**Online Payments Fraud Detection using ML**

**Team Members:-Aryan Tyagi(Team Lead)**

**Dev Kaushik**

**Vedansh Balooni**

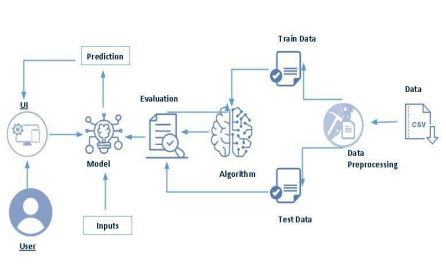
**Suryansh Patial**

**Project Description:**

The growth in internet and e-commerce appears to involve the use of online credit/debit card transactions. The increase in the use of credit / debit cards is causing an increase in fraud. The frauds can be detected through various approaches, yet they lag in their accuracy and its own specific drawbacks. If there are any changes in the conduct of the transaction, the frauds are predicted and taken for further process. Due to large amount of data credit / debit card fraud detection problem is rectified by the proposed method

We will be using classification algorithms such as Decision tree, Random forest, svm, and Extra tree classifier, xgboost Classifier**.**We will train and test the data with these algorithms. From this the best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

**Technical Architecture:**

****

**Pre requisites:**

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

**● Anaconda navigator and pycharm:**

**o** Refer the link below to download anaconda navigator

**o** Link : https://youtu.be/1ra4zH2G4o0

**● Python packages:**

o Open anaconda prompt as administrator

o Type“pip install numpy”and click enter.

o Type“pip install pandas”andclickenter.

o Type“pip install scikit-learn”andclickenter.

o Type”pip install matplotlib”andclickenter.

o Type”pip install scipy”andclickenter.

o Type”pip install pickle-mixin”andclickenter.

o Type”pip install seaborn”andclickenter.

o Type“pipinstallFlask”and click enter.

**Prior Knowledge:**

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

● **ML Concepts**

o Supervisedlearning:

https://www.javatpoint.com/supervised-machine-learning

o Unsupervisedlearning:

https://www.javatpoint.com/unsupervised-machine-learning

o Regression and classification

o Decisiontree:

https://www.javatpoint.com/machine-learning-decision-tree-classificatio n-algorithm

o Randomforest:

https://www.javatpoint.com/machine-learning-random-forest-algorithm **o xgboost Classifier**

https://www.javatpoint.com/xgboost-classifier-algorithm-for-machine-le arning

o Svm:

https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to understand-the-math-behind-Svm/

o Extra tree classifier:

https://www.javatpoint.com/Extratreeclassifier-algorithm-for-machine-learning

o Evaluationmetrics:

https://www.analyticsvidhya.com/blog/2019/08/11-important-model-eva luation-error-metrics/

o **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 about data.

● Have knowledge on pre-processing the data/transformation techniques on outlier and some visualisation concepts.

**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,

● Data collection

o Collect the dataset or create the dataset

● Visualising and analysing data

Importing the libraries

o Read the Dataset

o Univariate analysis

o Bivariate analysis

o Descriptive analysis

● Data pre-processing

o Checking for null values

o Handling outlier

o Handling categorical(object) data

o Splitting data into train and test

● Model building

o Import the model building libraries

o Initialising the model

o Training and testing the model

o Evaluating performance of model

o Save the model

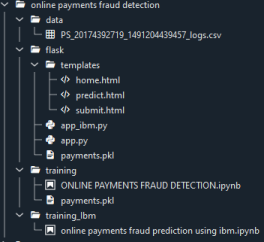
● Application Building

o Create an HTML file

o Build python code

**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.

● Model.pkl is our saved model. Further we will use this model for flask integration.

● Training folder contains model training files and the training\_ibm folder contains IBM deployment files.

**Milestone 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.

**Collect the dataset or create the dataset or Download the dataset:**

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

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

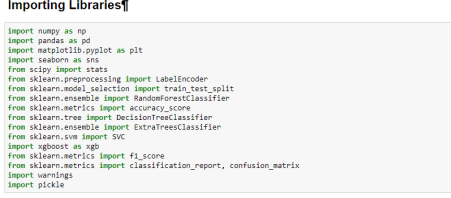
**Milestone 2: Visualising and analysing data**

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.**

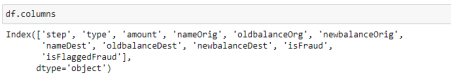
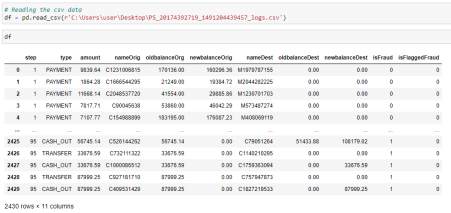
**Activity 1: Importing the libraries**

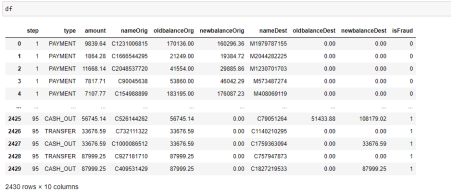
Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

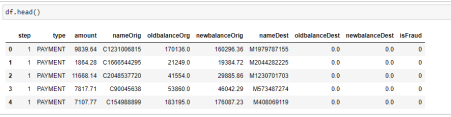
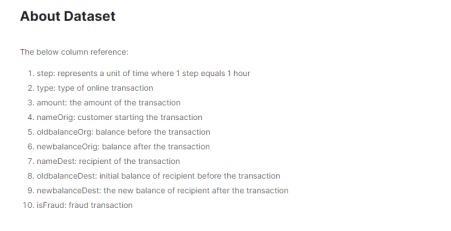
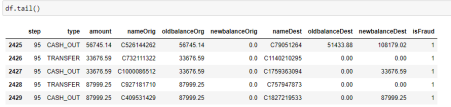
**Activity 2: Read the Dataset**

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

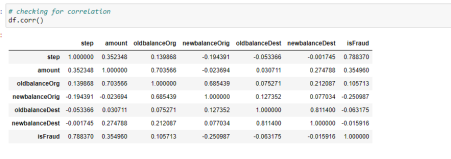
In pandas we have a function called read\_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

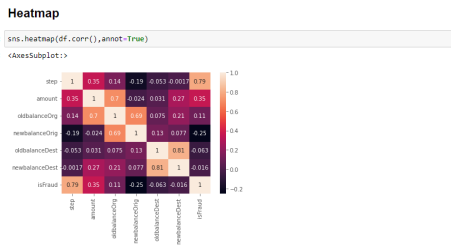
Here, the input features in the dataset are known using the df.columns function.

here, the dataset's superfluous columns are being removed using the drop method.

above, the dataset's first five values are loaded using the head method. above, the dataset's last five values are loaded using the tail method. 

utilising Style use here The Ggplot approach Setting "styles"—basically stylesheets that resemble matplotlibrc files—is a fundamental feature of mpltools. The "ggplot" style, which modifies the style to resemble ggplot, is demonstrated in this dataset.

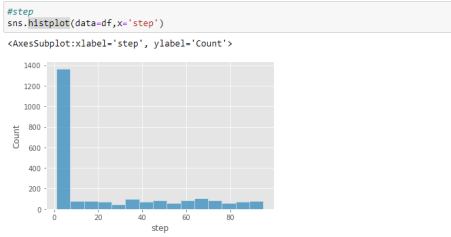
utilising the corr function to examine the dataset's correlation



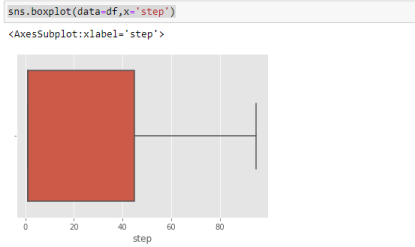
Here, a heatmap is used to understand the relationship between the input attributes and the anticipated goal value.

**Activity 3: Univariate analysis**

In simple words, univariate analysis is understanding the data with a single feature. Here I have displayed the graph such as histplot .



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.

Here, the relationship between the step attribute and the boxplot is visualised.



Here, the counts of observations in the type attribute of the dataset will be displayed using a countplot.



By creating bins along the data's range and then drawing bars to reflect the number of observations that fall within the amount attribute in the dataset.

Here, the relationship between the amount attribute and the boxplot is visualised.



By creating bins along the data's range and then drawing bars to reflect the number of observations that fall within the oldbalanceOrg attribute in the dataset.



utilising the value counts() function here to determine how many times the nameDest column appears.



Here, the relationship between the oldbalanceDest attribute and the boxplot is visualised.



Here, the relationship between the newbalanceDest attribute and the boxplot is visualised.



using the countplot approach here to count the number of instances in the dataset's target isFraud column.



Here, we're using the value counts method to figure out how many classes there are in the dataset's target isFraud column.

converting 0-means: is not fraud and 1-means: is fraud using the loc technique here **Activity 4: 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 type and isFraud.countplot 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 step.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 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 oldbalanceOrg. 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 newbalanceOrig. 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 oldbalanceDest. violinplot 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.



**Activity 5: 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.



**Milestone 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

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.



Here, I'm using the shape approach to figure out how big my dataset is 

here, the dataset's superfluous columns (nameOrig,nameDest) are being removed using the drop method.

**Activity 1: 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.



For checking the null values, data.isnull() function is used. To sum those null values we use the .sum() function to it. From the above image we found that there are no null values present in our dataset.So we can skip handling of missing values step.

determining the types of each attribute in the dataset using the info() function **Activity 2: Handling outliers**

****Here, a boxplot is used to identify outliers in the dataset's amount attribute.

Here, transformationPlot is used to plot the dataset's outliers for the amount

property.

**Activity 3: Object data labelencoding**

****using labelencoder to encode the dataset's object type

**Activity 4: Splitting data into train and test**

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.



**Milestone 4: Model Building**

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 four classification algorithms. The best model is saved based on its performance.

**Activity 1: 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.



**Activity 2: 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.



**Activity 3: 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.

**Activity 4: 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



preprocessing class of sklearn. LabelEncoder[source] 0 to n classes-1 as the range for the target labels to be encoded. Instead of encoding the input X, the target values, i.e. y, should be encoded using this transformer.



**Activity 5: 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



**Activity 6: 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 five models, the svc is performing well. From the below image, We can see the accuracy of the model is 79% accuracy. .

**Activity 7: Evaluating performance of the model and saving the model**

From sklearn, accuracy\_score is used to evaluate the score of the model. On the parameters, we have given svc (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model is svc by pickle.dump().



**Milestone 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

**Activity1: Building Html Pages:**

For this project create three HTML files namely

● home.html

● predict.html

● submit.html

and save them in the templates folder.

Let’s see how our home.html page looks like:



Now when you click on predict button from top right corner you will get redirected to predict.html

Let's look how our predict.html file looks like:



Now when you click on submit button from left bottom corner you will get redirected to submit.html

Let's look how our submit.html file looks like:

**Activity 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 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.

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.

Main Function:

**Activity 3: 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.

● 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.

**Output screensorts:**

****

****

****

****