A PROJECT REPORT

ON

Detecting Credit Card Fraud Using Machine Learning

Submitted in partial fulfillment for the requirement of the award of

TRAINING IN

Data Analytics, Machine Learning and AI using Python

Submitted By

Gorle.Jeevanashish(bhilai institute of technology)

Under the guidance of

Mr. Bipul Shahi

Source: https://giphy.com/gifs/glitch-money-shopping-d3mmdNnW5hkoUxTG

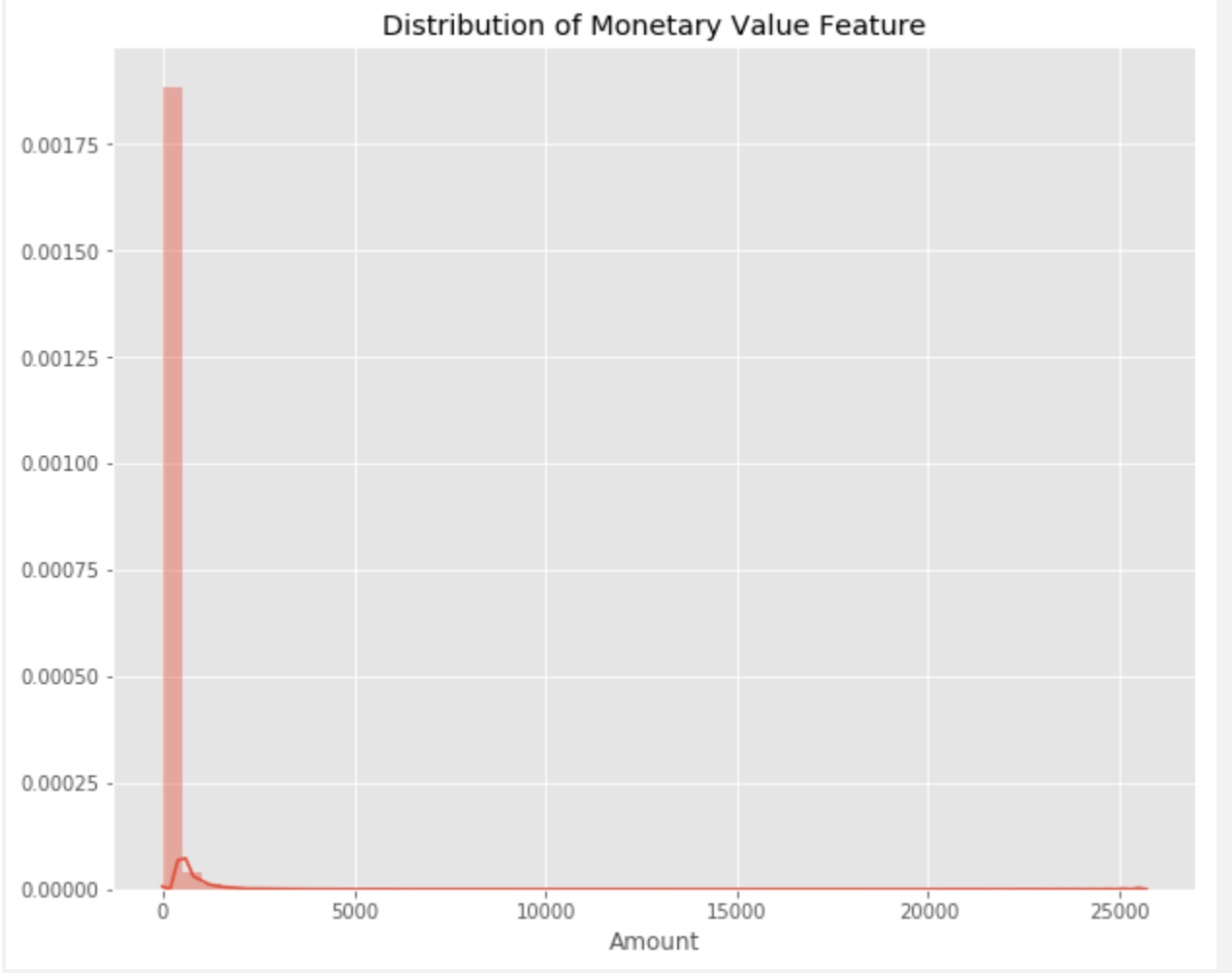
Introduction

Ever since starting my journey into data science, I have been thinking about ways to use data science for good while generating value at the same

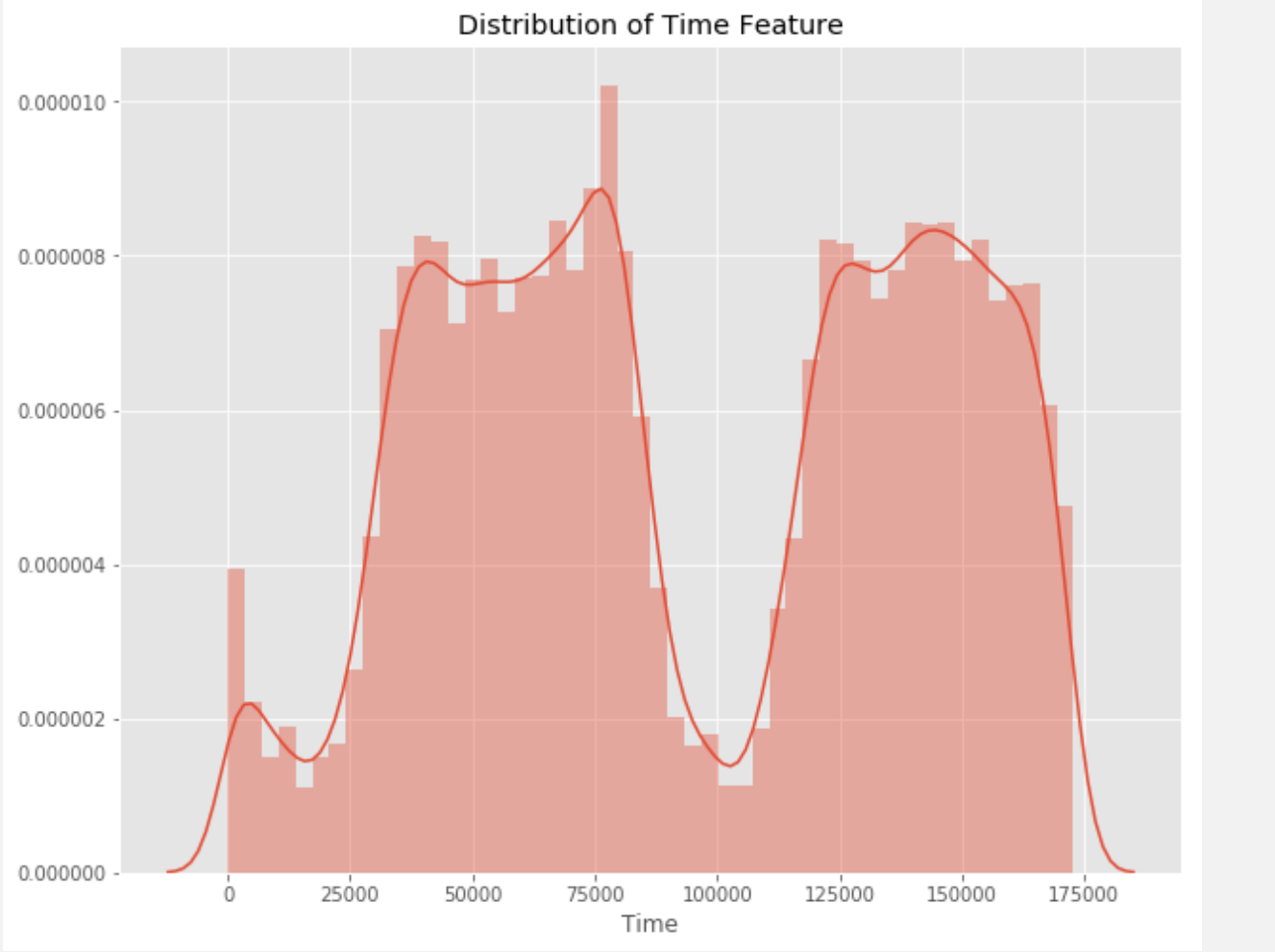
time. Thus, when I came across this data set on Kaggle dealing with credit card fraud detection, I was immediately hooked. The data set has 31 features, 28 of which have been anonymized and are labeled V1 through V28. The remaining three features are the time and the amount of the transaction as well as whether that transaction was fraudulent or not. Before it was uploaded to Kaggle, the anonymized variables had been modified in the form of a PCA (Principal Component Analysis). Furthermore, there were no missing values in the data set. Equipped with this basic description of the data, let's jump into some exploratory data analysis.

Exploratory Data Analysis (EDA)

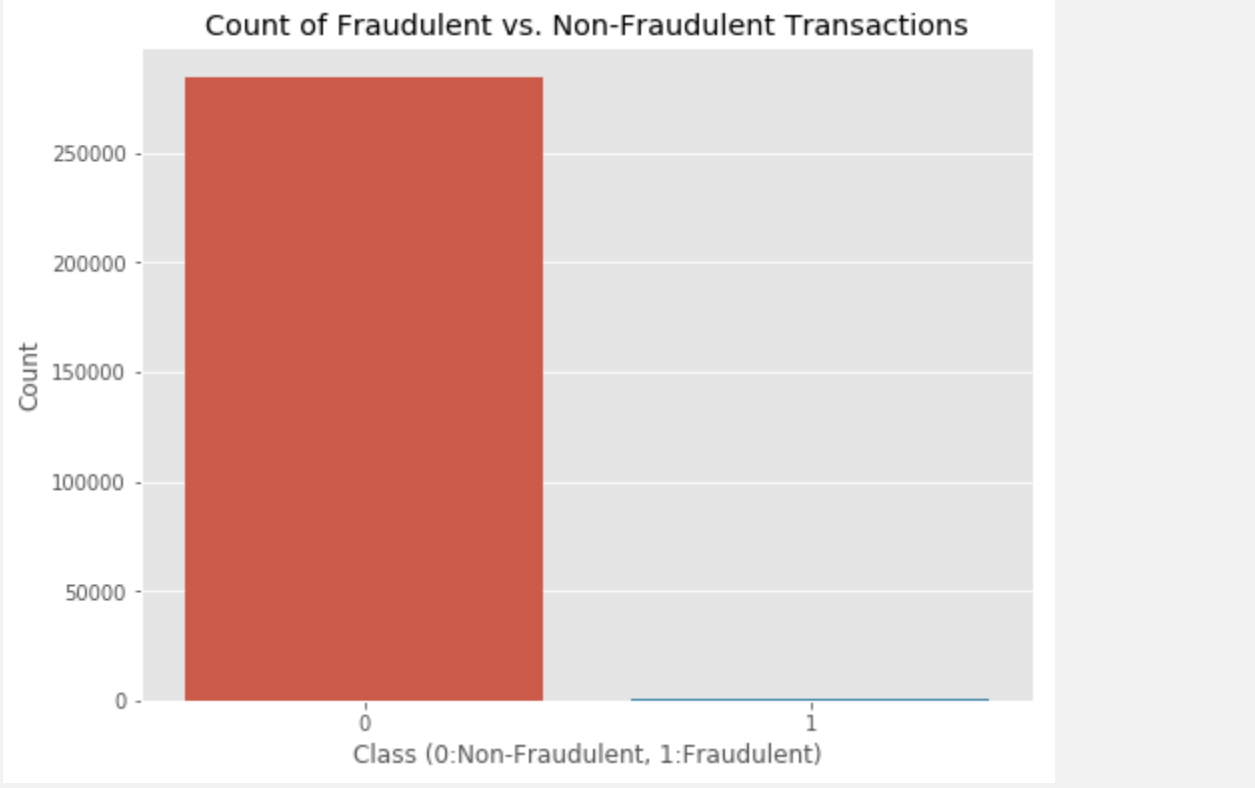
Since nearly all predictors have been anonymized, I decided to focus on the non-anonymized predictors time and amount of the transaction during my EDA. The data set contains 284,807 transactions. The mean value of all transactions is $88.35 while the largest transaction recorded in this data set amounts to $25,691.16. However, as you might be guessing right now based on the mean and maximum, the distribution of the monetary value of all transactions is heavily right-skewed. The vast majority of transactions are relatively small and only a tiny fraction of transactions comes even close to the maximum.



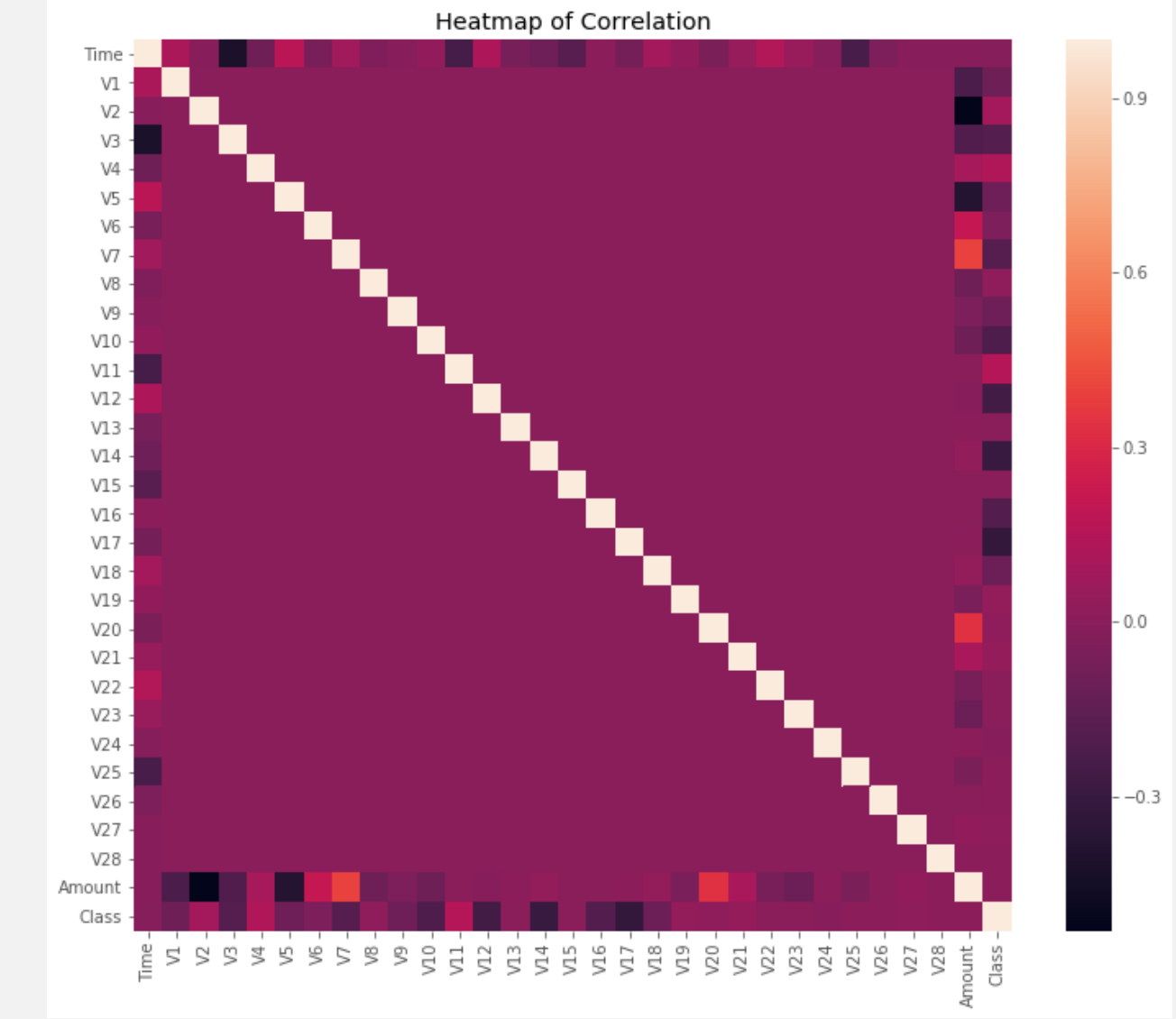
The time is recorded in the number of seconds since the first transaction in the data set. Therefore, we can conclude that this data set includes all transactions recorded over the course of two days. As opposed to the distribution of the monetary value of the transactions, it is bimodal. This indicates that approximately 28 hours after the first transaction there was a significant drop in the volume of transactions. While the time of the first transaction is not provided, it would be reasonable to assume that the drop in volume occurred during the night.



What about the class distributions? How many transactions are fraudulent and how many are not? Well, as can be expected, most transactions are non-fraudulent. In fact, 99.83% of the transactions in this data set were not fraudulent while only 0.17% were fraudulent. The following visualization underlines this significant contrast.



Finally, it would be interesting to know if there are any significant correlations between our predictors, especially with regards to our class variable. One of the most visually appealing ways to determine that is by using a heatmap.



As you can see, some of our predictors do seem to be correlated with the class variable. Nonetheless, there seem to be relatively little significant correlations for such a big number of variables. This can probably be attributed to two factors:

1. The data was prepared using a PCA, therefore our predictors are principal components.

2. The huge class imbalance might distort the importance of certain correlations with regards to our class variable.

Data Preparation

Before continuing with our analysis, it is important not to forget that while the anonymized features have been scaled and seem to be centered around zero, our time and amount features have not. Not scaling them as well would result in certain machine learning algorithms that give weights to features (logistic regression) or rely on a distance measure (KNN) performing much worse. To avoid this issue, I standardized both the time and amount column. Luckily, there are no missing values and we, therefore, do not need to worry about missing value imputation.

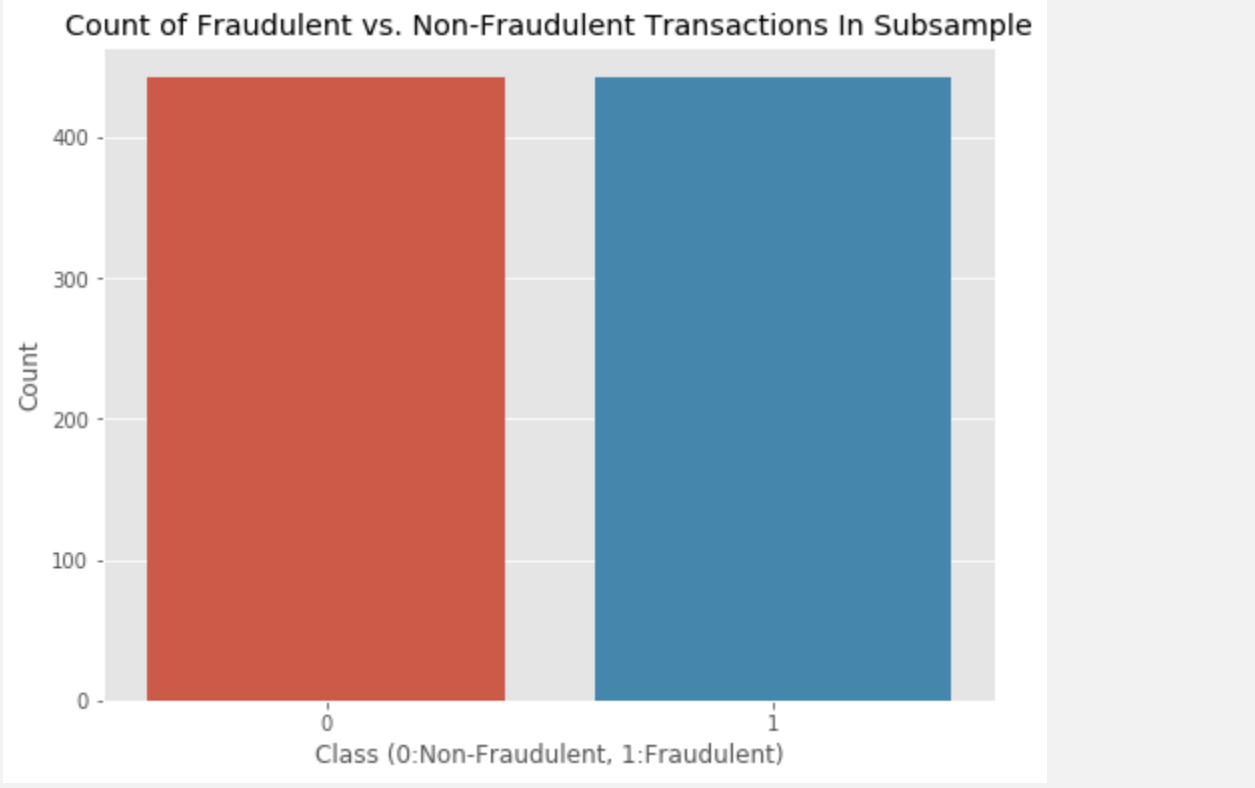
Creating a Training Set for a Heavily Imbalanced Data Set

Now comes the challenging part: Creating a training data set that will allow our algorithms to pick up the specific characteristics that make a transaction more or less likely to be fraudulent. Using the original data set would not prove to be a good idea for a very simple reason: Since over 99% of our transactions are non-fraudulent, an algorithm that always predicts that the transaction is non-fraudulent would achieve an accuracy higher than 99%. Nevertheless, that is the opposite of what we want. We do not

want a 99% accuracy that is achieved by never labeling a transaction as fraudulent, we want to detect fraudulent transactions and label them as such.

There are two key points to focus on to help us solve this. First, we are going to utilize **random under-sampling** to create a training dataset with a balanced class distribution that will force the algorithms to detect fraudulent transactions as such to achieve high performance. Speaking of performance, we are not going to rely on accuracy. Instead, we are going to make use of the Receiver Operating Characteristics-Area Under the Curve or ROC-AUC performance measure (I have linked further reading below this article). Essentially, the ROC-AUC outputs a value between zero and one, whereby one is a perfect score and zero the worst. If an algorithm has a ROC-AUC score of above 0.5, it is achieving a higher performance than random guessing.

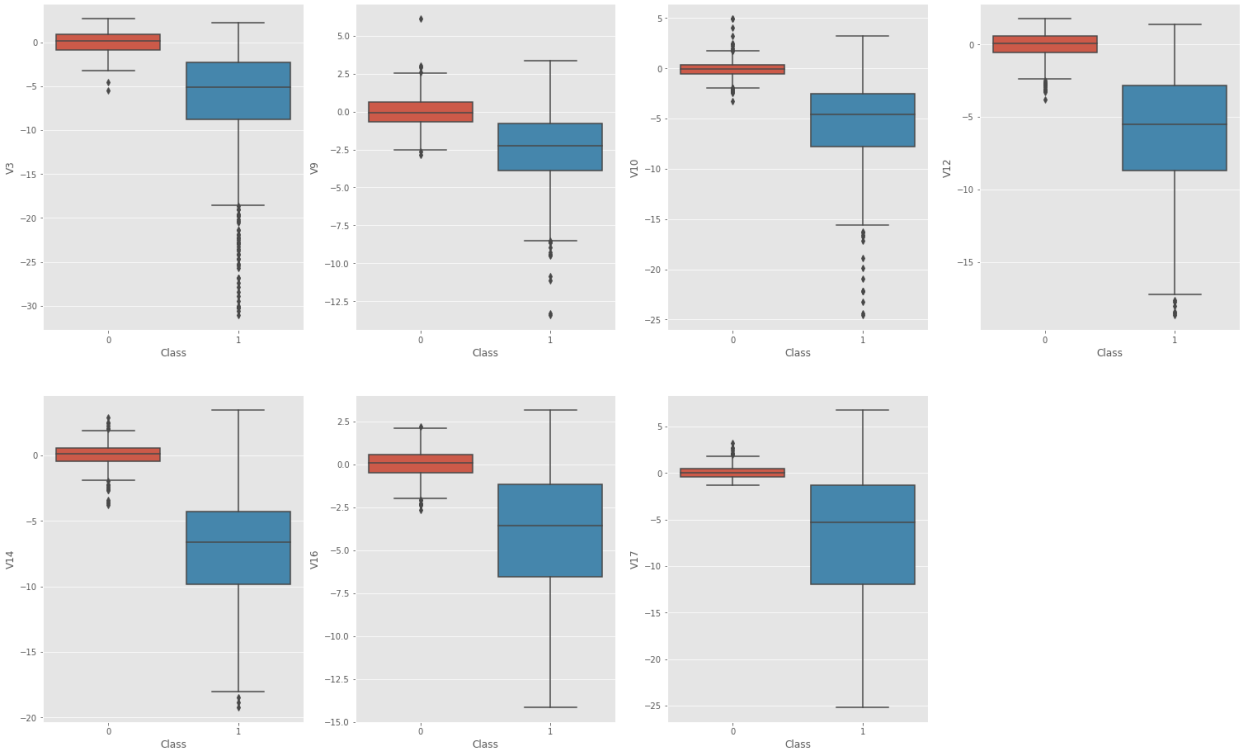
To create our balanced training data set, I took all of the fraudulent transactions in our data set and counted them. Then, I randomly selected the same number of non-fraudulent transactions and concatenated the two. After shuffling this newly created data set, I decided to output the class distributions once more to visualize the difference.



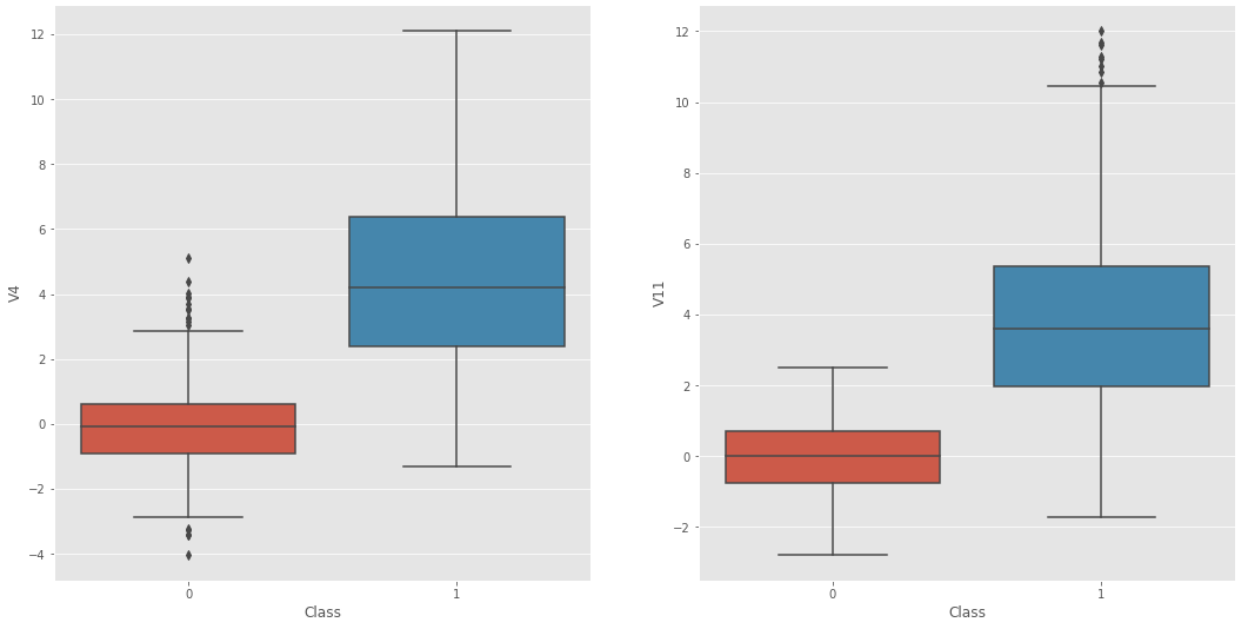
Outlier Detection & Removal

Outlier detection is a complex topic. The trade-off between reducing the number of transactions and thus volume of information available to my algorithms and having extreme outliers skew the results of your predictions is not easily solvable and highly depends on your data and goals. In my case, I decided to focus exclusively on features with a correlation of 0.5 or higher with the class variable for outlier removal. Before getting into the actual outlier removal, let's take a look at visualizations of those features:

Features With High Negative Correlation



Features With High Positive Correlation



Box plots provide us with a good intuition of whether we need to worry about outliers as all transactions outside of 1.5 times the IQR (Inter-Quartile Range) are usually considered to be outliers. However, removing all transactions outside of 1.5 times the IQR would dramatically decrease our training data size, which is not very large, to begin with. Thus, I decided to only focus on extreme outliers outside of 2.5 times the IQR.

Dimensionality Reduction With t-SNE Visualization

Visualizing our classes would prove to be quite interesting and show us if they are clearly separable. However, it is not possible to produce a 30-dimensional plot using all of our predictors. Instead, using a dimensionality reduction technique such as t-SNE, we are able to project these higher dimensional distributions into lower-dimensional visualizations. For this project, I decided to use t-SNE, an algorithm that I had not been working with before. If you would like to know more about how this algorithm works, see

Projecting our data set into a two-dimensional space, we are able to produce a scatter plot showing the clusters of fraudulent and non-fraudulent transactions:

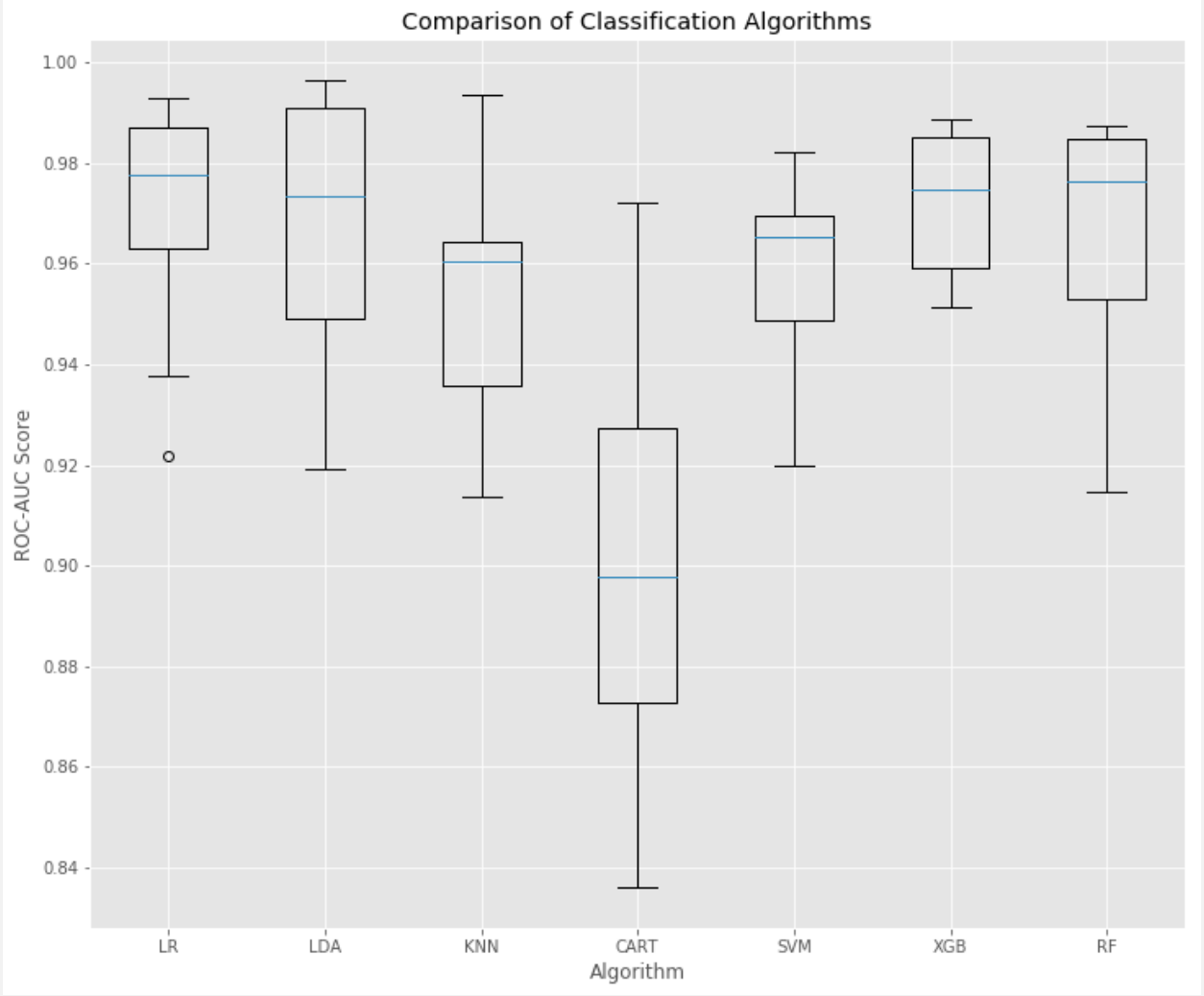
Classifications Algorithms

Onto the part you've probably been waiting for all this time: training machine learning algorithms. To be able to test the performance of our algorithms, I first performed an 80/20 train-test split, splitting our balanced data set into two pieces. To avoid overfitting, I used the very common resampling technique of k-fold cross-validation. This simply means that you separate your training data into k parts (folds) and then fit your model on k-1 folds before making predictions for the kth hold-out fold. You then repeat this process for every single fold and average the resulting predictions.

To get a better feeling of which algorithm would perform best on our data, let's quickly spot-check some of the most popular classification algorithms:

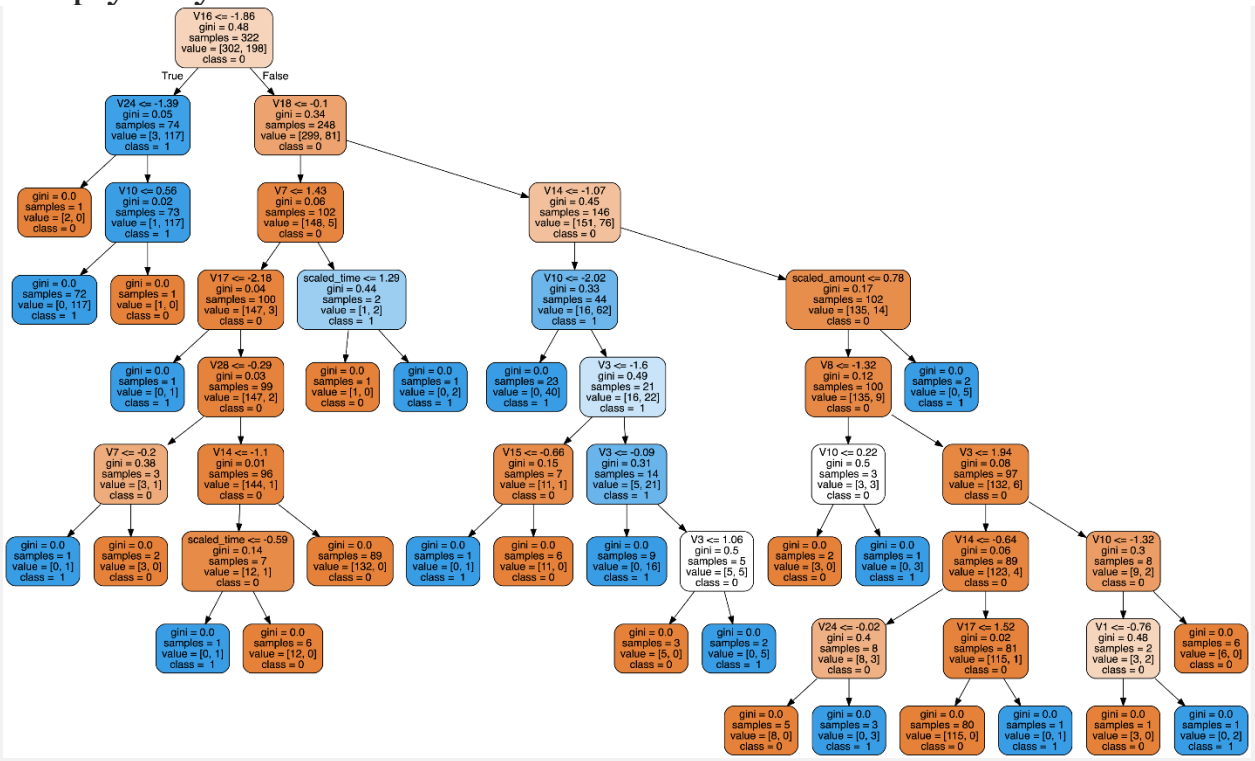
- Logistic Regression
- Linear Discriminant Analysis
- K Nearest Neighbors (KNN)
- Classification Trees
- Support Vector Classifier
- Random Forest Classifier
- XGBoost Classifier

The results of this spot-checking can be visualized as follows:



As we can see, there are a few algorithms that quite significantly outperformed the others. Now, what algorithm do we choose? As mentioned above, this project had not only the focus of achieving the highest accuracy but also to create business value. Therefore, choosing Random Forest over XGBoost might be a reasonable approach in order to achieve a higher degree of comprehensiveness while only slightly decreasing performance. To further illustrate what I mean by this, here is a visualization of our

Random Forest model that could easily be used to explain very simply why a certain decision was made:



Conclusion & Future Work

Fraud detection is a complex issue that requires a substantial amount of planning before throwing machine learning algorithms at it. Nonetheless, it is also an application of data science and machine learning for the good, which makes sure that the customer's money is safe and not easily tampered with.

Future work will include a comprehensive tuning of the Random Forest algorithm I talked about earlier. Having a data set with non-anonymized features would make this particularly interesting as outputting the feature importance would enable one to see what

specific factors are most important for detecting fraudulent transactions.

As always, if you have any questions or found mistakes, please do not hesitate to reach out to me. A link to the notebook with my code is provided at the beginning of this article.

References:

[1] L.J.P. van der Maaten and G.E. Hinton, Visualizing High-Dimensional Data Using t-SNE (2014), Journal of Machine Learning Research

[2] Machine Learning Group — ULB, Credit Card Fraud Detection (2018), Kaggle

[3] Nathalie Japkowicz, Learning from Imbalanced Data Sets: A Comparison of Various Strategies (2000), AAAI Technical Report WS-00–05