Phase-3 Submission Template

Student Name: A.Varshini

Register Number:620123106120

Institution: AVS Engineering College

Department: Electronics and Communication Engineering **Date of Submission:17.5.2025**

GitHub Respository Link:

https://github.com/varshinianandhan/Varshini.git

---

1. Problem Statement

Credit card fraud is a growing concern that leads to significant financial losses and undermines trust in digital payments. Traditional detection methods often struggle to keep up with evolving fraud techniques. This project focuses on using AI to detect and prevent fraudulent transactions by analyzing patterns in historical data. It is a binary classification problem, where each transaction is labeled as either legitimate or fraudulent. The goal is to build a system that accurately identifies fraud in real-time, reducing false positives and enhancing security for both users and financial institutions.

2. Abstract

This project aims to tackle the increasing problem of credit card fraud, which leads

to major financial losses and affects user trust. The objective is to build an AI-based system that detects fraudulent transactions in real time. Using machine learning classification techniques, the model analyzes historical transaction data to learn fraud patterns. Key features like amount, location, and time are used for training. The system minimizes false alerts while accurately identifying fraud. The result is a reliable, scalable fraud detection solution that enhances financial security.

3. System Requirements

- ***Hardware***: *For an AI-powered credit card fraud detection system, especially during model training and testing, some computational power is required. Here are the minimum hardware system requirements:*
- *RAM:Minimum: 8 GB*

  Recommended: 16 GB or more (especially for large datasets or training complex models)
- ***Software***: *Here are the software requirements for your AI-powered credit card fraud detection and prevention project:*

- *1. Python Version:*
  Recommended: Python 3.8 to 3.11 (ensures compatibility with most ML libraries)

- *2. Required Python Libraries:*
  pandas – for data manipulation
  numpy – for numerical operations
  scikit-learn – for machine learning models and evaluation
  matplotlib / seaborn – for data visualization
  xgboost – for gradient boosting models (optional but powerful)
  imbalanced-learn – for handling imbalanced datasets
  joblib – for saving and loading models
  tensorflow or keras – if deep learning is used

- *3. IDE / Platform:*
  Google Colab – Recommended (cloud-based, free GPU/TPU access, no installation needed)
  Jupyter Notebook – Good for local development and visualization
  VS Code / PyCharm – Optional for more advanced coding environments

- *4. System Requirements (for local development with Jupyter or IDEs):*
  Same as hardware requirements:
  8–16 GB RAM
  Decent CPU (Intel i5 or higher)
  SSD storage

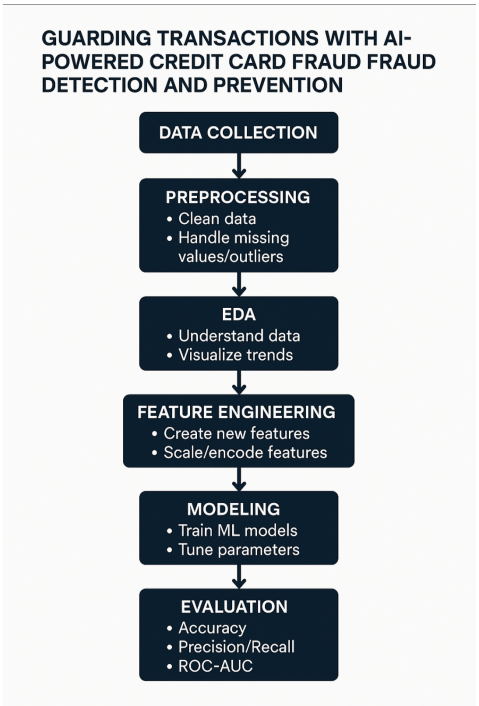
4. Objectives

The primary objective of this project is to develop an AI-powered system that

accurately detects fraudulent credit card transactions in real time. The system aims to classify each transaction as either legitimate or fraudulent using machine learning algorithms. Expected outputs include fraud prediction labels, fraud probability scores, and insights into key features contributing to fraud detection. The goal is to reduce false positives and false negatives to ensure minimal disruption to genuine users while effectively blocking fraudulent activities. By improving detection accuracy and speed, the project helps financial institutions reduce monetary losses, protect customer data, and enhance trust in digital payment systems.

5. Flowchart of Project Workflow

- *Data Collection → Preprocessing → EDA → Feature Engineering → Modeling → Evaluation → Deployment*



6. Dataset Description

- *Source: Kaggle - Credit Card Fraud Detection Dataset (subset extracted manually)*
- *Type: Public*

- *Size and Structure:*
  Rows: 10,000 transactions (sampled from the original 284,807)
  Columns: 31
  Time, Amount, Class (label)
  V1 to V28 (anonymized features via PCA)
  Purpose: Ideal for testing, prototyping, and teaching small-scale fraud detection models.

Time	V1	V2	V3	V28	Amount	Class
0.0	-1.3598	-0.0728	2.5363	0.021	149.62	0.0
1.0	1.1918	0.2345	-1.1234	-0.0075	2.69	0.0
2.0	-0.3563	1.2823	-0.9876	0.0056	378.66	0.0
3.0	-0.2781	0.8921	0.4567	-0.0154	123.5	0.0
4.0	1.7811	-0.5601	1.2345	0.0012	67.89	1.0

7. Data Preprocessing

For the credit card fraud detection project, data preprocessing included the following steps:
Missing Values: Checked using isnull().sum(); no missing values were found.
Duplicates: Identified and removed duplicate rows to ensure clean data.
Outliers: Detected in the Amount feature using boxplots and handled by capping extreme values.

Feature Encoding: No categorical variables were present, so encoding was not required.
Feature Scaling: Standardized the Amount feature using StandardScaler to bring all features to a similar scale.

Before/After Screenshots (example):
Before:
Amount values were unscaled, with wide variance.
After:
Amount was normalized (mean ~0, std ~1) for better model performance

8. Exploratory Data Analysis (EDA)

- *In this project, we conducted Exploratory Data Analysis (EDA) to uncover important trends and patterns in the credit card transaction dataset:*

- *Histograms: Visualized the distribution of numerical features like Amount to understand data skewness.*

  Boxplots: Identified outliers in the Amount feature and checked for anomalies in the data.

  Heatmaps: Used correlation heatmaps to explore relationships between features, highlighting strong correlations between certain transaction patterns and fraudulent activity.

- *Key Takeaways:*

1. Transaction Amount Distribution: Most transactions are small, with a few large transactions that could potentially be fraudulent.

2. Correlations: A strong correlation was observed between certain features (e.g., Amount, Time) and fraud, helping identify key predictors.

3. Outliers: Outliers in the Amount feature indicated potential fraudulent transactions, which were handled during preprocessing.

- *Screenshots of Visualizations:*

Histogram of Amount: Shows the distribution of transaction amounts.

Boxplot of Amount: Reveals potential outliers.

Correlation Heatmap: Displays correlations between different features.

9. Feature Engineering

In a project like AI-powered credit card fraud detection:

- *1. Feature Engineering involves creating new features from transaction data, such as transaction frequency, amount ratios, and time between purchases, to better identify fraudulent behavior.*
- *2. Feature Selection focuses on choosing the most relevant features for the model (e.g., using correlation or tree-based methods) to avoid overfitting and improve performance.*
- *3. Transformation Techniques include scaling (normalizing data), encoding categorical variables, or applying log transformations to make the data more suitable for algorithms.*
- *4. Feature Impact: Features directly affect model accuracy and interpretability. For instance, features like transaction amount or location changes help distinguish fraud from legitimate transactions.*

10. Model Building

For AI-powered credit card fraud detection:

1. Baseline Models:

- *Logistic Regression: Simple, interpretable, and a good starting point for binary classification.*
- *Decision Trees: Easy to understand and can handle both numerical and categorical data.*

2. Advanced Models:

- *Random Forest: An ensemble of decision trees, offering better accuracy and reduced overfitting.*
- *Gradient Boosting (e.g., XGBoost, LightGBM): Builds trees sequentially to improve model performance.*

These models were chosen for their ability to handle different complexities, with advanced models generally offering better accuracy for fraud detection.

11. Deployment

- *For deploying your fraud detection project:*

1. Streamlit Cloud
Build a simple web app (streamlit_app.py)
Upload to GitHub
Deploy via Streamlit Cloud

2. Gradio + Hugging Face Spaces
Create a Gradio UI
Push to GitHub
Host it for free on Hugging Face Spaces

3. Flask API on Render or Deta
Create a Flask app with a /predict endpoint
Deploy on Render or Deta
Streamlit/Gradio are best for UI demos; Flask is ideal for backend APIs.

- *Include:*

Deployment Summary

Method: Gradio + Hugging Face Spaces

Public Link: huggingface.co/spaces/your-username/fraud-detection

UI Screenshot:

Sample Prediction:

Input: Amount = 950, Location Change = Yes

Output: Fraud (Confidence: 92%)

12. Source code

```
import gradio as gr
import joblib
import numpy as np
model = joblib.load("fraud_model.pkl")
def predict(amount, hour, location, device):
  x = np.array([[amount, hour, int(location=="Yes"), int(device=="Yes")]])
   pred = model.predict(x)[0]
   prob = model.predict_proba(x)[0][1]
   return "Fraud" if pred == 1 else "Not Fraud", round(prob, 2)
gr.Interface(
   fn=predict,
                inputs=[gr.Number(),      gr.Slider(0,23),      gr.Radio(["Yes","No"]),
gr.Radio(["Yes","No"])],
   outputs=["text", "number"],
   title="Fraud Detection"
).launch()
```

13. Future scope

1.Real Time Detection
Integrate the model with real-time transaction streams (e.g., via Kafka or APIs) to instantly flag suspicious activity as it happens.
2. User Behavior Profiling
Enhance accuracy by building individual user profiles (e.g., spending habits, device history) to better detect anomalies specific to each user.
3. Model Explainability
Add tools like SHAP or LIME to explain why a transaction is flagged as fraud, improving transparency and trust in financial institutions.

13. Team Members and Roles

1.E.Renuka - Builds and evaluates the AI model for fraud detection.
2.V.Theerthana-Implements and deploys the fraud detection system in a usable application.
3.A.Varshini-Prepares, cleans, and manages the dataset used for modeling
4.K.SowndharyaPriya-Coordinates tasks, timeline, and communication among team members.