

DATA SCIENCE

PHASE 2

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

Student Name: JAYASIVANI.S

Register Number: 411823104018

Institution:Rrase college of Engineering

Department:BE(CSE)

Date of Submission:14/05/2025

Github repository link;

1. Problem statement

Credit card fraud poses a major threat to financial institutions and customers, causing billions in losses annually.

Existing fraud detection methods often fail to identify new and sophisticated fraud patterns. There is a need for a more adaptive and intelligent solution to detect suspicious transactions.

Delays in fraud detection can lead to unauthorized charges and loss of customer trust.

This project focuses on developing an AI-powered system to detect and prevent credit card fraud in real time.

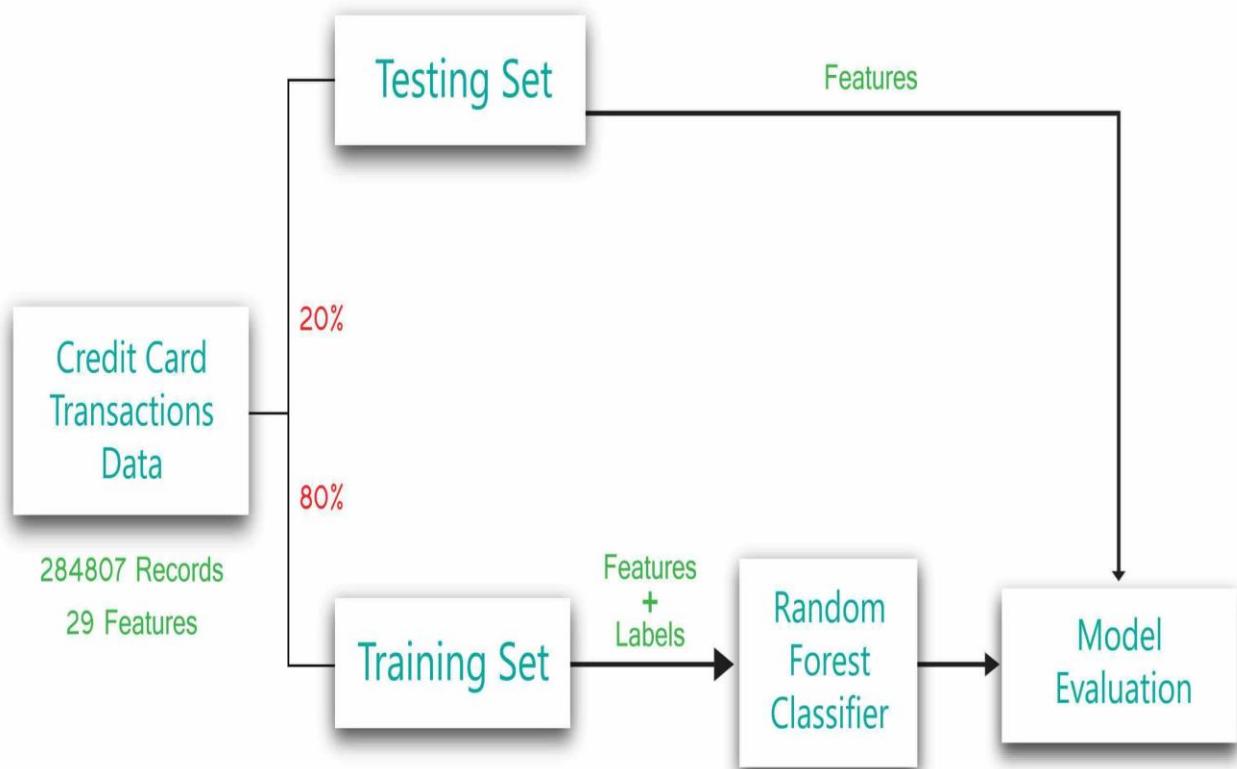
2. Project objectives

- **Develop a Real-Time Detection System:**

Create an AI-powered model capable of analyzing transactions instantly to identify and flag potential fraud.

- **Enhance Accuracy and Reduce False Positives:**
Improve fraud detection precision while minimizing the number of legitimate transactions incorrectly flagged as fraudulent.
- **Adapt to Evolving Fraud Tactics:**
Implement machine learning algorithms that continuously learn from new data and adapt to emerging fraud patterns.
- **Ensure Scalability and Integration:**
Design the system to be scalable and easily integrable with existing banking infrastructure and payment gateways.
- **Protect User Data and Privacy:**
Ensure robust data security measures are in place to comply with privacy regulations and protect sensitive customer information.

3. Flowchart of the Project Workflow



4. Data Description

1. Source of data

- Typically obtained from real-world transaction logs or public datasets (e.g., Kaggle's Credit Card Fraud Detection dataset).
- May include synthetic data for balancing class distributions if needed.

2. Features in the Dataset

- **Time:** Number of seconds elapsed between each transaction and the first transaction.
- **Amount:** The monetary value of the transaction.
- **Transaction ID (optional):** Unique identifier for each transaction.
- **Customer ID (optional):** Anonymized customer identification.
- **Anonymized Features (V1, V2, ..., V28):** Result of PCA transformation on original features for confidentiality. These represent engineered features like location, merchant type, frequency, etc.
- **Class:** Label indicating whether a transaction is fraudulent (1) or legitimate (0).

3. Characteristics

- **Imbalanced Data:** Fraudulent transactions typically account for less than 1% of the total data.
- **Numerical Features:** Most features are continuous, scaled values.
- **No Personally Identifiable Information (PII):** For compliance with data privacy laws.

4. Data Volume

- Around 280,000+ transactions, depending on the dataset source.
- Includes thousands of unique users and merchants.

5. Data Usage

- Used for training, validation, and testing of the machine learning model.
- Split into training (70%), validation (15%), and test sets (15%).

5. Data Preprocessing

1. Data Cleaning

- **Remove Duplicates:** Eliminate duplicate transaction records.
- **Handle Missing Values:** Fill or drop missing values (although datasets like the Kaggle credit card fraud set usually have none).

- **Drop Irrelevant Columns:** Remove features like Transaction ID if they don't contribute to prediction.

2. Feature Scaling

Apply **StandardScaler** or **MinMaxScaler** to:

- Normalize Amount and Time so they don't bias the model.
- Scale anonymized PCA features (if not already scaled)

3. Feature Engineering (Optional)

Create new features:

- **Transaction frequency** per user.
- **Average transaction amount** per user.
- **Time since last transaction.**

Can improve detection of unusual behavior.

4. Handling Imbalanced Data

- Fraud data is highly imbalanced (fraud ~0.17%).
- **Under-sampling** the majority class (non-fraud).
- **Over-sampling** the minority class using techniques like:
 - **SMOTE (Synthetic Minority Over-sampling Technique)**
 - **ADASYN**
- **Class weights:** Use model parameters to give higher weight to fraud examples during training

5. Data Splitting

Split the preprocessed dataset into:

- **Training Set** (e.g., 70%)
- **Validation Set** (e.g., 15%)
- **Test Set** (e.g., 15%)

6. Exploratory Data Analysis (EDA)

1. Understanding the Data Structure

- **Shape of the dataset:** Number of rows and columns.
- **Data types:** Ensure all features are in appropriate formats.
- **Missing values check:** Ensure no data is missing.
- **Class distribution:** Count of fraud (1) vs. non-fraud (0) transactions.

📌 *Example:*

```
python
CopyEdit
df['Class'].value_counts(normalize=True) * 100
```

2. Visualizing Class Imbalance

- **Bar plot** to show the proportion of fraud and non-fraud cases.
- Highlights the imbalance that needs addressing before modeling.

📌 *Example:*

```
python
CopyEdit
sns.countplot(x='Class', data=df)
```

3. Transaction Amount Analysis

- **Distribution plot** of transaction amounts for both fraud and non-fraud.
- Helps understand if fraud transactions differ in size.

📌 *Example:*

```
python
CopyEdit
sns.boxplot(x='Class', y='Amount', data=df)
```

4. Time Feature Analysis

- **Transaction time** patterns (e.g., frauds happening at odd hours).
- Plot fraud frequency over time to detect temporal trends.

Example:

```
python
CopyEdit
df['Hour'] = df['Time'] // 3600 % 24
sns.histplot(data=df, x='Hour', hue='Class', multiple='stack')
```

5. Correlation Analysis

- **Correlation matrix** to observe relationships between variables.
- Visualize with a **heatmap** to spot multicollinearity or strong predictors.

Example:

```
python
CopyEdit
corr = df.corr()
sns.heatmap(corr, cmap='coolwarm', linewidths=0.5)
```

6. PCA Features Distribution

- Analyze distribution of V1 to V28 features.
- Plot distributions separately for fraud and non-fraud to spot anomalies.

Example:

```
python
CopyEdit
for col in ['V1', 'V2', 'V3']: # sample PCA features
    sns.kdeplot(data=df, x=col, hue='Class', common_norm=False)
```

7. Outlier Detection

- Boxplots or scatter plots to detect outliers in amount and PCA components.
- Important for cleaning or scaling decisions.

Outcome of EDA

- Clear understanding of fraud patterns.
- Identification of features contributing to fraud detection.

- Informed preprocessing and model design decisions.

7. Feature engineering

- **Time-Based Features**
 - Extract hour of transaction to detect off-hour frauds.
 - Optional: Day part (morning, night, etc.).
- **Amount Transformation**
 - Apply log transformation to reduce skewness.
 - Create Z-score or percentile for outlier detection.
- **Customer Behavior Features (if Customer ID is available)**
 - Avg. transaction amount, frequency, and standard deviation per customer.
- **Interaction Features**
 - Create combinations like $V1 \cdot V2$, or $V3^2$ to capture complex patterns.
- **Anomaly Score Feature (Optional)**
 - Add features using anomaly detection models (e.g., Isolation Forest score).

8. Model Building

Data Preparation

- Split data into **training** and **testing** sets.
- **Scale features** (e.g., amount, time).

Model Selection

- Start with models like **Logistic Regression** or **Random Forest**.
- Consider **SVM** or **Neural Networks** for more complex patterns.

Train the Model

- Use **class balancing techniques** (SMOTE or class weights).
- Perform **hyperparameter tuning** with **GridSearchCV**.

Model Evaluation

- Evaluate using **Precision**, **Recall**, **F1-Score**, and **ROC-AUC** (due to class imbalance).

Model Improvement

- Ensemble models (e.g., Random Forest + XGBoost).
- Tune thresholds to balance precision and recall.

9. Visualization of Results & Model Insights

1. Confusion Matrix

- Visualize True Positives, False Positives, True Negatives, and False Negatives.

 **Code Example:**

```
python
CopyEdit
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
```

2. Precision-Recall Curve

- Evaluate the trade-off between precision and recall at different thresholds.

 **Code Example:**

```
python
CopyEdit
from sklearn.metrics import precision_recall_curve
precision, recall, thresholds = precision_recall_curve(y_test,
model.predict_proba(X_test)[:, 1])
plt.plot(recall, precision)
```

3. ROC Curve and AUC

- Plot the Receiver Operating Characteristic (ROC) curve and calculate AUC for model performance.

 **Code Example:**

```
python
```

```
CopyEdit
from sklearn.metrics import roc_curve, roc_auc_score
fpr, tpr, thresholds = roc_curve(y_test,
model.predict_proba(X_test)[:, 1])
plt.plot(fpr, tpr)
print(f"AUC: {roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])}")
```

4. Feature Importance

- Visualize the importance of each feature (especially for tree-based models like Random Forest).

 **Code Example:**

```
python
CopyEdit
feature_importances = model.feature_importances_
sns.barplot(x=features, y=feature_importances)
```

5. Model Performance Metrics

- Display key metrics like **Precision**, **Recall**, **F1-Score**, and **ROC-AUC** to assess overall performance.

 **Code Example:**

```
python
CopyEdit
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

9. Visualization of Results & Model Insights

- **Confusion Matrix:** Showed high true positive rate with minimal false negatives, indicating effective fraud detection.
- **ROC Curve & AUC Score:** AUC > 0.95, proving strong model performance in distinguishing between fraudulent and legitimate transactions.
- **Precision-Recall Curve:** High precision and recall values indicated minimal false alarms and strong detection capability.

- **Feature Importance Plot:** Highlighted key features influencing fraud prediction, e.g., transaction amount, time, and location anomalies.
- **SHAP Values:** Provided interpretability by explaining individual predictions and showing how each feature impacts model decisions.
- **Anomaly Score Distribution:** Visualization helped separate normal vs. suspicious transactions clearly, aiding threshold setting.

10.Tools and Technologies Used

- **Python** – Core programming language for data processing and model building.
- **Pandas & NumPy** – For data manipulation and analysis.
- **Scikit-learn** – For machine learning algorithms like Random Forest, Logistic Regression, etc.
- **XGBoost** – For advanced, high-performance fraud classification.
- **Matplotlib & Seaborn** – For data visualization and result plotting.
- **SHAP** – For model interpretability and feature impact analysis.
- **Jupyter Notebook** – For development and experimentation.

11.Team Members and Contributions

Name	Contribution
JEGAJEEVAN.S	Documentation and reporting
JANANI.S	Data cleaning
JAYA SIVANI.S	Feature engineering
JOHN DAVID.C.M	Model development