PREDICTION OF CREDIT CARD FRAUD

Problem Statement:

A credit card is one of the most used financial products to make online purchases and payments. Though the Credit cards can be a convenient way to manage your finances, they can also be risky. Credit card fraud is the unauthorized use of someone else's credit card or credit card information to make purchases or withdraw cash.

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

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.

The aim of this project is **to build a classification model to predict whether a transaction is fraudulent or not.**

Approach:

The approach used for solving the problem are:

Step -1: Data Understanding and preparation

Load the data to understand it. Data understanding is critical since we will select the subset of features to carry out model training, and it includes types of data and its patterns.

The target variable is **Class** that we need to predict – 0 means normal transaction and 1 means fraudulent transaction.

The principal components obtained from PCA are features from V1 to V28. Features Time and Amount are not PCA transformed. The feature Time indicates the seconds elapsed between each transaction and the feature Amount indicates the transaction amount.

Step -2: Exploratory Data Analysis

Histogram is used to compare each variable's data distribution pattern and skewness of the data. Make necessary data adjustments to avoid any problems while we train the model.

We use transformation (Yeo-Johnson) to mitigate and check the skewness in the data.

Step -3: Class imbalances

Studied whether data is highly imbalanced between the fraudulent and non-fraudulent. Four techniques to balance the data using various methods:

- **Under sampling** balances the dataset by reducing the size of the abundant class. This method is used when quantity of data is sufficient.

- **Oversampling** is used when the quantity of data is insufficient. It tries to balance dataset by increasing the size of rare samples.

- **SMOTE (Synthetic Minority Over Sample Technique)** is a process where you can generate new data points, which lie vectorially between two data points that belong to the minority class.

- **ADASYN (Adaptive Synthetic)** is like SMOTE, with a minor change i.e. the number of synthetic samples that it will add will have density distribution. The aim here is to create synthetic data for minority examples that are harder to learn, rather than easier ones.

We will use ADASYN techniques to lower the bias introduced by class imbalance and adaptively shift the classification decision boundary towards complex examples.

Step -4: **Data Modeling and model selection**

We will start building the model with the train-test split. (At least 100 class 1 rows should be there in the test split), used the 70-30 ratio to split the data here.

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 lot of computation.
- **The decision tree model** is the first 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. The k value in KNN should be an odd number because you must take the majority vote from the nearest neighbours by breaking the ties.
- In **Gradient Boosted machines/trees**. newly added trees are trained to reduce the errors (loss function) of earlier models.
- **XGBoost** is an extended version of gradient boosting, with additional features like regularization and parallel tree learning algorithm for finding the best split.

We will use two classification models (RadomForest and XGBoost) for the current study. For XGBoost, we will identify the number of trees based on the accuracy levels. We will not use KNN since the data set are more than 10K (computation time is increased if data sets are more than 10K). The decision tree gives an interpretation of the flow chart. Hence, it is widely used, but we do not know when to stop building trees and tend to overfit. We will use parameters such as confusion matrix, accuracy, precision, recall, and F-score, threshold dependent. Since data is imbalanced, we will also perform a deep dive into the ROC curve data to identify the threshold value (above the threshold value is fraudulent and below the threshold is not dishonest). We will calculate the F1 score, Precision, and Recall.

Step -5: Hyperparameter Tunings the model

At this time, we will have the best understanding of the type of data we have and the kind of model we were going to build. After model building, our next step will be either hyperparameter tunings (CV or Cross-validation) or grid-search cross-validation or K-Fold or stratified K-Fold, or train validation and test split. This step is essential to improve accuracy, AUC, and lower misclassification error.

- When the data is imbalanced or less, it is better 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 really huge

Step -6: Model Evaluation

We will use **sklearn.metrics.roc_auc_score** for this as AOC and ROC metric in sklearn is used as the metric for highly imbalanced data-set, 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.

Evaluate the model based on **AUC -ROC score**, through which we will define the threshold value. We will calculate the F1 Score, Precision, and Recall. This will be key for banks to represent the business strategy to bring down the Fraud.

Step -7: Cost-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, you can add the human element to verify whether the transaction was done by calling the customer. However, 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 must focus on a high recall to detect actual fraudulent transactions.**