Credit Card Fraud Detection

Problem Statement:

Fraudulent activities have increased several fold, with around 52,304 cases of credit/debit card fraud. Due to this step increase in banking frauds, it is the need of the hour to detect these fraudulent transactions in time in order 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.

Data Acquisition and Exploration:
Data Understanding, Data Preparation and EDA

First look at the data that we used here 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 have to predict where 0 is normal transaction and 1 is fraudulent transaction.

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 dependent cost- sensitive learning.

Since the PCA transformed variable are already Gaussian, there is no need for normalization. We can start with the basic EDA like correlation, boxplots etc. for outliers.

The initial step involves acquiring a suitable credit card transaction dataset.

Data Cleaning: Identifying and handling missing values, outliers, and inconsistencies. Data cleaning techniques like imputation or removal of outliers may be necessary.

Data Visualization: Employing visualizations like histograms, boxplots, and scatterplots to explore the distribution of features and identify potential relationships between them.

Feature Engineering: Creating new features from existing ones that might improve model performance. Examples include calculating the average transaction amount per merchant or the frequency of transactions within a specific timeframe.

Understanding Class Imbalance:
A critical aspect of credit card fraud data is class imbalance. Legitimate transactions typically far outnumber fraudulent ones. This imbalance can lead to models that prioritize predicting the majority class (legitimate transactions) and potentially miss fraudulent activities. Here, we explore techniques to address class imbalance:

Class Distribution Analysis: Calculating the percentage of fraudulent and legitimate transactions in the dataset. A significant disparity indicates class imbalance.

Oversampling: Replicating data points from the minority class (fraudulent) to increase its representation and balance the class distribution. Techniques like SMOTE (Synthetic Minority Oversampling Technique) can generate synthetic data points for the minority class.

Under sampling: Removing data points from the majority class (legitimate) to match the size of the minority class. However, this approach reduces the overall dataset size and might affect model performance.

Model Selection and Building:
We will evaluate various classification models commonly used for fraud detection. Here's an overview of some potential candidates:

Logistic Regression: A linear model suitable for binary classification problems. It's interpretable, efficient to train, and performs well when the data is linearly separable.

Random Forest: An ensemble method that combines predictions from multiple decision trees, leading to improved accuracy and robustness to overfitting. It can handle non-linear relationships between features and the target variable.
XGBoost: A powerful gradient boosting algorithm known for its effectiveness in various classification tasks. XGBoost uses decision trees as base learners and leverages a gradient boosting framework to achieve high accuracy and handle complex data structures.

For each chosen model, the following steps will be followed:

Data Preprocessing: Apply the necessary data cleaning and feature engineering techniques identified during data exploration.

Data Splitting: Divide the preprocessed data into training and testing sets. The training set is used to train the model, and the testing set is used to evaluate its performance on unseen data. Common splitting ratios are 80/20 (80% for training and 20% for testing).

Model Training: Train the model on the training set. Hyper parameters, which control the model's behavior, will be optimized using techniques like GridSearchCV or RandomizedSearchCV. This ensures the model is configured for optimal performance.

Hyperparameter Tuning:
Hyperparameters are crucial parameters that influence the learning process and model performance. Examples include the number of trees in a Random Forest or the learning rate in XGBoost. Tuning these parameters is essential to optimize the model's ability to learn from the data and generalize well to unseen transactions. Here's how we will approach hyperparameter tuning:

RandomizedSearchCV: This techniques systematically evaluate different combinations of hyperparameter values and identify the configuration that yields the best performance on a validation set (a subset of the training data). This approach helps prevent overfitting and ensures the model generalizes well to unseen data.

Evaluation Metrics: Metrics like accuracy, precision, recall, and AUC- ROC curve will be used to assess model performance on the validation set during hyperparameter tuning.

Accuracy: Measures the overall proportion of correctly classified transactions (fraudulent and legitimate).

Precision: Measures the proportion of transactions identified as fraudulent that are actually fraudulent.

Recall: Measures the proportion of actual fraudulent transactions that are correctly identified by the model.

AUC-ROC Curve (Area Under the Receiver Operating Characteristic Curve): A visualization that helps assess the model's ability to discriminate between fraudulent and legitimate transactions. A higher AUC indicates better performance.