

Project Documentation Phase

Date	09 November 2023
Team ID	591653
Project Name	Project – Online Fraud Detection

1. Introduction

1.1 Project Overview

The primary objective of our project is to develop a predictive model that can accurately classify transactions as either legitimate or fraudulent. The problem is classified as a binary classification problem, which requires the use of supervised machine learning algorithms. Our approach involves deploying well-established machine learning models that have been trained on a large dataset of labeled transactions to optimize prediction accuracy. Through the use of these models, we aim to provide businesses with a reliable and efficient tool to mitigate the risk of fraudulent transactions in their operations.

1.2 Purpose

Our objective is to accurately classify transactions as either legitimate or fraudulent, which is a binary classification problem. To achieve the highest prediction accuracy, we plan to leverage the power of Supervised Machine Learning models. By training these models on a large dataset of labeled transactions, we aim to develop a system that can reliably and efficiently detect fraudulent transactions. This approach has the potential to significantly reduce the financial losses that businesses face due to fraudulent activities.

2. Literature Survey

2.1 Existing Problem

- Each payment system has its limits regarding the maximum amount in the account, the number of transactions per day and the amount of output.
- If Internet connection fails, you cannot get to your online account.
- If you follow the security rules the threat is minimal. The worse situation when the system of processing company has been broken because it leads to the leak of personal data on cards and its owners.
- The information about all the transactions, including the amount, time and recipient are stored in the database of the payment system. And it means the intelligence agency has access to this information. Sometimes this is the path for fraudulent activities.

2.3 Problem Statement Definition

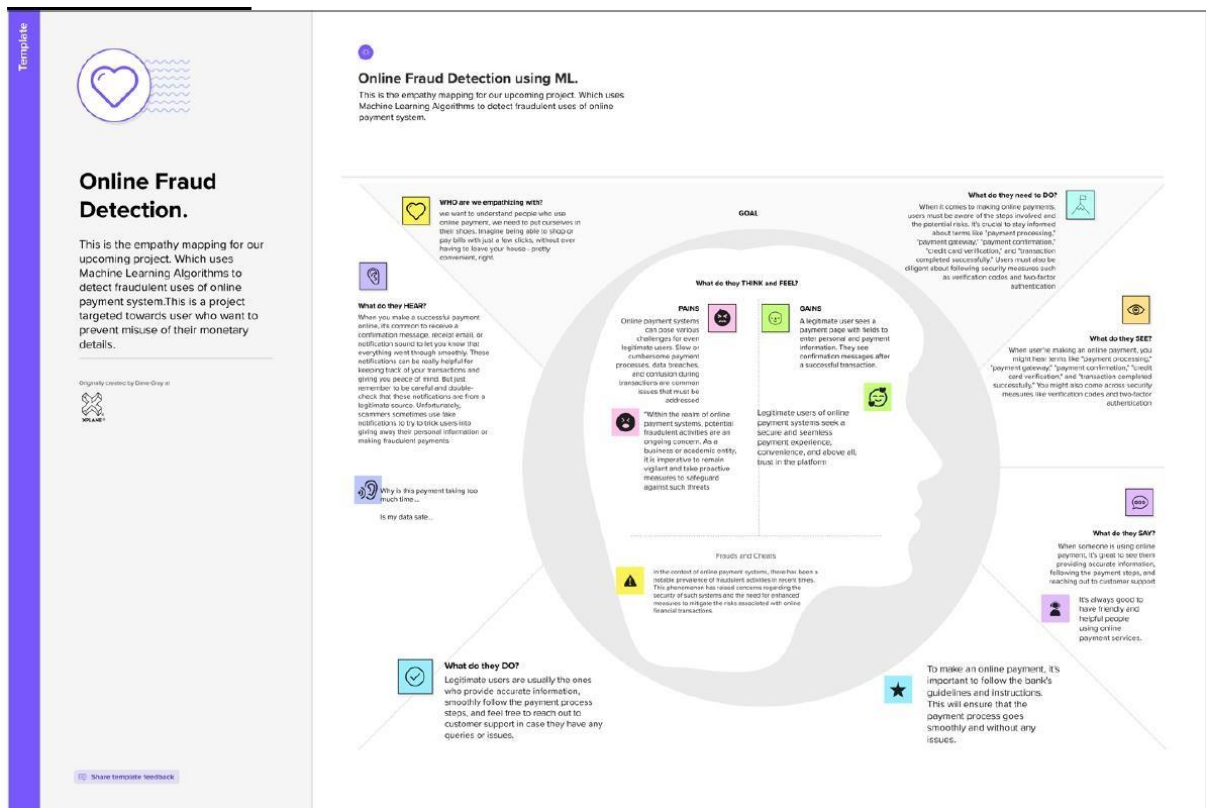
The proposed methodology aims to predict the legality of transactions and detect fraudulent transactions, making it a classification problem. To achieve the highest possible prediction accuracy, supervised machine learning models will be deployed. The perpetration of online fraudulent transactions constitutes a significant criminal violation that costs both individuals and financial institutions billions of dollars annually. It underscores the critical role of financial institutions in detecting and preventing fraudulent acts. To this end, machine learning algorithms provide a proactive mechanism for preventing online transaction frauds with a high degree of accuracy.

The proliferation of online payment methods in e-commerce and other online sites has made online transaction fraud an alluring target. However, the rise in fraud rates has necessitated the use of machine learning approaches to identify and evaluate fraud in online transactions. This project aims to implement supervised machine learning models for fraud detection, which involves analysing prior transaction information and classifying transactions into distinct groups based on their type. Subsequently, various classifiers are trained independently, and the models are evaluated for accuracy. The classifier with the highest rating score can then be selected as one of the best methods for predicting fraud.

We will be using classification algorithms such as Decision tree, Random forest, Extra tree classifier. We will train and test the data with these algorithms

3. Ideation and Proposed Solution 3.1

Empathy Map Canvas



3.2 Ideation and Brainstorming

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Pratham Swarnkar

Arpit Aditya

Find the fraudster	Notify the user	Support Vector Machine	Random Forest	KNN Classifier.
Decision Tree	Logistic Regression	Naive Bayes Classifier	Artificial Neural Network.	Awareness about the fraudulent payment.

4. Requirement Analysis

4.1 Functional Requirement

Fraud detection is a knowledge-intensive activity. The main AI techniques used for fraud detection include:

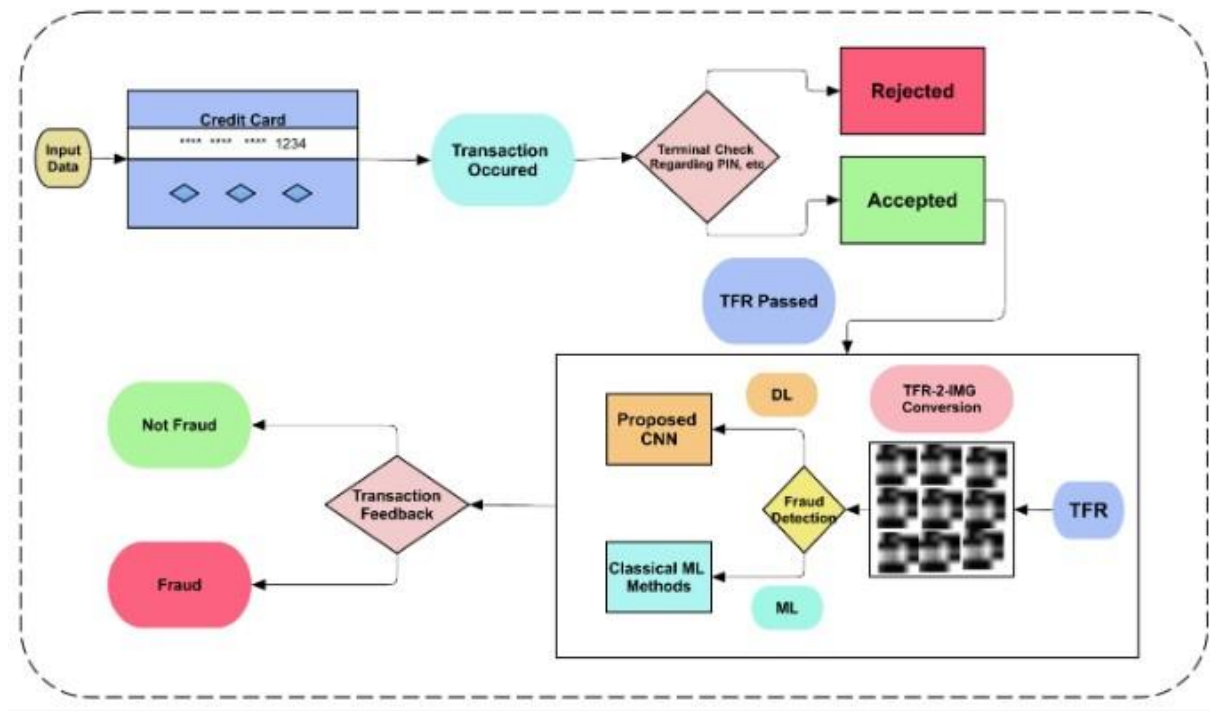
- Data mining to classify, cluster, and segment the data and automatically find associations and rules in the data that may signify interesting patterns, including those related to fraud.
- Expert systems to encode expertise for detecting fraud in the form of rules.
- Pattern recognition to detect approximate classes, clusters, or patterns of suspicious behavior either automatically (unsupervised) or to match given inputs.
- Machine learning techniques to automatically identify characteristics of fraud
- High Performance
- Secure
- Easy to Use

4.2 Nonfunctional Requirement

- Portable

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



		USN-2	Upon completion of the registration process for the application, the user will receive an email confirmation at the email address that they provided during the registration process. This email serves as a means of verifying the user's email address and confirming their successful registration for the application	Upon receiving the confirmation email, the user can click on the confirmation link provided within the email in order to confirm their email address and complete the registration process for the application	High	Sprint-1
		USN-3	As a user, you have the option to register for the application through an online payment app. This process involves completing the registration form within the payment app and providing the necessary details for registration	I can register & access the dashboard with online payment app	Low	Sprint-2
	Dashboard					
Customer (Web user)	Login	USN	As a user, you have the option to log in to the website using your Gmail credentials. This involves selecting the "Login with Gmail" option on the website's login page and entering your Gmail email address and password	After logging in to the website using your Gmail credentials, you may receive a confirmation email at the email address associated with your Gmail account. This confirmation email serves to verify your email address and confirm your successful login to the website	High	Sprint-1
Customer Care Executive		USN	As a user if you experience any issues while using the application, our customer care team is readily available to assist you. You can contact us via email at our Gmail address or by phone using the provided phone number.	Our team is dedicated to providing prompt and effective assistance in resolving any issues or concerns that you may have.	Very High	Sprint-1

5.2 Solution Architecture

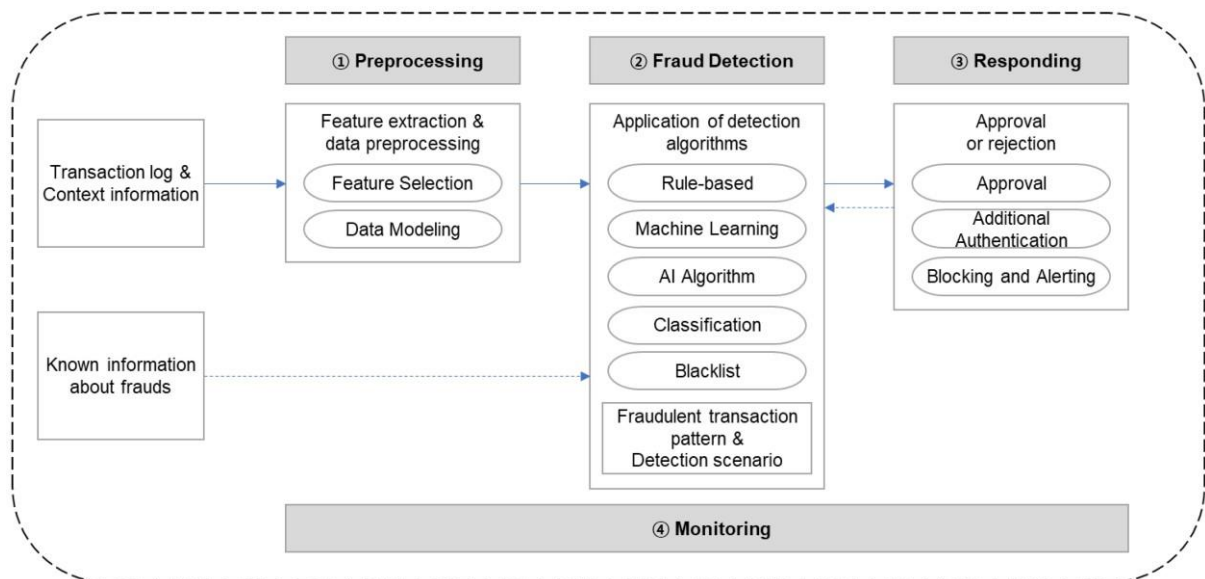
The objective of this study is to predict whether a transaction is legitimate or fraudulent. The proposed solution to address these issues involves undertaking a comprehensive data

exploration and cleaning process, followed by selecting an appropriate machine learning algorithm.

- Data Acquisition
- Exploratory Data Analysis
- Feature Engineering
- Data Processing
- Supervised Machine Learning Model Deployment
- Results Analysis

6. Project Planning & Scheduling

6.1 Technical Architecture



6.2 Sprint Planning & Estimation and Deleviring

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1 [Ideation]	7	3 Days	18 Oct 2023	20 Oct 2023	7	05 Nov 2023
Sprint-2 [project design]	16	4 Days	21 Oct 2023	25 Oct 2023	16	
Sprint-3 [Project Planning]	19	3 Days	25 Oct 2023	30 Oct 2023	19	

Sprint-4 [Project Development]	11	4 Days	30 Oct 202	02 Nov 2023	11	
-----------------------------------	----	--------	------------	-------------	----	--

BurnDown Chart			
Dates	Planned Tasks	Actual Tasks	Task name
18-10-2023	2	2	Ideation
19-10-2023	3	4	
20-10-2023	2	1	
21-10-2023	4	3	project design
22-10-2023	5	3	
23-10-2023	8	7	
24-10-2023	1	3	
25-10-2023	3	3	
26-10-2023	6	5	Project Planning
27-10-2023	2	3	
28-10-2023	3	4	
29-10-2023	5	4	
30-10-2023	1	2	
31-10-2023	2	3	Project Developpomen
01-11-2023	5	4	
02-11-2023	3	2	
03-11-2023	4	4	
04-11-2023	3	4	
05-11-2023	5	4	

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

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

Analysis and Results

Visualizations are performed in each step, in order to highlight new insights about the underlying patterns and relationships contained within the data. The data analysis process for the deployment of classification models is based on the following steps.

i) Data Acquisition

Download data

Upload data in Python environment

ii) Data Exploration

Checking data head, info, summary statistics and null value

iii) Feature Engineering

We can see from the above data that only two type of transactions are classified as fraud so we will drop the remaining types to generalize the data and we will only keep

Cash out and Transfer type

iv) Data Processing

After feature engineering we scaled the data

We dropped the uninterested and unscaled features from the dataset.

Supervised Machine Learning

- Modeling of the classifier system
- Validation of the model
- Visualization of the model
- Visualization and interpretation of the results

Results analysis

- Deployment of the models
 - Comparing prediction accuracy of ML models
 - Visualization of the results
- #### Application Building
- Create an HTML file
 - Build python code
- #### Project Structure:

Milestone 1: Data Collection

We will develop a predictive model using a provided dataset to detect fraudulent transactions at a credit card company. The goal is to ensure that customers are only charged for items they have purchased. The dataset, which includes transaction details such as amount, date, time, and location, will go through data preprocessing before employing machine learning algorithms such as logistic regression, decision trees, and random forests to develop the model. Model accuracy will be validated using various metrics, and techniques such as crossvalidation and hyperparameter tuning will be employed to optimize its performance. The dataset can be found here: [Dataset Link](#)

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.

i. Import the necessary packages and dataset into python

```
In [1]: # Importing necessary libraries
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

```
2023-11-07 18:54:17.024977: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-11-07 18:54:18.630148: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
```


ii. Read the Dataset

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.

```
In [2]: # Loading dataset for preprocessing
df=pd.read_csv('./Dataset/Unzip/Credit_Card.csv')
df=df[0:30000]

In [3]: # Extracting the column name for the dataset
column=df.columns
column

Out[3]: Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrig', 'newbalanceOrig',
              'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
              'isFlaggedFraud'],
              dtype='object')

In [4]: # df.drop_duplicates()
# Dropping duplicate value since dataset is very large

In [5]: # fig=plt.figure(figsize=(10,7))
# df['type'].value_counts(normalize=True).plot(kind='bar')
```

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

iii. Univariate analysis

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

Univariate Analysis

```
In [6]: sns.distplot(df['amount'])

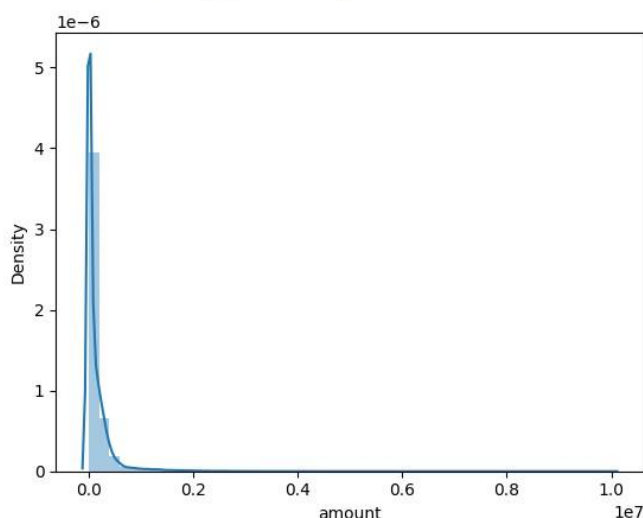
/tmp/ipykernel_14219/1086040597.py:1: UserWarning:
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(df['amount'])
```

```
Out[6]: <Axes: xlabel='amount', ylabel='Density'>
```

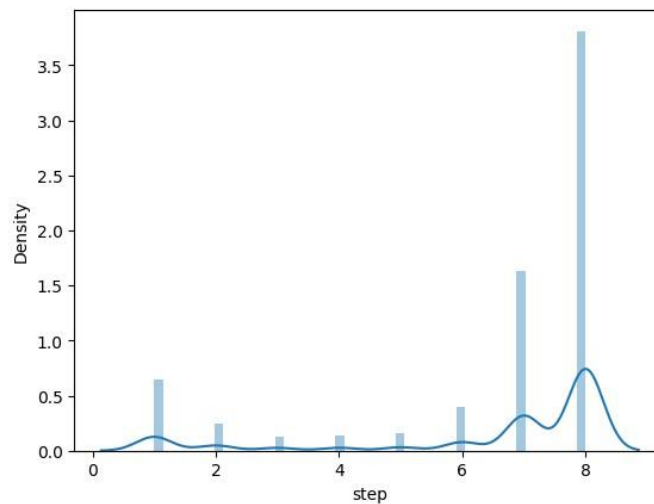


Here the relation of Amount is visualized by using `sns.distplot`. **Distplot** is used to observe how the data is distributed in a particular dataset

```
In [7]: sns.distplot(df['step'])
```

```
/tmp/ipykernel_14219/2318427200.py:1: UserWarning:  
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.  
  
Please adapt your code to use either `displot` (a figure-level function with  
similar flexibility) or `histplot` (an axes-level function for histograms).  
  
For a guide to updating your code to use the new functions, please see  
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751  
  
sns.distplot(df['step'])
```

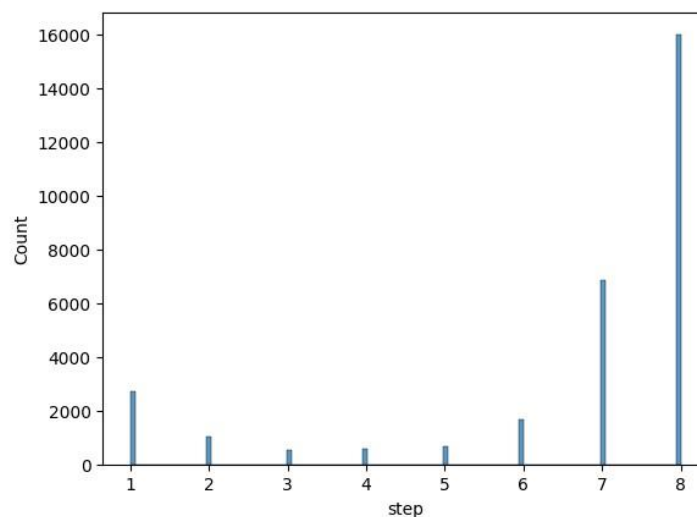
```
Out[7]: <Axes: xlabel='step', ylabel='Density'>
```



Here the relation step is visualized by using `sns.distplot`. **Distplot** is used to observe how the data is distributed in a particular dataset

```
In [8]: sns.histplot(data=df, x='step')
```

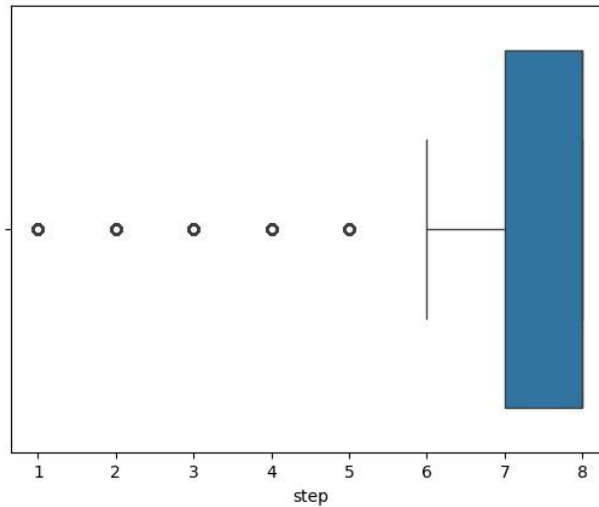
```
Out[8]: <Axes: xlabel='step', ylabel='Count'>
```



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 we count steps by using `sns.histplot`

```
In [9]: sns.boxplot(data=df,x='step')
```

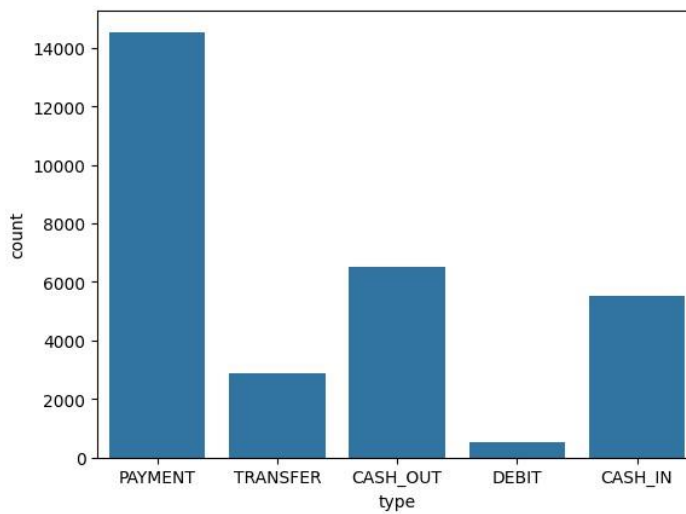
```
Out[9]: <Axes: xlabel='step'>
```



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

```
In [10]: sns.countplot(data=df,x='type')
```

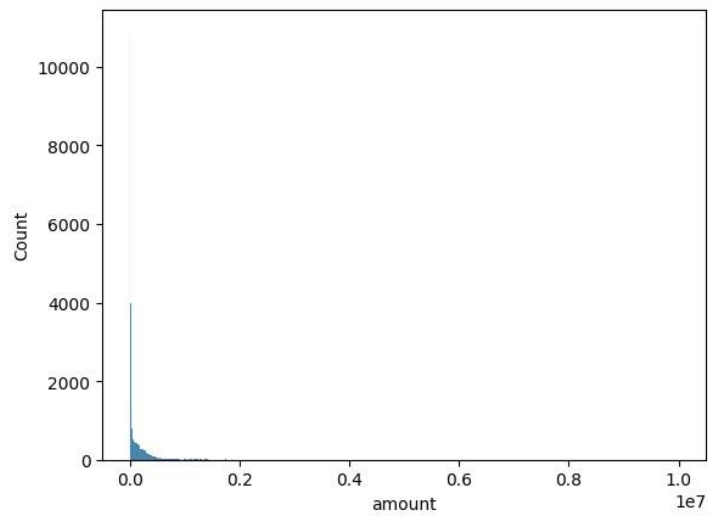
```
Out[10]: <Axes: xlabel='type', ylabel='count'>
```



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

```
In [11]: sns.histplot(data=df,x='amount')  
# amount
```

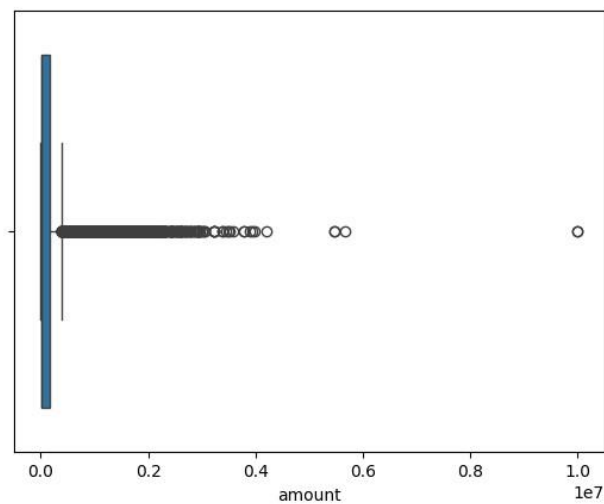
```
Out[11]: <Axes: xlabel='amount', ylabel='Count'>
```



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.

```
In [12]: sns.boxplot(data=df,x='amount')
```

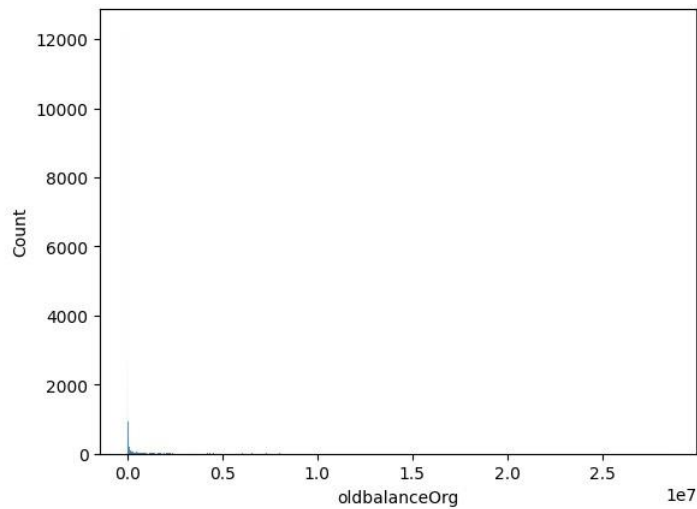
```
Out[12]: <Axes: xlabel='amount'>
```



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

```
In [13]: sns.histplot(data=df,x='oldbalanceOrg')

Out[13]: <Axes: xlabel='oldbalanceOrg', ylabel='Count'>
```



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.

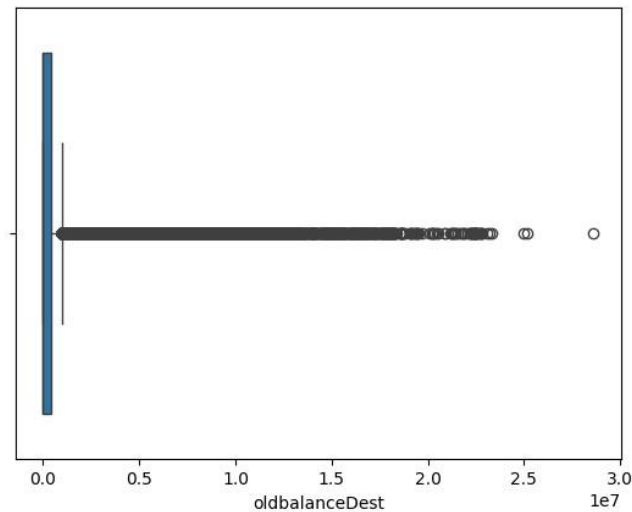
```
In [14]: df['nameDest'].value_counts()

Out[14]: nameDest
C985934102    71
C1286084959    61
C1590550415    60
C2083562754    59
C248609774     57
..
M1553206824     1
M961994506      1
M244114865      1
M1693575672     1
M640263118      1
Name: count, Length: 17801, dtype: int64
```

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

```
In [15]: sns.boxplot(data=df,x='oldbalanceDest')
```

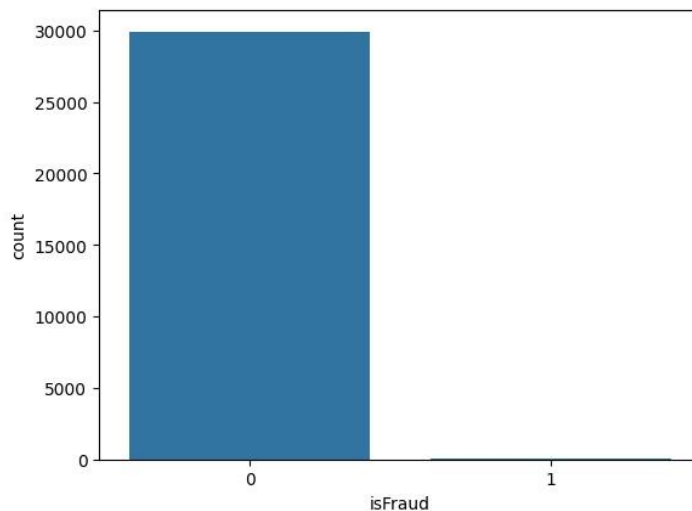
```
Out[15]: <Axes: xlabel='oldbalanceDest'>
```



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

```
In [17]: # isFraud
sns.countplot(data=df,x='isFraud')
```

```
Out[17]: <Axes: xlabel='isFraud', ylabel='count'>
```



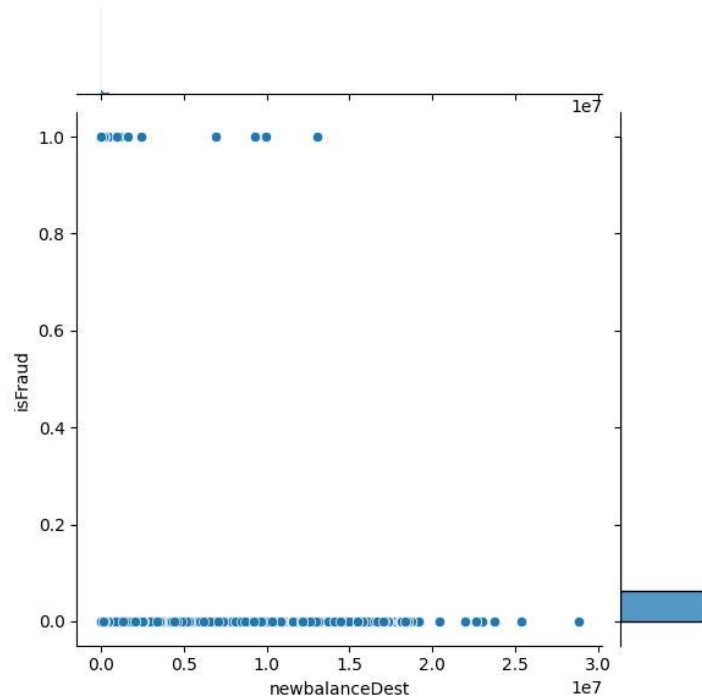
using the countplot approach here to count the number of instances in the dataset's target isFraud column.

iv. 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.

```
In [18]: sns.jointplot(data=df,x='newbalanceDest',y='isFraud')
```

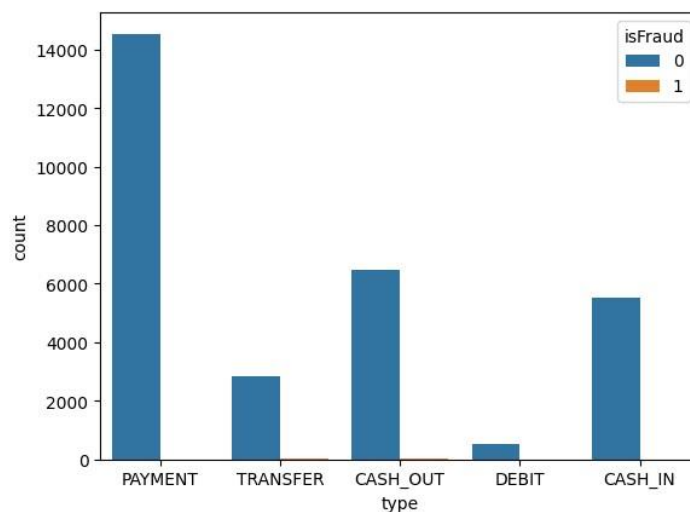
```
Out[18]: <seaborn.axisgrid.JointGrid at 0x7f9169074c40>
```



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

```
In [19]: sns.countplot(data=df,x='type',hue='isFraud')
```

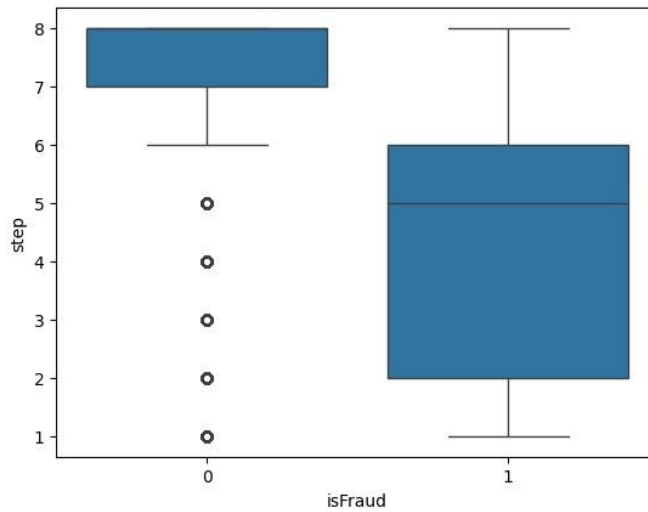
```
Out[19]: <Axes: xlabel='type', ylabel='count'>
```



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

```
In [20]: sns.boxplot(data=df, x='isFraud', y='step')
```

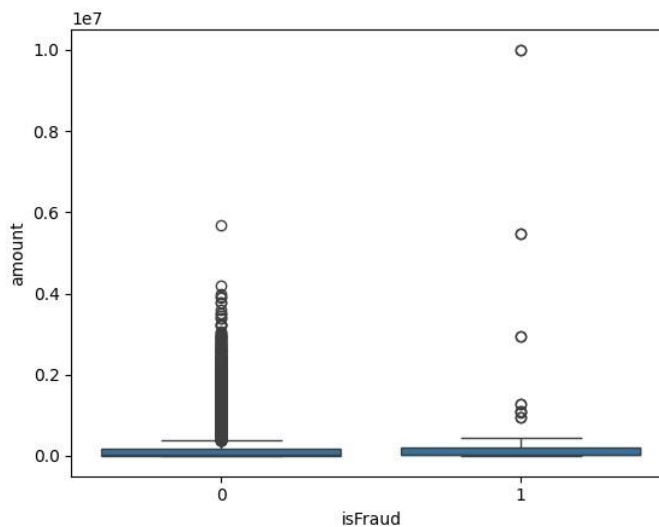
```
Out[20]: <Axes: xlabel='isFraud', ylabel='step'>
```



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.

```
In [21]: sns.boxplot(data=df, x='isFraud', y='amount')
```

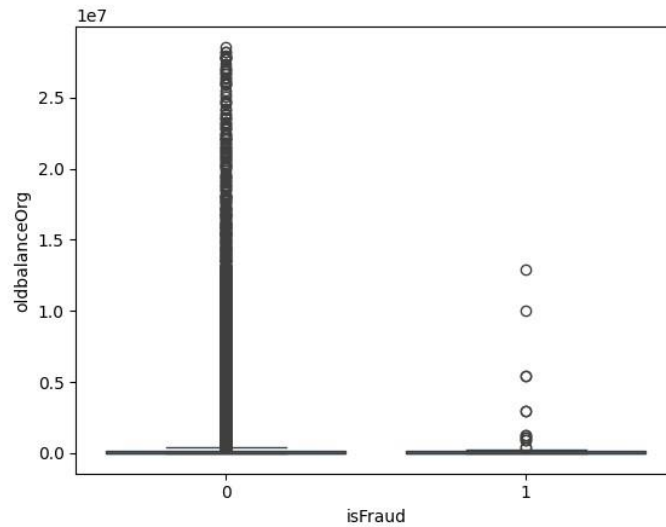
```
Out[21]: <Axes: xlabel='isFraud', ylabel='amount'>
```



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


```
In [22]: sns.boxplot(data=df,x='isFraud',y='oldbalanceOrg')
```

```
Out[22]: <Axes: xlabel='isFraud', ylabel='oldbalanceOrg'>
```

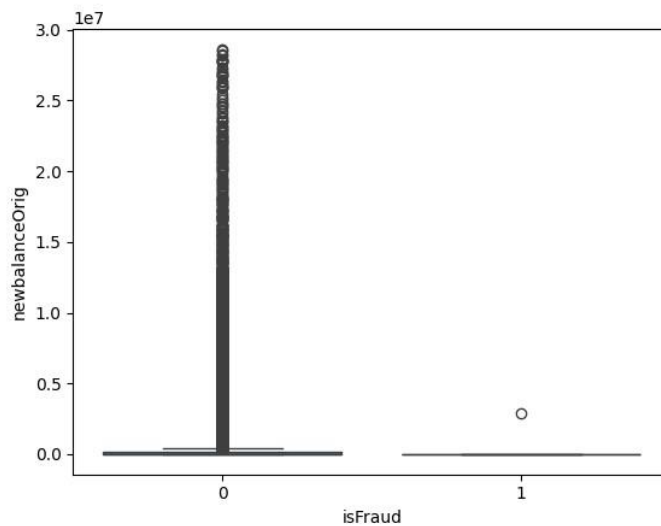


Here we are visualising the relationship between isFraud and newbalanceorg. boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value

isFraud

```
In [23]: sns.boxplot(data=df,x='isFraud',y='newbalanceOrig')
```

```
Out[23]: <Axes: xlabel='isFraud', ylabel='newbalanceOrig'>
```



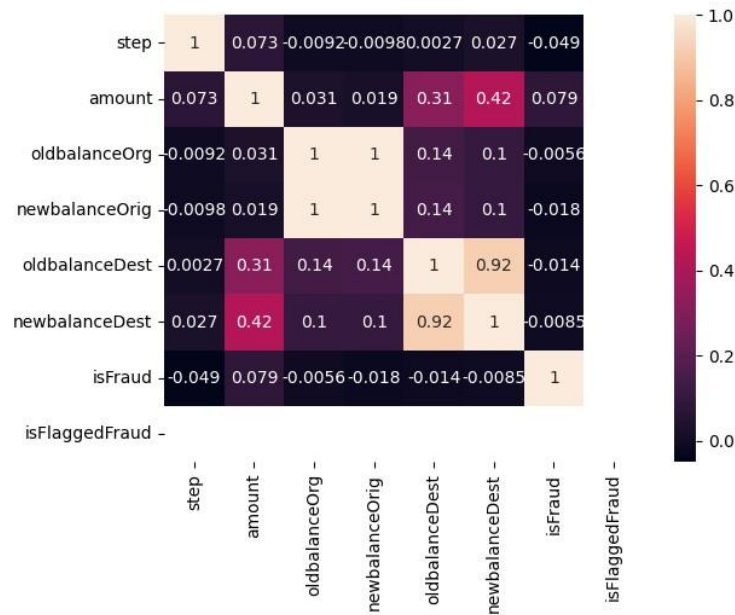
V.

Multivariate Analysis

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

```
In [24]: sns.heatmap(df.corr(numeric_only=True),annot=True)
```

```
Out[24]: <Axes: >
```



Milestone 3: Data Preprocessing

Before we can train a machine learning model on the downloaded data set, we need to preprocess it to remove any randomness and ensure accurate results. The preprocessing steps include:

- 1. Handling missing values:** We need to identify any missing data in the dataset and decide how to handle it. This could involve removing the data, replacing it with a default value, or using a machine learning algorithm to predict the missing values.
- 2. Handling Object data label encoding:** The machine learning algorithm requires numerical data to work with, so we need to convert any object data (such as text or categorical data) into numerical data using label encoding.
- 3. Splitting dataset into training and test set:** In order to evaluate the performance of the machine learning model, we need to split the dataset into a training set and a test set. The training set will be used to train the model and the test set will be used to evaluate its accuracy.

By following these pre-processing steps, we can ensure that our machine learning model is accurate and reliable.

```
In [29]: # Size of dataset
df.shape
```

```
Out[29]: (30000, 8)
```

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

```
In [27]: df.drop(['nameOrig', 'nameDest', 'isFlaggedFraud'], axis=1, inplace=True)
```

```
In [28]: # Getting info about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   step            30000 non-null  int64
1   type            30000 non-null  object
2   amount          30000 non-null  float64
3   oldbalanceOrig  30000 non-null  float64
4   newbalanceOrig  30000 non-null  float64
5   oldbalanceDest  30000 non-null  float64
6   newbalanceDest  30000 non-null  float64
7   isFraud         30000 non-null  int64
dtypes: float64(5), int64(2), object(1)
memory usage: 1.8+ MB
```

memory usage: 1.8+ MB

```
In [29]: # Size of dataset
df.shape
```

```
Out[29]: (30000, 8)
```

```
In [30]: # Describing the dataset
df.describe()
```

```
Out[30]:
```

	step	amount	oldbalanceOrig	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
count	30000.000000	3.000000e+04	3.000000e+04	3.000000e+04	3.000000e+04	3.000000e+04	30000.000000
mean	6.592933	1.376150e+05	8.663522e+05	8.829833e+05	8.633675e+05	1.193743e+06	0.002800
std	2.251725	3.033855e+05	2.495397e+06	2.534912e+06	2.541670e+06	3.104353e+06	0.052842
min	1.000000	1.770000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000
25%	7.000000	6.078300e+03	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000
50%	8.000000	2.075971e+04	2.005050e+04	4.246065e+03	0.000000e+00	0.000000e+00	0.000000
75%	8.000000	1.623463e+05	1.522215e+05	1.569253e+05	3.918882e+05	7.033162e+05	0.000000
max	8.000000	1.000000e+07	2.854724e+07	2.861740e+07	2.861740e+07	2.878359e+07	1.000000

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

vi. 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.

```
In [31]: # Checking for null values
df.isnull().any()
# no null values are present so no need for removal

Out[31]: step                False
type                False
amount              False
oldbalanceOrig      False
newbalanceOrig      False
oldbalanceDest      False
newbalanceDest      False
isFraud             False
dtype: bool
```

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.

```
In [28]: # Getting info about the dataset
df.info()

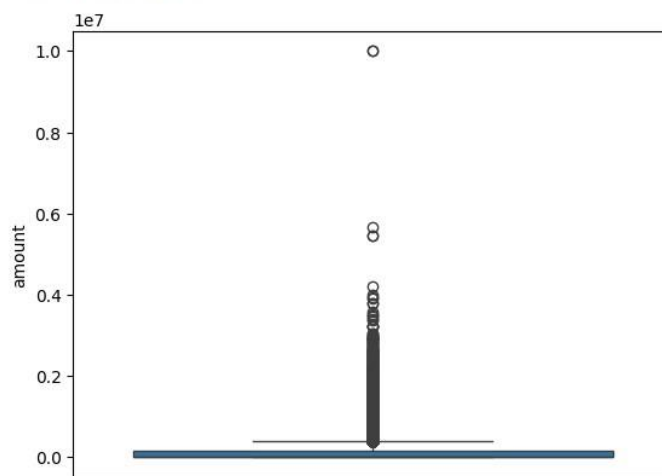
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   step                  30000 non-null  int64
1   type                  30000 non-null  object
2   amount                30000 non-null  float64
3   oldbalanceOrig        30000 non-null  float64
4   newbalanceOrig        30000 non-null  float64
5   oldbalanceDest        30000 non-null  float64
6   newbalanceDest        30000 non-null  float64
7   isFraud               30000 non-null  int64
dtypes: float64(5), int64(2), object(1)
memory usage: 1.8+ MB
```

determining the types of each attribute in the dataset using the `info()` function **vii.**

Handling outliers

```
In [34]: sns.boxplot(df['amount'])
# There are outliers in the amount column
```

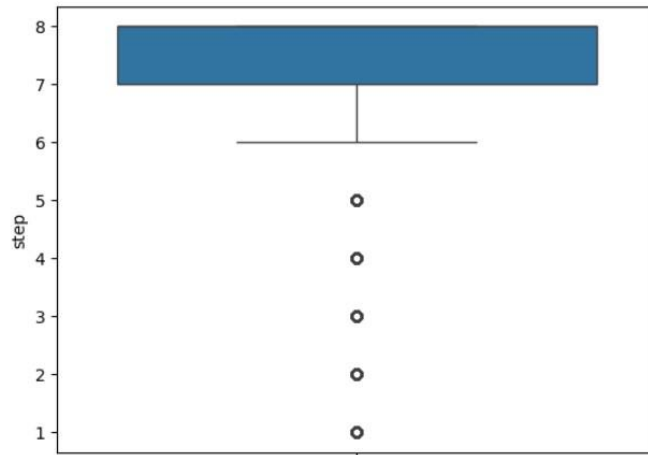
```
Out[34]: <Axes: ylabel='amount'>
```



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

```
In [35]: sns.boxplot(df['step'])
```

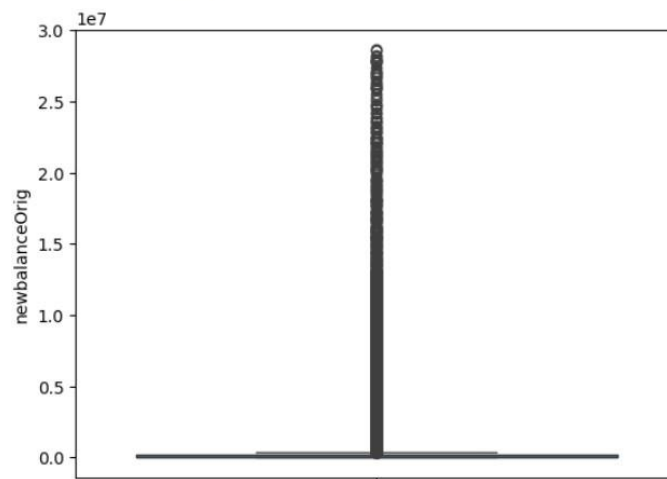
```
Out[35]: <Axes: ylabel='step'>
```



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

```
In [36]: sns.boxplot(df['newbalanceOrig'])
```

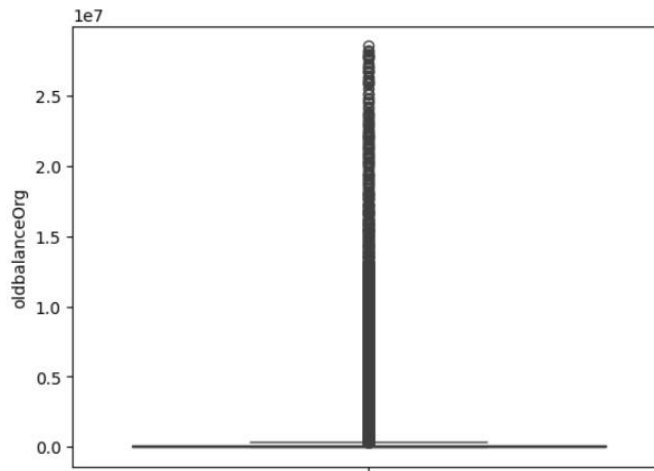
```
Out[36]: <Axes: ylabel='newbalanceOrig'>
```



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

```
In [37]: sns.boxplot(df['oldbalanceOrg'])
```

```
Out[37]: <Axes: ylabel='oldbalanceOrg'>
```



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

viii. Remove the using outlier

```
In [38]: # Removing outliers using loop
from scipy import stats
amount_zscore=stats.zscore(df['amount'])
step_zscore=stats.zscore(df['step'])
newbalanceOrig_zscore=stats.zscore(df['newbalanceOrig'])
oldbalanceOrg_zscore=stats.zscore(df['oldbalanceOrg'])
df=df[np.abs(amount_zscore)<3]
df=df[np.abs(step_zscore)<3]
df=df[np.abs(newbalanceOrig_zscore)<3]
df=df[np.abs(oldbalanceOrg_zscore)<3]
```

```
/tmp/ipykernel_14219/2992249858.py:8: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
df=df[np.abs(step_zscore)<3]
/tmp/ipykernel_14219/2992249858.py:9: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
df=df[np.abs(newbalanceOrig_zscore)<3]
/tmp/ipykernel_14219/2992249858.py:10: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
df=df[np.abs(oldbalanceOrg_zscore)<3]
```

```
In [39]: def transformationPlot(feature):
plt.figure(figsize=(10,6))
plt.subplot(1,2,1)
sns.distplot(feature)
plt.subplot(1,2,2)
stats.probplot(feature,plot=plt)
```

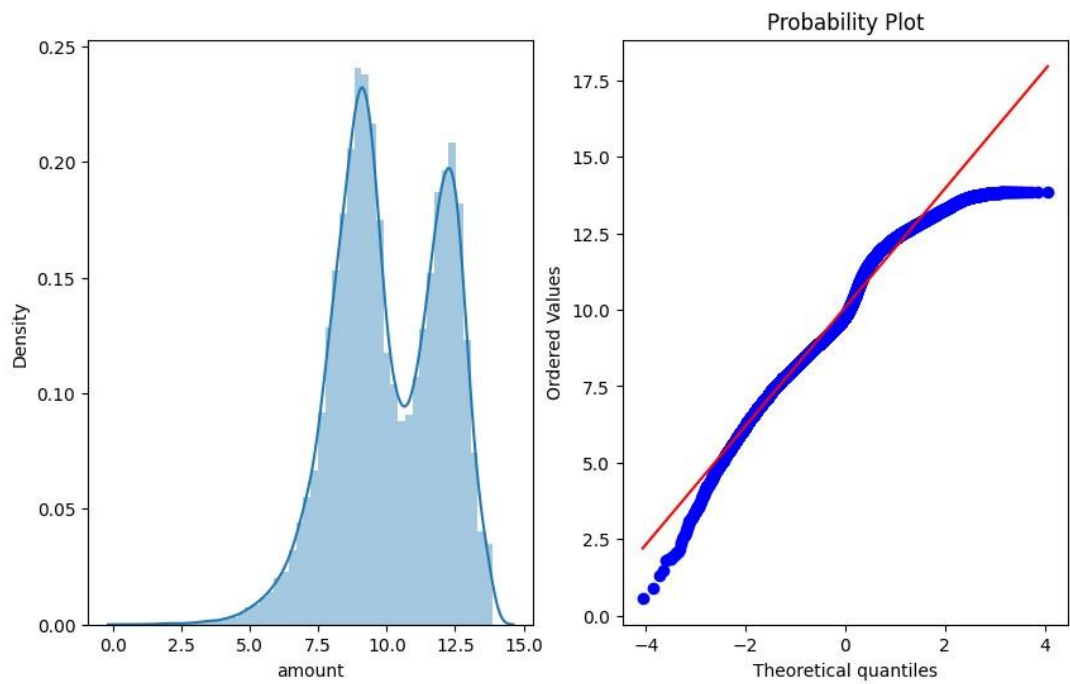
```
In [40]: transformationPlot(np.log(df['amount']))
```

```
/tmp/ipykernel_14219/479904814.py:4: UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```



```
In [41]: transformationPlot(np.log(df['step']))
```

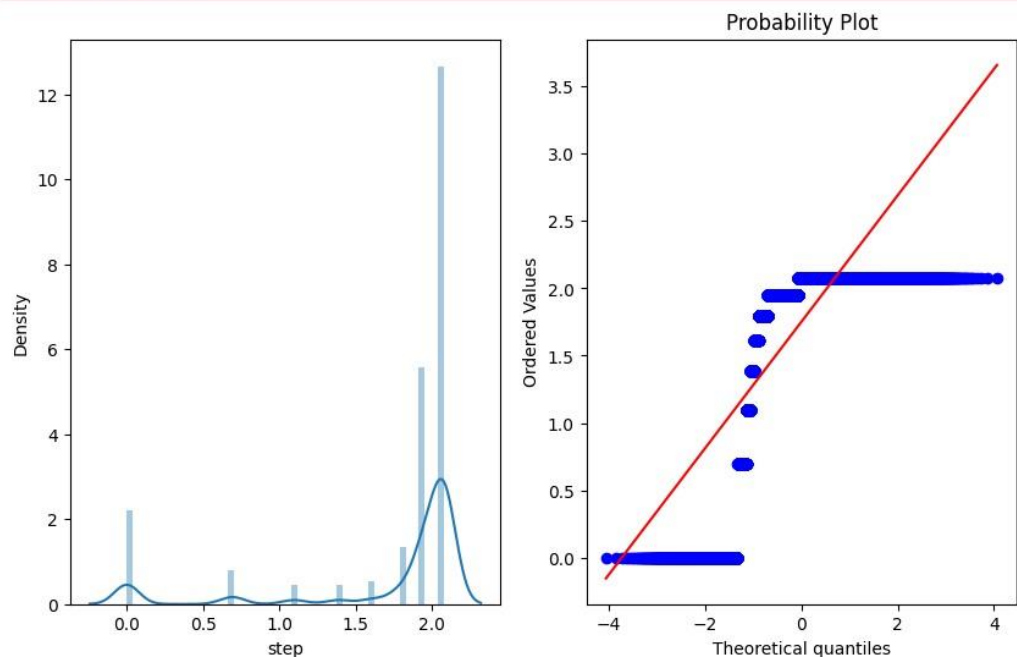
/tmp/ipykernel_14219/479904814.py:4: UserWarning:

'distplot' is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either 'displot' (a figure-level function with similar flexibility) or 'histplot' (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(feature)
```



ix. LabelEncoding

```
In [42]: # Label Encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['type']=le.fit_transform(df['type'])
```

Activity 4: Splitting data into train and test

Now let's split the Dataset into train and test sets. Changes: 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.

```
In [43]: # We will use the column isFraud as our target variable and we are Scaling the data

from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
y=df['isFraud']
X=df.drop('isFraud',axis=1)
X_scaled=pd.DataFrame(scale.fit_transform(X),columns=X.columns)
X_scaled.head()
```

```
Out[43]:
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
0	0.0	0.75	0.009399	0.020371	0.018889	0.000000	0.0
1	0.0	0.75	0.001779	0.002544	0.002284	0.000000	0.0
2	0.0	1.00	0.000171	0.000022	0.000000	0.000000	0.0
3	0.0	0.25	0.000171	0.000022	0.000000	0.00074	0.0
4	0.0	0.75	0.011145	0.004975	0.003522	0.000000	0.0

```
In [44]: # Splitting the data in training and testing
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.2,random_state=42)
```

```
In [45]: # The given dataset is highly Imbalanced so we need to balance it
from imblearn.over_sampling import SMOTE
y_train.value_counts()
sm=SMOTE(random_state=42)
x_train_smote,y_train_smote=sm.fit_resample(X_train,y_train)
print("The values after scaling are")
y_train_smote.value_counts()
```

The values after scaling are

```
Out[45]: isFraud
0      22862
1      22862
Name: count, dtype: int64
```

8. PERFORMANCE TESTING

8.1 Performace Metrics

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.

- x. **Logistic Regression:** This is a very intuitive algorithm and is often implemented for simple categorical problems, such as this one. Given a dataset, a linear function is constructed. Any test cases are given to the function as input — the output of which is a measure of confidence that the testcase is positive.

```
[In [46]:] # Linear Regression model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
lr=LogisticRegression()
lr.fit(x_train_smote,y_train_smote)
y_pred=lr.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,lr.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))
```

```
Training Score 0.8445455340740092
Testing Accuracy 0.8216404886561954
```

```
[In [47]:] confusion_matrix(y_test,y_pred)
```

```
Out[47]: array([[4698, 1016],
               [  6,   10]])
```

```
[In [48]:] pd.crosstab(y_test,y_pred)
```

```
Out[48]:
```

col_0	0	1
isFraud		
0	4698	1016
1	6	10

```
[In [49]:] print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.82	0.90	5714
1	0.01	0.62	0.02	16
accuracy			0.82	5730
macro avg	0.50	0.72	0.46	5730
weighted avg	1.00	0.82	0.90	5730

xi.

Decision Tree Classifier

A decision tree is a decision-making tool that employs a tree-like model of decisions and their potential outcomes, such as chance event outcomes, resource costs, and utility. It is one method of displaying an algorithm that consists solely of conditional control statements. Decision trees are a prominent method in machine learning and are often used in operations research, notably in decision analysis, to assist determine the approach most likely to achieve a goal.

```
In [50]: from sklearn.tree import DecisionTreeClassifier
dc = DecisionTreeClassifier(max_depth=3,splitter='best',criterion='entropy')
dc.fit(x_train_smote,y_train_smote)
```

```
Out[50]: DecisionTreeClassifier(criterion='entropy', max_depth=3)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

```
In [51]: y_pred=dc.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,dc.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))
# Plotting, showing the importance of each feature
```

```
Training Score 0.9462426734318957
Testing Accuracy 0.9738219895287958
```

```
In [52]: import pickle
pickle.dump(dc,open('model_dc.pkl','wb'))
```

```
In [53]: pd.crosstab(y_test,y_pred)
```

```
Out[53]:
```

	col_0	0	1
isFraud			
0	5571	143	
1	7	9	

```
In [54]: print(classification_report(y_test,y_pred))
```

```

              precision    recall  f1-score   support

     0               1.00      0.97      0.99         5714
     1               0.06      0.56      0.11           16

 accuracy               0.97         5730
  macro avg              0.53      0.77      0.55         5730
 weighted avg             1.00      0.97      0.98         5730
```

xii.

Random Forest Classifier

Random forest is an ensemble approach, because of its versatility and simplicity, it is one of the most often used algorithms. This model employs a large number of decision trees. Each of these decision trees separates a class of predictions, and the class with the most votes becomes our model's final output prediction. While growing trees in a random forest, rather than looking for the most significant characteristics for splitting, it seeks for the best features from a random selection of features for splitting the nodes. This results in a wide range of variety, which will provide us with a more accurate model. Because there is no association between the many models developed, the models provide ensemble forecasts that are more accurate than any of the individual projections. This is due to the fact that although certain trees may be incorrect.

Random Forest

```
In [55]: from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(criterion='entropy', max_depth=5,
                           min_samples_leaf=10, min_samples_split=10,
                           n_estimators=50)
rfc.fit(x_train_smote, y_train_smote)
```

```
Out[55]: RandomForestClassifier(criterion='entropy', max_depth=5, min_samples_leaf=10,
                               min_samples_split=10, n_estimators=50)
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
In [56]: y_pred=rfc.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,rfc.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))
print(X_test.shape)
```

```
Training Score 0.988955471962208
Testing Accuracy 0.9778359511343805
(5730, 7)
```

```
In [57]: pd.crosstab(y_test,y_pred)
```

```
Out[57]: col_0    0    1
isFraud
0    5593  121
1         6   10
```

```
In [58]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	5714
1	0.08	0.62	0.14	16
accuracy			0.98	5730
macro avg	0.54	0.80	0.56	5730
weighted avg	1.00	0.98	0.99	5730

xiii.

Extra Tree 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.

Extra Tree Classifier

```
In [59]: from sklearn.ensemble import ExtraTreesClassifier
etc=ExtraTreesClassifier()
etc.fit(x_train_smote,y_train_smote)
y_pred=etc.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,etc.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))

Training Score 1.0
Testing Accuracy 0.9975567190226876

In [60]: pd.crosstab(y_test,y_pred)

Out[60]:
```

col_0	0	1
isFraud		
0	5707	7
1	7	9

```


In [61]: print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

     0       1.00        1.00        1.00        5714
     1       0.56        0.56        0.56         16

 accuracy          0.78        0.78        1.00        5730
 macro avg         0.78        0.78        0.78        5730
 weighted avg      1.00        1.00        1.00        5730

In [81]: etc.predict([[0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00000000e+00,0.00000000e+00, 0.00000000e+00, 1.00000000e+00

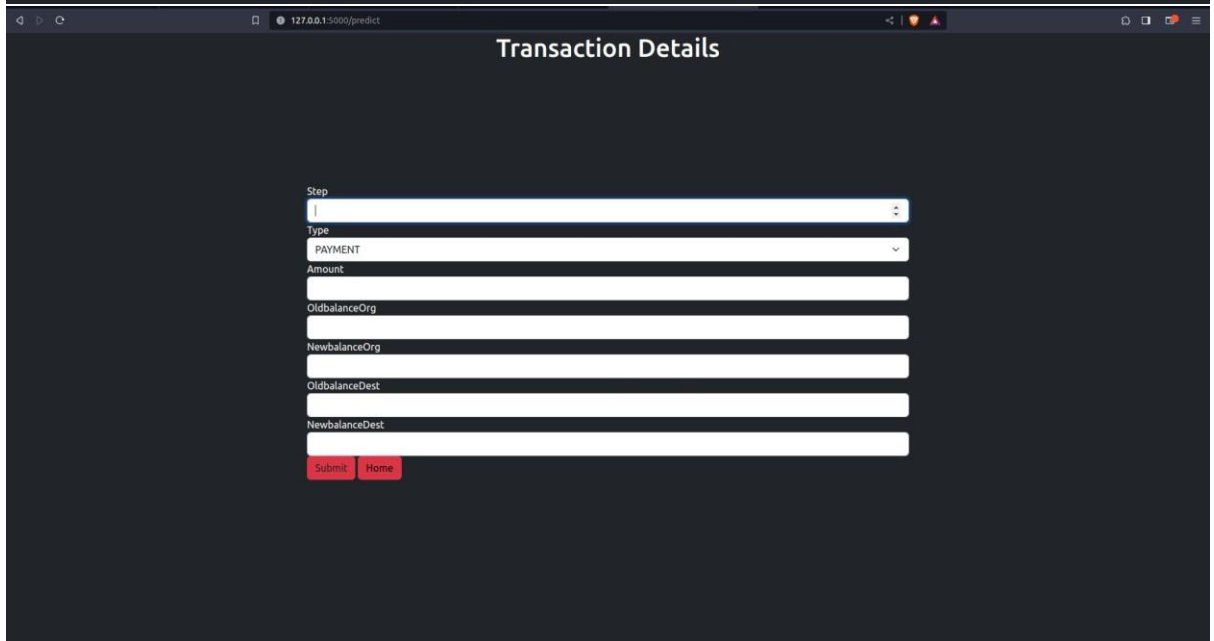
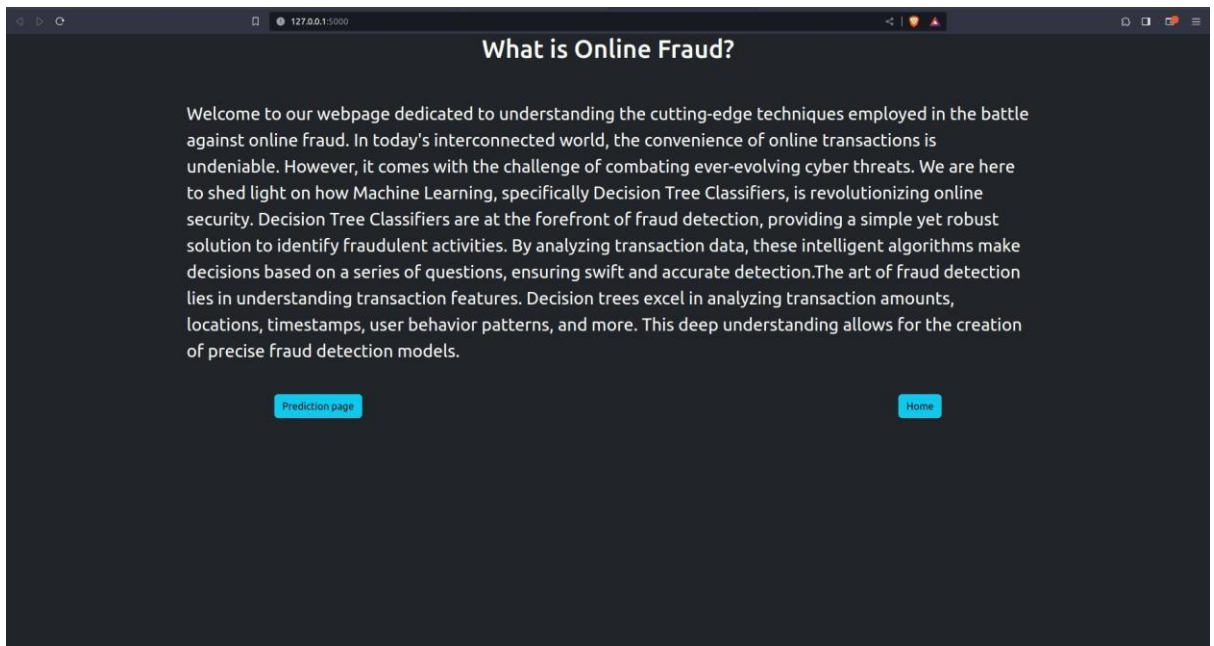
/home/exterminator/train/lib/python3.10/site-packages/sklearn/base.py:465: UserWarning: X does not have valid feature names, but ExtraTreesClassifier was fitted with feature names
  warnings.warn(

Out[81]: array([0])
```

Milestone 5: Application Building

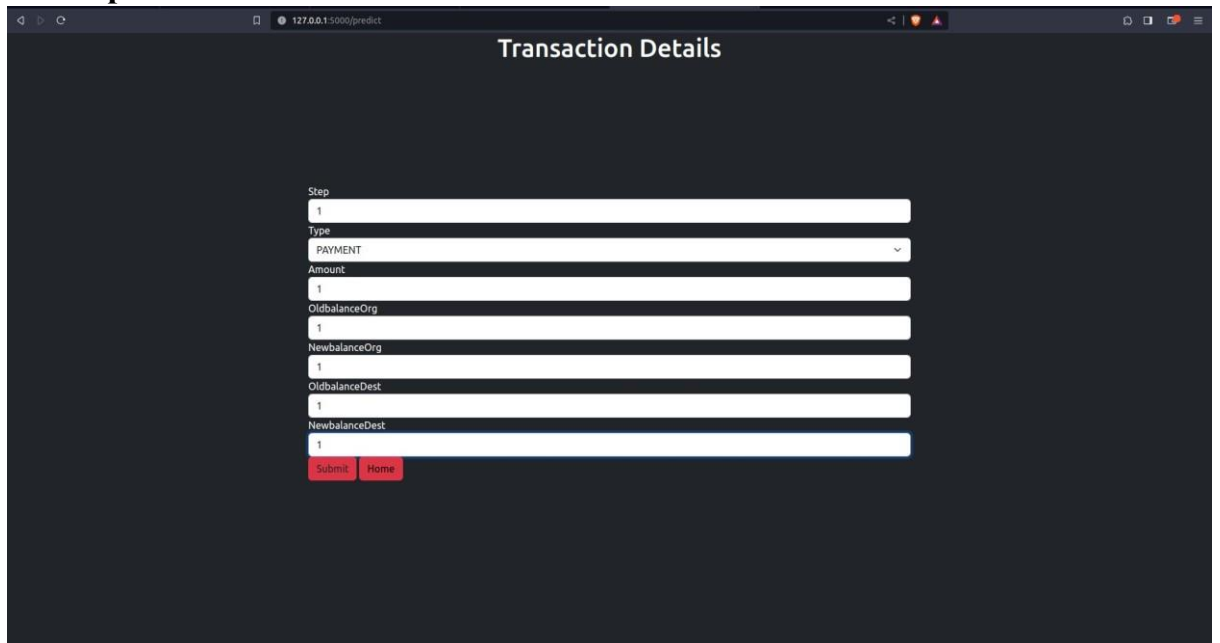
In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI. This section has the following tasks

- Building HTML Pages
- Building server side script



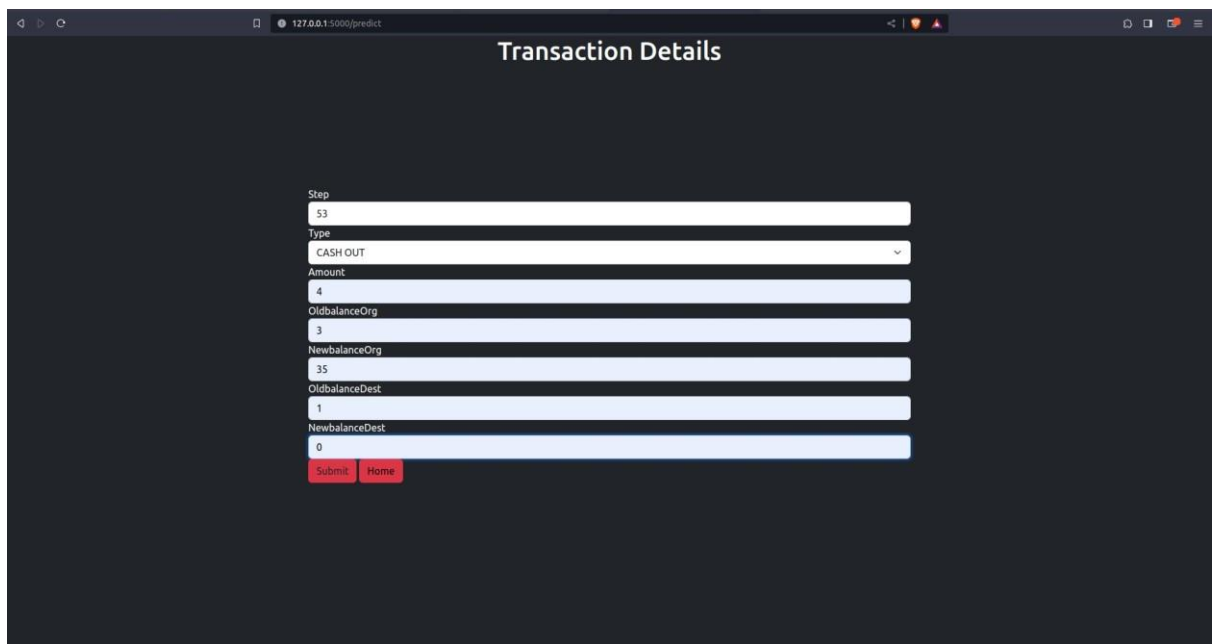
9. RESULTS

9.1 Output Screenshots



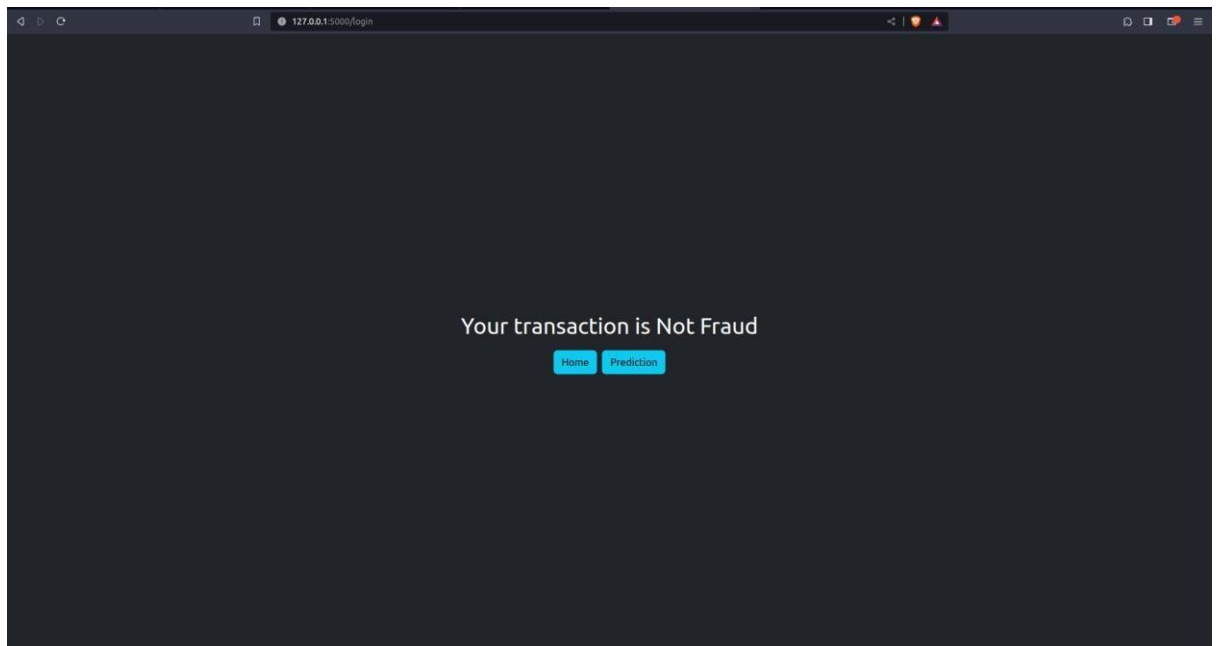
A screenshot of a web browser displaying a form titled "Transaction Details". The browser's address bar shows "127.0.0.1:5000/predict". The form contains several input fields, all of which have the value "1" entered. The fields are labeled: Step, Type (with a dropdown arrow), Amount, OldbalanceOrg, NewbalanceOrg, OldbalanceDest, and NewbalanceDest. At the bottom of the form, there are two red buttons labeled "Submit" and "Home".

Field	Value
Step	1
Type	PAYMENT
Amount	1
OldbalanceOrg	1
NewbalanceOrg	1
OldbalanceDest	1
NewbalanceDest	1



A screenshot of the same "Transaction Details" form, but with different values entered. The browser address bar remains "127.0.0.1:5000/predict". The values in the fields are: Step (53), Type (CASH OUT), Amount (4), OldbalanceOrg (3), NewbalanceOrg (35), OldbalanceDest (1), and NewbalanceDest (0). The "Submit" and "Home" buttons are still present at the bottom.

Field	Value
Step	53
Type	CASH OUT
Amount	4
OldbalanceOrg	3
NewbalanceOrg	35
OldbalanceDest	1
NewbalanceDest	0



10. ADVANTAGES & DISADVANTAGES

Advantages

Here, we use Machine Learning for detecting fraud. Here, a machine tries to learn by itself and becomes better by experience. Also, it is an efficient way of detecting fraud because of its fast computing. It does not even require the guidance of a fraud analyst. It helps in reducing false positives for transactions as the patterns are detected by an automated system for streaming transactions that are in huge volume

Disadvantages

There are some disadvantages in our machine learning model it can't predict real time transition till now and we are not able to connect many transition app till now we have to build a better website and connect to a database

11. CONCLUSION

Machine Learning is a technique used to extract vital information from existing huge amount of data and enable better decision-making for the banking and retail industries. They use data warehousing to combine various data from databases into an acceptable format so that the data can be mined. The data is then analyzed and the information that is captured is used throughout the organization to support decision-making. Data Mining techniques are very useful to the banking sector for better targeting and acquiring new customers, most valuable customer retention, automatic credit approval which is used for fraud prevention, fraud detection in real time, providing segment based products, analysis of the customers, transaction patterns over time for better retention and relationship, risk management and marketing

12. FUTURE SCOPE

Online fraudulent transactions represent a serious criminal offense that costs individuals and financial institutions billions of dollars each year. As a result, financial institutions play a critical role in detecting and preventing fraudulent activities. By adopting effective fraud detection measures, financial institutions can minimize the risks associated with online transactions and protect their customers' financial assets. for accuracy, and the classifier with the highest rating score will be selected as one of the best.

13. Appendix

[Project_Link](#)