Machine Learning Engineer Nanodegree

Capstone Project

Anshul Sharma

August 11th, 2019

I. Definition

Project Overview

Credit card fraud is a wide-ranging term for theft and fraud committed using or involving a payment card, such as a credit card or debit card, as a fraudulent source of funds in a transaction. The purpose may be to obtain goods without paying or to obtain unauthorized funds from an account. Credit card fraud is also an adjunct to identity theft.
Although incidences of credit card fraud are limited to about 0.1% of all card transactions, they have resulted in huge financial losses as the fraudulent transactions have been large value transactions. It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase. What we need is an algorithm, which could classify a transaction as fraudulent or non-fraudulent. Doing so will benefit both the credit card companies and the customers who have to go through the ordeal.

The dataset is provided by Kaggle and contains 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, the original features and more background information about the data could not be provided. 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-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Problem Statement

This is a binary classification problem. Inputs are the 30 features of the dataset and the goal is to predict whether the transaction is fraudulent. I will be tackling this as a classification problem. Then I plan to use a deep neural network to train the dataset. The features are pretty well drawn out, will have to see if any preprocessing is required. The target here is either “fraudulent” or “non-fraudulent” transaction.

Metrics

Given the class imbalance ratio, I would be using metrics such as Precision, Recall, Area Under the Precision-Recall Curve (AUPRC), ROC curve, e.t.c. to measure the accuracy. Confusion matrix accuracy is not meaningful for unbalanced classification because if out of millions of transactions I'm not able to detect 1000 fraudulent transactions and there are only these many fraudulent transactions then also the accuracy of the model would be great although the model is doing absolutely nothing. In order to evaluate the model, I'll be using a deep neural network as my benchmark model and compare it with other classifiers based on the metrics mentioned above. My main is to determine each and every fraudulent transaction with the least error.

II. Analysis

Data Exploration

First of all, to get a basic knowledge of the data I displayed the top 5 rows of the data and with that saw all the columns associated with it. Since a transaction can be fraud or non-fraud I checked the number of records of each class and found that it is an imbalanced dataset with 99.83% non-fraud and 0.17% fraud records. Due to some confidentiality issues, all the column names could not be provided. I then checked if any null values are present in the dataset and surprisingly it had none. Also, all out of all the 30 features, 28 were scaled except 'Time' and 'Amount'.So, the dataset didn't require much preprocessing, just some feature scaling for the two non-scaled features.

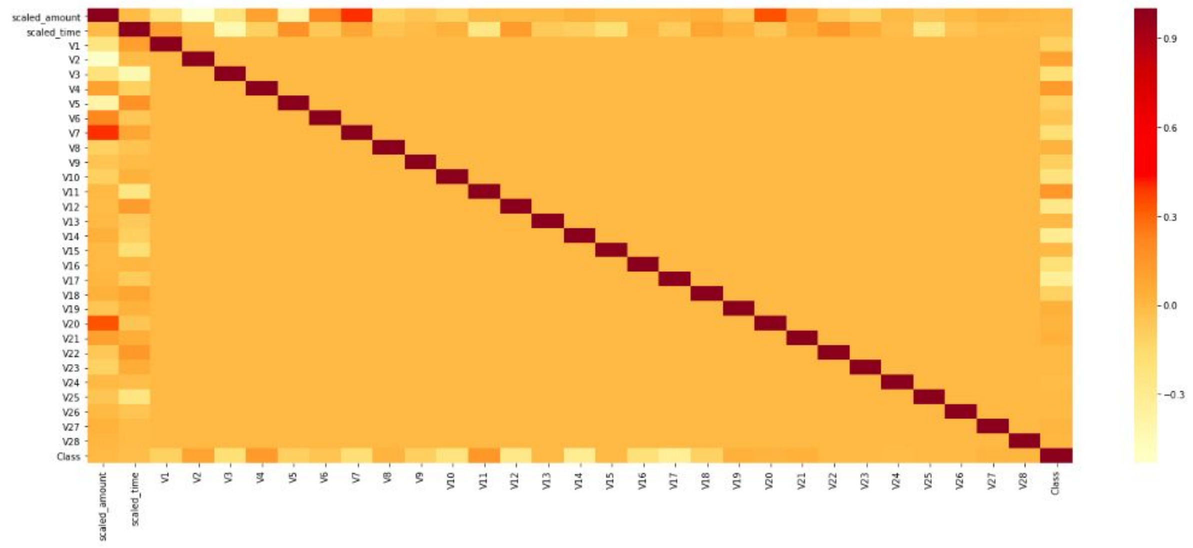
Exploratory Visualization

Now I will perform EDA on the dataset to study the various features of it. Since it is an imbalanced dataset, I would first undersample the majority class(i.e. Non-Fraud in this case) so that I can get an equal distribution of Fraud and Non-Fraud cases. For undersampling, I have used Random under sampling before crossvalidation. Although it is prone to overfitting, for EDA it can be performed. The following is the exploration I performed on the dataset:-

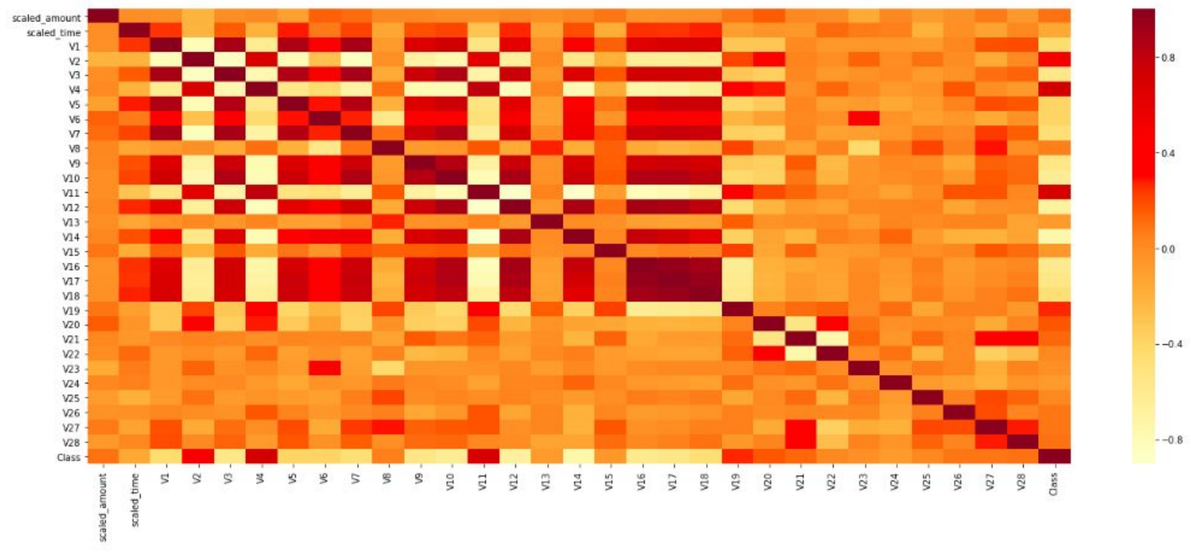
1. <u>Correlation Matrices</u>

   In order to get the essence of the data, I'll be plotting correlation matrices. Since there are 31 features it seems to be a better option to go for a heatmap kind of correlation matrix where different colors would indicate the correlation between any two features of the dataset. My main aim is to how the different features of the dataset help in determining a fraudulent or a non-fraudulent transaction.

   This is the heatmap for the original dataset.



   This is the heatmap for the randomly undersampled dataset.



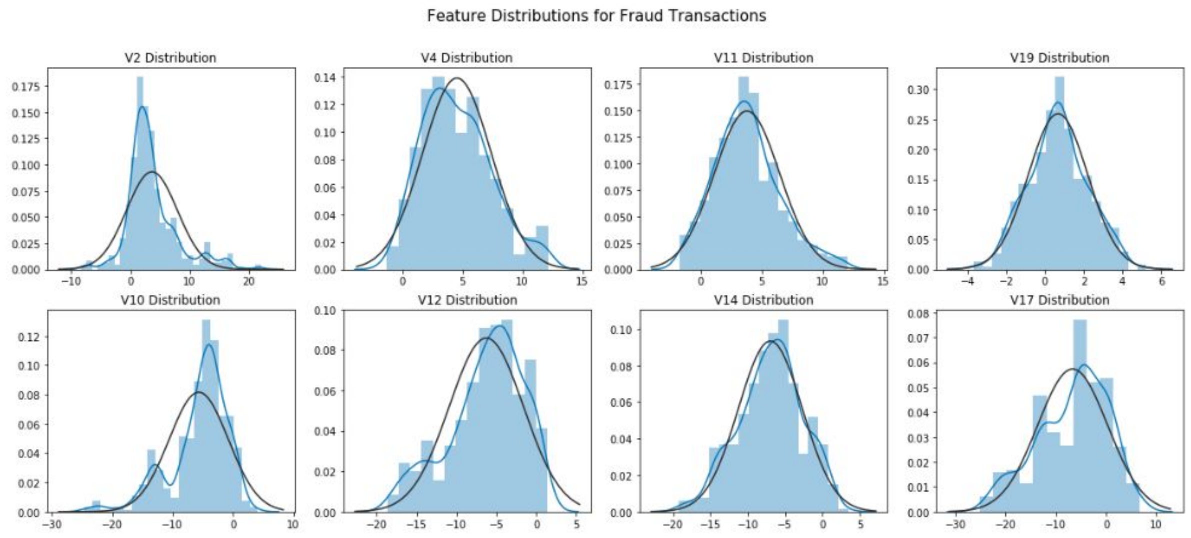
   My main aim is to find the features which influence a transaction to be fraudulent. In order to achieve this, I will focus on the last column of the plot where the correlation between various features and the Class is computed. The following is my analysis:-

   - **Positive Correlation:** Features *V2, V4, V11 & V19* show positive correlation with class. The higher the values of these features, the higher the chances of a transaction being fraudulant.

- **Negative Correlation:** Features *V10, V12, V14 & V17* show negative correlation with class. The lower the values of these features, the higher the chances of a transaction being fraudulant.

2. Histograms

I then plotted the histograms of the features listed above. The first row of histograms shows the distribution of positively correlated features with Class: *V2, V4, V11 & V19*, while the second row shows the distribution of negatively correlated features with Class: *V10, V12, V14 & V17*. I have also fitted each of the plots with normal distribution to judge how close the feature distribution is to a Gaussian distribution. **Note: All the feature distributions are for fraudulent transactions i.e Class = 1.**



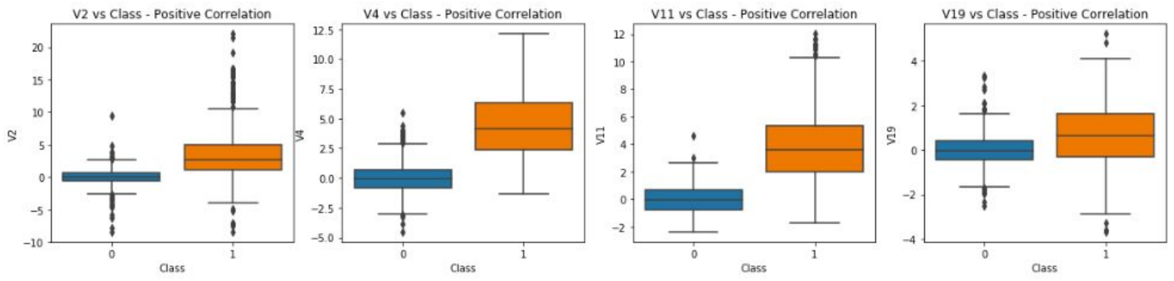
3. Boxplots

A boxplot is a standardized way of displaying the distribution of data based on a five-number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). It can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.

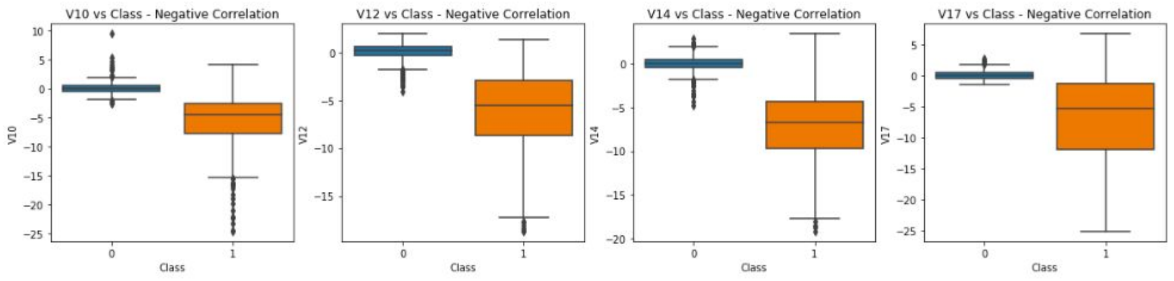
You need to have information on the variability or dispersion of the data. A boxplot is a graph that gives you a good indication of how the values in the data are spread out. Although box lots may seem primitive in comparison to a histogram or density plot, they have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets.

I plotted boxplots for all the features which show positive and negative correlations with Class. The boxplot representation for each feature is shown separately for each of the class(0 & 1).

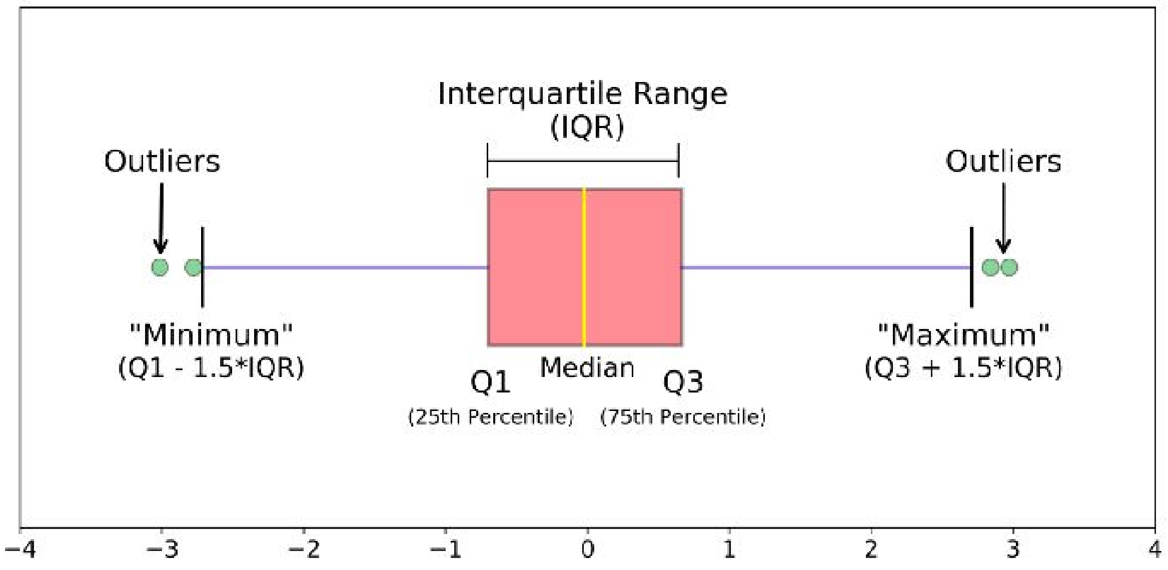
Boxplots of features showing Positive Correlation.



Boxplots of features showing Negative Correlation.

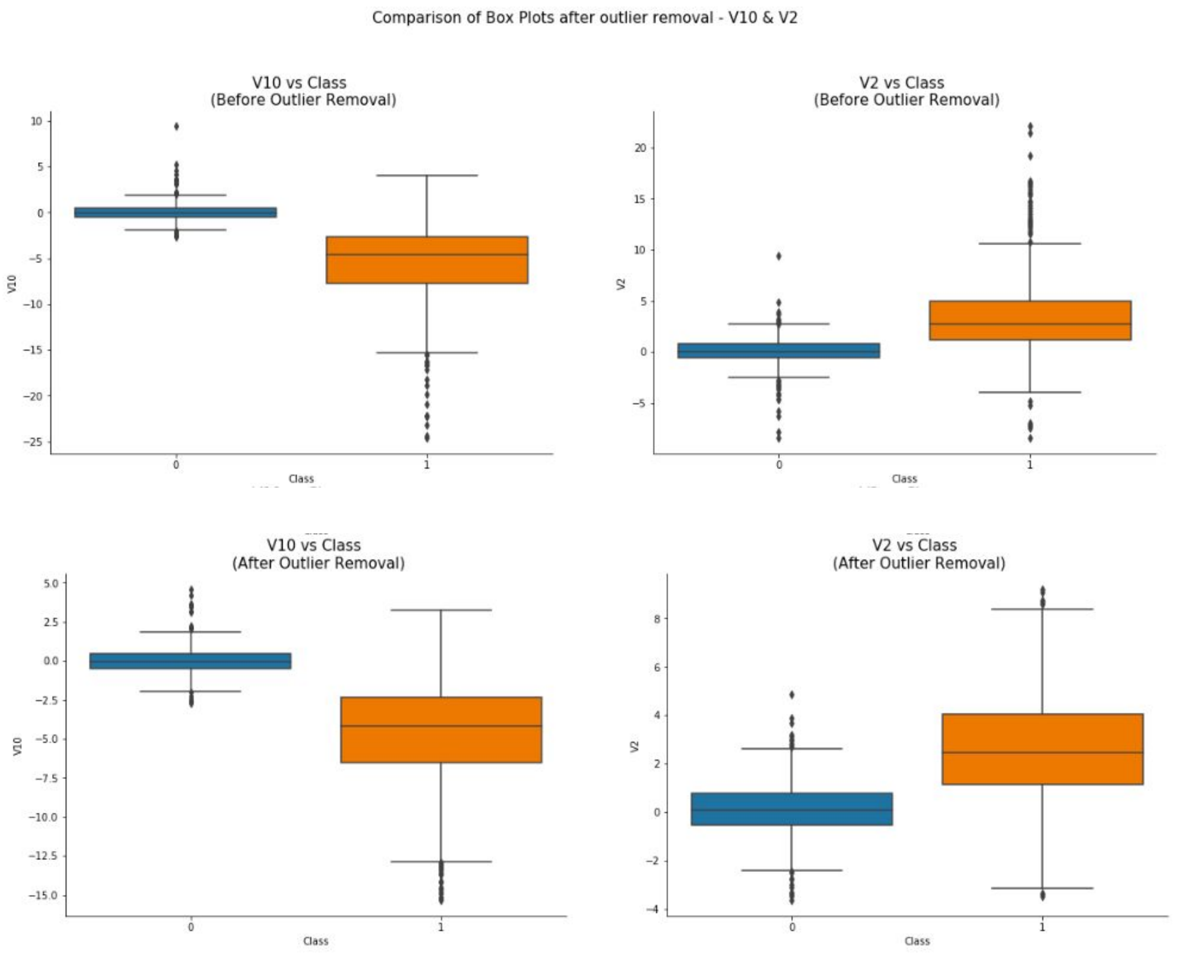


4. Outlier Detection and Removal



- **median (Q2/50th Percentile)**: the middle value of the dataset.
- **first quartile (Q1/25th Percentile)**: the middle number between the smallest number (not the “minimum”) and the median of the dataset.
- **third quartile (Q3/75th Percentile)**: the middle value between the median and the highest value (not the “maximum”) of the dataset.
- **interquartile range (IQR)**: 25th to the 75th percentile.
- **maximum**: Q3 + 1.5*IQR
- **minimum**: Q1 -1.5*IQR
- **whiskers**(shown in blue)
- **outliers**(shown as green circles): Anything below the minimum value and above the maximum value is an outlier

I then deviced a method, which will detect the outliers and then remove them from the dataset. I tried to remove the outliers for features *V10* and *V2* as through the above boxplots we can see that these have the maximum number of outliers.



Algorithms and Techniques

I plan to use four supervised classifiers to classify the data and to check the performance of each of them on the dataset. The classifiers used by me are:-

- Logistic Regression Classifier
- Support Vector Classifier
- Decision Tree Classifier
- KNearest Neighbours Classifier

First, I calculated the cross-validation score for each of the models using their default settings. Then, I applied the grid search technique to fine-tune the parameters of the models and the following were the hyperparameters used by me for each of the classifiers:-

- **Logistic Regression** : {"penalty": ['l1', 'l2'], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}

- **Support Vector Classifier** : {'C': [0.5, 0.7, 0.9, 1], 'kernel': ['rbf', 'poly', 'sigmoid', 'linear']}
- **Decision Tree**: {"criterion": ["gini", "entropy"], "max_depth": list(range(2,4,1)), "min_samples_leaf": list(range(5,7,1))}
- **KNearest** : {"n_neighbors": list(range(2,5,1)), 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']}

After finding the most optimal models, I studied their learning curves and area under the ROC curve to find the best supervised model for classification. I also created a simple neural network to compare its performance with those of the classifiers.
Please note all the analysis was done on the randomly undersampled dataset. Not on the original dataset

Benchmark

My benchmark would be a Naive Predictor. It will be a model which predicts all the transactions as Non-Fraudulant. The following will be the definitions for the model:-

- "Fraud(Class = 1)" is a **negative class**.
- "Non-Fraud(Class = 0)" is a **Positive class**.

Please note that the the purpose of generating a naive predictor is simply to show what a base model without any intelligence would look like. In the real world, ideally your base model would be either the results of a previous model or could be based on a research paper upon which you are looking to improve. When there is no benchmark model set, getting a result better than random choice is a place you could start from.

Naive predictor for original dataset:

[Accuracy score: 0.9983, precision: 0.9983, recall: 1.0000, f1-score: 0.9991]

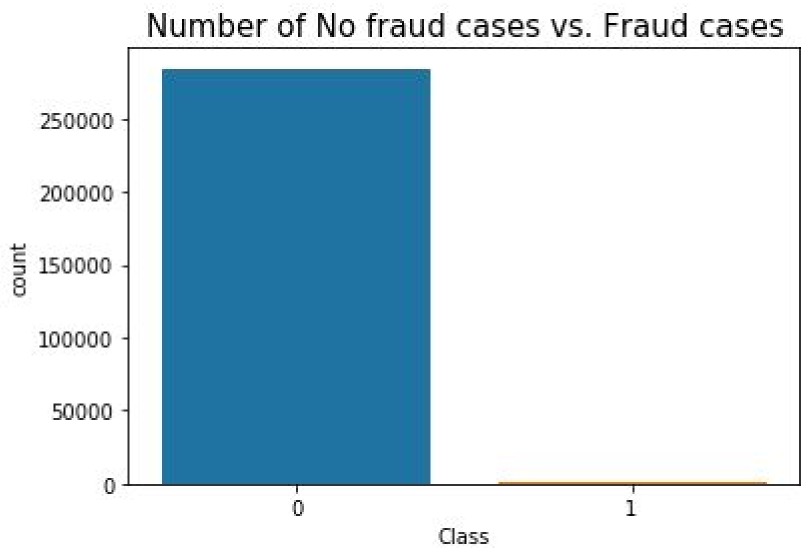
Naive predictor for undersampled dataset:
[Accuracy score: 0.5000, precision: 0.5000, recall: 1.0000, f1-score: 0.6667]

III. Methodology

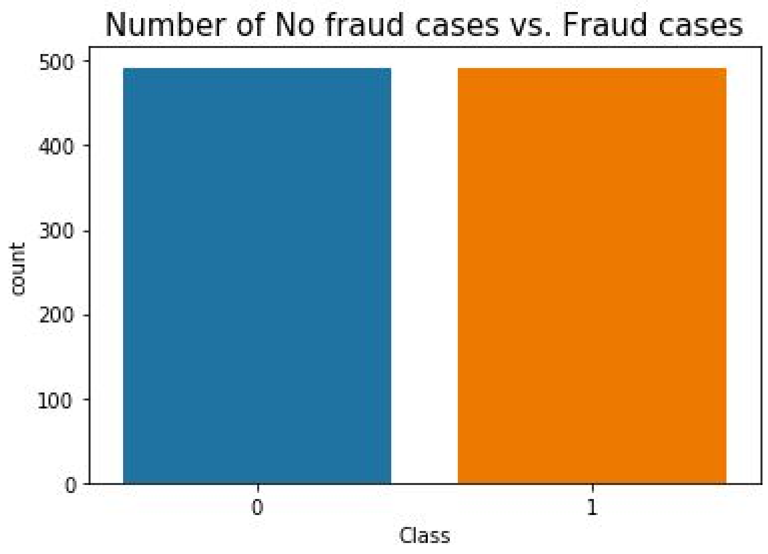
Data Preprocessing

As stated in the analysis section that the dataset did not have any null values and only two features: Time & Amount required scaling. Also, all the features were in correct format. So, no preprocessing was required as such.

The dataset was imbalanced as it had 99.83% non-fraud and just 0.17% fraud records.



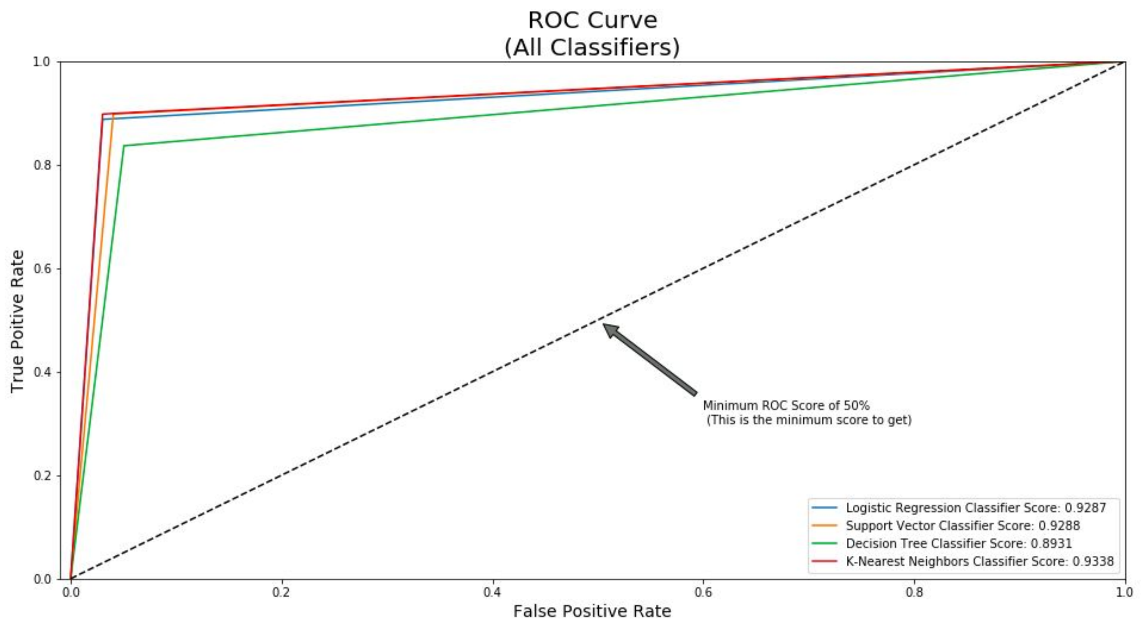
After applying random undersampling, it had 50:50 ratio of fraud vs non-fraud cases.



On analysing the boxplots of the features of the dataset I found that few features had a lot of outliers. So, I detected the outliers and tried to remove them using some code. That process is also explained in the analysis section.

Implementation

Usually, when testing a machine learning model, accuracy is the metric which comes to mind. Due to the class imbalance ratio in the dataset, accuracy clearly won't be a good metric to consider. Instead, I went with metrics such as precision, recall, and f1-score. I calculated the accuracy of the models using AUC-ROC(Area Under the Curve - Receiver Operating Characteristic) technique. Here, I tested which classifier out of the 4 chosen classifiers produced a value that is close to 1 and obviously greater than a 0.5. My findings were these:-



From the area under the ROC curve, it can be seen that all the four classifiers perform pretty well(have high accuracy) with the KNearest Neighbors classifiers showing the best results. For all other metrics, I used the classification report for the classifiers. A classification report is the best way to calculate precision, recall, and f1-score for all the classes in the dataset. Here is the result of the classification report:-

Logistic Regression:

```
          precision    recall  f1-score   support

  No Fraud       0.90      0.97      0.93        99
     Fraud       0.97      0.89      0.93        98

avg / total       0.93      0.93      0.93       197
```

Support Vector Classifier:

```
          precision    recall  f1-score   support

  No Fraud       0.90      0.96      0.93        99
     Fraud       0.96      0.90      0.93        98

avg / total       0.93      0.93      0.93       197
```

Decision Tree:

```
          precision    recall  f1-score   support

  No Fraud       0.85      0.95      0.90        99
     Fraud       0.94      0.84      0.89        98

avg / total       0.90      0.89      0.89       197
```

KNearest Neighbours:

```
             precision    recall  f1-score   support

   No Fraud       0.91      0.97      0.94        99
      Fraud       0.97      0.90      0.93        98

avg / total       0.94      0.93      0.93       197
```

Refinement

Initially, after declaring the classifiers, I calculated the cross-validation score for all on the randomly undersampled dataset. This score calculation was done using the default parameters of the models. In order to refine the models so that they could perform optimally, I defined a set of hyperparameters for the models and then used the grid search technique to fine-tune the hyperparameters. The following was the difference:-

Cross-Validation Scores before applying GridSearch:-
Logistic Regression: 94.28
Support Vector Classifier: 94.03
Decision Tree: 89.71
KNearest: 93.52

Cross-Validation Scores after applying GridSearch:-
Logistic Regression: 95.04
Support Vector Classifier: 94.92
Decision Tree: 92.63
KNearest: 94.66

It's pretty evident that the grid search has helped in the refinement of all the classifiers.

IV. Results

Model Evaluation and Validation

After analysing the performance of the benchmark model and beating it with that of the 4 classifers that I selected, I tried to find a model which would perform even better. So, I turned to a deep neural network. In order to keep it simple, I created a neural network and a single hidden layer. The following are some of the details of the Neural Network:-

- **Neural Network Structure**: As stated previously, this will be a simple model composed of one input layer (where the number of nodes equals the number of features) plus bias node, one hidden layer with 32 nodes and one output node composed of two possible results 0 or 1 (No fraud or fraud).

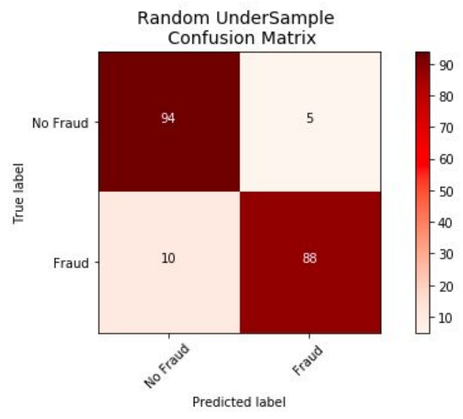
- **Other characteristics**: The learning rate will be 0.001, the optimizer we will use is the AdamOptimizer, the activation function that is used in this scenario is "Relu" and for the final outputs we will use sparse categorical cross entropy, which gives the probability whether an instance case is no fraud or fraud (The prediction will pick the highest probability between the two.)

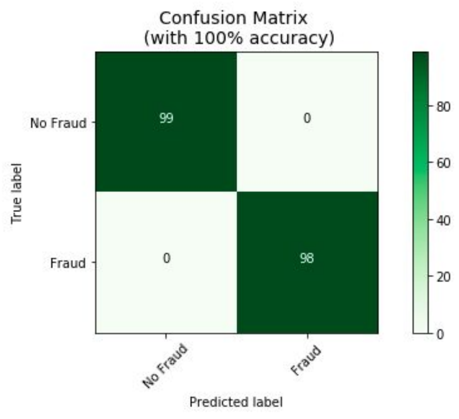
I used the Keras library for creating and testing the neural network. I tested it on two datasets and the following were the results:-(In case of the neural network i have used confusion matrix as the metric to calculate the accuracy)

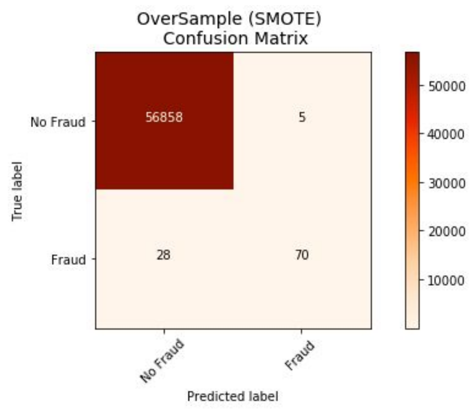
Dataset	Best Accuracy Reported
Randomly Undersampled	0.9714
Oversampled(using SMOTe)	0.9996

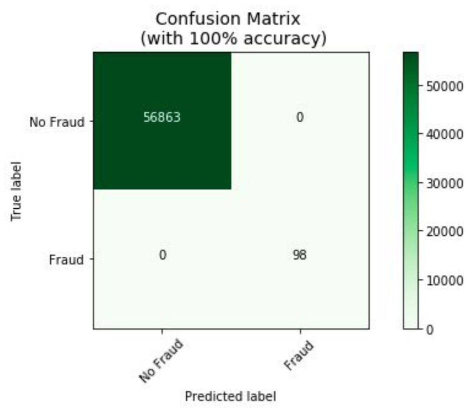
It can be seen that the NN performed pretty well on both the datasets. The best performance was produced on the oversampled dataset which was done using Synthetic Minority Oversammpling Technique(SMOTe).

Below are the confusion matrix for both the datasets along with the confusion matrix showing 100% accuracy :-









Justification

The benchmark model(Naive Predictor) produced the following metrics:-

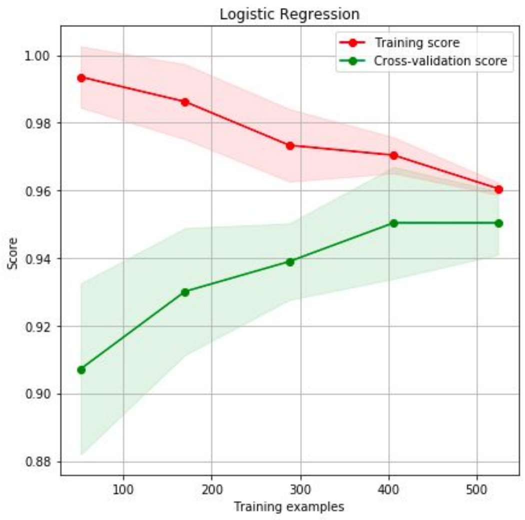
Dataset	Accuracy
Original	0.9983
Randomly undersampled	0.5

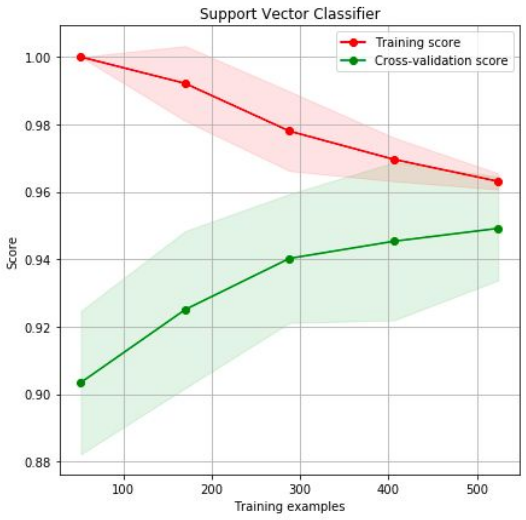
Obviously, the Naive predictor would perform well on the original dataset because of the class imbalance ratio and it considers all the records as non-fraudulent. Logically, it makes no sense to assume all the transactions as non-fraudulent as we are completely missing out on the fraudulent records.
A Neural network tested on an oversampled dataset above would be the best competitor for the benchmark model set earlier. As can be seen from the above section, the NN produces an accuracy of 0.996. Therefore, it is pretty accurate with it's predictions and is able to classify fraudulent and non-fraudulent records properly.

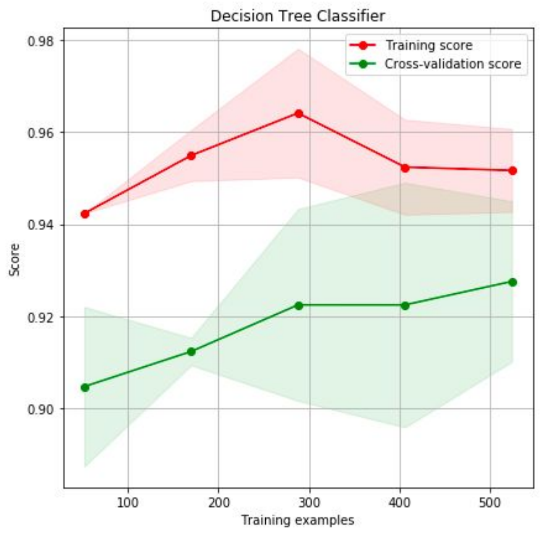
V. Conclusion

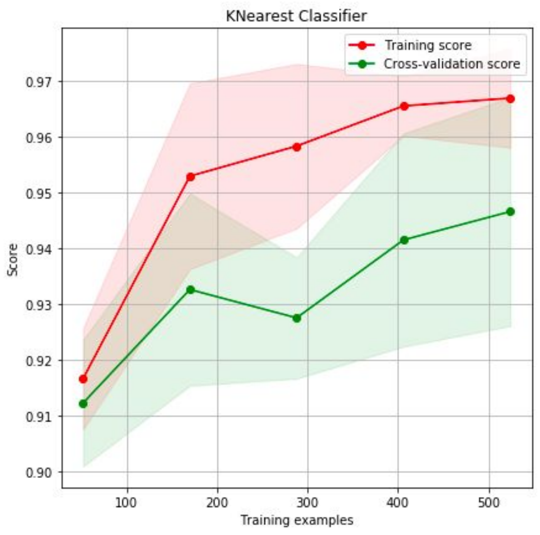
Free-Form Visualization

In order to access the performance of the classifiers listed in the above sections, I plotted learning curves for all of them to see how the classifiers perform on the training and the validation sets. The following are the learning curves:-









As can be seen through the learning curves above, **Logistic Regression** and **Support Vector Classifier** show the best score in training and cross-validation set and are pretty close to the ideal case. Ideal case is the one where the training and cross-validation scores remain high and tend to merge with each other as the number of training examples increase.

Reflection

In this project, I tried to find a model which would help me predict whether a transaction is fraudulent or non-fraudulent. In order to achieve this, I downloaded a dataset from kaggle whose information is available here. I then performed preprocessing on the dataset and later on did exploratory data analysis on it to study all of its features and to check which all influence a transaction to be fraudulent or non-fraudulent . Initially, I defined a Naive predictor which would act as the benchmark model for all other models that I put to test. After that, I took four different classifiers and tested them on the randomly undersampled dataset and evaluated several metrics on it: accuracy, precision, recall, f1-score. I also plotted learning curves and AUC-ROC curve to get a better understanding of the performance of the 4 classifiers. Oversampling using SMOTe technique was also done on the dataset and then also the metrics were evaluated. Lastly, I used a simple neural network to test the randomly undersampled and oversampled data. In the end, it can be concluded that the neural network performed the best on the oversampled dataset and will be the best model for fraud detection.

Improvement

From the start I have used the randomly undersampled dataset for Exploratory data analysis and further analysis using the benchmark model, the supervised classifiers and the neural network. I believe some different undersampling technique (eg. Nearmiss) can be used or even the entire analysis could be done on an oversampled dataset so that we don't need to worry about losing out on some of the details of the data.