Credit Card Fraud Detection Using Machine Learning With Python

Experiment Findings · November 2019

DOI: 10.13140/RG.2.2.22851.35361/1

Tamojit Das
Amity University Kolkata
6 PUBLICATIONS **10** CITATIONS

Credit Card Fraud Detection
Using Machine Learning with Python

Name: **TAMOJIT DAS**

DECLARATIONS

I do hereby declare that

a. The work contained in this report is original and has been done by me under the guidance of my supervisors.
b. The work has not been submitted to any other Institute for any degree or diploma as per best of my knowledge.
c. The guidelines provided by the Institute in preparing the thesis have been followed.
d. The norms and guidelines given in the Ethical Code of Conduct of the Institute have adhered to.
e. Wherever materials (data, theoretical analysis, figures and text) used from other sources, due credit has been given by citing them in the text of the thesis and giving their details in the reference.

(Student's Signature)
Name: **TAMOJIT DAS**

CERTIFICATE

Date:29.04.2019

This is to certify that the thesis entitled "**Credit Card Fraud Detection using Machine Learning with Python**" submitted by **TAMOJIT DAS** to Amity University, Kolkata, West Bengal, India, is a record of original research work carried under my supervision and is worthy of consideration for the award of the degree of **Bachelor of Technology (Computer Science & Engineering)** from the University.

Your Guide's Name Prof. Anjan Bandyopadhyay
Assistant Professor,
Department of CSE, ASETK,
Amity University, Kolkata,
West Bengal, India

Prof. (Dr.) Neeraj Sharma,
Head of Department,
Department of CSE, ASETK,
Amity University, Kolkata,
West Bengal, India

Dr. Birajashish Pattanaik,
Head of Institution,
Amity School of Engineering and Technology,
Amity University, Kolkata,
West Bengal, India

Acknowledgements

I would like to express my profound and sincere gratitude to my faculty guide and project guide Prof. Anjan Bandyopadhyay for guidance and support, who helped me to finish this project. His suggestion and instructions have served as the proper guide towards completion of this project.

I would also like to extend my gratitude to the HOD Prof. (Dr.) Neeraj Sharma and

HOI Dr. Birajashish Pattanaik for providing me with all the facility that was required.

Project Acknowledgement

The dataset has been collected and analyzed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (University Libre de Brucellas) on big data mining and fraud detection. More details on current and past projects on related topics are available on https://www.researchgate.net/project/Fraud-detection-5 and the page of the Defeat Fraud project

Tamojit Das

Abstract

Now a day's online transactions have become an important and necessary part of our lives. It is vital that credit card companies are able to identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase. As frequency of transactions is increasing, number of fraudulent transactions are also increasing rapidly. Such problems can be tackled with Machine Learning with its algorithms. This project intends to illustrate the modelling of a data set using machine learning with Credit Card Fraud Detection. The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the data of the ones that turned out to be fraud. This model is then used to recognize whether a new transaction is fraudulent or not. Our objective here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications. Credit Card Fraud Detection is a typical sample of classification. In this process, we have focused on analyzing and pre-processing data sets as well as the deployment of multiple anomaly detection algorithms such as Local Outlier Factor and Isolation Forest algorithm on the PCA transformed Credit Card Transaction data.

Contents

Tamojit Das -tamo.das.97@gmail.com

Introduction

Credit Card Fraud can be defined as a case where a person uses someone else's credit card for personal reasons while the owner and the card issuing authorities are unaware of the fact that the card is being used.

Due to rise and acceleration of E- Commerce, there has been a tremendous use of credit cards for online shopping which led to High amount of frauds related to credit cards. In the era of digitalization, the need to identify credit card frauds is necessary. Fraud detection involves monitoring and analyzing the behavior of various users in order to estimate detect or avoid undesirable behavior. In order to identify credit card fraud detection effectively, we need to understand the various technologies, algorithms and types involved in detecting credit card frauds. Algorithm can differentiate transactions which are fraudulent or not. Find fraud, they need to passed dataset and knowledge of fraudulent transaction. They analyze the dataset and classify all transactions.

Fraud detection involves monitoring the activities of populations of users in order to estimate, perceive or avoid objectionable behavior, which consist of fraud, intrusion, and defaulting.

Machine learning algorithms are employed to analyses all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent.

The investigators provide a feedback to the automated system which is used to train and update the algorithm to eventually improve the fraud-detection performance over time.

Literature Survey

S P Maniraj [1] In this paper, they describe Random forest algorithm applicable on Find fraud detection. Random forest has two types. They describe in detail and their accuracy 91.96% and 96.77% respectively. This paper summaries second type is better than the first type.

Suman Arora [2] In this paper, many supervised machine learning algorithms apply on 70% training and 30% testing dataset. Random forest, stacking classifier, XGB classifier, SVM, Decision tree and KNN algorithms compare each other i.e. 94.59%, 95.27%, 94.59%, 93.24%, 90.87%, 90.54% and 94.25% respectively. Summaries of this paper, SVM has the highest ranking with 0.5360 FPR, and stacking classifier has the lowest ranking with 0.0335.

Kosemani Temitayo Hafiz [3] In this paper, they describe flow chart of fraud detection process. i.e. data Acquisition, data pre-processing, Exploratory data analysis and methods or algorithms are in detail. Algorithms are K- nearest neighbor (KNN), random tree and Logistic regression accuracy are 96.91%, 94.32%, 57.73% and 98.24% respectively.

Methodology

The approach that this paper proposes, uses the latest machine learning algorithms to detect anomalous activities, called outliers.

The basic rough architecture diagram can be represented with the following figure:

When looked at in detail on a larger scale along with real life elements, the full architecture diagram can be represented as follows:

First of all, we obtained our dataset from Kaggle, a data analysis website which provides datasets.

Inside this dataset, there are 31 columns out of which 28 are named as v1-v28 to protect sensitive data.

The other columns represent Time, Amount and Class. Time shows the time gap between the first transaction and the following one. Amount is the amount of money transacted. Class 0 represents a valid transaction and 1 represents a fraudulent one.

We plot different graphs to check for inconsistencies in the dataset and to visually comprehend it:

This graph shows that the number of fraudulent transactions is much lower than the legitimate ones.

This graph shows the times at which transactions were done within two days. It can be seen that the least number of transactions were made during night time and highest during the days.

This graph represents the amount that was transacted. A majority of transactions are relatively small and only a handful of them come close to the maximum transacted amount.

After checking this dataset, we plot a histogram for every column. This is done to get a graphical representation of the dataset which can be used to verify that there are no missing any values in the dataset. This is done to ensure that we don't require any missing value imputation and the machine learning algorithms can process the dataset smoothly.

After this analysis, we plot a heatmap to get a colored representation of the data and to study the correlation between out predicting variables and the class variable. This heatmap is shown below:

The dataset is now formatted and processed. The time and amount column are standardized and the Class column is removed to ensure fairness of evaluation. The data is processed by a set of algorithms from modules. The following module diagram explains how these algorithms work together: This data is fit into a model and the following outlier detection modules are applied on it:

- Local Outlier Factor
- Isolation Forest Algorithm

These algorithms are a part of sklearn. The ensemble module in the sklearn package includes ensemble-based methods and functions for the classification, regression and outlier detection.

This free and open-source Python library is built using NumPy, SciPy and matplotlib modules which provides a lot of simple and efficient tools which can be used for data analysis

and machine learning. It features various classification, clustering and regression algorithms and is designed to interoperate with the numerical and scientific libraries.

Wave used Jupyter Notebook platform to make a program in Python to demonstrate the approach that this paper suggests. This program can also be executed on the cloud using Google Collab platform which supports all python notebook files.

Detailed explanations about the modules with pseudocodes for their algorithms and output graphs are given as follows:

1. Local Outlier Factor

   It is an Unsupervised Outlier Detection algorithm. 'Local Outlier Factor' refers to the anomaly score of each sample. It measures the local deviation of the sample data with respect to its neighbors.

More precisely, locality is given by k-nearest neighbors, whose distance is used to estimate the local data.

The pseudocode for this algorithm is written as:

On plotting the results of Local Outlier Factor algorithm, we get the following figure:

By comparing the local values of a sample to that of its neighbors, one can identify samples that are substantially lower than their neighbors. These values are quite amanous and they are considered as outliers.

As the dataset is very large, we used only a fraction of it in out tests to reduce processing times.

The final result with the complete dataset processed is also determined and is given in the results section of this paper.

2. Isolation Forest Algorithm

The Isolation Forest isolates observations by arbitrarily selecting a feature and then randomly selecting a split value between the maximum and minimum values of the designated feature.

Recursive partitioning can be represented by a tree, the number of splits required to isolate a sample is equivalent to the path length root node to terminating node.

The average of this path length gives a measure of normality and the decision function which we use.

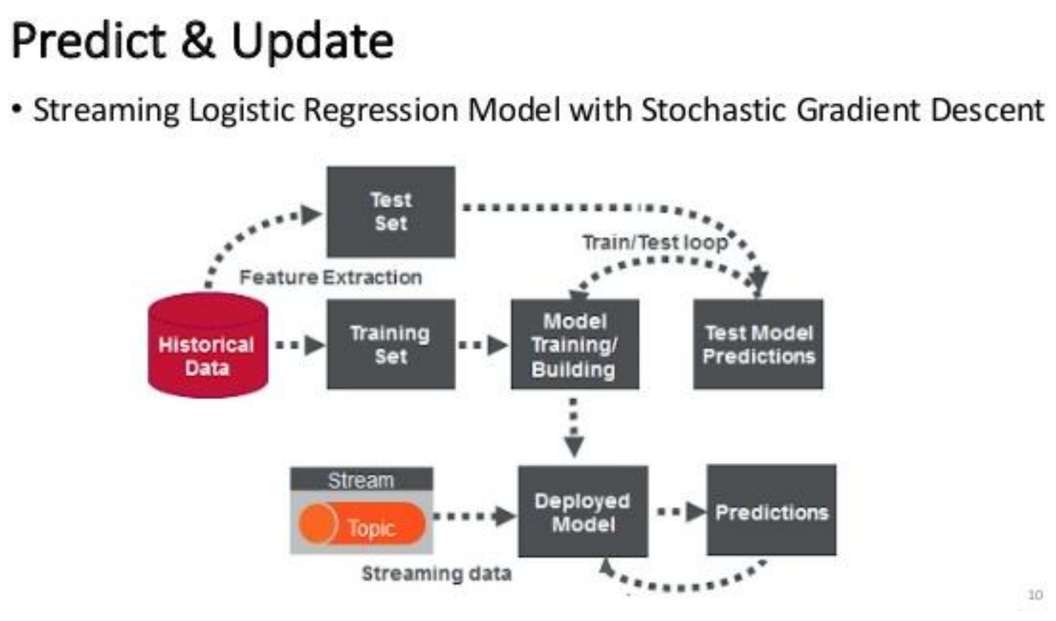
The pseudocode for this algorithm can be written as:

On plotting the results of Isolation Forest algorithm, we get the following figure:

Partitioning them randomly produces shorter paths for anomalies. When a forest of random trees mutually produces shorter path lengths for specific samples, they are extremely likely to be anomalies.

Once the anomalies are detected, the system can be used to report them to the concerned authorities. For testing purposes, we are comparing the outputs of these algorithms to determine their accuracy and precision.

Flow Diagram



Predict & Update

- Streaming Logistic Regression Model with Stochastic Gradient Descent

Tamojit Das -tamo.das.97@gmail.com

Implementation

About dataset

The datasets contain transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Machine Learning-Based Approaches

Below is a brief overview of popular machine learning-based techniques for anomaly detection.

a. Density-Based Anomaly Detection

Density-based anomaly detection is based on the k-nearest neighbors' algorithm.

Assumption: Normal data points occur around a dense neighborhood and abnormalities are far away.

The nearest set of data points are evaluated using a score, which could be Euclidian distance or a similar measure dependent on the type of the data (categorical or numerical). They could be broadly classified into two algorithms:
K-nearest neighbor: k-NN is a simple, non-parametric lazy learning technique used to classify data based on similarities in distance metrics such as Euclidian, Manhattan, Minkowski, or Hamming distance.
Relative density of data: This is better known as local outlier factor (LOF). This concept is based on a distance metric called reachability distance.

b. Clustering-Based Anomaly Detection

Clustering is one of the most popular concepts in the domain of unsupervised learning.

Assumption: Data points that are similar tend to belong to similar groups or clusters, as determined by their distance from local centroids.
K-means is a widely used clustering algorithm. It creates 'k' similar clusters of data points. Data instances that fall outside of these groups could potentially be marked as anomalies.

c. Support Vector Machine-Based Anomaly Detection

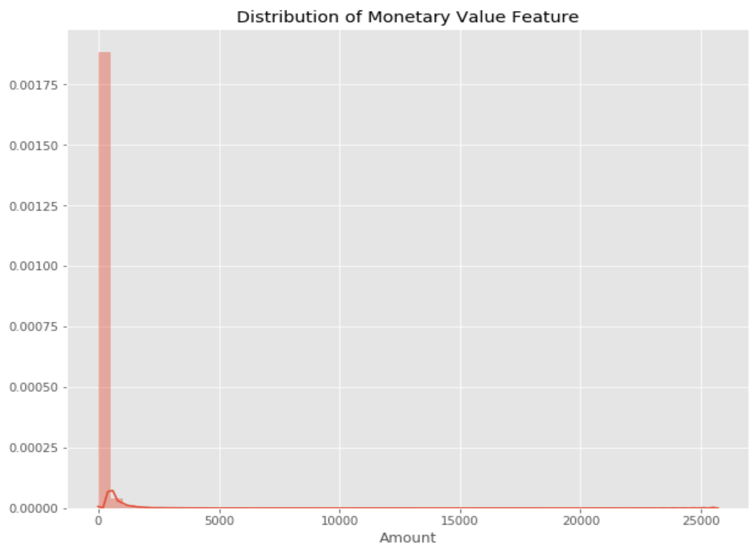
- A support vector machine is another effective technique for detecting anomalies.
- A SVM is typically associated with supervised learning, but there are extensions (OneClassCVM, for instance) that can be used to identify anomalies as an unsupervised problem (in which training data are not labeled).
- The algorithm learns a soft boundary in order to cluster the normal data instances using the training set, and then, using the testing instance, it tunes itself to identify the abnormalities that fall outside the learned region.
- Depending on the use case, the output of an anomaly detector could be numeric scalar values for filtering on domain-specific thresholds or textual labels (such as binary/multi labels).

In this jupyter notebook we are going to take the credit card fraud detection as the case study for understanding this concept in detail using the following Anomaly Detection Techniques namely

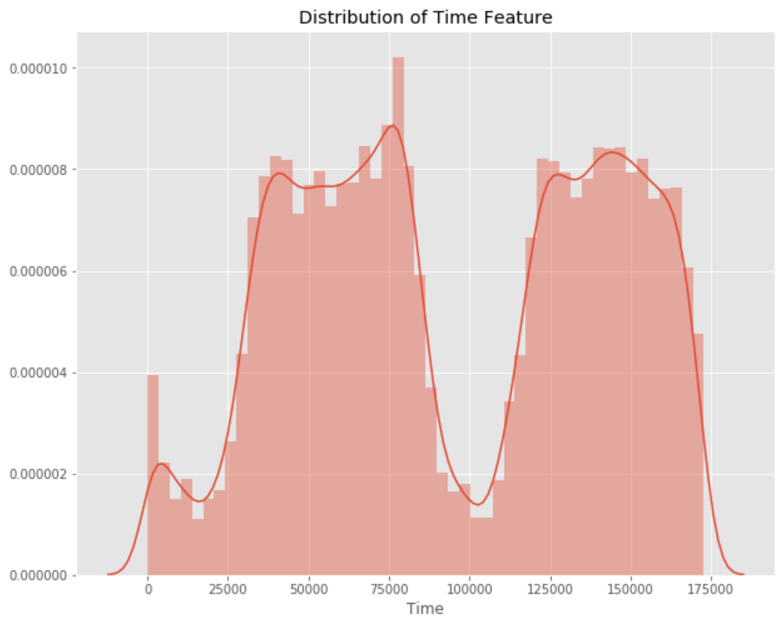
- **Isolation Forest Anomaly Detection Algorithm.**
- **Density-Based Anomaly Detection (Local Outlier Factor) Algorithm.**
- **Support Vector Machine Anomaly Detection Algorithm**

Exploratory Data Analysis (EDA)

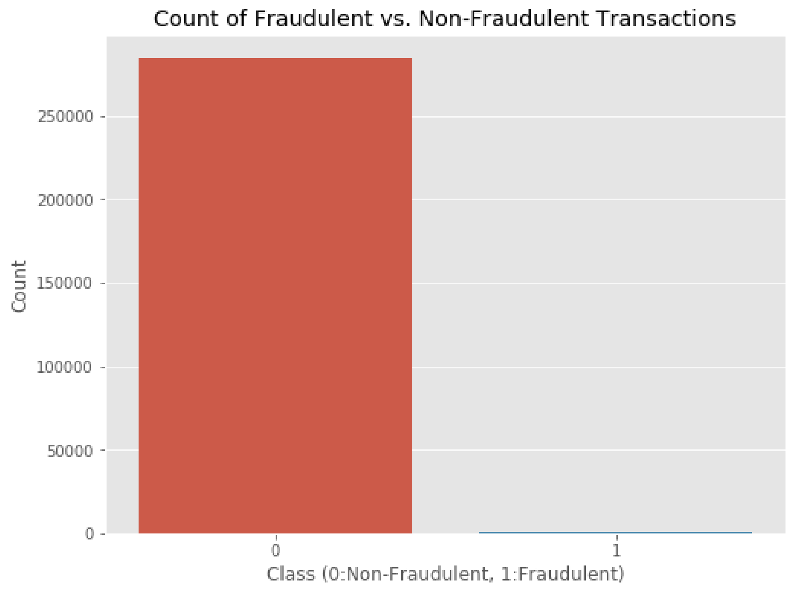
Since nearly all predictors have been anonymized, I decided to focus on the non-anonymized predictors time and amount of the transaction during my EDA. The data set contains 284,807 transactions. The mean value of all transactions is $88.35 while the largest transaction recorded in this data set amounts 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.



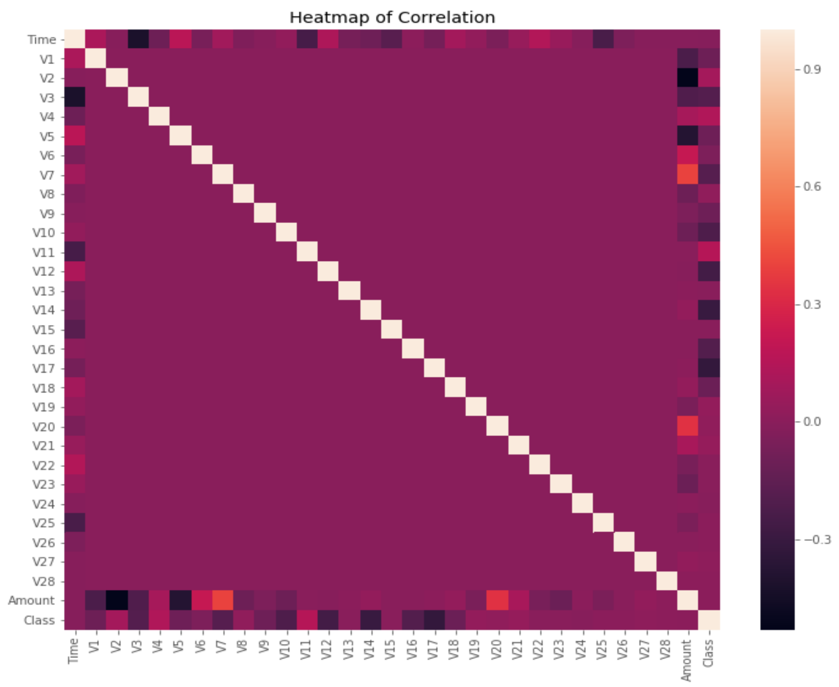
The time is recorded in the number of seconds since the first transaction in the data set. Therefore, we can conclude that this data set includes all transactions recorded over the course of two days. As opposed to the distribution of the monetary value of the transactions, it is bimodal. This indicates 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

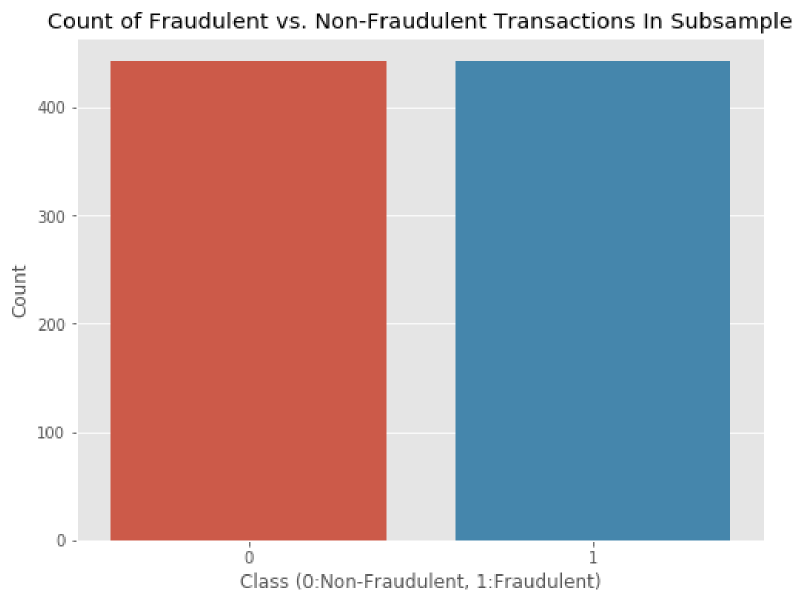
Before continuing with our analysis, it is important not to forget that while the anonymized features have been scaled and seem to be centered around zero, our time and amount features have not. Not scaling them as well would result in certain machine learning algorithms that give weights to features (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.

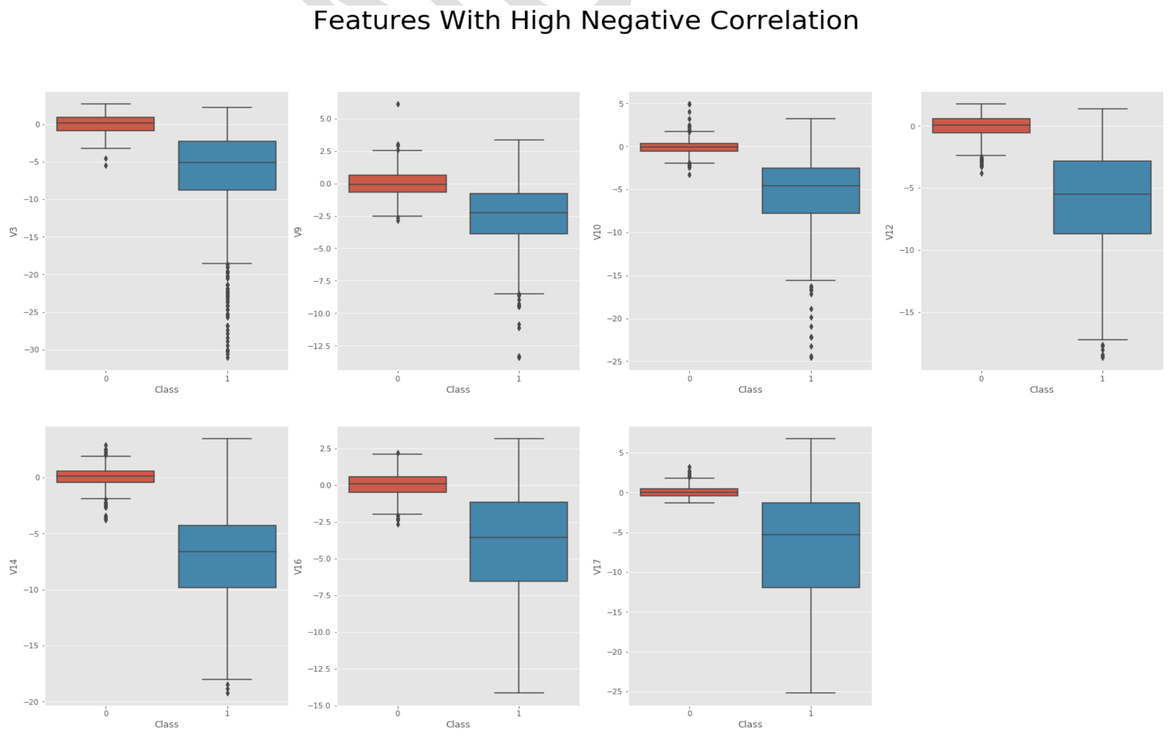
There are two key points to focus on to help us solve this. First, we are going to utilize **random under-sampling** to create a training dataset with a balanced class distribution that will force the algorithms to detect fraudulent transactions as such to achieve high performance. Speaking of performance, 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.

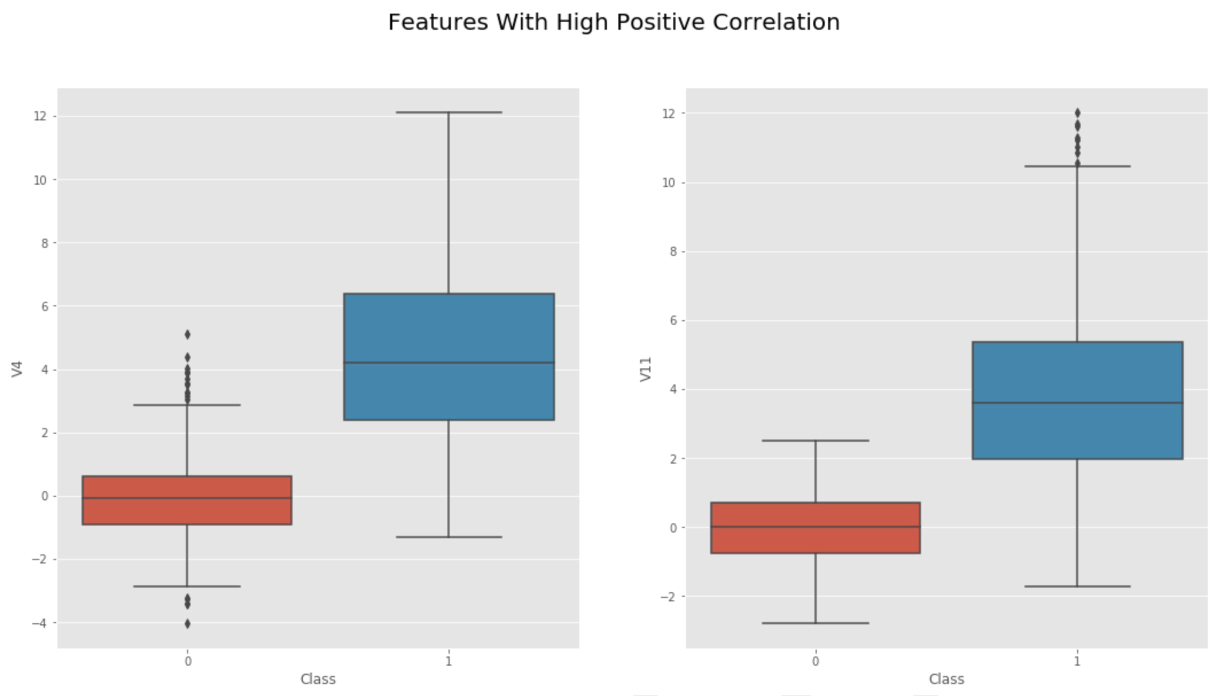
To create our balanced training data set, I took all of the fraudulent transactions in our data set and counted them. Then, I randomly selected the same number of non-fraudulent transactions and concatenated the two. After shuffling this newly created data set, I decided to output the class distributions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



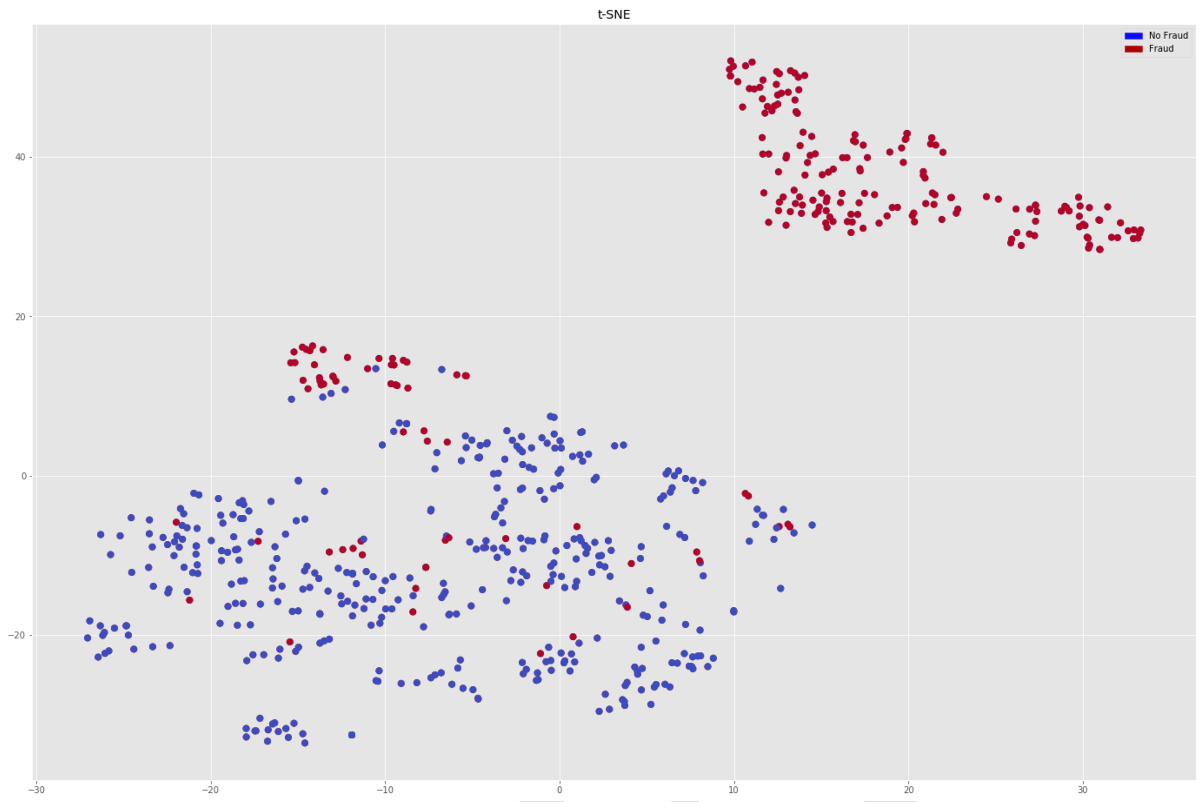


Box plots provide us with a good intuition of whether we need to worry about outliers as all transactions outside of 1.5 times the IQR (Inter-Quartile Range) are usually considered to be outliers. However, removing all transactions outside of 1.5 times the IQR would dramatically decrease our training 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 for 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.

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

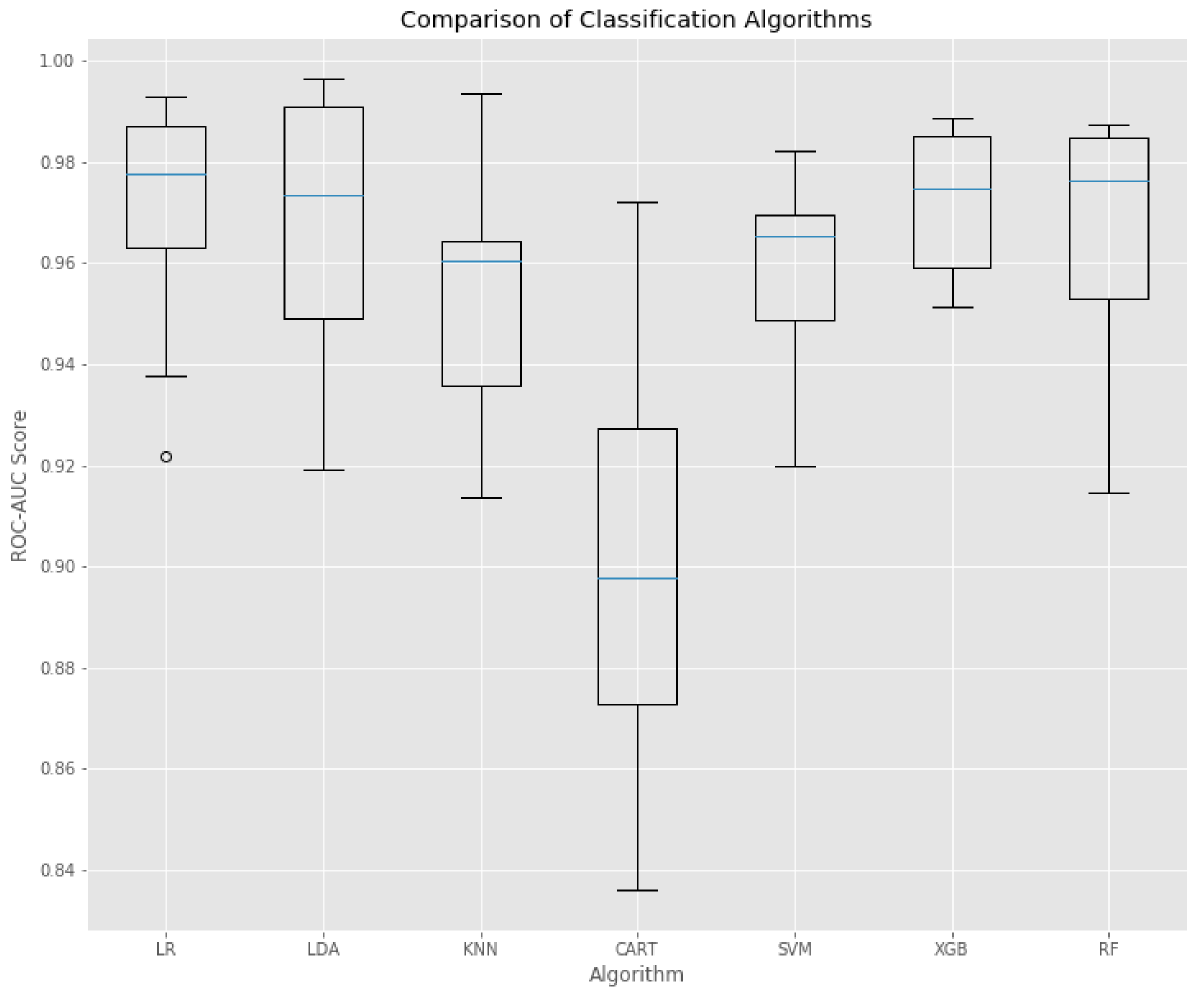
Onto the part you've probably been waiting for all this time: training machine learning algorithms. To be able to test the performance of our algorithms, I first performed an 80/20 train-test split, splitting our balanced data set into two pieces. To avoid overfitting, I used the very common resampling 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:

Result

Identify fraudulent credit card transactions.

Given the class imbalance ratio, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

The code prints out the number of false positives it detected and compares it with the actual values. This is used to calculate the accuracy score and precision of the algorithms.

The fraction of data we used for faster testing is 10% of the entire dataset. The complete dataset is also used at the end and both the results are printed.

These results along with the classification report for each algorithm is given in the output as follows, where class 0 means the transaction was determined to be valid and 1 means it was determined as a fraud transaction.

Conclusion

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. Credit Card Fraud Detection Based on Transaction Behavior -by John Richard D. Kho, Larry A. Vea published by Proc. of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, November 5-8, 2017

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

3. Machine Learning Group — ULB, Credit Card Fraud Detection (2018), Kaggle

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