Cost-Sensitive Learning and Decision Making for Massachusetts PIP Claim Fraud Data

Stijn Viaene¹, Richard Derrig², and Guido Dedene¹

¹ KBC Insurance Research Chair
Leuven Institute for Research on Information Systems
Department of Applied Economic Sciences
Katholieke Universiteit Leuven
Naamsestraat 69, B-3000 Leuven, Belgium
{stijn.viaene;guido.dedene}@econ.kuleuven.ac.be

² Automobile Insurers Bureau of Massachusetts &
Insurance Fraud Bureau of Massachusetts
101 Arch Street, Boston, Massachusetts
richard@aib.org

Abstract. In many real-life decision making situations the default assumption of equal (mis-)classification costs underlying pattern recognition techniques is most likely violated. Consider the case of insurance claim fraud detection for which an early claim screening facility is to be built to decide upon the nature of an incoming claim as either suspicious or not. This decision typically forms the basis for routing the claim through different claims handling workflows. Claims that pass the initial (automated) screening phase are settled swiftly and routinely, involving a minimum of transaction processing costs. Claims that are flagged as suspicious need to pass a costly state verification process, involving (human) resource-intensive investigation. Here, cost-sensitive learning and decision making bring help for making cost-benefit-wise optimal decisions. In this paper we investigate the issue of cost-sensitive classification for a data set of Massachusetts closed personal injury protection (PIP) insurance claims that were previously investigated for suspicion of fraud by domain experts and for which cost information has been obtained. After a theoretical exposition on cost-sensitive learning and decision making methods, we then apply these methods to the claims data at hand to contrast the predictive performance of the documented methods for a variety of decision tree and rule learners. Standard logistic regression and (smoothed) naive Bayes are used as benchmarks.

Keywords: Personal Injury Protection (PIP) Insurance, Claim Fraud, Decision Tree, Rule Learner, Cost-Sensitive Classification.
1 Introduction

Insurance companies and state/country-level fraud bureaus have invested considerably in more systematic electronic collection, organization, analysis, maintenance of, and access to insurance data to promote the coherent use and re-use of the embedded business knowledge. Fraud prevention and detection are definitely among those business functions that are able to benefit from the huge investments in a.o. data consolidation, network infrastructure and communication technology. Insurance fraud and abuse are widespread. Many believe that insurance fraud has evolved into one of the most prevalent and costly white collar crimes of which a sizeable amount of fraudulent activity remains undetected [1,2,3]. The emergent technical and organizational platforms of data sharing now lay the foundations for the development and application of business intelligence technologies to upgrade the measurement, detection, investigation and deterrence of insurance fraud.

Data-driven analysis and modelling allow to modernize the fraud detection process with sophisticated, (semi-)automated, intelligent tools such as unsupervised and supervised pattern recognition techniques. Classification is among the most widely-used supervised pattern recognition techniques. This is no different for fraud detection (see e.g. [4,5,6,7]). One of the complications that arise when applying these learning programs in practise, however, is to make them reduce the cost of (mis-)classification rather than the error rate. This is not unimportant, since in many real-life decision making situations the assumption of equal (mis-)classification costs, the default operating mode for many pattern learners, is most likely violated. Medical diagnosis is a prototypical example. Here a false negative prediction, i.e. failing to detect a disease, may well have fatal consequences, whereas a false positive prediction, i.e. diagnosing a disease for a patient that does not actually have it, may be less serious. A similar situation arises for insurance claim fraud detection, where an early claims screening facility is to help decide upon the routing of incoming claims through alternative claims handling workflows. Claims that pass the initial (automated) screening phase are settled swiftly and routinely, involving a minimum of transaction processing costs. Claims that are flagged as suspicious — usually only a minority of claims — need to pass a costly state verification process, involving (human) resource-intensive investigation. Early claims screening should thus be designed to take into account these cost asymmetries in order to make cost-benefit-wise optimal routing decisions.

Many practical situations are not unlike the ones above. They are typically characterized by a setting in which one of the predefined classes is a priori relatively rare, but also associated with relatively high cost if not detected. Automated pattern learning that is insensitive to this context is unlikely to be successful. For that reason Provost et al. [8,9] argue against error rate (a.k.a. zero-one loss) as a classifier performance criterion, as it assumes equal (mis-)classification costs and relatively balanced class distributions. Given a naturally very skewed class distribution and costly faulty predictions for the rare class, a model optimized on error rate alone may very well end up building a useless model, i.e.
one that always predicts the most frequent class. In these circumstances cost-sensitive learning and decision making bring help. For an up-to-date, on-line bibliography on cost-sensitive learning and decision making research, see Turney [10].

In this paper we investigate several methods for making classification algorithms sensitive to classification cost asymmetries. Rather than focus on work gone into making individual algorithms cost-sensitive (see e.g. [11,12]), we focus on methods that aim to make a broad variety of error-based learners, i.e. learners designed to minimize error rate and not necessarily generating models that produce (good) posterior class probability estimates for the application of direct minimum expected cost classification (see Section 2.1), cost-sensitive. The methods are empirically evaluated (by means of a repeated split-sample experiment) for a variety of error-based decision tree and rule learners, which remain extremely popular in practise due to their representational attractiveness (combining modelling flexibility and comprehensibility), using a data set of Massachusetts closed personal injury protection (PIP) automobile insurance claims that were previously investigated for suspicion of fraud by domain experts and for which cost information was obtained.

The decision tree and rule learners taken up in this study are all available in Weka 3-3-1 [13], open source machine learning software written in Java issued under the GNU General Public License, with the exception of C4.5rules, which is available as part of the freely available C4.5 Release 8 distribution [14]. Standard logistic regression and naive Bayes are used as benchmarks in light of their predictive performance reported in Viaene et al. [7], the widespread availability and use of these models in practise, and because previous research often uses them as a reference.

The paper is organized as follows. Section 2 contains a theoretical exposition on cost-sensitive learning and decision making methods, covering direct minimum expected cost classification (Section 2.1), MetaCost (Section 2.2), over- and undersampling (Section 2.3), and cost-sensitive boosting (Section 2.4). Section 3 covers the PIP automobile insurance claims data set characteristics and the corresponding cost information. The empirical evaluation of the methods presented in Section 2 using the data described in Section 3 is given in Section 4. Section 5 concludes the paper with a summary.

2 Cost-Sensitive Learning and Decision Making

In this section we give a brief overview of several more or less generic ways of making error-based classification algorithms cost-sensitive. In order of appearance we discuss: direct minimum expected cost classification (Section 2.1), MetaCost (Section 2.2), over- and undersampling (Section 2.3), and cost-sensitive boosting (Section 2.4).
Table 1. Cost matrix $C(\cdot, x)$ for binary classification. For a data instance with input vector $x \in \mathbb{R}^n$, the cost of a true positive (TP) is $C(+, +, x)$, the cost of a true negative (TN) is $C(-, -, x)$, the cost of a false positive (FP) is $C(+, -, x)$, and the cost of a false negative (FN) is $C(-, +, x)$.

<table>
<thead>
<tr>
<th>Predicted Target</th>
<th>Actual Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td></td>
<td>$C(-, -, x)$</td>
</tr>
<tr>
<td>$+$</td>
<td>$C(+, +, x)$</td>
</tr>
</tbody>
</table>

2.1 Direct Minimum Expected Cost Classification

Optimal Bayes classification dictates that a data instance with input vector $x \in \mathbb{R}^n$ should be assigned to the class $t \in \{1, \ldots, T\}$ associated with the minimum expected cost, i.e. the optimal Bayes classification $H(x) : \mathbb{R}^n \to \{1, \ldots, T\}$ then works as follows:

$$H(x) = \arg \min_{t \in \{1, \ldots, T\}} \sum_{j=1}^{T} p(j|x) C(t, j, x)$$

where $p(j|x)$ is the conditional probability of class $j$ given input vector $x$ and $C(t, j, x)$ is the cost of classifying a data instance with input vector $x$ and actual class $j$ as class $t$. Typically, the cost information $C$ is represented in the form of a cost matrix, where each row represents a single predicted class and each column an actual class. This is illustrated in Table 1 for the case of two classes. $C(\cdot, \cdot, x) > 0$ represents a cost. $C(\cdot, \cdot, x) < 0$ represents a benefit.

We assume a cost matrix $C$ that is: a) given (or estimated; see e.g. [15]) on beforehand; b) assumed to be independent of $x$; and c) stationary, i.e. it does not change during learning. For the two-class case this implies a fixed cost $C(+, +)$ for a true positive, a fixed cost $C(-, -)$ for a true negative, a fixed cost $C(+, -)$ for a false positive, and a fixed cost $C(-, +)$ for a false negative prediction. In addition, we assume that the cost matrix complies with the two reasonableness conditions formulated by Elkan in [16]. The first reasonableness condition implies that neither row dominates any other row, i.e. there are no two rows $m$ and $n$ ($m \neq n$) for which $C(m, j) \geq C(n, j)$ for $j = \{1, \ldots, T\}$. The second reasonableness condition implies that the cost of labelling a data instance incorrectly is always greater than the cost of labelling it correctly, i.e. $\min_{i \in \{1, \ldots, T\}} C(i, j) = C(j, j)$ for $j = \{1, \ldots, T\}$. Without loss of generality, we can thus limit our exposition to the case of cost matrices having only zeros on the diagonal and only non-negative off-diagonal elements.

Given a cost matrix $C$ that complies with the above reasonableness conditions, scaling all the entries in $C$ with a strictly positive constant factor, which corresponds to changing the measurement units for the costs, does not alter
the optimal decisions. Moreover, any such cost matrix $C$ can be transformed into an equivalent cost matrix $C'$ with zero diagonal elements and positive values for the off-diagonal elements by incrementing every column $j$ entry with $-C(j,j)$, again leaving the optimal classification decision unaffected [17]. This operation corresponds to what Elkan [16] calls "shifting the baseline"\(^1\) for the separate columns of the cost matrix.\(^2\) For a two-class cost matrix this is done by specifying $C'(+,+) = 0$, $C'(-,-) = 0$, $C'(+, -) = C(+, -) - C(-, -)$ and $C'(-, +) = C(-, +) - C(+, +)$.

In the minimum expected cost classification scenario the estimation of $p(t|x)$ is the crucial learning part. An advantage of this scenario is that, as costs are only introduced in the post-learning phase, it does not require the models to be retrained every time the costs change. Any error-based classifier that can produce an estimate of $p(t|x)$ can make direct use of Eq.(1) to calculate the optimal classification. However, the estimates that are produced by many error-based classifiers are neither unbiased, nor well-calibrated [18]. Decision tree learners such as C4.5 and rule learners such as C4.5rules [14] are well-known examples. They focus primarily on discrimination between the classes, and only produce posterior class membership probability estimates as a byproduct to the classification process. Several methods have been proposed for obtaining better probability estimates from decision trees and rules. Well-known are smoothing methods such as Laplace correction and $m$-estimation smoothing [19,20].

2.2 MetaCost

MetaCost is aimed at making an arbitrary error-based classification algorithm cost-sensitive by manipulation of the training data instance class labels. The underlying classification algorithm can be treated as a black box, requiring no knowledge of its internals or change to it. The original implementation of MetaCost due to Domingos [21] is based on relabelling each training data instance with its optimal class label according to minimum expected cost classification, and then learning the final model using the relabelled training data. Domingos estimates $p(t|x)$, needed for the application of minimum expected cost classification, using a variant of Breiman’s bootstrap resample aggregation (a.k.a. bagging) [22]. MetaCost then works as follows for training data $D = \{x_i, t_i\}_{i=1}^N$ with input vectors $x_i \in \mathbb{R}^n$ and class labels $t_i \in \{1, ..., T\}$, error-based learner $EBL$, and cost matrix $C$:

1. Construct internal cost-sensitive classifier $H^*(x) : \mathbb{R}^n \rightarrow \{1, ..., T\}$ on $D$ as follows for $k$ going from 1 to $K$ (predefined parameter):
   (a) Generate bootstrap resamples $D^{(k)}$ from $D$ having $m$ (predefined parameter) data instances.

---

1 The baseline stands for the starting situation against which costs and/or benefits of the alternative predictions are evaluated.

2 Note that this actually implies that the baseline need only be fixed per column when reasoning on the incurred costs following a prediction in order to populate the cost matrix entries.
(b) Learn models $H^{(k)}$ by applying $EBL$ to $D^{(k)}$. Let $H^{(k)}(x) \in \{1, ..., T\}$ denote the model’s class prediction, and $H^{(k)}_t(x) \in [0, 1]$ denote the model’s confidence in class prediction $t$ given input vector $x$, i.e. its estimate of $p(t|x)$, only applicable if the model generated by $EBL$ is able to produce the latter.

(c) Combine models $H^{(k)}$ into internal cost-sensitive classifier $H^*(x)$ using minimum expected cost classification as follows:

$$H^*(x) = \arg \min_{t \in \{1, ..., T\}} \sum_{j=1}^{K} h_{j}^{(k)}(x) C(t, j)$$

where $h_{j}^{(k)}(x) = H_{j}^{(k)}(x)$, if applicable, otherwise $h_{j}^{(k)}(x) = 1$ for $j = H^{(k)}(x)$ and $h_{j}^{(k)}(x) = 0$ for $j \neq H^{(k)}(x)$.

2. Relabel the data instances in $D$ using $H^*(x)$ to obtain $D^*$:

$$D^* = \{x_i, H^*(x_i)\}_{i=1}^{N}$$

where, optionally, the summation over $k$ in Eq.(2) may be chosen to range only over those models $H^{(k)}$ for which $(x_i, t_i) \notin D^{(k)}$.

3. Construct final cost-sensitive classifier $H(x) : \mathbb{R}^n \rightarrow \{1, ..., T\}$ by applying $EBL$ to $D^*$.

The MetaCost rationale of constructing a cost-sensitive classifier $H(x)$ by relabelling the training data $D$ according to an internal cost-sensitive classifier $H^*(x)$ (step 2) and reapplying $EBL$ to the relabelled training data $D^*$ (step 3) may be applied more generally [23]. We may in fact use any cost-sensitive classification method $CSC(D, EBL, C)$ in step 1 to yield $H^*(x)$. For instance, instead of using bagging as does the original MetaCost method to estimate $p(t|x)$ in step 1 for applying minimum expected cost classification, we may directly use the probability estimates produced by the model generated by $EBL$, if these are available. This implementation was used by Zadrozny and Elkan in [15]. We may also opt for a cost-sensitive version of $EBL$ that directly takes the cost matrix $C$ into consideration during optimization, producing classification without explicit use of minimum expected cost classification. Alternatively, we may use any of the cost-sensitive classification methods described in the following subsections.

An advantage of the MetaCost rationale of relabelling the training data is that it produces a single final model, irrespective of the internal cost-sensitive classifier used. It thus retains the comprehensibility of the models produced by the base learner (when they possess this quality, which is the case for simple decision tree and rule learners), while potentially drawing upon the strengths of a powerful internal cost-sensitive classifier (e.g. an ensemble-based classifier; see Section 2.4) [21,24].

2.3 Over- and Undersampling

Over- and undersampling are aimed at making an arbitrary error-based classification algorithm cost-sensitive by manipulation of the training data priors (which
are assumed to be representative of the population priors. The method can be applied to arbitrary two-class cost matrices that comply with the reasonableness conditions specified in Section 2.1. MetaCost then starts from the transformed cost matrix $C^0$ as specified in Section 2.1. Over- and undersampling cannot be directly applied to arbitrary multiclass cost matrices. Strictly speaking, it is only applicable to multiclass problems with a particular type of transformed cost matrix $C^0$, where there is a constant misclassification cost $C^0(j)$ for class $j$ irrespective of the predicted class. In other cases, Breiman et al. [25] suggest converting cost matrix $C^0$ into $C^{00}$ by employing $C^{00}(j) = \sum_{i \neq j} C^0(i, j)$ for $j = \{1, \ldots, T\}$ to make the method applicable.

The rationale of changing the priors of the classes according to the cost structure then goes as follows. After transforming the original cost matrix $C$ into a cost matrix $C$, the minimum expected cost classifier $H(x)$ can be specified as follows:

$$H(x) = \arg \min_{t \in \{1, \ldots, T\}} \sum_{j \in \{1, \ldots, T\}, j \neq t} p(j|x) C^0(j)$$

(4)

where $C^0(j)$ stands for the off-diagonal entry for column $j$ in $C^0$. If we use Bayes theorem and note that $p(x)$ does not depend on the class label, and therefore does not affect the classification decision, we can reformulate the decision in Eq.(4) as follows:

$$H(x) = \arg \min_{t \in \{1, \ldots, T\}} \sum_{j \in \{1, \ldots, T\}, j \neq t} p(j) p(x|j) C^0(j).$$

(5)

Now, the decision in Eq.(5) is equivalent to

$$H(x) = \arg \min_{t \in \{1, \ldots, T\}} \sum_{j \in \{1, \ldots, T\}, j \neq t} p'(j) p(x|j)$$

(6)

where we specify altered priors $p'(j) = \frac{p(j) C^0(j)}{\sum_{i \neq j} p(i) C^0(i)}$. In other words, this suggests that a natural way to deal with the problem of constant cost structure $C^0(j)$ is to redefine the priors and proceed as though it were a unit cost problem [25].

In the undersampling scenario all training data instances of the class $j$ with the highest $p'(j)$ are retained and a fraction $\frac{p'(j)}{p'(j)}$ of the training data instances of each other class $i$ are selected at random for inclusion in the resampled training set. The obvious disadvantage of this method is that it reduces the data available for training, which may be undesirable. Alternatively, to avoid the loss of training data, we may apply the oversampling scenario, where all training data instances of the class $j$ with the lowest $p'(j)$ are retained, and the training data instances of every other class $i$ are duplicated approximately $\frac{p'(i)}{p'(j)}$ times in the resampled training set [21]. This scenario may, however, seriously increase training time. Note that some learning algorithms can take into account weights specified on the training data instances during training. Resampling of the training data can then be substituted by reweighting of the training data instances according to the
altered priors, thus avoiding some of the disadvantages of resampling. However, when the cost matrix changes, the method will, just like the resampling scenarios, still need to be reiterated.

2.4 Cost-Sensitive Boosting

The mechanics of boosting rest on the construction of a sequence of classifiers, where each classifier is trained on a resampled (or reweighted, if applicable) training set where those training data instances that got poorly predicted in the previous runs receive a higher weight in the next run. At termination, i.e. after a fixed number of iterations, the constructed classifiers are then combined by weighted or simple voting schemes. The idea underlying the sequential perturbation of the training data is that the base learner gets to focus incrementally on those regions of the data that are harder to learn. For a variety of error-based learners boosting has been reported to reduce misclassification error, bias and/or variance (see e.g. [26,27,28,29,30]).

In this section, we concentrate on the use of boosting for cost-sensitive classification based on an arbitrary error-based base learner. We first describe the boosting method used in this paper, i.e. AdaBoost due to Shapire and Singer [31] designed for two-class problems. Then we present a number of cost-sensitive adaptations that have been proposed in the literature for this problem setting.

Following the generalized setting by Shapire and Singer [31], AdaBoost for a two-class problem works as follows. For training data \( \{x_i, t_i\}_{i=1}^N \) with input vectors \( x_i \in \mathbb{R}^n \) and class labels \( t_i \in \{1, -1\} \), error-based learner \( EBL \), and cost matrix \( C \), let the initial instance weights be \( w^{(1)}_i = \frac{1}{N} \). Then proceed as follows for run \( k \) going from 1 to \( K \) (predefined parameter):

1. Generate data \( D^{(k)} \) by resampling \( D \) using weights \( w^{(k)}_i \).
2. Learn model \( H^{(k)} \) by applying \( EBL \) to \( D^{(k)} \). Let \( H^{(k)}(x) \in \{1, -1\} \) denote the model’s class prediction, and \( h^{(k)}_t(x) \in [0, 1] \) denote the model’s confidence in class prediction \( t \) given input vector \( x \), i.e. its estimate of \( p(t|x) \), only applicable if the model generated by \( EBL \) is able to produce the latter.
3. Compute \( \alpha^{(k)} \in \mathbb{R} \) as follows:\(^3\)
   \[
   \alpha^{(k)} = \frac{1}{2} \ln \left( \frac{1 + r^{(k)}}{1 - r^{(k)}} \right) \tag{7}
   \]
   with \( r^{(k)} \) defined as follows:
   \[
   r^{(k)} = \sum_{i=1}^N w^{(k)}_i h^{(k)}(x_i) t_i \tag{8}
   \]
   where \( h^{(k)}(x_i) = H^{(k)}(x_i) h_{H^{(k)}(x_i)}(x_i) \), if applicable, otherwise \( h^{(k)}(x_i) = H^{(k)}(x_i) \).

\(^3\) This choice of \( \alpha^{(k)} \) was derived analytically by Freund and Shapire [32]. Shapire and Singer [31] leave this choice unspecified and discuss multiple tuning mechanisms.
4. Update instance weights as follows:

\[
w_i^{(k+1)} = w_i^{(k)} \exp \left( -\alpha^{(k)} h^{(k)}(x_i) t_i \right).
\] (9)

5. Normalize \( w_i^{(k+1)} \) by demanding that \( \sum_{i=1}^N w_i^{(k+1)} = 1 \).^4

The following final model combination hypothesis \( H(x) : \mathbb{R}^n \to \{-1, 1\} \) is then proposed for classifying a new case \([31]\):

\[
H(x) = \text{sign} \left( \sum_{k=1}^K \alpha^{(k)} h^{(k)}(x) \right).
\] (10)

Now we can try to make AdaBoost cost-sensitive. A first way of making AdaBoost cost-sensitive is by replacing the model combination hypothesis \( H(x) \) in Eq.(10) with minimum expected cost classification. This means we need to estimate \( p(t|x) \). Friedman et al. \([33]\) suggest passing the real-valued predictions of AdaBoost through a sigmoid transform, i.e. using a logistic model, to obtain an estimate of \( p(t = 1|x) \), i.e.

\[
\hat{p}(t = 1|x) = \frac{1}{1 + \exp \left( -\sum_{k=1}^K \alpha^{(k)} h^{(k)}(x) \right)}
\] (11)

which is equivalent to interpreting the real-valued output of AdaBoost as the log odds of a positive data instance. The rationale underlying this choice is the close connection between the exponential criterion that AdaBoost attempts to optimize and the negative log likelihood associated with the logistic model \([34]\).

A second way to make AdaBoost cost-sensitive is by using the AdaCost cost-sensitive learning variant of AdaBoost due to Fan et al. \([35]\). AdaCost uses the cost of misclassifications to update the training distribution on successive boosting runs, a method Domingos \([21]\) qualifies as more \textit{ad hoc}, lacking clear theoretical foundations for its resampling (or reweighting) scheme. AdaCost updates the instance weights of costly wrong classifications more aggressively and decreases the weight of costly correct classifications more conservatively. To create this effect AdaCost introduces a cost adjustment function \( \beta \left( H^{(k)}(x_i), t_i, c_i \right) \) into AdaBoost Eq.(8) and Eq.(9), with \( c_i \in \mathbb{R}^+ \) the instance-dependent, fixed, normalized\(^5\) cost of misclassifying training data instance \((x_i, t_i)\). It is assumed that correct classifications are associated with zero cost. Given a stable, instance-independent, two-class cost matrix \( C \) that is in line with the reasonableness conditions specified in Section 2.1, the cost specification required by AdaCost can always be mimicked by transforming \( C \) into cost matrix \( C' \) with zero diagonal elements and non-negative values for the off-diagonal elements, as described in

\(^4\) In order to avoid numerical underflow problems, instance weights of less than \( 10^{-8} \) are automatically set to \( 10^{-8} \) before normalizing \([26]\).

\(^5\) AdaCost starts from \( c_i \) that are normalized to \([0, 1]\). Therefore each cost value is divided by the maximum cost value.
Section 2.1. The cost adjustment function is chosen by Fan et al. [35] as follows: 
\[ \beta(1, c_i) = -0.5 c_i + 0.5 \quad \text{and} \quad \beta(-1, c_i) = 0.5 c_i + 0.5. \]
AdaCost then modifies Eq.(8) as follows:
\[ r^{(k)} = \sum_{i=1}^{N} w_i^{(k)} h^{(k)}(x_i) t_i \beta \left( H^{(k)}(x_i) t_i, c_i \right) \].  
(12)

The weight-update rule in Eq.(9) is modified as follows:
\[ w_i^{(k+1)} = w_i^{(k)} \exp \left( -\alpha^{(k)} h^{(k)}(x_i) t_i \beta \left( H^{(k)}(x_i) t_i, c_i \right) \right) \].  
(13)
Furthermore, the instance weights at run \( k = 1 \) are set as follows:
\[ w_i^{(1)} = \frac{c_i}{\sum_{i=1}^{N} c_i} \].  
(14)

As specified before, some learning algorithms can take into account weights specified on the training data instances during training. Resampling of the training data can then be substituted by reweighting of the training data instances, thus avoiding some of the disadvantages of resampling. A disadvantage of AdaCost remains, however, that it requires repeating all runs every time the costs change, which is not the case when applying AdaBoost in combination with minimum expected cost classification. Retraining can be a burdensome task. Also note that, like AdaBoost and similar multiple model or ensemble learning approaches which can improve significantly on the predictive power and stability of single models, AdaCost is not able to retain the comprehensibility of the models produced by the base learner (when they possess this quality; which is the case for simple decision tree and rule learners) [24].

3 PIP Claims Data

For the empirical part of the paper we used a representative data set of 1,399 PIP automobile insurance closed claim files from accidents that occurred in Massachusetts during 1993 and for which information was meticulously collected by the Automobile Insurers Bureau (AIB) of Massachusetts [7,36]. For all the claims, the AIB tracked information on twenty-five binary fraud indicators (a.k.a. red flags) and twelve non-indicator predictors, i.e. discretized continuous predictors, that were all supposed to make sense to claims adjusters and fraud investigators. The predictors pertained to characteristics of the accident, the claimant, the policy, the insured driver, the injury, the medical treatment and lost wages. All included predictors are typically available relatively early in the life of a claim [7,37].

\[ \text{Decision tree and rule learners are notoriously instable, i.e. in the face of limited data they produce models that can dramatically change with small changes in the data [24].} \]
### Table 2. Cost structure for the Massachusetts claim fraud data

<table>
<thead>
<tr>
<th>Cost</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost(TN)</td>
<td>no cost = 0</td>
</tr>
<tr>
<td>Cost(TP)</td>
<td>audit cost - saved average claim cost = 350 - 750 = 400</td>
</tr>
<tr>
<td>Cost(FP)</td>
<td>audit cost = 350</td>
</tr>
<tr>
<td>Cost(FN)</td>
<td>lost average claim cost = 750</td>
</tr>
</tbody>
</table>

Each claim file was reviewed by a senior claims manager on the basis of all available information. This closed claims reviewing was summarized into a ten-point scale expert assessment of suspicion of fraud, with zero being the lowest and ten the highest score. Each claim was also categorized in terms of the following five-level verbal assessment hierarchy: probably legitimate, excessive treatment (build-up) only, suspected opportunistic fraud, and suspected planned fraud. In auto insurance, a fraudulent claim was defined operationally for the Massachusetts study as a claim for an injury in an accident that did not happen or an injury unrelated to a real accident. The qualification of each available claim by both a verbal expert assessment of suspicion of fraud as well as a ten-point scale suspicion score gave rise to several alternative target encoding scenarios [7]. For instance, each definition threshold imposed on the ten-point scale then defines a specific view/policy towards the investigation of claim fraud; e.g., a 4+ target encoding, i.e. if suspicion rate < 4 THEN pass, ELSE no pass, then would make technical, if not verbal, sense. For the latter scenario, i.e. the one used in this study, about 28% of the claims contained enough suspicious elements to be further investigated for fraud. Details on the semantics of the data set can be found in [7,36].

After deliberation with the AIB we settled on a simplified cost model for the data at hand. The actual cost model used by insurers in practise may be more complex. We hypothesize the following situation. Whenever a new claim arrives, our claim screening facility makes a prediction (either non-suspicious=NS or suspicious=S), costlessly. If the claim gets predicted NS, then it is swiftly and routinely settled as stipulated by the obligations of the insurance contract. On the other hand, if predicted S, then the claim gets audited, i.e. S triggers a decision to acquire more information at a price. The claim investigation typically is costly but has no (direct) effect on the claim amount. We make abstraction of any other strategic reasons to audit claims, either by targeting or at random. Typically, the cost of audit will depend upon the degree to which the suspicion signal is strong. Here we settle for a fixed cost of $350, which is roughly the cost of an independent medical examination (IME). We furthermore assume that by auditing the claim we can exactly determine the nature of the claim. Studies by the AIB have shown PIP reductions — first party only, no liability — on average of about $750 when an IME is used successfully. For a false negative, i.e.
the claim gets predicted NS while it actually should have been audited, we can assume the claim would be paid without additional cost of investigation. The cost then would be the loss of a possible reduction. With the above information we specify the cost structure in Table 2.

4 Empirical Evaluation

The decision tree and rule learners taken up in this study — available in the Weka 3-3-1 open-source machine learning library\textsuperscript{7} [13] — are:\textsuperscript{8} C4.5 decision trees due to Quinlan \cite{quinlan}, unpruned C4.5 decision trees (C4.5\textsubscript{u}), a single conjunctive rule learner (ConjR), decision stumps (DStump) due to Iba and Thompson \cite{iba}, Holte's very simple classification rules (OneR) \cite{holte}, and Part rule lists (first matching rule wins) due to Frank and Witten \cite{frank}. We also include Quinlan’s C4.5rules rule lists (first matching rule wins) \cite{quinlan}, not available in Weka 3-3-1, but available as part of the freely available C4.5 Release 8 distribution \cite{quinlan}.\textsuperscript{9} All decision tree and rule learners were run using default parameter settings. We furthermore report on standard logistic regression (Logit), fitted with SAS for Windows V8 PROC LOGISTIC (default options: TECHNIQUE=FISHER and RIDGING=RELATIVE), and on naive Bayes (NB) with smoothing based on imposing a uniform Dirichlet prior on the tabular conditional probabilities (for individual predictors given the class) estimated from the training data \cite{murphy}.

We present the results of an empirical comparison of the cost-sensitive learning and decision making methods discussed in Section 2 using the above specified base learners and the data described in Section 3. We used a split-sample (a.k.a. hold-out) experimental setup for evaluating the methods: 2/3 of the data was used for training, the remaining 1/3 was left for testing. This basic experiment was repeated 30 times, each time using a different randomization of the data. The same randomizations were used for the evaluation of all the methods. For this repeated split-sample experiment, the performance of the methods is readily assessed by averaging over the 30 experiments. Performance differences can also be tested using paired t-tests. The performance measure used in this study is the total cost of classification evaluated on the test set. Each boosted classifier consisted of a sequence of 25 sub-classifiers and each bagged classifier used by MetaCost relied on 25 bootstraps of the training data (see Bauer and Kohavi \cite{auer} for the choice of these numbers). For both bagging and boosting the final classifier was built using all generated sub-classifiers (see Bauer and Kohavi \cite{auer}).

Remark that the purpose of the experiment reported here was not so much to identify a single best method for cost-sensitive classification, but rather to compare and illustrate the usefulness of the alternative methods discussed in

\textsuperscript{7} The source code can be obtained from http://www.cs.waikato.ac.nz/~ml/weka/.

\textsuperscript{8} Laplace corrected training frequency counts were used to estimate posterior class membership probabilities from the tree leaves and individual rules \cite{laplace,laplace2}.

\textsuperscript{9} The source code can be obtained from http://www.cse.unsw.edu.au/~quinlan/.
Section 2 for several error-based decision tree and rule learners and the claim fraud data described in Section 3.

Table 3 presents the average total cost of classification on the test set over 30 split-sample experiments using the base learners specified in Section 4 in combination with the following methods: direct minimum expected cost classification (DC), MetaCost using bagging and minimum expected cost classification in step 1 of the MetaCost method (MC), Oversampling (OS), Undersampling (US), cost-sensitive AdaBoost (AB), and AdaCost (AC).

Columns labelled bo used the binary output of the base classifiers. Columns labelled cr used confidence-rated base classifier predictions. This is especially relevant for the ensemble-based methods, i.e. MC, AB and AC, where we use the cost ratio to compare both. Boldfaced ratios indicate a significant difference between the denominator and the numerator, according to a paired two-sided t-test (0.05 significance level). The minimum cost per row is bold-faced, and those entries in that row showing no significant degradation, according to a paired one-sided t-test (0.025 significance level), are italicized. The minimum cost per column is double-underlined, and those entries in that column showing no significant degradation, according to a paired one-sided t-test (0.025 significance level), are single-underlined. Entries marked 1 in the superscript show no significant degradation vis-à-vis the overall minimum average cost, i.e. 1563 for NB OS-bo, according to a paired one-sided t-test (0.025 significance level). ER-bo stands for the simple error-based classification that makes abstraction of the cost information.

From the results in Table 3 it is clear that NB and Logit show superior performance to the decision tree and rule learners. We note the excellent performance of DC-cr for Logit and NB. This should not come as much of a surprise considering the fact that they are explicitly designed to yield probability estimates. Furthermore, the performance of MC-cr and AC-cr for C4.5, C4.5rules and Part is noteworthy in the positive sense, especially with respect to DC-cr and the corresponding bo-column variants, i.e. MC-bo and AC-bo. That ConjR, DStump and OneR produce bad posterior probability estimates should be clear from DC-cr. Also note the bad performance of the MC columns for ConjR, DStump and OneR. Though for ConjR and OneR MC-cr is superior to MC-bo, the performance remains unsatisfactory. Also, the cr-columns give no significant, if any, improvement on the bo-columns for AB and AC. OS-bo, US-bo, AB-bo, AC-bo and AC-cr show the best performance for ConjR, DStump and OneR.

Table 4 shows how the different learners compare with each other. The entries above the diagonal indicate the number of wins/ties/losses for the learner associated with the row versus the learner associated with the column, considering the average costs in Table 3 for all but the first column (ER-bo). The entries below the diagonal indicate the number of significant wins/ties/losses, according to a paired one-sided t-test (0.025 significance level). In the same way, Table 5 shows how the methods associated with the columns in Table 3 compare with each other.
As noted in Section 2.4, AB and AC are not able to retain the comprehensibility of the models produced by the base learner. One way to resolve this is to use the ensemble’s cost-sensitive classification in step 1 of the MetaCost method (instead of using bagging and minimum expected cost classification) and subsequently apply MetaCost steps 2 and 3 to produce a single, comprehensible model from the relabelled training data using the base learner that produced the sub-classifiers in the ensemble. Here we proceed in a way similar to Domingos’ CMM [24]. CMM is a meta-learner that combines multiple sub-classifiers into a single one with the aim of increasing comprehensibility, while retaining some of the strengths of the ensemble. While MetaCost uses the ensemble to relabel the training data, CMM uses it to relabel additional artificially generated examples [21]. Combining both ideas is left as an issue for further research. The same goes for the idea of using a different base learner for the ensemble and for learning the relabelled training data.

Table 6 presents the average total cost of classification on the test set over 30 split-sample experiments using the base learners specified in Section 4 in combination with the MetaCost rationale of relabelling the training data instances using: direct minimum expected cost classification (DC), bagging and minimum expected cost classification (= idem MC in Table 3), the binary predictions of Oversampling (OS), Undersampling (US), cost-sensitive AdaBoost (AB), and AdaCost (AC).

Columns labelled $bo$ used the binary output of the base classifiers. Columns labelled $cr$ used confidence-rated base classifier predictions. The triplet $a/b/c$ under each cost entry provides information on the following cost ratios: $a$ = the cost entry over the corresponding cost entry (= same column and row name) in Table 3, i.e. the cost entry associated with the direct application of the method that underlies the MetaCost rationale of relabelling the training data for the cost entry taken up in the numerator; $b$ = the cost entry over the DC-$cr$ cost entry in Table 3 for the learner under consideration; $c$ = the cost entry over $min(MC-bo,MC-cr)$ for the learner under consideration. Boldfaced ratios indicate a significant difference between the denominator and the numerator, according to a paired two-sided $t$-test (0.05 significance level). Entries marked 1 in the superscript show no significant degradation vis-à-vis the overall minimum average cost of 1563 for NB $OS-bo$ in Table 3, according to a paired one-sided $t$-test (0.025 significance level).

When we look at the results for C4.5, C4.5rule, C4.5rules and Part, we see that AC-$cr$, in contrast to the corresponding column in Table 3, is not capable of attaining similar performance to MC-$cr$. The combination of bagging using the confidence-rated predictions of the base classifier (a.k.a. $p$-bagging) and minimum expected cost classification is a clear winner here. Looking at ConjR, DSTump and OneR, we see that OS-$bo$, US-$bo$, AB-$bo$, AC-$bo$ and AC-$cr$, in general, stay close to the performance of the corresponding entries in Table 3. Finally, remark the beneficial effect of MetaCost post-processing on the AB-$bo$ and AB-$cr$ ensembles for NB.
5 Conclusion

We started this paper with a theoretical exposition on cost-sensitive learning and decision making, focusing on several more or less generic methods that aim to make a broad variety of error-based classification algorithms cost-sensitive. These methods were empirically evaluated (by means of a repeated split-sample experiment) for a variety of error-based decision tree and rule learners. For the empirical part we used a data set of Massachusetts closed personal injury protection (PIP) automobile insurance claims that were previously investigated for suspicion of fraud by domain experts and for which cost information was obtained. Standard logistic regression and (smoothed) naive Bayes were used as benchmarks. The purpose of the empirical evaluation was primarily to compare and illustrate the usefulness of the documented methods and to provide a baseline for further research.

References

15. Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: Seventh ACM SIGKDD Conference on Knowledge Discovery in Data Mining, San Francisco (2001)


Table 3. Average total cost of classification on the test set using the base learners specified in Section 4 in combination with the following methods: direct minimum expected cost classification (DC), MetaCost using bagging and minimum expected cost classification in step 1 of the MetaCost method (MC), Oversampling (OS), Undersampling (US), cost-sensitive AdaBoost (AB), and AdaCost (AC). Columns labelled \( \text{bo} \) used the binary output of the base classifiers. Columns labelled \( \text{cr} \) used confidence-rated base classifier predictions. Boldfaced ratios \( \frac{\text{cr}}{\text{bo}} \) indicate a significant difference between the denominator and the numerator, according to a paired two-sided \( t \)-test (0.05 significance level). The minimum cost per row is bold-faced, and those entries in that row showing no significant degradation, according to a paired one-sided \( t \)-test (0.025 significance level), are italicized. The minimum cost per column is double-underlined, and those entries in that column showing no significant degradation, according to a paired one-sided \( t \)-test (0.025 significance level), are single-underlined. Entries marked 1 in the superscript show no significant degradation vis-à-vis the overall minimum average cost, i.e. 1563 for NB \( \text{OS-bo} \), according to a paired one-sided \( t \)-test (0.025 significance level). ER-\( \text{bo} \) stands for the simple error-based classification that makes abstraction of the cost information.

<table>
<thead>
<tr>
<th></th>
<th>ER</th>
<th>DC</th>
<th>MC</th>
<th>OS</th>
<th>US</th>
<th>AB</th>
<th>AC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{bo} )</td>
<td>( \text{cr} )</td>
<td>( \text{bo} )</td>
<td>( \text{cr} )</td>
<td>( \frac{\text{cr}}{\text{bo}} )</td>
<td>( \text{bo} )</td>
<td>( \text{cr} )</td>
</tr>
<tr>
<td>C4.5</td>
<td>39545</td>
<td>14322</td>
<td>11998</td>
<td>5923</td>
<td>0.5</td>
<td>18537</td>
<td>9392</td>
</tr>
<tr>
<td>C4.5u</td>
<td>45363</td>
<td>17535</td>
<td>29685</td>
<td>9730</td>
<td>0.3</td>
<td>42570</td>
<td>27192</td>
</tr>
<tr>
<td>C4.5rules</td>
<td>24135</td>
<td>20995</td>
<td>14508</td>
<td>7848</td>
<td>0.5</td>
<td>31970</td>
<td>11743</td>
</tr>
<tr>
<td>ConjR</td>
<td>91147</td>
<td>59438</td>
<td>62622</td>
<td>42583</td>
<td>0.7</td>
<td>10012</td>
<td>11593</td>
</tr>
<tr>
<td>DStump</td>
<td>74610</td>
<td>82742</td>
<td>24457</td>
<td>30593</td>
<td>1.3</td>
<td>10012</td>
<td>10560</td>
</tr>
<tr>
<td>OneR</td>
<td>87615</td>
<td>56603</td>
<td>70917</td>
<td>32422</td>
<td>0.5</td>
<td>10012</td>
<td>10012</td>
</tr>
<tr>
<td>Part</td>
<td>35418</td>
<td>19252</td>
<td>15175</td>
<td>7640</td>
<td>0.5</td>
<td>27315</td>
<td>15992</td>
</tr>
<tr>
<td>Logit</td>
<td>25138</td>
<td>24031</td>
<td>14825</td>
<td>25451</td>
<td>0.2</td>
<td>3312</td>
<td>4618</td>
</tr>
<tr>
<td>NB</td>
<td>14587</td>
<td>25601</td>
<td>10450</td>
<td>5827</td>
<td>0.6</td>
<td>15631</td>
<td>20451</td>
</tr>
</tbody>
</table>
Table 4. How the different learners compare with each other. The entries above the diagonal indicate the number of wins/ties/losses for the learner associated with the row versus the learner associated with the column, considering the average costs in Table 3 for all but the first column (ER-bo). The entries below the diagonal indicate the number of significant wins/ties/losses, according to a paired one-sided t-test (0.025 significance level)

<table>
<thead>
<tr>
<th></th>
<th>C4.5</th>
<th>C4.5u</th>
<th>C4.5rules</th>
<th>ConjR</th>
<th>DStump</th>
<th>OneR</th>
<th>Part</th>
<th>Logit</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>-</td>
<td>9/0/0</td>
<td>7/0/2</td>
<td>6/0/3</td>
<td>6/0/3</td>
<td>6/0/3</td>
<td>7/0/2</td>
<td>1/0/8</td>
<td>0/0/9</td>
</tr>
<tr>
<td>C4.5u</td>
<td>0/2/7</td>
<td>-</td>
<td>1/0/8</td>
<td>3/0/6</td>
<td>2/0/7</td>
<td>3/0/6</td>
<td>1/0/8</td>
<td>0/0/9</td>
<td>0/0/9</td>
</tr>
<tr>
<td>C4.5rules</td>
<td>1/4/4</td>
<td>5/4/0</td>
<td>-</td>
<td>3/0/6</td>
<td>3/0/6</td>
<td>3/0/6</td>
<td>2/0/7</td>
<td>1/0/8</td>
<td>0/0/9</td>
</tr>
<tr>
<td>ConjR</td>
<td>3/2/4</td>
<td>6/0/3</td>
<td>4/2/3</td>
<td>-</td>
<td>3/3/3</td>
<td>1/3/5</td>
<td>5/0/4</td>
<td>1/0/8</td>
<td>1/0/8</td>
</tr>
<tr>
<td>DStump</td>
<td>3/2/4</td>
<td>6/1/2</td>
<td>4/3/2</td>
<td>1/7/1</td>
<td>-</td>
<td>2/3/4</td>
<td>5/0/4</td>
<td>1/0/8</td>
<td>1/0/8</td>
</tr>
<tr>
<td>OneR</td>
<td>3/2/4</td>
<td>6/0/3</td>
<td>4/2/3</td>
<td>0/9/0</td>
<td>1/7/1</td>
<td>-</td>
<td>5/0/4</td>
<td>1/0/8</td>
<td>1/0/8</td>
</tr>
<tr>
<td>Part</td>
<td>2/4/3</td>
<td>7/2/0</td>
<td>4/4/1</td>
<td>3/2/4</td>
<td>2/3/4</td>
<td>3/2/4</td>
<td>-</td>
<td>0/0/9</td>
<td>0/0/9</td>
</tr>
<tr>
<td>Logit</td>
<td>6/3/0</td>
<td>9/0/0</td>
<td>8/1/0</td>
<td>8/1/0</td>
<td>7/2/0</td>
<td>8/1/0</td>
<td>8/1/0</td>
<td>-</td>
<td>3/0/6</td>
</tr>
<tr>
<td>NB</td>
<td>7/2/0</td>
<td>9/0/0</td>
<td>8/1/0</td>
<td>8/1/0</td>
<td>8/1/0</td>
<td>8/1/0</td>
<td>1/7/1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5. How the methods associated with the columns in Table 3 compare with each other. The entries above the diagonal indicate the number of wins/ties/losses for the method associated with the row versus the method associated with the column, considering the average costs in Table 3. The entries below the diagonal indicate the number of significant wins/ties/losses, according to a paired one-sided t-test (0.025 significance level)

<table>
<thead>
<tr>
<th></th>
<th>ER</th>
<th>DC</th>
<th>MC-bo</th>
<th>MC-cr</th>
<th>OS</th>
<th>US</th>
<th>AB-bo</th>
<th>AB-cr</th>
<th>AC-bo</th>
<th>AC-cr</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER</td>
<td>-</td>
<td>1/0/8</td>
<td>0/0/9</td>
<td>0/0/9</td>
<td>1/0/8</td>
<td>0/0/9</td>
<td>1/0/8</td>
<td>1/0/8</td>
<td>0/0/9</td>
<td>0/0/9</td>
</tr>
<tr>
<td>DC</td>
<td>7/2/0</td>
<td>-</td>
<td>5/0/4</td>
<td>2/0/7</td>
<td>5/0/4</td>
<td>2/0/7</td>
<td>6/0/3</td>
<td>6/0/3</td>
<td>3/0/6</td>
<td>1/0/8</td>
</tr>
<tr>
<td>MC-bo</td>
<td>9/0/0</td>
<td>3/2/4</td>
<td>-</td>
<td>1/0/8</td>
<td>4/0/5</td>
<td>1/0/8</td>
<td>5/0/4</td>
<td>4/0/5</td>
<td>1/0/8</td>
<td>0/0/9</td>
</tr>
<tr>
<td>MC-cr</td>
<td>9/0/0</td>
<td>7/1/1</td>
<td>8/1/0</td>
<td>-</td>
<td>5/0/4</td>
<td>5/0/4</td>
<td>6/0/3</td>
<td>5/0/4</td>
<td>5/0/4</td>
<td>4/0/5</td>
</tr>
<tr>
<td>OS</td>
<td>7/1/1</td>
<td>3/2/4</td>
<td>5/0/4</td>
<td>4/1/4</td>
<td>-</td>
<td>4/1/4</td>
<td>7/0/2</td>
<td>6/0/3</td>
<td>5/0/4</td>
<td>1/3/5</td>
</tr>
<tr>
<td>US</td>
<td>9/0/0</td>
<td>5/3/1</td>
<td>5/4/0</td>
<td>4/1/4</td>
<td>4/5/0</td>
<td>-</td>
<td>7/0/2</td>
<td>8/0/1</td>
<td>5/0/4</td>
<td>1/1/7</td>
</tr>
<tr>
<td>AB-bo</td>
<td>8/0/1</td>
<td>3/0/6</td>
<td>4/1/4</td>
<td>3/0/6</td>
<td>1/5/3</td>
<td>0/3/6</td>
<td>-</td>
<td>4/0/5</td>
<td>3/0/6</td>
<td>0/0/9</td>
</tr>
<tr>
<td>AB-cr</td>
<td>8/0/1</td>
<td>3/2/4</td>
<td>4/2/3</td>
<td>3/2/4</td>
<td>1/4/4</td>
<td>0/4/5</td>
<td>4/3/2</td>
<td>-</td>
<td>1/0/8</td>
<td>0/0/9</td>
</tr>
<tr>
<td>AC-bo</td>
<td>9/0/0</td>
<td>6/2/1</td>
<td>7/2/0</td>
<td>3/2/4</td>
<td>4/5/0</td>
<td>2/7/0</td>
<td>6/3/0</td>
<td>6/3/0</td>
<td>-</td>
<td>0/0/9</td>
</tr>
<tr>
<td>AC-cr</td>
<td>9/0/0</td>
<td>7/2/0</td>
<td>8/1/0</td>
<td>4/3/2</td>
<td>4/5/0</td>
<td>3/6/0</td>
<td>6/3/0</td>
<td>6/3/0</td>
<td>4/5/0</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 6. Average total cost of classification on the test set using the base learners specified in Section 4 in combination with the MetaCost rationale of relabelling the training data instances using: direct minimum expected cost classification (DC), bagging and minimum expected cost classification (= idem MC in Table 3), the binary predictions of Oversampling (OS), Undersampling (US), cost-sensitive AdaBoost (AB), and AdaCost (AC). Columns labelled \( bo \) used the binary output of the base classifiers. Columns labelled \( cr \) used confidence-rated base classifier predictions. The triplet \( a=b=c \) under each cost entry provides information on the following cost ratios: \( a = \) the cost entry over the corresponding cost entry in Table 3; \( b = \) the cost entry over the DC-\( cr \) cost entry in Table 3 for the learner under consideration; \( c = \) the cost entry over min(MC-\( bo \),MC-\( cr \)) for the learner under consideration. Boldfaced ratios indicate a significant difference between the denominator and the numerator, according to a paired two-sided \( t \)-test (0.05 significance level). Entries marked 1 in the superscript show no significant degradation vis-à-vis the overall minimum average cost of 1563 for NB OS-\( bo \) in Table 3, according to a paired one-sided \( t \)-test (0.025 significance level).

<table>
<thead>
<tr>
<th></th>
<th>DC</th>
<th>OS</th>
<th>US</th>
<th>AB</th>
<th>AC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( cr )</td>
<td>( bo )</td>
<td>( cr )</td>
<td>( bo )</td>
<td>( cr )</td>
</tr>
<tr>
<td>C4.5</td>
<td>10978</td>
<td>11998</td>
<td>5923</td>
<td>0.5</td>
<td>16067</td>
</tr>
<tr>
<td>C4.5x</td>
<td>22348</td>
<td>29685</td>
<td>9730</td>
<td>0.3</td>
<td>39760</td>
</tr>
<tr>
<td>C4.5rules</td>
<td>20812</td>
<td>14508</td>
<td>7848</td>
<td>0.5</td>
<td>22335</td>
</tr>
<tr>
<td>ConjR</td>
<td>57585</td>
<td>62622</td>
<td>42583</td>
<td>0.7</td>
<td>10012</td>
</tr>
<tr>
<td>DStump</td>
<td>82742</td>
<td>24457</td>
<td>30593</td>
<td>1.3</td>
<td>10012</td>
</tr>
<tr>
<td>OneR</td>
<td>56603</td>
<td>70917</td>
<td>32422</td>
<td>0.5</td>
<td>10012</td>
</tr>
<tr>
<td>Part</td>
<td>15257</td>
<td>15175</td>
<td>7640</td>
<td>0.5</td>
<td>24830</td>
</tr>
<tr>
<td>Logit</td>
<td>3232(^2)</td>
<td>14825</td>
<td>2545(^2)</td>
<td>0.2</td>
<td>3677(^2)</td>
</tr>
<tr>
<td>NB</td>
<td>5795</td>
<td>10480</td>
<td>5827</td>
<td>0.6</td>
<td>5588</td>
</tr>
<tr>
<td></td>
<td>2.3/2.3/1.0</td>
<td>1.0/4.1/1.8</td>
<td>1.0/2.3/1.0</td>
<td>3.6/2.2/1.0</td>
<td>2.5/2.0/0.9</td>
</tr>
</tbody>
</table>