Phase-2 Submission

Student Name: N.Nantha kumar

Register Number: 732323104029

Institution: SSM College of Engineering (7323)

Department: BE., CSE

Date of Submission: 24.04.2025

Github Repository Link: https://github.com/Mowni-git/Phase2/blob/main/README.md

---

1. Problem Statement

Guarding transactions with AI-powered credit card fraud detection and prevention

1. Refined Problem Description

Credit card fraud is a growing threat in the digital payment landscape, resulting in significant financial losses for consumers and financial institutions. Traditional detection methods struggle to keep pace with the sophistication and volume of fraudulent activities. There is a critical need for an AI-powered system that can analyze real-time transactions to distinguish between legitimate and fraudulent actions swiftly.

2. Problem Type

Classification Problem: The task is to classify each transaction as either "fraudulent" or "legitimate," utilizing features such as transaction amount, location, time, and user behavior.

3. Importance of Solving This Problem

Financial Impact: Reducing losses from credit card fraud, which costs billions annually.

Consumer Trust: Enhancing user confidence in digital transactions, encouraging wider adoption.
Operational Efficiency: Automating fraud detection reduces workload and improves response times.
Regulatory Compliance: Helps financial institutions meet legal requirements to protect consumers.
Adaptability: AI models evolve with emerging fraud tactics, ensuring proactive defense against new threats.

In summary, an effective AI-powered fraud detection system is vital for securing credit card transactions, reducing losses, and maintaining consumer trust in digital payments.

2. Project Objectives

Key Technical Objectives

1. Accuracy: Achieve 95% accuracy in fraud detection.
2. Precision & Recall: Aim for 90% in both metrics.
3. F1 Score: Maximize for balanced detection.
4. Interpretability: Use SHAP and LIME for transparency.
5. Real-Time Processing: Enable immediate fraud detection.
6. Scalability: Support millions of transactions daily.
7. Robustness: Protect against adversarial attacks.
8. Integration: Seamlessly fit into existing systems.

Evolved Goals After Data Exploration

Focused Features: Narrow down to key performance indicators.
Class Imbalance: Address skewed data challenges.
Dynamic Updates: Keep the model current with evolving fraud patterns.
Stakeholder Feedback: Ensure alignment with business needs.

The project's focus has shifted to emphasize interpretability, real-time capabilities, and adaptability.

3. Flowchart of the Project Workflow

Start → Data Collection → Data Preprocessing → Exploratory Data Analysis (EDA) → Model Selection → Model Training → Model Evaluation → Model Interpretation → Deployment → Monitoring & Maintenance → Stakeholder Review → End

4. Data Description

Dataset Name and Origin:

The dataset used for this project is the "Credit Card Fraud Detection" dataset, available on [Kaggle](https://www.kaggle.com/datasets/dalpozz/creditcard-fraud).

Type of Data:

The dataset is structured data, consisting of numerical values that represent transactions.

Number of Records and Features:

The dataset contains 284,807 records (transactions) and 31 features, including time-based and transactional details, along with the target variable indicating whether a transaction is fraudulent (label).

Static or Dynamic Dataset:

This is a static dataset as it represents a snapshot of historical transaction data collected over a specific period, rather than continuously updated data.

Target Variable (if supervised learning):

The target variable for this supervised learning model is the binary feature "Class," where "1" indicates a fraudulent transaction and "0" indicates a legitimate transaction.

5. Data Preprocessing

In this section, we will document the steps taken for data cleaning and preparation to ensure that the dataset for our credit card fraud detection model is ready for analysis.

1. Handle Missing Values

We begin by checking the dataset for any missing values. In this case, we found that there are no missing values in the dataset, which allows us to proceed without needing to remove or impute any records. If there had been missing values, we would have either removed those records or employed imputation techniques to fill in the gaps based on the feature's characteristics.

2. Remove or Justify Duplicate Records

Next, we analyzed the dataset for duplicate records. Duplicate entries can artificially inflate the dataset and mislead our model during training. After checking, we determined that there were no duplicates present in the dataset. If duplicates had been found, we would have removed them to ensure the integrity of our analysis.

3. Detect and Treat Outliers

Identifying outliers is crucial, as they can unduly influence model training. We employed an outlier detection method to examine features such as transaction amounts. Through analysis, we created bounds using the Interquartile Range (IQR) method to identify outliers. Any transactions outside the defined range were removed to prevent skewing the results.

4. Convert Data Types and Ensure Consistency

We confirmed that all features in the dataset were in appropriate data types. For example, we ensured that features like 'Time' were converted to integer types when necessary. Consistency in data types is vital for model compatibility and performance.

5. Encode Categorical Variables

In this dataset, all features are numerical

6. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in understanding the structure, patterns, and relationships within the dataset. In this section, we perform both univariate and bivariate/multivariate analyses to extract insights that will guide model development.

1. Univariate Analysis

Distribution:

Histograms: Show the distribution of transaction amounts, revealing a right-skewed pattern.

Boxplots: Identify outliers in the transaction amounts and visualize data spread.

Count Plots: Display distribution of categorical variables (e.g., legitimate vs. fraudulent transactions).

2. Bivariate/Multivariate Analysis

Correlation Matrix: Identifies linear relationships between numerical features, highlighting strong correlations (e.g., transaction amount with fraud).

Pairplots: Visualize interactions between multiple features, revealing patterns and potential clustering.

Scatterplots: Show relationships between transaction amount and fraud status, indicating higher amounts associated with fraud.

Grouped Bar Plots: Compare categorical features against the target variable, revealing trends in different categories.

3. Insights Summary

Patterns/Trends:

Right-skew in transaction amounts; timing of fraud may peak during specific times/days.

Influential Features:

Transaction Amount: Critical for distinguishing between fraud and legitimacy.

Transaction Time: Certain times may correlate with increased fraud.

Observations:
Higher-value transactions often indicate fraud; uncommon geographical locations may show elevated fraud rates.

Conclusion
This EDA highlights crucial insights into the dataset, informing feature selection and modeling strategies for effective credit card fraud detection.

7. Feature Engineering

Feature engineering is a pivotal step in the machine learning pipeline as it can significantly enhance model performance by creating informative features or transforming existing ones. Below are strategies for feature engineering based on domain knowledge and insights from exploratory data analysis (EDA).

1. New Features:
Amount Binning: Categorize transaction amounts ("Low", "Medium", "High") to capture relationships and reduce outliers.

Time Features: Extract hour, day, and weekend indicators to identify time-related fraud patterns.

Transaction Frequency: Count transactions in a set period to detect unusual spikes.

2. Column Manipulation:
Split DateTime: Separate date and time for independent analysis.
User Aggregations: Summarize user transaction data (e.g., total and average) to spot behavior trends.

3. Transformations:
Polynomial Features: Create interaction terms to capture non-linear relationships.

Ratios: Analyze ratios (e.g., transaction amount to average) to gain insights into spending behaviors.

4. Dimensionality Reduction:
PCA: Reduce features while maintaining variance to streamline the model.

5. Feature Removal:
Unimportant Features: Eliminate low-correlation features to reduce complexity and overfitting.

Conclusion
Focused feature engineering improves the dataset for fraud detection, enhancing model performance through key informative features.

8. Model Building

Problem Type: Classification (distinguishing between fraudulent and legitimate transactions)

Data Characteristics: High dimensionality, Imbalanced classes (more legitimate transactions than fraudulent)
Selected Models:

1. Logistic Regression
Justification: Simple, interpretable, and effective for binary classification. Serves as a baseline model to compare with more complex algorithms.

2. Random Forest
Justification: Ensemble model that handles non-linear relationships and complex interactions between features. Robust to overfitting.

Data Splitting:
Stratified Split: Maintains class proportions (fraud/legitimate) in training and testing sets to address class imbalance (80/20 split).

Model Training and Evaluation:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

# Load and preprocess data
X = ...  # Features
y = ...  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

# Logistic Regression
logreg = LogisticRegression(random_state=42)
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)

# Evaluate Logistic Regression
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
precision_logreg = precision_score(y_test, y_pred_logreg)
recall_logreg = recall_score(y_test, y_pred_logreg)
f1_logreg = f1_score(y_test, y_pred_logreg)

# Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Evaluate Random Forest
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)
```

Metrics for Evaluation:

- Accuracy, Precision, Recall, F1-Score (focus on precision and recall due to class imbalance).

Final Step:

- Compare the models based on performance metrics and select the best model for fraud detection, considering business needs.

9. Visualization of Results & Model Insights

This summary provides key visualizations and interpretations for our credit card fraud detection models.

1. Confusion Matrix

```python

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

# Plot confusion matrix

cm = confusion_matrix(y_test, y_pred_rf)

ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Legitimate", "Fraud"]).plot(cmap=plt.cm.Blues)

plt.title("Confusion Matrix for Random Forest Model")

plt.show()

```

Interpretation:
TP:Fraud detected correctly.
TN: Legitimate transactions identified correctly.
FP: Legitimate transactions misclassified as fraud.
FN: Fraud missed.

High TP and TN with low FP and FN indicates good model performance.

2. ROC Curve

```python

from sklearn.metrics import roc_curve, auc

# Plot ROC curve

fpr, tpr, _ = roc_curve(y_test, rf.predict_proba(X_test)[:, 1])

roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' % roc_auc)

plt.plot([0, 1], [0, 1], color='red', linestyle='--')

plt.title('ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.legend(loc="lower right")

plt.show()

```

Interpretation:

- AUC close to 1 indicates a strong ability to distinguish between fraud and legitimate cases.

3. Feature Importance Plot

```python

import pandas as pd

import numpy as np

importances = rf.feature_importances_
```

```
feature_names = X.columns

plt.barh(feature_names, importances)

plt.title("Feature Importance")

plt.show()
```

Interpretation:
High importance features indicate significant predictors of fraud, such as "transaction amount" or "location".

4. Model Performance Comparison

```python
import seaborn as sns
performance_df = pd.DataFrame({

    'Model': ['Logistic Regression', 'Random Forest'],

    'Accuracy': [accuracy_logreg, accuracy_rf],

    'Precision': [precision_logreg, precision_rf],

    'Recall': [recall_logreg, recall_rf],

    'F1 Score': [f1_logreg, f1_rf]

}).melt(id_vars='Model', var_name='Metric', value_name='Score')

sns.barplot(data=performance_df, x='Metric', y='Score', hue='Model')

plt.title("Model Performance Comparison")

plt.ylim(0, 1)

plt.show()```
```

Interpretation:
Visual performance comparison facilitates identifying which model optimally balances accuracy, precision, and recall preferences in detecting fraud.

Conclusion

The visualizations collectively illustrate model performance, highlight significant predictors, and enable informed decisions in selecting the best fraud detection strategy.

10. Tools and Technologies Used

Here's a list of the tools and technologies utilized in this phase of the credit card fraud detection project:

1.Programming Language

Python: Chosen for its extensive libraries and community support in data science and machine learning.

2. IDE/Notebook

Google Colab: Used for its easy access to computational resources and collaboration features.
Jupyter Notebook: Utilized for interactive coding and visualization, allowing for seamless documentation of the analysis process.

3. Libraries

pandas: Used for data manipulation and analysis, particularly for handling large datasets.
numpy: Utilized for numerical operations and array manipulations.
scikit-learn: Essential for implementing machine learning algorithms, model training, and evaluation.
XGBoost: Implemented for boosting algorithms, particularly useful for handling imbalanced datasets in classification tasks.
seaborn: Employed for statistical data visualization, making it easier to create informative plots.
matplotlib: Used for creating static, animated, and interactive visualizations in Python.

4. Visualization Tools

Plotly: Utilized for creating dynamic visualizations that provide interactive experiences.

Tableau: Leveraged for creating advanced visual analytics (if applicable) in a user-friendly interface.

Power BI: Used for business intelligence reports and dashboards (if applicable) to summarize insights for stakeholders.

Summary

These tools and technologies facilitated data processing, modeling, and visualization, allowing for a robust framework for the credit card fraud detection analysis. Each component was selected to enhance productivity and effectiveness throughout the project workflow.

11.Report

Future iterations of this project could involve:

Testing additional model frameworks like Neural Networks.

Implementing online learning for real-time fraud detection.

Enhancing the model with additional features and larger datasets

Acknowledgments

We thank the community and data providers for the resources used in this analysis and the open-source tools that made this project feasible.

12 . Team Members and Contributions

- *E.Nithya bharadhan - Data cleaning*
- *M.Gopika - EDA*
- *M.Mathan - Feature engineering*
- *N.Nantha kumar - Model development*
- *J.Mownika - Documentation and reporting]*