Phase-2

Student Name: LOKESH.J

Register Number: 410723104040

Institution: Dhanalakshmi College of Engineering

Department: Computer Science and Department

Date of Submission: 08-05-2025

GitHub Repository Link: Lokesh151120 · GitHub

---

1. Problem Statement

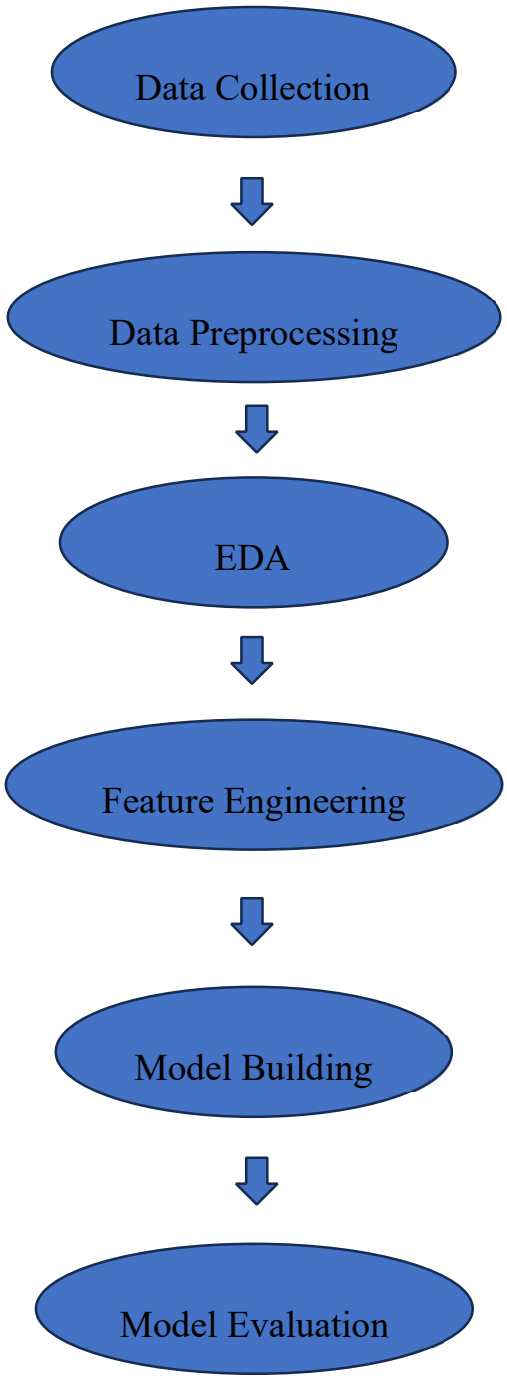
credit card fraud has become a major concern. Detecting fraudulent transactions accurately and efficiently is critical to reducing financial losses and maintaining customer trust. Traditional methods are time-consuming and may not adapt well to evolving fraud patterns. The goal of this project is to build a machine learning model that can automatically detect fraudulent credit card transactions by analysing transaction data, thereby reducing manual intervention and improving detection accuracy. The solution should handle imbalanced datasets and prioritize recall to minimize missed fraudulent cases.

2. Project Objectives

- Accurately classify credit card transactions as fraudulent or non-fraudulent using machine learning techniques.
- Maximize recall to ensure that as many fraudulent transactions as possible are detected, even if it means allowing some false positives.

- Minimize manual intervention by developing an automated and efficient fraud detection system.
- Ensure model interpretability, so the decisions made by the model can be understood and trusted by users.
- Address class imbalance effectively to avoid biased model predictions.
- Evaluate and compare multiple models, focusing on performance metrics suitable for imbalanced data (e.g., recall, precision, F1-score, ROC AUC).

3. Flowchart of the Project Workflow



4. Data Description

Dataset Overview:

- Source: Kaggle
- Total Records: 284,807 transactions
- Number of Features: 30

Target Variable:

- Class
- 0 = Legitimate (non-fraudulent transaction)
- 1 = Fraudulent transaction

Feature Details:

- Most features (V1 to V28) are anonymized principal components obtained through PCA (Principal Component Analysis) for confidentiality.
- Two non-anonymized features:
- Time – Time elapsed in seconds between the transaction and the first transaction in the dataset.
- Amount – Transaction amount in dollars.

Characteristics:

- Highly imbalanced dataset: Only 0.17% of the transactions are fraudulent.
- No missing values were found.
- Outliers were present in the Amount feature.
- Data is purely numerical and already pre-processed using PCA (except Time and Amount).

5. Data Preprocessing

Missing Value Handling:

- The dataset had no missing values, so no imputation or removal was necessary.

Duplicate Removal:

- Duplicate records were identified and removed to ensure data quality and avoid bias.

Outlier Handling:

- Outliers were detected in the Amount feature.
- A log transformation was applied to normalize the distribution:
- log_amount = np.log1p(Amount)

Feature Scaling:

- The Amount and Time features were standardized using StandardScaler:
- New features: norm amount and norm time
- The original Amount and Time columns were then dropped.
- Final Feature Set:
- The final dataset included:
- Scaled versions: norm amount, norm time
- PCA components: V1 to V28
- Target variable: Class

6. Exploratory Data Analysis (EDA)

1.Class Distribution (Imbalance):

- The dataset is highly imbalanced:
- Legit Transactions (Class 0): 99.83%

- Fraudulent Transactions (Class 1): 0.17%
- This imbalance necessitates special handling (e.g., resampling techniques or metric tuning).

2. Amount Feature:

- Wide range of transaction amounts.
- Presence of outliers, especially in large transaction values.
- Applied log transformation to normalize skewed distribution.

3. Time Feature:

- Distribution was uniform with no meaningful trend or seasonal pattern.
- Later standardized as norm_time.

4. PCA Features (V1–V28):

- These features are already transformed and anonymized via Principal Component Analysis.
- V14, V10, V17, V12 stood out as strong indicators of fraud based on distribution differences and feature importance analysis.
- No signs of multicollinearity (low correlations between features).

5. Correlation Analysis:

- Target variable had higher correlation with features:
- V14, V10, V17, V12
- No high correlations among input features, which is favorable for model training.

6. Visualizations Used:

- Histograms and box plots to understand the spread of Amount and Time.
- Correlation heatmap to identify relationships between features and the target.
- Count plot for class imbalance.
- Feature importance plots post-modelling

7. Feature Engineering

Handling Amount Feature:

- Applied log transformation to reduce the skewness and normalize the distribution:
- log_amount = np.log1p(Amount)

Handling Time Feature:

- Standardized using StandardScaler to bring it to a similar scale as other features:
- Created norm_time

Feature Scaling:

- Both log_amount and norm_time were standardized:

- norm_amount = StandardScaler().fit_transform(log_amount.reshape(-1,1))

Feature Reduction:

- Original Amount and Time columns were dropped after transformation to avoid redundancy.

PCA Features:

- Features V1 to V28 were already transformed through Principal Component Analysis (PCA), so no further engineering was done on them.

Final Feature Set:

- norm time, norm amount, and V1 to V28
- Class was retained as the target variable

8. Model Building

1. Train-Test Split:

- The dataset was split into:
- Training set: 70%
- Testing set: 30%

- Stratified split was used to preserve class imbalance proportions.

2. Handling Imbalanced Data:

- Since fraudulent transactions are very rare (0.17%), the following techniques were used:
- Oversampling with SMOTE (Synthetic Minority Oversampling Technique) to balance the training data.

3. Models Used:

a. Logistic Regression

- Baseline model for classification.
- Applied with regularization.
- Interpretable and fast.

b. Random Forest Classifier

- Ensemble-based model using multiple decision trees.
- Good performance on imbalanced data when tuned properly.
- Provides feature importance.
- c. XGBoost Classifier
- Gradient boosting model.
- Excellent performance with imbalanced and structured data.
- Tuned with parameters like max_depth, learning_rate, and n_estimators.

4. Evaluation Metrics:

- Given the class imbalance, the focus was on:
- Recall (to catch more frauds)
- Precision

- F1-score
- ROC-AUC

5. Model Selection:

- Models were compared based on the above metrics.
- XGBoost provided the best balance between recall and precision, making it the preferred model.
- Let me know if you'd like to see the evaluation results or confusion matrix breakdown too!

9. Visualization of Results & Model Insights

Confusion Matrix:

- Visualized for each model (Logistic Regression, Random Forest, XGBoost).

Shows:

- True Positives (TP) – correctly detected frauds
- False Positives (FP) – legitimate transactions wrongly flagged
- True Negatives (TN) – correctly identified legitimate transactions
- False Negatives (FN) – missed frauds (critical in fraud detection)

Goal: Minimize FN to catch as many frauds as possible.

- ROC Curve (Receiver Operating Characteristic):
- Plotted for each model.
- AUC (Area Under Curve) score used to compare model performance.

- XGBoost showed the highest AUC, indicating strong classification ability.

Precision-Recall Curve:

- Important for imbalanced datasets.
- Helps evaluate model performance in detecting rare classes (fraud).
- Again, XGBoost had the best balance between precision and recall.
- Feature Importance (from Random Forest and XGBoost):
- Visualized top contributing features to fraud detection:
- V14, V10, V17, V12 consistently ranked highest
- Indicates which PCA components are most associated with fraud patterns.

Class Distribution Plot:

- Bar chart showing the imbalance between fraud and non-fraud transactions.

Amount Distribution (before vs after transformation):

- Histograms show the effect of log transformation on the skewness of transaction amounts.

10. Tools and Technologies Used

Programming Language:

Python

- Chosen for its simplicity and rich ecosystem for data science and machine learning.
- Data Analysis & Processing Libraries:
- Pandas – for data manipulation and analysis
- NumPy – for numerical computations
- Matplotlib & Seaborn – for data visualization and plotting
- Machine Learning Libraries:

Scikit-learn – for:

- Train-test splitting
- Model building (Logistic Regression, Random Forest)
- Evaluation metrics
- Data preprocessing (StandardScaler, etc.)
- XGBoost – for gradient boosting classification
- Imbalanced-learn – for handling imbalanced data using SMOTE.

Jupiter Notebook:

- Used as the development environment for writing, running, and visualizing code and results interactively.

11. Team Members and Contributions

S.NO	NAMES	ROLES	RESPONSIBILITY
1	LOKESH J	TEAM LEADER	DATA COLLECTING
2	BINAPALLI MANOJ	MEMBER	DATA CLEANING AND FEATURE ENGINEERING
3	GUNA SEKHER REDDY	MEMBER	MODEL EVALUVATION AND MODEL BUILDING

4	POORNA CHANDRA REDDY	MEMBER	VISUALIZATION AND INTERPRETATION
5	ERUGU PURUSHOTHAM	MEMBER	EXPLORATORY DATA ANALYSIS