# Online Payments Fraud Detection using Machine Learning

## 1. Project Description:

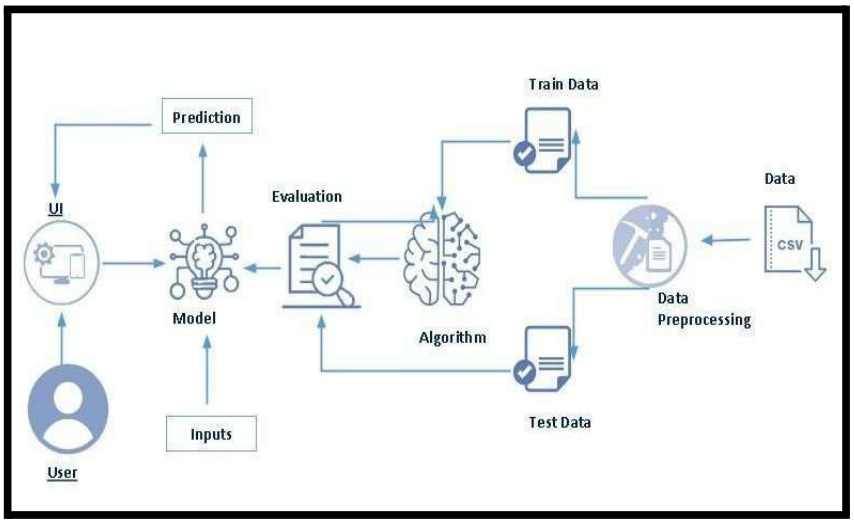
Online Payments Fraud Detection using Machine Learning is a proactive approach to identify and prevent fraudulent activities during online transactions. By leveraging historical transaction data, customer behavior patterns, and machine learning algorithms, this project aims to detect potential fraud 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:**




## 2. Pre requisites:

**To complete this project, you must required following software's, concepts and packages**

- **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.

## 3. Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques on outlier and some visualization concepts.

## 4. Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed 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,

- Data collection
  - Collect the dataset or create the dataset
- Visualizing and analyzing data
  - Univariate analysis
  - Bivariate analysis
  - Multivariate analysis
  - Descriptive analysis
- Data pre-processing
  - Checking for null values
  - Handling outlier
  - Handling categorical data
  - Splitting data into train and test
- Model building
  - Import the model building libraries
  - Initializing the model
  - Training and testing the model
  - Evaluating performance of model
  - Save the model
- Application Building
  - Create an HTML file
  - Build python code
  - Run the application

## 4.1. Data Collection

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

**Download the dataset**

In this project we have used PS_20174392719_1491204439457_logs.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.
Link: link

## 4.2. Visualizing and analyzing data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.
Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

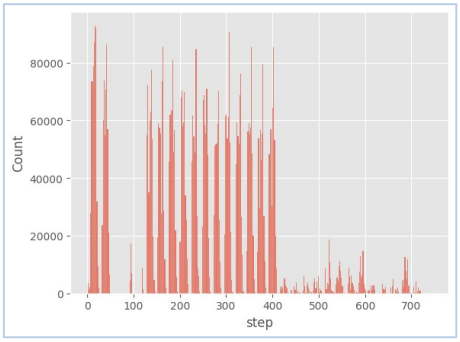
**Importing the libraries**

```
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
```

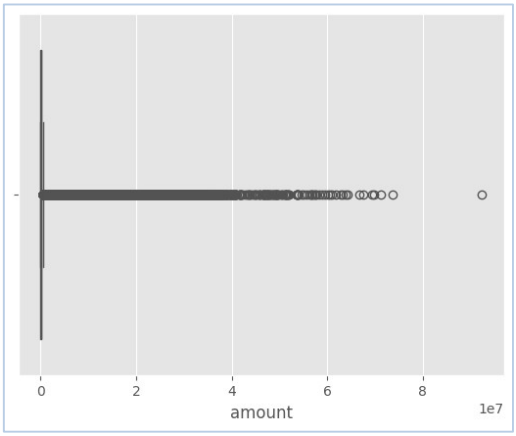
**Univariate analysis**

In simple words, univariate analysis is understanding the data with a single feature.

Histogram: The distribution of one or more variables is represented by a histogram, a traditional visualisation tool, by counting the number of observations that fall within.



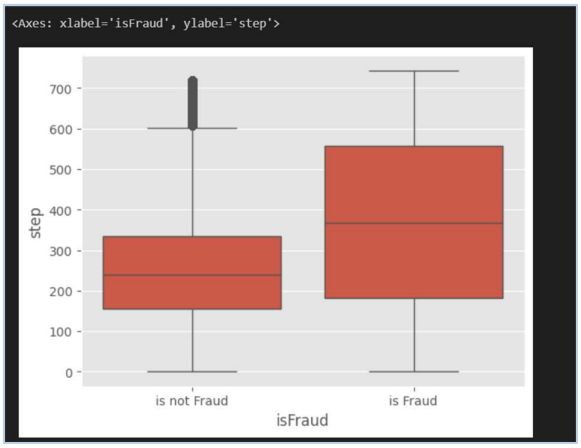
Boxplot: The relationship between the amount attribute and the boxplot is visualised.




**Bivariate analysis**

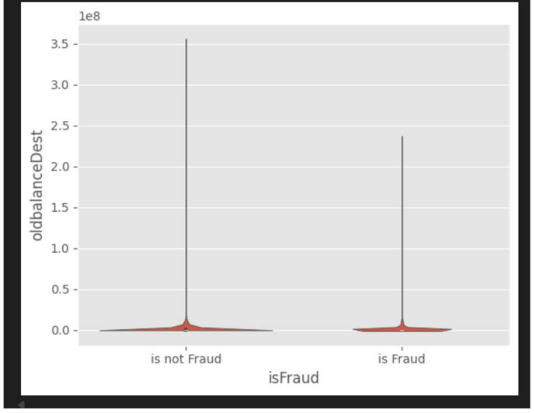
To find the relation between two features we use bivariate analysis. Here we are visualising the relationship between newbalanceDest and isFraud.
jointplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

Here we are visualising the relationship between isFraud and amount.boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.




Here we are visualising the relationship between isFraud and newbalanceDest. violinplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

<Axes: xlabel='isFraud', ylabel='oldbalanceDest'>




**Descriptive analysis**

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.

```
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 | C1677795071 | 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 |

## 4.3. Data Pre-processing

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 Object data label encoding
- Splitting dataset into training and test set

**Checking for null values**

Isnull is used (). sum() to check your database for null values. Using the df.info()

function, the data type can be determined.

**Handling outliers**

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

```
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('q3: ',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]))
```

```
q1:  13389.57
q3:  208721.4775
IQR:  195331.9075
upper_bound:  501719.33875
lower_bound:  -279608.29125
Skewed data:  338078
Skewed data:  0
```

**Object label encoding**

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

```
type
1    2237500
3    2151495
0    1399284
4     532909
2      41432
Name: count, dtype: int64
```

**Splitting the dataset**

Now let's split the Dataset into train and test setsChanges: 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 my target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
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(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

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

## 4.4. Model Building

We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

**Random Forest classifier**
A function named RandomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier 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. For evaluating the model, a confusion matrix and classification report is done.

**Decision tree Classifier**
A function named Decisiontree is created and train and test data are passed as the parameters. Inside the function, the DecisiontreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

**ExtraTrees Classifier**
A function named ExtraTree is created and train and test data are passed as the parameters. Inside the function, ExtraTreeClassifier 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. For evaluating the model, a confusion matrix and classification report is done.

**SupportVectorMachine Classifier**
A function named SupportVector is created and train and test data are passed as the parameters. Inside the function, the SupportVectorClassifier 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. For evaluating the model, confusion matrix and classification report is done

**xgboost Classifier**
A function named xgboost is created and train and test data are passed as the parameters. Inside the function, the xgboostClassifier 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. For evaluating the model, confusion matrix and classification report is done

**Compare the models**
For comparing the above four models, the compareModel function is defined.
After calling the function, the results of models are displayed as output.

```
[136]
def compareModel():
    print("train accuracy for rfc:", accuracy_score(y_train_predict,y_train))
    print("test accuracy for rfc:", accuracy_score(y_test_predict,y_test))
    print("train accuracy for dtc:", accuracy_score(y_train_predict2,y_train))
    print("test accuracy for dtc:", accuracy_score(y_test_predict2,y_test))
    print("train accuracy for etc:", accuracy_score(y_train_predict3,y_train))
    print("test accuracy for etc:", accuracy_score(y_test_predict3,y_test))
    print("train accuracy for svc:", accuracy_score(y_train_predict4,y_train))
    print("test accuracy for svc:", accuracy_score(y_test_predict4,y_test))
    print("train accuracy for xgb1:", accuracy_score(y_train_predict5,y_train1))
    print("test accuracy for xgb1:", accuracy_score(y_test_predict5,y_test1))

[138]
compareModel()

...
train accuracy for rfc: 1.0
test accuracy for rfc: 0.9996047225828354
train accuracy for dtc: 1.0
test accuracy for dtc: 0.999607080102222
train accuracy for etc: 1.0
test accuracy for etc: 0.9996330128154753
train accuracy for svc: 0.9592966419493856
test accuracy for svc: 0.9596054769890391
train accuracy for xgb1: 0.9998049152707532
test accuracy for xgb1: 0.9997178835133954
```

We have selected the **Support vector classifier model as the best model** and save the model with svc as the learning algorithm.

```
[146]
import pickle
pickle.dump(svc, open('payments.pkl','wb'))
```

## 5. Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to 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

**Building Html Pages:**

For this project create two HTML files namely

- index.html
- result.html

and save them in the templates folder.

And app.py for the backend file

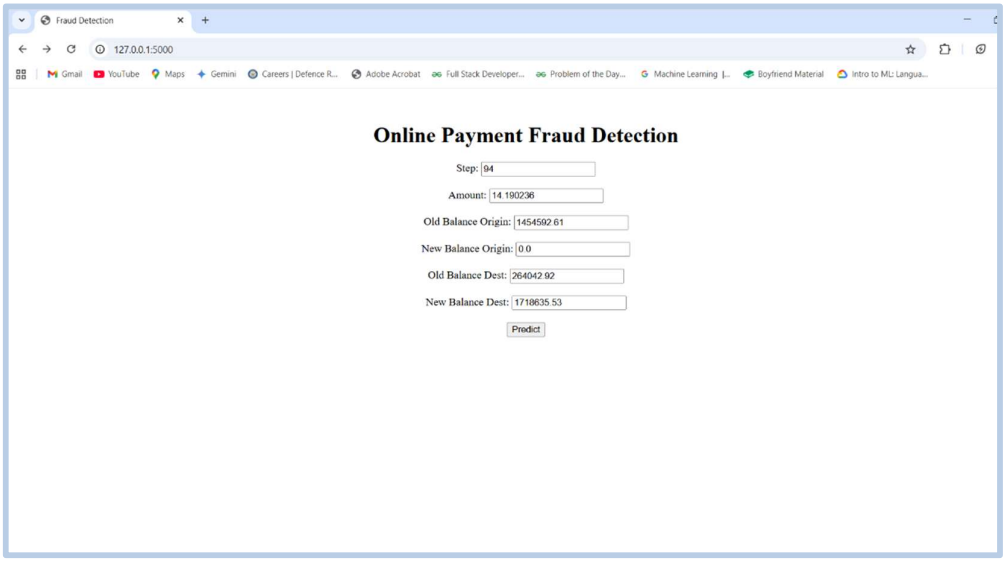
**Run the application**

- 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.

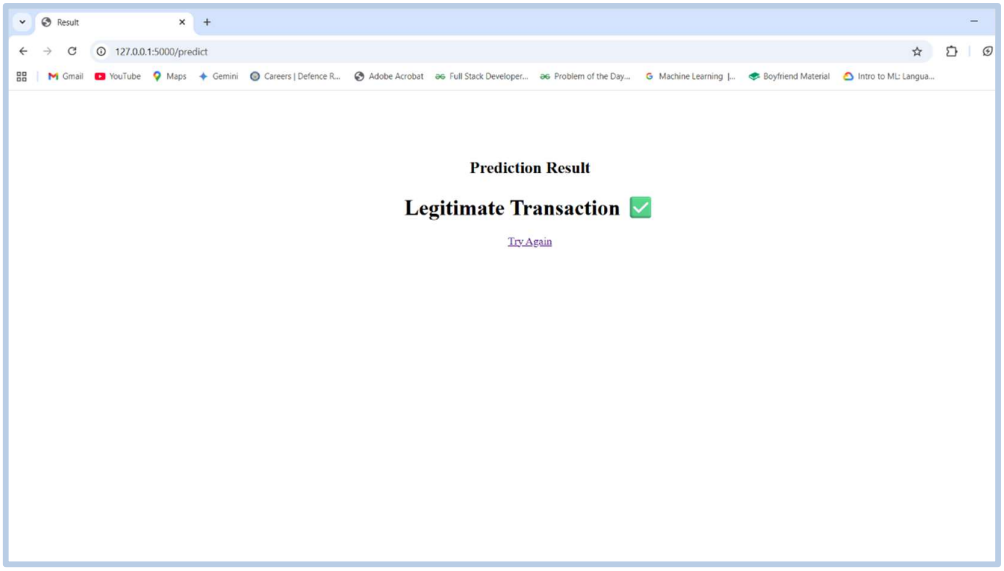
```
PS C:\Users\Hp\OneDrive\Desktop\smart_project> python app.py
C:\Users\Hp\AppData\Roaming\Python\Python314\site-packages\sklearn\base.py:463: InconsistentVersionWarning: Trying to unpickle esti
mator SVC from version 1.7.2 when using version 1.8.0. This might lead to breaking code or invalid results. Use at your own risk. F
or more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

The resultant website will be hosted at localhost: http://127.0.0.1:5000

Index.html:




After clicking on predict button, the result will be shown as follows:




The parameter's values evaluated to be a legitimate transaction i.e., not fraud.