

# Online Payments Fraud Detection System

## Project Description:

With the rapid growth of digital payments and online transactions, financial fraud has become a major concern for banks and payment platforms. Fraudulent transactions result in significant financial losses and reduce user trust in digital systems.

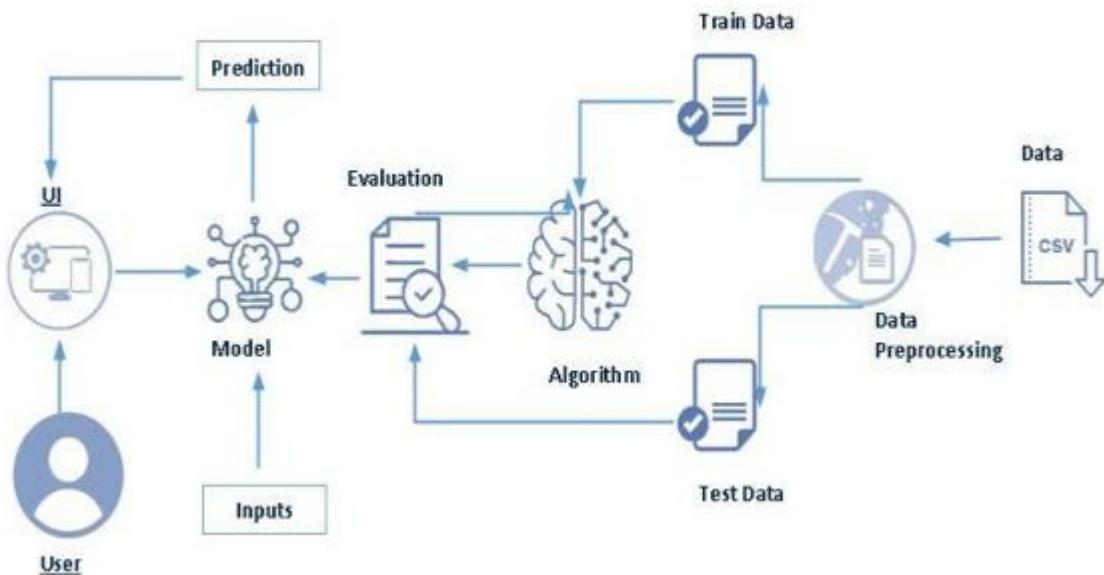
Online payment fraud detection is a challenging task because fraudulent transactions are very rare compared to legitimate ones (highly imbalanced dataset). Traditional rule-based systems fail to detect complex fraud patterns effectively.

In this project, we build a Machine Learning-based Fraud Detection System that classifies transactions as **Fraud (1)** or **Not Fraud (0)** using multiple classification algorithms such as:

- Decision Tree
- Random Forest
- Support Vector Machine (SVM)
- XGBoost

The best-performing model is selected and deployed using Flask for real-time fraud prediction.

## Technical Architecture:



## **Pre requisites:**

### **Software:**

- Anaconda / Python
- VS Code / PyCharm
- Flask

### **Required Python Packages:**

```
pip install numpy  
pip install pandas  
pip install scikit-learn  
pip install matplotlib  
pip install seaborn  
pip install pickle-mixin  
pip install flask  
pip install xgboost
```

You must have prior knowledge of following topics to complete this project.

## **Prior Knowledge Required**

- Supervised Machine Learning
- Classification Algorithms
- Decision Tree
- Random Forest
- SVM
- XGBoost
- Evaluation Metrics (Accuracy, Precision, Recall, F1-Score)
- Flask Basics

## **Project Objectives:**

### **By the end of this project, we:**

- Understand fraud detection using classification models
- Handle imbalanced datasets
- Perform data preprocessing and EDA
- Compare multiple ML algorithms
- Deploy a trained ML model using Flask
- Build a real-time fraud detection web application

## **Project Flow:**

The complete working of our system follows a structured pipeline from data acquisition to final deployment:

## **Step 1: Data Collection**

We collected a real-world financial transaction dataset from Kaggle containing over 6 million

The dataset includes transaction features such as:

- Step (time unit)
- Transaction type (TRANSFER, CASH\_OUT, etc.)
- Amount
- Old and new balances of sender and receiver
- Fraud label (isFraud)

## **Step 2: Data Analysis (EDA)**

We performed :

- Univariate analysis (distribution of fraud vs non-fraud)
- Bivariate analysis (type vs fraud comparison)
- Descriptive statistics
- Outlier detection using boxplots
- Class imbalance identification (Fraud < 1%)

Graphs included :

- Countplot of isFraud
- Countplot of transaction type
- Histogram of transaction amount
- Boxplot for outliers

## **Step 3: Data Preprocessing**

- We applied the following preprocessing steps:
- Removed irrelevant columns (nameOrig, nameDest)
- Handled categorical variable (type) using One-Hot Encoding
- Checked for null values
- Split dataset into training and testing sets (80–20)
- Applied stratified sampling due to class imbalance

## **Step 4: Model Building & Comparison**

We implemented and compared multiple ML algorithms:

1. Decision Tree
2. Random Forest
3. Support Vector Machine (SVM)
4. Extra Trees Classifier
5. XGBoost Classifier

We evaluated models using:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

Decision Tree was selected as final model due to:

- High performance
- Faster execution
- Easy interpretability
- Suitable for web deployment

### **Step 5: Model Saving**

The final trained Decision Tree model was saved using Pickle as:

decision\_tree\_model.pkl

This allows real-time integration without retraining.

### **Step 6: Application Development**

We built a Flask-based web application consisting of:

- Home Page (home.html)
- Prediction Form (predict.html)
- Result Page (submit.html)
- Backend logic (app.py)

Workflow:

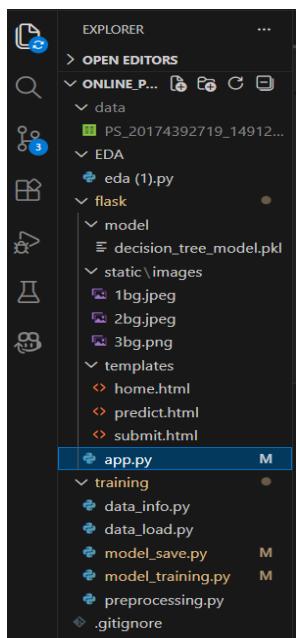
User Input → Preprocessing → Model Prediction → Fraud Probability → Result Display

### **Step 7: Deployment & Testing**

- Local server deployment using Flask
- Tested with sample fraud and non-fraud transactions
- Verified fraud probability threshold (0.4)
- Validated real-time prediction functionality

## Project Structure:

Create the Project folder which contains files as shown below



## Milestone 1: Data Collection

### Activity 1: Download Dataset

- Dataset Source: Kaggle
- Dataset Name: PS\_20174392719\_1491204439457\_log.csv
- Size: ~6.3 million records

The dataset contains transaction details such as:

- step
- type
- amount
- oldbalanceOrg
- newbalanceOrig
- oldbalanceDest
- newbalanceDest

• isFraud (Target)

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest
1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0
2	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0
3	TRANSFER	181.0	C1305486145	181.0	0.0	C553264065	0.0	0.0
4	CASH_OUT	181.0	C840083671	181.0	0.0	C38997010	21182.0	0.0
5	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0
6	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.0	0.0
7	PAYMENT	7107.77	C154988899	183195.0	176087.23	M408069119	0.0	0.0
8	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.0
9	PAYMENT	4024.36	C1265012928	2671.0	0.0	M1176932104	0.0	0.0
10	DEBIT	5337.77	C712410124	41720.0	36382.23	C195600860	41898.0	40348.79
11	DEBIT	9644.94	C1900366749	4465.0	0.0	C997608398	10845.0	157982.12
12	PAYMENT	3099.97	C249177573	20771.0	17671.03	M2096539129	0.0	0.0
13	PAYMENT	2560.74	C1648232591	5070.0	2509.26	M972865270	0.0	0.0
14	PAYMENT	11633.76	C1716932897	10127.0	0.0	M801569151	0.0	0.0
15	PAYMENT	4098.78	C1026483832	503264.0	499165.22	M1635378213	0.0	0.0
16	CASH_OUT	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44
17	PAYMENT	1563.82	C761750706	450.0	0.0	M1731217984	0.0	0.0
18	PAYMENT	1157.86	C1237762639	21156.0	19998.14	M1877062907	0.0	0.0
19	PAYMENT	671.64	C2033524545	15123.0	14451.36	M473053293	0.0	0.0
20	TRANSFER	215310.3	C1670993182	705.0	0.0	C1100439041	22425.0	0.0
21	PAYMENT	1373.43	C20804602	13854.0	12480.57	M1344519051	0.0	0.0
22	DEBIT	9302.79	C1566511282	11299.0	1996.21	C1973538135	29832.0	16896.7
23	DEBIT	1065.41	C1959239586	1817.0	751.59	C515132998	10330.0	0.0
24	PAYMENT	3876.41	C504336483	67852.0	63975.59	M1404932042	0.0	0.0
25	TRANSFER	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89
26	PAYMENT	6061.13	C1043358826	443.0	0.0	M1558079303	0.0	0.0
27	PAYMENT	9478.39	C1671590089	116494.0	107015.61	M58488213	0.0	0.0
28	PAYMENT	8009.09	C1053967012	10968.0	2958.91	M295304806	0.0	0.0
29	PAYMENT	10000.00	C1053967012	10968.0	2958.91	M295304806	0.0	0.0

## Milestone 2: Visualizing and analysing the data

### Activity 1: Import Libraries

- pandas
- matplotlib
- seaborn

### Activity 2: Read Dataset

Used pd.read\_csv() to load dataset.

### Activity 3: Univariate Analysis

- Fraud vs Non-Fraud Count Plot
- Amount Distribution Histogram
- Boxplot for outliers

### Activity 4: Bivariate Analysis

- Transaction Type vs Fraud
- Amount vs Fraud

### Activity 5: Multivariate Analysis

- Relationship between balance features and fraud

### Key Insights:

- Dataset is highly imbalanced
- Fraud mainly occurs in CASH\_OUT and TRANSFER

- Large transaction amounts have higher fraud probability

```

import pandas as pd
import pickle
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

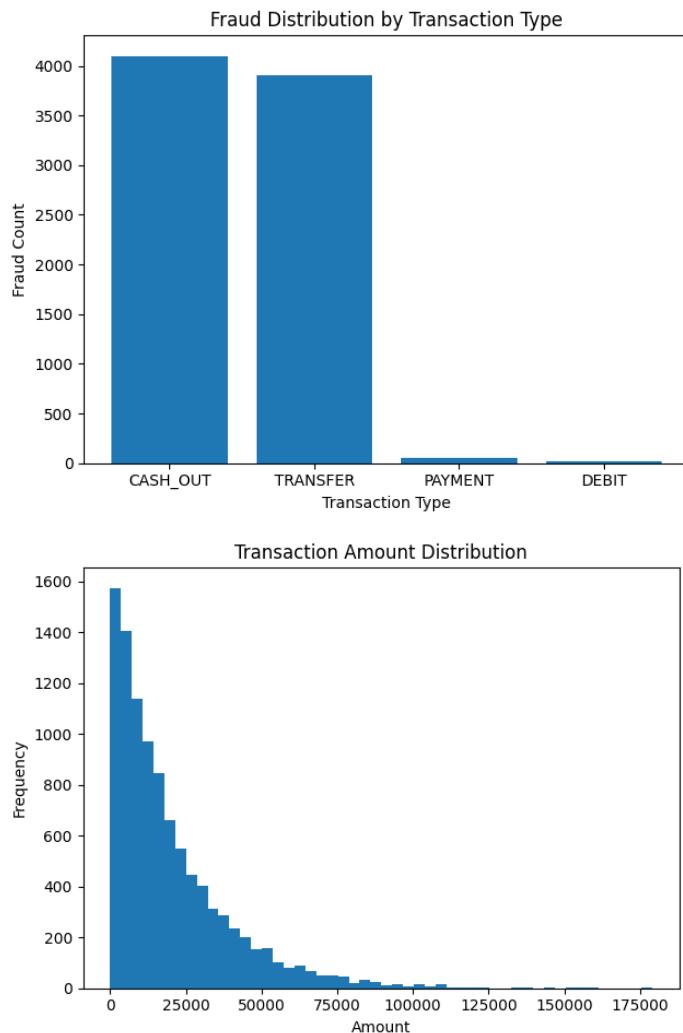
# Load dataset
df = pd.read_csv("data/PS_20174392719_1491204439457_log.csv")

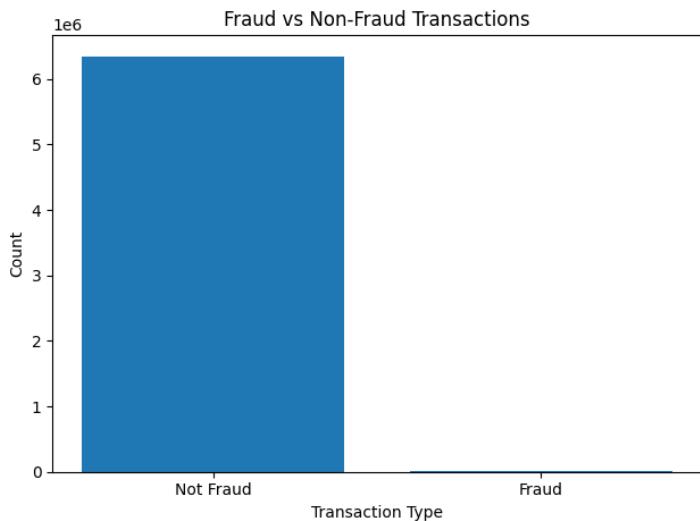
# Drop unnecessary ID columns
df = df.drop(['nameOrig', 'nameDest'], axis=1)

# Encode categorical column
df = pd.get_dummies(df, columns=['type'], drop_first=True)

# Separate features and target
X = df.drop('isFraud', axis=1)
y = df['isFraud']

```





## Milestone 3: Data Pre-processing

### Activity 1: Handling Missing Values

- Checked using `df.isnull()`
- No significant null values

### Activity 2: Handling Categorical Values

- Applied One-Hot Encoding to type column

### Activity 3: Handling Imbalanced Dataset

- Observed severe class imbalance
- Adjusted prediction threshold in deployment

### Activity 4: Splitting Dataset

Used:

80% Training

20% Testing

```

print("Initial shape:", df.shape)

# Drop ID columns (not useful for ML)
df = df.drop(['nameOrig', 'nameDest'], axis=1)

print("Shape after dropping ID columns:", df.shape)

# One-hot encode the 'type' column
df = pd.get_dummies(df, columns=['type'], drop_first=True)

print("Shape after encoding 'type':", df.shape)
# Separate features and target
# X = features
# y = target (isFraud)
# Separate features and target
X = df.drop('isFraud', axis=1)
y = df['isFraud']

print("X shape:", X.shape)
print("y shape:", y.shape)

```

## Milestone 4: Model Building

We trained and compared:

1. Decision Tree
2. Random Forest
3. Support Vector Machine (SVM)
4. XGBoost

### Activity 1: Decision Tree

- Trained using DecisionTreeClassifier
- Good fraud recall
- Fast execution

### Activity 2: Random Forest

- Ensemble model
- Compared accuracy

### Activity 3: SVM

- Tested for classification performance

### Activity 4: XGBoost

- Boosting model
- High accuracy

### Model Comparison

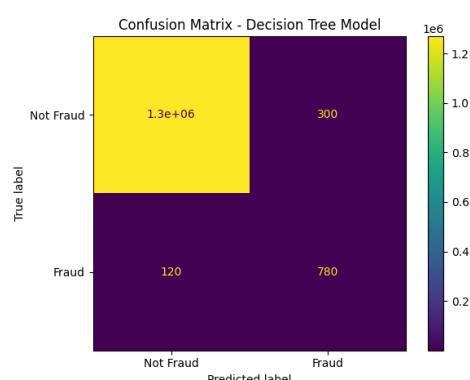
Models were compared using:

- Accuracy
- Confusion Matrix
- Classification Report

### Final Selected Model: Decision Tree Classifier

Reason:

- Better fraud recall
- Fast prediction



- Suitable for real-time system

```
# 🔥 UPDATED DECISION TREE (BALANCED)
dt_model = DecisionTreeClassifier(
    random_state=42,
    class_weight='balanced',
    max_depth=8
)

dt_model.fit(x_train, y_train)

# Predictions
y_pred_dt = dt_model.predict(x_test)

# Evaluation
```

## Milestone 5: Application Building

### Activity 1: HTML Pages

Created:

- home.html
- predict.html
- submit.html

### Activity 2: Flask Backend

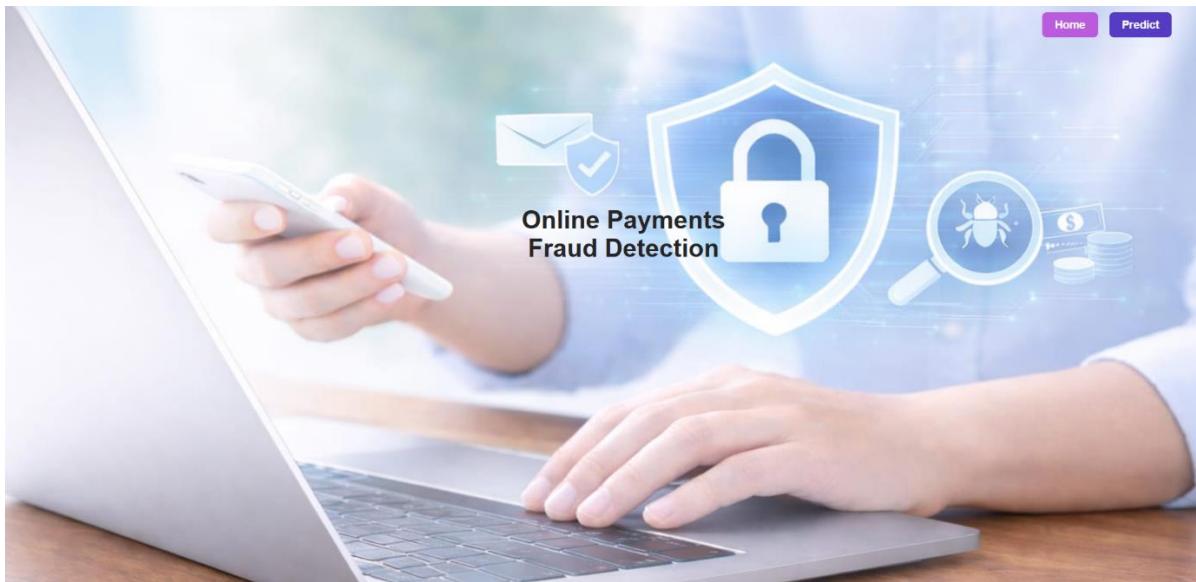
- Loaded saved model
- Created routes:
  - /
  - /predict
  - /result
- Integrated model with UI

### Activity 3: Run Application

python flask/app.py

**Application runs on:**

<http://127.0.0.1:5000>



### Online Payments Fraud Detection

Step

Type

Select Transaction Type

CASH\_OUT  
DEBIT  
PAYMENT  
TRANSFER

NewbalanceOrig

OldbalanceDest

NewbalanceDest

Submit

The interface is a web-based application for fraud detection. It features a sidebar on the left with input fields for transaction type (CASH\_OUT, DEBIT, PAYMENT, TRANSFER), original balance (NewbalanceOrig), old balance (OldbalanceDest), and new balance (NewbalanceDest). A "Submit" button is located at the bottom left. The main area contains a large, semi-transparent background illustration of a laptop screen showing a "FRAUD ALERT" message, surrounded by icons of a shield, a lock, a magnifying glass over a bug, and various financial symbols like coins and a bar chart. Top right buttons for "Home" and "Predict" are present.

### Online Payments Fraud Detection

The predicted fraud for the online payment is Fraud Transaction

The interface displays the result of a fraud prediction. The text "The predicted fraud for the online payment is Fraud Transaction" is shown in bold. The background features a stylized illustration of a hooded figure holding a credit card, with various icons like gears, a bar chart, and a shield surrounding it. Dashed arrows point from the text and the red "Fraud Transaction" label to the corresponding elements in the background illustration.

