Credit Card Fraud Detection Using Machine Learning

Abstract **-- This project focuses on developing a machine-learning model for detecting credit card fraud. Using a dataset with an imbalance between fraudulent and non-fraudulent transactions, we applied various preprocessing techniques and evaluated several models to identify the best-performing one. The final model demonstrates significant improvements in detection accuracy and has been deployed for practical use. Machine learning models are used and as a result, Random Forest has the highest accuracy among others i.e., approximately 99% accuracy, and its best fit for the dataset selected.**

TABLE OF CONTENTS

I. INTRODUCTION

The rise in online transactions has led to an increase in credit card fraud, causing significant financial losses. This project aims to develop a machine learning model to effectively detect fraudulent transactions. The dataset used consists of 284,807 transactions from European cardholders, with only 492 identified as fraudulent. The objective is to create a model that accurately identifies fraudulent transactions to help financial institutions reduce losses.

II. LITERATURE REVIEW

Various techniques have been used to detect credit card fraud, including logistic regression, decision trees, and ensemble methods like Random Forest and XGBoost. The challenge lies in handling the highly imbalanced nature of the dataset, where fraudulent transactions are a small fraction of the total. Oversampling techniques like SMOTE (Synthetic Minority Over-sampling Technique) and undersampling are commonly used to balance the dataset. Our approach leverages these techniques to improve model performance.

III. DATA COLLECTION

- **Data Sources:** The dataset was sourced from a public repository on Kaggle, containing anonymized credit card transactions.
- **Data Description:** The dataset includes 31 features. These features are:
    - Time: Number of seconds elapsed between this transaction and the first transaction in the dataset.
    - Amount: Transaction amount.
    - Class: Binary label for fraud detection (0 for non-fraud, 1 for fraud).
    - 28 anonymized features labeled V1 to V28 due to confidentiality issues.
- **Data Cleaning:** The dataset was found to have no missing values. Duplicate entries were removed, and the Amount feature was scaled for better model performance.

IV. EXPLORATORY DATA ANALYSIS (EDA)

EDA involves analyzing the main characteristics of the data, often visualizing them. The steps included:

- **Class Distribution:** Visualization showed that the dataset is highly imbalanced, with only 0.172% fraudulent transactions.
- **Data Visualization:** Various plots were created:

- Class Distribution: Bar and pie charts depicting the imbalance.
- Correlation Heatmap: Shows the correlation between different features.
- Amount Distribution: Histogram of transaction amounts for fraudulent and non-fraudulent transactions.
- Time Distribution: Distribution of transactions over time.
- Fraud vs. Normal Transactions by Time of Day/Hour: Density plots to observe patterns in fraudulent activities.
- Boxplots for Numerical Attributes: Distribution and outliers for numerical features.
- Amount vs. Class Distribution: Scatter plot to visualize the transaction amount against class labels.

V. METHODOLOGY

The methodology involved several steps to build and evaluate the model:

- **Algorithms Used:** The following algorithms were evaluated:
  - Logistic Regression
  - Random Forest
  - Decision Tree
  - XGBoost
- **Handling Imbalance:** Applied SMOTE for oversampling the minority class (fraudulent transactions) and random undersampling of the majority class (non-fraudulent transactions).
- **Evaluation Metrics:** Used to evaluate model performance included:
  - Accuracy: Proportion of correctly identified transactions (both fraudulent and non-fraudulent).
  - Precision: Proportion of true frauds among all detected frauds.
  - Recall: Proportion of detected frauds among all actual frauds.
  - F1 Score: Harmonic mean of precision and recall.
  - AUC-ROC: Area under the Receiver Operating Characteristic curve.

VI. IMPLEMENTATION

- **Data Preprocessing:**
  - Scaled the Amount feature using StandardScaler for better model performance.
  - Applied SMOTE to oversample the minority class and random undersampling to balance the dataset.
- **Model Training:**
  - Trained models using both undersampled and oversampled datasets to compare performance.
  - Performed hyperparameter tuning using RandomizedSearchCV to optimize model parameters.
- **Model Deployment:**
  - Selected the best-performing model based on evaluation metrics.
  - Saved the trained models using joblib for future deployment and practical use.

VII. WORKFLOW

The flowchart figure 1 below provides an overview of the steps involved in the credit card fraud detection process using the best-performing model, Random Forest.

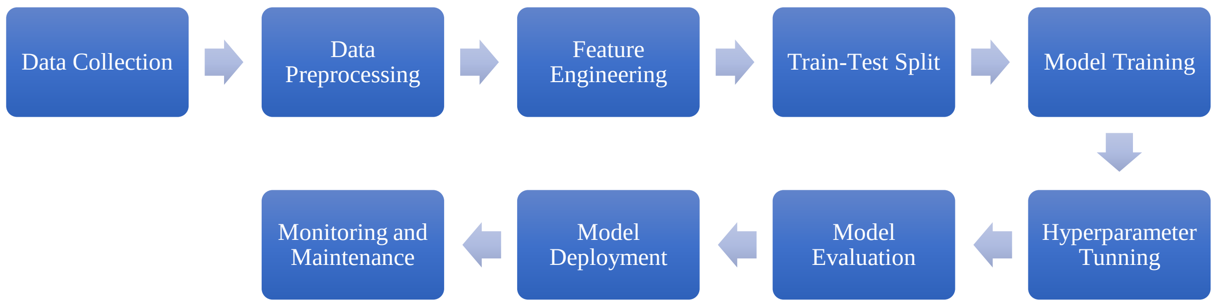

Fig. 1: Workflow Flowchart for Random Forest Model

Detailed Flowchart Steps:

1. **Data Collection:**
   - **Description:** Gather the raw transaction data that will be used for model training.
   - **Details:** This dataset includes features such as transaction time, amount, and anonymized transaction details.

2. **Data Preprocessing:**
   - **Description:** Preprocess the data to make it suitable for model training.
   - **Details:** Handle missing values, remove duplicates, scale numerical features, and encode categorical variables.
3. **Feature Engineering:**
   - **Description:** Enhance the dataset by creating new features and selecting the most relevant ones.
   - **Details:** Techniques such as creating interaction terms and using feature importance to select key features.
4. **Train-Test Split:**
   - **Description:** Split the dataset into training and testing sets.
   - **Details:** Typically, 70-80% of the data is used for training and 20-30% for testing.
5. **Model Training:**
   - **Description:** Train the Random Forest model using the training dataset.
   - **Details:** Use the training data to build multiple decision trees and aggregate their results.
6. **Hyperparameter Tuning:**
   - **Description:** Optimize the model by tuning hyperparameters.
   - **Details:** Use techniques like GridSearchCV or RandomizedSearchCV to find the best parameters for the model.
7. **Model Evaluation:**
   - **Description:** Evaluate the performance of the trained model.
   - **Details:** Use metrics such as accuracy, precision, recall, F1 score, and AUC-ROC on the test data.
8. **Model Deployment:**
   - **Description:** Deploy the best-performing model for practical use.
   - **Details:** Save the model using joblib and integrate it into the transaction processing system.
9. **Monitoring and Maintenance:**
   - **Description:** Continuously monitor the model's performance and update it as necessary.
   - **Details:** Track performance metrics and retrain the model periodically to maintain accuracy.

VIII. RESULTS

Table 1: Results of models trained

Model	Accuracy	Precision	Recall	F1 Score	AUC-ROC
Logistic Regression (Undersampled)	0.926316	0.945055	0.905263	0.924731	0.926316
Logistic Regression (Oversampled)	0.945819	0.972793	0.917294	0.944228	0.945819
Logistic Regression Tuned (Undersampled)	0.942105	0.988372	0.894737	0.939227	0.942105
Logistic Regression Tuned (Oversampled)	0.945837	0.972776	0.917347	0.944249	0.945837
Random Forest (Undersampled)	0.926316	0.945055	0.905263	0.924731	0.926316
Random Forest (Oversampled)	0.999866	0.999733	1.000000	0.999866	0.999866
Random Forest Tuned (Undersampled)	0.921053	0.934783	0.905263	0.919786	0.921053
Random Forest Tuned (Oversampled)	0.999795	0.999590	1.000000	0.999795	0.999795
Decision Tree (Undersampled)	0.905263	0.896907	0.915789	0.906250	0.905263
Decision Tree (Oversampled)	0.998315	0.997632	0.999002	0.998316	0.998315
Decision Tree Tuned (Undersampled)	0.926316	0.935484	0.915789	0.925532	0.926316
Decision Tree Tuned (Oversampled)	0.998164	0.997507	0.998823	0.998165	0.998164
XGBoost (Undersampled)	0.926316	0.955056	0.894737	0.923913	0.926316
XGBoost (Oversampled)	0.999706	0.999412	1.000000	0.999706	0.999706
XGBoost Tuned (Undersampled)	0.926316	0.955056	0.894737	0.923913	0.926316
XGBoost Tuned (Oversampled)	0.999474	0.998949	1.000000	0.999474	0.999474

IX. DISCUSSION

- **Interpretation of Results:** The models showed significant improvement in detecting fraudulent transactions when trained on the oversampled data. Random Forest and XGBoost performed better due to their ability to handle complex patterns.
- **Limitations:** The static nature of the dataset may limit the model's performance on highly dynamic and real-time data.
- **Implications:** Effective fraud detection can save financial institutions significant resources and enhance transaction security.

X. CONCLUSION

This project successfully developed a machine learning model to detect credit card fraud with high accuracy. The use of SMOTE and undersampling techniques effectively addressed the class imbalance issue, resulting in improved model performance. Future work can focus on real-time data processing and incorporating additional features for better accuracy.

XI. MODEL DEPLOYMENT PLAN

The deployment plan for the Random Forest model involves several steps to ensure the model is integrated effectively into the production environment. The steps include:

1. **Model Export:**
    - Save the trained Random Forest model using joblib to ensure it can be easily loaded for predictions in the production environment.
2. **Environment Setup:**
    - Ensure the production environment has all the necessary dependencies and libraries installed. This includes scikit-learn, joblib, pandas, and any other libraries used during model development.
3. **Integration:**
    - Integrate the model into the transaction processing system. This involves creating an API or a batch processing script that uses the model to predict fraudulent transactions.
4. **API Development:**
    - Develop an API endpoint using a framework like Flask or FastAPI. This API will accept transaction data, preprocess it, and return a fraud prediction.

5. **Batch Processing:**
    - Implement batch processing for offline predictions. This involves scheduling regular intervals (e.g., hourly or daily) to process new transaction data and predict fraud.
6. **Monitoring:**
    - Implement monitoring to track the model's performance in real-time. This includes logging prediction results and key performance metrics such as accuracy, precision, recall, F1 score, and AUC-ROC.
7. **Alerts:**
    - Set up alerts for any anomalies or significant changes in model performance. This ensures that any issues can be addressed promptly.
8. **Retraining:**
    - Plan for periodic retraining of the model using new transaction data to ensure it remains accurate and up-to-date with the latest fraud patterns.
9. **Documentation:**
    - Provide comprehensive documentation for the deployment process, including setup instructions, API endpoints, and batch processing scripts.
10. **Security:**
    - Ensure that the deployment environment is secure. This includes securing the API endpoints, encrypting data, and following best practices for data privacy and security.

This deployment plan ensures that the Random Forest model is effectively integrated into the production environment, providing accurate and timely fraud detection.

XII. REFERENCES

- Credit Card Fraud Detection Dataset (Kaggle)
- Various machine learning and data preprocessing libraries used.

XIII. APPENDICES

- **Code Files:** Included in the 'notebooks' and 'models' directories.
- **Visuals:** Plots and charts saved in the 'visuals' directory.
- **Data Files:** Raw and processed data available in the 'data' directory.