Machine Learning Engineer Nanodegree

Capstone Project: Credit Card Fraud Detection Model

Akbarali Shaikh July 19, 2020

Table of Content

1. Project Overview

The banking industry works on credit and income from the interest which can be in a form of mortgage, other types of loan, or from Credit Card. As the cost of having credit cards reduced it got broadly accepted in society and many people started having a 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 the banking industry and it is plaguing financial industries for years, some countries it is less problematic than others

Solving this problem could bring about a 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.

2. Problem Statement

Each record has multiple columns and in our dataset, there are 31 columns including classification column which confirms if the record is genuine or a fraud. In real-time when the record is asked to be classified it is not realistically with a human eye to do classification.

The objective is to create an ML model that can classify the transaction in real-time with zero human intervention. In the real world, we have higher genuine transactions than fraudulent so is a case with our dataset.

3. Datasets and Inputs

The dataset used for this project is hosted on data.world. Dataset is normalized and all columns names are changed before it is shared (due

to confidentiality). All features provided would be used in building the model.

The dataset contains 31 numerical features including record classification column means classifying the record as genuine or fraudulent. The first 28 features are labeled V1, V2 to V28, and these are assumed derived from principal components obtained with Principal Components Analysis of the raw/original data.

4. Solution Statement

Dataset is unbalanced with more than 99% of the transaction been genuine vs fraudulent transaction. If we run the ML model on as is data it is the highly likely model will be biased.

To solve this problem, first, we will baseline the model on as is data then we will implement SMOTE function which will create more fraudulent data on train data set to train the model, post which we will implement min-max scaler to normalize the data and so the entire dataset is on the same scaler. Post which we will apply naive_predictor_accuracy assuming all entire data set is zero then what is a baseline accuracy of the model.

Once we have a baseline and cleaned data we will run a model on the model on different machine learning algorithms and it will be tested on validation set data. Post which we use hyper meter tuning on the best-chosen model.

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

5. Benchmark Model

Checked the details at scores but there are limited to know benchmark details but similar work is done as part of academics and competition

and looks like, Accuracy, and F1 score are 2 important metrics frequently used in the classification of fraudulent transaction

6. Evaluation Metrics

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

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

Accuracy is an important measure of a 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.

7. Project Design

As we start with the project we will follow the following steps to choose the right ML model for our problem statement

1. We will load data from source (data.world)
2. Post loading in a data frame, we need to familiarize ourself with a dataset
3. Visualize the data
4. We will then start with ML
    1. We will start with importing relevant libraries for ML
    2. Splitting the data into train/validation/test sets
    3. Baselining the model with Naive Predictor - assuming if all predicting value is of one category and also on Logistic regression model
    4. Since we now have a baseline, we will try improving the model by
        - Balancing the dataset via SMOTE function and
        - Normalizing the dataset
        - **Note**: this will be applied only on the test dataset
    5. Now we will run the model on the different algorithm (we have a binary classification problem statement), hence we will try the following models
        - Logistic Regression
        - KNN Classifier
        - Decision Tree Classifier
        - Random Forest Classifier
        - Gradient Boosting Classifier
        - GaussianNB
        - SVC
        - BaggingClassifier
        - ExtraTreesClassifier
        - XGBClassifier
        - AdaBoostClassifier
    6. Apply Hyper-parameter tuning on top models

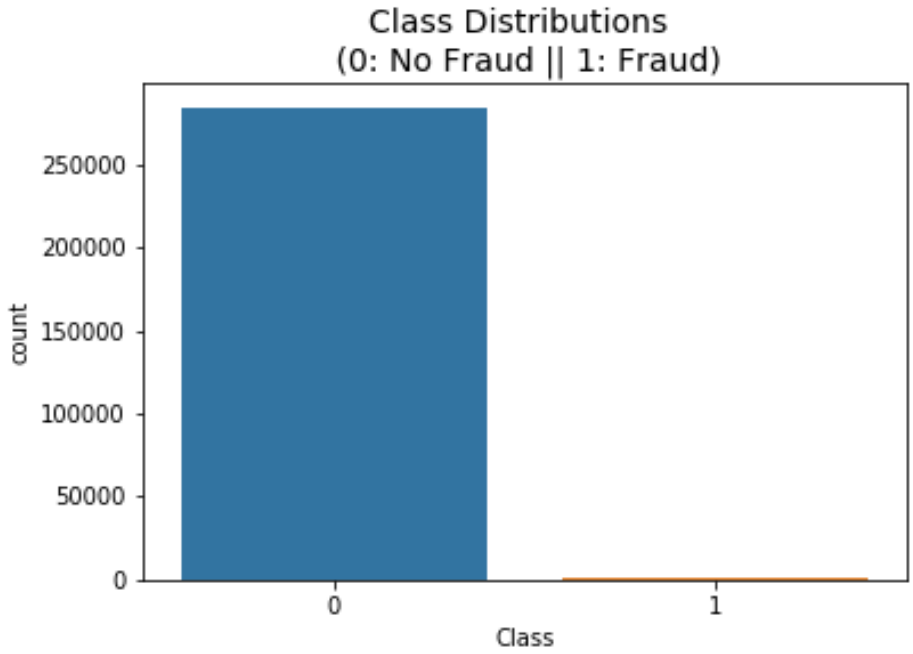
- Observation: Since we have an Imbalanced dataset we need to not only check for accuracy but also for Fi Score.

8. Analysis

Data Exploration
Due to confidentiality, the dataset used is anonymized. The dataset contains 31 numerical features., the first 28 features are labeled V1, V2 to V28, and these are assumed derived from principal components obtained with Principal Components Analysis of the raw/original data.

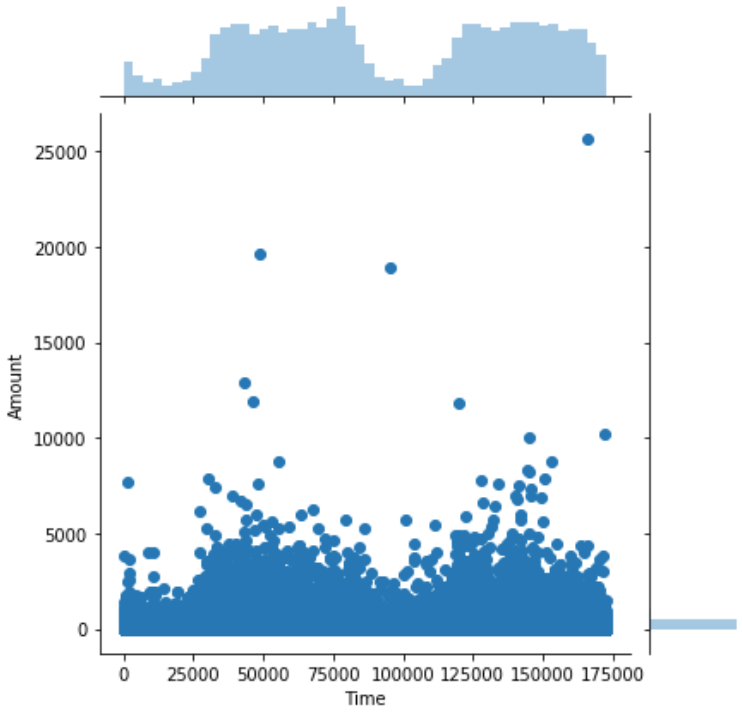
The dataset presents 492 frauds resource out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for .17% of all transactions.

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 analyzed directly model will be biased.

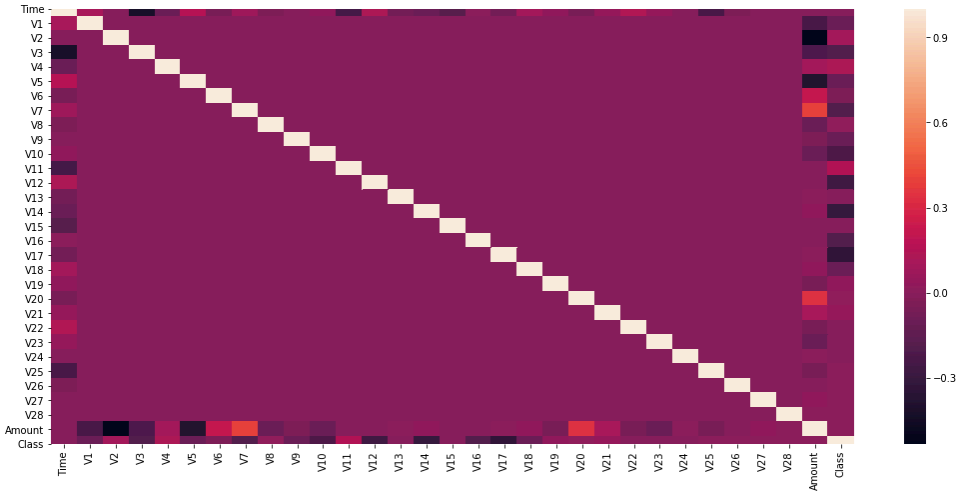



```
# The classes are heavily skewed we need to solve this issue later.
print('No Frauds', round(df['Class'].value_counts()[0]/df.shape[0] * 100,2), '% of the dataset')
print('Frauds', round(df['Class'].value_counts()[1]/df.shape[0] * 100,2), '% of the dataset')
```

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



Correlation



9. Modelling

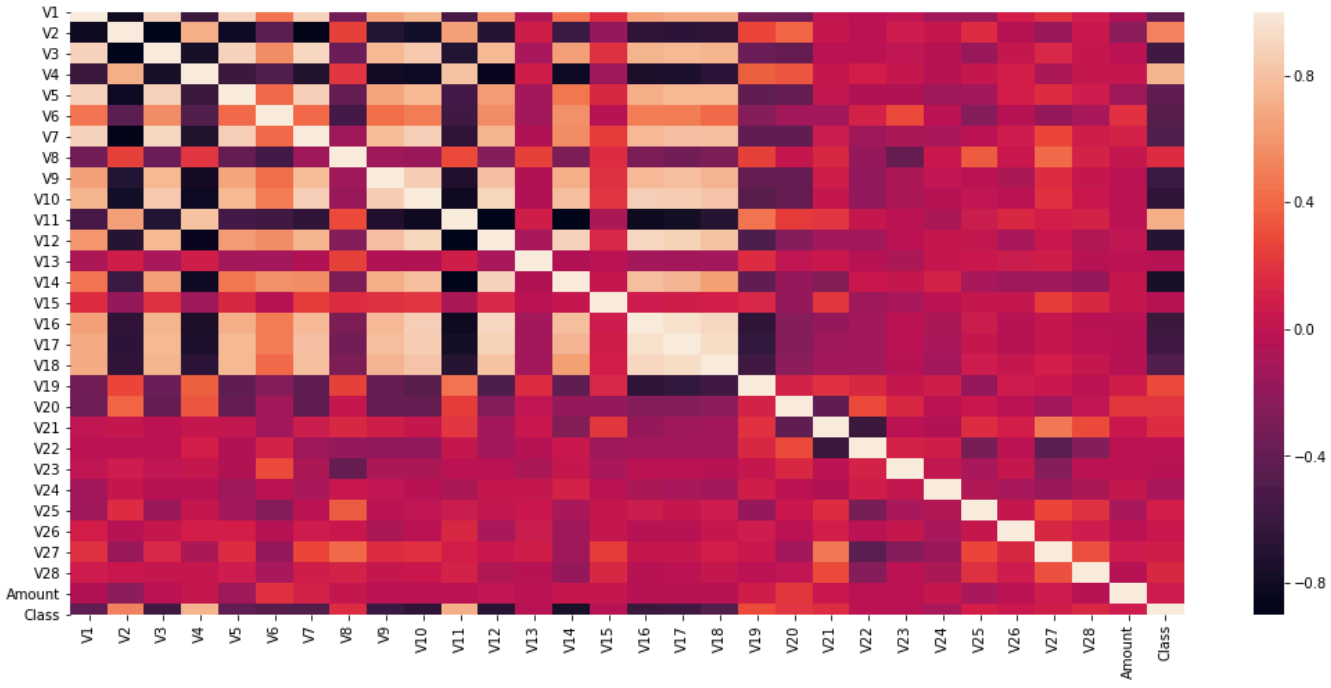
Before we proceed, let's try baselining the model.
Step 1: Split the dataset into Train / Validation and Test datasetÂ
Step 2: Run a Logistic Regression model on as is data - baseline the model and scores.

	Model Name	Accuracy	F1 Score	Recall	Precision	F1 beta
0	Logistic Regression - Baselining - Train	0.999174	0.666667	0.553191	0.83871	0.617916

Step 3: Now we implement SMOTE and Minmax scaler,Â

- SMOTE will increase the minority class, this will have help to balance the dataset and have an equal number of fraud and genuine transactions. This needs to be applied to the training dataset.
- Run min-max scaler to have similar scales on all columns values between -1 and 1 to avoid biases during the ML

Step 4: Visualise and check if we have any correlation, if we check the above correlation plot, the plot is very different and below plot shows columns having relation.



Step 5: Run a Naive predictor, in this algorithm we assume all values are 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 different models with no hyper-parameter tuning to check the accuracy and f1 score to check them 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 the above image - 8 models have worked better than the baseline logistic model. The difference in accuracy between the 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 consider the F1 score. The difference between the top 4 F1 scores is 0,018891. Since all top 4 models, accuracy, and F1 range are in very narrow range hence, let's try implementing hyper-parameter tuning on all the 4 models and see the best model with test data.

The ML model was trained on training data and tested in the validation set. Now we will tune the model with hyper-parameterized models and test the model on the test dataset. Post-hyper-parameter tuning - model execution we see the following data

	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

We executed on 4 models of which ExtraTreeClassifier, XGB, and Random Forest have scored the same on test data set.

Since **XGB** is more scalable and accurate on gradient boosting ML models and it has proven to push the limits of computing power for boosted tree algorithms off model performance and computational speed. Hence, the suggestion is to go for the XGB model for production deployment.

10. Conclusion

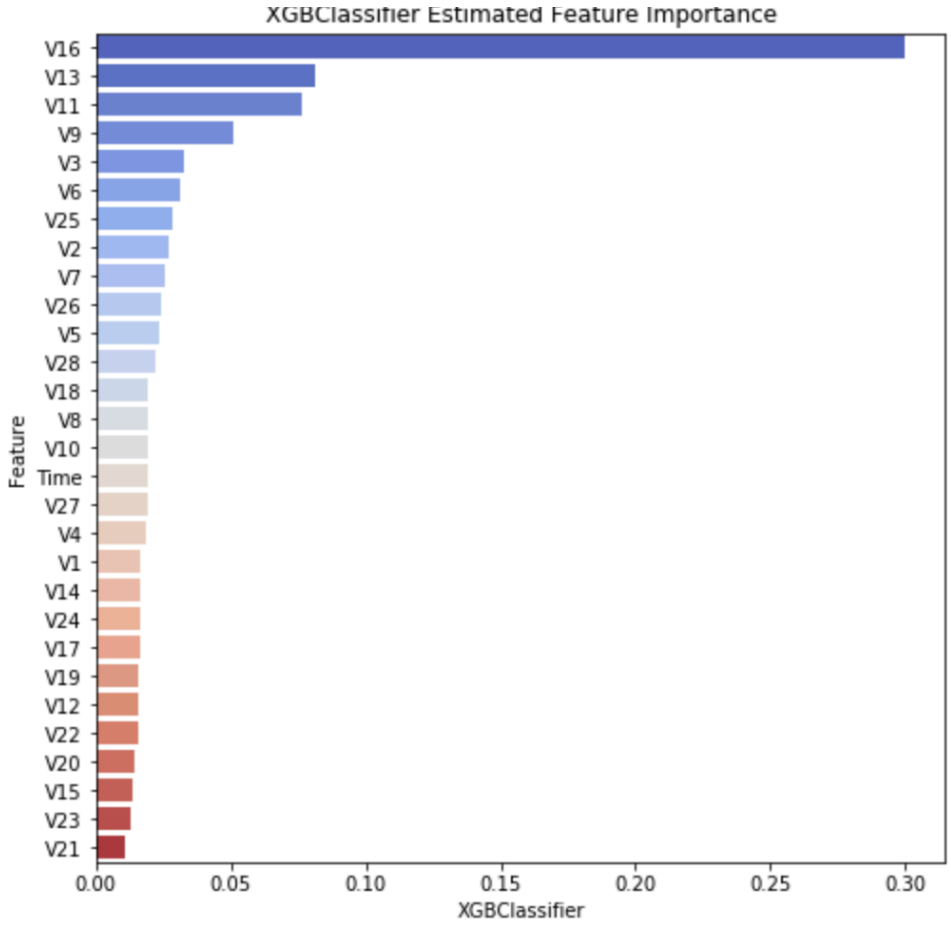
The importance of each feature is the highlight in the following image

	feature	Extra Tree
0	V16	0.142127
1	V13	0.116976
2	V11	0.112680
3	V15	0.076344
4	V9	0.066386

	feature	XGB
0	V16	0.299744
1	V13	0.081401
2	V11	0.076274
3	V9	0.051112
4	V3	0.032577

	feature	Random Forest
0	V16	0.299744
1	V13	0.081401
2	V11	0.076274
3	V9	0.051112
4	V3	0.032577

Above, images are the top 5 important features, influencing the model. Features V16, V13, v11, v9 are common in all 3 models and v16 contributing been the highest influencer. Common features in Extratree are contributing to 43,81 % whereas XGM and Random Forest are contributing to 50,85%. Complete feature importance list of XGB is as follows:



11. Reflection

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

1. Getting the dataset - this was the easiest phase for me as the dimension of the data had already been reduced into Principal Components.
2. Exploratory data analysis - as the data set was clean, not many activities had to be done except getting familiar with the data.
3. Model Selection - we had skewed data and choosing the right model was one of the most challenging tasks with running the model on multiple models and choosing the best models for hyper-parameter training.

12. Improvement

We can improve the model score by using unsupervised learning multi-layer models.