Summer of Code 2022
Credit Card Fraud Detection

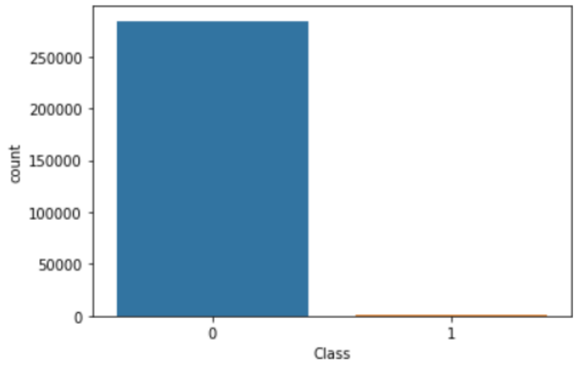
Introduction

Credit Card Fraud Detection with Machine Learning is a process of data investigation by a Data Science team and the development of a model that will provide the best results in revealing and preventing fraudulent transactions. This is achieved through bringing together all meaningful features of card users' transactions, such as Date, User Zone, Product Category, Amount, Provider, Client's Behavioral Patterns, etc. The information is then run through a subtly trained model that finds patterns and rules so that it can classify whether a transaction is fraudulent or legitimate. All big banks like Chase use fraud monitoring and detection systems.

About Dataset :

The dataset 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. 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 a value 1 in case of fraud and 0 otherwise.

Evaluation Metrics:

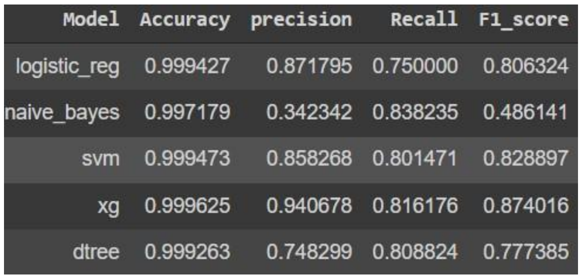
Given the class imbalance ratio, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

EDA and Data Pre-processing:

After using .skew() function came to know that some column values have skewed data. So we used the cube root function to remove the skewness of the data. The Amount column is not standardized so we use the StandardScaler() from sklearn to standardize the data.

Building the model:

So first create a function in which we had defined our model. First, we split the dataset as train and test with the split percentage of 0.3. then we called the LogisticRegression(), GaussianNB(), SVC(), XGBClassifier(), DecisionTreeClassifier() Classifiers from the sklearn library, and then fitted, predicted the data, and print the confusion matrics, classification report, and the precision, recall, F1 score and the accuracy of the model.



Model	Accuracy	precision	Recall	F1_score
logistic_reg	0.999427	0.871795	0.750000	0.806324
naive_bayes	0.997179	0.342342	0.838235	0.486141
svm	0.999473	0.858268	0.801471	0.828897
xg	0.999625	0.940678	0.816176	0.874016
dtree	0.999263	0.748299	0.808824	0.777385

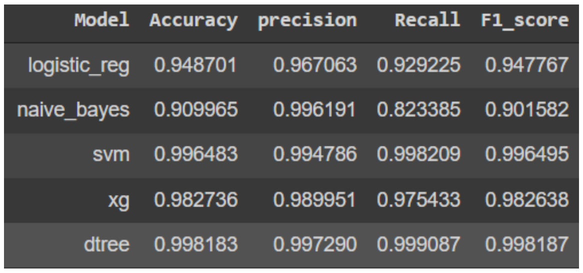
It's clear from the results that our model accuracy is very good but as it is an imbalance dataset accuracy is not a good parameter to judge it so we should judge the model by its F1_score. But, the F1_score is very bad as compared to the accuracy of the model.

Using SMOTE :

As our dataset is imbalanced the models are not able to predict well. So we need to balance out our data set. For this, we will be using the SMOTE: Synthetic Minority Oversampling Technique.

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.

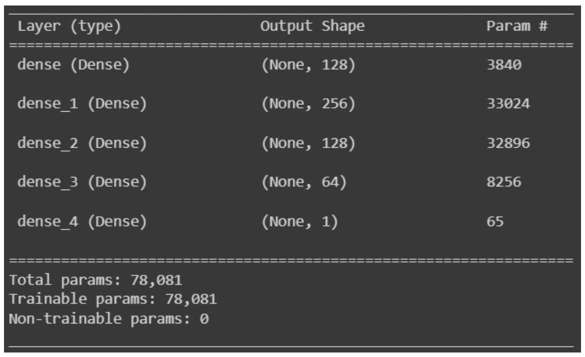
After using the smote technique the F1_score was improved a lot.



Model	Accuracy	precision	Recall	F1_score
logistic_reg	0.948701	0.967063	0.929225	0.947767
naive_bayes	0.909965	0.996191	0.823385	0.901582
svm	0.996483	0.994786	0.998209	0.996495
xg	0.982736	0.989951	0.975433	0.982638
dtree	0.998183	0.997290	0.999087	0.998187

Now, our model is performing, even more, better but let's used Deep Learning to improve the score again.

Using Deep Learning:



```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               3840

 dense_1 (Dense)             (None, 256)               33024

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 1)                 65

=================================================================
Total params: 78,081
Trainable params: 78,081
Non-trainable params: 0
```

Using the Deep neural network our score improved even more Accuracy: 0.999695, Precision: 0.999392, Recall: 1.000000, F1_score: 0.999696. We achieved Recall = 1 i.e we had not predicted any fraud as the nonfraud transaction.