A replication of a cost-sensitive decision tree approach for fraud detection

Kristófer Reynisson
School of Computer Science
Reykjavík University

***Abstract*—Payment card fraud poses a problem with widespread implications. However, the class imbalance between genuine and fraudulent transactions presents a challenge to traditional learning methods. Cost-based methods address this problem by assigning different costs to the misclassification of each class. Sahin et al. [7] define a cost-sensitive decision tree algorithm, and this study attempts to both replicate and build on their findings. The results presented here do not support the hypothesis that the cost-sensitive algorithm provides an improvement over traditional decision tree algorithms. However, the results do suggest that different methods of prioritizing alerted transactions can lead to performance improvements over some measures.**

I. Introduction

Fueled by the growth in cashless payments, payment card fraud is becoming an increasingly relevant problem to machine learning communities. For cardholders, payment card fraud is both a stressful and time-consuming experience. For merchants, the costs associated with fraud often stack up to far exceed the price of their products, with every $1[1] of chargebacks costing an extra $2.17 to e-commerce businesses [5]. That is in addition to the potential impact on brand and customer loyalty. Moreover, society as a whole also loses out. The total economic impact of payment card fraud was forecast to be $416 bn in 2017 [6]. A common misconception is that payment card fraud is committed by bedroom hackers. However, the reality is far bleaker as the proceeds of payment card fraud are known to finance terrorism, arms, and drug crime [6].

When it comes to fraud detection, research has failed to have a significant impact on practical applications [6]. One of the reasons for this, as identified by Ryman-Tubb et al. [6], is that there is an insignificant industry incentive to improve, as the financial industry has long looked at fraud as a cost of doing business. Moreover, academic work in this area is difficult and lacks both funding and data philanthropy [6]. Dal Pozzolo et al. [3] argue that payment card fraud presents a particularly challenging problem from a learning perspective due to three main factors. First, the *class imbalance* between genuine and fraudulent transitions presents a challenge to traditional learning methods that fail to yield effective classifiers for detecting the minority class. Second, consumers change their spending patterns, and fraudsters evolve their strategy over time. As a result, the statistical properties of transactions change over time. This is defined as *concept drift*. Last is a problem called *verification latency*, which is defined as the delay in feedback to a *fraud detection system* (FDS). The combination of concept drift and verification latency compound the effects of each problem on the effectiveness of FDSs, but this is something Dal Pozzolo et al. [3] argue has not been sufficiently addressed in previous studies. Furthermore, research in this area often fails to report practically relevant measures when assessing model performance [3, 7].

[1]A prefix of $ signifies values in US Dollar

II. Background

The following section provides background information about payment card fraud and the systems employed in industry for its detection. Moreover, important notation is also defined and explained in order to provide clarity for later sections.

A. Categories of Payment Card Fraud

Van Vlasselaer et al. [8] describe payment card fraud as falling into one of four categories. *Application fraud* occurs when a fraudster obtains a payment card using false information. *Lost or stolen card fraud* is, as the name states, when a card is lost or stolen, and then used to pay for a good or service by someone other than the cardholder. *Card not present fraud* occurs when a fraudster uses the card details. They can either do so in a physical store at a point of sale (POS) system or, more commonly, online at an e-commerce business. The last category of payment card fraud is *counterfeit card fraud*, which is when the details of a payment card, typically details on the magnetic strip, are copied to create a counterfeit physical card. The counterfeit is then used by a fraudster to pay at a POS system. This study will focus on detecting card not present fraud and counterfeit card fraud.

B. The Fraud Detection Process

Payment card fraud detection can be defined as the process of determining the legitimacy of payment card transactions. In particular, such processes attempt to learn and detect specific sequences of operations performed by fraudsters while carrying out payment card fraud. These sequences are referred to as *fraud vectors*, and are known to follow certain patterns [6]. Although fraud detection is carried out by multiple stakeholders within the payment industry, this study focuses on the fraud detection process at card issuers.

The fraud detection process at a card issuer involves actively monitoring the stream of transactions made by its cardholders.

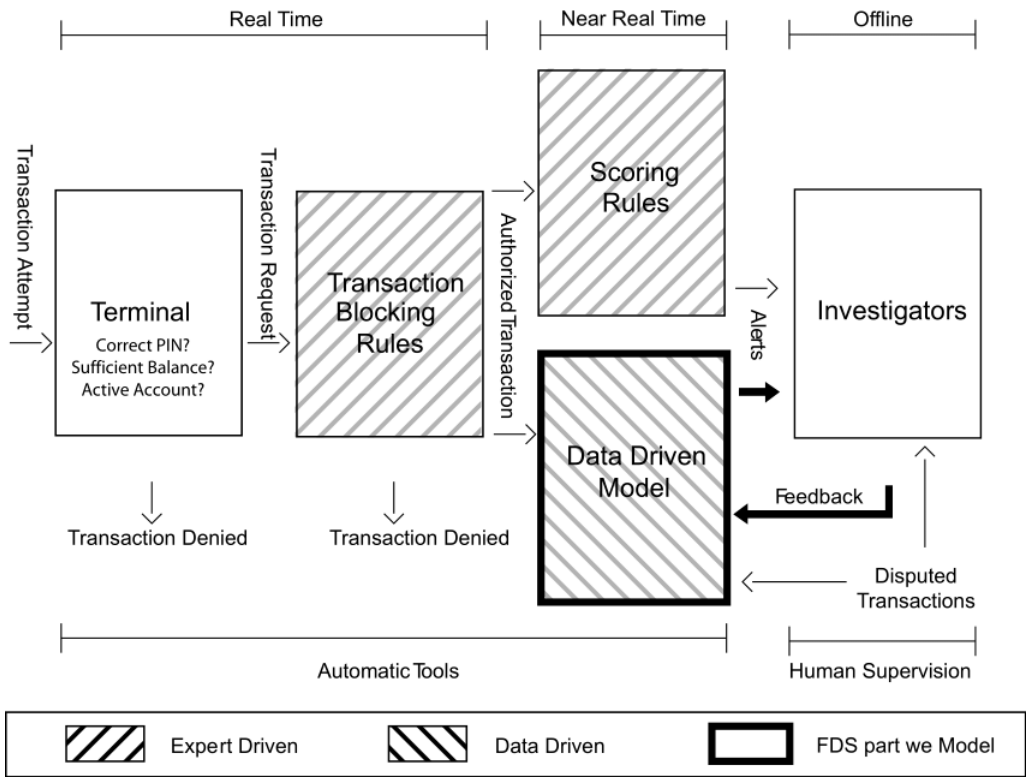

Fig. 1: A FDS Scheme. Source: [3]

When a cardholder attempts to pay with their card, the merchant accepting the payment sends all relevant details to the card's issuer. Upon receiving these details, the issuer runs the details through their FDS in an attempt to determine the nature of the transaction. Dal Pozzolo et al. [3] describe the general structure of such systems and provide the scheme shown in Fig. 1. The following is a description of the stages in Fig. 1, based on the descriptions of such systems in [3, 8].

At the top of Fig 1 is a scale indicating the time frame within which the stages if the FDS are executed. Since the terminal check and transaction blocking rules have to be executed in real-time, they are comprised of simple binary checks. The terminal check includes simple rules, such as if the card has sufficient balance to cover the bill amount of the transaction. The transaction blocking rules differ from terminal checks in that they are set by the *fraud monitoring team* (FMT) and evolve more rapidly. These rules are used to block transactions that exhibit characteristics of known fraud vectors and thus have a high likelihood of being fraudulent. Due to the existence of concept drift, the FMT monitors the effectiveness of these rules and adjusts them to reflect the evolution in fraud vectors over time. An example of a potential transaction blocking rule is to reject multiple e-commerce transactions made within a short amount of time and without strong customer authentication. If a transaction follows all of the rules defined by both the terminal check or the transaction blocking rules, the transaction is denied.

A transaction that passes both the terminal test and the transaction blocking rules becomes authorized, at which point the customer usually receives the product that the transaction paid for. Authorized transactions are then run through two additional tools in near real-time. One of these tools are the scoring rules. Similar to the transaction blocking rules, the scoring rules are dynamically managed binary checks set by the FMT to reflect known fraud vectors. However, these checks occur once the transaction has been authorized and therefore generate alerts rather than denying the transaction. One of the reasons for separating the two sets of rules is that issuers generally do not want to deny too many of their cardholders' transactions. Therefore, the scoring rules usually reflect patterns that do not have as high a correlation with being fraudulent as the transaction blocking rules. Whereas the checks in the three previously described are defined by humans, the data-driven model is typically a learned model and as such, not defined by humans. This means that the data-driven model complements the other parts of the FDS by generating alerts based on previously unknown fraud vectors. This study will examine the effectiveness of a cost-sensitive decision tree algorithm in operating as the data-driven model within an issuer's FDS.

Alerts raised by the scoring rules and the data-driven model are reviewed by investigators who produce feedback by marking the alerted transactions as either genuine or fraudulent. To determine the nature of a transaction, the investigators either rely on domain expertise or contact the cardholder directly. Fraudulent transactions that are missed by the FDS can be disputed by the cardholders when they notice them on their account statements. Once a sufficient time passes from the date of a transaction, if it has not yet be marked as fraudulent, then it is considered to be genuine. Note that the issue of verification

latency stems from the fact that not all transactions are reviewed, and that several days can pass between a transaction being authorized and a cardholder disputing it. For a detailed discussion on the effects of verification latency on an FDS and methods to mitigate them see Dal Pozzolo et al. [3]

In practice, a limit is usually put on the number of alerts that a FDS can generate [3, 6, 7]. This is not only because the human investigator cannot reliably check every transaction that enters the system, but also because investigators tend to develop a mistrust for the information generated by the FMS if it raises too many false alerts [6]. Thus, the number of investigators limits the number of transactions that can be alerted over a given time period.

C. Notation

For readability purposes, we will now define notation inspired by Dal Pozzolo et al. [3]. This notation will be used throughout the study when referring to key concepts in payment card fraud detection and machine learning. The set of transactions T consists of two disjoint subsets, F, the set of fraudulent transactions, and G, the set of genuine transactions, such that $T = F \cup G$. Given a classifier $\mathcal{K}$, the probability that a transaction x is fraudulent based on $\mathcal{K}$ will be expressed as $\mathcal{P}_{\mathcal{K}}(+|x)$. This is also referred to as the posterior of $\mathcal{P}_{\mathcal{K}}$. The limit on the number of transactions that investigators can check given a set of transactions T is defined as an integer $k > 0$. The set of alerts generated by $\mathcal{K}$ for a given T, are the $k-$most risky alerts defined as

$$A = \{x|r(x) \leq k \text{ and } x \in T) \tag{1}$$

where $r(x) \in 1, 2, ..., |T|$ is the rank of transaction x, with rank 1 being the rank of the transaction most likely to be fraudulent based on $\mathcal{P}_{\mathcal{K}}$.

By convention, the class of fraudulent transactions is referred to as the positive class and the class of genuine transactions as the negative class. When referring to the performance of a classifier $\mathcal{K}$, a true positive (TP) is a fraudulent transaction $f \in F$ such that $f \in A$. Similarly, false negatives (FN) are fraudulent transactions $f' \in F, f' \notin A$, and true negatives (TN) and false positives (FP) are transactions g, g' such that $g \notin F, g \notin A$ and $g' \notin F, g' \in A$ respectively.

III. Method

In their work, Sahin et al. [7] define a new cost-sensitive decision tree algorithm for fraud detection. In this, we attempt to replicate and build on the findings in [7]. Therefore, the method described in this section replicates the method employed by Sahin et al. [7] where possible. However, some parts of their method are not fully described. Consequently, parts of the method followed in this study have to be defined by its author. This is highlighted where it occurs and done in a way that tries to resemble what we assume was done in Sahin et al. [7].

A. Feature Extraction

A common approach for learning in payment card fraud detection is to pre-process the transaction data to reflect the cardholder's spending behavior rather than raw transaction details. Due to privacy constraints set by their data provider, Sahin et al. [7] cannot adequately describe the input variables used to train their models. However, they explain that the variables in their data describe card usage profiles, and are derived from both the transaction itself and past transaction history of the card. Moreover, they divulge that all their variables fall under one of the following five variable types: all transaction statistics, merchant type statistics, time-based amount statistics, and time-based number of transaction statistics.

In order to make the input variables used in this study comparable to the ones used in [7], *intrinsic feature extraction* based on work by Van Vlasselaer et al. [8] was employed. The variables extracted by this method are split up into several variable categories that try to capture the card usage profile. For each transaction, the following variables are extracted. *Recency* variables describe the amount of time that has passed since the cardholder's last transaction. Similarly, *Frequency* and *Monetary Value* values capture the total number of transactions and the average amount of those transactions, respectively. These variables are then calculated over three different time intervals, the last hour, the last day, and the last week. Moreover, variables are calculated for five different levels of aggregation. For a given transaction, the levels of aggregation can be described as at the same merchant, at the same merchant category, in the same country, in the same currency, and globally over all transactions on the same card.

Van Vlasselaer et al. [8] provide a fourth category of variables, *event occurrence*, which describes whether the transaction is the cardholder's first within a given time frame and aggregation level. This category was not used in this study. The reason for this is that the same information is already captured in the frequency variables. If a given transaction is the cardholder's first transaction within some level of aggregation and time frame, then that information provided by the fact that the corresponding frequency variable will have a value of zero.

In order to provide the model with information about general spending patterns over all cards, the average monetary value over all transactions made by all cards in the past week is calculated at both a global and merchant level. Moreover, the proportion of fraudulent transactions at the same merchant and in the same country is given in order to provide their respective sensitivity to fraud. This differs slightly from Van Vlasselaer et al. [8] implementation as they provide dummy variables for the location of the transaction (Belgium, EU, Rest of the World) as well as dummy variables for the merchant category risk level (low, medium, high). However, they do not provide a method for splitting merchant categories into these levels, and we believe that providing the proportion of fraud is more informative and allows the models to learn where to partition these sensitivities. A summary of these variable types and aggregation levels is given in Table I.

TABLE I: Description of input features

Variable	Description
Amount	Amount of the transaction
Recency	Time passed since transaction
MC	at the same merchant
MC Category	at the same merchant category
Global	across all transactions
Country	in the same country
Currency	with the same currency
Frequency	Total number of transactions
MC	at the same merchant
MC Category	at the same merchant category
Global	across all transactions
Country	in the same country
Currency	with the same currency
Monetary Value	Average amount of transactions
MC	at the same merchant
MC Category	at the same merchant category
Global	across all transactions
Country	in the same country
Currency	with the same currency
Average Transactions	Average amount of transactions over all cards
MC	at the same merchant
Global	across all transactions
Sensitivity to fraud	Fraction of transactions that are fraudulent
MC Category	at the same merchant category
Country	in the same country

B. Addressing the Class Imbalance

As previously mentioned, the class distribution in payment card transaction data is highly imbalanced towards genuine transactions. The consequence of this imbalance is that it limits the ability of traditional learning algorithms to correctly predict fraudulent transactions [3]. The methods used to address this issue generally fall into one of two categories: *sampling methods* and *cost-based* methods. Sampling methods either involve removing instances of the majority class, referred to as *undersampling*, or replicating instances of the minority class, known as *oversampling*. Unlike sampling methods, which modify the input to classification algorithms, cost-based methods attempt to address the imbalance by assigning different costs to the misclassification of each class. In doing so, cost-based methods modify the output of classification algorithms. Consequently, an implication of this method is determining a representative value for these costs. This is a non-trivial task in the case of payment card fraud detection, as it is unclear what the total cost of misclassifying an individual fraudulent transaction is. One way to estimate this cost depends on the fact that, once a fraudster has made a fraudulent transaction using a given card, he will usually then go on to spend the remaining available usable limit of a credit card [7]. Based on this, Sahin et al. [7] claim that if one could detect a fraudulent transaction, then it would be possible to prevent any subsequent fraudulent transactions on the same card, the cost of which would roughly amount to the available usable limit left on the card. Sahin et al. [7], therefore, define the cost, C_{FP}, of misclassifying a fraudulent transaction as the remaining available usable limit on the card used in the transaction. Furthermore, they also define the cost of a false negative to be some fixed number C_{FN}.

C. Implementation of the Proposed Learning Strategy

The following is a brief discussion of the algorithm defined by Sahin et al. [7]. For a more comprehensive overview, please refer to the original paper. The distinguishing feature of the algorithm is that the splitting criteria are influenced by changes in the individual misclassification costs. Sahin et al. [7] propose four different methods for calculating these costs. The first method, CS - Direct Cost, is not influenced by any impurity measure. Instead, the cost of labeling a node as negative is the sum of the available limit on cards associated with all fraudulent transactions at that node (2). Similarly, the cost of labeling a node as positive is the fixed C_{FP} cost multiplied by the number of genuine transactions at that node (3).

$$C_N = \sum_{i=1}^{|F|}(C_{FN_i}) \tag{2}$$

$$C_P = |G| \times C_{FP} \tag{3}$$

The following three methods all depend on a combination of the misclassification costs and a traditional measure of impurity. The cost functions used in these methods are stated in equations (4 - 9) below.

CS - Class Probability:

$$C_N = \left(\frac{|F|}{|T|}\right) \times \left(\sum_{i}^{|F|}(C_{FN_i})\right) \tag{4}$$

$$C_P = \left(\frac{|G|}{|T|}\right) \times |G| \times C_{FP} \tag{5}$$

CS - Gini:

$$C_N = \left(\frac{|F|}{|T|}\right)^2 \times \left(\sum_{i}^{|F|}(C_{FN_i})\right) \tag{6}$$

$$C_P = \left(\frac{|G|}{|T|}\right)^2 \times |G| \times C_{FP} \tag{7}$$

CS - Information Gain:

$$C_N = -\log\left(\frac{|G|}{|T|}\right) \times \left(\sum_{i}^{|F|}(C_{FN_i})\right) \tag{8}$$

$$C_P = -\log\left(\frac{|F|}{|T|}\right) \times |G| \times C_{FP} \tag{9}$$

At the start of the algorithm, all instances are assigned to the root node. Then the cost (10) of every possible partition of the training set into subsets, given some attribute test of the input variables, is calculated. If some partition leads to a reduction in costs, then the set is split. This procedure is then repeated to grow the decision tree in a recursive fashion by partitioning the training records into sets with successively lower misclassification costs. If the set of instances at a node

cannot be split such that it leads to a lower cost, then the node becomes a leaf node and is assigned either a positive or a negative label (11).

$$C_{Node} = min(C_N, C_P) \tag{10}$$

$$Label_{Node} = \text{If } (C_N < C_P) \text{ then } N, \text{ else } P \tag{11}$$

Another essential feature of the algorithm by Sahin et al. [7] is that the probability assigned to a given class at a leaf node is defined by the misclassification costs and not the relative frequencies of training instances that ended up at that note. This results in the following probability functions for cost-sensitive classifier $\mathcal{K}$

$$\mathcal{P}_{\mathcal{K}}(-|x_n) = \frac{(C_P)_n}{(C_N)_n + (C_P)_n} \tag{12}$$

$$\mathcal{P}_{\mathcal{K}}(+|x_n) = \frac{(C_N)_n}{(C_N)_n + (C_P)_n} \tag{13}$$

where x_n is a transaction which ends up at leaf node n in $\mathcal{K}$, and $(C_P)_n$, $(C_N)_n$ are the costs of labeling the n as positive and negative respectively.

D. Assessment Criteria

As mentioned above, this study attempts to both replicate and build on the findings of Sahin et al. [7]. The assessment criteria associated with these two objectives are described in the following section.

1) The first objective is to attempt to replicate the findings of Sahin et al. [7]. In particular, this study checks whether it can reproduce their finding that models built using their cost-sensitive algorithm outperform traditional decision tree algorithms in terms of both *true positive rate* (TPR) and *saved loss rate* (SLR), a measure defined by them as the saved percentage of potential financial loss (14).

$$SLR = \frac{\sum_{j=1}^{k}(C_{FN})_j}{\sum_{i=1}^{f}(C_{FN})_i} \tag{14}$$

where k is the number of alerts, $(C_{FN})_j$ is the cost of fraud $j \in A$, and $(C_{FN})_i$ is the cost of fraud $i \in F$. As discussed above, the number of alerts that can be investigated at any given time is limited due to the availability of investigators. Therefore, studies [3, 7] into payment card fraud detection usually set this number to a fixed value and then assess the performance of their methods on the top k-most risky transactions in the test set. In their study, Sahin et al. [7] claim that the bank which supplied their data could check 8% of all transactions. Thus, they present their findings based on model performance where $k = 0.08 * |T|$. This study will do the same when attempting to replicate their findings.

2) The second objective is to assess the practical applicability of the cost-sensitive algorithm as compared to traditional decision tree algorithms. In order to do so, this study attempts to give the results in a wider practical context. This is due to the fact that, in practice, the value for k varies significantly. For example, the value for k used in [7] differs significantly from the number used by Dal Pozzolo et al. [3], who only allow 100 alerts per day. This is equivalent to roughly 0.06% of all transactions. The reason for this wide range of values for k is that individual issuers have to balance the financial trade-off between FDS performance and the cost of investigators. Therefore, both TPR and SLR will be plotted against a range of thresholds for k. This allows practitioners to understand model performance within the context of their specific needs [6].

IV. Results

A. Data Sets

The original data set contains 9,468,112 transactions made over 90 days in 2017. Included are all authorization attempts for every card within the data set over the time period. The complete set was used to calculate the input features described above. However, once all input variables had been calculated, a number of transactions were removed before the data was used for training and testing the models. One reason for this is that calculations for the available usable limit have to take into account transactions from 30 days prior. Therefore, the transactions from the first 30 days were removed. Moreover, since this study focuses on card not present fraud and counterfeit card fraud, all transactions that could not fall into either of these categories were removed. An example of a transaction that cannot fall into either of these categories is a transaction made with the card's chip and confirmed with its PIN. Furthermore, transactions that were not authorized, and therefore cannot be disputed as fraud were removed. Lastly, inspired by Van Vlasselaer et al. [8], transactions whose bill amount exceeded 500,000 ISK were removed in order to create more stable models. This is due to the fact that although these transactions are anomalies in terms of general consumption patterns, none of them are labeled as fraudulent. One of the reasons for this could be that a common transaction blocking rule to decline transactions whose bill amount exceeds a certain value.

The resulting data set, containing transactions over a period of 60 days, was then divided into two parts. The first part contained transactions from the first 45 days and was further divided into three sets. A training set was created containing all fraudulent transactions in the first 30 days, and an undersample of the genuine transactions from the same period. The genuine transactions were undersampled such that the set contained 9 to 1 genuine to fraud ratio. This mirrors the ratio used in the training set in [7]. The transactions from the remaining 15 days were then put into a validation set. These two sets were then used in a grid search for hyperparameters. A second training set containing the fraudulent transactions from first the training set and the validation set was also created. This also included the genuine transactions from the training set and an undersample of the genuine transactions from the validation set. Training Set II was used when training the final models used to produce the results presented in the next section. The second part of the data, containing transactions from the last

15 days was used as a test set. Detailed information about these sets and the populations from which they were drawn can be seen in Table II.

TABLE II: Data Sets

Set	Class	Record count in population	Record count in set
Training set I	Genuine	2,682,218	5,445
	Fraud	605	605
Verification set	Genuine	1,470,143	1,470,603
	Fraud	460	460
Training set II	Genuine	4,152,361	9,585
	Fraud	1.065	1,065
Test set	Genuine	1,618,055	1,618,055
	Fraud	458	458

B. Classifiers

A grid search was employed for the hyperparameters for each of the cost-sensitive methods. The hyperparameters searched for were the minimum internal node size, or in other words, the minimum number of samples at a node to be considered for splitting, and the fixed C_{FP}. Models were trained on combinations of values for minimum internal node size in the range $(2, ..., 250)$ and C_{FP} in the range $(1, ..., 6000)$. Their performance with regards to SLR was then assessed using the verification set. While conducting the grid search, it became apparent that the models were overfitting to the data, and changing the hyperparameters mentioned above was not having a significant effect. Therefore, feature importance was calculated using scikit learn's [4] Extra Trees Classifier, based on the method described in [2]. The result of this calculation was used to guide ad hoc tests on the validation set to decide on a subset of the original input features that produced the best improvement in SLR across all cost-sensitive methods. Table III lists the input features used when training the cost-sensitive models.

TABLE III: Hyper parameters used when training models using the cost sensitive algorithm

Variable Category	Level of Aggregation	Time period
Amount	NA	NA
Recency	Global	Hour
Recency	Country	Hour
Recency	Currency	Week
Frequency	Country	Week
Monetary Value	Global	Hour
Monetary Value	Global	Week
Average Transactions	Merchant	Week
Average Transactions	Global	Week
Fraud Sensitivity	Merchant Category	NA
Fraud Sensitivity	Country	NA

The hyperparameters that produced the best results were then used when training the model on Training Set II and evaluated based on their performance on the test set. The hyperparameters used when training each model can be found in Table IV.

As in Sahin et al. [7], models were also trained using traditional decision tree algorithms. This included models trained with the C&RT, CHAID, and Exhaustive CHAID algorithms implemented in IBM SPSS Statistics Subscription [1]. However, unlike Sahin et al. [7], no model was trained using the C5.0 algorithm since it was not available in the software's current version. These models where trained using fixed misclassification costs of $C_{FP} = 1$ and $C_{FN} = 5$. Otherwise, the default parameters set in SPSS Statistics Subscription were used when training these models.

TABLE IV: Hyper parameters used when training models using the cost sensitive algorithm

Model	Minimum internal node size	C_{FP}
CS - Direct Cost	60	4500
CS - Class Probability	60	5850
CS - Gini	50	900
CS - Information Gain	50	800

C. Prediction Results

Table V shows model performance over alerts A where A contains the top 8% most risky transactions. Firstly, it is apparent that models trained with the traditional algorithms outperform the cost-sensitive algorithms both in TPR and SLR. This contradicts the findings of Sahin et al. [7]. However, these results do support the findings in [7], in that the classifier trained with the CS - Direct Cost method performs notably worse than all other models. Since the other models were all trained using methods relying on some measure of impurity, this indicates that implementing an impurity measure within a decision tree algorithm facilitates learning in this domain. It is also of note that the classifiers trained using the cost-sensitive algorithm all perform better in terms of SLR than TPR, while the traditional algorithms all produce models that either perform the same or worse in SLR as compared to TPR.

TABLE V: Model performance over A where k = 8% of transactions

Model	TPR	SLR
C&RT	0.87	0.87
CHAID	0.90	0.90
Exhaustive CHAID	0.91	0.90
CS - Direct Cost	0.60	0.56
CS - Class Probability	0.83	0.85
CS - Gini	0.81	0.86
CS - Information Gain	0.81	0.83

As previously mentioned, two essential features of the cost-sensitive algorithm by Sahin et al. [7] are that both the splitting decisions and the probability of fraud assigned to an instance are influenced by misclassification costs. In an attempt to isolate the effect of these two features, the effect of two different methods of ranking the transactions in T were examined. First, to test whether the traditional algorithms can achieve SLR performance similar to the SLR statistics reported for the cost-sensitive methods Sahin et al. [7], an alternative definition of the set of alerts, A_c, was constructed where A_c includes the k-most expensive transactions based on expected cost. This defines A_c as

$$A_c = \{x | r_c(x) \leq k \text{ and } x \in T) \tag{15}$$

where $r_c(x) \in 1, 2, ..., |T|$ is the rank of transaction x within

T, with 1 being the rank of the transaction x_i with the highest expected cost calculated as $(C_{FP})_i \times \mathcal{P}_{\mathcal{K}}(+|x_i)$.

Table VI shows the performance of the models when measured over A_c, where the values in the brackets indicate the change in performance when measured over A. These results indicate that all models gain an increase in SLR when performance is measured over A_c. Moreover, in general, the models trained on the traditional algorithms gain a more substantial increase in SLR, but all models seem to pay for the increase in SLR with a proportionately larger decrease in TPR. There are however, two exceptions to the general observations made thus far. Firstly, CS - Direct Cost gains the most substantial increase in SLR. Secondly, CS - Gini does not show any decrease in TPR.

TABLE VI: Model performance over A_c where k = 8% of transactions

Model	TPR	SLR
C&RT	0.78 (-0.09)	0.92 (+0.05)
CHAID	0.81 (-0.09)	0.96 (+0.06)
Exhaustive CHAID	0.79 (-0.12)	0.95 (+0.05)
CS - Direct Cost	0.38 (-0.22)	0.65 (+0.09)
CS - Class Probability	0.79 (-0.04)	0.88 (+0.03)
CS - Gini	0.81 (0)	0.88 (+0.02)
CS - Information Gain	0.80 (-0.01)	0.85 (+0.02)

An alternative probability function for the fraud probability assigned to a transaction run through the cost-sensitive algorithms was also developed, defined as

$$\mathcal{P}_{\mathcal{K}}(+|x_n) = \frac{|F_n|}{|T_n|} \tag{16}$$

where x_n is a transaction which ends up at leaf node n in $\mathcal{K}$, and F_n and T_n are the sets of fraudulent transactions and all transactions within the training set that ended up at n respectively. Unlike equations (12, 13) defined by Sahin et al. [7] which depend on misclassification costs, this probability function depends only on the proportion of fraudulent transactions within the training set that ended up at a node n. The results of this revised probability calculation is shown in Table VII.

TABLE VII: Cost sensitive model performance over A where k = 8% of transactions and transactions are ranked based on class probability

Model	TPR	SLR
CS - Direct Cost	0.43 (-0.17)	0.44 (-0.12)
CS - Class Probability	0.84 (+0.01)	0.86 (+0.01)
CS - Gini	0.83 (+0.02)	0.87 (+0.01)
CS - Information Gain	0.81 (0)	0.83 (0)

The results in Table VII show a minor increase in both TPR and SLR for all models except for the one trained with the CS - Direct Cost method, which shows a substantial decrease over both measures and the model trained with the CS - Information Gain method which shows no change. This could indicate that the performance of models trained using the cost-sensitive algorithm could be improved by using the alternative probability function given in (16).

In order to give these results a wider practical context, SLR and TPR are plotted at different thresholds for k as a ratio of $|T|$

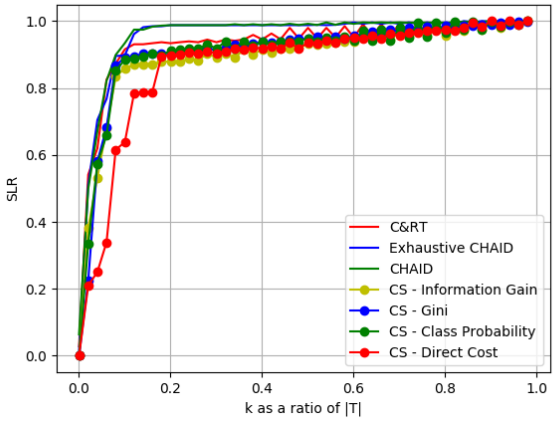

Fig. 2: SLR at different thresholds of k

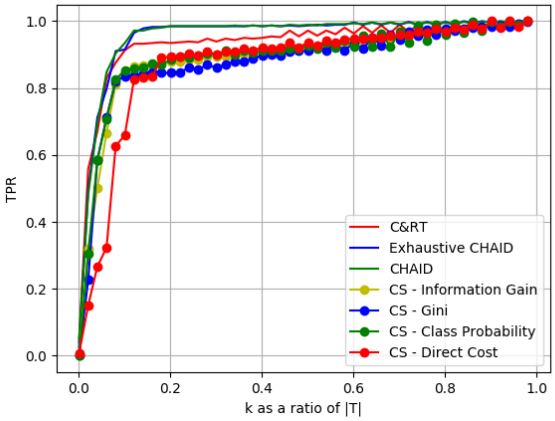

Fig. 3: TPR at different thresholds of k

in Fig. 2 and Fig. 3 respectively. The information portrayed by these figures suggests that the models trained with the traditional algorithms outperform the models trained with the cost-sensitive algorithms at most values for k. However, the difference in performance is larger in terms of TPR than in terms of SLR. Furthermore, the models have the fastest rate of improvement between the values of 0 and 0.16, with the traditional models maintaining the rate of improvement for longer. Consequently, these models show a large improvement over the models trained with the cost-sensitive algorithm past a rate of about 0.1. This difference then slowly decreases as k approaches $|T|$ in size.

V. Conclusion

This study set out with two objectives. *1)* The first objective was to replicate the results of Sahin et al. [7]. The study failed in that objective since it was unable to show that models trained with the cost-sensitive algorithm performed better than traditional algorithms in terms of TPR and SLR. The reasons behind this could be multiple, ranging from different methods of feature extraction to improved hyperparameter selection. Since these results do not provide a test of statistical significance,

we cannot state that they disprove the results in [7]. Therefore, future research could look into replicating these methods in a repeated fashion in an attempt to provide such statistical significance. This could be done while also attempting to identify different ways to improve the performance of the cost-sensitive algorithm, including better methods to avoid overfitting.

Moreover, this study provides two suggestions for improving the performance of models trained with the cost-sensitive algorithm. Firstly, ordering the transactions in $|T|$ by expected cost seems to provide an improvement in SLR while coming at the cost of a loss in TPR. This effect is more pronounced in the models trained using traditional algorithms. The implication of this is that traditional learning methods could be used to achieve an SLR similar to that of cost-sensitive ones. Further research is needed in order to prove this effect. Furthermore, since random forests and neural networks have been shown to have the best performance in practice [6], future research could look into measuring models trained by these methods over A_c.

Secondly, redefining the fraud probability $\mathcal{P}_{\mathcal{K}}(+|x_n)$ assigned to transactions classified by models trained with the cost-sensitive algorithm to depend solely on class distribution at leaf notes, similar to the probability function of traditional classifiers, seems to have the possibility of improving the performance of these models in terms of both TPR and SLR. However, future studies are needed to provide statistical significance to this finding.

2) The study's second objective was to build on the findings in [7] by providing a wider practical context to the results. This was done by plotting both TPR and SLR over a range of thresholds for k. The results indicate that models trained with traditional methods would generally outperform the ones trained with the cost-sensitive algorithm in practice. However, future research could provide several improvements to the method employed here in order to provide a much greater practical relevance. Firstly, as ensemble methods are typically preferred over decision trees in practice, future studies could look at ways of integrating the cost-sensitive decision tree algorithm within an ensemble method. This would allow for a much fairer comparison with other methods used in practice and could, therefore, determine the ultimate practical value of the cost-sensitive algorithm. Furthermore, this research does not address issues raised in [3] caused by verification latency. Training a model on 45 days of data and then testing that model on the subsequent 15 days of data is not representative of model learning within an FDS. Future research should look into implementing the cost-sensitive algorithm within a solution addressing this issue, similar to the two classifier approach presented in [3].

REFERENCES

[1] *IBM SPSS Statistics w/Forecasting and Decision Trees Add-on*. IBM Corporation.

[2] J. Brownlee. Feature selection for machine learning in python, 2019. URL https://machinelearningmastery.com/feature-selection-machine-learning-python/.

[3] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi. Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3784–3797, 2018. doi: 10.1109/tnnls.2017.2736643.

[4] 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.

[5] Ravelin. *Online payment fraud*. Ravelin, 2019. URL https://www.ravelin.com/insights/online-payment-fraud.

[6] N. F. Ryman-Tubb, P. Krause, and W. Garn. How artificial intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark. *Engineering Applications of Artificial Intelligence*, 76:130–157, 2018. doi: 10.1016/j.engappai.2018.07.008.

[7] Y. Sahin, S. Bulkan, and E. Duman. A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15):5916–5923, 2013. doi: 10.1016/j.eswa.2013.05.021.

[8] 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. doi: 10.1016/j.dss.2015.04.013.