Report:

Title: Credit Card Fraud Detection Project

Author: Kushagra Jaiswal

Date: [24-5-24]

Introduction

The Credit Card Fraud Detection Project aims to develop a predictive model to detect fraudulent transactions, thereby helping financial institutions minimize losses and protect their customers.

Data Overview

The dataset consists of credit card transactions, with 284,807 entries and 31 features. Before preprocessing, the data contains no missing values. The dataset was imbalanced, with fraudulent transactions accounting for only 0.17% of the total transactions.

Exploratory Data Analysis (EDA)

Key insights from EDA:

The distribution of transaction amounts is heavily right-skewed. Fraudulent transactions tend to have higher transaction amounts on average compared to legitimate transactions. Time does not seem to have a significant impact on the likelihood of a transaction being fraudulent. Outliers were handled using the Interquartile Range (IQR) method, removing transactions beyond 1.5 times the IQR.

Data Preprocessing

Missing values: No missing values were found in the dataset. Data transformation: The 'Amount' feature was scaled using a standard scaler. SMOTE for class balancing: Synthetic Minority Over-sampling Technique (SMOTE) was applied to address the class imbalance. 5. Modeling

Two models were used: Logistic Regression and Random Forest.(there were others models used in RAW code but after these two objective was fulfilled so not using them in Pipeline)

Logistic Regression: After hyperparameter tuning, the best parameters were C=1.0 and penalty='l2'. Random Forest: The optimal hyperparameters were n_estimators=100 and max_depth=10. 6. Performance Evaluation

Outlier Handling Section for the Report

Why Outliers Were Not Handled

Model Robustness:

Tree-based algorithms such as Random Forest and Gradient Boosting are inherently robust to outliers. These models are less sensitive to the presence of outliers because they use the median value to split nodes, which is not affected by extreme values. Performance Metrics:

The performance metrics (accuracy, precision, recall, F1-score) were satisfactory without handling outliers. This indicates that the model was able to perform well despite the presence of outliers. Specifically, the Random Forest model achieved an F1-score of 0.86 for the minority class, indicating good performance in detecting fraud. Retaining Data Integrity:

Outliers can sometimes contain important information, especially in the context of fraud detection. Removing outliers might lead to a loss of potentially valuable insights. Cross-Validation Stability:

Cross-validation scores were consistent and high, suggesting that the model generalizes well across different subsets of the data, even with outliers present.

When to Consider Handling Outliers

Model Sensitivity:

For models that are sensitive to outliers, such as linear regression or k-means clustering, handling outliers is crucial.

Data Quality:

If outliers are due to data entry errors or measurement errors, they should be corrected or removed. Exploratory Data Analysis:

During EDA, if outliers are found to disproportionately affect the data distribution or model performance, it might be necessary to handle them.

[After removing the outliers random model efficiency dropped quite much so I have not removed the outliers as model working quite well as it is]

Performance metrics for both models:

Logistic Regression: Accuracy: 99.92% Precision: 88.89% Recall: 59.18% F1-score: 71.23% Random Forest: Accuracy: 99.95% Precision: 91.67% Recall: 77.69% F1-score: 83.13% Confusion matrices and classification reports are provided in the appendix.

About Deployment

The Credit Card Fraud Detection project aims to deploy a machine learning model as a Flask API for real-time predictions of fraudulent credit card transactions. The model, a random forest classifier, has been trained on historical transaction data to accurately classify transactions as either fraudulent or legitimate.

Deployment Strategy

The deployment strategy involves containerizing the Flask application using Docker, allowing for easy portability and reproducibility across different environments. The following steps outline the deployment process:

Setup Directory Structure:

Ensure the directory structure is organized with the following layout:

```
Credit Card Fraud Detection Project/
├── Models/
│   └── random_forest_model.pkl
├── app.py
├── requirements.txt
└── Dockerfile
```

Flask Application Code:

Develop the Flask API in app.py, which loads the pre-trained model from the Models folder and exposes a /predict endpoint to receive input data for prediction. Requirements File:

Create a requirements.txt file listing the necessary Python packages required for the project, including Flask, joblib, and numpy. Dockerfile:

Create a Dockerfile specifying the Docker image configuration, including setting the working directory, copying project files, installing dependencies, exposing the necessary port, and defining the command to run the Flask application. Build and Run Docker Container:

Build the Docker image using the docker build command. Run the Docker container with the docker run command, mapping port 5000 to enable communication with the Flask API.

Testing (Optional):

Test the deployed API locally using tools like Postman or curl, sending POST requests to http://localhost:5000/predict with sample data.

Future Work

Potential improvements and future research directions include exploring advanced anomaly detection techniques and ensemble methods for further improving model performance.

Conclusion

In conclusion, both Logistic Regression and Random Forest models showed high accuracy in detecting fraudulent credit card transactions. However, Random Forest outperformed Logistic Regression in terms of precision and recall. Further refinement and exploration of advanced techniques could enhance the model's effectiveness in real-world scenarios.The deployment of the Credit Card Fraud Detection API using Docker provides a scalable and efficient solution for real-time fraud detection in credit card transactions.