5828
Ecoli034vs5	0.9153	0.8611	0.9271	**0.8889**	0.9250	0.8861	0.8750	0.8639	0.8750	0.8639	0.8847	0.8556	0.8972	**0.8889**
Yeast2vs4	0.6691	0.6691	0.9090	**0.8896**	0.9084	0.8885	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.7408	0.7360
Ecoli067vs35	0.8999	**0.8525**	0.9276	0.8325	0.9239	0.8350	0.8363	0.8025	0.7807	0.7050	0.8468	0.8300	0.8733	0.8275
Ecoli0234vs5	0.9229	0.8667	0.9302	**0.8892**	0.9205	**0.8892**	0.8813	0.8417	0.8813	0.8417	0.8834	0.8140	0.9292	0.8696
Glass015vs2	0.5000	0.5000	0.5943	0.5094	0.5961	**0.5191**	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Yeast0359vs78	0.6067	0.6067	0.7476	**0.7451**	0.7522	0.7450	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.6067	0.6067
Yeast02579vs368	0.8090	0.8006	0.9137	0.9013	0.9143	**0.9069**	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.8890	0.8981
Yeast0256vs3789	0.5524	0.5486	0.8102	0.7940	0.8098	**0.8018**	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.8034	0.8000
Ecoli046vs5	0.9028	0.8696	0.9213	**0.8869**	0.9130	**0.8869**	0.8875	0.8696	0.8875	0.8696	0.8806	0.8669	0.8966	0.8642
Ecoli01vs235	0.8863	0.8359	0.9393	0.8505	0.9420	**0.8550**	0.8429	0.7805	0.8429	0.7805	0.8796	0.8582	0.9029	0.7959
Ecoli0267vs35	0.8899	0.8526	0.9162	0.8255	0.9156	**0.8530**	0.8346	0.7851	0.8346	0.7851	0.8288	0.8251	0.8717	0.8079
Glass04vs5	0.8893	0.8500	0.9638	**0.9566**	0.9638	0.9507	0.8893	0.9000	0.8893	0.9000	0.8983	0.9129	0.8893	0.9000
Ecoli0346vs5	0.9035	0.8696	0.9191	0.8926	0.9287	0.8926	0.8688	0.8946	0.8688	0.8946	0.8743	**0.8973**	0.9279	0.8088
Ecoli0347vs56	0.9123	0.8935	0.9219	**0.9082**	0.9224	0.9061	0.8550	0.8135	0.8500	0.8135	0.8545	0.8135	0.9191	0.8848
Yeast05679vs4	0.5000	0.5000	0.8016	**0.8075**	0.7977	0.7875	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.7371	0.7135
Ecoli067vs5	0.9094	0.8425	0.9213	0.8475	0.9238	0.8075	0.8500	0.7450	0.8500	0.7450	0.8775	0.8325	0.9013	**0.9125**
Vowel0	0.9096	0.8950	0.9793	**0.9622**	0.9795	**0.9622**	0.8655	0.8461	0.9432	0.9244	0.9420	0.9172	0.9477	0.9489
Glass016vs2	0.5000	0.5000	0.6462	**0.5336**	0.6520	0.5267	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Glass2	0.5000	0.5000	0.6883	0.6155	0.6852	**0.6905**	0.7051	0.5953	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Ecoli0147vs2356	0.8635	0.8434	0.8973	**0.8828**	0.9060	0.8727	0.7801	0.7267	0.7801	0.7267	0.7882	0.7101	0.8885	0.8568
Led7digit02456789vs1	0.9051	**0.8901**	0.8981	0.8851	0.8850	0.8891	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.9027	0.8272
Glass06vs5	0.7071	0.6500	0.9520	**0.9437**	0.9520	0.9437	0.6929	0.6500	0.6929	0.6500	0.8262	0.6245	0.6929	0.6500
Ecoli01vs5	0.9273	0.8364	0.9648	0.8364	0.9608	0.8364	0.8813	0.7909	0.8813	0.7909	0.8864	0.7909	0.9403	**0.8864**
Glass0146vs2	0.5000	0.5000	0.6631	0.6121	0.6729	**0.6310**	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Ecoli0147vs56	0.9080	**0.8719**	0.9181	0.8612	0.9205	0.8546	0.8400	0.7967	0.8400	0.7967	0.8742	0.8335	0.8984	0.8519
Cleveland0vs4	0.9403	0.7483	0.9619	0.8785	0.9627	**0.9149**	0.9318	0.7483	0.9318	0.7483	0.9503	0.7483	0.8966	0.8014
Ecoli0146vs5	0.8798	0.8635	0.9269	**0.8904**	0.9404	0.8808	0.8438	0.7923	0.8438	0.7923	0.8620	0.8154	0.8865	0.8654
Ecoli4	0.5875	0.5750	0.9743	0.9200	0.9739	0.9200	0.9834	**0.9529**	0.5000	0.5000	0.5000	0.5000	0.6313	0.6000
Yeast1vs7	0.5000	0.5000	0.7746	**0.7861**	0.7664	0.7741	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Shuttle0vs4	1.0000	**1.0000**	1.0000	0.9960	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	0.9960	1.0000	**1.0000**
Glass4	0.6157	0.5592	0.9546	**0.9576**	0.9621	0.9101	0.9615	0.9126	0.6064	0.5617	0.5964	0.5592	0.7529	0.6733
Page-blocks13vs2	0.8896	0.8332	0.9654	0.9561	0.9654	**0.9640**	0.8513	0.8566	0.6777	0.7757	0.6654	0.6325	0.7104	0.6738
Abalone9vs18	0.5029	0.5000	0.8161	0.8127	0.8257	0.8128	0.8352	**0.8740**	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Glass016vs5	0.5839	0.4971	0.9536	0.9429	0.9521	**0.9457**	0.5554	0.5000	0.5554	0.5000	0.6346	0.5886	0.5825	0.5471
Shuttle2vs4	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	0.9990	0.9960	1.0000	**1.0000**
Yeast1458vs7	0.5000	0.5000	0.6926	**0.6373**	0.7032	0.6266	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Glass5	0.5554	0.5000	0.9518	0.9512	0.9488	0.9415	0.9713	**0.9732**	0.5554	0.5000	0.5554	0.5500	0.5143	0.5000
Yeast2vs8	0.7739	0.7739	0.8201	0.7663	0.8183	0.7642	0.8223	0.7664	0.5500	0.5739	0.5500	0.5739	0.7739	**0.7739**
Yeast4	0.5000	0.5000	0.8571	0.8241	0.8560	**0.8258**	0.8604	0.8155	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Yeast1289vs7	0.5000	0.5000	0.7401	**0.7194**	0.7455	0.7077	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Yeast5	0.5000	0.5000	0.9641	0.9653	0.9642	0.9628	0.9648	**0.9656**	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Ecoli0137vs26	0.8733	**0.8500**	0.9571	0.7990	0.9521	0.8044	0.8733	**0.8500**	0.8733	**0.8500**	0.8720	0.8481	0.8553	0.8463
Yeast6	0.5000	0.5000	0.8886	0.8730	0.8867	0.8696	0.8807	**0.8758**	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Abalone19	0.5000	0.5000	0.8039	**0.7930**	0.8150	0.7873	0.8170	0.7615	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Average	0.7563	0.7341	0.8806	0.8514	0.8826	**0.8517**	0.7869	0.7651	0.6981	0.6820	0.7077	0.6895	0.7656	0.7461

Table A.17
Complete table of results using the AUC measure for the FH-GBML variety of algorithms.

FH-GBML	None		SMOTE		SENN		CS		Wr_SMOTE		Wr_US		Wr_SENN	
Dataset	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst
Glass1	0.8103	0.7199	0.8194	0.7313	0.8220	0.7331	0.8270	0.7414	0.8263	0.6852	0.8278	0.6964	0.8192	**0.7540**
Ecoli0vs1	0.9958	0.9762	0.9926	0.9627	0.9837	0.9532	0.9942	**0.9765**	0.9959	0.9729	0.9928	0.9550	0.9878	0.9698
isconsin	0.9818	0.9620	0.9811	0.9638	0.9785	0.9720	0.9828	**0.9780**	0.9841	0.9704	0.9829	0.9704	0.9739	0.9507

Table A.17 (*continued*)

FH-GBML	None		SMOTE		SENN		CS		Wr_SMOTE		Wr_US		Wr_SENN	
Dataset	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst
Pima	0.7410	0.6980	0.7684	**0.7381**	0.7494	0.7061	0.7772	0.7274	0.7770	0.7235	0.7776	0.7304	0.7619	0.7321
Iris0	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	0.9950	0.9713	0.9850	1.0000	**1.0000**
Glass0	0.8246	0.7524	0.8515	0.7542	0.8325	0.7901	0.8500	0.7709	0.8565	0.8036	0.8542	0.7738	0.8342	**0.8043**
Yeast1	0.6717	0.6611	0.7310	0.7004	0.7211	0.7044	0.7353	0.7016	0.7351	0.7115	0.7346	**0.7234**	0.7272	0.6965
Vehicle1	0.6642	0.6164	0.7655	0.7106	0.7469	0.7049	0.7615	0.6982	0.7655	0.7126	0.7549	**0.7202**	0.7284	0.6902
Vehicle2	0.8257	0.8204	0.8917	0.8718	0.8812	0.8697	0.8900	0.8732	0.8932	**0.8768**	0.8859	0.8704	0.8916	0.8582
Vehicle3	0.6454	0.6121	0.7520	0.7128	0.7501	**0.7275**	0.7500	0.6942	0.7485	0.6929	0.7493	0.6966	0.7339	0.6748
Haberman	0.6284	0.5130	0.7080	0.6136	0.6851	0.6067	0.7498	0.6061	0.7547	0.6132	0.7506	**0.6141**	0.6345	0.5163
Glass0123vs456	0.9651	0.8888	0.9722	0.9307	0.9704	**0.9430**	0.9796	0.9154	0.9774	0.8474	0.9803	0.9082	0.9617	0.8954
Vehicle0	0.8618	0.8348	0.9062	0.8938	0.9007	0.8697	0.8997	0.8878	0.9194	**0.9053**	0.9131	0.9050	0.8988	0.8837
Ecoli1	0.9083	0.8480	0.9276	**0.8763**	0.9174	0.8704	0.9346	0.8659	0.9303	0.8732	0.9297	0.8731	0.9198	0.8557
New-thyroid2	0.9893	0.9546	0.9986	**0.9802**	0.9931	0.9774	1.0000	0.9516	1.0000	0.9544	0.9979	0.9659	0.9579	0.8944
New-thyroid1	0.9982	**0.9931**	1.0000	0.9516	0.9944	0.9917	1.0000	0.9659	1.0000	0.9659	0.9958	0.9405	0.9608	0.9488
Ecoli2	0.9296	0.8550	0.9538	0.8861	0.9471	**0.9369**	0.9569	0.8974	0.9564	0.9044	0.9536	0.8943	0.9428	0.9343
Segment0	0.9724	0.9709	0.9837	0.9772	0.9829	0.9741	0.9891	0.9806	0.9827	**0.9828**	0.9855	0.9802	0.9840	0.9736
Glass6	0.9656	0.9032	0.9772	0.8827	0.9743	0.8298	0.9854	0.8384	0.9949	0.8605	0.9899	0.8771	0.9509	**0.9252**
Yeast3	0.8673	0.8321	0.9432	0.9293	0.9362	0.9165	0.9447	0.9076	0.9419	0.9212	0.9424	**0.9298**	0.9341	0.9089
Ecoli3	0.8240	0.7674	0.9405	0.8847	0.9443	0.8787	0.9516	**0.8864**	0.9554	0.8502	0.9524	0.8772	0.9222	0.8283
Page-blocks0	0.8170	0.8116	0.9012	0.8938	0.8939	0.8983	0.9028	0.8944	0.9003	0.9017	0.8996	**0.9023**	0.8927	0.8868
Ecoli034vs5	0.9743	0.8569	0.9865	0.8944	0.9865	0.8444	0.9997	**0.9125**	1.0000	0.8236	0.9979	0.8861	0.9597	0.8972
Yeast2vs4	0.8859	0.8328	0.9442	0.9073	0.9504	0.8972	0.9626	0.8931	0.9610	0.9056	0.9606	**0.9196**	0.9019	0.8809
Ecoli067vs35	0.9324	0.8575	0.9458	0.8125	0.9539	**0.8750**	0.9828	0.8188	0.9831	0.8075	0.9863	0.8375	0.9036	0.8350
Ecoli0234vs5	0.9688	0.8890	0.9856	0.8572	0.9769	0.8434	0.9993	0.8059	0.9979	0.8696	0.9903	0.8227	0.9501	**0.9306**
Glass015vs2	0.5886	0.4887	0.8709	0.6008	0.8576	**0.7204**	0.9246	0.6481	0.9267	0.6191	0.9141	0.7167	0.7967	0.6013
Yeast0359vs78	0.6100	0.5889	0.7995	0.7226	0.7977	0.7351	0.8204	**0.7573**	0.8234	0.7030	0.8262	0.6879	0.7895	0.7004
Yeast02579vs368	0.8998	0.8619	0.9248	0.9099	0.9232	0.8938	0.9330	0.9001	0.9325	0.8982	0.9311	**0.9071**	0.9270	0.9029
Yeast0256vs3789	0.7259	0.6911	0.8283	0.7851	0.8252	0.7942	0.8374	0.7945	0.8388	0.7818	0.8359	**0.7970**	0.8226	0.7778
Ecoli046vs5	0.9688	0.8973	0.9877	0.8326	0.9829	0.8061	0.9986	**0.9669**	0.9973	0.8142	0.9963	0.8669	0.9682	0.9337
Ecoli01vs235	0.9407	0.7882	0.9693	0.8075	0.9625	**0.8482**	0.9781	0.7955	0.9804	0.8409	0.9794	0.8320	0.9276	0.7900
Ecoli0267vs35	0.9314	**0.8551**	0.9599	0.8331	0.9479	0.7991	0.9864	0.8315	0.9842	0.8103	0.9855	0.8303	0.9326	0.8216
Glass04vs5	1.0000	0.8441	0.9868	**0.9673**	0.9925	0.8574	1.0000	0.9199	1.0000	0.9375	0.9895	0.7195	0.9687	0.8188
Ecoli0346vs5	0.9556	0.7946	0.9823	0.8331	0.9872	0.9142	0.9986	0.8919	0.9990	0.8669	0.9926	0.9061	0.9627	**0.9223**
Ecoli0347vs56	0.9339	0.8357	0.9663	0.8600	0.9608	0.8525	0.9855	0.8320	0.9847	0.8737	0.9844	0.8731	0.9423	**0.8792**
Yeast05679vs4	0.7084	0.6514	0.8559	**0.8064**	0.8456	0.7312	0.8690	0.7703	0.8665	0.7842	0.8693	0.7832	0.8476	0.7782
Ecoli067vs5	0.9375	0.8613	0.9600	0.8338	0.9656	0.8750	0.9903	0.8613	0.9897	0.8863	0.9869	0.8150	0.9050	**0.9125**
Vowel0	0.8924	0.8256	0.9661	**0.9561**	0.9565	0.9135	0.9663	0.9394	0.9630	0.9352	0.9563	0.9352	0.9521	0.9466
Glass016vs2	0.5727	0.5233	0.8671	0.6343	0.8498	**0.6895**	0.9046	0.6636	0.8973	0.5976	0.8912	0.5860	0.8092	0.5400
Glass2	0.5659	0.4885	0.8603	0.6771	0.8210	0.5991	0.8972	0.7098	0.9050	**0.8172**	0.8957	0.5978	0.7961	0.6106
Ecoli0147vs2356	0.8934	0.7936	0.9467	0.8508	0.9489	0.8457	0.9651	0.8622	0.9624	0.8077	0.9607	**0.8792**	0.8995	0.8043
Led7digit02456789vs1	0.9069	**0.8938**	0.9235	0.8839	0.9039	0.8900	0.9440	0.8745	0.9454	0.8741	0.9459	0.8666	0.9079	0.8823
Glass06vs5	1.0000	0.8925	0.9859	0.9320	0.9862	0.8925	1.0000	0.9100	1.0000	0.8747	0.9975	0.8950	0.9756	**0.9374**
Ecoli01vs5	0.9750	0.8648	0.9892	**0.8989**	0.9835	0.8864	0.9994	0.8432	1.0000	0.8875	0.9966	0.8886	0.9543	0.8693
Glass0146vs2	0.5368	0.4961	0.8510	0.7064	0.8352	0.6345	0.9111	**0.7618**	0.8996	0.6367	0.8947	0.6756	0.8079	0.7020
Ecoli0147vs56	0.9296	0.8667	0.9669	0.8045	0.9648	0.8605	0.9862	**0.8955**	0.9888	0.8388	0.9866	0.8596	0.9561	0.8820
Cleveland0vs4	0.9219	0.6939	0.9431	0.7520	0.9317	0.7056	0.9832	0.6861	0.9798	0.6348	0.9829	**0.7876**	0.9519	0.7541
Ecoli0146vs5	0.9495	0.7913	0.9786	**0.9202**	0.9856	0.8750	0.9990	0.8529	0.9983	0.7808	0.9962	0.9000	0.9418	0.8231
Ecoli4	0.9563	0.8703	0.9876	0.9302	0.9858	0.9294	0.9972	**0.9421**	0.9968	0.8873	0.9972	0.8905	0.9484	0.8913
Yeast1vs7	0.6786	0.5358	0.8396	0.7191	0.8543	0.6424	0.8673	**0.7389**	0.8773	0.7026	0.8724	0.6655	0.8012	0.6882
Shuttle0vs4	1.0000	0.9960	1.0000	0.9980	1.0000	**1.0000**	1.0000	0.9920	1.0000	0.9958	1.0000	**1.0000**	1.0000	0.9958
Glass4	0.9021	0.6479	0.9775	0.8867	0.9657	**0.9613**	0.9969	0.8746	0.9963	0.7505	0.9957	0.8684	0.9259	0.6868
Page-blocks13vs2	0.9375	0.9272	0.9866	0.9515	0.9882	0.9459	0.9958	0.9749	0.9949	**0.9787**	0.9959	0.9498	0.9532	0.9142
Abalone9vs18	0.6085	0.5912	0.7917	0.7165	0.7979	0.7376	0.8440	0.7737	0.8308	0.7774	0.8346	0.7797	0.7972	**0.7948**
Glass016vs5	0.9107	0.8136	0.9752	**0.8993**	0.9768	0.8921	0.9993	0.8193	1.0000	0.8443	0.9975	0.8300	0.9486	0.8964
Shuttle2vs4	1.0000	0.9500	1.0000	0.9940	1.0000	0.9877	1.0000	**1.0000**	1.0000	0.8500	1.0000	0.9500	0.9200	0.8500
Yeast1458vs7	0.5333	0.4985	0.7761	0.6287	0.7620	**0.6597**	0.8021	0.6319	0.7925	0.6370	0.7955	0.6237	0.7385	0.5822
Glass5	0.8797	0.8201	0.9899	0.7671	0.9848	0.7970	0.9988	0.8841	0.9994	0.7427	0.9976	**0.9201**	0.9636	0.8165
Yeast2vs8	0.8125	0.7478	0.8723	0.7442	0.8555	0.7226	0.8877	0.7411	0.8916	0.7839	0.8892	**0.8180**	0.8196	0.7076
Yeast4	0.5659	0.5167	0.8806	0.8137	0.8785	0.7947	0.8945	**0.8222**	0.8962	0.8027	0.8898	0.8214	0.8261	0.7394
Yeast1289vs7	0.6250	0.5820	0.8096	**0.7238**	0.7943	0.7175	0.8425	0.6393	0.8369	0.7076	0.8457	0.6441	0.6868	0.5299
Yeast5	0.7206	0.6783	0.9735	0.9469	0.9796	**0.9778**	0.9885	0.9740	0.9875	0.9314	0.9861	0.9396	0.9575	0.8958
Ecoli0137vs26	0.8767	0.7472	0.9824	0.8236	0.9820	0.8208	0.9991	0.7891	0.9989	0.8363	0.9966	**0.8445**	0.8544	0.7982
Yeast6	0.6243	0.6270	0.9204	0.8646	0.9215	0.8591	0.9296	0.8426	0.9317	**0.8713**	0.9302	0.8300	0.8716	0.8302
Abalone19	0.5000	0.5000	0.8322	0.6708	0.8250	**0.7297**	0.8387	0.6627	0.8493	0.6816	0.8321	0.6914	0.6293	0.5726
Average	0.8352	0.7692	0.9181	0.8364	0.9127	0.8350	0.9328	**0.8373**	0.9330	0.8244	0.9304	0.8322	0.8866	0.8168

Table A.18
Complete table of results using the AUC measure for the k-NN variety of algorithms.

3-NN	None		SMOTE		SENN		CS		Wr_SMOTE		Wr_US		Wr_SENN	
Dataset	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst
Glass1	0.7583	0.7460	0.8273	0.7805	0.8398	0.7761	0.7583	0.7460	0.7583	0.7460	0.7567	0.7350	0.8593	**0.8147**
Ecoli0vs1	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.9691	0.9600	0.9690	**0.9766**	0.9707	0.9533

(*continued on next page*)

Table A.18 (*continued*)

3-NN	None		SMOTE		SENN		CS		Wr_SMOTE		Wr_US		Wr_SENN	
Dataset	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst	Tr	Tst
Wisconsin	0.9636	0.9658	0.0214	0.9698	0.0209	**0.9729**	0.9636	0.9658	0.9636	0.9658	0.9641	0.9658	0.9647	0.9658
Pima	0.6686	0.6703	0.7479	0.6865	0.7682	0.7099	0.6686	0.6703	0.6686	0.6703	0.6696	0.6711	0.7986	**0.7297**
Iris0	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	1.0000	**1.0000**	0.9988	**1.0000**	1.0000	**1.0000**
Glass0	0.8144	0.8027	0.8184	0.8185	0.8299	**0.8361**	0.7884	0.7771	0.7884	0.7771	0.7529	0.7670	0.8025	0.7809
Yeast1	0.6559	0.6539	0.7864	0.6772	0.7734	0.7071	0.6740	0.6858	0.6740	0.6858	0.6745	0.6841	0.7772	**0.7190**
Vehicle1	0.6741	0.6314	0.8454	0.6985	0.8230	**0.7752**	0.7665	0.7476	0.7665	0.7476	0.7664	0.7474	0.7781	0.7472
Vehicle2	0.9743	**0.9736**	0.9753	0.9692	0.9690	0.9620	0.9578	0.9541	0.9578	0.9541	0.8942	0.8960	0.9635	0.9519
Vehicle3	0.6395	0.6529	0.8539	0.7085	0.8291	**0.7636**	0.7365	0.7355	0.7365	0.7355	0.7367	0.7355	0.7416	0.7474
Haberman	0.5463	0.5310	0.6955	0.5633	0.6906	0.5767	0.6167	0.6510	0.6167	0.6510	0.6100	**0.6516**	0.6546	0.5729
Glass0123vs456	0.8859	0.8888	0.9709	0.9164	0.9620	0.9334	0.9424	0.9331	0.9424	0.9331	0.9338	**0.9399**	0.9407	0.9199
Vehicle0	0.9446	0.9379	0.9548	0.9471	0.9493	0.9415	0.9473	0.9461	0.9473	0.9461	0.9371	0.9363	0.9535	**0.9479**
Ecoli1	0.7693	0.7636	0.8484	0.8085	0.8345	0.8089	0.8019	0.8036	0.8789	0.8749	0.8721	0.8730	0.9165	**0.9065**
New-thyroid2	0.9508	0.9373	0.9889	0.9889	0.9875	0.9861	0.9831	**0.9917**	0.9831	**0.9917**	0.9854	0.9833	0.9688	0.9516
New-thyroid1	0.9401	0.9659	0.9917	0.9889	0.9889	0.9861	0.9831	**0.9917**	0.9831	**0.9917**	0.9818	0.9806	0.9816	0.9631
Ecoli2	0.8253	0.8302	0.8674	0.8382	0.8622	0.8276	0.8307	0.8276	0.9102	0.9154	0.9082	0.9066	0.9396	**0.9294**
Segment0	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.9970	**0.9980**	0.9941	0.9937	0.9977	0.9962
Glass6	0.9147	0.9140	0.9824	**0.9419**	0.9770	0.9338	0.9366	**0.9419**	0.9366	**0.9419**	0.9305	0.9365	0.9286	**0.9419**
Yeast3	0.8231	0.8171	0.9541	0.8681	0.9470	0.8634	0.8827	0.8777	0.8827	0.8777	0.8859	0.8803	0.9300	**0.8824**
Ecoli3	0.6798	0.6598	0.8514	0.7283	0.8443	0.7772	0.7766	0.7502	0.8432	0.8209	0.8428	0.8478	0.8932	**0.8863**
Page-blocks0	0.9040	0.9075	0.9796	0.9326	0.9748	0.9316	0.9409	**0.9370**	0.9409	0.9370	0.9409	**0.9370**	0.9530	0.9193
Ecoli034vs5	0.8306	0.8222	0.8813	0.8222	0.8813	0.8222	0.8556	0.8361	0.9389	**0.9333**	0.9389	0.9222	0.9535	0.9167
Yeast2vs4	0.7485	0.7368	0.8573	0.8073	0.8554	0.8073	0.7903	0.7938	0.8680	0.8771	0.8677	0.8771	0.9128	**0.8803**
Ecoli067vs35	0.7109	0.7625	0.8531	0.8200	0.8525	0.8150	0.7724	0.8550	0.8563	**0.8900**	0.8623	0.8800	0.8627	0.8275
Ecoli0234vs5	0.8125	0.8500	0.8746	0.8530	0.8746	0.8530	0.8328	0.8612	0.9294	0.9308	0.9329	0.9280	0.9446	**0.9336**
Glass015vs2	0.5943	0.5788	0.8637	0.6750	0.8637	0.6935	0.7036	**0.7097**	0.7036	**0.7097**	0.7073	0.6685	0.7225	0.6067
Yeast0359vs78	0.6522	0.6468	0.8736	**0.7247**	0.8733	0.7203	0.6762	0.6923	0.6762	0.6923	0.6852	0.6979	0.8005	0.6977
Yeast02579vs368	0.8845	0.8834	0.9507	0.9024	0.9506	0.9013	0.8970	0.8988	0.8970	0.8988	0.8852	0.8922	0.9269	**0.9082**
Yeast0256vs3789	0.7580	0.7658	0.9066	0.7728	0.9066	0.7655	0.8096	**0.7916**	0.8096	**0.7916**	0.8087	0.7861	0.8407	0.7836
Ecoli046vs5	0.9111	0.9250	0.9781	0.9282	0.9754	0.9282	0.9328	0.9365	0.9328	0.9365	0.9172	**0.9450**	0.9335	0.9392
Ecoli01vs235	0.7733	0.7700	0.8705	0.7936	0.8705	0.7936	0.8099	0.7850	0.8926	0.8827	0.8973	**0.8959**	0.9181	0.8564
Ecoli0267vs35	0.7263	0.7725	0.8629	0.8401	0.8604	0.8327	0.7745	0.8026	0.8516	0.8526	0.8491	**0.8677**	0.8566	0.8150
Glass04vs5	0.8702	0.9441	0.9412	0.9632	0.9397	0.9511	0.9789	**0.9941**	0.9789	**0.9941**	0.9517	0.9761	0.9789	**0.9941**
Ecoli0346vs5	0.8368	0.8000	0.8791	0.8169	0.8791	0.8169	0.8434	0.8419	0.9282	0.9419	0.9096	0.9095	0.9329	**0.9446**
Ecoli0347vs56	0.7925	0.7735	0.8623	0.7920	0.5000	0.5000	0.8263	0.8363	0.9109	0.9119	0.9089	0.9212	0.9317	**0.9227**
Yeast05679vs4	0.6288	0.6257	0.8954	0.7440	0.8978	0.7682	0.7443	**0.7968**	0.7443	**0.7968**	0.7383	0.7915	0.8557	0.7825
Ecoli067vs5	0.8031	0.8225	0.9500	0.8375	0.9456	0.8250	0.8769	0.8675	0.8769	0.8675	0.8750	**0.8825**	0.8863	0.8600
Vowel0	0.9915	0.9939	0.9999	**0.9994**	0.9999	**0.9994**	0.9975	**0.9994**	0.9975	**0.9994**	0.9808	0.9800	0.9975	**0.9994**
Glass016vs2	0.5629	0.6357	0.8800	0.7169	0.8771	0.6445	0.7477	**0.7893**	0.7477	**0.7893**	0.7640	0.7864	0.7982	0.7560
Glass2	0.5474	0.5302	0.9150	0.7162	0.8984	**0.7717**	0.6969	0.6954	0.6969	0.6954	0.7254	0.7334	0.8470	0.6733
Ecoli0147vs2356	0.7838	0.7968	0.8605	0.7959	0.8609	0.7959	0.8160	0.8272	0.8969	0.9057	0.8907	0.9041	0.9262	**0.9170**
Led7digit02456789vs1	0.7696	0.7747	0.8618	0.8215	0.8642	0.8465	0.8261	0.8297	0.8261	0.8297	0.8311	0.8223	0.9018	**0.8639**
Glass06vs5	0.8725	0.9500	0.9786	0.9847	0.9786	0.9847	0.9240	**1.0000**	0.9240	**1.0000**	0.9205	0.9400	0.9383	**1.0000**
Ecoli01vs5	0.8932	0.9000	0.9739	0.9023	0.9733	0.9023	0.9216	0.9136	0.9216	0.9136	0.9239	0.9068	0.9312	**0.9159**
Glass0146vs2	0.5302	0.5727	0.8903	0.7019	0.8923	0.7018	0.6940	**0.7567**	0.6940	**0.7567**	0.7339	0.7458	0.7404	0.6447
Ecoli0147vs56	0.8793	0.8551	0.9666	0.9139	0.9601	0.9025	0.9221	0.9189	0.9221	0.9189	0.9238	0.9156	0.9340	**0.9254**
Cleveland0vs4	0.7726	0.7136	0.9320	0.8346	0.9320	0.8346	0.8487	**0.8584**	0.8487	**0.8584**	0.8448	0.8553	0.8727	0.8583
Ecoli0146vs5	0.9058	0.9231	0.9740	0.9019	0.9745	0.9000	0.9168	0.9135	0.9168	0.9135	0.9159	**0.9250**	0.9197	0.9192
Ecoli4	0.8238	0.7734	0.8865	0.8421	0.8846	0.8108	0.8366	0.8187	0.9163	0.9155	0.9217	0.9107	0.9281	**0.9202**
Yeast1vs7	0.6153	0.6109	0.8802	0.7390	0.8811	0.6998	0.7170	**0.7453**	0.7170	**0.7453**	0.7175	0.7406	0.8039	0.6177
Shuttle0vs4	0.9959	0.9960	1.0000	0.9960	1.0000	0.9960	0.9959	0.9960	0.9959	0.9960	0.9959	**1.0000**	0.9959	0.9960
Glass4	0.7628	0.8425	0.9689	0.8917	0.9627	**0.9151**	0.8885	0.8868	0.8885	0.8868	0.8835	0.8868	0.8912	0.8843
Page-blocks13vs2	0.9724	0.9433	0.9963	0.9978	0.9963	**0.9989**	0.9963	0.9977	0.9963	0.9977	0.9859	0.9888	0.9972	0.9977
Abalone9vs18	0.5987	0.6332	0.9099	0.7525	0.9023	0.7416	0.6990	**0.7637**	0.7998	0.7334	0.8097	0.7408	0.7117	0.6482
Glass016vs5	0.9121	0.8971	0.9686	0.9271	0.9664	0.9186	0.9871	**0.9857**	0.9871	**0.9857**	0.9757	0.9686	0.9850	**0.9857**
Shuttle2vs4	0.8750	0.9500	0.9959	**1.0000**	0.9959	**1.0000**	0.9600	0.9500	0.9600	0.9500	0.9078	0.9140	0.9600	0.9500
Yeast1458vs7	0.5163	0.5144	0.8852	**0.6944**	0.8812	0.6929	0.6249	0.6609	0.6249	0.6609	0.6228	0.6654	0.6719	0.5729
Glass5	0.8439	0.8976	0.9780	0.9378	0.9689	**0.9732**	0.9717	0.9329	0.9717	0.9329	0.9799	0.9256	0.9580	0.9280
east2vs8	0.7236	0.7239	0.9656	0.7208	0.9608	0.7371	0.7930	**0.8012**	0.7930	**0.8012**	0.7846	**0.8012**	0.8131	0.7631
Yeast4	0.5966	0.5947	0.9594	0.7444	0.9520	0.7571	0.7281	0.7489	0.7281	0.7489	0.7279	0.7489	0.8787	**0.7708**
Yeast1289vs7	0.5520	0.5484	0.9185	0.6586	0.9170	**0.6764**	0.6677	0.6462	0.6677	0.6462	0.6671	0.6629	0.7135	0.6154
Yeast5	0.8056	0.8128	0.9836	0.9503	0.9812	**0.9566**	0.9357	0.9424	0.9357	0.9424	0.9394	0.9389	0.9530	0.9174
Ecoli0137vs26	0.7730	0.7982	0.8680	0.7691	0.5000	0.5000	0.7607	0.7800	0.8361	0.8281	0.8293	0.8244	0.8516	**0.8445**
Yeast6	0.7570	0.7527	0.9720	0.8442	0.9676	0.8540	0.8145	0.8368	0.8145	0.8368	0.8198	0.8497	0.8890	**0.8678**
Abalone19	0.4998	0.4998	0.9780	0.5216	0.9737	0.5205	0.5402	0.5184	0.7576	0.5193	0.7573	**0.5357**	0.6215	0.5114
Average	0.7697	0.7752	0.8880	0.8212	0.8743	0.8166	0.8229	0.8295	0.8594	**0.8596**	0.8564	0.8561	0.8849	0.8509

results for the C4.5 algorithm versions. Next, the results for the SVM versions used in the study are shown in Table A.16. Later, the results for the FH-GBML algorithm versions are presented in Table A.17. Finally, Table A.18 show the average results for each dataset for the 3-NN algorithm. We stress in **boldface** the best results achieved by a version.

References

Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In *ECML* (pp. 39–50).

Alaiz-Rodríguez, R., Guerrero-Curieses, A., & Cid-Sueiro, J. (2009). Improving classification under changes in class and within-class distributions. In *IWANN'09: Proceedings of the 10th international work-*

conference on artificial neural networks (pp. 122–130). Berlin, Heidelberg: Springer.

Alaiz-Rodríguez, R., & Japkowicz, N. (2008). Assessing the impact of changing environments on classifier performance. In *Canadian AI'08: Proceedings of the Canadian society for computational studies of intelligence, 21st conference on advances in artificial intelligence* (pp. 13–24). Berlin, Heidelberg: Springer-Verlag.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., et al. (2011). KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multi-Valued Logic and Soft Computing*, *17*, 255–287.

Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M. J., Ventura, S., Garrell, J. M., et al. (2008). Keel: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, *13*, 307–318.

Barandela, R., Sánchez, J. S., García, V., & Rangel, E. (2003). Strategies for learning in class imbalance problems. *Pattern Recognition, 36*, 849–851.

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behaviour of several methods for balancing machine learning training data. *SIGKDD Explorations, 6*, 20–29.

Bermejo, P., Gámez, J., & Puerta, J. (2011). Improving the performance of Naive Bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets. *Expert Systems with Applications, 38*, 2072–2080.

Bickel, S., Brückner, M., & Scheffer, T. (2009). Discriminative learning under covariate shift. *Journal of Machine Learning Research, 10*, 2137–2155.

Bradford, J. P., Kunz, C., Kohavi, R., Brunk, C., & Brodley, C. E. (1998). Pruning decision trees with misclassification costs. In *ECML* (pp. 131–136).

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition, 30*, 1145–1159.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009). Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Proceedings of the 13th Pacific-Asia conference on advances in knowledge discovery and data mining PAKDD'09* (pp. 475–482).

Quiñonero Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2009). *Dataset shift in machine learning*. The MIT Press.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991). Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks, 4*, 759–771.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligent Research, 16*, 321–357.

Chawla, N. V., Cieslak, D. A., Hall, L. O., & Joshi, A. (2008). Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery, 17*, 225–252.

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explorations, 6*, 1–6.

Chen, M.-C., Chen, L.-S., Hsu, C.-C., & Zeng, W.-R. (2008). An information granulation based data mining approach for classifying imbalanced data. *Information Sciences, 178*, 3214–3227.

Chen, X., Fang, T., Huo, H., & Li, D. (2011). Graph-based feature selection for object-oriented classification in vhr airborne imagery. *IEEE Transactions on Geoscience and Remote Sensing, 49*, 353–365.

Cieslak, D. A., & Chawla, N. V. (2009). A framework for monitoring classifiers performance: When and why failure occurs? *Knowledge and Information Systems, 18*, 83–108.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning, 20*, 273–297.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*, 21–27.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Denil, M., & Trappenberg, T. (2010). Overlap versus imbalance. In *Canadian AI 2010, LNAI* (Vol. 6085, pp. 220–231).

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the 5th international conference on knowledge discovery and data mining (KDD'99)* (pp. 155–164).

Ducange, P., Lazzerini, B., & Marcelloni, F. (2010). Multi-objective genetic fuzzy classifiers for imbalanced and cost-sensitive datasets. *Soft Computing, 14*, 713–728.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the 17th IEEE international joint conference on artificial intelligence (IJCAI'01)* (pp. 973–978).

Estabrooks, A., Jo, T., & Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence, 20*, 18–36.

Fernández, A., García, S., del Jesus, M. J., & Herrera, F. (2008). A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. *Fuzzy Sets and Systems, 159*, 2378–2398.

Fernández, A., del Jesus, M. J., & Herrera, F. (2009). Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. *International Journal of Approximate Reasoning, 50*, 561–577.

Fernández, A., del Jesús, M. J., & Herrera, F. (2010). On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets. *Information Sciences, 180*, 1268–1291.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*, 675–701.

Gama, J. (2004). Functional trees. *Machine Learning, 55*, 219–250.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Computing, 13*, 959–977.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences, 180*, 2044–2064.

García, S., & Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research, 9*, 2607–2624.

García, V., Mollineda, R., & Sánchez, J. S. (2008). On the k-NN performance in a challenging scenario of imbalance and overlapping. *Pattern Analysis Applications, 11*, 269–280.

Globerson, A., Teo, C. H., Smola, A., & Roweis, S. (2009). An adversarial view of covariate shift and a minimax approach. In J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, & N. D. Lawrence (Eds.), *Dataset shift in machine learning* (pp. 179–198). The MIT Press.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., & Schölkopf, B. (2009). Covariate shift by kernel mean matching. In J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, & N. D. Lawrence (Eds.), *Dataset shift in machine learning* (pp. 131–160). The MIT Press.

Han, H., Wang, W., & Mao, B. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *Proceedings of the international conference on intelligent computing* (pp. 878–887).

Hand, D. J., & Vinciotti, V. (2003). Choosing k for two-class nearest neighbour classifiers with unbalanced classes. *Pattern Recognition Letters, 24*, 1555–1562.

He, H., Bai, Y., Garcia, E., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of international joint conference on neural networks* (pp. 1322–1328).

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*, 1263–1284.

Ho, T., & Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*, 289–300.

Huang, J., & Ling, C. X. (2005). Using auc and accuracy in evaluating learning algorithms – appendices. *IEEE Transactions on Knowledge and Data Engineering*, 17.

Ishibuchi, H., & Nakashima, T. (2001). Effect of rule weights in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems, 9*, 506–515.

Ishibuchi, H., & Yamamoto, T. (2005). Rule weight specification in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems, 13*, 428–435.

Ishibuchi, H., Yamamoto, T., & Nakashima, T. (2005). Hybridization of fuzzy GBML approaches for pattern classification problems. *IEEE Transactions on System, Man and Cybernetics B, 35*, 359–365.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal, 6*, 429–450.

Kaburlasos, V. G. (2007). Granular enhancement of fuzzy art/som neural classifiers based on lattice theory. In *Computational intelligence based on lattice theory* (pp. 3–23).

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *International conference on machine learning* (pp. 179–186).

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Proceedings of the conference on AI in medicine in Europe: Artificial intelligence medicine* (pp. 63–66).

Ling, C. X., Yang, Q., Wang, J., & Zhang, S. (2004). Decision trees with minimal costs. In *ICML*.

López, V., Fernández, A., & Herrera, F. (2010). A first approach for cost-sensitive classification with linguistic genetic fuzzy systems in imbalanced data-sets. In *10th International conference on intelligent systems design and applications, ISDA 2010* (pp. 676–681).

Luengo, J., Fernández, A., García, S., & Herrera, F. (2011). Addressing data complexity for imbalanced data sets: Analysis of SMOTE-based oversampling and evolutionary undersampling. *Soft Computing, 15*, 1909–1936.

Martín-Félez, R., & Mollineda, R. (2010). On the suitability of combining feature selection and resampling to manage data complexity. In *CAEPIA 2009, LNAI* (Vol. 5988, pp. 141–150).

Moreno-Torres, J. G., & Herrera, F. (2010). A preliminary study on overlapping and data fracture in imbalanced domains by means of genetic programming-based feature extraction. In *Proceedings of the 10th international conference on intelligent systems design and applications (ISDA 2010)* (pp. 501–506).

Moreno-Torres, J. G., Llorà, X., Goldberg, D. E., & Bhargava, R. (in press). Repairing fractures between data using genetic programming-based feature extraction: A case study in cancer diagnosis. *Information Sciences*, doi:10.1016/j.ins.2010.09.018.

Orriols-Puig, A., & Bernadó-Mansilla, E. (2009). Evolutionary rule-based systems for imbalanced datasets. *Soft Computing, 13*, 213–225.

Prati, R. C., & Batista, G. E. A. P. A. (2004). Class imbalances versus class overlapping: An analysis of a learning system behavior. In *Proceedings of Mexican international conference on artificial intelligence (MICAI)* (pp. 312–321).

Quinlan, J. (1993). *C4.5: Programs for machine learning*. Morgan Kauffman.

Riddle, P., Segal, R., & Etzioni, O. (1994). Representation design and brute-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence, 8*, 125–147.

Shaffer, J. (1986). Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association, 81*, 826–831.

Sheskin, D. (2006). *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall.

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

Sikora, M. (2011). Induction and pruning of classification rules for prediction of microseismic hazards in coal mines. *Expert Systems with Applications, 38*, 6748–6758.

Sugiyama, M., Krauledat, M., & Müller, K.-R. (2007). Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research, 8*, 985–1005.

Sun, Y., Kamel, M. S., Wong, A. K. C., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition, 40*, 3358–3378.

Sun, Y., Wong, A. K. C., & Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence, 23*, 687–719.

Ting, K. M. (2002). An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering, 14*, 659–665.

Vapnik, V. (1998). *Statistical learning theory*. New York, USA: Wiley.

Vapnik, V., & Lerner, A. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control, 24*.

Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI* (pp. 55–60).

Villar, P., Fernández, A., & Herrera, F. (2010). A genetic algorithm for feature selection and granularity learning in fuzzy rule-based classification systems for highly imbalanced data-sets. In *13th International conference on information processing and management of uncertainty in knowledge-based systems (IPMU2010), CCIS* (Vol. 80, pp. 741–750).

Villar, P., Fernández, A., Sánchez, A., & Herrera, F. (2009). Un algoritmo genético para selección de características en sistemas de clasificación basados en reglas difusas para conjuntos de datos altamente no balanceados. In *Actas de la XIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA09)* (pp. 519–528).

Wang, B., & Japkowicz, N. (2004). Imbalanced data set learning with synthetic samples. In *Proceedings of IRIS machine learning workshop*.

Wang, K., Zhou, S., Fu, C. A., Yu, J. X., Jeffrey, F., & Yu, X. (2003). Mining changes of classification by correspondence tracing. In *Proceedings of the 2003 SIAM international conference on data mining (SDM 2003)*.

Weiss, G. M., & Provost, F. J. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research, 19*, 315–354.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics, 2*, 408–421.

Wu, G., & Chang, E. Y. (2005). Kba: Kernel boundary alignment considering imbalanced data distribution. *IEEE Transactions on Knowledge and Data Engineering, 17*, 786–795.

Xu, L., Chow, M.-Y., & Taylor, L. S. (2007). Power distribution fault cause identification with imbalanced data using the data mining-based fuzzy classification e-algorithm. *IEEE Transactions on Power Systems, 22*, 164–171.

Yang, Q., & Wu, X. (2006). 10 Challenging problems in data mining research. *International Journal of Information Technology and Decision Making, 5*, 597–604.

Yang, Y., Wu, X., & Zhu, X. (2008). Conceptual equivalence for contrast mining in classification learning. *Data and Knowledge Engineering, 67*, 413–429.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the 7th international conference on knowledge discovery and data mining (KDD'01)* (pp. 204–213).

Zadrozny, B., Langford, J., & Abe, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the 3rd IEEE international conference on data mining (ICDM'03)* (pp. 435–442).

Zhou, Z.-H., & Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering, 18*, 63–77.