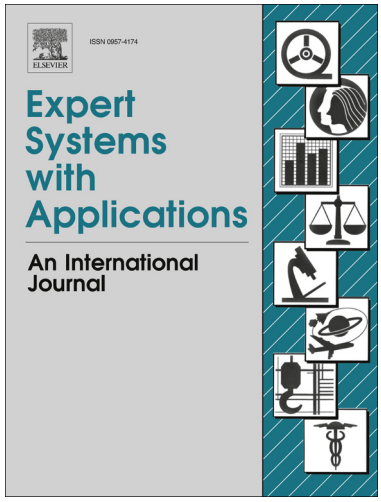



# Example-Dependent Cost-Sensitive Decision Trees

Alejandro Correa Bahnsen, Djamila Aouada, Björn Ottersten

PII: S0957-4174(15)00284-5
DOI: http://dx.doi.org/10.1016/j.eswa.2015.04.042
Reference: ESWA 9989

To appear in: *Expert Systems with Applications*

Please cite this article as: Bahnsen, A.C., Aouada, D., Ottersten, B., Example-Dependent Cost-Sensitive Decision Trees, *Expert Systems with Applications* (2015), doi: http://dx.doi.org/10.1016/j.eswa.2015.04.042

# Example-Dependent Cost-Sensitive Decision Trees

Alejandro Correa Bahnsen, Djamila Aouada, Björn Ottersten

*Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg*

**Abstract**

Several real-world classification problems are example-dependent cost-sensitive in nature, where the costs due to misclassification vary between examples. However, standard classification methods do not take these costs into account, and assume a constant cost of misclassification errors. State-of-the-art example-dependent cost-sensitive techniques only introduce the cost to the algorithm, either before or after training, therefore, leaving opportunities to investigate the potential impact of algorithms that take into account the real financial example-dependent costs during an algorithm training. In this paper, we propose an example-dependent cost-sensitive decision tree algorithm, by incorporating the different example-dependent costs into a new cost-based impurity measure and a new cost-based pruning criteria. Then, using three different databases, from three real-world applications: credit card fraud detection, credit scoring and direct marketing, we evaluate the proposed method. The results show that the proposed algorithm is the best performing method for all databases. Furthermore, when compared against a standard decision tree, our method builds significantly smaller trees in only a

*Email addresses:* alejandro.correa@uni.lu (Alejandro Correa Bahnsen), djamila.aouada@uni.lu (Djamila Aouada), bjorn.ottersten@uni.lu (Björn Ottersten)

fifth of the time, while having a superior performance measured by cost savings, leading to a method that not only has more business-oriented results, but also a method that creates simpler models that are easier to analyze.
*Keywords:* Cost-sensitive learning, Cost-Sensitive Classifier, Credit scoring, Fraud detection, Direct marketing, Decision trees

## 1. Introduction

Classification, in the context of machine learning, deals with the problem of predicting the class of a set of examples given their features. Traditionally, classification methods aim at minimizing the misclassification of examples, in which an example is misclassified if the predicted class is different from the true class. Such a traditional framework assumes that all misclassification errors carry the same cost. This is not the case in many real-world applications. Methods that use different misclassification costs are known as cost-sensitive classifiers. Typical cost-sensitive approaches assume a constant cost for each type of error, in the sense that, the cost depends on the class and is the same among examples (Elkan, 2001; Kim et al., 2012). Although, this class-dependent approach is not realistic in many real-world applications, for example in credit card fraud detection, failing to detect a fraudulent transaction may have an economical impact from a few to thousands of Euros, depending on the particular transaction and card holder (Sahin et al., 2013). In churn modeling, a model is used for predicting which customers are more likely to abandon a service provider. In this context, failing to identify a profitable or unprofitable churner have a significant different financial impact (Glady et al., 2009). Similarly, in direct marketing, wrongly predicting

that a customer will not accept an offer when in fact he will, has a different impact than the other way around (Zadrozny et al., 2003). Also in credit scoring, where declining good customers has a non constant impact since not all customers generate the same profit (Verbraken et al., 2014). Lastly, in the case of intrusion detection, classifying a benign connection as malicious have a different cost than when a malicious connection is accepted (Ma et al., 2011).

Methods that use different misclassification costs are known as cost-sensitive classifiers. In particular we are interested in methods that are example-dependent cost-sensitive, in the sense that the costs vary among examples and not only among classes (Elkan, 2001). However, the literature on example-dependent cost-sensitive methods is limited, mostly because there is a lack of publicly available datasets that fit the problem (Aodha and Brostow, 2013). Example-dependent cost-sensitive classification methods can be grouped according to the step where the costs are introduced into the system. Either the costs are introduced prior to the training of the algorithm, after the training or during training (Wang, 2013). In Figure 1, the different algorithms are grouped according to the stage in a classification system where they are used.

The first set of methods that were proposed to deal with cost-sensitivity consist in re-weighting the training examples based on their costs, either by cost-proportionate rejection-sampling (Zadrozny et al., 2003), or cost-proportionate over-sampling (Elkan, 2001). The rejection-sampling approach consists in selecting a random subset by randomly selecting examples from a training set, and accepting each example with probability equal to the

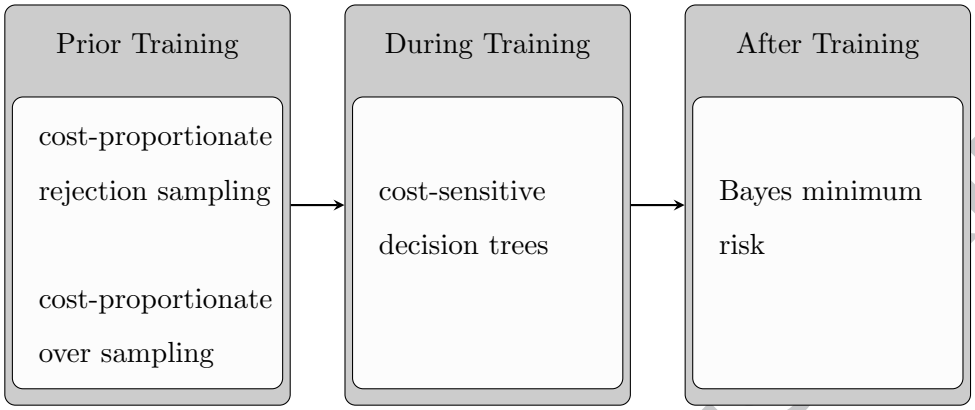



Figure 1: Different example-dependent cost-sensitive algorithms grouped according to the stage in a classification system where they are used.

normalized misclassification cost of the example. On the other hand, the over-sampling method consists in creating a new set, by making $n$ copies of each example, where $n$ is related to the normalized misclassification cost of the example. Recently, we proposed a direct cost approach to make the classification decision based on the expected costs. This method is called Bayes minimum risk ($BMR$), and has been successfully applied to detect credit card fraud (Correa Bahnsen et al., 2013, 2014c). The method consists in quantifying tradeoffs between various decisions using probabilities and the costs that accompany such decisions.

Nevertheless, these methods still use a cost-insensitive algorithm, and either by modifying the training set or the output probabilities convert it into a cost-sensitive classifier. Therefore, leaving opportunities to investigate the potential impact of algorithms that take into account the real financial example-dependent costs during the training of an algorithm.

The last way to introduce the costs into the algorithms is by modifying the methods. The main objective of doing this, is to make the algorithm take

into account the example-dependent costs during the training phase, instead of relying on a pre-processing or post-processing method to make classifiers cost-sensitive. In particular this has been done for decision trees (Draper et al., 1994; Ting, 2002; Ling et al., 2004; Li et al., 2005; Kretowski and Grześ, 2006; Vadera, 2010). In general, the methods introduce the misclassification costs into the construction of a decision trees by modifying the impurity measure, and weight it with respect of the costs (Lomax and Vadera, 2013). However, in all cases, approaches that have been proposed only deal with the problem when the cost depends on the class and not on the example.

In this paper we formalize a new measure in order to define when a problem is cost-insensitive, class-dependent cost-sensitive or example-dependent cost-sensitive. Moreover, we go beyond the aforementioned state-of-the-art methods, and propose a decision tree algorithm that includes the example-dependent costs. Our approach is based first on a new example-dependent cost-sensitive impurity measure, and secondly on a new pruning improvement measure which also depends on the cost of each example.

We evaluate the proposed example-dependent cost-sensitive decision tree using three different databases. In particular, a credit card fraud detection, a credit scoring and a direct marketing databases. The results show that the proposed method outperforms state-of-the-art example-dependent cost-sensitive methods. Furthermore, when compared against a standard decision tree, our method builds significantly smaller trees in only a fifth of the time. Furthermore, the source code used for the experiments is publicly available

as part of the *CostSensitiveClassification*[1] library.

By taking into account the real financial costs of the different real-world applications, our proposed example-dependent cost-sensitive decision tree is a better choice for these and many other applications. This is because, our algorithm is focusing on solving the actual business problems, and not proxies as standard classification models do. We foresee that our approach should open the door to developing more business focused algorithms, and that ultimately, the use of the actual financial costs during training will become a common practice.

The remainder of the paper is organized as follows. In Section 2, we explain the background behind example-dependent cost-sensitive classification and we define a new formal definition of cost-sensitive classification problems. In Section 3, we make an extensive review of current decision tree methods, including by the different impurity measures, growth methods, and pruning techniques. In Section 4, we propose a new example-dependent cost-sensitive decision tree. The experimental setup and the different datasets are described in Section 5. Subsequently, the proposed algorithm is evaluated on the different datasets. Finally, conclusions of the paper are given in Section 7.

## 2. Cost-Sensitive Cost Characteristic and Evaluation Measure

In this section we give the background behind example-dependent cost-sensitive classification. First we present the cost matrix, followed by a formal definition of cost-sensitive problems. Afterwards, we present an evaluation

[1] https://github.com/albahnsen/CostSensitiveClassification

measure based on cost. Finally, we describe the most important state-of-the-art methods, namely: Cost-proportionate sampling and Bayes minimum risk.

### *2.1. Binary classification cost characteristic*

In classification problems with two classes $y_i \in \{0, 1\}$, the objective is to learn or predict to which class $c_i \in \{0, 1\}$ a given example $i$ belongs based on its $k$ features $X_i = [x_i^1, x_i^2, ..., x_i^k]$. In this context, classification costs can be represented using a 2x2 cost matrix (Elkan, 2001), that introduces the costs associated with two types of correct classification, true positives ($C_{TP_i}$), true negatives ($C_{TN_i}$), and the two types of misclassification errors, false positives ($C_{FP_i}$), false negatives ($C_{FN_i}$), as defined below:

| | Actual Positive $y_i = 1$ | Actual Negative $y_i = 0$ |
|---|---|---|
| Predicted Positive $c_i = 1$ | $C_{TP_i}$ | $C_{FP_i}$ |
| Predicted Negative $c_i = 0$ | $C_{FN_i}$ | $C_{TN_i}$ |

Table 1: Classification cost matrix

Conceptually, the cost of correct classification should always be lower than the cost of misclassification. These are referred to as the "reasonableness" conditions Elkan (2001), and are defined as $C_{FP_i} > C_{TN_i}$ and $C_{FN_i} > C_{TP_i}$. Taking into account the "reasonableness" conditions, a simpler cost matrix with only one degree of freedom has been defined in Elkan (2001), by scaling and shifting the initial cost matrix, resulting in:

A classification problem is said to be cost-insensitive if costs of both errors are equal. It is class-dependent cost-sensitive if the costs are different

| | |
|---|---|
| Negative | $C^*_{FN_i} = \frac{(C_{FN_i} - C_{TN_i})}{(C_{FP_i} - C_{TN_i})}$ |
| Positive | $C^*_{TP_i} = \frac{(C_{TP_i} - C_{TN_i})}{(C_{FP_i} - C_{TN_i})}$ |

Table 2: Simplified cost matrix

but constant. Finally we talk about an example-dependent cost-sensitive classification problem if the cost matrix is not constant for all the examples.

However, the definition above is not general enough. There are many cases when the cost matrix is not constant and still the problem is cost-insensitive or class-dependent cost-sensitive. For example, if the costs of correct classification are zero, $C_{TP_i} = C_{TN_i} = 0$, and the costs of misclassification are $C_{FP_i} = a_0 \cdot z_i$ and $C_{FN_i} = a_1 \cdot z_i$, where $a_0$, $a_1$, are constant and $z_i$ a random variable. This is an example of a cost matrix that is not constant. However, $C^*_{FN_i}$ and $C^*_{TP_i}$ are constant, i.e. $C^*_{FN_i} = (a_1 \cdot z_i)/(a_0 \cdot z_i) = a_1/a_0$ and $C^*_{TP_i} = 0 \ \forall i$. In this case the problem is cost-insensitive if $a_0 = a_1$, or class-dependent cost-sensitive if $a_0 \neq a_1$, even given the fact that the cost matrix is not constant.

Nevertheless, using only the simpler cost matrix is not enough to define when a problem is example-dependent cost-sensitive. To achieve this, we define the classification problem cost characteristic as

$$b_i = C^*_{FN_i} - C^*_{TP_i}, \tag{1}$$

and define its mean and standard deviation as $\mu_b$ and $\sigma_b$, respectively.

Using $\mu_b$ and $\sigma_b$, we analyze different binary classification problems. In the case of a cost-insensitive classification problem, for every example $i$ $C_{FP_i} = C_{FN_i}$ and $C_{TP_i} = C_{TN_i}$, leading to $b_i = 1 \ \forall i$ or more generally $\mu_b = 1$

and $\sigma_b = 0$. For class-dependent cost-sensitive problems, the costs are not equal but constants $C_{FP_i} \neq C_{FN_i}$ or $C_{TP_i} \neq C_{TN_i}$, leading to $b_i \neq 1 \ \forall i$, or $\mu_b \neq 1$ and $\sigma_b = 0$. Lastly, in the case of example-dependent cost-sensitive problems, the cost difference is non constant or $\sigma_b \neq 0$.

In summary a binary classification problem is defined according to the following conditions:

| $\mu_b$ | $\sigma_b$ | Type of classification problem |
|---|---|---|
| 1 | 0 | cost-insensitive |
| $\neq 1$ | 0 | class-dependent cost-sensitive |
| | $\neq 0$ | example-dependent cost-sensitive |

### *2.2. Example-dependent cost-Sensitive evaluation measures*

Common cost-insensitive evaluation measures such as misclassification rate or $F - Score$, assume the same cost for the different misclassification errors. Using these measures is not suitable for example-dependent cost-sensitive binary classification problems. Indeed, two classifiers with equal misclassification rate but different numbers of false positives and false negatives do not have the same impact on cost since $C_{FP_i} \neq C_{FN_i}$; therefore there is a need for a measure that takes into account the actual costs $C_i = [C_{TP_i}, C_{FP_i}, C_{FN_i}, C_{TN_i}]$ of each example $i$, as introduced in the previous section.

Let $S$ be a set of $N$ examples $i$, $N = |S|$, where each example is represented by the augmented feature vector $X_i^a = [X_i, C_i]$ and labelled using the class label $y_i \in \{0, 1\}$. A classifier $f$ which generates the predicted label $c_i$

for each element $i$ is trained using the set $S$. Then the cost of using $f$ on $S$ is calculated by

$$Cost(f(S)) = \sum_{i=1}^{N} \Big( y_i(c_i C_{TP_i} + (1 - c_i) C_{FN_i}) + \tag{2}$$

$$(1 - y_i)(c_i C_{FP_i} + (1 - c_i) C_{TN_i}) \Big). \tag{3}$$

Moreover, by evaluating the cost of classifying all examples as the class with the lowest cost $Cost_l(S) = \min\{Cost(f_0(S)), Cost(f_1(S))\}$ where $f_0$ refers to a classifier that predicts all the examples in $S$ as belonging to the class $c_0$, and similarly $f_1$ predicts all the examples in $S$ as belonging to the class $c_1$, the cost improvement can be expressed as the cost savings as compared with $Cost_l(S)$.

$$Savings(f(S)) = \frac{Cost_l(S) - Cost(f(S))}{Cost_l(S)}. \tag{4}$$

*2.3. State-of-the-art example-dependent cost-sensitive methods*

As mentioned earlier, taking into account the different costs associated with each example, some methods have been proposed to make classifiers example-dependent cost-sensitive. These methods may be grouped in two categories. Methods based on changing the class distribution of the training data, which are known as cost-proportionate sampling methods; and direct cost methods (Wang, 2013).

A standard method to introduce example-dependent costs into classification algorithms is to re-weight the training examples based on their costs, either by cost-proportionate rejection-sampling (Zadrozny et al., 2003), or over-sampling (Elkan, 2001). The rejection-sampling approach consists in selecting a random subset $S_r$ by randomly selecting examples from $S$, and

accepting each example $i$ with probability $w_i / \max_{1,\ldots,N}\{w_i\}$, where $w_i$ is defined as the expected misclassification error of example $i$:

$$w_i = y_i \cdot C_{FN_i} + (1 - y_i) \cdot C_{FP_i}. \quad (5)$$

Lastly, the over-sampling method consists in creating a new set $S_o$, by making $w_i$ copies of each example $i$. However, cost-proportionate over-sampling increases the training since $|S_o| >> |S|$, and it also may result in over-fitting (Drummond and Holte, 2003). Furthermore, none of these methods uses the full cost matrix but only the misclassification costs.

In a recent paper, we have proposed an example-dependent cost-sensitive Bayes minimum risk (BMR) for credit card fraud detection (Correa Bahnsen et al., 2014c). The BMR classifier is a decision model based on quantifying tradeoffs between various decisions using probabilities and the costs that accompany such decisions (Jayanta K. et al., 2006). This is done in a way that for each example the expected losses are minimized. In what follows, we consider the probability estimates $p_i$ as known, regardless of the algorithm used to calculate them.

The risk that accompanies each decision is calculated. In the specific framework of binary classification, the risk of predicting the example $i$ as negative is $R(c_i = 0|X_i) = C_{TN_i}(1 - \hat{p}_i) + C_{FN_i} \cdot \hat{p}_i$, and $R(c_i = 1|X_i) = C_{TP_i} \cdot \hat{p}_i + C_{FP_i}(1 - \hat{p}_i)$, is the risk when predicting the example as positive, where $\hat{p}_i$ is the estimated positive probability for example $i$. Subsequently, if $R(c_i = 0|X_i) \leq R(c_i = 1|X_i)$, then the example $i$ is classified as negative. This means that the risk associated with the decision $c_i$ is lower than the risk associated with classifying it as positive. However, when using the output of a binary classifier as a basis for decision making, there is a need for a probability

that not only separates well between positive and negative examples, but that also assesses the real probability of the event (Cohen and Goldszmidt, 2004).

## 3. Decision trees

Decision trees are one of the most widely used machine learning algorithms Maimon (2008). The technique is considered to be white box, in the sense that is easy to interpret, and has a very low computational cost, while maintaining a good performance as compared with more complex techniques Hastie et al. (2009). There are two types of decision tree depending on the objective of the model. They work either for classification or regression. In this section we focus on binary classification decision tree.

### *3.1. Construction of classification trees*

Classification trees is one of the most common types of decision tree, in which the objective is to find the $Tree$ that best discriminates between classes. In general the decision tree represents a set of splitting rules organized in levels in a flowchart structure. In the $Tree$, each rule is shown as a node, and it is represented as $(X^j, l^j)$, meaning that the set $S$ is split in two sets $S^l$ and $S^r$ according to $X^j$ and $l^j$:

$$S^l = \{X_i^a | X_i^a \in S \wedge x_i^j \leq l^j\} \quad \text{and} \quad S^r = \{X_i^a | X_i^a \in S \wedge x_i^j > l^j\}, \tag{6}$$

where $X^j$ is the $j^{th}$ feature represented in the vector $X^j = [x_1^j, x_2^j, ..., x_N^j]$ and $l^j$ is a value such that $min(X^j) \leq l^j < max(X^j)$.

The $Tree$ is constructed by testing all possible $l^j$ for each $X^j$, and picking the rule $(X^j, l^j)$ that maximizes a specific splitting criteria. Then the training data is split according to the best rule, and for each new subset the procedure

is repeated, until one of the stopping criteria is met. Afterwards, taking into account the number of positive examples in each set $S_1 = \{X_i^a | X_i^a \in S \wedge y_i = 1\}$, the percentage of positives $\pi_1 = |S_1|/|S|$ of each set is used to calculate the impurity of each leaf using either the entropy $I_e(\pi_1) = -\pi_1 \log \pi_1 - (1 - \pi_1) \log(1 - \pi_1)$ or the Gini $I_g(\pi_1) = 2\pi_1(1 - \pi_1)$ measures. Finally the gain of the splitting criteria using the rule $(X^j, l^j)$ is calculated as the impurity of $S$ minus the weighted impurity of each leaf:

$$Gain(X^j, l^j) = I(\pi_1) - \frac{|S^l|}{|S|} I(\pi_1^l) - \frac{|S^r|}{|S|} I(\pi_1^r), \tag{7}$$

where $I(\pi_1)$ can be either of the impurity measures $I_e(\pi_1)$ or $I_g(\pi_1)$.

Subsequently, the gain of all possible splitting rules is calculated. The rule with maximal gain is selected

$$(best_x, best_l) = \arg\max_{(X^j, l^j)} Gain(X^j, l^j), \tag{8}$$

and the set $S$ is split into $S^l$ and $S^r$ according to that rule. Furthermore, the process is iteratively repeated for each subset until either there is no more possible splits or a stopping criteria is met.

### *3.2. Pruning of a classification tree*

After a decision tree has been fully grown, there is a risk for the algorithm to over fit the training data. In order to solve this, pruning techniques have been proposed in Breiman et al. (1984). The overall objective of pruning is to eliminate branches that are not contributing to the generalization accuracy of the tree Rokach and Maimon (2010).

In general, pruning techniques start from a fully grown tree, and recursively check if by eliminating a node there is an improvement in the error or

misclassification rate $\epsilon$ of the $Tree$. The most common pruning technique is cost-complexity pruning, initially proposed by Breiman Breiman et al. (1984). This method evaluates iteratively if the removal of a node improves the error rate $\epsilon$ of a $Tree$ in the set $S$, weighted by the difference of the number of nodes.

$$PC_{cc} = \frac{\epsilon(EB(Tree, node), S) - \epsilon(Tree, S)}{|Tree| - |EB(Tree, node)|}, \tag{9}$$

where $EB(Tree, node)$ is an auxiliary function that removes $node$ from $Tree$ and returns a new $Tree$. At each iteration, the current $Tree$ is compared against all possible nodes.

## 4. Example-Dependent Cost-Sensitive Decision Trees

Standard decision tree algorithms focus on inducing trees that maximize accuracy. However this is not optimal when the misclassification costs are unequal Elkan (2001).This has led to many studies that develop algorithms that aim to introduce the cost-sensitivity into the algorithms Lomax and Vadera (2013). These studies have focused on introducing the class-dependent costs Draper et al. (1994); Ting (2002); Ling et al. (2004); Li et al. (2005); Kretowski and Grześ (2006); Vadera (2010), which is not optimal for some applications. For example in credit card fraud detection, it is true that false positives have a different cost than false negatives, nevertheless, false negatives may vary significantly, which makes class-dependent cost-sensitive methods not suitable for this problem.

In this section, we first propose a new method to introduce the costs into the decision tree induction stage, by creating new-cost based impurity mea-

sures. Afterwards, we propose a new pruning method based on minimizing the cost as pruning criteria.

### *4.1. Cost-sensitive impurity measures*

Standard impurity measures such as misclassification, entropy or Gini, take into account the distribution of classes of each leaf to evaluate the predictive power of a splitting rule, leading to an impurity measure that is based on minimizing the misclassification rate. However, as has been previously shown Correa Bahnsen et al. (2013), minimizing misclassification does not lead to the same results than minimizing cost. Instead, we are interested in measuring how good is a splitting rule in terms of cost not only accuracy. For doing that, we propose a new example-dependent cost based impurity measure that takes into account the cost matrix of each example.

We define a new cost-based impurity measure taking into account the costs when all the examples in a leaf are classified both as negative using $f_0$ and positive using $f_1$

$$I_c(S) = \min\Big\{Cost(f_0(S)), Cost(f_1(S))\Big\}. \tag{10}$$

The objective of this measure is to evaluate the lowest expected cost of a splitting rule. Following the same logic, the classification of each set is calculated as the prediction that leads to the lowest cost

$$f(S) = \begin{cases} 0 & \text{if } \; Cost(f_0(S)) \leq Cost(f_1(S)) \\ 1 & \text{otherwise} \end{cases} \tag{11}$$

Finally, using the cost-based impurity, the splitting criteria cost based gain of using the splitting rule $(X^j, l^j)$ is calculated with (7).

### *4.2. Cost-sensitive pruning*

Most of the literature in class-dependent cost-sensitive decision tree focuses on using the misclassification costs during the construction of the algorithms Lomax and Vadera (2013). Only few algorithms such as AUCSplit Ferri et al. (2002) have included the costs both during and after the construction of the tree. However, this approach only used the class-dependent costs, and not the example-dependent costs.

We propose a new example-dependent cost-based impurity measure, by replacing the error rate $\epsilon$ in (9) with the cost of using the $Tree$ on $S$ i.e. by replacing with $Cost(f(S))$.

$$PC_c = \frac{Cost(f(S)) - Cost(f^*(S))}{|Tree| - |EB(Tree, node)|}, \tag{12}$$

where $f^*$ is the classifier of the tree without the selected node $EB(Tree, node)$.

Using the new pruning criteria, nodes of the tree that do not contribute to the minimization of the cost will be pruned, regardless of the impact of those nodes on the accuracy of the algorithm. This follows the same logic as in the proposed cost-based impurity measure, since minimizing the misclassification is different than minimizing the cost, and in several real-world applications the objectives align with the cost not with the misclassification error.

## 5. Experimental setup

In this section we present the datasets used to evaluate the example-dependent cost-sensitive decision tree algorithm $CSDT$ proposed in the Section 4. We used datasets from three different real world example-dependent cost-sensitive problems: Credit scoring, direct marketing and credit card

| Database | Set | Observations | %Positives | Cost |
|---|---|---|---|---|
| Credit Scoring | Total | 112,915 | 6.74 | 83,740,181 |
| | Training | 45,264 | 6.75 | 33,360,130 |
| | Under-sampled | 6,038 | 50.58 | 33,360,130 |
| | Rejection-sampled | 5,271 | 43.81 | 29,009,564 |
| | Over-sampled | 66,123 | 36.16 | 296,515,655 |
| | Validation | 33,919 | 6.68 | 24,786,997 |
| | Testing | 33,732 | 6.81 | 25,593,055 |
| Direct Marketing | Total | 37,931 | 12.62 | 59,507 |
| | Training | 15,346 | 12.55 | 24,304 |
| | Under-sampled | 3,806 | 50.60 | 24,304 |
| | Rejection-sampled | 1,644 | 52.43 | 20,621 |
| | Over-sampled | 22,625 | 40.69 | 207,978 |
| | Validation | 11,354 | 12.30 | 16,154 |
| | Testing | 11,231 | 13.04 | 19,048 |
| Credit Card | Total | 236,735 | 1.50 | 895,154 |
| Fraud Detection | Training | 94,599 | 1.51 | 358,078 |
| | Under-sampled | 2,828 | 50.42 | 358,078 |
| | Rejection-sampled | 94,522 | 1.43 | 357,927 |
| | Over-sampled | 189,115 | 1.46 | 716,006 |
| | Validation | 70,910 | 1.53 | 274,910 |
| | Testing | 71,226 | 1.45 | 262,167 |

Table 3: Summary of the datasets

fraud detection. For each dataset we define a cost matrix, from which the algorithms are trained. Additionally, we perform an under-sampling, cost-proportionate rejection-sampling and cost-proportionate over-sampling procedures. In Table 3, information about the different datasets is shown.

### *5.1. Credit scoring*

Credit scoring is a real-world problem in which the real costs due to misclassification are not constant, but are example-dependent. The objective in credit scoring is to classify which potential customers are likely to default

a contracted financial obligation based on the customer's past financial experience, and with that information decide whether to approve or decline a loan Anderson (2007). This tool has become a standard practice among financial institutions around the world in order to predict and control their loan portfolios. When constructing credit scores, it is a common practice to use standard cost-insensitive binary classification algorithms such as logistic regression, neural networks, discriminant analysis, genetic programing, decision tree, among others Correa Bahnsen and Gonzalez Montoya (2011); Hand and Henley (1997); Ong et al. (2005); Yeh and Lien (2009). However, in practice, the cost associated with approving a bad customer is quite different from the cost associated with declining a good customer. Furthermore, the costs are not constant among customers. This is because loans have different credit line amounts, terms, and even interest rates.

| | Actual Positive $y_i = 1$ | Actual Negative $y_i = 0$ |
|---|---|---|
| Predicted Positive $c_i = 1$ | $C_{TP_i} = 0$ | $C_{FP_i} = r_i + C^a_{FP}$ |
| Predicted Negative $c_i = 0$ | $C_{FN_i} = Cl_i \cdot L_{gd}$ | $C_{TN_i} = 0$ |

Table 4: Credit scoring example-dependent cost matrix

For this paper we follow the example-dependent cost-sensitive approach for credit scoring proposed in (Correa Bahnsen et al., 2014b). In Table 4, the credit scoring cost matrix is shown. First, the costs of a correct classification, $C_{TP_i}$ and $C_{TN_i}$, are zero for all customers, $i$. Then, $C_{FN_i}$ are the losses if the customer $i$ defaults, which is calculated as the credit line $Cl_i$ time the loss given default $L_{gd}$. The cost of a false positive per customer $C_{FP_i}$ is

defined as the sum of two real financial costs $r_i$ and $C^a_{FP}$, where $r_i$ is the loss in profit by rejecting what would have been a good customer. The second term $C^a_{FP}$, is related to the assumption that the financial institution will not keep the money of the declined customer idle it will instead give a loan to an alternative customer (Nayak and Turvey, 1997). Since no further information is known about the alternative customer, it is assumed to have an average credit line $\overline{Cl}$ and an average profit $\overline{r}$. Then, $C^a_{FP} = -\overline{r} \cdot \pi_0 + \overline{Cl} \cdot L_{gd} \cdot \pi_1$, in other words minus the profit of an average alternative customer plus the expected loss, taking into account that the alternative customer will pay his debt with a probability equal to the prior negative rate, and similarly will default with probability equal to the prior positive rate.

We apply the previous framework to a publicly available credit scoring dataset. The dataset is the 2011 Kaggle competition Give Me Some Credit[2], in which the objective is to identify those customers of personal loans that will experience financial distress in the next two years. The Kaggle Credit datasets contain information regarding the features, and more importantly about the income of each example, from which an estimated credit limit $Cl_i$ can be calculated (see (Correa Bahnsen et al., 2014a)).

The dataset contains 112,915 examples, each one with 10 features and the class label. The proportion of default or positive examples is 6.74%. Since no specific information regarding the datasets is provided, we assume that they belong to average European financial institution. This enabled us to find the different parameters needed to calculate the cost matrix. In particular we

[2]http://www.kaggle.com/c/GiveMeSomeCredit/

| | Actual Positive $y_i = 1$ | Actual Negative $y_i = 0$ |
|---|---|---|
| Predicted Positive $c_i = 1$ | $C_{TP_i} = C_a$ | $C_{FP_i} = C_a$ |
| Predicted Negative $c_i = 0$ | $C_{FN_i} = Int_i$ | $C_{TN_i} = 0$ |

Table 5: Direct marketing example-dependent cost matrix

used the same parameters as in (Correa Bahnsen et al., 2014a), the interest rate $int_r$ to 4.79%, the cost of funds $int_{cf}$ to 2.94%, the term $l$ to 24 months, and the loss given default $L_{gd}$ to 75%.

### *5.2. Direct marketing*

In direct marketing the objective is to classify those customers who are more likely to have a certain response to a marketing campaign (Ngai et al., 2009). We used a direct marketing dataset (Moro et al., 2011) available on the UCI machine learning repository (Bache and Lichman, 2013). The dataset contains 45,000 clients of a Portuguese bank who were contacted by phone between March 2008 and October 2010 and received an offer to open a long-term deposit account with attractive interest rates. The dataset contains features such as age, job, marital status, education, average yearly balance and current loan status and the label indicating whether or not the client accepted the offer.

This problem is example-dependent cost sensitive, since there are different costs of false positives and false negatives. Specifically, in direct marketing, false positives have the cost of contacting the client, and false negatives have the cost due to the loss of income by failing to contact a client that otherwise would have opened a long-term deposit.

We used the direct marketing example-dependent cost matrix proposed in (Correa Bahnsen et al., 2014c). The cost matrix is shown in Table 5, where $C_a$ is the administrative cost of contacting the client, as is credit card fraud, and $Int_i$ is the expected income when a client opens a long-term deposit. This last term is defined as the long-term deposit amount times the interest rate spread.

In order to estimate $Int_i$, first the long-term deposit amount is assumed to be a 20% of the average yearly balance, and lastly, the interest rate spread is estimated to be 2.463333%, which is the average between 2008 and 2010 of the retail banking sector in Portugal as reported by the Portuguese central bank. Given that, the $Int_i$ is equal to $(balance * 20\%) * 2.463333\%$.

### *5.3. Credit card fraud detection*

A credit card fraud detection algorithm, consisting on identifying those transactions with a high probability of being fraud, based on historical customers consumer and fraud patterns. Different detection systems that are based on machine learning techniques have been successfully used for this problem, in particular: neural networks (Maes et al., 2002), Bayesian learning (Maes et al., 2002), hybrid models (Krivko, 2010), support vector machines (Bhattacharyya et al., 2011) and random forest (Correa Bahnsen et al., 2013).

Credit card fraud detection is by definition a cost sensitive problem, since the cost of failing to detect a fraud is significantly different from the one when a false alert is made (Elkan, 2001). We used the fraud detection example-dependent cost matrix proposed in (Correa Bahnsen et al., 2013). In Table 6, the cost matrix is presented. Where $Amt_i$ is the amount of transaction $i$, and

$C_a$ is the administrative cost of investigating a fraud alert. This cost matrix differentiates between the costs of the different outcomes of the classification algorithm, meaning that it differentiates between false positives and false negatives, and also the different costs of each example.

For this paper we used a dataset provided by a large European card processing company. The dataset consists of fraudulent and legitimate transactions made with credit and debit cards between January 2012 and June 2013. The total dataset contains 120,000,000 individual transactions, each one with 27 attributes, including a fraud label indicating whenever a transaction is identified as fraud. This label was created internally in the card processing company, and can be regarded as highly accurate. In the dataset only 40,000 transactions were labeled as fraud, leading to a fraud ratio of 0.025%.

From the initial attributes, an additional 260 attributes are derived using the methodology proposed in (Bhattacharyya et al., 2011; Whitrow et al., 2008; Correa Bahnsen et al., 2013). The idea behind the derived attributes consists in using a transaction aggregation strategy in order to capture consumer spending behavior in the recent past. The derivation of the attributes consists in grouping the transactions made during the last given number of

| | Actual Positive $y_i = 1$ | Actual Negative $y_i = 0$ |
|---|---|---|
| Predicted Positive $c_i = 1$ | $C_{TP_i} = C_a$ | $C_{FP_i} = C_a$ |
| Predicted Negative $c_i = 0$ | $C_{FN_i} = Amt_i$ | $C_{TN_i} = 0$ |

Table 6: Credit card fraud detection example-dependent cost matrix

hours, first by card or account number, then by transaction type, merchant group, country or other, followed by calculating the number of transactions or the total amount spent on those transactions.

For the experiments, a smaller subset of transactions with a higher fraud ratio, corresponding to a specific group of transactions, is selected. This dataset contains 236,735 transactions and a fraud ratio of 1.50%. In this dataset, the total financial losses due to fraud are 895,154 Euros. This dataset was selected because it is the one where most frauds are being made.

## 6. Results

In this section we present the experimental results. First, we evaluate the performance of the proposed $CSDT$ algorithm and compare it against a classical decision tree ($DT$). We evaluate the different trees using them without pruning (*notp*), with error based pruning (*errp*), and also with the proposed cost-sensitive pruning technique (*costp*). The different algorithms are trained using the training ($t$), under-sampling ($u$), cost-proportionate rejection-sampling ($r$), and cost-proportionate over-sampling ($o$) datasets. Lastly, we compare our proposed method versus state-of-the-art example-dependent cost-sensitive techniques.

### *6.1. Results CSDT*

We evaluate a decision tree constructed using the Gini impurity measure, with and without the pruning defined in (9). We also apply the cost-based pruning procedure given in (12). Lastly, we compared against the proposed $CSDT$ constructed using the cost-based impurity measure defined in (10), using the two pruning procedures.

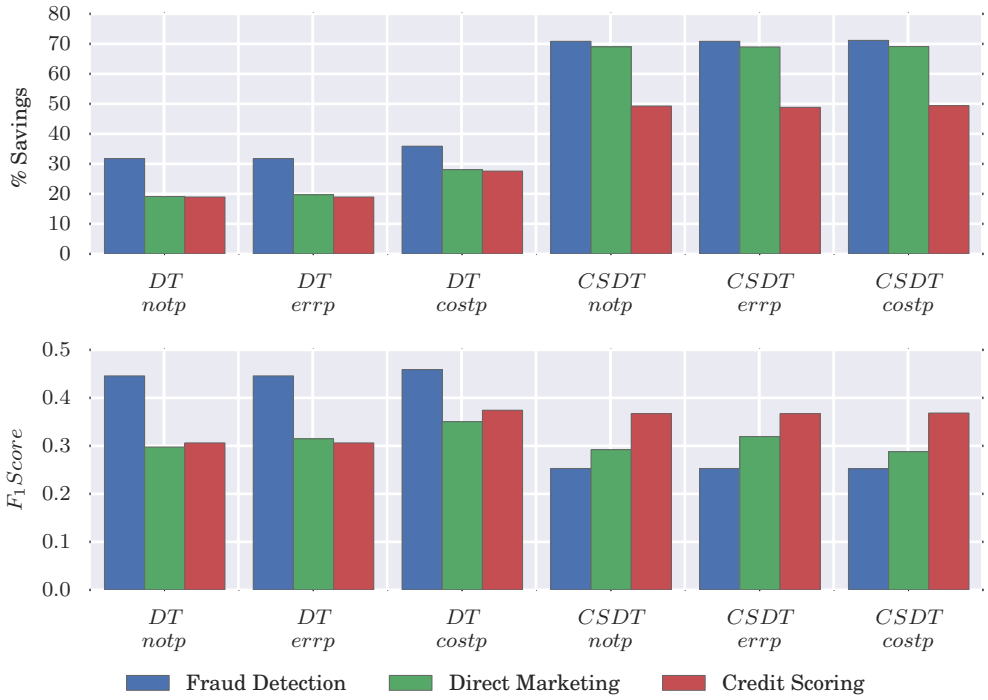



Figure 2: Results of the $DT$ and the $CSDT$. For both algorithms, the results are calculated with and without both types of pruning criteria. There is a clear difference between the savings of the $DT$ and the $CSDT$ algorithms. However, this difference is not observable on the $F_1Score$ results. Since the $CSDT$ is focused on maximizing the savings not the accuracy or $F_1Score$. There is a small increase in savings when using the $DT$ with cost-sensitive pruning. Nevertheless, in the case of the $CSDT$ algorithm, there is no change when using any pruning procedure, neither in savings or $F_1Score$.

In Figure 2, the results using the three databases are shown. In particular we first evaluate the impact of the algorithms when trained using the training set. There is a clear difference between the savings of the $DT$ and the $CSDT$ algorithms. However, that difference is not observable on the $F_1Score$ results. Since the $CSDT$ is focused on maximizing the savings not the accuracy or $F_1Score$. There is a small increase in savings when using the $DT$ with cost-

| | | Fraud Detection | | | Direct Marketing | | | Credit Scoring | | |
|---|---|---|---|---|---|---|---|---|---|---|
| set | Algorithm | $\%Sav$ | $\%Accu$ | $F_1Score$ | $\%Sav$ | $\%Accu$ | $F_1Score$ | $\%Sav$ | $\%Accu$ | $F_1Score$ |
| t | $DT_{notp}$ | 31.76 | 98.76 | 0.4458 | 19.11 | 88.24 | 0.2976 | 18.95 | 93.42 | 0.3062 |
| | $DT_{errp}$ | 31.76 | 98.76 | 0.4458 | 19.70 | 88.28 | 0.3147 | 18.95 | 93.42 | 0.3062 |
| | $DT_{costp}$ | 35.89 | 98.71 | 0.4590 | 28.08 | 88.28 | 0.3503 | 27.59 | 93.41 | 0.3743 |
| | $CSDT_{notp}$ | 70.85 | 95.07 | 0.2529 | 69.00 | 85.51 | 0.2920 | 49.28 | 93.19 | 0.3669 |
| | $CSDT_{errp}$ | 70.85 | 95.07 | 0.2529 | 68.97 | 88.18 | 0.3193 | 48.85 | 93.19 | 0.3669 |
| | $CSDT_{costp}$ | 71.16 | 94.98 | 0.2522 | 69.10 | 81.75 | 0.2878 | 49.39 | 90.28 | 0.3684 |
| u | $DT_{notp}$ | 52.39 | 85.52 | 0.1502 | 49.80 | 70.80 | 0.3374 | 48.91 | 75.96 | 0.2983 |
| | $DT_{errp}$ | 52.39 | 85.52 | 0.1502 | 49.80 | 70.80 | 0.3374 | 48.91 | 75.96 | 0.2983 |
| | $DT_{costp}$ | 70.26 | 92.67 | 0.2333 | 53.20 | 74.51 | 0.3565 | 49.77 | 79.37 | 0.3286 |
| | $CSDT_{notp}$ | 12.46 | 69.34 | 0.0761 | 64.21 | 52.34 | 0.2830 | 30.68 | 93.19 | 0.2061 |
| | $CSDT_{errp}$ | 14.98 | 70.31 | 0.0741 | 66.21 | 60.06 | 0.2822 | 41.49 | 93.19 | 0.2564 |
| | $CSDT_{costp}$ | 15.01 | 70.31 | 0.0743 | 68.07 | 62.11 | 0.2649 | 44.89 | 78.08 | 0.2881 |
| r | $DT_{notp}$ | 34.39 | 98.70 | 0.4321 | 68.59 | 72.73 | 0.3135 | 48.97 | 87.07 | 0.3931 |
| | $DT_{errp}$ | 34.39 | 98.70 | 0.4321 | 68.79 | 73.39 | 0.3196 | 48.97 | 87.07 | 0.3931 |
| | $DT_{costp}$ | 38.99 | 98.64 | 0.4478 | 69.27 | 72.58 | 0.3274 | 50.48 | 82.69 | 0.3501 |
| | $CSDT_{notp}$ | 70.85 | 95.07 | 0.2529 | 66.87 | 57.91 | 0.2761 | 31.25 | 93.19 | 0.1940 |
| | $CSDT_{errp}$ | 70.85 | 95.07 | 0.2529 | 67.47 | 63.31 | 0.2581 | 40.69 | 93.19 | 0.2529 |
| | $CSDT_{costp}$ | 71.09 | 94.94 | 0.2515 | 68.08 | 62.60 | 0.2642 | 44.51 | 77.82 | 0.2869 |
| o | $DT_{notp}$ | 31.72 | 98.77 | 0.4495 | 60.30 | 79.46 | 0.3674 | 47.39 | 88.28 | 0.3994 |
| | $DT_{errp}$ | 31.72 | 98.77 | 0.4495 | 60.30 | 79.46 | 0.3674 | 47.39 | 88.28 | 0.3994 |
| | $DT_{costp}$ | 37.35 | 98.68 | 0.4575 | 69.44 | 70.07 | 0.3108 | 50.92 | 87.52 | 0.3977 |
| | $CSDT_{notp}$ | 70.84 | 95.06 | 0.2529 | 68.75 | 64.72 | 0.2935 | 41.37 | 93.19 | 0.2205 |
| | $CSDT_{errp}$ | 70.84 | 95.06 | 0.2529 | 68.75 | 64.72 | 0.2935 | 41.38 | 90.63 | 0.3457 |
| | $CSDT_{costp}$ | 71.09 | 94.94 | 0.2515 | 68.75 | 64.72 | 0.2935 | 41.65 | 78.26 | 0.2896 |

Table 7: Results on the three datasets of the cost-sensitive and standard decision tree, without pruning (*notp*), with error based pruning (*errp*), and with cost-sensitive pruning technique (*costp*). Estimated using the different training sets: training, under-sampling, cost-proportionate rejection-sampling and cost-proportionate over-sampling

sensitive pruning. Nevertheless, in the case of the $CSDT$ algorithm, there is no change when using any pruning procedure, neither in savings or $F_1Score$.

In addition, we also evaluate the algorithms on the different sets, under-

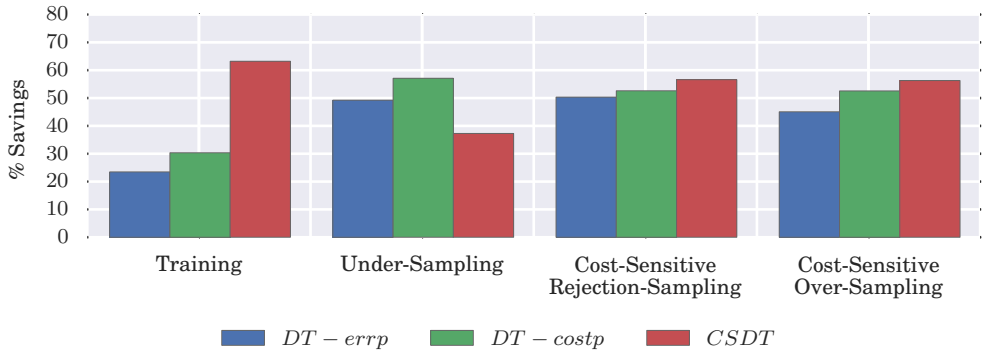



Figure 3: Average savings on the three datasets of the different cost-sensitive and standard decision tree, estimated using the different training sets: training, under-sampling, cost-proportionate rejection-sampling and cost-proportionate over-sampling. The best results are found when using the training set. When using the under-sampling set there is a decrease in savings of the algorithm. Lastly, in the case of the cost-proportionate sampling sets, there is a small increase in savings when using the $CSDT$ algorithm.

sampling, rejection-sampling and over-sampling. The results are shown in Table 7. Moreover, in Figure 3, the average results of the different algorithms measured by savings is shown. The best results are found when using the training set. When using the under-sampling set there is a decrease in savings of the $CSDT$ algorithm. Lastly, in the case of the cost-proportionate sampling sets, there is a small increase in savings when using the $CSDT$ algorithm.

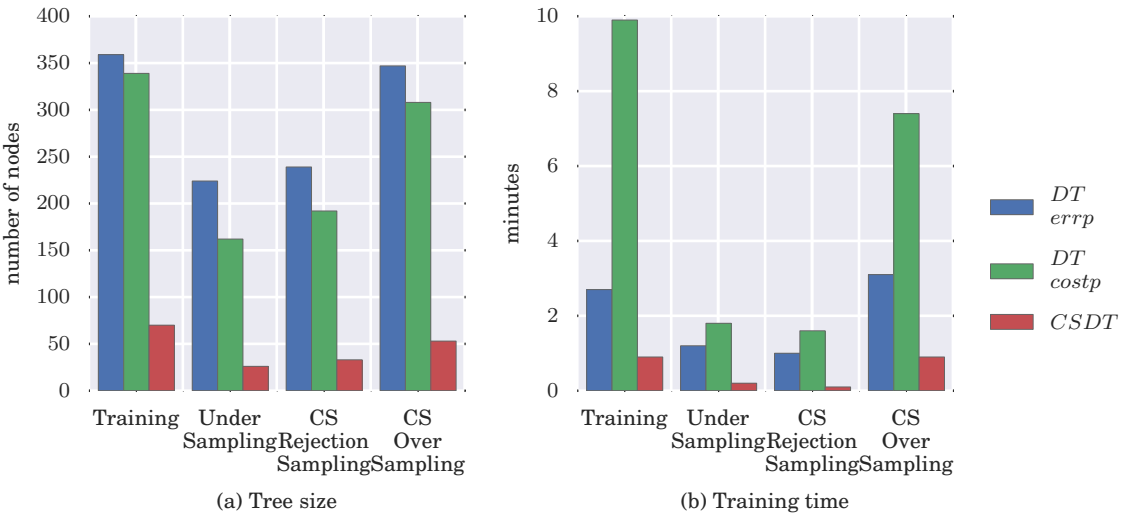
Finally, we also analyze the different models taking into account the complexity and the training time. In particular we evaluate the size of each $Tree$. In Table 8, and Figure 4, the results are shown. The $CSDT$ algorithm creates significantly smaller trees, which leads to a lower training time. In particular this is a result of using the non weighted gain, the $CSDT$ only accepts splitting rules that contribute to the overall reduction of the cost, which is

| | | Fraud Detection | | Direct Marketing | | Credit Scoring | |
|---|---|---|---|---|---|---|---|
| set | Algorithm | $\|Tree\|$ | Time | $\|Tree\|$ | Time | $\|Tree\|$ | Time |
| t | $DT_{notp}$ | 488 | 2.45 | 298 | 1.58 | 292 | 1.58 |
| | $DT_{errp}$ | 488 | 3.90 | 298 | 2.13 | 292 | 2.13 |
| | $DT_{costp}$ | 446 | 19.19 | 291 | 5.23 | 280 | 5.23 |
| | $CSDT_{notp}$ | 89 | 1.47 | 51 | 0.40 | 69 | 0.40 |
| | $CSDT_{errp}$ | 88 | 1.87 | 51 | 0.64 | 69 | 0.64 |
| | $CSDT_{costp}$ | 89 | 1.74 | 51 | 0.45 | 69 | 0.45 |
| u | $DT_{notp}$ | 308 | 1.10 | 198 | 1.00 | 167 | 1.00 |
| | $DT_{errp}$ | 308 | 1.43 | 198 | 1.14 | 167 | 1.14 |
| | $DT_{costp}$ | 153 | 2.59 | 190 | 1.34 | 142 | 1.34 |
| | $CSDT_{notp}$ | 14 | 0.20 | 23 | 0.17 | 42 | 0.17 |
| | $CSDT_{errp}$ | 14 | 0.23 | 23 | 0.19 | 42 | 0.19 |
| | $CSDT_{costp}$ | 14 | 0.24 | 23 | 0.18 | 42 | 0.18 |
| r | $DT_{notp}$ | 268 | 0.98 | 181 | 0.90 | 267 | 0.90 |
| | $DT_{errp}$ | 268 | 1.24 | 181 | 0.95 | 267 | 0.95 |
| | $DT_{costp}$ | 153 | 2.48 | 162 | 1.20 | 261 | 1.20 |
| | $CSDT_{notp}$ | 18 | 0.22 | 10 | 0.07 | 70 | 0.07 |
| | $CSDT_{errp}$ | 18 | 0.23 | 10 | 0.07 | 70 | 0.07 |
| | $CSDT_{costp}$ | 18 | 0.23 | 10 | 0.07 | 70 | 0.07 |
| o | $DT_{notp}$ | 425 | 2.30 | 340 | 1.80 | 277 | 1.80 |
| | $DT_{errp}$ | 425 | 3.98 | 340 | 2.65 | 277 | 2.65 |
| | $DT_{costp}$ | 364 | 10.15 | 288 | 5.99 | 273 | 5.99 |
| | $CSDT_{notp}$ | 37 | 1.58 | 51 | 0.38 | 70 | 0.38 |
| | $CSDT_{errp}$ | 37 | 1.90 | 51 | 0.45 | 70 | 0.45 |
| | $CSDT_{costp}$ | 37 | 1.98 | 51 | 0.42 | 70 | 0.42 |

Table 8: Training time and tree size of the different cost-sensitive and standard decision tree, estimated using the different training sets: training, under-sampling, cost-proportionate rejection-sampling and cost-proportionate over-sampling, for the three databases.

not the case if instead the weighted gain was used. Even that the $DT$ with cost pruning, produce a good result measured by savings, it is the one that takes the longer to estimate. Since the algorithm first creates a big decision tree using the *Gini* impurity, and then attempt to find a smaller tree taking

into account the cost. Measured by training time, the $CSDT$ is by all means faster to train than the $DT$ algorithm, leading to an algorithm that not only gives better results measured by savings but also one that can be trained much quicker than the standard $DT$.




(a) Tree size

(b) Training time

Figure 4: Average tree size (a) and training time (b), of the different cost-sensitive and standard decision tree, estimated using the different training sets: training, under-sampling, cost-proportionate rejection-sampling and cost-proportionate over-sampling, for the three databases. The $CSDT$ algorithm create significantly smaller trees, which leads to a lower training time.

### *6.2. Comparison with state-of-the-art methods*

Additionally to the comparison of the $CSDT$ and a $DT$, we also evaluate and compare our proposed method with the standard example-dependent cost-sensitive methods, namely, cost-proportionate rejection-sampling (Zadrozny et al., 2003), cost-proportionate over-sampling (Elkan, 2001) and Bayes minimum risk ($BMR$) (Correa Bahnsen et al., 2014c).

| | | Fraud Detection | | | Direct Marketing | | | Credit Scoring | | |
|---|---|---|---|---|---|---|---|---|---|---|
| set | Algorithm | $\%Sav$ | $\%Accu$ | $F_1Score$ | $\%Sav$ | $\%Accu$ | $F_1Score$ | $\%Sav$ | $\%Accu$ | $F_1Score$ |
| t | $DT$ | 31.76 | 98.76 | 0.4458 | 19.70 | 88.28 | 0.3147 | 18.95 | 93.42 | 0.3062 |
| | $DT-BMR$ | 60.45 | 65.05 | 0.2139 | 69.27 | 63.09 | 0.2416 | 33.25 | 79.06 | 0.2450 |
| | $LR$ | 0.92 | 99.75 | 0.1531 | 14.99 | 88.25 | 0.2462 | 3.28 | 93.47 | 0.0811 |
| | $LR-BMR$ | 45.52 | 66.42 | 0.1384 | 68.46 | 70.73 | 0.2470 | 29.55 | 80.86 | 0.2883 |
| | $RF$ | 33.42 | 76.33 | 0.2061 | 20.10 | 86.94 | 0.2671 | 15.38 | 93.57 | 0.2720 |
| | $RF-BMR$ | 64.14 | 62.85 | 0.2052 | 67.30 | 69.07 | 0.3262 | 47.89 | 81.54 | 0.2698 |
| u | $DT$ | 52.39 | 85.52 | 0.1502 | 49.80 | 70.80 | 0.3374 | 48.91 | 75.96 | 0.2983 |
| | $LR$ | 12.43 | 73.08 | 0.0241 | 49.60 | 73.32 | 0.3396 | 45.38 | 85.40 | 0.3618 |
| | $RF$ | 56.84 | 54.48 | 0.0359 | 41.64 | 67.14 | 0.3069 | 49.53 | 79.02 | 0.3215 |
| r | $DT$ | 34.39 | 98.70 | 0.4321 | 68.79 | 73.39 | 0.3196 | 48.97 | 87.07 | 0.3931 |
| | $LR$ | 30.77 | 76.02 | 0.1846 | 62.85 | 72.63 | 0.3313 | 48.80 | 84.69 | 0.3660 |
| | $RF$ | 38.12 | 75.03 | 0.2171 | 61.45 | 66.06 | 0.2949 | 47.34 | 81.47 | 0.3284 |
| o | $DT$ | 31.72 | 98.77 | 0.4495 | 60.30 | 79.46 | 0.3674 | 47.39 | 88.28 | 0.3994 |
| | $LR$ | 27.93 | 76.79 | 0.1776 | 55.63 | 81.65 | 0.3540 | 34.05 | 91.55 | 0.3923 |
| | $RF$ | 36.12 | 75.62 | 0.2129 | 22.80 | 85.52 | 0.2871 | 21.72 | 93.22 | 0.3301 |

Table 9: Results on the three datasets of the decision tree, logistic regression and random forest algorithms, estimated using the different training sets: training, under-sampling, cost-proportionate rejection-sampling and cost-proportionate over-sampling

Using each database and each set, we estimate three different algorithms, in particular a decision tree ($DT$), a logistic regression ($LR$) and a random forest ($RF$). The $LR$ and $RF$ algorithms were trained using the implementations of Scikit-learn Pedregosa et al. (2011), respectively. We only used the $BMR$ algorithm using the training set, as it has been previously shown that it is where the model gives the best results (Correa Bahnsen et al., 2013).

The results are shown on Table 9. Measured by savings, it is observed that regardless of the algorithm used for estimating the positive probabilities, in all cases there is an increase in savings by using BMR . In general, for all datasets the best results are found when using a random forest algorithm

for estimating the positive probabilities. In the case of the direct marketing dataset, the results of the different algorithms are very similar. Nevertheless, in all cases the $BMR$ produce higher savings. When analyzing the $F_1Score$, it is observed that in general there is no increase in results when using the $BMR$. It is observed that the best models selected by savings are not the same as the best ones measured by $F_1Score$. And the reason for that, is because the $F_1Score$ treat the false positives and the false negatives as equal, which as discussed before is not the case in example-dependent cost-sensitive problems.

Finally, we compare the results of the standard algorithms, the algorithms trained using the cost-proportionate sampling sets, the $BMR$, and the $CSDT$. Results are shown in Figure 5. When comparing by savings, for all databases the best model is the $CSDT$, closely follow by the $DT$ with cost-based pruning. It is interesting to see that both algorithms that the algorithms that incorporates the costs during construction, $BMR$ and $CSDT$, gives the best results when trained using the training set. When measured by $F_1Score$, there is not a clear trend regarding the different results. In the case of fraud detection the best model is the $DT$, however measure by savings that model performs poorly. In the case of direct marketing, by $F_1Score$, $DT$ with cost pruning performs the best, but that model is the second worst by savings. In the credit scoring dataset the best model is the same when measured by savings or $F_1Score$.


Figure 5: Comparison of the different models on the three databases. Measured by savings, $CSDT$ is the overall best method. However, by $F_1Score$, there is not a clear trend regarding the different results.

## 7. Conclusions and future work

Several real-world business applications of classification models are example-dependent cost-sensitive, in the sense that the objective of using an algorithm is related to maximizing the profit of the company. Moreover, the different costs due to misclassification vary among examples. In this paper, we focus on three different applications: credit card fraud detection, credit scoring and direct marketing. In all cases, evaluating a classification algorithm using traditional statistics such as misclassification rate or $F_1Score$, do not accurately represent the business oriented goals.

State-of-the-art example-dependent cost-sensitive techniques only introduce the cost to the algorithm, either before or after training, therefore, leaving opportunities to investigate the potential impact of algorithms that take into account the real financial example-dependent costs during an algorithm training. In this paper, we proposed a new example-dependent cost-sensitive decision tree algorithm, by incorporating the different example-dependent costs into a new cost-based impurity measure and a new cost-based pruning criteria. We show the importance of including the costs during the algorithm construction, both by using a classical decision tree and then the cost-based pruning procedure, and by fully creating a decision tree taking into account the costs per example.

Our proposed algorithm maintains the simplicity and interpretability of decision trees, therefore, the resulting tree is easy to analyze and to obtain straightforward explanations of the decisions made by the algorithm. This is a highly desirable feature, since in real-world applications, it is important to explain the rationale behind the models decisions. Moreover, when comparing the results of the proposed method and state-of-the-art algorithms, we found that in all cases our method provides the best performance measured by savings.

Furthermore, when comparing our method with standard decision trees in terms of complexity and training time, our proposed algorithm creates significantly smaller trees in a fraction of the time. This, is an interesting result, as simpler trees are found to perform better when maximizing the savings than when maximizing standard impurity measures. However, this may cause our algorithm to struggle with high cost outliers, as they may be

ignored by the method. Also, individual decision trees typically suffer from high variance.

To overcome the algorithm's limitations, a future research should be focused on evaluating the algorithm in a bagging framework, specifically, by learning different example-dependent cost-sensitive decision trees on random subsets of the training set, and then combining them in order to produce a more robust result. This approach, should take care of most outliers and should arise to better results, as it has been proved to do with standard decision trees, i.e. random forest. Moreover, we are aware that drawing conclusions from only three databases is not ideal. Future work should include focusing efforts on finding more example-dependent cost-sensitive databases. Lastly, it is worth investigating the combination of traditional impurity measures and the proposed cost-sensitive impurity measure, as a measure that takes both the information gain and the savings gain may produce good results.

## References

Anderson, R., 2007. The Credit Scoring Toolkit : Theory and Practice for Retail Credit Risk Management and Decision Automation. Oxford University Press.

Aodha, O. M., Brostow, G. J., Dec. 2013. Revisiting Example Dependent Cost-Sensitive Learning with Decision Trees. In: 2013 IEEE International Conference on Computer Vision. Washington, DC, USA, pp. 193–200.

Bache, K., Lichman, M., 2013. UCI Machine Learning Repository. School of Information and Computer Science, University of California.

Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, J. C., Feb. 2011. Data mining for credit card fraud: A comparative study. Decision Support Systems 50 (3), 602–613.

Breiman, L., Friedman, J., Stone, C. J., Olshen, R., 1984. Classification and Regression Trees. Chapman and Hall/CRC.

Cohen, I., Goldszmidt, M., 2004. Properties and Benefits of Calibrated Classifiers. In: Knowledge Discovery in Databases: PKDD 2004. Vol. 3202. Springer Berlin Heidelberg, pp. 125–136.

Correa Bahnsen, A., Aouada, D., Ottersten, B., 2014a. Example-Dependent Cost-Sensitive Credit Scoring using Bayes Minimum Risk. In: International Conference on Machine Learning and Applications.

Correa Bahnsen, A., Aouada, D., Ottersten, B., 2014b. Example-Dependent Cost-Sensitive Logistic Regression for Credit Scoring. In: 2014 13th International Conference on Machine Learning and Applications. IEEE, Detroit, USA.

Correa Bahnsen, A., Gonzalez Montoya, A., Dec. 2011. Evolutionary Algorithms for Selecting the Architecture of a MLP Neural Network: A Credit Scoring Case. In: IEEE 11th International Conference on Data Mining Workshops. Ieee, pp. 725–732.

Correa Bahnsen, A., Stojanovic, A., Aouada, D., Ottersten, B., Dec. 2013. Cost Sensitive Credit Card Fraud Detection Using Bayes Minimum Risk.

In: 2013 12th International Conference on Machine Learning and Applications. IEEE, Miami, USA, pp. 333–338.

Correa Bahnsen, A., Stojanovic, A., Aouada, D., Ottersten, B., Apr. 2014c. Improving Credit Card Fraud Detection with Calibrated Probabilities. In: Proceedings of the fourteenth SIAM International Conference on Data Mining. No. January 2012. Philadelphia, PA, pp. 677 – 685.

Draper, B., Brodley, C., Utgoff, P., 1994. Goal-directed classification using linear machine decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (9), 888–893.

Drummond, C., Holte, R., 2003. C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In: Workshop on Learning from Imbalanced Datasets II, ICML. Washington, DC, USA.

Elkan, C., 2001. The Foundations of Cost-Sensitive Learning. In: Seventeenth International Joint Conference on Artificial Intelligence. pp. 973–978.

Ferri, C., Flach, P., Hernández-Orallo, J., 2002. Learning decision trees using the area under the ROC curve. In: ICML '02 Proceedings of the Nineteenth International Conference on Machine Learning. San Francisco, CA, USA, pp. 139–146.

Glady, N., Baesens, B., Croux, C., Aug. 2009. Modeling churn using customer lifetime value. European Journal of Operational Research 197 (1), 402–411.

Hand, D. J., Henley, W. E., 1997. Statistical Classification Methods in Consumer Credit Scoring: A Review. Journal of the Royal Statstical Society. Series A (Statistics in Society) 160 (3), 523–541.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition. Springer, Stanford, CA.

Jayanta K., G., Mohan, D., Tapas, S., Apr. 2006. Bayesian Inference and Decision Theory. In: An Introduction to Bayesian Analysis. Vol. 13. Springer New York, pp. 26–63.

Kim, J., Choi, K., Kim, G., Suh, Y., Mar. 2012. Classification cost: An empirical comparison among traditional classifier, Cost-Sensitive Classifier, and MetaCost. Expert Systems with Applications 39 (4), 4013–4019.

Kretowski, M., Grześ, M., 2006. Evolutionary induction of cost-sensitive decision trees. In: Foundations of Intelligent Systems. Springer Berlin Heidelberg, pp. 121–126.

Krivko, M., Aug. 2010. A hybrid model for plastic card fraud detection systems. Expert Systems with Applications 37 (8), 6070–6076.

Li, J., Li, X., Yao, X., 2005. Cost-Sensitive Classification with Genetic Programming. In: 2005 IEEE Congress on Evolutionary Computation. Vol. 3. IEEE, pp. 2114–2121.

Ling, C. X., Yang, Q., Wang, J., Zhang, S., 2004. Decision trees with minimal costs. In: Twenty-first international conference on Machine learning - ICML '04. No. Icml. ACM Press, New York, New York, USA, p. 69.

Lomax, S., Vadera, S., Feb. 2013. A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys 45 (2), 1–35.

Ma, J., Saul, L. K., Savage, S., Voelker, G. M., Apr. 2011. Learning to detect malicious URLs. ACM Transactions on Intelligent Systems and Technology 2 (3), 1–24.

Maes, S., Tuyls, K., Vanschoenwinkel, B., Manderick, B., 2002. Credit card fraud detection using Bayesian and neural networks. In: Proceedings of NF2002.

Maimon, L. R. O., 2008. Data Mining with Decision Trees: Theory and Applications. World Scientific Publishing Company.

Moro, S., Laureano, R., Cortez, P., 2011. Using data mining for bank direct marketing: An application of the crisp-dm methodology. In: European Simulation and Modelling Conference. No. Figure 1. Guimares, Portugal, pp. 117–121.

Nayak, G. N., Turvey, C. G., 1997. Credit Risk Assessment and the Opportunity Costs of Loan Misclassification. Canadian Journal of Agricultural Economics 45 (3), 285–299.

Ngai, E., Xiu, L., Chau, D., Mar. 2009. Application of data mining techniques in customer relationship management: A literature review and classification. Expert Systems with Applications 36 (2), 2592–2602.

Ong, C.-s., Huang, J.-j., Tzeng, G.-h., 2005. Building credit scoring models using genetic programming. Expert Systems With Applications 29, 41–47.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J.,

Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Rokach, L., Maimon, O., 2010. Decision Trees. In: Rokach, L., Maimon, O. (Eds.), Data Mining and Knowledge Discovery Handbook, 2nd Edition. Springer US, Ch. 9, pp. 149–174.

Sahin, Y., Bulkan, S., Duman, E., Nov. 2013. A cost-sensitive decision tree approach for fraud detection. Expert Systems with Applications 40 (15), 5916–5923.

Ting, K., 2002. An instance-weighting method to induce cost-sensitive trees. IEEE Transactions on Knowledge and Data Engineering 14 (3), 659–665.

Vadera, S., 2010. CSNL: A cost-sensitive non-linear decision tree algorithm. ACM Transactions on Knowledge Discovery from Data 4 (2), 1–25.

Verbraken, T., Bravo, C., Weber, R., Baesens, B., Oct. 2014. Development and application of consumer credit scoring models using profit-based classification measures. European Journal of Operational Research 238 (2), 505–513.

Wang, T., 2013. Efficient Techniques for Cost-Sensitive Learning with Multiple Cost Considerations. Ph.D. thesis, University of Technology, Sydney.

Whitrow, C., Hand, D. J., Juszczak, P., Weston, D. J., Adams, N. M., Jul. 2008. Transaction aggregation as a strategy for credit card fraud detection. Data Mining and Knowledge Discovery 18 (1), 30–55.

Yeh, I.-C., Lien, C.-h., Mar. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications 36 (2), 2473–2480.

Zadrozny, B., Langford, J., Abe, N., 2003. Cost-sensitive learning by cost-proportionate example weighting. In: Third IEEE International Conference on Data Mining. IEEE Comput. Soc, pp. 435–442.

- Example-Dependent Cost-sensitive tree algorithm
- Each example is assumed to have different financial cost
- Application on credit card fraud detection, credit scoring and direct marketing
- Focus on maximizing the financial savings instead of accuracy
- Code is open source and available at albahnsen.com/CostSensitiveClassification