university of groningen

faculty of science and engineering

Optimizing the financial gain for credit card fraud detection systems using machine learning techniques

Rik Vegter

university of groningen / faculty of science and engineering

University of Groningen

Optimizing the economic efficiency for credit card fraud detection systems using machine learning techniques

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
Prof. dr. L.R.B. Schomaker (Artificial Intelligence, University of Groningen)
and
Maruf Dhali (Artificial Intelligence, University of Groningen)

Rik Vegter
s3147495

November 10, 2022

Contents

Abstract

Credit card fraud detection is a worldwide problem that comes with significant financial losses for credit card service companies. Due to the high number of transactions that occur daily, there is a need for the automatization of the classification of these transactions using machine learning techniques. One of the main problems within credit card fraud detection is the imbalance between *fraudulent* and *genuine* transactions. Machine learning algorithms tend to be biased towards the majority class. Therefore data balancing techniques are required to overcome the problem of class imbalance. This thesis provides a comparison of state-of-the-art methods in terms of a new proposed, more realistic financial metric. The financial metric is defined based on expert advice from icscards and the literature of credit card fraud detection. While the financial metrics in literature only take into account a percentual gain and loss of transactions, this metric also takes external costs that credit card service companies make into account. Also, a class imbalance machine learning method used to optimize the decision threshold from another domain (called Ghost) where class imbalance occurs is trained on credit card fraud data. Ghost finds the optimal decision threshold combined with a Random Forest classifier. The results were tested on two commonly used credit card fraud detection data sets and showed that boosting methods perform state-of-the-art results both in terms of the number of correct classifications and in terms of economic efficiency resulting in near-to-perfect classification. The best performance of classification on the commonly used Europe data set containing 284,807 transactions with 0.172% fraud transactions is obtained by XGBoost in terms of AUPRC and resulted in an AUPRC of 0.961 using 5-fold cross-validation. The highest AUROC obtained on the Europe dataset is 0.997 using Ghost. Statistical analysis showed that the differences between the different algorithms used were significant for Economic Efficiency and AUPRC. This study also analyzes the accepted loss in terms of AUROC/AUPRC of classifiers optimized on the Economic Efficiency and where the economic efficiency fits within the precision-recall trade-off. Due to the significant financial loss for false negatives, a slight decrease is found in Economic Efficiency for low precision.

1 Introduction

In the year of 2019, there were approximately 39.6 billion credit card transactions in the United States [2]. While a credit card is a very common payment method in the United States, the USA is also the leading country of fraudulent credit card transactions. 38.6% of all fraudulent credit card transactions worldwide are reported within the U.S [3]. The number of credit card fraud reports has been growing for the last decade. Fig. 1 shows the growth of the losses due to credit card fraud worldwide. This Figure shows that the financial losses due to credit card fraud grew every year from 2010-2019.

Figure 1: Growth of financial losses due to credit card fraud worldwide. Directly taken from [4].

The Federal Trade Commission (FTC) reports that the COVID-19 pandemic might have resulted in an increase in the growth rate of the number of fraud cases. They reported an increase of 44% of identified fraud credit card reports from 2020 to 2021 to support their claim. An obvious disadvantage of credit card fraud is that it involves financial risk. Depending on the type of fraud and the credit card company, a bank will refund the amount of money involved in the fraudulent transaction to the customer. Due to a delay in this refund, a credit card holder might get into an unwanted financial situation where he/she might not be able to fulfil his/her financial duties like paying rent. Secondly, credit card fraud can harm the relationship between customers and a bank [5].

1.1 Credit card fraud detection

To reduce the number of credit card fraud reports, there are two ways to minimize the risk of fraudsters succeeding; credit card fraud detection (CFD from here) and fraud prevention [6]. Ideally, credit card fraud is prevented through cyber security and other fraud prevention systems. A review and a good

starting point about several fraud prevention systems are presented in [7]. Since fraudsters remain to change their strategies, it is difficult for a credit card fraudulent prevention system to prevent all fraud transactions [8]. Therefore, CFD remains a relevant part for credit card companies to reduce their financial risk and enhance the relationship between credit card service companies and customers.

Due to the high number of credit card transactions financial services need to process on a daily basis, it is essential to automate the classification of transactions being a fraud or genuine. To create such a system, systems should be trained on fraud/genuine labeled transactions to detect a pattern for fraudulent transactions in the future without a human having to interfere with the process. This is done using machine learning where a system is trained on a dataset to extract patterns from that dataset. The transactions in these datasets contain features that describe the transactions. Examples of these features are the customer ID, the timestamp of the transaction, the amount of the transaction, or the place of the transaction. Besides the features that describe the transactions, every transaction within the dataset should also come with a label indicating whether it is a fraudulent transaction or not. Machine learning systems can learn patterns within datasets allowing them to make predictions of new data.

1.1.1 Types of credit card fraud

The term *credit card fraud* is a catch-all term for several types of fraud. Several of the most common types of credit card fraud transactions according to Australian Payments Network (APN) [9] are listed below. These examples help the reader to get a feeling about what types of fraud occur.

The APN mentions that the most common type of fraud for Australian merchants is the *card-not-present (CNP)* fraud. These types of transactions do not require a physical credit card and mainly occur on the phone or online. The second most common type of fraud is called *counterfeit/skimming*. Skimming refers to a device copying the details of the magnetic stripe of a credit card. With skimming, a counterfeit can be created which essentially is a duplicate of the original credit card. The third most common type of fraud is when a credit card is *lost or stolen*. If the credit card is not returned, fraudulent transactions can be made. A credit card that is lost or stolen is especially sensitive to fraud since a credit card (usually) does not need a unique pin code only known to the owner. These three types of credit card fraud explain 96.8% of all fraud transactions in Australia with a credit card [9].

1.2 Class imbalance

The datasets used in CFD that machine learning classifiers are trained on are usually extremely imbalanced due to the majority of transactions being *genuine*. An example of such a dataset is the Europe dataset which is publicly available on Kaggle. Only 0.172% of the 284,807 transactions are labeled *fraud*. Most standard machine learning binary classifiers assume a balanced distribution of the two classes [10]. If this assumption is violated the classifier will be biased toward the majority class which means that the classifier will tend to classify a transaction to the majority class. This is an unwanted consequence since the goal of fraud detection is to find fraudulent transactions. As mentioned in a frequently cited review on imbalanced data [11] there are three main approaches to handling imbalance within distributions. The first approach modifies the collection of data points to reduce the imbalance of the distributions. These methods are called *data-level methods*. The second method modifies the algorithms used for the classification of imbalanced datasets. The goal of these so-called *algorithm-level methods* is to reduce the bias towards the majority class (non-fraud transaction). The third approach is a combination of the two approaches mentioned above and are called *hybrid methods*. The background literature section of this thesis will provide a detailed review of each of these

approaches and provides several examples combined with their advantages and disadvantages.

Class imbalance is a machine learning problem that can not only be found within the field of credit card fraud detection. *Medical diagnosis* also faces an imbalance in distributions between sick/healthy patients since disease cases are often underpopulated as compared to healthy populations. Another example is *dangerous behavior detection* where the goal is to find dangerous behavior based on character traits [12]. However due to the low number of samples of dangerous behavior when compared to non-dangerous behavior, the distributions are heavily skewed towards the majority class.

Since class imbalance is a problem among several applications of machine learning, techniques that show state-of-the-art performance are not necessarily known in the literature on other applications. Therefore researchers need to keep track of the state-of-the-art techniques of other applications.

Not only does the imbalance in class distribution play a role in the performance of classification. As mentioned in [13] the sample size and separability also affect the performance. An experimental study found that as the sample size decreases, the error rate due to the class imbalance increases [14]. The relation between the separability and the error rate of classification is obvious. If the characteristic patterns of the classes are different from the opposite class in a binary classification problem, the error rate will decrease. This finding is supported by multiple experimental studies [14, 15]. Since the performance of classification is based on multiple characteristics of data, it is difficult to quantify the proportion class imbalance plays in the decrease performance of classification.

1.3 Decision threshold

Several classification algorithms attach a probability to a data point belonging to a certain class. Examples of such so-called *probabilistic models* are random forest [16], logistic regression [17], and gradient boosting [18]. The definitions given in this subsection are based on the definition given in [19].

A binary classifier is a function that maps instances x from an input space to a target label $Y = \{0,1\}$. A model is a function $m : X \to \mathbb{R}$ that attaches a *score* to a data point. By convention, a high score expresses a strong belief that the data point should be labelled 1 while a low score expresses a strong belief that the data point should be labelled 0. Model $m : X \to [0,1]$ maps a data point x to a score $\hat{p}(1|x)$ which represents the probability of example x belonging to instance 1. An important part of classification is to convert these scores to predictions in the Y domain. This is done with a so-called decision threshold t. Given a score $s = m(x)$, instance x is classified 1 if $s > t$ and 0 otherwise.

For binary classification, by default, the decision threshold is set to $t = 0.5$. However, this threshold might not be optimal. Therefore it needs to be optimized for classification depending on the application. Some applications require a high specificity (proportion of negative classifications) and for some applications, it is important to maximize the sensitivity (proportion positive classifications). As previously mentioned, undetected fraudulent transactions can have an impact on financial risk. Therefore, within CFD sensitivity is an important metric. However, it is undesired to classify *genuine* transactions as *fraud* since it might be time-consuming to manually find out whether these transactions are *genuine* by an expert. It has been shown that as the decision threshold increases, the sensitivity decreases while the specificity increases [20]. Specificity and sensitivity are often referred to as the true negative rate (TNR) and true positive rate (TPR) respectively.

1.4 Metrics

The traditional and perhaps most frequently used performance metric for a machine learning classifier is accuracy. However, this metric is sensitive to the distribution of the classes if they are not equal

[10]. Again, looking at the previously mentioned CFD dataset where only 0.172 % of all transactions are fraud. A classifier that is maximally biased towards the majority class (and classifies every data point as *non-fraud*) will result in an accuracy of $100 - 0.172 = 99.828\%$ which is considered a good accuracy in most applications. Due to the potential financial loss that a credit card service company risks when not detecting these transactions [5] it is important to classify the fraud transactions (minority class) correctly. Therefore the accuracy does not reflect the performance of the model adequately. Precision and recall are more suitable when measuring the performance of models for imbalanced data distributions. Precision refers to how many predicted fraudulent transactions were fraud transactions. This is calculated by Equation 1. The recall is calculated by Equation 2 and refers to the fraction of positives (true fraud transactions) that were classified correctly. Both equations rely on numbers drawn from the confusion matrix. A confusion matrix is a summary of the prediction results. An example of a confusion matrix explaining the different terms in a confusion matrix in terms of CFD is shown in Fig. 2. These terms help to understand the components of the calculation of Eq. 1 and 2.

		Prediction label	
		Fraud	**Non-fraud**
True label	**Fraud**	**True Positives(TP)**: Number of true fraud transactions classified as fraud	**False Negatives (FN):** Number of true fraud transactions classified as non-fraud
	Non-fraud	**False Positives (FP):** Number of true non-fraud transactions classified as fraud	**True Negatives (TN):** Number of non-fraud transactions classified as non-fraud

Figure 2: Confusion matrix with explanation in terms of CFD. Red boxes indicate incorrect classifications. Green boxes indicate correct classifications.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

The precision and the recall represent a trade-off. If the precision increases, the recall decreases and vice versa [21]. This trade-off is visualized in Fig. 3. If only fraudulent transactions with a high probability of being fraudulent are classified as fraud, we will have high precision. However, this also results in some true fraud transactions not detected by the CFD system.

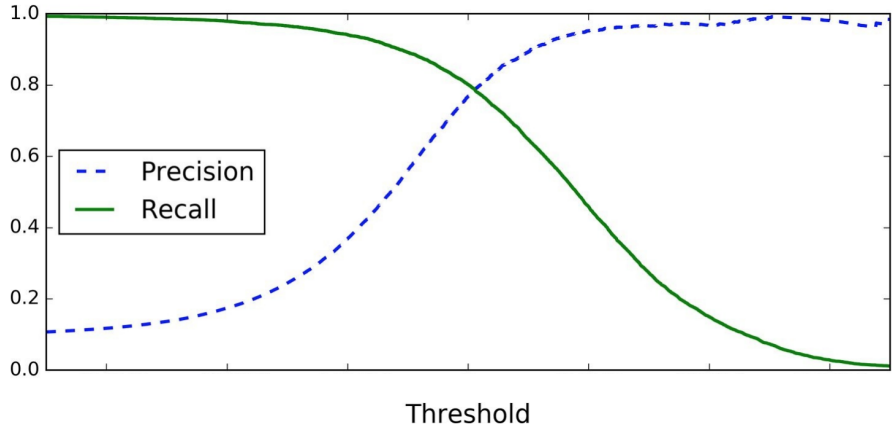

Figure 3: The precision and recall show a trade-off that can be set by the decision threshold. If the precision increases, the recall decreases, and vice versa. Directly taken from [22].

1.4.1 Financial metrics

The ultimate goal of CFD is to detect all fraudulent transactions. However, this is not always a realistic strive. Therefore the metric for measuring the performance of a CFD system might not be optimally measured in terms of the number of correct classifications. Instead calculating a financial gain/loss might be a more suitable metric since it represents the financial impact of using one system over another in a more suitable manner. Several studies in literature already measure performance of CFD systems using a financial metric [23, 24, 25, 26]. However, this is a metric that is not universally defined in comparison with traditional metrics such as accuracy, precision, or recall and therefore makes studies difficult to compare in terms of financial metrics.

A credit card service company processing credit card transactions usually receive a small percentage of every genuine transaction as a financial gain. Secondly, a fraudulent transaction that is classified as genuine results in a financial loss. This results in certain transactions of a greater amount having higher importance to classify correctly due to the potential financial loss/gain that it brings. Keeping these percentages in mind, we can measure the performance of our system in terms of economic efficiency.

If fraudulent transactions are detected, credit card service companies usually contact the credit card holder to confirm whether a transaction is fraudulent. However, it might not be financially worth to do this for every small potential fraudulent transactions. Therefore a threshold might be used to determine whether it is financially worth to manually classify a transaction if it is classified as fraud by the CFD system. [27] mentions that according to expert domains this threshold is a roughly constant and set to 150 Euros (approximately 2 hours of work). No source of expert domain is given by the authors. However, after having consulted with icscards this 2 hours of work is not as constant as the author suggests since it may depend on several factors. Examples of such factors are:

1. Type of fraud (CNP, skim, card stolen, ID fraud)

2. Reaction time of the credit card holder for confirmation of fraudulent activity on the card

3. Professionality of the fraudster; some fraudsters are more difficult to find and trace than other fraudsters.

4. Other factors depending on the case

These factors make it difficult to determine a threshold. However, due to practical reasons for the design of a CFD system, it is inevitable to come to a consensus and set a threshold. Due to confidentially reasons, credit card services cannot share this threshold. To my knowledge, the only known threshold known in the literature is [27].

The percentages that a credit card company receives for transactions and loses when incorrectly classifying fraudulent transactions, combined with the cost of a manual classification by a credit card company expresses an economic efficiency (EE). This EE takes the financial risk into account where the traditional metrics such as accuracy, precision or recall do not.

1.5 Problem statement

As previously mentioned, credit card fraud remains a problem for banks and other related institutions. Credit card fraud detectors are trained on datasets where the problem of imbalance within distributions of the data needs to be tackled. The literature presents several solutions to overcome the problem of class imbalance. However, within the field of CFD, the focus should not lie on optimizing the number of correct classifications but rather on optimizing economic efficiency [23].

Therefore this thesis compares the economic efficiency of state-of-the-art methods for credit card fraud classification. Secondly, as mentioned, state-of-the-art methods from other domains might obtain results outperforming the state-of-the-art CFD systems presented in the literature. This thesis introduces a recently presented technique for the classification of imbalanced data from another application.

1.6 Research Questions

To summarize, this thesis focuses on the following problems:

- This thesis presents proposes a more realistic financial metric called Economic Efficiency (EE).
- This thesis intends to compare the state-of-the-art CFD systems in terms of Economic Efficiency.
- This thesis includes the application of a recently published technique from another domain used for optimizing the decision threshold to overcome the class imbalance problem.

1.7 Thesis Outline

This section introduced several problems concerning CFD. Also, the research question and the specific problem statement are mentioned in this section. Section 2 is the background literature where solutions from the literature on the problems introduced in the introduction are presented. The methods section will discuss the methods used to overcome the problem statement as mentioned in the introduction. In the results section, the outcome of the experiments is discussed and the discussion will put these results in the perspective of the literature and draw conclusions based on the experiments.

2 Background Literature

This section provides a detailed explanation of the different concepts introduced in the literature of machine learning throughout the years for overcoming the problem of class imbalance. Also several classification and performance evaluation concepts are discussed which are frequently used in the field of CFD. With the concepts introduced, several pipelines which show state-of-the-art results in the literature of CFD are discussed.

A pipeline in CFD usually consists of several components. The order of the components is not fixed and depends on the situation. Since there exists a class imbalance in fraud data sets, it is good practice to apply a data balancing technique which reduces the bias towards the majority class. Secondly, a strategy for feature selection can significantly improve the performance of classification. Certain features have low explained variance between the classes and/or are strongly correlated with other features making them redundant for classification. After these two steps, a classification method needs to be applied which groups the *fraud* and *non-fraud* classes based on similarities observed in the data. All components and the theory to understand the techniques will be explained in this section.

2.1 Credit Card Fraud Data sets

Publicly available CFD data sets are spare since they contain personal data. For privacy reasons it is not allowed to make this data publicly available. Fortunately there are several data sets available containing transactions labelled as *fraud* or *genuine*. One of the most frequently used data sets in literature which is publicly available is the Europe data set containing real-world transactions. This data set is available because the numerical variables are made anonymous using principal component analysis which will be explained later in this subsection. There are also synthetic data sets available. These data sets are generated using algorithms which create artificial data sets based on learned distributions from real-world data. The most frequently used example is PaySim [28]. PaySim creates 'mobile simulates mobile money transactions based on an original data set'. Several pipelines have been tested on PaySim [29, 30, 31, 1, 32]. While the usage of synthetic data can be useful for CFD, a major disadvantage is that the data is not directly from the real world. Since these synthetic data sets are created from distributions of real world data sets they can be useful for research purposes.

2.1.1 Principal Component Analysis

Principal component analysis [33] is a method which maps features to principal components. Principal components are dimensions which explain the variance within a data set maximally. The first principal component is the dimension which explains the most variance while the second principal components is the second best dimension explaining the variance within the data. Obtaining these components is a straightforward method. The first step is to standardize the elements in the data. The reason for standardizing is that PCA is sensitive for variances due to the variance being influenced by a high range of variables. The second step is to calculate the covariance matrix which is defined by Eq. 3 where $\Sigma_{i,j} = cov(X_i, X_j)$. The covariance matrix describes the covariance between the variables where element $[i, j]$ is the covariance between variable i and j.

$$\Sigma = \begin{bmatrix} \Sigma_{1,1} & \cdots & \Sigma_{1,n} \\ \cdots & \cdots & \cdots \\ \Sigma_{n,1} & \cdots & \Sigma_{n,n} \end{bmatrix} \tag{3}$$

The principal components are defined by the eigenvectors and eigenvalues. Figure 4 shows an example of PCA. The original dimensions are x_1, x_2, x_3. The principal components are described by the eigenvectors shown in red, green and blue. The green vector is the first principal component, hence this vector explains the most variance.

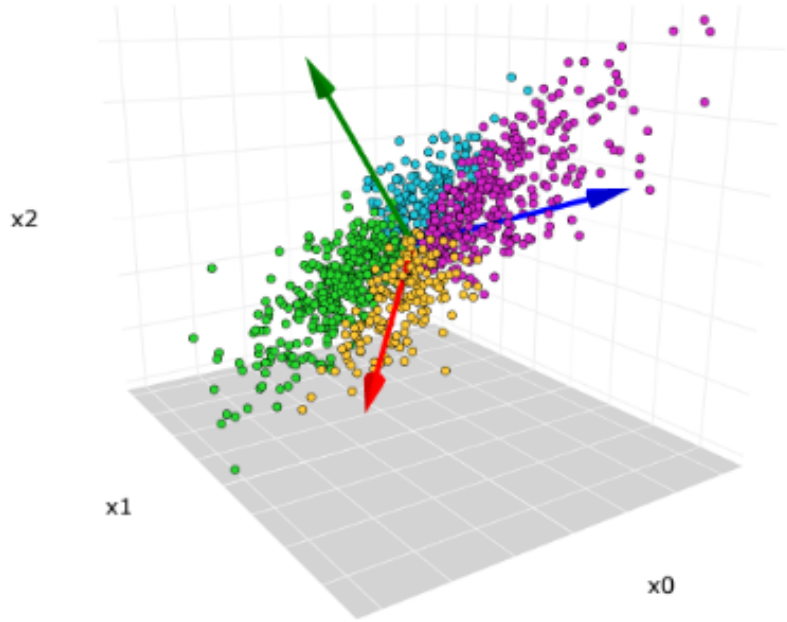

Figure 4: Example of PCA. The green, blue and red vectors describe the principal components. Every data point can be rewritten in terms of the principal components. Directly taken from [34].

PCA is a technique used for several purposes in the field of machine learning. It is mainly used for dimensionality reduction. One can achieve this by selecting the first N principal components as features. This can be achieved by plotting the explained variance per principal component. From this plot it is possible to select a number of principal components explaining a required variance. It is common practice to select the number of features that explain $> 95\%$ of the variance. However, this choice depends on the amount of dimension reduction that is achieved. PCA can also be used to visualize data which can be achieved by plotting the data points on the first two principal components. As previously mentioned, the Europe data set contains real-world transactions. The features that describe the transactions of this data set are obtained by applying PCA. Since the original data cannot be retrieved from the data on the principal components, this data set is in accordance with the European privacy law protecting personal data of citizens.

2.2 Overfitting and underfitting

Before the concept of class imbalance can be understand correctly, first two other fundamental machine learning concepts should be introduced. The explanation and notation below is heavily based on [35].

Given a set of training samples $(x_1, y_1), ..., (x_n, y_n)$ the goal is to learn a function $h_n(x)$ that is able to predict an unseen data point to the correct label y. The predictor $h_n(x)$ is a machine learning classifier. This can be a neural network or any other classification method. Supervised learning can be seen as finding a predictor function h out of all possible functions $\mathcal{H}$. The error of classification

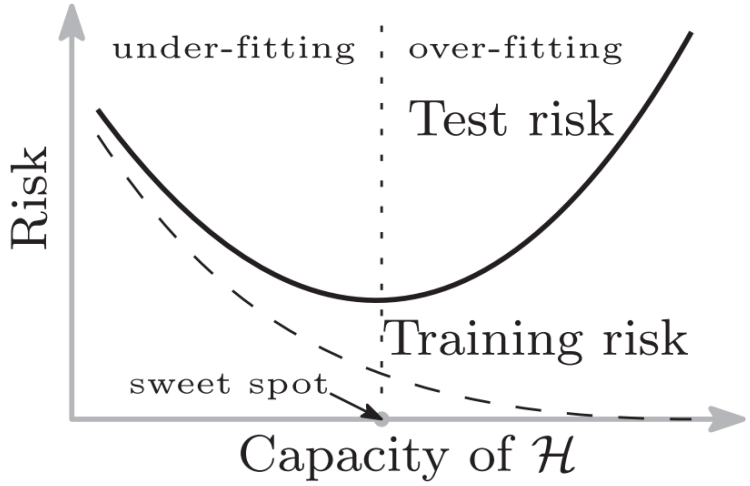

Figure 5: Figure showing the trade-off between decreasing the empirical risk (dashed line) and the true risk (solid line). Directly from [35].

on the training data can be written by Eq. 4. This is also called the empirical risk which should be minimized (ERM)

$$\text{Empirical risk} = \frac{1}{n}\Sigma_{i=1}^{n}\ell(h(x_i), y_i) \tag{4}$$

where ℓ is a loss function representing the error of classification on training data. For binary classification problems this is defined by Eq. 5.

$$\ell(y, y') = \begin{cases} 1, & \text{if } \{y \neq y'\} \\ \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Usually the training data is randomly sampled from a population P. The performance of $h_n(x)$ is evaluated on other independent randomly sampled examples from the population. The ultimate goal is to have an empirical risk similar to the true risk. The true risk is the error that we achieve by testing the trained classifier on the unseen test data. This can be rewritten by Eq. 6. This equation represents a trade-off. Minimizing the empirical risk (error on the training data) increases the error on the test data at a certain point. This is visualized by Fig. 5. The x-axis represents the complexity of the model where $h \in \mathcal{H}$ becomes more complex when moving to the right. The figure shows that at a certain point if the model becomes too complex, the test risk increases while the training risk (empirical risk) still decreases. The so called 'sweet spot' represents the ideal trade-off between the empirical risk and the true risk. However, finding this ideal model h is a difficult task. The parameters that comes with machine learning algorithms determine the complexity of the algorithm. If we tune the parameters to minimize the empirical risk maximally and the true risk increases, the model is overfitting the training data (right side of Fig. 5). Contrary, if the empirical risk is not low enough, the true risk will be high since the model is not learning the patterns of the training data enough (left side of Fig. 5).

$$\mathbb{E}_{(x,y)} \approx_P [\ell(h(x), y)] \tag{6}$$

In machine learning, often the terms *overfitting* and *underfitting* are used. If a model is overfitting it means the model has low bias (low error on the training data) and high variance (high error on the

test data). If a model is underfitting it means the model has high bias (high error on the training data) which results in the model not learning patterns captured in P properly. This also results in a low performance on the test data. However, since the true risk is structurally high for several parameter settings of h, the variance on the test set is also low.

2.3 Class imbalance problem

Credit card fraud detection is not the only application of machine learning where class imbalance is a problem. Class imbalance is a problem that is seen in many machine learning applications. However there are several ways to tackle the problem. This subsection will present several methods to overcome the problem of class imbalance. Where some methods try to balance a data set by manipulating the distribution (data-level methods) other methods try to overcome the class imbalance problem by modifying the algorithm that is trained on the data set (algorithm-level methods).

2.3.1 Over- and under sampling

A straightforward method of balancing the distribution of the classes is using under- and oversampling. Undersampling means that random samples from the majority class are deleted from the data to improve the balance of the data. For extremely unbalanced data (for example credit card fraud data sets), this means that a lot of data of the majority must be disregarded to balance both of the classes. The other possibility is oversampling where data points of the minority class are duplicated. The danger of this method is that the risk of overfitting increases [36]. In order to balance an extremely unbalanced data set, oversampling needs to be repeated several times. Every time the minority class is oversampled, the risk of overfitting increases since the trained model becomes more biased towards the data set.

2.3.2 SMOTE

A more complex oversampling algorithm, which is heavily used in literature is SMOTE [37]. SMOTE increases the minority data class by creating artificial data points. These artificial data points are obtained by interpolation of the data points of the monitory class. The pseudocode as mentioned in the original paper can be seen in Fig. 6 (directly taken from the original paper [37]). SMOTE loops over the data points from the minority class. It then finds the K-nearest neighbors. One of these neighbors is randomly chosen and the vector to the original data point and the selected neighbor is calculated. This vector can be multiplied with a random decimal number between 0 and 1 which results in a new vector with a terminal point between the selected neighbor and the original data point. The terminal point of the vector in the feature space is the new data point that can be added to the data set. This process can be repeated several times to oversample the minority class.

Research has shown good results with SMOTE on the Europe CFD data set. [38] reported a precision of 0.96 by using a simple system consisting of SMOTE in combination with Logistic Regression on the Europe data set. A major drawback of SMOTE is that the synthetic data points created are sensitive to being overgeneralized. This means that the data points that are created by SMOTE do not take into consideration the location in the feature space of neighboring data points of the other class. This might result in artificial data points overlapping the classes [10].

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100)$ * T synthetic minority class samples
1. (∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
8. $k =$ Number of nearest neighbors
9. $numattrs =$ Number of attributes
10. $Sample[\,][\,]$: array for original minority class samples
11. $newindex$: keeps a count of number of synthetic samples generated, initialized to 0
12. $Synthetic[\,][\,]$: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
13. **for** $i \leftarrow 1$ **to** T
14. Compute k nearest neighbors for i, and save the indices in the $nnarray$
15. Populate(N, i, $nnarray$)
16. **endfor**

Populate(N, i, $nnarray$) (∗ *Function to generate the synthetic samples.* ∗)
17. **while** $N \neq 0$
18. Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
19. **for** $attr \leftarrow 1$ **to** $numattrs$
20. Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
21. Compute: $gap =$ random number between 0 and 1
22. $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
23. **endfor**
24. $newindex{+}{+}$
25. $N = N - 1$
26. **endwhile**
27. **return** (∗ *End of Populate.* ∗)
End of Pseudo-Code.

Figure 6: Pseudocode of SMOTE directly taken from the original paper [37]

2.3.3 Variations of SMOTE

Several adaptations of SMOTE have been proposed to overcome this problem. An example of such an adaptation is Borderline-SMOTE [39] where, as the name suggests, artificial minority class data points are created which lie close to the border of the majority class. Because of this, data points are more likely to be misclassified. Another adaptation of the SMOTE data balancing techniques is SVM SMOTE [40] which uses an SVM algorithm to create artificial data points which lie around

the borderline of the support vector. This data balancing technique has shown to outperform regular SMOTE, Borderline-SMOTE and ADASYN [41] when combined with a random forest classifier in [36].

2.3.4 Threshold-moving

Another method for increasing performance of classification on imbalanced data is moving the decision threshold. Based on score $s = m(x)$ and the decision threshold t a probabilistic model is able to attach a label to a score: 1 if $s > t$. The default decision threshold in many classifiers $t = 0.5$ might not be optimal depending on the wanted precision or recall. This trade-off depends on the application. In CFD a high recall is very important due to the potential financial loss of misclassified fraud transactions. As mentioned by [42], '*The bottom line is that when studying problems with imbalanced data, using the classifiers produced by standard machine learning algorithms without adjusting the output threshold may well be a critical mistake*'. This quote suggests that many traditional classifiers operating on imbalanced data may increase performance by setting the correct threshold. While this is a statement from more than 2 decades ago, recent literature still supports the claim that imbalanced data needs a more detailed threshold analysis for optimal performance [43, 44, 45].

It is well-known that finding the optimal threshold is essential for certain applications. Secondly, it is obvious that finding the optimal decision threshold based on a single performance metric is a straightforward process since we can simply loop over a range of threshold, and choose the threshold which optimizes the metric. It is common practice to use N-fold cross validation for finding the optimal decision because cross validation reduces the bias towards the training data set. Using N-fold cross validation is computationally inefficient since we need to train the algorithm N times. However, when evaluating the performance of a classifier operating on imbalanced data, it is application dependent which metric should be optimized which is a problem in CFD since there is not a universal agreed metric which should be optimized.

2.4 Classification

Within the field of CFD several classifiers are optional. In this subsection the most frequently used, best performing classifiers are discussed and explained. Finally a recently published, promising method for classification is discussed from another domain of classification of imbalanced data.

2.4.1 Random forest

The random forest classifier has shown success ever since its first publication in 2001 [16]. In order to understand a random forest classifier, one must first understand the basic principles of a decision tree which are the building blocks of a random forest.

A Decision Tree is an intuitive supervised learning algorithm. A DT creates rules which are inferred from the data. Figure 7 presents an example of such a DT. A DT can be created by several learning algorithms of which iterative dichotomiser 3 (ID3) [46] is one of the most commonly used ones. This algorithm iteratively divides features into two or more parts. Per iteration ID3 selects the feature with the highest information gain (IG). The IG of a feature, $IG(S)$, is calculated by Eq. 7 and is strictly dependent on the entropy which is a measure of disorder of a feature. While there are more methods of calculating the disorder like Gini coefficient, the most common method is entropy. $E(s)$ is given by Eq. 8 where p_i is the probability of an element with class i occurring in the data.

$$IG(S) = 1 - E(S) \tag{7}$$

$$E(S) = \Sigma_{i=1}^{c} - p_i log_2 p_i \tag{8}$$

The feature with the highest IG will split the data set into different groups where each group belongs to a certain condition to the variable. In Fig. 7 the feature with the highest IG is *Outlook*, hence it becomes the root node of the DT. If all rows belong to the same class, a leaf is created with the corresponding class as its label. A DT can be cut-off at a certain point to reduce the depth of the tree. This results in the reduction of overfitting of the data. Note that ID3 chooses the best feature locally per iteration, making it a greedy algorithm. Greedy algorithms might get stuck in a local minima instead of finding the global optimal solution.

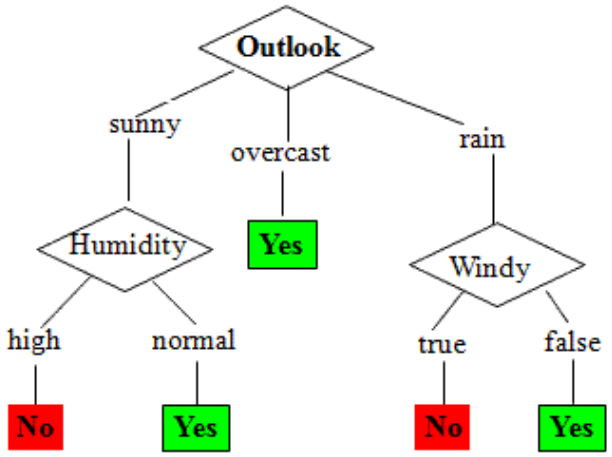

Figure 7: Example of a decision tree where the goal is to predict whether the weather is suitable to play tennis (yes/*no*) based on the features *Outlook*, *Humidity* and *Windy*. Directly taken from [47].

As its name suggest, a random forest is a collection of multiple DTs making it an ensemble classifier. Each DT in the forest outputs a prediction. The prediction with the most votes is the output of the forest. A key element in the initialization of the random forest is that it needs to create DTs which are different from each other. With ID3, the best IG feature selection considers all features. However, during a random forest initialization, the best feature selection only considers a random subset of the features. This ensures that all decision trees are different from each other. Creating a random forest comes with setting several parameters that need to be optimized to optimize performance. While more parameters can be considered, the three parameters below can be considered the most important.

1. **Number of trees in the forest:** Increasing the number of trees generally improves the performance of classification due to the reduction of variance [48]. However, as Breiman stated, this does not result in overfitting [16]. If the forest grows it will always decrease the training speed due to the increase of DTs that need to be trained.

2. **Max depth of the trees**: This parameters influences the maximum number of splits that a decision tree is allowed to make. Increasing the allowed number of splits increases the maximum depth of a decision tree. If a decision tree increases in depth, it tends to overfit the data. Restricting the depth of a tree is also called pruning which reduces unnecessary complexity of decision trees [49]. It is difficult to set this parameter due to the Horizon-Effect. We might not know in front if increasing the depth of a tree (and allowing another split) decreases the error significantly. This makes it difficult to set this parameter.

3. **Max randomly selected features:** If this parameter is increased, the maximum number of features to consider in an individual decision tree increases. Increasing the number of features to consider per DT generally improves the performance on the training data. However, it also means that the difference between the individual decision trees will decrease which might result in overfitting. Increasing the number of features to consider will increase the run time of a random forest due to the increasing number of features to consider.

Since a random forest is based on a bagging algorithm, the variance is significantly reduced in comparison with a decision tree. However, a disadvantage of using a random forest is that (mainly depending on the size of the forest), training time can be long due to all the individual trees that need to be trained.

2.4.2 Support vector machine

Another frequently used method used for classification (which we will see later used in a well performing CFD pipeline) is the support vector machine (SVM) [50]. The SVM is a method where a hyperplane is calculated based on the data points in the input space. This hyperplane separates the classes. In binary classification, data points on one side of the plane belong to class A where data points on the other side of the plane belong to class B. SVM tries to find this optimal hyperplane. While there are multiple variations of a support vector machine, only the linear SVM will be explained since this is used later in this section.

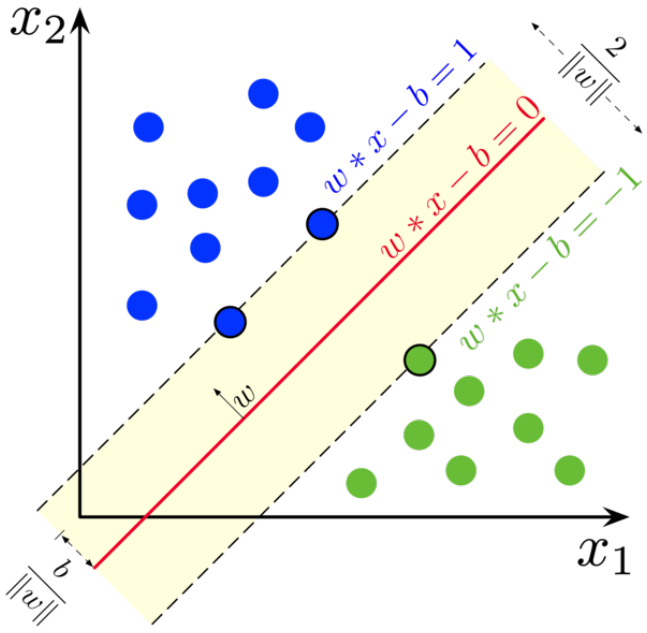

Figure 8: Explanation of the working of a support vector machine. Directly taken from [51].

The hyperplane is described by $Wx - b = 0$ where W is a vector of $n-1$ dimensionality where n is the number of dimensions of input x. Assuming the data is linearly separable (as in Fig. 8), 3-dimensional data can be separated by a 2-dimensional surface and 2-dimensional data can be separated by a single line. We first define the two parallel hyperplanes that separate the two classes of data. A property of these so called 'support vectors' is that the Euclidean distance to the hyperplane, parallel to the support vectors is maximal. The distance between the two support vectors can be described by $\frac{2}{||\mathbf{w}||}$ which means in order to maximize the distance between the support vectors, $||\mathbf{w}||$ should be

minimal. In the example shown in Fig. 8 a linear kernel is used since the data is linearly separable. However, in many practical cases of machine learning this is not the case. Therefore the linear SVM algorithm can be extended by changing the kernel function. Other options for kernel functions are polynomials of a manually specified degree and a radial basis function (RBF). An RBF-kernel is able to partition the data in more than two regions.

2.5 Artificial neural networks

The usage of artificial neural networks (ANNs) within the field of machine learning has gained much attention since the computational resources necessary for training have become available. Due to the attention ANNs have received in the last decades, the CFD literature presents several methods for classifying credit card transactions with ANNs which come close to state-of-the-art results. This is a reason to briefly explain the workings of this type of classification and to present several CFD systems proposed in literature.

An ANN is a method for classification inspired by a biological neural brain. An ANN consists of three layers which can be described by vectors: (1) Input layer $\overrightarrow{x}$, (2) hidden layer(s) $\overrightarrow{f_i}$ and the (3) output layer $\overrightarrow{y}$. Although there are several types of neural networks, the basic form of an ANN is the feed forward neural network. In this type of ANN the input layer sends signals to the hidden layer and the hidden layer to the output layer. Neurons of layer k are connected to layer $k-1$ and $k+1$. The connections between neurons represent weights.

An example of an ANN with a single hidden layer is shown in Figure 9. In this example the input vector $\mathbf{x} \in \mathbb{R}$ is multiplied by the weights connecting the input layer with the hidden layer which results in the hidden layer vector. In order to obtain the output layer the hidden layer vector is multiplied with the weights connecting the hidden layer to the output vector $y \in \mathbb{R}$. The ANN makes a classification with an activation function. An activation function decides whether a neuron should be activated based on a non-linear function. There are several activation functions. Examples of the most common used activation functions are sigmoid, ReLu, leaky ReLu and softmax. The activation function ensures that the input can be mapped with a non-linear transformation to the output.

During the training phase of an ANN, the weights are determined. Although there are multiple ways to train an ANN, the most common algorithm is backpropagation. Backpropagation iteratively changes the weights per epoch based on the error of classification.

CFD literature proposes several systems with good performance using ANNs. An example is [53] where the data is first balanced using an adaptation of SMOTE called SMOTE-ENN. This balancing method uses undersampling and oversampling simultaneously. For classification a long short term memory (LSTM) network is used. An LSTM differs from a feed-forward network since it contains a cell with an internal cell state which maintains its state over the epochs.

[54] proposes a deep neural network for CFD. The first layer of their network consists of 512 hidden units with dropout. Dropout means that during training, a specified percentage of random nodes are dropped. The effect of a dropout in a hidden layer is that it increases the regularization of the network resulting in a smaller chance of overfitting. In this layer the input is also standardized. The second layer consists of 256 hidden units and the third of 32 units. A sigmoid function is used to determine the classification output. While this method achieves an AUROC of 0.901 (which will be discussed layer in this section), it is outperformed by several other methods as shown in Table 2 in terms of AUROC.

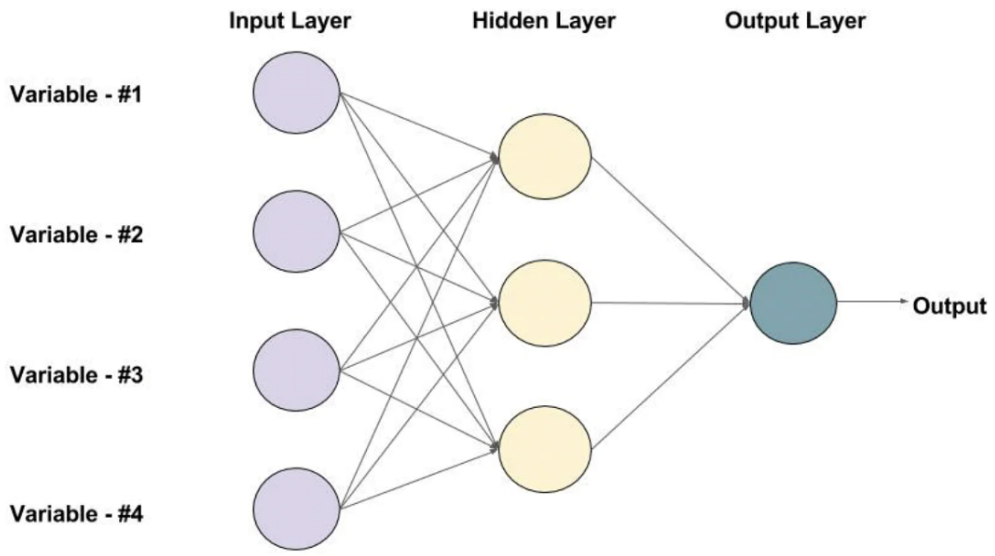

Figure 9: Example of a neural network with one hidden layer. Directly taken from [52].

2.6 Boosting algorithms

Boosting algorithms have shown great success with classification on imbalanced data due to its ability to reduce variance and bias [55]. The idea of a boosting algorithms is to sequentially train k models on a subset of the training data.

Data points which were classified wrong by a model before have a higher probability of getting in the next drawn sub set. This probability is represented by a weight on every data point which increases after incorrect classification. A boosting algorithm combines a set of weak learners to an ensemble classifier that outperforms all of the weak learners individually. One of the first boosting algorithms is AdaBoost [56] which works as followings:

Consider data set $D \in \mathbb{R}^n, y_i \in \{0,1\}$ where n is the number of dimensions, x is the set of data points and y is the target variable being either 0 or 1. AdaBoost assigns a weight to every training data point. The initialization of AdaBoost attaches the same weight to every data point which is $w = 1/N \in [0,1]$ where N is the number of data points. Now a random subset of the data is drawn where the weight of each data point represents the chance of a data point to go in the subset. From here, the steps below are executed.

1. For $i = 1, 2, ..., \max$ epoch

   (a) Draw a random subset D_i of size M from training data D based on the weights. Data points with higher weight have a higher chance to be in subset D_i

   (b) Train a model m_i on D_i

   (c) Test performance of m_i on D

   (d) For all misclassified data points $P \in D_i$, increase the weight w_i

   (e) Based on the number of correct classifications, attach a weight α_i on m_i.

2. For final model m, calculate the sum of the classification weights α multiplied by the output of m_i for all $i = 1, 2, ..., N$. This results in the output of classification of the ensemble classifier obtained by AdaBoosting.

Every iteration during boosting, a 'goodness' of the model is calculated. This is called the loss function and measures the performance of the model over the epochs.

Since AdaBoost, several other boosting methods have been proposed like XGBoost [57], BrownBoost [58] and CatBoost [59]. Although these boosting algorithms all have their strong and weak points, all boosting algorithms share some advantages and disadvantages. While some studies show that boosting reduces the chance of overfitting greatly [60] some studies show that that boosting methods are sensitive to noise and overfit on this noisy data [61]. This can also be proven mathematically by showing that the generalization error is bounded [62]. Although there has been debate about whether boosting methods overfit, it has shown success with classification of imbalanced data [63, 64, 65, 66].

2.6.1 CatBoost

A previously mentioned adaption of AdaBoost is CatBoost which is a recently published boosting algorithm. This algorithm is able to handle categorical features in the training phase by transforming them into numerical features. One of the biggest improvements of CatBoost is the unbiased gradient estimation which controls the overfit better in comparison with AdaBoost. This is achieved within the calculation of the gradient of each sample by excluding that sample from the calculation of the gradient.

2.6.2 XGBoost

XGBoost is a parallel tree boosting algorithm which can be used either for classification or regression. Since the regression variant is not relevant for classification, only XGBoost classification will be explained. The explanation relies on the mathematical notation as explained by [67].

As with all boosting algorithms, XGBoost builts boosting trees. XGBoost uses two derivatives in order to measure the performance of the system. The gradient (g) and the hessian (h). In order to know that the gradient is going into the right direction, not only the derivative is taken, but also the second order derivative. This is one of the key solutions ensuring the speed of XGBoost. XGBoost also uses parallel computing for the calculation of the trees resulting in a decrease of execution time. Also, during the training phase of the decision trees, pruning is done automatically using a threshold. In order to prevent overfitting, an L1/L2 regularization term is added to the loss function. XGBoost comes with many hyperparameters that can be tuned. The most important parameters are:

1. **Regularization parameter** λ : This parameter controls the overfitting of XGBoost. If λ is increased, the chance of overfitting is reduced. However, if λ is too too high, there exists a chance of underfitting the data. λ also controls the effect of outliers of the data. This means that a higher λ reduces the effect of outliers.

2. **Prune threshold** γ : A main advantage of XGBoost is that decision trees are automatically pruned. This is done by calculating the IG per split. If the IG is below threshold γ the split will not occur and the tree will be pruned. γ ensures that no uninformative splits occur. Increasing γ decreases the bias towards the training data.

3. **Learning rate** η: The learning rate η controls the speed of convergence of XGBoost. Every iteration the prediction of the next tree is changed based on the prediction error and the learning rate. Increasing the learning rate results in faster learning. However, if the learning rate is too high XGBoost might never be able to find the optimal solution since the step size is too high. This could result in over correction.

2.7 Metrics

The introduction already covered several metrics which can be used to measure the performance of a classifier operating on imbalanced data. However, these metrics represent a trade-off. If the precision increases, the recall decreases and vice versa [21]. Several alternatives exist for precision and recall.

2.7.1 F_1 Score

An aggregated metric of the precision and recall is called the F1-measure and is calculated by Eq. 9. This metric is heavily used in literature when one wants to directly compare one system with another system through one metric taking precision and recall into account.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{9}$$

A major drawback of the F_1 score is that it disregards the number of true negatives which (independent of the application) is important. Another drawback is that the F_1 score is sensitive to swapping the labels which make its performance dependent on arbitrary labels [68]. Although criticism about the F1-score is uttered [69, 70], it is still a heavily used metric for the classification of imbalanced data due to its intuitive interpretation.

2.7.2 AUROC

Another metric that is heavily used in the performance evaluation of binary classification is the **A**rea **U**nder the **R**eceiver **O**perating **C**haracteristics. This metric calculates the area under an ROC-curve. The definition given here relies heavily on [71]. The ROC-curve is a plot of the true positive rate (TPR) on the y-axis against the false positive rate (FPR) on the x-axis. The TPR is defined in Equation 10. The true negative rate is defined in Equation 11.

$$TPR = \frac{Positives\ correctly\ classified}{Total\ positives} \tag{10}$$

$$FPR = \frac{Negatives\ incorrectly\ classified}{Total\ negatives} \tag{11}$$

For various decision thresholds, the TPR and FPR can be seen from the ROC-curve. In an ROC-curve, random classification represents a diagonal curve with $y = x$. In an ROC graph, the lower left point (0,0) means a classifier that always classifies a data point as negative meaning it has no true positives and no false positives. A point on the top left (1,0) means perfect classification. When an ROC-curve is on the bottom right of the diagonal $y = x$ which means that the classifier is performing worse than random guessing. When inversing the labels this classifier will perform better than random guessing. ROC-curves are especially useful for probabilistic models because they can visualize the performance based on different thresholds. Based on the ROC-curve, a wanted TPR-FPR rate can be chosen.

One of the main advantages of ROC-curves is that they are insensitive to changes in class distribution [71]. A disadvantage of the AUROC is that it shows good performance if only all of the true negatives (non-fraudulent transactions) are classified correctly which is shown by [72]. They showed using an isolation forest [73] that they were able to match state-of-the-art results in terms of AUROC on the Europe data set. This is due to the false positive rate being relatively low as a result of the high number of total negatives.

2.7.3 AUPRC

The **A**rea **U**nder the **P**recision-**R**ecall **C**urve is a similar metric in terms of interpretation. Where the AUROC shows the trade-off between the TPR and the FPR, the AUPRC (as the name suggests) shows the trade-off between precision and recall for various thresholds. The PR-curve shows the precision as a function of recall. The reason why the AUPRC would be preferred over the the AUROC is because of its tolerance for imbalanced data [74]. The interpretation of the AUPRC is more difficult to understand compared to the AUROC. The reason is that the AUPRC depends on the class distribution. A random classifier will get an AUROC of 0.5. However, the AUPRC will depend on the fraction of positives in the data set. This means that if the minority class is 12% of the data set, 0.12 is the baseline. Obtaining an AUPRC of 0.50 on such a data set is considered a good result depending on the expectancy of performance. A disadvantage of the AUPRC is that it does not involve the number of true negatives in any way.

2.7.4 Financial metrics

All metrics discussed above are based on the number of correct classifications. Every classification has equal weight when evaluating the correctness of the classification of the transaction. Within the field of credit card fraud detection, one might argue that not every transaction has equal weight. Consider two fraudulent transactions T_1 and T_2. T_1 is a transaction of €40.000,- and T_2 is a transaction of €10,-. As previously mentioned in the Introduction, depending on the transaction, banks sometimes cover the financial losses of a customer of a credit card company due to credit card fraud. Therefore these credit card companies have a bigger financial interest to classify T_1 correctly than T_2 due to the higher amount of T_1. Based on this example one might argue that we need to take the amount of the transaction into consideration when evaluating the performance of a credit card fraud detection system.

In the literature on CFD, only several evaluation metrics are proposed [23, 27, 75]. [23] proposed a metric that they call *economic Efficiency (EE)*. '*The EE accounts for the economic returns a company receives when it correctly classifies legitimate transactions subtracted by the losses accumulated for no identifying a fraud*'. The EE is calculated by Eq. 12 and 13. These equations are directly taken from [23].

$$EE = \sum_{i=1}^{\#Transactions} Returns \tag{12}$$

$$Returns = \begin{cases} \sum_{i=1}^{x}(v_i \cdot k) & \text{if i is legitimate (TN)} \\ \sum_{i=1}^{y}(-v_i \cdot (1-k)) & \text{if i is fraudulent (FN)} \end{cases} \tag{13}$$

where v_i is the amount of the transaction and i is the index of the transaction. The value for k is the financial gain that their contact, PagSeguro UOL (a financial services and digital payments company) uses and is set to 0.03. This means that UOL receives 3% of every legitimate transaction that is

classified correctly as non-fraud (TN). From Eq. 13 we can also observe that for every fraudulent transaction classified as legitimate (FN), 97% of the value of the transaction is lost.

[76] presents an economic efficiency similar to [23]. However, the main difference is that [23] does not penalize false positives (genuine transactions classified as fraud). This is done in [76] by subtracting 1% of the amount of all of the misclassified transactions. The formula, as given in [76] is given by Eq. 14

$$EE_{Technique} = \sum_{j=1}^{n} G * r - (L * (1-r) + NG * P) \tag{14}$$

where G is the amount of the transactions and r is the percentage that the company gains for successful transactions. L is the financial value of falsely negative classified transactions. NG is the amount of the misclassified transactions and P is the penalty ratio which penalties misclassified transactions. P is set to 1%. The EE given here is also normalized to make it comparable with the current strategy used by the credit card company. Two more concepts are introduced for this. The first one is EE_{Real} which is a simplification of the profit of the company and is given by Eq. 15.

$$EE_{Real} = \sum_{j=1}^{n} NF * r - F * (1-r) \tag{15}$$

where NF represents the amount of non-fraud transactions and F is the financial value of fraud transactions. Secondly, EE_{Max} is the maximum gain that a credit card company can have when no fraud occurs and all transactions are classified correctly meaning this is r multiplied by the sum of all transactions. With these new concepts, the authors define the EE as in Eq. 16.

$$EE = \frac{EE_{Technique} - EE_{Real}}{EE_{Max} - EE_{Real}} \tag{16}$$

With the EE as defined in Eq. 16 100% represents the maximum possible gain and 0% is the scenario where no credit card fraud detection is used. The specific value for r is not given due to privacy reasons of the cooperating company of the study.

[27] proposes a utility-based method. This method calculates the expected utility of classifying a transaction. This expected utility is based on the costs and benefits of a transaction being classified manually or by the detection system. The authors claim that according to expert domains the cost of a manual inspection by a credit card fraud worker is 'roughly constant and was set to 150 Euros (approximately 2 hours of work)'. Unfortunately, no source is given for this estimation of manual classification cost.

[77] utilize a financial metric where each type of classification of the confusion matrix is handled differently to measure the savings of a credit card service company. This is given by a cost matrix as proposed by [78] which is shown in Table 1. In this matrix, C_k where $k \in \{TP, FP, FN, TN\}$ defines the cost that comes per type of classification. C_a defines the administration cost for handling a TP or a FP. From Table 1 it is clear that the financial gain of true negatives (non-fraud transactions classified as non-fraud) is not taken into account. While the measure as presented by [77] should represent the *savings*, which is the amount of money that is saved by a certain pipeline in comparison to using no pipeline, this metric does not include the total financial gain since credit card service companies also financially benefit from TN transactions [23].

	Actual Positive ($y_i = 1$)	Actual negative ($y_i = 0$)
Predictive positive ($c_i = 1$)	$C_{TP_i} = C_a$	$C_{FP_i} = C_a$
Predictive negative ($c_i = 0$)	$C_{FN_i} = Amt_i$	$C_{TN_i} = 0$

Table 1: Credit card fraud cost matrix (taken from [78])

	Method	AUPRC	AUROC	Precision	Recall	F_1
1	Feed-forward neural network[79]	0.6876	0.8675	0.9351	0.7351	0.8229
2	Deep neural network with Full Center Loss function [80]	0.879	-	-	-	0.805
3	General Adversarial Network using autoencoder [81]	0.9759	0.7937	0.9759	0.7937	0.8736
4	Sampling based on data distribution with Ensemble Learning model (HELMDD)[82]	-	0.985	0.721	0.799	0.758*
5	SMOTE + random forest[83]	-	-	1.00	1.00	0.994
6	**XGBoost with implicit data balancing**[67]	-	0.981	-	-	0.857
7	Autoencoder + random forest	-	0.962	0.998	0.891	0.941*
8	**SVM recursive feature elimination + random forest** [1]	0.97	1.00	1.00	1.00	0.99
9	Undersampling + Logistic Regression [84]	-	0.888	1.0	1.0	0.875
10	**SMOTE + CatBoost (using Bayes Minimum Risk)** [63]	-	0.999	-	-	1.00

Table 2: Table providing state-of-the-art results on the Kaggle Europe data set based on the following ranking of metrics: (1) AUROC, (2) F_1-score In the column of the F_1 score the value was calculated afterward with the precision and the recall according to Equation 9. These F_1 values are annotated with a (*). If a field is not filled in, the authors of the paper did not present this metric.

2.8 State-of-the-art Credit Card Fraud Detection

This subsection will cover the best-performing pipelines presented in the literature on CFD. Since many studies show pipelines performing on a private data set, these studies cannot be included due to a lack of reproducibility. One of the most frequently used data sets in CFD literature is the previously mentioned Europe data set hence this will be used as the benchmark data set. Only pipelines performing on this data set will be included in the state-of-the-art analysis. A crucial definition here is which metric we base our definition of the *best-performing* on. The metrics previously described, each with its advantages and disadvantages, show that it is difficult to choose a metric to base the state-of-the-art on. While there is no definitive answer, some metrics are used more often in the literature than others. The AUROC is a frequently used metric. However, the AUROC is sensitive to skew in class distribution [71] which is undesirable for the classification of imbalanced data. Therefore the AUPRC should be preferred as a primary metric to determine the best-performing credit card fraud detection systems. Unfortunately, not all papers present an AUPRC. Therefore the AUROC was taken as the primary metric on which to base the state-of-the-art result. When both of these metrics were similar or not present, the F_1-score was chosen.

To make the different systems in the literature comparable only performances were chosen that were tested on the Europe CFD data set. The reason for the selection of this data set is that this is by far the most frequently used publicly available data set in the literature on CFD.

Table 2 shows an overview of the best pipelines used on the above criteria. For the sake of completeness, precision and recall are also mentioned. The highest AUROC listed in Table 2 is (8) with an AUROC of 1.00. Note that this is a rounded value and does not mean perfect classification. The second highest AUROC is (10). Although (4) achieves the third highest AUROC, due to a lack of implementation details for the reproduction of the system, (6) was chosen as the third best performing pipeline. These systems will be considered state-of-the-art based on their high AUROC score.

2.8.1 SVM Recursive feature elimination and random forest

[1] presents a pipeline that operates on the Europe data set. This system consists of three stages:

1. **Feature elimination:** Support vector machine recursive feature elimination
2. **Data-level balancing method:** SMOTE
3. **Classification:** Random forest with grid search for parameter optimization

The first step is a recursive feature extraction method (RFE) using an SVM with 10-fold cross validation. RFE is a method to reduce the number of features. It trains a classifier (SVM in this case) on the data. Secondly, a score is given to the features which rank them based on importance. 28 features are selected by the author (including 'Time' and 'Amount'). After splitting the data into train- and test data, SMOTE is applied to the training data. For classification, bayesian belief networks, random forest, K-nearest neighbors, SVM, artificial neural networks, and a hidden Markov model was tested. The random forest classifier showed the best performance and was optimized using a grid search. As can be seen from Table 2, the AUROC measured using this pipeline is 1.00 which suggests perfect classification. However, since the F_1 score is not 1 it can be concluded that the AUROC presented is rounded above.

2.8.2 SMOTE + CatBoost

[63] presents s system which resulted in an AUROC of 0.999. This is achieved by first balancing the training data set using SMOTE. SMOTE is applied repeatedly to the minority class until the data points of the minority class equal the data points of the majority class. Secondly, CatBoost is used for classification since it outperformed logistic regression, random forest, and XGBoost.

2.8.3 XGBoost with implicit data balancing

[67] presents an optimized XGBoost pipeline which they call OXGBoost (Optimized XGBoost). The optimized values for the setting have been calculated by using `RandomizedSearchCV` from sklearn and 5-fold cross-validation. Two data sets were used for this study. The Europe data set and a private data set that was obtained by cooperating with a financial institution. For both data sets, different parameter settings were obtained. The parameters that are optimized using the Europe data set are shown in Table 3. The authors mention an *implicit* data-balancing technique. This is obtained by the usage of the parameter *scale_pos_weight* which is the imbalance ratio defined by the number of negative instances divided by the number of positive instances. The parameter adjusts the weights of all data points of the minority class which results in the minority class having a higher chance to end up in the subsets. The decision threshold is set to 0.5.

2.9 Optimizing the decision threshold

By default, the decision threshold of classifiers is set to 0.5. While this is an intuitive setting, it might not be optimal due to class imbalance. A recently published method used for optimizing the decision threshold was presented in [45]. This method is called **G**eneralized t**H**resh**O**ld **S**hif**T**ing Procedure (Ghost from here).

This method was tested on 138 public drug discovery data sets that contain biological activity measurements where there is an imbalance of active vs inactive compounds resulting in an imbalance

Hyperparameter	Optimal value
n_estimators	400
min_child_weight	3
max_depth	15
learning_rate	0.3
gamma	0.1
colsample_bytree	0.4

Table 3: Optimal parameters for XGBoost found by RandomizedSearchCV in [67] using xgb library name convention.

of distribution in a binary classification problem. This application is similar to credit card fraud detection in terms of binary classification and class imbalance. The Ghost algorithm is a procedure to determine the decision threshold based on Cohen's kappa [85]. After setting the decision threshold, the initially trained classifier is also used for testing. This results in an entire data balancing + classification pipeline. The details about the Ghost algorithm are explained below and follow the explanation as in the original paper.

1. Split the data into 80% training data and 20% test data
2. Train a probabilistic model on the training data
3. Extract prediction probabilities for all data points in the training set
4. Randomly draw N stratified subsets from the training set (e.g., 20% of the training data)
5. Loop over a set of possible decision thresholds $T = \{t_1 = 0.05, t_2 = 0.1, ..., t_m = 0.95\}$ (0.05 to 0.95 with 0.05 spacing)
6. For every $t \in T$ with do:
   (a) Label the training samples. 1 if $m(x) > t$, 0 otherwise.
   (b) Calculate Cohen's kappa for all N subsets
   (c) Compute the median of Cohen's kappa over the N subsets. This is the aggregated value of Cohen's kappa over the subsets
7. Select the threshold that maximizes Cohen's kappa on the training subsets
8. Use the trained probabilistic model in combination with the optimal decision threshold to obtain a binary classifier and classify the test data set

The Ghost algorithm starts by splitting the data into train and test data with an 80%-20% split respectively. Then a probabilistic model is trained on the training data. Now we can provide probabilities on the transactions of the training data. Here every probability represents the chance of a transaction being a fraud. The next step is to draw N stratified subsets from the training set. The default setting of Ghost takes 100 training subsets with a size of 20% of the training data resulting in overlapping subsets. To find the optimal decision threshold, Ghost iterates over possible thresholds ranging from 0.05 to 0.95 using 0.05 spacing. For every threshold, the training samples are labelled

with respect to the decision threshold. Now Cohen's kappa is calculated for all N subsets. Cohen's kappa is calculated by Eq. 17

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{17}$$

where p_0 is the accuracy of the model predictions. p_e is the expected accuracy. The expected accuracy is the accuracy that is observed by a model that makes predictions based on the distribution within the classes. For binary classification with equal distributions within the classes, p_0 is set to 0.5. Intuitively this is seen as the accuracy that a random classifier based on the confusion matrix would obtain. The authors use Cohen's kappa since it is a metric taking class imbalance into consideration [86, 87]. This results from the observed accuracy which is compared against a random classifier that takes into account the distributions of the classes. After all the kappas are calculated, the median of all the kappas of the subsets is calculated. The optimal decision threshold is the threshold that maximizes the median of Cohen's kappa of the training subsets. Finally, the initial trained probabilistic model combined with the decision threshold can be used to obtain a classifier.

Ghost allows freedom in choosing the probabilistic model. The authors experimented with four different classifiers, random forest [16], gradient boosting [18], extreme gradient boosting (XGB)[88] and logistic regression [17]. Since the goal of the authors was to 'compare the impact of different strategies for handling imbalanced data sets, the classifiers' parameters were not optimized to achieve optimal classification. However, since the goal of this thesis is to create a model to optimize a certain metric, hyperparameter tuning was done for all four classifiers. For **random forest** the number of trees (n) was optimized ($n \in \{50, 100, 150, 200, 300, 400, 500, 600\}$). Also, the max depth (d) was optimized where $d \in \{11, 13, 15, 17\}$. For **gradient boosting** the learning rate $\eta \in \{0.01, 0.05, 0.15\}$ and the loss function $f \in \{logloss, deviance, exponential\}$ were optimized. For **extreme gradient boosting** the same parameters were optimized within the same ranges as GB. For the **logistic regression**, the penalty $p \in \{L_1, L_2\}$ was optimized. All of the classifiers were implemented using `sklearn 1.0.2` [89] To prevent a massive grid search and to reduce run time, the parameters not mentioned were left to their default values according to the documentation page adequately.

3 Methods

This section will explain the methods to solve the problems as mentioned in the introduction using the concepts introduced in the background literature. To remind the reader of the problems addressed in this thesis, they will be repeated for the sake of readability.

Research questions

To summarize, this thesis focuses on the following problems:

- This thesis presents a new more realistic financial metric (Economic Efficiency).
- This thesis intends to compare the state-of-the-art CFD systems in terms of Economic Efficiency.
- This thesis includes the application of a recently published technique used for the optimization of the decision threshold.

In short, these research questions will be answered with the following steps:

1. Present the new proposed Economic Efficiency and justify the choices made.
2. Find the state-of-the-art for the Europe data set and find the EE for these detection systems.
3. Find promising recently published state-of-the-art techniques from other applications with imbalanced data and test its performance on the Europe data set.

3.1 Data

A major issue with comparing pipelines within the field of credit card fraud detection is that the data contains personal data. Due to confidential issues that authors have with the financial institutions that provide real-world data, this data is often unavailable [23, 90, 91, 77]. However, there are publicly available data sets of credit card fraud transactions for research purposes. The data sets used for this thesis are the Europe data set which is publicly available on Kaggle and the PaySim data set. The reason these data sets were chosen is that they are frequently used data sets in literature which means that results can be replicated and compared in terms of AUROC and AUPRC.

3.1.1 Europe data set

The data set consists of credit card transactions of European cardholders in September 2013. All of the transactions in the data set happened in two days. It consists of a total of 284,807 transactions of which 492 transactions are labelled as *fraud*. This means 0.172% of all transactions of the data are fraudulent transactions. The numerical features that describe the transactions are obtained by a PCA transformation. Due to confidentiality issues, it is not possible to obtain the original features that describe the transactions. The features are called *V1, V2, ..., V28*. The only conclusion we can draw from this is that the original data consisted of $N \geq 28$ features. Besides the 28 anonymous features, the *amount* and *time* of the transaction are provided and were not transformed in any way. The *amount* refers to the amount in Euros of the credit card transaction. *time* refers to the time of occurrence of the transaction. The starting time is $t = 0$ seconds while the last transaction occurred at $t = 172792$ seconds. Every transaction is labelled *fraud* (1) or *genuine* (0).

3.1.2 PaySim data set

The PaySim data set is a synthetic data set based on the distribution of a real-world data set. This data set simulates mobile money transactions. It uses a multi-agent systems approach to simulate a network of financial transactions. In the simulation, an agent is the main entity called the *client*. A client has a profile describing the behaviour of that client. Such a profile consists of several factors such as the transaction limit and the maximum balance of a client. The behavior of a client is affected by the age, behavior, and several other factors describing a client. Actions executed by clients are based on probabilities that are obtained by the distribution of the original data set. The simulation starts with the initialization of the neighbors of agents which is based on user input using several parameters. The profile that is assigned to every client is based on probabilities based on the distribution of the original data set. The initial balance of the client's bank accounts is based on probability distributions for specific profiles. Again, these distributions are based on the original data. At this point, the actual simulation starts. The simulation consists of steps representing one hour. Every step, a client has a probability P_i to make a transaction to neighbor i. P_i can change based on several restrictions put on a client its profile. Examples are the max number of steps a client can perform on a day and the balance on an account of a client. To generate fraudulent transactions, PaySim injects fraudulent agents with criminal patterns based on a certain probability obtained from the original data.

The simulation ran for 30 days which resulted in over 6000,000 transactions. From these 6000,000 transactions, 600,000 transactions were randomly sampled in order to reduce the size of the data set. Of these 600,000 transactions, there are 785 fraudulent transactions. The original data contains 9 features and a target label indicating whether the transaction is fraudulent or genuine. The 9 features of the data are:

- **Step** is the time unit. One step indicates one hour from the initial state.
- **Type** represents what sort of transaction occurred. The possible types of transactions considered in PaySim are *CASH-IN*, *CASH-OUT*, *DEBIT*, *PAYMENT*, and *TRANSFER*.
- **amount** is the amount in Euros of the transaction.
- **nameOrig** is the ID of the client starting the transaction.
- **old balanceOrg** indicates the balance of the account of the client before the transaction.
- **newbalanceOrig** is the balance of the account of the client after the transaction.
- **nameDest** is the unique customer ID of the recipient client. In other words, this is the ID of the customer on the other end of the client that initiates it.
- **oldbalanceDest** is the balance of the account of the recipient before the transaction.
- **newbalanceDest** is the recipient balance after the transaction.

3.2 Preprocessing

Both data sets require different preprocessing steps which will be described individually.

3.2.1 Europe data set

The Europe data set was directly downloaded from Kaggle where it is publicly available. Secondly, the data was split into 80% train data and 20% test data. The time of the transactions is in seconds after the first transaction of the data set which means the first transaction in the data set has a timestamp of 0. Since the obtained data is a PCA transformation of the original data, no scaling needs to be applied on *V1, V2, ..., V28*. However, the feature *amount* is not scaled and contains the original amount of Euros per transaction. Since the amount can be a relevant feature for the classification the amount is scaled for classification. For the calculation of the Economic Efficiency, the original, non-scaled amounts of the transactions were used.

3.2.2 PaySim data set

The PaySim data set contains two nominal categorical variables; *nameOrig* and *nameDest* which represent the customer IDs of the clients. Although this might be a very interesting feature for classification of fraud transactions due to fraudsters being identifiable by a unique ID, these two features were removed. The reason for removing these features is that there are 437,154 unique values in nameOrig and 599,913 in nameDest. Note that there are 600,000 transactions in total. Due to the high diversity of unique values, the entropy of these features is very high resulting in a small information gain when used for classification making the feature unsuitable for classification.

3.3 Implementation details

The background literature section presents the best-performing pipelines presented in the literature on the Europe data set based on several criteria. The best performing pipelines in terms of AUROC are implemented according to the description in the paper. As previously mentioned, method (4) from Table 2 is not chosen due to a lack of reproducibility. Therefore (6) is chosen. The details of the implementation will be discussed here. The systems were implemented using Python 3.8.8. All machine learning techniques were implemented using `sklearn 1.0.2` [92].

3.3.1 SVM Recursive Feature Elimination and random forest [1]

In this method, the features *time* and *amount* were left in the data set without scaling. The algorithm for this method is explained in Section 2.8.1. The data is resampled using SMOTE. Specifications for the sampling strategy are not provided. Therefore the 'auto' sampling strategy in sklearn is assumed which applies SMOTE to the minority class until the classes are equally distributed.

After SMOTE, RFE is applied. In order to rank features with RFE, a choice needs to be made to rank features based on a statistical method or by using a classifier. Certain classifiers offer the ability to rank the features based on importance. This pipeline uses a support vector machine which results in an SVM-RFE model. 28 features were selected using the `sklearn.feature_selection.RFECV()` function with 10-fold cross-validation.

`sklearn.ensemble.RandomForestClassifier` was used for classification to create a random forest. After hyperparameter optimization, the optimal hyperparameters found can be seen in Table 4. Note that the parameters were optimized by using the AUROC instead of a financial metric. random forest attaches probabilities to classes using the voting system which means that a decision threshold is needed. No decision threshold is given in the paper. Therefore the default decision threshold of sklearn is assumed which is 0.5.

Hyperparameter	Value
bootstrap	True
n_estimators	100
max-depth	10
max-features	2
min-sample-leaf	2
min-samples-split	12
max-leaf-nodes	10

Table 4: Optimal parameters for a random forest classifier found in [1] using sklearn name convention.

3.3.2 SMOTE + CatBoost (Bayes Minimum Risk)

The pipeline proposed in [1] is presented in Section 2.8.2. It consists of SMOTE for class balancing and utilizes CatBoost used for classification.

The *time* and *amount* features are scaled to unit variance and zero mean. SMOTE equalizes the number of samples in the minority class with the majority class by iteratively applying SMOTE to the minority class. This is achieved by using `imblearn.over_sampling.SMOTE` with the 'auto' sampling strategy and using 5 neighbors. Afterward, the data is standardized again with zero mean and unit variance. For classification, CatBoost is used. The CatBoost classifier is implemented using `catboost.CatBoostClassifier` with default parameters.

3.3.3 Ghost

The authors of Ghost provide a library that is used to reproduce the algorithm. The random forest classifier outperformed the other classifiers that were tested. Therefore, a random forest was chosen. Since the parameters can influence the performance of the model depending on the application, hyperparameter tuning was performed for the Europe data set. To prevent a massive grid search, the number of trees in the forest, max depth of trees and max randomly selected features were tuned which arguably are one of the most performance-sensitive features of a random forest. The possible values for these parameters that were tried are shown in Table 5. Since this pipeline was tuned on drug discovery data sets, the hyperparameter optimization is done again on both the Europe and the PaySim data set individually. The optimal parameters are presented in the results section.

hyperparameter	Possible values
Number of trees	{100, 150, 200, 250, 300, 400}
Max depth of the trees	{10, 13, 15, 20}
Max randomly selected features	{sqrt, log2}

Table 5: Ranges used for hyperparameter tuning used in the random forest of the Ghost procedure.

3.3.4 OXGBoost

The implementation of OXGBoost is rather straightforward. From the `xgboost` library the function `XGBClassifier` is imported. Secondly, the parameters as shown in Table. 3 are set. The imbalance

ratio (`scale_pos_weight`) is calculated per data set by Eq. 18. No preprocessing or other data-level balancing methods are applied to this pipeline.

$$\text{Imbalance ratio} = \frac{\text{Number of non-fraud transactions}}{\text{Number of fraud transactions}} \tag{18}$$

3.3.5 Ghost + OXGBoost

OXGBoost uses a default threshold of 0.5 which might not be optimal. Since finding the optimal decision threshold using Ghost can be done using any classifier that attaches a probability to a data sample, a random forest is easily interchangeable by a boosting classification method. The implementation of the combination of Ghost and an XGBoost classification method uses the parameter settings from OXGBoost and the decision threshold obtained by the Ghost procedure.

3.4 Evaluation

The evaluation of the pipeline consists of several steps. As previously mentioned, Economic Efficiency is calculated partly on expert advice and partly on literature. Additionally, cross-fold validation is applied in order to obtain a more stable evaluation of the pipeline.

3.4.1 Economic Efficiency

In binary classification there are four types of classifications as shown in Fig. 2. The proposed Economic Efficiency handles true positives, true negatives, false negatives, and false positives in a way to create a representative financial evaluation of a pipeline for credit card service companies.

- **True positives:** TPs are true fraud transactions classified as fraud. Usually, if a fraud transaction is discovered by a CFD system, an attempt to reclaim the amount of the transaction is initiated. This is a difficult process and it is difficult to attach a financial cost to this process. Therefore the EE proposed here assumes that the reclaim has a 100% success rate meaning that the amount of the transaction is always retrieved by a financial service and no financial loss concerning the amount of the transaction is suffered by the credit card service company. Every reclaim of fraudulent transactions is different since it depends on the type of fraud, the reaction time of the credit card holder, and the professionality of the fraudster since some fraudsters are more difficult to find and trace than other fraudsters. However, since there exists a financial loss in terms of salary paid by the work to retrieve the amount of the transaction, this needs to be taken into account when calculating the EE. The financial cost for the reclaim of a fraud transaction β is set to 150 Euros. This value is purely based on [27] where it is mentioned that 'the inspection cost of each report is roughly constant and was set to 150 Euros (approximately 2 hours of work)'.Therefore the EE of the TPs is calculated by Eq. 19 which means β is multiplied by the number of fraud transactions detected by the CFD system.

$$EE_{TP} = \Sigma_{i \in TP}(-\beta) \tag{19}$$

- **True negatives:** TNs are genuine transactions classified as non-fraud. For every credit card transaction, a credit card service company obtains a certain percentage of the amount of the transaction, γ as financial gain. Although these percentages are usually not publicly available, [23] mentions that PagSeguro handles a percentage of 3% [23] . To calculate the EE of the

TNs the amount of the TN transactions (X_i, i) are multiplied by γ which can be mathematically written as in Eq. 20

$$EE_{TN} = \Sigma_{i \in TN(X_i) * \gamma} \tag{20}$$

- **False positives:** False positives are non-fraud transactions classified as fraud. According to icscards, the credit card service company will contact the credit card user. Pending on the reaction of the user, the credit card will be blocked meaning no other transactions can be done. The process of contacting the user, blocking the card, and the other situational actions that need to be undertaken to prevent further escalation of the potential fraud case, takes time for the fraud expert which comes with a financial loss in terms of salary α. [27] mentions this is roughly constant and comes down to two hours of work which is 150 Euros. With an FP the company will observe that it was not a fraudulent transaction. Therefore the percentage of the amount of the transaction is still obtained as a financial gain. The EE of the FPs can be calculated by Eq. 21.

$$EE_{FP} = \Sigma_{i \in FP}(X_i * \gamma) - \alpha \tag{21}$$

- **False negatives:** False negatives are fraud transactions classified as non-fraud. This is the worst-case scenario for a credit card service company since the fraud transaction is not detected resulting in a financial loss for the credit card service company. Secondly, and more importantly FNs often come with a significant financial loss for the credit card user. As mentioned in [23], PagSeguro loses 97% of the amount of an FN transaction. Since there is a significant financial loss based on the amount, this needs to be taken into consideration when calculating the EE of the FNs. The EE of the false negatives can be calculated by multiplying the amount of the FN transaction by δ which is written in Eq. 22.

$$EE_{FN} = \Sigma_{i \in FN}(-X_i * \delta) \tag{22}$$

The total Economic Efficiency is calculated by Eq. 24 and 23. Note that the EE is divided by EE_{max} which is the maximal Economic Efficiency if all classifications are done correctly. This normalization results in a metric that is more intuitive and comparable to different data sets of different ranges of amounts of transactions. The maximum Economic Efficiency is obtainable by handling all fraud transactions as a TP and all genuine transactions as a TN. Therefore EE_{max} can be calculated by Eq. 25. Since EE_{max} is the EE when all classifications are correct, the Economic Efficiency as in Eq. 24 can never be more than 1.00. However, the EE may be lower than 0.00 since FNs have a relatively high negative financial weight.

$$EE = (EE_{TP} + EE_{TN} + EE_{FP} + EE_{FN})/EE_{max} \tag{23}$$

$$EE = \frac{\Sigma_{i \in TP}(-\beta) + \Sigma_{i \in FN}(-x_i * \delta) + \Sigma_{i \in FP}(x_i * \gamma - \alpha) + \Sigma_{i \in TN}(x_i * \gamma)}{EE_{max}} \tag{24}$$

$$EE_{max} = \Sigma_{i \in P}(-\beta) + \Sigma_{i \in N}(x_i \gamma) \tag{25}$$

where

- i is the index of the transactions in the test set.
- EE_{max} is the sum of the amount of all transactions in the test set. This means that all classifications are correct. Therefore we only have TPs and TN.

- x_i is the amount of transaction i.
- α is the amount of money it costs to manually check a transaction whether it is *fraud* or *genuine*. This is set to €150,- based on [27].
- β is the amount of money it cost to retrieve the lost money due to detected fraud transactions.
- γ is the percentage that is the profit a company gains from a transaction. This is set to 3% based on [23].
- δ is the percentage that is lost for a missed fraud. This is set to 97% based on [23].
- FN is the set of all false negative transactions. This means every true fraud transaction that is classified as genuine.
- TP is the set of all false negative transactions. This means every true fraud transaction labelled as fraud.
- FP is the set of all false positive transactions This means every genuine transaction labelled as fraud.
- TN is set of all true negative transactions. This means every genuine transaction labelled as non-fraud.
- N is the set of all negative transactions. This means every genuine transaction.
- P is the set of all positive transactions. This means every fraudulent transaction.

This formula comes with several assumptions

1. If a transaction is classified manually, it is always classified correctly
2. Every transaction is independent of each other meaning transactions have no relationship to one another.
3. If fraud is detected, the total amount of money of that transaction can be returned with the cost of β.
4. If fraud is detected, no financial gain on that transaction is made by the credit card service company.

The EE represents the financial gain for a credit card service company when using a certain pipeline. A high economic efficiency represents a better pipeline in terms of financial gain for the credit card company. However, this does not necessarily mean that this metric optimizes the number or correct classification in terms of precision/recall. From the definition of the EE, it is easily visible that several crucial parameters need to be set, namely α, β, γ, and δ. The value of these parameters will probably vary per credit card company.

The main advantage of this EE over the EE presented in [27] and [23] is that all incorrect classifications are penalized depending on the amount of the transaction. Besides the EE, the AUROC and AUPRC will also be measured to compare the robustness of the threshold decision to other state-of-the-art pipelines.

3.5 Cross-fold validation

In order to obtain results, the build classifiers will be evaluated using k-fold cross-validation. This common method for estimating the prediction error first splits the data set in K randomly selected subsets of the data. Every subset is used once as test data while the other $K-1$ folds are used as training data. Afterward, the mean value of the performance over the different folds is given. $K = 5$ was chosen since this is a common value in literature. For the OXGBoost pipeline, an imbalance ratio is calculated. Since this might vary per fold, the imbalance ratio is also calculated every iteration of the cross-fold validation.

The mean of the results including the standard deviation over the folds will be reported. The SD is noteworthy since it represents the deviation within the performances over the folds. SD is calculated by Eq. 26.

$$SD = \sqrt{\frac{\Sigma_{i=1}^{N}(X_i - \mu)^2}{N}} \tag{26}$$

where N is the number of samples, X_i is the obtained result of fold i, and μ is the mean over the folds. SD is an important result for financial institutions because a high SD represents an unstable performance of the CFD system. If the system its performance turns out to be unstable over the different folds, this might result in transactions with a high amount being misclassified which can lead to significant unwanted financial losses.

3.6 Experiments

To get a good understanding of the experiments and the different pipelines that are used an overview is given here. An overview of the different pipelines and their components is given in Fig. 10. Every pipeline will be executed on both the Europe and the PaySim data set using 5-fold cross-validation. The AUROC, AUPRC, and EE are reported. Based on these evaluation metrics the performance is compared. Statistical analysis is performed to measure significant differences between the performances of the classification methods. The code used to perform the experiments is available on https://github.com/rikvegter/fraud_detection. The run instructions can be found in appendix A.

3.6.1 Economic Efficiency vs traditional metrics

The EE as presented in section 3.4.1 measures the financial gain based on a cost matrix with the financial weights of the data samples. The AUPRC measures the performance of classification with equal weights among all data points. Since the EE presented is a new metric, it is an important insight to analyze the accepted loss (if any) of AUPRC/AUROC and precision/recall that is observed when optimizing a system with the EE. Therefore the performance of AUROC/AUPRC will be measured for various parameter settings to detect an (if any) accepted loss. Also, Economic Efficiency over a variety of decision thresholds will be measured to look at how this relates to the precision-recall trade-off as mentioned in the introduction. These analyses should give insight into the relationship between the existing well-known metrics and the Economic Efficiency.

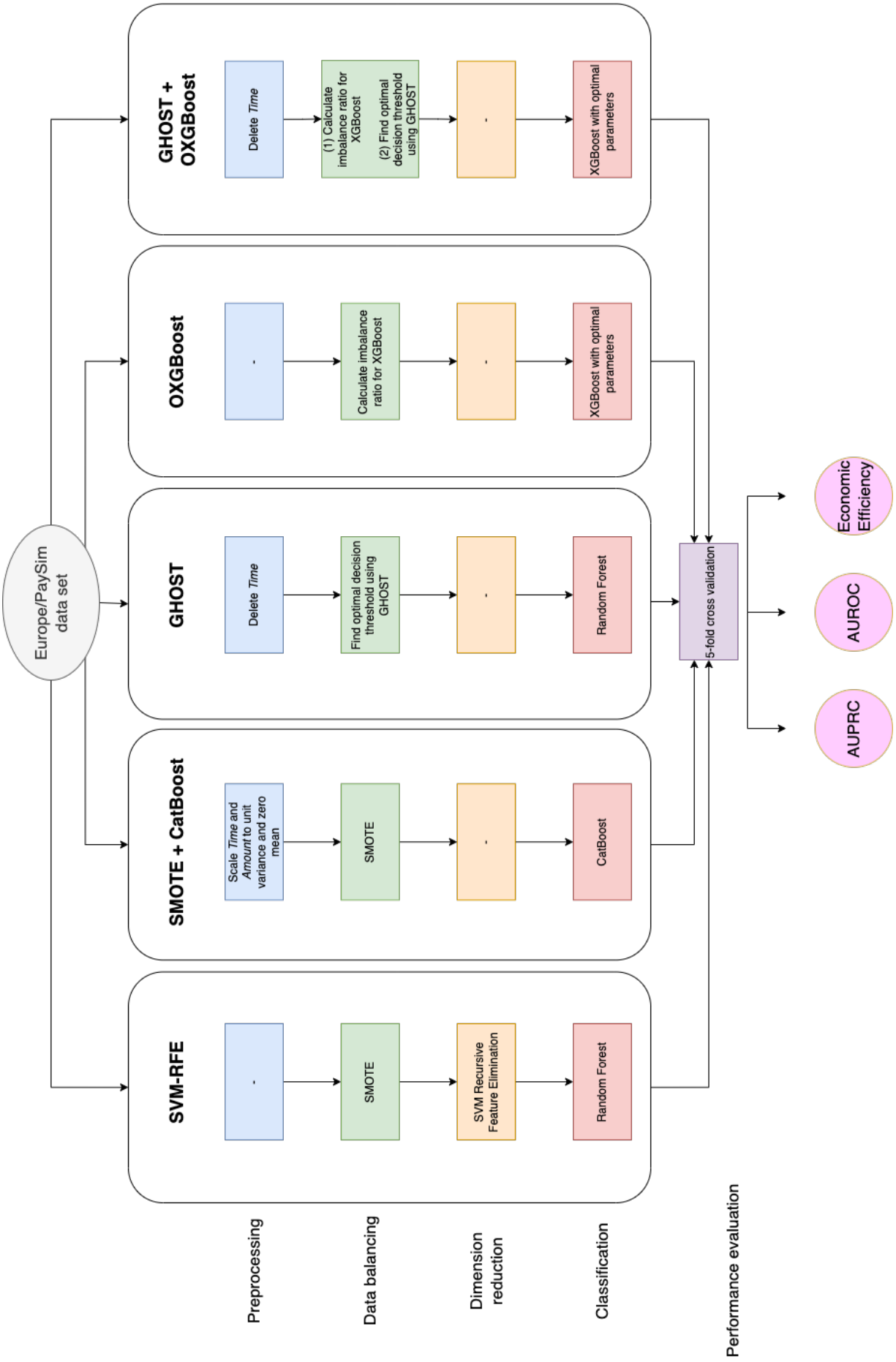

Figure 10: Overview of the pipelines used in the different experiments.

4 Results

This section describes the results of the experiments as mentioned in the Methods section.

4.1 Exploratory analysis

To get a good understanding of the data sets we are working with, an exploratory analysis was done on both data sets.

4.1.1 Europe data set

Table 6 shows a histogram of the *amount* of the transactions in the data set. From this histogram we can calculate that 99.9% of the transactions in the data set have an amount within the range of €0.00 - €2569.00 euros. From this Table, we can also see that some transactions have a much higher *amount* than most transactions. However, since these transactions in particular are important to classify correctly, no outliers based on the amount of the transaction were removed. While this

Amount range	Frequency
0-2569	284395
2569-5138	360
5138-7707	36
707-10276	10
10276-12845	2
12845-15414	1
15414-17983	0
17983-20552	2
20552-23122	0
23122-25691	1

Table 6: Histogram of the amount of the transactions of the Europe data set in Euros

histogram provides a first look at the distribution of the amount of the transactions, it is interesting to look more in detail at the amounts of the transactions within the range €0.00-€2569.00 since this covers most of the data. More specifically, 96.7% of the amounts in the data set are within the range of €0.00 - €500.00. A histogram of these transactions is shown in Fig. 11. The first bin in this histogram represents the frequency of transactions with an amount in the range of €0.00 - €5.00 Euros. Almost 70000 transactions are within this range. Further analysis of the amounts of the transactions showed that 1825 transactions have an amount of €0.00. The reason for transactions of €0.00 is unknown. However, since 27 of these zero-value amount transactions are labelled as fraud, it is undesirable to exclude these transactions since it increases the imbalance of the data.

To make the data anonymous, PCA was applied to the data. Analysis of the variance per component was done to get a better insight into the explained variance of the components. To achieve this, the cumulative explained variance was plotted against the number of components. The result is shown in Fig. 12 from which we can see that the explained variance decreases linearly when excluding principal components. The number of fraud transactions over time is shown in Fig. 13. Several spikes of

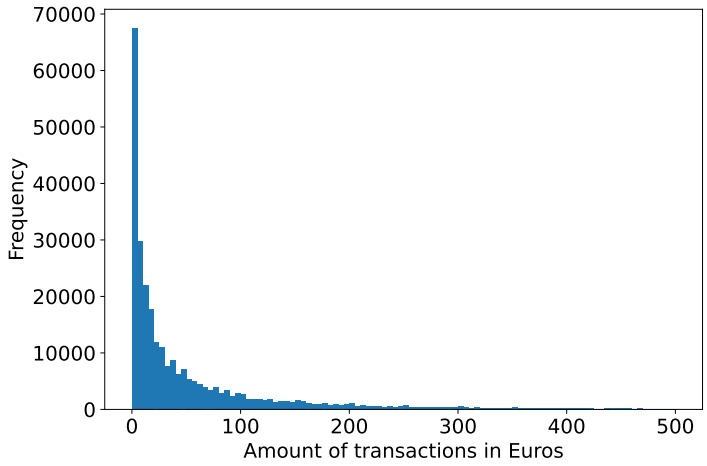

Figure 11: Histogram of the transactions in the Europe data set with an amount within the range €0.00 - €500.00 Euros. This histogram contains 100 bins where every bin represents a range of 5 Euros

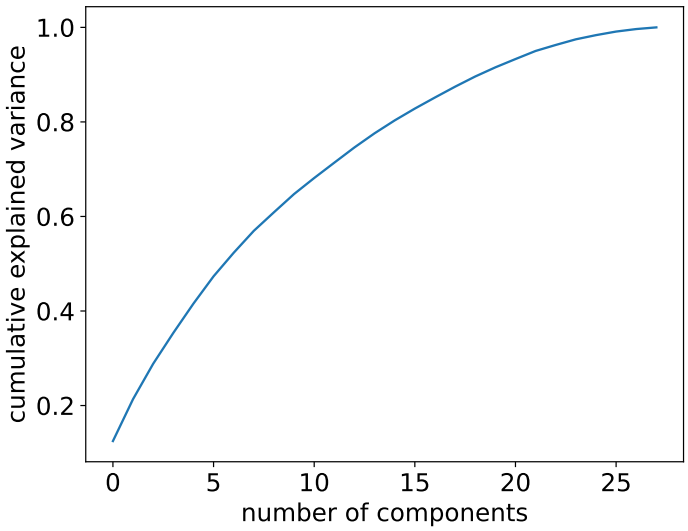

Figure 12: Explained variance per PCA component

fraud transactions are visible which suggests that the *time* feature is a good predictor for several fraud data points.

4.1.2 PaySim data set

Table 7 shows the distribution of the amounts of the transactions of the PaySim data set. From this table, we can see that 99.9% of the transactions have an amount within the range of €0.00 - €6750076.13. Similar to the EA of the Europe data set, it is useful to look at the distribution within this big part of the data to provide a better overview. Since this first bin is still a great range and 332718 of the transactions are within the range of €0.00 - €100,000.00. A plot of the distribution of this range is provided in Fig. 14. From this histogram one can observe that many transactions are

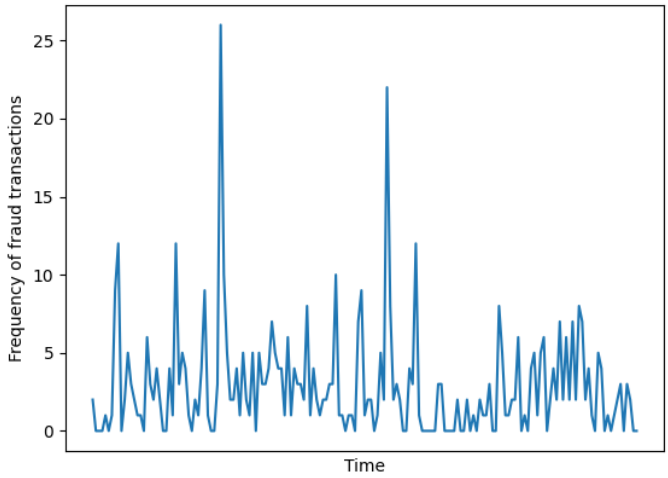

Figure 13: This Figure shows the frequency of fraudulent transactions over time. The time was split into 1000 equally spaced time intervals. The number of fraud transactions within these intervals is shown on the y-axis.

on the lower side of the range. This distribution is similar to the distribution of the Europe data set where the vast majority of the data lies within the lower side of the distribution and there are several transactions with a high amount. Again, due to these data points being important to classify correctly, these data points (which can be seen as outliers in some sense) are kept in the data set.

Amount range	**Frequency**
0 - 6,750,076.13	599215
6,750,076.13 - 1,3500,152.26	623
13,500,152.26 - 20,250,228.39	81
20,250,228.39 - 27,000,304.52	42
27,000,304.52 - 33,750,380.65	14
33,750,380.65 - 40,500,456.77	20
40,500,456.77 - 47,250,532.90	2
47,250,532.90 - 54,000,609.03	2
54,000,609.03 - 60,750,685.16	0
60,750,685.16 - 67,500,761.29	1

Table 7: Histogram of the amount of the transaction of the PaySim data set in Euros

The number of fraud transactions over time is also visualized in Fig. 15. From this figure we can see that the distribution of the spikes is more distributed than the distribution of the spikes in the Europe data set where there are 2 big spikes.

4.2 Outcome of the experiments

The results of the AUROC, AUPRC, and the EE of the Europe and PaySim data set are presented in Table 9.

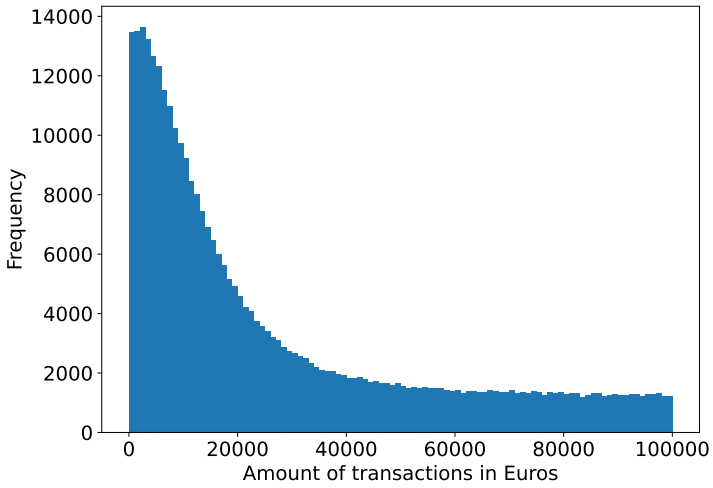

Figure 14: Histogram of the transactions in the PaySim data set with an amount within the range €0.00 - €100,000 Euros. This histogram contains 100 bins where every bin represents a range of 1000 Euros

Hyperparameter	**Value**
Number of trees	250
Max depth of the trees	15
Max randomly selected features	sqrt

Table 8: Optimal parameters found by Ghost + RF

4.2.1 Europe data set

The most optimal decision threshold found by the Ghost algorithm was 0.15 both for OXGBoost and for Random Forest. The Random Forest found the optimal EE with 15 trees, a max depth of 15 and using sqrt as the maximum randomly selected features.

In terms of AUROC, the best performing pipeline on the Europe data set is Ghost which results in an AUROC of 0.997 with an SD of 0.0056. Although OXGBoost + Ghost also obtains an AUROC of 0.997, the SD over the folds is slightly lower for the Ghost pipeline. The other three pipelines result in a comparable AUROC. The best pipeline in terms of EE is SMOTE + CatBoost which reaches an EE of 0.986 with an SD of 0.00690 over the folds. In terms of AUPRC, the best performing pipeline is OXGBoost + Ghost which reaches an AUPRC of 0.0.961. Although OXGBoost shows the same AUPRC, the SD over the folds is slightly lower when OXGBoost is combined with Ghost indicating slightly more stable performance. The SVM-RFE seems to generally perform much worse on the Europe dataset reaching an AUPRC of 0.761.

The PR-curves and ROC-curves of the pipelines on the Europe data set are shown in Fig. 18, 19, 20, 22, and 21. These curves are generated by the evaluation of fold 5 during cross-fold validation. Therefore the AUROC/AUPRC shown on the bottom-right of these figures is not the same as in Table 9.

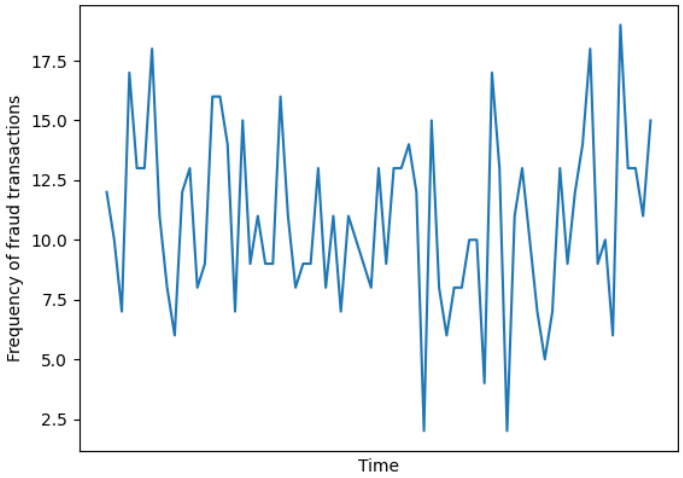

Figure 15: Number of fraud transactions over time of the PaySim data set

	Europe data set			PaySim data set		
Pipeline	**EE**	**AUROC**	**AUPRC**	**EE**	**AUROC**	**AUPRC**
Ghost	0.952 ± 0.0862	$\mathbf{0.997 \pm 0.00560}$	0.949 ± 0.0756	0.985 ± 0.00820	0.999 ± 0.000300	0.970 ± 0.033
SVM-RFE	0.0157 ± 0.00180	0.980 ± 0.0129	0.761 ± 0.0597	-0.140 ± 0.00650	0.991 ± 0.0023	0.675 ± 0.0112
Smote + CatBoost	$\mathbf{0.986 \pm 0.00690}$	0.995 ± 0.00990	0.948 ± 0.0798	0.935 ± 0.0122	$\mathbf{0.999 \pm 0.000100}$	0.9603 ± 0.0219
OXGBoost	0.957 ± 0.0862	0.996 ± 0.0072	0.961 ± 0.0778	$\mathbf{0.999 \pm 0.0009}$	0.999 ± 0.000200	$\mathbf{0.987 \pm 0.0255}$
OXGBoost + GHOST	0.955 ± 0.0896	0.997 ± 0.00715	$\mathbf{0.961 \pm 0.0772}$	0.955 ± 0.0896	0.999 ± 0.000200	0.987 ± 0.0270

Table 9: This table shows the Economic Efficiency (EE), AUROC, and AUPRC. The mean values over 5-fold cross-validation are shown including the standard deviation of the performance over the 5 folds. The results of both data sets are shown. The highest mean performance per metric per data set is indicated by bold numbers.

4.2.2 PaySim data set

The optimal decision threshold found by Ghost is 0.20 for OXGBoost and Random Forest in the Ghost pipeline. Ghost, OXGBoost, OXGBoost + Ghost, and SMOTE + CatBoost obtained an AUROC of 0.999. However, since SMOTE + CatBoost has the lowest SD it is ranked as the best performing system based on AUROC. The best performance in terms of AUPRC, is OXGBoost and OXGBoost + Ghost which both resulted in an AUROC of 0.987. Note that the SD of OXGBoost is slightly lower indicating more stable performance. The AUPRC of 0.97 for SVM-RFE as mentioned in [1] could not be reproduced. Instead, an AUPRC of 0.675 was obtained. The best performance in terms of EE is obtained by OXGBoost which is 0.999 ± 0.0009. Note the low SD of OXGBoost on the PaySim data set which suggests very stable performance over the folds. In terms of EE OXGBoost, Ghost, SMOTE + CatBoost and Ghost + OXGBoost significantly outperform the SVM-RFE both in terms of stability and mean EE.

The SD over the folds of all systems performing on the PaySim data set is significantly lower than the SD of the AUROC over the folds of the Europe data set. The PR-curves and ROC-curves of the pipelines on the Europe data set are shown in Fig. 23, 24, 25, and 26. These curves are generated by the evaluation of fold 5 during cross-fold validation. Therefore the AUROC/AUPRC shown on the bottom-right of these figures is not the same as in Table 9.

4.2.3 Overall difference in performance between methods

A contingency table was created from the individual performances over the folds where the performances in terms of EE, AUROC and AUPRC were multiplied by the test size of the data set. Note that the test data is 20% of the data size. The performance of SVM-RFE was left out from this analysis since the AUPRC as presented in [1] could not be reproduced. All differences between methods and data sets are highly significant due to the large number of samples in the data sets for EE and AUPRC. For AUROC the result was not significant with $X^2(3, N = 3532879) = 1.233, p = 0.745$.

4.3 Optimizing classification performance based on economic efficiency vs AUROC and AUPRC

As previously mentioned, it is an important insight to measure the accepted loss (if any) in terms of AUROC and AUPRC, when optimizing a classification method on EE. To measure this, an optimization parameter of a classification algorithm is manipulated. Since the number of trees in a random forest is important for the performance of classification, Ghost with a random forest was used. All parameters were left the same as in the Ghost algorithm. Only the number of trees were manipulated from 50 to 450 using 50 spacing. For all parameter settings the EE, AUPRC, and AUROC were measured over a 5-fold cross-validation on the Europe data set. The results are shown in Fig. 16. This figure shows that the AUROC on average is higher than the AUPRC and the EE. Furthermore, we can observe a slight increase in AUPRC while the EE decreases. However, this is a really small decrease/increase in performance.

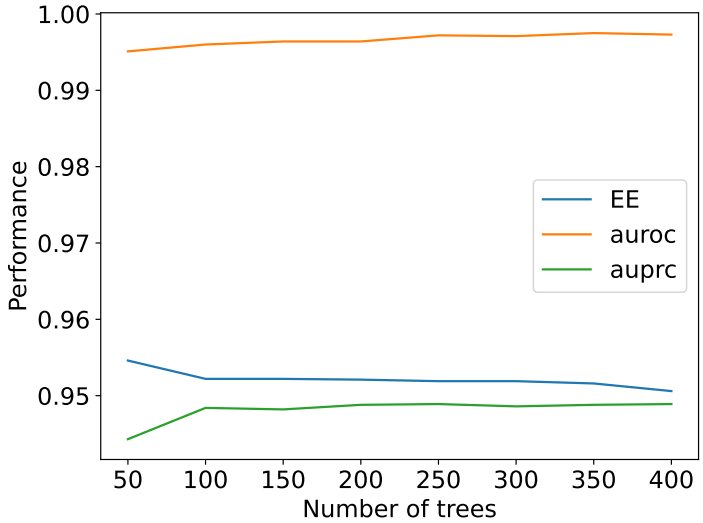

Figure 16: Economic efficiency, AUROC, and AUPRC of the Ghost algorithm with a random forest. The x-axis represents the number of trees that were used in the random forest. The other parameters of the random forest remained the same as presented in Table. 8. The performance is measured on the Europe data set using a 5-fold cross-validation

As mentioned in the introduction, precision and recall represent a trade-off. To analyze how the economic efficiency relates to this trade-off, precision and recall are measured for a variety of decision thresholds ranging from 0.01 to 0.99 using 0.01 spacing. For this experiment, OXGBoost was used meaning the parameter settings of this classifier are as presented in Table. 3. The results are shown in

Fig. 17. From this figure one can observe that as precision increases, recall decreases. However, the economic efficiency seems to increase when the precision also increases and remains stable after the decision threshold reaches 0.1.

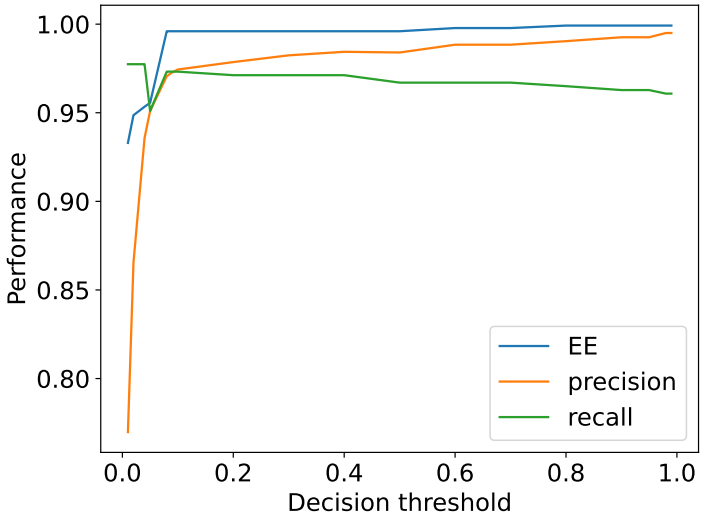

Figure 17: Recall, precision, and EE of OXGBoost for different decision thresholds. The x-axis represents the decision threshold where the y-axis is the performance. The performance is measured on the Europe data set using 5-fold cross-validation.

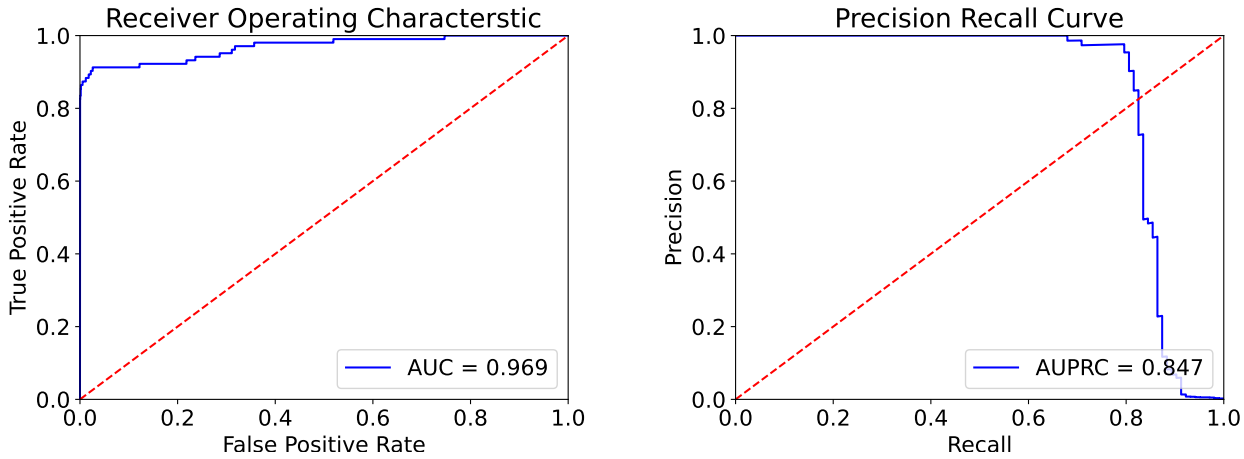

Figure 18: AUROC- (left) and AUPRC (right) curve Europe data set using Ghost

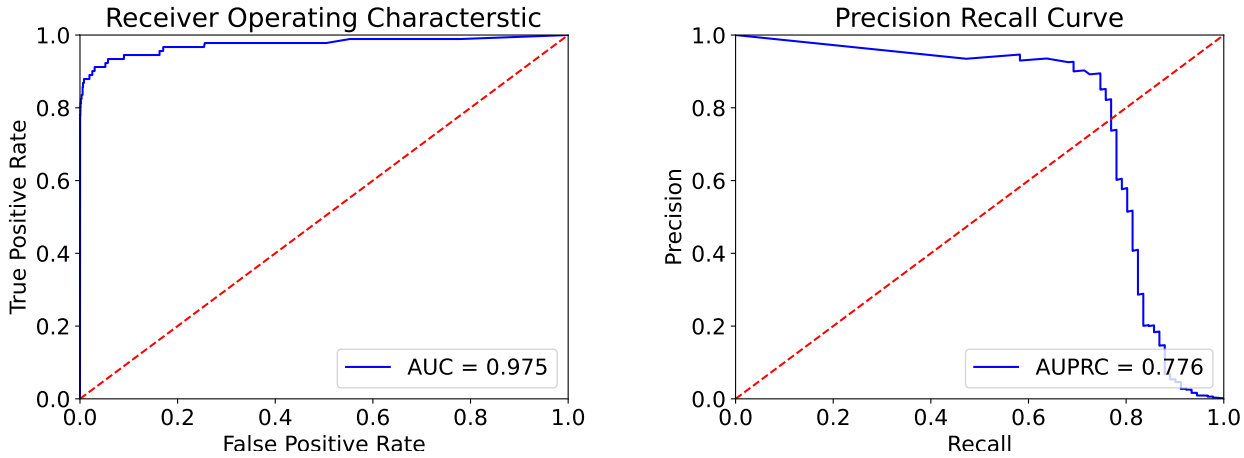

Figure 19: AUROC- (left) and AUPRC (right) curve Europe data set using SVM-RFE

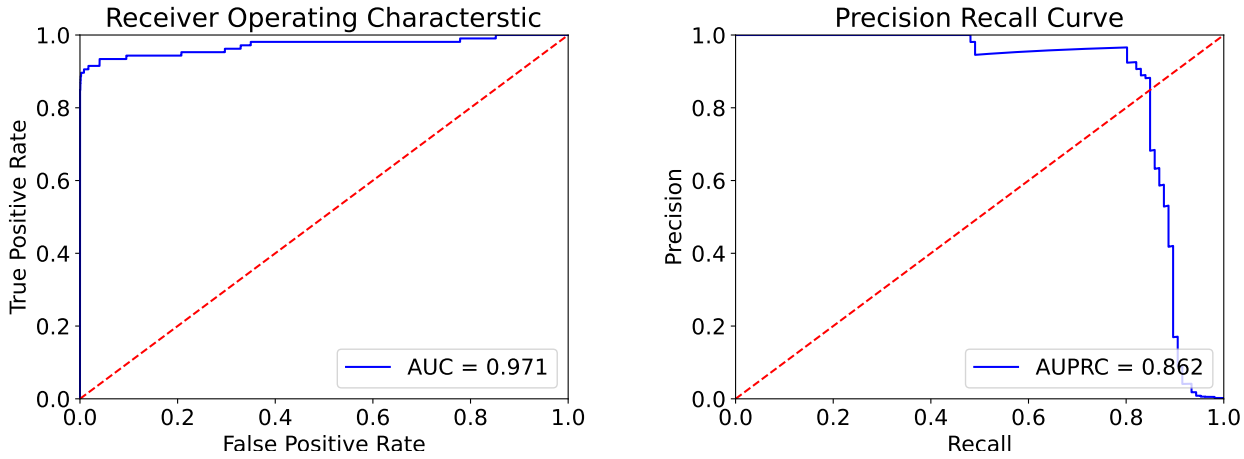

Figure 20: AUROC- (left) and AUPRC (right) curve Europe data set using SMOTE + CatBoost

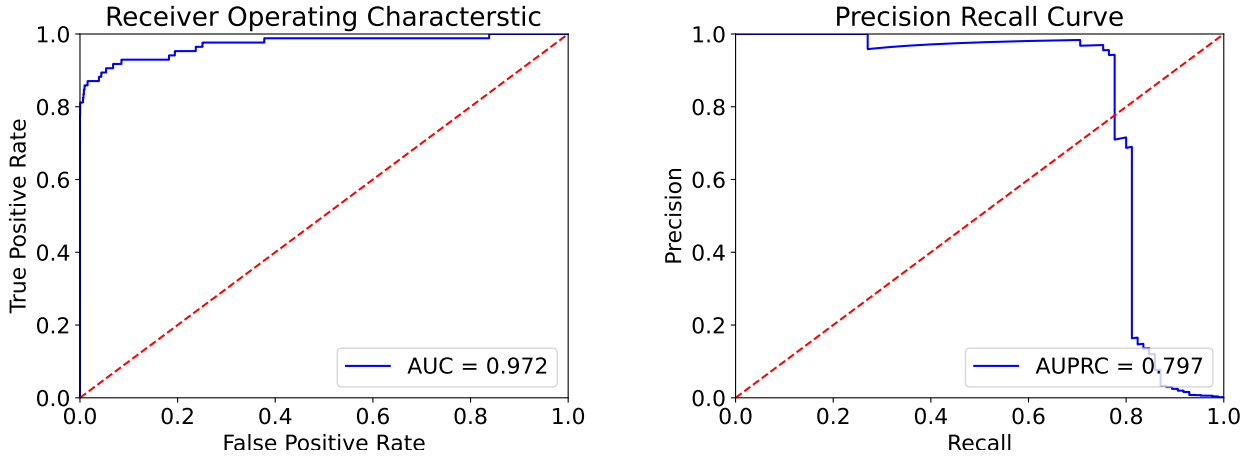

Figure 21: AUROC- (left) and AUPRC (right) curve Europe data set using OXGBoost

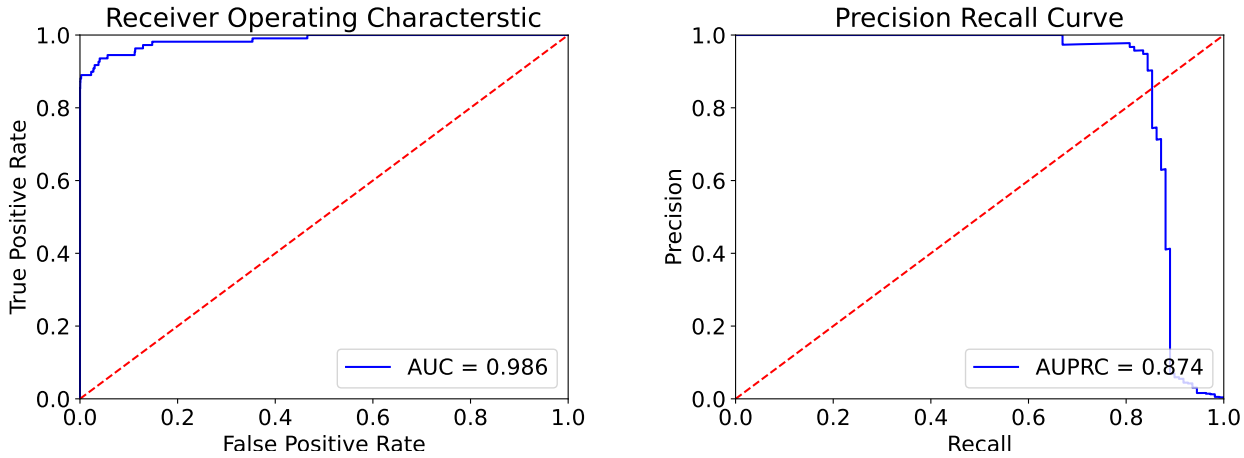

Figure 22: AUROC- (left) and AUPRC (right) curve Europe data set using OXGBoost + Ghost

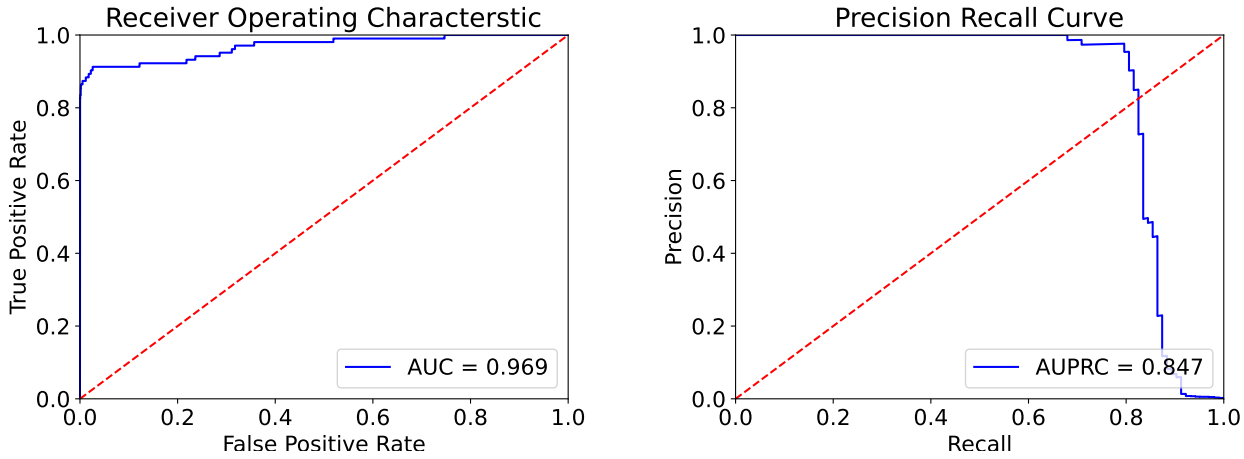

Figure 23: AUROC- (left) and AUPRC (right) curve PaySim data set using Ghost

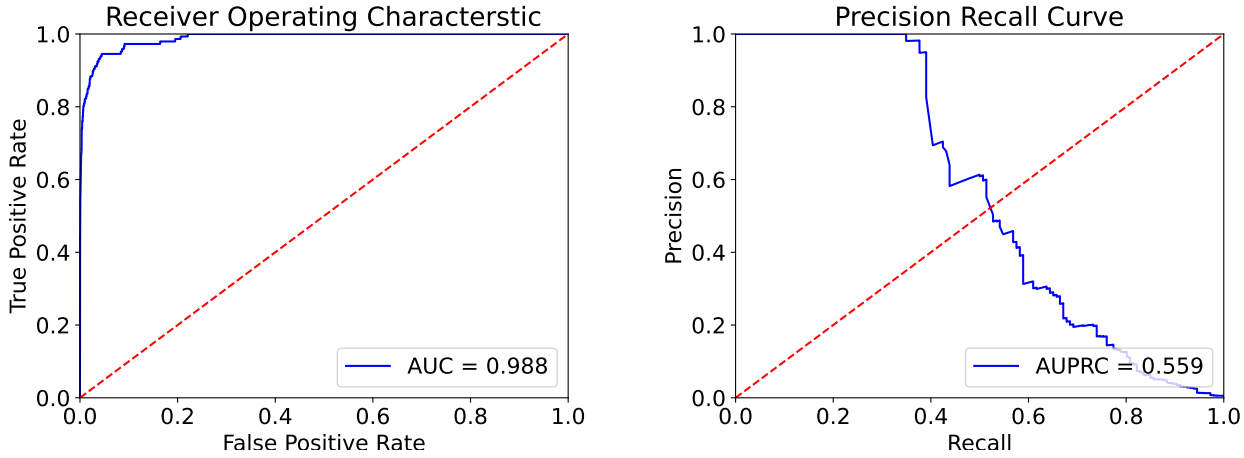

Figure 24: AUROC- (left) and AUPRC (right) curve PaySim data set using SVM-RFE

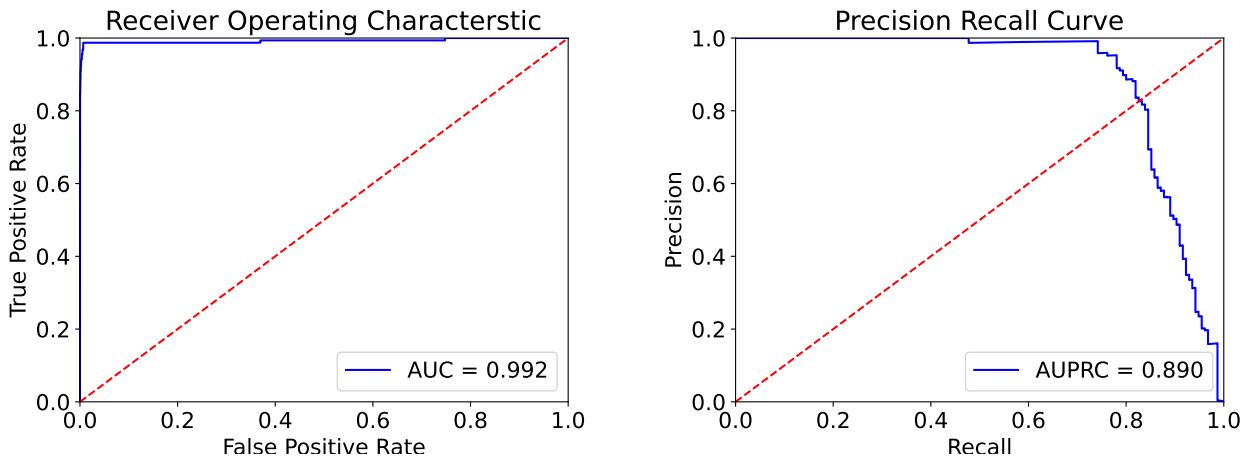

Figure 25: AUROC- (left) and AUPRC (right) curve PaySim data set using SMOTE + CatBoost

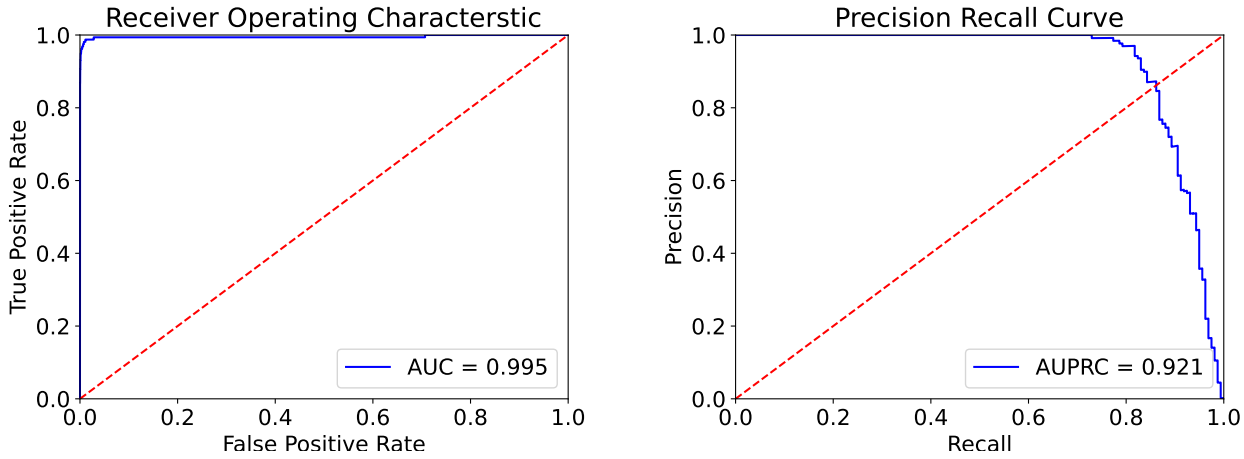

Figure 26: AUROC- (left) and AUPRC (right) curve PaySim data set using OXGBoost

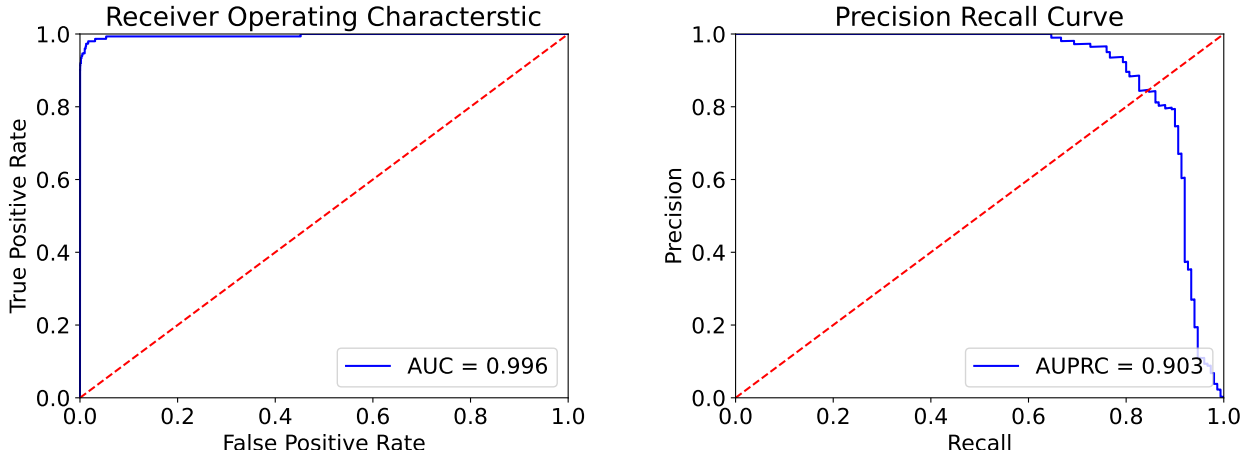

Figure 27: AUROC- (left) and AUPRC (right) curve PaySim data set using Ghost + OXGBoost

5 Discussion

This section reflects on Economic Efficiency and the best peforming pipeline in terms of the metrics measured. It also summarizes the thesis and its main contributions. It ends with potential future directions for CFD.

5.1 On the economic efficiency

As mentioned in the results section, the best-performing pipeline depends on the metric we are using to evaluate performance. A CFD system trained to maximize the number of correct classifications does not mean this system optimizes the financial gain for the credit card service company. From the results, we see that although the AUROC is correlated with the EE, a higher AUROC/AUPRC does not necessarily mean the EE is better. While the change in EE might seem insignificant, note that the EE is normalized by the EE achieved by perfect classification. Depending on the sum of the amounts of the transactions, this might result in a significant financial gain. Due to the significant financial gain a credit card company can benefit from such small changes in correct numbers of classifications in terms of AUROC, I argue that besides looking at the number of correct classifications, when comparing the performance of CFD systems one must also take into consideration the EE. This statement is supported by the high AUROC that the SVM-RFE method obtains. While this AUROC is considered a good result, the economic efficiency and AUPRC is significantly lower for this method than for the other methods indicating that SVM-RFE has low recall. The values that credit card service companies use for α, β, δ, and γ are usually not known. Therefore it is assumed that these values vary per company that uses a CFD system. Due to a lack of known values for these parameters, this thesis uses the values of [23] and [27]. However, every credit card service company should measure the performance of their systems with their intern values used for these parameters.

The correct classification of the minority class is often more important than the classification of the majority class in different fields. In medical diagnosis Type II errors (false negatives) are more important to eliminate than Type I errors (false positives) since incorrectly classifying sick patients can be catastrophic whereas diagnosing a healthy patient is unwanted but manageable with further testing on the patient. Expressing the importance of these errors numerically is a difficult task. In CFD this is easier since we can express the importance of a transaction based on the amount of the transaction. Although several financial metrics for the evaluation of CFD pipelines are known in the literature, the EE as proposed in this thesis distinguishes itself from other financial metrics because all elements from the confusion matrix is given a financial weight with respect to the amount of the transactions and internal costs for credit card service companies.

An analysis to measure the accepted loss in terms of AUROC/AUPRC when optimizing classification based on EE was done. When the number of trees in a random forest increased, the performance of all three metrics (EE, AUROC, AUPRC) was not affected significantly. However, manipulating the decision threshold had a minor effect on the economic efficiency related to the precision and recall. If the decision threshold increases, the precision increases which means the false negative rate decreases. Since false negatives in credit card fraud detection come with significant financial losses, economic efficiency seems to be slightly correlated with precision. However, the expectancy is that this is strongly dependent on the data set that is used. If the amount of transactions increases, the economic efficiency will be more correlated with the recall since false negatives will significantly decrease economic efficiency. This is also dependent on the cost that we set for α since false positives are penalized based on α. However, the setting of the decision threshold is a trade-off that the credit card service company should set based on the wanted trade-off of precision and recall.

5.2 Boosting algorithms in credit card fraud

Due to several previously mentioned reasons, the main performance estimator of this thesis is the EE. In terms of EE, the best-performing pipeline is OXGBoost and SMOTE + CatBoost on the Europe data set and OXGBoost for the PaySim data set. Ghost and Ghost + OXGBoost obtain similar results in terms of EE, AUROC, AUPRC, and its standard deviation. This indicates that optimizing the decision threshold with Ghost is not very beneficial for the boosting algorithm.

The OXGBoost shows the best performance on the imbalanced data sets without a data-level balancing method. The way OXGBoost handles the class imbalance is by changing the weights of the data points (which represent the chance of becoming training data for the next model) according to the distribution within the classes. Depending on the data set this algorithm-level balancing method seems to outperform the data-level methods. However, for the PaySim data set the data-level balancing method seems to slightly outperform the other pipelines in terms of EE.

A possible reason for the high performance of OXGBoost is that since the balancing method is applied during the training stage, the voting system of the boost DTs optimizes itself based on an assumed decision threshold of 0.5. Every time a data point is misclassified the chance increases that this data point is used for the next model that is trained. Combined with the `scale_pos_weight` parameter which increases the chance of a minority data point ending up in the data of the next model that is trained, the cumulative weights of the minority class become similar to the cumulative weights of the majority class. This is similar to oversampling of the minority class. If the imbalance ratio is extremely high, this might result in overfitting since the data points in the minority class will repeatedly be used as training data. This effect was not visible in the results of this study since the performance on the unseen test set was almost optimal.

5.3 Limitations of CFD literature

While there exists rich literature on CFD, the motivation and resources for the projects are limited in comparison with the resources for projects by credit card service companies. The reason is that the potential financial gain of a high-performance CFD system is higher for credit card companies than it is for research teams of universities. While literature presents many cooperations with credit card companies, it is not beneficial for these companies to share their current state-of-the-art CFD system due to rivals of these companies being able to copy the techniques they use. When credit card companies decide to cooperate with university research teams, they are provided only a segment of the data set they possess. And since these data sets are not publicly available due to personal data not being allowed to be public, it is difficult to compare the performance of these systems. The only method to compare these pipelines is by recreating the systems. However, testing whether the implementation is identical is impossible since the performance on these specific data sets cannot be matched due to the unavailability of the data set itself. Therefore it is encouraged for these types of studies to also provide the performance of CFD systems on publicly available data sets like the Europe data set.

5.4 Conclusion

This study compared state-of-the-art machine learning methods used for determining whether credit card transactions are fraudulent or genuine. A major challenge here is that the data sets are highly imbalanced due to fraud transactions occurring significantly less often than genuine transactions. An unwanted result of training machine learning classifiers on imbalanced data is that the classification

tends to be biased towards the majority data class. Therefore a data balancing technique is required to prevent fraudulent transactions from being classified as genuine. The evaluation metrics of pipelines in CFD is often measured through the number of correct classifications. Examples of such metrics are precision, recall, F_1, AUROC, and AUPRC. Since not every credit card transaction is equally important to classify correctly (based on the amount of the transaction) it is important to take this unequal weight in transactions into consideration when evaluating the performance of a pipeline. In addition, every type of classification in a confusion matrix comes with certain financial costs or gains which also need to be taken into consideration to get a realistic view of the economic efficiency of using a certain pipeline.

The economic efficiency proposed in this study differs from the known financial metrics proposed in literature since it takes all elements from the confusion matrix into account for the calculation of the financial gain. Where most financial metrics only base the financial gain on the true positives and false negatives, the EE proposed here also calculates the financial gain of both the false positives and true negatives. The reason this should be taken into consideration is that, based on expert advice, these incorrect classified transactions come with a financial loss. While this loss is difficult to determine and is not a constant value, it is important to quantify these types of classifications. Since financial institutions have different values for the parameters in the EE, every credit card service company should quantify their financial loss using the intern values used in the company.

This study also tested a recently published method from another domain called Ghost which overcomes the problem of class imbalance by finding an optimal threshold by maximizing Cohen's Kappa. While this method obtained state-of-the-art results on two frequently used credit card fraud data sets in the literature, the benefit of using this method in combination with an XGBoost classification method is insignificant. The outcome of the experiments was that the best-performing systems in terms of the traditional CFD evaluation metrics, do not necessarily maximize the financial gain for a credit card service company. Furthermore, the best pipeline in terms of EE was OXGBoost and SMOTE + CatBoost. Another finding is that when optimizing the EE based on the decision threshold, depending on the amounts of the credit card transactions, the EE correlates with either the recall or the precision. Since false negatives come with a significant financial loss, the EE will correlate more with the recall if the amounts of transactions are higher. The EE will correlate with the precision when the transactions are lower since the financial loss of false positives are lower if the amount of the transactions decrease. This trade-off is also dependent on the value of α which represents the financial manual classification cost and is a fixed financial loss on the false positives.

5.5 Summary of Main Contributions

- This thesis presents a realistic economic to measure the performance in terms of financial gain of a pipeline on a credit card fraud detection data set and relates this metric to the existing metrics for classification of imbalanced data.

- This thesis provides a comparison of state-of-the-art methods within the credit card fraud detection literature. This comparison is done both with the more standard metrics (AUROC, AUPRC) but also on the proposed economic efficiency. In terms of Economic Efficiency, boosting methods show state-of-the-art results.

- This thesis introduces a recently published pipeline used in other domains to overcome the problem of class imbalance in the field of credit card fraud detection which showed close to state-of-the-art results.

5.6 Future Work

During this research project, several additions to CFD that did not fit within the scope of this project came to mind.

The first possible future direction is to make better use of the feature *time*. As can be seen from Fig 13, several spikes exist in a plot of fraud transactions over time. A useful additional feature could be to include *delta time*. Every time a fraud transaction is detected, this timestamp can be set as $t = 0$. If we get a transaction that has a time close to $t = 0$ we might decide to decrease the decision threshold to create a 'safe' classification strategy. However, a big disadvantage of this method is obviously that we increase the chance of obtaining false positives.

Another future direction could be to study the utility of CFD data sets. The utility is defined as the probability of a transaction being a fraud, multiplied by the amount of that transaction. The utility can be considered as the financial risk of a transaction. However, even though the probability of certain transactions might be low, if the amount of the transaction is very high (similar to the transactions in the Europe and PaySim data set where several transactions have a much higher amount), this utility will always be high since the scale of the probability is not equal to the distribution of the amounts of the transactions. A research project on utilizing different thresholds based on certain utility ranges could be interesting. If the utility increases, we could decrease the decision threshold to obtain a more 'safe' strategy of classification in terms of economic efficiency.

Finally, to better fine-tune the economic efficiency proposed in this thesis and to get a more realistic financial metric, the parameters α and β can be predicted through regression. These parameters are currently both set to €150,- as mentioned in [27]. However, according to icscards, these values are not constant and depend on several factors. If these values were to be predicted, Economic Efficiency becomes more realistic. To do so, the costs that it takes for manual classification on every transaction (α) and the costs for retrieving lost money due to detected fraud transactions (β) are needed.

Acknowledgments

I thank ICScards for answering questions about the practical vision of credit card fraud systems

Bibliography

[1] N. Rtayli and N. Enneya, "Enhanced credit card fraud detection based on SVM-recursive feature elimination and hyper-parameters optimization," *Journal of Information Security and Applications*, vol. 55, no. September, p. 102596, 2020.

[2] "The Average Number of Credit Card Transactions Per Day Year." https://www.cardrates.com/advice/number-of-credit-card-transactions-per-day-year/. Accessed: 2022-17-5.

[3] "Credit Card Fraud Statistics quick credit card theft statistics." https://shiftprocessing.com/credit-card-fraud-statistics/. Accessed: 2022-17-5.

[4] A. Joshi, S. Soni, and V. Jain, "An experimental study using unsupervised machine learning techniques for credit card fraud detection,"

[5] A. O. Hoffmann and C. Birnbrich, "The impact of fraud prevention on bank-customer relationships: An empirical investigation in retail banking," *International journal of bank marketing*, 2012.

[6] D. S. Sisodia, N. K. Reddy, and S. Bhandari, "Performance evaluation of class balancing techniques for credit card fraud detection," in *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pp. 2747–2752, IEEE, 2017.

[7] S. T. King, N. Scaife, P. Traynor, Z. Abi Din, C. Peeters, and H. Venugopala, "Credit card fraud is a computer security problem," *IEEE Security & Privacy*, vol. 19, no. 2, pp. 65–69, 2021.

[8] M. Rezapour, "Anomaly detection using unsupervised methods: credit card fraud case study," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 11, 2019.

[9] "Australian Payment Fraud 2020 ." https://www.auspaynet.com.au/sites/default/files/2020-08/Fraud_Report_2020.pdf. Accessed: 2022-17-5.

[10] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[11] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[12] A. Azaria, A. Richardson, S. Kraus, and V. Subrahmanian, "Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 2, pp. 135–155, 2014.

[13] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International journal of pattern recognition and artificial intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

[14] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[15] R. C. Prati, G. E. Batista, and M. C. Monard, "Class imbalances versus class overlapping: an analysis of a learning system behavior," in *Mexican international conference on artificial intelligence*, pp. 312–321, Springer, 2004.

[16] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[17] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[18] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[19] J. Hernández-Orallo, P. Flach, and C. Ferri Ramírez, "A unified view of performance metrics: Translating threshold choice into expected classification loss," *Journal of Machine Learning Research*, vol. 13, pp. 2813–2869, 2012.

[20] J. Chen, C.-A. Tsai, H. Moon, H. Ahn, J. Young, and C.-H. Chen, "Decision threshold adjustment in class prediction," *SAR and QSAR in Environmental Research*, vol. 17, no. 3, pp. 337–352, 2006.

[21] M. Buckland and F. Gey, "The relationship between recall and precision," *Journal of the American society for information science*, vol. 45, no. 1, pp. 12–19, 1994.

[22] "Accuracy, Confusion Matrix, Precision/Recall, Oh My!." `https://medium.com/swlh/precision-recall-tradeoff-29e57ce22a13`. Accessed: 29-8-2022.

[23] A. G. de Sá, A. C. Pereira, and G. L. Pappa, "A customized classification algorithm for credit card fraud detection," *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 21–29, 2018.

[24] E. Caldeira, G. Brandao, and A. C. Pereira, "Fraud analysis and prevention in e-commerce transactions," in *2014 9th Latin American Web Congress*, pp. 42–49, IEEE, 2014.

[25] J. F. Júnior, A. Pereira, W. Meira Júnior, and A. Veloso, "Methodology for fraud detection in electronic transactions," in *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, pp. 289–292, 2012.

[26] P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo, "Distributed data mining in credit card fraud detection," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 6, pp. 67–74, 1999.

[27] L. Torgo, E. Lopes, and S. L.-i. Porto, "Utility-Based Fraud Detection," pp. 1517–1522.

[28] E. Lopez-Rojas, A. Elmir, and S. Axelsson, "Paysim: A financial mobile money simulator for fraud detection," in *28th European Modeling and Simulation Symposium, EMSS, Larnaca*, pp. 249–255, Dime University of Genoa, 2016.

[29] M. Habibpour, H. Gharoun, M. Mehdipour, A. Tajally, H. Asgharnezhad, A. Shamsi, A. Khosravi, M. Shafie-Khah, S. Nahavandi, and J. P. Catalao, "Uncertainty-aware credit card fraud detection using deep learning," *arXiv preprint arXiv:2107.13508*, 2021.

[30] A. M. Mubalaike and E. Adali, "Deep learning approach for intelligent financial fraud detection system," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, pp. 598–603, IEEE, 2018.

[31] S. Sa'adah and M. S. Pratiwi, "Classification of customer actions on digital money transactions on paysim mobile money simulator using probabilistic neural network (pnn) algorithm," in *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pp. 677–681, IEEE, 2020.

[32] E. Balagolla, W. Fernando, R. Rathnayake, M. Wijesekera, A. Senarathne, and K. Abeywardhana, "Credit card fraud prevention using blockchain," in *2021 6th International Conference for Convergence in Technology (I2CT)*, pp. 1–8, IEEE, 2021.

[33] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[34] "Principal Component Analysis (PCA) Explained Visually with Zero Math ." `https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d`. Accessed: 2022-15-9.

[35] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15849–15854, 2019.

[36] S. Taneja, B. Suri, and C. Kothari, "Application of balancing techniques with ensemble approach for credit card fraud detection," in *2019 International Conference on Computing, Power and Communication Technologies (GUCON)*, pp. 753–758, IEEE, 2019.

[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[38] N. Mqadi, N. Naicker, and T. Adeliyi, "A smote based oversampling data-point approach to solving the credit card data imbalance problem in financial fraud detection," *International Journal of Computing and Digital Systems*, vol. 10, no. 1, pp. 277–286, 2021.

[39] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*, pp. 878–887, Springer, 2005.

[40] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 3, no. 1, pp. 4–21, 2011.

[41] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328, IEEE, 2008.

[42] F. Provost, "Machine learning from imbalanced data sets 101," *Proceedings of the AAAI'2000 Workshop on . . .*, p. 3, 2000.

[43] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[44] C. Ferri, J. Hernández-Orallo, and P. Flach, "Setting decision thresholds when operating conditions are uncertain," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 805–847, 2019.

[45] C. Esposito, G. A. Landrum, N. Schneider, N. Stiefl, and S. Riniker, "Ghost: adjusting the decision threshold to handle imbalanced data in machine learning," *Journal of Chemical Information and Modeling*, vol. 61, no. 6, pp. 2623–2640, 2021.

[46] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[47] E. E. Ogheneovo and P. A. Nlerum, "Iterative dichotomizer 3 (id3) decision tree: A machine learning algorithm for data classification and predictive analysis," *International Journal of Advanced Engineering Research and Science*, vol. 7, no. 4, pp. 514–521, 2020.

[48] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[49] W. N. H. W. Mohamed, M. N. M. Salleh, and A. H. Omar, "A comparative study of reduced error pruning method in decision tree algorithms," in *2012 IEEE International conference on control system, computing and engineering*, pp. 392–397, IEEE, 2012.

[50] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[51] "Support-vector machine ." `https://en.wikipedia.org/wiki/Support-vector_machine`. Accessed: 2022-15-9.

[52] "Understanding Feedforward Neural Networks ." `https://learnopencv.com/understanding-feedforward-neural-networks/`. Accessed: 2022-19-9.

[53] E. Esenogho, I. D. Mienye, T. G. Swart, K. Aruleba, and G. Obaido, "A neural network ensemble with feature engineering for improved credit card fraud detection," *IEEE Access*, vol. 10, pp. 16400–16407, 2022.

[54] X. Yu, X. Li, Y. Dong, and R. Zheng, "A deep neural network algorithm for detecting credit card fraud," in *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pp. 181–183, IEEE, 2020.

[55] L. Breiman, "Bias, variance, and arcing classifiers," tech. rep., Tech. Rep. 460, Statistics Department, University of California, Berkeley ..., 1996.

[56] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[57] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[58] Y. Freund, "An adaptive version of the boost by majority algorithm," *Machine learning*, vol. 43, no. 3, pp. 293–318, 2001.

[59] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[60] A. Vezhnevets and O. Barinova, "Avoiding boosting overfitting by removing confusing samples," in *European Conference on Machine Learning*, pp. 430–441, Springer, 2007.

[61] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[62] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial," *arXiv preprint arXiv:1905.12787*, 2019.

[63] D. Almhaithawi, A. Jafar, and M. Aljnidi, "Example-dependent cost-sensitive credit cards fraud detection using smote and bayes minimum risk," *SN Applied Sciences*, vol. 2, no. 9, pp. 1–12, 2020.

[64] A. A. Taha and S. J. Malebary, "An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine," *IEEE Access*, vol. 8, pp. 25579–25587, 2020.

[65] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, "Boosting methods for multi-class imbalanced data classification: an experimental review," *Journal of Big Data*, vol. 7, no. 1, pp. 1–47, 2020.

[66] B. X. Wang and N. Japkowicz, "Boosting support vector machines for imbalanced data sets," *Knowledge and information systems*, vol. 25, no. 1, pp. 1–20, 2010.

[67] C. V. Priscilla and D. P. Prabha, "Influence of optimizing xgboost to handle class imbalance in credit card fraud detection," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 1309–1315, IEEE, 2020.

[68] B. Ozenne, F. Subtil, and D. Maucort-Boulch, "The precision-recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases," *Journal of Clinical Epidemiology*, vol. 68, no. 8, pp. 855–859, 2015.

[69] P. Flach and M. Kull, "Precision-recall-gain curves: Pr analysis done right," *Advances in neural information processing systems*, vol. 28, 2015.

[70] D. Hand and P. Christen, "A note on using the f-measure for evaluating record linkage algorithms," *Statistics and Computing*, vol. 28, no. 3, pp. 539–547, 2018.

[71] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[72] U. Porwal and S. Mukund, "Credit Card Fraud Detection in e-Commerce: An Outlier Detection Approach," 2018.

[73] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*, pp. 413–422, IEEE, 2008.

[74] H. R. Sofaer, J. A. Hoeting, and C. S. Jarnevich, "The area under the precision-recall curve as a performance metric for rare binary events," *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 565–577, 2019.

[75] E. Kim, J. Lee, H. Shin, H. Yang, S. Cho, S. kwan Nam, Y. Song, J. a. Yoon, and J. il Kim, "Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning," *Expert Systems with Applications*, vol. 128, pp. 214–224, 2019.

[76] E. Caldeira, G. Brandão, H. Campos, and A. Pereira, "Characterizing and evaluating fraud in electronic transactions," *Proceedings - 2012 8th Latin American Web Congress, LA-WEB 2012*, pp. 115–122, 2012.

[77] A. Correa Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature engineering strategies for credit card fraud detection," *Expert Systems with Applications*, vol. 51, pp. 134–142, 2016.

[78] A. Correa Bahnsen, D. Aouada, and B. Ottersten, "Ensemble of example-dependent cost-sensitive decision trees," *arXiv e-prints*, pp. arXiv–1505, 2015.

[79] J. Forough and S. Momtazi, "Ensemble of deep sequential models for credit card fraud detection," *Applied Soft Computing*, vol. 99, p. 106883, 2021.

[80] Z. Li, G. Liu, and C. Jiang, "Deep representation learning with full center loss for credit card fraud detection," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 2, pp. 569–579, 2020.

[81] J. Chen, Y. Shen, and R. Ali, "Credit card fraud detection using sparse autoencoder and generative adversarial network," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 1054–1059, IEEE, 2018.

[82] Y. Xie, A. Li, L. Gao, and Z. Liu, "A heterogeneous ensemble learning model based on data distribution for credit card fraud detection," *Wireless Communications and Mobile Computing*, vol. 2021, 2021.

[83] A. Singh, R. K. Ranjan, and A. Tiwari, "Credit Card Fraud Detection under Extreme Imbalanced Data: A Comparative Study of Data-level Algorithms," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 34, no. 4, pp. 571–598, 2021.

[84] F. Itoo, S. Singh, *et al.*, "Comparison and analysis of logistic regression, naïve bayes and knn machine learning algorithms for credit card fraud detection," *International Journal of Information Technology*, vol. 13, no. 4, pp. 1503–1511, 2021.

[85] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[86] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[87] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[88] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[89] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[90] J. J. Chen, C. A. Tsai, H. Moon, H. Ahn, J. J. Young, and C. H. Chen, "Decision threshold adjustment in class prediction," *SAR and QSAR in Environmental Research*, vol. 17, no. 3, pp. 337–352, 2006.

[91] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang, and C. Jiang, "Random forest for credit card fraud detection," in *2018 IEEE 15th international conference on networking, sensing and control (ICNSC)*, pp. 1–6, IEEE, 2018.

[92] 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*, vol. 12, pp. 2825–2830, 2011.

Appendices

A Run instructions

The code for the execution of the experiments is available at https://github.com/rikvegter/fraud_detection
Setup:

1. Clone the github repository
2. Download the zipped data from `https://www.dropbox.com/home/Master\%20Thesis\%20data` and unzip in same directory
3. Create a folder 'data' and put both data files in this folder.
4. Download requirements with the following command:
   `pip3 install -r requirements.txt`
5. The `imblearn` library should be downloaded separately. Run the following command:
   `pip install -U imbalanced-learn`.
6. Follow run instructions

Run instructions:

1. `python3 main.py`
2. Choose data set you want to test by typing 1 or 2 followed by hitting 'Enter'
3. Choose pipeline you want to test by typing 1, 2, 3 or 4 followed by hitting 'Enter'. The code was tested using Python 3.8.8 and above on Linux and MacOS.