Guarding transactions with AI-powered credit card fraud detection and prevention

Student Name: KARTHIKEYAN K

Register Number: 510623243012

Institution: C ABDUL HAKEEM COLLEGE OF ENGINEERING AND TECHNOLOGY

Department: BTECH AI&DS

Date of Submission: 08/05/2025

Github Repository Link: ***karthi15724/karthikeyan-Guarding-transactions-with-AI-powered-credit-card-fraud-detection-and-prevention***

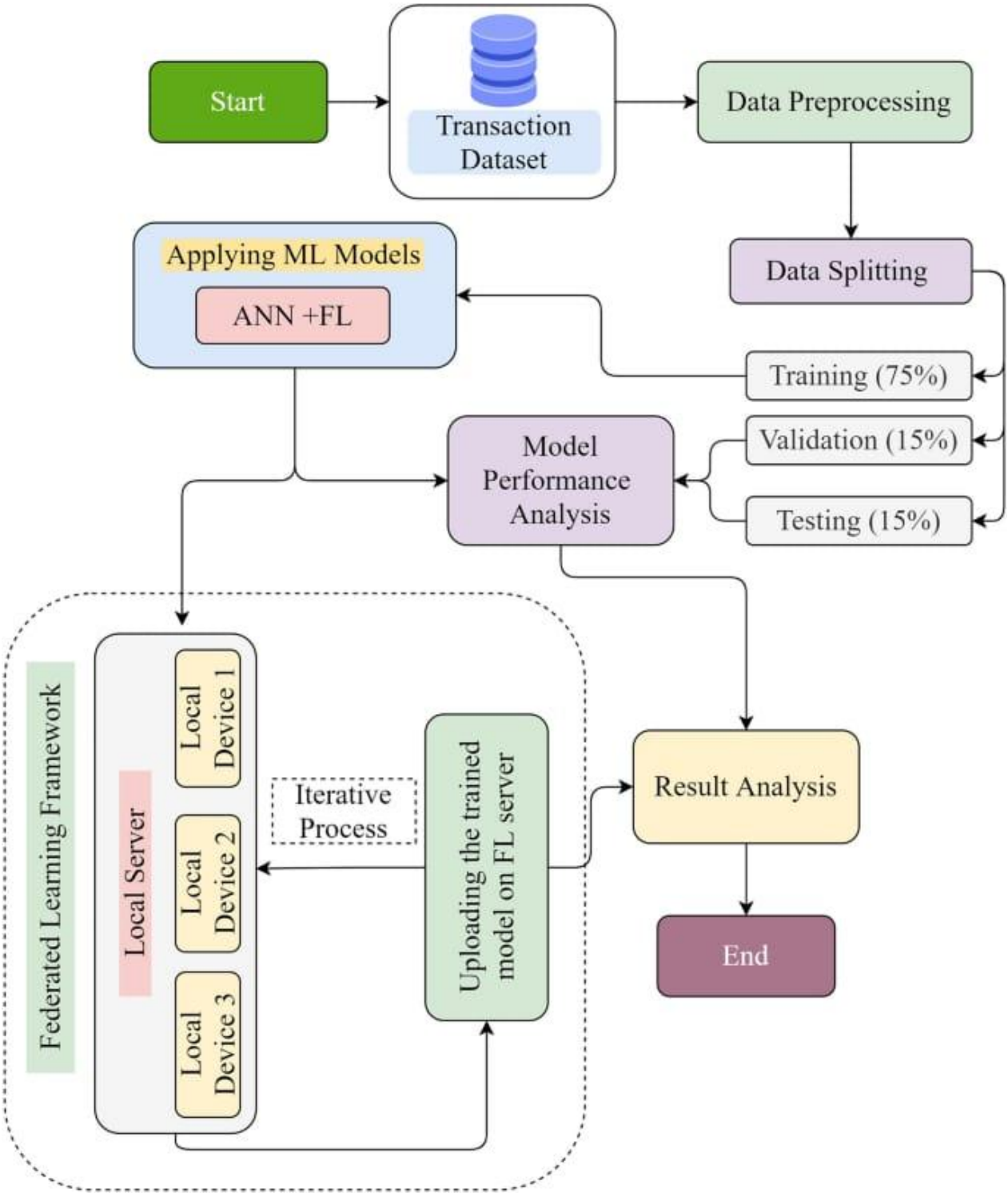
1. Problem Statement

With the rapid growth of digital payments and e-commerce, credit card fraud has become increasingly sophisticated, causing substantial financial losses to consumers and financial institutions alike. Traditional rule-based fraud detection systems often fail to adapt to evolving fraud patterns and generate high false positive rates, which negatively impact customer experience and operational efficiency. There is a pressing need for an intelligent, adaptive system that can accurately detect and prevent fraudulent transactions in real time. This project aims to develop an AI-powered credit card fraud detection and prevention system that leverages machine learning techniques to analyze transaction patterns, identify anomalies, and enhance the security of digital payment ecosystems.

2. Project Objectives

- *Identifying anomalous patterns in transaction data that may indicate fraudulent behavior.*
- *Minimizing false positives to ensure a smooth customer experience without unnecessarily blocking legitimate transactions.*
- *Providing real-time alerts and automated responses to suspicious activities to prevent financial losses.*
- *Improving fraud detection accuracy over time using adaptive learning techniques*
- *Ensuring scalability and security to handle large volumes of transactions across diverse platforms.*

3. Flowchart of the Project Workflow



4. Data Description

1. Transaction ID

Type: String
Description: Unique identifier for each transaction.

2. Timestamp

Type: DateTime
Description: Date and time when the transaction was made.

3. Amount

Type: Float
Description: Monetary value of the transaction.

4. Merchant Category Code (MCC)

Type: Categorical
Description: Type of merchant or service provider.

5. Transaction Type

Type: Categorical
Description: Indicates whether the transaction was online, in-store, contactless, etc.

5. Data Preprocessing

1. Data Cleaning

Remove duplicates: Eliminate any repeated transactions.

Handle missing values: Impute or remove rows with missing data (e.g., missing location or amount).

Correct data types: Ensure numerical, categorical, and datetime values are properly typed.

2. Feature Engineering

Time-based features: Extract features like transaction hour, day of the week, or time since last transaction.

Statistical features: Compute user-specific metrics (average transaction amount, standard deviation).

Geolocation distance: Calculate the distance from previous known location.

Transaction velocity: Number of transactions within a short time frame (e.g., 10 minutes).

3. Encoding Categorical Variables

One-hot encoding: For categorical features like merchant type or transaction type.

Label encoding: For ordinal data (if applicable).

4. Scaling Numerical Features

Standardization (Z-score normalization): Especially important for models like logistic regression or SVM.

Min-Max scaling: Useful for neural networks or distance-based models.

5. Outlier Detection

Identify and handle unusually large or small values that may skew the model.

6. Exploratory Data Analysis (EDA)

1. Understand Data Shape and Structure

- ***df.shape*** *— Check number of rows and columns.*
- ***df.info()*** *— Review data types and missing values.*
- ***df.describe()*** *— Summary statistics for numerical features.*

---

2. Class Distribution (Imbalance Check)

python

CopyEdit

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

- *Visualize with a **bar chart** or **pie chart**.*
- *Fraud cases are typically <1%, highlighting the need for careful modeling.*

---

3. Missing Values

python

CopyEdit

df.isnull().sum().sort_values(ascending=False)

- *Heatmap or bar plot to visualize missingness.*

---

4. Univariate Analysis

- ***Transaction Amount***
  - *Histogram and boxplot.*
  - *Compare distribution for fraud vs. non-fraud.*
- ***Time***
  - *Analyze fraud frequency by time of day (hour-wise plot).*
  - *Example: Are most frauds happening at night?*

---

5. Bivariate Analysis

- ***Fraud vs. Transaction Amount***
    - *Plot amount vs. fraud label using violin or box plots.*
- ***Correlation Matrix***

python

CopyEdit

sns.heatmap(df.corr(), annot=True, cmap="coolwarm")

-

7. Feature Engineering

1. Time-Based Features

- ***Hour of Day / Day of Week:***

python

CopyEdit

df['Hour'] = df['Timestamp'].dt.hour

df['DayOfWeek'] = df['Timestamp'].dt.dayofweek

Fraud is more likely during non-business hours or weekends.

- ***Time Since Last Transaction:***

python

CopyEdit

df = df.sort_values(['CardholderID', 'Timestamp'])

df['TimeDelta'] = df.groupby('CardholderID')['Timestamp'].diff().dt.total_seconds()

---

2. Behavioral Features

- ***Transaction Frequency (Last 24h / 1h):***
  Use rolling windows or count transactions in a time window.

- ***Average Transaction Amount (User-level):***

python

CopyEdit

user_avg = df.groupby('CardholderID')['Amount'].transform('mean')

df['UserAvgAmount'] = user_avg

- ***Amount Deviation from User Average:***

python

CopyEdit

df['AmountDiff'] = df['Amount'] - df['UserAvgAmount']

---

3. Geolocation-Based Features

- ***Distance from Previous Location:***
  If location (latitude/longitude) is available, compute geospatial distance.

- ***Unusual Location Flag:***
  Compare current location to known frequent locations for the cardholder.

---

4. Device & Channel Features

- ***New Device Used:***
  Flag if a new device/browser is used by the cardholder.
- ***Channel Type (Online vs. POS):***
  Encode as binary or one-hot; fraud is more common in certain channels.

---

5. Risk Encoding (Target Encoding)

- ***Merchant Risk Score:***
  Assign a fraud rate to each merchant based on historical data.
- ***Country/IP Risk Level:***
  Encode countries or IPs with high historical fraud rates.

8. Model Building

☑ *1. Define the Problem*

- ***Goal:*** *Predict whether a transaction is fraudulent (1) or legitimate (0)*
- ***Type:*** *Binary classification*
- ***Challenge:*** *Highly imbalanced dataset*

🩹 *2. Train-Test Split*

python

CopyEdit

```
from sklearn.model_selection import train_test_split


X = df.drop('Class', axis=1)  # Features
y = df['Class']               # Target


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42)
```

⚖️ *3. Handle Class Imbalance*

Use either:

- ***Class weights:*** *Supported by many models like Logistic Regression, RandomForest, XGBoost.*
- ***SMOTE (Synthetic Minority Oversampling Technique):***

  python

  CopyEdit

  ```
  from imblearn.over_sampling import SMOTE
  ```

sm = SMOTE(random_state=42)

X_train_res, y_train_res = sm.fit_resample(X_train, y_train)

🧰 *4. Choose and Train Models*

Try several models and compare performance:

A. Logistic Regression

python

CopyEdit

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(class_weight='balanced')

model.fit(X_train_res, y_train_res)
```

B. Random Forest

python

CopyEdit

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100,
class_weight='balanced')

model.fit(X_train_res, y_train_res)
```

C. XGBoost (Recommended for Imbalanced Data)

python

CopyEdit

```
from xgboost import XGBClassifier

model = XGBClassifier(scale_pos_weight=ratio_of_neg_to_pos,
use_label_encoder=False, eval_metric='logloss')

model.fit(X_train, y_train)
```

📊 *5. Evaluate the Model*

Use metrics suitable for imbalanced data:

python

CopyEdit

```
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score


y_pred = model.predict(X_test)

print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))

print("ROC AUC Score:", roc_auc_score(y_test,
model.predict_proba(X_test)[:,1
```

9. Visualization of Results & Model Insights

1. Class Distribution

Shows how imbalanced the dataset is (essential context).

python

CopyEdit

```
sns.countplot(x='Class', data=df)
plt.title("Distribution of Fraud vs Legitimate Transactions")
```

---

2. Transaction Amount by Class

Visualize how fraudulent transactions differ in value.

python

CopyEdit

```
sns.boxplot(x='Class', y='Amount', data=df)
plt.title("Transaction Amounts by Class")
```

---

3. Time of Day vs Fraud Occurrence

Discover time-based fraud patterns.

python

CopyEdit

```
df['Hour'] = df['Timestamp'].dt.hour
sns.histplot(data=df[df['Class']==1], x='Hour', bins=24, kde=True)
plt.title("Fraudulent Transactions by Hour")
```

---

4. Correlation Heatmap

Understand relationships between features.

python

CopyEdit

```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), cmap='coolwarm', annot=False)
plt.title("Feature Correlation Heatmap")
```

---

5. Confusion Matrix

Visualize model performance.

python

CopyEdit

```
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_estimator(model, X_test, y_test)
```

plt.title("Confusion Matrix")

10. Tools and Technologies Used

💻 *Programming Language*

- ***Python*** *– Primary language for data analysis, machine learning, and model deployment.*

📊 *Data Analysis & Visualization*

- ***Pandas*** *– Data manipulation and analysis.*
- ***NumPy*** *– Numerical computing.*
- ***Matplotlib*** *– Static data visualization.*
- ***Seaborn*** *– Advanced visualization (heatmaps, pair plots).*
- ***Plotly*** *(optional) – Interactive charts and dashboards.*

🧠 *Machine Learning & Modeling*

- ***Scikit-learn*** *– Classical ML algorithms (Logistic Regression, Random Forest, SVM).*
- ***XGBoost*** *– Gradient boosting optimized for imbalanced classification.*
- ***Imbalanced-learn (imblearn)*** *– For resampling techniques like SMOTE.*

- ***LightGBM / CatBoost*** *(optional) – Efficient gradient boosting frameworks.*

11. Team Members and Contributions

Name	***Role***	***Responsibilities***
KARTHIKEYAN K	***Project Lead***	*Oversee project development, ensure timely delivery, and manage final documentation.*
PAVITHRAN T	***Data Engineer***	*Collect and preprocess transaction data, balance dataset, and ensure quality*
TAMILSELVAN K	***ML Expert***	*Build and evaluate fraud detection models , tune hyperparameters.*
RASHIQUE K	***Visualization Lead***	*Perform EDA, interpret results, and build performance dashboards.*