Machine Learning Engineer Nanodegree Capstone Project

Credit Card Fraud Detection Challenge

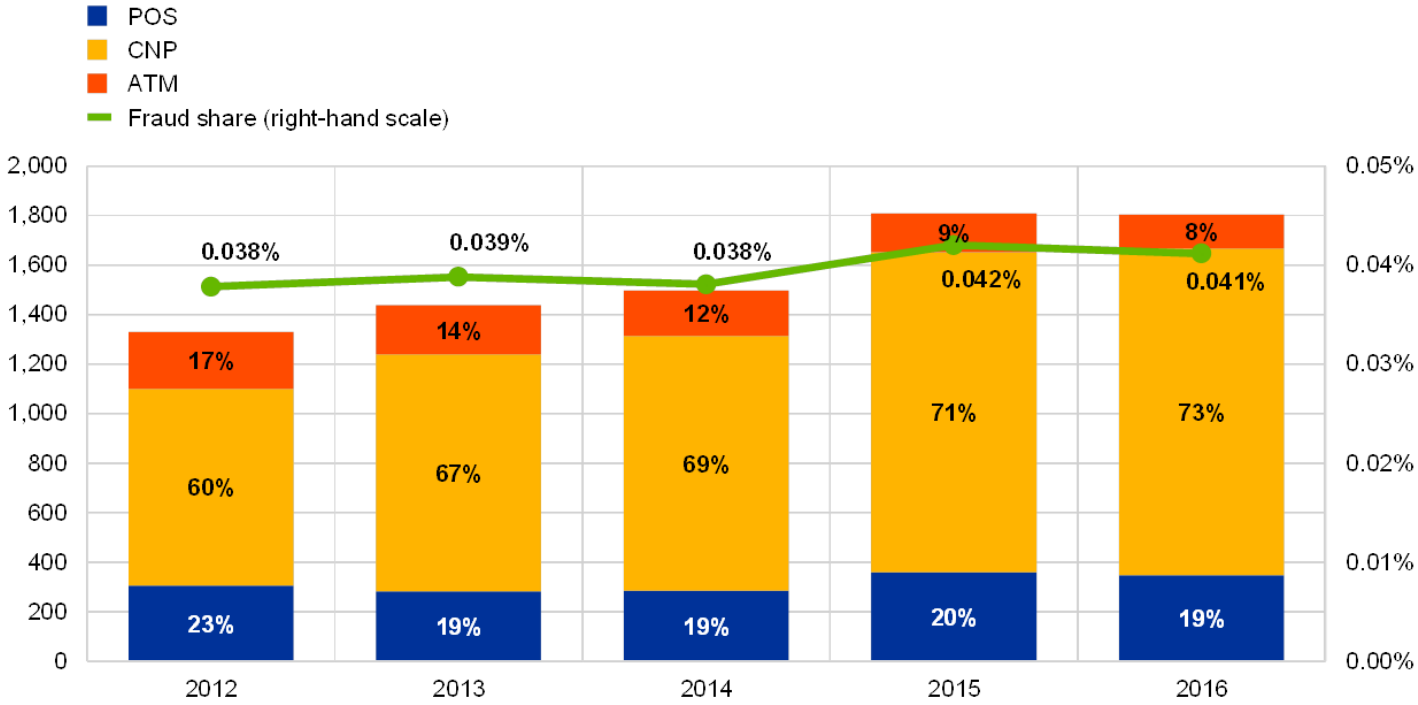
Rohit Venkatachalam
3rd June, 2019

Definition

Project Overview

Digital transactions are on the rise. And so are Credit Card and identity thefts. Below is a stats published by the European Central bank.

The image depicts Evolution of the total value of card fraud using cards issued within SEPA. (Single Euro Payment Account)



Source: All reporting CPSs.

Preventing fraudulent transactions is important, as it is a key component to promote digital transactions. Increase in digital transactions will lead to reduced cash transactions, and help in reduce crimes, arising out of cash transaction.

ML for Fraud Detection

AI/MI based solutions have just started skimming the surface when it comes to fighting financial crime.

Some of the past and future work in the related areas, can be viewed at

https://www.researchgate.net/project/Fraud-detection-with-machine-learning

Problem Statement

Recognize fraudulent credit card transactions so that customers are not charged by credit card companies for items that they did not purchase.

Datasets and Inputs

The dataset provided in Kaggle, was collected by Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

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.

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.

Dataset Kaggle Url: https://www.kaggle.com/mlg-ulb/creditcardfraud

Solution Overview

The problem at hand is a binary classification problem. The solution will be capable of predicting whether a given transaction is fraudulent or not.

To solve this problem, a prediction model would be developed using

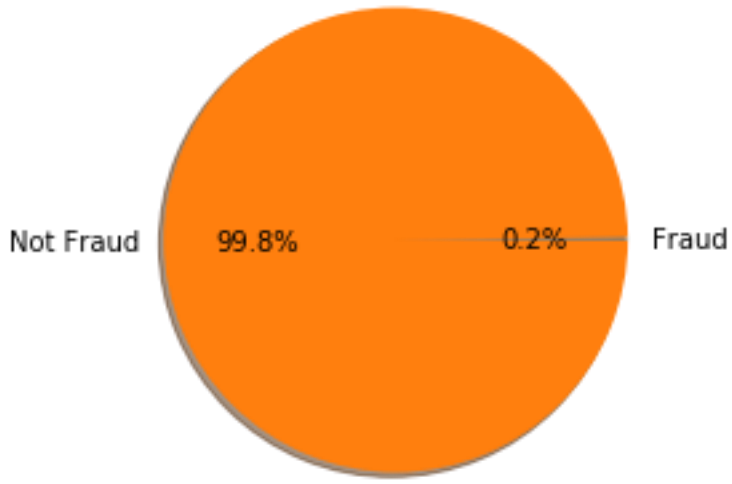
1. Light Gradient Boost
2. Logistic Regression
3. Multi layer Perceptron CNN.

The logistic regression will be used as a benchmark model.

LGB and CNN will be tuned to enhance the accuracy.

The dataset provided, already been dimensionality reduced. All the features will be utilized.

This is a highly imbalanced data. Only 0.2% of the data is classified as fraudulent. Hence if the data is used as is, the model will be biased and will throw false negatives.



A high percentage of data classified as Not-Fraud will be problematic, as it will make the model biased towards, classifying data as Not-Fraud. As our hypothesis, assumes data to be Fraud, this will cause an increase in False Negative. While we do want our model to be highly precise, we would prefer it to err in the side of more FALSE POSITIVES

To address this bias will be using a technique called oversampling, as suggested, by the reviewer in the capstone proposal, I will be using SMOTE .(Synthetic Minority Over Sampling Technique)

This has also been suggested in the data set description in Kaggle

References for SMOTE:

1. https://medium.com/erinludertblog/smote-synthetic-minority-over-sampling-technique-caada3df2c0a
2. https://imbalanced-learn.org/en/stable/generated/imblearn.over_sampling.SMOTE.html
3. https://beckernick.github.io/oversampling-modeling/

Evaluation Metrics.

For the problem at hand accuracy is not a good enough metric. We should be able to predict the probability with which the model will specify a outcome, as the cost of **False Negatives** is more severe than that of **False Positives.**

Two diagnostic tools that help in the interpretation of probabilistic forecast for binary (two-class) classification predictive modeling problems are ROC Curves and Precision-Recall curves.

ROC v Precision/Recall curves

ROC Curves:

ROC(Receiver Operating characteristic curves are a useful tool for predicting the probability of a binary outcome.

Precision- Recall Curves:

Precision is a ratio of the number of true positives divided by the sum of the true positives and false positives. It describes how good a model is at predicting the positive class. Precision is referred to as the positive predictive value.

Precision = True Positives / (True Positives + False Positives)

Recall is calculated as the ratio of the number of true positives divided by the sum of the true positives and the false negatives. Recall is the same as sensitivity.

Recall = True Positives / (True Positives + False Negatives)

Sensitivity = True Positives / (True Positives + False Negatives)

Recall == Sensitivity

Reviewing both precision and recall is useful in our case as there is an imbalance in the observations between the two classes. Specifically, there are many examples of no event (class 0) and only a few examples of an event (class 1).

The reason for this is that typically the large number of class 0 examples means we are less interested in the skill of the model at predicting class 0 correctly, e.g. high true negatives.

A precision-recall curve is a plot of the precision (y-axis) and the recall (x-axis) for different thresholds, much like the ROC curve.

The no-skill line is defined by the total number of positive cases divide by the total number of positive and negative cases. For a dataset with an equal number of positive and negative cases, this is a straight line at 0.5. Points above this line show skill.

A model with perfect skill is depicted as a point at [1.0,1.0]. A skillful model is represented by a curve that bows towards [1.0,1.0] above the flat line of no skill.

We will be using the AUC ROC curve to evaluate the performance. The original data set is extremely imbalanced. But using techniques like SMOTE, we have overcome this. The data against which we training our model, has an equal distribution of data. Hence we can use AUC_ROC curves, instead of the AUC/Precision Recall curve, as suggested in the data set. We will evaluate how relevant AUC_ROC is against AUC Precision Recall curve.

For more details, please refer this excellent article.

https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

Analysis

Data Exploration

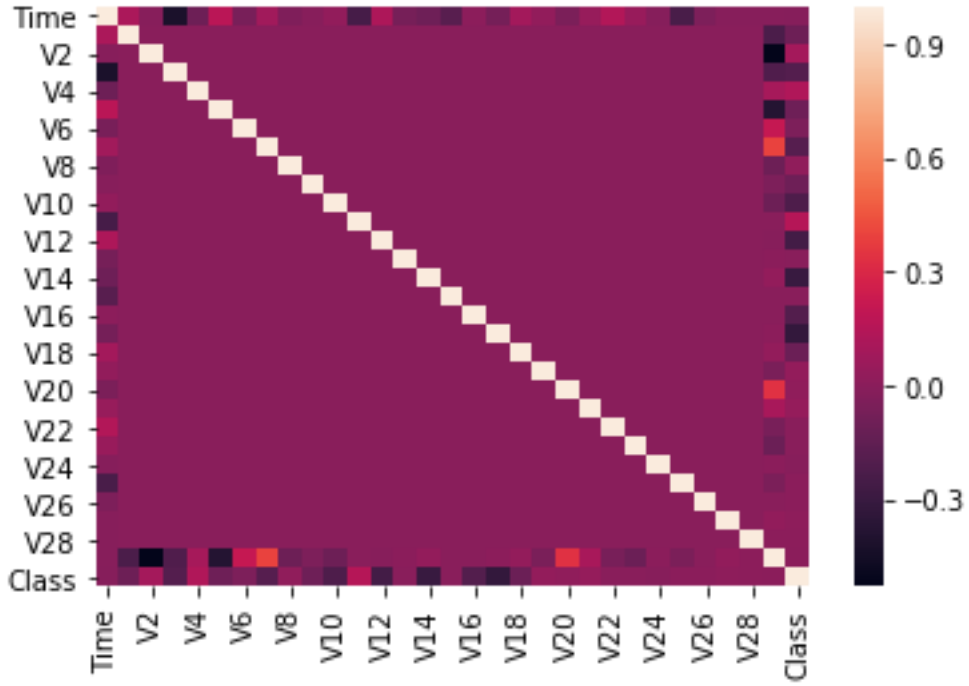
The dataset to be used in solving this problem is an anonymized set of credit card transactions labelled as fraudulent or genuine. Due to confidentiality issues, the original features and more background information about the data were not provided.

The dataset presents transactions that occurred in two days, where there are 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.2% of all transactions.

The dataset contains 31 numerical features. The first 28 features are labelled V1, V2….V28 and these are principal components obtained with Principal Components Analysis of the raw/original data.

Correlation between Features:

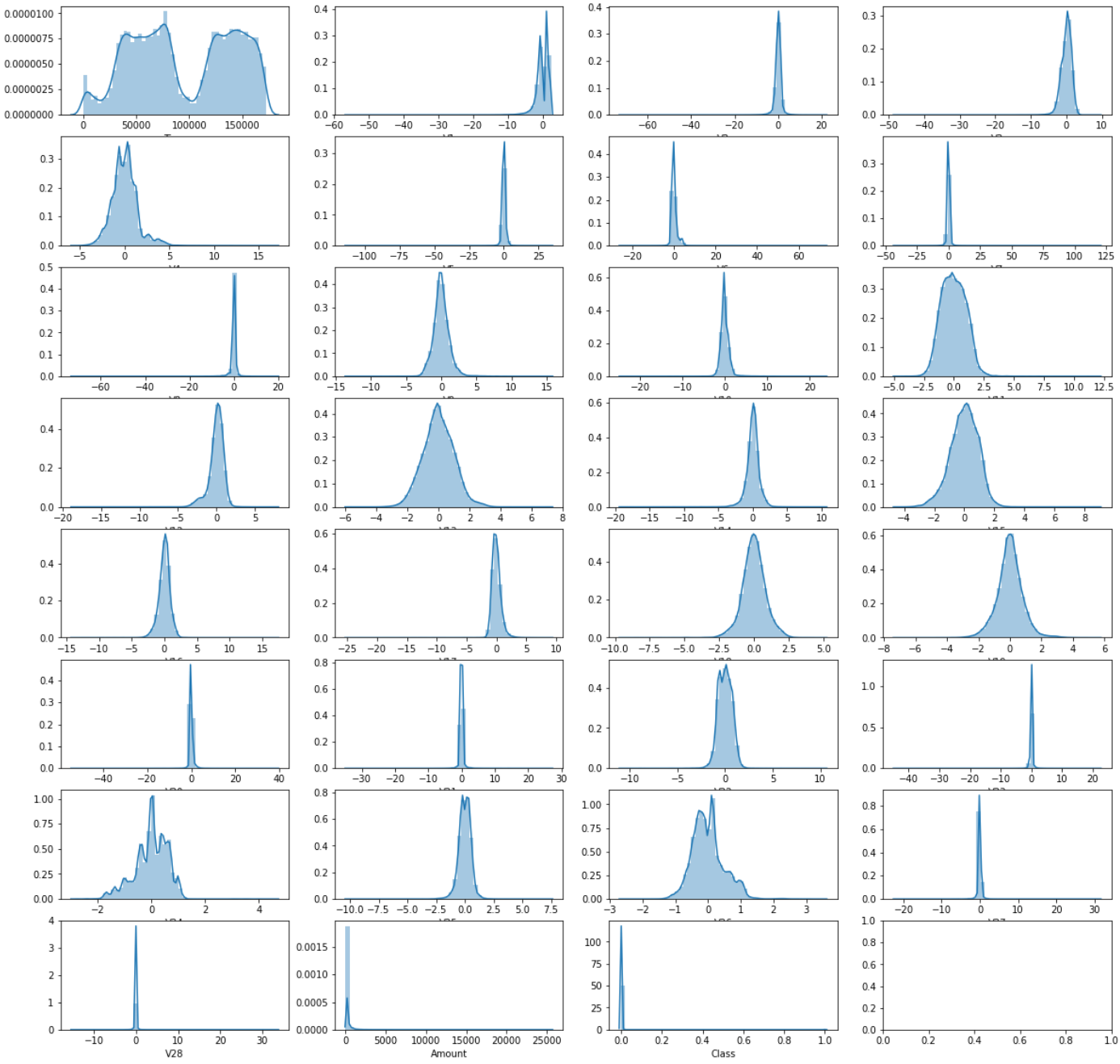
The heatmap suggests that there is very little correlation between the features. Feature engineering has already been applied. So this is expected, only the most essential features are provided in the data set



Feature Distribution Data:

Below is the plot of all the features, after normalization of the amount column.

Most of them are normally distributed.



AS per the information given in Kaggle, only features which have not been transformed with PCA are:

1. Time - contains the seconds elapsed between each transaction and the first transaction in the dataset
2. Amount - transaction amount . This needs to be scaled down. There are other columns with skewed distribution, but there are many outliers, as in the case of Amount column.

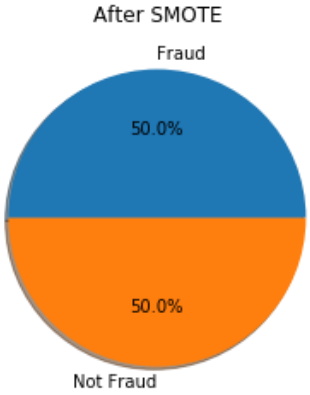
The last feature to be discussed is ‘Class’ which is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Exploratory Visualization

Fig 1. A plot showing the ratio of fraudulent transactions to genuine transactions, from this visualization it is obvious that fraudulent cases are grossly misrepresented and may do some harm while training our model.

If we train our model in this data, the model will be inclined to mark more transactions as negative. We want to minimize the false negative classification as little as possible

To overcome this bias, we will be using a technique called Synthetic oversampling technique.



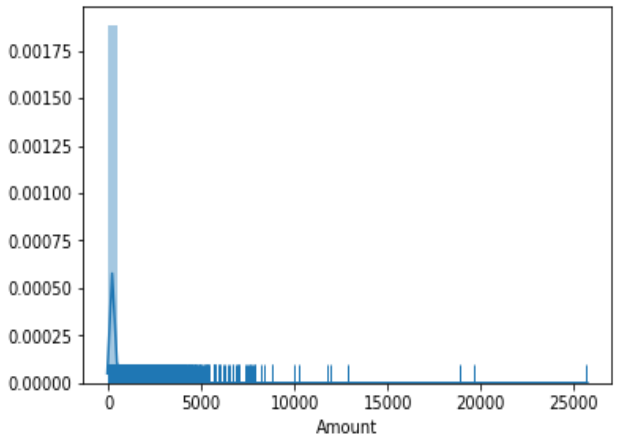
The Amount column is not inline in scale with the other columns:

The statistics for the Amount column:

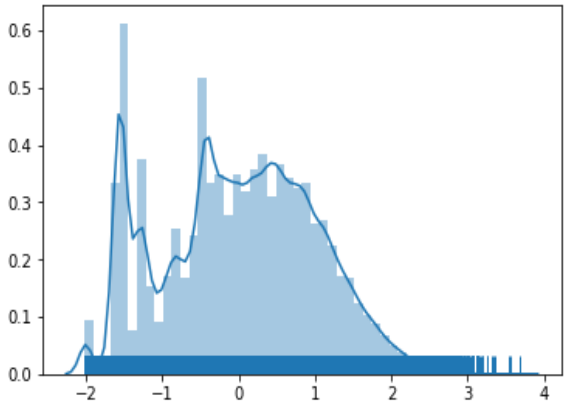
count	284807.000000
mean	88.349619
std	250.120109
min	0.000000
25%	5.600000
50%	22.000000
75%	77.165000
max	25691.160000

There is a huge difference between min, max and the mean values, indicating skewness in distribution and presence of outliers. The histogram for the amount column, further confirms that this is a Right Skewed data. We will have to apply scaling on this column to apply transformation. After trying multiple scalers, a PowerTransformer will be applied to normalize the data for this .

Amount Column Raw Data



Amount column Normalized Data



Algorithms and Techniques:

One of the goals is the project is to see how CNN's perform vis-à-vis conventional Machine Learning algorithms.

We have chosen 3 algorithms.

1. Light Gradient Boosting
2. LogisticRegression
3. Multi Layer Perceptron:

Light Gradient Boosting(LGB)

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

Logistic Regression(LR):

The logistic regression algorithm is a popular algorithm used in classification problems. This algorithm works by learning a numerical weight for each feature and a constant term from our training dataset. Given a new point from the test dataset, it multiplies each feature by the corresponding learnt weight and adds all the product together with the learnt constant term. The result of this operation is then fit into a logit function which scales it down to a number between 0 & 1. The final number gotten can be interpreted as probabilities with results lesser than 0.5 indicating a prediction of false and results greater than or equals to 0.5 indicating a prediction of true.

The following parameters can be tuned to optimize the logistic regression classifier:

I. Penalty - specifies the kind of regularization option: L1 OR L2

II. C – Inverse of regularization strength, smaller values specify stronger regularization.

In the grid search we have only provided, different values of the parameter C. L2 Regularization is chosen, as most of the recommended solvers support L2 over L1.

Logistic Regression will act as our baseline Algorithm. We will train Light Gradient Boost and Multi Layer Perceptron to beat the bench mark set by LR

Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is a Feed Forward artificial neural network, which consists of at least three layers of nodes. An input node, a hidden node and an output node. Its multiple layers and non-linear activation makes it suitable for data that is non-linearly separable.

For this project the MLP would have 3 hidden layers. The output layer would consist of just one node with a sigmoid activation function that outputs numbers between 0 & 1 to represent the probability of predicting a value for fraudulent transactions.

The Multi-Layer Perceptron deals very well with a complex decision boundary, but might have a risk of overfitting. The choice of the MLP classifier was to introduce and evaluate neural network algorithm to already existing options.

Below is the architecture of MLP used in the solution.

Except for the output layer which uses a Sigmoid activation function, all other layers use a RELU activation function.

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 16)	480
dense_2 (Dense)	(None, 18)	306
dropout_1 (Dropout)	(None, 18)	0
dense_3 (Dense)	(None, 20)	380
dense_4 (Dense)	(None, 24)	504
dense_5 (Dense)	(None, 1)	25

Total params: 1,695

Trainable params: 1,695

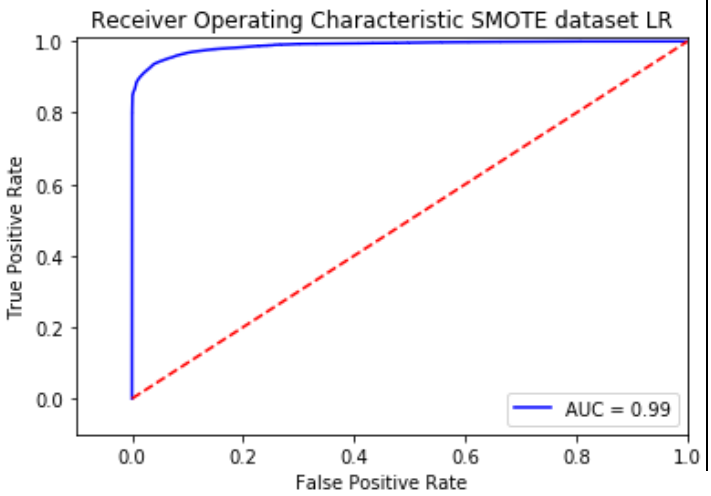
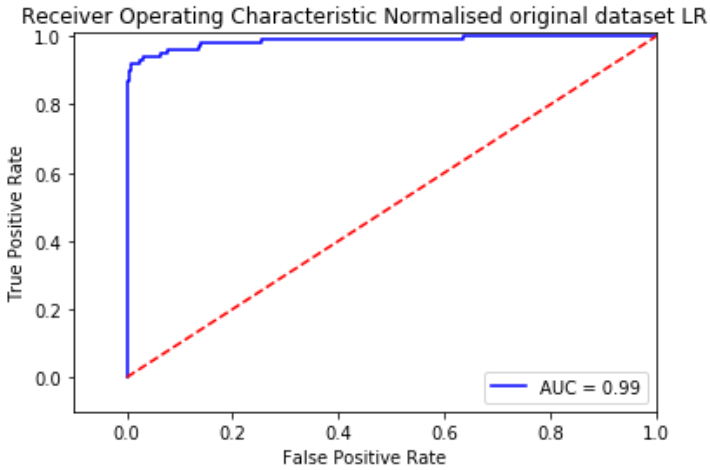
Non-trainable params: 0

Benchmark:

For the problem, given the imbalance in dataset a simple accuracy cannot be used as a benchmark metric. We would be using the area under the curve for the ROC curve and the Precision/Recall curve as our bench mark. Logistic Regression has been selected as the algorithm for benchmark results. The results for Logistic Regression are as follows.

Logistic Regression:

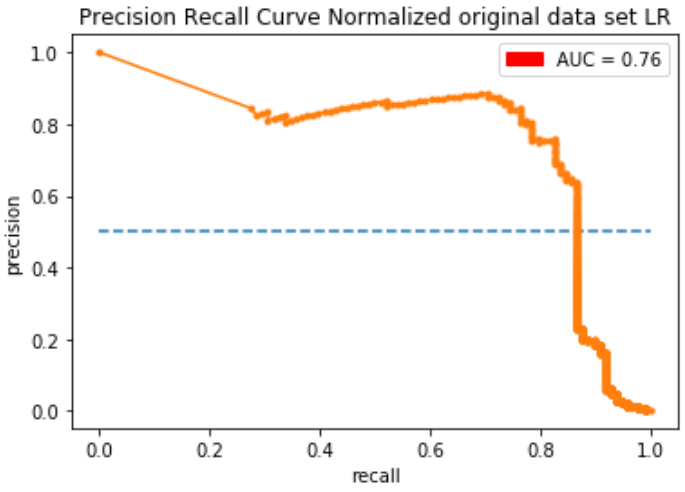
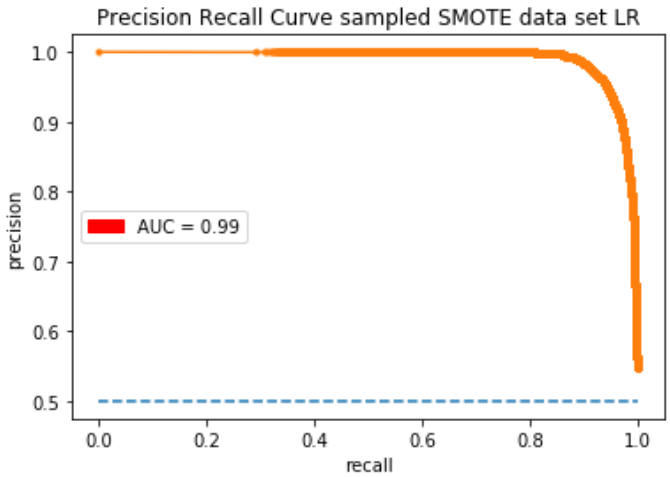
Receiver Operating Characteristic:

Result for predictions on the over Sampled Data Set	Result for prediction on the original normalized data set
	

The area under the curve for the ROC curve, seems to suggest that this is almost a perfect model

Now let's see what the Precision Recall curve suggests.

Precision Recall Curve

Result of predictions on the oversampled SMOTE dataset	Result of prediction on the original normalized data set
	

The AUC ROC seems to be overly optimistic. We would take the AUC for Precision Recall with a score of 0.76 as our bench mark

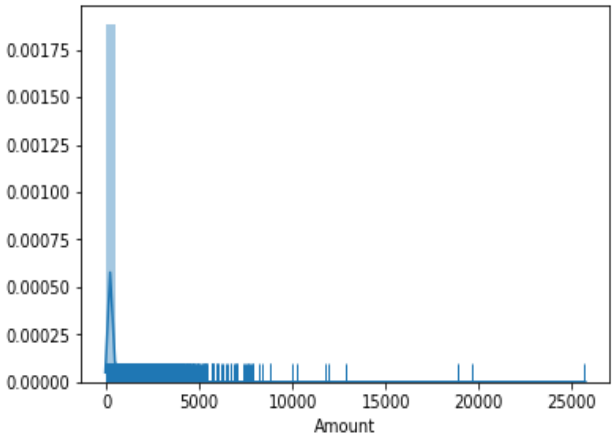
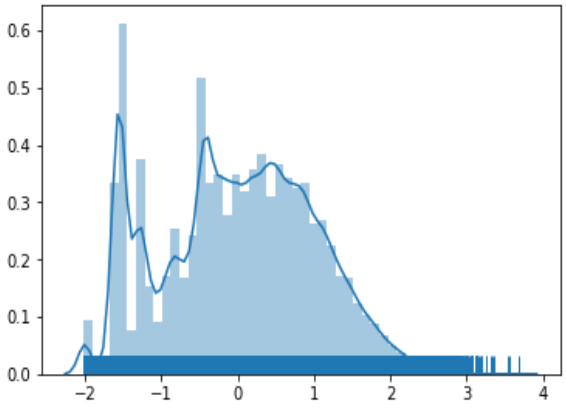
Methodology

Normalization:

The dataset has already undergone a dimensionality reduction.

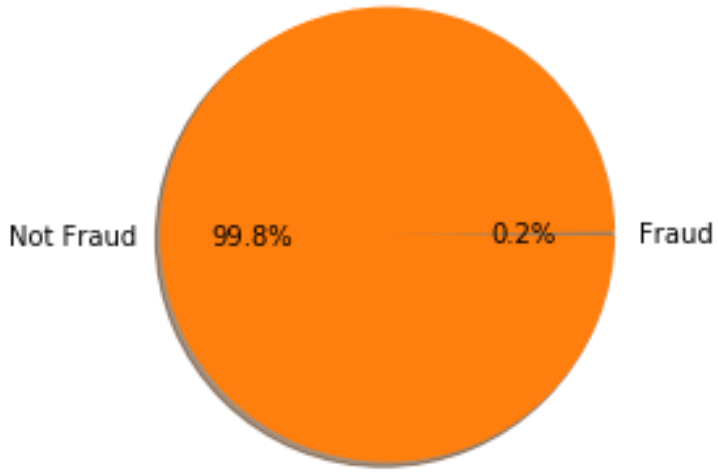
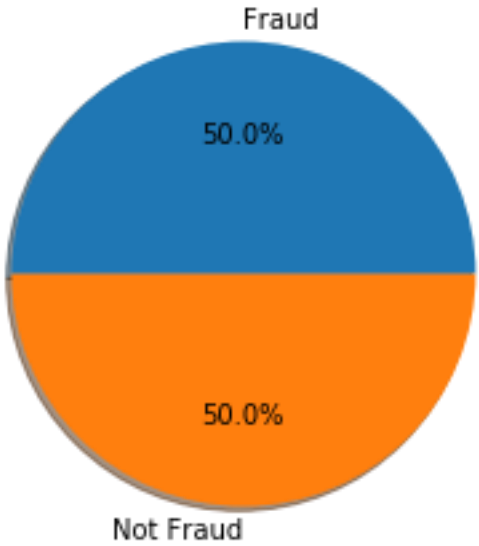
The amount field however has to be normalized, since it was not in the same scale as other figures.

PowerTransformer was used a scaler, because of the normalized distribution of it's output.

The distribution plot for the amount column before scaling.	Distribution Plot for the amount column after scaling
	

Sampling:

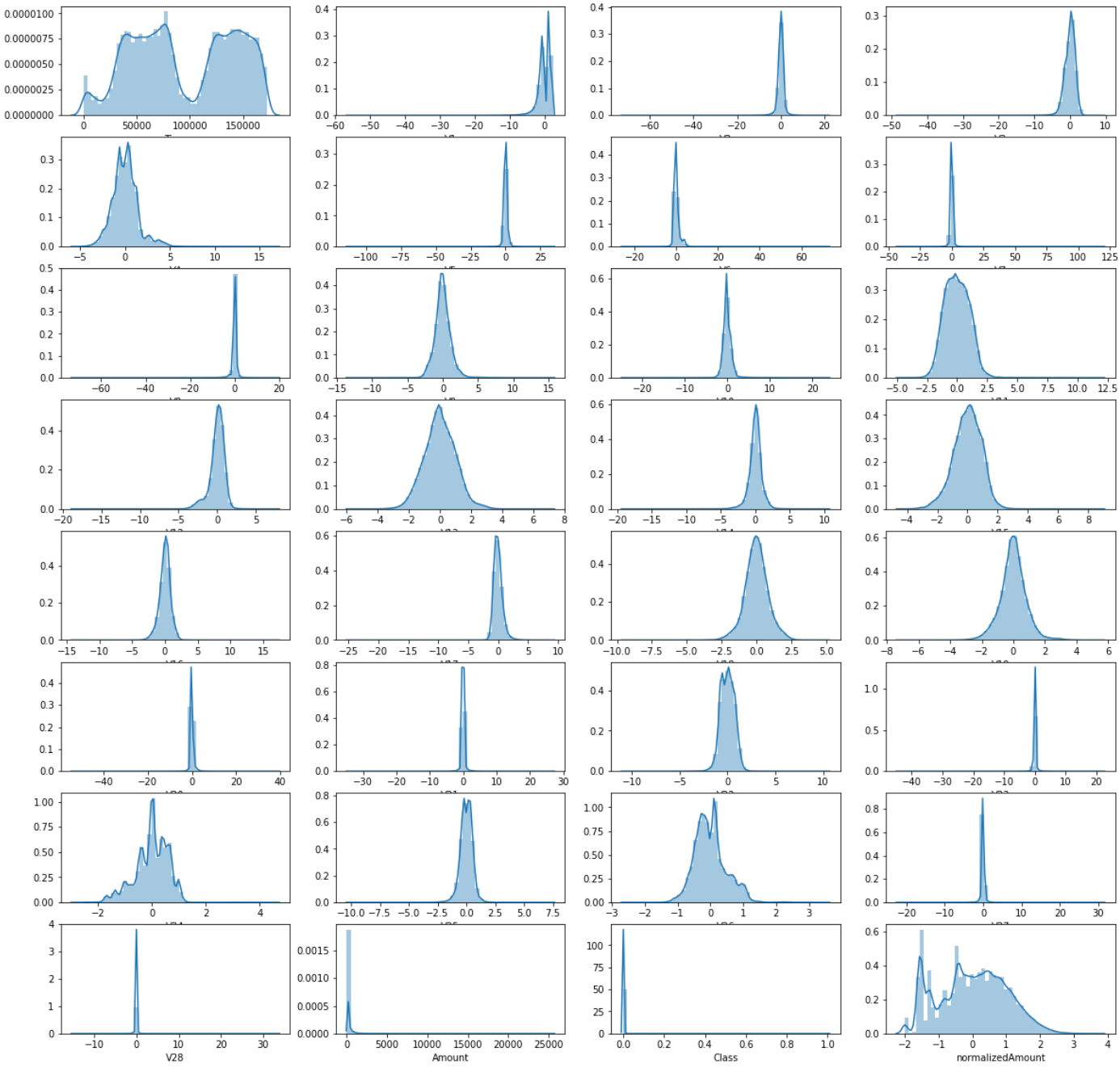
Next, the **SMOTE** algorithm was used to balance out the class ratio. It works by constructing new points from the minority class until it evens out the deficit in the class ratio.

Before SMOTE:	AFTER SMOTE:
	

Feature Distribution Data:

Below is the plot of all the features, after normalization of the amount column.

Most of them are normally distributed. The scaler chosen for the amount column is also fairly resilient to outliers. So we can assume, that the model trained will be fairly resistant to outliers



Implementation:

The implementation stage involved creating a training and predicting pipeline.

- The Logistic Regression was used to arrive at a bench mark score.
- The grid search classifier was used to arrive at the best estimator.
- The estimator was then trained against the oversampled(SMOTE) training data set.
- The final scores were evaluated against both the test data set from the original normalized data, as well as the oversampled data set.
- The benchmark AUC /precision –recall score of 0.77 was then chosen as the benchmark for the multi layer perceptron as well as Light Gradient Boosting algorithm.

The same process was then repeated for the LightGradientBoosting and the Multi Layer Perceptron.

Refinement:

Hyper Parameter tuning was performed to optimize the model for Light Gradient Boosting and Logistic Regression.

Logistic Regression:

Grid Search was used to arrive at the optimal C value

The penalty values are dependent on the scorer selected. Three scorers were considered. Since the oversampled data set is fairly large, >5 Lakh records, the scorer considered were.

1. Sag
2. Saga
3. Lbfgs

Sag and saga were discarded, because of the time taken to train the model. Going by the sklearn documentation for LogisticRegression, the default is going to change to lbfgs from the version 0.22, lbfgs was considered for the final model. Since lbfgs only supports l2, this helped in reducing th optimal model for LogisticRegression

Light Gradient Boosting

Multi Layer Perceptron

For Light gradient Boosting and Multi Layer perceptron, not much tuning had to be done. The parameters used are fairly standard, and yielded excellent results, on the test straight away. Grid Search in combination with cross validation was good enough, to yield an AUPRC of ~0.9

Results:

As suggested in the Kaggle dataset description the AUPRC is a good metric to test the model performance.
The AUPRC for Light Gradient Boosting was 0.91

The AUPRC for MLP was 0.88

The model was tested against a test data set from both the oversampled data set and the normalized data set to test for robustness. The LGB and MLP are comparable. The LGB a derivative of the gradient boosting tree is indeed a swiss army knife of model. It is a lot faster than MLP too.

Justification:

The final AUPRC score of 0.91 and 0.88 is much better than the benchmark score 0.76. An approximate 20% increase, from the benchmark.

This might not be an earth shattering score, in the context of real world use for fraud detection. But has promise. Even though, the LGB performs marginally better, in terms of score, the MLP model is more consistent. Even at high recalls, the precision does not drop significantly.

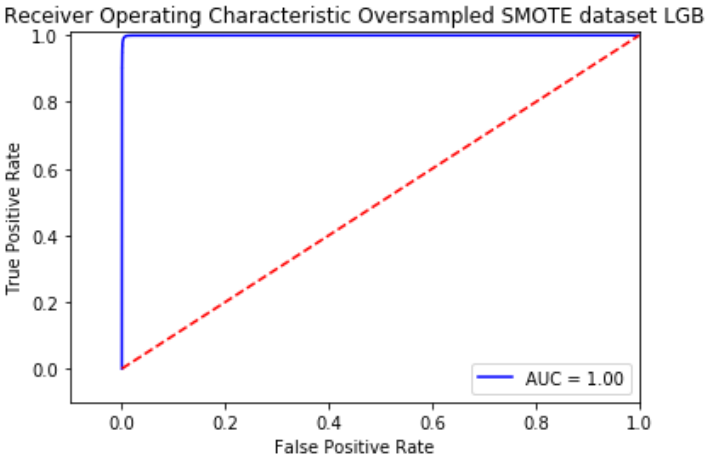
That's why I would prefer going ahead with MLP. MLP in combination with some RNN models can learn better, than LGB. In fact, in real world, this problem, can be changed to a multi classification problem, like "Suspicious", "Fraud" and "Not Fraud". This model can be generalized and extended to other financial crime problems too

Conclusion

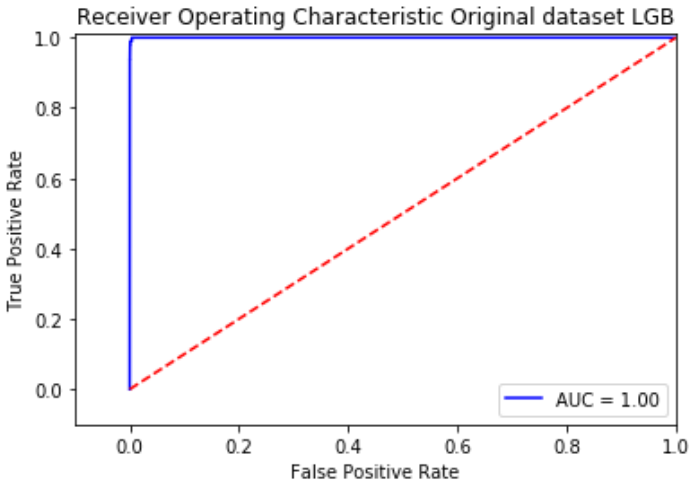
LIGHT GRADIENT BOOSTING

RECEIVER OPERATING CHARACTERISTIC

Test results from the over sampled data set:



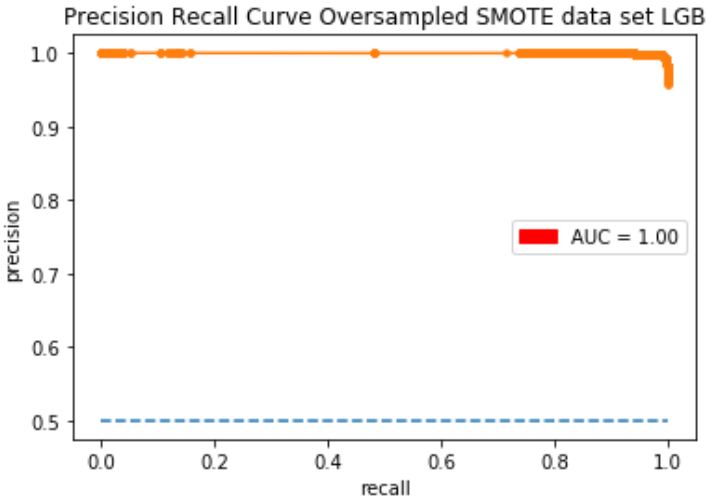
Test Results from the original normalized data set



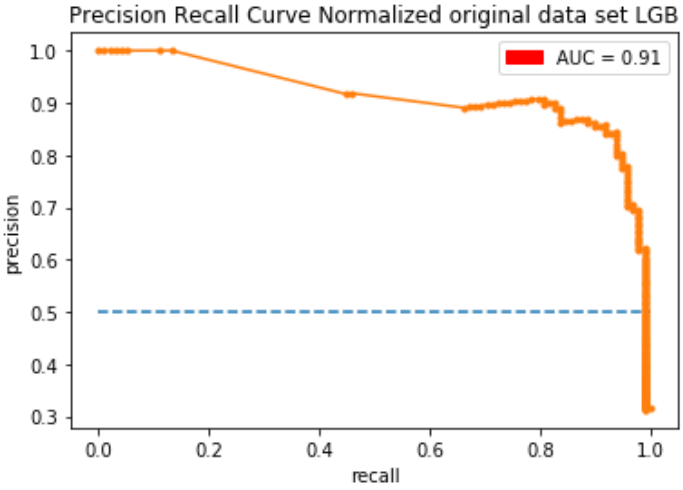
The AUC for the ROC is too perfect to believe.
Next section we evaluate the AUC for the Precision recall curve

PRECISION RECALL CURVE LIGHT GRADIENT BOOSTING

Test results from the oversampled data set



Test results from the original normalized data set

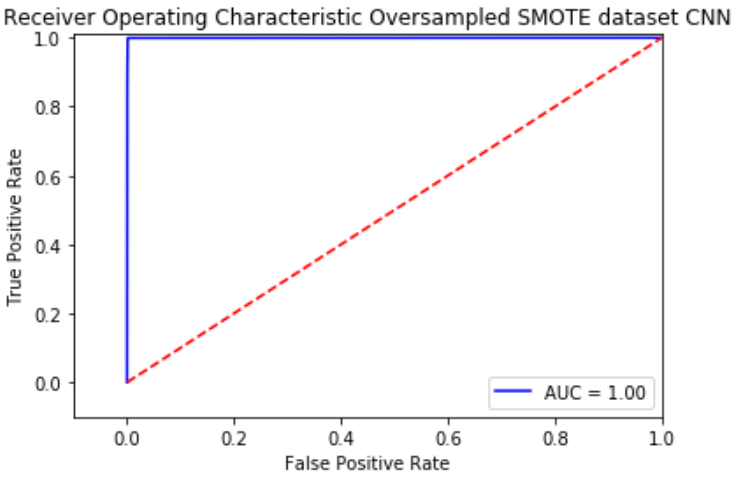


The area under the curve is an impressive 0.91. However the precision falls of steeply at higher values of recall

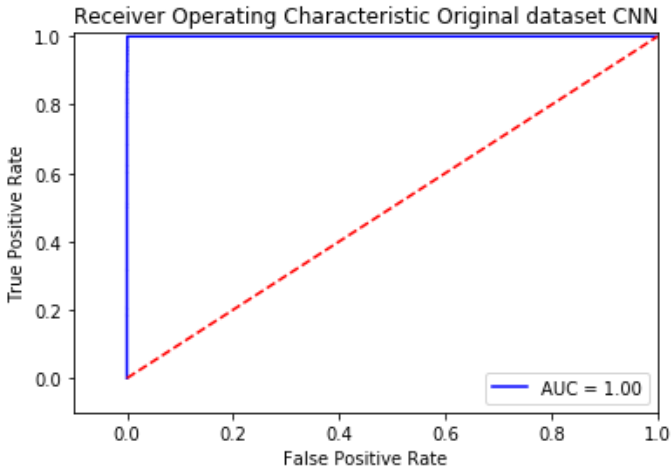
MULTI LAYER PERCEPTRON

RECEIVER OPERATING CHARACTERISTIC MLP

Test results from the over sampled data set

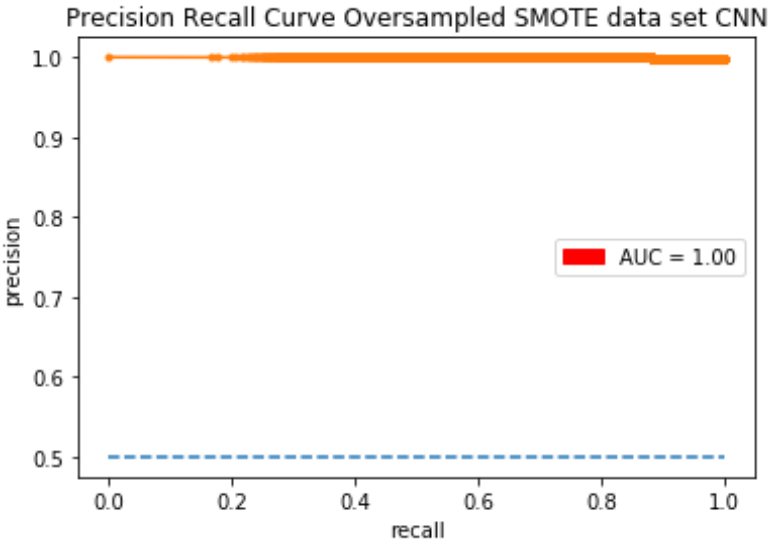


Test results from the original normalized data set

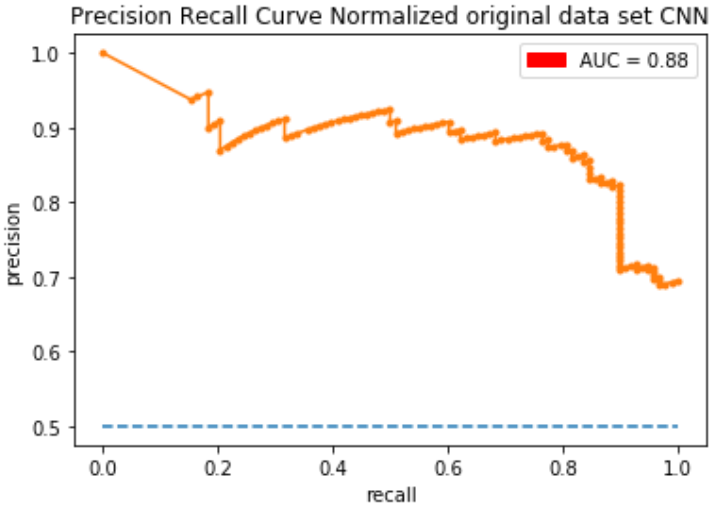


Precision Recall Curve MLP

Test results from the oversampled SMOTE data set



Test results from the original data set



The area under the curve for the ROC curve is 0.88 for MLP. This is lower than the value of 0.91 for LGB but better than 0.76 of Logistic Regression.

The ROC curve for the MLP suggests that the precision never falls of drastically even at higher values for Recall.

We can train the MLP to have a better score.

The area under the curve for the ROC curve is an idealistic value of 1 for all of the algorithms we evaluated

The reference materials, did suggest that ROC is not a good fit when there is imbalance in data.

Our tests seem to agree that this statistic is not reliable and is overly optimistic.

However the even the AUC for precision recall is a much improve ~0.9,

A significant improvement from the benchmark score of 0.77

Reflection:

The project can be broken down into 3 phases

1. Searching for a Kaggle data set
2. Data Exploration
3. Model Selection
   The problem is of particular interest to me. Being a software architect, who works mostly in banking domain, coming up with solutions to solve various areas of financial crime like Fraud, AML has become a necessity.

   Getting the data set was fairly easy, however, given the sensitive nature of data, the data provided already had it's dimensionality reduced. This did make my life a lot easier, as a lot of data cleaning/dimensionality etc was already done for me.
   However, it meant that, I knew very little about the data, for me to explore.
   Working on this, did give me some reflections to the real world challenges, I might have to face.
   Selecting the right algorithm, was another tricky little problem, but I do have good insights now when to use which, or at-least how do I go about pruning.

   While the algorithms were mostly evaluated for their efficiency vis-à-vis metrics like precision recall,
   Another parameter which I tested to evaluate the performance is the time taken to train the model.
   Light Gradient Boosting, was the fastest of all, and even did outperform the MLP. The MLP is heavy.
   While hardware is getting cheap, this might still be an important benchmark to evaluate algorithms.
   Thanks to Google's colab , running this code was a breeze, especially in the TPU runtime

Improvements:

In the context of predicting credit card fraud, this is but a first stop.

1. But this model can be enhanced, with better data to many aspects of financial crimes. Especially the Multi Layer Perceptron, provides a model which can be generalized.
2. Use some of the ensemble chaining techniques to enhance the MLP model

References:

https://www.kaggle.com/mlg-ulb/creditcardfraud

https://www.kaggle.com/gargmanish/how-to-handle-imbalance-data-study-in-detail

https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport201809.en.html

https://towardsdatascience.com/deep-learning-unbalanced-training-data-solve-it-like-this-6c528e9efea6

https://www.kaggle.com/varunsharma3/credit-card-fraud-detection-using-smote

https://www.kaggle.com/omkarsabnis/credit-card-fraud-detection-using-neural-networks