**PHASE-4 SUBMISSION TEAM 4**

**Credit Card Fraud Detection**

**Feature Engineering:**

Feature engineering is a critical step in preparing the data for training machine learning models. It involves creating new features or transforming existing ones to improve model performance and fraud detection accuracy.

some feature engineering techniques and ideas for credit card fraud detection:

**Transaction Frequency Features:** Transaction frequency features help identify unusual transaction patterns based on the number of transactions within specific time intervals. In this case, we calculate the number of transactions in the last hour for each cardholder.

**CODE:** df['transactions\_last\_hour'] = df.groupby('cardholder')['Time'].transform(lambda x: x.diff().lt(3600).cumsum())

**Transaction Amount Features:** Transaction amount features are important to understand the spending behavior of cardholders. We compute statistics like the average, maximum, and minimum transaction amounts for each cardholder.

**CODE:**

df['avg\_transaction\_amount'] = df.groupby('cardholder')['Amount'].transform('mean')

df['max\_transaction\_amount'] = df.groupby('cardholder')['Amount'].transform('max')

df['min\_transaction\_amount'] = df.groupby('cardholder')['Amount'].transform('min')

**Time-Based Patterns:** Time-based features capture patterns in transaction times. By extracting information such as the hour of the day, day of the week, and time since the last transaction, we can identify suspicious temporal behavior.

**CODE:**

df['hour'] = pd.to\_datetime(df['Time']).dt.hour

df['day\_of\_week'] = pd.to\_datetime(df['Time']).dt.dayofweek

df['time\_since\_last\_transaction'] = df.groupby('cardholder')['Time'].diff()

**Cardholder Behavior:** Features related to cardholder behavior provide insights into spending habits. We calculate the average number of transactions per day and the average amount spent per transaction.

**CODE:**

df['avg\_transactions\_per\_day'] = df.groupby(['cardholder', ‘day\_of\_week']) ['Amount'] .transform('count')

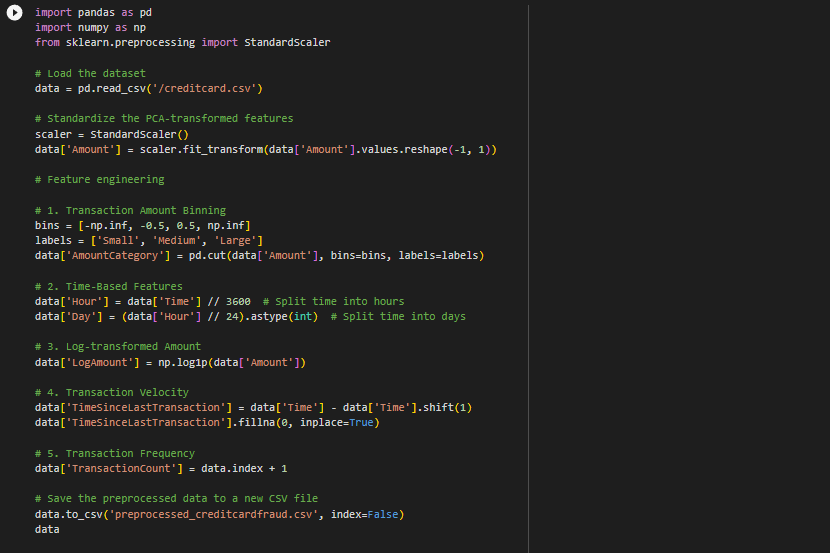
df['avg\_amount\_per\_transaction'] = df.groupby(['cardholder'])['Amount'].transform('mean')

**Merchant-Based Features**: Merchant-based features focus on the characteristics of merchants. These features include the average transaction amount at each merchant and the number of transactions in specific merchant categories.

**CODE:**

df['avg\_transaction\_amount\_merchant'] = df.groupby ('MerchantID') ['Amount']. transform('mean')

df['num\_transactions\_merchant\_category'] = df.groupby ('MerchantCategory') ['Amount']. Transform('count')



**Model Training:**

Model training involves selecting the right machine learning algorithms, tuning hyperparameters, and training the models to detect credit card fraud effectively.

**Data Collection:**

* Gather a dataset of historical credit card transactions. This dataset should include information about each transaction, including transaction amount, timestamp, and features that you've engineered during the feature engineering process.
* The dataset should also include labels indicating whether each transaction is legitimate (non-fraudulent) or fraudulent.

**Data Preprocessing:**

* Clean and preprocess the data, handling missing values and outliers appropriately.
* Normalize or scale numerical features to ensure they have a similar range. This step is essential for many machine learning algorithms.

**Data Splitting:**

* Divide the dataset into training, validation, and test sets. Common splits include 70-80% for training, 10-15% for validation, and the remaining 10-15% for testing. The validation set is used to fine-tune hyperparameters, while the test set is used to evaluate the model's performance.

**Choosing an Algorithm:**Select a suitable machine learning algorithm for credit card fraud detection. Common choices include:

* Logistic Regression
* Random Forest
* Gradient Boosting (e.g., XGBoost, LightGBM)
* Neural Networks (Deep Learning)
* Anomaly Detection algorithms (e.g., Isolation Forest, One-Class SVM)

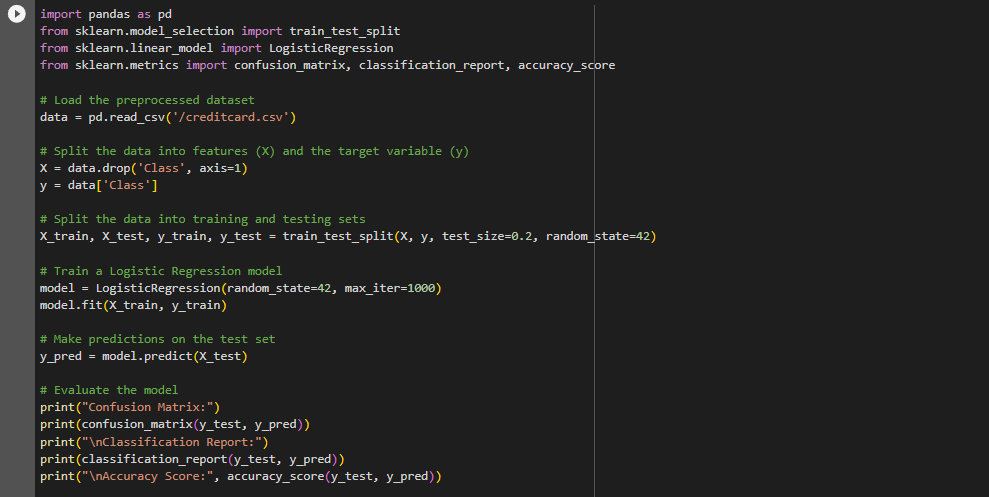
We using Logistic Regression and Random Forest

**Model Selection:** Choose machine learning models suitable for credit card fraud detection. Common options include Logistic Regression, Random Forest, and Gradient Boosting models.

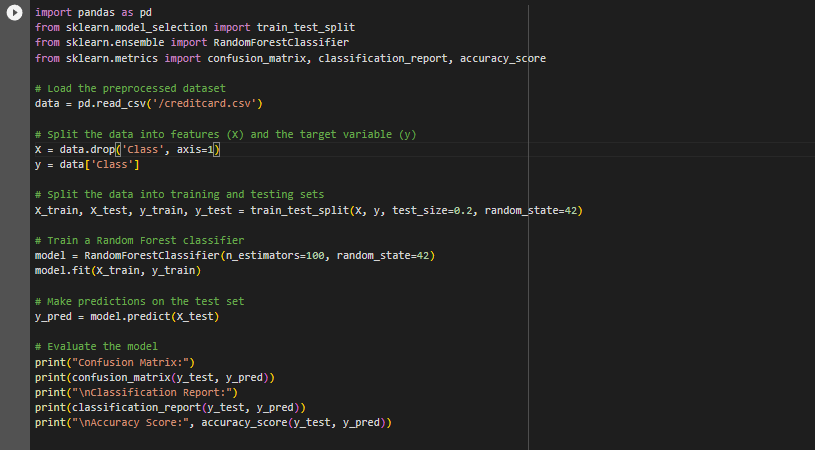
**Hyperparameter Tuning:** Tune the hyperparameters of selected models to optimize their performance. Techniques like Grid Search or Random Search can be used.

**Model Training:** Train the chosen algorithm on the training dataset. The model learns to distinguish between legitimate and fraudulent transactions based on the labeled data.

**The below example using Logistic regression:**

****

**The below example using Random Forest:**

****

**Evaluation:**

Evaluating the model's performance is crucial to assess its effectiveness in detecting credit card fraud.

**Accuracy:** Accuracy measures the proportion of correctly classified transactions. However, in the presence of imbalanced data, accuracy can be misleading because it can be high even when the model fails to detect most of the fraud cases.

**Precision (Positive Predictive Value**): Precision measures the fraction of true positive predictions (correctly identified fraud cases) out of all positive predictions (total cases predicted as fraud). A higher precision indicates that the model is better at not misclassifying legitimate transactions as fraud.

Precision = True Positives / (True Positives + False Positives)

**Recall (Sensitivity, True Positive Rate):** Recall measures the proportion of actual fraud cases that were correctly identified by the model. A higher recall means that the model is better at detecting most of the fraud cases.

Recall = True Positives / (True Positives + False Negatives)

**F1-Score:** The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, taking both false positives and false negatives into account. It is especially useful when there is an imbalance between the classes.

F1-Score = 2 \* (Precision \* Recall) / (Precision + Recall)

**Evaluation Metrics:** Define the evaluation metrics to assess the performance of the models. Common metrics include precision, recall, F1-score, and the area under the Receiver Operating Characteristic (ROC-AUC) curve.

**Model Performance:** Present the results of model evaluation on the test data, including relevant metrics and visualizations like confusion matrices and ROC curves.

**Confusion Matrix:** Examining the confusion matrix, which includes True Positives, False Positives, True Negatives, and False Negatives, provides a more detailed view of model performance.

**CODE:**

