Fraud Transaction Detection Using Transactions Dataset

1. Objective

The goal of this project is to build a machine learning model for detecting fraudulent transactions. The model will utilize a dataset of transaction records to predict whether a transaction is fraudulent (TX_FRAUD). The project includes steps like data loading, preprocessing, feature engineering, model training, hyperparameter tuning, and evaluation.

2. Data Acquisition and Preprocessing

2.1 Loading Data

- The project begins by loading multiple .pkl files containing transaction data from a specified directory. All relevant files are loaded and combined into a single DataFrame for further processing.
- A directory is also extracted from a zipped dataset, which includes the transaction files.

2.2 Feature Engineering

- Several new features are created to enrich the dataset for model training:
    - **Date Features:** The TX_DATETIME column is converted to datetime format, and additional features like the day and hour of the transaction are extracted.
    - **Amount Flag:** A binary flag is created to indicate whether a transaction amount exceeds a specific threshold (220 in this case).
    - **Fraud Count:** A rolling sum of fraudulent transactions per terminal is calculated for the past 28 days.
    - **Average Customer Spend:** The average transaction amount per customer is calculated and added as a feature.
    - Missing values are filled with 0 to handle incomplete data.

2.3 Target and Features

- The model is trained to predict TX_FRAUD, a binary column indicating whether a transaction is fraudulent or not.
- The features (X) are selected by dropping columns like TRANSACTION_ID, TX_DATETIME, and TX_FRAUD from the dataset, while the target variable (y) is the TX_FRAUD column.

3. Model Training and Hyperparameter Tuning

3.1 Handling Class Imbalance

- Given that fraud detection datasets are often imbalanced (with far fewer fraudulent transactions), Synthetic Minority Over-sampling Technique (SMOTE) is applied to

balance the dataset by generating synthetic samples of the minority class (fraudulent transactions).

3.2 Model Selection

- A **Random Forest Classifier** is selected for training the model. This is a powerful ensemble learning method suitable for classification tasks.

3.3 Hyperparameter Tuning

- A grid search with cross-validation is performed to find the best hyperparameters for the model. The key parameters tuned include:
    - Number of trees (n_estimators)
    - Maximum depth of the trees (max_depth)
    - Minimum number of samples required to split a node (min_samples_split)
    - Minimum number of samples required at a leaf node (min_samples_leaf)
    - Class weight for handling class imbalance (class_weight)

4. Model Evaluation

4.1 Performance Metrics

- The model's performance is evaluated on the test set using several metrics:
    - **Accuracy**: Measures the proportion of correctly classified transactions.
    - **ROC-AUC Score**: Provides an indication of the model's ability to distinguish between the classes (fraudulent and non-fraudulent).
    - **Classification Report**: Includes precision, recall, F1-score, and support for each class.
    - **Confusion Matrix**: Displays the true positives, false positives, true negatives, and false negatives.

4.2 Model Saving

- If the model achieves an accuracy of 90% or higher, it is saved as a .pkl file for future use. This file contains the trained model that can be loaded to make predictions on new data.

5. Predictions and Results

5.1 Full Dataset Prediction

- After training and hyperparameter tuning, the model is applied to the entire dataset to predict the fraud status of each transaction.
- The predicted fraud labels are added as a new column (PREDICTED_TX_FRAUD) to the dataset.

5.2 Results Output

- The results, including the actual and predicted fraud labels, are saved to a CSV file (fraud_detection_results.csv) for further analysis or reporting.

6. Optional Model Loading and Testing

- The saved model is loaded using the joblib library to demonstrate how it can be reused for future predictions on new data.
- Sample predictions are generated on a small subset of the test data and printed for inspection.