**Phase-2 Submission Template**

**Student Name:** [S.Pavithra]

**Register Number:** [620123243035]

**Institution:** [AVS ENGINEERING COLLEGE]

**Department:** [B.TECH AI&DS]

**Date of Submission:** [08.05.2025]

**Github** ( <https://github.com/pavithra-shreya/Fake-news-detection-powered-by-natural-language-processing-Exposing-the-truth-with-advanced.git> )



# Project Statement

Guarding transaction with AI powered credit card fraud detection and prevent

“Guarding transactions with AI-powered credit card fraud detection and prevention” refers to using artificial intelligence (AI) techniques to detect and stop fraudulent activities in real-time or near real-time. The “high methodology” in this context refers to advanced methods and best practices used to maximize the effectiveness of such systems. Here’s an overview of the high-level methodology:

# Project Objectives

1. Detect Fraudulent Transactions in Real-Time: Utilize AI and machine learning algorithms to identify suspicious credit card activities instantly.
2. Reduce False Positives and False Negatives: Improve the accuracy of fraud detection models to minimize incorrectly flagged transactions and undetected fraud.
3. Develop a Scalable and Adaptive AI Model: Build a system that continuously learns from new data and adapts to evolving fraud patterns.
4. Ensure Data Security and Privacy: Implement robust data protection mechanisms to ensure sensitive customer data remains secure.

# Flowchart of the Project Workflow

1. **Transaction Initiation**

**→ User attempts a credit card transaction.**

1. **Initial Checks**

**→ Validate card status (active, not expired)**

**→ Check available balance or credit limits**

1. **Data Collection**

**→ Capture transaction details (amount, location, device, time)**

1. **AI Fraud Detection**

**→ Feed data into machine learning model**

**→ Analyze behavior and detect anomalies**

**→ Generate fraud risk score**

1. **Risk Assessment**

**→ If Low Risk → Approve transaction**

**→ If Medium Risk → Request additional verification (OTP, biometrics)**

**→ If High Risk → Block transaction and flag as potential fraud.**

1. **Post-Transaction Processing**

**→ Log and monitor transaction**

**→ Get feedback from user if flagged**

**→ Retrain model with new data**

# Data Description

The system utilizes transactional and customer-related data to detect and prevent fraudulent credit card activities in real time. The data used typically includes the following components:

1. Transactional Data

This is the primary data used to detect fraud and includes:

Transaction ID: Unique identifier for each transaction.

Timestamp: Date and time of the transaction.

Amount: Value of the transaction.

Merchant ID/Name: Identifier or name of the business where the transaction occurred.

Location: Geolocation data such as city, state, and country.

Currency: Currency in which the transaction was made.

Device Type: Type of device used for the transaction (e.g., mobile, desktop, POS).

Channel: Online, offline, mobile app, ATM, etc.

1. Customer Profile Data

Used to understand and model typical customer behavior:

Customer ID: Unique identifier for the cardholder.

Age, Gender: Demographic information.

Account Age: How long the account has been active.

Credit Score: May be included to assess risk.

Average Transaction Value: Helps in modeling typical spending behavior.

Preferred Locations and Merchants: Patterns of normal use.

1. Card Usage History

Past Transaction Logs: Helps train models on both normal and fraudulent behavior.

Transaction Frequency: Daily, weekly patterns.

Disputed Transactions: Labeled historical fraud cases.

Data PThe effectiveness of fraud detection relies heavily on how data is collected, cleaned, transformed, and prepared for model training and inference. Below are the key stages involved in the data processing pipeline:

1. Data Collection

Source Aggregation: Data is collected from transaction logs, customer profiles, bank records, and third-party risk intelligence databases.

Real-Time & Batch Inputs: Both historical (batch) and streaming (real-time) data pipelines are used to ensure up-to-date analysis.

1. Data Cleaning

Handling Missing Values: Imputation or removal of records with incomplete information.

Noise Reduction: Filtering out duplicate or irrelevant transactions.

Outlier Detection: Identifying abnormal values that may distort model training.

1. Data Integration

Combining Sources: Merging customer data, transaction records, and fraud reports using unique identifiers (e.g., customer ID, transaction ID).

Temporal Alignment: Ensuring consistent time formats and alignment for time-series analysis.

1. Data Transformation

Feature Engineering:

Time-based features (e.g., transaction hour, day of week)

Transaction velocity (e.g., number of transactions in last 1 hour)

Location-based deviation (distance from typical locations)

Merchant category trends

# ExporaltoryData Analysis (EDA)

For your project “Guarding Transactions with AI-Powered Credit Card Fraud Detection and Prevention”, Exploratory Data Analysis (EDA) is a crucial step to understand the dataset, detect patterns, and identify anomalies that may indicate fraudulent behavior.

Here’s a structured outline for EDA, along with Python code using common libraries like pandas, matplotlib, and seaborn.

* 1. Load and Inspect the Dataset

Import pandas as pd

# Load the dataset

Df = pd.read\_csv(‘creditcard.csv’) # Replace with your dataset path

# Basic structure

Print(df.shape)

Print(df.info())

Print(df.describe())

* 1. Class Distribution (Fraud vs. Non-Fraud)

# Count of each class

Print(df[‘Class’].value\_counts())

# Visualize

Import seaborn as sns

Import matplotlib.pyplot as plt

Sns.countplot(x=’Class’, data=df)

Plt.title(“Class Distribution (0: Non-Fraud, 1: Fraud)”)

Plt.show()

* 1. Missing Values Check

# Check for missing values

Print(df.isnull().sum())

1. Feature Correlations

# Correlation matrix

Corr\_matrix = df.corr()

# Visualize

Plt.figure(figsize=(15, 10))

Sns.heatmap(corr\_matrix, cmap=’coolwarm’, vmax=0.8)

Plt.title(“Feature Correlation Matrix”)

Plt.show()

1. Analyze Transaction Amount and Time.

# Feature Engineering

Feature engineering for AI-powered credit card fraud detection and prevention is a crucial step in building effective machine learning models. It involves creating relevant input features from raw transaction data to help models differentiate between legitimate and fraudulent transactions.

Here are key feature engineering techniques commonly used in this domain:

* 1. Transaction-Based Features

These are directly derived from individual transactions:

Transaction amount: Raw or normalized amount.

Transaction time: Time of day, weekday/weekend, etc.

Merchant category: One-hot encoded or embedded.

Payment method: Online, POS, mobile, etc.

* 1. User Behavior Features

Patterns of a user’s typical transaction behavior:

Average transaction amount (over past n days).

Transaction frequency: Number of transactions in the past 1 hour/day/week.

Time since last transaction.

Change in transaction pattern: e.g., spending spike.

* 1. Geolocation and Device Features

Useful for detecting unusual access:

Transaction location: Country, city, or distance from previous transaction.

IP address and device ID: Has this device been used before?

Velocity features: Speed between two transactions (if locations vary widely in short time).

* 1. Historical and Aggregated Features

Summarized data over a time window:

# Model Building

The model building phase involves selecting, training, evaluating, and tuning machine learning models that can accurately detect fraudulent transactions. Here’s a structured overview of how to approach this:

Model Building for Credit Card Fraud Detection

1. Data Preparation

Before modeling, ensure your data is:

Cleaned (handle missing values, duplicates)

Balanced (class imbalance is common; use techniques like SMOTE or undersampling)

Feature-engineered (based on previous step)

1. Model Selection

Start with a few baseline models, then iterate with more complex ones:

Baseline Models

Logistic Regression: Good baseline for interpretability.

Decision Trees: Handles non-linear relationships.

Random Forests: Ensemble model that reduces overfitting.

Gradient Boosting (XGBoost, LightGBM, CatBoost): High-performance for tabular data.

Advanced Models

Neural Networks: Deep learning models, especially with temporal features.

Autoencoders: Unsupervised anomaly detection.

Isolation Forests / One-Class SVM: Good for rare-event fraud detection.

1. Handling Class Imbalance

Fraud detection is an imbalanced classification task (frauds are <1%):

Resampling techniques: SMOTE, ADASYN, Tomek links.

Cost-sensitive learning: Penalize false negatives more heavily.

# Visualization of Results & Model Insights

**To create a compelling visualization section for “Guarding Transactions with AI-Powered Credit Card Fraud Detection and Prevention”, we can divide it into two parts:**

1. Visualization of Results

This section shows how well the AI model performs. Recommended plots and visuals:

* 1. Confusion Matrix

Shows the number of true positives, true negatives, false positives, and false negatives.

Helps in understanding misclassification.

* 1. ROC Curve & AUC Score

Demonstrates the model’s ability to distinguish between fraudulent and legitimate transactions.

AUC closer to 1 indicates strong performance.

* 1. Precision-Recall Curve

Particularly useful when dealing with imbalanced datasets, such as fraud detection.

Highlights trade-offs between precision and recall.

* 1. Classification Metrics Summary

A table with Accuracy, Precision, Recall, F1-Score.

Gives a numerical overview of performance.

1. Model Insights

This part reveals why the model behaves the way it does.

* 1. Feature Importance (Tree-based models)

Bar chart showing the most influential features (e.g., transaction amount, time, location variance).

Helps stakeholders understand decision factors.

* 1. SHAP (Shapley Additive exPlanations) Values

Visual explanation of how each feature contributes to a particular prediction.

Useful for high-stakes environments like fraud detection.

* 1. Anomaly Scores (if using Isolation Forest, Autoencoders, etc.)

Heatmap or line graph showing the anomaly score per transaction.

Helps in identifying suspicious patterns over time.

* 1. Cluster Visualization (PCA or t-SNE)

2D or 3D scatter plot showing clusters of legitimate vs. fraudulent transactions.

Helps validate that the model can distinguish patterns in data space.

Would you like me to generate Python code to create these visuals or help you build a report layout with them?

# Tools and Technologies Used

**For the “Guarding Transactions with AI-Powered Credit Card Fraud Detection and Prevention” project, the Tools and Technology section should outline the key components used to develop, train, and deploy the fraud detection system. Here’s a breakdown:**

Tools and Technology

1. Programming Languages

Python: Primary language used for data preprocessing, model development, and evaluation.

SQL: Used for querying transactional data from databases.

1. Data Handling and Preprocessing

Pandas: For data manipulation and cleaning.

NumPy: For numerical operations.

Scikit-learn: For scaling, encoding, and splitting datasets.

1. Machine Learning & AI Frameworks

Scikit-learn: For traditional ML models like Logistic Regression, Decision Trees, Random Forests.

XGBoost / LightGBM: For high-performance gradient boosting models.

TensorFlow / Keras: For deep learning models (e.g., autoencoders).

PyOD: For specialized outlier detection techniques.

1. Model Explainability

SHAP: For interpreting model predictions and understanding feature impact.

LIME: Local interpretation of individual predictions.

1. Data Visualization

Matplotlib / Seaborn: For visualizing trends, patterns, and model performance.

Plotly / Dash: For interactive visual dashboards.

1. Model Evaluation

Scikit-learn Metrics: For computing accuracy, precision, recall, F1-score, ROC-AUC.

Yellowbrick: For visual model evaluation and diagnostics.

1. Deployment & Integration

Flask / FastAPI: To expose the model as a web API.

Docker: For containerization and deployment.

AWS / Azure / GCP: For cloud-based model hosting and scalability.

1. Monitoring & Logging

Prometheus + Grafana / ELK Stack: For real-time monitoring of model predictions and system health.

MLflow: For experiment tracking and model versioning.

Would you like this as a formatted section for a report or presentation slide?

# Team Members and Contributions

**1.R.Devashree** o ***Data Cleaning****: Preprocessing the dataset, handling missing values, and removing any irrelevant or erroneous data.*

o ***Documentation and Reporting****: Writing the project report, summarizing the methodology, results, and providing insights into the model performance.*

1. **S.Varshini** o ***Exploratory Data Analysis (EDA)****: Performing data analysis, visualizing relationships between variables, and identifying trends i n the dataset.*

o ***Feature Engineering****: Creating new features, transforming data, and selecting key features t o enhance the model's performance.*

1. **A.K.dakshinya**o ***Model Development****: Building and training the linear regression model, optimizing it, and evaluating i t s accuracy and performance.*