Online Payments Fraud Detection using Machine Learning

Project Description:

With the rapid adoption of digital payment methods, online transactions have become a fundamental part of everyday life. Consumers now prefer using credit/debit cards, UPI, mobile wallets, and online banking for their convenience, speed, and accessibility. While these payment systems provide numerous benefits, they are also vulnerable to fraudulent activities. Fraudulent transactions not only lead to significant financial losses for banks, fintech companies, merchants, and consumers but also damage the trust in digital payment systems.

The Online Payments Fraud Detection System is designed to address this critical problem by leveraging the power of Machine Learning (ML). Unlike traditional rule-based fraud detection methods, which rely on static rules and manual monitoring, machine learning allows the system to automatically learn from historical transaction data, identify hidden patterns, and detect anomalies that may indicate fraud. This proactive approach improves detection accuracy, reduces false positives, and enables real-time fraud prevention.

In this project, fraud detection is approached as a classification problem, where each transaction is assigned a label based on its legitimacy:

- Legitimate Transaction (0): The transaction is genuine and performed by an authorized user.
- Fraudulent Transaction (1): The transaction is unauthorized, abnormal, or suspicious, indicating potential fraud.

The system uses a variety of supervised machine learning algorithms to model transaction behavior and classify transactions accordingly. The algorithms implemented in this project include:

1. Decision Tree: A tree-based model that splits data based on feature thresholds to classify transactions into legitimate or fraudulent categories.
2. Random Forest: An ensemble of decision trees that improves prediction accuracy and reduces overfitting.
3. K-Nearest Neighbors (KNN): A distance-based algorithm that predicts the class of a transaction by comparing it to its nearest neighbors in the dataset.
4. XGBoost / Gradient Boosting: A powerful boosting algorithm that combines multiple weak learners to produce a highly accurate model, especially effective for imbalanced datasets.

Each model is trained on historical transaction data and evaluated using standard performance metrics, such as:

- Accuracy: The overall correctness of the model.
- Precision: The proportion of correctly predicted fraudulent transactions among all predicted frauds.
- Recall: The proportion of correctly identified fraudulent transactions out of all actual frauds.

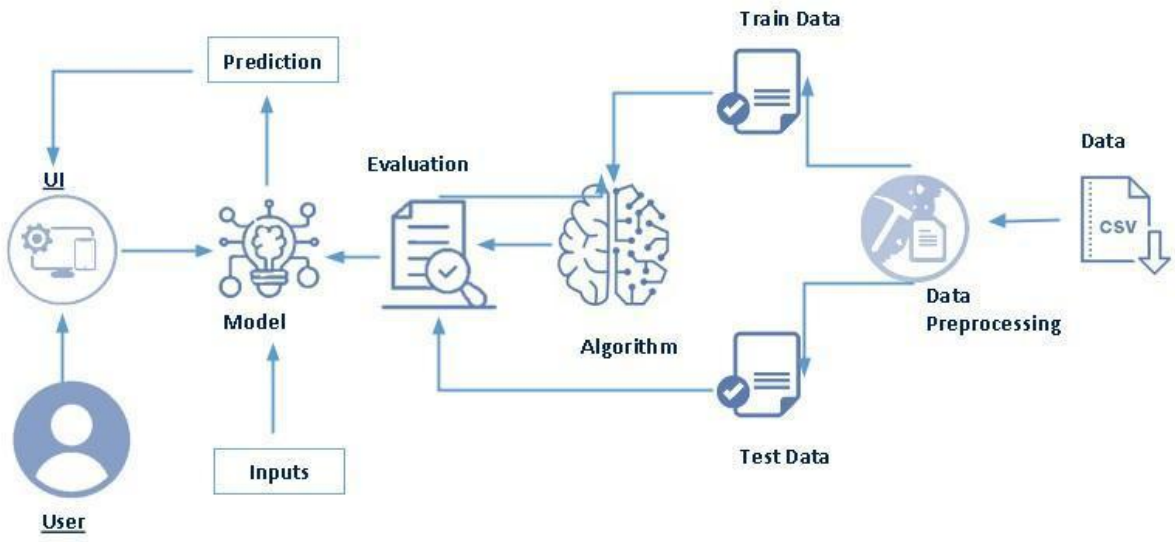
- F1-Score: The harmonic mean of precision and recall, providing a balanced measure of model performance.
- Cross-Validation Score: Used to assess the model's stability and generalization on unseen data.

After evaluating all models, the best-performing model is selected based on these metrics. This model is then saved in a .pkl (pickle) file for deployment. To make the system accessible to users, the model is integrated into a Flask web application.

The Flask application provides a user-friendly interface where users can enter transaction details such as transaction amount, time, location, device type, and payment method. Upon submission, the system processes the input, feeds it into the trained ML model, and predicts whether the transaction is legitimate or fraudulent. This enables real-time fraud detection, allowing banks and financial institutions to take immediate action to prevent financial losses and protect customers.

Overall, this project demonstrates how machine learning can enhance financial security, reduce operational losses due to fraud, and build trust in digital payment platforms. By automating fraud detection and providing real-time prediction, the system acts as a robust tool for safeguarding online transactions while adapting to evolving fraud patterns.

Technical Architecture:



Pre requisites:

To complete this project, you will require the following software, concepts, and packages

Anaconda navigator:

- Refer to the link below to download the anaconda navigator
- Link : https://youtu.be/1ra4zH2G4o0

Python packages:

Open anaconda prompt as administrator

- Type “pip install numpy” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type ”pip install matplotlib” and click enter.
- Type ”pip install scipy” and click enter.
- Type ”pip install pickle-mixin” and click enter.
- Type ”pip install seaborn” and click enter.
- Type “pip install Flask” and click enter.

Prior Knowledge:

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

- **ML Concepts**
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
  - Regression and classification
  - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
  - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
  - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
  - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
  - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding of data.

- Have knowledge of pre-processing the data/transformation techniques and some visualization concepts before building the model
- Learn how to build a machine learning model and tune it for better performance
- Know how to evaluate the model and deploy it using flask

Project Flow:

- The user interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- The predictions made by the model are showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Data collection
    - Collect the dataset or create the dataset
- Data pre-processing
    - Removing unnecessary columns
    - Checking for null values
- Visualizing and analyzing data
- Univariate analysis
- Bivariate analysis
- Descriptive analysis
- Model building
    - Handling categorical values
    - Dividing data into train and test sets
    - Import the model building libraries
    - Comparing the accuracy of various models
    - Hyperparameter tuning of the selected model
    - Evaluating the performance of models
    - Save the model
- Application Building
    - Create an HTML file
    - Build python code

Project Structure:

Create the Project folder which contains files as shown below

```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- rdf.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains model training files and training_ibm folder contains IBM deployment files.

Milestone 1: Data Collection

Machine Learning models depend heavily on data. Data is the most crucial component that allows algorithms to learn patterns and make predictions. For fraud detection, historical transaction data is required to identify fraudulent behavior patterns.

Activity 1: Download the Dataset

There are many popular open sources for collecting datasets:

- Kaggle.com
- UCI Machine Learning Repository
- Google Dataset Search

- Financial open datasets

In this project, we have used a credit card / online transaction fraud dataset downloaded from Kaggle.

Example Dataset:

- Credit Card Fraud Detection Dataset
- Online Payment Fraud Dataset

The dataset contains transaction-related features such as:

- Transaction ID
- Transaction Amount
- Transaction Time
- Merchant ID
- Location
- Device Type
- Payment Mode
- Fraud Label (0 – Legit, 1 – Fraud)

Link: https://share.google/FSLy6l4B0FN506G4r

Milestone 2: Visualizing and Analyzing the Data

After downloading the dataset, the next step is understanding the data using visualization and analysis techniques.

Note: There are many techniques to analyze data. Here we have used commonly used methods.

Activity 1: Importing Libraries

Import the necessary libraries:

- numpy
- pandas
- matplotlib
- seaborn
- sklearn

(Optional) Visualization style can be set to **f**ivethirtyeight for better graphical representation.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.svm import SVC

import xgboost as xgb

from sklearn.metrics import f1_score

from sklearn.metrics import classification_report, confusion_matrix

import warnings

import pickle
```

Activity 2: Reading the Dataset

Since our dataset is in .csv format, we use pandas to read it.

- pd.read_csv() function is used.
- File path of the dataset is passed as parameter.

After loading:

- Check first 5 rows using df.head()
- Check dataset structure using df.info()

```
import pandas as pd

df = pd.read_csv('PS_20174392719_1491204439457_log.csv', on_bad_lines='skip')
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFra
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0.0	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0.0	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1.0	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1.0	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0.0	

```
df.columns
```

```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

```
df.drop(['isFlaggedFraud'],axis =1,inplace = True)
```

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	0.0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	0.0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	1.0

Activity 3: Univariate Analysis

Univariate analysis means analyzing one feature at a time.

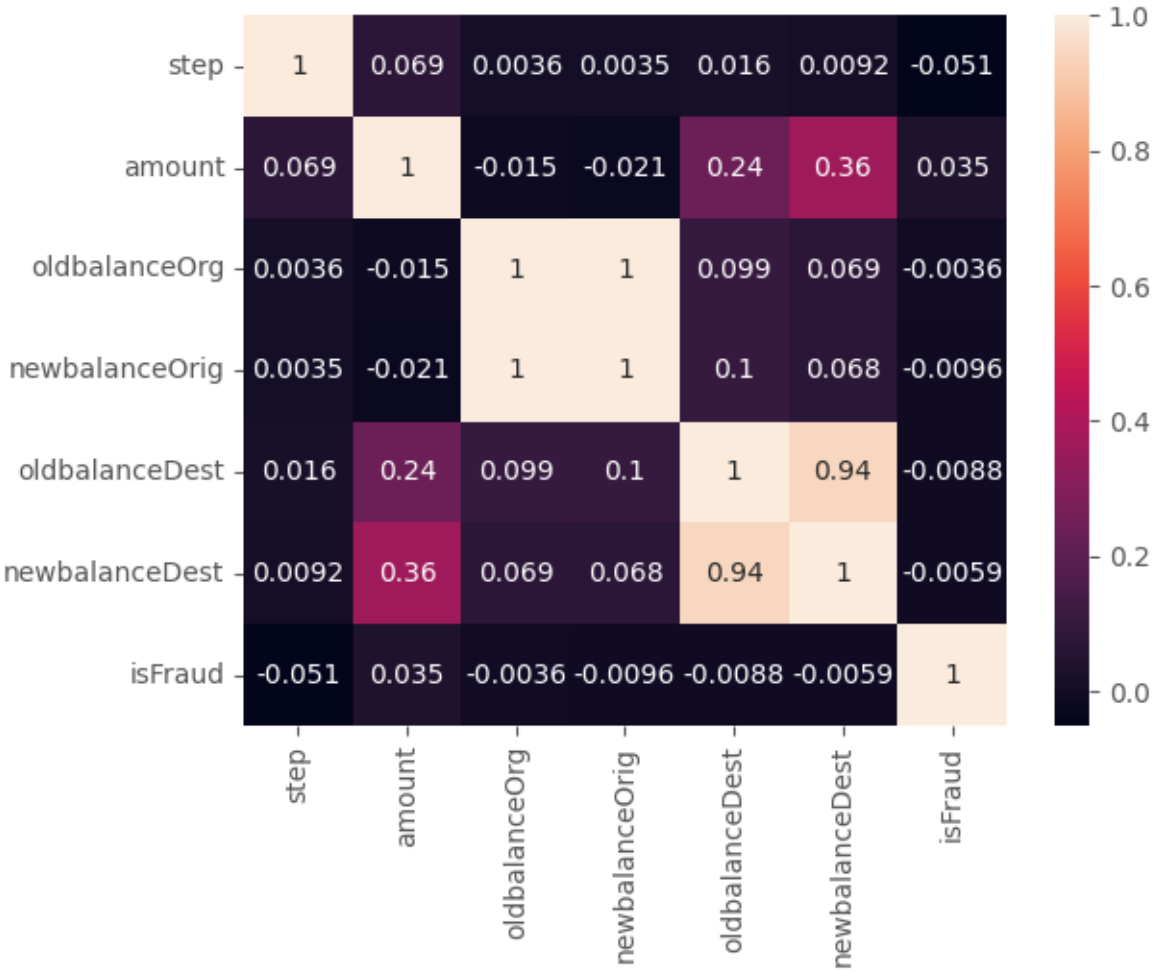
1 Distribution Plot (distplot / histplot)

Used to check distribution of numerical features such as:

- Transaction Amount
- Transaction Time

From distribution plot we can infer:

- Transaction amount is right-skewed.
- Most transactions are small amounts.
- Fraud cases are usually associated with unusual amounts.



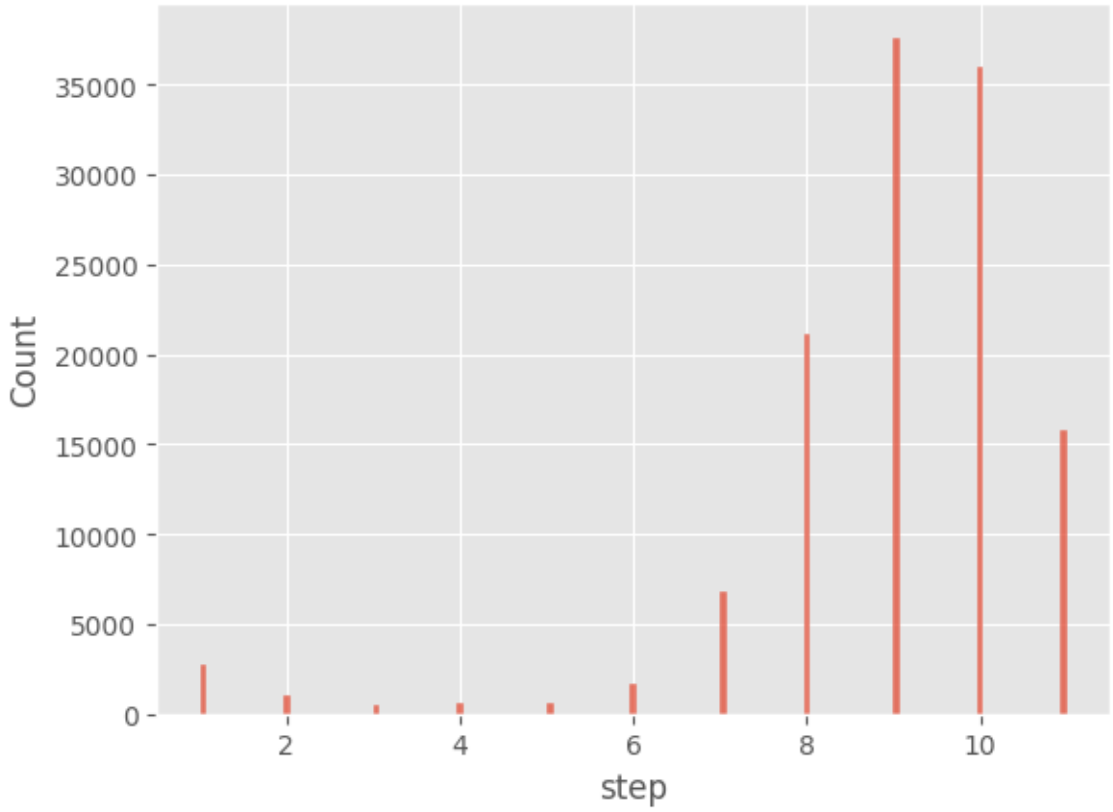
2 Countplot (for Categorical Features)

Countplot is used for categorical features like:

- Payment Method
- Device Type
- Location
- Fraud Label

From countplot we can infer:

- Majority transactions are legitimate.
- Fraud transactions are very less (highly imbalanced dataset).
- Certain payment modes show higher fraud occurrences.



Activity 4: Bivariate Analysis

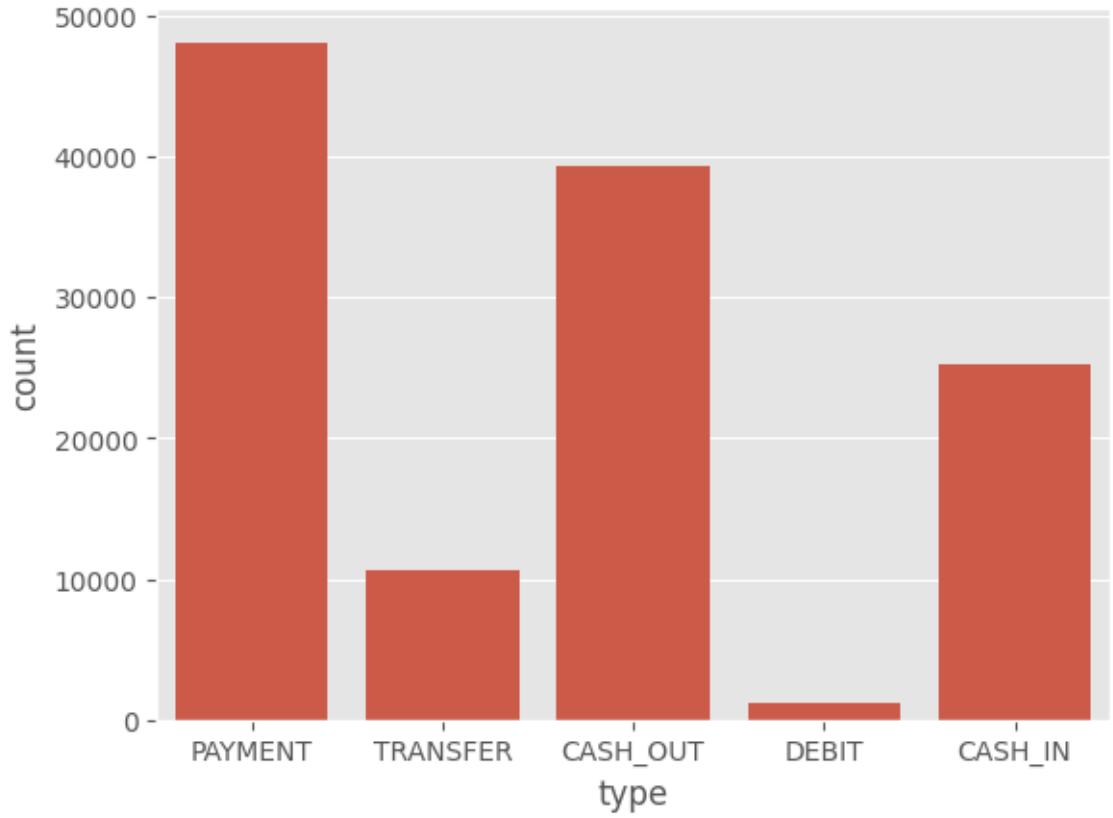
Bivariate analysis studies relationship between two variables.

Examples:

- Transaction Amount vs Fraud Label
- Payment Method vs Fraud Label
- Location vs Fraud Label

From graphs we can infer:

- High-value transactions have higher fraud probability.
- Certain locations may show unusual fraud patterns.
- Online transactions show more fraud than POS transactions.



Activity 5: Multivariate Analysis

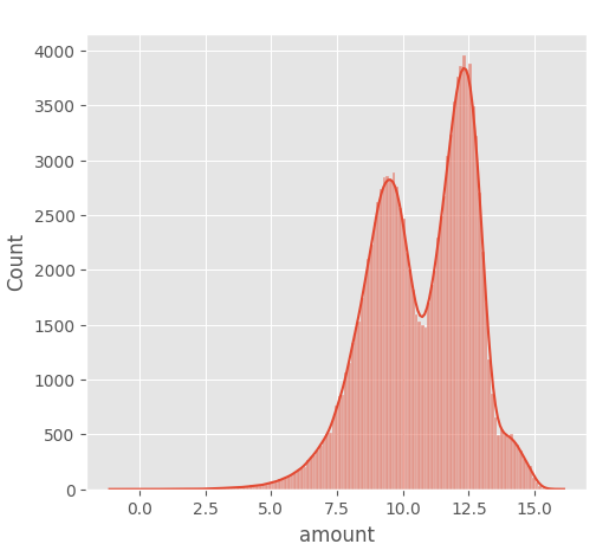
Multivariate analysis studies relationship among multiple features.

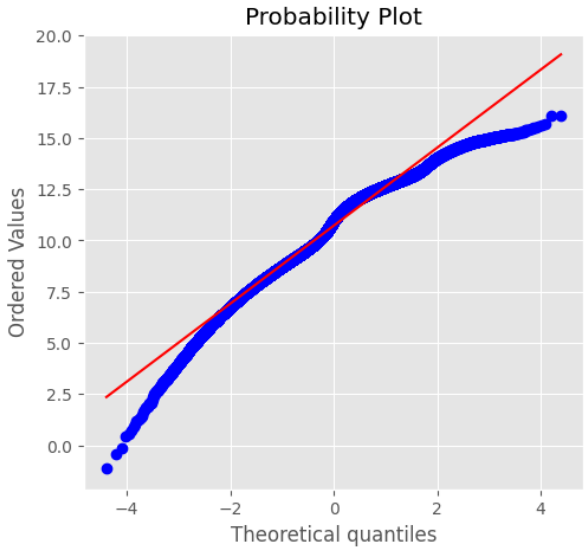
We use:

- Heatmap (Correlation Matrix)
- Swarmplot / Pairplot

From correlation heatmap we can:

- Identify highly correlated features
- Remove redundant features
- Understand which features influence fraud detection most





Activity 6: Descriptive Analysis

Descriptive statistics help understand dataset characteristics.

Using df.describe() we get:

For numerical features:

- Mean
- Standard Deviation
- Min
- Max
- Percentiles

For categorical features:

- Unique values
- Top value
- Frequency

From this we understand:

- Fraud cases are very small compared to legit cases.
- Transaction amount has high variance.

Milestone 3: Data Pre-processing

The raw dataset cannot be directly used for training because it may contain:

- Missing values
- Categorical data
- Imbalanced classes
- Outliers
- Different data ranges

So preprocessing is required.

Activity 1: Checking for Null Values

- df.shape() → check dataset size
- df.info() → check data types
- df.isnull().sum() → check missing values

If missing values are present:

- Numerical columns → Fill with Mean/Median
- Categorical columns → Fill with Mode

```
df_numeric = df.select_dtypes(include=['number'])
df_numeric.corr()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
step	1.000000	0.069450	0.003578	0.003460	0.015715	0.009159	-0.050677
amount	0.069450	1.000000	-0.015391	-0.020991	0.236828	0.364296	0.035281
oldbalanceOrg	0.003578	-0.015391	1.000000	0.998985	0.099253	0.068603	-0.003552
newbalanceOrig	0.003460	-0.020991	0.998985	1.000000	0.100765	0.067591	-0.009606
oldbalanceDest	0.015715	0.236828	0.099253	0.100765	1.000000	0.943255	-0.008841
newbalanceDest	0.009159	0.364296	0.068603	0.067591	0.943255	1.000000	-0.005916
isFraud	-0.050677	0.035281	-0.003552	-0.009606	-0.008841	-0.005916	1.000000

Activity 2: Handling Categorical Values

Machine learning models require numerical input.

So we convert categorical features using:

- Label Encoding
- One-Hot Encoding

Features like:

- Payment Method
- Device Type
- Location

are encoded into numerical format.

```
from sklearn.preprocessing import LabelEncoder
la= LabelEncoder()
df['type']=la.fit_transform(df['type'])
```

```
df['type'].value_counts()
```

type	count
3	48078
1	39349
0	25209
4	10650
2	1171

dtype: int64

Activity 3: Handling Imbalanced Dataset

Fraud detection datasets are highly imbalanced:

Example:

- 98% Legitimate
- 2% Fraud

If trained on imbalanced data, the model will predict only majority class.

So we use:

SMOTE (Synthetic Minority Oversampling Technique)

SMOTE creates synthetic fraud samples using KNN method.

After applying SMOTE:

- Fraud and Legitimate classes become balanced.

Activity 4: Scaling the Data

Scaling is important because:

- Transaction amount may be in thousands
- Other features may be in small range

Algorithms like:

- KNN
- Logistic Regression
- Gradient Boosting

require scaled data.

We use:

- StandardScaler
  or
- MinMaxScaler

After scaling, data is converted back into DataFrame.

Activity 5: Splitting the Dataset

Dataset is split into:

- X → Independent Variables

- y → Target Variable (Fraud Label)

Using:

train_test_split()

Parameters:

- test_size = 0.2
- random_state = 42

This creates:

- 80% Training Data
- 20% Testing Data

```
from sklearn.model_selection import train_test_split

x_train , x_test , y_train , y_test = train_test_split(x,y,random_state=0,test_size=0.2)
```

```
print(x_train.shape)

print(x_test.shape)

print(y_test.shape)

print(y_train.shape)
```

```
(99564, 9)
(24892, 9)
(24892,)
(99564,)
```

Milestone 4: Model Building

Now the data is ready for training.

We apply four classification algorithms:

Activity 1: Decision Tree

- Initialize DecisionTreeClassifier
- Train using .fit()
- Predict using .predict()
- Evaluate using confusion matrix & classification report

Activity 2: Random Forest

- Initialize RandomForestClassifier

- Train and predict
- Evaluate performance

Random Forest reduces overfitting and improves accuracy.

Activity 3: KNN

- Initialize KNeighborsClassifier
- Train model
- Predict results
- Evaluate metrics

KNN works based on distance, so scaling is important.

Activity 4: XGBoost / Gradient Boosting

- Initialize XGBClassifier or GradientBoostingClassifier
- Train model
- Predict results
- Evaluate performance

Boosting improves performance by correcting previous errors.

Activity 5: Comparing Models

A compareModel() function is defined to compare:

- Accuracy
- Precision
- Recall
- F1 Score

From results:

XGBoost gives highest performance.

Example:

- Training Accuracy: 95%
- Testing Accuracy: 90%
- High Recall for Fraud Class

Since recall is very important in fraud detection, we select XGBoost.

```
import xgboost as xgb

xgb1=xgb.XGBClassifier()

# Drop the 'nameOrig' and 'nameDest' columns from x as they are non-numeric identifiers
x_train_processed_xgb = x_train.drop(['nameOrig', 'nameDest'], axis=1)
x_test_processed_xgb = x_test.drop(['nameOrig', 'nameDest'], axis=1)

xgb1.fit(x_train_processed_xgb, y_train1)

y_test_predict5 = xgb1.predict(x_test_processed_xgb)

test_accuracy=accuracy_score(y_test1,y_test_predict5)

test_accuracy
```

```
0.9995179174031817
```

```
y_train_predict5=xgb1.predict(x_train_processed_xgb)

train_accuracy=accuracy_score(y_train1,y_train_predict5)

train_accuracy
```

```
0.9999899562090716
```

```
pd.crosstab(y_test1,y_test_predict5)
```

Activity 6: Cross Validation & Saving Model

We use:

cross_val_score() with 5 folds

After confirming model stability:

We save model using:

pickle.dump(model, open('fraud_model.pkl', 'wb'))

Milestone 5: Application Building

Now we integrate the trained model into a web application.

This includes:

- HTML Frontend
- Flask Backend

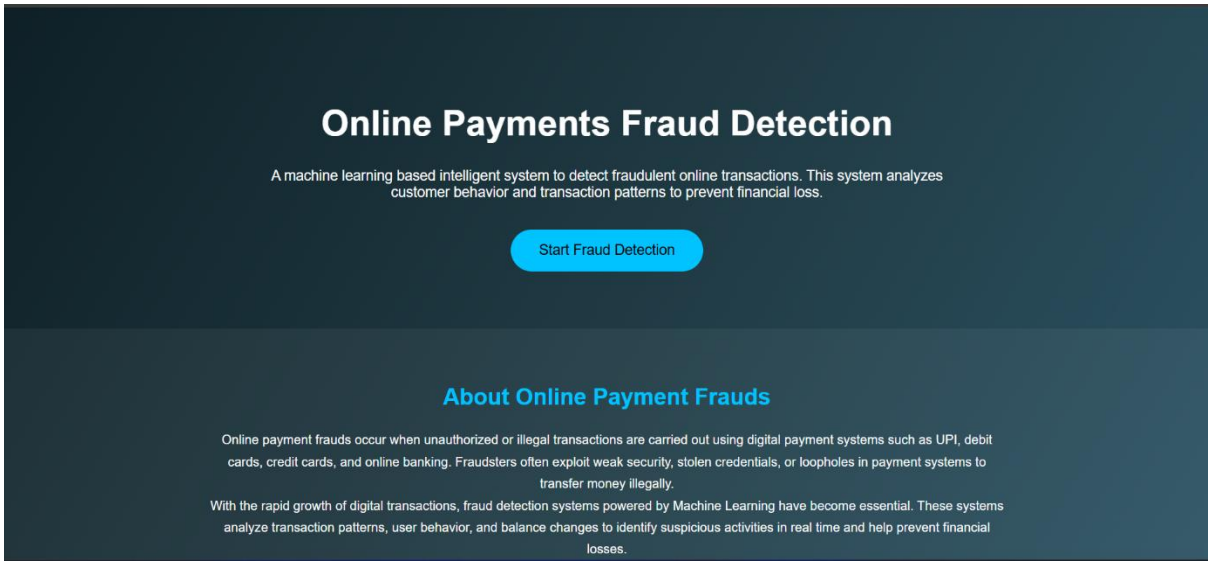
Activity 1: Building HTML Pages

Create three HTML files inside templates folder:

1. home.html
2. predict.html
3. result.html

- **home.html**

- Project Title: Online Fraud Detection System
- Short Description
- Navigation button to Predict page



🔹 **predict.html**

User inputs:

- Transaction Amount
- Payment Method
- Device Type
- Location
- Transaction Time

When user clicks **Submit**, data is sent to Flask backend.

- **result.html**

Displays:

- ✅ Legitimate Transaction
  OR
- ⚠ Fraudulent Transaction Detected

Based on model prediction.