**Credit Card Fraud Detection**

**Problem Statement:**

Fraudulent activities have increased severalfold, with around 52,304 cases of credit/debit card fraud reported in FY'19 alone. Due to this steep increase in banking frauds, it is the need of the hour to detect these fraudulent transactions in time to help consumers as well as banks, who are losing their credit worth each day.

Every fraudulent credit card transaction that occurs is a direct financial loss to the bank as the bank is responsible for the fraud transactions as well it also affects the overall customer satisfaction adversely.

The aim of this project is to identify and predict fraudulent credit card transactions using machine learning models.

**Approach:**

The approach to the problem can be divided into below parts:

1. **Data Understanding, Data Preparation and EDA**

First look at the data used here from the Kaggle dataset suggests that it is highly imbalanced in nature. The positive class (frauds) account for only 0.172% of all transactions.

|  |  |  |
| --- | --- | --- |
| Class | 0 | 1 |
| Count | 284315 | 492 |

Class is the target variable which we must predict where 0 is normal transaction and 1 is fraudulent transaction.

Features V1, V2, … V28 are the principal components derived 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.

As the PCA transformed variable are Gaussian, normalisation is not required.

We can do the EDA to analyse the data like correlation, boxplots etc for outliers & skewness in the data.

The imbalance in the data can dealt through the techniques below:

* SMOTE is a process where you can generate new data points, which lie vectorially between two data points that belong to the minority class.
* ADASYN is like SMOTE, with a minor change i.e. the number of synthetic samples that it will add will have a density distribution.

1. **Model Selection and Model Building:**

We will start building the model with the train-test split. We need to find which ML model works good with the imbalance data and have better results on the test data.

* Logistic regression works best when the data is linearly separable and needs to be interpretable.
* KNN is also highly interpretable, but not preferred when we have a huge amount of data as it will consume a more computing power.
* The decision tree model is the model of choice when we want the output to be intuitive, but they tend to overfit if left unchecked.
* KNN is a simple, supervised machine learning algorithm used for both classification and regression tasks.
* In Gradient Boosted machines/trees. newly added trees are trained to reduce the errors (loss function) of earlier models.
* XGBoost is an improved version of gradient boosting, with additional features like regularization and parallel tree learning algorithm for finding the best split.
* We will start with the Logistic regression model with the different value of regularisation and hyper param tuning, then go to the decision tree and so on etc. We will also try ensemble models and use voting classifier, bagging, boosting etc on the best performing individual models to find the best model.

1. **Hyperparameter Tuning:**

* When the data is imbalanced or less, it is best to use K-Fold Cross Validation for evaluating the performance when the data set is randomly split into ‘k’ groups.
* Stratified K-Fold Cross Validation is an extension of K-Fold cross-validation, in which we rearrange the data to ensure that each fold is a good representative of all the strata of the data.
* When you have a small data set, the computation time will be manageable to test out different hyperparameter combinations. In this scenario, it is advised to use a grid search.
* But, with large data sets, it is advised to use a randomized search because the sampling will be random and not uniform. In our case, we will use the Stratified K-Fold Cross Validation as the dataset is not huge.

1. **Model Evaluation:**

* We will use AOC and ROC metric for highly imbalanced dataset, rest all fails. ROC have better false negative than the false positives.
* ROC-Curve = Plot between TPR and FPR
* The threshold with highest value for TPR-FPR on the train set is usually the best cut-off. We should not use the confusion matrix as the performance metrics as well as they have internally defined hard threshold of 0.5.
* We also can’t completely rely on the precision, recall and F1-score for now as they also have their strings attached of some threshold value. ROC curve takes into cognizance of all the possible threshold values.
* The ROC curve is used to understand the strength of the model by evaluating the performance of the model at all the classification thresholds. Because the ROC curve is measured at all thresholds, the best threshold would be one at which the TPR is high and FPR is low, i.e., misclassifications are low.

1. **Benefit Analysis:**

* Depending on the use case, we must account for what we need: high precision or high recall.
* For banks with smaller average transaction value, we would want high precision because we only want to label relevant transactions as fraudulent.
* For every transaction that is flagged as fraudulent, it needs to be verified with the customer. When precision is low, such tasks are a burden because the human element must be increased.
* For banks having a larger transaction value, if the recall is low, i.e., it is unable to detect transactions that are labelled as non-fraudulent. So, consider the losses if the missed transaction was a high-value fraudulent one, for e.g., a transaction of $10,000?
* So here, to save banks from high-value fraudulent transactions, we have to focus on a high recall in order to detect actual fraudulent transactions.
* We need to determine how much profit or dollar/rupee value we are saving with our best selected model.