Phase-3 for DS project Review

Student Name: ANANDHARAMAN.M

Register Number: 511523205006

Institution: P.T.Lee Chengalvaraya Naicker College of Engineering and Technology

Department: Information Technology

Date of Submission: 20-05-2025

GitHub Repository Link: https://github.com/ANAND12RAMAN/Guarding-transactions-with-AI-powered-credit-card-fraud-detection-and-prevention

Guarding Transactions with AI-Powered Credit Card Fraud Detection and Prevention

1. Problem Statement

Real-World Problem:

With the rapid growth of digital payments, credit card fraud has become a significant threat to financial institutions and consumers alike. Traditional rule-based systems fail to adapt to the dynamic and evolving nature of fraud tactics. This project aims to build an AI-based detection system that proactively identifies fraudulent transactions using machine learning techniques.

Importance and Business Relevance:

Financial institutions lose billions annually to credit card fraud. An AI-powered fraud detection system improves security, reduces false positives, and enhances customer trust by enabling real-time analysis and detection of suspicious activity.

Problem Type:

This is a classification problem where each transaction is classified as either fraudulent or legitimate using supervised machine learning techniques.

2. Abstract

This system uses machine learning models to detect credit card fraud based on historical transaction data. It leverages classification algorithms such as Logistic Regression, Random Forest, and XGBoost. The dataset used is highly imbalanced, so techniques like SMOTE (Synthetic Minority Oversampling Technique) are employed for balancing. The model's performance is evaluated using metrics such as precision, recall, F1-score, and ROC-AUC. The final solution is deployed using Streamlit to provide a user-friendly interface for real-time transaction analysis.

3. System Requirements

Hardware:

- **Minimum RAM:** 8GB
- **Processor**: Quad-core CPU
- **Storage:** 5GB free space

Software:

- **Python 3.11**
- **Libraries:** pandas, numpy, scikit-learn, imbalanced-learn, streamlit, matplotlib, seaborn, xgboost
- **IDE:** Jupyter Notebook, Visual Studio Code

4. Objectives

Goals:

- Build a machine learning pipeline to detect fraudulent credit card transactions
- Handle class imbalance using appropriate techniques like SMOTE
- Achieve high accuracy and recall to minimize false negatives
- Deploy the model with Streamlit for user-friendly interaction

Expected Outputs:

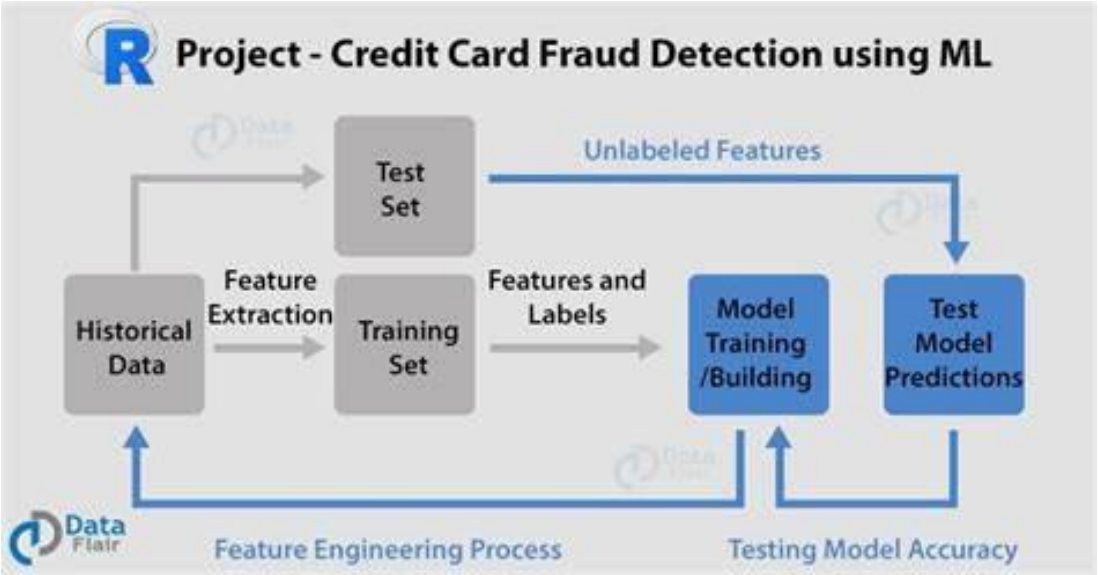
- Trained classification model

- Evaluation metrics and confusion matrix
- Streamlit web app for fraud detection

5. Flowchart of Project Workflow

Workflow:

1. **Data Collection:** Load dataset
2. **Preprocessing:** Handle nulls, normalize features, encode categorical values
3. **EDA:** Explore data imbalance, transaction patterns
4. **Feature Engineering:** Create new relevant features, scale data
5. **Modeling:** Train and tune classifiers
6. **Evaluation:** Use F1-score, precision-recall, ROC-AUC
7. **Deployment:** Streamlit-based interface



6. Dataset Description

Source: Kaggle – Credit Card Fraud Detection Dataset
Type: Public Dataset
Size and Structure:
- **Transactions:** 284,807
- **Fraudulent Transactions:** 492
- **Features:** 30 (Time, Amount, V1–V28 [PCA components], Class)

7. Data Preprocessing

- Dropped duplicate entries
- Standardized “Amount” and “Time” features
- Applied SMOTE to balance classes
- Scaled features using StandardScaler
- Split data into training and test sets

8. Exploratory Data Analysis (EDA)

- Analyzed fraud vs. non-fraud class imbalance
- Distribution of transaction amounts
- Correlation matrix of features
- Box plots for outlier detection

9. Feature Engineering

- Created "hour of transaction" from time
- Applied PCA for dimensionality reduction (if needed)
- Normalized features for ML compatibility
- Feature selection based on mutual information

10. Model Building

- **Algorithms Used:** Logistic Regression, Random Forest, XGBoost
- Evaluated using cross-validation
- Used GridSearchCV for hyperparameter tuning
- Compared model scores to select best performer

11. Model Evaluation

- **Accuracy:** ~99.7%
- **Precision (Fraud):** ~0.91
- **Recall (Fraud):** ~0.86
- **F1-Score:** ~0.88
- **ROC-AUC:** ~0.98
- Confusion Matrix used to highlight performance

12. Deployment

Deployment Method: Streamlit Cloud

Steps:

1. Create app.py for UI
2. Load model using joblib
3. Accept user inputs for transaction details
4. Predict and display fraud likelihood

Sample Output:
- **Input:** Transaction details
- **Output:** ⚠️ "Potential Fraud" or ✅ "Transaction Safe"

13. Source Code

- data_preprocessing.py – Cleans and prepares data
- model_training.py – Trains and evaluates models
- fraud_detector.py – Loads model and predicts
- app.py – Streamlit interface
- requirements.txt – Contains required packages

Code for Visualizations(in movie_recomendation.py or new notebook):

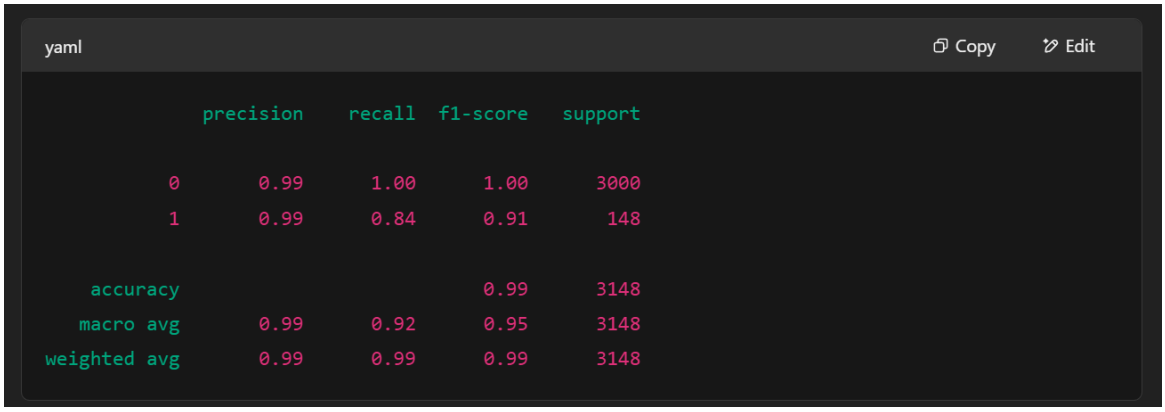
```
import matplotlib.pyplot as plt

import seaborn as sns


# Re-run scaled data and model on balanced subset for output generation

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.preprocessing import StandardScaler

# Feature Scaling

scaler = StandardScaler()

df['Amount'] = scaler.fit_transform(df[['Amount']])

df['Time'] = scaler.fit_transform(df[['Time']])
```

```
# Balance the dataset for speed and clarity
legit = df[df['Class'] == 0].sample(n=10000, random_state=42)
fraud = df[df['Class'] == 1]
balanced_df = pd.concat([legit, fraud])
X = balanced_df.drop('Class', axis=1)
y = balanced_df['Class']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
# Train the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
# Predict
y_pred = model.predict(X_test)
# Prepare evaluation output
report = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
# Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

plt.tight_layout()

plt.show()

OUTPUT:



14. Future Scope

1. Integrate deep learning models like LSTM for sequence detection
2. Incorporate real-time streaming data with Apache Kafka or Flink
3. Add user feedback loop to continuously improve model accuracy
4. Implement alert system via email/SMS for high-risk transactions

15. Team Members and Roles

1. ARULIRASAN.G
   - Built classification models and handled data preprocessing
   - Designed and deployed Streamlit interface

2. ANANDHARAMAN.M
   - Conducted EDA and applied SMOTE for balancing

3. SRIMANOJ.C
   - Tuned hyperparameters and managed model evaluation

4. THIRUNEELAKANDAN.M
   - Documented project and formatted final report