The quest for the optimal class distribution: an approach for enhancing the effectiveness of learning via resampling methods for imbalanced data sets

Iñaki Albisua · Olatz Arbelaitz · Ibai Gurrutxaga ·
Aritz Lasarguren · Javier Muguerza · Jesús M. Pérez

Received: 4 July 2012 / Accepted: 12 October 2012 / Published online: 24 November 2012
© Springer-Verlag Berlin Heidelberg 2012

Abstract Class imbalance problems have lately become an important area of study in machine learning and are often solved using intelligent resampling methods to balance the class distribution. The aim of this work is to show that balancing the class distribution is not always the best solution when intelligent resampling methods are used, i.e. there is often a class distribution other than 50 % that improves the results. This paper presents a methodology to find a (near-)optimal class distribution. Experiments have been performed using 29 databases, 8 different resampling methods and two learning algorithms, and have been evaluated using AUC performance metric and statistical tests. The results go beyond the original objective, because they show that although intelligent resampling methods are mostly used to solve class imbalance problems they can also be used to improve results, using the class distribution found in data sets that are not imbalanced. As a consequence, we can state that when solving any machine learning problem it is always worth looking for the optimal class distribution.

Keywords Optimal class distribution · Class imbalance problems · Resampling methods · SMOTE

1 Introduction

Class imbalance problems are one of the areas of machine learning on which much effort has been focused in recent years. This problem is considered one of the emerging chal-

lenges in the machine learning area [19,22,28,42]. In class imbalance problems, the number of examples of one class (minority class) is much smaller than the number of examples of the other classes, with the minority class being the class of greatest interest one and that with the biggest error cost from the point of view of learning.

It is easily seen why this problem is a challenge for classifiers: if 99 % of the examples in a data set belong to the same class, a classifier that classifies new cases with the majority class will achieve 99 % accuracy. However, this classifier will have learned nothing from the problem we wanted to solve. Since most classification algorithms are designed to minimize the error rate, they tend to build very simple and useless classifiers with this kind of data sets [9,29].

As a consequence, many approaches have been proposed to deal with class imbalance problems and they can be divided into two main groups [11,19,22]: algorithmic approaches and data approaches. The former propose modifications in the algorithms, such as the improved boosting proposed by Joshi et al. [31]; the modification of SVM proposed by Wu et al. [40], an alternative based on one-class learning [33], and some other options that can be seen in [15,16,34,38,43]. On the other hand, the data approaches usually consist of resampling (subsampling or oversampling) the data in order to balance the classes before building the classifier. The latter approaches are much more versatile since they are independent of the learning algorithm used and most of the research has been done in this direction [8,4,6,17,30]. One of the most popular techniques is SMOTE [10]: an intelligent oversampling technique to synthetically generate more minority class examples. A broad analysis and comparison of some variants can be found in [5,22,36].

As a third option, some authors have proposed solving this problem by combining the algorithmic and data approaches and have obtained good results [3,37].

I. Albisua · O. Arbelaitz · I. Gurrutxaga · A. Lasarguren · J. Muguerza · J. M. Pérez (✉)

Computer Science Faculty, University of the Basque Country (UPV/EHU), Manuel Lardizabal 1, 20018 Donostia, Gipuzkoa, Spain
e-mail: txus.perez@ehu.es

Although our work is focused on the data approaches, it is not our aim to propose a new resampling method, but to describe a methodology to obtain better results when we use these methods irrespective of which resampling method we use.

However, Weiss and Provost [38] showed that there is usually a class distribution different to that appearing in the data set, with which better results are obtained. To draw these conclusions, Weiss and Provost performed an experiment to find the optimal class distribution for 26 real-world databases (20 from the UCI [20] and 6 from their own environment). They worked with the C4.5 algorithm and randomly undersampled the training data to perform the experiment with 13 different class distributions in each domain, and then analyzed the results using AUC and error estimates.

Based on Weiss and Provost's work, Albisua et al. [2] confirmed that changes in the class distribution of the training samples improve the performance of the classifiers. However, in contrast to what Weiss and Provost pointed out in their work, they found that the optimal class distribution depends on the learning algorithm used (even if there are decision tree learners using the same split criteria, such as C4.5 and CTC) and also on whether or not the trees are pruned.

This made us suspect that the results could differ depending on the class distribution used, independent of the resampling technique used before training the classifier. In most of the above-cited techniques for solving the class imbalance problem, the class distribution of the sample generated is usually 50 %: i.e. researchers tend to balance the classes. However, this does not need to be so, since most methods allow a class distribution other than 50 % to be used and, furthermore, some authors suggest that this option could be better [5, 12, 27]. Moreover, some methods do not guarantee that the final class distribution is the balanced one, for instance: SMOTE-ENN [5], EUSTSS [22], SMOTE-RSB* [36], and others.

In this work, we propose an approach for enhancing the effectiveness of the learning process that combines the use of resampling methods with the optimal class distribution (instead of balancing the classes). It should be noted that the use of this approach is not restricted to imbalanced data sets but can be applied to any data set (imbalanced or not) to improve the results of the learning process. The aim of Chawla et al. [12] was similar but they proposed a very different procedure that tackles only class imbalance problems; an heuristic procedure that applies first random subsampling to undersample the majority class while the results improve and then SMOTE to oversample the minority class until no improvement is found.

The proposed approach is the result of some research questions we tried to answer. As previously mentioned, resampling methods are used to balance class distribution. However, is the 50 % class distribution always the best one?

Weiss and Provost [38] have already answered this question for the case of random subsampling, but what about the resampling methods we mentioned above? Also, is the optimal class distribution independent of the resampling method and algorithm used? And, finally, is it worth resampling even for data sets with a balanced class distribution?

The experiments described in this work confirm that an optimal class distribution exists, but that it depends not only on the data's characteristics but also on the algorithm and on the resampling method used.

It is easy to assume that to obtain the optimal class distribution it would be necessary to analyze the results for an infinite (or sufficiently broad) range of class distributions. Limited by reality, we can describe our aim as to define a methodology that will help us to obtain a class distribution that achieves better results (with statistically significant differences) than the balanced distribution, whichever resampling method is used.

To corroborate the effectiveness of this approach, we performed experiments with 29 real problems (balanced and imbalanced ones) extracted from the UCI Repository benchmark [20] using eight different resampling methods, C4.5 and PART algorithms and the AUC performance measure. For estimating performance we used a 10-fold cross-validation methodology executed five times ($5 \times 10\text{CV}$). Finally, we used the non-parametric statistical tests proposed by Demšar [14] and García et al. [23, 24] to evaluate the statistical significance of the results.

Section 2 provides a brief description of the resampling methods, algorithms and performance metric to be used. In Sect. 3, we describe the experimental methodology used to corroborate the previously mentioned hypothesis, and in Sect. 4 we present an analysis of the experimental results. Finally, in Sect. 5 we summarize the conclusions and suggest further work.

2 Background

In this section, we briefly describe some of most popular and interesting resampling methods used to tackle the class imbalance problem found in bibliography. This section also includes a brief description of the two algorithms (C4.5 and PART) and performance metric or measure used to evaluate the classifiers (AUC) obtained by applying these methods.

2.1 Resampling methods

As stated in the introduction, one of the approaches used to solve class imbalance problems is to use a resampling method to balance the class distribution in the training data. Random subsampling and random oversampling can be considered as baseline methods because they are the simplest methods

and they do not use any kind of knowledge about the data set. In contrast, the rest of the methods can be considered as intelligent methods.

Random subsampling (RANSUB)

This consists of erasing randomly selected examples from the majority class in order to reduce it to the same number of examples the minority class has. Note that we could stop the process earlier to obtain a class distribution other than 50 %. We could even undersample the minority class to obtain a class distribution lower than the original one.

Random oversampling (RANOVER)

This is another non-heuristic resampling method whose idea is to replicate randomly selected examples in both the minority and majority classes, depending on whether the class distribution of the minority class has to be increased or decreased. It is normally used to replicate cases of the minority class to balance the training data, thus obtaining competitive results [5].

SMOTE

SMOTE (synthetic minority oversampling technique) [10] is an oversampling algorithm where the minority class is over-sampled to generate new synthetic examples. The basic idea is to generate new examples that are located between each of the minority class examples and one of its k nearest neighbours. The synthetic examples are generated with the following procedure: calculate the difference between the feature vector of the current example (a minority class example) and the feature vector of an example selected randomly from its nearest neighbours. Then, multiply the difference vector by a random value between 0 and 1 and finally, add this vector to the feature vector of the current example. The new vector will be the synthetic example. The number of times that a neighbour has to be selected to be used to generate a new example depends on the number of new examples that must be generated; for example, if we need to duplicate the number of examples in the minority class it will be sufficient to use one neighbour for each of the minority class examples.

Borderline-SMOTE

The authors proposed two different approaches [25] and named them Borderline-SMOTE1 and Borderline-SMOTE2. The main difference between these methods and SMOTE is that only the borderline minority examples are oversampled. The minority class examples will be considered to be in the borderline if more than half of their m nearest neighbours belong to the majority class; i.e. those examples on the bor-

deline between the majority and minority class. The authors consider these examples to be in danger; i.e. they could be confused with majority class examples. The Borderline-SMOTE1 option (B_SMOTE1) uses just the minority class neighbours of the borderline examples to generate the synthetic examples, whereas the Borderline-SMOTE2 option (B_SMOTE2) uses all the neighbours (minority and majority class). If the selected neighbour belongs to the majority class, the random value generated to multiply the difference vector will be in the range between 0 and 0.5 (in order to create the new example closer to the minority class).

ENN

Wilson's Edited Nearest Neighbour Rule (ENN) [39] is a cleaning algorithm that removes an example if it could be misclassified by its three nearest neighbours. The method is applied to both the majority and minority classes in the training set; i.e. the ENN algorithm consists of erasing those examples having at least two of their three nearest neighbours belonging to the other class. It is worth noting that this is a deterministic process where the class distribution of the final sample cannot be chosen.

SMOTE-ENN

This hybrid method is a combination of an oversampling method (SMOTE) and a cleaning algorithm (ENN) to be applied to the over-sampled training set. The motivation behind this method is to reduce the overfitting risk in the classifier due to the introduction of artificial minority class examples too deeply in the majority class space. This was one of the methods proposed in [5] achieving very good results in particular for data sets with few minority examples and it is considered as a reference method by some authors [22,32,36].

ENN-SMOTE

This is a variant of SMOTE-ENN we propose in this work. In this case, the cleaning process is done before applying SMOTE. It seems to make sense to clean the data before oversampling with SMOTE. Moreover, this method has a lower computational cost because it reduces the size of the sample and there is no need to calculate the distance of the new synthetic cases from the rest.

2.2 Learning algorithms

As we mentioned in the introduction the proposed approach has been tested in two algorithms: C4.5 and PART.

C4.5

This is a supervised learning algorithm used to build decision trees using divide-and-conquer strategy [35]. The algorithm uses the gain ratio as split criteria to divide the training sample based on the most discriminant variable according to the class. Once a decision tree is built a pruning procedure is carried out to avoid overfitting. C4.5 achieves high-quality results as single classifier with an added value: it provides with explanation the proposed classification. It was identified as one of the top 10 algorithms in data mining in the IEEE International Conference in Data Mining held in 2006 [41]. It is one of the most used learning algorithms specially as base classifier in multiple classifier systems (MCS) and as the classifier to be combined with intelligent resampling methods to solve class imbalance problems.

PART

PART [21] is a supervised learning algorithm that builds a rule set. It was designed by the authors of the WEKA platform [7] with the aim of combining the capacities of two algorithms: C4.5 [35] and RIPPER [13]. C4.5 is used to build a partial decision tree that will be used to extract one rule (the branch with greatest weight in the tree); the examples not covered by this rule are used to generate a new sample. This new sample is used to build a new partial C4.5 decision tree; the process is repeated until every example in the remaining sample is assigned to the same leaf node of a C4.5 tree; i.e. they are all in the root node. This algorithm has been used with C4.5 as representative of learning algorithms with explaining capacities by other authors [22].

2.3 Performance metric

Accuracy (*Acc*), the ratio of correctly classified examples to examples to classify, is a traditional performance metric of a classifier in the machine learning context. Its opposite, error rate ($\text{Err} = 1 - \text{Acc}$), is also widely used. However, these metrics are strongly biased in favour of the majority class when the prior class probabilities are very different as in class imbalance problems. In these cases, another kind of metric is required, where the performance of the classifier is measured based on the confusion matrix. Table 1 shows a confusion matrix for a two-class problem having positive (minority) and negative (majority) class values.

Table 1 Confusion matrix of a 2-class problem

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

From the confusion matrix we can derive performance metrics that directly measure the classification performance for positive and negative classes independently:

- True positive rate:

$$\text{TP rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (1)$$

- True Negative rate:

$$\text{TN rate} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (2)$$

- False negative rate:

$$\text{FN rate} = \frac{\text{FN}}{\text{TP} + \text{FN}} \quad (3)$$

- False positive rate:

$$\text{FP rate} = \frac{\text{FP}}{\text{TN} + \text{FP}} \quad (4)$$

For instance, True Positive rate (also called Recall or Sensibility) measures the percentage of correctly classified positive examples.

- Precision: Another interesting metric is Precision which measures the percentage of correctly classified positive examples with respect to the examples predicted to be positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (5)$$

Unfortunately, for most real-world applications it is impossible to maximize all the metrics and a trade-off must be found between pairs of them for example, between TP rate and TN rate, between FN rate and FP rate, between Precision and Recall, etc. Hence, some metrics have been designed to analyze the relationship between these two kinds of components. AUC is one of these metrics and, as mentioned before, the metric we selected for this study.

- AUC (area under ROC curve): The ROC (relative operating characteristic) curve is a graphical representation to compare TP rate and FP rate while changing the decision threshold of an example to belong to a class. The AUC metric tries to maximize the area under this curve. Therefore, AUC evaluates the classifier in multiple contexts of the classification space [5]. This is one of the most used metrics in the bibliography for class imbalance problems. A good description of this metric can be found in [18].

3 Experimental setup

Before we use any resampling method, there is a question that needs to be answered, but which is often ignored: what is the final class distribution we want to obtain?

As we mentioned in the Sect. 1, most of the published works use resampling methods to obtain balanced class distributions. Although this is in general a good value, it does not take into account the singularity of the specific database (problem) and it obviates the fact that a better one could exist.

In this work, we will divide the process of finding the optimal class distribution into two stages. First, we will use the simplest and computationally most efficient resampling method (random subsampling) to find the best class distribution. We will then use this class distribution to efficiently find a distribution that is better than the balanced distribution for the selected resampling method.

In order to obtain the optimal class distribution for each data set, we used the partial results for C4.5 published in the technical report [1] based on Weiss and Provost's work [38]. Although 30 databases were used in the technical report, in this work we decided not to use the Fraud database (a fraud detection problem in car insurance companies) as it was too big (with oversampling techniques we would generate samples of 200,000 cases) and it would not give us any additional information.

Weiss and Provost stated that the best value for the class distribution is data dependent (context or problem dependent) and they used random subsampling to determine the best of 13 different class distributions: 2, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90 and 95 %, and the original (Albisua et al. added the 98 % value in order to make the scanning symmetric and we did the same for this work). We carried out the same experiments with the PART algorithm.

We first performed the experiments with 29 two-class real problems, all belonging to the UCI Repository benchmark [20]. In Table 2, we present a summary of the characteristics of the databases used in the experiment ordered according to their original class distribution from more imbalanced to more balanced. Just as Weiss and Provost did in their work, we transformed all databases with more than two classes to two-class problems (in the table we show the number of classes before the transformation). We used a 10-fold cross-validation methodology five times ($5 \times 10\text{CV}$) to estimate the generalization capacity of the classifiers based on the AUC performance metric. As a consequence, we obtained 50 pairs of training and test samples for each database. From each training sample, in order to reduce the effect of chance, we then generated 100 subsamples for each of 14 class distributions (from 2 to 98 %). Figure 1 represents this sample generation process for a database. Thus, in this experiment, we used $5 \times 10 \times 14 \times 100$ (70,000) samples per database. All the samples generated in one database were of identi-

cal size: the number of examples of the minority class. We repeated the same subsampling process as Weiss and Provost did in their work [38]. For the 29 databases, the average size of the generated samples is 27.64 % of the training samples (the same as the mean of the original class distributions).

We used these samples to build C4.5 and PART classifiers. In each fold we built a classifier with each one of the 1,400 samples generated (14 class distributions $\times$ 100 samples). Thus, for the 29 databases we used 2,030,000 C4.5 trees and the same amount of PART sets of rules.

As Weiss and Provost mentioned, when the class distribution is changed in the sample used to induce a classifier, a corrector (oversampling ratio) has to be applied in the test process so that the induced model is adapted to the distribution expected in reality. This ratio has been used in our experiment.

Once we know the optimal class distribution for each database, we want to use this value with different resampling methods and to determine whether the results are better than those obtained with balanced samples. However, since we suspect that this optimal value may not be the optimal one for every different resampling technique, we will also use two more values that are close to the optimal one.

We repeated almost the same experiment (10-fold cross-validation five times), but using different resampling methods and limiting the scope of the scanned class distributions. We only used the optimal class distribution obtained in the first step (*ocd*), the next value (*ocd* + 10 %) and the previous value (*ocd* - 10 %) (see Fig. 2). Moreover, we wanted to compare the results obtained with these class distributions with those obtained with the balanced samples. For this reason, if the 50 value was not among those scanned for a data set this value was also added. For example, if the optimal class distribution for a data set is 30 (*ocd* = 30 %) we would use 20, 30, 40 and 50 % values.

In addition, we evaluated the proposal for eight different resampling methods. We used six well-known resampling methods that were described in Sect. 2.1 (RANSUB, RANOVER, SMOTE, B_SMOTE1, B_SMOTE2 and SMOTE-ENN) where we used two versions of RANSUB, and we also tried a variant of SMOTE-ENN (ENN-SMOTE).

One version of the RANSUB method used is that explained previously, i.e. the one used to determine the optimal class distribution (*ocd*). We will refer to this version as *RANSUB_{ocd}*. Since the size of these samples is very small (following Weiss and Provost's methodology), we also used a random subsampling method with no size limitations, i.e. randomly erasing selected examples from one of the classes only until the desired class distribution is achieved. We will refer to this version as RANSUB.

Since we are conscious that randomness plays an important role in every resampling method, for each training sample in the $5 \times 10\text{CV}$, we generated 50 samples for each method

Table 2 Domains used in the experiment and their characteristics

Domain	Number of examples	Number of features	Number of classes	Original class distribution	Missing values
<i>Hypo</i>	3,163	25	2	4.77	Y
<i>Abalone</i>	4,177	8	29	8.6	N
<i>Sick_euthyroid</i>	3,120	25	2	9.26	Y
<i>Optdigits</i>	5,620	64	10	9.9	N
<i>Soybean-Large</i>	290	35	15	13.79	Y
<i>Segment210</i>	210	19	7	14.29	N
<i>Segment2310</i>	2,310	19	7	14.29	N
<i>Solar_flare</i>	1,389	13	2	15.7	N
<i>Kddcup</i>	4,941	41	2	19.69	N
<i>Hepatitis</i>	155	19	2	20.65	Y
<i>Vehicle</i>	846	18	4	23.52	N
<i>Glass</i>	214	9	7	23.83	N
<i>Splice_junction</i>	3190	60	3	24.1	N
<i>Yeast</i>	1,484	8	10	28.9	N
<i>Breast-y</i>	286	9	2	29.72	Y
<i>Credit-g</i>	1,000	20	2	30.0	N
<i>Car</i>	1,728	6	4	30.0	N
<i>Iris</i>	150	4	3	33.33	N
<i>Breast-w</i>	699	10	2	34.5	Y
<i>Pima</i>	768	8	2	34.9	N
<i>Heart-h</i>	294	13	2	36.05	Y
<i>Voting</i>	435	16	2	38.62	Y
<i>Spam</i>	4,601	57	2	39.4	N
<i>Lymph</i>	148	18	4	41.22	N
<i>Liver</i>	345	6	2	42.03	N
<i>Bands</i>	540	39	2	42.2	Y
<i>Credit-a</i>	690	14	2	44.49	N
<i>Heart-c</i>	303	13	2	45.87	Y
<i>Ks-vs-kp</i>	3,196	36	2	47.8	N

Maximum and minimum values are shown in italics

Fig. 1 Schema repeated in each database for sample generation in order to discover the optimal class distribution (Step 1)

and class distribution as it is shown in the Fig. 2. The SMOTE-ENN method is an exception; in this case we generated a single sample, instead of 50. In order to apply this method

50 times, we had to recalculate a new distance matrix for each sample generated with SMOTE and then apply ENN. This caused problems with disk space and extended the

Fig. 2 Schema repeated in each database and in each fold of the 5×10 CV for sample generation related to every resampling method we used (Step 2)

experimental time for months and months. Addressing this problem will be a further work.

Figure 2 summarizes the resampling process carried out for each training sample of the 5×10 CV. The methods used have been grouped according to the size of the samples to be generated. We try to reflect this size proportionately in the figure, as well as its difference depending on the distribution value applied for each resampling method. In fact, as stated above, using the mean of the original class distribution for the 29 data sets, the average size of the samples generated using $RANSUB_{ocd}$ is 27.64 % of the training samples; the same for each sample in each data set regardless of the class distribution used, just as for Weiss and Provost.

If we suppose the optimal class distribution (ocd) is 50 %, for an original class distribution of 27.64 % $RANSUB$ would generate double-sized subsamples (55.28 %). In addition, 69.10 and 46.07 % would be the sizes for $ocd - 10$ % and $ocd + 10$ %, respectively. However, for the other four over-sampling methods (from $RANOVER$ to B_SMOTE2), the sizes would be 120.60, 144.72 and 180.90 % (for $ocd - 10$ %, ocd and $ocd + 10$ %, respectively).

Finally, we cannot know *a priori* the size of the samples generated by the combination of the SMOTE and ENN methods because we do not know how many examples will be erased with the ENN cleaning method before or after applying SMOTE. However, we can guarantee that their size will be smaller than the size of the samples of the previous group.

As we previously indicated, AUC was the performance metric selected for our experiment to determine the best value for the class distributions. Moreover, we used the non-parametric tests proposed by Demšar [14] and García et al. [23, 24] to evaluate the statistical significance of the results.

3.1 Implementation issues

In some of these methods a distance metric is required to calculate which are the k nearest neighbours (k -NN algorithm) of a minority class example. The distance between examples

is usually calculated based on Euclidean distance; however, Euclidean distance is not adequate for qualitative (rather than quantitative) features. We have implemented SMOTE using the HVDM (Heterogeneous Value Difference Metric) distance [39] which uses Euclidean distance for quantitative attributes and VDM distance for the qualitative ones. The VDM metric takes into account the similarities between the possible values of each qualitative attribute.

A further problem is that the examples can have missing values. These are usually replaced by the average of that attribute for the rest of examples of its class in the case of quantitative attributes and for the mode in the case of qualitative attributes.

It should be noted that we used the same distance implementations used in the well-known WEKA machine learning workbench [7] for our implementation of the SMOTE method and its variants.

In most of the works using this kind of method where a k -NN algorithm is used, the value of k is not specified. Chawla et al. [10] concluded that 5 was a good value for k and this is the value assumed in the revised works.

On the other hand, for the Borderline-SMOTE1 and Borderline-SMOTE2 methods, the authors explain that half of the minority class should be on the borderline before applying SMOTE, but the way the m value should be selected and the process to determine the cases in danger (those on the borderline) is presented as a further work in their paper [25]. Therefore, let us explain how we implemented the danger case detection.

We first instantiate m with a small value (5). Then we check which examples would be in danger when taking into account its m nearest neighbours. If at least half of the minority class examples are in danger, the search is finished; if not, we duplicate the value of m and search again for cases in danger. The idea is to repeat the procedure until we have sufficient danger cases, but there is a small problem: if m becomes greater than double the minority class size, all the minority examples will be in danger and, therefore, all the cases will

be selected; as a result, Borderline-SMOTE methods would become standard SMOTE. In order to avoid this, our search will finish before m becomes too big (the algorithm is shown in Appendix 3).

Furthermore, in order to reduce the disk space required, and knowing that only in step 1 we already had 70,000 subsamples of the original sample for each database, we generated the samples by simply saving in a file the position of the selected example in the original database. The training and test samples were saved in this way. This method of storing samples also reduces the computational cost of the resampling processes using k -NN since the distance matrix of the complete data set can be generated only once and then used in every intelligent resampling method to generate the set of samples in each fold.

4 Experimental results

This section is devoted to showing the results obtained in the two phases described: the calculation of the optimal class distribution (*ocd*) for each database (based on the random subsampling method) and a second phase where we try to determine whether the *ocd* value helps us to achieve better results than the balanced class distribution for a set of different and well-known resampling methods. Both phases require an estimation of the best class distribution within a range of values. As we wanted these estimations to be as realistic as possible, we divided the whole sample belonging to each data set in training, validation and test data [26] and we made the estimation of the optimal class distribution based only on validation data.

In the first step of our experiment, we calculated the optimal class distribution for each database. For the C4.5 algorithm, a more detailed report of the results for each data set can be found in Appendix 1 of the technical report [1]. We can see all the results obtained for the C4.5 and PART algorithms, 14 class distributions and the AUC performance metric in Table 3. Results for the C4.5 algorithm are shown in the upper part of the table whereas results the PART algorithm are shown in lower part. The information in the columns is for different class distributions and the results for each data set are shown in different rows. For each database, the best result is marked in bold while all the results improving the results for the original distribution have a grey background. Moreover, the relative position of the original class distribution within the range of evaluated class distributions is denoted using a vertical double bar between columns (just as Weiss and Provost did). For example, for the Hypo data set the vertical bar indicates that the original distribution falls between the 2 and 5 % distributions (4.77 % from Table 2). It should be noted that each value in the table represents the average performance of 5,000 classifiers ($5 \times 10 \times 100$) whereas in

Weiss and Provost's work [38] this estimate was done with only 30 classifiers.

Observing the shape of the grey background in Table 3 we can conclude that, for both algorithms, the more imbalanced the data set, the greater the number of class distribution values there are that improve the performance of the classifier for the original class distribution (orig column). The best results for each data set appear for class distributions around 50 % and only in two data sets for C4.5 (*Car* and *Credit-a*) and three datasets for PART (*Car*, *Voting* and *Kr-vs-kp*) out of 29, the best AUC is achieved with the original class distribution. Furthermore, if we analyze the average results C4.5 and PART (*Mean* and *Median* rows in Table 3) the average AUC achieved for the 29 databases was greater than that achieved with the original class distribution for a wide range of data distributions (between 30 and 70 % for both algorithms) and the best average results were obtained for the 50 % distribution.

Based on these results, we determined the corresponding class-distribution values for the second step. We selected the class-distribution with the best AUC value (those marked in bold in Table 3) and two more values obtained by subtracting and adding 10 % to this value. For the cases where the balanced distribution is not one of these three values (marked with * in the *ocd* column in Table 4), we also considered the balanced distribution to compare the results. Thus, Table 4 shows the class-distribution values applied (Applied CDs) for each algorithm and database in the second step of the experiment and the original (original CD) class distribution.

In the *ocd* column of Table 4, the cell has been shaded if this value is other than 50 %. As can be observed, this occurs in 16 data sets out of 29 for the C4.5 algorithm and in 12 data sets for the PART algorithm. The optimal values of the table range from 30 % up to as much as 95 %.

Once we had the class distribution values, we were able to carry out the second part of the experiment, a five times 10-fold cross-validation with 50 samples per fold, for each of the proposed class distributions and resampling methods. We then built C4.5 and PART classifiers for each sample and tested the results using AUC performance metric.

The results are shown in Table 5. The ORIGINAL row shows the average values of the AUC obtained using the whole training sample to build the classifiers; i.e. without a resampling method. This will be a reference value to evaluate whether or not it is worth using any resampling method.

In the rows below we can see the mean of these values obtained for each resampling method. The *bal* column shows the results for the balanced distribution, the *ocd* column shows the results for the optimal class distribution and the *orm* column shows the results for the optimal value obtained in step 2 [note that 3 (or 4) class distributions have been used].

By way of example, to better understand where these results come from, we have added (in Appendix 1) a table

Table 3 Average AUC values, 5×10 CV, for C4.5 and PART classifiers

		C4.5													
	orig	2	5	10	20	30	40	50	60	70	80	90	95	98	
Hypo	87.66	66.64	87.50	92.41	95.20	95.97	96.44	96.74	96.78	96.53	96.05	94.78	92.69	81.20	
Abalone	56.50	50.09	50.61	52.51	69.24	71.24	72.74	73.36	73.37	72.26	66.69	57.51	52.89	50.16	
Sick_euthyroid	91.01	63.98	85.95	91.30	93.32	94.26	94.86	95.39	95.67	95.03	92.90	88.27	84.50	70.21	
Optdigits	59.33	50.02	50.96	58.97	78.16	86.47	90.66	89.73	87.57	84.39	80.75	73.50	64.60	50.80	
Soybean_Large	72.94	50.00	50.00	65.25	80.18	84.05	85.97	85.18	83.75	80.27	76.29	69.56	59.66	50.00	
Segment210	86.06	50.00	50.00	79.29	87.50	90.30	91.09	91.21	89.76	88.52	83.72	78.62	75.42	50.00	
Segment2310	97.11	84.25	92.22	96.39	97.74	96.24	98.35	98.30	98.19	97.92	97.41	94.71	89.55	76.88	
Solar_flare	54.30	50.05	50.31	50.76	56.04	65.64	70.66	72.06	71.96	66.85	52.40	50.21	50.00	50.01	
Kddcup	98.85	94.92	97.21	98.54	98.85	99.10	99.18	99.18	99.09	98.93	98.71	98.55	98.31	98.80	
Hepatitis	73.21	50.00	50.00	60.58	71.25	76.30	78.78	79.64	79.46	77.09	72.69	62.41	55.93	50.00	
Vehicle	92.25	56.17	76.29	85.51	91.19	93.42	94.56	94.92	94.86	94.26	92.70	88.45	80.21	69.09	
Glass	89.51	50.00	73.24	80.54	88.27	90.45	92.24	93.10	93.28	92.98	92.24	90.47	87.88	50.00	
Splice_junction	95.36	50.00	66.55	90.23	94.75	95.80	95.93	95.65	94.87	93.74	92.71	90.40	86.75	79.01	
Yeast	74.12	50.08	51.24	58.36	69.60	74.42	76.45	77.18	76.78	74.98	71.02	61.68	55.50	50.30	
Breast-y	62.49	50.01	50.11	51.11	57.06	62.79	64.82	67.23	65.78	55.38	50.28	50.00	50.00	50.00	
Credit-g	70.99	50.01	50.10	50.97	61.89	70.96	73.10	73.66	72.65	70.02	65.80	51.54	50.19	50.02	
Car	96.05	50.00	52.68	88.51	94.98	96.02	95.48	94.50	93.42	92.54	91.90	90.99	83.15	74.34	
Iris	98.67	50.00	88.80	93.57	97.51	98.39	98.94	99.20	99.41	99.56	99.75	99.86	99.92	50.00	
Breast-w	94.77	50.12	65.10	89.13	93.54	94.86	95.62	96.44	96.67	96.47	96.01	94.35	50.00	50.00	
Pima	78.52	51.32	57.54	64.81	72.85	77.24	79.37	79.99	79.32	77.33	73.85	67.79	62.37	52.99	
Heart-h	81.94	51.50	57.89	72.45	76.21	80.24	83.06	84.82	84.83	83.72	81.30	67.64	50.51	50.00	
Voting	97.58	78.60	92.44	95.80	97.44	97.65	97.57	97.38	97.09	96.77	96.48	96.22	94.57	88.42	
Spam	94.67	64.37	80.16	88.64	93.68	95.48	95.92	94.99	94.69	93.68	91.53	88.49	81.59	67.88	
Lymph	84.48	50.00	50.11	56.14	77.65	81.56	84.23	84.90	84.20	81.57	74.25	64.23	55.34	52.46	
Liver	73.45	50.16	51.14	56.59	64.52	69.75	73.09	74.21	73.05	69.55	63.74	56.27	51.46	50.43	
Bands	83.59	52.67	58.31	65.87	75.30	80.85	83.35	84.20	83.67	81.68	76.64	59.12	51.66	50.23	
Credit-a	90.30	54.18	68.66	85.39	89.94	89.94	90.27	89.93	88.57	87.17	86.20	85.25	63.54	50.81	
Heart-c	83.81	51.02	54.94	68.39	75.09	80.44	82.98	84.11	83.04	80.37	76.94	64.78	51.21	50.26	
Kr-vs-kp	99.72	80.99	91.55	95.52	98.75	99.39	99.66	99.72	99.66	99.37	98.52	96.53	93.21	86.76	
Mean	82.31	56.71	65.05	74.56	82.23	85.33	86.90	87.32	86.73	84.95	81.93	76.62	69.10	60.03	
Median	85.27	50.10	57.71	75.87	83.84	88.20	90.46	89.83	88.07	85.78	82.51	76.06	62.95	50.37	

PART

	orig	2	5	10	20	30	40	50	60	70	80	90	95	98
Hypo	86.83	57.46	87.33	92.33	95.09	95.90	96.43	96.76	96.81	96.51	95.98	94.62	92.60	79.07
Abalone	55.21	50.07	50.49	51.90	67.93	70.64	71.98	72.70	73.11	67.46	60.51	54.63	52.03	50.16
Sick_euthyroid	91.35	63.45	86.62	91.54	93.36	94.02	94.53	95.08	95.52	95.41	93.65	89.68	84.95	72.70
Optdigits	61.67	50.01	51.06	58.08	69.49	85.31	90.32	90.94	89.52	86.44	83.33	78.81	77.72	51.51
Soybean_Large	72.90	50.00	50.00	65.24	80.40	83.67	85.68	86.11	85.04	82.17	79.63	73.61	62.11	50.00
Segment210	96.97	83.88	91.98	96.28	97.56	98.05	98.17	98.21	98.18	97.95	97.28	93.91	89.47	74.18
Segment2310	85.97	50.00	50.00	79.34	87.30	89.88	90.77	90.90	89.41	88.23	83.57	78.53	75.41	50.00
Solar_flare	53.32	50.05	50.30	50.70	53.78	61.73	68.07	71.04	65.04	59.85	51.73	50.21	50.00	50.01
Kddcup	98.86	92.21	96.58	98.25	98.79	99.00	99.04	99.05	98.98	98.85	98.71	98.60	98.27	98.80
Hepatitis	72.64	50.00	50.00	60.34	70.07	75.12	77.89	79.00	78.64	76.40	72.26	62.94	56.03	50.00
Vehicle	92.36	55.80	75.16	85.56	91.34	93.52	94.44	94.64	94.62	94.02	92.35	87.67	79.82	67.16
Glass	88.92	50.00	73.24	80.22	87.61	89.89	91.83	92.77	93.03	92.81	92.02	90.12	87.91	50.00
Splice_junction	95.85	50.00	50.14	57.09	94.90	95.90	95.53	94.91	94.31	93.58	91.50	89.16	87.48	80.06
Yeast	70.72	50.08	51.01	55.51	65.35	71.03	73.27	75.77	75.17	72.05	67.25	59.55	54.87	50.31
Breast-y	62.34	50.01	50.11	51.06	56.30	62.88	65.17	66.49	59.50	52.11	50.17	50.00	50.00	50.00
Credit-g	57.66	50.01	50.07	50.54	51.90	57.25	70.83	72.26	71.07	68.29	52.70	50.50	50.12	50.02
Car	94.56	50.00	50.99	75.34	91.85	94.37	94.20	93.64	92.97	92.37	91.88	90.99	83.39	74.31
Iris	98.67	50.00	88.80	93.57	97.51	98.39	98.94	99.20	99.41	99.56	99.75	99.86	99.92	50.00
Breast-w	96.36	50.24	67.20	91.33	95.53	96.31	96.37	95.95	94.97	93.79	92.42	90.12	50.00	50.00
Pima	76.57	51.19	56.72	62.46	70.16	75.09	77.74	78.78	78.12	75.71	72.16	66.36	61.19	52.77
Heart-h	80.81	51.39	57.47	71.04	76.10	79.59	81.69	83.91	84.65	82.96	77.68	61.63	50.51	50.00
Voting	97.43	78.00	90.39	92.96	96.26	97.27	97.39	97.28	97.04	96.76	96.32	95.13	92.56	88.24
Spam	93.87	59.62	77.51	87.30	92.29	93.94	94.97	94.37	93.88	92.99	91.24	88.04	80.21	63.30
Lymph	84.60	50.00	50.11	55.85	75.84	81.31	84.23	84.96	84.00	81.08	73.76	63.60	55.47	52.51
Liver	70.82	50.15	51.03	55.16	61.21	65.73	69.87	72.82	70.78	65.82	60.33	54.93	51.23	50.33
Bands	78.71	52.36	57.61	64.98	70.38	73.62	77.92	80.68	79.73	72.45	64.85	56.09	51.44	50.19
Credit-a	89.39	54.01	66.83	81.18	88.87	90.61	89.98	88.55	87.60	86.97	86.20	83.68	60.87	50.77
Heartc	83.20	50.88	54.48	63.69	72.79	79.36	82.37	83.54	82.62	79.07	71.92	58.11	51.10	50.28
Kr-vs-kp	99.49	62.72	84.42	93.01	97.61	99.12	99.45	99.48	99.25	98.62	98.24	96.36	88.44	82.90
Mean	82.35	55.64	64.40	72.82	80.95	84.43	86.52	87.23	86.31	84.15	80.67	76.12	69.83	59.99
Median	85.97	50.15	56.72	71.04	87.30	89.88	90.32	90.90	89.41	86.97	83.57	78.81	62.11	50.33

We present values for the 29 databases. Each column represents the results for one of the 14 class distributions

Table 4 Original, optimal and selected class distributions for every data set

	Original CD	Applied CDs: C4.5			Applied CDs: PART		
		<i>ocd</i> − 10	<i>ocd</i>	<i>ocd</i> + 10	<i>ocd</i> − 10	<i>ocd</i>	<i>ocd</i> + 10
<i>Hypo</i>	4.77	50	60	70	50	60	70
<i>Abalone</i>	8.70	50	60	70	50	60	70
<i>Sick_euthyroid</i>	9.26	50	60	70	50	60	70
<i>Optdigits</i>	9.90	30	40	50	40	50	60
<i>Soybean_large</i>	13.79	30	40	50	40	50	60
<i>Segment210</i>	14.29	40	50	60	40	50	60
<i>Segment2310</i>	14.29	30	40	50	40	50	60
<i>Solar_flare</i>	15.70	40	50	60	40	50	60
<i>Kddcup</i>	19.69	30	40	50	40	50	60
<i>Hepatitis</i>	20.65	40	50	60	40	50	60
<i>Vehicle</i>	23.52	40	50	60	40	50	60
<i>Glass</i>	23.83	50	60	70	50	60	70
<i>Splice_junction</i>	24.10	30	40	50	20	30*	40
<i>Yeast</i>	28.90	40	50	60	40	50	60
<i>Breast-y</i>	29.72	40	50	60	40	50	60
<i>Credit-g</i>	30.00	40	50	60	40	50	60
<i>Car</i>	30.00	20	30*	40	20	30*	40
<i>Iris</i>	33.33	90	95*	98	90	95*	98
<i>Breast-w</i>	34.48	50	60	70	30	40	50
<i>Pima</i>	34.90	40	50	60	40	50	60
<i>Heart-h</i>	36.05	50	60	70	50	60	70
<i>Voting</i>	38.62	20	30*	40	30	40	50
<i>Spam</i>	39.40	30	40	50	30	40	50
<i>Lymph</i>	41.22	40	50	60	40	50	60
<i>Liver</i>	42.03	40	50	60	40	50	60
<i>Bands</i>	42.20	40	50	60	40	50	60
<i>Credit-a</i>	44.49	30	40	50	20	30*	40
<i>Heart-c</i>	45.87	40	50	60	40	50	60
<i>Kr-vs-kp</i>	47.80	40	50	60	40	50	60

Italic values indicate that the optimal class distribution is other than 50%

Asterisks indicate the cases when the balanced class distribution is not one of the selected ones

Table 5 Mean values of AUC for each of the evaluated algorithms, resampling methods and class distributions used

	ORIGINAL	C4.5 (83.47)			PART (79.92)		
		<i>bal</i>	<i>ocd</i>	<i>orm</i>	<i>bal</i>	<i>ocd</i>	<i>orm</i>
RANSUB _{<i>ocd</i>}		83.97	84.17	84.19	84.11	84.23	84.27
RANSUB		85.62	85.28	85.69	85.52	85.20	85.54
RANOVER		84.18	84.17	84.63	84.84	85.09	85.37
SMOTE		84.80	84.84	85.13	84.44	84.76	85.23
B_SMOTE1		84.42	84.47	84.91	84.48	84.69	85.11
B_SMOTE2		84.96	85.12	85.44	84.78	84.67	85.68
SMOTE-ENN		84.15	82.63	83.82	83.97	82.09	83.46
ENN-SMOTE		83.29	83.55	83.82	83.45	83.50	83.61

Italic values indicate the cases where the results without resampling method are not improved

Bold values indicate the cases where the results obtained with balanced samples are improved

with the results for the 29 data sets, but only for the SMOTE cases and PART algorithm. This same analysis was done for all the methods and both algorithms, but for the sake of readability we summarize it in Table 5.

The results improving those obtained with no-resampling method (ORIGINAL) have a grey background, while those that show an improvement on the results obtained with balanced samples (*bal* column) are marked in bold.

The values in Table 5 show that, for both algorithms, most of the results obtained improve those obtained with the whole

sample and shown in the ORIGINAL row (grey background). According to these results, for both algorithms, the optimal class distribution found in step 2 (*orm*) performs better than the balanced version for almost every resampling method; i.e. for every data set and every resampling method there is a class distribution better than 50 %.

With regard to the results obtained with *ocd* class distribution (*ocd* column), it can be observed that they are better (marked in bold) than the balanced ones (*bal* column) for more than half of the resampling methods evaluated.

Table 6 Average ranks and Friedmans test values related to AUC evaluation metric

	C4.5					PART				
	ORIGINAL	<i>bal</i>	<i>ocd</i>	<i>orm</i>	Friedman (%)	ORIGINAL	<i>bal</i>	<i>ocd</i>	<i>orm</i>	Friedman (%)
RANSUB _{ocd}	2.0690 (1)	2.8621 (4)	2.5862 (3)	2.4828 (2)	87.13	3.6207 (4)	2.3448 (3)	2.0000 (1)	2.0345 (2)	100.00
RANSUB	2.9138 (4)	2.4828 (2)	2.6207 (3)	1.9828 (1)	95.47	3.7241 (4)	2.1034 (2)	2.4483 (3)	1.7241 (1)	100.00
RANOVER	2.7069 (3.5)	2.6034 (2)	2.7069 (3.5)	1.9828 (1)	<i>90.56</i>	3.9655 (4)	2.2759 (3)	1.9483 (2)	1.8103 (1)	100.00
SMOTE	2.8966 (4)	2.7759 (3)	2.5690 (2)	1.7586 (1)	99.77	3.8621 (4)	2.5172 (3)	2.1552 (2)	1.4655 (1)	100.00
B_SMOTE1	2.8966 (4)	2.6207 (2)	2.6724 (3)	1.8103 (1)	99.36	3.5517 (4)	2.3966 (3)	2.2759 (2)	1.7759 (1)	100.00
B_SMOTE2	2.7069 (3)	2.8966 (4)	2.4138 (2)	1.9828 (1)	96.22	3.7241 (4)	2.4655 (3)	2.3103 (2)	1.5000 (1)	100.00
SMOTE-ENN	2.3448 (1)	2.6207 (3)	2.6552 (4)	2.3793 (2)	27.47	2.8621 (4)	2.3793 (2)	2.5862 (3)	2.1724 (1)	78.94
ENN-SMOTE	2.3966 (2)	2.8621 (4)	2.6379 (3)	2.1034 (1)	86.45	3.5172 (4)	2.3793 (3)	2.1379 (2)	1.9655 (1)	100.00

Bold values indicate statistically significant differences with an over 95 % significance level

Italic values indicate statistically significant differences with an over 90 % significance level

We have analyzed the results based on the average values for the 29 data sets, but are the differences found between these results statistically significant?

We used the non-parametric tests proposed by Demšar [14] and García et al. [23,24] to evaluate if there is any statistically significant difference between the results obtained with the class distributions, we propose and those obtained with balanced class distribution. Demšar proposed the use of the non-parametric Friedman test [14] to compare more than two algorithms (or possibilities) and discover whether or not significant differences between their behaviours appear. The Friedman test ranks the algorithms for each data set separately and compares the obtained average ranks using the FF statistic. Table 6 shows the average ranks and the result of the Friedman test for each resampling method.

As it can be observed in Table 6, *orm*, the optimal class distribution found related to each resampling method has the best average rank for most of the methods. It can also be observed that for the C4.5 algorithm statistically significant differences were found for four resampling methods with an over 95 % ($\alpha = 0.05$) significance level (marked in light grey) and for one more with an over 90 % ($\alpha = 0.1$) significance level (marked in dark grey). However, for the PART algorithm SMOTE-ENN was the only method without significant differences (with a 100 % significance level).

Since significant differences were found in most cases, the next step was to perform post-hoc tests to discover between which pairs of options the differences appeared. We performed three different post-hoc tests. On the one hand, we performed the Nemenyi and Bonferroni-Dunn classic tests and obtained the CD (critical difference) diagrams (graphical representation of the average ranks and significant differences when multiple algorithms are compared proposed by Demšar). On the other hand, we performed the Holm test (focused on the comparison between a control method and a set of options) because being a more powerful test it could detect more significant differences.

For the sake of clarity we moved to Appendix 2 the figures of the CD diagrams for C4.5 and PART. We included a CD diagram for each resampling method we analyzed.

Results regarding to Holm's procedure are shown in Fig. 3. We applied the Holm's procedure to compare the results achieved with the *orm* class distribution (the control method) with the results achieved with other possible options (ORIGINAL, marked with squares, *bal*, marked with x, and *ocd*, marked with triangles). The graphic in the left part of the figure shows results for the C4.5 algorithm, where as the graphic in the right part does it for the PART algorithm. The graphics include 1 axe per resampling method evaluated where adjusted p values are shown. In addition to these p values, two octagons are drawn: the external one (dashed line) marks the p value $\alpha = 0.1$ and the internal one the p value $\alpha = 0.05$. Thus, we could interpret the figures the following way: for each resampling method, the options that fall inside the internal octagon have significant differences according to the *orm* option with a 95 % significance level and the options that appear between the two octagons have differences with a 90 % significance level.

Results in Fig. 3 show that for most of the resampling methods the *orm* possibility performs better than the rest, and for many of them, with statistically significant differences. More specifically, in the case of the PART algorithm, significant differences were found between the *orm* and the ORIGINAL options with a 95 % confidence level for every method but SMOTE-ENN where the differences have been found with a 90% significance level. However, for the C4.5 algorithm significant differences between the *orm* and the ORIGINAL options with a 95 % significance level were found for RANSUB, SMOTE and B_SMOTE1 resampling methods and with a 90 % significance level for RANOVER and B_SMOTE2. If we compare the performance of the *orm* and the *bal* options, even if the first one obtained better ranks for every resampling method and algorithms, there are fewer significant differences. Specifically for the C4.5 algorithm,

Fig. 3 Star plot diagrams related to Holm's adjusted p values for C4.5 and PART and the eight resampling methods. The values in the axis are shown in logarithmic scale

Table 7 Relative improvement and Wilcoxon's test values comparing SMOTE-ENN variants

ENN-SMOTE vs. SMOTE-ENN	bal (%)	ocd (%)	orm (%)
C4.5			
Rel.Impr	−0.99	1.93	0.04
Wilcoxon	80.18	40.37	55.73
PART			
Rel.Impr	−0.66	2.47	0.25
Wilcoxon	31.09	27.87	6.03

significant differences with a 95 % significance level have been found for SMOTE, B_SMOTE1 and B_SMOTE2 and with a 90 % significance level for ENN-SMOTE. With regard to the PART algorithm, differences with a 95 % significance level were found for SMOTE and B_SMOTE2. If we finally compare the results obtained with *orm* and *ocd* very similar conclusions are obtained.

As a secondary result, we have also analyzed whether there are statistically significant differences between the two combinations of SMOTE and ENN: the SMOTE-ENN and the ENN-SMOTE we proposed. The test proposed by Demšar for pairwise comparisons is the Wilcoxon signed-ranks test. Table 7 shows the average relative improvement values (Rel.Impr. row) and the Wilcoxon's Tests values (Wilcoxon row) for the three values used for class distribution: *bal*, *ocd* and *orm*.

Based on Table 7, we can conclude that there are not significant differences between using the ENN cleaning method before or after applying the SMOTE method. However, as stated above, ENN-SMOTE has a computationally lower cost and it also allows us to tune the final class distribution of the sample.

Finally, we wanted to know how many of the optimal class distribution values were other than 50 %, and, when they were 50 %, how many times this value was outside our selected range ($ocd \pm 10$). Table 8 shows the optimal class distribution value for each database and resampling method. Column 1 shows the database name, while column 2 shows its original class distribution. After that, for each algorithm, the optimal value obtained in step 1 is shown first and then, the final class distribution value obtained for each resampling method. Values other than 50 % have a grey background. The table includes as a summary in the lower part the mean of the optimal class distribution (*Mean* row) and the standard deviation (*Std Dev* row) as well as the number of times the optimal class distribution is 50 % (row #*orm* = 50) and the number of times it is the same as the optimal value obtained in step 1 (row #*orm*=*ocd*). As a conclusion, we can say that although the optimal class distribution is usually a nearly balanced class distribution (50 %), as indicated by the mean values, this is the best value in only a few cases. There are some results that need further explanation in order to better understand them. Although the optimal value for the two columns Optimal CD and RANSUBocd might be expected to be the same, since the same resampling method is used, this does not always happen (see row #*orm*=*ocd*). This is because chance plays a part in the method and, besides, a different number of samples was used for each option (100 subsamples for Optimal CD (step 1) and only 50 in all the resampling methods used in step 2, including RANSUBocd).

Having analyzed the results of the wide range of experiments, we can claim to have proposed a method that is able to find a pseudo-optimal class distribution to be used for a set of intelligent resampling methods.

Table 8 Optimal class distribution values for all data sets and resampling methods using AUC metric

		C4.5									PART								
Original CD	Optimal CD	RANSUB _{oed}	RANSUB	RANOVER	SMOTE	B.SMOTE1	B.SMOTE2	SMOTE-ENN	ENN-SMOTE	Optimal CD	RANSUB _{oed}	RANSUB	RANOVER	SMOTE	B.SMOTE1	B.SMOTE2	SMOTE-ENN	ENN-SMOTE	
Hypo	4.77	60	50	50	60	70	70	70	60	50	60	50	50	50	50	70	60	50	
Abalone	8.70	60	50	70	60	50	50	60	70	70	60	50	70	60	70	60	60	70	
Sick_euthyroid	9.26	60	60	50	50	50	70	50	70	50	60	60	70	50	60	50	70	60	
Optdigits	9.90	40	40	50	40	30	30	30	50	30	50	50	60	60	50	60	60	60	
Soybean_Large	13.79	40	40	40	50	40	40	30	50	30	50	50	60	60	60	40	60	60	
Segment210	14.29	50	50	30	30	30	50	50	50	40	50	50	40	50	60	60	60	60	
Segment2310	14.29	40	40	40	60	50	40	60	60	60	50	40	60	40	40	40	60	60	
Solar_flare	15.70	50	50	50	40	60	40	40	50	60	50	50	50	40	40	40	50	60	
Kddcup	19.69	40	50	30	40	30	50	50	50	30	50	50	40	40	40	40	40	40	
Hepatitis	20.65	50	50	50	60	50	50	60	60	60	50	50	50	60	50	40	60	60	
Vehicle	23.52	50	60	50	40	60	60	60	60	60	50	60	40	40	60	60	50	40	
Glass	23.83	60	60	50	50	60	50	50	60	60	60	60	50	50	60	60	60	70	
Splice_junction	24.10	40	40	50	30	30	40	30	50	30	30	30	40	30	30	30	40	30	
Yeast	28.90	50	50	60	40	40	40	40	50	60	50	50	50	60	60	50	50	60	
Breast-y	29.72	50	50	50	60	50	60	60	50	50	50	40	50	50	40	40	40	40	
Credit-g	30.00	50	60	40	40	50	60	40	60	50	50	50	60	40	60	50	40	60	
Car	30.00	30	40	50	40	40	50	50	40	40	30	40	40	20	20	40	40	20	
Iris	33.33	95	95	95	90	90	90	90	90	90	95	95	95	90	90	90	90	90	
Breast-w	34.48	60	60	50	60	70	50	50	70	50	40	50	50	30	30	40	30	30	
Pima	34.90	50	50	40	40	40	40	40	60	60	50	50	50	40	40	40	50	60	
Heart-h	36.05	60	50	50	50	60	60	70	60	70	60	60	50	60	70	70	50	70	
Voting	38.62	30	30	40	30	20	20	30	30	30	38.62	50	40	40	50	50	50	50	
Spam	39.40	40	40	40	30	30	30	40	50	50	40	40	50	50	50	50	50	50	
Lymph	41.22	50	50	50	40	60	60	40	50	50	50	50	50	40	40	40	50	40	
Liver	42.03	50	50	50	50	60	60	50	50	60	50	50	50	50	60	60	50	40	
Bands	42.20	50	40	60	40	40	40	40	40	40	50	50	50	50	50	40	50	40	
Credit-a	44.49	44.49	44.49	44.49	30	30	44.49	30	44.49	30	30	30	40	30	20	20	40	20	
Heart-c	45.87	50	50	50	50	60	60	40	50	40	50	50	50	60	40	40	50	40	
Kr-vs-kp	47.80	50	50	60	40	60	60	50	40	40	47.80	50	50	40	40	50	50	50	
Mean	27.64	50.33	49.98	49.64	46.71	48.62	50.00	49.12	54.14	50.15	50.52	50.17	51.21	49.31	50.69	49.66	49.66	53.10	
Std Dev	12.66	11.58	11.43	12.18	12.84	15.75	14.64	13.90	12.11	14.30	11.83	11.38	11.15	13.87	15.10	13.49	13.49	10.72	
#orm=50	13	14	15	6	6	7	7	13	7	7	17	18	16	10	8	7	5	12	
#orb=oed		21	11	11	9	5	6	13	9	9		22	13	11	7	10	6	14	

5 Conclusions and further work

The aim of this work was to offer an approach (or methodology) that would help us to obtain better results in the context of machine learning by resampling data to obtain an optimum class distribution.

Most of the resampling methods are used to balance the classes to solve class imbalance problems. However, in this paper we have proposed using a class distribution value other than 50 % to improve the results, whatever the original class distribution may be. Our approach can be explained in a simple algorithmic way:

1. Determine the best class distribution for a data set using the simplest and fastest resampling method (random subsampling) for 14 different class distribution values, from 2 to 98 %.

2. Use the selected resampling method with this optimal class distribution and, also, with two values around it (± 10) to select the best value within this range.

The results obtained show that our methodology finds a class distribution which gives better results than the balanced one with statistically significant differences (in many cases) for eight resampling methods and two learning algorithms. Based on these results, we can conclude that our hypothesis is satisfied with independence from the resampling method used and learning algorithm.

Since the use of this approach is not restricted to imbalanced data sets, we can conclude that when we want to address a learning problem, it is always worth changing the class distribution of the sample to be used for training, using any resampling method for this. Although beyond the context of this work, our experiment showed that a simple

and non-intelligent resampling method, random subsampling without size limitations (RANSUB), can achieve very competitive results.

As a complementary result we have also shown the results of ENN-SMOTE, a variant of the known SMOTE-ENN method, where the ENN cleaning process is applied before SMOTE. Its performance is similar to SMOTE-ENNs performance (in the context of the data sets used in this paper) but its computational cost is lower, since ENN reduces the size of the sample before applying SMOTE. In addition, the same distance matrix can be used after applying ENN. Therefore, a way to analyze this drawback of SMOTE-ENN could be investigated in the future, as previously stated.

In addition to proposing the methodology, we answered some previously mentioned questions. It has been shown that each database has its own optimal class distribution, which may or may not be other than 50 %. This is not an important factor for our methodology, since it will be found even if it is 50 %. However, this optimal value also depends on the resampling method used, which is taken into account in the second step of the methodology. We are conscious that this value could be improved, since we search for it within a small range of values. As a related further work, we have it in mind to make a trade-off between the broadening of the range to search the optimal value and the time taken to perform it. Moreover, in order to make this search, the use of optimization techniques such as genetic algorithms, simulated annealing, etc. would be of interest. These kinds of techniques could be also applied in the search of the Step 1 or this step could even be left out.

Another important question has been answered. We wanted to know whether it is worth resampling a non-imbalanced data set. As the improvement obtained for non-imbalanced data sets has been as significant as that obtained for imbalanced ones, we can be sure that the methodology guides us to a better result whatever the original class distribution.

Acknowledgments This work was funded by the University of the Basque Country, general funding for research groups, ALDAPA (GIU10/02); by the Science and Education Department of the Spanish Government, ModelAccess (TIN2010 - 15549 project) and by the Basque Government's SAIOTEK program, Dataacc (S-PE11UN097). The *lymphography* domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data. Finally, we would like to thank Francisco Herrera for being always willing to help us and his helpful comments.

Appendix 1: Additional results

Table 9 shows average AUC values achieved with the PART algorithm for the 29 databases when the SMOTE resampling method and validation data-based estimation are used. In

addition, the table also shows as a reference the results for the non-resampling method (ORIGINAL). For the ORIGINAL case we can find the original class distribution of the data set (%C=orig) and average AUC value (AUC). For SMOTE we can find the results obtained with this resampling method when the balanced distribution is used (%C=50), when the optimal class distribution obtained in the step 1 is used (optimal class distribution using random subsampling, %C=opt column), and its average AUC value for SMOTE, AUC column) and when the optimal class distribution for SMOTE obtained in step 2 is used (optimal class distribution using SMOTE, %C=opt column, and its average AUC value, AUC column).

With regard to the AUC values, the results improving those obtained with the whole training sample (ORIGINAL) have a grey background, while for class distribution values the cells that have a (lighter) grey background indicate that the optimal class distribution is not balanced (50 %).

The last two rows show the average (*Mean*) and median (*Median*) values, for AUC and class distribution, calculated for the 29 data sets. The reader can find here the *bal*, *ocd* and *orm* values (84.44, 84.76 and 85.23, respectively) for the PART algorithm, the AUC metric and the SMOTE method appearing in Table 5 in the main text.

Appendix 2: CD diagrams

This appendix includes the CD diagrams for C4.5 and PART. We included a CD diagram for each resampling method we analyzed. Demšar proposed a visual representation his post-hoc tests when multiple algorithms are compared: CD (critical difference) diagrams. The CD diagrams show graphically the existence or otherwise of significant differences based on the distance between the average ranks of two algorithms. For the Nemenyi test and 95 % significance level (for 4 algorithms and 29 data sets) a critical value (maximum distance) of 0.8710 is defined and for the Bonferroni-Dunn test (under the same conditions) the critical value is 0.8116. Both these critical values have been graphically represented in the CD diagrams in the upper left and lower right corners of Figs. 4 and 5 in order to provide a visual reference.

The Nemenyi test is used to perform comparisons of all algorithms against each other. In this kind of CD diagram groups of algorithms that are not significantly different are connected with a line (above the *x*-axis). The Bonferroni-Dunn test is also used to perform comparisons of one algorithm (the control algorithm) against the others. All algorithms with ranks outside the marked interval (below the *x*-axis) are significantly different from the control option. In this case, we have included two control options in each CD diagram, which have been drawn at two different heights: the best (first) option and the ORIGINAL case.

Table 9 Average AUC values, 5×10 CV for PART sets of rules using SMOTE resampling method

			%c=50	SMOTE			
	ORIGINAL			Step1		Step2	
	%C=orig	AUC		AUC	%C=opt	AUC	%C=opt
Hypo	4.77	71.92	85.31	50	85.31	60	87.04
Abalone	8.70	86.50	95.64	60	95.87	50	95.64
Sick_euthyroid	9.26	54.83	66.54	60	68.17	70	69.95
Optdigits	9.90	90.85	93.03	60	93.49	50	93.03
Soybean_large	13.79	61.43	87.40	50	87.40	60	89.65
Segment210	14.29	96.60	99.34	50	99.34	60	99.38
Segment2310	14.29	83.76	92.34	50	92.34	40	92.94
Solar_flare	15.70	52.94	62.26	50	62.26	40	63.89
Kddcup	19.69	98.62	99.39	50	99.39	60	99.43
Hepatitis	20.65	67.09	62.90	50	62.90	50	62.90
Vehicle	23.52	90.04	94.29	50	94.29	60	94.51
Glass	23.83	85.84	90.60	60	90.71	60	90.71
Splice_junction	24.10	95.35	95.84	30	96.65	30	96.65
Yeast	28.90	67.87	73.29	50	73.29	60	73.74
Breast-y	29.72	59.46	59.92	50	59.92	40	60.53
Credit-g	30.00	55.91	69.27	50	69.27	60	68.41
Car	30.00	93.54	97.61	30	98.41	20	98.94
Iris	33.33	98.08	99.50	95	99.50	90	99.50
Breast-w	34.48	95.53	95.75	40	96.29	30	96.93
Pima	34.90	71.33	74.79	50	74.79	50	74.79
Heart-h	36.05	77.11	80.25	60	82.21	70	85.54
Voting	38.62	96.79	97.69	50	97.69	50	97.69
Spam	39.40	91.87	93.73	40	93.52	50	93.73
Lymph	41.22	78.12	80.95	50	80.95	40	81.52
Liver	42.03	60.84	65.02	50	65.02	50	65.02
Bands	42.20	72.14	73.72	50	73.72	50	73.72
Credit-a	44.49	87.86	86.37	30	89.49	20	89.61
Heart-c	45.87	76.19	76.37	50	76.37	60	76.60
Kr-vs-kp	47.80	99.31	99.59	50	99.59	40	99.62
Mean	27.64	79.92	84.44	50.52	84.76	50.69	85.23
Median	29.72	83.76	87.40	50.00	89.49	50.00	89.65

We present values for the 29 databases

Italic values indicate that the cases when the optimal class distribution is not balanced

Bold values indicate that the results improving those obtained with the original class distribution

Fig. 4 C4.5 algorithm. Validation data-based estimation. CD diagrams for the eight resampling methods related to AUC evaluation metric

Fig. 5 PART algorithm. Validation data-based estimation. CD diagrams for the eight resampling methods related to AUC evaluation metric

Appendix 3: Danger case detection algorithm

This appendix includes the algorithm we have implemented to determine which cases in the minority class are in danger for both the Borderline-SMOTE1 and Borderline-SMOTE2 methods, since this is not specified in the original work [25].

Algorithm 1 Danger case detection

Require:

S: data sample

m: number of neighbors to take into account (default value=5)

```

1: CasesInDanger = 0;
2: MinClassSize = CountNumberOfMinCases(S);
3: while ((CasesInDanger < MinClassSize/2) and (m < 2*(MinClassSize-1))) do
4:   foreach minority class example e in the sample do
5:     MajNeighbors = DetermineNumberOfMajNeighbors(m, e);
6:     if (MajNeighbors >= m/2) then
7:       MarkExampleAsCaseInDanger(e);
8:       CasesInDanger = CasesInDanger + 1;
9:     end if
10:  end for
11:  m = m * 2;
12: end while

```

References

- Albisua, I., Arbelaitz, O., Gurrutxaga, I., Lasarguren, A., Muguerza, J., Pérez, J.: Analysis of the effect of changes in class distribution in C4.5 and consolidated C4.5 tree learners. Tech. Rep. EHU-KAT-IK-01-12, University of the Basque Country (UPV-EHU) (2012). <http://www.sc.ehu.es/acwaldap/Argitalpenak/12.EHU-KAT-IK-01-12.pdf>
- Albisua, I., Arbelaitz, O., Gurrutxaga, I., Martín, J., Muguerza, J., Pérez, J., Perona, I.: Obtaining optimal class distribution for decision trees: comparative analysis of CTC and C4.5. In: Proceedings of the Current Topics in Artificial Intelligence, and 13th Conference on Spanish Association for Artificial Intelligence, CAEPIA'09, pp. 101–110 (2010)
- Albisua, I., Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J.: C4.5 consolidation process: an alternative to intelligent oversampling methods in class imbalance problems. In: Proceedings of the 14th International Conference on Advances in Artificial intelligence: Spanish Association for Artificial Intelligence, CAEPIA'11, pp. 74–83 (2011)
- Artis, M., Ayuso, M., Guillen, M.: Modelling different types of automobile insurance fraud behaviour in the Spanish market. *Insur. Math. Econ.* **24**(1–2), 67–81 (1999)
- Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor.* **6**, 20–29 (2004)
- Berry, M., Linoff, G.: *Mastering Data Mining: The Art and Science of Customer Relationship Management*. Wiley, New York (2000)
- Bouckaert, R.R., Frank, E., Hall, M.A., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., Sonnenburg, S.: WEKA—experiences with a java open-source project. *J. Mach. Learn. Res.* **11**, 2533–2541 (2010)
- Chan, P., Stolfo, S.: Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In: Proc. of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 164–168 (1998)
- Chawla, N.: C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In: Proceedings of the International Conference on Machine Learning (ICML'03) Workshop on Learning from Imbalanced Datasets (2003)
- Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002)
- Chawla, N., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor.* **6**(1), 1–6 (2004)
- Chawla, N.V., Cieslak, D.A., Hall, L.O., Joshi, A.: Automatically countering imbalance and its empirical relationship to cost. *Data Min. Knowl. Discov.* **17**(2), 225–252 (2008)
- Cohen, W.W.: Fast effective rule induction. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 115–123 (1995)
- Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)
- Domingos, P.: Metacost: a general method for making classifiers cost-sensitive. In: Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, pp. 155–164. (1999)
- Elkan, C.: The foundations of cost-sensitive learning. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pp. 973–978 (2001)
- Estabrooks, A., Jo, T., Japkowicz, N.: A multiple resampling method for learning from imbalanced data sets. *Comput. Intell.* **20**(1), 18–36 (2004)
- Fawcett T ROC graphs: Notes and practical considerations for data mining researchers. Tech. rep., HP Laboratories (2004). http://home.comcast.net/tom.fawcett/public_html/papers
- Fernández, A., García, S., Herrera, F.: Addressing the classification with imbalanced data: Open problems and new challenges on class distribution. In: Proceeding of the 6th International Conference on Hybrid Artificial Intelligent Systems, HAIS'11, pp. 1–10 (2011)
- Frank, A., Asuncion, A.: UCI machine learning repository. <http://archive.ics.uci.edu/ml> (2010)
- Frank, E., Witten, I.: Generating accurate rule sets without global optimization. In: Shavlik, J. (ed.) Proceedings of the 15th International Conference on Machine Learning, pp. 144–151 (1998)
- García, S., Fernández, A., Herrera, F.: Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems. *Appl. Soft Comput.* **9**, 1304–1314 (2009)
- García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Inf. Sci.* **180**, 2044–2064 (2010)
- García, S., Herrera, F.: An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. *J. Mach. Learn. Res.* **9**, 2677–2694 (2008)
- Han, H., Wang, W., Mao, B.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D.S., Zhang, X.P., Huang, G.B. (eds.) Advances in Intelligent Computing, International Conference on Intelligent Computing, ICIC 2005, vol. 3644, pp. 878–887. (2005)
- Hastie, T., Tibshirani, R., Friedman, J.: *The elements of statistical learning*. Springer, New York (2003)
- He, H., Bai, Y., Garcia, E., Li, S.: ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IJCNN 2008, pp. 1322–1328 (2008)

28. He, H., Garcia, E.: Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* **21**(9), 1263–1284 (2009)
29. Japkowicz, N.: Learning from imbalanced data sets: a comparison of various, strategies. pp. 10–15 (2000)
30. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. *Intell. Data Anal.* **6**(5), 429–449 (2002)
31. Joshi, M.V., Kumar, V., Agarwal, R.C.: Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In: *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM '01*, pp. 257–264 (2001)
32. Luengo, J., Fernández, A., García, S., Herrera, F.: Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling. *Soft Comput.* **15**(10), 1909–1936 (2011)
33. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. *J. Mach. Learn. Res.* **2**, 139–154 (2002)
34. Orriols-Puig, A., Bernadó-Mansilla, E.: Evolutionary rule-based systems for imbalanced data sets. *Soft Comput.* **13**, 213–225 (2008)
35. Quinlan, J.R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, Burlington (1993)
36. Ramentol, E., Caballero, Y., Bello, R., Herrera, F.: SMOTE-RS*: A hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. In: *Knowledge and Information Systems*, pp. 1–21 (2012). doi:[10.1007/s10115-011-0465-6](https://doi.org/10.1007/s10115-011-0465-6)
37. Wang, B., Japkowicz, N.: Boosting support vector machines for imbalanced data sets. *Knowl. Inf. Syst.* **25**(1), 1–20 (2010)
38. Weiss, G.M., Provost, F.: Learning when training data are costly: the effect of class distribution on tree induction. *J. Artif. Intell. Res.* **19**(1), 315–354 (2003)
39. Wilson, D., Martinez, T.: Reduction techniques for exemplar-based learning algorithms. *Mach. Learn.* **38**(3), 257–286 (2000)
40. Wu, G., Chang, E.: Class-boundary alignment for imbalanced dataset learning. In: *Proceedings of the International Conference on Machine Learning (ICML'03) Workshop on Learning from Imbalanced Datasets*, pp. 49–56 (2003)
41. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. *Knowl. Inf. Syst.* **14**(1), 1–37 (2008)
42. Yang, Q., Wu, X.: 10 challenging problems in data mining research. *Int. J. Inf. Technol. Decis. Mak.* **5**(4), 597–604 (2006)
43. Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: *Proceeding of the Seventh International Conference on Knowledge Discovery and Data Mining*, pp. 204–213 (2001)