Credit Card Fraud Detection

Introduction

Ever since starting my journey into data science, I have been thinking about ways to use data science for good while generating value at the same time. Thus, when I came across this data set on Kaggle dealing with credit card fraud detection, I was immediately hooked. The data set has 31 features, 28 of which have been anonymized and are labeled V1 through V28. The remaining three features are the time and the amount of the transaction as well as whether that transaction was fraudulent or not. Before it was uploaded to Kaggle, the anonymized variables had been modified in the form of a PCA (Principal Component Analysis). Furthermore, there were no missing values in the data set. Equipped with this basic description of the data, let's jump into some exploratory data analysis.

The Dataset

- Our dataset, first of all, has a tremendous imbalance regarding its two transaction classes, fraudulent and non-fraudulent.
- Most of the transactions were **Non-Fraud** (99.83%) of the time, while **Fraud** transactions occurs (017%) of the time in the dataframe.
- There are no **"Null"** values, so we don't have to work on ways to replace values.
- The transaction amount is relatively **small**. The mean of all the mounts made is approximately USD 88.
- Other than that, when looking at our columns, we can see that, for the sake of privacy, 28 out of 31 columns have been anonymized and normalized.
- The description of the data says that all the features went through a PCA transformation (Dimensionality Reduction technique) (Except for time and amount)
- Keep in mind that in order to implement a PCA transformation features need to be previously scaled. (In this case, all the V features have been scaled or at least that is what we are assuming the people that develop the dataset did.)
- For more information on the dataset you can check out its Kaggle page.

The Problem

So, our main problem is: how to make our model learn with data that doesn't make that possible. It's clear that the best choice for our model, in order to be **accurate** would be to label every single example as non-fraudulent, and it would still get more than 99% accuracy!

It's clear that the problem relies not solely on accuracy, but also recall.

Equipped with this basic description of the data, let's jump into some exploratory data analysis.

Exploratory Data Analysis

- 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.
- 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.
-
- The Feature 'Class' is the response or target variable and it takes value 1 in case of fraud and 0 otherwise.
-
- 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.
- For Fraudulent Transaction Mean transaction is around 122 and standard deviation is around 256 and Maximum Transaction was 2125 and minimum was 0.
- For Normal Transaction Mean transaction is around 88 and standard deviation is around 250.Maximum Transaction was 25691 and minimum was 0.

- 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.
- There are 113 fraud transactions for just one dollar and 27 fraud transactions for 99.99 dollars. Also, there are 27 fraud transactions for zero amount.
- The reason for zero transaction can be the zero Authorization which is an account verification method for credit cards that is used to verify a cardholder's information without charging the consumer.
- I was not able to find any useful observation/trend from The plots of both hours and minutes.
- For some of the features we can observe a good selectivity in terms of distribution for the two values of Class: V4, V11 have clearly separated distributions for Class values 0 and 1, V12, V14, V18 are partially separated, V1, V2, V3, V10 have a quite distinct profile, whilst V20-V28 have similar profiles for the two values of Class and thus not very useful in differentiation of both the classes.

Data Cleaning and Preprocessing

Outlier Removing

- As we saw, that amount column has a extreme outliers so it necessary to remove them as they can effect the model's performance. We will used Interquartile range to detect outliers which removes anything below the lower limit (25 percentile) and anything above upper limit (75 Percentile).
- Note that, the data we have for fraudulent cases is very low so we wanna keep our cutoff a bit high so as avoid removing much of the fraud cases. Here, as the data is skewed (kind of exponential) so having high cutoff will help us. Let's take the cutoff value as 5.0 instead of 1.5 which is usually used.
- Total Number of outliers found were 11366 out of them 41 were in Fraudulent class and 11325 were found in Normal Class. Percentage of Fraud Amount outliers were 0.36%.

Feature Scaling

- As the amount column is highly skewed so it will be better to apply log transoformation as it can result in nearly normal distribution which is suited for most of the algorithms.

Dimensionality Reduction and Clustering

- Imbalanced data is a problem in supervised learning problems which can result in a high bias towards majority class. Therefore to balance the data we use SMOTE.
- After the data is balanced we use some data visualization technique.
- For the dimensionality reduction I will use t-SNE. It is state-of-the-art in visualising high dimensional data and can help us understand how our model will perform or how separable the data is using clusters. It outperforms PCA.
- As expected, t-SNE is performed well, separating both the classes. The clusters are not perfectly separated but it is able to seperate almost 80% of the sample data.

Splitting the data

As our data is imbalanced so we will not use train_test_split and instead we will use stratified split which will take the representative of respective populations i.e Fraud Transactions and Normal Transactions.

Here splitting is done before handling the imbalance data so as to reduce the chances of duplicacy in the test set.

Modelling

One thing we should keep in mind is that we might get very high accuracy but we should focus on optimising our f1_score and recall as we want to perform better on fraud cases as they are the most important.

- **Logistic Regression(On Imbalanced Dataset):-**
    - Logistic Regression did not work well on the imbalanced data as the recall value came out to be just 43% . Here we will not focus on Accuracy score as this is an highly imbalanced dataset.
    - We get a total of 55 errors or misclassification

- We can see that Recall value is pretty low which implies that False Negative is high, 51 Fraud Transaction were classified as Normal Transactions.
- By HIGHER Recall value we identify that the actual fraud transactions which were Incorrectly Predicted as Normal Transaction(i.e. False Negative) is LOW.

- **Random Forest Classifier(On Imbalanced Dataset): ;-**
  - Random Forest did work pretty well on the imbalanced data as the recall value came out to be 67% . Here we will not focus on Accuracy score as this is an highly imbalanced dataset.
  - We get a total of 30 errors or misclassification which are all the False Negative with 0 False Positive.
  - We can see that Recall value is better than the Logistic Regression which implies that False Negative is low, 30 Fraud Transaction were classified as Normal Transactions.
  - By HIGHER Recall value we identify that the actual fraud transactions which were Incorrectly Predicted as Normal Transaction(i.e. False Negative) is LOW.
- **OverSampling:-**
  - **Random OverSampling:-**
    - Random Oversampling randomly replicates the minority class instances. It has some drawbacks.
    - After Random OverSampling the size becomes 436784 for both the classes.
    - Then applied Logistic Regression.
    - Within Random Oversampling we can observe that we get a good recall score of 88% and on changing the ratio of sampling strategy from default ratio auto(resample all classes but the majority class) to 0.4 (desired ratio of the number of samples in the minority class over the number of samples in the majority class after resampling) we get a recall score of 86%
    - Random Oversampling did work pretty well on the balanced data via oversampling technique as the recall value came out to be 88% .
    - We get a total of 441(auto) and 195(0.4) errors or misclassification.
    - we can see that Recall value is pretty high which implies that False Negative is low, just 11 and 13 Fraud Transaction were classified as Normal Transactions.

- we see a good decrease in the amount of False positive {430 -> 182} by changing the sampling strategy from auto to 0.4 which is good as it increases the True Negative classification without any significant increase in False Negative.
- By HIGHER Recall value we identify that the actual fraud transactions which were Incorrectly Predicted as Normal Transaction(i.e. False Negative) is LOW.

- **SMOTE**
  - To avoid the over-fitting problem, Chawla et al. (2002) propose the Synthetic Minority Over-sampling Technique (SMOTE).This method is considered a state-of-art technique and works well in various applications.
  - This method generates synthetic data based on the feature space similarities between existing minority instances.
  - In order to create a synthetic instance, it finds the K-nearest neighbors of each minority instance, randomly selects one of them, and then calculates linear interpolations to produce a new minority instance in the neighborhood.
  - Within SMOTE we can observe that we get a good recall score of 87% along with an accuracy of 99.18% but we care about recall score more

  - SMOTE did work pretty well on the balanced data via oversampling technique as the recall value came out to be 87% .

  - We get a total of 448 errors or misclassification

  - Here we can see that Recall value is pretty high which implies that False Negative is low, 12 Fraud Transaction were classified as Normal Transactions

  - We can observe a high amount of False positive(436)

  - By HIGHER Recall value we identify that the actual fraud transactions which were Incorrectly Predicted as Normal Transaction(i.e. False Negative) is LOW
  -

- **Adaptive Synthetic Sampling:-**
  - ADASYN is, like SMOTE, a synethtic data generator. It also generates data through a K-Neighbours perspective, but it also takes into account the density of a certain neighbour of K-Neighbours.
  - ADASYN did work pretty well on the balanced data via oversampling technique as the recall value came out to be 91% .
  -
  - ► We get a total of 2223 errors or misclassification
  -
  - ► Here we can see that Recall value is pretty high which implies that False Negative is low, Only 8 Fraud Transaction were classified as Normal Transactions
  -
  - We can observe a very high amount of False positive(2215)
  -
  - By HIGHER Recall value we identify that the actual fraud transactions which were Incorrectly Predicted as Normal Transaction(i.e. False Negative) is LOW
  -
  - These are the oversampling techniques used here. We can clearly see that we made progress. Jumping from 0 to 91% recall is pretty good with algorithms that didn't get too complex.
  -
  - The drawback that follows every single one of these Oversampling techniques is that our happiness may be founded upon overfitting.But since this type of data is itself rare, therefore we can hope that this model did a pretty good job.

- **UnderSampling:-**
- Opposed to oversampling, undersampling changes the proportion between the classes by removing examples from the majority class until we have a more even distribution.
  - **Random UnderSampling:-**
    - As the name suggests, Random Undersampling removes example randomly from the dataset. It follows the same logic as Random Oversampling. Again, we will be using imblearn.

        - Within Random Undersampling we get the same recall value as compared to the ADASYN i.e. 91.1%
        - By randomly removing examples, we might be throwing away valuable data. Data that is representative of the real problem that we are trying to solve. We simply do not have any mechanisms that are built in the algorithm to deal with the possible loss of information.
- **Outlier Detection:-**
    - Up until now, we've been dealing with the imbalance of our dataset from a perspective of more data. If we get more data, then our lives will be easier.
    - When we consider our dataset, the fraud transactions consist less than 1% of total entries. Therefore it is, by definition, an outlier. Therefore, we could apply algorithms that are really good at detecting an outlier and then label its predictions of outliers as fraudulent transactions.
    - We'll approach Isolation Forest Algorithm that can do this
      .
    - **Isolation Forest:-**
        - The Isolation Forest algorithm isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The logic argument goes: isolating anomaly observations is easier because only a few conditions are needed to separate those cases from the normal observations. On the other hand, isolating normal observations require more conditions. Therefore, an anomaly score can be calculated as the number of conditions required to separate a given observation.

        - The that the algorithm constructs the separation is by first creating isolation trees, or random decision trees. Then, the score is calculated as the path length to isolate the observation.
        - Within Isolation Forest we get the recall value of 90% on an imbalanced dataset
        - We get a total of 2994 errors or misclassification

- Here we can see that Recall value is pretty high which implies that False Negative is low, Only 9 Fraud Transaction were classified as Normal Transactions

- We can observe a very high amount of False positive(2810). We can reduce this further by using neural network to increase the accuracy for detecting fraud and non-fraud transactions
- By HIGHER Recall value we identify that the actual fraud transactions which were Incorrectly Predicted as Normal Transaction(i.e. False Negative) is LOW
- This was an Outlier Detection techniques used here. We can clearly see that we made progress by increasing the recall value from 43% to 90% on an completely imbalanced data.