



# ONLINE PAYMENTS FRAUD DETECTION USING MACHINE LEARNING

## **Team Members**

Ponn Oviyaa S

Srikakolapu Veera Venkata Amaradh Babu

Saksham Khurana

Harshavardhanan R

# Online Payments Fraud Detection using Machine Learning

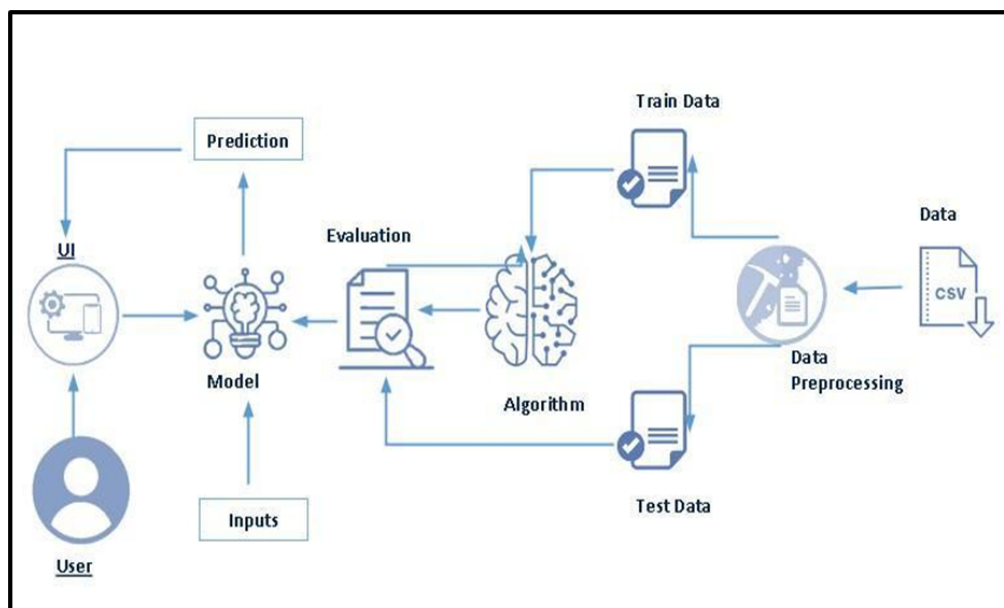
Online Payments Fraud Detection using Machine Learning is a proactive approach to identify and prevent fraudulent activities during online transactions. By leveraging historical transaction data, customer behavior patterns, and machine learning algorithms, this project aims to detect potential fraud in real time, ensuring secure and trustworthy online payment experiences for users and businesses alike.

**Scenario 1: Real-time Fraud Monitoring** The system continuously monitors online payment transactions in real time. By analyzing transaction features such as transaction amount, location, device information, and user behavior, it can flag suspicious transactions for further investigation, preventing fraudulent activities before they occur.

**Scenario 2: Fraudulent Account Detection** Machine learning models can detect patterns indicative of fraudulent accounts or activities. By analyzing user behavior over time, such as unusual login times, multiple failed login attempts, or sudden changes in spending patterns, the system can identify and block potentially fraudulent accounts, protecting legitimate users and businesses.

**Scenario 3: Adaptive Fraud Prevention** The system adapts and improves its fraud detection capabilities over time. By continuously learning from new data and adjusting its algorithms, it can stay ahead of evolving fraud techniques and trends, providing ongoing protection against online payment fraud for businesses and their customers.

## Technical Architecture



## Project Flow

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

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

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

## Prior Knowledge

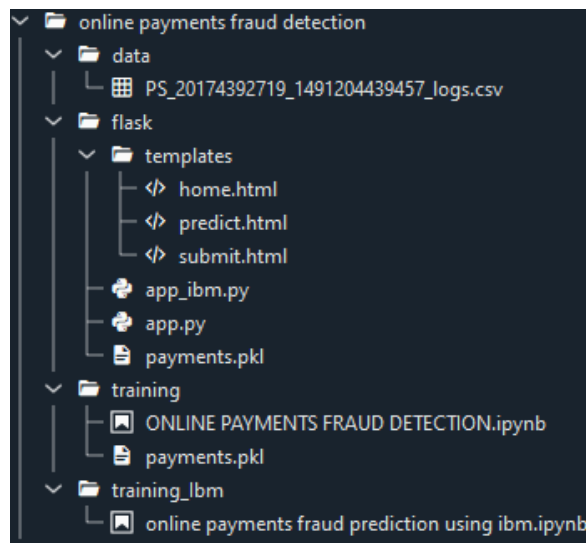
You must have prior knowledge of the following topics to complete this project.

1. ML Concepts

- a. Supervised learning: <https://www.javatpoint.com/supervised-machine-learning>
- b. Unsupervised learning: <https://www.javatpoint.com/unsupervised-machine-learning>
2. Decision tree: <https://www.javatpoint.com/machine-learning-decision-tree-classificationalgorithm>
3. Random forest: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>
4. KNN: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
5. Xgboost: <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>
6. Evaluation metrics: <https://www.analyticsvidhya.com/blog/2019/08/11-important-modelevaluation-error-metrics/>
7. Flask Basics : [https://www.youtube.com/watch?v=lj4I\\_CvBnt0](https://www.youtube.com/watch?v=lj4I_CvBnt0)

## Project Structure

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- payments.pkl is our saved model. Further we will use this model for flask integration.
- Data Folder contains the Dataset used
- The Notebook file contains procedure for building the model.

## **Milestone 1: Define Problem / Problem Understanding**

### **Activity 1: Specify the business problem**

With the surge in digital transactions, financial institutions and online businesses face significant challenges in identifying and preventing fraudulent transactions. Fraudulent payments not only result in direct financial losses but also damage customer trust and brand reputation. The core business problem is the lack of an effective, real-time system that can intelligently detect and prevent online payment fraud without disrupting the user experience for legitimate customers.

### **Activity 2: Business requirements**

The project must fulfil the following key business requirements:

- **Real-time Detection:** The system should analyze transactions in real time and instantly flag potentially fraudulent activity.
- **High Accuracy:** It should minimize false positives and false negatives to reduce both customer inconvenience and fraud risk.
- **Scalability:** The system should be scalable to handle a high volume of concurrent transactions.
- **Adaptability:** The solution must adapt to evolving fraud patterns using continuous learning.
- **Compliance:** The system must follow data privacy and security standards (e.g., GDPR, PCI DSS).
- **User Experience:** It should be non-intrusive for legitimate users, preserving a smooth transaction experience.

### **Activity 3: Literature Survey**

A literature review for this project involves analyzing existing fraud detection methods using machine learning. Research covers:

- Classical models such as Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines.
- Ensemble models and boosting techniques like XGBoost and LightGBM.
- Deep learning-based fraud detection systems (LSTM, Autoencoders).
- Research papers on real-time fraud detection systems and their success/failure cases.
- Evaluation metrics like precision, recall specifically for imbalanced datasets.
- Public datasets used for benchmarking (e.g., the dataset from Kaggle used in this project).

This survey helps identify the most suitable techniques for handling highly imbalanced data and achieving effective fraud detection.

### **Activity 4: Social or Business Impact.**

**Social Impact:**

- Enhances trust and safety for customers in online financial ecosystems.
- Reduces mental stress and inconvenience associated with financial fraud.

**Business Impact:**

- Helps businesses prevent financial losses due to fraudulent transactions.
- Improves brand reputation and customer retention by ensuring secure transactions.
- Reduces operational cost and effort required for manual fraud investigation.

## **Milestone 2: Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible.

### **Activity 1: Collect the dataset**

There are many popular open sources for collecting data. Eg: kaggle.com, UCI repository, etc. In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

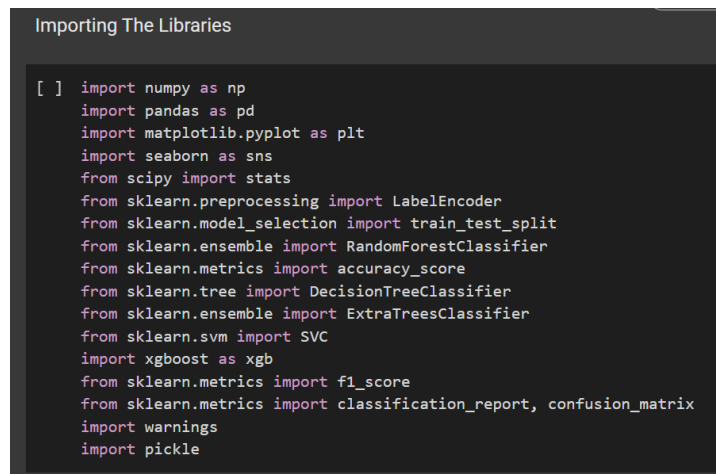
Link: <https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it.

### **Activity 1.1: Importing the libraries**

Importing the necessary libraries as shown in the image.



```
[ ] 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 1.2: Read the Dataset**

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read\_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
[ ] data=os.path.join(path,"PS_20174392719_1491204439457_log.csv")
    df=pd.read_csv(data)
```

```
[ ] df.head()
```

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

## Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness, so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning.

### Activity 2.1: Handling missing values

For checking the null values, `df.isna().any()` function is used. To sum those null values we use `.sum()` function. From the image below we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
[15] df.isna().any()
```

	0
step	False
type	False
amount	False
nameOrig	False
oldbalanceOrg	False
newbalanceOrig	False
nameDest	False
oldbalanceDest	False
newbalanceDest	False
isFraud	False
dtype:	bool

## Activity 2.2: Handling Outliers

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of amount feature with some mathematical formula.

Boxplot from seaborn library is used here.



To understand the central tendency of the amount column.

```
print(stats.mode(df['amount']))
print(np.mean(df['amount']))
print(np.median(df['amount']))
```

ModeResult(mode=np.float64(10000000.0), count=np.int64(3207))  
179861.90354913071  
74871.94

- To compute IQR-based upper and lower bounds for detecting outliers.
- To define a reusable function transformationPlot() to visualize feature distribution and normality.



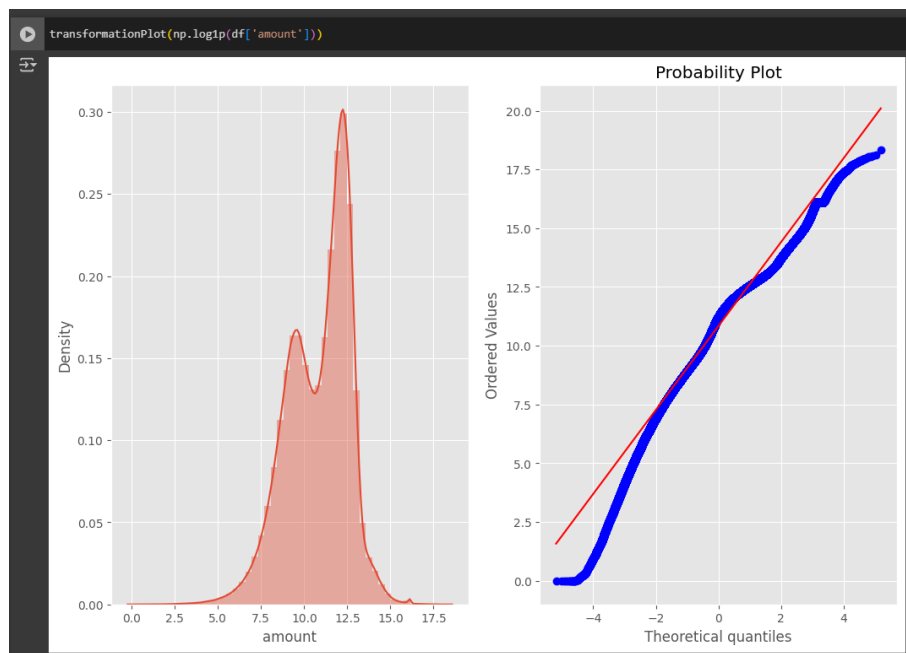
```
[ ] q1 = np.quantile(df['amount'],0.25)
    q3 = np.quantile(df['amount'],0.75)
    IQR = q3-q1
    upper_bound = q3+(1.5*IQR)
    lower_bound = q1-(1.5*IQR)

    print('q1 : ',q1)
    print('q2 : ',q3)
    print('IQR : ',IQR)
    print('Upper Bound : ',upper_bound)
    print('Lower Bound : ',lower_bound)
    print('Skewed data : ',len(df[df['amount']>upper_bound]))
    print('Skewed data : ',len(df[df['amount']<lower_bound]))

    def transformationPlot(feature):
        plt.figure(figsize=(12,8))
        plt.subplot(1,2,1)
        sns.distplot(feature)
        plt.subplot(1,2,2)
        stats.probplot(feature,dist='norm',plot=plt)
        plt.show()
```

```
q1 : 13389.57
q2 : 208721.4775
IQR : 195331.9075
Upper Bound : 501719.33875
Lower Bound : -279608.29125
Skewed data : 338078
Skewed data : 0
```

To check the distribution and normality of the amount column after applying a log transformation using `np.log1p()`



## Milestone 3: Exploratory Data Analysis

### Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

Descriptive Analysis

```
df.describe(include='all')
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
count	6.362620e+06	6362620	6.362620e+06	6362620	6.362620e+06	6.362620e+06	6362620	6.362620e+06	6.362620e+06	6362620
unique	NaN	5	NaN	6353307	NaN	NaN	2722362	NaN	NaN	2
top	NaN	CASH_OUT	NaN	C1530544995	NaN	NaN	C1286084959	NaN	NaN	is not Fraud
freq	NaN	2237500	NaN	3	NaN	NaN	113	NaN	NaN	6354407
mean	2.433972e+02	NaN	1.798619e+05	NaN	8.338831e+05	8.551137e+05	NaN	1.100702e+06	1.224996e+06	NaN
std	1.423320e+02	NaN	6.038582e+05	NaN	2.888243e+06	2.924049e+06	NaN	3.399180e+06	3.674129e+06	NaN
min	1.000000e+00	NaN	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
25%	1.560000e+02	NaN	1.338957e+04	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
50%	2.390000e+02	NaN	7.487194e+04	NaN	1.420800e+04	0.000000e+00	NaN	1.327057e+05	2.146614e+05	NaN
75%	3.350000e+02	NaN	2.087215e+05	NaN	1.073152e+05	1.442584e+05	NaN	9.430367e+05	1.111909e+06	NaN
max	7.430000e+02	NaN	9.244552e+07	NaN	5.958504e+07	4.958504e+07	NaN	3.560159e+08	3.561793e+08	NaN

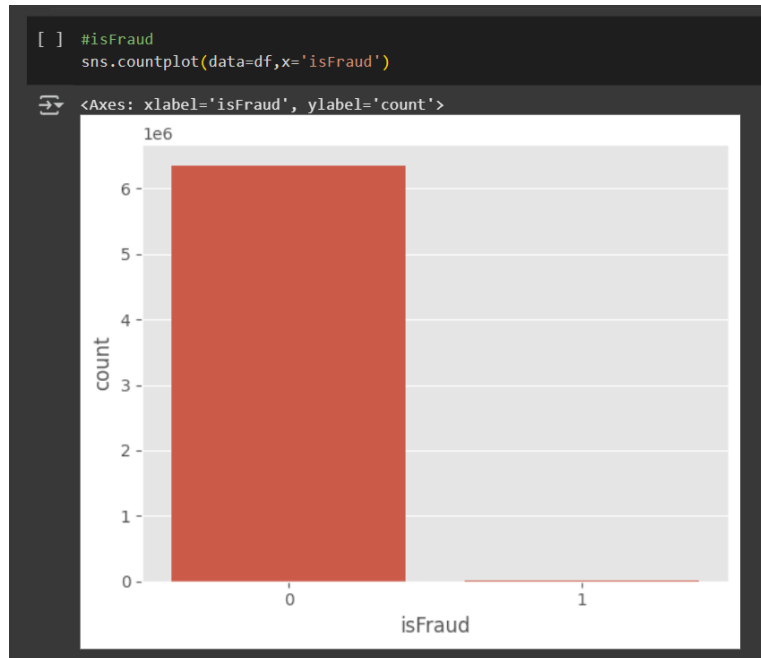
### Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

#### Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs, such as Pie chart and count plot.

Seaborn package provides a wonderful function count plot. It is more useful for categorical features. With the help of count plot, we can Number of unique values in the feature.



This countplot shows the frequency of different transaction types in the dataset. CASH\_OUT and PAYMENT are the most common, while DEBIT is the least frequent.



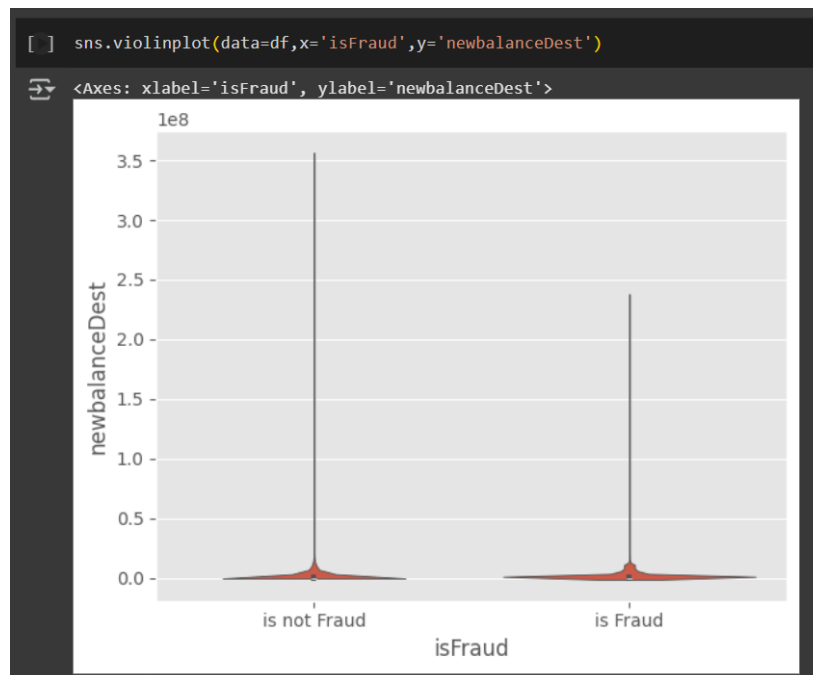
## Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis.

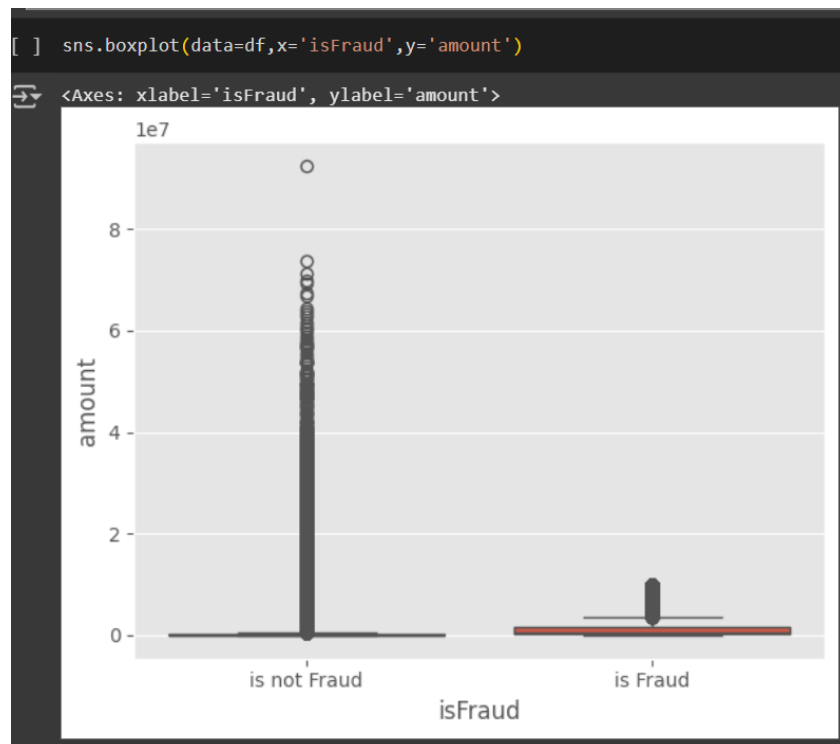
This boxplot shows that fraudulent transactions (isFraud = 1) tend to occur later (higher step values) compared to non-fraudulent ones. Also, fraud has a wider spread of step values, indicating variability in timing.



This violin plot shows that newbalanceDest is mostly near zero for both fraudulent and non-fraudulent transactions, but some very large values occur—more extreme in the non-fraud category.

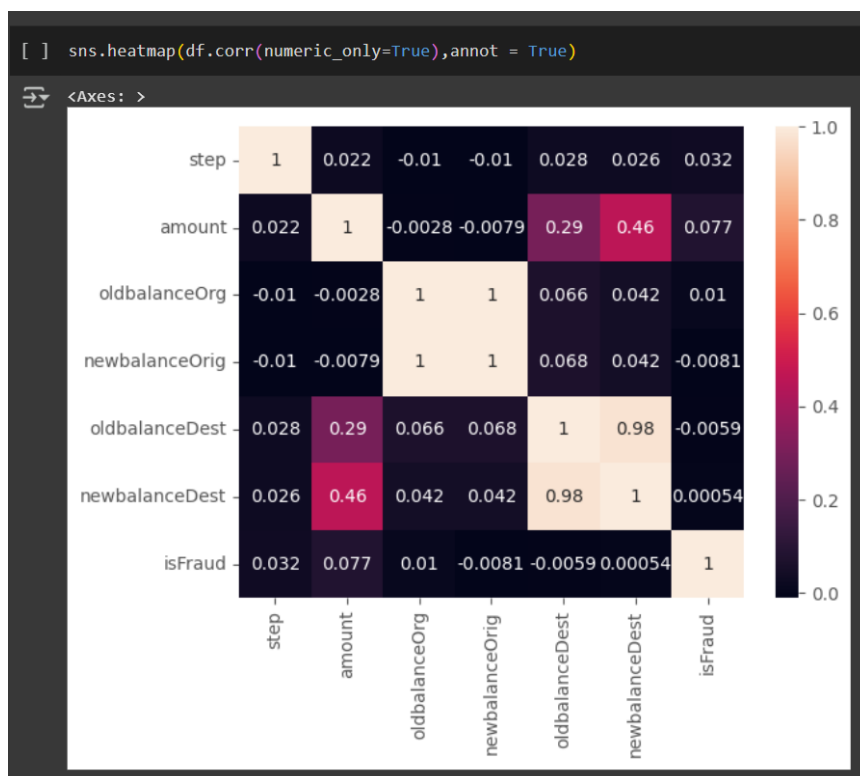


This boxplot shows that fraudulent transactions generally involve lower amounts, while non-fraudulent ones have a wider range and many high-value outliers.



### Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.



## Encoding the Categorical Features

- The categorical Features can't be passed directly to the Machine Learning Model. So, we convert them into Numerical data based on their order. This Technique is called Encoding.
- Here we are importing Label Encoder from the Sklearn Library.
- Here we are applying fit\_transform to transform the categorical features to numerical features.

```
Object Data LabelEncoding

la = LabelEncoder()
df['type'] = la.fit_transform(df['type'])
df['type'].value_counts()
```

	count
1	2237500
3	2151495
0	1399284
4	532909
2	41432

dtype: int64

## Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set.

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train\_test\_split() function from sklearn. As parameters, we are passing x, y, test\_size, random\_state.

```
Splitting Data Into Train And Test

[ ] 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)
```

```
(5090096, 7)
(1272524, 7)
(1272524,)
(5090096,)
```

## Milestone 4: Model Building

### Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

#### Activity 1.1: Random forest model

First Random Forest Model is imported from sklearn Library then RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X\_train and X\_test.

```
Random Forest Classifier

[ ] rfc = RandomForestClassifier()
    rfc.fit(x_train,y_train)
    y_test_predict1 = rfc.predict(x_test)
    test_accuracy = accuracy_score(y_test,y_test_predict1)
    test_accuracy

0.9997092392756443

[ ] y_train_predict1 = rfc.predict(x_train)
    train_accuracy = accuracy_score(y_train,y_train_predict1)
    train_accuracy

1.0
```

#### Activity 1.2: Decision tree model

First Decision Tree is imported from sklearn Library then DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X\_train and X\_test.

```
Decision Tree Classifier

[ ] dtc = DecisionTreeClassifier()
    dtc.fit(x_train,y_train)

    y_test_predict2 = dtc.predict(x_test)
    test_accuracy = accuracy_score(y_test,y_test_predict2)
    test_accuracy

0.999704524236871

[ ] y_train_predict2 = dtc.predict(x_train)
    train_accuracy = accuracy_score(y_train,y_train_predict2)
    train_accuracy

1.0
```

### Activity 1.3: ExtraTrees model

First, the ExtraTreesClassifier is imported from the sklearn.ensemble library. The algorithm is then initialized and the training data is passed to the model using the .fit() function. Predictions on the test data are made using the .predict() function and stored in a new variable. The training and testing accuracy can be evaluated using X\_train and X\_test respectively.

```
ExtraTrees Classifier

[ ] etc = ExtraTreesClassifier()
    etc.fit(x_train,y_train)

    y_test_predict3 = etc.predict(x_test)
    test_accuracy = accuracy_score(y_test,y_test_predict3)
    test_accuracy

0.9997029525572798

[ ] y_train_predict3 = etc.predict(x_train)
    train_accuracy = accuracy_score(y_train,y_train_predict3)
    train_accuracy

1.0
```

### Activity 1.4: SVM model

SVM Model is imported from sklearn Library then SVM algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
Support Vector Machine Classifier

[ ] from sklearn.svm import LinearSVC
    svc = LinearSVC()
    svc.fit(x_train,y_train)
    y_test_predict4 = svc.predict(x_test)
    test_accuracy = accuracy_score(y_test,y_test_predict4)
    test_accuracy

0.9975466081582745

[ ] y_train_predict4 = svc.predict(x_train)
    train_accuracy = accuracy_score(y_train,y_train_predict4)
    train_accuracy

0.9975587886751055
```



## Activity 1.5: Xgboost model

The XGBClassifier is imported from the xgboost library. The model is initialized and trained on the training dataset using the .fit() method. Predictions for the test set are made using the .predict() function and saved in a separate variable. The accuracy for both training and test data is then calculated using X\_train and X\_test.

```
Xgboost Classifier

[ ] xgb1 = xgb.XGBClassifier()
    xgb1.fit(x_train,y_train1)

    y_test_predict5 = xgb1.predict(x_test)
    test_accuracy = accuracy_score(y_test1,y_test_predict5)
    test_accuracy

0.9997776073378577

[ ] y_train_predict5 = xgb1.predict(x_train)
    train_accuracy = accuracy_score(y_train1,y_train_predict5)
    train_accuracy

0.999851279818691
```

## Milestone 5: Performance Testing & Hyperparameter Tuning

### Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks include accuracy, precision, recall, support and F1-score.

### Activity 1.1: Compare the model

For comparing the above four models, the compareModel() function is defined

```
Compare The Models

[ ] def compareModel():
    print("Train accuracy for rfc",accuracy_score(y_train,y_train_predict1))
    print("Test accuracy for rfc",accuracy_score(y_test,y_test_predict1))
    print("Train accuracy for dtc",accuracy_score(y_train,y_train_predict2))
    print("Test accuracy for dtc",accuracy_score(y_test,y_test_predict2))
    print("Train accuracy for etc",accuracy_score(y_train,y_train_predict3))
    print("Test accuracy for etc",accuracy_score(y_test,y_test_predict3))
    print("Train accuracy for svc",accuracy_score(y_train,y_train_predict4))
    print("Test accuracy for svc",accuracy_score(y_test,y_test_predict4))
    print("Train accuracy for xgb1",accuracy_score(y_train1,y_train_predict5))
    print("Test accuracy for xgb1",accuracy_score(y_test1,y_test_predict5))

    compareModel()

Train accuracy for rfc 1.0
Test accuracy for rfc 0.9997092392756443
Train accuracy for dtc 1.0
Test accuracy for dtc 0.999704524236871
Train accuracy for etc 1.0
Test accuracy for etc 0.9997029525572798
Train accuracy for svc 0.9975587886751055
Test accuracy for svc 0.9975466081582745
Train accuracy for xgb1 0.999851279818691
Test accuracy for xgb1 0.9997776073378577
```

## Activity 2: Classification report

After making predictions on the test dataset using the `.predict()` function, the `classification_report` from `sklearn.metrics` is used to evaluate the model's performance. This report includes metrics such as precision, recall, f1-score, and support for each class. It gives a detailed view of how well the model is performing, especially useful for imbalanced datasets.

### Activity 2.1: Random forest model

This output shows the performance of a **Random Forest Classifier** for fraud detection.

The model achieves **high precision (0.98)** for detecting fraud but a slightly lower **recall (0.79)**, meaning it misses some fraud cases (342 false negatives).

Despite class imbalance, the overall accuracy is **nearly perfect (1.00)**, but care should be taken as the dataset is dominated by non-fraud cases.

```
[ ] pd.crosstab(y_test,y_test_predict1)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1299	342
is not Fraud	28	1270855

```
[ ] print(classification_report(y_test,y_test_predict1))
```

	precision	recall	f1-score	support
is Fraud	0.98	0.79	0.88	1641
is not Fraud	1.00	1.00	1.00	1270883
accuracy			1.00	1272524
macro avg	0.99	0.90	0.94	1272524
weighted avg	1.00	1.00	1.00	1272524

### Activity 2.2: Decision tree model

This output shows the performance of a **Decision Tree Classifier** for fraud detection.

It achieves slightly **lower precision (0.89)** than Random Forest but **higher recall (0.88)**, detecting more actual fraud cases (fewer false negatives).

While the overall accuracy is still nearly perfect due to class imbalance, Decision Tree trades a bit of false positive rate for better fraud recall.

```
[ ] pd.crosstab(y_test,y_test_predict2)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1439	202
is not Fraud	174	1270709

```
[ ] print(classification_report(y_test,y_test_predict2))
```

	precision	recall	f1-score	support
is Fraud	0.89	0.88	0.88	1641
is not Fraud	1.00	1.00	1.00	1270883
accuracy			1.00	1272524
macro avg	0.95	0.94	0.94	1272524
weighted avg	1.00	1.00	1.00	1272524

## Activity 2.3: ExtraTrees model

This output shows the performance of an **ExtraTrees Classifier** for fraud detection.

It yields the **highest precision (0.99)** among the models but with slightly lower **recall (0.78)**, missing more fraud cases compared to the Decision Tree.

Despite excellent overall accuracy, this model favors reducing false positives at the cost of more false negatives in fraud detection.

```
[ ] pd.crosstab(y_test,y_test_predict3)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1278	363
is not Fraud	15	1270868

```
[ ] print(classification_report(y_test,y_test_predict3))
```

	precision	recall	f1-score	support
is Fraud	0.99	0.78	0.87	1641
is not Fraud	1.00	1.00	1.00	1270883
accuracy			1.00	1272524
macro avg	0.99	0.89	0.94	1272524
weighted avg	1.00	1.00	1.00	1272524

## Activity 2.4: SVM model

This output shows the performance of an **SVM model**, which performs **poorly** on fraud detection.

It achieves **only 22% precision and 35% recall** for fraud, misclassifying most fraud cases as non-fraud due to **extreme class imbalance**.

The label encoding result confirms that the dataset is highly imbalanced (mostly label 1 = "not fraud"), which causes SVM to bias toward the majority class.

```
[ ] pd.crosstab(y_test,y_test_predict4)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	570	1071
is not Fraud	2051	1268832

```
[ ] print(classification_report(y_test,y_test_predict4))
```

	precision	recall	f1-score	support
is Fraud	0.22	0.35	0.27	1641
is not Fraud	1.00	1.00	1.00	1270883
accuracy			1.00	1272524
macro avg	0.61	0.67	0.63	1272524
weighted avg	1.00	1.00	1.00	1272524

```
[ ] la = LabelEncoder()
    y_train1 = la.fit_transform(y_train)
```

```
[ ] y_test1 = la.transform(y_test)
    y_test1
```

```
⇒ array([1, 1, 1, ..., 1, 1, 1])
```

```
[ ] y_train1
```

```
⇒ array([1, 1, 1, ..., 1, 1, 1])
```

## Activity 2.5: Xgboost model

This output shows the performance of the **XGBoost model**, which performs very well on fraud detection. It achieves a strong **precision of 0.96** and **recall of 0.87** for the fraud class, resulting in a high **F1-score of 0.91**.

Among all models, XGBoost provides a good balance between detecting fraud and minimizing false positives, making it highly suitable for imbalanced classification.

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

```
⇒
```

col_0	0	1
row_0		
0	1422	219
1	64	1270819

```
[ ] print(classification_report(y_test1,y_test_predict5))
```

```
⇒
```

	precision	recall	f1-score	support
0	0.96	0.87	0.91	1641
1	1.00	1.00	1.00	1270883
accuracy			1.00	1272524
macro avg	0.98	0.93	0.95	1272524
weighted avg	1.00	1.00	1.00	1272524

## **Milestone 6: Model Deployment**

### **Activity 1: Save the best model**

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance. This can be useful in avoiding the need to retrain the model every time it is needed and to be able to use it in the future.

```
[ ] pickle.dump(svc,open('payments.pkl','wb'))
```

### **Activity 2: Integrate with Web Framework**

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he must enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

#### **Activity 2.1: Building Html Page**

For this project create HTML file namely

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

and save them in the templates folder.

#### **Activity 2.2: Build Python code**

Import the libraries

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current as an argument.

**Render HTML page:**

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

When the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

**Retrieves the value from UI:**

Here we are routing our app to predict() function.

This function retrieves all the values from the HTML page using Post request.

That is stored in an array.

This array is passed to the model.predict() function.

This function returns the prediction.

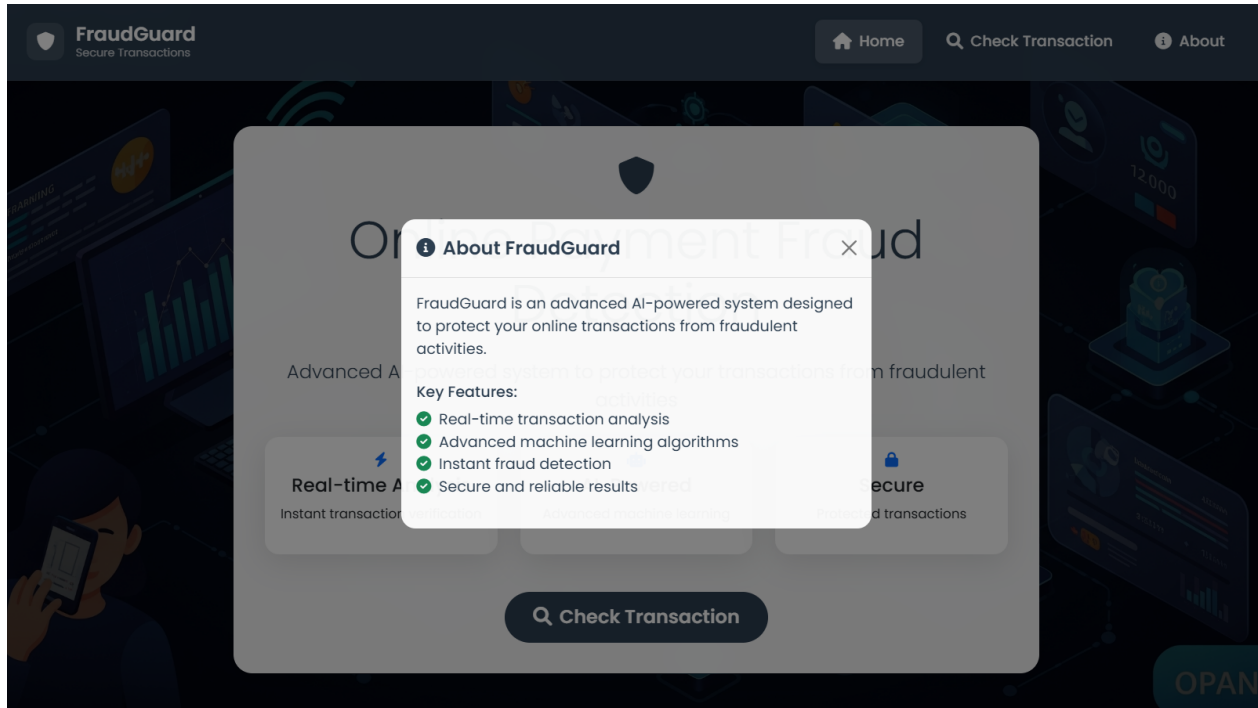
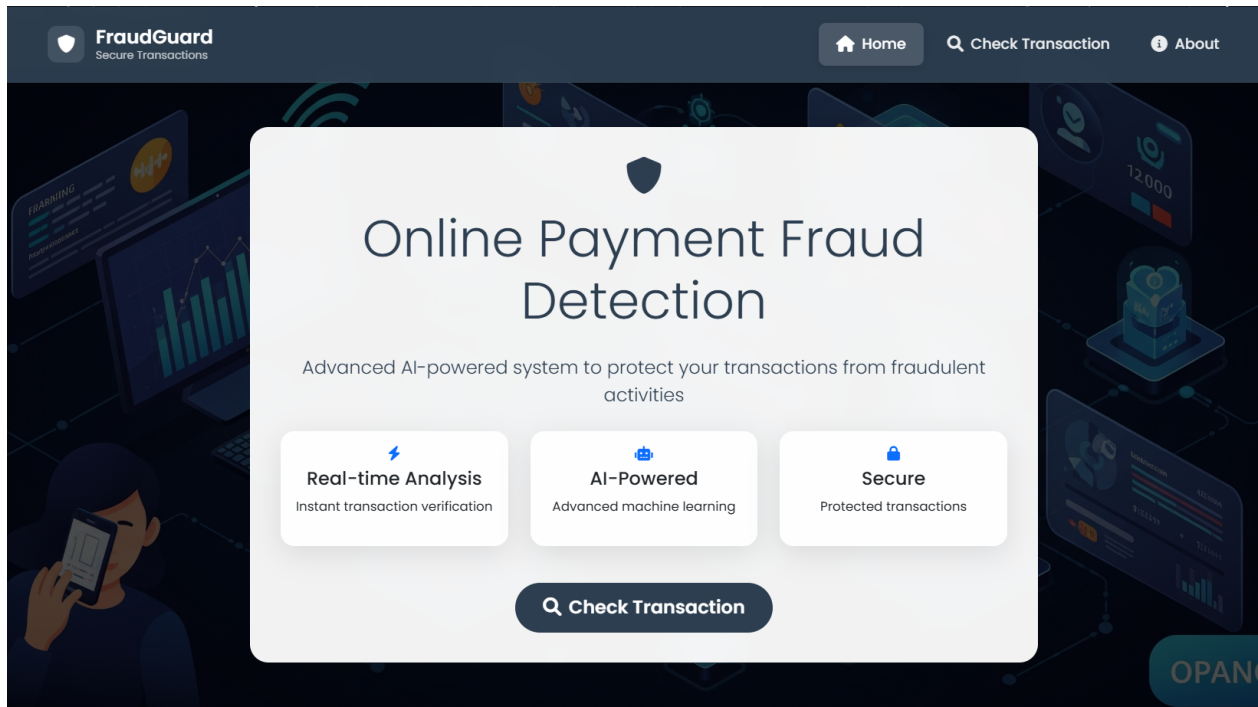
And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.


**Activity 2.3: Run the web application**

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.


Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result

## Output Screenshots



 **FraudGuard**  
Secure Transactions

[Home](#) [Check Transaction](#) [About](#)



## Transaction Analysis

Enter the transaction details below for fraud detection

🕒 Step

1

🔗 Transaction Type

PAYMENT

\$ Amount

9839

📄 Old Balance Origin

170136

📄 New Balance Origin


160296


📄 Old Balance Destination

0


📄 New Balance Destination

0


 **Analyze Transaction**

 **FraudGuard**  
Secure Transactions


[Home](#) [Check Transaction](#) [About](#)




## Transaction Verified


 **Transaction is Legitimate**

This transaction appears to be legitimate and safe to proceed.


 **Check Another Transaction**

 **Back to Home**



 **FraudGuard**  
Secure Transactions

[Home](#) [Check Transaction](#) [About](#)



## Transaction Analysis

Enter the transaction details below for fraud detection

🕒 Step

1

🔄 Transaction Type

TRANSFER

\$ Amount

181

📄 Old Balance Origin

181

📄 New Balance Origin


0


📄 Old Balance Destination

0


📄 New Balance Destination

0


 **Analyze Transaction**

 **FraudGuard**  
Secure Transactions


[Home](#) [Check Transaction](#) [About](#)




## Fraud Alert!

 **Fraudulent Transaction Detected (Confidence: 99.94%)**

This transaction has been flagged as potentially fraudulent. Please review the details carefully.

 **Check Another Transaction**

 **Back to Home**

OPANOM