Credit Card Fraud Detection Using a Machine Learning Approach

Student Numbers: 133348, 116853, 133816 and 133802

Data Mining, Machine Learning, Deep Learning, 15 pages

MSc in Business Administration and Information Systems - Data Science

Supervisor: Raghava Rao Mukkamala

Copenhagen, 11.06.2020

Abstract

Credit card fraud causes serious economic damage and destabilizes the trust in industries. Due to the complexity of the problem, identifying fraudulent transactions based on manual techniques has shown to be ineffective and inefficient. In this paper, different supervised machine learning based approaches are tested in order to identify a high number of fraudulent credit card transactions. The goal is to contain criminal action and to provide an economically viable solution to financial institutions. The research was conducted based on a dataset with credit card transactions from European cardholders, where different sampling methods (SMOTE, Borderline-SMOTE and ADASYN) are applied to handle the imbalance of the data. Subsequently, a Random Forest, a Logistic Regression, SVM, and an Artificial Neural Network (ANN) are developed. The work shows that all model types are able to prevent fraudulent transactions to some degree and contribute to contain criminal action. However, only the Random Forest model in combination with Borderline-SMOTE and the SVM model with SMOTE produces economically feasible results that could be implemented by a financial institution.

Keywords: Credit Card Fraud Detection, Imbalanced Data, Classification, Supervised Machine Learning, Data Science

1 Introduction (133348)

In 2019, credit card transactions and migrations into a digital or mobile wallet accounted for 66 per cent of all payments in e-commerce and 42.4 per cent for points of sale (POS). According to FIS's Global Payment Report, by 2023 this value is expected to increase even further by 5 percentage points in e-commerce and 9.5 percentage points for POS (FIS 2020).

These transactions provide a target for financial fraudsters, leading to serious problems through direct economic damage, the erosion of trust and therefore causing a potentially destabilizing long-term effect on industries as a whole (Ngai et al. 2010, West & Bhattacharya 2016). Financial fraud can be defined as "[a] deliberate act [..] that is contrary to law, rule, or policy with the intent to obtain unauthorized financial benefit [. . .]" (The Ohio State University 2006, p.1). Three of the major financial fraud types are insurance fraud, corporate fraud and bank fraud, which include credit card fraud (West & Bhattacharya 2016).

Research divides credit card fraud into two categories, *application fraud* (someone else applies as a new cardholder with a stolen identity) and *behavioral fraud* (someone else uses stolen credit card illegitimately) (Abdallah et al. 2016, Bhattacharyya et al. 2011). A contributing factor to today's high prevalence of behavioral fraud is the rising amount of online transactions, where so-called *card holder not present* fraud occurs. It describes a situation where credit card details of another person are used unknowingly (Bhattacharyya et al. 2011). The high and continuously rising volume of credit card transactions entails a lot of potential for fraudulent activity. In 2016, the total amount of global fraudulent transactions by credit cards issued in single euro payments area (SEPA), caused a total damage of 1.8 billion Euro, constituting a share of 0.041 per cent of all transactions. As the merchant typically bears all the cost from charge-back over shipping costs to administrative costs caused by a fraudulent transaction, it is vital for them to rely on financial fraud detection to reduce these incidental costs (Quah & Sriganesh 2008).
Financial fraud detection comprises the disclosure of fraudulent procedures by differentiating fraudulent from authentic transactions (Ngai et al. 2010). In the past, this was done based on manual techniques. Due to the complexity of the problem, this approach became unreliable and inefficient (West & Bhattacharya 2016). Machine learning based approaches show the ability to determine unusual patterns in large datasets and can therefore play a vital role to solve this problem (Ngai et al. 2010).

Within the scope of this paper, different supervised machine learning methods are applied to detect fraudulent credit card transactions. The outcomes of the methods are compared and evaluated to identify the best model. The objective is to develop an economically viable solution that detects as many fraudulent transactions as possible, thereby preventing criminal action and saving the financial institutions and merchants money in the future. Additionally, the misclassified valid transactions shall be kept as low as possible to avoid inconvenience for the cardholders (Jain et al. 2020).

2 Related Work (116853)

This chapter starts by summarizing popular machine learning algorithms used in related studies. Then, important characteristics for dealing with credit card fraud are highlighted. Lastly, encountered challenges during preprocessing and different metrics for performance evaluation are compared.

In their comparative study, Delamaire et al. (2009) discuss the application of different credit card fraud detection techniques. Based on their findings, decision trees, genetic algorithms, bayesian networks and neural networks are among the most commonly used algorithms. Albashrawi & Lowell (2016) compare 65 studies about data mining techniques applied to credit card fraud. They note that, although commonly used, techniques such as logistic regression (LR), decision tree (DT), support-vector machine (SVM), artificial neural networks (ANN) and Bayesian networks may not always yield the best results.
Adding to the comparative studies of Delamaire et al. (2009) and Albashrawi & Lowell (2016), Table 1 summarizes applications of supervised machine learning methods in studies which analyzed credit card datasets:

Method / **Study**	k-NN	LR	Random Forest	DT	Naive Bayes	SVM	ANN
Whitrow et al. (2009)	x	x	x			x	
Bhattacharyya et al. (2011)	x	x			x		
Dal Pozzolo et al. (2014)				x		x	x
Randhawa et al. (2018)		x		x	x	x	x
Venkata Suryanarayana et al. (2018)		x	x	x			
Dornadula & Geetha (2019)		x	x	x			
Niu et al. (2019)	x	x	x	x		x	
Husejinović (2020)				x	x		

Table 1: Related work regarding credit card fraud sorted ascending by publication date

During preprocessing, Whitrow et al. (2009) aggregate credit card transactions in the dataset from the transaction level to the account level by setting a fixed time window for a series of transactions (1,3 and 7 days). They find that, for the classifiers which they employ, the metric minimum normalized loss (total cost divided by theoretical maximum cost) decreases when increasing the transaction time window.
In contrast, Bhattacharyya et al. (2011) use daily time stamp data for the credit card dataset available. Consequently, the derived attributes yield a lower precision across the employed supervised models (see Table 1). Accuracy and precision are applied as base parameters for performance measurement of the classification results throughout four of the eight compared studies (Bhattacharyya et al. 2011) (Venkata Suryanarayana et al. 2018), (Randhawa et al. 2018) (Dornadula & Geetha 2019).

In order to deal with imbalanced data, different sampling techniques can be employed to the training data. Researchers apply over-sampling with e.g. Synthetic Minority Oversampling Technique (SMOTE) or under-sampling (random under-sampling) to further increase their classifier performance (Dal Pozzolo et al. 2014) (Dornadula & Geetha 2019) (Randhawa et al. 2018)).

Anis et al. (2015) suggest to consider the following characteristics for choosing a classifier dealing with credit card fraud:

1. accurate identification of frauds - high true positive (TP) rate
2. genuine transactions should be correctly classified - low false positive (FP) rate
3. quick detection of frauds

When evaluating the performance of the supervised machine learning classifiers, literature mostly agrees that defining a proper cost measure is challenging. Accuracy for instance is seen as an inadequate measurement (considering the high class imbalances which are typical for fraud detection datasets), because even with a high accuracy the few fraudulent cases may be missed by the classifier (Whitrow et al. 2009) (Bhattacharyya et al. 2011) (Dal Pozzolo et al. 2014) (Niu et al. 2019)).
Hand et. al state that "misclassifying a fraudulent transaction as legitimate is much more serious than the reverse" (Hand et al. 2008, p.960). Hence, they suggest a cost weighted performance measure for classification, which combines weighted misclassifications on the dataset.

Similar to weighing the cost, the Matthews Correlation Coefficient (MCC) is deemed a suitable performance measure to deal with an imbalanced dataset (Dornadula & Geetha 2019, Randhawa et al. 2018). MCC is a statistical measure used for binary classification, which takes into account the true and false positives, and remains balanced even with different class sizes (Powers 2011). The formula is stated in equation 1.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{1}$$

MCC used by Dornadula & Geetha (2019) yields an almost perfect score of 0.9996 for a random forest model after applying SMOTE on the dataset. Randhawa et al. (2018) report a MCC score of 1 (meaning it perfectly predicts all transactions) with their hybrid model using Adaptive Boosting and majority voting combination methods (combining multiple algorithms for one method). Based on the performance results from the comparative study of Randhawa et al. (2018), a framework consisting of two propositions is developed:

- **Proposition 1:** Supervised machine learning can contribute to the containment of criminal actions by identifying fraudulent transactions better than guessing.

To quantify the classifier's contribution to the containment of criminal transactions, the above mentioned measure MCC is used (equation 1). The variables TP, FP, TN, FN, which appear in the calculation of this measures, is explained in Table 2.

Actual \ Predicted	**Positive**	**Negative**
Positive	# of True Positives (TP)	# of False Negatives (FN)
Negative	# of False Negatives (FP)	# of True Negatives (TN)

Table 2: The Confusion Matrix, a concept to systematically evaluate the performance of a binary classifier (Sokolova & Lapalme 2009).

- **Proposition 2:** Supervised machine learning constitutes an economically viable fraud detection solution for organizations. Thus, a supervised machine learning algorithm creates more economic benefits through the correct classification of fraudulent transactions than it creates cost by misclassifying non-fraudulent transactions as fraudulent.

To measure the second proposition, the economic viability score (EVS) compares the total value saved by the classifier with the total value lost by falsely blocking non-fraudulent transactions (compare formula 2). Whereas the value saved is calculated by multiplying TP with the average value of a fraudulent transaction ($\frac{1}{n_1}\sum_{i=1}^{n} y_i v_i$), the value lost is calculated by multiplying FP with the average value of a non-fraudulent transaction ($\frac{1}{n_0}\sum_{i=1}^{n}(1-y_i)v_i$). FNs are not considered as we assume that these transactions would not be detected without a classification algorithm in place. This metric allows an economic evaluation by weighing up implementation costs with savings.

$$EVS = TP \cdot \frac{1}{n_1}\sum_{i=1}^{n} y_i v_i - FP \cdot \frac{1}{n_0}\sum_{i=1}^{n}(1-y_i)v_i, \quad y_i = \begin{cases} 1 & if \quad transaction_i\, fraudulent \\ 0 & if \quad transaction_i\, non-fraudulent \end{cases} \tag{2}$$

3 Conceptual Framework (133802)

In order to create a fraud detection algorithm, the development process is divided into six parts; loading the credit card fraud dataset, exploratory data analysis (EDA) (see 4.1), preprocessing (see 4.2), model development (see 4.2.1 to 4.5) and testing (see 4.6) and evaluation (see 5). The described process can be observed visually in Figure 1.

The work begins by loading the credit card fraud dataset which is provided by the examiners of the CBS machine learning class. This data has already been processed beforehand (Blackbox) as recorded in paragraph 4.1. In addition to obtaining and loading the dataset, the data is analysed in an exploratory manner. EDA is relevant to attain a deeper understanding of the problem at hand (compare paragraph 4.1) and is the basis for future steps. According to the newly gained insights, step three, the preprocessing of the data, is performed using different over-/ and undersampling techniques as described in section 4.2.1.

By knowing the data in detail and shaping it according to the purposes of fraud detection, the fundamental basis for the classification algorithms is built. Considering the insights from chapter 2, logistic regression, a

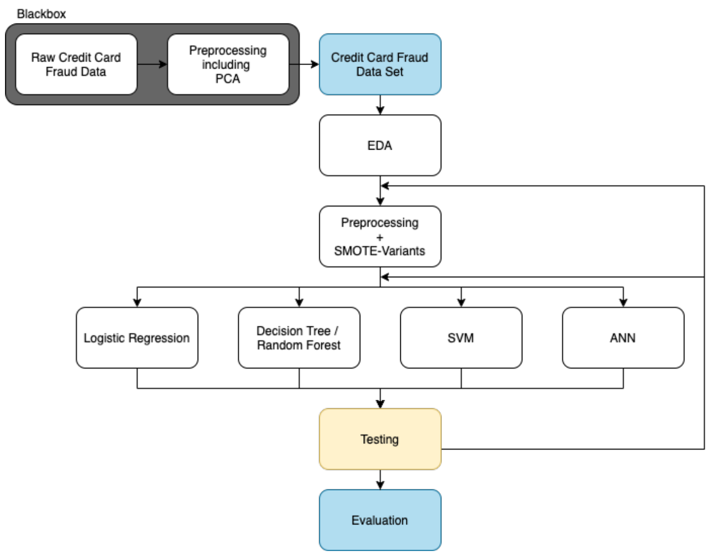
Blackbox
Raw Credit Card Fraud Data
Preprocessing including PCA
Credit Card Fraud Data Set
EDA
Preprocessing + SMOTE-Variants
Logistic Regression
Decision Tree / Random Forest
SVM
ANN
Testing
Evaluation

Figure 1: The steps performed to develop a fraud detection algorithm using multiple classification models.

decision tree based approach, a support vector machine as well as an artificial neural network are used to solve the classification problem. The models' output is subsequently tested on the development set, to achieve better classifiers by tuning the hyper-parameters. The described testing step is iterative in its nature, and therefore indicated with a loop reaching back to the preprocessing step (see Figure 1). Since not only the models' hyperparameters, but also preprocessing stage can be modified, the backwards loop in the visualization arches both above and underneath the preprocessing step (see Figure 1).

Once the models are sufficiently improved based on hyperparameter optimization, in chapter 5 the models' performance is evaluated and compared on unseen test data. Chapter 6, elaborates on encountered limitations and possible future work.

4 Methodology

4.1 Data Exploration (133348)

Dataset The dataset used in this paper is available on kaggle.com. It contains data of credit card transactions by European cardholders in September 2013 (ULB 2016). The data set originates from a research collaboration between Worldline and the Machine Learning Group of the Université Libre de Bruxelles. In order to ensure confidentiality of the personal data and to reduce the dimensionality of the data, a principal component analysis (PCA) was applied. No further information of the original features are provided ULB (2016).

EDA Sahoo et al. defined EDA as "an approach to summarize the data by taking their main characteristics and visualize it with proper representations" (2019, p.4727). As humans show high cognitive abilities to acquire information through data visualization, they serve as a powerful tool. When executed well, it is possible to observe and recognize patterns even within huge amounts of data immediately (Ware 2019).

The data set at hand contains 284,807 credit card transactions. There are 31 variables in total. A time variable, 28 numerical columns resulting from PCA, the amount transferred and the label of the transaction with 0 for a regular transaction and 1 for a fraudulent transaction.

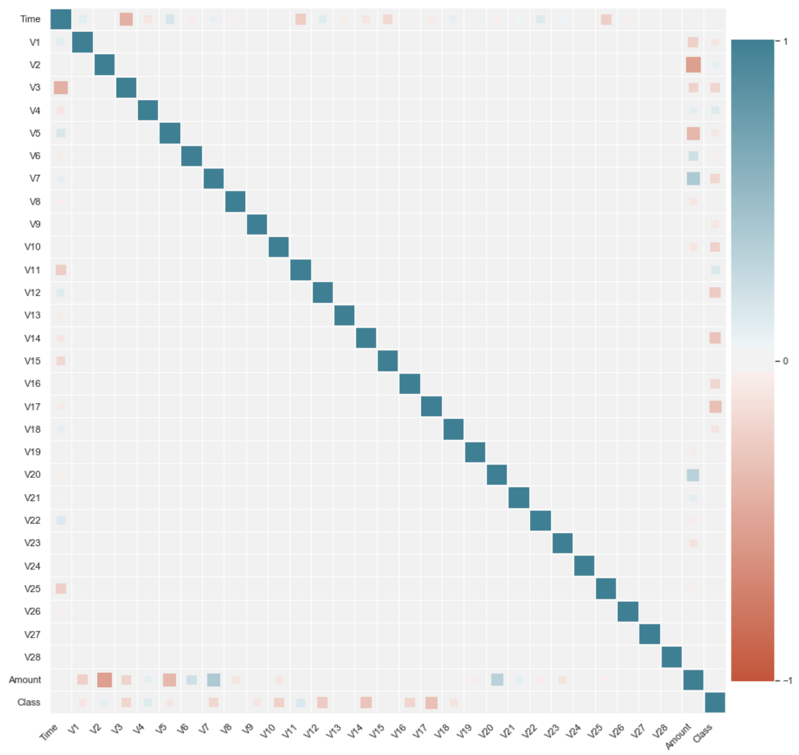

Figure 2: Correlation plot of all features in the data set.

The correlation plot (Figure 2) shows that the variables V1 to V28 do not correlate with each other. This can be explained by PCA which has been applied before, as the method reduces the dimensionality of the dataset by calculating new, largely uncorrelated features that explain the maximum variance within the data (James et al. 2013). Additionally, in Figure 2 linear correlation of some features with respect to the label feature can be observed. Since they jointly flow into the machine learning models, the correlation will not be investigated on an individual level.

The time variable counts up in the unit of second, starting at zero with the first transaction. It shows that transactions were recorded over a time period of 48 hours. The amount of transactions plotted over time (see Figure 3) show a regularity with two noticeable drops and leads to the assumption that a high number of transactions takes place during the day.

The average amount of non-fraudulent transactions is 88.29 Euro (the currency is assumed since the data set contains only European credit card holders). Since the median, which is less affected by outliers than the mean, differs from the mean (88.29 to 22.00 Euro), there must be outliers causing the difference. The maximum transaction with 25,691.16 Euro differs greatly from the average amount. Also, the distribution plot (see Figure 4 in the Appendix) confirms that the majority of the transactions are of low volume. 1798 non-fraudulent and 27 fraudulent transactions have a zero amount. The 27 fraudulent transactions are dropped (further explained in section 4.2). Looking only at the fraudulent transactions, the maximum amount is 2125.87 Euro, which is much lower than the maximum amount of regular transactions. The average amount per fraudulent transaction is 129.31 Euro and therefore higher than the average non-fraudulent transaction. Here, the difference between the mean and median (129.31 to 17.06 Euro) is even greater. Looking at the distribution (see Figure 5 in the Appendix), a high volume of fraudulent transactions in the positive range up to 20 Euro can be noticed, which could be explained by the limit of contact-less pay and another spike with the amounts close to 100 Euro, which is likely an amount that could be considered as suspicious by manual fraud detection approaches.

After dropping the zero amounts of the fraudulent transactions, from the total 284,315 transactions only 465 (0.16 per cent) are fraudulent, resulting in a highly imbalanced dataset. With regards to the total amount, non-fraudulent transactions sum up to 25,102,462.04 Euro compared to 60,127.96 Euro in fraudulent transactions. The implications of this observed imbalance and how to take it properly into account before applying algorithms to the data will be discussed in the following section.

4.2 Data Preprocessing (133816)

The preprocessing includes the following steps. First, the amount column is smoothed by applying a log transformation which moves outliers closer to the other values. Second, the time column is dropped as it can only interpreted after transforming it to a categorical feature such as day of the week or hour of the day. While this is generally possible, the potential increase in model performance is not considered, to outweigh the higher computational costs leading to the feature's exclusion. Third, fraudulent transactions with an amount of zero are excluded from the data set. As they do not cause any economic harm, not classifying them as fraudulent does not have any economic relevance for the company.
In the next step, the data set is split into a train and test set, where 75 per cent of rows are assigned to the train set. The split is performed in a stratified way, ensuring that the proportion of fraudulent samples is the same for both sets. Afterwards, the features are brought to the same scale by fitting a standard scaler on the train set and applying it to both the train and test set.
As a final step, the aforementioned class imbalance is addressed using different over-sampling methods. Since those methods significantly impact the classifiers' performance, they are introduced in detail below.

4.2.1 Over- and Under-Sampling (133816)

Handling imbalanced data is focused on increasing the predictability of the minority class in an imbalanced dataset. Solutions for that emerge both on a data and algorithmic level (Han et al. 2005). Having the former in mind, this section considers re-sampling methods that can be applied to the data in order to positively affect the performance of algorithms that are subsequently applied.

Random Under-/over-sampling The most basic methods in that regard are random under- and over-sampling. Random under-sampling works by randomly removing samples from the majority class. Random over-sampling, in turn, expands the minority class by creating exact duplicates of minority class samples. Both methods can be applied either separately or combined with each other and have shown to improve the performance of classifiers in terms of predicting minority class samples (Chawla et al. 2002). However, there are more sophisticated methods producing superior results which are presented in this section.

SMOTE One popular method of over-sampling the minority class is the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al. 2002). This method creates new data points in the minority class by detecting the k minority class nearest neighbors of a given minority example and creating a synthetic example (x_{new}) on a random point on the line connecting the original example (x) with one or more of its neighbors ($\tilde{x}$).

$$x_{new} = x + rand(0,1) * (\tilde{x} - x) \tag{3}$$

By doing so, SMOTE expands the decision region of the minority class, allowing for an improved generalizability of the trained classifiers. Chawla et al. (2002) further suggest a combination of over- and under-sampling. While the SMOTE algorithm provides a simple and effective way of expanding the minority class decision space by generating synthetic samples without overfitting the data, it risks shrinking the decision space of majority class samples which can lead to poor predictions (Hu & Li 2013).

Borderline-SMOTE This problem is avoided by Borderline-SMOTE. As its name suggests, this algorithm is mainly concerned with samples that lie at the borderline between the majority and minority class (Han et al. 2005). For many classification algorithms, samples that are positioned at this borderline have a much bigger impact on the classification than those that are further out. This is the point of entry of the Borderline-SMOTE which solely oversamples borderline samples of the minority class contrasting it from the "normal" SMOTE that creates synthetic samples based on all minority class samples. From a procedural perspective, those minority class samples, whose k nearest neighbors include mostly – but not solely - majority class neighbors, are used for the creation of new synthetic samples. New samples are created by applying the aforementioned SMOTE-procedure to the selected samples. When a minority class' sample's neighbors all belong to the majority class, it is considered noise and is excluded from further steps. Through this procedure, Borderline-SMOTE enlarges the decision space of the minority class without compromising the decision space of the majority class. This method, however, comes with the downside that in a particularly small minority class, most of the samples would be considered noise, which would prevent it from creating the synthetic samples needed, to increase predictability of minority class samples.

ADASYN The third method considered for minority class over-sampling is adaptive synthetic (ADASYN) sampling (He et al. 2008). Depending on the difficulty of learning a minority class sample - which is indicated by the ratio of majority class samples by which it is surrounded - a larger number of synthetic data points is created. In doing so, the learning bias is reduced in favour of minority class samples and the decision boundary is shifted, which increases the focus on samples that are difficult to learn (He et al. 2008). The ADASYN algorithm creates synthetic data points in a similar way as the SMOTE algorithm. However, it takes the surroundings of every minority class sample into account and generates more synthetic data points for those samples in areas where the density of majority class samples is high. Particularly in extreme cases, where a minority class sample's k nearest neighbors solely consist of majority class samples, ADASYN deviates substantially from Borderline-Smote. While the first puts a special emphasis on such samples, the latter disregards them entirely.

Even though the step of creating a synthetic data point is similar for both SMOTE, Borderline-SMOTE, and ADASYN, the general ideas behind them differ from each other. SMOTE simply creates synthetic samples without regard to the surroundings of minority class samples. Borderline-SMOTE and ADASYN look at their surroundings, yet they emphasize different things. Borderline-SMOTE is mostly concerned about samples at the borderline between minority and majority class whereas ADASYN supports difficult to learn samples. Due to their different approaches, all three algorithms are separately applied to the analysis to see which one most positively affects the model performance. As a combination of under- and over-sampling is argued to perform better then pure over-sampling (Chawla et al. 2002), this analysis combines the over-sampling methods with random under-sampling. As a first step, the minority class is over-sampled to 50 per cent of the size of the majority clas. Then, the majority class is under-sampled to half its size. As a result, both classes have an equal size which is exactly half the size of the initial majority class. Those classes are then used for training the classifiers in the upcoming sections.

4.3 Decision tree (116853)

A decision tree (DT) is a tree-based algorithm which splits up data based on a set of decision rules (Geron 2019). In machine learning, decision tree learning is defined as a "non-parametric supervised approach used for classification and regression" (Anis et al. 2015, p. 90). Non-parametric means that the algorithm learns the functional structure from the dataset without making former assumptions about the distribution or it's errors which, in turn, increases its degrees of freedom. This allows a DT algorithm to fit closely on complex datasets, but also makes it prone to overfitting on the training set (Geron 2019).
A DT consists of nodes and branches. Starting from a top root node (the entire population/sample), a tree-like structure is constructed downwards and from left to right by splitting the data based on decisions occurring at the internal nodes. A node corresponds to a certain characteristic in the data and the branches to a range of possible values (Ali et al. 2012). During the construction of a DT, the recursive creation of sub-nodes happens based on impurity measures, such as *Gini impurity* or *Entropy*. In this paper, *Gini impurity* is explained which is the default impurity measure used in the implementation of the Python module *sklearn.tree.DecisionTreeClassifier* (DTC). The *Gini impurity* is shown in equation **4**; $p_{i,k}$ is the the ratio of class k instances in the i^{th} node.

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{4}$$

The goal of a DT is to reduce the Gini impurity for each node and consequently increase the overall homogeneity of its sub-nodes in order to find an optimal split for the dataset with respect to the target variable (Venkata Suryanarayana et al. 2018). A node is considered "pure" (with corresponding Gini impurity of 0) if it only contains training instances from the same class (Geron 2019). Once a chain of decisions in the tree ends, it reaches terminal node and no further branching occurs. These terminal nodes represent the target variables of the dataset (Husejinović 2020).

One implementation for training a DT is the *Classification and Regression Tree* (CART) algorithm, which was developed by (Breiman et al. 1984), and is implemented in DTC. CART constructs binary trees, meaning it only allows for two sub-nodes per node during each decision. Once a DT is constructed, it can be further optimized by tuning its hyperparameters. The hyperparameters allow to set constraints for the model (regularization) which minimize the risk of overfitting. It is possible to change the maximum depth and width of a DT by setting the amount of leaves or introducing a minimum amount of splits (Geron 2019).

4.3.1 Random forest (116853)

Random forest describes a method which constructs an ensemble of DTs and aggregates their prediction results by having them vote for the most popular class (majority vote) (Anis et al. 2015). This method was first developed and mentioned by Breiman (2001). For classification tasks, Breiman (2001) showed that the accuracy could be significantly improved by using random forest models compared to single decision trees. Furthermore, random forest models are more robust to overfitting on the training data if a large enough ensemble of trees is grown Bhattacharyya et al. (2011). The amount of trees grown is handled by the hyperparameter *n_ estimators*. Additionally, random forest models can be similarly regularized as DTs (Geron 2019).

4.4 Logistic Regression (133348)

Logistic Regression is among the most widely-used classification algorithms. It enables to calculate a probability for y to assign it to a category. Since there are only two possible classes for the credit card fraud data - fraudulent and non-fraudulent - a binary logistic regression is suitable (James et al. 2013). Multinomial logistic regression (with more than two classes) is not covered in this paper. In the following, the equations are stated in vectorized form.

Similarly to linear regression, the weighted sum is calculated from the input features and a bias term ($h_w(X^T, c) = X^T w + c$) (*1.1. Linear Models* n.d.). In contrast to linear regression, additionally the sigmoid function ($\sigma(t) = \frac{1}{1+e^{-t}}$) is applied to transform the input into a probability by normalizing the output into a value between 0 and 1 (Geron 2019). Due to its S-shape the method can create more sensible predictions and is able to capture the span of probabilities in a better way (James et al. 2013). The logistic function looks as the following in vectorized form:

$$\hat{p} = h_w(X^T, c)) = \sigma(X^T w + c) = \frac{1}{1 + e^{-(X^T w + c)}} \tag{5}$$

Since we have a binary classifier, an output of equation 5 equal to, or above 0.5 gets labelled as 1 (fraudulent), whereas an output below 0.5 is labelled as 0 (non-fraudulent).

$$\hat{y} = \begin{cases} 0, & \text{if } \hat{p} < 0.5 \\ 0, & \text{if } \hat{p} \geq 0.5. \end{cases} \tag{6}$$

To assign a probability as low as possible to negative cases (non-fraudulent) and the highest possible probability to positive cases (fraudulent) a cost function is minimized (Geron 2019). Logistic regression was applied on the data set with the Python package *sklearn*. Instead of the default solver *lbfgs*, *saga* is used, since it is fast on large datasets and, based on the scikit-learn documentation, often a good choice. It is a modification of the *sag* solver, which uses a Stochastic Average Gradient Descent solving algorithm. However, besides *l1* and *l2*, it additionally enables *elasticnet* as penalty option. *Elasticnet* is a combined penalty of *l1* and *l2*, for which ratio is determined by ρ, that can be controlled with the *l1_ratio* parameter. The penalty term is added to the cost function J. Additionally, different C values on a logarithmic scale are tested. C determines the strength of the regularization by being multiplied with the cost function (*1.1. Linear Models* n.d.). A low C value is expected to produce the best results for the data at hand, since it results in a stronger regularization and therefore prevents overfitting of the model (*sklearn.linear_model.LogisticRegression* n.d.).

Penalty	**Loss Function**
l1	$\|w\|_1 + C \cdot J(y, \hat{y})$
l2	$\frac{1}{2} w^T w + C \cdot J(y, \hat{y})$
elasticnet	$\frac{1-\rho}{2} w^T w + \rho \|w\|_1 + C \cdot J(y, \hat{y})$

Table 3: Regularization types

4.5 Support-Vector-Machine (133816)

Support-Vector-Machines (SVMs) are relevant machine learning models in the domain of classifying complex data sets of limited size (Geron 2019). Both linear and non-linear classification tasks can be handled by SVMs, yet only the second case is considered in this paper, as the complexity of the underlying dataset does not allow for a straight line to make a clear distinction between the two classes.

Regarding the types of classification SVMs can implement, one can differentiate between *soft margin* and *hard margin* classification. The latter is only applicable to linearly separable data sets and highly sensitive to outliers, which is why it is not suitable for this analysis. Soft margin classification, in turn, strives to find a balance between keeping the gap between the different classes as big as possible, while simultaneously limiting the number of instances, that end up in between classes or in the wrong class (Geron 2019). This trade-off is mediated by the C hyperparameter, which is subject to the parameter optimization process.

Data sets that are not linearly separable can be handled in different ways. One of them is the addition of more features, which can - under certain circumstances - make the data set linearly separable. For example, a second feature x_2 can be added to x_1 where $x_2 = (x_1)^2$ thereby putting the newly generated feature into two-dimensional space where it can potentially be linearly separable from other features that went through the same transformation. This approach, however, does not scale well to larger data sets as a high polynomial degree negatively influences the model's speed. This problem is circumvented by applying the kernel trick to the SVM. By utilizing a polynomial kernel, the same result is achieved through the inclusion of additional polynomial features without actually adding them (Geron 2019).

Another way of performing a non-linear classification is by adding similarity features. Those features are determined by applying a similarity function - such as Gaussian Radial Basis Function (RBF) (see equation **7**) - which estimates new features based on the original feature's proximity to a specific landmark (Geron 2019).

$$\phi\gamma(x, l) = exp(-\gamma \|x - l\|^2) \tag{7}$$

As a result, the newly generated features form part of a higher dimensional space in which they could be linearly separable. Also, in case of the similarity features method the kernel trick can be used to decrease computational costs without sacrificing the quality of the result. Relevant parameters for this Gaussian RBF kernel include γ and C, where γ constitutes a regularization parameter by regulating the influence of each instance (Geron 2019).

4.6 Artificial Neural Network (133802)

As a fourth model to detect credit card fraud, an artificial neural network (ANN) is used. The network is built from scratch in Python by using the *numpy* package as the only development framework. The implementation of the ANN is based on the mathematical concepts taught by Andrew Ng in his deep learning online course (Ng 2020). In the following, the specifications of each of the network's main parts are described.

Initialization The model's parameters, which are weight and bias matrices, are created dynamically, depending on the networks architecture. The weight matrices are initialized according to the proposal of He et al. (n.d.), where the randomly chosen weight in position ij and layer l, ξ_l^{ij}, is scaled by the size of the previous layer n_{l-1} (compare equation 8). This initialization helps the network, which uses the Rectified Linear Unit (ReLU) activation function, to converge faster by reducing the risk of vanishing or exploding weights, especially in large networks.

$$w_l^{ij} = \xi^{ij} * \sqrt{\frac{2}{n_{l-1}}}, \quad \xi^{ij} \in [0, 1] \tag{8}$$

Forward and Backward Propagation The forward propagation of a layer in a neural network can be divided in a linear and a non-linear transformation applied on the output values of the previous layer. For calculation optimization purposes, all linear transformations are implemented in a vectorized manner. As non-linearities, the ReLU activation function is used in all layers except the final layer. Because the sigmoid function comes with the useful property of scaling the output between 0 and 1, it is used to generate the network's prediction probabilities. The commonly known mathematical functions, which are used for forward propagation, can be found in the code attached to this work.

Cost Function To measure the error between the network's predictions $\hat{Y}$ and the actual observations Y, the cross-entropy cost function is used. The intention of the cost function is to weigh the difference between prediction and actual values exponentially which lets the ANN focus on the optimization of the most wrong neurons first.

Back Propagation and Weight Updates The commonly known gradient descent algorithm is implemented for updating the weights. As the data is less than 200MB in size, batch gradient descent is not necessary to be implemented. The implementation of the ANN is rather primitive which can lead to numerical instability. For this reason a mechanism is implemented which reduces the learning rate by a factor of ten, once a predefined checkpoint is reached. This mechanism helps the network to converge closer to the minimum loss without using batch normalization or other regulatory means. For the sake of simplicity, we did not implement momentum or ADAM gradient descent, although these help make the network converge faster.

4.7 Model Selection (133816)

The models are trained using the pipeline object from the *imblearn* package, where both, the over- and under-sampling as well as the respective model are initialized. This pipeline is chosen over the comparable *sklearn* implementation as it is capable of only re-sampling the train data at each iteration of the model fitting process. This stands in contrast to the *sklearn* package which inadequately fits both train and test data. Three-fold cross validation is used to ensure the models' generalizability by avoiding overfitting. Furthermore, *GridSearchCV* from *sklearn* is applied over a pre-specified parameter grid in order to detect on which parameter configuration each model performs best based on the lowest roc_auc value. In total, each model is trained thrice - once for every over-sampling technique. The results can be seen in Table 5 in the Appendix.

5 Results (All)

To select the best model out of the pre-selected models two criteria will be taken into account: one technical evaluation and one economical evaluation.
The technical evaluation is based on the *MCC* score. Since a score of zero is equal to a random guess and the values of all models lay above zero, all models are able to contribute to the containment of criminal action to a varying degree. The SVM model applied to a SMOTE-transformed dataset is particularly performative achieving an MCC score of 0.67. Clearly performing better than random, *proposition 1* can be supported.
The models' economic performance is measured based on the previously introduced EVS score. Also in this regard, the SVM in combination with SMOTE performs best with a score of 7,033.53. Since the training set represents only 25 per cent of the data flowing into this equation, the EVS score represents half a day. The SVM could save more than 5 million euros over the period of one year confirming the model's economic viability and therefore *proposition 2*. The performance measures obtained by the application of the different models on the test set can be observed in Table 4.

Model		**Metrics**	
Preprocessing	**Algorithm**	**MCC**	**EVS**
SMOTE	Random Forest	0.48	-14,057.22
SMOTE	Logistic Regression	0.24	- 127,033.65
SMOTE	SVM	0.67	7,033.53
SMOTE	ANN$_{iter=4200}$	0.25	-108,256.40
Borderline	Random Forest	0.66	3,424.20
Borderline	Logistic Regression	0.29	-69,844
Borderline	SVM	0.27	-80,173.93
Borderline	ANN$_{iter=800}$	0.27	-81,851.44
ADASYN	Random Forest	0.19	-236,172.59
ADASYN	Logistic Regression	0.13	-533,139.13
ADASYN	SVM	0.35	-38,895.23
ADASYN	ANN$_{iter=2200}$	0.25	-325,109.14

Table 4: Each sampling variant is shown with the best performing models of every model category. The models' parameters are stated in Table 5

6 Discussion and Conclusion (All)

Contributions The main contribution of the paper is to (1) determine whether supervised machine learning algorithms can contain criminal action by successfully identifying fraudulent transactions. Furthermore, the economic viability score (EVS) based on weighing the *True Positives* (TP) and *False Positives* (FP) is introduced (2) as a contextual measure, connecting the technical aspect with the economic application. From the results, it can be seen that applying different sampling methods results in vastly different classification performances. In comparison to related work, we include the MCC as a performance measure. The different algorithms were tested based on a variety of hyperparameter values. Also, the different methods of over-sampling, each combined with under-sampling, contribute positively to the individual classifier performance. Especially Borderline-SMOTE seems to outperform SMOTE and ADASYN as over-sampling method. The process of testing different sampling strategies, applying hyperparameter-tuning and comparing different algorithms based defined evaluation measures, led to a diverse outcome of the different algorithms. The supervised machine learning algorithms mostly did not reach a positive EVS. Especially logistic regression and the ANN performed insufficiently. However, the Random Forest and SVM models produced good results, which classifies enough fraudulent transactions as fraudulent in order to yield a positive EVS. However, with a positive MCC score for all models, it can be concluded, that the ensemble of supervised machine learning algorithms contributes positively to the containment of criminal activity concerning credit card fraud.

Limitations As the underlying dataset of this study has already been PCA transformed, the interpretability regarding the decision-making of the classifiers can not be provided. In fact, it is entirely unknown which features were even included in the initial dataset. Hence, this study is not informative about possible feature selection practices or direct feature effects on the outcome. A further limitation is the short time span for which data was collected. It is reasonable to assume that not all fraudulent schemes can be captured in such a short period of time which can result in future fraudulent actions being missed. From a model training perspective, another shortcoming is that the models utilized in this study only went through one iteration of grid search, potentially missing a more favourable parameter selection in the process. One reason for that are limited computational capacities that negatively influenced the time it took the models to train. Apart from that, as the focus of re-sampling was on over-sampling of the minority class, there was a shortcoming in testing different under-sampling methods besides the random under-sampling. The vast and steadily increasing body of literature on different over- and under-sampling procedures offers many opportunities to expand further on these methods. Finally, while this paper has a strong focus on handling imbalanced data through manipulations on the data level, potentials on the algorithmic levels are barely explored. Instead, as algorithms are only applied in a conventional manner, this paper does not show how algorithms by themselves can be applied to handle imbalanced data. During the work the high computational cost, especially of the SVM model, was noticed. However, this was not further dealt with in this paper.

Future Research Building on top of this paper's contributions and circumventing its limitations, future research can further advance knowledge on effective fraud detection procedures. On the algorithmic level, voting classifiers can potentially improve the model's performance by combining predictors from different models. The good performance of the Random Forest model and SVM in this study provides an indication for the suitability of these models. Another path that can be conducive to model performance is investigating in how far different majority-/minority class proportions influence the result by testing different class configurations. Future studies can also build on top of the performance measures introduced in this study, mainly the measure for economic viability. By introducing an economic dimension into usually non-contextual performance measures, models can be tailored to their circumstances and thereby potentially deliver more relevant results. In contrast to the *roc_ auc* optimization applied in this paper, future research can consider to optimize their models on the EVS score or a comparable measure to maximize economic utility of the model.

Conclusion In the end, dealing with imbalanced data usually results in a trade-off where reliable crime prevention and economic value creation are barely achieved simultaneously. If crime prevention is the focus, an optimization based on recall or MCC makes sense in order to prevent fraudulent transactions. For an economically viable model, however, detection of actual frauds must not come at the expense of precision. Since every legitimate transaction falsely classified, and is therefore prevented, costs the bank money. This weighing can be effectively captured by the EVS. Therefore, whether or not a model is suitable ultimately depends on the project's requirements.

References

1.1. Linear Models (n.d.).
URL: *https://scikit-learn.org/stable/modules/linear$_m$odel.htmllogistic − regression*

Abdallah, A., Maarof, M. A. & Zainal, A. (2016), 'Fraud detection system: A survey', *Journal of Network and Computer Applications* **68**, 90–113.

Albashrawi, M. & Lowell, M. (2016), 'Detecting Financial Fraud Using Data Mining Techniques : a', *Journal of Data Science* **14**(3), 553–570.

Ali, J., Khan, R., Ahmad, N. & Maqsood, I. (2012), 'Random Forests and Decision Trees', *International Journal of Computer Science Issues* **9**(5), 272–278.

Anis, M., Ali, M. & Yadav, A. (2015), 'A comparative study of decision tree algorithms for class imbalanced learning in credit card fraud detection', *International journal of economics, commerce and management* **3**(12), 86–102.

Bhattacharyya, S., Jha, S., Tharakunnel, K. & Westland, J. C. (2011), 'Data mining for credit card fraud: A comparative study', *Decision Support Systems* **50**(3), 602–613.

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**(1), 5–32.

Breiman, L., Friedman, J., Stone, C. J. & Olshen, R. A. (1984), *Classification and regression trees*, CRC press.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), 'SMOTE: Synthetic Minority Over-sampling Technique', *Journal of Artificial Intelligence Research* **16**(1), 321–357.

Dal Pozzolo, A., Caelen, O., Le Borgne, Y. A., Waterschoot, S. & Bontempi, G. (2014), 'Learned lessons in credit card fraud detection from a practitioner perspective', *Expert Systems with Applications* **41**(10), 4915–4928.

Delamaire, L., Abdou, H. & Pointon, J. (2009), 'Credit card fraud and detection techniques: A review', *Banks and Bank Systems* **4**(2), 57–68.

Dornadula, V. N. & Geetha, S. (2019), 'Credit Card Fraud Detection using Machine Learning Algorithms', *Procedia Computer Science* **165**, 631–641.

FIS (2020), Global payment report – The pathways of people and payments, Technical Report January.

Geron, A. (2019), *Hands–On Machine Learning with Scikit–Learn and TensorFlow 2nd edition.*

Han, H., Wang, W.-Y. & Mao, B.-H. (2005), Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning, *in* 'Lecture Notes in Computer Science', Vol. 3644, pp. 878–887.

Hand, D. J., Whitrow, C., Adams, N. M., Juszczak, P. & Weston, D. (2008), 'Performance criteria for plastic card fraud detection tools', *Journal of the Operational Research Society* **59**(7), 956–962.

He, H., Bai, Y., Garcia, E. A. & Li, S. (2008), ‘ADASYN: Adaptive synthetic sampling approach for imbalanced learning’, *Proceedings of the International Joint Conference on Neural Networks* (3), 1322–1328.

He, K., Zhang, X., Ren, S. & Sun, J. (n.d.), Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, Technical report.

Hu, F. & Li, H. (2013), ‘A Novel Boundary Oversampling Algorithm Based on Neighborhood Rough Set Model: NRSBoundary-SMOTE’, *Mathematical Problems in Engineering* **2013**, 1–10.

Husejinović, A. (2020), ‘Credit card fraud detection using naive Bayesian and c4.5 decision tree classifiers’, *Periodicals of Engineering and Natural Sciences* **8**(1), 1–5.

Jain, L. C., Peng, S.-L., Alhadidi, B. & Pal, S. (2020), *Intelligent Computing Paradigm and Cutting-edge Technologies*, Vol. 9 of *Learning and Analytics in Intelligent Systems*, Springer International Publishing, Cham.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013), *An Introduction to Statistical Learning*, Vol. 103 of *Springer Texts in Statistics*, Springer New York, New York, NY.

Ng, Andrew, D. (2020), ‘Neural Networks and Deep Learning’.
URL: *https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning*

Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y. & Sun, X. (2010), ‘The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature’.

Niu, X., Wang, L. & Yang, X. (2019), ‘A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised’.

Powers, D. M. W. (2011), ‘EVALUATION: FROM PRECISION, RECALL AND F-MEASURE TO ROC, INFORMEDNESS, MARKEDNESS & CORRELATION’, *Journal of Machine Learning Technology* **2**(1), 37–63.

Quah, J. T. & Sriganesh, M. (2008), ‘Real-time credit card fraud detection using computational intelligence’, *Expert Systems with Applications* **35**(4), 1721–1732.

Randhawa, K., Loo, C. K., Seera, M., Lim, C. P. & Nandi, A. K. (2018), ‘Credit Card Fraud Detection Using AdaBoost and Majority Voting’, *IEEE Access* **6**, 14277–14284.

Sahoo, K., Samal, A. K., Pramanik, J. & Pani, S. K. (2019), ‘Exploratory Data Analysis using Python’, *International Journal of Innovative Technology and Exploring Engineering* **8**(12), 4727–4735.

sklearn.linear_ model.LogisticRegression (n.d.).
URL: *https://scikit-learn.org/stable/modules/generated/sklearn.linear$_m$odel.LogisticRegression.html*

Sokolova, M. & Lapalme, G. (2009), 'A systematic analysis of performance measures for classification tasks', *Information Processing and Management* **45**(4), 427–437.
URL: *http://dx.doi.org/10.1016/j.ipm.2009.03.002*

The Ohio State University (2006), Reporting and Investigating Financial Fraud University Policy, Technical report.
URL: *https://busfin.osu.edu/sites/default/files/119$_r$eportinginvestigatingfinancialfraud.pdf*

ULB, M. L. G. (2016), 'Credit Card Fraud Detection'.
URL: *https://www.kaggle.com/mlg-ulb/creditcardfraud*

Venkata Suryanarayana, S., Balaji, G. N. & Venkateswara Rao, G. (2018), 'Machine learning approaches for credit card fraud detection', *International Journal of Engineering and Technology(UAE)* **7**(2), 917–920.

Ware, C. (2019), 'Exploratory Data Analysis using Python', *International Journal of Innovative Technology and Exploring Engineering* **8**(12), 4727–4735.

West, J. & Bhattacharya, M. (2016), 'Intelligent financial fraud detection: A comprehensive review', *Computers and Security* **57**, 47–66.

Whitrow, C., Hand, D. J., Juszczak, P., Weston, D. & Adams, N. M. (2009), 'Transaction aggregation as a strategy for credit card fraud detection', *Data Mining and Knowledge Discovery* **18**(1), 30–55.

7 Appendix

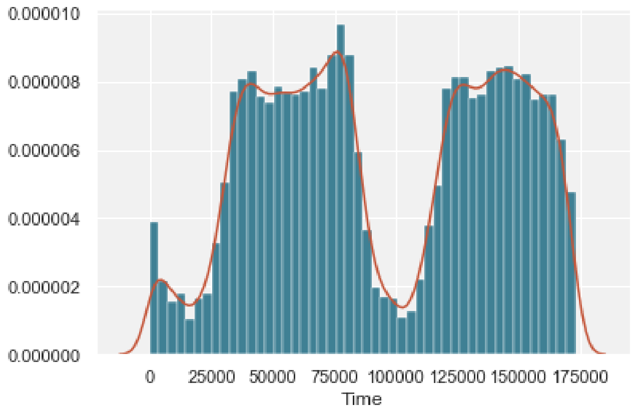

Figure 3: Distribution of the number of transactions over time.

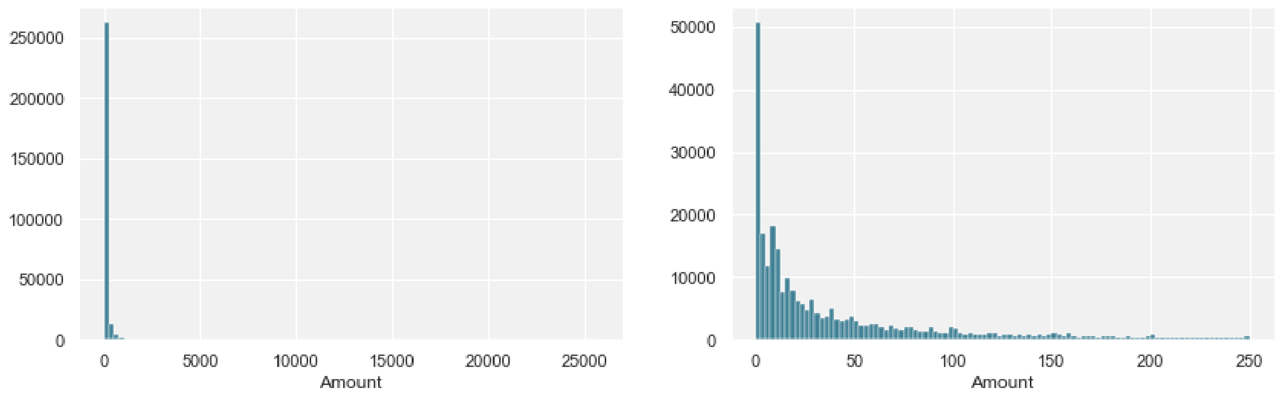

Figure 4: Distribution of the number of non-fraudulent transactions based on the amount. Showing on the left hand side all transaction amounts and on the right hand side limited to the amount of 250 Euro.

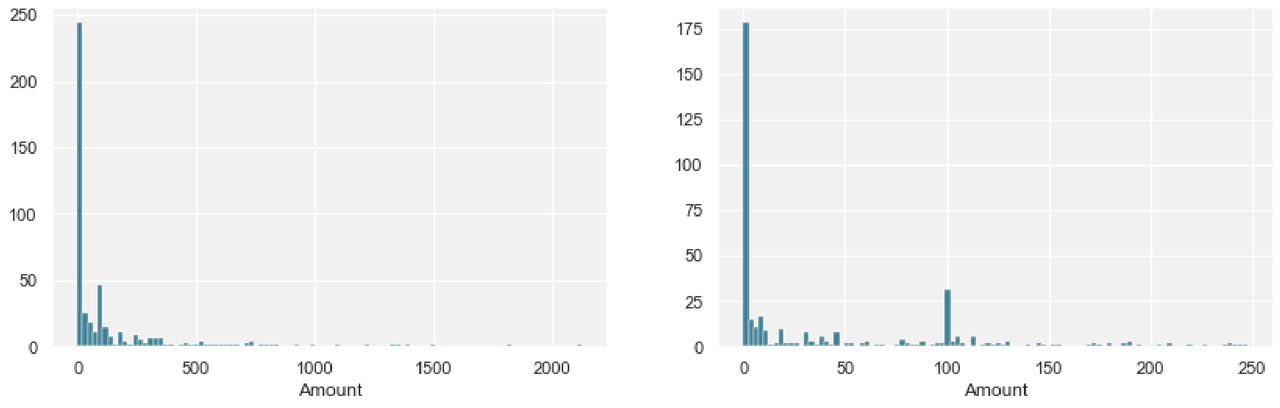

Figure 5: Distribution of the number of fraudulent transactions based on the amount. Showing on the left hand side all transaction amounts and on the right hand side limited to the amount of 250 Euro.

Model		**Parameters**				**Metric**
Algorithm	**Preprocessing**	**max_depth**		**n_est**		**ROC_AUC**
Random Forest	SMOTE	5		200		0.99
Random Forest	Borderline	10		200		0.98
Random Forest	ADASYN	5		200		0.98
Model	**Preprocessing**	**solver**	**penalty**	**C**	**l1_ratio**	**ROC_AUC**
Logistic Regression	SMOTE	saga	l1	0.001	0	0.98
Logistic Regression	Borderline	saga	l1	0.001	0	0.98
Logistic Regression	ADASYN	saga	l1	0.001	0	0.98
Model	**Preprocessing**	**kernel**	**C**	**degree**	**gamma**	**ROC_AUC**
SVM	SMOTE	poly	0.001	5	scale	-*
SVM	Borderline	poly	0.001	1	scale	-*
SVM	ADASYN	poly	0.001	2	scale	-*
Model	**Preprocessing**	**num_dims**		α	**num_iter**	**ROC_AUC**
ANN	SMOTE	[29, 15, 20, 1]		0.01	300	0.93
ANN	Borderline	[29, 15, 20, 1]		0.01	300	0.93
ANN	ADASYN	[29, 15, 20, 1]		0.01	300	0.90

Table 5: Presents the results of the model selection process. The models are stated with the parameters that led to the best results in the grid search process. More detailed information can be found in the code that is delivered with this work. * values not available due to server issues.