Creating Value Through Data Mining - Project

Predicting Online Payment Fraud by the help of Machine Learning Algorithms

Hannes Guth
Artem Demidov
Purity Gikonyo

A project presented for the subject
Creating Value Through Data Mining

Geneva School of Economics and Management
University of Geneva
Switzerland

Predicting Online Payment Fraud by the help of Machine Learning Algorithms

Hannes Guth, Artem Demidov, Purity Gikonyo

December 20, 2022

Contents

1 Introduction

Online payment fraud is an increasing problem and can happen to every person who makes payments using credit or prepaid cards. To tackle an upcoming issue like this, one needs to analyze its roots in-depth and mitigate the risk of occurring. This project will deal with this first part, namely analyzing and predicting fraudulent transactions by applying machine learning methods on a comprehensive data set, containing more than 6 million transactions. But before methods can be successfully deployed and interpreted, the term Online Payment Fraud shall be introduced and understood. It can be considered as a form of theft where criminals either obtain cash or execute transactions with a credit card that is assigned to another person. [exp] The damage that is caused by such transactions is tremendous. In 2022, issuers, merchants, and consumers suffered a loss of more than USD 41 billion. Most of these cases are reported from e-commerce platforms. North America is the region with the highest number of fraudulent cases, recording up to 30% of all the reported cases. [Cop21]
It is very important to avoid fraud before it happens. This project will focus on identifying fraudulent transactions by several characteristics. Applying techniques like this may help banks flag and treat suspicious transactions before they are executed. Special focus will be put on identifying as many fraudulent transactions as possible while giving less weight to false positive predictions, i.e. false alerts.

2 Exploratory Analysis

2.1 Data overview

As a first step, the features of the data set will be introduced.

- *step*: 1 step equals 1 hour, the number describes how long it took to execute this payment
- *type*: Refers to the type of transaction
- *amount*: This is the amount that has been transferred
- *nameOrig*: The anonymous account number from where the transaction originated
- *oldbalanceOrg*: The balance of the origin account before the transaction took place
- *newbalanceOrig*: The balance of the origin account after the transaction took place
- *nameDest*: The anonymous account number to which the transaction has been executed
- *oldbalanceDest*: The balance of the destination account before the transaction
- *newbalanceDest*: The balance of the destination account after the transaction
- *isFraud*: Dependent variable which shall be predicted, indicates if this transaction was fraud

In the following data overview, distributions of variables and some patterns were analyzed. For example, the distribution of the amount of all transactions is very smooth as can be seen in Figure 1 that shows the amount of transactions on the y-axis with respect to the amount itself on the x-axis:

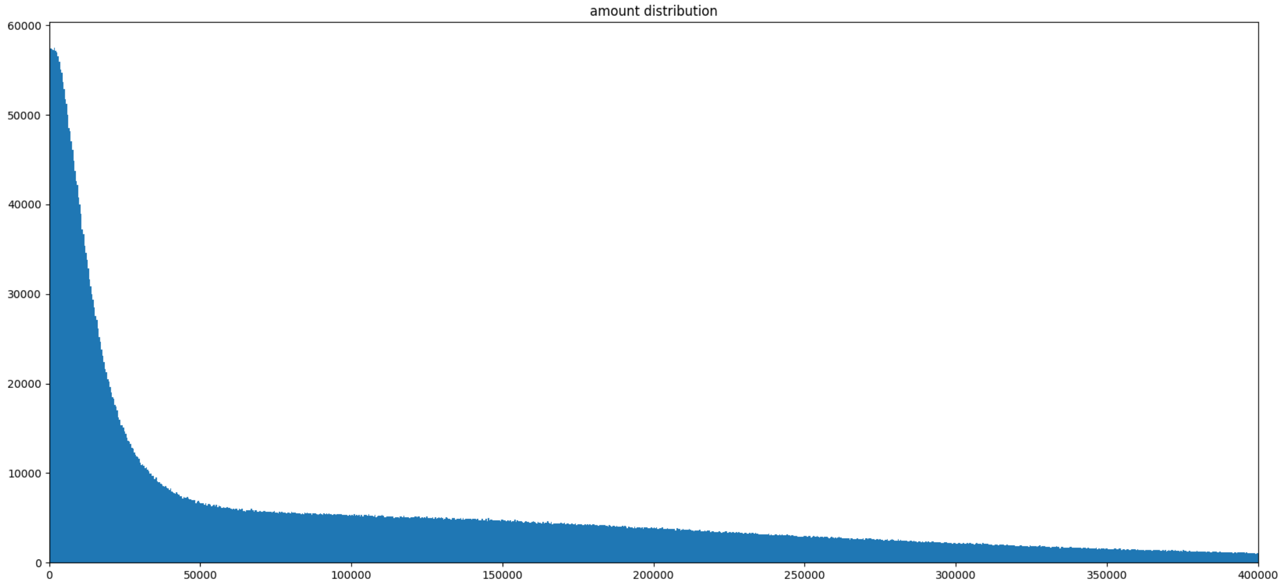


Figure 1: Distribution of amount of transactions

The maximum x value was limited to demonstrate shape of the distribution, which was slightly reduced with the same dynamic. The majority of transactions was below USD 50,000. With rising amount, significantly less transactions are executed.
The figure that follows shows the plot of amount distribution against fraud transactions:

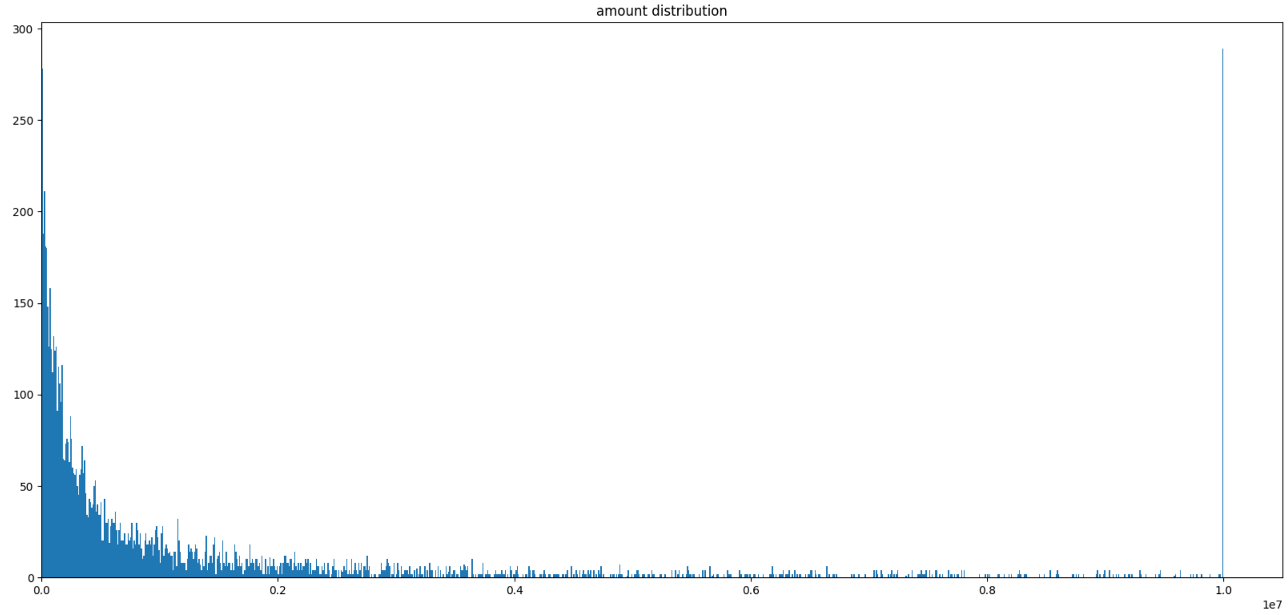


Figure 2: Distribution of amount of transactions against fraud

This plot has a pike near 10^7, which points out that a lot of fraudulent transactions had such an amount transferred. This finding emphasizes the importance of the amount feature. That is why it was very important to investigate discrepancies between transaction amounts and changing balances.

The other important finding was the distribution of fraudulent transactions depending on the type of transaction. This chart is represented below:

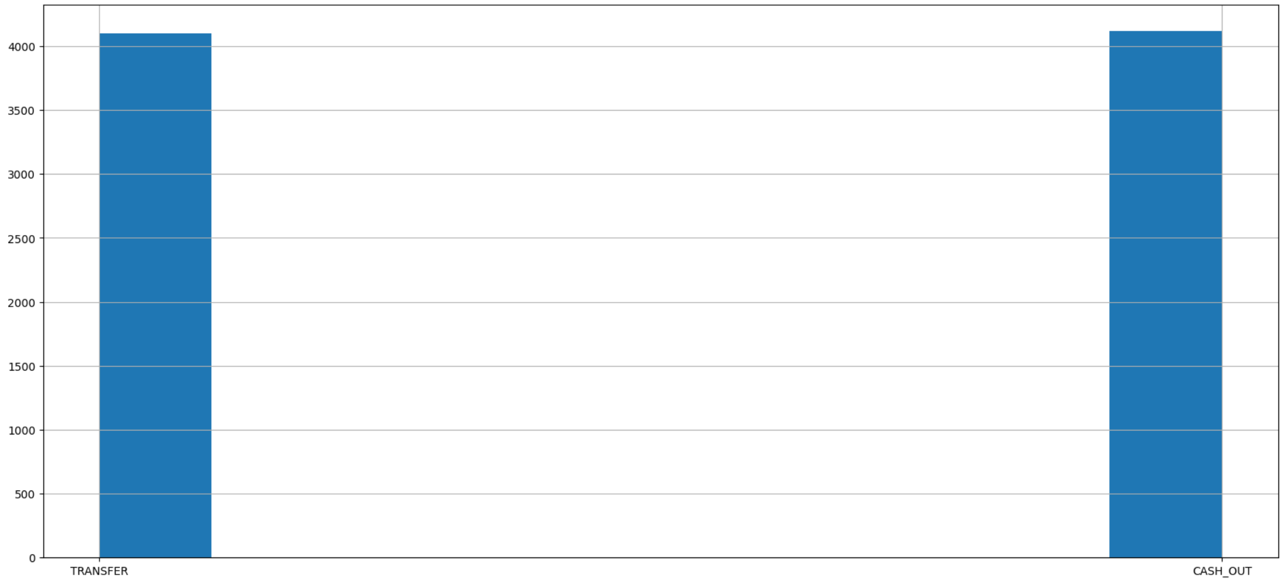


Figure 3: Fraud among different types of transactions

As we can see, fraud only occurs in transfer or cash_out types of transactions, so only these two types of transactions will be retained for the rest of the project.

Summarizing the main findings from the EDA, the following things could be learned:

- Only transactions with types TRANSFER and CASH_OUT are fraud
- Amount of transaction is often not equal to change in the balance of origin or destination bank accounts
- Many transactions have the same destination account, so they are not independent
- Sometimes the amount of the transaction exceeds the current balance, but the final balance is always not negative
- Classes are extremely unbalanced - only about 0.1% of all transactions are fraud transactions
- All names of bank accounts start with the letter M or C, which indicates that we have specific types of accounts.

2.2 Discrepancy investigation

During EDA we recognized, that about 80% of all transactions have discrepancies. In our case discrepancy is when the difference between the old and new balance of the source or destination does not equal the amount of the transaction. One of the reasons for such discrepancy is the comparison of float numbers, which sometimes can provide us with incorrect results. That is why we rounded all transactions to avoid this problem. Moreover, as mentioned above, the only types of transactions we were interested in are TRANSFER and CASH_OUT, because fraud is possible only in those cases. Therefore, only these two types of transactions were kept. After these steps, we discovered that the number of transactions with problems was significantly reduced - less than 0.7% of transactions with type TRANSFER or CASH_OUT had actual spent value ($old\ balance - new\ balance$) not equal to the amount, what can be explained by outliers. About 8% of CASH_OUT transactions had the actual amount received ($new\ balance - old\ balance$) not equal to the amount, so it is also a minor discrepancy. The other picture is for TRANSFER transactions - about 24% of the transactions have discrepancies in the actual amount received. To explore it, we plotted the actual spent value (actual_spent_value) and the difference between the actual received value and amount (diff_received) in the same plot for transactions with discrepancy.

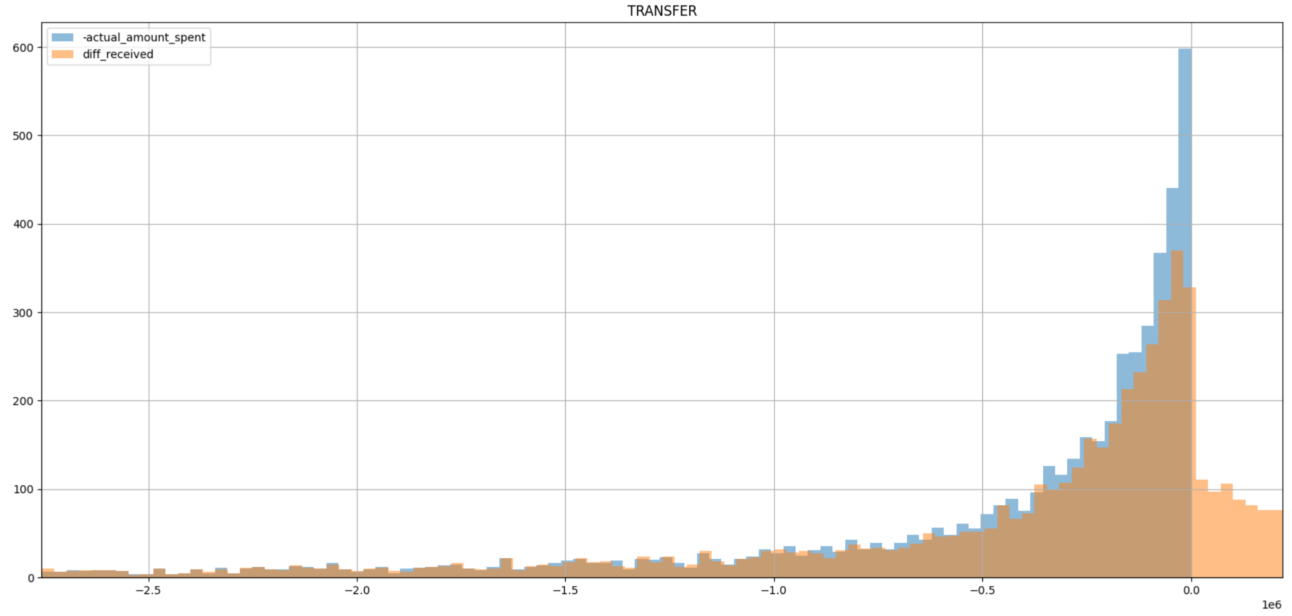


Figure 4: Distribution of spent value and difference between received and sent value

According to the plot, both variables have almost the same distribution, which leads us to the conclusion that in these transactions destination balance was almost the same, which can be caused by the delay between payment and actual receiving and money. Thus, we investigated all problems with the data and explain most of them. Now we can consider our data reliable.

2.3 Feature generation

Due to the rather small number of features in this dataset, it might make sense to "generate" new features in order to bring up hidden information for the analysis. This process is described below:

2.3.1 Names of accounts

As mentioned above, all names of bank accounts start with the letter M or C (the letter M is possible only in the destination account's name). To use this fact we added a feature that indicates if the name of the destination account starts with C or not.

2.3.2 Debt features

According to our research, sometimes the amount of transaction exceeds the current balance. In this case, the origin bank account should have debt, but this is not represented in the data, because the balance is never negative. To take debt into consideration we added a feature "has_debt", which is an indicator of whether the amount exceeds the balance or not, and a feature "debt", which is equal to the difference between the amount and current balance for transactions that have debt and zero otherwise. These features allow us to cover cases when the amount exceeds the balance and provide our ML models with full data.

2.3.3 General knowledge

Fraud is a widespread problem, and that is why we have some hypotheses based on our own experience about what factors can indicate fraud. The first idea is that in fraudulent transactions, amount is often equal to the current balance - fraudsters try to steal as much money as possible. To detect such transactions, we add a flag that the amount of the transaction is equal to the current balance. The second idea is that fraudsters may prefer to use only round numbers (without cents, units, tens, or hundreds) in the number of transactions because they don't know the actual balance or they try to make a transaction as soon as possible. For this hypothesis, we generate an indicator if the amount has cents, units, tens, or hundreds.

2.3.4 Group features

Another interesting finding is that a lot of transactions have the same destination account, which points out that they are connected. To use this fact we grouped transactions by destination name and generated min, max, median, mean, and std over the group for all numerical features.

2.4 Data split

As we emphasized above, a lot of transactions have the same destination bank account, so they are related. Another important thing to remember is that we have a very little percentage of fraud transactions, which is why in common sampling we can not guarantee that we will have enough positive cases in each set. To solve the listed problems, we split data into train, test, and validation sets, keeping all transactions with the same destination in one set and keeping the balance of classes the same for each set.

2.5 Normalization

Some of the models we plan to use are sensitive to absolute values of data. That is why we will normalize all numerical features. Doing so, we used linear transformation which converts the minimal value of the feature to zero, and the maximal value is converted to 1. We train the scaler on the trained dataset to prevent data leaks, and then we apply it to test and validation sets.

2.6 Choice of features

To choose the most important features, we use CatBoost - one of the most powerful models we used in our research. Firstly, we trained the model on all features and measured its accuracy. Then we used some variation of the backward elimination method. We chose top-20 the most important features and received the same accuracy. We repeated this procedure with the top 10 features of the received set and achieved the same accuracy again. After this we started to remove the least important feature one by one and received the following dynamic of accuracy:

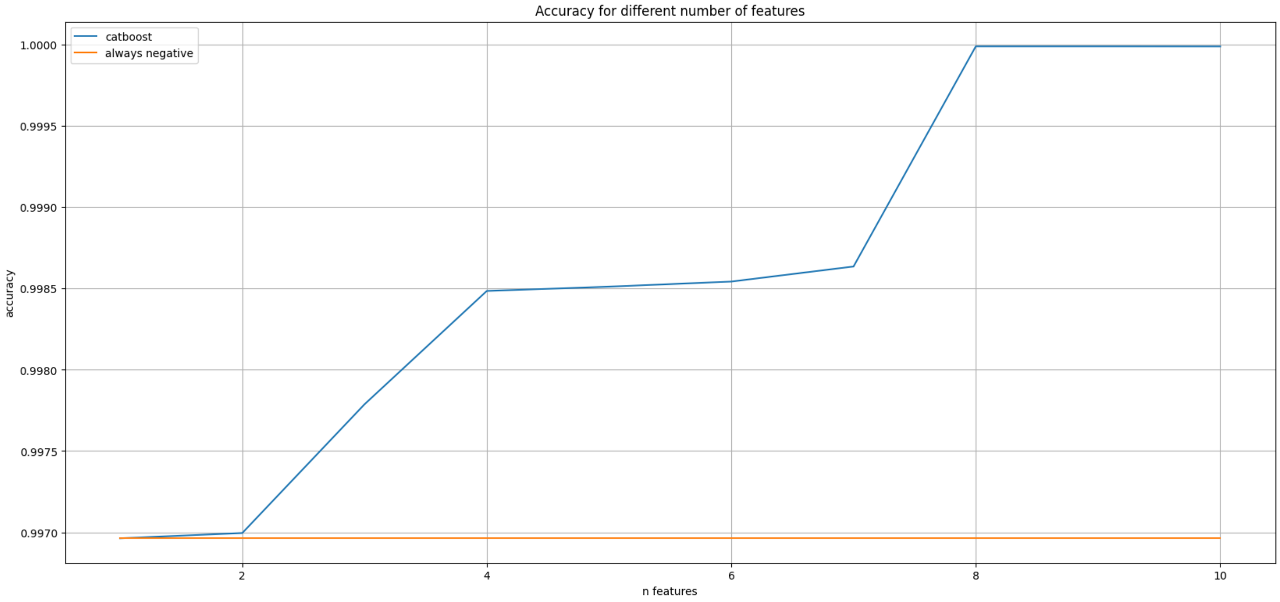


Figure 5: Dependency of accuracy from the number of chosen features

On this plot we have a line that represents the accuracy of the model which always predicts a negative class, to demonstrate that accuracy of 99.70% is not good in our case because of extremely unbalanced classes. According to the received plot, we can achieve maximum possible accuracy with 8 most important features: oldbalanceOrg, step, std_amount, std_oldbalanceOrg, std_oldbalanceDest, amount_is_equal_to_balance, newbalanceDest and debt. As we can see, most of these features were generated by us, so it indicates that our ideas and assumptions allowed us to make data more informative. We will use these features for all models to facilitate a fair comparison.

3 Predictive Analytics

3.1 Baseline Models

To prove the predictive power of the models in this project it needs some simple baselines via the most primitive models. In this work, the following models are used:

- All transactions are not fraud
- All transactions are fraud
- If the amount is equal to balance then fraud, otherwise not fraud

Performance table Baseline models			
Model	Sensitivity	Specificity	Accuracy
All transactions are not fraud	0.00 %	100 %	99.70 %
All transactions are fraud	100.00 %	0.00 %	0.30 %
Fraud when amount equal to balance	97.75 %	100.00 %	99.99 %

Table 1: Performance baseline models

Due to the imbalance in classes (*fraud* and *nonFraud*) in this dataset, the baseline model *All transactions are not fraud* could reach a very high accuracy (and self-evidently a Specificity of 100%). Therefore, accuracy will not be considered a valid instrument to measure model performance in this project. The model *All transactions are fraud* logically only reaches an accuracy of 0.30% and the reverse values for sensitivity and specificity. In order to predict Fraud, none of these models seems appropriate.

Throughout the analysis, the variable *amount_is_equal_to_balance* appeared to be very insightful, so the third baseline model predicts the fraudulent character of transactions with the help of this feature. If the amount (transferred amount) is equal to the balance, the transaction will be flagged as fraud and vice versa. This method performs surprisingly well and will be the baseline model to beat by the machine learning methods in the upcoming pages.

3.2 Logistic Regression for Classification

3.2.1 Method Description

Logistic regression is one of the basic machine learning algorithms and distinguishes itself by its applicability and simplicity. It gives in the first place no classification to one class or the other but a probability for an observation to belong to either of these classes. This probability can be expressed as follows.

$$p = \frac{e^y}{1 + e^{-y}}$$

where

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i * x_i$$

To get a linear combination of all predictors, the logarithm is taken and the so-called logit function is derived [Kas18]

$$log\left[\frac{p}{1-p}\right] = \beta_0 + \sum_{i=1}^{n} \beta_i * x_i$$

To finally derive p, one can take the previously received result for this observation and de-logarithmize it by using the exponential function to get rid of the logarithm and get the odds. [Sau22] Odds are the ratio of the probability of the positive class to the probability of the negative class. From this point, one can then easily compute p by [Kas18]

$$p = \frac{Odds}{1 + Odds}$$

In this project, the two classes are *not Fraud* and *Fraud*. Converting calculated probabilities to classes is done via a cutoff value of 0.5. This value is a general assumption and can be adapted but in this project, there is no base to assume this value to be incorrect.

3.2.2 Data Transformation

Data does not need to be normalized or standardized, even though, categorical variables must be transformed into dummy variables. In our case, the data is already normalized and there is no need to reconvert it since it has no impact on our results.

3.2.3 Method application

The logistic regression for classification was implemented in R. The method that was used to run the logistic regression in this project is the `glm()`-function from the `stats()`-package. The default settings of the function were kept and no further optimization was done as it will be for several other methods in the upcoming report. Also, since this method is computational not very intensive, the function could be executed on the whole training set and validated with the test set.

3.2.4 Performance and method evaluation

The logistic regression with default values in the `glm()`-function works very well and could already beat the "best" baseline model. 8 records were mistakenly considered not fraud while being so (out of 1644 fraudulent records). 2 values were considered false positives which is satisfying as well even though sensitivity is considered the more important metric in this project. The most important variable is by far *oldbalanceOrg*, followed by *newbalanceDest* and the newly created *debt*, see Figure 7.

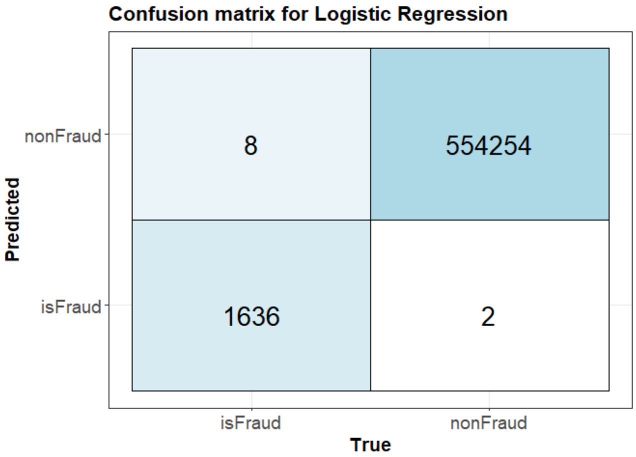


Figure 6: Confusion matrix for the logistic regression

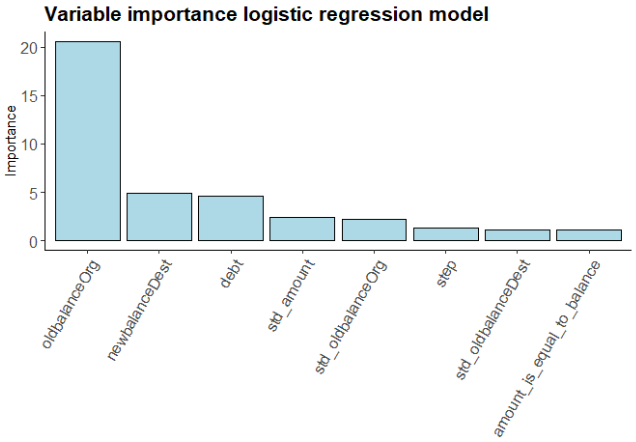


Figure 7: Variable importance logistic regression model

3.3 K Nearest Neighbours for Classification

3.3.1 Method Description

K Nearest Neighbours falls in the supervised learning technique, which tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the number of k points closest to the test data. The algorithm calculates the probability of the test data belonging to the classes of 'K training data and the class holding the highest probability of being selected. The probability of the class i can be defined as:

$$p_i = \frac{1}{\sum\limits_{j=1}^{k} w_j} \sum_{j=1}^{k} w_j \cdot I(point\ has\ class\ j)$$

3.3.2 Data Transformation

The KNN is sensitive to the scale of data because it relies on the distance between points which depends on scale. That is why it is important that data was normalized in advance.

3.3.3 Method application

The K Nearest Neighbours method for classification was implemented in Python. The KNN object has the class KNeighborsClassifier which is a part of the package sklearn.neighbors. The method has many different parameters, we varied only three of them:

- Distance metric - Manhattan or Euclidean
- Weights of neighbors - all points have the same weight or weight is equal to $\frac{1}{distance\ to\ the\ point}$
- Number of neighbors - from 1 to 10

Chosen parameters match the intuition that in the case of highly unbalanced classes considering more than one neighbor can lead to poor quality. Fitting the algorithm consist of saving the training dataset, while prediction consists of calculation of $train\_size \times test\_size$ distances between points, which is a very time-consuming procedure in the considered case. That is why the usage of this method is recommended only in case it demonstrates outstanding results.

3.3.4 Performance and method evaluation

The method with the default number of neighbors equal to 5 demonstrated extremely low accuracy because of unbalanced classes. With the reduction of the number of neighbors method demonstrated much better results, but significantly worse than other models. The best model has the following parameters:

- Number of neighbors is equal to one
- All weights of points are the same
- p = 1, i.e. Manhattan distance was used

The most important variable is by far *amount_is_equal_to_balance* while *oldbalanceOrg* is on second rank but distinct less important. The sensitivity and specificity of the method are not better than these values for other models, so usage of this model is not recommended.

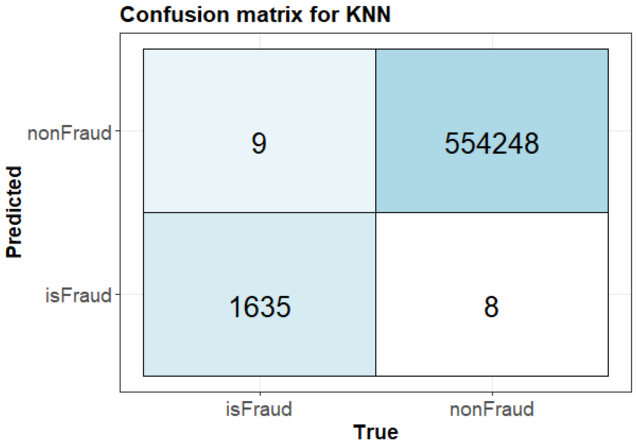


Figure 8: Confusion matrix for KNN

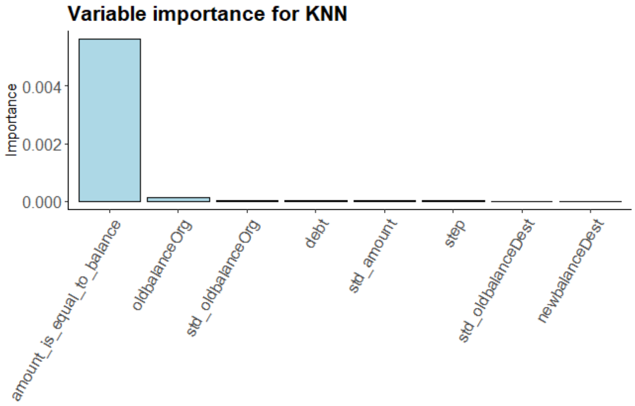


Figure 9: Variable importance for KNN

3.4 Naive Bayes for Classification

3.4.1 Method Description

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. The assumption made is that the value of a particular feature is independent of the values of any other features given the class variable. The advantage of the Naive Bayes method is that it requires a small amount of training data to estimate the parameters necessary for classification and that the classifier can be trained incrementally. The most popular examples of Naive Bayes are Spam filtration, Sentimental analysis, and classifying articles. The Naive Bayes is made up of two words: Naive assumes the occurrence of a certain feature is independent of the occurrence of other features while Bayes depends on the principle of Bayes' Theorem. The Bayes Theorem formula is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

3.4.2 Data Transformation

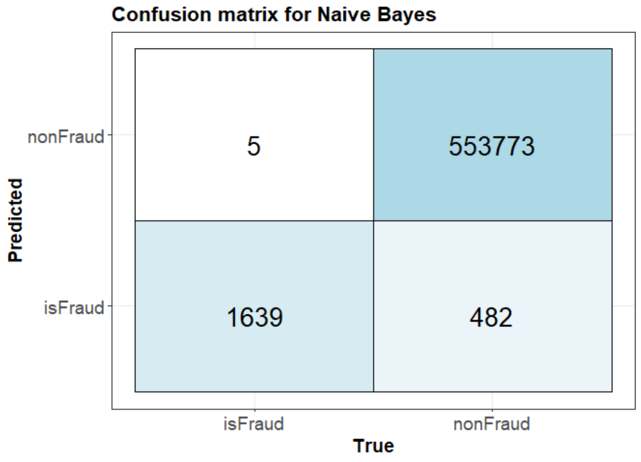
Naive Bayes for classification is not affected by the scaling of variables. In our case, the variables are already scaled and there is no need to convert them back since there is no effect on the final output.

3.4.3 Method application

The Naive Bayes method for classification was implemented in Python. The method is implemented in class GaussianNB which is a part of the package `sklearn.naive_bayes`. It doesn't have any parameters, so the only version of the algorithm was tested.

3.4.4 Performance and Method evaluation

This tested version delivered a sensitivity of 99.70% and therefore 5 false negative values what is an acceptable result but also predicted nearly 500 false positive values. Even though the main focus of this project is sensitivity, this value is high and not satisfying. Again, *amount_is_equal_to_balance* is the most important feature. On ranks 2 and 3, one can find similar values *std_amount* and *oldbalanceOrg*, respectively, but both with low values compared to *amount_is_equal_to_balance*, see Figure 11. It means that the most of predictive ability of the model is explained by *amount_is_equal_to_balance*, and the other features have a smaller impact on the final prediction.



Predicted \ True	isFraud	nonFraud
nonFraud	5	553773
isFraud	1639	482

Figure 10: Confusion matrix Naive Bayes

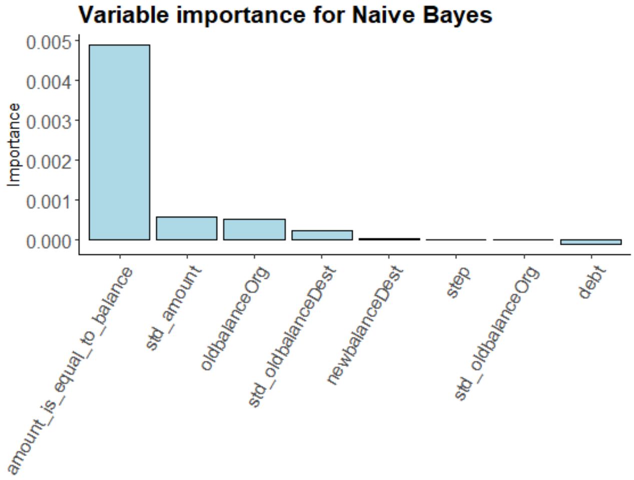


Figure 11: Variable importance Naive Bayes

3.5 Classification Trees

3.5.1 Method Description

The Classification Tree method is a type of decision tree where the outcome is discrete, in this case, binary (*fraud* or *nonFraud*). [Liu20] The principle how the method classifies data is based on decision rules, therefore *Decision*tree. The data are partitioned into nodes, according to these learned rules. In the learning phase, the algorithm learns these rules when trying to create as pure nodes as possible. These rules are then applied to predict the outcomes for test data sets or single observations.

3.5.2 Data Transformation

The classification tree method is a tree-based algorithm and is therefore insensitive to the scaling of features. [Ary22] Nevertheless, the features are already normalized, and will not be reverted into their original format because it has no impact on the result.

3.5.3 Method application

This method was implemented in R. As a function, the typical `rpart()`-function was used. As for neural networks later on in this project, the function offers the possibility to set several parameters, here in the `rpart.control()`-section. For instance, one can select the maximum depth of the tree, `maxdepth`, the minimum observations in the final leaf, `minbucket`, and the complexity parameter `cp`. To learn about the performance of different combinations of these parameters, 2 smaller training sets (roughly 4,000 observations, each) were created, one with both classes balanced and one with random selection, so keeping the imbalance of classes. To limit complexity, only the parameters `maxdepth` and `cp` were varied via 5 levels each, so that 25 different combinations were created. For each combination, the respective tree was calculated for both of the small training sets. These trees were then used to make predictions on the validation dataset, so that they could be stored, compared, and assessed. The combination that provided the best compromise between sensitivity, specificity and complexity was the one with `maxdepth` = 2 and `cp` = 0.01. Finally, this specification of the classification tree was trained on the whole training set to then predict the test set. Following performance could be achieved.

3.5.4 Performance and method evaluation

The classification tree with the specification from above achieved the following results. 6 fraud transaction were not detected by the model, leading to an sensitivity of nearly 99.64%. Two predictions were false positive what is quite satisfying even though, as mentioned above, the focus is on sensitivity. The by far most important feature is *amount_is_equal_to_balance*, followed by *oldbalanceOrg*.

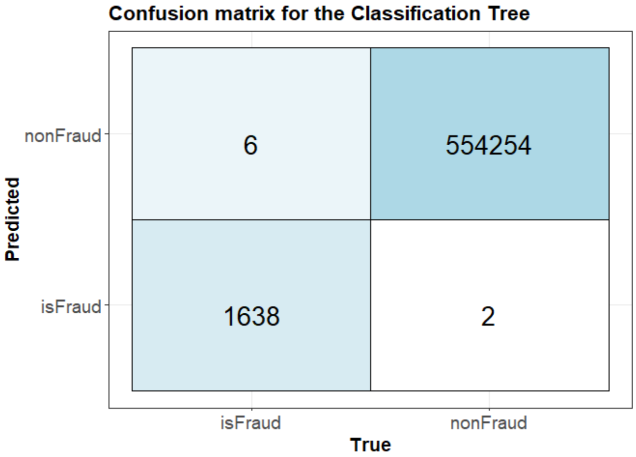


Figure 12: Confusion matrix classification tree

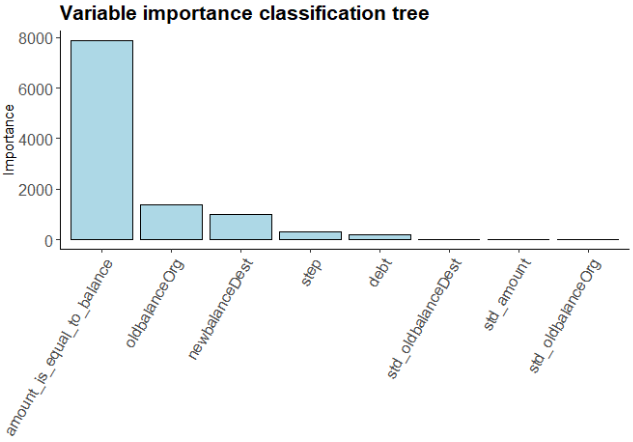


Figure 13: Variable importance classification tree

3.6 Random Forest for Classification

3.6.1 Method Description

This method belongs to supervised learning as the methods before. It incorporates the principles of decision trees, ensembles, and bootstrapping. [Beh22] Since decision trees and ensembles are explained in other parts of this report, only bootstrapping will be explained in this chapter. Bootstrapping means taking a high number of random samples (with replacement) of a given dataset, on which one can run different machine learning methods like decision trees or simply take the means of them. Bagging, a kind of antecedent to random forest means applying classification or regression trees on these subsamples and taking the majority vote or the average, respectively, to finally obtain a result in form of a classification or a numeric value for example. This method should be superior to simple trees. Nevertheless, it appears that single trees or their results are correlated. To overcome this problem, the random forest method can be applied. In this method, the bagging procedure is principally executed but without taking all input variables for each subset. So, a random set of features is selected by the model for each tree, which should improve the ensemble prediction and should therefore be superior to bagging.

However, the performance on imbalanced datasets, as in this project, is limited. [Bro20] Also, one can expect a higher computational time for a random forest than for a single regression tree for obvious reasons. On the other hand, the problem of over-fitting is addressed, compared to single tree models. [R21]

3.6.2 Data Transformation

As this method is based on building decision trees, as described above, it is insensitive to the scaling of features. [Ary22] Since the data have been normalized before, they will not be un-normalized but used as they are since they will not influence the result.

3.6.3 Method application

This method was approached in R. Due to computational time, it was executed on two sub-samples of the training set and evaluated on the test set. The two sub-samples were the randomly drawn set (with the approximately original ratio between *fraud*- and *non-fraud* transactions) and the balanced set (with equal amount of *fraud* and *non-fraud* transaction). Following the introduction to this method, one should expect to receive better values from the balanced set but to check the introduction and the expectation, both sets will be used. As a function in R, the `randomForest`-function from the `randomForest` package was used. To save computational time, the function `randomForest` was previously compiled, using `cmpfun()`.

3.6.4 Performance and method evaluation

Referring to Figures 15 and 14, several conclusions can be drawn. The first observation is that both approaches produced acceptable and similarly good results but still differ. The model from the balanced sample delivers obviously a lot more mis-predictions in sum, nevertheless, the composition of errors makes it reasonable to consider using this method as the better one. While the model from the random sample has 0 false positives, the model from the balanced sample has 136 of this kind that would vote against it. On the other hand, it has only 6 False Negatives which is much better, compared to the model derived from the random sample which has 37 of this kind. This is interesting since in this special application (predicting fraud), one can consider it much more relevant to detect a higher ratio of indeed fraudulent transactions, taking into account a higher error rate in terms of ”false alerts”. Since this is an arguable topic, this project will not suggest a clear favorite at least between these two approaches, but there are other methods that performed better than either of both.

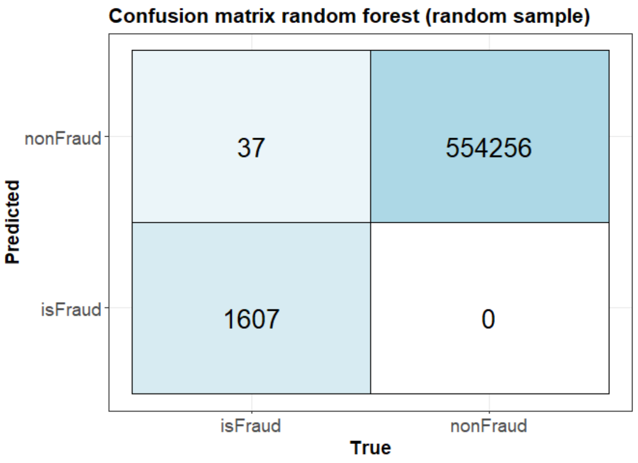


Figure 14: Confusion matrix random forest (random sample)

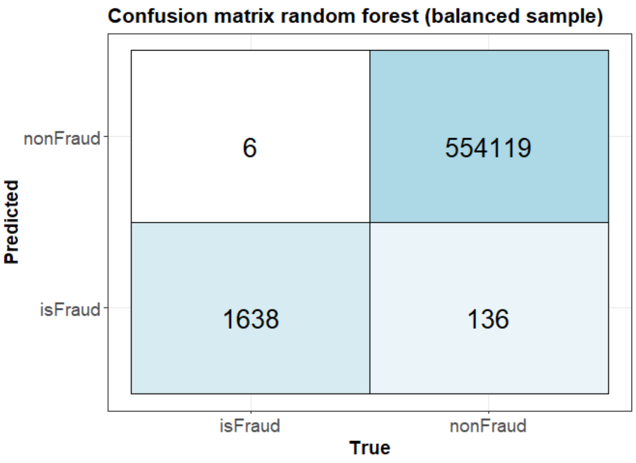


Figure 15: Confusion matrix random forest (balanced sample)

The concrete performance values can be seen in Table 2 below.

Performance table for the two random forest versions			
Model	Sensitivity	Specificity	Accuracy
Model from random sample	97.75 %	100.00 %	99.99 %
Model from balanced sample	99.64 %	99.98 %	99.97 %

Table 2: Performance table random forests

Also regarding variable importance, both methods differ. Both consider *amount_is_equal_to_balance* as the most important one but while the model from the random sample put barely any importance on other variables, the model from the balanced sample gives also considerable weight to *debt* and *oldBalanceOrg*. The corresponding graphics can be seen in the appendix.

3.7 Neural Networks for Classification

3.7.1 Method Description

Neural Networks are especially useful when a complex relationship between input and outcome variables is expected. The supervisor does not indicate a type of relationship in advance. A network consists of 3 layer types, one input layer, hidden layers, and one output layer, each consisting of nodes. One node of the input layer takes the value of one feature of one observation. Therefore, one has as many input layers as features. In this project, a classification in *Fraud* and *not Fraud* is targeted. Hence, in the output layer, there exist 2 nodes, one for each class. The nodes in the hidden layers take the values of every node of the previous layer, process them and return the result until the output layer is reached. The functions with which the nodes process data might be linear but can also be exponential or any other function. Each node has a "personal" bias and each value that is transmitted to the next layer has a weight. After the computation of the values for the output nodes, the error is calculated and propagated back through the model where each node that was involved in this error gets partly responsible for this mistake. In the next runs, the weights and biases are adapted to minimize the error. This is being conducted for all observations. [Shm18]

3.7.2 Data Transformation

In case of using a logistic function at the beginning, it will be useful to normalize the data in advance, this means bringing them on a scale from 0 to 1. [Shm18] Since the data have already been normalized, no further transformation needed to be done.

3.7.3 Method application

This method was implemented in R. As a function the `neuralnet()`-function from the neural net package was used. This method gives the chance to select an appropriate number of layers and nodes for each layer. In order to choose the optimal combination, the following approach has been employed. At first, it is computationally very intense to create a neural net with a large training set. Therefore, two smaller samples of the training set have been created in advance, one with an equalized ratio between fraud and non-fraud observations and one just as a random sample of the initial training set. Both have due to computational time not more than slightly above 4,000 observations. As a second step, all combinations of up to 9 layers and 9 nodes have been created by two for-loops. With every one of these combinations, a neural net has been calculated for both, the balanced and the random training set. With these neural nets, the fraud/non-fraud observations were predicted for the test dataset. To measure now if the networks of one training set are superior over the ones of the other training set, and if they suggested the same combination of layers and nodes, the results of the predictions in the sense of sensitivity, specificity, and accuracy were stored, compared, and assessed. The combination of 1 layer and 4 nodes appears to be a good choice because it has on the one hand the 4. best sensitivity for the balanced training set (0.9999820) and also very high sensitivity for the random training set (0.9999621) and is on the other hand much less complex than comparable well-performing combinations. In this context shall be noted that the values for specificity for both training sets were except for a few examples equal to 0.9963504. Therefore and because of the project's intention to detect as many fraud cases as possible, sensitivity was considered a better basis for selecting the most suitable number of nodes and layers. Also should be noted that the values for sensitivity were in most cases better than 99.8%. Finally, to run the neural network, the combination mentioned above was used. Following performance could be achieved.

3.7.4 Performance and method evaluation

As could already be expected from the description in Method Application, the model could also score high values when being trained on the whole training set, see Figure 16. The network reached a sensitivity of over 99.57% (7 undetected fraud transactions out of 1,644) and a specificity of exactly 100%, meaning that not a single value was flagged fraud while actually not being fraud. The overall accuracy is also very close to 100%.
Since measuring variable importance in neural networks is not as straightforward as it is for the other methods, the second graph, besides the confusion matrix will not consider variable importance but the neural network itself. In this Figure 17, one can see the scheme of the implemented neural net with one hidden layer with 4 nodes. One can clearly see the two output nodes, marking 0 and 1 as possible outputs. For reasons of clarity, it was refrained from displaying weights.

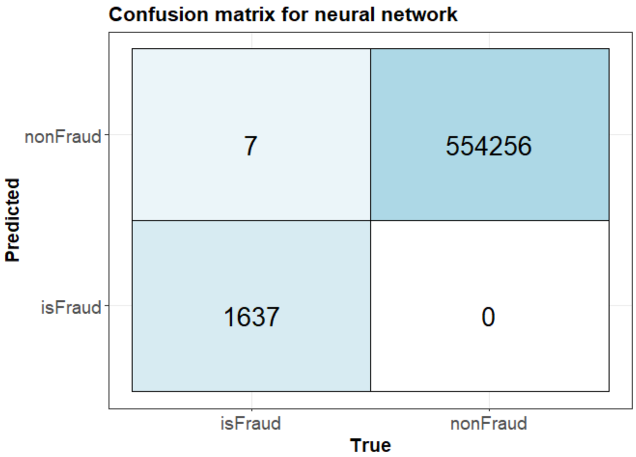


Figure 16: Confusion matrix for neural network

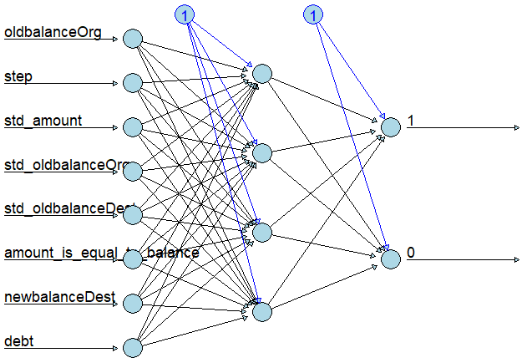


Figure 17: Neural Net

3.8 Ensemble

3.8.1 Method Description

It is a machine learning technique that combines several base models in order to produce one optimal predictive model. This approach allows the production of better predictive performance compared to when using a single model. In this work, CatBoost was used as one of the most powerful models which can perform very well without any adjustment of parameters. The basis of the model is gradient boosting over decision trees. In the training process, the trees are built consequently - the new tree always has reduced loss than the previous one. Building each tree consists of the following steps:

- Calculation of splits
- Transformation of categorical and text features into numerical
- Choosing the structure of the tree
- Calculating leaves of tree

3.8.2 Data Transformation

CatBoost does not need any data transformation. Moreover, it can convert categorical and text features into numerical variables by itself.

3.8.3 Method application

This method was implemented in Python. The implementation of the method is class CatBoostClassifier from the package catboost. This class has a good built-in adjustment of hyperparameters, so the default configuration was kept.

3.8.4 Performance and method evaluation

According to the received confusion matrix, CatBoost demonstrated extremely high predictive power. It has only 6 FN cases, while all other cases were predicted correctly. It allows for achieving very high values of target metrics. Moreover, according to the feature importance chart, all features have close values of importance to each other, which may mean that all features are necessary and each of them contributes to the final prediction well enough. This assumption was proven by conducting research on needed features based on backward elimination.

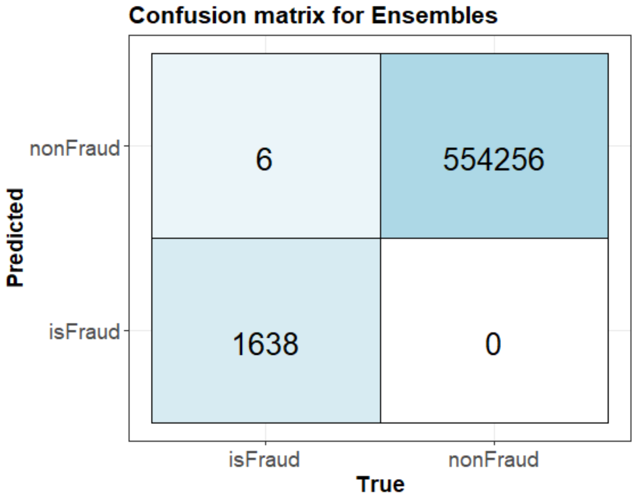


Figure 18: Confusion matrix for Ensembles

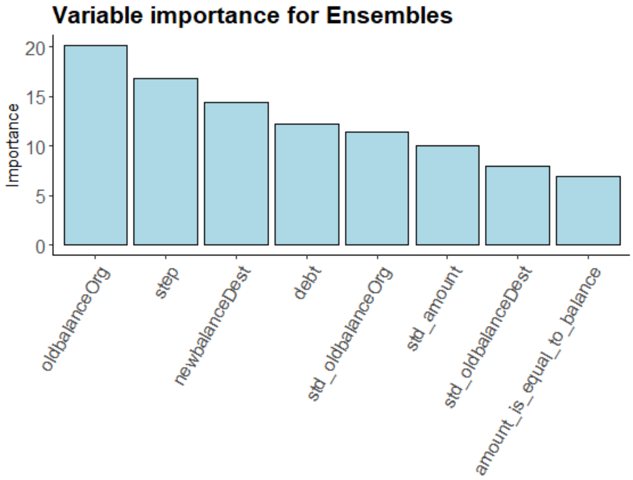


Figure 19: Variable importance Ensembles

4 Assessing Model Performances

4.1 Performance

In this paragraph, the performance of the models that were established in this report will be compared with each other in order to finally state the overall ”best” model or alternatively the model that finds the best compromise between different aspects of performance. Table 3 includes all values for the evaluation metrics that were used in the report, *Sensitivity*, *Specificity* and *Accuracy*. The values are rounded to the second digit and visualized in Figure 20 below.

Performance table Baseline models			
Model	Sensitivity	Specificity	Accuracy
Nothing is Fraud	0.00 %	100.00 %	99.70 %
Everything is Fraud	100.00 %	0.00 %	0.30 %
Fraud when amount equal to balance	97.75 %	100.00 %	99.99 %
Logistic regression	99.51 %	100.00 %	100.00 %
KNN	99.45 %	100.00 %	100.00 %
Naive Bayes	99.70 %	99.91 %	99.91 %
Classification trees	99.64 %	100.00 %	100.00 %
Random Forest (random sample)	97.75 %	100.00 %	99.99 %
Random Forest (balanced sample)	99.64 %	99.98 %	99.97 %
Neural Networks	99.57 %	100.00 %	99.70 %
Ensemble (Catboost)	99.64 %	100.00 %	100.00 %

Table 3: Performance all models

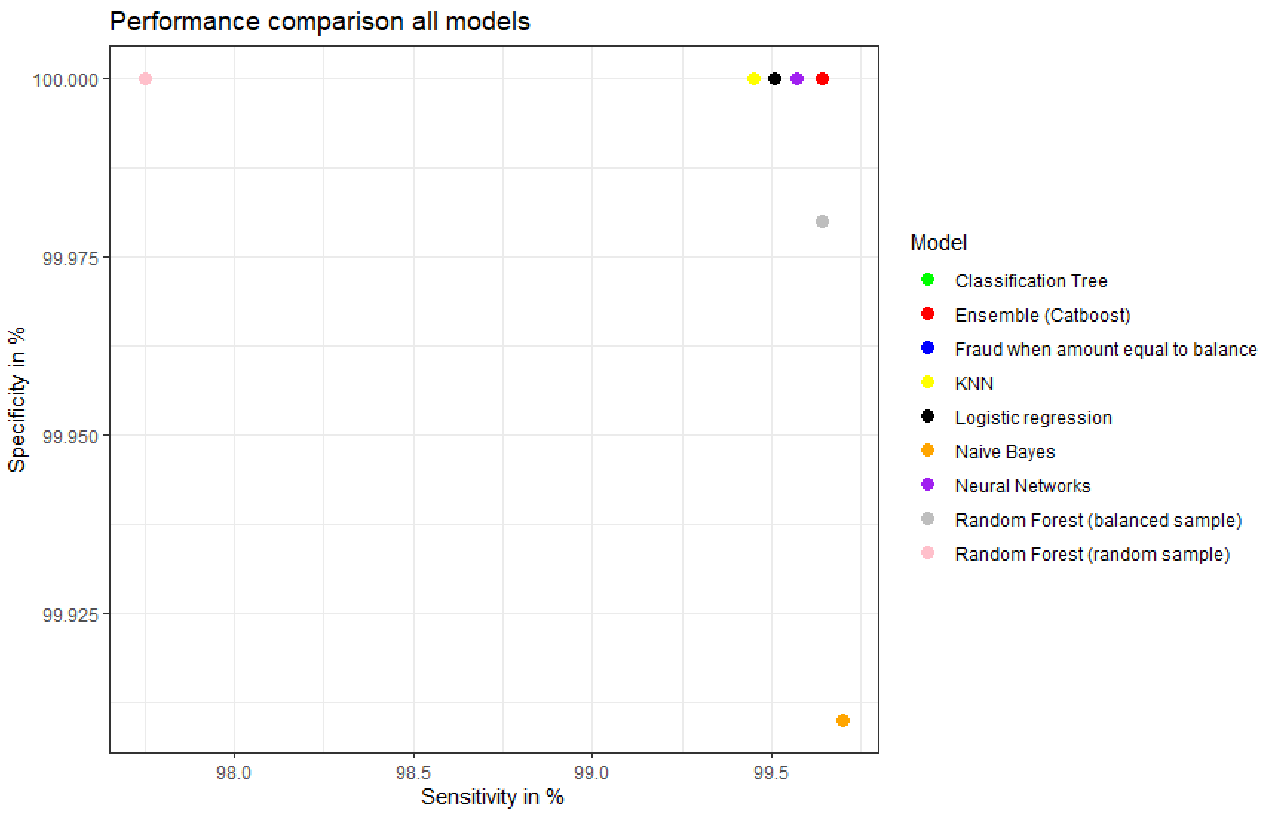


Figure 20: Comparison of models in terms of sensitivity and specificity

The first 3 models are the baseline models. Due to the major imbalance in classes, labeling nothing as fraud can reach a very high overall accuracy of more than 99% but, self-speaking does not identify any fraudulent observations. Similarly useful is the opposite, labeling everything as fraud, even though it identifies every fraud observation. The third baseline model tries to identify fraudulent observations with the help of the created feature *amount_is_equal_to_balance* which has two levels. This method performs well and reaches a sensitivity of more than 97% while maintaining a specificity of nearly 100%. This model is the only acceptable baseline model and was selected at the beginning as the baseline model to compare against the machine learning algorithms.
Since this project's predominant aim is to find as many fraud transactions as possible, sensitivity is the No. 1 metric to evaluate model performance. The model which performed best in that sense is the Naive Bayes model which misses only 5 fraud observations and achieves a sensitivity of 99.70%. Unfortunately, it also creates a lot of false positive values. There are 3 models (Classification trees, Random Forest (random sample) and Ensemble) that miss only 1 fraud transaction more and which perform much better in the sense of specificity. The Ensemble method is the only one of these three that does not create a single false positive observation. Therefore, this method creates the best combination of sensitivity and specificity. This results can also be seen in Figure 20.

4.2 Feature importance

This additional section shall give a brief overview about the use of features throughout the models. Table 4 provides a summary of the features that were considered the most or second most important by at least one model. It also says which model ranked them with such high importance. As one can easily see, *amount_is_equal_to_balance* is the most dominant feature as it was said to be the most important one by 5 out of 7 models. So, using this single feature as a baseline model could find some justification.

Most important features		
Feature	First	Second
oldbalanceOrg	Ensembles Logistic Regression	Random Forest (random sample) Classification Tree KNN
step		Ensembles
amount_is_equal_to_balance	Random Forest (random sample) Random Forest (balanced sample) Classification Tree Naive Bayes KNN	
debt		Random Forest (balanced sample)
std_amount		Naive Bayes
newbalanceDest		Logistic Regression

Table 4: Most important features

5 Conclusion

Online payment fraud is an upcoming topic that leads to large annual losses. To tackle this issue, actions must be taken by different agents, let it be the banks or the clients themselves. The aim of this project was to train different machine learning models to accurately predict fraudulent transactions using a comprehensive data set of more than 6 million transactions. The primary aim was to precisely predict such transactions and by this outperform a baseline model. This goal was achieved at a satisfying level. The project proposed models that detected nearly every fraudulent transaction while minimizing false alerts, and therefore maintaining high values in both, sensitivity and specificity. Both metrics are important especially for banks to terminate fraudulent transactions but also to not terminate transactions that are legal in order to provide high customer satisfaction and security at the same time.

To finally state the "best" model is not unambiguously possible. One could see that a simple logistic regression with default values delivered quite acceptable results. When putting more computational power in place and searching for the best neural network, one can beat this result by a few predictions. Regarding the combination of sensitivity and specificity, the ensemble method delivers the best result.

6 Appendix

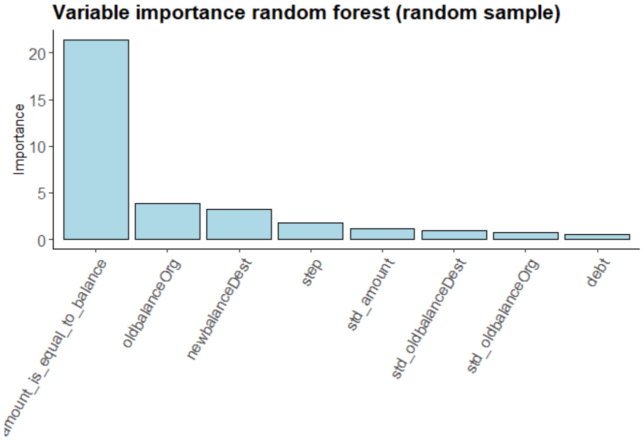


Figure 21: Variable importance random forest (random sample)

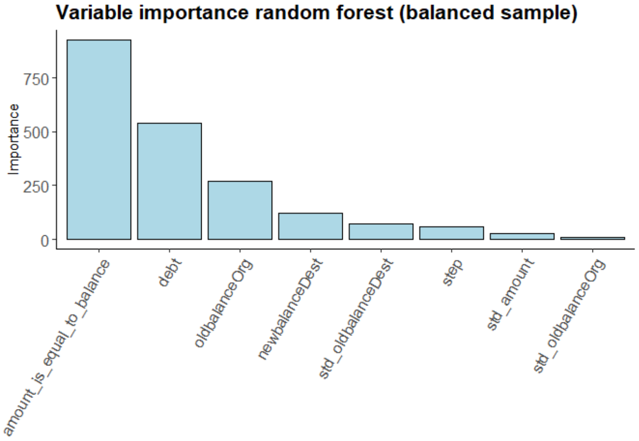


Figure 22: Variable importance random forest (balanced sample)

References

[Ary22] Nisha Arya. Does the random forest algorithm need normalization? https://www.kdnuggets.com/2022/07/random-forest-algorithm-need-normalization.html, 2022. Accessed on 2014-12-14.

[Bec18] Marcus W. Beck. NeuralNetTools: Visualization and analysis tools for neural networks. *Journal of Statistical Software*, 85(11):1–20, 2018.

[Beh22] Nima Beheshti. Random forest classification. https://towardsdatascience.com/random-forest-classification-678e551462f5, 2022. Accessed on 2014-12-16.

[Bro20] Jason Brownlee. Bagging and random forest for imbalanced classification. https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/#:~:text=Bagging%20is%20an%20ensemble%20algorithm,used%20in%20each%20data%20sample., 2020. Accessed on 2014-12-17.

[Cop21] Daniela Coppola. E-commerce payment fraud losses worldwide 2020-2023. https://www.statista.com/statistics/1273177/ecommerce-payment-fraud-losses-globally/, 2021. Accessed on 2022-11-22.

[DS22] Matt Dowle and Arun Srinivasan. *data.table: Extension of 'data.frame'*, 2022. R package version 1.14.6.

[exp] experian.com. Steps to take if you are a victim of credit card fraud. https://www.experian.com/blogs/ask-experian/credit-education/preventing-fraud/credit-card-fraud-what-to-do-if-you-are-a-victim/. Accessed on 2014-12-11.

[FGW19] Stefan Fritsch, Frauke Guenther, and Marvin N. Wright. *neuralnet: Training of Neural Networks*, 2019. R package version 1.44.2.

[Kas18] Alboukadel Kassambara. Convert logit to probability. http://www.sthda.com/english/articles/36-classification-methods-essentials/151-logistic-regression-essentials-in-r/, 2018. Accessed on 2014-12-16.

[Kuh22] Max Kuhn. *caret: Classification and Regression Training*, 2022. R package version 6.0-93.

[Liu20] Clare Liu. Decision tree intuition: From concept to application. https://www.vebuso.com/2020/01/decision-tree-intuition-from-concept-to-application/, 2020. Accessed on 2014-12-23.

[LW02] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[R C22] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2022.

[R21] Sruthi E R. Understanding random forest. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/, 2021. Accessed on 2014-12-17.

[Sau22] Sebastian Sauer. Convert logit to probability. https://sebastiansauer.github.io/convert_logit2prob/#:~:text=To%20convert%20a%20logit%20(%20glm,%2F%20(1%20%2B%20odds)%20., 2022. Accessed on 2014-12-16.

[Shm18] Galet Shmueli. *Data Mining for Business Analytics: Concepts, Techniques, and Applications in R.* Wiley, Hoboken, 2018.

[TA22] Terry Therneau and Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2022. R package version 4.1.19.

[WFHM22] Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. *dplyr: A Grammar of Data Manipulation*, 2022. R package version 1.0.10.

[Wic16] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York, 2016.

[Xie22] Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2022. R package version 1.41.