Capstone Project HarvardX Data Science

Credit Card Fraud Detection

Jan L. Döring

2022-11-03

Contents

Introduction

In the modern world, credit card purchases have become indispensable. There are several reasons why digital banking is superior to cash payments and is becoming more and more popular. However, the digital system also has some weaknesses, most of which come from the user. Carelessness and negligence on the internet make users vulnerable and open up ways for criminals to authenticate fraudulent credit card purchases. For this reason, banks are willing to invest a lot of money to detect these frauds with the aim of protecting their customers and maximizing their satisfaction with the bank's service.

Of all the criminal acts in the financial sector, credit card fraud is the most widespread and has the greatest negative impact due to the ease and sheer volume in which it occurs.Although many companies take serious measures to combat and prevent these scams, they can never be completely eradicated because the criminals eventually find a weakness in these measures. The benefits of fraud detection technology still outweigh the risks of the large investment and inconvenience caused. Therefore, combating fraud activities by continuously improving data mining and machine learning is one of the most important approaches to prevent the losses caused by illegal activities.

In this practical approach we want to use different machine learning models to identify fraudulent credit card transactions. In addition to overall accuracy, we pay particular attention to the ratio of specificity and sensitivity, as we want to detect as many fraudulent transactions as possible but not on the cost of false positive detection of non fraudulent transactions.

Dataset

The dataset is very imbalanced, only 0,017% of transactions are fraudulent while the rest represents legitimate transactions. The described imbalance forms a problem for predictive modeling as most of the machine learning algorithms which are used for classification were designed to be applied to balanced dataset. In many cases the minority class is the class of interest, so it is in credit card fraud detection. In order to overcome this problem we will use Synthetic Minority Oversampling Technique (SMOTE). The SMOTE algorithm can be described as follows:

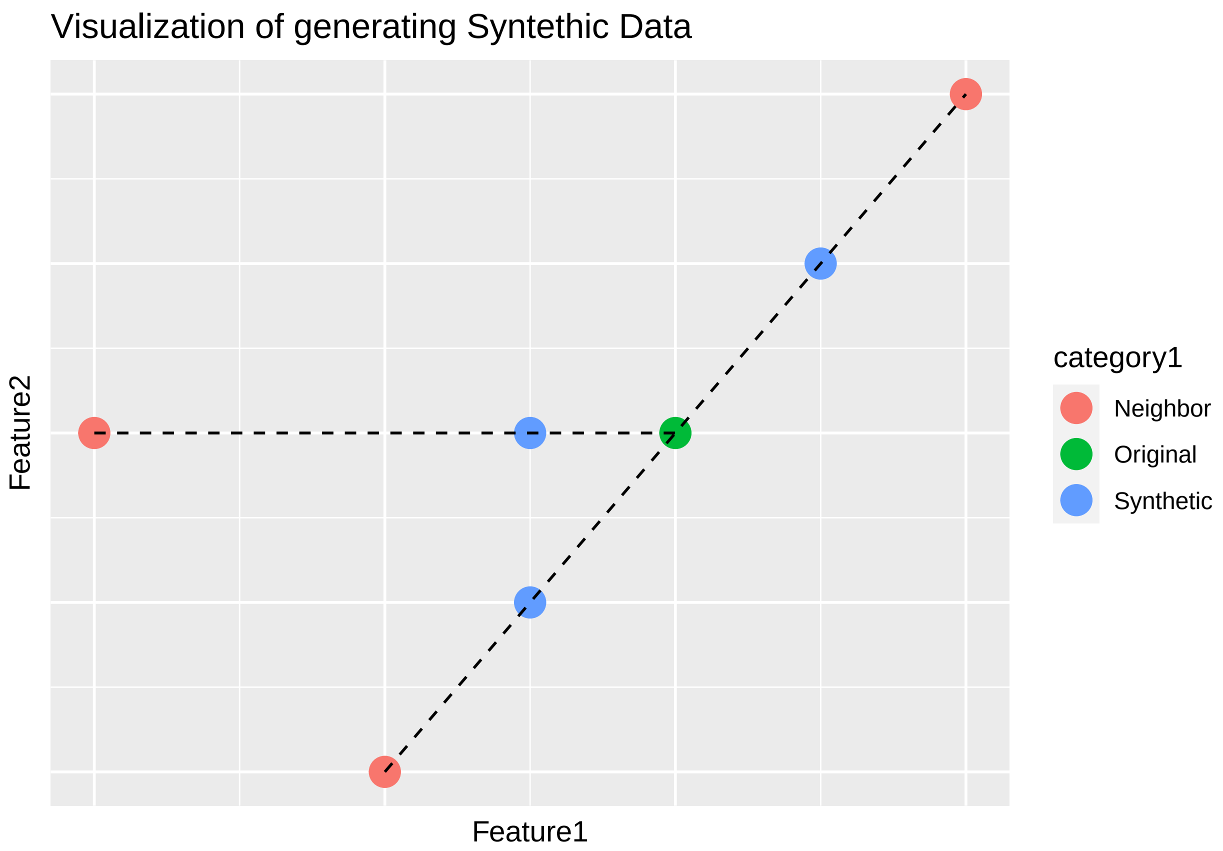
1. Take the difference between a sample and its nearest neighbor

2. Multiply the difference by a random number between 0 and 1

3. Add the difference to the sample to generate a synthetic example in the feature space

4. Continue with the next nearest neighbor up to the user defined number

Lets assume we have an imbalanced dataset. We plot a subset of the minority class in order to visualize the idea of SMOTE. For simplicity we only consider two features, although we can imagine that the subsequent steps may take place in a N-dimensional space.



We consider the first row of a dataset and compute the k nearest neighbors for this data point. We select a random nearest neighbor out of the k nearest neighbors, and compute the difference between the two points. This difference is multiplied by a random numeric between 0 and 1. The resulting value represents a synthetic example along the dashed line between these two points. The number of repetitions of this process, is depending on the amount of synthetic data we want to generate. The number of duplicates can be calculated by the following formula:

$$n = \frac{1 - r_0}{r_0} \cdot \frac{n_0}{n_1} - 1$$

$$r_0 = desired\ ratio$$

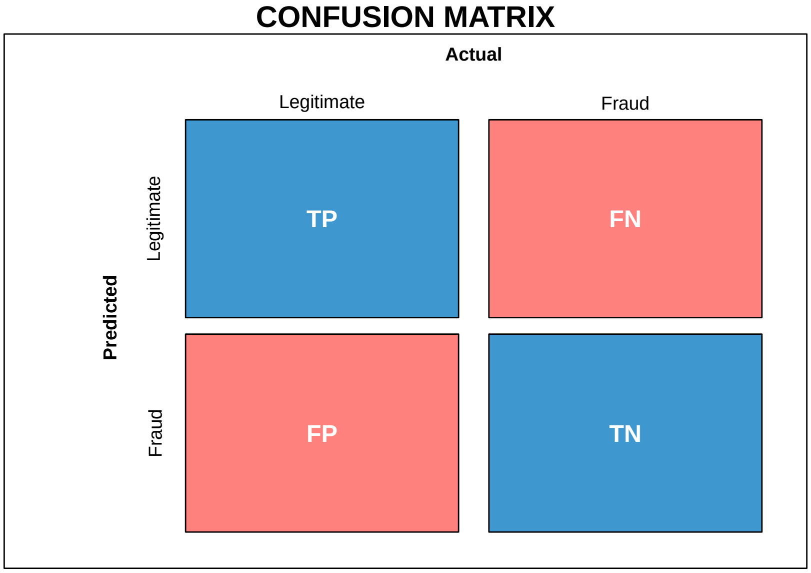
$$n_0 = number\ of\ rows\ corresponding\ to\ the\ majority\ class$$

$$n_1 = number\ of\ rows\ corresponding\ to\ the\ minority\ class$$

The SMOTE algorithm has some advantages in comparison to other balancing techniques. We generate synthetic data which are not just a copy of other existing data points so we avoid over fitting like its often times the case with the simple up sample algorithm. Down sampling would be even worse since the machine learning algorithms we want to apply on our dataset would have less training data to learn from.

Model evaluation

Credit card fraud detection is a classic binary classification problem. There are two possibilities, either the purchase is valid or it is fraudulent. In order to describe the performance of our classification models we use a confusion matrix and receiver operating characteristic (ROC).


CONFUSION MATRIX

		Actual	
		Legitimate	Fraud
Predicted	Legitimate	TP	FN
	Fraud	FP	TN

In a confusion matrix, the number of correct and incorrect predictions are compared.As we can see in the table a confusion matrix gives us a quick overview how many predictions for the two binary outcomes TRUE and FALSE (resp. 1 and 0) are actually right and wrong. With sensitivity we describe the ability of our algorithm to predict a positive outcome when the actual outcome is positive. This metric on its own is not enough to judge if our algorithm works well. The algorithm could predict always TRUE and cover all the cases in where the actual value is TRUE and we would measure a sensitivity of 100%. In this case we would ignore all cases where the model predicts TRUE although the actual value is FALSE. This why we also examine the specificity which is a metric for the ability of the algorithm to predict FALSE, when the actual value is FALSE.

The mathematical description for sensitivity (True Positive Rate) and specificity (True Negative Rate) is the following:

$$TPR = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP}$$

Besides these metrics its common to calculate the overall accuracy, precision, prevalence F-Score and Cohen's Kappa for binary classifiers. The overall accuracy is defined as:

$$ACC = \frac{TP + TN}{Total}$$

The problem with the overall accuracy alone as a metric is similar to the use of sensitivity. Depending on the dataset and the ratio of actual negative and positive values, we can achieve a high accuracy like close to 100% with a high sensitivity but still have a very low specificity.

The precision is the ability of a model to predict TRUE when its actually TRUE:

$$PREC = \frac{TP}{TP + FP}$$

The prevalence is a metric for the actual number of TRUE (NOT) that occurs in our dataset:

$$PREV = \frac{NOT}{TOTAL}$$

The F-Score is a weighted average of TPR and precision:

$$F1 = \frac{2TP}{2TP + FP + FN} = 2 \cdot \frac{PREC \cdot TPR}{PREC + TPR} = \frac{1}{\frac{1}{2}(\frac{1}{TPR} + \frac{1}{PREC})}$$

In receiver operating characteristics the true positive rate(sensitivity) is plotted versus the false positive rate (1- specificity). For ROC-Plots the are under the curve (AUC) can be compared for different algorithms. The higher the area under the curve the better is the prediction by the algorithm.

Data Science Process

The main steps in a data science project include:

1. Data collection
2. Data preparation
3. Data exploration
4. Data cleaning
5. Data analyses and visualization
6. Model building and validation
7. Creating a report and publishing the results

Methods

Data Collection/Preparation

The dataset is downloaded from [kaggle.com] and provides a huge dataset in .csv format with a size of 147,3 Megabyte. The dataset is PCA transformed so the features for each observation are obtained by principal compound analysis.

Data Exploration

As a first step the we discover the structure of the dataset, we call the str() function in order to receive some information about the variables and datatypes in the data frame.

```
str(creditcard)
```

```
## 'data.frame':    284807 obs. of  31 variables:
##  $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
##  $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
##  $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
##  $ V3    : num  2.536 0.166 1.773 1.793 1.549 ...
##  $ V4    : num  1.378 0.448 0.38 -0.863 0.403 ...
##  $ V5    : num  -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...
##  $ V6    : num  0.4624 -0.0824 1.8005 1.2472 0.0959 ...
##  $ V7    : num  0.2396 -0.0788 0.7915 0.2376 0.5929 ...
##  $ V8    : num  0.0987 0.0851 0.2477 0.3774 -0.2705 ...
##  $ V9    : num  0.364 -0.255 -1.515 -1.387 0.818 ...
##  $ V10   : num  0.0908 -0.167 0.2076 -0.055 0.7531 ...
##  $ V11   : num  -0.552 1.613 0.625 -0.226 -0.823 ...
##  $ V12   : num  -0.6178 1.0652 0.0661 0.1782 0.5382 ...
##  $ V13   : num  -0.991 0.489 0.717 0.508 1.346 ...
##  $ V14   : num  -0.311 -0.144 -0.166 -0.288 -1.12 ...
##  $ V15   : num  1.468 0.636 2.346 -0.631 0.175 ...
##  $ V16   : num  -0.47 0.464 -2.89 -1.06 -0.451 ...
##  $ V17   : num  0.208 -0.115 1.11 -0.684 -0.237 ...
##  $ V18   : num  0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...
##  $ V19   : num  0.404 -0.146 -2.262 -1.233 0.803 ...
##  $ V20   : num  0.2514 -0.0691 0.525 -0.208 0.4085 ...
##  $ V21   : num  -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...
##  $ V22   : num  0.27784 -0.63867 0.77168 0.00527 0.79828 ...
##  $ V23   : num  -0.11 0.101 0.909 -0.19 -0.137 ...
##  $ V24   : num  0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...
##  $ V25   : num  0.129 0.167 -0.328 0.647 -0.206 ...
##  $ V26   : num  -0.189 0.126 -0.139 -0.222 0.502 ...
##  $ V27   : num  0.13356 -0.00898 -0.05535 0.06272 0.21942 ...
##  $ V28   : num  -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
##  $ Amount: num  149.62 2.69 378.66 123.5 69.99 ...
##  $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

We can see that the features V1-V28 obtained by PCA are hidden features due to confidentiality issues. Only Time, Amount and Class are known observations, while Class represents our desired value to predicted. The dataset contains 284807 observations and the datatype of all variables is numeric.

With the head() function we can call the first six rows of the data frame:

```
head(creditcard)
```

```
##   Time         V1          V2        V3         V4          V5          V6
## 1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##           V7          V8         V9         V10        V11         V12
## 1  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086
## 2 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531
## 3  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369
## 4  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823
## 5  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##          V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
##           V19         V20          V21          V22         V23         V24
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692807
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658
##          V25        V26          V27         V28 Amount Class
## 1  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62     0
## 2  0.1671704  0.1258945 -0.008983099  0.01472417   2.69     0
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66     0
## 4  0.6473760 -0.2219288  0.062722849  0.06145763 123.50     0
## 5 -0.2060096  0.5022922  0.219422230  0.21515315  69.99     0
## 6 -0.2327938  0.1059148  0.253844225  0.08108026   3.67     0
```

In order to make sure that we don't have any missing values we check for missing values with the following line:

```
colSums(is.na(creditcard))
```

```
##   Time     V1     V2     V3     V4     V5     V6     V7     V8     V9    V10 
##      0      0      0      0      0      0      0      0      0      0      0 
##    V11    V12    V13    V14    V15    V16    V17    V18    V19    V20    V21 
##      0      0      0      0      0      0      0      0      0      0      0 
##    V22    V23    V24    V25    V26    V27    V28 Amount  Class 
##      0      0      0      0      0      0      0      0      0
```

Since all the columns are 0 we know that there is no missing values in our dataset.

Data Cleaning

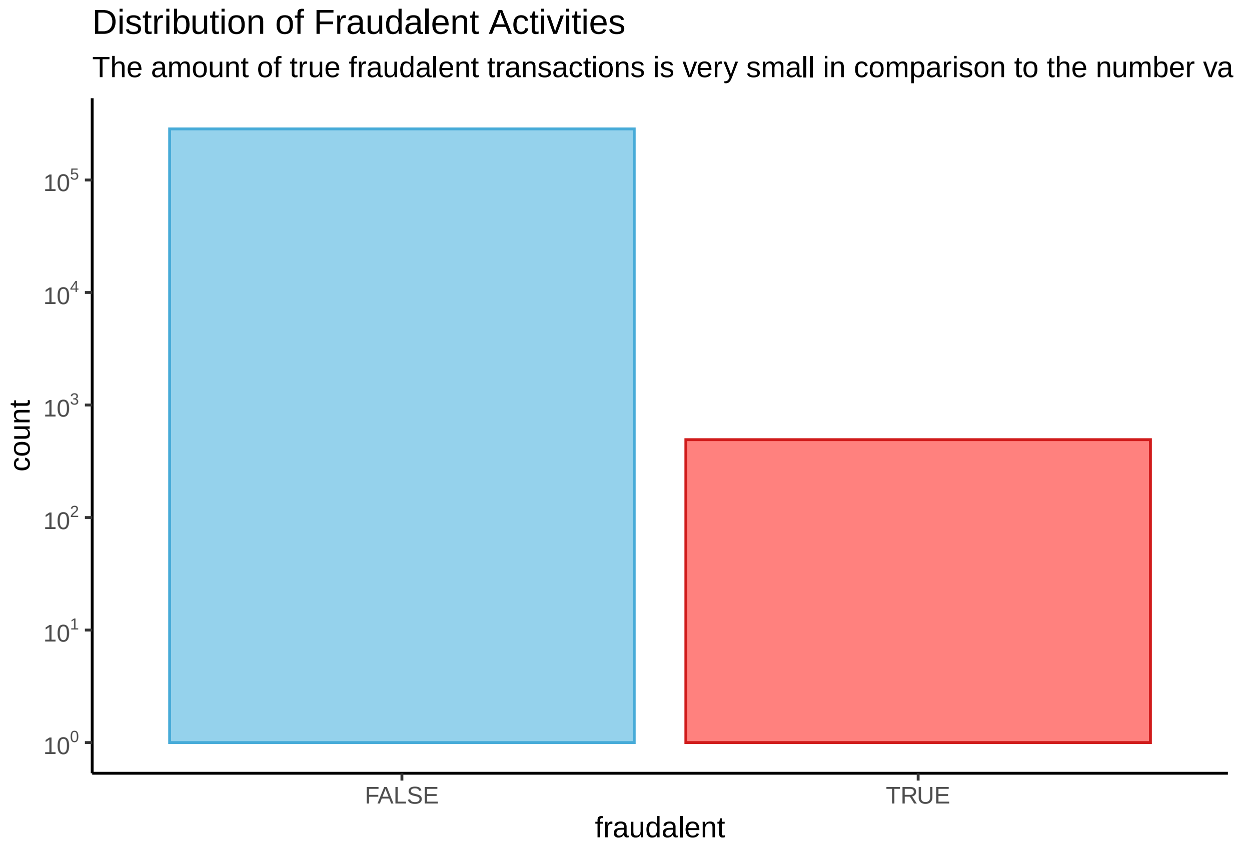
The Time feature contains the seconds elapsed between each transaction and the first transaction in the dataset. That means it is not so much an actual time but rather a kind of chronological order in which the transactions occur. Since this might not have any importance for our predictions we remove the time feature in a later step.

Dataanalysis and Visualization

In this section we will visualize interrelationships and facts about the dataset. We will visualize the ratio of fraudulent and legitimate transactions, correlations between the features and dependencies of the known features time and amount.

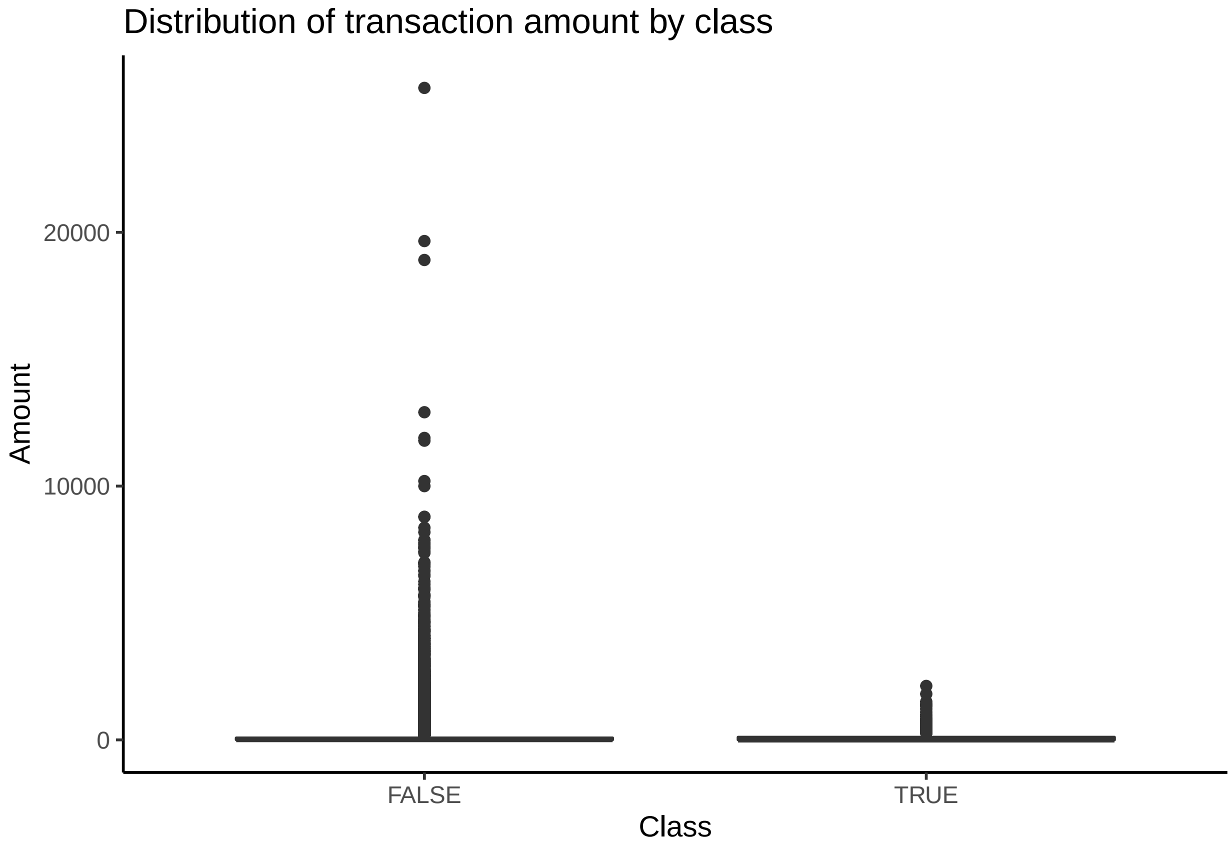
Imbalance

As already mentioned the dataset is very imbalanced. Only 0.017% of the transactions in the dataset are fraudulent. That means there is not so much training data for the used algorithms to learn from.



Amount of Transactions

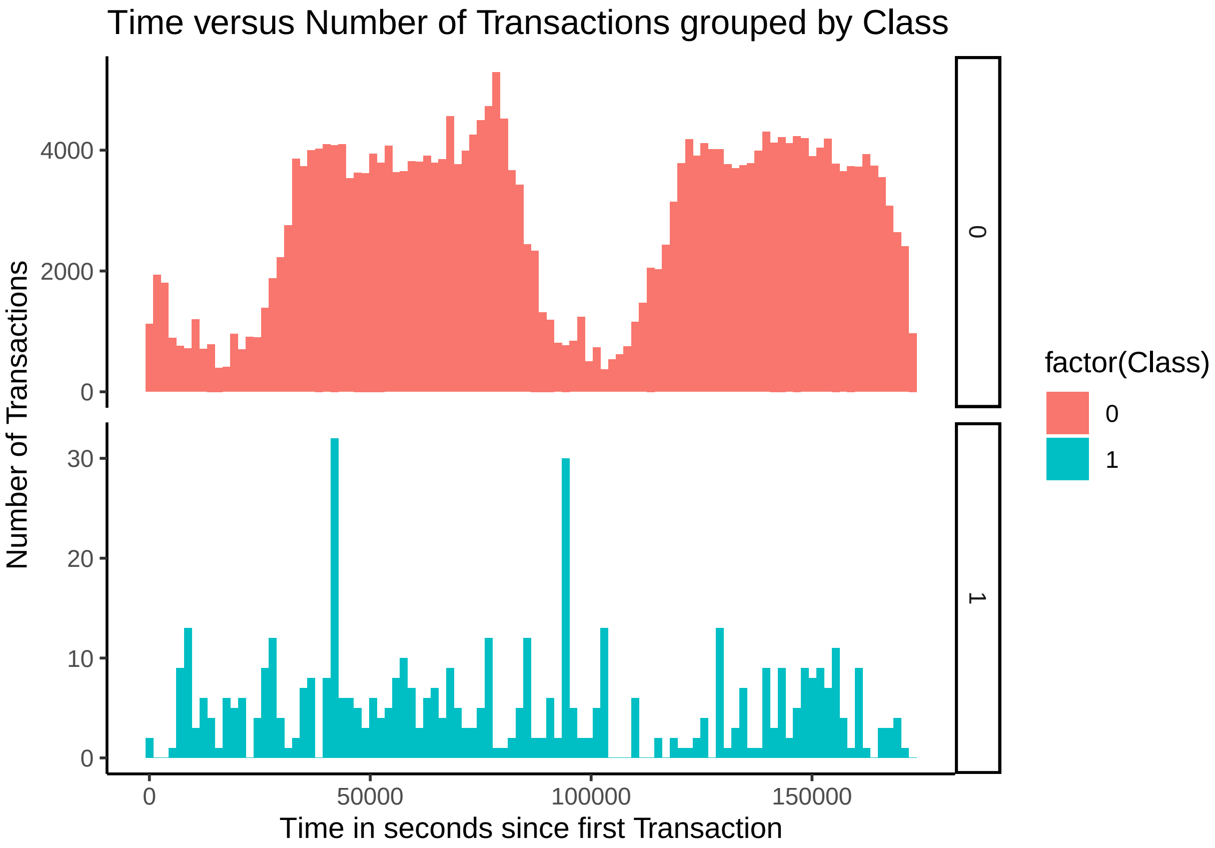
In the following plot we show the amount of money which is transferred with the transactions.The boolean expression FALSE represents legitimate transaction, while TRUE represents a fraudulent transaction.



We can see that for non-fraudulent transactions the distribution of transferred amounts of money is much larger compared to fraudulent transactions.

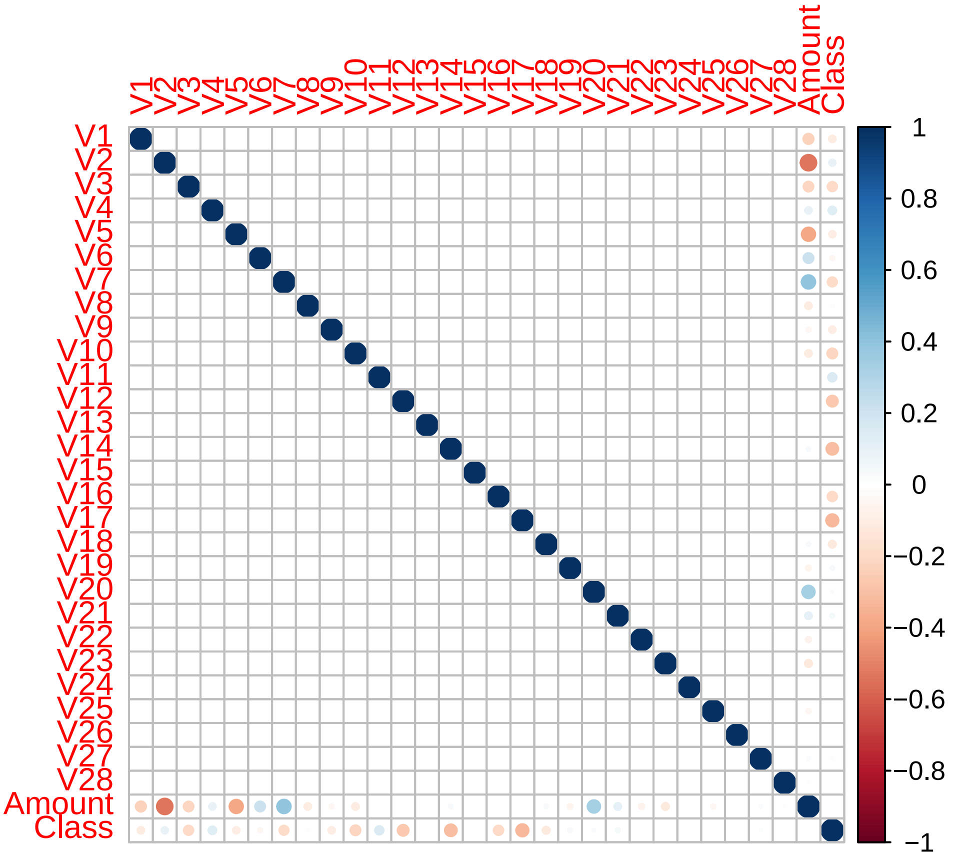
Time

The time feature is not an actual Date Time. It more represents the chronological order of transactions, since its the number of seconds that passed after the first transaction. With a plot of the number of transactions versus time we want to investigate if there is any anomalies. The class 0 represents non fraudulent transactions and the class 1 represents fraudulent transactions.



Correlations

In order to see if there is any features that have strong correlations we are going to plot all the correlations between every feature of the dataset.



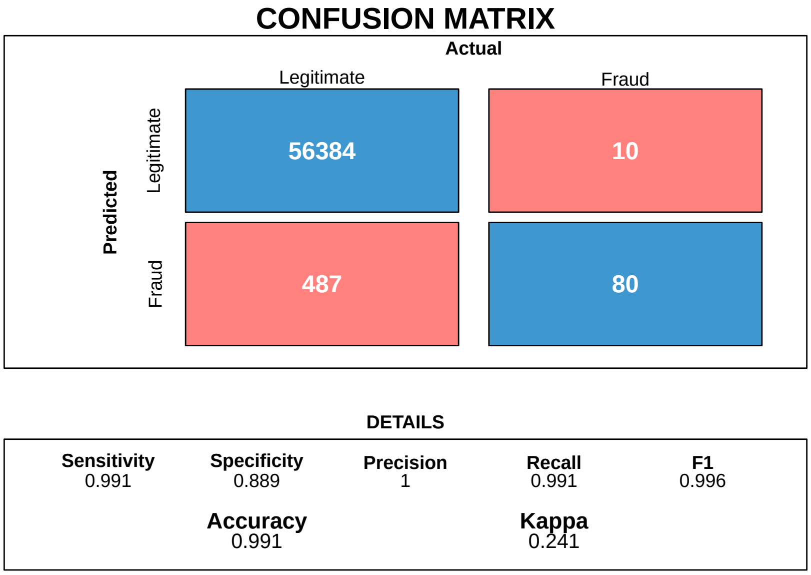
We can see that there is no correlations between the features V1-V28. This is the result of the PCA transformation of this hidden features. Some of these features have stronger or less strong correlations to the known features amount and class.

Models

The underlying task is a binary classification problem. We select three of The most popular models for binary classification and compare the results of these models. We will perform logistic regression,a random forest and a neuronal network. Since the original dataset is largely imbalanced, we will use an up sampling method for the training sets of all three algorithms. We will also perform a k-fold cross validation on the training sets in order to find the best settings for the algorithms.

Results

Logistic Regression



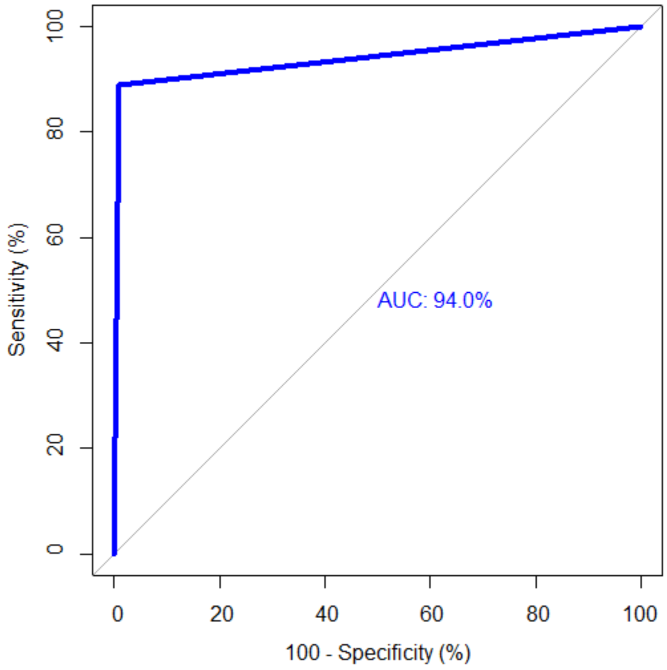
CONFUSION MATRIX

		Actual	
		Legitimate	Fraud
Predicted	Legitimate	56384	10
	Fraud	487	80

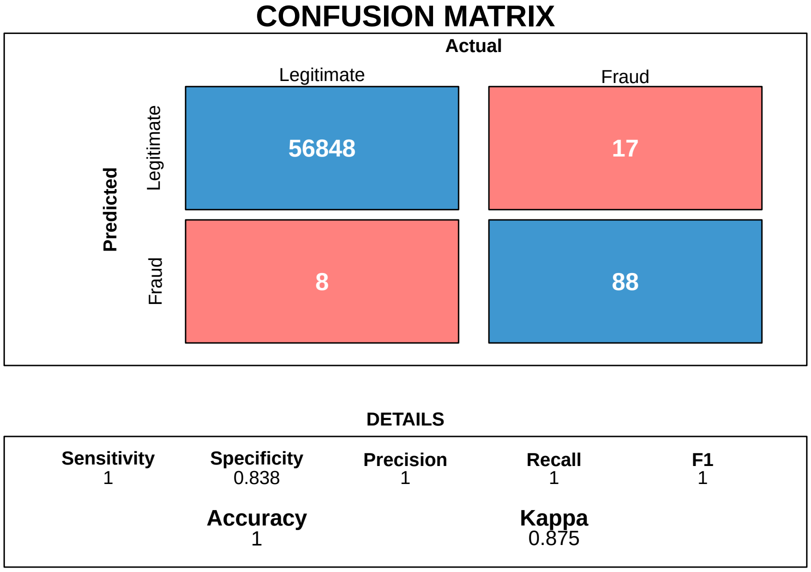
DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.991	0.889	1	0.991	0.996

Accuracy	Kappa
0.991	0.241



Random Forest



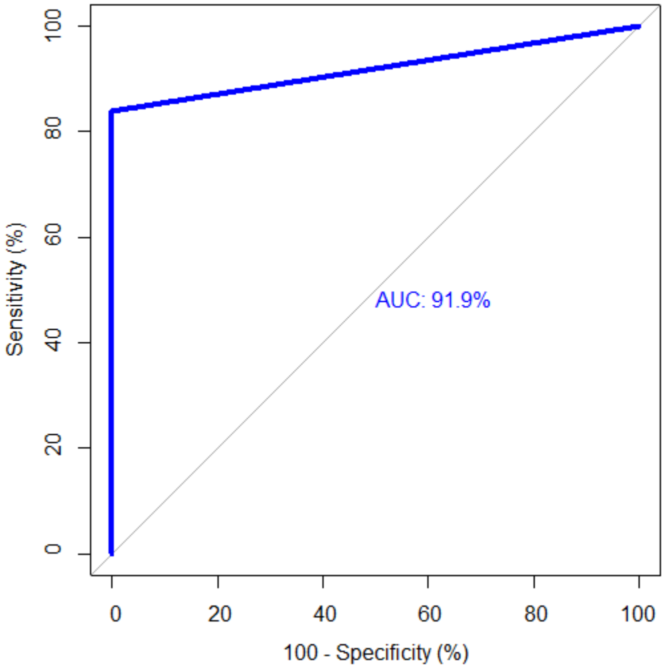
CONFUSION MATRIX

		Actual	
		Legitimate	Fraud
Predicted	Legitimate	56848	17
	Fraud	8	88

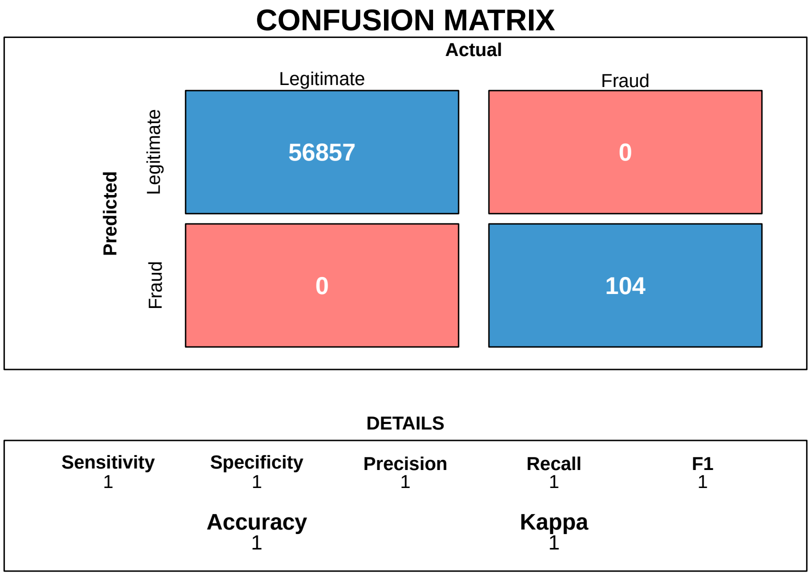
DETAILS

Sensitivity	Specificity	Precision	Recall	F1
1	0.838	1	1	1

Accuracy	Kappa
1	0.875



Neuronal Network



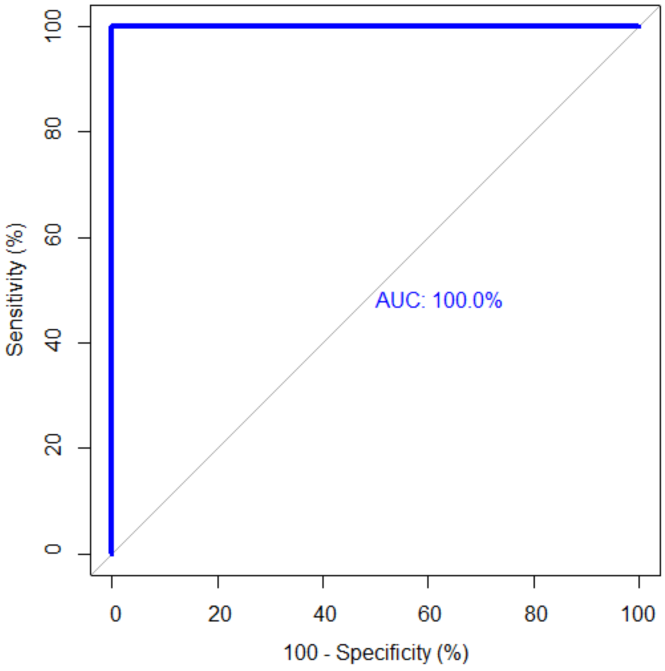
CONFUSION MATRIX

		Actual	
		Legitimate	Fraud
Predicted	Legitimate	56857	0
	Fraud	0	104

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
1	1	1	1	1

Accuracy	Kappa
1	1



Conclusion

Three different algorithms were used in order to classify if a transaction is legitimate or fraudulent. In order to overcome the negative influence of imbalance of the training data we performed Synthetic Minority Oversampling Technique on our training sets. All three methods gave really good results, however the neuronal network performs the best and gave perfect accuracy with 100% AUC. Logistic regression and the random forest model gave quite similar results with 94.0% AUC and 91.9% AUC. The sensitivity of the random forest is better than the one of logistic regression but the specificity of logistic regression is better than the one of the random forest. One might think its more important to focus on specificity since we want to detect fraudulent transactions but i guess both specificity and sensitivity are very important since every false positive and false negative prediction has an influence on costumer satisfaction. In case of logistic regression much more non-fraudulent transactions were classified as fraudulent compared to the random forest model. However the random forest takes much more time for computation. The neuronal network not only gave the best result it also has the shortest run time. So considering results and run time the neuronal network performed the best by far.

Limitations

Even if the results for the random forest and logistic regression have high ROC-AUC percentages I could imagine that its possible to achieve better results with better tuning. Especially the run time for the random forest made it hard to perform more tuning since a k-fold repeated cross validation on this algorithm takes several hours of computation.

The interpretation of the important variables for our predictions is simply not possible since we don't know any information about the features V1-V28.

References

1. Rafael A. Irizarry (2019), Introduction to Data Science: Data Analysis and Prediction Algorithms with R
2. https://cran.r-project.org/web/packages/smotefamily/smotefamily.pdf
3. https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf
4. https://cran.r-project.org/web/packages/caret/vignettes/caret.html