A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data

Gustavo E. A. P. A. Batista
Ronaldo C. Prati
Maria Carolina Monard
Instituto de Ciências Matemáticas e de Computação
Caixa Postal 668, 13560-970
São Carlos - SP, Brazil
{gbatista, prati, mcmonard}@icmc.usp.br

ABSTRACT

There are several aspects that might influence the performance achieved by existing learning systems. It has been reported that one of these aspects is related to class imbalance in which examples in training data belonging to one class heavily outnumber the examples in the other class. In this situation, which is found in real world data describing an infrequent but important event, the learning system may have difficulties to learn the concept related to the minority class. In this work we perform a broad experimental evaluation involving ten methods, three of them proposed by the authors, to deal with the class imbalance problem in thirteen UCI data sets. Our experiments provide evidence that class imbalance does not systematically hinder the performance of learning systems. In fact, the problem seems to be related to learning with too few minority class examples in the presence of other complicating factors, such as class overlapping. Two of our proposed methods, Smote + Tomek and Smote + ENN, deal with these conditions directly, allying a known over-sampling method with data cleaning methods in order to produce better-defined class clusters. Our comparative experiments show that, in general, over-sampling methods provide more accurate results than under-sampling methods considering the area under the ROC curve (AUC). This result seems to contradict results previously published in the literature. Smote + Tomek and Smote + ENN presented very good results for data sets with a small number of positive examples. Moreover, Random over-sampling, a very simple over-sampling method, is very competitive to more complex over-sampling methods. Since the over-sampling methods provided very good performance results, we also measured the syntactic complexity of decision trees induced from over-sampled data. Our results show that these trees are usually more complex then the ones induced from original data. Random over-sampling usually produced the smallest increase in the mean number of induced rules and Smote + ENN the smallest increase in the mean number of conditions per rule, when compared among the investigated over-sampling methods.

1. INTRODUCTION

Most learning systems usually assume that training sets used for learning are balanced. However, this is not always the case in real world data where one class might be represented by a large number of examples, while the other is represented by only a few. This is known as the class imbalance problem and is often reported as an obstacle to the induction of good classifiers by Machine Learning (ML) algorithms. Generally, the problem of imbalanced data sets occurs when one class represents a circumscribed concept, while the other class represents the counterpart of that concept, so that examples from the counterpart class heavily outnumber examples from the class of interest. This sort of data is found, for example, in medical record databases regarding a rare disease, were there is a large number of patients who do not have that disease; continuous fault-monitoring tasks where non-faulty examples heavily outnumber faulty examples, and others.

In recent years, there have been several attempts at dealing with the class imbalance problem in the field of Data Mining and Knowledge Discovery in Databases, to which ML is a substantial contributor. Related papers have been published in the ML literature aiming to overcome this problem. The ML community seems to agree on the hypothesis that the imbalance between classes is the major obstacle in inducing classifiers in imbalanced domains. However, it has also been observed that in some domains, for instance the Sick data set [3], standard ML algorithms are capable of inducing good classifiers, even using highly imbalanced training sets. This shows that class imbalance is not the only problem responsible for the decrease in performance of learning algorithms.

In [18] we developed a systematic study aiming to question whether class imbalances hinder classifier induction or whether these deficiencies might be explained in other ways. Our study was developed on a series of artificial data sets in order to fully control all the variables we wanted to analyze. The results of our experiments, using a discrimination-based inductive scheme, suggested that the problem is not solely caused by class imbalance, but is also related to the degree of data overlapping among the classes.

The results obtained in this previous work motivated the proposition of two new methods to deal with the problem of learning in the presence of class imbalance. These methods ally a known over-sampling method, namely Smote [5], with two data cleaning methods: Tomek links [22] and Wilson's Edited Nearest Neighbor Rule [24]. The main motivation behind these methods is not only to balance the training

data, but also to remove noisy examples lying on the wrong side of the decision border. The removal of noisy examples might aid in finding better-defined class clusters, therefore, allowing the creation of simpler models with better generalization capabilities.

In addition, in this work we perform a broad experimental evaluation involving ten methods, three of them proposed by the authors, to deal with the class imbalance problem in thirteen UCI data sets. We concluded that over-sampling methods are able to aid in the induction of classifiers that are more accurate than those induced from under-sampled data sets. This result seems to contradict results previously published in the literature. Two of our proposed methods performed well in practice, in particular for data sets with a small number of positive examples. It is worth noting that Random over-sampling, a very simple over-sampling method, is very competitive to more complex over-sampling methods.

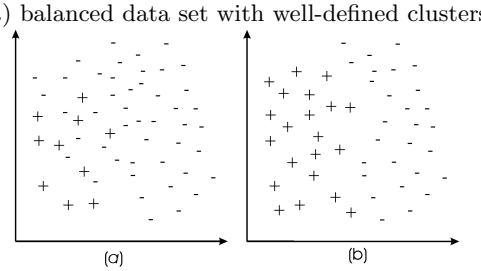
The remainder of the paper is organized as follows: Section 2 discusses why learning from imbalanced data sets might be a difficult task. Section 3 describes the drawbacks of using accuracy (or error rate) to measure the performance of classifiers and discusses alternative metrics. Section 4 presents the methods employed in the experimental evaluation, including the three methods proposed by the authors. Section 5 discusses the methodology used in the experiments, as well as the results achieved. Finally, Section 6 presents the conclusions and outlines future research.

2. WHY LEARNING FROM IMBALANCED DATA SETS MIGHT BE DIFFICULT

Learning from imbalanced data sets is often reported as being a difficult task. In order to better understand this problem, imagine the situation illustrated in Figure 1. In Fig. 1(a) there is a large imbalance between the majority class (-) and the minority class (+), and the data set presents some degree of class overlapping. A much more comfortable situation for learning is represented in Fig. 1(b), where the classes are balanced with well-defined clusters.

In a situation similar to the one illustrated in Fig. 1(a), spare cases from the minority class may confuse a classifier like *k-Nearest Neighbor (k-NN)*. For instance, 1-NN may incorrectly classify many cases from the minority class because the nearest neighbors of these cases are examples belonging to the majority class. In a situation where the imbalance is very high, the probability of the nearest neighbor of a minority class case is a case of the majority class is likely to be high, and the minority class error rate will tend to have high values, which is unacceptable.

Figure 1: Many negative cases against some spare positive cases (a) balanced data set with well-defined clusters (b).



Decision trees also experience a similar problem. In the presence of class overlapping, decision trees may need to create many tests to distinguish the minority class cases from majority class cases. Pruning the decision tree might not necessarily alleviate the problem. This is due to the fact that pruning removes some branches considered too specialized, labelling new leaf nodes with the dominant class on this node. Thus, there is a high probability that the majority class will also be the dominant class of those leaf nodes.

3. ON EVALUATING CLASSIFIERS IN IMBALANCED DOMAINS

The most straightforward way to evaluate the performance of classifiers is based on the confusion matrix analysis. Table 1 illustrates a confusion matrix for a two-class problem having **positive** and **negative** class values. From such a matrix it is possible to extract a number of widely used metrics for measuring the performance of learning systems, such as Error Rate, defined as $Err = \frac{FP+FN}{TP+FN+FP+TN}$ and Accuracy, defined as $Acc = \frac{TP+TN}{TP+FN+FP+TN} = 1 - Err$.

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)

However, when the prior class probabilities are very different, the use of such measures might lead to misleading conclusions. Error rate and accuracy are particularly suspicious performance measures when studying the effect of class distribution on learning since they are strongly biased to favor the majority class. For instance, it is straightforward to create a classifier having an accuracy of 99% (or an error rate of 1%) in a domain where the majority class proportion corresponds to 99% of the examples, by simply forecasting every new example as belonging to the majority class.

Another fact against the use of accuracy (or error rate) is that these metrics consider different classification errors to be equally important. However, highly imbalanced problems generally have highly non-uniform error costs that favor the minority class, which is often the class of primary interest.

Finally, another point that should be considered when studying the effect of class distribution on learning systems is that the class distribution may change. Consider the confusion matrix shown in Table 1. Note that the class distribution (the proportion of positive to negative examples) is the relationship between the first and second lines. Any performance metric that uses values from both lines will be inherently sensitive to class skews. Metrics such as accuracy and error rate use values from both lines of the confusion matrix. As class distribution changes, these measures will change as well, even if the fundamental classifier performance does not. All things considered, it would be more interesting if we use a performance metric that disassociates the errors (or hits) that occurred in each class. From Table 1 it is possible to derive four performance metrics that directly measure the classification performance on positive and negative classes independently:

False negative rate $FN_{rate} = \frac{FN}{TP+FN}$ is the percentage of positive cases misclassified 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;

True negative rate $TN_{rate} = \frac{TN}{FP+TN}$ is the percentage of negative cases correctly classified as belonging to the negative class;

True positive rate $TP_{rate} = \frac{TP}{TP+FN}$ is the percentage of positive cases correctly classified as belonging to the positive class.

These four performance measures have the advantage of being independent of class costs and prior probabilities. The aim of a classifier is to minimize the false positive and negative rates or, similarly, to maximize the true negative and positive rates. Unfortunately, for most real world applications there is a tradeoff between FN_{rate} and FP_{rate}, and similarly between TN_{rate} and TP_{rate}. ROC (Receiver Operating Characteristic) graphs [19] can be used to analyze the relationship between FN_{rate} and FP_{rate} (or TN_{rate} and TP_{rate}) for a classifier.

Some classifiers, such as the Naïve Bayes classifier or some Neural Networks, yield a score that represents the degree to which an example is a member of a class. Such ranking can be used to produce several classifiers, by varying the threshold of an example pertaining to a class. Each threshold value produces a different point in the ROC space. These points are linked by tracing straight lines through two consecutive points to produce a ROC curve. For decision trees, we could use the class distributions at each leaf as a score or, as proposed in [9], by ordering the leaves by their positive class accuracy and producing several trees by re-labelling the leaves, one at a time, from all forecasting negative class to all forecasting positive class in the positive accuracy order.

A ROC graph characterizes the performance of a binary classification model across all possible trade-offs between the classifier sensitivity (TP_{rate}) and false alarm (FP_{rate}). ROC graphs are consistent for a given problem, even if the distribution of positive and negative examples is highly skewed. A ROC analysis also allows the performance of multiple classification functions to be visualized and compared simultaneously. The area under the ROC curve (AUC) represents the expected performance as a single scalar. The AUC has a known statistical meaning: it is equivalent to the Wilcoxon test of ranks, and is equivalent to several other statistical measures for evaluating classification and ranking models [10]. In this work, we use the method proposed in [9] with Laplace correction for measuring the leaf accuracy to produce ROC curves. We also use the AUC as the main method for assessing our experiments.

4. METHODS

This section describes the notation used as well as our implementation of the k-NN algorithm, since this algorithm plays an important role in the behavior of the methods considered. Finally, an explanation of each balancing method is given.

4.1 Notation

In supervised learning, the inducer is fed with a data set $E = \{E_1, E_2, \ldots E_N\}$, in which each example $E_i \in E$ has an associated label. This label defines the class the example belongs to. Each example $E_i \in E$ is a tuple $E_i = (\vec{x_i}, y_i)$ in which $\vec{x_i}$ is a vector of feature (or attribute) values of the example E_i, and y_i is its class value. The objective of a supervised learning algorithm is to induce a general mapping of vectors $\vec{x}$ to values y. Thus, the learning system aims to construct a model $y = f(\vec{x})$, of an unknown function f, also known as *concept function*, that enables one to predict the y values for previously unseen examples. In practice, learning systems are able to induce a function $\mathbf{h}$ that approximates f, *i.e.*, $\mathbf{h}(\vec{x}) \approx f(\vec{x})$. In this case, $\mathbf{h}$ is called the *hypothesis* of the concept function f.

In Table 2 a data set with N examples and M attributes is presented. Columns $(A_1, \ldots A_M)$ represent the attributes and lines $(E_1, \ldots E_N)$ represent the examples. For instance, line i in Table 2 refers to the ith example and the entry x_{ij} refers to the value of the jth attribute, A_j, of example i. For classification problems, the class-attribute Y is a qualitative attribute that may assume a set of N_{Cl} discrete values $C = \{C_1, C_2, \ldots C_{N_{Cl}}\}$.

Table 2: Data set in attribute-value form.

	A_1	A_2	$\cdots$	A_M	Y
E_1	x_{11}	x_{12}	$\cdots$	x_{1M}	y_1
E_2	x_{21}	x_{22}	$\cdots$	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	x_{N1}	x_{N2}	$\ldots$	x_{NM}	y_N

As stated earlier, in this work we consider two-class problems where $C_1 = +$ represents the circumscribed concept class and $C_2 = -$ represents the counterpart of that concept. Furthermore, the examples from the negative class outnumber the examples from the positive class.

4.2 Our Implementation of the k-NN Algorithm

Several research papers use the Euclidean distance as a distance metric for the k-NN algorithm. However, this distance function might not be appropriate when the domain presents qualitative attributes. For those domains, the distance for qualitative attributes is usually calculated using the overlap function, in which the value 0 (if two examples have the same value for a given attribute) or the value 1 (if these values differ) are assigned. Our implementation of the k-NN algorithm uses the *Heterogeneous Value Difference Metric (HVDM)* distance function [25]. This distance function uses the Euclidean distance for quantitative attributes and the VDM distance [21] for qualitative attributes. The VDM metric provides a more appropriate distance function for qualitative attributes if compared with the overlap metric, since the VDM metric considers the classification similarity for each possible value of a qualitative attribute to calculate the distances between these values.

Another refinement to the basic k-NN algorithm is to weigh the contribution of each of the k neighbors according to their distance to the query example E_q, giving greater weight to closer neighbors. The vote of each neighbor is weighed according to the inverse square of its distance from E_q [17]. Given $\hat{E} = \{\hat{E}_1, \hat{E}_2, \ldots \hat{E}_k\}$, the set of k nearest neighbors of E_q, according to the distance function d the final classification is given by Equation 1.

$$\mathbf{h}(E_q) = \arg\max_{c \in C} \sum_{i=1}^{k} \omega_i \delta(c, f(\hat{E}_i)) \qquad \omega_i = \frac{1}{d(E_q, \hat{E}_i)^2} \quad (1)$$

and $\delta(a, b) = 1$ if $a = b$ otherwise $\delta(a, b) = 0$.
As the balancing methods make severe use of distance computations, we implemented an indexing structure namely M-tree [6] to speed up the execution of k-NN queries. M-tree only considers relative distances of examples rather than their absolute positions in a multi-dimensional space, to organize and partition the *metric space*. In a metric space, example proximity is only defined by a distance function that satisfies the positivity, symmetry and triangle inequality postulates.

4.3 Methods

In this work, we evaluate ten different methods of under and over-sampling to balance the class distribution on training data. Two of these methods, Random over-sampling and Random under-sampling, are non-heuristic methods that were initially included in this evaluation as baseline methods. The evaluated methods are described next.

Random over-sampling is a non-heuristic method that aims to balance class distribution through the random replication of minority class examples.

Random under-sampling is also a non-heuristic method that aims to balance class distribution through the random elimination of majority class examples.

Several authors agree that Random over-sampling can increase the likelihood of occurring overfitting, since it makes exact copies of the minority class examples. In this way, a symbolic classifier, for instance, might construct rules that are apparently accurate, but actually cover one replicated example. On the other hand, the major drawback of Random under-sampling is that this method can discard potentially useful data that could be important for the induction process. The remainder balancing methods use heuristics in order to overcome the limitations of the non-heuristic methods.

Tomek links Tomek links [22] can be defined as follows: given two examples E_i and E_j belonging to different classes, and $d(E_i, E_j)$ is the distance between E_i and E_j. A (E_i, E_j) pair is called a Tomek link if there is not an example E_l, such that $d(E_i, E_l) < d(E_i, E_j)$ or $d(E_j, E_l) < d(E_i, E_j)$. If two examples form a Tomek link, then either one of these examples is noise or both examples are borderline. Tomek links can be used as an under-sampling method or as a data cleaning method. As an under-sampling method, only examples belonging to the majority class are eliminated, and as a data cleaning method, examples of both classes are removed.

Condensed Nearest Neighbor Rule Hart's Condensed Nearest Neighbor Rule (CNN) [11] is used to find a consistent subset of examples. A subset $\hat{E} \subseteq E$ is consistent with E if using a 1-nearest neighbor, $\hat{E}$ correctly classifies the examples in E. An algorithm to create a subset $\hat{E}$ from E as an under-sampling method is the following [14]: First, randomly draw one majority class example and all examples from the minority class and put these examples in $\hat{E}$. Afterwards, use a 1-NN over the examples in $\hat{E}$ to classify the examples in E. Every misclassified example from E is moved to $\hat{E}$. It is important to note that this procedure does not find the smallest consistent subset from E. The idea behind this implementation of a consistent subset is to eliminate the examples from the majority class that are distant from the decision border, since these sorts of examples might be considered less relevant for learning.

One-sided selection One-sided selection (OSS) [14] is an under-sampling method resulting from the application of Tomek links followed by the application of CNN. Tomek links are used as an under-sampling method and removes noisy and borderline majority class examples. Borderline examples can be considered "unsafe" since a small amount of noise can make them fall on the wrong side of the decision border. CNN aims to remove examples from the majority class that are distant from the decision border. The remainder examples, *i.e.* "safe" majority class examples and all minority class examples are used for learning.

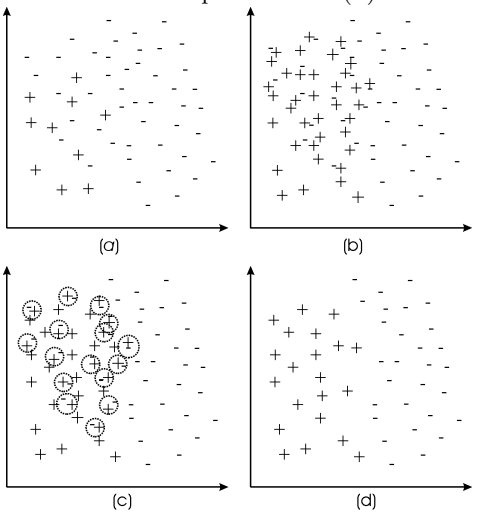
CNN + Tomek links This is one of the methods proposed in this work. It is similar to the one-sided selection, but the method to find the consistent subset is applied before the Tomek links. Our objective is to verify its competitiveness with OSS. As finding Tomek links is computationally demanding, it would be computationally cheaper if it was performed on a reduced data set.

Neighborhood Cleaning Rule Neighborhood Cleaning Rule (NCL) [15] uses the *Wilson's Edited Nearest Neighbor Rule (ENN)* [24] to remove majority class examples. ENN removes any example whose class label differs from the class of at least two of its three nearest neighbors. NCL modifies the ENN in order to increase the data cleaning. For a two-class problem the algorithm can be described in the following way: for each example E_i in the training set, its three nearest neighbors are found. If E_i belongs to the majority class and the classification given by its three nearest neighbors contradicts the original class of E_i, then E_i is removed. If E_i belongs to the minority class and its three nearest neighbors misclassify E_i, then the nearest neighbors that belong to the majority class are removed.

Smote Synthetic Minority Over-sampling Technique (Smote) [5] is an over-sampling method. Its main idea 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.

Smote + Tomek links Although over-sampling minority class examples can balance class distributions, some other problems usually present in data sets with skewed class distributions are not solved. Frequently, class clusters are not well defined since some majority class examples might be invading 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 in the majority class space. Inducing a classifier under such a situation can lead to overfitting. In order to create better-defined class clusters, we propose applying Tomek links to the over-sampled training set as a data cleaning method. Thus, instead of removing only the majority class examples that form Tomek links, examples from both classes are removed. The application of this method is illustrated in Figure 2. First, the original data set (a) is over-sampled with Smote (b), and then

Figure 2: Balancing a data set: original data set (a); over-sampled data set (b); Tomek links identification (c); and borderline and noise examples removal (d).



Tomek links are identified (c) and removed, producing a balanced data set with well-defined class clusters (d). The Smote + Tomek links method was first used to improve the classification of examples for the problem of annotation of proteins in Bioinformatics [1].

Smote + ENN The motivation behind this method is similar to Smote + Tomek links. ENN tends to remove more examples than the Tomek links does, so it is expected that it will provide a more in depth data cleaning. Differently from NCL which is an under-sampling method, ENN is used to remove examples from both classes. Thus, any example that is misclassified by its three nearest neighbors is removed from the training set.

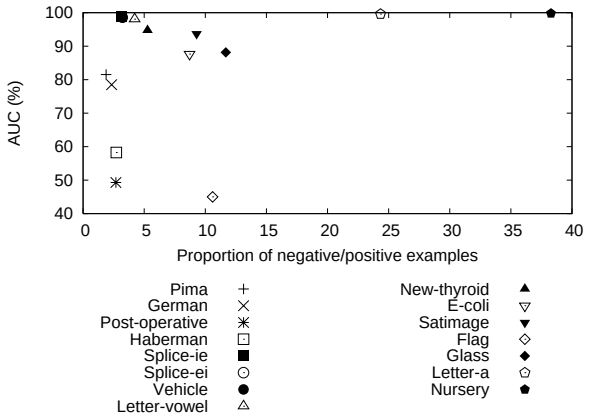
5. EXPERIMENTAL EVALUATION

The main objective of our research is to compare several balancing methods published in the literature, as well as the three proposed methods, in order to verify whether those methods can effectively deal in practice with the problem of class imbalance. To make this comparison, we have selected thirteen data sets from UCI [3] which have different degrees of imbalance. Table 3 summarizes the data employed in this study. For each data set, it shows the number of examples (#Examples), number of attributes (#Attributes), number of quantitative and qualitative attributes, class attribute distribution and the majority class error. For data sets having more than two classes, we chose the class with fewer examples as the positive class, and collapsed the remainder as the negative class. As the Letter and Splice data sets have a similar number of examples in the minority classes, we created two data sets with each of them: Letter-a and Letter-vowel, Splice-ie and Splice-ei.

In our experiments, we used release 8 of the C4.5 symbolic learning algorithm to induce decision trees [20]. Firstly, we ran C4.5 over the original (imbalanced) data sets and calculated the AUC for each data set using 10-fold cross-validation. The results obtained in this initial experiment are shown in a graph in Figure 3.

Figure 3 plots the proportion of negative/positive examples versus the mean AUC values for the original data sets. If class imbalances can systematically hinder the performance of imbalanced data sets, then it would be expected that AUC decreases for highly imbalanced data sets. However, in spite of a large degree of imbalance the data sets Letter-a and Nursery obtained almost 100% AUC.

Figure 3: Proportion of negative/positive examples versus AUC.



The results obtained in the UCI data sets seem to be compatible with previous work of the authors [18] conducted on a series of experiments with artificial domains, in which we varied the degree of overlapping between the classes. It was concluded that class imbalance, by itself, does not seem to be a problem, but when allied to highly overlapped classes, it can significantly decrease the number of minority class examples correctly classified. Domains with non-overlapping classes do not seem to be problematic for learning no matter the degree of imbalance. Moreover, in [12] Japkowicz performed several experiments on artificial data sets and concluded that class imbalances do not seem to systematically cause performance degradation. She concludes that the imbalance problem is a relative problem depending on both the complexity of the concept[1] and the overall size of the training set.

The relationship between training set size and improper classification performance for imbalanced data sets seems to be that on small imbalanced data sets the minority class is poorly represented by an excessively reduced number of examples, that might not be sufficient for learning, especially when a large degree of class overlapping exists and the class is further divided into subclusters. For larger data sets, the effect of these complicating factors seems to be reduced, as the minority class is better represented by a larger number of examples. This trend is confirmed by the graph shown in Figure 4 which shows how the AUC is affected by the number of positive training examples in the data sets.

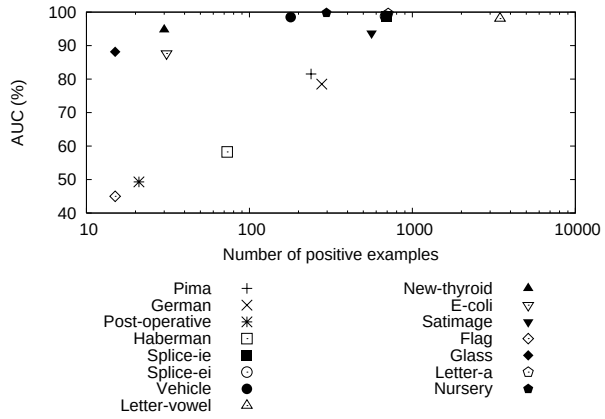
In a second stage, the over and under-sampling methods described in Section 4 were applied to the original data sets. Smote, Random over-sampling, Random under-sampling and CNN methods have internal parameters that allow the user to set up the resulting class distribution obtained after the application of these methods. We decided to add/remove examples until a balanced distribution was reached. This

[1]Where the "concept complexity" corresponds to the number of subclusters into which the classes are subdivided.

Table 3: Data sets summary descriptions.

Data set	#Examples	#Attributes (quanti., quali.)	Class (min., maj.)	Class % (min., maj.)	Majority Error
Pima	768	8 (8,0)	(1, 0)	(34.77%, 65.23%)	65.23%
German	1000	20 (7,13)	(Bad, Good)	(30.00%, 70.00%)	70.00%
Post-operative	90	8 (1,7)	(S, remainder)	(26.67%, 73.33%)	73.33%
Haberman	306	3 (3,0)	(Die, Survive)	(26.47%, 73.53%)	73.53%
Splice-ie	3176	60 (0,60)	(ie, remainder)	(24.09%, 75.91%)	75.91%
Splice-ei	3176	60 (0,60)	(ei, remainder)	(23.99%, 76.01%)	76.01%
Vehicle	846	18 (18,0)	(van, remainder)	(23.52%, 76.48%)	76.48%
Letter-vowel	20000	16 (16,0)	(all vowels, remainder)	(19.39%, 80.61%)	80.61%
New-thyroid	215	5 (5,0)	(hypo, remainder)	(16.28%, 83.72%)	83.72%
E.Coli	336	7 (7,0)	(iMU, remainder)	(10.42%, 89.58%)	89.58%
Satimage	6435	36 (36,0)	(4, remainder)	(9.73%, 90.27%)	90.27%
Flag	194	28 (10,18)	(white, remainder)	(8.76%, 91.24%)	91.24%
Glass	214	9 (9,0)	(Ve-win-float-proc, remainder)	(7.94%, 92.06%)	92.06%
Letter-a	20000	16 (16,0)	(a, remainder)	(3.95%, 96.05%)	96.05%
Nursery	12960	8 (8,0)	(not_recom, remainder)	(2.55%, 97.45%)	97.45%

Figure 4: Number of positive training examples versus AUC.



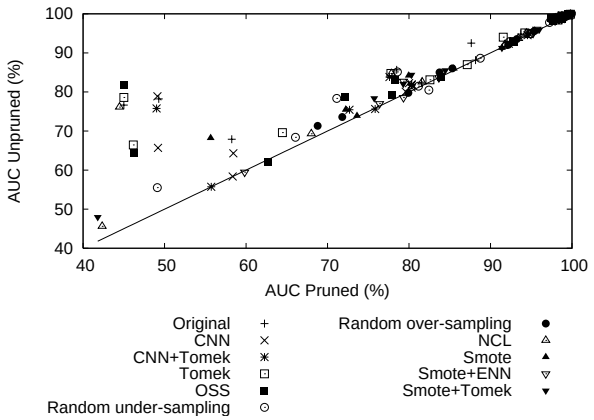
decision is motivated by the results presented in [23], in which it is shown that when AUC is used as performance measure, the best class distribution for learning tends to be near the balanced class distribution.

The results obtained in our experiments are summarized in Tables 4 and 5. Table 4 shows the performance results for the original, as well as for the over-sampled data sets. Table 5 shows the results obtained for the under-sampled data sets. The performance results are reported in terms of AUC. The numbers between brackets are the corresponding standard deviations. As stated earlier, these results were obtained with 10-fold cross-validation. AUCs were measured over decision trees pruned with the default C4.5 pruning parameter setting (25% confidence level) and over unpruned decision trees.

Although some research papers state that pruning might be helpful with imbalanced data sets in some circumstances [4], other papers indicate that when target misclassification costs or class distributions are unknown, then pruning should be avoided [26; 2]. One reason to avoid pruning is that most pruning schemes, including the one used by C4.5, attempt to minimize the overall error rate. These pruning schemes can be detrimental to the minority class, since reducing the error rate in the majority class, which stands for most of the examples, would result in a greater impact over the overall error rate. On the other hand, it still seems to be an open-ended question if pruning can lead to a performance improvement for decision trees grown over artificially balanced data sets. One argument against pruning is that if pruning is allowed to execute under these conditions, the learning system would prune based on false assumption, *i.e.*, that the test set distribution matches the training set distribution [23].

Figure 5 shows a comparison of the effect of pruning decision trees on the original and balanced data sets. Line $x = y$ represents when both pruned and unpruned decision trees obtain the same AUC. Plots above this line represent that unpruned decision trees obtained better results, and plots under this line the opposite. Figure 5 clearly shows that pruning rarely leads to an improvement in AUC for the original and balanced data sets.

Figure 5: AUC of pruned versus unpruned decision trees for the original and balanced data sets.



In Tables 4 and 5 the results in bold indicate the best AUCs obtained for each data set considering pruned and unpruned decision trees independently. In order to facilitate the analysis of the results, Tables 6 and 7 present these results as a ranking of methods for pruned and unpruned decision trees respectively. The over-sampling methods are highlighted with a light gray color, and the results obtained with the original data sets with a dark gray color. Note that, in general, over-sampling methods are better ranked than the under-sampling methods. Hsu's Multiple Comparison with the Best (MCB) test was performed in order to verify if significant differences exist, with 95% confidence level, among

Table 4: AUC results for the original and over-sampled data sets.

Data set	Pruning	Original	Rand Over	Smote	Smote+Tomek	Smote+ENN
Pima	yes	81.53(5.11)	85.32(4.17)	**85.49(5.17)**	84.46(5.84)	83.66(4.77)
	no	82.33(5.70)	**86.03(4.14)**	85.97(5.82)	85.56(6.02)	83.64(5.35)
German	yes	79.19(5.84)	**84.65(3.80)**	80.74(5.43)	81.75(4.78)	80.91(4.36)
	no	**85.94(4.14)**	85.56(4.31)	84.51(4.55)	84.02(3.94)	83.90(3.70))
Post-operative	yes	49.29(2.26)	**68.79(23.93)**	55.66(24.66)	41.80(16.59)	59.83(33.91)
	no	**78.23(15.03)**	71.33(23.43)	68.19(26.62)	47.99(16.61)	59.48(34.91)
Haberman	yes	58.25(12.26)	71.81(13.42)	72.23(9.82)	75.73(6.55)	**76.38(5.51)**
	no	67.91(13.76)	73.58(14.22)	75.45(11.02)	**78.41(7.11)**	77.01(5.10)
Splice-ie	yes	98.76(0.56)	**98.89(0.47)**	98.46(0.87)	98.26(0.51)	97.97(0.74)
	no	**99.30(0.30)**	99.09(0.27)	99.19(0.28)	99.13(0.31)	98.88(0.34)
Splice-ei	yes	98.77(0.46)	98.80(0.44)	**98.92(0.44)**	98.87(0.44)	98.85(0.60)
	no	99.47(0.61)	**99.52(0.60)**	**99.52(0.26)**	99.51(0.32)	99.49(0.16)
Vehicle	yes	98.49(0.84)	**99.14(0.73)**	98.96(0.98)	98.96(0.98)	97.92(1.09)
	no	98.45(0.90)	**99.13(0.75)**	99.04(0.85)	99.04(0.85)	98.22(0.90)
Letter-vowel	yes	98.07(0.63)	98.80(0.32)	98.90(0.20)	98.90(0.20)	**98.94(0.22)**
	no	98.81(0.33)	98.84(0.27)	99.15(0.17)	99.14(0.17)	**99.19(0.15)**
New-thyroid	yes	94.73(9.24)	98.39(2.91)	98.91(1.84)	98.91(1.84)	**99.22(1.72)**
	no	94.98(9.38)	98.89(2.68)	98.91(1.84)	98.91(1.84)	**99.22(1.72)**
E.Coli	yes	87.64(15.75)	93.24(6.72)	95.49(4.30)	**95.98(4.21)**	95.29(3.79)
	no	92.50(7.71)	93.55(6.89)	95.49(4.30)	**95.98(4.21)**	95.29(3.79)
Satimage	yes	93.73(1.91)	95.34(1.25)	95.43(1.03)	95.43(1.03)	**95.67(1.18)**
	no	94.82(1.18)	95.52(1.12)	95.69(1.28)	95.69(1.28)	**96.06(1.20)**
Flag	yes	45.00(15.81)	**79.91(28.72)**	73.62(30.16)	79.30(28.68)	79.32(28.83)
	no	76.65(27.34)	79.78(28.98)	73.87(30.34)	**82.06(29.52)**	78.56(28.79)
Glass	yes	88.16(12.28)	92.20(12.11)	91.40(8.24)	91.40(8.24)	**92.90(7.30)**
	no	88.16(12.28)	92.07(12.09)	91.27(8.38)	91.27(8.38)	**93.40(7.61)**
Letter-a	yes	99.61(0.40)	99.77(0.30)	**99.91(0.12)**	**99.91(0.12)**	**99.91(0.12)**
	no	99.67(0.37)	99.78(0.29)	**99.92(0.12)**	**99.92(0.12)**	99.91(0.14)
Nursery	yes	99.79(0.11)	**99.99(0.01)**	99.21(0.55)	99.27(0.36)	97.80(1.07)
	no	99.96(0.05)	**99.99(0.01)**	99.75(0.34)	99.53(0.31)	99.20(0.51)

the best ranked method and the remaining methods. The results are also summarized in Tables 6 and 7, where methods marked with an asterisk obtained statistically inferior results when compared to the top ranked method.

Conversely, over-sampling methods in general and Random over-sampling in particular are well-ranked among the remainder methods. This result seems to diverge with several papers previously published in the literature. Drummond and Holte [8] report that when using C4.5's default settings, over-sampling is surprisingly ineffective, often producing little or no change in performance in response to modifications of misclassification costs and class distribution. Moreover, they note that over-sampling prunes less and therefore generalizes less than under-sampling, and that a modification of the C4.5's parameter settings to increase the influence of pruning and other overfitting avoidance factors can reestablish the performance of over-sampling. In our experiments, Random over-sampling did not produce overfitted decision trees even when these trees were left unpruned, as it can be confirmed by the higher AUC values obtained by this method for unpruned trees. In addition, under-sampling methods did not perform as well as over-sampling methods, even when heuristics to remove cases were considered in under-sampling.

Moreover, Domingos [7] reports that concerning concept learning problems, C4.5 Rules produces lower cost classifiers using under-sampling than over-sampling. Ling and Li [16] compare over and under-sampling for boosted C4.5 and report that under-sampling produces better lift index, although extreme over-sampling performs almost as well. On the other hand, Japkowicz and Stephen [13] compare several methods of over and under-sampling on a series of artificial data sets and conclude that over-sampling is more effective than under-sampling at reducing error rate.

In our opinion, the good results obtained by over-sampling are not completely unexpected. As stated before, it seems that the loss of performance is directly related to the lack of minority class examples in conjunction with other complicating factors. Over-sampling is the class of methods that most directly attack the problem of the lack of minority class examples.

It is worth mentioning that two of our proposed methods, Smote + Tomek and Smote + ENN are generally ranked among the best for data sets with a small number of positive examples. Considering only data sets with less than 100 positive examples (in our experiments there are 6 of them: Flag, Glass, Post-operative, New-thyroid, E.Coli and Haberman), at least one of the proposed methods provided meaningful results for all 6 data sets for pruned trees – Table 6, and for 5 of the 6 data sets for unpruned trees – Table 7. This seems to indicate that these methods could be appropriate in domains having such conditions.

Since over-sampling methods, as well as unpruned decision trees obtained very good performance results, further analysis will focus on these results. In addition to classifier performance results, we also attempted to measure the syntactic complexity of the induced models. Syntactic complexity is given by two main parameters: the mean number of induced rules (branches) and the mean number of conditions per rule. Tables 8 and 9 respectively show the mean number of induced rules and the mean number of condition per rule for the over-sampling methods and the original data sets with unpruned decision trees. The best results are shown in bold, and the best results obtained by an over-sampling method, not considering the results obtained in the original data sets, are highlighted with a light gray color.

Figure 6 shows the results in Table 8 in graphical form, where it can be observed that over-sampled data sets usually lead to an increase in the number of induced rules if compared to the trees induced with the original data sets.

Table 5: AUC results for the under-sampled data sets.

Data set	Pruning	Rand Under	CNN	CNN+Tomek	Tomek	OSS	NCL
Pima	yes	81.17(3.87)	79.60(6.22)	80.30(3.86)	82.56(5.11)	77.89(5.37)	81.61(4.48)
	no	81.49(4.29)	80.08(5.82)	81.71(3.69)	83.11(4.65)	79.23(4.81)	82.55(3.53)
German	yes	79.85(3.05)	79.85(5.56)	79.48(5.01)	78.87(4.27)	79.20(3.15)	77.89(3.85)
	no	84.54(3.32)	82.25(5.59)	81.70(4.00)	85.90(3.99)	82.96(3.22)	85.07(3.54)
Post-operative	yes	49.11(14.07)	49.20(8.91)	49.02(11.34)	46.16(5.89)	46.31(18.77)	42.34(28.12)
	no	55.52(24.47)	65.69(21.64)	75.79(16.86)	66.45(23.29)	64.44(20.88)	45.62(32.71)
Haberman	yes	66.07(10.26)	58.36(10.26)	55.73(14.31)	64.46(10.95)	62.70(11.50)	68.01(13.99)
	no	68.40(10.17)	58.36(10.26)	55.73(14.31)	69.59(13.30)	62.03(11.82)	69.29(14.13)
Splice-ie	yes	97.46(1.10)	98.39(0.64)	97.55(0.46)	98.69(0.51)	97.37(0.84)	98.38(0.57)
	no	98.80(0.40)	99.17(0.36)	98.82(0.32)	99.18(0.43)	98.93(0.30)	99.15(0.36)
Splice-ei	yes	98.74(0.46)	98.78(0.46)	98.85(0.42)	98.78(0.46)	98.83(0.45)	98.77(0.47)
	no	99.25(0.48)	99.27(0.77)	99.47(0.27)	99.44(0.60)	99.33(0.66)	99.40(0.66)
Vehicle	yes	97.25(1.95)	98.62(0.67)	98.34(1.32)	98.26(0.90)	98.79(0.67)	97.94(1.05)
	no	97.80(0.94)	98.64(0.63)	98.42(1.02)	98.41(0.90)	98.71(0.97)	98.17(1.12)
Letter-vowel	yes	97.69(0.43)	98.03(0.37)	97.97(0.46)	98.18(0.53)	97.66(0.30)	98.17(0.30)
	no	98.26(0.28)	98.49(0.31)	98.39(0.22)	98.90(0.18)	98.27(0.19)	98.81(0.17)
New-thyroid	yes	94.87(5.00)	94.79(10.14)	94.54(10.10)	94.73(9.24)	92.72(10.55)	93.44(9.74)
	no	94.87(5.00)	94.79(10.14)	94.54(10.10)	94.98(9.38)	92.72(10.55)	93.69(9.90)
E.Coli	yes	88.75(12.45)	80.32(19.96)	80.34(19.85)	91.57(7.81)	83.97(21.27)	91.73(8.00)
	no	88.64(12.46)	81.13(20.00)	81.95(19.90)	94.03(5.56)	83.76(21.17)	92.04(8.15)
Satimage	yes	92.34(1.27)	92.25(1.45)	92.73(1.38)	94.21(1.76)	92.85(1.19)	94.42(1.53)
	no	92.86(1.29)	92.35(1.35)	92.90(1.38)	95.11(1.29)	92.84(1.22)	95.06(1.27)
Flag	yes	71.13(28.95)	49.12(21.57)	75.85(30.26)	45.00(15.81)	45.00(15.81)	44.47(15.71)
	no	78.35(29.98)	78.90(28.63)	75.64(29.37)	78.59(28.75)	81.73(29.51)	76.13(27.80)
Glass	yes	82.44(8.99)	58.44(13.15)	72.69(14.07)	87.15(16.47)	72.16(16.84)	91.67(12.76)
	no	80.47(13.25)	64.31(14.21)	75.44(11.61)	87.00(16.75)	78.76(12.52)	91.67(12.76)
Letter-a	yes	99.35(0.48)	99.60(0.37)	99.61(0.37)	99.61(0.40)	99.66(0.46)	99.60(0.40)
	no	99.46(0.42)	99.66(0.37)	99.65(0.38)	99.67(0.37)	99.67(0.45)	99.67(0.37)
Nursery	yes	97.52(0.82)	99.55(0.21)	98.77(0.35)	99.80(0.08)	99.47(0.19)	99.79(0.12)
	no	98.76(0.22)	99.84(0.13)	99.57(0.21)	99.89(0.08)	99.83(0.08)	99.89(0.09)

Table 6: Performance ranking for original and balanced data sets for pruned decision trees.

Data set	1°	2°	3°	4°	5°	6°	7°	8°	9°	10°	11°
Pima	Smt	RdOvr	Smt+Tmk	Smt+ENN	Tmk	NCL	Original	RdUdr	CNN+Tmk	CNN*	OSS*
German	RdOvr	Smt+Tmk	Smt+ENN	Smt	RdUdr	CNN	CNN+Tmk*	OSS*	Original*	Tmk*	NCL*
Post-operative	RdOvr	Smt+ENN	Smt	Original	CNN	RdUdr	CNN+Tmk	OSS*	Tmk*	NCL*	Smt+Tmk*
Haberman	Smt+ENN	Smt+Tmk	Smt	RdOvr	NCL	RdUdr	Tmk	OSS*	CNN*	Original*	CNN+Tmk*
Splice-ie	RdOvr	Original	Tmk	Smt	CNN	NCL	Smt+Tmk	Smt+ENN*	CNN+Tmk*	RdUdr*	OSS*
Splice-ei	Smt	Smt+Tmk	Smt+ENN	CNN+Tmk	OSS	RdOvr	Tmk	CNN	NCL	Original	RdUdr
Vehicle	RdOvr	Smt	Smt+Tmk	OSS	CNN	Original	CNN+Tmk	Tmk	NCL*	Smt+ENN*	RdUdr*
Letter-vowel	Smt+ENN	Smt+Tmk	Smt	RdOvr	Tmk*	NCL*	Original*	CNN*	CNN+Tmk*	RdUdr*	OSS*
New-thyroid	Smt+ENN	Smt+Tmk	Smt	RdOvr	RdUdr	CNN	Original	Tmk	CNN+Tmk	NCL	OSS
E.Coli	Smt+Tmk	Smt	Smt+ENN	RdOvr	NCL	Tmk	RdUdr	Original	OSS	CNN+Tmk*	CNN*
Satimage	Smt+ENN	Smt	Smt+Tmk	RdOvr	NCL	Tmk	Original*	OSS*	CNN+Tmk*	RdUdr*	CNN*
Flag	RdOvr	Smt+ENN	Smt+Tmk	CNN+Tmk	Smt	RdUdr	CNN*	OSS*	Tmk*	Original*	NCL*
Glass	Smt+ENN	RdOvr	NCL	Smt	Smt+Tmk	Original	Tmk	RdUdr	CNN+Tmk*	OSS*	CNN*
Letter-a	Smt+Tmk	Smt+ENN	Smt	RdOvr	OSS	Original	Tmk	CNN+Tmk	NCL	CNN	RdUdr*
Nursery	RdOvr	Tmk	Original	NCL	CNN*	OSS*	Smt+Tmk*	Smt*	CNN+Tmk*	Smt+ENN*	RdUdr*

Table 7: Performance ranking for original and balanced data sets for unpruned decision trees.

Data set	1°	2°	3°	4°	5°	6°	7°	8°	9°	10°	11°
Pima	RdOvr	Smt	Smt+Tmk	Smt+ENN	Tmk	NCL	Original	CNN+Tmk	RdUdr	CNN*	OSS*
German	Original	Tmk	RdOvr	NCL	RdUdr	Smt	Smt+Tmk	Smt+ENN	OSS	CNN	CNN+Tmk
Post-operative	Original	CNN+Tmk	RdOvr	Smt	Tmk	CNN	OSS	Smt+ENN	RdUdr	Smt+Tmk*	NCL*
Haberman	Smt+Tmk	Smt+ENN	Smt	RdOvr	Tmk	NCL	RdUdr	Original	OSS*	CNN*	CNN+Tmk*
Splice-ie	Original	Smt	Tmk	CNN	NCL	Smt+Tmk	RdOvr	OSS*	Smt+ENN*	CNN+Tmk*	RdUdr*
Splice-ei	RdOvr	Smt	Smt+Tmk	Smt+ENN	Original	CNN+Tmk	Tmk	NCL	OSS	CNN	RdUdr
Vehicle	RdOvr	Smt	Smt+Tmk	OSS	CNN	Original	CNN+Tmk	Tmk	Smt+ENN	NCL	RdUdr*
Letter-vowel	Smt+ENN	Smt	Smt+Tmk	Tmk*	RdOvr*	NCL*	Original*	CNN*	CNN+Tmk*	OSS*	RdUdr*
New-thyroid	Smt+ENN	Smt	Smt+Tmk	RdOvr	Original	Tmk	RdUdr	CNN	CNN+Tmk	NCL	OSS
E.Coli	Smt+Tmk	Smt	Smt+ENN	Tmk	RdOvr	Original	NCL	RdUdr	OSS	CNN+Tmk*	CNN*
Satimage	Smt+ENN	Smt	Smt+Tmk	RdOvr	Tmk	NCL	Original	CNN+Tmk*	RdUdr*	OSS*	CNN*
Flag	Smt+Tmk	OSS	RdOvr	CNN	Tmk	Smt+ENN	RdUdr	Original	NCL	CNN+Tmk	Smt
Glass	Smt+ENN	RdOvr	NCL	Smt	Smt+Tmk	Original	Tmk	RdUdr	OSS*	CNN+Tmk*	CNN*
Letter-a	Smt	Smt+Tmk	Smt+ENN	RdOvr	Tmk	OSS	NCL	Original	CNN	CNN+Tmk	RdUdr*
Nursery	RdOvr	Original	NCL	Tmk	CNN	OSS*	Smt*	CNN+Tmk*	Smt+Tmk*	Smt+ENN*	RdUdr*

Comparing the mean number of rules obtained with the over-sampled data sets, Random over-sampling and Smote + ENN are the methods that usually provide a smaller increase in the mean number of rules. It was expected that the application of over-sampling would result in an increase in the mean number of rules, since over-sampling increases the total number of training examples, which usually generates larger decision trees. It can be considered unexpected

Table 8: Number of rules (branches) for the original and over-sampled data sets and unpruned decision trees.

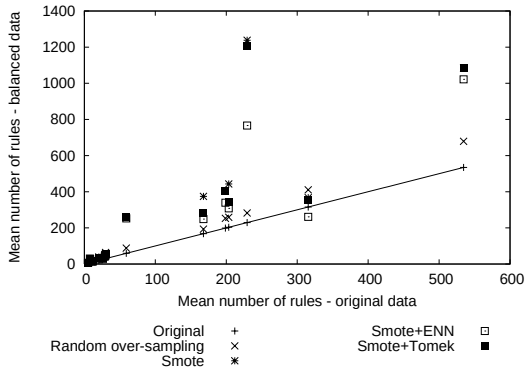
Data set	Original	Rand Over	Smote	Smote+Tomek	Smote+ENN
Pima	**29.90(6.06)**	63.80(13.15)	57.70(11.52)	54.20(12.91)	47.50(8.76)
German	315.50(21.41)	410.60(28.64)	367.30(20.85)	355.10(24.20)	**261.00(28.08)**
Post-operative	**20.40(3.86)**	36.80(3.05)	38.60(4.35)	32.70(5.87)	25.90(4.09)
Haberman	**7.80(3.79)**	25.20(10.94)	23.20(9.61)	25.00(7.70)	30.30(4.92)
Splice-ie	**203.50(7.78)**	258.70(13.07)	443.20(16.69)	340.60(21.34)	307.90(17.21)
Splice-ei	**167.80(9.40)**	193.30(7.41)	374.50(20.41)	283.90(14.90)	248.80(12.90)
Vehicle	**26.20(3.29)**	28.90(2.60)	34.90(3.38)	34.90(3.38)	29.20(2.82)
Letter-vowel	**534.50(11.92)**	678.80(19.07)	1084.50(19.61)	1083.20(20.12)	1022.00(26.34)
New-thyroid	5.40(0.84)	**5.10(0.32)**	6.90(1.29)	6.90(1.29)	6.90(0.99)
E-coli	**11.60(3.03)**	17.70(2.91)	16.70(3.20)	16.50(3.84)	12.70(3.23)
Satimage	**198.80(11.04)**	252.70(9.33)	404.60(12.97)	404.60(12.97)	339.40(13.80)
Flag	**28.60(6.52)**	46.30(7.72)	52.50(12.47)	46.50(13.36)	40.30(9.09)
Glass	**9.40(2.22)**	13.00(1.33)	17.70(1.77)	17.70(1.77)	15.50(1.58)
Letter-a	**59.10(3.45)**	88.00(5.56)	257.60(15.42)	257.60(15.42)	252.60(18.23)
Nursery	**229.40(4.65)**	282.50(5.34)	1238.30(28.91)	1204.70(27.94)	766.30(77.24)

Table 9: Mean number of conditions per rule for the original and over-sampled data sets and unpruned decision trees.

Data set	Original	Rand Over	Smote	Smote+Tomek	Smote+ENN
Pima	**6.21(0.61)**	7.92(0.64)	7.74(0.44)	7.59(0.54)	7.27(0.67)
German	**6.10(0.17)**	6.89(0.25)	10.27(0.51)	9.68(0.32)	7.35(0.58)
Post-operative	**3.61(0.41)**	4.86(0.26)	5.36(0.37)	4.75(0.52)	4.46(0.50)
Haberman	**3.45(1.36)**	5.71(1.43)	5.61(1.27)	5.81(1.02)	6.45(0.60)
Splice-ie	6.04(0.09)	6.15(0.04)	6.08(0.08)	6.00(0.09)	**5.58(0.11)**
Splice-ei	5.46(0.14)	5.70(0.08)	5.51(0.07)	5.41(0.09)	**4.91(0.09)**
Vehicle	7.21(0.70)	7.03(0.44)	7.09(0.50)	7.09(0.50)	**6.63(0.38)**
Letter-vowel	20.96(1.19)	19.32(0.82)	18.78(0.40)	18.78(0.40)	**18.32(0.43)**
New-thyroid	**2.76(0.39)**	2.85(0.17)	3.12(0.26)	3.12(0.26)	3.08(0.20)
E-coli	4.43(0.79)	5.48(0.41)	4.98(0.60)	4.92(0.65)	**4.15(0.49)**
Satimage	**12.13(0.46)**	15.93(0.42)	13.89(0.64)	13.89(0.64)	12.54(0.36)
Flag	**3.92(0.70)**	5.42(0.55)	9.43(1.04)	8.75(1.53)	6.71(1.23)
Glass	**4.20(0.61)**	5.80(0.51)	5.92(0.50)	5.92(0.50)	5.51(0.32)
Letter-a	**7.30(0.22)**	10.35(0.64)	10.97(0.38)	10.97(0.38)	10.86(0.36)
Nursery	6.51(0.01)	6.84(0.03)	6.87(0.03)	6.84(0.03)	**6.41(0.12)**

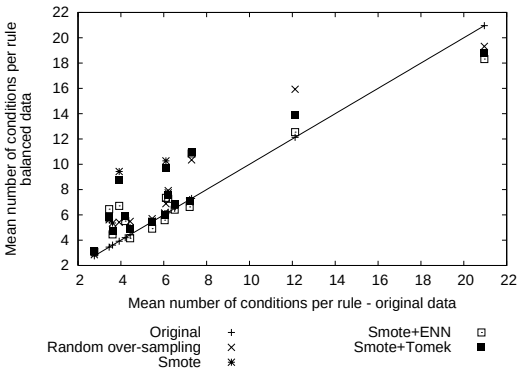
that Random over-sampling is competitive with Smote + Tomek and Smote + ENN in the number of induced rules, once Tomek and ENN were applied as data cleaning methods with the objective of eliminating noise examples and thus simplifying the induced decision trees.

Figure 6: Mean number of induced rules for original and balanced data sets and unpruned decision trees.



The results presented in Table 9 are shown in a graph in Figure 7 allowing a clearer comparison for the mean number of conditions per rule for the over-sampled data sets. The Smote + ENN method provided very good results. In fact, it was the best ranked in 10 data sets. Furthermore, this method was even able to obtain smaller values than those achieved by decision trees induced from the original data sets in 6 data sets. Moreover, considering only the over-sampled data sets, this method was the best ranked for another 4 data sets.

Figure 7: Mean number of conditions per rule for original and balanced data sets and unpruned decision trees.



6. CONCLUSION AND LIMITATIONS

In this work we analyze the behavior of several over and under-sampling methods to deal with the problem of learning from imbalanced data sets. Our results show that the over-sampling methods in general, and Smote + Tomek and Smote + ENN (two of the methods proposed in this work) in particular for data sets with few positive (minority) examples, provided very good results in practice. Moreover, Random over-sampling, frequently considered an unprosperous

method provided competitive results with the more complex methods. As a general recommendation, Smote + Tomek or Smote + ENN might be applied to data sets with a small number of positive instances, a condition that is likely to lead to classification performance problems for imbalanced data sets. For data sets with larger number of positive examples, the Random over-sampling method which is computationally less expensive than other methods would produce meaningful results.

It should be noted that allocating half of the training examples to the minority class does not always provide optimal results [23]. We plan to address this issue in future research. Furthermore, some under-sampling methods, such as Tomek links and NCL, that do not originally allow the user to specify the resulting class distribution, must be improved to include this feature. Another natural extension to this work is to analyze the ROC curves obtained from the classifiers. This might provide us with a more in depth understanding of the behavior of balancing and cleaning methods.

Acknowledgements. We wish to thank the anonymous reviewers and Dorival Leão Pinto Júnior for their helpful comments. This research was partially supported by the Brazilian Research Councils CAPES and FAPESP.

7. REFERENCES

[1] Batista, G. E. A. P. A., Bazan, A. L., and Monard, M. C. Balancing Training Data for Automated Annotation of Keywords: a Case Study. In *WOB* (2003), pp. 35–43.

[2] Bauer, E., and Kohavi, R. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning 36* (1999), 105–139.

[3] Blake, C., and Merz, C. UCI Repository of Machine Learning Databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[4] Chawla, N. V. C4.5 and Imbalanced Data Sets: Investigating the Effect of Sampling Method, Probabilistic Estimate, and Decision Tree Structure. In *Workshop on Learning from Imbalanced Data Sets II* (2003).

[5] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. SMOTE: Synthetic Minority Over-sampling Technique. *JAIR 16* (2002), 321–357.

[6] Ciaccia, P., Patella, M., and Zezula, P. M-tree: an Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB* (1997), pp. 426–435.

[7] Domingos, P. MetaCost: A General Method for Making Classifiers Cost-Sensitive. In *KDD* (1999), pp. 155–164.

[8] Drummond, C., and Holte, R. C. C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-sampling. In *Workshop on Learning from Imbalanced Data Sets II* (2003).

[9] Ferri, C., Flach, P., and Hernández-Orallo, J. Learning Decision Trees Using the Area Under the ROC Curve. In *ICML* (2002), pp. 139–146.

[10] Hand, D. J. *Construction and Assessment of Classification Rules*. John Wiley and Sons, 1997.

[11] Hart, P. E. The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory IT-14* (1968), 515–516.

[12] Japkowicz, N. Class Imbalances: Are We Focusing on the Right Issue? In *Workshop on Learning from Imbalanced Data Sets II* (2003).

[13] Japkowicz, N., and Stephen, S. The Class Imbalance Problem: A Systematic Study. *IDA Journal 6*, 5 (2002), 429–449.

[14] Kubat, M., and Matwin, S. Addressing the Course of Imbalanced Training Sets: One-sided Selection. In *ICML* (1997), pp. 179–186.

[15] Laurikkala, J. Improving Identification of Difficult Small Classes by Balancing Class Distribution. Tech. Rep. A-2001-2, University of Tampere, 2001.

[16] Ling, C. X., and Li, C. Data Mining for Direct Mining: Problems and Solutions. In *KDD* (1998), pp. 73–79.

[17] Mitchell, T. M. *Machine Learning*. McGraw-Hill, 1997.

[18] Prati, R. C., Batista, G. E. A. P. A., and Monard, M. C. Class Imbalances *versus* Class Overlapping: an Analysis of a Learning System Behavior. In *MICAI* (2004), pp. 312–321. LNAI 2972.

[19] Provost, F. J., and Fawcett, T. Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. In *KDD* (1997), pp. 43–48.

[20] Quinlan, J. R. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, CA, 1988.

[21] Stanfill, C., and Waltz, D. Instance-based Learning Algorithms. *Communications of the ACM 12* (1986), 1213–1228.

[22] Tomek, I. Two Modifications of CNN. *IEEE Transactions on Systems Man and Communications SMC-6* (1976), 769–772.

[23] Weiss, G. M., and Provost, F. Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction. *JAIR 19* (2003), 315–354.

[24] Wilson, D. L. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Communications 2*, 3 (1972), 408–421.

[25] Wilson, D. R., and Martinez, T. R. Reduction Techniques for Exemplar-Based Learning Algorithms. *Machine Learning 38*, 3 (2000), 257–286.

[26] Zadrozny, B., and Elkan, C. Learning and Making Decisions When Costs and Probabilities are Both Unknown. In *KDD* (2001), pp. 204–213.