False Positives in Credit Card Fraud Detection: Measurement and Mitigation

Florian Wallny
flo.wallny@gmail.com

Abstract

Credit Card Fraud Detection is a classification problem where different types of classification errors cause different costs. Previous works quantified the financial impact of data-driven fraud detection classifiers using a cost-matrix based evaluation approach, however, none of them considered the significant financial impact of false positives. Analysts reported that fraud prediction in e-commerce still has to deal with false positive rates of 30-70%, and many cardholders reduce card usage after being wrongly declined. In our paper, we propose a new method for assessing the cost of false positives and evaluate several state-of-the-art fraud detection classifiers using this method. Further, we investigate the effectiveness of ensemble learning as previous work supposed that a combination of diverse, individual classifiers can improve performance. Our results show that cost-based evaluation yields valuable insights for practitioners and that our ensemble learning strategy indeed cuts fraud cost by almost 30%.

1. Introduction

Credit card fraud is a problem with strong economic impact. Although a lot of effort has gone into the design of secondary verification layers like *3D Secure* or biometric transaction authentication, credit cards are still very susceptible to fraud, especially in e-commerce transactions [1]. Worldwide losses due to credit card fraud have been estimated to $28.65 billion in 2019, and are projected to reach $32.04 billion in 2021 [2]. Credit card fraud is commonly subdivided into Card-Present-Fraud (CP), where the card is physically present at the time of the fraudulent transaction, and Card-Not-Present-Fraud (CNP), where it is not, like with online transactions. According to recent data, 15 out of 100 online transactions are turned down compared to only three out of 100 for in-store transactions [3].

Card issuers (mostly banks) and network providers (e.g. VISA or MasterCard) rely on Machine Learning (ML) for credit card fraud detection (CCFD). State-of-the-art fraud detection systems combine a (confidential) set of transaction blocking rules with a data-driven ML model. This model typically employs a classifier, trained by ML algorithms on examples of genuine and fraudulent transactions, which can assign a suspiciousness score to an incoming transaction. Transactions exceeding a pre-set threshold are declined. [4]. Due to the economic potential, ML for CCFD has received a lot of attention, however, it remains challenging to achieve satisfying performance in practice. In particular, the susceptibility of the models to generate *false declines* (or *false positives)* is a major problem. They cause embarrassing situations for customers, lost revenue for merchants, and administrative overhead for issuers. Especially during the ongoing COVID-19 pandemic and the connected increase in online transactions, false positives became an even bigger problem [3]. Analysts pointed out that false positives might cost more than fraud itself: Analysts estimated that in 2014 in the U.S., $9 billion were lost due to card fraud while $118 billion – 3% of the total U.S. retail market – of sales have been wrongly declined due to the fear of fraud. 26% of the surveyed customers reduced their card usage after being falsely declined, and 32% completely abandoned the card afterwards [5].

Previous work on CCFD did not take into account the adverse financial impact of false positives. Most publications report only standard classification metrics such as accuracy, precision and recall [6,7,8]. Although some attempts have been made to quantify the financial impact of using ML for CCFD, all of them reduce the cost of false positives to a fixed administrative overhead, which heavily underestimates the true financial damage they cause. Our study addresses this gap by asking the following first research question (RQ): *RQ1: How can classifiers for CCFD be evaluated in financial terms while incorporating the cost connected with false positives?*

Using statistical data on card usage from the Euro area, we construct an evaluation function for CCFD models that incorporates the operative cost as well as the potentially lost revenue that false positive cause for card issuers. We then evaluate the most prominent models for CCFD using this evaluation function.

URI: https://hdl.handle.net/10125/79527
978-0-9981331-5-7

Ensemble learning showed to be effective for reducing false positives in other domains [9,10,11]. However, no publication has yet investigated its effectiveness in CCFD using state-of-the-art ML algorithms. Only one recent publication by [12] suggested combining a Random Forest with a Neural Network classifier as the predictions differ in the kind of frauds they detect. Following this call for further research, we also investigate: *RQ2: Can ensemble learning help to mitigate the false positives problem?* We will demonstrate that already simple ensemble learning strategies like majority voting can indeed help to mitigate the false positive problem, even if the participating models are already ensembles like Random Forests.

The remainder of this paper is structured as follows. In Section 2, we introduce the reader to the problem of CCFD and name the most important approaches found during our literature review for solving it. In Section 3, we review existing cost-based CCFD model evaluation strategies. Section 4 introduces the dataset used for training our models and presents the associated feature engineering steps. Section 5 describes the associated models Random Forests, Gradient-Boosted Decision Trees and Neural Networks. Further, we present our model evaluation strategy, including our proposed cost-based evaluation of false positives. Section 6 reports results, while Section 7 discusses theoretical and practical implications as well as limitations of our work.

2. Credit Card Fraud Detection

Credit card fraud can be defined as any illegitimate use of a credit card against the interest of the cardholder, and CCFD is defined as the task of identifying the fraudulent transactions [13]. More formally, given a set of credit card transactions $X = [x_1, \dots, x_N]$, where each transaction $x_i = [x_i^1, x_i^2, \dots, x_i^K]$ contains details like amount, currency, timestamp, etc., CCFD can be formulated as a supervised learning problem [14]. Each transaction has a class label $y_i \in \{0,1\}$ assigned, where $y = 1$ denotes a fraudulent transaction and $y = 0$ a genuine one. The goal of a machine learning algorithm for CCFD is to learn the class-conditional posterior distribution $P(y|X)$ and predict the label $\hat{y}_{N+1}$ of an unseen transaction x_{n+1}.

Seminal publications on algorithmic CCFD exclusively used *Neural Networks* [15 – 18]. Since the mid-2000s, a vast set of models has been applied to solve the CCFD problem.[1] The most prominent ML model for CCFD in research and practice nowadays is *Random Forests* [4,8]. Several studies have reported their superior performance when compared to others [7, 14, 21]. Another effective model for CCFD is *Gradient-Boosted Decision Trees* [22]. Although this algorithm has not yet received broad attention in academic research on CCFD, it is heavily used in competitive machine learning. In the IEEE-CIS Fraud Detection Competition on kaggle[2], for example, all top-ranked teams that reported on their solutions used some implementation of Gradient Boosting.

Breiman [23] firstly introduced Ensemble Learning, the aggregation of multiple classification outcomes into one single prediction as *Bagging*. He proposed an ensemble consisting of individual classifiers built on random subsamples of the training set. Ensemble methods received broad attention with the 2006 awarded Netflix prize, where the winning solution was a classifier ensemble [24]. However, in CCFD, ensembling has received little attention so far. The only work that found a performance boost after ensembling Random Forests with Neural Networks was carried out by [25].

3. Cost-Based Evaluation of Credit Card Fraud Detection Models

Cost-based evaluation is an appropriate method of evaluating a CCFD classifier, because firstly, businesses are usually interested in minimizing the cost accountable to card fraud and therefore need to trade off costs and potential savings when using an automated fraud detection system. Second, it is easier to compare a data-driven model to a human investigators team if both are evaluated in terms of financial savings. Third, CCFD is an instance of *cost-sensitive* learning problems [26], where different classification outcomes have a different financial impact. Table 1 shows a *cost matrix*, which assigns each classification result (true positive, false positive, true negative, false negative) a cost value $(C_{TP}, C_{FP}, C_{TN}, C_{FN})$.

Table 1. Cost Matrix.

	$y = 1$ (Fraud)	$y = 0$ (Genuine)
$\hat{y} = 1$	C_{TP}	C_{FP}
$\hat{y} = 0$	C_{FN}	C_{TN}

Table 2 gives an overview of how previous studies populated this cost matrix. Cost-based evaluation for CCFD has been proposed firstly in the paper by Chan et al (1999) [27]. They assume no cost for true negatives and full cardholder imbursement in case of a missed fraud. If a transaction is wrongly declined or a fraud

[1] [6, 19,20] have carried out systematic literature reviews.

[2] https://www.kaggle.com/c/ieee-fraud-detection/overview

caught, they assume a constant administrative overhead C_a between \$50 and \$100 to incur, because in both cases, the cardholder must be contacted, the transaction must be investigated, a new card must be issued, et cetera. A transaction with a lower amount than C_a is, however, not worth investigating.

Table 2: Cost Matrices of Previous Work.

	C_{TP}	C_{FP}	C_{TN}	C_{FN}
Chan, et al. (1999) [27]	C_a if $T_{Amt} > C_a$, else T_{Amt}	C_a if $T_{Amt} > C_a$, else 0	0	T_{Amt}
Hand, et al. (2008) [28]	C_a	C_a	0	$100C_a$
Bahnsen, et al. (2016) [29]	C_a	C_a	0	T_{Amt}
Wedge, et al. (2018) [32]	0	$0.5 \times 1{,}75\% \times T_{Amt}$	0	T_{Amt}

The study by Hand, et al. (2008) [28] does not use example-dependent costs because of the "fraud breeds fraud" argument – fraudsters getting away with small-scale frauds are likely to continue in the future, and therefore all frauds have to be prevented. To illustrate magnitudes, they assume a false negative cost 100 times of C_a. Bahnsen, et al. (2013, 2015, 2016) [29, 30, 31] assume a constant C_a for both false positives and false negatives and an example-dependent C_{FN} (the respective transaction amount) in their work on cost-sensitive algorithms for CCFD. Wedge, et al. (2018) [32] is the only study so far that uses an example-dependent C_{FP}. They assume that a wrongly declined transaction goes through with 50% probability with the second try. If it doesn't, then the issuer loses a transaction processing fee assessed at 1.75% of the transaction amount. Their evaluation approach, however, doesn't include overheads. We argue that none of the previously explained approaches captures all aspects of the true cost, especially for false positives. Clearly, administrative costs occur with both false positives and false negatives. However, reducing $\boldsymbol{C_{FP}}$ to a fixed administrative overhead ignores the variable cost for the issuer caused by the lost processing fee. Taking a fixed percentage only of the current transaction amount is also too short-sighted.

As the study [5] reported, 32% of cardholders stop using a card after being falsely declined. Therefore, if a cardholder decides to switch the issuer because of too many wrong declines, the card issuer loses not the transaction amount of this transaction, but also all other future transactions that this cardholder could have made. If the issuer charges service fees or a fixed yearly credit card cost, this money is lost as well.

4. Data

Because of the intrinsically private nature of credit card data, publicly available, real-world credit card transaction datasets are scarce. To the best of our knowledge, there are currently only two: The first dataset was released together with the Ph.D. thesis of Dal Pozzolo (2015) [4]. It consists of 284,807 credit card and is available on kaggle[3]. The second dataset was issued as a part of the 2019 IEEE-CIS Fraud Detection competition on kaggle. It contains 590,540 labeled credit card transactions and was extracted from a production fraud detection system. We decided to use the latter dataset in our study because of the larger size and feature set. We don't present a full feature list here for space reasons, but it is available online.[4] Table 3 reports a set of aggregated descriptive metrics ('#' denotes numbers, and '%' percentages).

4.1 Feature Engineering

Because spending behavior varies from cardholder to cardholder, transactions that do not look outstanding from a global point of view might be unusual for a certain cardholder and indicate fraud. It is therefore helpful to add a set of derived features which form a cardholder profile. Here, we will use three feature sets: a baseline feature set containing only original features, an augmented feature set containing engineered features and a reduced feature set that removes correlated and unimportant features. Feature engineering for credit card fraud detection is a widely discussed topic, and many studies use a strategy that aggregates transaction data over a pre-defined timeframe [7,12,14,21,31] in a *Recency-Frequency-Monetary* style. The normal

[3] https://www.kaggle.com/mlg-ulb/creditcardfraud

[4] https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203

cardholder behavior is characterized by the (1) *Recency* or the average time between previous transactions, by the (2) *Frequency* or the average amount spent in a given timeframe and the (3) *Monetary value* or the average amount spent in a given timeframe.

Let $S_{t,id}$ be the set of transactions made by a specific cardholder (named with an id) in the timeframe $t \in \{1h, 3h, 6h, 12h, 18h, 24h, 72h, 168h\}$.

Table 3. Essential Descriptive Metrics.

#Transactions	Timespan	#Features	#Genuine	#Fraudulent
590,540	**6 Months**	**434**	**569,877**	**20,663**
%Genuine	%Fraud	Total Amount	Fraudulent Amount	%Fraudulent Amt
96.5	**3.5**	**$79,738,948.74**	**$3,083,844.86**	**3.87**

Our derived features are (where $TxDT$ is a Timedelta in seconds from a given reference datetime and denotes the point of time when the transaction was made):

- Average timespan ($AvgTsp$) between the transactions in the given timeframe (Recency):

$$AvgTsp = \frac{1}{|S_{id,t} - 1|}\sum_{i=0}^{|S_{id,t}|-1} x_{i+1}^{TxDT} - x_i^{TxDT}$$

- Number of transactions ($Tx\ \ Count$) in the given timeframe (Frequency):

$$Tx_{Count} = |S_{id,t}|$$

- Average transaction amount ($Avg\ \ TxAmt$) in the past t hours for cardholder id (Monetary Value):

$$Avg_{TxAmt} = \frac{1}{|S_{id,t}|}\sum_{i=0}^{|S_{id,t}|} x_i^{TransactionAmt}$$

To calculate the aggregated features, we firstly engineered a cardholder identifier by using adversarial validation[5]. In real-world datasets, this identifier is given by the card number, but in the dataset at hand, the card number has been removed for privacy reasons. Next, we added time-based features based on the $TxDT$ field to track transaction month, day and hour. Finally, we calculated the aggregation values separately for each of the 8 timeframes and obtained 28 engineered features in total, therefore, we have 452 features in the augmented feature set.

4.2 Feature Selection

To remove unimportant features, we use two techniques: correlation analysis and a modified version of backward feature selection (for tree-based models only). For correlation analysis, we identified groups of correlated features ($\rho > 0.9$) and replaced the whole group by one feature. For backward feature selection, we reviewed the feature importances of the tree-based models, and removed zero-importance features as well as the least important 5% of the features. As soon as performance decreased, we stopped eliminating features. This process removed 26 features for Random Forests, 206 features for CatBoost and 17 features for XGBoost

5. Model

This section briefly introduces the individual classifiers Random Forests, Neural Networks, and the two Gradient Boosting variants XGBoost and CatBoost together with the parameterizations we used for training. Subsequently, we will outline the ensembling strategy to combine the predictions of the individual classifiers. Finally, we describe conventional and cost-based metrics to evaluate the models as well as the ensemble.

5.1 Random Forests

As described in Section 2, Random Forests are the industry standard in CCFD. Random Forests were introduced by [34] and are an ensemble method that combines predictions of multiple, de-correlated decision trees by majority vote. We used the implementation and the hyperparameter tuning strategy by scikit-learn [35]. Firstly, we successively increased the number of trees in the forest and did not find any further improvements for more than 1000 trees. Afterwards, we performed a grid search (an exhaustive search over a manually specified subset of the hyperparameter space) over different subsampling ratios and found the best results with training each tree on 20% of the features and 80% of the rows. To address the fact that fraudulent transaction represent only a tiny fraction of all transactions, we used the inverse of the imbalance ratio as class weights.

[5] http://fastml.com/adversarial-validation-part-two/

Random Forests cannot deal with categorical features natively, so we mapped each category to an integer.

5.2 Gradient-Boosted Decision Trees

Gradient-Boosted Decision Trees are another method of ensembling single tree predictions where each tree contributes a small "boost", scaled by a learning rate, to the output [22]. The state-of-the-art implementation of gradient boosting is *XGBoost* ("eXtreme Gradient Boosting") [36]. Because of its performance and scalability, XGBoost has quickly become a dominating solution for a wide range of problems, especially in competitive machine learning. We used the python implementation together with the parameters selected by the winning team of the kaggle competition. They identified a learning rate of $\rho = 0.02$, a row subsampling rate of 0.8 and an ensemble size of 2000 estimators as optimal.[6] We mapped each categorical feature to an integer value since XGBoost cannot handle categorical features natively. Another novel, but already quite popular boosting solution is CatBoost [37], a modified version of the standard boosting algorithm that can handle categorical features natively. It is interesting to see how CatBoost will perform on our dataset, because it contains a lot of categorical features, some of them with a high number of unique values. We had to limit the ensemble size to 1000 because of RAM constraints. We set class weights as with Random Forests because without, performance decreased.

5.3 Neural Networks

Neural Networks (NNs) are a large family of statistical models and learning methods. NNs are a highly active area of machine learning research, and, besides that, have already found a large variety of applications from machine translation over medical imaging to financial analysis. Our Neural Network Architecture was adapted from a successful kaggle submission.[7] It consists of three hidden layers, 256 nodes each, with a learning rate of ρ=0,001 and a weight decay coefficient of $\lambda = 0{,}0005$. Before training, we standardized all numerical features and log-transformed those with large value range. For categorical features, we used One-Hot-Encoding, after reducing the cardinality for high-cardinality features by replacing infrequent categories with a uniform label. We tried to use learned embeddings as alternative, but saw worse results. As further regularization techniques, we use Batch Normalization [38] and Dropout [39] with 0.3 dropping probability. We also tried computations with increased depth of the NN up to ten hidden layers, but didn't see any significant improvements.

5.4 Ensembling Strategy

Inspired by the work of [12], we investigate how strong the true positives of our models overlap to see if they detect different kinds of fraud. If they do so, combining their predictions in an ensemble probably improves the performance. We use the Jaccard Score to measure the classifiers' prediction similarity. Given two classifiers' predictions, it is defined as

$$JaccardScore = \frac{N_{1,1}}{N_{1,0} + N_{0,1} + N_{1,1}}$$

where $N_{i,j}$ is the number of transactions for which classifier 1 predicted class i and classifier 2 predicted class j [40]. For $i = j$, we therefore say that the classifiers *agree*, and *disagree* otherwise. A Jaccard Score of 1 means that both classifiers made identical predictions on all transactions and a Jaccard Score of 0 means that the classifiers did not agree on a single transaction. As we strive for maximum diversity within the ensemble, we suspect that adding a classifier with low prediction similarity to the models already in the ensemble improves the score stronger than adding a classifier with high similarity.

We use majority voting for combining predictions: For a given transaction, the ensemble's prediction is the class that the majority of the models predicted.

5.5 Conventional Evaluation Metrics

We evaluate each model against a set of conventional classification metrics (described in this section) and in financial terms (described below). Firstly, we report Precision and Recall for visualizing the proportion between false positives and false negatives. Like frequently found in the literature, as an overall classification quality metric, we use Area under the Precision-Recall Curve (AUPRC) instead of the popular Area under the Receiver Operating Curve (AUROC). It summarizes the trade-off between Recall and False Positive Rate, while AUPRC summarizes the trade-off between Precision and Recall [41]. Although they look similar, AUPRC is more informative in imbalanced classification settings. [42,43]. As a further standard metric that captures the trade-off between

[6] https://www.kaggle.com/cdeotte/xgb-fraud-with-magic-0-9600

[7] https://www.kaggle.com/c/ieee-fraud-detection/discussion/111476

precision and recall, we report the F1 score (harmonic mean of precision and recall).

To measure the relative improvement or decline ensembling yielded, we do the following: For a given performance metric z, let z_{avg} be the average cross-validated score of the individual classifiers participating in the ensemble and z_{ens} be the cross-validated ensemble score. We then report the relative improvement score

$$\%Change = (z_{ens} - z_{avg})/z_{avg}.$$

5.6 Financial Evaluation

For financial evaluation, we use the cost matrix described in Section 3. Like in previous work, we assume no cost in case of a true negative, and an administrative overhead C_a of \$10 in case of false positives and true positives. If a fraud remains undetected, the lost amount is equal to the transaction amount ($C_{FN} = TransAmt$).

We quantify the additional financial impact of a false positive as follows: From the Statistical Data Warehouse of the ECB, we know that the number of issued credit cards in the Euro area in 2018 was 108,071,580 and the total transaction value made with them was 336,679.541 Mln. € [44, 45]. Therefore, by dividing the total transaction value by the number of cards, we obtain an average yearly transaction value per card of 3,115.34 €. From the study by [5], we know that 32% of wrongly declined customers drop the card. Like in the study by [32], we assume that that the issuer loses a transaction processing fee of 1,75% on this and the future potential transactions of this customer in 50% of the cases. Summarizing, to assess the cost of a false positive, we proceed as follows:

- If a false positive occurs, the transaction passes with probability 50% at a second try. If it does, no cost occurs
- If the transaction is rejected the second time as well, the false positive causes the administrative cost because the cardholder needs to be contacted. In this case, the customer drops his card with a probability of .32.
- If the customer decides to drop the card, the next year's revenue generated by this customer $1.75\% \times 3{,}114.34€ = 54{,}51€$ is lost.

Therefore, on average, each false positive costs the issuer

$C_{FP} = (0.5 \times 10) + (0.5 \times 0.32 \times 0.175 \times 3.115{,}34€) = 15{,}63€$.

As the currency of the transactions in the dataset and in the literature are in USD, we use the dollar equivalent amount of \$18.93[8]. After having derived the cost of a single classification outcome, we can calculate the total cost accountable to credit card fraud like in the study by [31]:

$$TotalFraudCost = \sum_{i=1}^{N} y_i(\hat{y}_i C_{TP} + (1-\hat{y}_i)C_{FN}) + (1-y_i)(\hat{y}_i C_{FP} + (1-\hat{y}_i)C_{TN}),$$

where y_i is the class label and $\hat{y}_i$ is the predicted class label of the transaction x_i [31]. Because it is easier to interpret, we report the financial savings after applying this classifier by comparing it to a status quo where no classifier is used. If a card issuer decides not to use a fraud detection system at all, the lost amount on a given transaction set is equal to the sum of the amounts of the fraudulent transactions, that is

$$C_{NoFDS} = \sum_{i=1}^{N} x_i^{TransAmt} \mid y_i = 1$$

[31]. We then define financial savings that could be achieved after applying this classifier as the difference between the cost when no fraud detection system is used and $FraudCost(f)$:

$$FinancialSavings(f) = C_{NoFDS} - FraudCost(f),$$

We also report savings as percentage of the total transaction amount. For evaluating the ensemble, we evaluate same as defined in Section 5.5.

6. Results

This section presents the results from evaluating the individual classifiers and the respective ensemble. All reported scores are results of a six-fold cross validation with months as folds. We didn't intend to use more folds, as more folds would increase the computational burden and may hurt the model performance as many transactions recur on a regular, monthly basis. We further tried time series cross validation with the same number of folds and had hardly differing results.

Table 4 shows the results of Random Forests and CatBoost, Table 5 of XGBoost and the NN. Firstly, we note that all models performed better than if no classifier was used. Even the worst classifier (CatBoost/Baseline) would still save the card issuer \$140,006.27, which is approximately 0.9% of the total transaction amount. The

[8] We used the exchange rate of 12th of June, 2021 provided by Yahoo Finance.

performance varies heavily for different models and feature sets. Classification quality ranges from .465 AUPRC up to .665 AUPRC. Interestingly, the biggest difference in terms of classification performance is within one model: CatBoost achieves .465 AUPRC without feature engineering, which is the worst result among all models, and .665 after feature augmentation and reduction, which is the best AUPRC seen across all models. In financial terms, the worst model achieved savings of $140,006.27 (CatBoost/Baseline), while the best one (NN/Baseline) achieved savings of $325,111.82. XGBoost and CatBoost achieve similar scores in financial impact while they differ in the kind of errors made: CatBoost yields more false positives and less false negatives (lower precision), XGBoost yields less false positives and more false negatives. This difference reflects in the F1-Score: XGBoost has a consistently higher F1-Score than CatBoost as the number of false positives and false negatives is more balanced.

A higher classification performance score does not always coincide with higher financial savings. For example, savings decrease with Random Forests after feature augmentation, but at the same time, AUPRC increases. The same observation holds for NNs.
NN/Baseline has a Precision (Recall) of .315 (.755), while CatBoost/Reduced has precision (Recall) .276 (.793). Even though the NN detected less fraudulent transactions (lower Recall), its cost accountable to false negatives is lower in absolute terms. This suggests that the NN successfully learned to classify high-value fraudulent transactions correctly because the false negative loss depends on the actual transaction amounts, while the false positive cost is a fixed average amount (see Section 5.6). Some models seem to profit more, some less from feature engineering. For NNs, adding derived features and removing correlated features hurts the predictive power of the model: In comparison to the baseline feature set, NNs on the reduced feature set show a .007 decrease in AUPRC and achieve $23,937.24 (8.2%) fewer savings. In contrast, all tree-based models show a positive response to the feature engineering. The most stunning result is for CatBoost: Without derived features, the model performs worse than all other models.

Adding derived features and removing correlated ones boosts the savings by $131,457.73 (93,8%) and increases AUPRC by 0.2. For XGBoost and Random Forests, the effect is more moderate. Both models show a steady increase in predictive performance after augmentation and reduction. The savings decrease after augmentation, but the decrease is more than offset by the reduction. Quantitatively, Random Forests have $5,761 (1.3%) higher savings and .043 higher AUPRC on the reduced feature set than on the baseline feature set. XGBoost shows with $6,441 (1.9%) increase in savings and .024 increase in AUPRC the same magnitude of response to feature engineering. We now turn to investigate the ensembling results. Figure 1 shows the Jaccard similarity scores between the trained models. We see that, unsurprisingly, most models from the same family have high Jaccard scores over .5. One exception is the CatBoost model with a strongly worse performance, as described above: It shared only .38/.39 prediction similarity with the other CatBoost models.

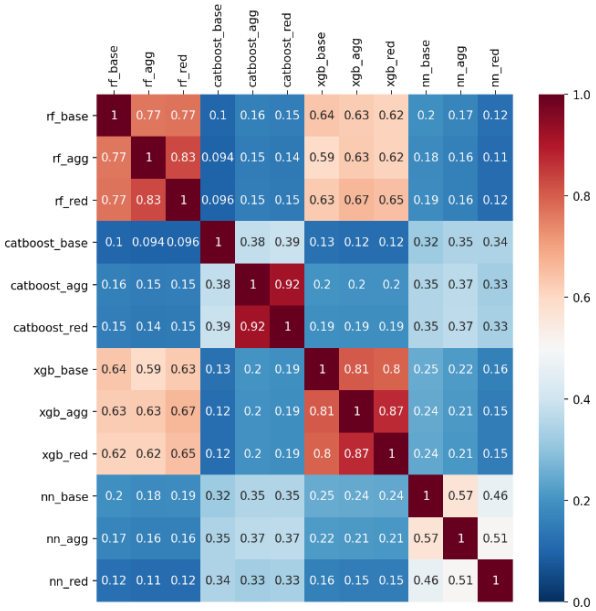

Figure 1. Jaccard prediction similarity between all models.

However, models from different families show a rather low prediction similarity. While the predictions of Random Forests and XGBoost still are relatively similar, CatBoost and NNs agree with other models maximally in 37% of transactions, but mostly in less. Among all possible combinations of our twelve models, the ensemble consisting of CatBoost/Reduced, NN/Baseline and XGBoost/Reduced showed the best result. Table 6 shows the corresponding scores. Interestingly, the ensemble yields benefits, even if some participating models are already ensembles by themselves (Random Forest and Gradient Boosting).

7. Discussion and Conclusion

This study evaluated multiple state-of-the-art ML models and an ensemble consisting of three models for CCFD. In contrast to previous studies, we incorporated both the operational cost and the threatened revenue of card issuers in case of false positives in our cost-based evaluation model. By that,

we answered *RQ1*. The evaluation showed that using NNs yields the highest financial savings, despite not showing the best classification performance by classical means (AUPRC/F1-Score). During experimentation, we also noted that the performance varied massively with the model parameters and with the feature engineering strategy.

Table 4: Cross-validated scores for Random Forests and CatBoost.

Feature Set	Random Forests			CatBoost		
	Baseline	*Augmented*	*Reduced*	*Baseline*	*Augmented*	*Reduced*
Precision	0,68	0,73	0,75	0,166	0,273	0,276
Recall	0,39	0,38	0,39	0,749	0,792	0,793
AUPRC	0,513	0,530	0,556	0,465	0,664	0,665
F1	0,492	0,491	0,508	0,271	0,402	0,404
Fraud Cost	$449.523,19	$459.188,74	$443.762,39	$475.762,52	$347.353,91	$345.304,57
Savings	$167.245,77	$157.580,23	$173.006,58	$140.006,27	$269.415,06	$271.464,00
%TotalTrans	1,05%	0,99%	1,09%	0,88%	1,69%	1,70%

Table 5: Cross-validated scores for XGBoost and the Neural Network.

Feature Set	XGBoost			Neural Networks		
	Baseline	*Augmented*	*Reduced*	*Baseline*	*Augmented*	*Reduced*
Precision	0,5	0,602	0,584	0,315	0,291	0,291
Recall	0,589	0,556	0,577	0,755	0,764	0,762
AUPRC	0,594	0,612	0,618	0,624	0,626	0,617
F1	0,526	0,566	0,562	0,441	0,419	0,416
Fraud Cost	$344.342,60	$346.252,73	$337.900,90	$291.657,15	$301.391,46	$315.594,39
Savings	$272.426,36	$270.516,24	$278.868,08	$325.111,82	$315.377,51	$301.174,58
%TotalTrans	2,16%	2,17%	2,12%	1,83%	1,89%	1,98%

Table 6: Scores of the ensemble consisting of CatBoost/Red, XGBoost/Red and NN/Baseline.

Precision	Recall	F1	AUPRC	Fraud Cost	Savings
0,486 (+24,1%)	**0,735** (+3,8%)	**0,58** (+23,7%)	**0,663** (+4,3%)	**$239,728.38** (−26,2%)	**$377,040.59** (+29.2%)

This highlights the importance of extensive model tuning and feature engineering. Different models differ in the kind of errors made. In this regard, although being superior in savings, NNs/CatBoost generated more false positives than Random Forests and XGBoost. This can be undesirable in a practical setting. Especially when a bank has a shortage of card fraud investigators, they are interested in receiving few but precise suspicious transaction alerts [8]. Which algorithms are appropriate to use in a concrete setting must be decided individually, given the specific resources and requirements a fraud detection system operator has: A card issuing fintech with a small team might tolerate less false positives than an established bank. Our developed cost-based classifier evaluation method can provide valuable insights for both.

We answered *RQ2* by demonstrating that forming an ensemble of multiple models with low prediction similarity can significantly improve performance through a reduction in false positives: Precision increases by almost 25%, compared to the average Precision of the individual models. This increase is reflected in increased savings: By reducing the cost accountable to false positives, our ensemble saves almost 30% more than the average of savings achieved by the individual models. Comparing to other strategies for improving CCFD performance, ensembling has an impact of similar magnitude. The study by [31] observed a 35% increase in saving when adding features that capture the periodic spending patterns of cardholders. [32] found cost reduction potential of approximately 40% when using a deep feature synthesis algorithm (Both studies didn't consider the potentially lost revenue in their evaluation, however).

As with every study, our work has some limitations. First, our quantification of the cost associated with false positives is an underestimation, because a customer might switch to a new card issuer permanently, not only for one year. Further, it is likely that not only false positives have a deteriorating effect on customer experience and therefore cause cost. Also false negatives are likely to cause cost additional to the lost transaction amount. A customer who repeatedly found his card charged by fraudsters, without the interference of his bank, might also consider abandoning that card and switching to a new issuer even if she gets reimbursed. Our study encourages future research on the financial impacts of false positives. Practitioners like banks and card issuers can do such studies best because they have comprehensive data about their customers available. It would be interesting to see if some customers are more likely to drop their card after a false decline and how high the actual loss is, given that a cardholder decided abandoning his card. These losses depend on details of the invididual issuers' business model.

Further, future studies should explore the potential of ensemble learning for CCFD in greater detail. It would be interesting to verify our results on a more extended dataset with fully available features, and to explore larger and/or more diverse ensembles comprising models with different architecture.

8. References

[1] Wang, D.; Chen, B.and Chen, J.: "Credit card fraud detection strategies with consumer incentives", Omega, 88, 2019, pp. 179-95.

[2] The Nilson Report: "The Nilson Report", 1187, https://nilsonreport.com/publication_newsletter_archive_issue.php?issue=1187, 2020.

[3] Crawshaw, James: "COVID-19 Increase in Card Payments is leading to an increase in Adapted Fraud Schemes", PaymentsJournal,, https://www.paymentsjournal.com/covid-19-increase-in-card-payments-is-leading-to-an-increase-in-adapted-fraud-schemes/, 2021

[4] Dal Pozzolo, A.: "Adaptive Machine Learning for Credit Card Fraud Detection". PhD thesis, University of Brussels, 2015.

[5] Javelin Research: "Overcoming False Positives: Saving the Sale and the Customer Relationship", Javelin Research Whitepaper, 2015, https://www.javelinstrategy.com/coverage-area/overcoming-false-positives-saving-sale-and-customer-relationship

[6] Bhattacharyya, S., Jha, S.,Tharakunnel, K. & Westland, J. C.: "Data mining for credit card fraud: A comparative study", Decision Support Systems, 50 (3), 2011, pp.602-13.

[7] Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M. & Baesens, B.: "APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions", Decision Support Systems, 75, 2015, pp. 38-48.

[8] Dal Pozzolo, A., Bontempi, G., Caelen, O., Alippi, C. and Boracchi, G.: "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy", IEEE Transactions on Neural Networks and Learning Systems, 29 (8), 2018, pp. 3784-3797.

[9] Khasawneh, K., Ozsoy, M., Donovick, C., Abu-Ghzaleh, N. and Ponomarev, D.: "Ensemble Learning for Low-Level Hardware-Supported Malware Detection", International Symposium on Recent Advances in Intrusion Detection RAID ,2015, pp. 3-25.

[10] Bejani, M. M. and Ghatee, M.G.: "A context aware system for driving style evaluation by an ensemble learning on smartphone sensors data", Transportation Research 89, 2018, pp. 303-320.

[11] Liu, S., Wang, Y., Zhang, J., Chen, C. and Xiang, Y.: "Adressing the class imbalance problem in Twitter spam detection using ensemble learning", Computers&Security 69, 2017,pp. 35-49.

[12] Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P., He-Guelton, L. and Caelen, Olivier: "Sequence classification for credit-card fraud detection", Expert Systems with Applications 100, 2018, pp. 234-45.

[13] Maes, Sam; Tuyls, Karl; Vanschoenwinkel, Bram & Manderick, Bernard : "Credit Card Fraud Detection. Applying Bayesian and Neural networks", 2002, https://www.researchgate.net/publication/248809471_Credit_Card_Fraud_Detection_Applying_Bayesian_and_Neural_networks

[14] Dal Pozzolo, A., Caelen, O., Bontempi, G. and Waterschoolt, S.: "Learned lessons in credit card fraud detection from a practitioner prespective", *Expert Systems with Applications* 41 (10), 2014, pp. 4915-28.

[15] Ghosh, S. and Reilly, D.: “Credit Card fraud detection with a neural-network”, Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences 3, 1994, pp. 621-30.
[16] Aleskerov, E., Freisleben, B. and Rao, B.: CARDWATCH: a neural network based database mining system for credit card fraud detection”, IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr), 1997.
[17] Dorronsoro, J.R., Ginel, F., Sgnchez, C. and Cruz, C.S.: “Neural fraud detection in credit card operations”, IEEE Transactions on Neural Networks 8 (4), 1997.
[18] Brause, R, Langsdorf, T. and Hepp, M.: “Neural data mining for credit card fraud detection, 11th International Conference on Tools with Artificial Intelligence”, Chicago, USA, 1999.
[19] Priscilla C.V., Prabha D.P. “Credit Card Fraud Detection: A Systematic Review.”, Intelligent Computing Paradigm and Cutting-edge Technologies, 2020.
[20] Delamaire, L., Abdou, H. and Potinon, J.: “Credit card fraud and detection techniques: a review.”, Banks and Bank Systems 4 (2), 2009, pp. 57-68.
[21] Whitrow, C., Hand, D. J, Juszczak, P., Weston, D. and Adams, N. M.: “Transaction aggregation as a strategy for credit card fraud detection, Data Mining and Knowledge Discovery” 18 (1), 2008, pp. 30-55.
[22] Friedman, Jerome H.: “Greedy Function Approximation: A Gradient Boosting Machine”, The Annals of Statistics 29 (5), 2001, pp. 1189-232.
[23] Breiman, L.: Bagging Predictors, Machine Learning 24, 1996, pp. 123-140.
[24] Kuncheva, L.: “Combining Pattern Classifiers: Methods and Algorithms”: Wiley, 2014.
[25] Kim, E., Lee, J., Shin, H., Yang, H. Cho, S., Nam, S., Song, Y., Yoon, J and Kim, J.: “Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning”, Expert Systems With Applications 128, 2019, pp. 214-224.
[26] Elkan, C.: The Foundations of Cost-Sensitive Learning, IJCAI'01: Proceedings of the 17th international joint conference on Artificial intelligence, 2001, pp. 973-978.
[27] Chan, Philip K.; Prodromis, A.; Fan, W. and Stoflo, S. (1999): “Distributed Data Mining in Credit Card Fraud Detection, IEEE Intelligent Systems” 14 (6), 1999, pp. 67-74.
[28] Hand, D. J., Whitrow, C., Adams, N. M., Juszczak, P. and Weston, D.: “Performance criteria for plastic card fraud detection tools”, Journal of the Operational Research Society 59, 2008, pp. 956-62.
[29] Bahnsen, A.C., Stojanovic, A., Aouada, D. and Ottersten, B.: “Cost Senitive Credit Card Fraud Detection Using Bayes Minium Risk”,12th International Conference on Machine Learning and Appliacations, Miami, USA, 2013
[30] Bahnsen, A.C., Aouada, D. and Ottersten, B.: “Ensemble of Example-Dependent Cost-Sensitive Decision Trees”, *arXiv* 1505.04637, 2015.
[31] Bahnsen, A.C., Aouada, D., Stojanovic, A. and Ottersten, B.: “Feature engineering strategies for credit card fraud detection”, Expert Systems with Applications 51, 2016, pp. 134-42.
[32] Wedge, R., Kanter, J. M., Veeramachaneni, K., Rubio, S. M. and Perez, S. I.: “Solving the False Positives Problem in Fraud Prediction Using Automated Feature Engineering”, ECML 2018. In Proceedings of European Conference on Machine Learning, Ireland, 2018.
[33] Molina, L. C., Belanche, L. and Nebot, A.: “Feature selection algorithms: a survey and experimental evaluation”, Proceedings of the 2002 IEEE International Conference on Data Mining, 2002, pp. 306-313.
[34] Breiman, L.: Random Forests, Machine Learning 45, 2001, pp. 5-32.
[35] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., and Thirion, B.: “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research 12, 2011, pp. 2825-30.
[36] Chen, Tianqi and Guestrin, Carlos: “XGBoost”, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[37] Prokorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A. V. & Gulin, A. (2019): CatBoost: unbiased boosting with categorical features, arXiv Preprint 1706.09516v5
[38] Ioffe, S. and Szegedy, C.: ”Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 448-456.
[39] Srivastava, N.; Hinton, G.; Krizhevsky, A., Sutskever, I. & Salakhutdinov, R.: “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research 15 (56), 2014, pp. 1929-58.
[40] Tan, P., Steinbach, M., Karpatne, A., Kumar, V. (2019): “Introduction to Data Mining”, Second Edition: Pearson.
[41] Carcillo, F., Le Borgne, Y. Caelen, O., Kessaci, Y., Oblé, F. and Bontempi, G.: “Combining unsupervised and supervised learning in credit card fraud detection”, Information Sciences 10 (22), 2019, pp. 10-27.
[42] Davis, J. and Goadrich, M.: “The relationship between Precision-Recall and ROC curves”, Proceedings of the 23rd international conference on Machine Learning, 2006, pp. 233-240.
[43] Saito, T. and Rehmsmeier, M.: “The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets”, Plos One 10 (3), 2015.
[44] European Central Bank (2020a): Statistical Data Warehouse: Number of payment cards with a credit function - issued by resident PSPs https://sdw.ecb.europa.eu/quickview.do?SERIES_KEY=169.PSS.A.D0.S101.I13.Z00Z.NT.X0.20.Z0Z.Z
[45] European Central Bank (2020b): Statistical Data Warehouse: Value of payments with cards with a credit function - cards issued by resident PSPs, https://sdw.ecb.europa.eu/quickview.do?SERIES_KEY=169.PSS.A.D0.F000.I13.Z00Z.VT.X0.20.Z01.E