ENHANCING ONLINE PAYMENT FRAUD DETECTION

DEEKSHITHA CHIKKALA KEERTHI YALAMANCHILI PRABHITHA VEERAMACHANENI

project-dchikkal-kyalaman-pveeram

ABSTRACT

To begin with, online payments, which bring efficiency and convenience, have become an essential component of everyday life in the contemporary digital era. As the number of electronic transactions grows, protecting the security of online payments becomes more crucial. Using advanced data mining techniques, this initiative aims to improve the effectiveness of online payment fraud detection. The goal is to create a strong system that can detect and prevent fraudulent transactions in real time, hence creating a secure and trustworthy online payment environment.

Our main objective is to efficiently investigate, comprehend, and categorize fraudulent and non-fradulent transactions by utilizing data analytic techniques. This project's purpose is to improve online payment fraud detection by employing a scientific, data-driven methodology. The goal of this project is to do thorough dataset analysis on the "Online payments fraud detection" dataset in order to detect online payment fraud. The collection, which contains historical data on fraudulent transactions, allows us to understand more about the patterns and characteristics of fraudulent conduct.

We conducted an extensive exploratory data analysis (EDA) to understand the dataset's structure, distribution, and anomalies. Following that, we use cutting-edge techniques such as Decision tree, ada-boost and gradient boost to precisely categorize fraudulent and non-fraudulent payments, with an emphasis on model accuracy and performance.

The initiative's purpose is to reduce financial fraud and improve the security of internet payments. This strategy intends to make a significant contribution to the reduction of financial fraud and the security of online payments by providing a comprehensive foundation for achieving these vital goals.

KEY WORDS :

- Decision Tree
- Exploratory Data Analysis
- Visualizations
- Univariate Analysis
- Inter Quartile Range
- Adaptive Boosting
- Gradient Boosting
- Pearson Correlation
- HeatMap
- Searborn Distplot
- Exploratory Box plots
- Outlier Detection
- Skewness Test
- Normality Test
- Histogram
- Whiskers
- User Interactive Fraud Detection

CHAPTER 1
INTRODUCTION

I. OVERVIEW OF ONLINE PAYMENTS :

Businesses have historically faced several increased risks to their operations during periods of economic volatility, such as supply chain interruption, workforce retention, and cyber threats. This has resulted in an increase in fraudulent activity in 2023, as fraudsters deploy advanced tools to take advantage of the situation.

Fraud detection is the process of monitoring transactions and customer behavior in order to identify and combat fraudulent conduct. It is typically a key component of a company's loss prevention strategy and, in certain cases, a component of its overall anti-money laundering (AML) compliance protocols.

Criminals can gain your data, access finances, or steal assets in a variety of methods thanks to the internet. Logging into an unprotected Wi-Fi network is enough to expose your personal information to naive scammers nearby. However, it can also occur in the privacy of your own home or place of business. It is vital to have a method to detect fraud before it occurs in order to avoid being another victim of a cybercrime. For some businesses, this means losing thousands, if not millions, of dollars. Fraud can take place in a variety of ways.

Today's most frequent types are :

- Identity theft : is the theft of a person's private data, such as their birth date, Social Security number, addresses, and phone numbers, in order to qualify for credit.
- Phishing scam : Sending emails disguised as a vendor or government agency with links or attachments carrying malware in order to gain login credentials.
- Malware is software that secretly records your keystrokes in order to steal account usernames and passwords.

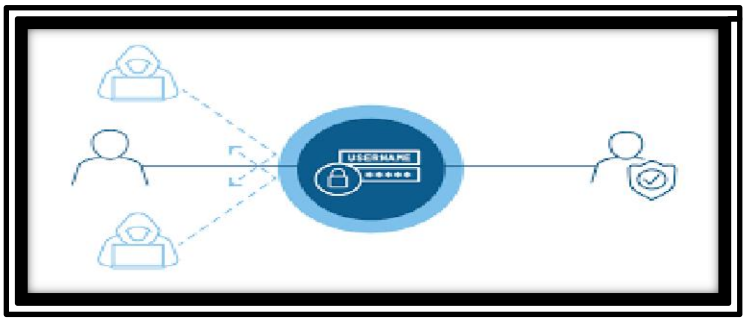


Fig. 1.1 Online Fraud Detection

II. OBJECTIVE :

This project aims to address the issues raised by online payment fraud by constructing a sophisticated and effective fraud detection system. Traditional rule-based solutions, while valuable, frequently struggle to keep up with fraudsters' shifting techniques. As a result, this research makes use of approaches which were taught during the class to improve the accuracy and efficiency of real-time fraud detection.

The fundamental purpose of this study is to create a comprehensive fraud detection model that goes beyond traditional methods. The system intends to detect complex patterns and abnormalities within transaction data by using data mining methods, such as ada boost, gradient boost and to react dynamically to the ever-changing landscape of online fraud. Evaluate the suggested model's performance rigorously using a diverse dataset that includes both genuine and fraudulent transactions.

Conduct comparative analyses with existing fraud detection systems to showcase the superiority of the developed model.

Our major goal is to improve online payment security by employing a meticulous, data- driven approach to fraud detection.

We hope to improve the accuracy and efficiency of detecting fraudulent transactions by employing advanced data analytics and classification techniques.

This program is critical in strengthening the resilience of online payment systems, minimizing risks, and building a safer digital financial ecosystem.

CHAPTER 2
DATASET

The initial stage in our project work was to select the appropriate data set. There are numerous online resources that provide access to a multitude of financial fraud investigation datasets containing transaction information but no personal user information. We discovered several data sets from kaggle, known as "online- payment-fraud-detection".

We obtained this dataset from Kaggle for this job, which contains historical information on fraudulent transactions that can be utilized to detect fraud in online payments.

All of the columns from the dataset we're utilizing are listed below :

a) step : A time unit in which one step equals one hour
b) type : Online transaction type.
c) amount : The amount of the transaction
d) nameorig : Customer starting the transaction
e) oldbalanceorg : Balance prior to the transaction
f) newbalanceorig : Balance following the transaction
g) namedest : The transaction's receiver
h) oldbalancedest : The recipient's starting balance prior to the transaction newbalancedest: The recipient's new balance after the transaction.
i) isfraud : Fraud transaction

The dataset had 1 Million rows. And the dataset is of size 186MB.
We imported different libraries in order to move forward with the code.

We used the following tools in order to implement the project. They are shown as follows:

a) Google colab
b) Python
c) Libraries(numpy, pandas, matplotlib, searborn, sklearn.model_selection, sklearn.tree, sklearn.utils, plotpy.express etc)
d) Jupyter Notebook

CHAPTER 3
EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a type of data analysis that seeks out broad patterns in data. EDA is a data mining methodology of evaluating datasets to summarize their essential properties, generally using visual tools. EDA is used before modeling to see what the data can tell us. It is difficult to detect essential data properties by looking at a column of numbers or an entire spreadsheet. Looking at basic numbers can be time-consuming, uninteresting, and/or daunting. In this case, exploratory data analysis approaches have been developed to help. Outliers and unexpected data features are examples of these patterns. EDA is a critical first step in any data analysis process.

Exploratory data analysis is often divided into two categories. To begin, each method is either non-graphic or graphic. Second, each approach is either univariate or multivariate (also mostly known as bivariate).

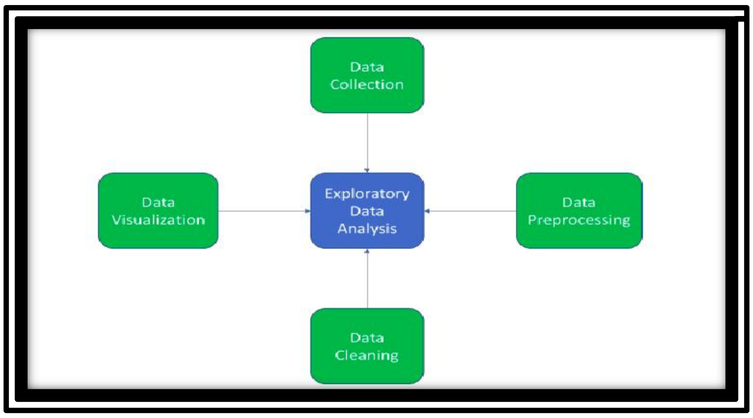
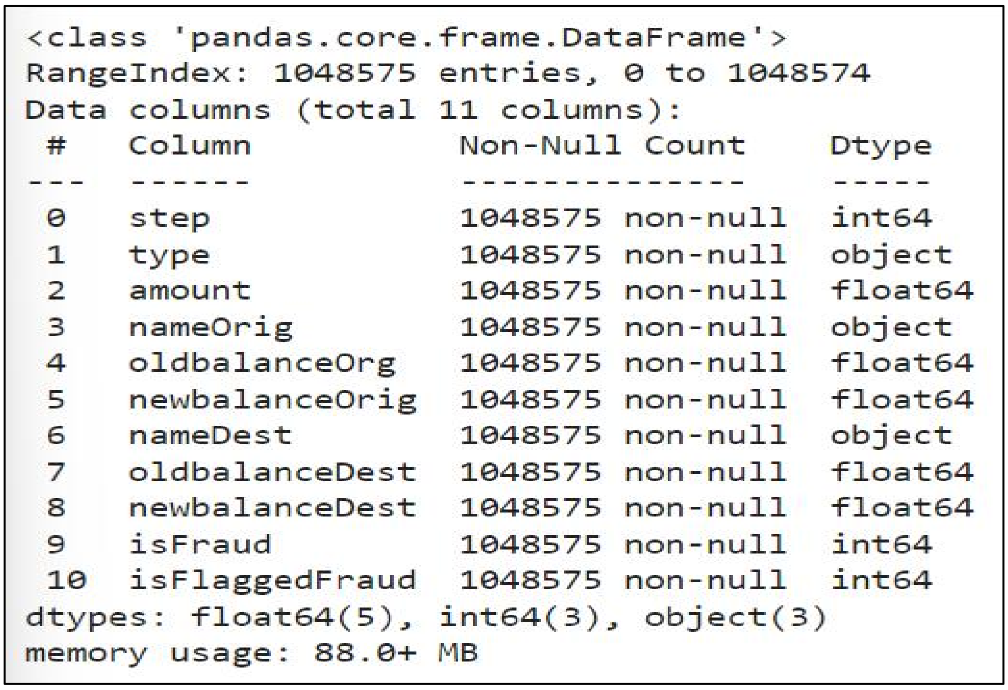


Fig. 3.1 Exploratory Data Analysis

In this Exploratory Data Analysis (EDA), categorical variables like transaction types and fraud labels are numerically encoded for analysis. Visualizations, including a pie chart for transaction type distribution and bar plots for type and fraud labels, offer insights into data characteristics. The Pearson correlation heatmap illuminates relationships between features, aiding in understanding potential patterns and informing subsequent steps in fraud detection.

I. EDA ON DATASET COLUMNS :

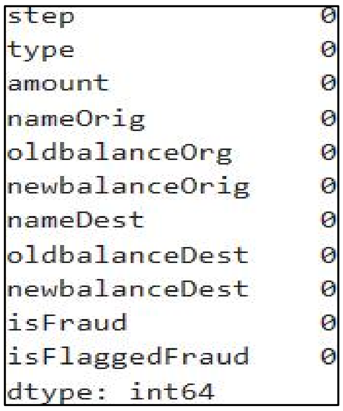
When observed the columns, we found the datatype of each column , calculated the non-null values for each column, found the how many data types are present with their count and finally found the memory usage of the whole dataset.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   step            1048575 non-null  int64
 1   type            1048575 non-null  object
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  int64
 10  isFlaggedFraud  1048575 non-null  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 88.0+ MB
```

Fig. 3.2 EDA on Dataset columns

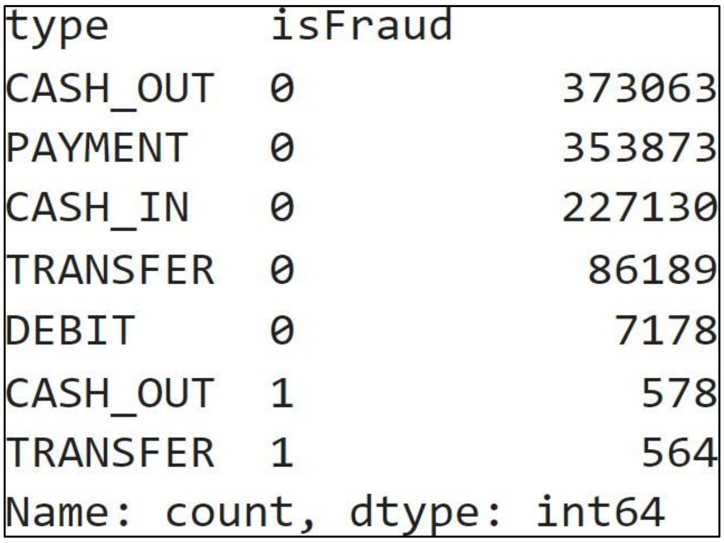
It is used to count the number of missing (null or NaN) values in each column of the DataFrame **opfd**. The result will be a Series containing a count of null values for each column in opfd. This data is useful for determining the completeness of our dataset and how to handle missing values during data preprocessing.



```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

Fig. 3.3 EDA on Dataset columns

The following output appears to count the number of distinct combinations of values in the DataFrame opfd's 'type' and 'isFraud' columns. It chooses a subset of the DataFrame opfd that only contains the 'type' and 'isFraud' columns. The function is then executed to the subset of interest, returning a Series holding the counts of unique combinations of values in the columns supplied. Where “0” appears to be not fraud and “1” value appears to be fraud.



```
type      isFraud
CASH_OUT  0          373063
PAYMENT   0          353873
CASH_IN   0          227130
TRANSFER  0           86189
DEBIT     0            7178
CASH_OUT  1             578
TRANSFER  1             564
Name: count, dtype: int64
```

Fig. 3.4 Fraud in Each column

The snapshot below leverages the Plotly library to generate a pie chart depicting the distribution of transaction types in a DataFrame (opfd). It depicts the distribution of transaction types in the DataFrame, with each transaction type shown as a slice in the pie chart and the colors distinguishing them visually. The proportion of each transaction type in the dataset is represented by the size of each slice.

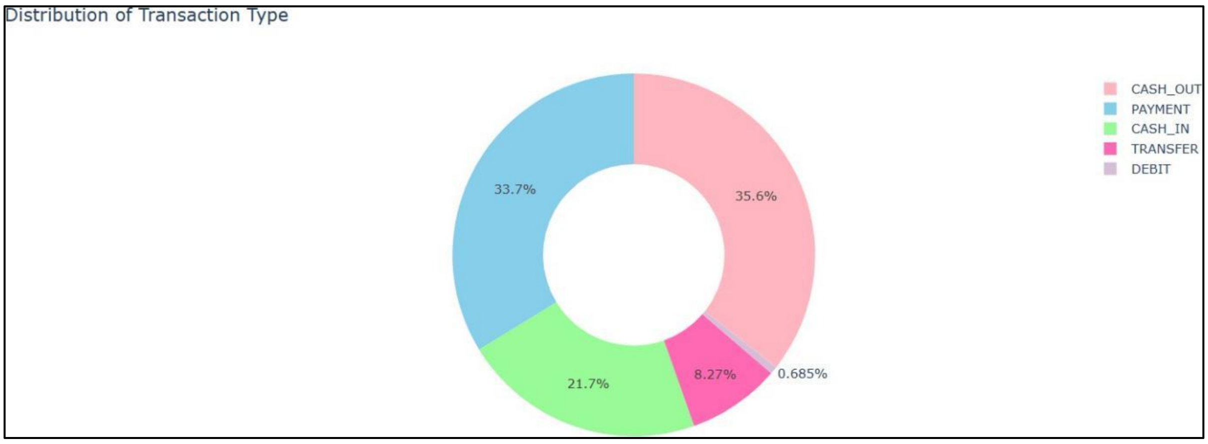


Fig. 3.5 Distribution of Transaction Type

The graphic below depicts the occurrences of each unique value in the 'type' column. The bar chart's x-axis normally indicates the unique values of the 'type' column, and the height of each bar represents the count of occurrences for each kind.
In summary, this code offers a textual summary (by the printed value counts) as well as a visual depiction (via the bar chart) of the distribution of transaction types in the DataFrame's 'type' column. The bar chart can help you rapidly grasp the relative frequency of each transaction type.

```
<Axes: xlabel='type'>
```

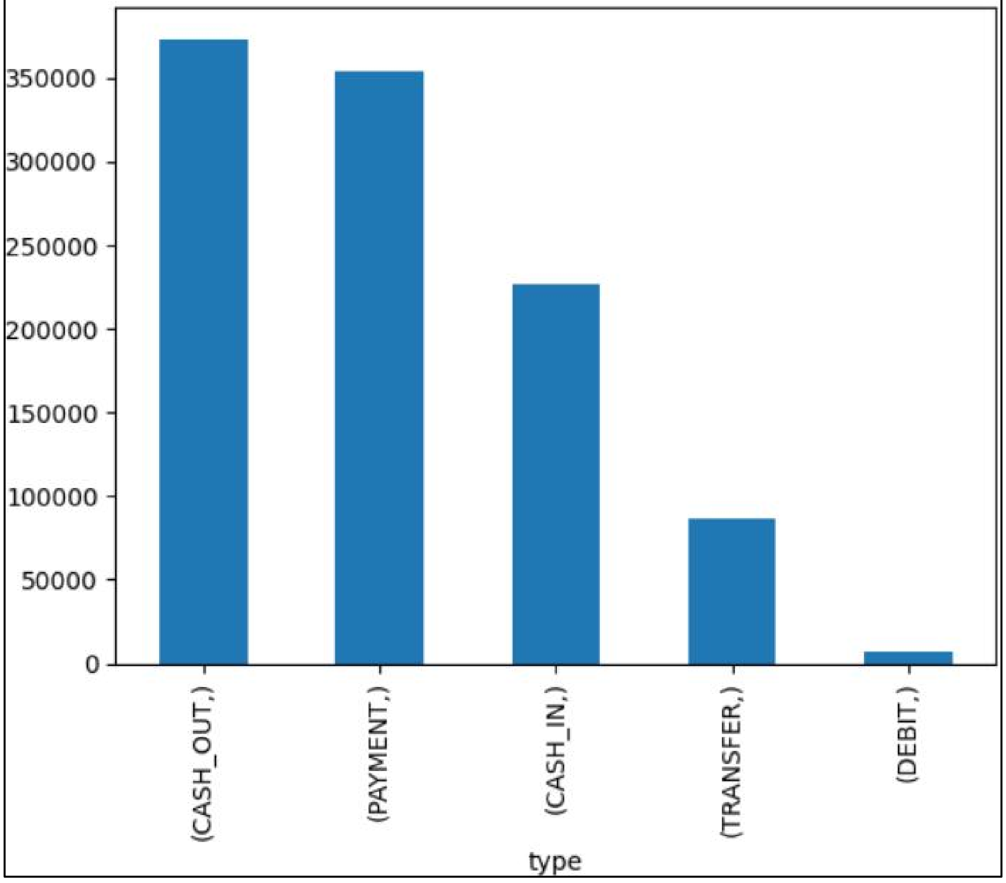


Fig. 3.6 Bar plot of Transaction Types

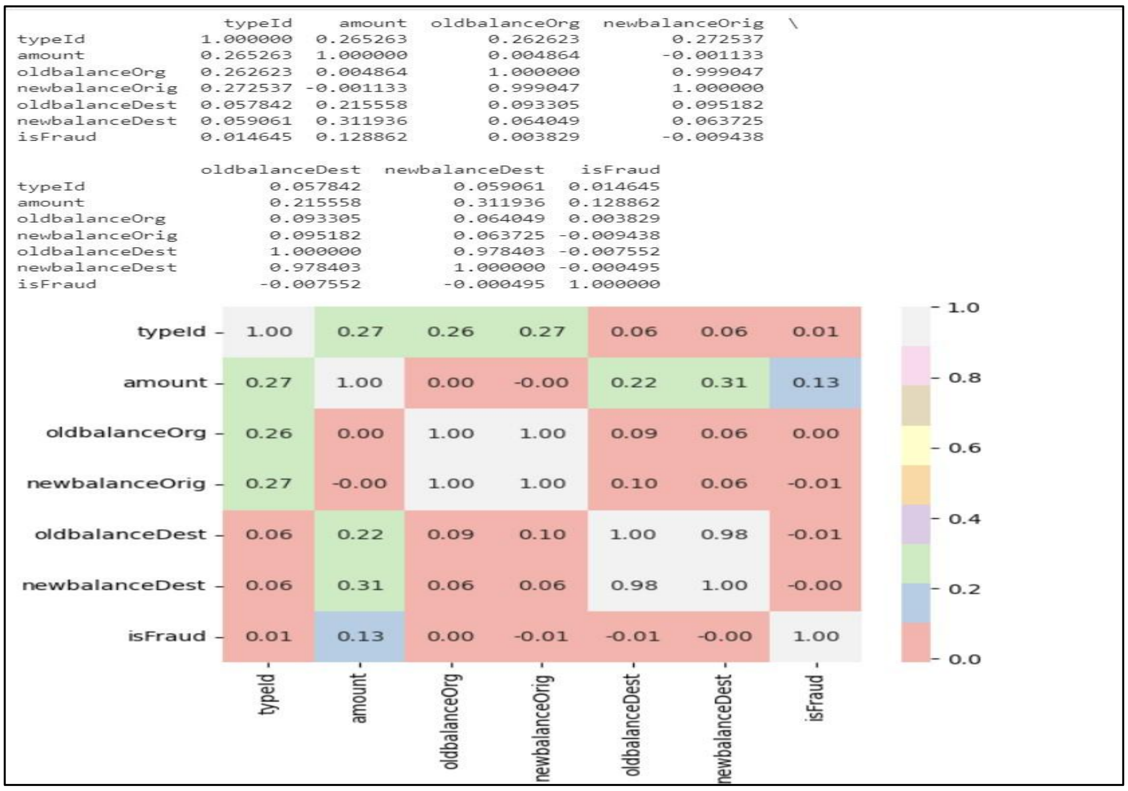
II. METHODS :

a) Correlation and HeatMap -

The Python function does Pearson correlation analysis on a subset of the DataFrame's columns (opfd) and generates a heatmap to display the correlation coefficients between these columns. In summary, this code does correlation analysis on all the columns of the DataFrame, computes Pearson correlation coefficients, and then generates a heatmap to visualize the strength and direction of these relationships. The heatmap is an effective tool for discovering correlations between variables in a dataset.
To further understand the fraud detection we are now exploring pearson correlation and heatmap. Pearson correlation provides a quantative measure of linear relationships. The key variables selected for analysis are all the dataset columns.

When we observe the heatmap, it is either in positive or negative linear relationship.
It is said to be +ve corr when: variable at x-axis and it's corresponding value at y-axis increases and

corr coeff is closer to +1
It is said to be -ve corr when: variable at x-axis and it's corresponding value at y-axis decreases and corr coeff is closer to -1
All the white columns represent all the values in +ve correlation and the rest represent -ve correlation.



```
                   typeId    amount  oldbalanceOrg  newbalanceOrig  \
typeId           1.000000  0.265263       0.262623        0.272537
amount           0.265263  1.000000       0.004864       -0.001133
oldbalanceOrg    0.262623  0.004864       1.000000        0.999047
newbalanceOrig   0.272537 -0.001133       0.999047        1.000000
oldbalanceDest   0.057842  0.215558       0.093305        0.095182
newbalanceDest   0.059061  0.311936       0.064049        0.063725
isFraud          0.014645  0.128862       0.003829       -0.009438

                 oldbalanceDest  newbalanceDest   isFraud
typeId                 0.057842        0.059061  0.014645
amount                 0.215558        0.311936  0.128862
oldbalanceOrg          0.093305        0.064049  0.003829
newbalanceOrig         0.095182        0.063725 -0.009438
oldbalanceDest         1.000000        0.978403 -0.007552
newbalanceDest         0.978403        1.000000 -0.000495
isFraud               -0.007552       -0.000495  1.000000
```

Fig. 3.7 Correlation and heatmap for all the columns

b) Searborn Distplot -

We are now moving into seaborn distplot, it is useful for visualizing the frequency distribution of a continuous variable and understanding its shape and characteristics.
The primary component of a distplot is a histogram, we are representing this data by dividing it into bins. In the below we divided diagram into 50 bins.
The x-axis represents the range of values in the 'step' coloumn, and the y-axis represents the density of observations in each bin.
The smooth line represented in the graph is known as kernel density estimate, it helps to visualize the data, by providing a smoother representation compared to the raw histogram.
It generates a new 15x6-inch figure and then draws a distribution plot (distplot) for the'step' column from the DataFrame opfd. The distplot is a seaborn function that combines a histogram with an estimate of kernel density. Bins=50 defines the number of bins in the histogram, and color="pink" changes the plot's color to pink.

In summary, this code snippet performs multiple operations, including estimating the amount of loss due to fraudulent transactions, producing value counts for specific columns, and utilizing a distplot to visualize the distribution of the'step' column. The seaborn library is utilized for creating the distribution plot.

```
The amount of loss due to fraud transaction:
<Axes: xlabel='step', ylabel='Density'>
```

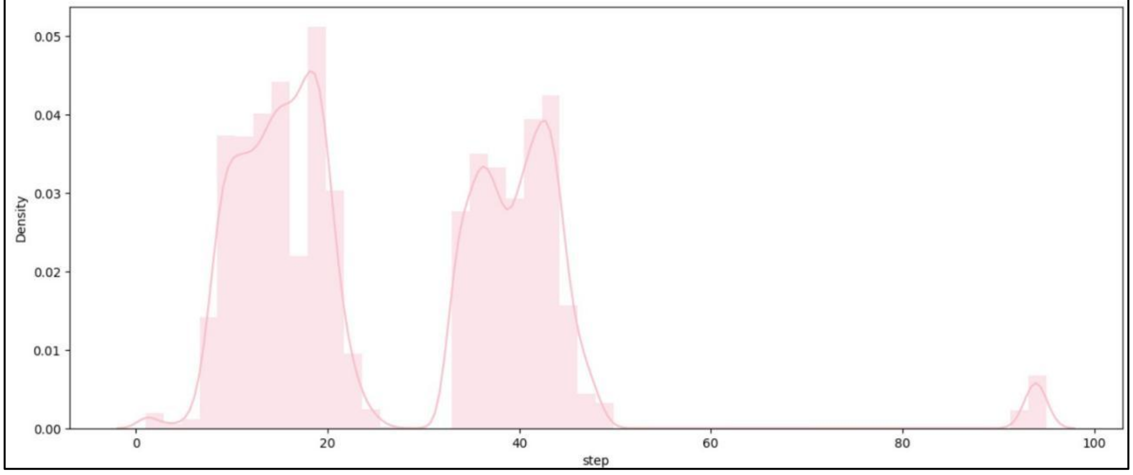


Fig. 3.8 Searborn Distplot

c) Box Plot Exploratory Analysis -

Box plot exploratory analysis, we created box plot for the relationship between the target variable and each numerical feature. The box plot is displayed in the top row of subplots.
we created box plot for the relationship between the categorical variable (transaction type) and each numerical feature. The box plot is displayed in the bottom row of subplots.
Box in the middle represents the interquartile range (IQR), which is the range between the 25th percentile (Q1) and the 75th percentile (Q3) of the data.
The length of the box illustrates the spread of the middle 50% of the data.
The line inside the box represents the median (50th percentile) of the data.It indicates the central tendency of the dataset.
Individual points beyond the whiskers are considered outliers.

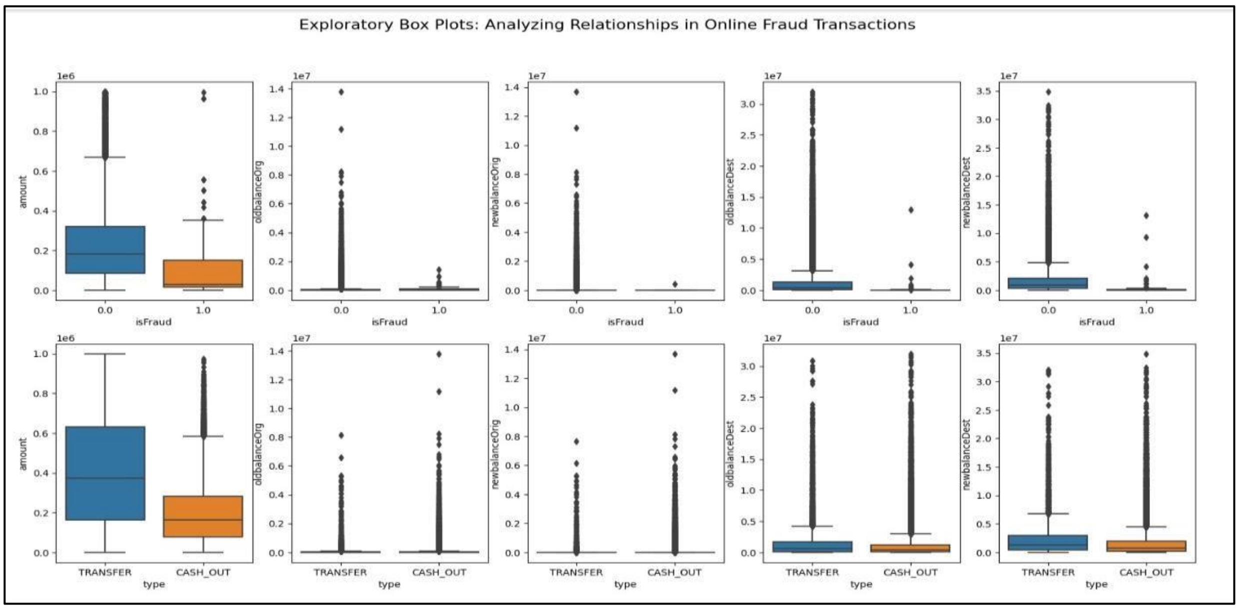


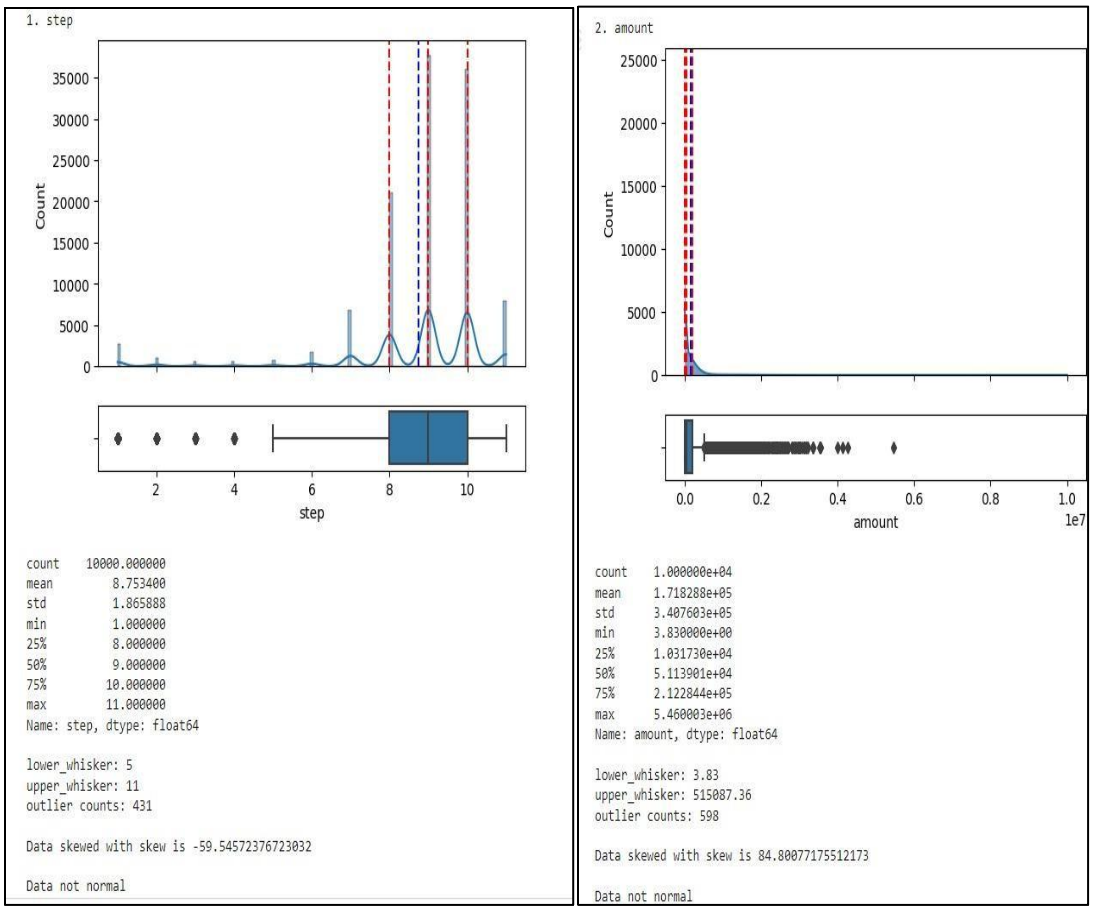
Fig. 3.9 Exploratory Box Plots

d) Univariate Analysis -

Univariate analysis is a fundamental component of exploratory data analysis, providing valuable insights into the characteristics of individual variables within a dataset.
Box in the middle represents the interquartile range (IQR), which is the range between the 25th percentile (Q1) and the 75th percentile (Q3) of the data.
The length of the box illustrates the spread of the middle 50% of the data.The line inside the box represents the median (50th percentile) of the data. It indicates the central tendency of the dataset.
Individual points beyond the whiskers are considered outliers.
In the context of our project, univariate analysis serves as the initial step to comprehend the distribution, central tendency, and variability of each feature independently.

The code evaluates numerical variables in several of ways :

- Histogram : a graphic representation of the distribution of data.
- Boxplot : provides a summary of the distribution of data, including percentiles and any outliers.
- Loop of univariate analysis : The code then iterates through each numerical variable (opfd2_num). It generates a subplot with two axes for each variable, one for a histogram with percentiles and another for a boxplot. Descriptive statistics tests provide a numerical summary of the distribution's central tendency, variability, and form.
- Outlier detection : identifies extreme values that deviate from the normal distribution.
- Skewness test : determines whether the distribution of data is symmetric or skewed.
- The Normality test : determines if the data has a normal distribution.
- Univariate analysis, provides the characteristics of individual variables within the dataset,It calculates the lower and upper fences using the interquartile range (IQR) method

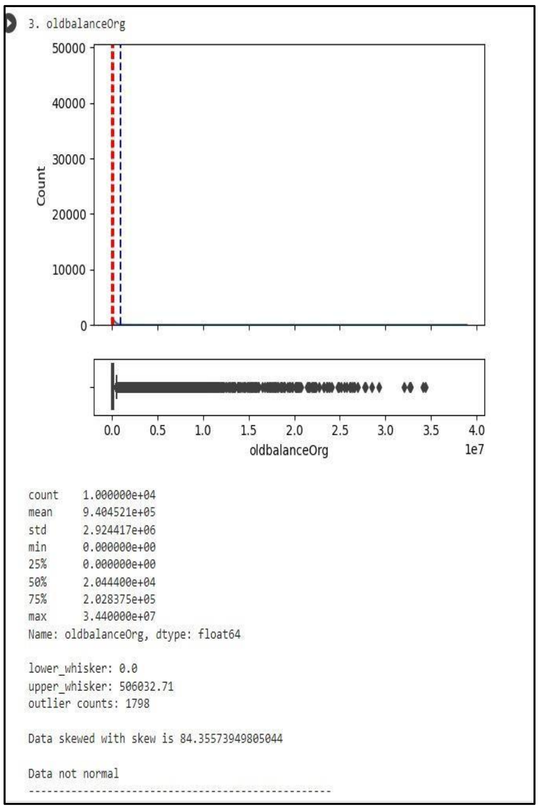


```
count    10000.000000
mean         8.753400
std          1.865888
min          1.000000
25%          8.000000
50%          9.000000
75%         10.000000
max         11.000000
Name: step, dtype: float64

lower_whisker: 5
upper_whisker: 11
outlier counts: 431

Data skewed with skew is -59.54572376723032

Data not normal
```

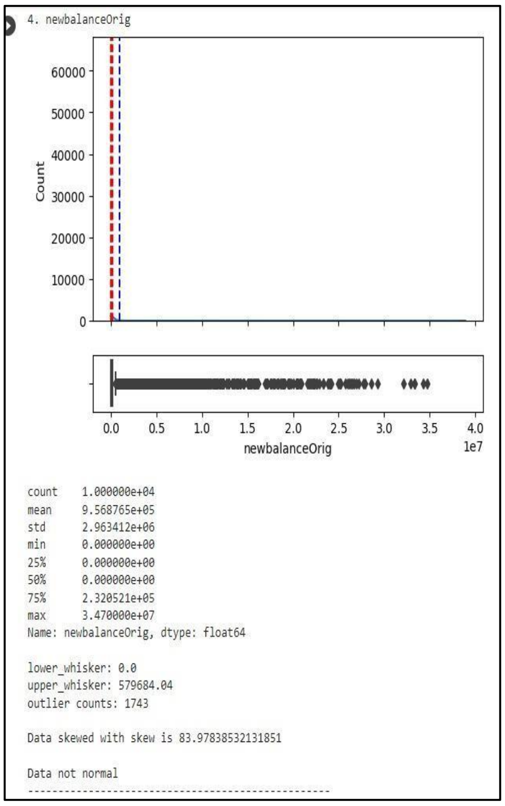
```
count    1.000000e+04
mean     1.718288e+05
std      3.407603e+05
min      3.830000e+00
25%      1.031730e+04
50%      5.113901e+04
75%      2.122844e+05
max      5.460003e+06
Name: amount, dtype: float64

lower_whisker: 3.83
upper_whisker: 515087.36
outlier counts: 598

Data skewed with skew is 84.80077175512173

Data not normal
```

Fig. 3.10 (a)(b) Univariate Analysis Plots

For each variable, it creates a subplot with two axes: one for a histogram with percentiles and another for a boxplot.
The histogram includes red dashed lines at the 25th, 50th (median), and 75th percentiles, as well as a blue dashed line at the mean.



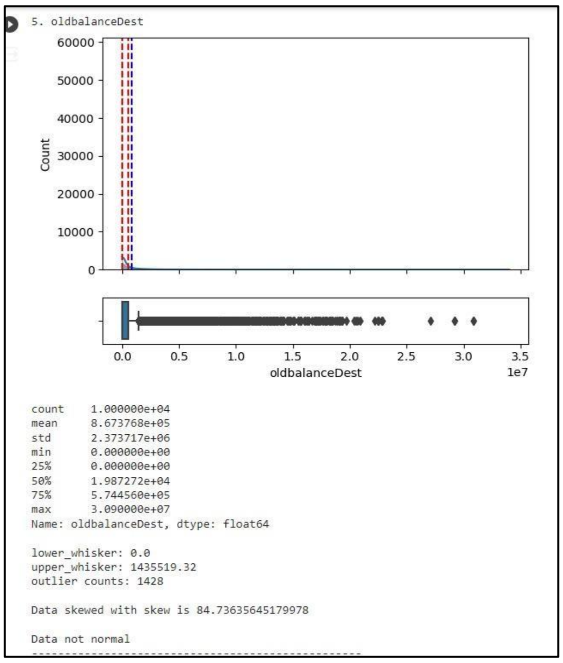
```
count    1.000000e+04
mean     9.404521e+05
std      2.924417e+06
min      0.000000e+00
25%      0.000000e+00
50%      2.044400e+04
75%      2.028375e+05
max      3.440000e+07
Name: oldbalanceOrg, dtype: float64

lower_whisker: 0.0
upper_whisker: 506032.71
outlier counts: 1798

Data skewed with skew is 84.35573949805044

Data not normal
--------------------------------------------------
```



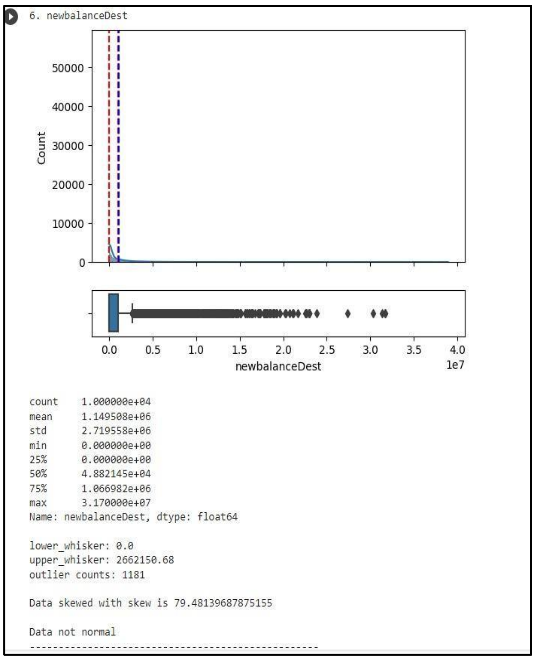
```
count    1.000000e+04
mean     9.568765e+05
std      2.963412e+06
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      2.320521e+05
max      3.470000e+07
Name: newbalanceOrig, dtype: float64

lower_whisker: 0.0
upper_whisker: 579684.04
outlier counts: 1743

Data skewed with skew is 83.97838532131851

Data not normal
--------------------------------------------------
```

Fig. 3.10 (c)(d) Univariate Analysis Plots

A boxplot is also displayed for a visual representation of the data distribution.



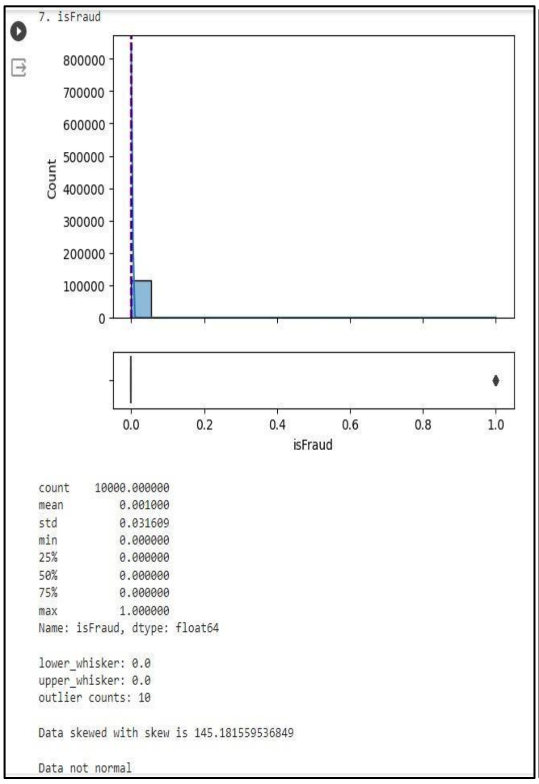
```
count    1.000000e+04
mean     8.673768e+05
std      2.373717e+06
min      0.000000e+00
25%      0.000000e+00
50%      1.987272e+04
75%      5.744560e+05
max      3.090000e+07
Name: oldbalanceDest, dtype: float64

lower_whisker: 0.0
upper_whisker: 1435519.32
outlier counts: 1428

Data skewed with skew is 84.73635645179978

Data not normal
--------------------------------------------------
```



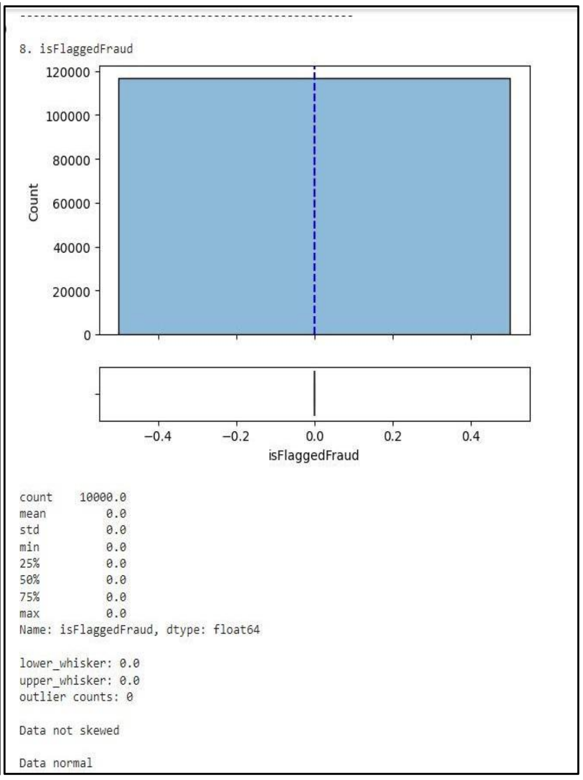
```
count    1.000000e+04
mean     1.149508e+06
std      2.719558e+06
min      0.000000e+00
25%      0.000000e+00
50%      4.882145e+04
75%      1.066982e+06
max      3.170000e+07
Name: newbalanceDest, dtype: float64

lower_whisker: 0.0
upper_whisker: 2662150.68
outlier counts: 1181

Data skewed with skew is 79.48139687875155

Data not normal
--------------------------------------------------
```

Fig. 3.10 (e)(f) Univariate Analysis Plots

Skewness is calculated if the p-value is less than 0.05, it indicates that the data is skewed. Normality is calculated if the p-value is less than 0.05, it indicates that the data is not normal.



```
7. isFraud

count    10000.000000
mean         0.001000
std          0.031609
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          1.000000
Name: isFraud, dtype: float64

lower_whisker: 0.0
upper_whisker: 0.0
outlier counts: 10

Data skewed with skew is 145.181559536849

Data not normal
```



```
---------------------------------------------------

8. isFlaggedFraud

count    10000.0
mean         0.0
std          0.0
min          0.0
25%          0.0
50%          0.0
75%          0.0
max          0.0
Name: isFlaggedFraud, dtype: float64

lower_whisker: 0.0
upper_whisker: 0.0
outlier counts: 0

Data not skewed

Data normal
```

Fig. 3.10 (g)(h) Univariate Analysis Plots

CHAPTER 4
PROPOSED METHOD AND RESULTS

I. USER INTERACTIVE FRAUD DETECTION USING DECISION TREE

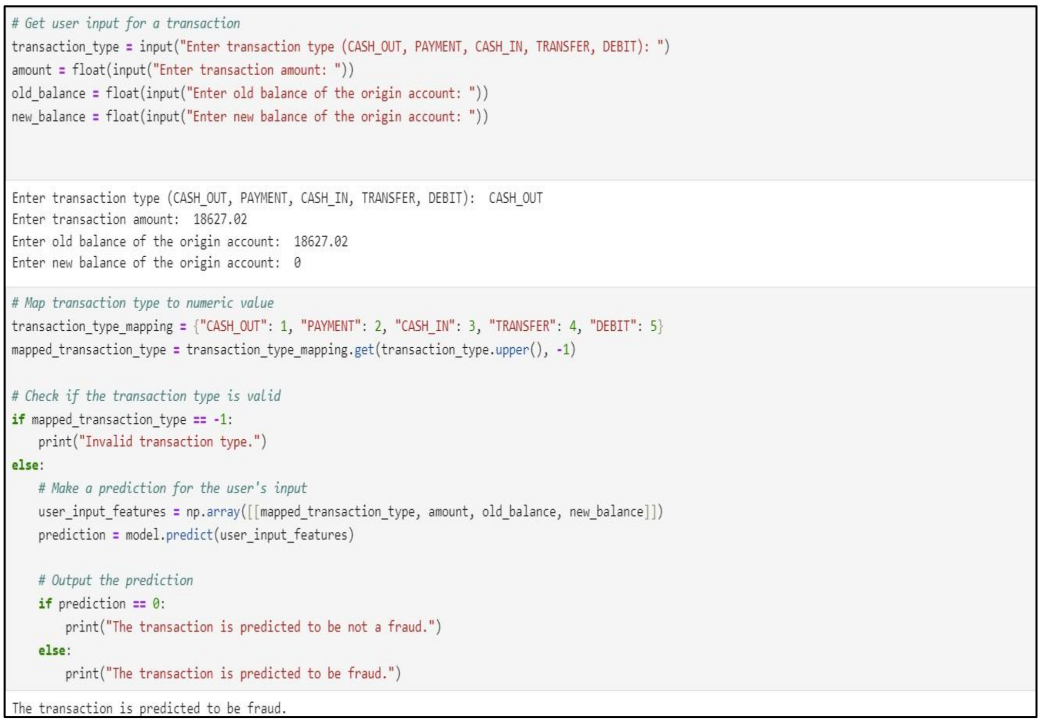
we move on to feature engineering where Features used for training the model include 'type', 'amount', 'oldbalanceOrg', and 'newbalanceOrig.' These capture relevant information about the transaction.

The dataset is split into training and testing sets. A Decision Tree Classifier is employed to learn patterns in the features that distinguish between fraud and non-fraud transactions.

We used a decision tree classifier here as it's transparent structure aids in explaining decisions, making it suitable for a user-interactive system, while its versatility accommodates mixed data types commonly found in fraud detection datasets.

Users provide input for a transaction, including type, amount, old balance, and new balance. The input is transformed into a numpy array, and the model is used to predict whether the transaction is fraudulent based on mapping the transaction type to the numerical values.

The code then outputs whether the transaction is predicted to be fraud or not based on the Decision Tree model's prediction.
Here we can see the examples when we tested out with different transactions where one kind of transaction was found to be fraud and other to be not a fraud.



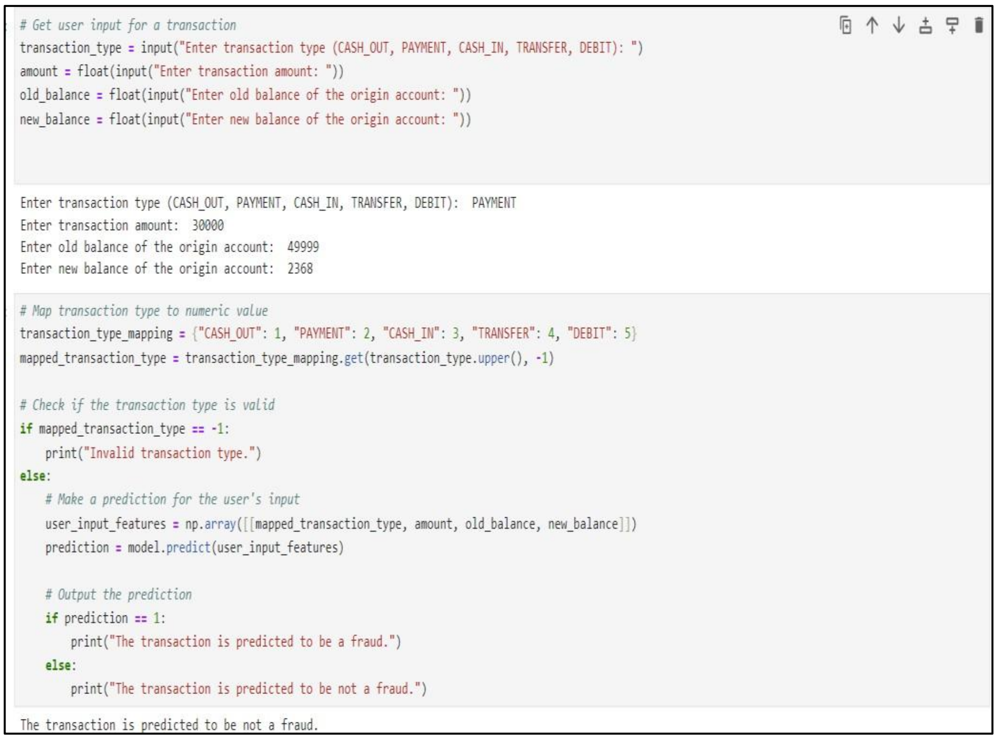
```
# Get user input for a transaction
transaction_type = input("Enter transaction type (CASH_OUT, PAYMENT, CASH_IN, TRANSFER, DEBIT): ")
amount = float(input("Enter transaction amount: "))
old_balance = float(input("Enter old balance of the origin account: "))
new_balance = float(input("Enter new balance of the origin account: "))
```

```
Enter transaction type (CASH_OUT, PAYMENT, CASH_IN, TRANSFER, DEBIT):  CASH_OUT
Enter transaction amount:  18627.02
Enter old balance of the origin account:  18627.02
Enter new balance of the origin account:  0
```

```
# Map transaction type to numeric value
transaction_type_mapping = {"CASH_OUT": 1, "PAYMENT": 2, "CASH_IN": 3, "TRANSFER": 4, "DEBIT": 5}
mapped_transaction_type = transaction_type_mapping.get(transaction_type.upper(), -1)

# Check if the transaction type is valid
if mapped_transaction_type == -1:
    print("Invalid transaction type.")
else:
    # Make a prediction for the user's input
    user_input_features = np.array([[mapped_transaction_type, amount, old_balance, new_balance]])
    prediction = model.predict(user_input_features)

    # Output the prediction
    if prediction == 0:
        print("The transaction is predicted to be not a fraud.")
    else:
        print("The transaction is predicted to be fraud.")
```

```
The transaction is predicted to be fraud.
```

Fig. 4.1 Transaction is detected to be Fraud



```
# Get user input for a transaction
transaction_type = input("Enter transaction type (CASH_OUT, PAYMENT, CASH_IN, TRANSFER, DEBIT): ")
amount = float(input("Enter transaction amount: "))
old_balance = float(input("Enter old balance of the origin account: "))
new_balance = float(input("Enter new balance of the origin account: "))
```

```
Enter transaction type (CASH_OUT, PAYMENT, CASH_IN, TRANSFER, DEBIT):  PAYMENT
Enter transaction amount:  30000
Enter old balance of the origin account:  49999
Enter new balance of the origin account:  2368
```

```
# Map transaction type to numeric value
transaction_type_mapping = {"CASH_OUT": 1, "PAYMENT": 2, "CASH_IN": 3, "TRANSFER": 4, "DEBIT": 5}
mapped_transaction_type = transaction_type_mapping.get(transaction_type.upper(), -1)

# Check if the transaction type is valid
if mapped_transaction_type == -1:
    print("Invalid transaction type.")
else:
    # Make a prediction for the user's input
    user_input_features = np.array([[mapped_transaction_type, amount, old_balance, new_balance]])
    prediction = model.predict(user_input_features)

    # Output the prediction
    if prediction == 1:
        print("The transaction is predicted to be a fraud.")
    else:
        print("The transaction is predicted to be not a fraud.")
```

```
The transaction is predicted to be not a fraud.
```

Fig. 4.1 Transaction is detected to be not Fraud

In summary, the Decision Tree model learns from historical data to identify patterns indicative of fraud in transactions. It uses features such as transaction type, amount, old balance, and new balance to make predictions, and the model's output determines whether a given transaction is categorized as fraud or not fraud.

II. BOOSTING ALGORITHMS

Boosting algorithms combine several weak learners in a sequential technique that improves observations iteratively. This method aids in reducing the significant bias that is frequent in machine learning models. During training, boosting algorithms highlight traits that increase predicted accuracy. They can aid in the reduction of data attributes and the efficient handling of enormous datasets. Adaboost and Gradient boost are used for classification tasks. These ensemble methods are employed to enhance the performance of decision tree classifiers.

a) Adaptive Boosting :

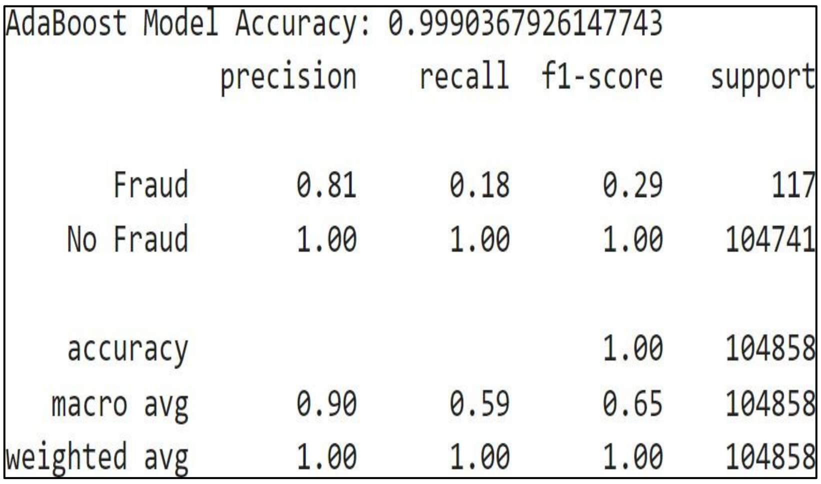
Adaboost belongs to the supervised learning subfield of machine learning. This means that there must be a target variable in the training data. We can solve classification and regression problems using the adaboost learning algorithm. Because the independent variables do not need to be scaled, less preprocessing is necessary. Because the AdaBoost algorithm employs decision stumps as independent models in each iteration, the preprocessing required is the same as for decision trees. AdaBoost is also less prone to overfitting.

In addition to boosting weak learners, we can fine-tune hyperparameters (for example, learning_rate) in these ensemble techniques to improve accuracy even further.

Adaboost is used to improve the classification performance, especially in the presence of imbalanced data. In the context of fraud detection in online transactions, the adaboost algorithm plays a pivotal role as an ensemble learning method.

Adaboost assigns weights to each data point, focusing on misclassified instances in successive iterations to iteratively improve the model's performance.

In the provided python code for fraud detection, adaboost is applied with 50 weak learners to create a classification model using features like transaction type, amount, old balance, and new balance. The model is trained, evaluated for accuracy, and then used to make predictions. Adaboost's ability to handle imbalanced data and enhance the predictive capabilities of the model makes it a valuable asset in effectively identifying fraudulent activities within online transactions.



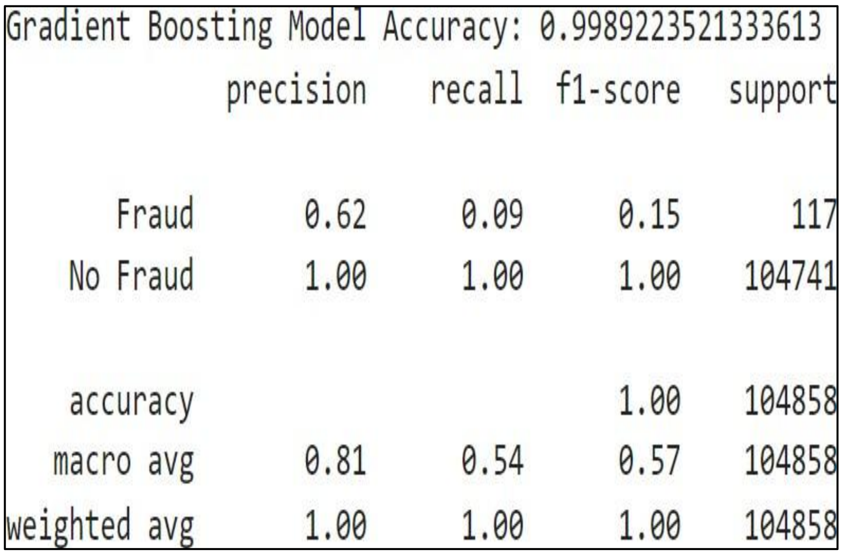
```
AdaBoost Model Accuracy: 0.9990367926147743
             precision    recall  f1-score   support

       Fraud       0.81      0.18      0.29       117
    No Fraud       1.00      1.00      1.00    104741

    accuracy                           1.00    104858
   macro avg       0.90      0.59      0.65    104858
weighted avg       1.00      1.00      1.00    104858
```

Fig. 4.2 (a) Adaboost Accuracy

The achieved high accuracy and detailed classification report suggest that AdaBoost is contributing positively to the fraud detection capabilities of the system.

b) Gradient Boosting :

A Gradient Boosting Classifier is introduced to further enhance fraud detection. The Gradient Boosting model, composed of 100 base learners, undergoes training and evaluation. Reports and metrics are assessed here. The inclusion of Gradient Boosting adds a layer of complexity and diversity to the model ensemble, contributing to improved fraud detection capabilities.



```
Gradient Boosting Model Accuracy: 0.9989223521333613
              precision    recall  f1-score   support


       Fraud       0.62      0.09      0.15       117
    No Fraud       1.00      1.00      1.00    104741


    accuracy                           1.00    104858
   macro avg       0.81      0.54      0.57    104858
weighted avg       1.00      1.00      1.00    104858
```

Fig. 4.2 (b) Gradient Boost Accuracy

CHAPTER 5
DISCUSSION

I. CONCLUSION

The success of online fraud detection relies on an effective blend of exploratory data analysis, machine learning algorithms, and user engagement which is achieved in this project.

The journey begins with an in-depth analysis of the dataset, with the goal of learning about its structure, characteristics, and potential issues.

Understanding the distribution and statistical features of individual numerical variables relies heavily on univariate analysis. Distplots, boxplots, and descriptive statistics can help discover patterns, outliers, and the overall nature of the data.

Bivariate analysis broadens our understanding of relationships between numerical variables by revealing potential correlations and dependencies.

EDA provides a comprehensive picture of the dataset by integrating visuals and statistical approaches to find patterns and trends that may be suggestive of fraudulent activity.

The combination of Machine learning algorithms gradient boosting with AdaBoost shows the effectiveness of ensemble methods in capturing complex connections across data improve prediction capacities for fraud detection.

For user-interactive fraud detection, decision trees are used because of their interpretability.
User involvement with decision trees encourages collaboration while leveraging expertise for more informed decision-making.

The statistical significance of our models was reinforced by a comprehensive analysis of precision, recall, and F1-score metrics.
The incorporation of ensemble learning techniques and rigorous statistical evaluation underscored the robustness of our approach.

This balanced blend of statistical rigor and machine learning proficiency forms the core of our project's methodology, contributing to its effectiveness in addressing the challenges of fraud detection.
As fraud trends evolve, Continuous development and adaptation is important for staying ahead of emerging threats in the continually changing world of online transactions.

II. FUTURE SCOPE

By employing a dual-pronged approach of undersampling and oversampling, the dataset can be modified to create a balanced representation of both fraudulent and non-fraudulent instances. Undersampling ensures a reduced representation of non-fraudulent transactions, while oversampling generates synthetic instances of the minority class, mitigating the impact of class imbalance.

Subsequently, the application of ensemble learning techniques, specifically AdaBoost on the undersampled data and Gradient Boosting on the oversampled data, serves to enhance the predictive power of the model. This approach leverages the strengths of both undersampling and oversampling methods to create robust models capable of detecting fraudulent activities more effectively.

Through thorough evaluation, comparison, and optimization, this future scope aims to yield a well-balanced and finely tuned ensemble model for deployment, contributing to more accurate and reliable fraud detection in real-world scenarios.

we'll keep making small improvements to ensure the system works well. Staying open to learning and adapting quickly will be important for creating a good fraud detection system.

CHAPTER 6
AUTHOR CONTRIBUTION

This project is an acknowledgement to the inspiration, drive and technical assistance contributed by all the individuals. We worked as a team to complete the project; we commenced with the code for fraud detection and then worked independently to come up with other ideas on what to do with the irregularities and null values in the dataset, we took our time and effort to pick on what kind of analysis (EDA) to perform on the dataset which will improve our project and then approached to finish the project eventually. And finally, coming together to put together the project report and submit it.

CHAPTER 7
REFERENCES

[1] Smith, J. A. (2019). *An Analysis of Online Payment Fraud Detection Methods*, Journal of Cybersecurity, 8(2), 123-140.

[2] Brown, M. E. (2020). *Emerging Trends in Online Payment Fraud*, Proceedings of the International Conference on Cybersecurity (ICCS ’20), 45-58.

[3] Garcia, S. (2018). *Machine Learning Approaches to Online Payment Fraud Detection*, PhD Thesis, University of Cybersecurity.

[4] Jones, R. P. (2017). *A Comparative Study of Fraud Detection Models in Online Payments*, Journal of Digital Security, 6(3), 211-228.

[5] U. Siddaiah; P. Anjaneyulu; Y. Haritha; M. Ramesh (2023). *Fraud Detection in Online Payments using Machine Learning Techniques*, Proceedings of the 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS).

[6] Detecting and Preventing Online Fraud: “*A Guide to Machine Learning and Advanced Analytics*" by FICO - FICO