Chapter 1

Fraud Detection by Stacking Cost-Sensitive Decision Trees

Alejandro Correa Bahnsen*, Sergio Villegas*, Djamila Aouada†
and Björn Ottersten†

*Easy Solutions a Cyxtera Company
8550 NW 33 Street, Doral, FL 33122, USA
acorrea@easysol.net, svillegas@easysol.net

†Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg
2, avenue de l'Université, 4365 Esch-sur-Alzette, Luxembourg
djamila.aouada@uni.lu, bjorn.ottersten@uni.lu

Worldwide, billions of euros are lost every year due to credit card fraud. Increasingly, fraud has diversified to different digital channels, including mobile and online payments, creating new challenges as innovative new fraud patterns emerge. Hence, it remains challenging to find effective methods of mitigating fraud. Existing solutions include simple if-then rules and classical machine learning algorithms. Credit card fraud is by definition an example-dependent and cost-sensitive classification problem, in which the costs due to misclassification vary between examples and not only within classes, i.e., misclassifying a fraudulent transaction may have a financial impact ranging from a few to thousands of euros. In this paper, we propose an extension to the cost-sensitive decision trees algorithm, by creating an ensemble of such trees, and combining them using a stacking approach with a cost-sensitive logistic regression. We compare our method with standard machine learning algorithms and state-of-the-art cost-sensitive classification methods using a real credit card fraud dataset provided by a large European card processing company. The results show that our method achieves savings of up to 73.3%, more than 2 percentage points more than a single cost-sensitive decision tree.

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 to minimize the misclassification of examples, in which an example is misclassified if the class it predicts 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.[1]

Fraud detection has been shown to be a classification problem in which this traditional cost-insensitive approach is not suitable.[2] This is because incorrectly predicting a legitimate transaction to be fraud makes the financial institution incur several different kinds of administrative costs. On the other hand, failing to detect a fraudulent transaction may have a financial impact of a few to thousands of euros, depending on the particular transaction and card holder.[3]

Some methods have been proposed to deal with these example-dependent costs. Standard solutions consist of re-weighting the training examples based on their costs, either by cost-proportionate rejection-sampling,[4] or by cost-proportionate-sampling.[1] Recently, we have proposed different methods that take into account the different example-dependent costs, in particular: Bayes minimum risk (BMR),[2,5] cost-sensitive logistic regression ($CSLR$),[6] and cost-sensitive decision tree ($CSDT$).[7]

A comparison of the above mentioned methods was perform in,[7] the results showed that the most successful algorithm for detecting credit card fraud is the $CSDT$ algorithm. This method is based on a new splitting criteria which is cost-sensitive, used during the tree construction. Then, after the tree is fully grown, it is pruned by using a cost-based pruning criteria. The $CSDT$ algorithm only creates one tree in order to make a classification, however, individual decision trees typically suffer from high variance.[8] A very efficient and simple way to address this flaw is to use them in the context of ensemble methods.

Ensemble learning is a widely studied topic in the machine learning community. The main idea behind the ensemble methodology is to combine several individual base classifiers in order to have a classifier that outperforms each of them.[9] Nowadays, ensemble methods are one of the most popular and well studied machine learning techniques.[10] The core principle in ensemble learning is to induce random permutations into the

learning procedure in order to produce several different base classifiers from a single training set, then combining the base classifiers in order to make the final prediction. Typically these base classifiers are combined using majority voting.[10]

In this paper we propose a new method to create an ensemble of example-dependent cost-sensitive decision trees by combining them using a cost-sensitive stacking algorithm. The stacking algorithm consists of combining the different base classifiers by learning a second level algorithm on top of them.[11] Following the logic used in,[12] our proposed cost-sensitive stacking consists of learning the second level classifier using a cost-sensitive logistic regression ($CSLR$).[6] Furthermore, our source code, as used for the experiments, is publicly available as part of the *CostSensitiveClassification*[a] library.

The remainder of the paper is organized as follows. In Section 2, we expose the downsides of the traditional classification methods on credit card fraud and present an evaluation method based on savings. In Section 3, we present the proposed stacking of cost-sensitive decision-trees algorithm. In Section 4, we describe our experimental setup and present the results of the experiments. Finally, conclusions are given in Section 5.

2. Credit Card Fraud Detection

Online fraud costs the global economy more than $400 billion, with more than 800 million personal records stolen in 2013 alone.[13] Increasingly, fraud has diversified to different digital channels, including mobile and online payments, creating new challenges as innovative fraud patterns emerge. One of the tools used to identify fraudulent transactions is machine learning.

At its essence, machine learning is the study and creation of algorithms that learn and make relevant predictions from data. Within machine learning a sub-family of algorithms called classification methods are being used to distinguish between fraudulent and non fraudulent transactions. The following are examples of models that have been used in the past in fraud detection: neural networks,[14] Bayesian learning,[14] artificial immune systems,[15] association rules,[16] hybrid models,[17] support vector machines,[18] peer group analysis,[19] online learning,[20] discriminant analysis[21] and social network analysis.[22] In this section we will talk about how the traditional evaluation methods for classification methods on credit card detection fail to take into account monetary costs associated with fraud and fraud detec-

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

tion. Afterwards, we will present an evaluation method that corrects for this downfall by optimizing for savings rather than raw predictive power. This correction will serve as a definition for the following machine learning models in subsequent sections.

2.1. *Fraud detection evaluation: traditional versus cost-sensitive approach*

A credit card fraud detection model is based on a classification algorithm that attempts to predict class y_i of a set $\mathcal{S}$ of transactions, given their k features $\mathbf{x}_i \in \mathbb{R}^k$. The objective is to construct a function $f(\cdot)$ that makes a prediction c_i of the class of each example using its variables $\mathbf{x}_i$. Traditionally such systems are evaluated using a standard binary classification measure, such as misclassification error, receiver operating characteristic (ROC), Kolmogorov-Smirnov (KS) or $F_1 Score$ statistics.[20,23,24] However, these measures may be not the most appropriate evaluation criteria when evaluating fraud detection models because they seek to minimize the misclassification rate assuming all hits and misses carry the same value. This assumption does not hold in fraud prevention, since wrongly predicting a fraudulent transaction as legitimate carries a significantly different financial cost than in the case of a false positive. Furthermore, the accuracy measure also assumes that the class distribution among transactions is constant and balanced,[25] and typically the distributions of a fraud detection dataset are skewed, with a percentage of fraud ranging from 0.005% to 0.5%.[15,18]

To take into account the costs of fraud detection operation during evaluation, different costs are assigned to the different misclassification errors. Table 1 is an example matrix representation of the different classification outcomes with the associated costs. Hand et al.[24] proposed a cost matrix, where in the case of a false positive the associated cost is the administrative cost $C_{FP_i} = C_a$ related to analyzing the transaction and contacting the card holder. This cost is the same assigned to a true positive $C_{TP_i} = C_a$, because in this case, the card holder will have to be contacted. However, in the case of a false negative, in which fraud is not detected, the cost is defined to be one hundred times larger, i.e. $C_{FN_i} = 100C_a$. This same approach was also used in.[15]

Nevertheless, in practice, losses due to a particular fraudulent transaction can vary from a couple to thousands of euros, meaning that assuming constant cost for false negatives is unrealistic. In order to address this limitation, we propose a cost matrix that takes into account the actual

Table 1. Credit card fraud detection cost matrix

	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 $c_i = 0$	$C_{FN_i} = Amt_i$	$C_{TN_i} = 0$

example-dependent financial costs. Our cost matrix defines the cost of a false negative to be the amount $C_{FN_i} = Amt_i$ of the transaction i. We argue that this cost matrix is a better representation of the actual cost, since when a fraud is not detected, the losses of that particular fraud correspond to the stolen amount. The costs are summarized in Table 1.

2.2. *Credit card fraud detection model evaluation*

Once the different costs have been defined we must incorporate them into the classification model so it can maximize savings. We first define the cost given a model f. After, we define a way to measure the savings of the model f compared to using no model at all. Both of these definitions are going to be used in further sections.

Let $\mathcal{S}$ be a set of N transaction $\mathbf{x}_i$, where each transaction is represented by the augmented feature vector $\mathbf{x}_i^* = [\mathbf{x}_i, C_{TP_i}, C_{FP_i}, C_{FN_i}, C_{TN_i}]$ and labelled using the class label y_i. A classifier f which generates the predicted label c_i for each transaction i is trained using the set $\mathcal{S}$. Using the cost matrix, an example-dependent cost statistic,[2] is defined as:

$$\begin{aligned} Cost(f(\mathbf{x}_i^*)) =& y_i(c_i C_{TP_i} + (1 - c_i)C_{FN_i}) + \\ & (1 - y_i)(c_i C_{FP_i} + (1 - c_i)C_{TN_i}), \end{aligned} \tag{1}$$

leading to a total cost of:

$$Cost(f(\mathcal{S})) = \sum_{i=1}^{N} Cost(f(\mathbf{x}_i^*)). \tag{2}$$

However, the total cost may not be easy to interpret. A better way to analyze the performance of an algorithm is by using the savings.[6] The savings corresponding to using an algorithm are defined as the cost of the algorithm versus the cost of using no algorithm at all. To do that, the cost of the costless class is defined as

$$Cost_l(\mathcal{S}) = \min\{Cost(f_0(\mathcal{S})), Cost(f_1(\mathcal{S}))\}, \tag{3}$$

where $f_0(\mathcal{S})$ and $f_1(\mathcal{S})$ are trivial classifiers that predict all examples of $\mathcal{S}$ to be either negatives or positives, respectively. Then, the cost improvement can be expressed as the cost of savings as compared with $Cost_l(\mathcal{S})$.

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

3. Stacking cost-sensitive decision-trees

In this section, we present our proposed framework for stacking of example-dependent cost-sensitive decision-trees ($ECSDT$). The framework is based on expanding our previous contribution on example-dependent cost-sensitive decision trees ($CSDT$).[7] In particular, we create many different $CSDT$ on random subsamples of the training set, and then we stack them using a cost-sensitive logistic regression ($CSLR$).[6]

The remainder of the section is organized as follows: First, we introduce the example-dependent cost-sensitive decision tree. Then we show how to ensemble different example-dependent cost-sensitive decision trees. Finally, we present our proposed cost-sensitive stacking algorithm.

3.1. *Cost-sensitive decision tree (CSDT)*

Introducing the cost into the training of a decision tree has been a widely studied way of making classifiers cost-sensitive.[26] However, in most cases, approaches that have been proposed only deal with the problem when the cost difference between false positives and false negatives is constant.[27–32] In order to take into account the varying costs for the different misclassification errors, in,[7] we proposed an example-dependent cost-sensitive decision tree ($CSDT$) algorithm that takes into account the example-dependent costs during the training and pruning of a tree.

The $CSDT$ method uses a new splitting criteria during the construction of a decision tree's construction. In particular, instead of using a traditional splitting criteria such as Gini, entropy or misclassification, the example-dependent cost, as defined in (1), is calculated for each tree node. Then, the gain of using each different split is evaluated as the decrease in total cost.

The cost-based impurity measure compares the costs when all the transactions in a leaf are classified as legitimate and as fraudulent,

$$I_c(\mathcal{S}) = \min\Big\{Cost(f_0(\mathcal{S})), Cost(f_1(\mathcal{S}))\Big\}. \tag{5}$$

Afterwards, using the cost-based impurity, the gain of using the splitting rule $(\mathbf{x}^j, l^j)$, that is the rule of splitting the set $\mathcal{S}$ on feature $\mathbf{x}^j$ on value l^j, is calculated as:

$$Gain(\mathbf{x}^j, l^j) = I_c(\mathcal{S}) - \frac{|\mathcal{S}^l|}{|\mathcal{S}|} I_c(\mathcal{S}^l) - \frac{|\mathcal{S}^r|}{|\mathcal{S}|} I_c(\mathcal{S}^r), \tag{6}$$

where $\mathcal{S}^l = \{\mathbf{x}_i^* | \mathbf{x}_i^* \in \mathcal{S} \wedge x_i^j \leq l^j\}$, $\mathcal{S}^r = \{\mathbf{x}_i^* | \mathbf{x}_i^* \in \mathcal{S} \wedge x_i^j > l^j\}$, and $|\cdot|$ denotes the cardinality. Afterwards, using the cost-based gain measure, a decision tree is grown until no further splits can be made.

Lastly, after the tree is constructed, it is pruned by using a cost-based pruning criteria

$$PC_c = Cost(f(S)) - Cost(f^*(S)), \tag{7}$$

where f^* is the classifier of the tree without the pruned node.

3.2. *Ensembles of Cost-Sensitive Decision Trees*

The main idea behind the ensemble methodology is to combine several individual classifiers, referred to as base classifiers, in order to have a classifier that outperforms every one of them.[9] The most typical form of an ensemble is made by combining T different base classifiers. Each base classifier $M(\mathcal{S}_j)$ is trained by applying algorithm M to a random subset $\mathcal{S}_j$ of the training set $\mathcal{S}$. For simplicity we define $M_j \equiv M(\mathcal{S}_j)$ for $j = 1, \ldots, T$, and $\mathcal{M} = \{M_j\}_{j=1}^T$ a set of base classifiers. Then, these models are combined using majority voting to create the ensemble H as follows

$$f_{mv}(\mathcal{S}, \mathcal{M}) = \arg \max_{c \in \{0,1\}} \sum_{j=1}^{T} \mathbf{1}_c(M_j(\mathcal{S})). \tag{8}$$

Importantly, the base classifiers are typically created by selecting random subsamples $\mathcal{S}_j$ for $j = 1, \ldots, T$, of the training set $\mathcal{S}$. Then, a $CSDT$ algorithm is evaluated on each one of the partitions. To create the different random subsamples we used three different methods: bagging,[33] pasting[34] and random patches.[8]

The bagging[33] method consists in randomly drawn bootstrap subsets of the original data. Pasting[34] is a similar method in which the random samples are extracted without replacement. Lastly, the random patches algorithm,[8] consists of creating base classifiers by randomly drawn bootstrap subsets of both examples and features.

3.3. *Cost-sensitive stacking*

Once different base classifiers are learned, the next step involves combining them. Typically this is done by majority voting as described in (8). However, majority voting gives the same weight to all the base classifiers, but our hypothesis is that individual classifiers should have a different weights according to their performance measured by financial savings. To get there we use stacking learning.

The stacking algorithm consists in combining the different base classifiers by learning a second level algorithm on top of them.[11] In this framework, once the base classifiers are constructed using the training set $\mathcal{S}$, a new set is constructed where the output of the base classifiers is now considered as the feature, while keeping the class labels.

Even though there is no restriction on which algorithm can be used as a second level learner, it is common to use a linear model,[10] such as

$$f_s(\mathcal{S}, \mathcal{M}, \beta) = g\left(\sum_{j=1}^{T} \beta_j M_j(\mathcal{S})\right), \tag{9}$$

where $\beta = \{\beta_j\}_{j=1}^{T}$, and $g(\cdot)$ is the sign function $g(z) = sign(z)$ in the case of a linear regression or the sigmoid function, defined as $g(z) = 1/(1+e^{-z})$, in the case of a logistic regression.

Following the logic used in,[12] we propose learning the set of parameters β using a cost-sensitive logistic regression ($CSLR$).[6] The $CSLR$ algorithm consists of introducing example-dependent costs into a logistic regression, by changing the objective function of the model to one that is cost-sensitive. For the specific case of cost-sensitive stacking, we define the cost function as:

$$\begin{aligned} &J(\mathcal{S}, \mathcal{M}, \beta) = \\ &\sum_{i=1}^{N} \Big[y_i \Big(f_s(\mathbf{x}_i, \mathcal{M}, \beta) \cdot (C_{TP_i} - C_{FN_i}) + C_{FN_i} \Big) + \\ &(1 - y_i) \Big(f_s(\mathbf{x}_i, \mathcal{M}, \beta) \cdot (C_{FP_i} - C_{TN_i}) + C_{TN_i} \Big) \Big]. \end{aligned} \tag{10}$$

Then, the parameters β that minimize the logistic cost function are used in order to combine the different base classifiers. However, as discussed in,[6] this cost function is not convex for all possible cost matrices, therefore, we use genetic algorithms to minimize it.

This method guarantees that the base classifiers that contribute to a higher increase in savings have more importance in the ensemble. Furthermore, by learning an additional second-level cost-sensitive method, the combination is made such that the overall savings measure is maximized.

4. Experiments

In this section, the dataset used for the evaluation of the algorithms is described. Then the partitioning of the dataset and the algorithms used for fraud detection are given. Lastly, we present the experimental results.

4.1. *Database*

For this paper we use 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 750,000 individual transactions, each one with 27 attributes as summarized in Table 2. The database also includes 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 0.467% of the transactions correspond to fraud. Moreover, the total financial losses due to fraud are 866,410 euros.

Using the initial attributes we derived additional 260 attributes using the transaction aggregation methodology.[18,35–37] The idea behind the de-

Table 2. Database attributes

Attribute name	Description
Date	Date and hour of the transaction
Account number	Identification number of the account
Card number	Identification number of the card
Transaction type	Type of transaction (Internet, Card present, ATM)
Amount	Amount of transaction in euros
Merchant ID	Identification of the merchant
Merchant group	Merchant group identification provided by the card processing company
Country	Country where the transaction took place
Country 2	Country of residence of the card holder
Type of card	Card brand (Visa debit, Visa Classic, Mastercard ...)
Gender	Gender of the card holder
Age	Card holder age
Bank	Issuer bank of the card
Fraud	Whenever the transaction was or not fraud

rived attributes consists of using a transaction aggregation strategy in order to capture consumer spending behavior in the recent past. The derivation of the attributes consists of grouping the transactions made during the last given number of 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 made in those transactions. An example of a derived attribute is: number of transactions made during the past 6 hours on the internet by the same individual in the same country.

4.2. *Database partitioning*

First, the database is partitioned in 3 different sets: training(t), validation and testing. Each one containing 50%, 25% and 25% of the transactions, respectively. Afterwards, we perform the cost-proportionate rejection-sampling(r)[4] and cost-proportionate over sampling(o)[1] procedures. The rejection-sampling approach consists of selecting a random subset $\mathcal{S}_r$ by randomly selecting examples from $\mathcal{S}$, and accepting each example i with probability $w_i / \max_{1,\dots,N}\{w_i\}$, where w_i is defined as the expected misclassification error of example i:

$$w_i = y_i \cdot C_{FN_i} + (1 - y_i) \cdot C_{FP_i}. \tag{11}$$

Lastly, the over-sampling method consists of creating a new set $\mathcal{S}_o$, by making w_i copies of each example i. However, cost-proportionate over-sampling increases the training since $|\mathcal{S}_o| >> |\mathcal{S}|$, and it also may result in over-fitting.[38] Furthermore, none of these methods uses the full cost matrix, only the misclassification costs.

Table 3, summarizes the different sets. It is important to note that the sampling procedures were only applied to the training dataset since the validation and test datasets must reflect the real distribution.

Table 3. Summary of the datasets

Set	**# Examples**	**% of Positives**	**Cost**
total	236,735	1.50	895,154
Training (t)	94,599	1.51	358,078
Under-Sampling (u)	2,828	50.42	358,078
Cost-Sensitive Rejection-Sampling (r)	94,522	1.43	357,927
Cost-Sensitive Over-Sampling (o)	189,115	1.46	716,006
Validation	70,910	1.53	274,910
Testing	71,226	1.45	262,167

4.3. *Results*

For the experiments we first used three classification algorithms, decision tree (DT), logistic regression (LR) and random forest (RF). Using the implementation of Scikit-learn,[39] each algorithm is trained using the different training sets: training (t), under-sampling (u), cost-proportionate rejection-sampling (r)[4] and cost-proportionate over-sampling (o).[1] Afterwards, we evaluated the cost-sensitive decision tree ($CSDT$)[7] and cost-sensitive logistic regression ($CSLR$)[6] were also evaluated. Lastly, we calculated the ensemble of cost-sensitive decision trees algorithms using majority voting (mv) and cost-sensitive stacking (s), and the following random inducer methods: bagging (CSB), pasting (CSP), random forests ($CSRF$) and random patches ($CSRP$). Unless otherwise stated, the random selection of the training set was repeated 50 times, and each time the models were trained and results collected, this allows us to measure the stability of the results.

The results are shown in Table 4. Observing the results on the t datasets are not as good as the ones on the u, this is highly related to the unbalanced distribution of the legitimate and fraudulent transactions.

In the case of cost-proportionate sampling methods (CPS), specifically the cost-proportionate rejection sampling (r) and cost-proportionate over sampling (o), it is observed that these methods do not outperform the algorithms trained on the under-sampled set. This may be related to the fact that in this database the initial percentage of positives is 1.5% which is similar to the percentage in the r and o sets. However it is 50.42% in the u set, which may help explain why this method performs much better as measured by savings. Moreover, if we see the results of the family of algorithms is the cost-sensitive training, which includes the $CSLR$ and $CSDT$ techniques. We can observe a significant increase in the performance of the algorithms. The $CSDT$ significantly outperforms the previous algorithms.

Lastly, we evaluate the proposed $ECSDT$ algorithms. Initial results show that the bagging algorithm does not improve the results of the $CSDT$ regardless of the method used for combining the base classifiers. This is already an indicator of the good overall performance of the $CSDT$ algorithm. It is in the case of the random patches that we see an increase in performance measured by savings. In particular, we see that by using $CSRP$ with majority voting there is an increase to 72.2% of savings, and by using the proposed cost-sensitive stacking, we reach a financial savings of 73.3%.

Table 4. Results of the algorithms measured by savings and F1Score

Family	Algorithm	Savings	F1Score
CI	DT-t	0.3176±0.0357	0.4458±0.0133
	LR-t	0.0092±0.0002	0.1531±0.0045
	RF-t	0.3342±0.0156	0.2061±0.0041
	DT-u	0.5239±0.0118	0.1502±0.0066
	LR-u	0.1243±0.0387	0.0241±0.0163
	RF-u	0.5684±0.0097	0.0359±0.0065
CPS	DT-r	0.3439±0.0453	0.4321±0.0086
	LR-r	0.3077±0.0301	0.1531±0.0045
	RF-r	0.3812±0.0264	0.2171±0.0100
	DT-o	0.3172±0.0274	**0.4495±0.0063**
	LR-o	0.2793±0.0185	0.1776±0.0117
	RF-o	0.3612±0.0295	0.2129±0.0080
CST	CSLR-t	0.6113±0.0262	0.2031±0.0065
	CSDT-t	0.7116±0.2557	0.2522±0.0980
ECSDT Majority Voting	CSB-mv-t	0.7124±0.0162	0.2112±0.0125
	CSP-mv-t	0.7106±0.0113	0.2098±0.0126
	CSRP-mv-t	0.7220±0.0082	0.2691±0.0054
ECSDT Stacking	CSB-s-t	0.7181±0.0109	0.2072±0.0103
	CSP-s-t	0.7212±0.0067	0.2064±0.0069
	CSRP-s-t	**0.7336±0.0108**	0.2735±0.0148

(the model with the highest savings and F1Score are marked as bold)

5. Conclusions and future work

In this paper we proposed a new method for stacking example-dependent cost-sensitive decision-trees. The proposed method was applied to credit card fraud detection using a real world database. We have shown experimentally that our method outperforms state-of-the-art classification algorithms and example-dependent cost-sensitive methodologies, when measured by financial savings.

The best results are found when creating and stacking cost-sensitives decision trees created by sub-sampling the training data using the random patches algorithm. Furthermore, the random patches algorithm is the one with the lowest complexity, as each base classifier is learned on a smaller subset than with bagging or pasting. We also found that the proposed stacking method provides an additional percentage point in savings compared to using only majority voting to combine the base classifiers.

Moreover, our results show the importance of using the real example-dependent financial costs associated with credit card fraud detection. In particular, we found significant differences in the results when evaluating a model using a traditional cost-insensitive measure such as F1Score, than when using financial savings.

To improve the results, future research should be focused on developing an example-dependent cost-sensitive boosting approach. For some applications, boosting methods have proven to outperform the bagging algorithms. Futhermore, the methods covered in this work are all batch, in the sense that the batch algorithms keeps the system weights constant while calculating the evaluation measures. However, in fraud detection, the evolving patterns in fraudster behavior are not captured by using batch methods. Therefore, there is a need to investigate this problem from an online-learning perspective.

References

1. C. Elkan. The Foundations of Cost-Sensitive Learning. In *Seventeenth International Joint Conference on Artificial Intelligence*, pp. 973–978 (2001).
2. A. Correa Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten. Cost Sensitive Credit Card Fraud Detection Using Bayes Minimum Risk. In *2013 12th International Conference on Machine Learning and Applications*, pp. 333–338, IEEE, Miami, USA (2013).
3. E. Ngai, Y. Hu, Y. Wong, Y. Chen, and X. Sun, The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature, *Decision Support Systems.* **50**(3), 559–569 (2011).
4. B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Third IEEE International Conference on Data Mining*, pp. 435–442, IEEE Comput. Soc (2003).
5. A. Correa Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten. Improving Credit Card Fraud Detection with Calibrated Probabilities. In *Proceedings of the fourteenth SIAM International Conference on Data Mining*, pp. 677 – 685, Philadelphia, USA (2014).
6. A. Correa Bahnsen, D. Aouada, and B. Ottersten. Example-Dependent Cost-Sensitive Logistic Regression for Credit Scoring. In *2014 13th International Conference on Machine Learning and Applications*, pp. 263–269, IEEE, Detroit, USA (2014).
7. A. Correa Bahnsen, D. Aouada, and B. Ottersten, Example-Dependent Cost-Sensitive Decision Trees, *Expert Systems with Applications.* **49**(19) (2015).
8. G. Louppe and P. Geurts. Ensembles on random patches. In *ECML PKDD'12 Proceedings of the 2012 European conference on Machine Learning and*

Knowledge Discovery in Databases, pp. 346–361, Springer Berlin Heidelberg (2012).
9. L. Rokach, Ensemble-based classifiers, *Artificial Intelligence Review*. **33**(1-2), 1–39 (2009).
10. Z.-H. Zhou, *Ensemble Methods Foundations and Algorithms*. CRC Press, Boca Raton, FL, US (2012).
11. D. H. Wolpert, Stacked generalization, *Neural Networks*. **5**, 241–259 (1992).
12. T. A. Nesbitt. *Cost-Sensitive Tree-Stacking : Learning with Variable Prediction Error Costs*. PhD thesis, University of California, Los Angeles (2010).
13. C. for Strategic, I. Studies, and McAfee. Net Losses: Estimating the Global Cost of Cybercrime. URL `https://www.mcafee.com/mx/resources/reports/rp-economic-impact-cybercrime2.pdf` (2014).
14. S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick. Credit card fraud detection using Bayesian and neural networks. In *Proceedings of NF2002* (2002).
15. S. Bachmayer, Artificial Immune Systems, *Artificial Immune Systems*. **5132**, 119–131 (2008).
16. D. Sánchez, M. Vila, L. Cerda, and J. Serrano, Association rules applied to credit card fraud detection, *Expert Systems with Applications*. **36**(2), 3630–3640 (2009).
17. M. Krivko, A hybrid model for plastic card fraud detection systems, *Expert Systems with Applications*. **37**(8), 6070–6076 (2010).
18. S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, Data mining for credit card fraud: A comparative study, *Decision Support Systems*. **50** (3), 602–613 (2011).
19. D. J. Weston, D. J. Hand, N. M. Adams, C. Whitrow, and P. Juszczak, Plastic card fraud detection using peer group analysis, *Advances in Data Analysis and Classification*. **2**(1), 45–62 (2008).
20. A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, Learned lessons in credit card fraud detection from a practitioner perspective, *Expert Systems with Applications*. **41**(10), 4915–4928 (2014).
21. N. Mahmoudi and E. Duman, Detecting credit card fraud by Modified Fisher Discriminant Analysis, *Expert Systems with Applications*. **42**(5), 2510–2516 (2015).
22. V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, APATE: A Novel Approach for Automated Credit Card Transaction Fraud Detection using Network-Based Extensions, *Decision Support Systems*. **75**, 38–48 (2015).
23. R. J. Bolton, D. J. Hand, F. Provost, and L. Breiman, Statistical Fraud Detection: A Review, *Statistical Science*. **17**(3), 235–255 (2002).
24. D. J. Hand, C. Whitrow, N. M. Adams, P. Juszczak, and D. J. Weston, Performance criteria for plastic card fraud detection tools, *Journal of the Operational Research Society*. **59**(7), 956–962 (2007).
25. F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 445–453, Morgan Kaufmann

(1998).
26. S. Lomax and S. Vadera, A survey of cost-sensitive decision tree induction algorithms, *ACM Computing Surveys.* **45**(2), 1–35 (2013).
27. B. Draper, C. Brodley, and P. Utgoff, Goal-directed classification using linear machine decision trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence.* **16**(9), 888–893 (1994).
28. K. Ting, An instance-weighting method to induce cost-sensitive trees, *IEEE Transactions on Knowledge and Data Engineering.* **14**(3), 659–665 (2002).
29. C. X. Ling, Q. Yang, J. Wang, and S. Zhang. Decision trees with minimal costs. In *Twenty-first international conference on Machine learning - ICML '04*, number Icml, p. 69, ACM Press, New York, New York, USA (2004).
30. J. Li, X. Li, and X. Yao. Cost-Sensitive Classification with Genetic Programming. In *2005 IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2114–2121, IEEE (2005).
31. M. Kretowski and M. Grześ. Evolutionary induction of cost-sensitive decision trees. In *Foundations of Intelligent Systems*, pp. 121–126. Springer Berlin Heidelberg (2006).
32. S. Vadera, CSNL: A cost-sensitive non-linear decision tree algorithm, *ACM Transactions on Knowledge Discovery from Data.* **4**(2), 1–25 (2010).
33. L. Breiman, Bagging predictors, *Machine Learning.* **24**(2), 123–140 (1996).
34. L. Breiman, Pasting small votes for classification in large databases and online, *Machine Learning.* **103**, 85–103 (1999).
35. C. Whitrow, D. J. Hand, P. Juszczak, D. J. Weston, and N. M. Adams, Transaction aggregation as a strategy for credit card fraud detection, *Data Mining and Knowledge Discovery.* **18**(1), 30–55 (2008).
36. A. Correa Bahnsen, D. Aouada, and B. Ottersten. Detecting Credit Card Fraud Using Periodic Features. In *2015 14th International Conference on Machine Learning and Applications*, IEEE, Detroit, USA (2015).
37. A. Correa Bahnsen, D. Aouada, and B. Ottersten, Feature engineering strategies for credit card fraud detection, *Expert Systems with Applications.* **51**(1), 134–142 (2016).
38. C. Drummond and R. Holte. 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 (2003).
39. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research.* **12**, 2825–2830 (2011).