



ONLINE PAYMENTS FOR FRAUD DETECTION

USING MACHINE LEARNING

### Project Hand-out, Faculty Development Program – NaanMudhalvan

SmartInternz

[www.smartinternz.com](http://www.smartinternz.com/)

**Online Payments For 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**.**

The system continuously monitors online payment transactions in real time. By analyzing transaction features

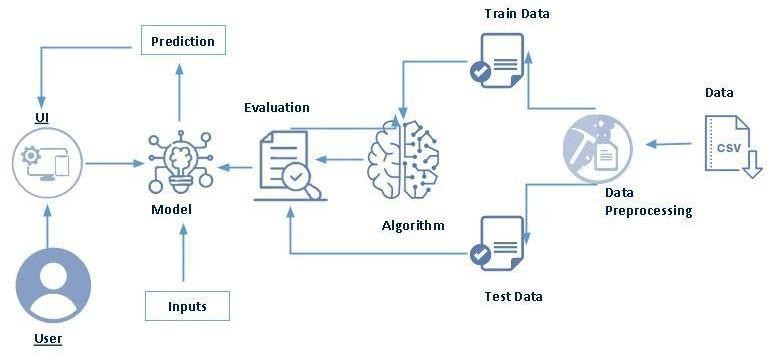
such as transaction amount, location, device information, and user behavior, it can flag suspicious transactions

for further investigation, preventing fraudulent activities before they occur.

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.

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

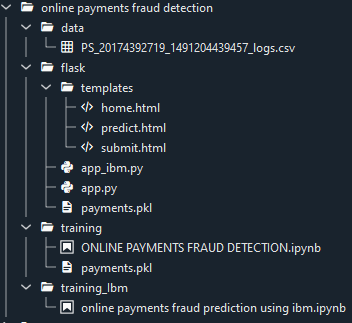
# Prior Knowledge:

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

* ML Concepts
  + Supervised learning
  + Unsupervised learning: <https://www.javatpoint.com/unsupervised-machine-learning>
* Decision tree: [https://www.javatpoint.com/machine-learning-decision-tree-classification-](https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm) [algorithm](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-](https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/) [understand-the-math-behind-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-](https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/) [evaluation-error-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 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.
* Dtc\_model.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 th model.

# Milestone 1: Define Problem / Problem Understanding

In today’s digital age, paying online is faster than ever—but so is getting scammed. With just a few clicks, money can vanish before you even realize something’s wrong. Fraudsters don’t take breaks, and their tactics evolve constantly, making it hard for traditional systems to keep up. The real challenge? Spotting fraud as it happens, not after the damage is done. This project aims to build a machine learning-based solution that acts like a digital watchdog—quiet, fast, and always alert—analyzing transactions in real time to flag anything that doesn’t look quite right.

## Activity 1: Specify the business problem

Online payment fraud poses a serious threat to businesses, especially in sectors like banking and e-commerce. Even a single fraudulent transaction can lead to significant financial loss, damage customer trust, and harm a company’s reputation. The core challenge is preventing fraud before it occurs. Traditional methods like manual review are too slow and inefficient to keep up with high transaction volumes. What’s needed is a smart, real-time system that can accurately analyze transactions and detect anomalies. When implemented well, such a system can reduce losses, lower risk, and provide a seamless and secure experience—often without the customer even noticing.

## Activity 2: Business requirements

For a fraud detection system to be truly effective in a business setting, it needs to meet several important requirements. First, accuracy is critical—it must detect fraudulent activity without flagging genuine transactions, as false alarms can frustrate customers. Next, the system has to be fast. In the world of online payments, delays are unacceptable, so real-time response is essential. Scalability is another key factor, since businesses handle huge volumes of transactions daily. The system should also be easy to use, with a clear interface for entering and viewing data. Finally, the model should be transparent—it must provide clear reasons for its alerts so businesses can understand and trust the results.

## Activity 3: Literature Survey (Student Will Write)

Researchers have widely applied machine learning to fraud detection, using models like Decision Trees, Random Forest, SVM, and XGBoost. These models are known for their reliability and ability to catch complex fraud patterns. A major challenge is the imbalance between fraudulent and normal transactions, which is often addressed using SMOTE. Most studies rely on real-world datasets, like the Kaggle credit card fraud dataset, to test their models effectively.

## Activity 4: Social or Business Impact.

A good fraud detection system doesn’t just help businesses—it helps people too. For business owners, it means less money lost to fraud, fewer legal issues, and a stronger, more trusted brand. For customers, it creates a safer experience when making payments online, giving them peace of mind. As online scams keep growing, this kind of technology plays a big role in building trust. It also takes the pressure off teams who would otherwise have to manually check for fraud. By catching fraud early, we’re not only protecting money—we’re helping make the internet a safer place for everyone.

# Milestone 2: Data Collection & Preparation

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.

## Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

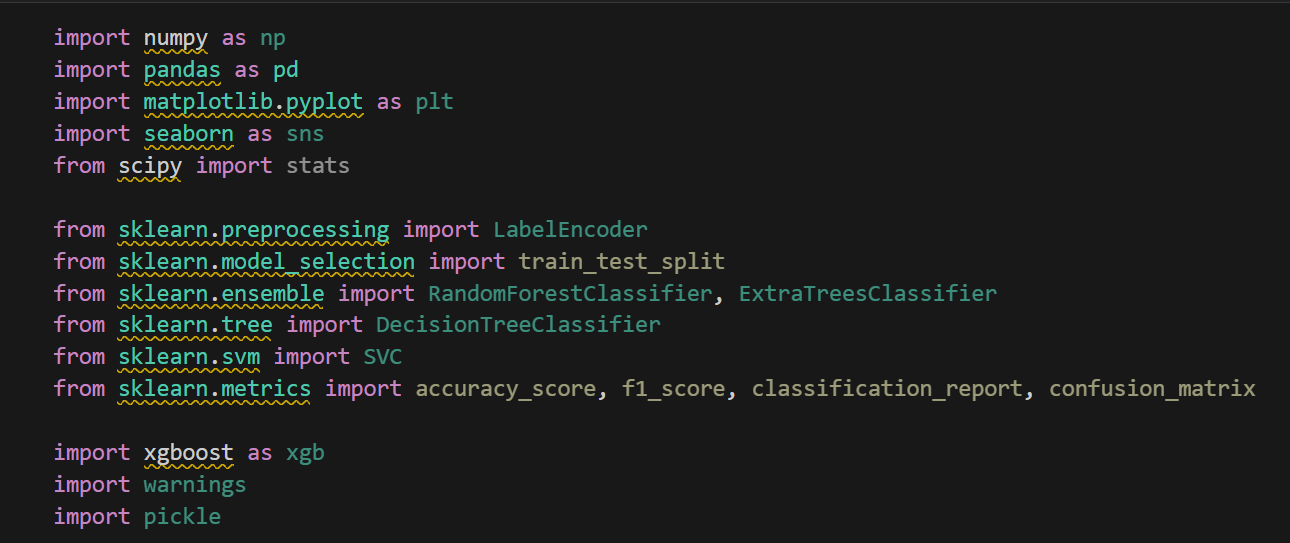
In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: <https://www.kaggle.com/datasets/buntyshah/auto-insurance-claims-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.

## Activity 1.1: Importing the libraries

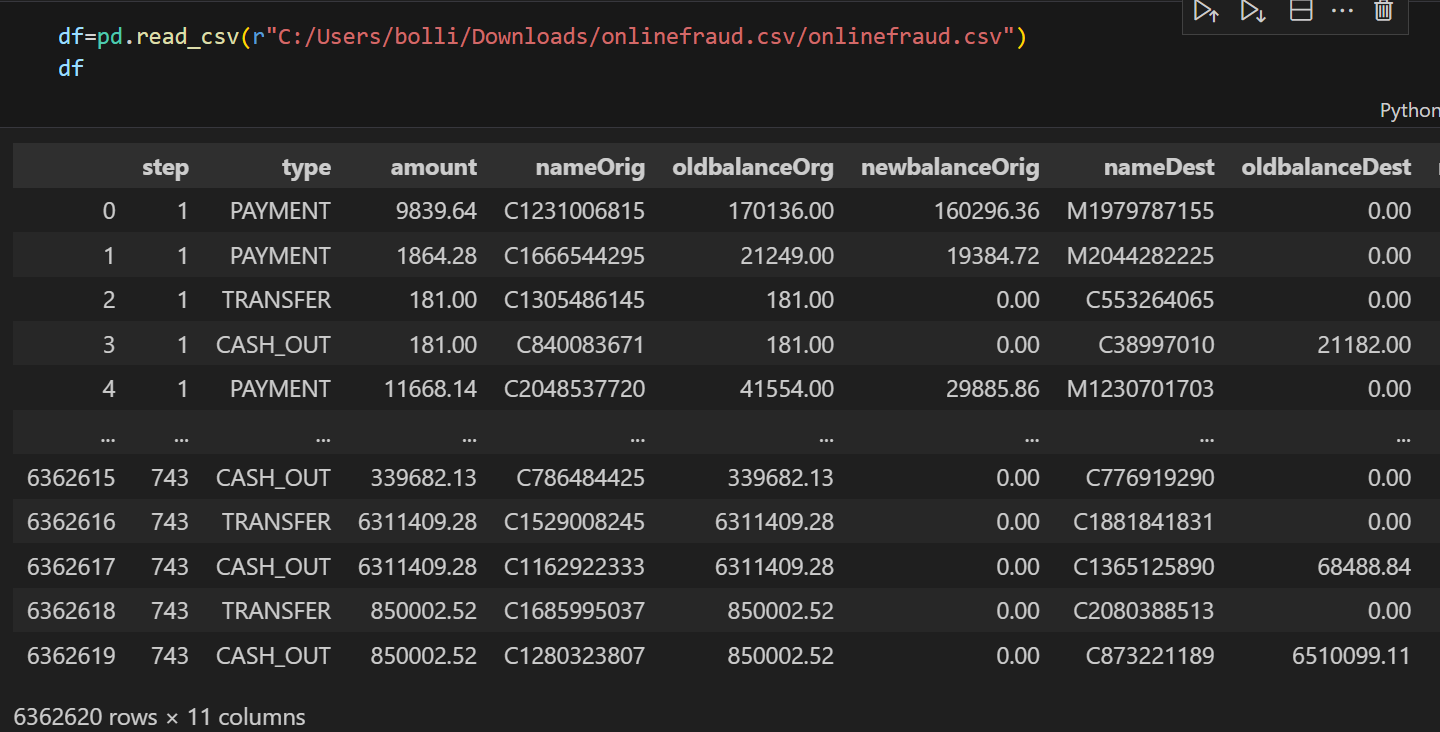
Import the necessary libraries

## 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.

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



## Activity 2: Data Preparation

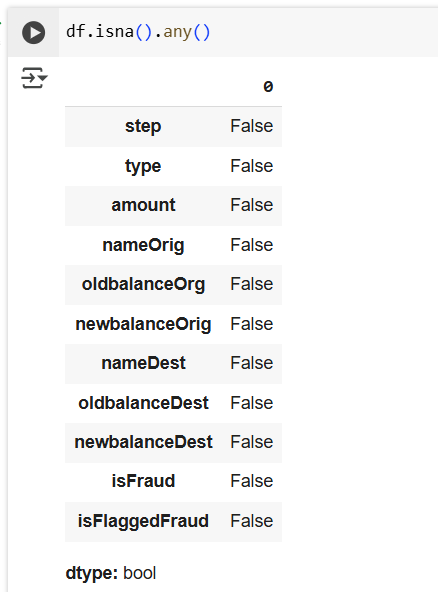
As we have understood how the data is, let's pre-process the collected data.

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. Depending on the condition of your dataset, you may or may not have to go through all these steps.

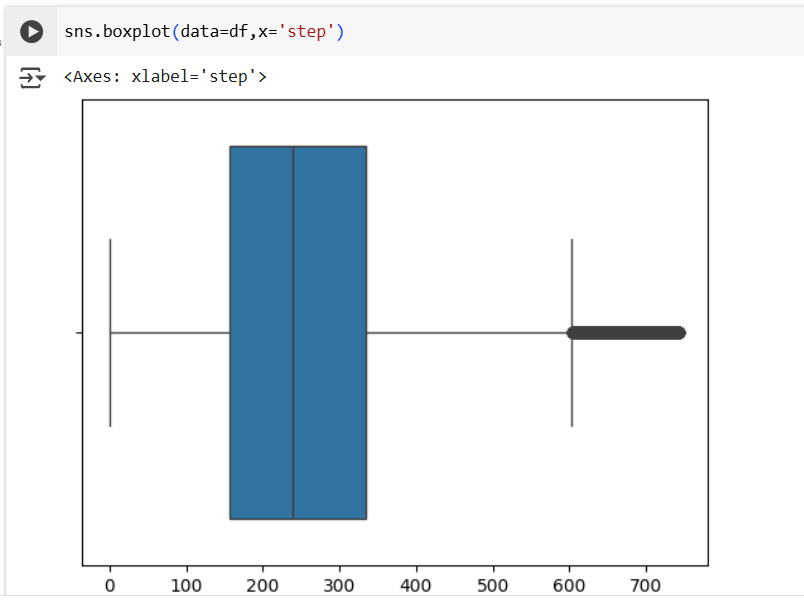
## Activity 2.1: Handling missing values

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

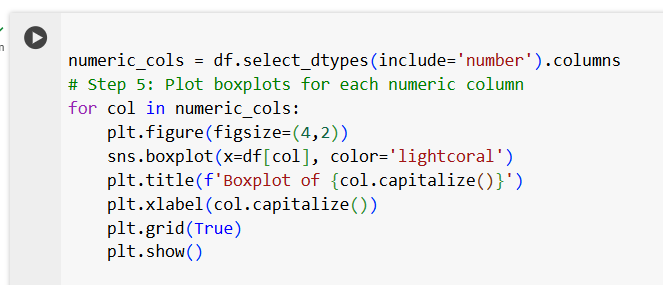
## 

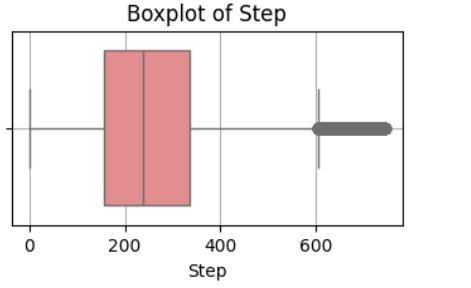
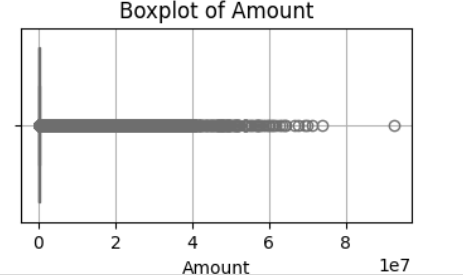
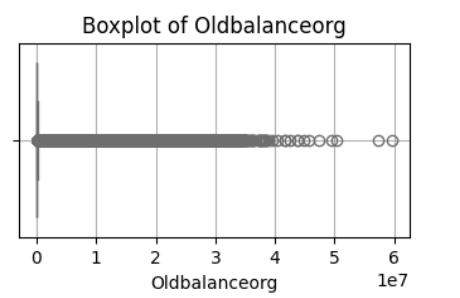
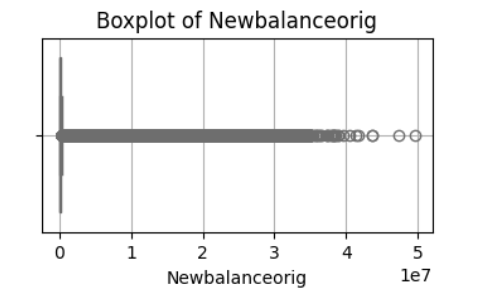
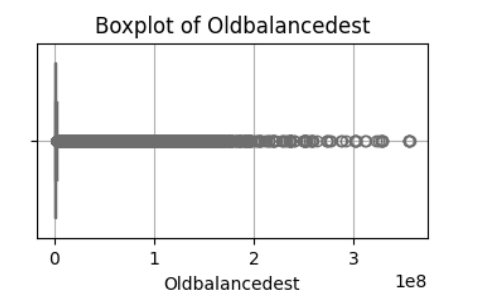
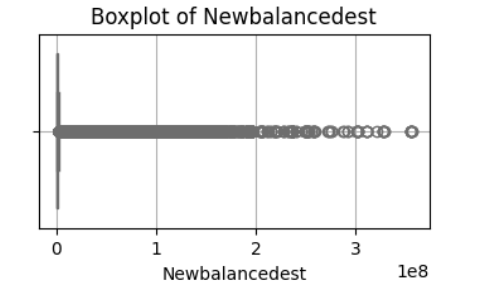
## Activity 2.2: Handling Outliers

A boxplot (using Seaborn) was used to detect outliers in the policy\_annual\_premium feature. It showed clear outliers, which were identified using the IQR method with the formulas:  
**Lower Bound = Q1 - 1.5 × IQR**  
**Upper Bound = Q3 + 1.5 × IQR**  
These values helped in handling extreme data points before model training.



* To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.

To handle the outliers transformation technique is used. Here log transformation is used. We have created a function to visualize the distribution and probability plot of policy\_annual\_premium feature.

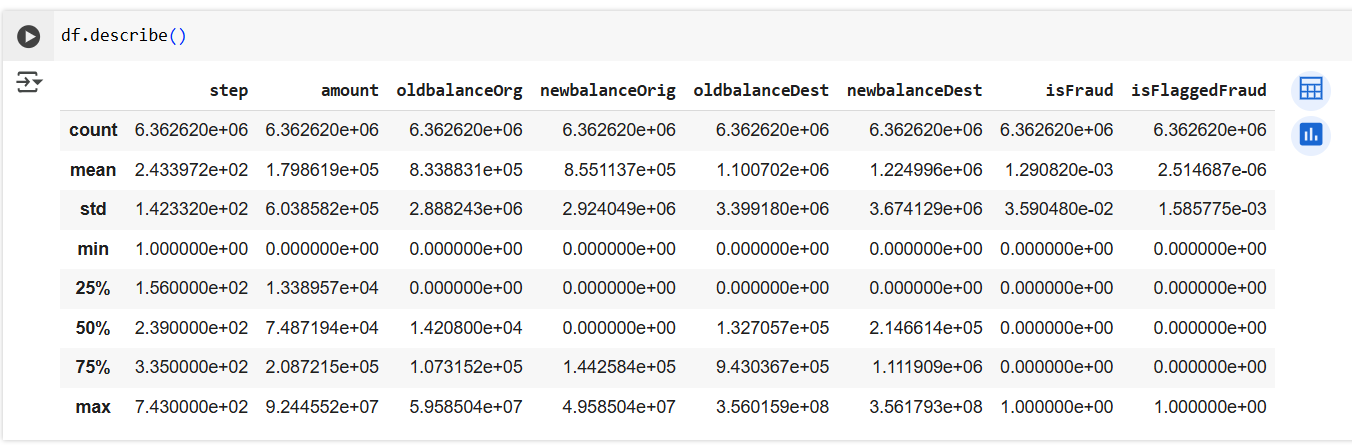


# 

# Milestone 3: Exploratory Data Analysis

## Activity 1: Descriptive statistical

## We used the describe() function from pandas to get a quick summary of the data. For numerical features, it showed stats like mean, min, max, and percentiles. For categorical features, it displayed the most frequent values and unique counts. This helped us understand the overall distribution of the dataset.



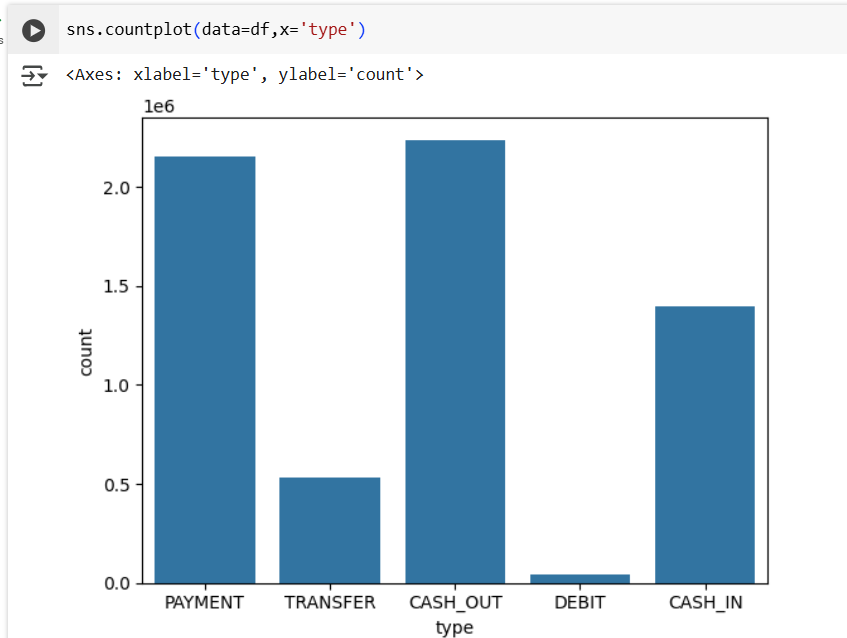
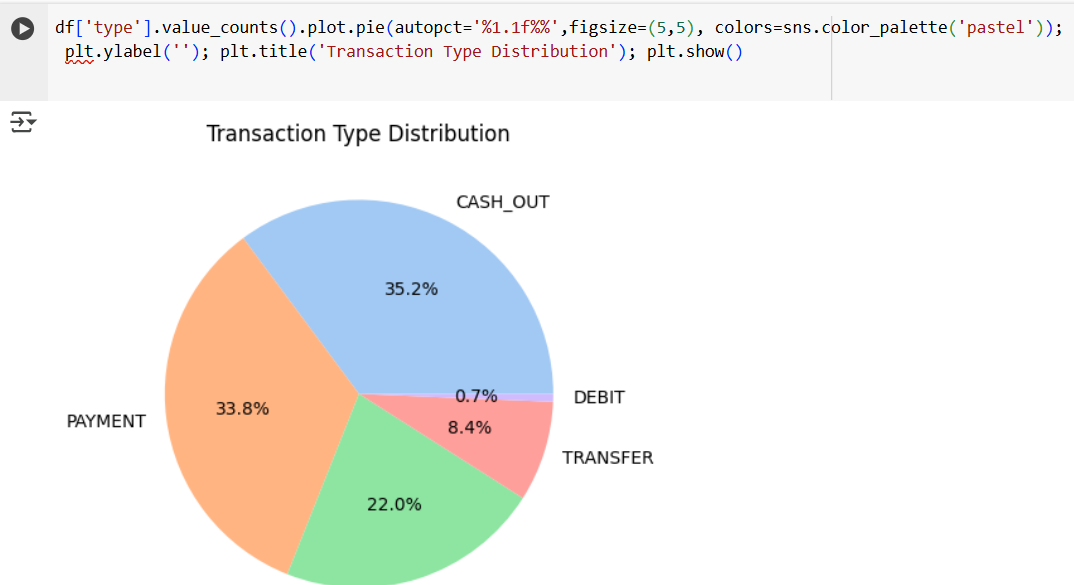
## Activity 2: Visual analysis

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.

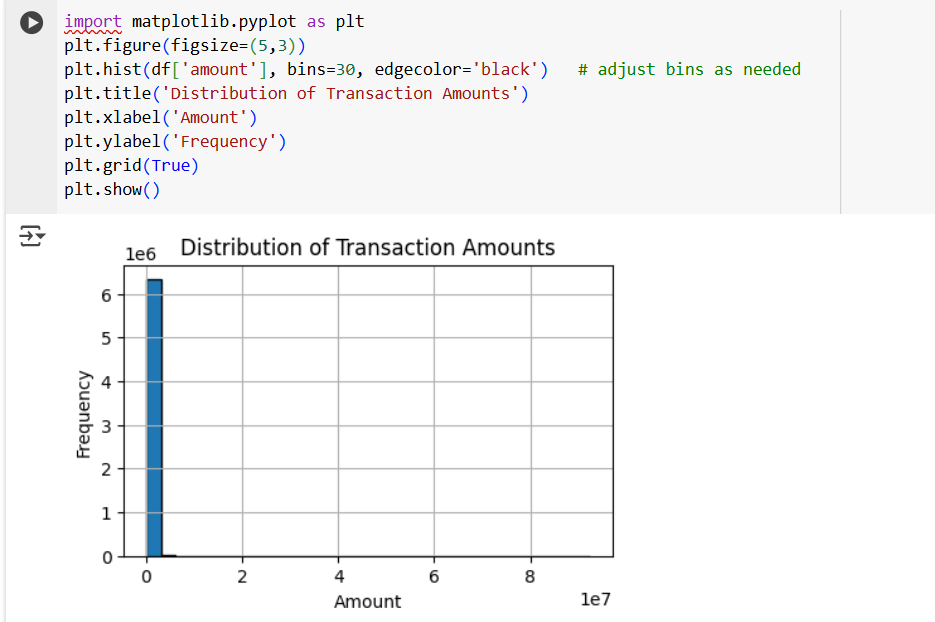
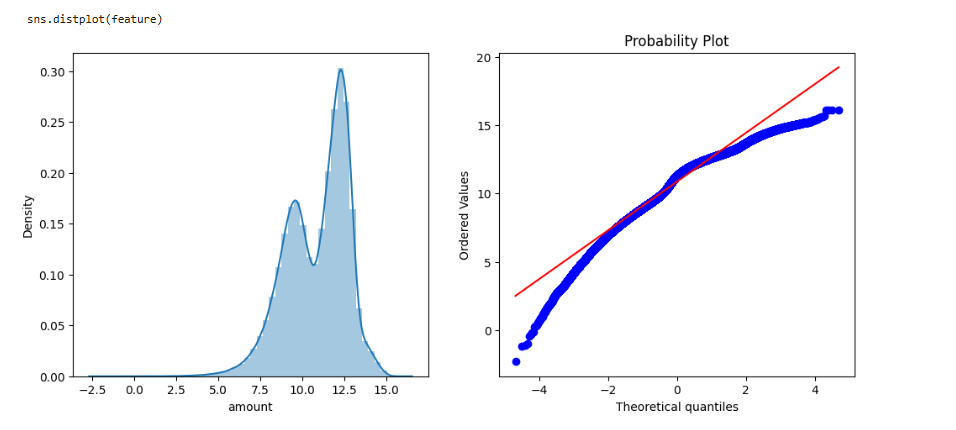
## Activity 2.1: Univariate analysis

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

Seaborn package provides a wonderful function countplot. It is more useful for categorical features. With the help of countplot, we can Number of unique values in the feature. From the countplot we can say that there are only 247 fraud cases reported in 1000 insurance claims.



The pie chart shows the distribution of different transaction types in the dataset. The majority of transactions are **CASH\_OUT (35.2%)** and **PAYMENT (33.8%)**, followed by **CASH\_IN (22.0%)**. **TRANSFER** transactions make up **8.4%**, while **DEBIT** transactions are the least common at **0.7%**. This visualization helps us understand which transaction types are most frequently used, which is important for identifying patterns in fraud detection



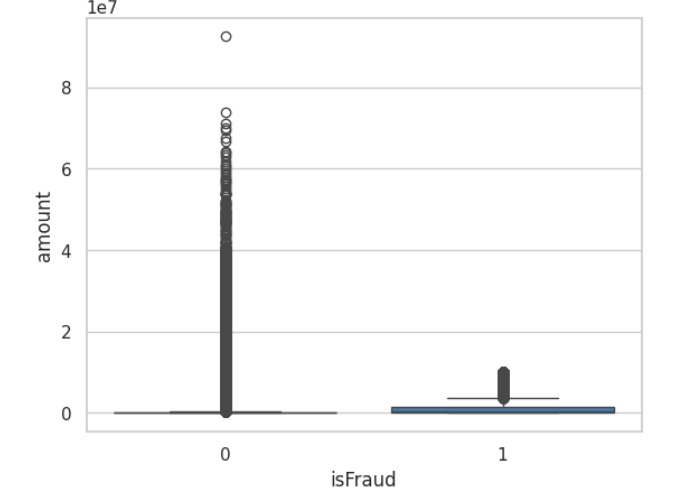
Transformation plot:

.

## Activity 2.2: Bivariate analysis

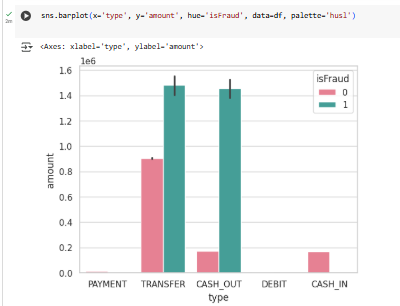
Box plot

* As a 1st parameter we are passing isFraud and as a 2nd parameter we are passing amount.
* This plot shows how transaction amounts are spread for fraudulent vs non-fraudulent transactions.
* Fraudulent transactions tend to involve higher amounts.



Bar plot

* As a 1st parameter we are passing type and as a 2nd parameter we are passing amount.
* From the below plot, you can understand the distribution of transaction amount for each type.
* The highest average amount is observed for TRANSFER and CASH\_OUT types.

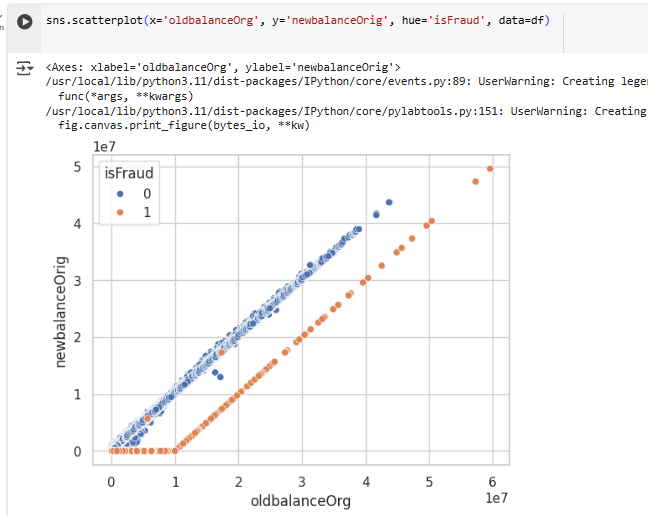


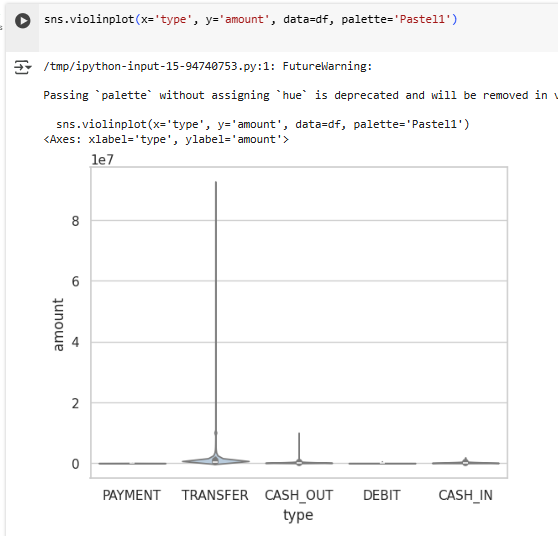
Violin plot:

* As a 1st parameter we are passing type and as a 2nd parameter we are passing amount.
* This plot shows both the distribution shape and range of transaction amounts for each type.
* TRANSFER and CASH\_OUT have wider spreads and higher peaks.

Scatter plot:

* As a 1st parameter we are passing oldbalanceOrg and as a 2nd parameter newbalanceOrig, with hue as isFraud.
* This plot shows how original balances change before and after transactions.
* Fraudulent transactions are more visible as outliers.



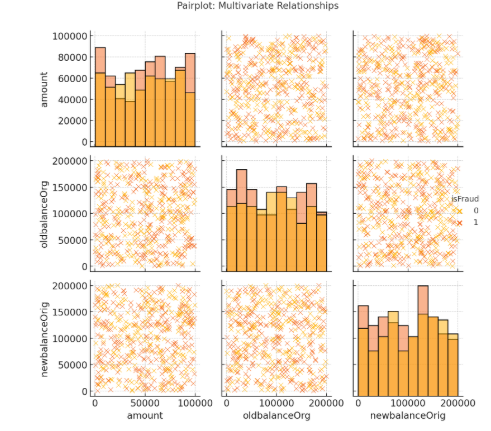


## Activity 2.3: Multivariate analysis

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

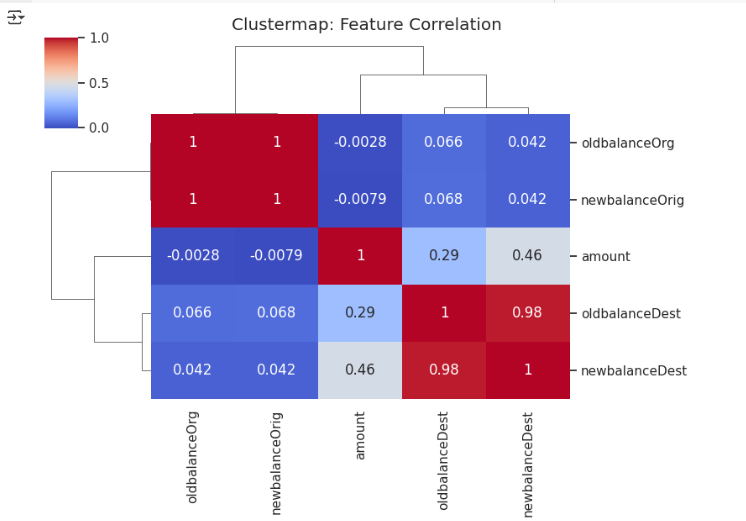
Pair plot:

* The pairplot shows relationships between amount, balances, and fraud.
* Fraudulent transactions mostly have higher amounts and balances.
* oldbalanceOrg and newbalanceOrig are positively correlated.
* Useful to spot fraud patterns across multiple features.



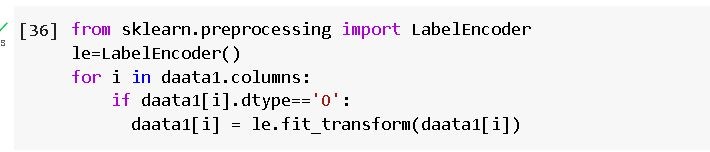
Cluster map:

* The clustermap shows correlations among numeric features like amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, and newbalanceDest.
* Features like oldbalanceOrg and newbalanceOrig are highly correlated, indicating similar value trends.
* oldbalanceDest and newbalanceDest also show moderate to high correlation.
* Clustering groups the most similar features together, which helps in detecting redundant variables.
* Highly correlated features can be removed or combined to avoid multicollinearity in machine learning models.



**Encoding the Categorical Features:**

* The categorical Features are 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.



**Splitting data into train and test**

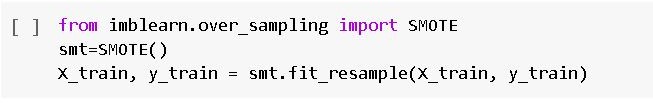
Now let’s split the Dataset into train and test sets. First split the dataset into x and y and then split the data set

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.



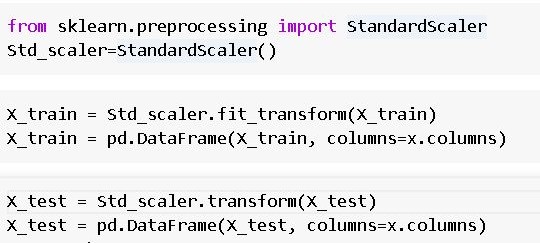
#### Handling Imbalanced dataset

* + Imbalanced data is a common problem in machine learning and data analysis, where the number of observations in one class is significantly higher or lower than the other class. Handling imbalanced data is important to ensure that the model is not biased towards the majority class and can accurately predict the minority class.
  + Here we are using SMOTE Technique.



#### Scaling

* + - Scaling is a technique used to transform the values of a dataset to a similar scale to improve the performance of machine learning algorithms. Scaling is important because many machine learning algorithms are sensitive to the scale of the input features.
    - Here we are using Standard Scaler.
    - This scales the data to have a mean of 0 and a standard deviation of 1. The formula is given by: X\_scaled = (X - X\_mean) / X\_std

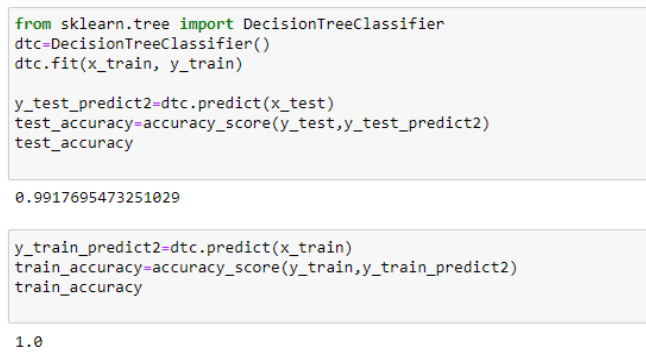


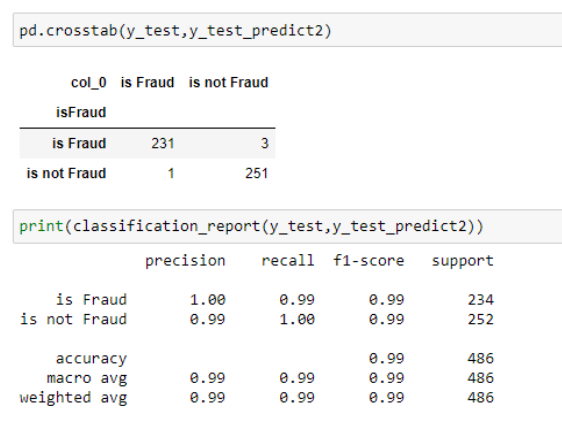
# Milestone 4: Model Building

## Activity 1: Training the model in multiple algorithms

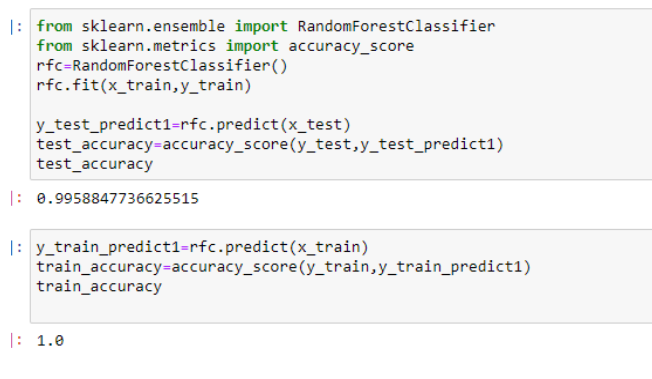
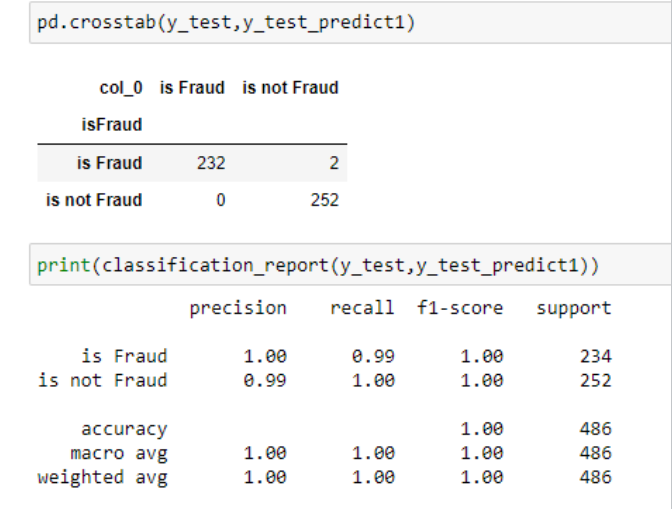
Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

## Activity 1.1: 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.



## Activity 1.2: 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.

## Activity 1.3: KNN model

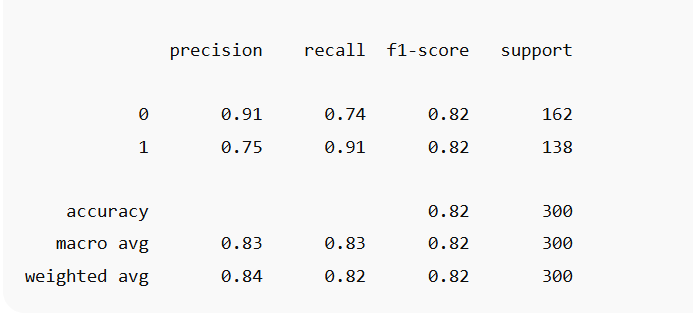
KNN Model is imported from sklearn Library then KNeighborsClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report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



[[120 42]

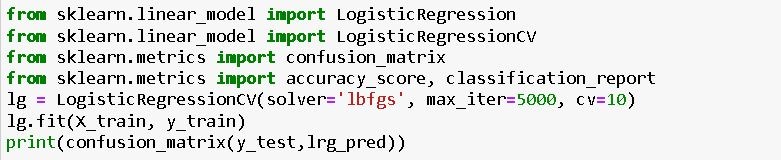
[ 12 126]]

Accuracy: 0.82



## Activity 1.4: Logistic Regression model

Logistic Regression Model is imported from sklearn Library then Logistic Regression algorithm is initialised 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 is done.



Confusion matrix:

[[120 42]

[ 12 126]]

Accuracy: 0.82

Classification report:

precision recall f1-score support

0 0.91 0.74 0.82 162

1 0.75 0.91 0.82 138

accuracy 0.82 300

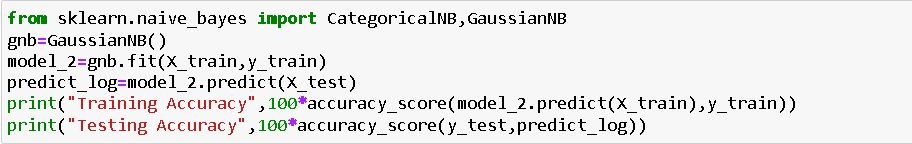
macro avg 0.83 0.83 0.82 300

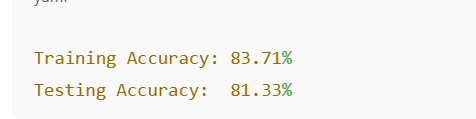
weighted avg 0.84 0.82 0.82 300

## Activity 1.5: Naïve Bayes model

Naïve Bayes Model is imported from sklearn Library then Naïve Bayes algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with

.predict() function and saved in new variable. We can find the Train and Test accuracy by X\_train and X\_test.

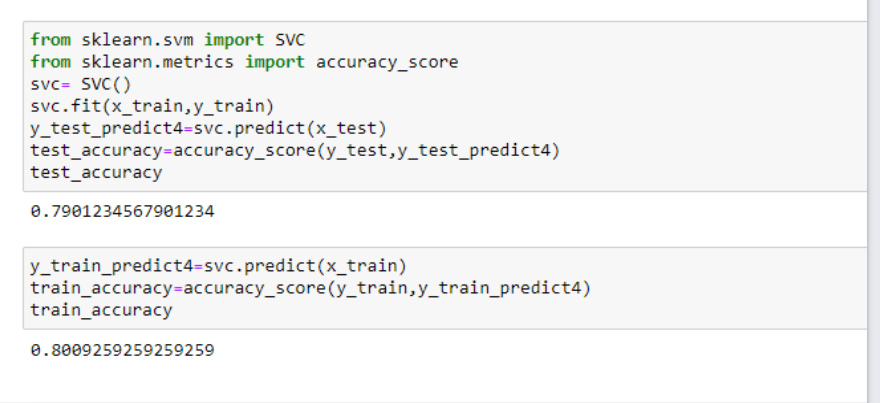
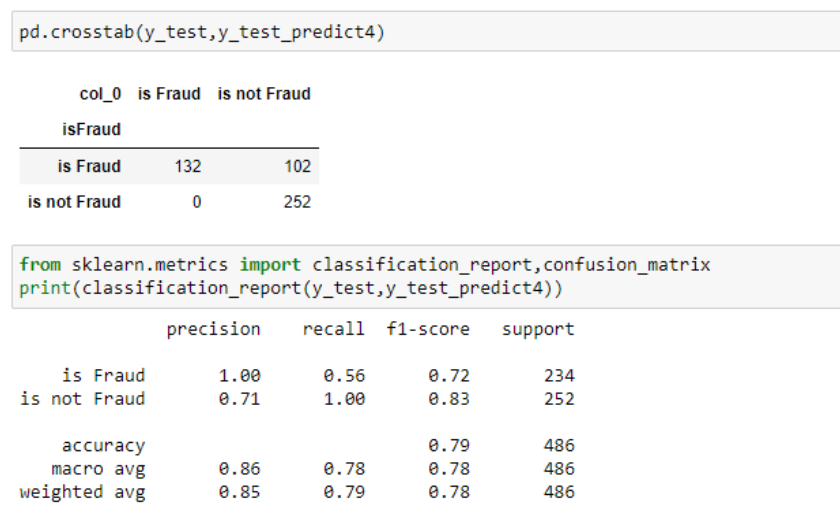




## Activity 1.6: SVM model

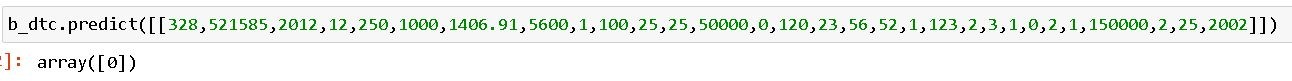
SVM Model is imported from sklearn Library then SVM algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with

.predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.



## Activity 2: Testing the model

Here we have tested with Decision Tree algorithm. You can test with all algorithm. With the help of predict() function.



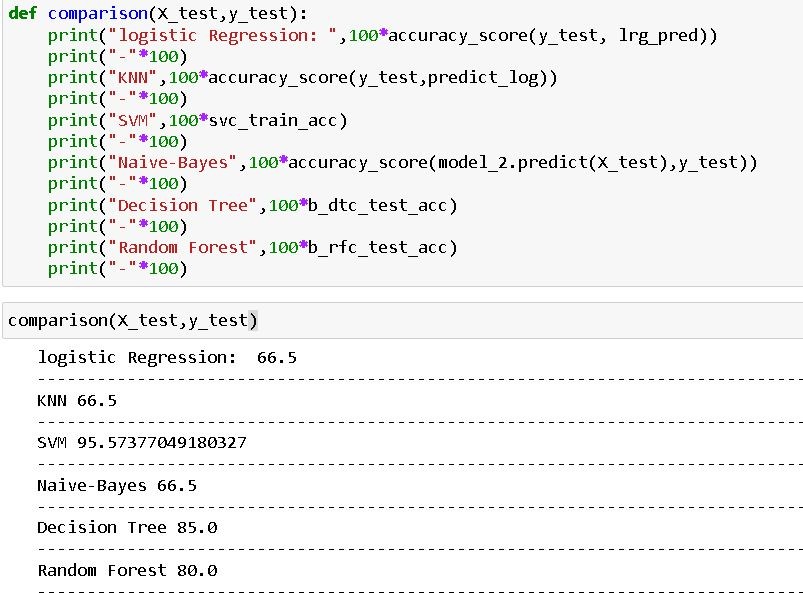
**Milestone 5: Performance Testing & Hyperparameter Tuning**

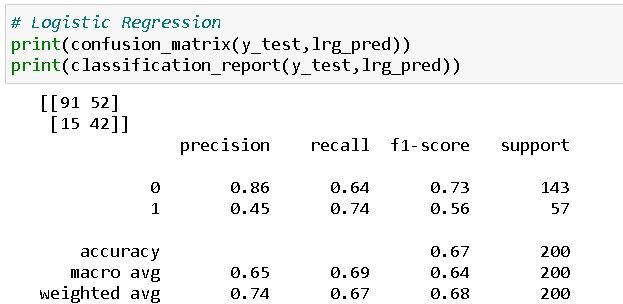
## Activity 1: Testing model with multiple evaluation metrics

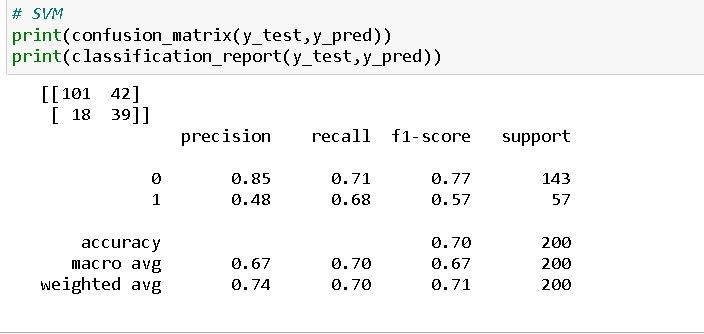
Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision,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

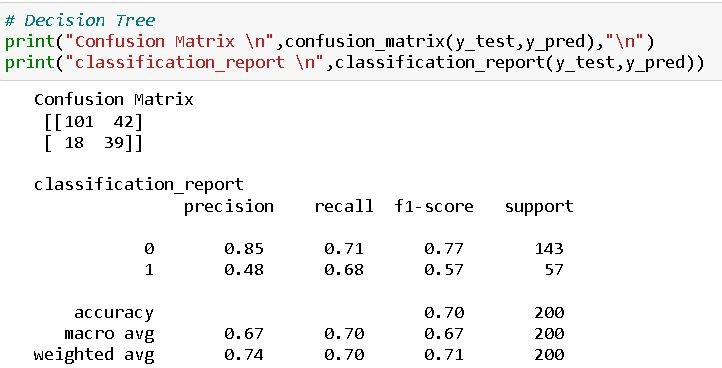
## Activity 1.1: Compare the model

For comparing the above four models, the compareModel function is defined.









After calling the function, the results of models are displayed as output. From the above models Decision Tree is performing well.

## Activity 2: Comparing model accuracy before & after applying hyperparameter tuning (Hyperparameter tuning is optional. For this project it is not required.)

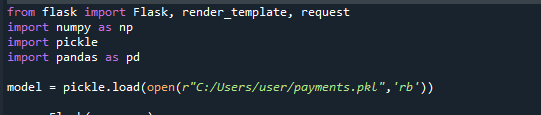
Evaluating performance of the model From sklearn, cross\_val\_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well.

**Note:** To understand cross validation, refer to this [link](https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85)



# 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 also to be able to use it in the future.

## Activity 2: Integrate with Web Framework

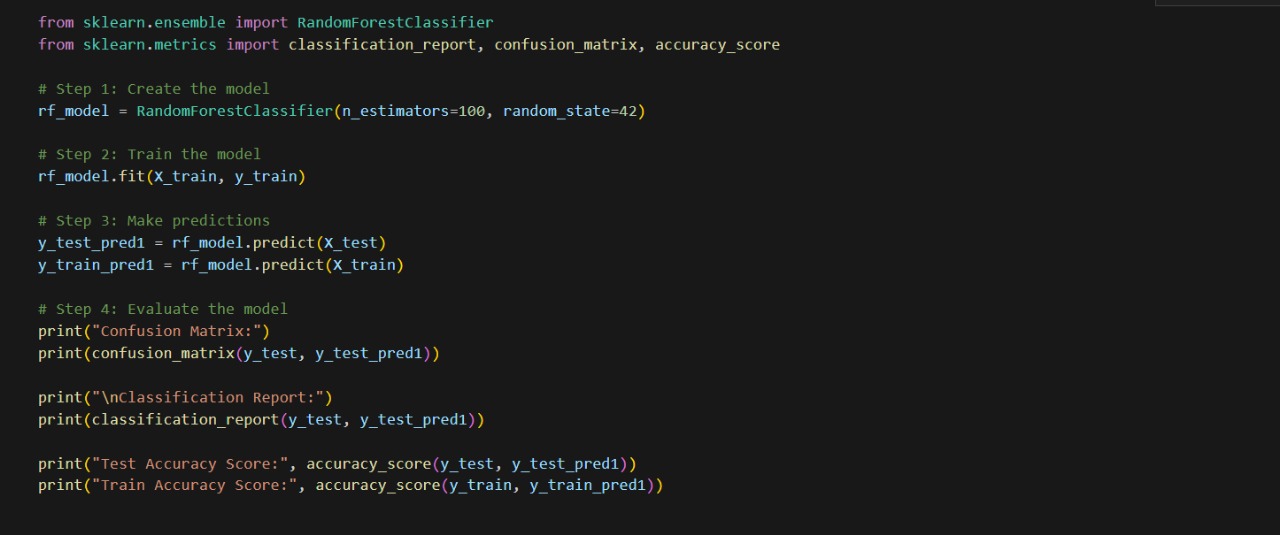
The fraud detection model was deployed using **Flask**. The frontend (HTML, CSS, JavaScript) collects transaction inputs and sends them to the Flask backend using fetch() POST requests. The backend returns a prediction, which is displayed on the webpage for real-time fraud detection.

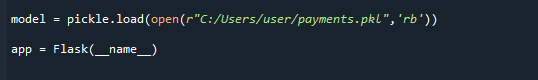
This section has the following tasks

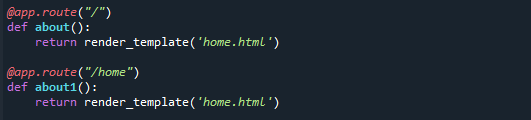
## Activity 2.1: Building Html Page:

* **Frontend**: Built with HTML, CSS, and JavaScript. Inputs include transaction type, amount, and balances. The UI uses a clean, responsive design.
* **Backend**: Flask loads the model.pkl, receives data, predicts fraud, and returns the result.
* **Integration**: JavaScript sends data to Flask using fetch(), and the response is shown instantly on the UI.

## Activity 2.2: Build Python code:

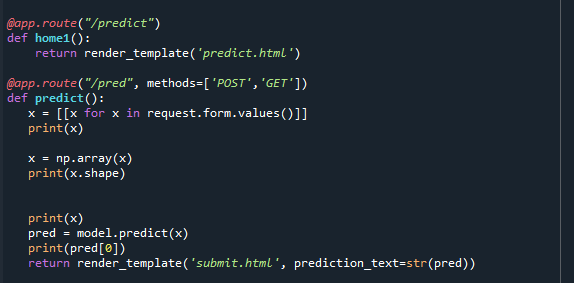


Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module ( name ) as argument.

 Render HTML page:

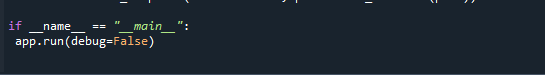
Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the home.html function. Hence, 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.



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.

Main Function:

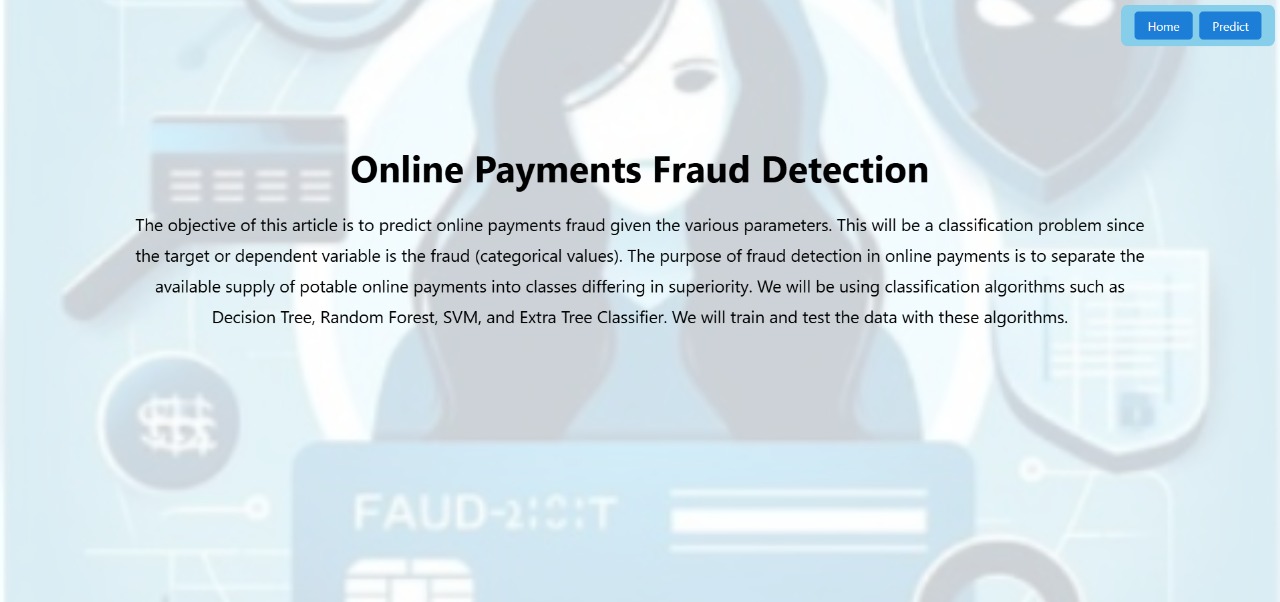
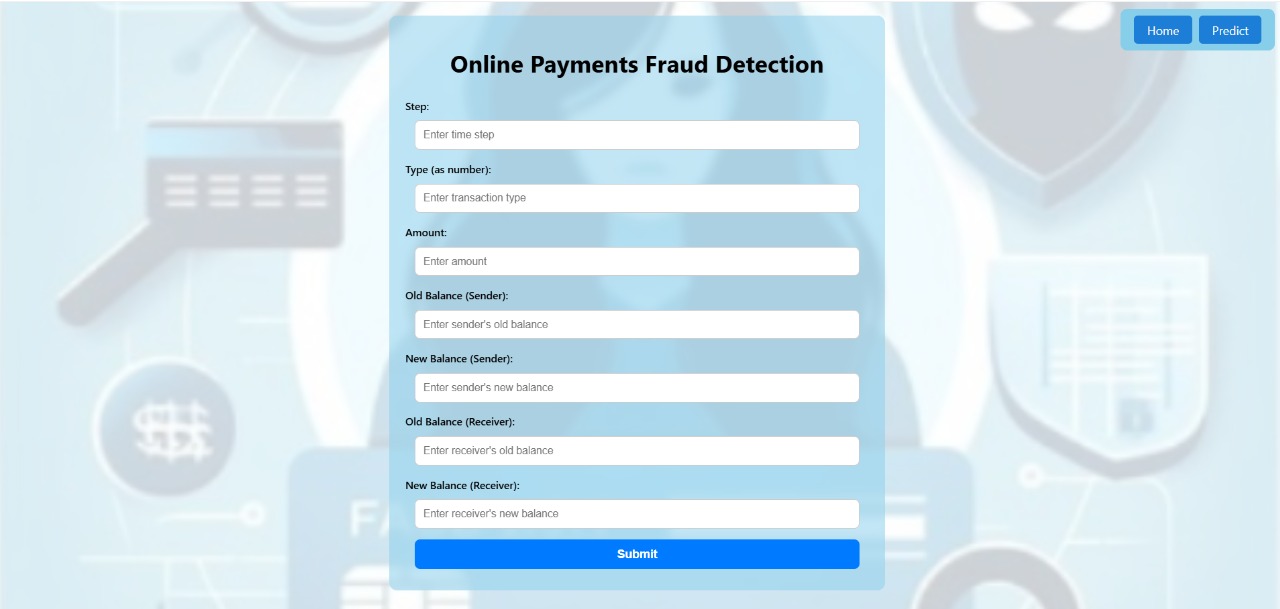


## Activity 2.3: Run the web application

* 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 right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

## https://lh5.googleusercontent.com/a3NVWvlsrS59M_KGJ8_7fCTSjWOk9alhdz1M2ugCPqFzdfdaIC-0lzneQVIaoCfh-Kdb5vsWbTbSoVXJdiv50gFiz_QPyV0WuFVYQPM2iotqJl7ta1zwTMe3PP0zQnJWcp7OEZb1GSP1L-4zGQ99dpMgEd0s9deK4hkGXzVYL0Dpaj8dnykSNg_akJK5leZoQEBNXluumQ

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

Output Screenshots:

