Credit Card Fraud Detection

Objective

The goal of this project is to develop a machine learning model that can classify credit card transactions as fraudulent or non-fraudulent. The model aims to minimize false positives while maximizing fraud detection accuracy. By using various machine learning algorithms, including XGBoost and TabNet, the system identifies patterns in transaction data that distinguish fraudulent activity from legitimate transactions.

Dataset:

The dataset used in this project contains transaction details such as:

- **amount**: The amount of the transaction.
- **category**: The type of item or service purchased.
- **merchant**: The merchant involved in the transaction.
- **user_id**: The unique identifier for the user making the transaction.
- **timestamp**: The time at which the transaction occurred.
- **fraud**: Target column indicating whether the transaction is fraudulent (1) or legitimate (0).

The dataset is imbalanced, with a significantly smaller number of fraudulent transactions compared to legitimate ones.

Installation:

Prerequisites

To run this project, you need the following libraries:

Install dependencies by running: **pip install -r requirements.txt**

Run Flow:

Follow the steps below to run the notebook files sequentially:

1. Run train_data.ipynb

This notebook handles the data preprocessing:

- Cleans the dataset by handling missing values and converting categorical features.
- Scales numerical features and splits the data into training and testing sets.
- Apply techniques like SMOTE to handle class imbalance.

2. Run model_XGBOOST.ipynb

In this notebook, the XGBoost model is trained:

- A machine learning pipeline is constructed to train the XGBoost model using the preprocessed data.
- The model's performance is evaluated using metrics such as accuracy, confusion matrix, and ROC-AUC score.

3. Run model_tagnet_1.ipynb

In this notebook, the TabNet model is trained:

- A deep learning model (TabNet) is trained using the preprocessed data.
- The performance of TabNet is compared with the XGBoost model in terms of accuracy, precision, recall, and ROC-AUC score.
- Graphs related to TabNet's performance, such as training and validation loss curves, are generated to visualize the model's learning process.

Technologies Used:

1. XGBoost

Benefit: XGBoost is a powerful gradient boosting algorithm known for its high efficiency and performance. It works well with imbalanced datasets, provides fast model training, and supports parallel processing, making it a great choice for large datasets like credit card transactions.

2. TabNet

Benefit: TabNet is a deep learning model specialized for tabular data. It handles categorical features and missing values better than traditional neural networks. It also automatically learns which features to focus on, providing state-of-the-art performance on tabular data tasks, including fraud detection.

The trained TabNet model is saved in ZIP format for easy sharing and future use.

3. SMOTE (Synthetic Minority Over-sampling Technique)

Benefit: SMOTE is an over-sampling technique used to balance class distributions by generating synthetic samples of the minority class. It helps prevent the model from being biased toward the majority class (non-fraudulent transactions), ensuring better performance for fraud detection.

4. Scikit-learn

Benefit: Scikit-learn provides easy-to-use and efficient tools for data preprocessing, model building, and evaluation. It is used for tasks such as feature scaling, model evaluation, and cross-validation, helping to ensure a clean and reproducible workflow.

5. PyTorch

Benefit: PyTorch is a powerful deep learning framework that enables the development of complex neural networks. It supports dynamic computation graphs, making it flexible for experimentation. PyTorch is used in the TabNet model to handle deep learning aspects of the project.

Code Structure:

- **train_data.ipynb**: Data preprocessing, handling missing data, scaling features, and balancing the dataset.
- **model_XGBOOST.ipynb**: Training and evaluating the XGBoost model.
- **model_tagnet_1.ipynb**: Training and evaluating the TabNet model, including generating performance graphs.
- **requirements.txt**: List of Python dependencies required for the project.
- **README**: Documentation describing the project.
- **TabNet_model.zip**: Saved TabNet model in ZIP format for future use.

Graphs:

- **Confusion Matrix** (confusion_matrix.png): Visual representation of the true positives, false positives, true negatives, and false negatives.
- **Feature Importance** (feature_importance.png): A plot showing the relative importance of each feature used in the model.
- **Precision-Recall Curve** (pr_curve.png): A plot showing the trade-off between precision and recall for various thresholds.
- **ROC Curve** (roc_curve.png): A plot illustrating the trade-off between the true positive rate and false positive rate for various thresholds.
- **Training History** (training_history.png): A plot showing the training and validation loss curves during model training for TabNet.

Model Performance:

Here’s the performance of the models after training:

XGBoost Model:

- ROC-AUC: 0.98
- Precision: 0.95
- Recall: 0.92
- F1-Score: 0.93

TabNet Model:

- ROC-AUC: 0.97
- Precision: 0.93
- Recall: 0.91
- F1-Score: 0.92

Note: The performance of the models, particularly the TabNet model, can be improved further with higher epochs. However, this would require significant computational power and time, especially when training on large datasets.