Machine Learning Engineer Nanodegree

Capstone Project: Credit Card Fraud Detection Model

Akbarali Shaikh July 19, 2020

Table of Content

1. Project Overview

Banking industry works on credit and income from interest which can be in a form of mortgage, other types of loan or from Credit Card. As cost of having credit card reduced it got broadly accepted in society and many people started having credit card. As the credit card reached the masses, new ways of fraud also came to existence. Today, CC fraud is a very big problem for a banking industry and it is plaguing financial industries for years, some countries it is more less problem than others

Solving this problem could bring about reduced volume of fraudulent transactions in financial industries hereby saving them a huge volume of money lost and this problem can be avoided by using predictive analytics where machine learning algorithms are used to detect fraud patterns and determine future probabilities and trends.

The dataset to be used for this project was gotten from data.world

2. Problem Statement

Just by looking at the transaction, it is often difficult know which is fraudulent or genuine. We have a binary classification problem i.e. either a transaction is fraudulent or genuine. Objective is to create a model that accurately predicts whether a transaction is fraudulent or not.

To solve this problem, first we will baseline the model and than run a predictive model on different machine learning algorithms on same dataset to see which will give the best performance which is better than the others,

These algorithms include Naïve Bayes Classifier, SVC, GaussianNB, LogisticRegression, KNeighborsClassifier, DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, BaggingClassifier, ExtraTreesClassifier, XGBClassifier and AdaBoostClassifier.

Dataset is normalized and all columns names changed before it is shared, it could also be a case where some of the features were reduced from original dataset, no other form of reduction would be made going forward. All features provided would be used in building the model.

3. Metrics

In binary classification model, accuracy with minimal type 1 and 2 error is an important metric.

Type I error also referred as False Positive: model predicts positive when it isn't.

Type II error also referred as False Negative: model predict negative when it isn't.

Accuracy is an important measure of binary classification model it shows how accurate the model predicts true and false.

In our case, we have a highly imbalanced dataset, other evaluation metrics consider such class imbalance:

1. Precision
2. Recall

Recall measures the fraction of fraudulent transactions that are correctly predicted while measures that fraction of cases predicted to be fraudulent that are truly fraudulent.

Precision and Recall are defined as follows:

Recall: When the focus in catching all fraudulent transactions even in case if few genuine transactions are also categorized as fraudulent then this is a go-to metric. When the cost of failures is high, we want to recall the rate to be high.

Precision: How accurate the model performs e.g., correctly categorizing true as true. Ideally. When raising false alerts is high we want to have

high precision

F beta score: It combines precision and recalls into one metric. The higher the score the better model. Value high than 1 means more emphasis on Recall as compared to Precision and if less than 1 it is vice versa.

F1 score (beta=1) It's the mean of precision and recall

F2 score (beta=2) It's a metric that combines precision and recall, putting 2x emphasis on recall - consider using it when recalling positive observations (fraudulent transactions) is more important than being precise.

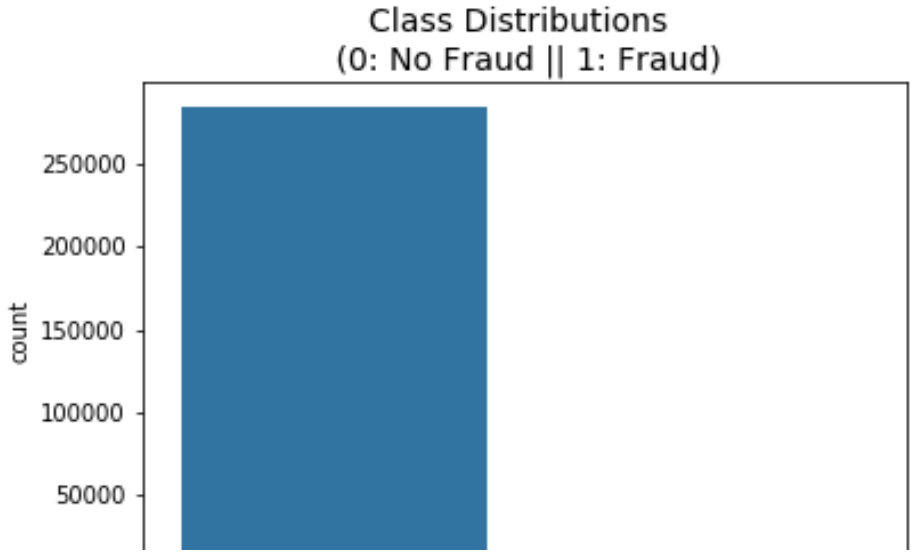
4. Analysis

Data Exploration

The dataset used is 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 .17% of all transactions.

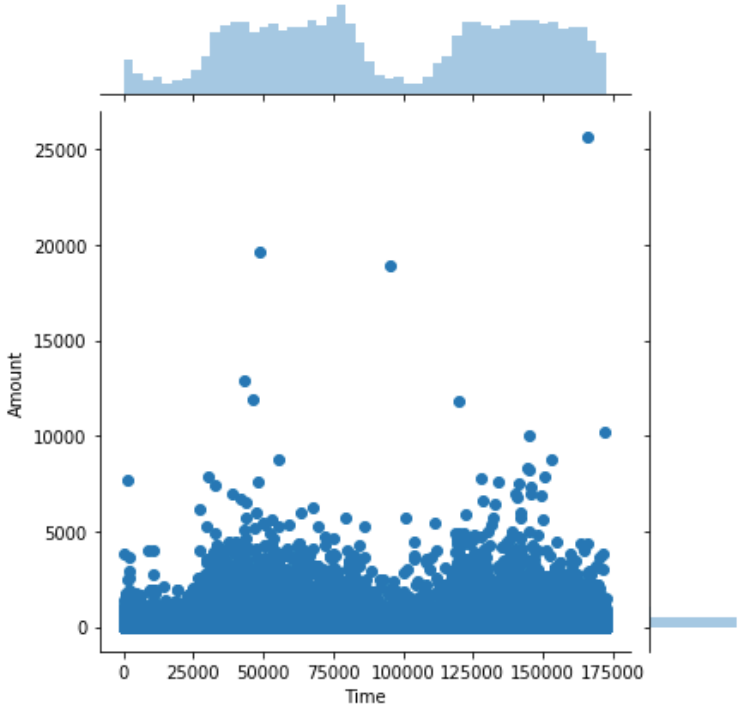
The dataset contains 31 numerical features. The first 28 features are labelled V1, V2 to V28 and these are assumed derived from principal components obtained with Principal Components Analysis of the raw/ original data.

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 more and if analysed directly model will be biased.

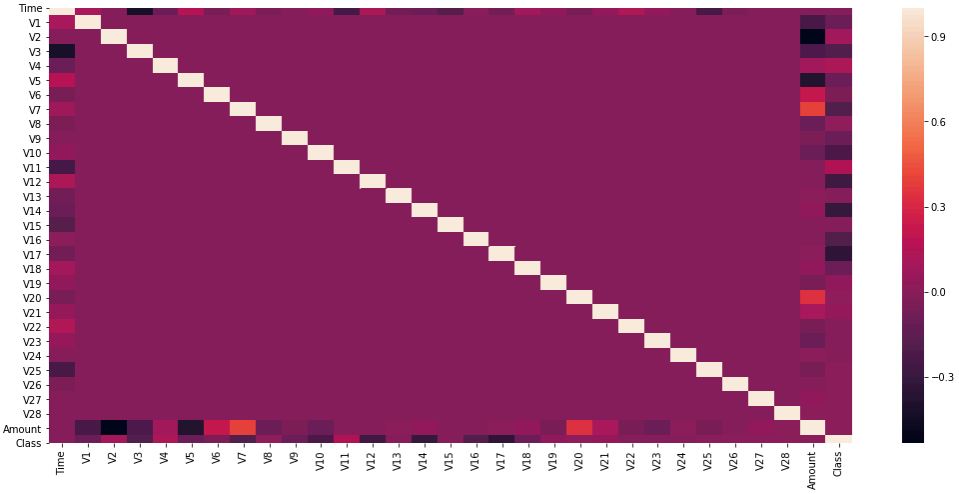



```
# The classes are heavily skewed we need to solve this issue later.
print('No Frauds', round(df['Class'].value_counts()[0]/df.shape[0] * 100,2), '% of the dataset')
print('Frauds', round(df['Class'].value_counts()[1]/df.shape[0] * 100,2), '% of the dataset')
```

```
No Frauds 99.83 % of the dataset
Frauds 0.17 % of the dataset
```



Correlation

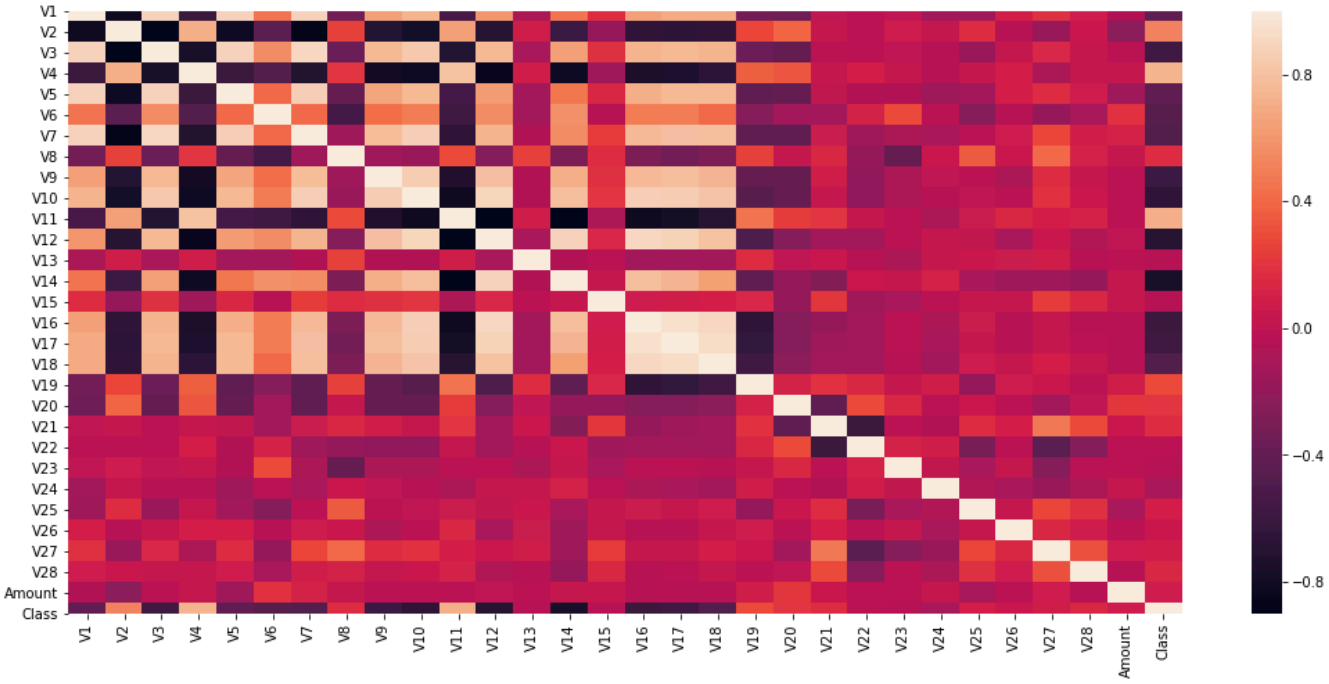


5. Modelling

Before we proceed, lets try baselining the model.
Step 1: Run a Logistic Regression model on as is data - baseline the model and scores.
Step 2: Now we implement SMOTE and Minmax scaller,

- SMOTE will increase the minority class, this will have equal number of fraud and genuine transaction

Step 3: Visualise and check if we have any correlation



Step 4: Run minmax scaller will support to have values columns values between -1 and 1

Step 5: Run a Naive predictor, in this algorithm we assume all values is zeros than what will be the accuracy and f1 score of the model

```
Naive predictor accuracy: 0.998
Naive predictor f1-score: 0.000
```

Steps 6: Run the model with no hyper parameter tuning to check the accuracy and f1 score to check the against the baseline

	Model Name	Accuracy	F1 Score	Recall	Precision	F1 beta
11	XGBClassifier	0.999587	0.845238	0.755319	0.959459	0.808231
5	RandomForestClassifier	0.999571	0.838323	0.744681	0.958904	0.799649
9	BaggingClassifier	0.999571	0.838323	0.744681	0.958904	0.799649
10	ExtraTreesClassifier	0.999539	0.826347	0.734043	0.945205	0.788225
3	KNeighborsClassifier	0.999508	0.812121	0.712766	0.943662	0.770796
8	SVC	0.999460	0.808989	0.765957	0.857143	0.791878
4	DecisionTreeClassifier	0.999301	0.755556	0.723404	0.790698	0.742857
12	AdaBoostClassifier	0.999254	0.728324	0.670213	0.797468	0.704819
0	Logistic Regression - Baselining - Train	0.999174	0.666667	0.553191	0.838710	0.617916
1	Logistic Regression - Baselining - Test	0.999159	0.697318	0.579618	0.875000	0.646802
6	GradientBoostingClassifier	0.999127	0.640523	0.521277	0.830508	0.588725
2	LogisticRegression	0.999063	0.581560	0.436170	0.872340	0.515474
7	GaussianNB	0.978228	0.099803	0.808511	0.053184	0.150564

We can see from above image - 8 models have worked better than the baseline. Difference in accuracy between top 4 models is 0,000048 - not much too compare but we need to keep in mind that the dataset is biased hence we should also take F1 score in consideration. Difference between the top 4 F1 score is 0,018891. Since all top 4 models accuracy and F1 range is in very narrow range hence, implemented hyper-parameter tuning on all the 4 models and tested the model with test data.

Model was trained on training data and tested in validation set. We will use the test data on hyper-parametered models on all the top 4 models

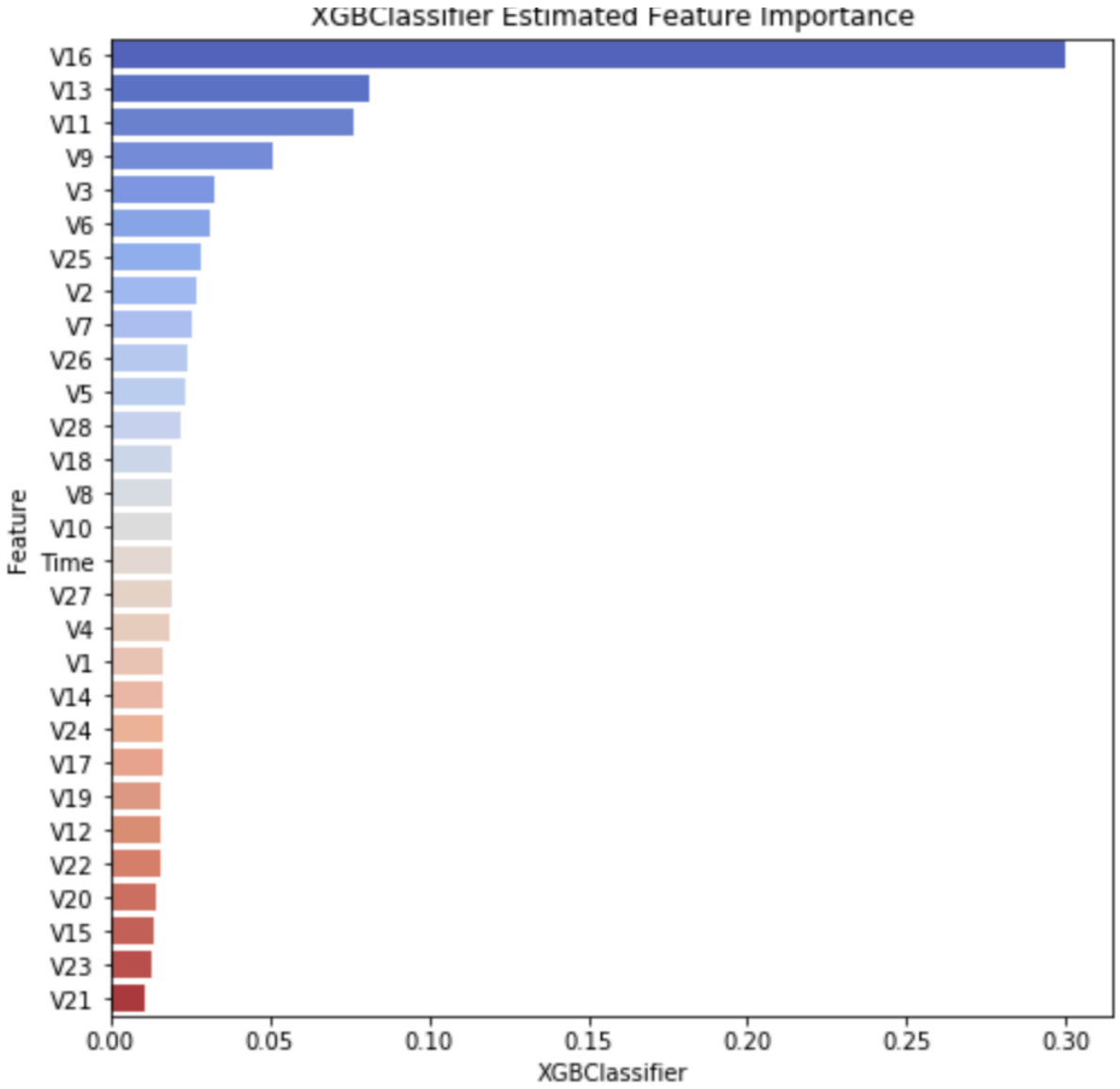
	Model Name	Accuracy	F1 Score	Recall	Precision	F1 beta
0	XGBClassifier	0.999587	0.845238	0.755319	0.959459	0.808231
1	RandomForestClassifier	0.999571	0.838323	0.744681	0.958904	0.799649
2	BaggingClassifier	0.999571	0.838323	0.744681	0.958904	0.799649
13	ExtraTreesClassifier - Test Set	0.999564	0.858131	0.789809	0.939394	0.830500
14	XGBClassifier - Test Set	0.999564	0.858131	0.789809	0.939394	0.830500
15	Random Forest - Test Set	0.999564	0.858131	0.789809	0.939394	0.830500
3	ExtraTreesClassifier	0.999539	0.826347	0.734043	0.945205	0.788225
4	KNeighborsClassifier	0.999508	0.812121	0.712766	0.943662	0.770796
5	SVC	0.999460	0.808989	0.765957	0.857143	0.791878
16	BaggingClassifier - Test Set	0.999436	0.810036	0.719745	0.926230	0.772751
6	DecisionTreeClassifier	0.999301	0.755556	0.723404	0.790698	0.742857
7	AdaBoostClassifier	0.999254	0.728324	0.670213	0.797468	0.704819
8	Logistic Regression - Baselining - Train	0.999174	0.666667	0.553191	0.838710	0.617916
9	Logistic Regression - Baselining - Test	0.999159	0.697318	0.579618	0.875000	0.646802
10	GradientBoostingClassifier	0.999127	0.640523	0.521277	0.830508	0.588725
11	LogisticRegression	0.999063	0.581560	0.436170	0.872340	0.515474
12	GaussianNB	0.978228	0.099803	0.808511	0.053184	0.150564

As we can see from above image, ExtraTreeClassifier, XGBClassifier and Random Forest have scored the same on test data set.

Since **XGB** is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. Suggestion is to go for XGB model for production deployment.

6. Conclusion

The importance of each feature is highlight in following image



7. Reflection

This project can be broken down into phases based on my encounter;

I. Getting the dataset - this was the easiest phase for me as the dimension of the data had already been reduced into Principal Components.

II. Exploratory data analysis - as the data set was clean, not many activity had to be done except getting familiarizing with the data.

III. Model Selection – we had skewed data and choosing right model was one of the most challenging task with running the model on multiple models and choosing the best models for hyper-parameter training.

8. Improvement

Work with more enhanced model with multi layer models to improve the model score.