Weighted Classification Using Decision Trees for Binary Classification Problems

José Luis Polo Fernando Berzal Juan Carlos Cubero

Department of Computer Science and Artificial Intelligence

ETSIIT, University of Granada, 18071 Granada, Spain

jlpolo@decsai.ugr.es fberzal@decsai.ugr.es jc.cubero@decsai.ugr.es

Abstract

Weighted classification is targeted at classification problems where obtaining a good classification model for particular classes is all-important. Weighted classification techniques provide simpler models for the important classes. Moreover, they do so without severely affecting the resulting classifiers accuracy.

In this paper, we propose weighted versions of decision-tree-based classification models in order to address the weighted classification problem. In particular, we have adapted the standard TDIDT model (Top-Down Induction of Decision Trees) as well as the ART model (Association Rules Trees).

Weighted classification is specially well-suited to binary classification problems when we are interested in obtaining a simple model for one of the problem classes, a relatively common situation in practice. The experiments we have performed on binary classification problems show promising results.

1 Introduction

Classification is a key problem addressed by Data Mining techniques. The aim of any classification algorithm is to build a classification model given some examples of the classes we are trying to model. The model we obtain can then be used to classify new examples or simply to achieve a better understanding of the available data.

In traditional classification algorithms, all the classes are assumed to have exactly the same importance and a classification model is built without taking into account the differences among classes that may exist. However, the specific context of a given classification problem might spur our interest in obtaining classification models focused, somehow, on particular classes that might exhibit singular features.

Some techniques have been proposed that do not treat all classes equally. They include unbalanced learning methods, cost-sensitive techniques, and subgroup discovery.

- Unbalanced learning methods [1, 4] focus on those classes with a very low support level . Traditional learning algorithms tend to ignore such low-support classes and build classification models that do not represent members from these classes.

- Cost-sensitive classification techniques [6, 5] take into account that the cost of a false positive is not the same for all the problem classes. When different costs are involved, traditional learning methods could lead to too many high-cost errors, making classification models useless in practice. Cost-sensitive techniques introduce biases so that certain kinds of errors are avoided when those errors incur in a high cost.

- Subgroup discovery [7] tries to find interesting subgroups within the training data set. These subgroups must exhibit differentiating features. In a classification

setting, subgroup discovery can be used to discover descriptions for a particular class. In this case, the resulting model is focused only on one of the problem classes.

Weighted classification [9] assigns different importance degrees to different classes. In this paper, we assign weights to the different problem classes in order to represent the relative importance of each class. w_i will represent the weight corresponding to the i-th class. In binary classification problems, w_0 will represent the first class importance, while w_1 will correspond to the second class importance. For convenience, we will assume that both weights will belong to the the [0,1] interval (if this were not the case, we could always normalize them).

Weighted classification is well-suited to many real-world binary classification problems, since we are often more interested in some particular classes than others. Hence the proper description of the important classes might be desirable. For example, the ADULT data set from the UCI Machine Learning Repository collects census data about individuals. Two classes are defined: one class correspond to people who earn more than 50000 US dollars (i.e. `income>50K$`) and the other lumps together those who do not reach that income level (i.e., `income<=50K$`). Tax collection agencies, however, will probably be more interested in the first group of people when assigning their limited resources to perform fiscal inspections.

A different context where weighted classification might be useful involves classes whose importance varies depending on user's needs, even for the same problem. An example of such a situation is represented by the MUSHROOM data set, also from the UCI Machine Learning Repository, whose classes respectively correspond to edible and poisonous mushrooms. A mycologist performing a field study on poisonous mushrooms would be interested in obtaining a simple model allowing the prompt identification of poisonous mushrooms. An excursionist, looking for mushrooms for personal consumption, would prefer a simple model focused on edible mushrooms.

Regardless the relative importance we assign to each class, classification errors might or might not be allowable. In the previous example, misclassifying a poisonous mushroom for an edible one might have nefarious consequences. Therefore, weighted classification and cost-sensitive classification [6], despite being related, focus on different aspects. Weighted classification focuses on building classification models that try to concisely represent the most important classes without a significative loss of the overall classifier accuracy, let it be cost-sensitive or not. Unlike subgroup discovery techniques, weighted classification leads to complete classification models, i.e. models involving all the problem classes.

The rest of our paper is organized as follows. Section 2 shows how to incorporate the weighted classification bias in the typical top-down induction of decision trees. Section 3 briefly describes the ART classification model and introduces the modifications needed to address weighted classification in ART. Section 4 describes the experiments we have performed with decision trees for weighted classification problems. Finally, our conclusions appear in Section 5.

2 The Weighted TDIDT Model

Decision trees [10, 12] are one of the most widely used classification models due to their interpretability and also due to the availability of efficient and scalable learning algorithms, which makes them suitable for a wide range of situations.

As their name implies, TDIDT algorithms build decision trees top-down, starting from the root of the tree. At each node, the most promising attribute (or combination of attributes) is chosen to branch the tree. This selection is performed according to an heuristic splitting criterion and this splitting process is recursively repeated until a stopping criterion is met (e.g. when all the examples in a node correspond to the same class). The

resulting node is then labeled with the most common class within the examples belonging to that node. Once the tree is completely built, or even during its construction, the decision tree is pruned to avoid overfitting.

We have adapted the standard TDIDT algorithm to weighted classification by modifying both the heuristic splitting criterion and the tree pruning strategy.

2.1 Splitting Criteria

Many different splitting criteria have been proposed in the literature and they all tend to provide similar results [2].

Traditional algorithms, such as ID3 [10], employ an heuristic based on Information Theory: the information gain resulting from a given partition of the training set. The attribute that maximizes the information gain is chosen to branch the tree. This information gain can be expressed as

$$G(A) = H(C) - H(C|A)$$

where A is the attribute to be evaluated, $H(C)$ is the class entropy, and $H(C|A)$ is the class entropy given attribute A. Class entropy can be computed as follows:

$$H(C) = -\sum_{k=1}^{K} p(c_k) \cdot log_2(p(c_k))$$

$$H(C|A) = \sum_{j=1}^{J} p(a_j) \cdot H(C|a_j)$$

where K and J represent the number of classes and the number of different valued for the A attribute, respectively. Finally, $H(C|a_j)$ represents the class entropy for the a_j value of the A attribute:

$$H(C|a_j) = -\sum_{k=1}^{K} p(c_k|a_j) \cdot log_2(p(c_k|a_j))$$

The above entropy-based splitting criterion measures the resulting partition node impurity, i.e. how good an attribute is in separating the examples belonging to different classes, but it does not take into account the relative importance of each class. In a weighted classification context, it would be desirable to bias the decision tree growing process towards nodes with a better representation of examples from the most important classes.

Class entropy gives the same importance to every class by averaging the contribution of each class. We could also compute a weighted average so that the contribution of important classes biases the final attribute selection. The resulting criterion, **Weighted Premium Gain (*WPGain*)**, considers the relative importance of each class and it can be formalized as follows for a K-class classification problem:

$$W(C|a_j) = -\sum_{k=1}^{K} f(w_k)p(c_k|a_j)log_2(p(c_k|a_j))$$

where $f(w_k)$ is the function used to aggregate the contributions from each class according to its relative importance, which its uniquely determined by its weight.

We could have directly resorted to the class weights, i.e. $f(w_k) = w_k$. However, this would lead to classifiers that would completely ignore the least important classes as we increase the relative importance of some classes with respect to the others. Hence, we need a softened aggregation function $f(w_k)$ for the computation of $W(C|a_j)$.

We use the *weighted premium* (WP) heuristic that has proved to be effective for different weight distributions in a recent study [9]. For a given class weight w_k, its weighted premium is

$$WP(w_k) = 1 + \frac{w_k - min}{max}$$

where min corresponds to the weight of the least important class and max represents the most important class weight. Therefore, the weighted premium is 1 for the least important class and greater than one for more important classes. This heuristic favors the most important classes without ignoring the least important ones.

The $f(w_k)$ aggregation function can then be computed as the normalized weighted premium for the w_k weights:

$$f(w_k) = \frac{WP(w_k)}{\sum_{i=1}^{K} WP(w_i)}$$

Figure 1 depicts the resulting weighted splitting criterion for binary classification problems with respect to the most important class probability. The figure shows the splitting criterion when both classes have the same importance (`W=1`), which corresponds to the traditional splitting criterion without weights. It also shows the bias we introduce when one class is twice as important than the other (`W=2`) and when it is nine times more important (`W=9`).

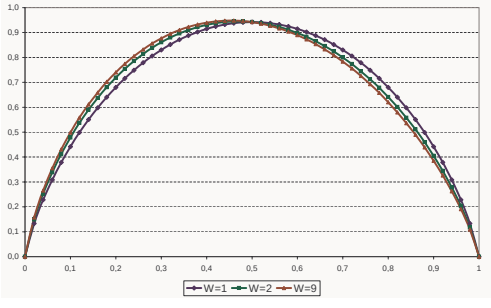

Figure 1: Weighted gain splitting criterion.

2.2 Tree Pruning

Tree pruning is necessary in TDIDT algorithms to avoid overfitting. Quinlan's pessimistic pruning [11], for instance, is one of the available pruning strategies. This pruning technique performs a postorder traversal of the tree internal nodes in order to decide, for each subtree, if it should be replaced for a single leaf node, which would be labeled with the most common class in that subtree.

Quinlan's pruning algorithm is called pessimistic because it pessimistically estimates the subtree error rate, as well as the error rate that would result from replacing the whole subtree with a single leaf node.

If N is the number of examples covered by a node and E is the number of errors we make when we label the node with the majority class, we could assume that the training examples are distributed following the binomial distribution $B(N, E/N)$. Then, we can estimate the error rate as the upper limit of the confidence interval for this distribution. The subtree rooted at a given node is substituted by a single leaf node when the estimated error rate for the single leaf is lower than the estimated error for the whole subtree.

As we did above with the splitting criterion, we could also modify the standard pruning criterion so that class weights are taken into account.

A subtree that is not pruned by the standard criterion should be pruned if the errors that pruning introduces correspond to examples belonging to the less important classes. Standard pruning would only see the overall number of errors and deem it inappropriate to prune the tree, even though it might be desirable for weighted classification. Since reducing classifier complexity for the important classes is the main driver behind weighted classification, a more aggressive pruning strategy would then lead to smaller classification models. As a consequence, a higher number of false positives for the important classes might appear. In other words, with simpler models, recall might improve at the cost of a potentially lower precision for the most important classes (a typical trade-off in information retrieval systems, now seen from a completely different perspective).

In order to perform this decision tree pruning, we can define a weighted error rate taking into account class weights:

$$error_w = \frac{\sum_{k=1}^{K} w_k \cdot e_k}{\sum_{k=1}^{K} w_k \cdot n_k}$$

where e_k is the number of misclassified training examples from the k-th class and n_k is the support of the k-th class in the training set.

This modified error rate is used to prune a subtree if the pessimistic estimation of the

weighted error for the subtree is higher than the weighted error for a single leaf node.

3 The Weighted ART Model

ART [3], which stands for *Association Rule Tree*, is an associative classification model that employs efficient association rule mining techniques to build a classifier. The resulting classifier can be seen either as a decision tree or as a decision list. As a decision list learner, ART is a generalized "Separate and Conquer" algorithm, in contrast to the standard "Divide and Conquer" approach of TDIDT algorithms. However, ART is faster than general decision list and rule inducers which need to discover rules one at a time. Moreover, ART classifiers tend to be smaller than the decision trees generated by standard TDIDT algorithms.

3.1 ART Classifier Construction

Instead of discovering rules one at a time, as most decision list learners do, ART discovers multiple rules simultaneously. ART builds partial classification models using sets of association rules. The instances in the input dataset which are not covered by the selected association rules are then grouped together in 'else' branches to be further processed following the same algorithm.

The special kind of decision list ART obtains can be considered as a degenerate, polythetic decision tree. Unlike most traditional TDIDT algorithms, ART can employ several attributes at once to branch the decision tree. This ability improves the performance of decision tree learning in terms of both higher prediction accuracy and lower theory complexity.

ART begins by using simple hypotheses to classify the training data and makes more complex hypothesis only when no simple hypotheses are found to work well. ART thus incorporates an explicit preference for simpler models, the bias behind Occam's Razor.

ART looks for sets of rules that might be suitable for classification, i.e. those whose confidence is high enough to be accurate for classification. Among the discovered rules, ART groups them according to the attributes that appear in their antecedents, as shown in Figure 2.

$$\{A_3\&B_2 \rightarrow C_1,\ B_4\&C_2 \rightarrow C_2,\ A_4\&B_1 \rightarrow C_2,\ B_0\&C_1 \rightarrow C_1,\ A_2\&C_2 \rightarrow C_1\}$$
$$\Downarrow$$
$$\{A_3\&B_2 \rightarrow C_1,\ A_4\&B_1 \rightarrow C_2\}$$
$$\{B_4\&C_2 \rightarrow C_2,\ B_0\&C_1 \rightarrow C_1\}$$
$$\{A_2\&C_2 \rightarrow C_1\}$$

Figure 2: Grouping compatible antecedents in ART.

Finally, the best group of rules is chosen by taking into account their coverage (i.e. their support within the training dataset) and this group is used to branch decision tree. All the examples that do not satisfy any of the selected rules are lumped together into an 'else' branch and the whole process is iteratively repeated until too few examples are left to be classified or we reach a pure node (i.e. a node with all examples belonging to the same class). In any case, a final leaf node is added and it is labeled with the most common class among the remaining examples.

ART automatically sets a suitable confidence threshold for selecting rules at each iteration. Once the most accurate association rule has been identified (i.e., the one with the best confidence value $MaxConf$), only similarly accurate rules are used to build the classifier. In particular, only those rules with confidence above $MaxConf - \Delta$ are considered. The Δ parameter establishes a "tolerance" interval for the confidence of the rules. We select the best possible rule and only allow slightly worse rules to be included in each set of good rules. This rather restrictive approach ensures that no bad rules will be ever considered to build the classifier.

3.2 ART for Weighted Classification

Weighted classification targets at obtaining simpler models for important classes. This requires bringing rules closer to the root in

ART classification models when those rules correspond to the most important classes.

Therefore, the rule selection criterion in ART should not be equally demanding for all problem classes. If we allow for slightly worse rules for important classes during classifier construction, this will lead to simpler models for important classes (even though potentially less accurate).

We can this adjust the Δ tolerance margin according to our needs, as specified by each class relative importance. A weighted premium to the tolerance margin is then suitable for adjusting confidence thresholds so that less accurate rules are accepted for the more important classes.

We can define a weighted tolerance margin Δ_{WP} as

$$\Delta_{WP}(w) = \Delta \cdot WP(w) = \Delta \cdot (1 + \frac{w - min}{max})$$

where Δ is ART tolerance parameter, min is the minimum class weight, and max is the maximum class weight.

Using the above heuristic, tolerance is maintained for the least important classes while it is increased for the more important ones.

This single modification to the original ART algorithm lets us build ART classification models that take weights into account in the desired way (the rest of the algorithm is left untouched).

4 Experimentation

This section describes the approach we have followed to evaluate weighted classification models and the results we have obtained with actual data sets.

4.1 Weighted Classifier Evaluation

We have used three metrics to evaluate weighted classification models: their relative complexity, their overall accuracy, and their false positive rate.

4.1.1 Classifier Complexity

First, we will describe the complexity measure we call *weighted length*. Basically, it is just the weighted version of the average tree depth typically used to measure decision tree complexity.

Without taking into account class weights, the average tree depth can be computed as

$$AvgDepth = \frac{\sum_{k=1}^{K} \sum_{l=1}^{L} n_{kl} \cdot l}{\sum_{k=1}^{K} n_k}$$

where K is the number of classes, L is the number of tree levels (i.e. its depth), and n_{kl} is the number of examples belonging to the k-th that reach the l-th tree level.

Our weighted complexity just performs a weighted average by taking class weights into account, so that each class importance determines that class contribution to the overall classifier complexity:

$$wLen = \frac{\sum_{k=1}^{K} w_k \sum_{l=1}^{L} n_{kl} \cdot l}{\sum_{k=1}^{K} w_k n_k}$$

This way, a decision tree with important class leaves nearer to the root will have a lower weighted complexity. Please note that, when all weights are the same, our weighted classifier complexity measure equals the average tree depth.

4.1.2 Classifier Accuracy

Even though our goal is achieving simpler models for important classes, we must also take into account the resulting classifier accuracy. The simplest classification model, from a weighted classification perspective, would be useless if it misclassified too many examples. Hence we also compute the standard classifier accuracy in our experiments:

$$Accuracy = \frac{\sum_{k=1}^{K} TP_k}{N}$$

where TP_k is the number of true positives belonging to the k-th class and N is the total number of examples.

4.1.3 F Measure: False Positives and False Negatives

Finally, we must also check that the complexity reduction we achieve does not come at the cost of an inordinate number of false positives. We use an F measure borrowed from information retrieval that computes the harmonic mean of the resulting classifiers precision and recall. In particular, we resort to the macro-averaging F measure [13] that is defined as follows

$$MacroF = \frac{2 \cdot Pr^M \cdot Re^M}{Pr^M + Re^M}$$

where Pr^M and Re^M represent the macro-averaged precision and recall measures for our classifiers.

A macro-averaged measure is just the average of the individual measurements performed for each one of the problem classes, i.e.

$$Pr^M = \frac{\sum_{k=1}^{K} Pr_k}{K}$$

$$Re^M = \frac{\sum_{k=1}^{K} Re_k}{K}$$

where Pr_k and Re_k stand for the classifier precision and recall for the k-th class, respectively.

4.2 Experimental Results

We have tested both Weighted TDIDT and Weighted ART classification models against some well-known binary classification problems from the UCI Machine Learning Repository [8].

For each one of the classification problems, we have perform two experiment suites: one considering the first class as the important one, another choosing the second class as the important one. Each suite, itself, includes experiments with seven different relative weights: when both classes are equally important (same weights) and when the important class if 50%, 100%, 150%, 200%, 400%, and 800% more important than the less important class. In other words, we have checked what happens when the higher weight is 1, 1.5, 2, 2.5, 3, 5, and 9 times the lower weight, where 1 corresponds to the non-weighted case, 1.5 corresponds to a 50% premium, and so on. Since each particular weight assigned is evaluated by a 10-fold cross validation, that leads to 140 experiments for each algorithm tested on a particular dataset (3080 classifiers overall).

Data set	Examples	Attributes
ADULT	48842	15
AUSTRALIAN	690	15
BREAST	699	9
CHESS	3196	36
HEART	270	14
MUSHROOM	8124	23
PIMA	768	9
TICTACTOE	958	10
TITANIC	2201	4
VOTES	435	17

Table 1: Data sets used to evaluate weighted classification models.

Figures 3 and 4 summarize the results we have obtained with Weighted TDIDT and Weighted ART models with respect to their non-weighted counterparts.

The tables in these figures show the average relative differences (in %) between traditional and weighted classifiers for the three metrics we introduced in the previous section: `wLen` (*weighted length*), `acc` (*classifier accuracy*), and `macroF` (*macro-averaged F measure*).

They also detail the results we have obtained with respect to our target measure `wLen`: how many times weighted complexity is increased by the weighted versions of our classifiers (`[g]reater`), how many times it is reduced (`[l]ower`), and how many times the introduction of weights does not affect the resulting classifier complexity (`[e]quivalent`). Results are considered to be equivalent if their relative difference is below 1%.

Each table row shows the results for a particular weight assignment, where the number represents the ratio between the higher and lower weights. This lets us visually explore the evolution of the evaluation

TDIDT	% variation			wLen		
w	wLen	acc	macroF	g	e	l
1	0	0	0	0	20	0
1.5	-6.11	-1.00	-1.15	5	3	12
2	-6.79	-2.15	-2.23	6	3	11
2.5	-7.28	-2.74	-2.82	5	4	11
3	-9.36	-3.74	-5.00	6	4	10
5	-11.10	-4.76	-5.99	5	2	13
9	-19.78	-5.44	-6.55	5	1	14

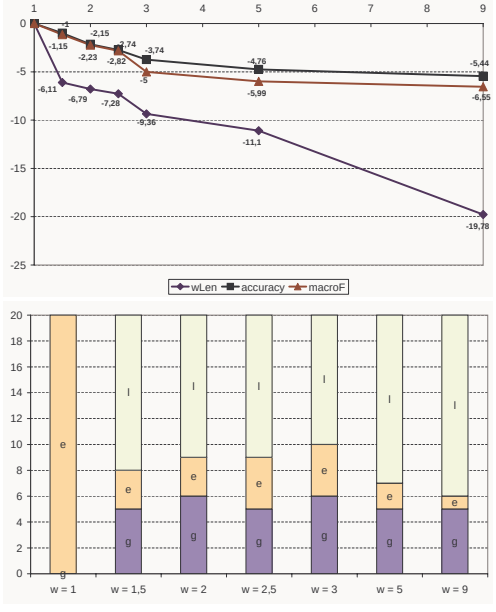

Figure 3: Weighted TDIDT results.

ART	% variation			wLen		
w	wLen	acc	macroF	g	e	l
1	0	0	0	0	20	0
1.5	-7.14	-0.47	-0.59	1	6	13
2	-11.40	-0.46	-0.55	3	5	12
2.5	-13.13	-0.46	-0.54	1	6	13
3	-16.20	-0.50	-0.60	1	6	13
5	-19.97	-0.36	-0.40	1	6	13
9	-23.00	-0.25	-0.28	1	6	13

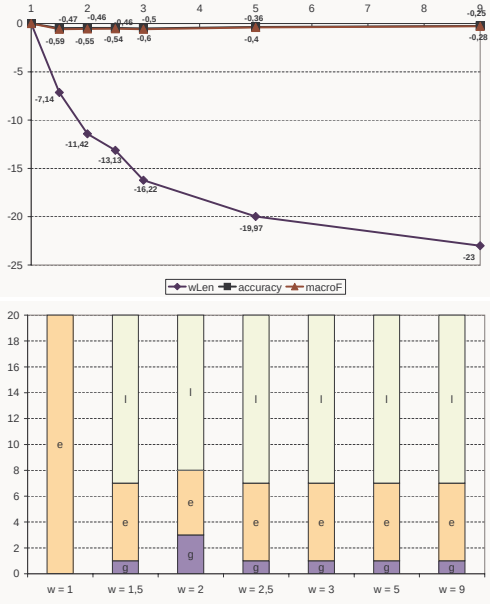

Figure 4: Weighted ART results.

measures as we increase the difference between weights (i.e. as we increase the relative importance of the important class with respect to the less important class). The graphs below the tables illustrate this evolution. Please note that the X-axis would correspond to the results we would achieve by using a traditional (non-weighted) classifier, which also corresponds to the results we obtain with equal class weights ($w = 1$).

Weighted TDIDT classification models, as shown in Figure 3, help us reduce classifier complexity (more than 6% on average when one class is just 50% more important than the other) even though classifier accuracy and precision also suffer. In other words, complexity and accuracy/precision are traded off, as expected, when weights are introduced into the TDIDT model. It should be noted, however, that complexity reductions always outweigh accuracy and precision losses. Moreover, complexity reductions are consistently achieved in a wide majority of classification problems for every weight assignment.

Figure 4 summarizes the same experiments when performed with Weighted ART classification models. Here, we also consistently achieve a substantial classification model complexity reduction, surprisingly without significative accuracy and precision losses. ART rule selection approach lets us focus on important classes while, at the same time, we still achieve excellent

accuracy results. Weights adequately guide classifier construction but do not degrade ART performance.

When we compare the accuracy results corresponding to traditional classifiers (`w=1`) with those we can achieve with weighted classifiers (`w=9`) we observe that, while TDIDT classification models reduce their average accuracy from 87.91% to 83.13% when weights are introduced, ART is able to keep its average accuracy level: just a slight decrease from 86.85% to 86.63% when weights are taken into account.

5 Conclusions

The goal of weighted classification is obtaining simple, yet accurate, models for the most important classes in a classification problem. In this paper, we have described how to build TDIDT and ART classifiers that take class importance into account during decision tree construction.

In particular, we have demonstrated how TDIDT algorithms can be adapted to weighted classification by means of weighted splitting criteria and pruning strategies, while ART can be used for weighted classification with a simple modification that tailors rule selection tolerance to class importance.

The results we have obtained show a reasonable trade-off between classification complexity reduction and accuracy/precision loss is possible using both Weighted TDIDT and Weighted ART classification models. Results are specially promising in the case of ART, where accuracy and precision are preserved at the same time that weighted classifier complexity is significantly reduced.

References

[1] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6(1):20–29, 2004.

[2] F. Berzal, J. C. Cubero, F. Cuenca, and M. J. Martín-Bautista. On the quest for easy-to-understand splitting rules. *Data and Knowledge Engineering*, 44(1):31–48, 2003.

[3] F. Berzal, J. C. Cubero, D. Sánchez, and J.-M. Serrano. Art: A hybrid classification model. *Machine Learning*, 54(1):67–92, 2004.

[4] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations*, 6(1):1–6, 2004.

[5] P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[6] C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[7] D. Gamberger and N. Lavrac. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.

[8] C. B. D. Newman and C. Merz. UCI repository of machine learning databases, 1998.

[9] J. L. Polo, F. Berzal, and J. C. Cubero. Taking class importance into account. *Lecture Notes in Computer Science*, 4413, 2007.

[10] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[11] J. R. Quinlan. Simplifying decision trees. Technical report, Cambridge, MA, USA, 1986.

[12] J. R. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, January 1993.

[13] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.