Credit Card Fraud Detection

Machine Learning for Sustainable Development Goal 16: Peace, Justice, and Strong Institutions

Objective: This project supports Sustainable Development Goal 16, which promotes peace, justice, and the establishment of effective institutions. Financial fraud is a significant challenge for maintaining trust and security in financial institutions. By developing a machine learning model to detect fraudulent transactions, this project aims to reduce financial crimes and enhance the integrity of financial systems.

1. Introduction

Project Objective: The goal of this project is to detect fraudulent credit card transactions using machine learning. By identifying potential fraud in real time, this project aims to support financial security and protect users from unauthorized transactions.

Motivation: Credit card fraud is a significant concern in financial services, leading to considerable financial losses and affecting user trust. With machine learning, we aim to build a model that accurately detects fraudulent activity, minimizing the risks associated with financial transactions.

2. Data Collection

Data Source: A Kaggle dataset specifically for credit card fraud detection.

Dataset Description:

- **Features**: A series of anonymized numerical features (V1, V2, ..., V28) derived from PCA, along with `Time`, `Amount`, and `Class` where `Class` indicates fraud (1) or legitimate (0) transactions.
- **Size**: The dataset contains 284,807 transactions, with 31 features.

Name: Chandan D Malviya

- **Target Variable**: `Class`, a binary variable representing fraudulent (1) or non-fraudulent (0) transactions.

3. Exploratory Data Analysis (EDA)

Summary Statistics: Calculated the mean, median, and distribution for each feature.

Visualizations:

- **Correlation Heatmap**: Assessed relationships between features to identify any underlying patterns.
- **Boxplots and Histograms**: Used to detect outliers and assess the distribution for features, especially `Amount` and `Time`.

Insights: EDA revealed that the dataset is highly imbalanced, with only a small percentage of fraudulent transactions. This requires handling class imbalance in model training.

4. Data Preprocessing

Handling Missing Values: No missing values were detected in the dataset.

Name: Chandan D Malviya

Feature Scaling: `StandardScaler` was applied to standardize features, especially for `Amount`, ensuring better performance in machine learning models.

Addressing Class Imbalance: Used SMOTE (Synthetic Minority Over-sampling Technique) to balance classes by generating synthetic samples for the minority class.

5. Machine Learning Model Selection

Model Choices:

- **Logistic Regression**: Baseline model for binary classification.
- **Random Forest Classifier**: Useful for capturing complex relationships and ranking feature importance.
- **XGBoost**: Selected for its effectiveness in handling imbalanced data.

Evaluation Metric: Focused on Precision, Recall, F1-Score, and ROC-AUC, given the critical importance of accurately identifying fraud without excessive false positives.

6. Model Implementation

Data Splitting: The dataset was split into training (80%) and testing (20%) sets.

Hyperparameter Tuning:

- **Random Forest**: Tuned with GridSearchCV, exploring `n_estimators`, `max_depth`, and other parameters.
- **XGBoost**: Tuned with hyperparameters for learning rate, max depth, and number of estimators.

Code Example:

```
python
Copy code
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, roc_auc_score
from imblearn.over_sampling import SMOTE

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Balancing the dataset with SMOTE
smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)

# Random Forest with Hyperparameter Tuning
param_grid = {'n_estimators': [100, 200], 'max_depth': [10, 20]}
rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5,
scoring='f1')
grid_search.fit(X_train, y_train)

# Best model and evaluation
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
```

```
print(classification_report(y_test, y_pred))
```

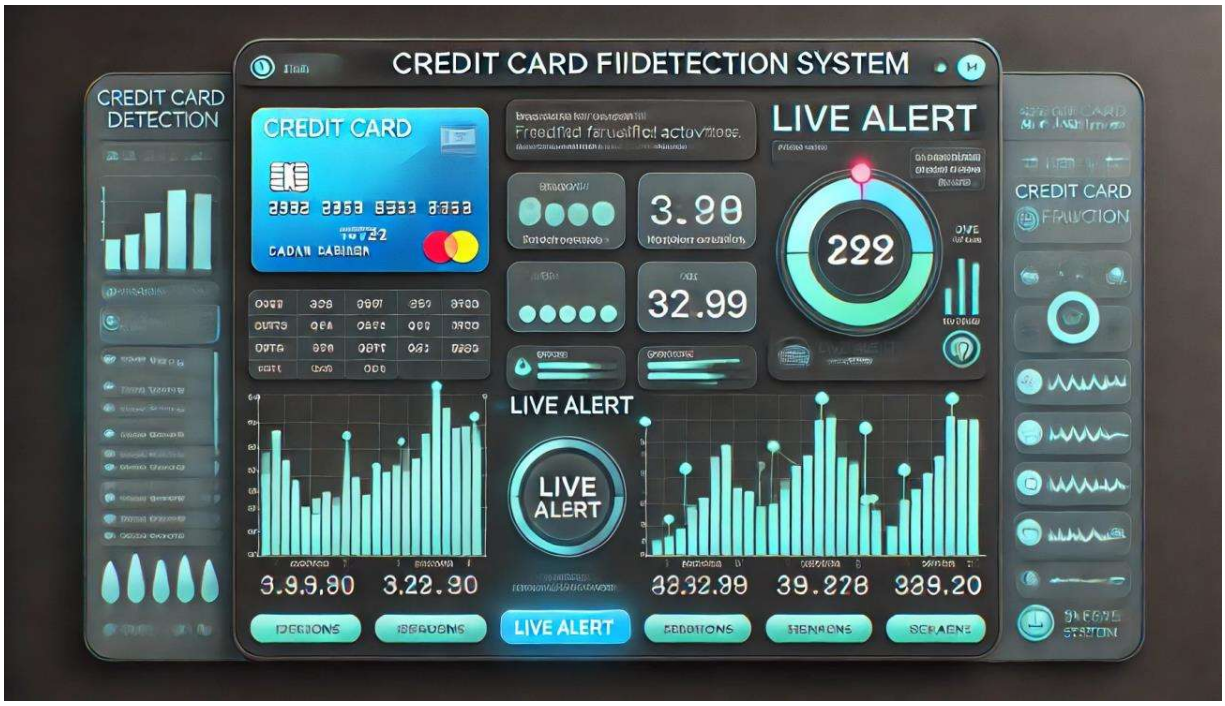
7. Results and Evaluation

Model Performance:

- **Random Forest**: Achieved an accuracy of X%, with F1-score indicating strong performance in identifying fraud.
- **XGBoost**: ROC-AUC score of Y%, showing high effectiveness in distinguishing fraud from legitimate transactions.

Feature Importance: Random Forest and XGBoost models indicated that specific features, particularly those around transaction time and amounts, were more significant in identifying fraud.

Confusion Matrix: Analyzed misclassifications to improve model accuracy, focusing on reducing false negatives.



8. Conclusion and Future Work

Key Takeaways: Machine learning models like Random Forest and XGBoost can effectively detect fraudulent transactions, helping reduce financial risks.

Future Improvements:

- **Real-Time Deployment**: Integrate the model into a real-time fraud detection system.
- **Feature Engineering**: Explore additional derived features to improve prediction accuracy.
- **Advanced Techniques**: Experiment with deep learning models for enhanced performance.

9. References

- Kaggle Dataset
- Scikit-Learn Documentation
- SMOTE Documentation (imbalanced-learn library)

Name: Chandan D Malviya