The Hidden Cost of Fraud: An Instance-Dependent Cost-Sensitive Approach for Positive and Unlabeled Learning

Carlos Ortega Vázquez CARLOSEDUARDO.ORTEGAVAZQUEZ@KULEUVEN.BE
Jochen De Weerdt JOCHENDEWEERDT@KULEUVENBE
Research Center for Information Systems Engineering (LIRIS), KU Leuven, Leuven, Belgium

Seppe vanden Broucke SEPPEVANDENBROUCKE@UGENT
Department of Business Informatics and Operations Management, Ghent University, Ghent, Belgium

Editor: Nuno Moniz, Paula Branco, Luís Torgo, Nathalie Japkowicz, Michał Woźniak and Shuo Wang.

Abstract

Financial institutions have increasingly suffered pressure to implement better and faster fraud detection systems to minimize the cost of fraud. This issue has attracted attention from the literature over recent years. Despite the practical relevance, few works have considered label uncertainty in fraud detection. The incomplete label information naturally arises because fraudsters strive to go undetected. Most fraud detection systems operate by spending more resources to investigate only few suspicious cases and quickly process the rest as unsuspicious. That is, we only have positive label information of some fraudsters whereas the rest of the positives, together with legitimate non-fraudsters, remain unlabeled. This setting is referred to as learning from positive and unlabeled data, or PU learning. Besides the issue of undetected fraudsters, fraud detection is commonly regarded as a cost-sensitive classification task in which the misclassification cost can substantially vary between examples. Thus, this work introduces a novel technique that integrates PU learning and the instance-dependent cost-sensitive framework: PU-CSBoost. PU-CSBoost can directly minimize financial loss through an instance-dependent cost measure that also incorporates the misclassification cost due to hidden fraudsters. Our empirical analysis compares PU-CSBoost with CSBoost, its non-PU counterpart, and other PU techniques specialized in imbalanced learning. The experimental results emphasize the PU-CSBoost's potential to diminish financial losses under the PU setting. Moreover, the results suggest a quick drop in cost-sensitive performance by CSBoost when hidden fraudsters are present. Thus, ignoring the issue of hidden fraudsters can lead to an underwhelming performance in cost savings for techniques based on the cost-sensitive framework.

Keywords: positive and unlabeled learning, instance-dependent cost-sensitive learning, fraud detection

1. Introduction

The ubiquity of e-commerce in our society has posed increasing pressure for implementing better fraud-detection systems. For example, according to the sixth report on card fraud by the European Central Bank, the total value of fraudulent transactions conducted by cards issued within SEPA and acquired worldwide reached €1.8 billion in 2018 (European Central

Bank, 2020). As a result, financial institutions have relied on more data-driven solutions to decrease the monetary loss due to fraud.

Several works have addressed the challenge of fraud detection (Bolton and Hand, 2002; Ngai et al., 2011; Van Vlasselaer et al., 2015; Dal Pozzolo et al., 2014). Most literature has focused on improving statistical metrics to evaluate predictive performance. However, some works have adopted a cost-sensitive approach incorporating class-dependent misclassification costs of fraud (Chan; Sahin et al., 2013). More recently, instance-dependent cost-sensitive (IDCS) methods have exploited the information on the misclassification cost at transaction or example level for fraud detection (Bahnsen et al., 2015; Zelenkov, 2019; Höppner et al., 2022). IDCS techniques have the potential to outperform class-dependent methods in terms of costs (Vanderschueren et al., 2022).

In the literature on fraud detection, few works have considered the setting of weakly supervised learning (Chen et al., 2021; Jiang et al., 2020). The assumption of complete and perfect label information is often violated in fraud detection. For example, in insurance fraud detection, it has been reported that a high proportion of fraudulent claims go unnoticed by investigators (Caron and Dionne, 1999; Šubelj et al., 2011). This paper considers a specific setting of weakly supervised learning: positive and unlabeled (PU) learning. PU learning assumes that the model can only observe labeled and unlabeled examples; labeled instances are positive, but the unlabeled examples can belong to either the positive or negative class. This setting describes more realistically several real-world applications, including fraud detection: biological and medical diagnosis (Chen et al., 2020; Mordelet and Vert, 2014); banking services (Jiang et al., 2020); and e-commerce (Wu et al., 2018).

Given the business objective in financial institutions to minimize the loss of fraud, together with the issue of label uncertainty in fraud detection, we address an important gap in the literature. To the best of our knowledge, no IDCS method for PU learning has been proposed before. Therefore, we introduce the first IDCS method for fraud detection that can deal with positive and unlabeled data: PU-CSBoost. Our method extends upon a previous IDCS method (Höppner et al., 2022), CSBoost, that successfully outperforms XGBoost (Chen and Guestrin, 2016) in terms of cost-sensitive metrics. CSBoost adopts a customized loss function related to the cost of fraud per instance. Despite the strong performance in cost-sensitive metrics, CSBoost assumes perfect label information, excluding hidden fraudsters' existence. PU-CSBoost deals with the issue of hidden fraudsters by estimating the probability that a seemingly legitimate example is actually fraudulent. Thus, PU-CSBoost utilizes this probability to enable PU learning. Our empirical analysis shows the potential benefits of PU-CSBoost over CSBoost in the PU setting; the experiments also emphasize how detrimental the presence of hidden fraudsters is to the performance of cost-sensitive techniques.

2. Related Work

2.1. Cost Sensitive Learning

In numerous domain applications of binary classification, including fraud detection, the class of interest (i.e., minority class) is rare. The rarity of the positive class constitutes a challenge for standard classifiers, as most conventional algorithms are biased towards the majority class. Specifically, minority class examples are misclassified more often when compared to those from the majority class. Thus, several techniques have been proposed to address the class imbalance issue. These methods can be divided into four main categories (Fernández et al., 2018): data-level methods, algorithm-level methods, cost-sensitive learning, and ensemble-based approaches.

An algorithm can integrate the costs at different stages: direct approaches introduce the misclassification costs during the training procedure; meta-learning approaches modify the data or output based on the cost information rather than adapting the algorithm. Most cost-sensitive algorithms utilize class-dependent (Höppner et al., 2020; Krawczyk et al., 2014; Domingos, 1999; Chawla et al., 2008). However, recent instance-dependent cost-sensitive (IDCS) algorithms have been proposed in the literature (Zelenkov, 2019; Höppner et al., 2022; Bahnsen et al., 2015). Furthermore, IDCS techniques have the potential to outperform their class-dependent counterparts in terms of cost savings at the expense of lower performance in standard metrics (Vanderschueren et al., 2022).

In fraud detection, CSBoost shows a solid capability to reduce financial loss due to fraud by including a customized instance-dependent cost matrix. CSBoost, however, assumes that the data contains complete label information, which rules out the existence of hidden or undetected fraudsters. Therefore, we can further adapt the cost matrix used in CSBoost to incorporate the existence of hidden fraudsters. That is, we can extend CSBoost into the PU setting.

2.2. PU Learning

PU learning is a setting related to weakly supervised learning, in which only positive and unlabeled examples exist. Different assumptions can be made regarding labeled positive examples or the underlying labeling mechanism. Most PU learning methods are built on the selected completely at random (SCAR) assumption. The SCAR assumption states that the positively labeled examples are a randomly selected subset of the set of positives. Other labeling mechanisms, such as selected at random (SAR), have been less explored by the literature (Bekker and Davis, 2020; He et al., 2018).

PU learning methods can be divided into three categories: two-step techniques, biased learning, and class-prior-based methods (Bekker and Davis, 2020). In this work, we focus on the class-prior-based methods as they have been considered state-of-the-art in the field (Kiryo et al., 2017; Du Plessis et al., 2015; Su et al., 2021; Chen et al., 2021). Most of these methods modify the objective function to accommodate the PU setting, which can be readily used in many algorithms, such as neural networks or logistic regressions.

Although the PU learning literature has developed a plethora of methods, most works have focused on a balanced setting. Likewise, no study has focused on evaluating well-known PU learning methods in highly imbalanced settings. Compared to imbalanced classification with complete label information, PU classification under class imbalance suffers even worse from severe underrepresentation of the positive class. In such a scenario, the bias towards the majority class increases for most PU methods. A few studies have proposed methods to handle class imbalance. These PU methods for the imbalanced setting have used class-dependent cost-sensitive learning (Chen et al., 2021) and algorithm-level approaches (Sakai et al., 2018; Su et al., 2021). We consider for the empirical analysis cost-sensitive PU learning (CS-PU) (Chen et al., 2021) and Imbalanced Non-Negative PU (imbalanced nnPU) (Su et al., 2021). CS-PU and imbalanced nnPU are based on the risk minimization framework to enable PU learning (Du Plessis et al., 2015). CS-PU utilizes a weighted PU loss function that includes the class-dependent costs for training the model. Unlike the CS-PU, imbalanced nnPU does not require misclassification cost information as it relies on internally oversampling to balance the class distribution. This method requires the oversampling rate to be tuned as an additional hyperparameter. Imbalanced nnPU offers better protection against overfitting compared to previous works on PU loss functions (Du Plessis et al., 2015; Du Plessis and Sugiyama, 2014), which can be more easily implemented in strong classifiers, such as XGBoost or neural networks. To our knowledge, no IDCS technique has been proposed for fraud detection under the PU setting. Our contribution to the literature consists of proposing PU-CSBoost, which can directly minimize the financial loss of fraud, including the cost of hidden fraudsters.

3. Instance-Dependent Cost-Sensitive Learning with PU Data for Fraud Detection

3.1. Towards an Instance-Dependent Cost Matrix for PU learning

A fraud detection system aims to identify the examples with the highest probability of being fraudulent. In most cases, this task can be regarded as a binary classification problem in which fraudulent examples belong to the positive class (class 1), and the legitimate instances, to the negative class (class 0). The outcome of a binary classifier can be represented in a confusion matrix as shown in Table $1(a)$. The confusion matrix describes misclassification costs and the benefits of correctly classified examples. Let $C_i(\hat{y}|y)$ be the monetary outcome of predicting class $\hat{y}$ for an example i in case that the true class is y. The costs and benefits can be different at each cell of a confusion matrix or specific to the example i; in credit card fraud detection, each example i consists of specific amounts that distinguish it from the rest. Previous works (Höppner et al., 2022; Hand et al., 2008) in fraud detection have assumed a fixed cost c_f for the false positive and the true negative: $C_i(1|0) = C_i(1|1) = c_f$. The fixed cost reflects the administrative cost that includes at least, investigating the particular transaction and contacting the customer. Moreover, the cost of undetected fraud (i.e., false negative) is defined by the amount of the transaction: $C_i(0|1) = A_i$. Table $1(a)$ describes a cost matrix in case of complete label information used in CSBoost, which reflects the standard positive-negative classification.

Despite the popularity of the positive-negative classification, label uncertainty naturally arises in fraud detection. Some fraudsters successfully avoid detection, so their true label is never revealed. Given the issue of hidden fraudsters, the cost matrix shown in Table $1(a)$ is no longer valid. In contrast to Table $1(a)$, the left column in Table $1(b)$ now refers to "Seemingly legitimate". In this setting, true label y is unavailable, and we only observe the indicator l for the examples with the positive label, which means that they are known fraudsters. We are no longer certain which examples truly belong to the negative class. To reflect this uncertainty, the cost $C_i(\hat{y} = 0 \mid l = 0)$ in Table $1(b)$ is calculated as follows:

$$\begin{aligned} C_i(\hat{y} = 0 \mid l = 0) =& P(y = 1 \mid l = 0, x_i) C_i(\hat{y} = 0 \mid y = 1) + \\ & P(y = 0 \mid l = 0, x_i) C_i(\hat{y} = 0 \mid y = 0) \\ =& P(y = 1 \mid l = 0, x_i) A_i, \end{aligned} \tag{1}$$

where $C_i(\hat{y} = 0 \mid y = 0) = 0$. Thus, the cost of considering an instance i as legitimate depends on the transaction amount and the probability of being a hidden fraudster. Also, the cost $C_i(\hat{y} = 1 \mid l = 0)$, in principle, incorporates the probability of being a hidden fraudster, but the cost is simplified to the fixed cost c_f. The cost of predicting a transaction as fraudulent is the same regardless of the label. Table 1(*b*) describes more realistically the costs due to hidden fraudsters. Although we have assumed a fixed cost for false and true positives, any cost matrix can be used. As suggested before (Höppner et al., 2022), we can incorporate a variable cost for false positives that reflects the customer churn due to negative experiences.

Table 1: Confusion matrix of a binary classification task. Between square brackets, the related instance-dependent classification costs for transaction fraud are given.

(*a*) Positive and Negative Setting as in CSBoost

	Actual legitimate $y = 0$	Actual fraudulent $y = 1$
Predicted as legitimate (negative) $\hat{y} = 0$	True negative $[C_i(0\|0) = 0]$	False negative $[C_i(0\|1) = A_i]$
Predicted as fraud (positive) $\hat{y} = 1$	False positive $[C_i(1\|0) = c_f]$	True positive $[C_i(1\|1) = c_f]$

(*b*) Positive and Unlabeled Setting as in PU-CSBoost

	Seemingly legitimate $l = 0$	Actual fraudulent $l = 1$
Predicted as legitimate (negative) $\hat{y} = 0$	True or false negative $[C_i(0\|0) = P_i A_i]$	False negative $[C_i(0\|1) = A_i]$
Predicted as fraud (positive) $\hat{y} = 1$	True or false positive $[C_i(1\|0) = c_f]$	True positive $[C_i(1\|1) = c_f]$

3.2. The Cost of the Fraud Detection Model under the PU Setting

Following the notation as in Höppner et al. (2022), $\mathcal{D}$ consists of a set of N examples, which each contains a tuple of feature set with d variables $\mathcal{X}_i = (x_{i1}, ..., x_{id})$, positive label indicator $l_i \in \{0, 1\}$, and ground-truth label $y_i \in \{0, 1\}$: $\{(\mathcal{X}_i, l_i, y_i)\}_i^N$. A classifier f can learn from a data set $\mathcal{D}$ to provide a fraud score $f : \mathcal{X}_i \to s_i \in [0, 1]$, which can be converted to a predicted label $\hat{y}_i$ based on a threshold t. Based on the cost matrix under the PU setting (PU-cost), shown in Table 1(*b*), the cost that a classifier incurs is the sum of the amounts of undetected fraud (known positives) and hidden fraudsters (unlabeled positives) plus the administrative cost spent. This cost can be calculated as follows:

$$\begin{aligned} PU - Cost(f(\mathcal{X}_i)) &= \sum_{i=1}^{N} \Big(l_i \Big[\hat{y}_i C_i(1 \mid 1)(1 - \hat{y}_i) C_i(0 \mid 1) \Big] + \\ &\quad (1 - l_i) \Big[\hat{y}_i C_i(1 \mid)(1 - \hat{y}_i) C_i(0 \mid 0) \Big] \Big) \\ &= \sum_{i=1}^{N} l_i (1 - \hat{y}_i) A_i + (1 - l_i)(1 - \hat{y}_i) P(y = 1 \mid l = 0, x_i) A_i + \hat{y}_i c_f. \end{aligned} \tag{2}$$

The computation of PU-cost, as shown in Equation 2, depends on the predicted label $\hat{y}_i$. Notice that the conditional expected value of the predicted label $\hat{y}_i$ approximates the fraud score: $E[\hat{y}_i \mid x_i] \approx s(x_i)$. Thus, we can define the PU average expected cost (PU-AEC) of a classifier as follows:

$$\begin{aligned} PU - AEC(f(\mathcal{X}_i)) &= \frac{1}{N}\sum_{i=1}^{N} E[PU - Cost(f(\mathcal{X}_i)) \mid \mathcal{X}] \\ &= l_i(1-s_i)A_i + (1-l_i)(1-s_i)P(y=1 \mid l=0, x_i)A_i + s_i c_f. \end{aligned} \tag{3}$$

The PU average expected cost (PU-AEC) utilizes the estimated fraud probabilities s_i instead of the predicted labels. When the transaction is labeled positive (i.e., known fraudster), then the estimated proportion of the amount that is not detected $(1 - s_i)A_i$ is the expected cost of the instance. If the transaction is an unlabeled positive (i.e., hidden fraudster), then the expected cost of the instance is the estimated proportion of undetected amount weighted by the probability of being hidden fraudster $(1-s_i)P(y = 1 \mid l = 0, x_i)A_i$. The element $s_i c_f$ is the expected fraction of the fixed cost spent in transaction i.

As PU-AEC reflects the financial loss due to fraud at the instance level under the PU setting, we can exploit it as the loss function of an algorithm. In this work, we focus on XGBoost (Chen and Guestrin, 2016), a tree-based gradient boosting algorithm. XGBoost allows the end-user to specify the PU-AEC as the loss function, which enables cost-sensitive PU learning. Our technique PU-CSBoost is, thus, an adaptation for cost-sensitive PU learning.

3.3. Estimation of the Probability of Being Hidden Fraudster

The probability of being hidden fraudster $P(y = 1 \mid l = 0, x_i)$ is the key element that enables PU learning in our technique. However, the estimation of this probability is not trivial because the ground-truth label y is unavailable. One solution is suggested from the PU learning literature (Elkan and Noto, 2008). If the selected completely at random assumption (SCAR) holds, the probability $P(y = 1 \mid l = 0, x_i)$ can be computed by using the class prior in the following fashion:

$$P(y = 1 \mid l = 0, x_i) = \frac{1-c}{c}\frac{P(l = 1 \mid x_i)}{1 - P(l = 1 \mid x_i)}, \tag{4}$$

where c is the label frequency, the proportion of positives that are labeled. The label frequency is closely related to the class prior $P(y = 1)$: $c = P(l = 1)/P(y = 1)$. The conditional probability $P(l = 1 \mid x_i)$ can be estimated by training a standard classifier on the available PU data. The same authors recommend using a classifier that can provide calibrated probabilities, for example, logistic regression.

4. Experimental Setup

The main goal of the experimental evaluation is to demonstrate the cost-related performance achieved by PU-CSBoost when applied to imbalanced PU data. Therefore, we compare PU-CSBoost with four benchmark techniques, including CS-Boost, on two credit card transaction data sets. We use two evaluation metrics that have been used in imbalanced learning (Fernández et al., 2018; Höppner et al., 2022; Vanderschueren et al., 2022): F1-score, as well as disaggregated into precision and recall, and expected savings. Unlike expected savings, F1-score depends on a threshold that might disadvantage techniques that do not provide calibrated scores, for instance, tree-based methods. Thus, the threshold is optimized on a validation set to maximize the F1-score for all techniques for each experimental setting. We also report the precision and recall that correspond to each computed F1-score.

Cost savings have been used as an evaluation metric for cost-sensitive algorithms in previous studies (Whitrow et al., 2009; Bahnsen et al., 2015; Höppner et al., 2022; Vanderschueren et al., 2022). Cost savings is formulated as the improvement in costs by utilizing a technique compared to a baseline cost,

$$Savings(f(\mathcal{X}_i)) = \frac{Cost_{baseline} - Cost(f(\mathcal{X}_i))}{Cost_{baseline}}. \tag{5}$$

The baseline cost refers to having no fraud detection system at all. The cost consists of the sum of the amounts of all fraudulent examples. Savings can be transformed into the expected savings, which is threshold-independent. Expected savings are calculated as follows:

$$\begin{aligned} ExpectedSavings(f(\mathcal{X}_i)) &= \frac{Cost_{baseline} - E[Cost(f(\mathcal{X}_i)) \mid \mathcal{X}]}{Cost_{baseline}} \\ &= \frac{\sum_{i=1}^{N} y_i s_i A_i - s_i c_f}{y_i A_i}. \end{aligned} \tag{6}$$

Equation 6 shows that higher savings can be achieved by providing a higher fraud score to actual fraudulent examples; the improvement in savings is even better for high-amount fraudulent transactions. Notice that for evaluating the techniques, we assume that the ground-truth labels are available, as in most works in PU learning literature. This assumption holds in an experimental evaluation. In practice, we might not be able to compute evaluation metrics without the assistance of financial institutions. Studies have aimed to estimate evaluation metrics under the PU setting (Sakai et al., 2018; Lee and Liu, 2003).

4.1. Data

The first benchmark data set is the credit card fraud provided by Worldline and ULB (MLG, 2018) on Kaggle. This data set contains 284807 transactions, in which the fraudulent examples correspond to 0.17% of all transactions. The features of the ULB data set are numerical because they are the output of a PCA transformation. Therefore, following Höppner et al. (2022), we use the original 28 PCA features plus the logarithmically transformed amount as the set of features for the classification task.

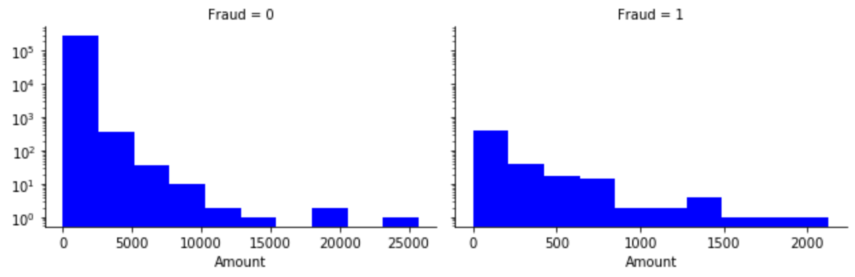
The second data set, referred to as Bank (BAN), has been given access to our research group by a large bank. The data consists of 298587 transactions, in which 0.81% of the total examples have been considered fraudulent. The transactions were made in October 2013 during the first three days of the month. The features represent customers' behavior regarding recency, frequency, and monetary value. As in the ULB data set, we add the logarithmic transformation of the transaction amount into the feature set.

Figure 1 displays the distribution of the transaction amounts for each data set and class. The fraudulent or positive class is generally more dispersed than the negative class. Another noticeable difference is that the BAN data set exhibits more extreme amounts values than the ULB data set. Table 2 summarizes the aforementioned information of the data sets.

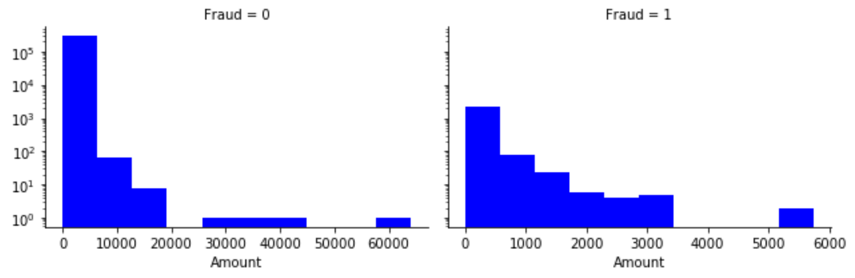
To create the PU data, we flip at random (i.e., SCAR) a proportion of the positives or fraudulent transactions into unlabeled examples in each data set. Across the experiments, the number of labeled positives is determined by the flip ratio, which is the ratio of positives to be unlabeled. Three values of the flip ratio are considered: 25, 50, and 75%. For each data set, we perform 20 repetitions of a holdout validation that splits the data into a training set (70%) and a test set (30%). Following a similar preprocessing (Höppner et al., 2022), the splits are stratified so that both training and test contain an equal quantity of high, middle, and low amounts of fraudulent cases. Hence, we calculate the 33 and 66th percentiles, and we divide every example into one of these categories (i.e., low, middle, or high). In total, there exist 120 settings (2 datasets $\times$ 20 repetitions $\times$ 3 flip ratios).

Table 2: Summary of data sets

Data set	Examples	Features	$P(y = 1)$
Kaggle ULB (ULB)	284807	29	0.17%
Bank (BAN)	298587	29	0.81%



(a) Histogram of Amounts in ULB data set



(b) Histogram of Amounts in BAN data set

Figure 1: Histogram of Amounts

4.2. Techniques

We compare our PU-CSBoost with four techniques, two of them are PU methods: imbalanced non-negative PU learning (imbalanced nnPU) (Su et al., 2021), cost-sensitive PU learning (CS-PU) (Chen et al., 2021), CSBoost (Höppner et al., 2022), and XGBoost (Chen and Guestrin, 2016). Imbalanced nnPU, CS-PU, and CSBoost are based on XGBoost since they adapt the gradient boosting algorithm into PU or cost-sensitive learning. The PU techniques in this empirical analysis belong to the class-prior-based category, including PU-CSBoost, because they all depend on the class prior $P(y = 1)$. In this work, we assume that the class prior is known at training time. In practice, the class prior can be estimated using several methods from the literature (Elkan and Noto, 2008; Bekker and Davis, 2018; Ramaswamy et al., 2016). Table 3 summarizes the hyperparameter configuration of the techniques in the experimental setup.

Apart from the main comparison between techniques, we explore the effect of the amount-based outliers on PU-CSBoost. Previous works have reported that CSBoost can suffer a drop in performance, measured in cost-insensitive metrics, due to a sensitivity to outliers Vanderschueren et al. (2022); Höppner et al. (2022). This reflects a trade-off between cost-sensitive and cost-insensitive metrics. An end-user might prefer a more intermediate solution in which the cost-sensitive model can remain competitive in standard metrics. One strategy from robust statistics is the winsorization (Ronchetti and Huber, 2009). We implement a one-sided 95% winsorization during the training procedure: all amounts above the 95th percentile are to such a percentile. This strategy is referred to as "Winsorized PU-CSBoost" in the rest of the paper.

Table 3: Experimental hyperparameters

Technique	Setting
PU-CSBoost	classifier = XGBoost, loss function = PU average expected cost,
	class prior = $P(y = 1)$, cost matrix as defined in Table 1(b),
	$P(l = 1 \mid x)$ is estimated by a default logistic regression as in `scikit-learn`
Winsorized PU-CSBoost	winsorization = one-sided 95%,
	the rest of the hyperparameters as in PU-CSBoost
CSBoost (Höppner et al., 2022)	classifier = XGBoost, loss function = average expected cost,
	cost matrix as defined in Table 1(a)
imbalanced nnPU (Su et al., 2021)	classifier = XGBoost, loss function = binary cross-entropy,
	class prior = $P(y = 1)$, oversampling rate = balanced ratio
CS-PU (Chen et al., 2021)	classifier = XGBoost, loss function = weighted binary cross-entropy,
	class prior = $P(y = 1)$,
	class-level costs = average amount of the positive and negative class
XGBoost (Chen and Guestrin, 2016)	hyperparameters recommended by authors

5. Results & Discussion

The average and standard deviation of the evaluation metrics across different flip ratios are shown in Tables 4, 5, 6, and 7. Tables 4 and 5 present results for the ULB data set whereas Tables 6 and 7, for the BAN data set. We report precision and recall as well as F1-scores for more granular analysis. The bold and underlined value indicates the best performing model for a given flip ratio.

PU-CSBoost outperforms other techniques, including CSBoost, in expected savings in all experimental settings. At higher flip ratios, the advantage of PU-CSBoost over CSBoost on expected savings is more prominent in both data sets. Likewise, CSBoost performs similarly in cost savings to other techniques specialized for PU learning at a 75% flip ratio. For instance, in the ULB data set at 75% flip ratio, PU-CSBoost (43.0%) performs around three times better in terms of expected savings than CS-Boost (14.7%); The performance of CSBoost is instead closer to Imbalanced nnPU's, which is a cost-insensitive technique. Thus, CSBoost shows a substantial weakness in the PU setting, which might jeopardize its usefulness in fraud detection. Unlike CS-Boost, PU-CSBoost can improve the cost savings in fraud detection tasks with potential hidden fraudsters. Moreover, despite being a cost-sensitive PU method, CS-PU lacks performance in expected savings at a high flip ratio; CS-PU is the only technique that shows negative expected savings at the high flip ratio, which means that the incurred cost of the algorithm surpasses the baseline cost.

Table 4: Recall & Precision (%) across flip ratios for ULB data set when $c_f = 10$

	Recall			Precision		
Models Flip Ratio	25%	50%	75%	25%	50%	75%
PU-CSBoost	40.6 ± 4.2	35.1 ± 4.2	30.1 ± 5.0	67.7 ± 4.9	60.4 ± 7.4	54.8 ± 9.8
Winsorized PU-CSBoost	42.6 ± 5.0	36.8 ± 5.0	33.6 ± 4.4	75.0 ± 3.0	73.5 ± 3.3	69.5 ± 4.7
CS-Boost	39.8 ± 4.9	27.5 ± 6.3	16.1 ± 6.0	82.3 ± 4.1	**83.1 ± 5.9**	**83.4 ± 8.4**
Imbalanced nnPU	**80.9 ± 9.1**	**76.4 ± 7.1**	**58.7 ± 10.3**	71.3 ± 10.0	72.1 ± 8.1	74.5 ± 8.6
CS-PU	78.7 ± 4.2	72.3 ± 3.9	52.1 ± 20.9	62.8 ± 15.3	50.6 ± 15.7	27.7 ± 16.8
XGBoost	78.1 ± 2.7	70.6 ± 6.4	52.1 ± 6.8	**85.6 ± 2.5**	81.3 ± 5.4	81.1 ± 5.5

Best performing model is in **bold and underlined** in each column.

Table 5: F1-score & Expected Savings (%) across flip ratios for ULB data set when $c_f = 10$

	F1-score			Expected Savings		
Models Flip Ratio	25%	50%	75%	25%	50%	75%
PU-CSBoost	50.5 ± 3.4	44.0 ± 3.7	38.4 ± 5.1	64.0 ± 6.4	55.0 ± 6.9	43.0 ± 7.7
Winsorized PU-CSBoost	54.2 ± 4.5	48.8 ± 4.4	45.1 ± 4.2	**64.9 ± 4.5**	**55.3 ± 5.8**	**44.9 ± 6.0**
CS-Boost	53.5 ± 4.9	40.8 ± 7.0	26.5 ± 8.8	54.3 ± 7.7	34.0 ± 8.3	14.7 ± 5.9
Imbalanced nnPU	74.7 ± 3.6	73.5 ± 4.0	**64.9 ± 8.0**	45.6 ± 4.5	26.9 ± 6.2	11.1 ± 3.6
CS-PU	68.6 ± 15.1	57.9 ± 14.5	34.5 ± 19.5	51.5 ± 36.1	45.2 ± 29.1	-4.1 ± 104.7
XGBoost	**81.7 ± 1.2**	**75.2 ± 3.4**	63.1 ± 5.6	45.9 ± 4.4	27.0 ± 7.6	10.6 ± 3.5

Best performing model is in **bold and underlined** in each column.

PU-CSBoost, by algorithmic design, incorporates the amounts of likely hidden fraudsters. This algorithmic adaptation into the PU settings implies a trade-off between recall and precision: more hidden fraudsters are recovered, but more non-fraudsters are misclassified. Hence, PU-CSBoost generally obtains a higher recall than CS-Boost, particularly at higher flip ratios; CS-Boost outperforms most techniques in terms of precision. We also observe this pattern between XGBoost and Imbalanced nnPU, two cost-insensitive techniques. Between these two techniques, XGBoost reaches a higher precision, whereas Imbalanced nnPU obtains a higher recall. Thus, the best performing technique regarding the F1-score is either XGBoost or Imbalanced nnPU.

One significant difference between the two data sets is that the BAN data set contains more high-amount examples than the ULB data set, as shown in Figure 1. These outliers can disproportionately affect a cost-sensitive algorithm, especially in techniques that exploit instance-level costs (Vanderschueren et al., 2022). Winsorized PU-CSBoost curbs the effect of the outliers through winsorization. This results in a slight advantage of Winsorized PU-CSBoost over PU-CSBoost across all evaluation metrics in the ULB data set. Nevertheless, compared to PU-CSBoost, Winsorized PU-CSBoost drops in expected savings while substantially increasing precision and F1-score in the BAN data set. Despite the drop in performance, the expected savings achieved by Winsorized PU-CSBoost is still higher than the rest of the techniques. This suggests that PU-CSBoost exploits hidden fraudsters with an outlier amount, which boosts expected savings. Based on the experimental results, winsorization offers a strategy to increase the robustness of PU-CSBoost in standard evaluation metrics. Depending on the presence of outliers in the transaction's amount distribution, winsorization can moderately decrease or slightly increase the expected savings.

Table 6: Recall & Precision (%) across flip ratios for BAN data set when $c_f = 10$

	Recall			Precision		
Models Flip Ratio	25%	50%	75%	25%	50%	75%
PU-CSBoost	47.4 ± 3.6	46.5 ± 3.1	37.4 ± 4.1	63.2 ± 9.2	41.0 ± 11.3	35.9 ± 16.6
Winsorized PU-CSBoost	49.6 ± 3.5	47.0 ± 4.1	37.6 ± 3.9	80.0 ± 11.5	75.4 ± 14.1	74.7 ± 12.4
CS-Boost	49.9 ± 3.8	44.9 ± 3.2	28.9 ± 5.2	68.6 ± 19.7	**81.5 ± 9.0**	**82.4 ± 5.1**
Imbalanced nnPU	**72.7 ± 4.6**	**67.8 ± 2.9**	**55.5 ± 9.0**	27.7 ± 16.0	27.2 ± 9.7	43.6 ± 11.7
CS-PU	52.5 ± 7.4	52.0 ± 8.5	45.3 ± 6.7	77.7 ± 15.5	67.3 ± 21.2	62.1 ± 17.5
XGBoost	57.5 ± 2.1	54.1 ± 2.9	45.7 ± 4.5	**85.0 ± 3.6**	75.9 ± 6.6	69.4 ± 4.6

Best performing model is in **bold and underlined** in each column.

Table 7: F1-score & Expected Savings (%) across flip ratios for BAN data set when $c_f = 10$

	F1-score			Expected Savings		
Models Flip Ratio	25%	50%	75%	25%	50%	75%
PU-CSBoost	53.6 ± 3.3	42.6 ± 4.6	34.9 ± 6.4	**45.5 ± 3.5**	**47.8 ± 4.7**	**40.5 ± 6.0**
Winsorized PU-CSBoost	60.5 ± 2.3	56.9 ± 2.5	49.4 ± 3.6	42.8 ± 3.5	41.0 ± 1.9	32.7 ± 3.2
CS-Boost	56.0 ± 7.7	57.5 ± 1.9	42.4 ± 5.5	33.3 ± 4.1	27.6 ± 3.3	12.6 ± 2.8
Imbalanced nnPU	37.5 ± 11.9	37.9 ± 6.4	46.9 ± 4.6	22.5 ± 2.0	14.6 ± 1.8	6.7 ± 1.2
CS-PU	60.9 ± 5.9	54.7 ± 12.8	50.1 ± 7.2	26.7 ± 2.6	22.2 ± 3.0	16.8 ± 4.4
XGBoost	**68.5 ± 1.1**	**62.9 ± 2.1**	**54.9 ± 3.1**	23.2 ± 2.0	15.1 ± 1.8	7.2 ± 1.5

Best performing model is in **bold and underlined** in each column.

6. Conclusions

Motivated by the issue of hidden fraudsters, we introduce an instance-dependent cost-sensitive technique that can natively learn from PU data: PU-CSBoost. Similar to CS-Boost, PU-CSBoost minimizes the financial loss due to fraud; unlike CSBoost, PU-CSBoost incorporates the probability of being a hidden fraudster into the model construction to enable PU learning. Based on our empirical analysis, PU-CSBoost achieves the best expected savings under all experimental settings, surpassing CSBoost. Additionally, we show that winsorization on the transaction amounts can limit the effect of outliers and improve the PU-CSBoost's robustness in cost-insensitive metrics, particularly in precision. Moreover, we emphasize that CSBoost quickly deteriorates its cost-sensitive performance with an increasing presence of hidden fraudsters.

There are several potential research directions for future work. In this work, we estimate the probability of being a hidden fraudster by assuming that labeled positives represent a random subset of the positive class. This assumption in the PU learning literature corresponds to being selected completely at random (SCAR). The SCAR assumption, however, fails to describe how successful fraudsters go undetected: for instance, hidden fraudsters might concentrate close to the negative class. Hence, we could explore other methods to estimate the probability of being a hidden fraudster to accommodate more realistic assumptions. Another line of work relates to the effect of extreme amounts on PU-CSBoost. Again, we could experiment with more strategies that strengthen the technique's robustness.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Alejandro Correa Bahnsen, Djamila Aouada, and Björn Ottersten. Example-dependent cost-sensitive decision trees. *Expert Systems with Applications*, 42(19):6609–6619, 2015.

Jessa Bekker and Jesse Davis. Estimating the class prior in positive and unlabeled data through decision tree induction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: a survey. *Machine Learning*, 109:719–760, 2020. ISSN 0885-6125.

Richard J Bolton and David J Hand. Statistical fraud detection: A review. *Statistical science*, 17(3):235–255, 2002.

Louis Caron and Georges Dionne. Insurance fraud estimation: more evidence from the quebec automobile insurance industry. pages 175–182, 1999.

Philip K Chan. Sjs 2001. toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168.

Nitesh V Chawla, David A Cieslak, Lawrence O Hall, and Ajay Joshi. Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery*, 17(2):225–252, 2008.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Xiuhua Chen, Chen Gong, and Jian Yang. Cost-sensitive positive and unlabeled learning. *Information Sciences*, 558:229–245, 2021.

Xuxi Chen, Wuyang Chen, Tianlong Chen, Ye Yuan, Chen Gong, Kewei Chen, and Zhangyang Wang. Self-PU: Self boosted and calibrated positive-unlabeled training. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1510–1519. PMLR, 2020.

Andrea Dal Pozzolo, Olivier Caelen, Yann-Aël Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, 41(10):4915–4928, 2014. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2014.02.026.

Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164, 1999.

Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *International conference on machine learning*, pages 1386–1394. PMLR, 2015.

Marthinus Christoffel Du Plessis and Masashi Sugiyama. Class prior estimation from positive and unlabeled data. *IEICE TRANSACTIONS on Information and Systems*, 97(5): 1358–1362, 2014.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, 2008.

European Central Bank. Report on sixth report on card fraud. Technical report, European Central Bank, 2020.

Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*, volume 11. Springer, 2018.

David J Hand, Christopher Whitrow, Niall M Adams, Piotr Juszczak, and Dave Weston. Performance criteria for plastic card fraud detection tools. *Journal of the Operational Research Society*, 59(7):956–962, 2008.

Fengxiang He, Tongliang Liu, Geoffrey I Webb, and Dacheng Tao. Instance-dependent pu learning by bayesian optimal relabeling. *arXiv preprint arXiv:1808.02180*, 2018.

Sebastiaan Höppner, Eugen Stripling, Bart Baesens, Seppe vanden Broucke, and Tim Verdonck. Profit driven decision trees for churn prediction. *European journal of operational research*, 284(3):920–933, 2020.

Sebastiaan Höppner, Bart Baesens, Wouter Verbeke, and Tim Verdonck. Instance-dependent cost-sensitive learning for detecting transfer fraud. *European Journal of Operational Research*, 2022.

Liwei Jiang, Dan Li, Qisheng Wang, Shuai Wang, and Songtao Wang. Improving positive unlabeled learning: Practical aul estimation and new training method for extremely imbalanced data sets. *arXiv preprint arXiv:2004.09820*, 2020.

Ryuichi Kiryo, Gang Niu, Marthinus C du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. *arXiv preprint arXiv:1703.00593*, 2017.

Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14:554–562, 2014.

Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pages 448–455, 2003.

MLG. Credit card fraud version 3, 2018. URL https://www.kaggle.com/mlg-ulb/creditcardfraud.

Fantine Mordelet and J-P Vert. A bagging svm to learn from positive and unlabeled examples. *Pattern Recognition Letters*, 37:201–209, 2014.

Eric WT Ngai, Yong Hu, Yiu Hing Wong, Yijun Chen, and Xin 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.

Harish G. Ramaswamy, Clayton Scott, and Ambuj Tewari. Mixture proportion estimation via kernel embeddings of distributions. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2052–2060. JMLR.org, 2016.

Elvezio M Ronchetti and Peter J Huber. *Robust statistics*. John Wiley & Sons, 2009.

Yusuf Sahin, Serol Bulkan, and Ekrem Duman. A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15):5916–5923, 2013.

Tomoya Sakai, Gang Niu, and Masashi Sugiyama. Semi-supervised auc optimization based on positive-unlabeled learning. *Machine Learning*, 107(4):767–794, 2018.

Guangxin Su, Weitong Chen, and Miao Xu. Positive-unlabeled learning from imbalanced data. In *International Joint Conferences on Artificial Intelligence IJCAI*, pages 2995–3001, Montreal, 2021. ijcai.org.

Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48, 2015.

Toon Vanderschueren, Tim Verdonck, Bart Baesens, and Wouter Verbeke. Predict-then-optimize or predict-and-optimize? an empirical evaluation of cost-sensitive learning strategies. *Information Sciences*, 594:400–415, 2022.

Lovro Šubelj, Štefan Furlan, and Marko Bajec. An expert system for detecting automobile insurance fraud using social network analysis. *Expert Systems With Applications*, 38(1): 1039–1052, 2011. ISSN 0957-4174.

Christopher Whitrow, David J Hand, Piotr Juszczak, David Weston, and Niall M Adams. Transaction aggregation as a strategy for credit card fraud detection. *Data mining and knowledge discovery*, 18(1):30–55, 2009.

Zhiang Wu, Jie Cao, Yaqiong Wang, Youquan Wang, Lu Zhang, and Junjie Wu. hpsd: a hybrid pu-learning-based spammer detection model for product reviews. *IEEE transactions on cybernetics*, 50(4):1595–1606, 2018.

Yuri Zelenkov. Example-dependent cost-sensitive adaptive boosting. *Expert Systems with Applications*, 135:71–82, 2019.