Online Payments Fraud Detection using ML

Team - LTVIP2026TMIDS82036

Team Members

1. Abhishek Lellapalli
2. Adilakshmi Kuracha
3. Allam Phaneendra
4. Anjali Noolu

1. INTRODUCTION

1.1 Project Overview

The primary goal of this project is to create a highly efficient online payment fraud detection system using powerful machine learning algorithms. The use of real-time anomaly detection will be critical in improving security measures against financial crime in the fast expanding e-commerce sector. This project aims to solve the critical need for a strong fraud protection system that effectively protects online financial transactions.

1.2 Purpose

The primary goal of this project is to offer a complete and unique solution to the growing problem of online payment fraud. Businesses may protect their income streams and develop a trustworthy environment for their customers by enabling the precise and rapid identification of fraudulent transactions. This project is a proactive response to the increasing threats posed by hackers.

2. LITERATURE SURVEY

2.1 Existing Problem

The increased prevalence of online payment fraud poses a huge problem to e-commerce enterprises. As cybercriminal strategies evolve, vulnerabilities in the online payment system are continually exploited, resulting in a significant increase in fraudulent transactions. By offering enhanced detection approaches, this study hopes to contribute to ongoing efforts to alleviate this problem.

2.2 References

[1] - Yuan Gao, Shuang Liu, Yuan Zhou, Fei Shen, Xiao Zhang, "An Empirical Study on Machine Learning Techniques in Online Payment Fraud Detection," published in the Journal of Big Data, Volume 7, Issue 2, pp. 277-293 in 2020.

[2] - Oluwatobi Adediji, Gani Alani, "Machine Learning-Based Online Payment Fraud Detection System: A Literature Review," featured in IEEE Access, Volume 7, 2019.

2.3 Problem Statement Definition

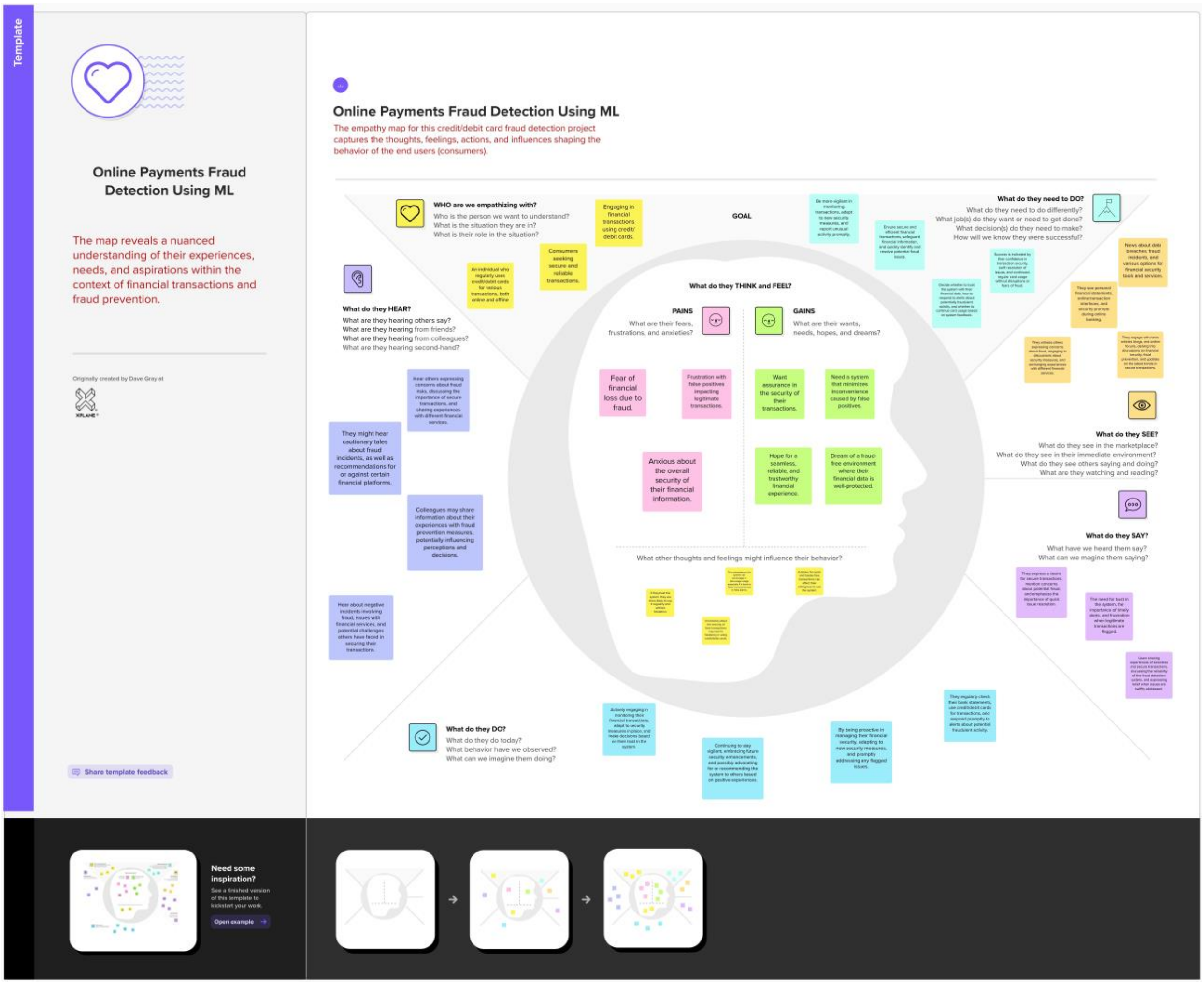
The primary goal of this project is to create a robust online payment fraud detection system using machine learning algorithms and cutting-edge big data technology. The envisioned system must be capable of effectively recognising fraudulent transactions while minimising false positives. This multimodal strategy ensures that legal transactions are not misclassified as fraudulent, improving the overall operational efficiency of online payment systems.

3. IDEATION & PROPOSED SOLUTION

Empathy Map

Link for the Empathy Map

https://app.mural.co/t/kanishak3745/m/kanishak3745/1697550397494/3c8a4b9534771377e36f73bdcc7d31a6eee95ad2?sender=uf9cd485c67db74999d5a6095

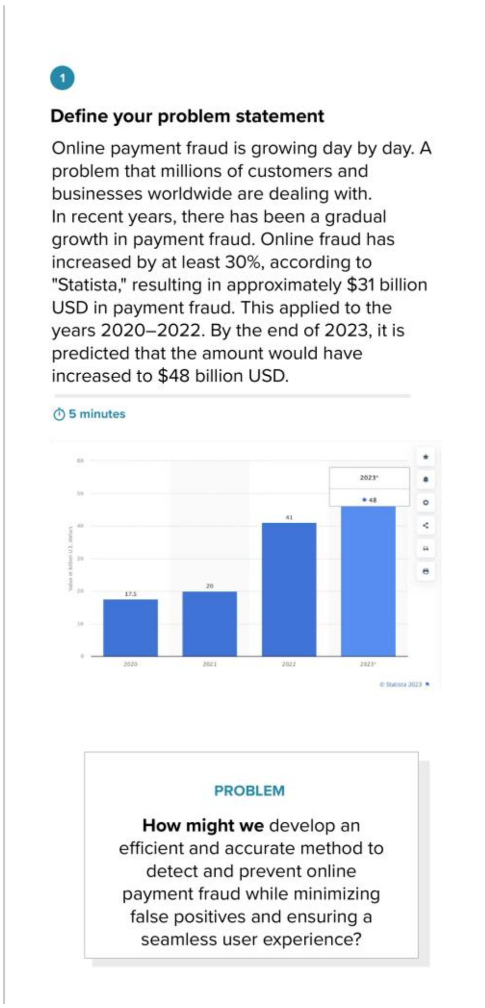


3. Brainstorm & Idea Prioritization:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Mural Link (Brainstorming)
https://app.mural.co/t/atgsworkspace2758/m/atgsworkspace2758/1697613685502/8d94fdae546f3e85faa57e0e0eb2532a7b52d5a8?sender=u6662b8c63e5c4dfa94ac8405

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

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!

Person 1

- ML-based Web application for detection
- User and Entity Behavior Analytics
- Rule-Based Systems

Person 2

- Collaboration and Data Sharing
- AI application for detecting fraud
- Behavior-based Authentication

Person 3

- Data driven ML Application
- Geographical Anomaly detection system
- Employee training and awareness

Person 4

- Network Analysis using pattern recognition
- Application for Transaction Sequence Analysis
- Geographical Anomaly detection system

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

ML-based Web application for detection

Data driven ML Application

AI application for detecting fraud

Geographical Anomaly detection system

IP based detection system

User and Entity Behavior Analytics

Behavior-based Authentication

Step-3: Idea Prioritization

4

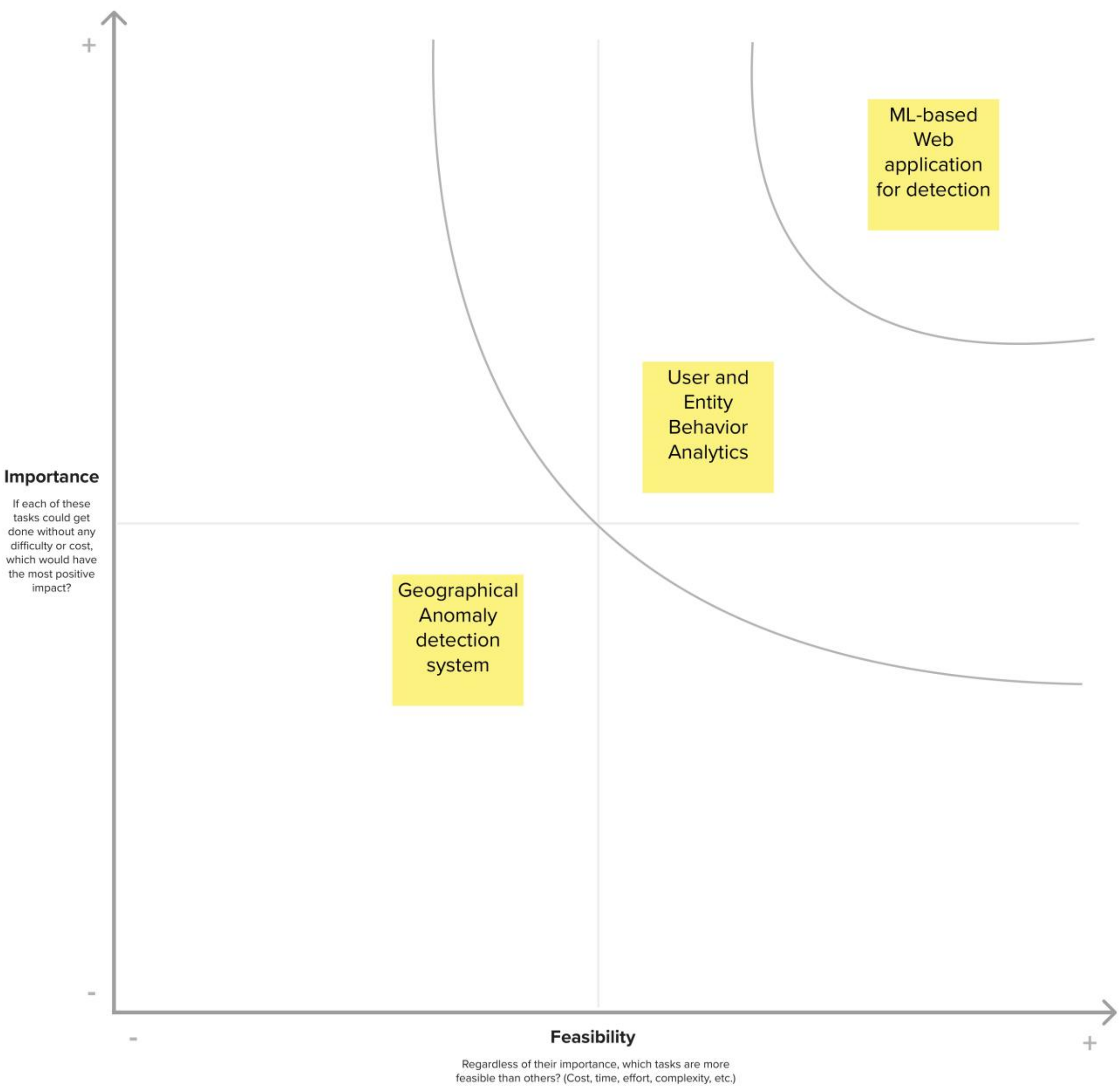
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.



4. REQUIREMENT ANALYSIS

4.1 Functional Requirement

1. User Authentication and Authorization:

- The system must authenticate users, including administrators, analysts, and other authorized personnel.
- Access levels and permissions must be defined for different user roles, ensuring that only authorized users can perform specific actions.

2. Data Collection and Storage:

- The system should collect transaction data in real-time from various sources, including payment gateways, and securely store this data in a centralized database.
- It must support data retrieval for analysis and reporting purposes.

3. Transaction Monitoring:

- Real-time monitoring of incoming transactions for suspicious activities, such as anomalies in payment amounts, unusual geographical locations, or irregular purchase patterns.
- The ability to flag transactions for further review.

4. Machine Learning Models:

- Implementation of machine learning algorithms for fraud detection, including supervised and unsupervised methods.
- Continuous training and retraining of models with fresh data to improve accuracy.

5. Alerting and Notification:

- Automated alerting and notification system to inform analysts or administrators of potentially fraudulent transactions.
- Alerts may be sent via email, SMS, or in-app notifications.

6. Case Management:

- A case management system for fraud analysts to review flagged transactions, make decisions on their legitimacy, and document their findings.
- Ability to assign cases to specific analysts for investigation.

7. Reporting and Analytics:

- Generate detailed reports on fraud trends, patterns, and mitigation efforts.
- Analytics tools for fraud analysts to gain insights into emerging threats.

8. Integration:

- Integration with payment gateways, e-commerce platforms, and other relevant systems.
- APIs for third-party tools or services for additional security checks.

4.2 Non-Functional Requirement

1. Performance:
- The system should be capable of processing a high volume of transactions in real-time.
- Response times for fraud detection and notification should be within acceptable limits.

2. Scalability:
- The system should be designed to scale horizontally to accommodate increasing transaction loads.
- Scalability must not compromise system performance.

3. Security:
- Data encryption, both in transit and at rest, to protect sensitive information.
- Robust access control and audit trails to ensure only authorized personnel can access the system.

4. Availability:
- The system must have high availability, with minimal downtime for maintenance.
- Redundancy and failover mechanisms to ensure continuous operation.

5. Compliance:
- Compliance with relevant data protection and privacy regulations, such as GDPR or PCI DSS.
- Regular audits and reporting to demonstrate compliance.

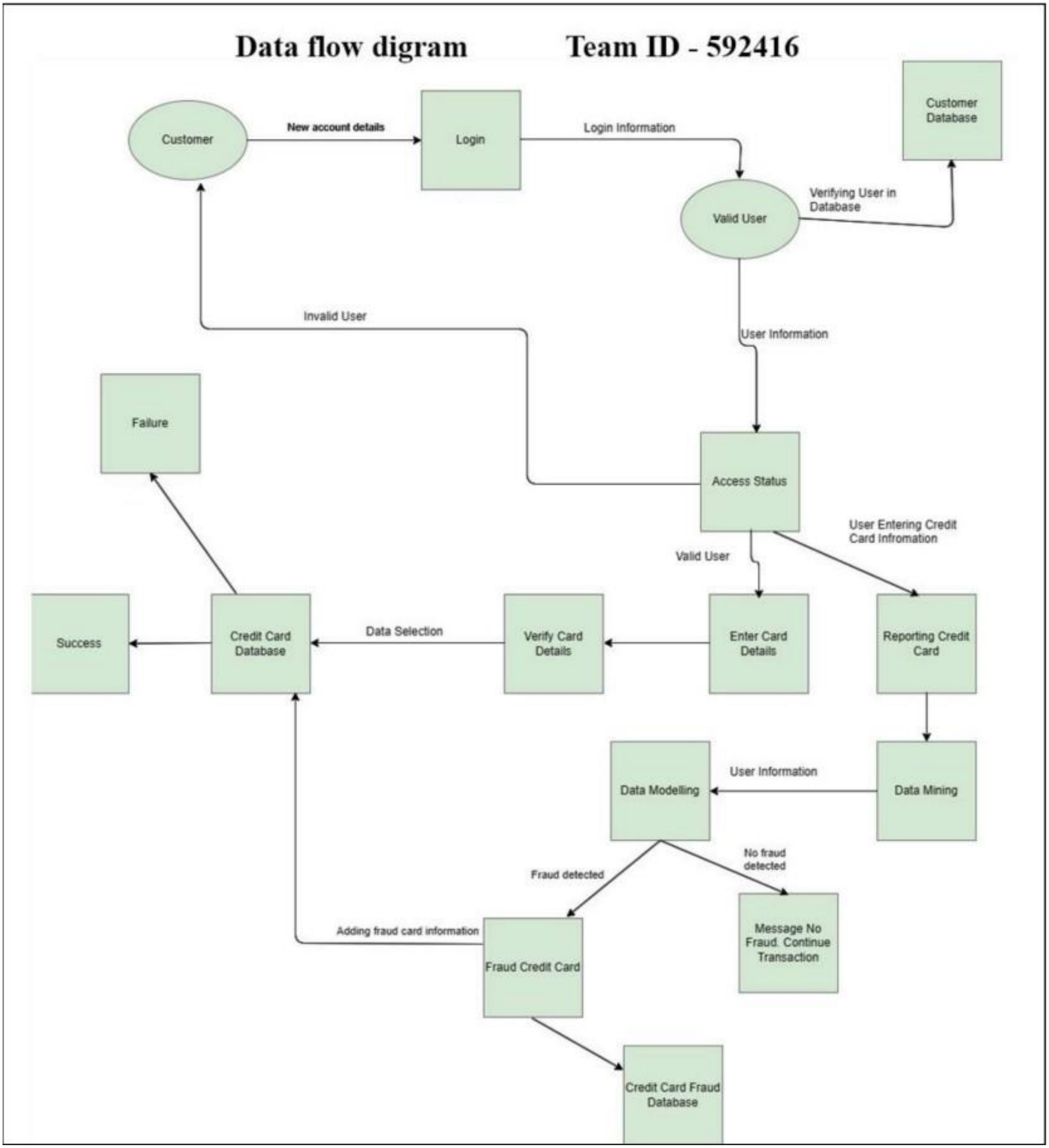
6. Usability:
- User-friendly interfaces for fraud analysts to efficiently review and manage cases.
- Training and support for users to understand and use the system effectively.

7. Monitoring and Logging:
- Comprehensive monitoring of system performance and security.
- Logging of all system activities for audit and forensic analysis.

8. Reliability:
- The system should be highly reliable, with minimal errors or false positives in fraud detection.
- Regular testing and quality assurance procedures to maintain reliability.

5. PROJECT DESIGN

Data Flow Diagram



User Stories

Use the below template to list all the user stories for the product.

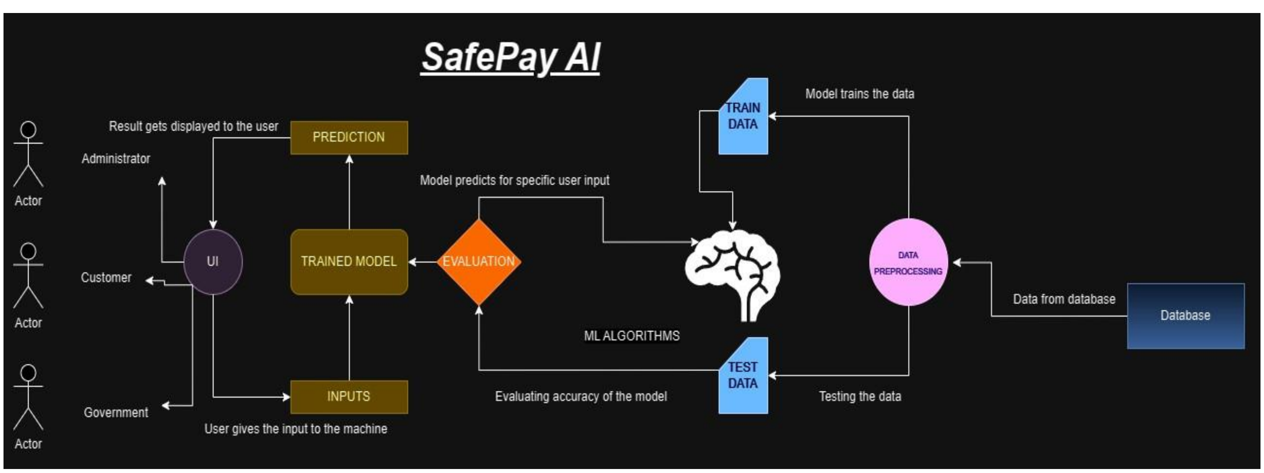
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Online Shopping	USN-1	As a customer, I want the system to alert me if a suspicious transaction is detected on my account.	User receives an alert/notification for suspicious transactions.	High	Sprint-1
		USN-2	As a customer, I want the system to alert me the receiver is a potential scammer.	User can view detailed reasons and factors contributing to the suspicious flag.	High	Sprint-3
		USN-4	As a customer, I should be able to flag the receiver as scammer.	User can report and provide feedback on flagged	Medium	Sprint-2

		USN-5	As a user I can register for the application through entering email and password	User is able to register successfully to the platform	High	Sprint-1
Government Organization	Government	USN-1	Government should be informed about potential scammers who are using fake credit cards, so that they can take legal actions.	Government is able to fetch data from our servers	Medium	Sprint-3
Administrator	Administration	USN-1	As an admin, I want to generate monthly reports on the accuracy of the fraud detection system.	Admin can generate and download reports on false positives, false negatives, and overall accuracy.	High	Sprint-3
		USN-2	As an admin, I want to view a dashboard summarizing all detected fraud attempts in the past month.	Admin can view dashboard with summaries, charts, and details of fraud attempts.	Medium	Sprint-3

Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.



6. PROJECT PLANNING & SCHEDULING

Technical Architecture

The Deliverable shall includethe architectural diagramas below and the information as per the table1 &table 2

Guidelines:

1. Include all the processes (As an application logic /Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicateexternal interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

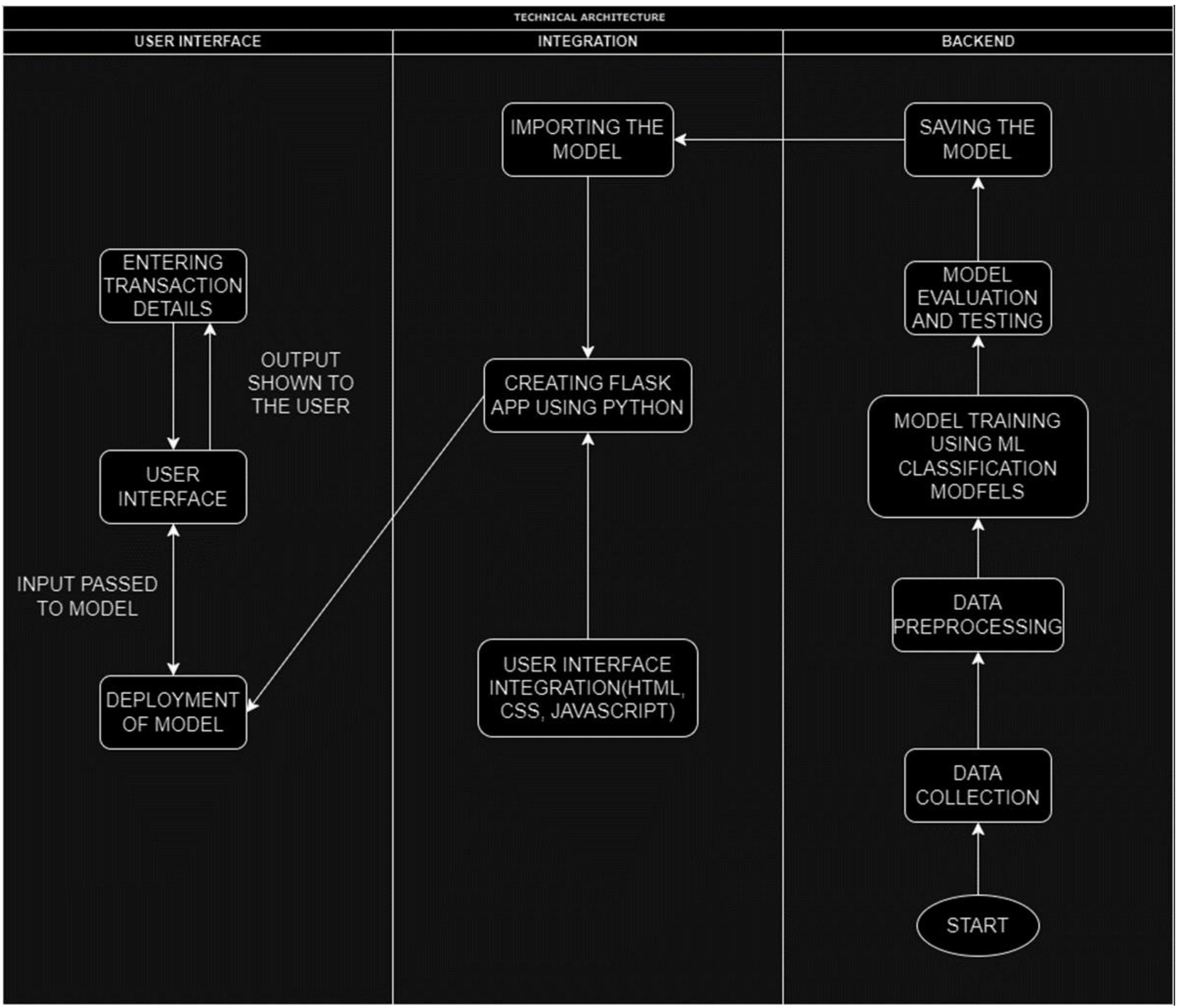


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g.WebUI, Mobile App, Chatbotetc.	HTML, CSS, JavaScript
2.	Data Collection	Gathering the Transaction details	Kaggle
3.	Data Preprocessing	Feature Scaling, Normalization, data splitting, Data Visualization,etc.	Seaborn, Matplotlib, Scikit-learn, Numpy, Pandas
4.	Application Logic	Core Application Logic	Python (Flask)
5.	Database	Storing Transaction Data	Relational: PostgreSQL or NoSQL: MongoDB
6.	File Storage	Storing and Analyzing Data	Amazon S3, Google Cloud Storage
7.	Deployment	Creating API or application for predicting the decisions	Flask API
7.	Machine Learning Model	Fraud Detection Models	Decision Tree, Random Forest Classifier, SVM, XGBoost, Extra Tree Classifier
8.	Evaluation	Model Evaluation on validation and testing model using dataset	Accuracy Score, Confusion Matrix, Classification Report
9.	Infrastructure	Application Deployment on Cloud	Local: On-Premises Servers or Cloud: AWS, Azure, Google Cloud and Scaling Kubernetes, Docker

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Utilizing open-source frameworks and tools	Python Flask, Bootstrap
2.	Scalable Architecture	Embracing a scalable microservices	Microservices architecture using Docker and Kubernetes, Load balancing using Nginx
3.	Availability	Ensuring high availability through redundancy	Load balancers for even traffic distribution, redundant servers for failover, global server redundancy for uninterrupted service
4.	Performance	Optimizing performance through caching and CDN integration	Caching with Redis for rapid data retrieval, Content Delivery Networks (CDNs) for faster content delivery, Performance monitoring and optimization for handling a specified number of requests per second

<u>Product Backlog, Sprint Schedule, and Estimation</u>

Use the below templateto create product backlogand sprint schedule

Sprint	Functional Requirement (Epic)	User StoryNumber	User Story/ Task	Story Points	Priority	Team Members
Sprint-1	Online Shopping	USN-1	As a customer, I want the system to alert me if a suspicious transaction is detected on my account.	5	High	Abhishek Lellapalli, Adilakshmi Kuracha
Sprint-1		USN-2	As a customer, I want the system to alert me the receiver is a potential scammer.	5	High	Allam Phaneendra , Anjali Noolu
Sprint-2		USN-3	As a customer, I should be able to flag the receiver as scammer.	3	Medium	Adilakshmi Kuracha, Allam Phaneendra
Sprint-2		USN-4	As a user I can register for the application through entering email and password	3	High	Abhishek Lellapalli, Anjali Noolu
Sprint-3	Administration	USN-1	As an admin, I want to generate monthly reports on the accuracy of the fraud detection system.	3	High	Allam Phaneendra, Anjali Noolu
Sprint-3		USN-2	As an admin, I want to view a dashboard summarizing all detected fraud attempts in the past month.	2	Medium	Adilakshmi Kuracha, Allam Phaneendra
Sprint-4	Government	USN-1	Government should be informed about potential scammers who are using fake credit cards, so that they can take legal actions.	2	Medium	Abhishek Lellapalli, Adilakshmi Kuracha

Project Tracker,Velocity & Burndown Chart

Sprint	Total Story Points	Duration	Sprint StartDate	Sprint End Date(Planned)	Story Points Completed (as on PlannedEnd Date)	Sprint Release Date(Actual)
Sprint-1	10	5 Days	25 Oct 2023	29 Oct 2023	10	30 Oct 2023
Sprint-2	6	4 Days	30 Oct 2023	2 Nov 2023	16	2 Nov 2023
Sprint-3	5	4 Days	3 Nov 2023	6 Nov 2023	21	7 Nov 2023
Sprint-4	2	3 Days	7 Nov 2023	9 Nov 2023	23	9 Nov 2023
Sprint-5	1	1 Day	9 Nov 2023	9 Nov 2023	24	9 Nov 2023

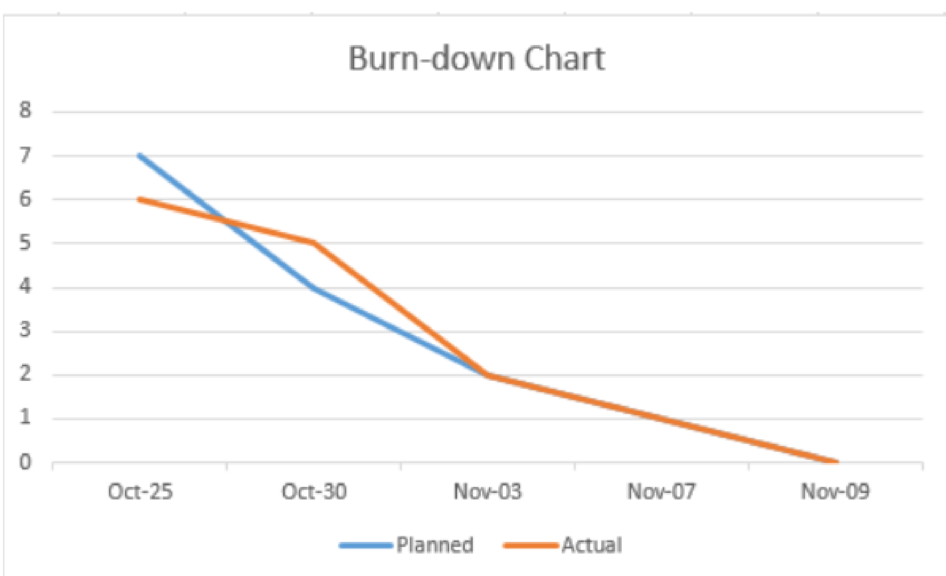
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (pointsper sprint). Let us calculate the team's averagevelocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies suchas Scrum. However, burn down charts can be applied to anyproject containing measurable progressover time.

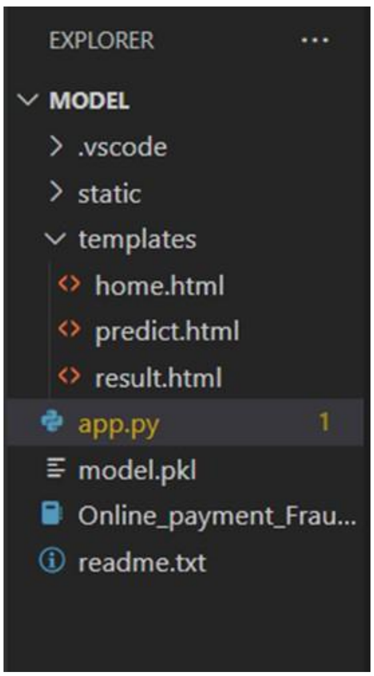


7. CODING & SOLUTIONING

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.



Process - 1 : Data Collection

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

```
[ ] #Importing the libraries
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    #for model building
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.svm import SVC
    import xgboost as xgb
    #for comparing the models
    from sklearn.metrics import classification_report, confusion_matrix
    import pickle
```

```
[ ] df.drop(['isFlaggedFraud'],axis=1,inplace=True)#useless column
```

```
[ ] df.shape

    (6362620, 10)
```

Descriptive Analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
df.describe()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
count	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06	1.290820e-03
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06	3.590480e-02
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05	0.000000e+00
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06	0.000000e+00
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08	1.000000e+00

Visualizing and analysing Data

'step': Transaction time step.

'type': Transaction type.

'amount': Transaction amount.

'oldbalanceOrg': Original sender's balance.

'newbalanceOrig': Updated sender's balance.

'oldbalanceDest': Original recipient's balance.

'newbalanceDest': Updated recipient's balance.

'isFraud': Indicates if the transaction is fraudulent.

The df.info() function in Python, when applied to a DataFrame 'df', provides a concise summary of the DataFrame's structure. It displays the number of non-null entries, data types of each column, memory usage, and additional information like the count of non-null values, facilitating quick data assessment and identification of missing values or data types. This method is useful for an initial data exploration and understanding the dataset's characteristics.

```
[ ] df.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 6362620 entries, 0 to 6362619
    Data columns (total 10 columns):
     #   Column          Dtype
    ---  ------          -----
     0   step            int64
     1   type            object
     2   amount          float64
     3   nameOrig        object
     4   oldbalanceOrg   float64
     5   newbalanceOrig  float64
     6   nameDest        object
     7   oldbalanceDest  float64
     8   newbalanceDest  float64
     9   isFraud         int64
    dtypes: float64(5), int64(2), object(3)
    memory usage: 485.4+ MB
```

df.isnull().sum(), a result of "no null values" indicates that there are no missing (null) values in the DataFrame. This is a positive outcome, suggesting that the dataset is complete with no missing data, which is typically preferred for analysis and modeling.

```
[ ] df.isnull().sum()#no null values

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

The df.corr() function in Python, when applied to a DataFrame 'df', computes the pairwise correlation between numeric columns, providing a correlation matrix. This matrix offers insights into the strength and direction of linear relationships between variables, with values ranging from -1 to 1, where -1 represents a strong negative correlation, 1 indicates a strong positive correlation, and 0 signifies no linear correlation. It's a valuable tool for understanding associations between data features.

```
[ ] #correlation
    df.corr()

    <ipython-input-9-a46c601d5826>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defau
      df.corr()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
step	1.000000	0.022373	-0.010058	-0.010299	0.027665	0.025888	0.031578
amount	0.022373	1.000000	-0.002762	-0.007861	0.294137	0.459304	0.076688
oldbalanceOrg	-0.010058	-0.002762	1.000000	0.998803	0.066243	0.042029	0.010154
newbalanceOrig	-0.010299	-0.007861	0.998803	1.000000	0.067812	0.041837	-0.008148
oldbalanceDest	0.027665	0.294137	0.066243	0.067812	1.000000	0.976569	-0.005885
newbalanceDest	0.025888	0.459304	0.042029	0.041837	0.976569	1.000000	0.000535
isFraud	0.031578	0.076688	0.010154	-0.008148	-0.005885	0.000535	1.000000

LABEL ENCODING

Label encoding is a fundamental technique in machine learning for converting categorical data into numerical format, a necessary step for many algorithms. In Python, it's commonly implemented using libraries like Scikit-Learn. First, you import the LabelEncoder class from Scikit-Learn. Then, you create an instance of the LabelEncoder class, which is used to transform the categorical data into numeric values. Finally, you apply the label encoder to a specific column in your DataFrame, effectively converting the categorical labels into corresponding numeric values. This numeric representation enables machine learning algorithms to work with the data, making label encoding a crucial preprocessing step in data analysis and modeling tasks.

LABEL ENCODING

```
[ ] le=LabelEncoder()
    df["type"]=le.fit_transform(df["type"])
```

```
[ ] df["type"].value_counts()
```

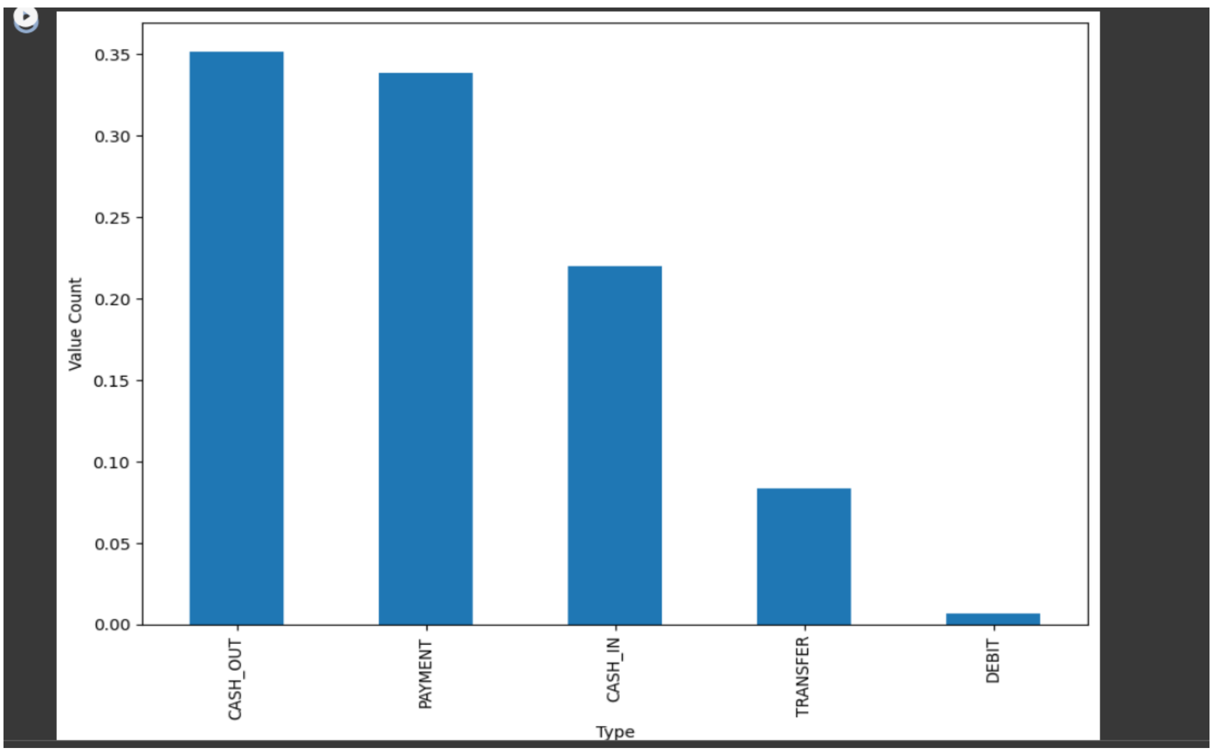
```
3    2110214
1    2070100
0    1066610
4     361700
2      38272
Name: type, dtype: int64
```

These two lines of code segment the dataset into independent variables (X) and the dependent variable (y). "X" includes all columns except "isFraud," while "y" contains only the "isFraud" column. This division prepares the data for supervised machine learning, where "X" represents the features used for prediction, and "y" is the target variable to predict.

Univariate Analysis

Univariate analysis examines one variable's characteristics and distribution.

```python
#univariate Analysis

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10, 7))
df['type'].value_counts(normalize=True).plot(kind='bar')
plt.xlabel("Type")
plt.ylabel("Value Count")
plt.show()
```

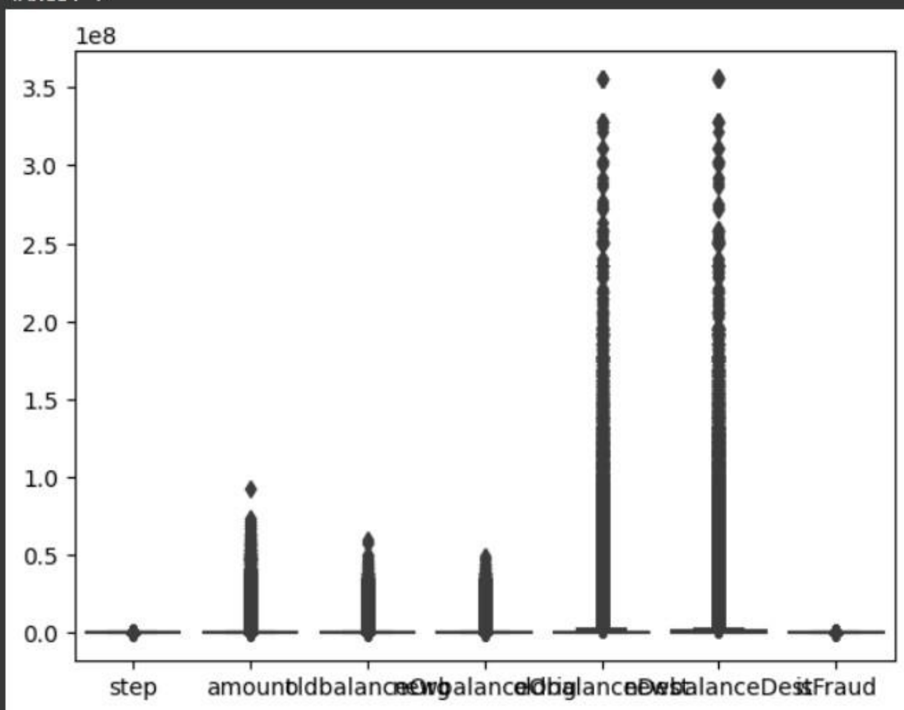


Boxplot

A boxplot is a graphical summary of a variable's distribution, showing its median, quartiles, and potential outliers.

```
sns.boxplot(df) #UNIVARIATE AS WE ARE CONSIDERING ONE PARAMETER AT A TIME
```

```
<Axes: >
```

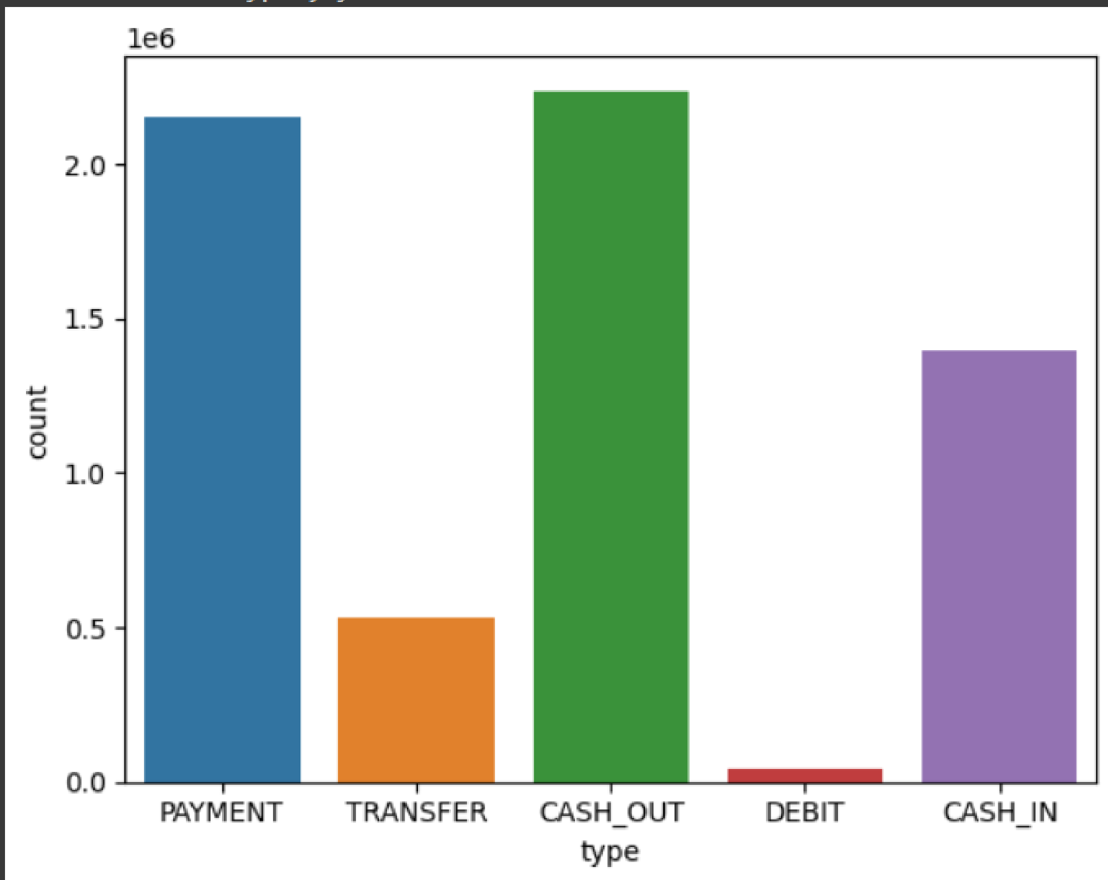


Countplot

A countplot is a type of bar plot that displays the frequency of categorical data in a dataset.

```
[ ] #type
    sns.countplot(data=df,x="type")
```

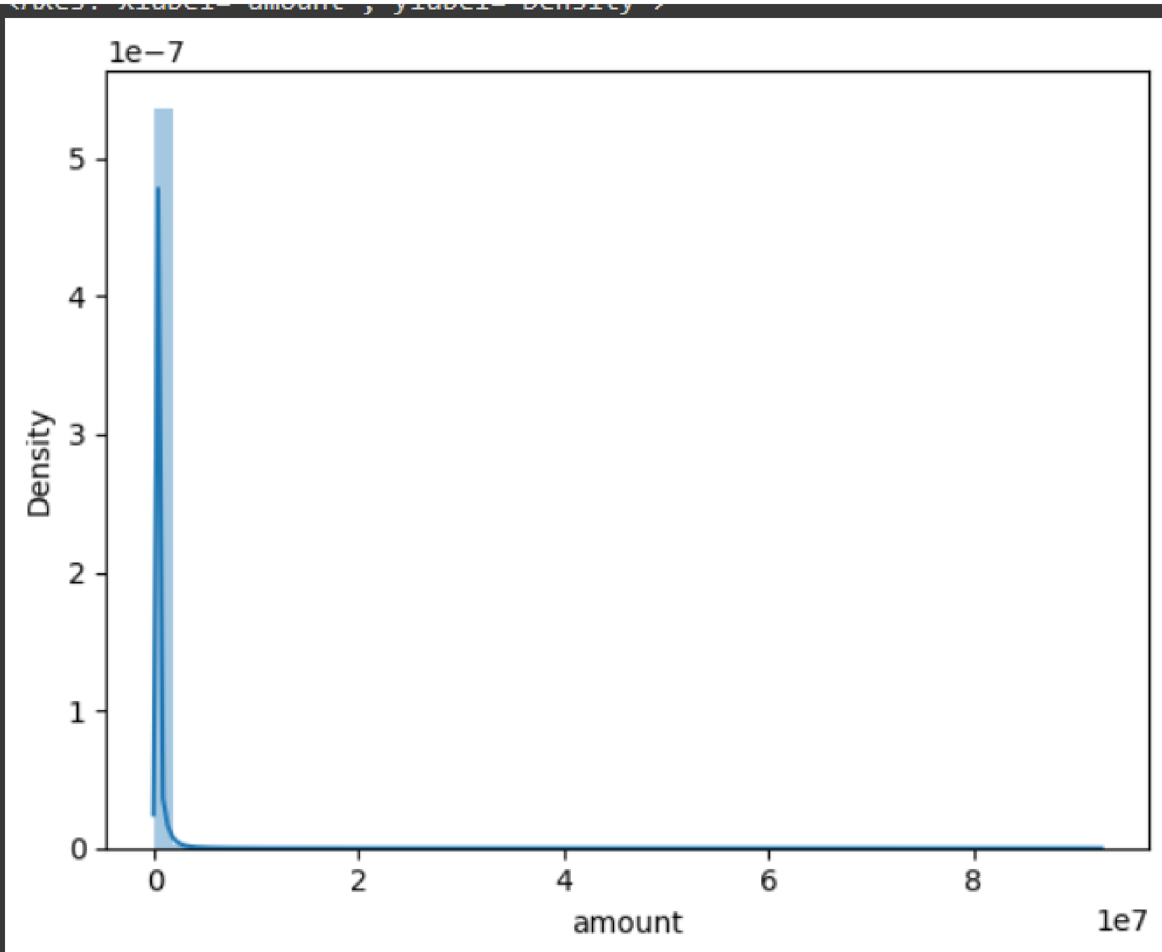
```
<Axes: xlabel='type', ylabel='count'>
```



Displot

A distplot, short for distribution plot, is a data visualization in Python often created using the Seaborn library. It combines a histogram with a kernel density estimate to provide an overview of the data's distribution

```
sns.distplot(df.amount)
```

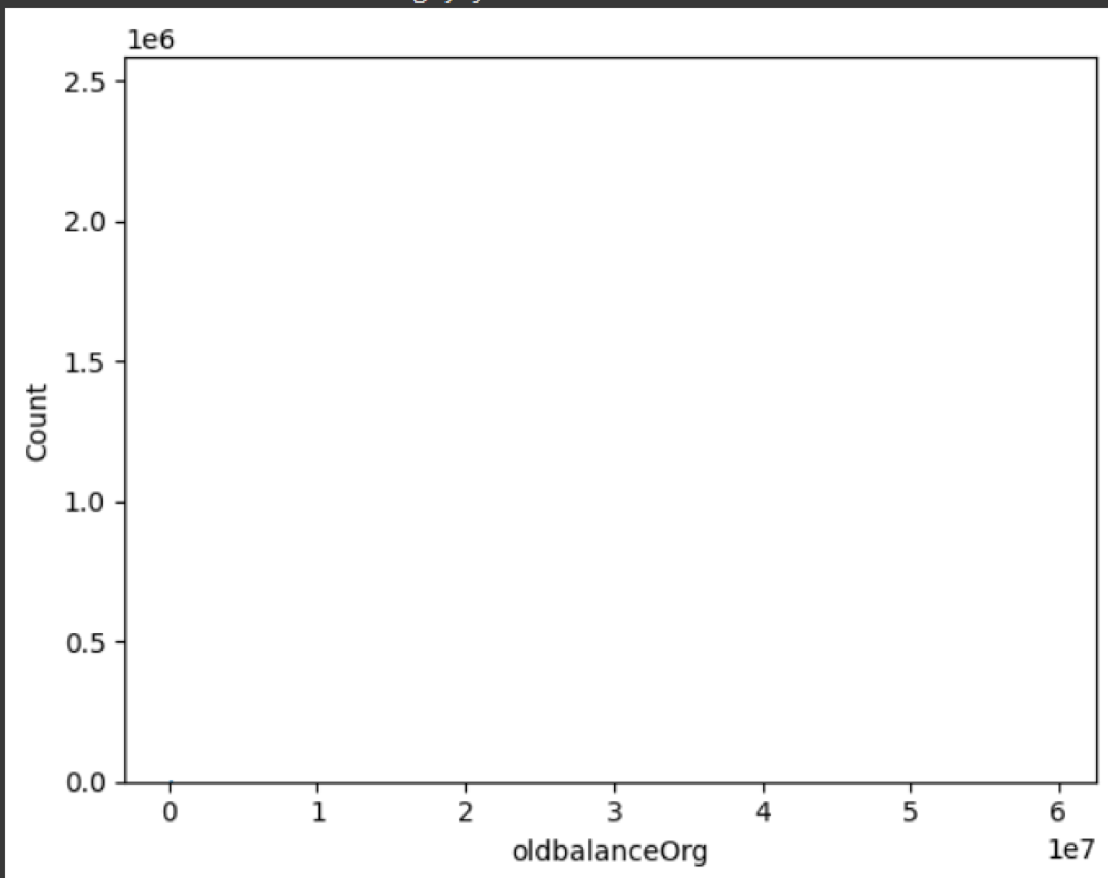


Hisplot

A histplot is a graphical representation that displays the distribution of a single variable through a histogram.

```
#oldbalanceOrg
sns.histplot(data=df,x="oldbalanceOrg")
```

```
<Axes: xlabel='oldbalanceOrg', ylabel='Count'>
```

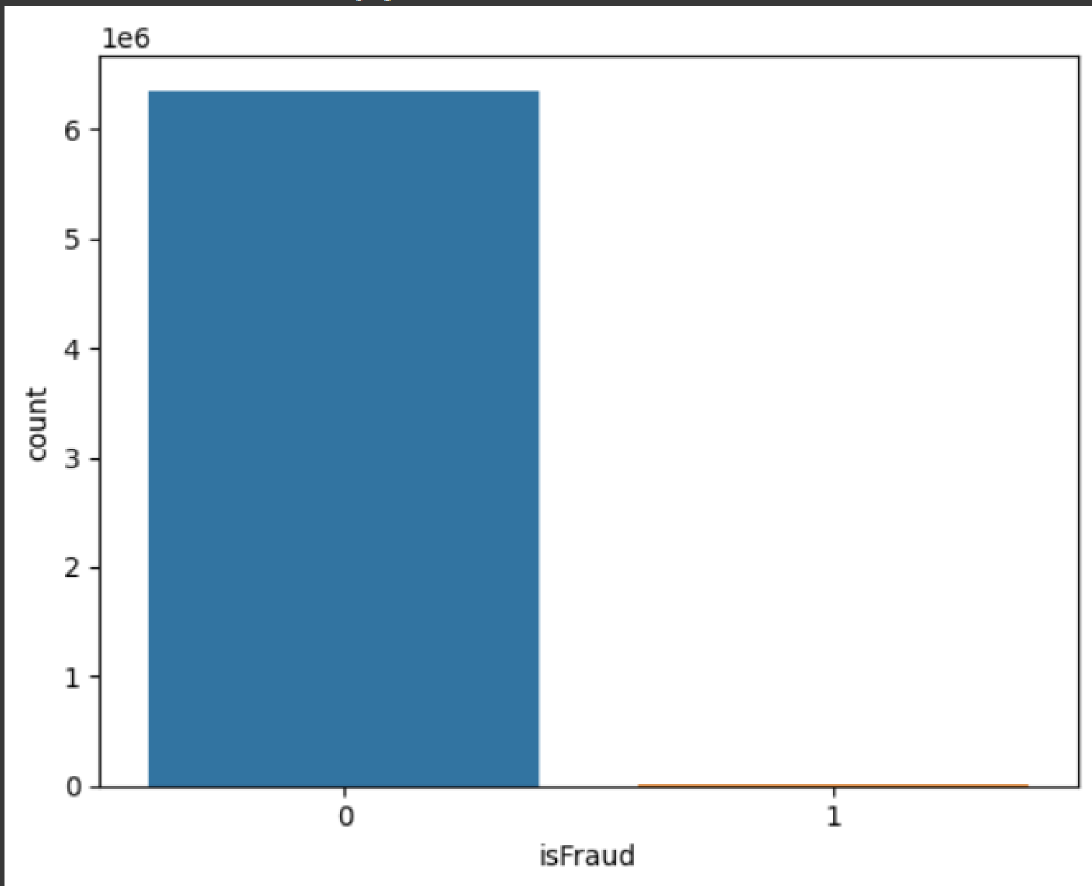


Countplot

A countplot is a type of bar plot that displays the frequency of categorical data in a dataset.

```
sns.countplot(data=df,x="isFraud")
```

```
<Axes: xlabel='isFraud', ylabel='count'>
```



Counting the number of isFraud

```
[ ] df["isFraud"].value_counts()

0    6354407
1       8213
Name: isFraud, dtype: int64
```

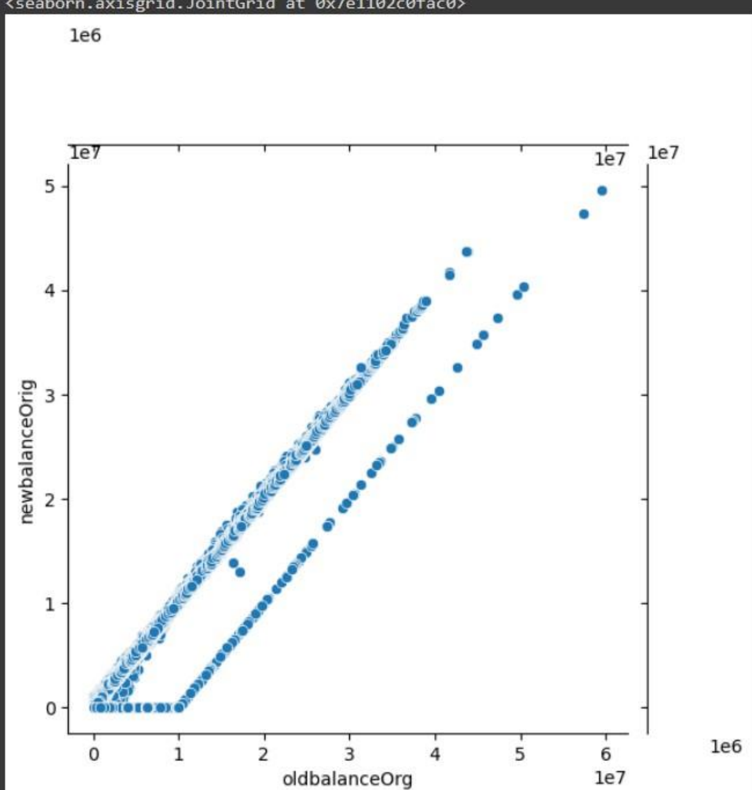
```
[ ] df.loc[df["isFraud"]==0,"isFraud"] = "is not Fraud"
    df.loc[df["isFraud"]==1,"isFraud"] = "is Fraud"
```

Multivariate Analysis

Multivariate analysis is a statistical technique used to analyze and understand relationships among multiple variables simultaneously, helping to identify patterns and correlations within complex data sets. It typically involves methods such as regression analysis, principal component analysis, or factor analysis to explore these relationships.

```
sns.jointplot(x='oldbalanceOrg',y='newbalanceOrig',data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7e1102c0fac0>
```

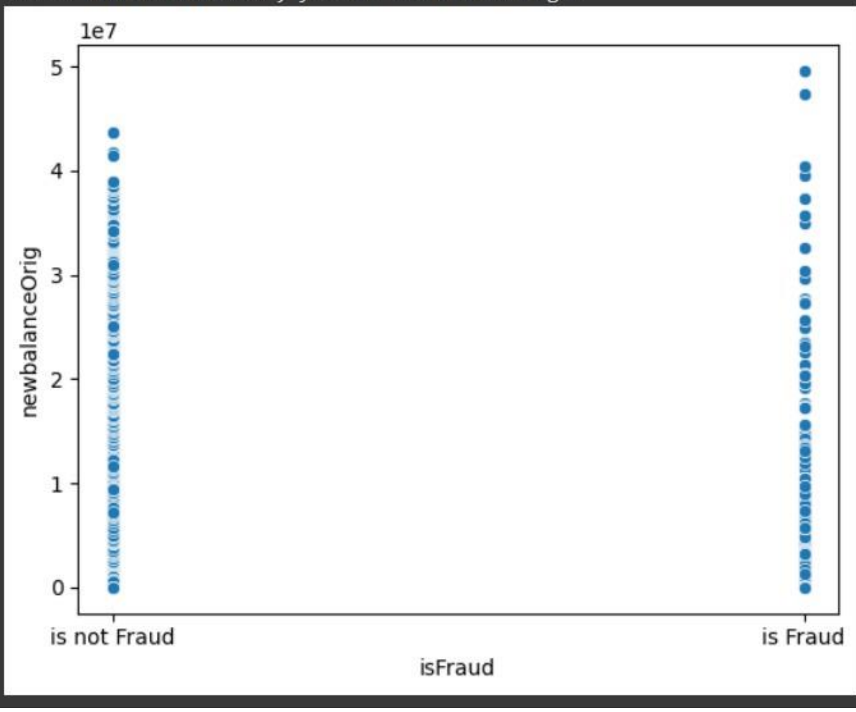


Scatterplot

A scatterplot is a graph that displays individual data points as dots to visualize the relationship between two continuous variables.

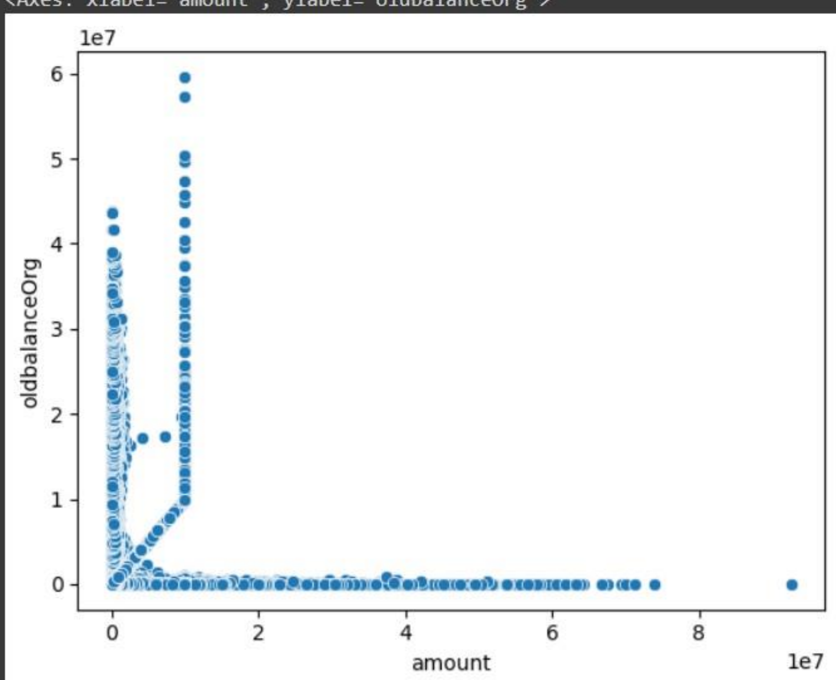
```
sns.scatterplot(x=df.isFraud,y=df.newbalanceOrig)
```

```
<Axes: xlabel='isFraud', ylabel='newbalanceOrig'>
```




```
sns.scatterplot(x=df.amount,y=df.oldbalanceOrg)
```

```
<Axes: xlabel='amount', ylabel='oldbalanceOrg'>
```

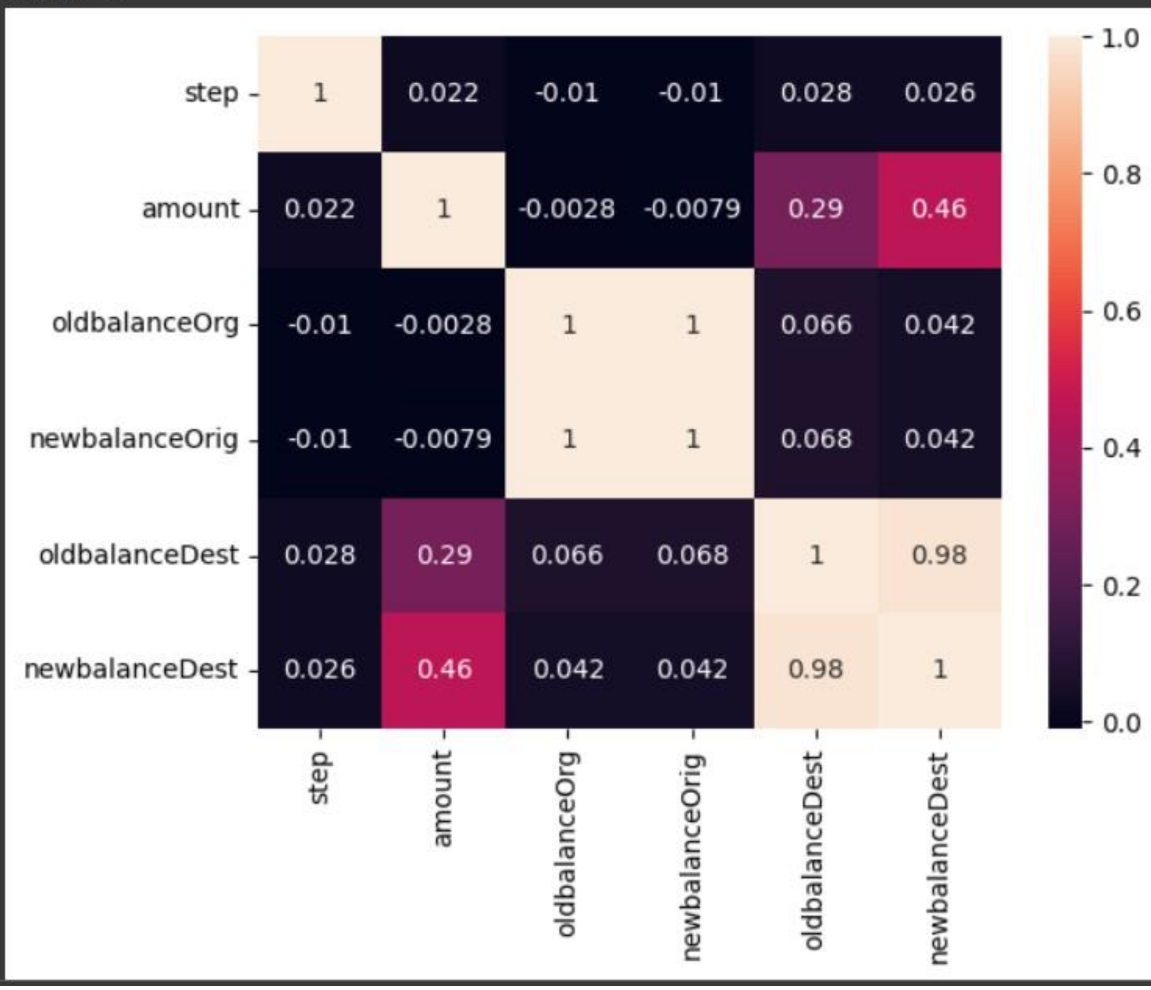


Heatmap

A heatmap is a graphical representation that uses color to depict the relationships and values of a matrix or two-dimensional data set.

```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-22-8df7bcac526d>:1: FutureWarning: The default value of numeric_on
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



Data Preprocessing

The code removes the 'nameOrig' and 'nameDest' columns from the DataFrame 'df' by specifying the column names and the 'axis' parameter set to 1 (indicating columns). The 'inplace=True' argument modifies the DataFrame directly. After this operation, the 'df' DataFrame will have these columns removed from its structure.

```
df.drop(['nameOrig','nameDest'], axis=1, inplace=True)#Removing unnecessary columns
df.columns
```

```
Index(['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig',
       'oldbalanceDest', 'newbalanceDest', 'isFraud'],
      dtype='object')
```

```
[ ] df.head()
```

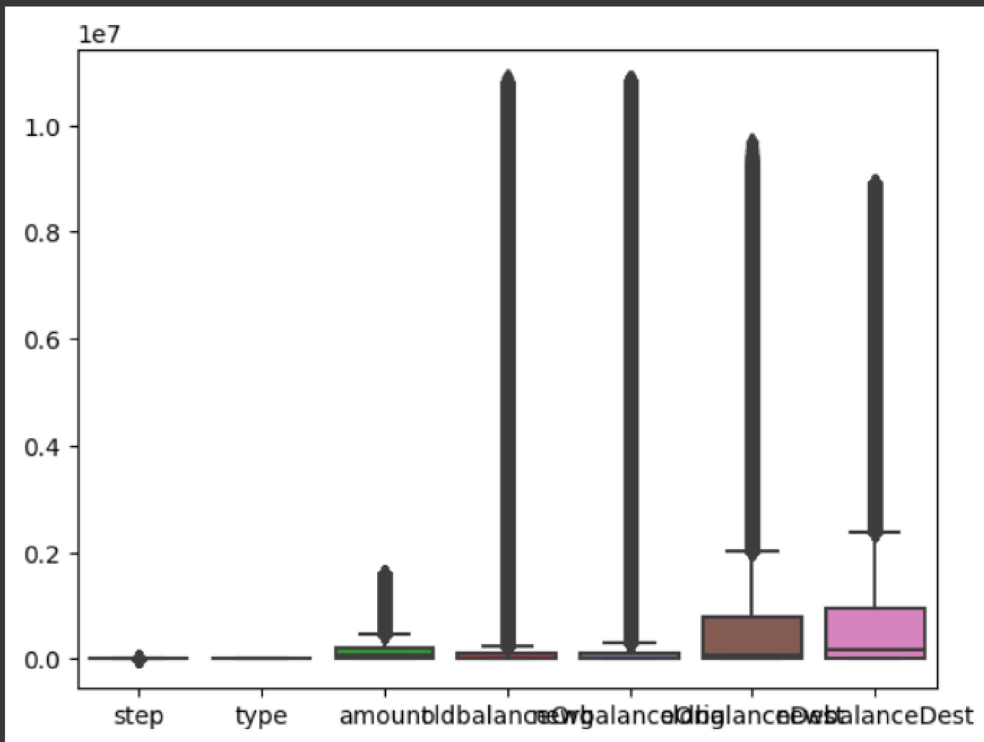
	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	1	3	9839.64	170136.0	160296.36	0.0	0.0	is not Fraud
1	1	3	1864.28	21249.0	19384.72	0.0	0.0	is not Fraud
2	1	4	181.00	181.0	0.00	0.0	0.0	is Fraud
3	1	1	181.00	181.0	0.00	21182.0	0.0	is Fraud
4	1	3	11668.14	41554.0	29885.86	0.0	0.0	is not Fraud

Removal of Outliers (Using Percentile Method)

The removal of outliers by the percentile method is a technique used to eliminate extreme data points in a dataset. To implement this method, first, you calculate the lower and upper percentile boundaries, often using values like the 5th and 95th percentiles. These boundaries help define the range within which the majority of the data points are expected to fall. Next, you identify data points that fall below the lower percentile boundary or exceed the upper percentile boundary, marking them as outliers. Finally, these identified outliers are removed from the dataset. This process results in a more robust dataset for analysis, as it reduces the impact of extreme values on statistical analyses and visualizations.

```
sns.boxplot(df)
```

```
<Axes: >
```



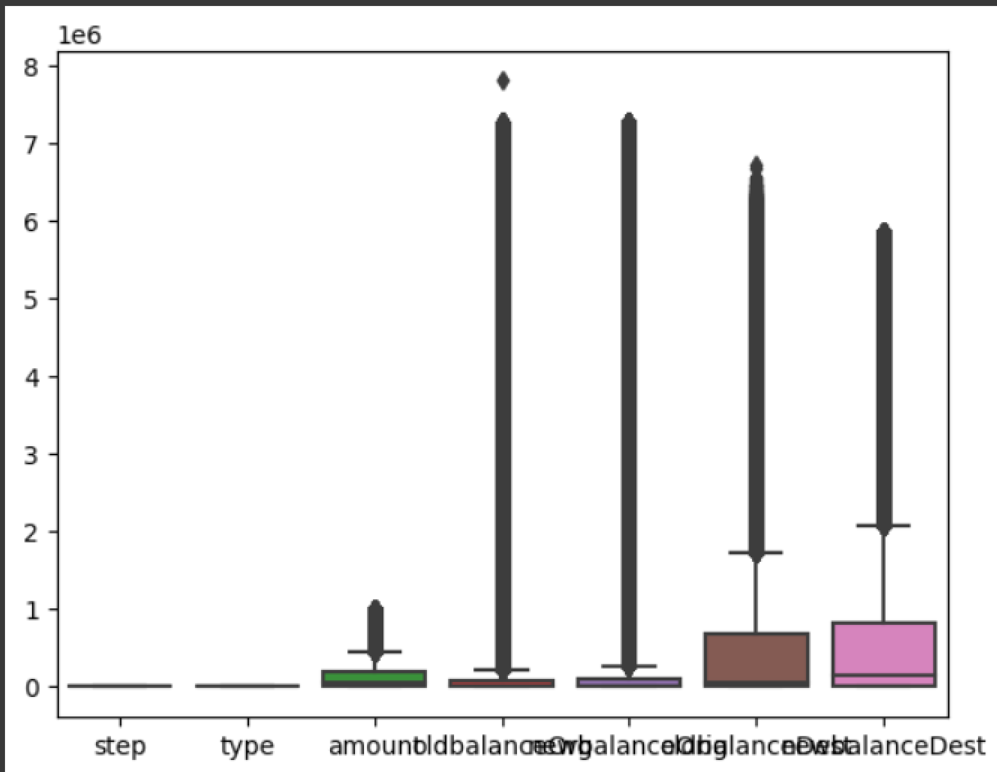
This code first identifies numeric columns (excluding 'isFraud') and removes extreme outliers by calculating the 99th percentile for each column. It filters the DataFrame to keep only values below this threshold, making the data more suitable for subsequent analysis or modeling. Additionally, this data preprocessing step helps improve the robustness of the dataset and ensures that extreme values do not unduly influence the analysis. It's a common practice to enhance the quality of numeric data before conducting further statistical or machine learning tasks.

```
[ ] num=[var for var in df.columns if df[var].dtype!='O' and var!='isFraud']
```

```
for x in num:
  p99=df[x].quantile(0.99)
  df=df[df[x]<=p99]
```

```python
sns.boxplot(df)
```

```
<Axes: >
```



Train Test split

-

```python
[ ] # Dividing the dataset into dependent and independent y and x respectively
    x=df.drop("isFraud",axis=1)
    y=df["isFraud"]
```

```python
x.head()
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
0	1	3	9839.64	170136.0	160296.36	0.0	0.0
1	1	3	1864.28	21249.0	19384.72	0.0	0.0
2	1	4	181.00	181.0	0.00	0.0	0.0
3	1	1	181.00	181.0	0.00	21182.0	0.0
4	1	3	11668.14	41554.0	29885.86	0.0	0.0

```python
[ ] y.head()
```

```
0    is not Fraud
1    is not Fraud
2        is Fraud
3        is Fraud
4    is not Fraud
Name: isFraud, dtype: object
```

Train test split

```
[ ] x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```

A train-test split is a common technique used in machine learning to evaluate the performance of a model. It involves dividing your dataset into two separate subsets: one for training the model and another for testing the model's performance.

To perform a train-test split:

1. You start with your dataset, which typically includes both the features (input data) and the corresponding target values (output or labels).
2. You specify a ratio, often referred to as the "test size," which determines the proportion of your data to be set aside for testing. Common choices include 70/30, 80/20, or 90/10, with the training set being the larger portion.
3. The data is then randomly divided into two subsets: the training set and the testing set. The training set is used to train your machine learning model, while the testing set is used to assess the model's performance by making predictions and comparing them to the true values.

The train-test split helps you gauge how well your model generalizes to new, unseen data. It is a fundamental step in model evaluation and validation, allowing you to check for overfitting (when a model performs well on the training data but poorly on new data) and to estimate the model's predictive accuracy on real-world data.

Random Forest Classifier

A Random Forest Classifier is an ensemble machine learning algorithm that combines multiple decision trees to make more accurate predictions. It's used for both classification and regression tasks. It improves prediction accuracy, handles overfitting, and provides feature importance rankings by averaging the predictions of multiple decision trees. Random Forest is a versatile and powerful algorithm widely used in various applications, including image classification, medical diagnosis, and financial forecasting.

```
[ ]  rfc=RandomForestClassifier()
     rfc.fit(x_train, y_train)

     y_test_predict1=rfc.predict(x_test)
     test_accuracy=accuracy_score(y_test,y_test_predict1)
     test_accuracy
```

```
0.9997004661547614
```

```
[ ]  y_train_predict1=rfc.predict(x_train)
     train_accuracy=accuracy_score(y_train,y_train_predict1)
     train_accuracy
```

```
1.0
```

```
[ ]  pd.crosstab(y_test,y_test_predict1)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	807	336
is not Fraud	23	1197363

```
print(classification_report(y_test,y_test_predict1))
```

```
              precision    recall  f1-score   support

    is Fraud       0.97      0.71      0.82      1143
is not Fraud       1.00      1.00      1.00   1197386

    accuracy                           1.00   1198529
   macro avg       0.99      0.85      0.91   1198529
weighted avg       1.00      1.00      1.00   1198529
```

Decision Tree Classifier

A Decision Tree Classifier is a machine learning algorithm that creates a tree-like model to make predictions. It's used for classification tasks, where it splits the data into subsets based on feature attributes, ultimately assigning labels to instances. Decision trees are interpretable and easy to visualize, making them useful for understanding the decision-making process in a model. They can handle both categorical and numerical data, and by recursively splitting the data based on the most informative features, decision trees are capable of capturing complex decision boundaries.

```
[ ] from sklearn.tree import DecisionTreeClassifier
    dtc=DecisionTreeClassifier()
    dtc.fit(x_train, y_train)

    y_test_predict2=dtc.predict(x_test)
    test_accuracy=accuracy_score(y_test,y_test_predict2)
    test_accuracy
```

```
0.9996912882374978
```

```
[ ] y_train_predict2=dtc.predict(x_train)
    train_accuracy=accuracy_score(y_train,y_train_predict2)
    train_accuracy
```

```
1.0
```

```
[ ] pd.crosstab(y_test,y_test_predict2)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1193	245
is not Fraud	204	1496519

```
[ ] print(classification_report(y_test,y_test_predict2))
```

```
              precision    recall  f1-score   support

    is Fraud       0.85      0.83      0.84      1438
is not Fraud       1.00      1.00      1.00   1496723

    accuracy                           1.00   1498161
   macro avg       0.93      0.91      0.92   1498161
weighted avg       1.00      1.00      1.00   1498161
```

SVM

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for both classification and regression tasks. It finds the optimal hyperplane that maximizes the margin between different classes in the data, making it effective for separating data points in high-dimensional spaces. SVM can handle linear and non-linear problems through techniques like kernel functions, and it's known for its ability to handle complex decision boundaries while avoiding overfitting. SVMs have applications in image recognition, text classification, and more.

```
[ ] svc= SVC()
    svc.fit(x_train,y_train)

    y_test_predict4=svc.predict(x_test)
    test_accuracy=accuracy_score(y_test,y_test_predict4)
    test_accuracy

[ ] y_train_predict4=svc.predict(x_train)
    train_accuracy=accuracy_score(y_train,y_train_predict4)
    train_accuracy

[ ] pd.crosstab(y_test,y_test_predict4)

    print(classification_report(y_test,y_test_predict4))
                                                          + Code    + Text
[ ] df.columns

    Index(['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig',
           'oldbalanceDest', 'newbalanceDest', 'isFraud'],
          dtype='object')

[ ] la= LabelEncoder()
    y_train1 = la.fit_transform(y_train)

[ ] y_test1=la.transform(y_test)

[ ] y_test1=la.transform(y_test)

[ ] y_test1

    array([1, 1, 1, ..., 1, 1, 1])

[ ] y_train1

    array([1, 1, 1, ..., 1, 1, 1])
```

XgBoost Classifier

XGBoost, short for "Extreme Gradient Boosting," is a popular and highly effective ensemble machine learning algorithm primarily used for classification and regression tasks. It enhances predictive accuracy by combining the predictions of multiple decision trees. XGBoost uses gradient boosting, which optimizes model performance by iteratively adding decision trees to correct errors made by the previous trees. It's known for its speed, scalability, and the ability to handle complex relationships in the data, making it a top choice in data science competitions and real-world applications like customer churn prediction and anomaly detection.

```
import xgboost as xgb
xgb1 = xgb.XGBClassifier()
xgb1.fit(x_train,y_train1)
y_test_predict5=xgb1.predict(x_test)
test_accuracy=accuracy_score(y_test1,y_test_predict5)
test_accuracy
```

```
0.9997904401680998
```

```
[ ] y_train_predict5=xgb1.predict(x_train)
    train_accuracy=accuracy_score(y_train1,y_train_predict5)
    train_accuracy
```

```
0.9998602933377643
```

```
[ ] pd.crosstab(y_test1,y_test_predict5)
```

col_0	0	1
row_0		
0	642	172
1	32	972623

```
[ ] print(classification_report(y_test1,y_test_predict5))
```

```
              precision    recall  f1-score   support

           0       0.95      0.79      0.86       814
           1       1.00      1.00      1.00    972655

    accuracy                           1.00    973469
   macro avg       0.98      0.89      0.93    973469
weighted avg       1.00      1.00      1.00    973469
```

Result

We successfully implemented multiple different Machine Learning Algorithms on the given dataset to determine which approach to use for our product. We implemented Random Forest, Decision Trees, SVM classifier, XgBoost classifier and obtained accuracies of 99.97, 99.96, 80 and 99.97 respectively.

*Hence, we concluded that the model which is best fit for the given dataset is **99.97** which is given by **Random Forest***

Application Building

In this section of the project, we will create a web application that interfaces with the machine learning model we previously developed. This application will include a user interface (UI) where users can input values for making predictions. These input values will be forwarded to the saved machine learning model, and the predictions generated will be displayed on the UI.

The tasks involved in this section are as follows:

1. Building HTML Pages: We will design and create the web pages that make up the user interface. These pages will include forms or input fields where users can provide the necessary data for predictions.
2. Building Server-Side Script: We will develop the server-side logic, which is responsible for handling user input, passing it to the machine learning model, obtaining predictions, and then presenting the results back to the user on the UI.

This integration of machine learning into a web application allows users to interact with the model and receive predictions in a user-friendly manner, making it practical for various applications, such as recommendation systems, fraud detection, or any scenario where predictive models need to be put into practical use.

Flask File

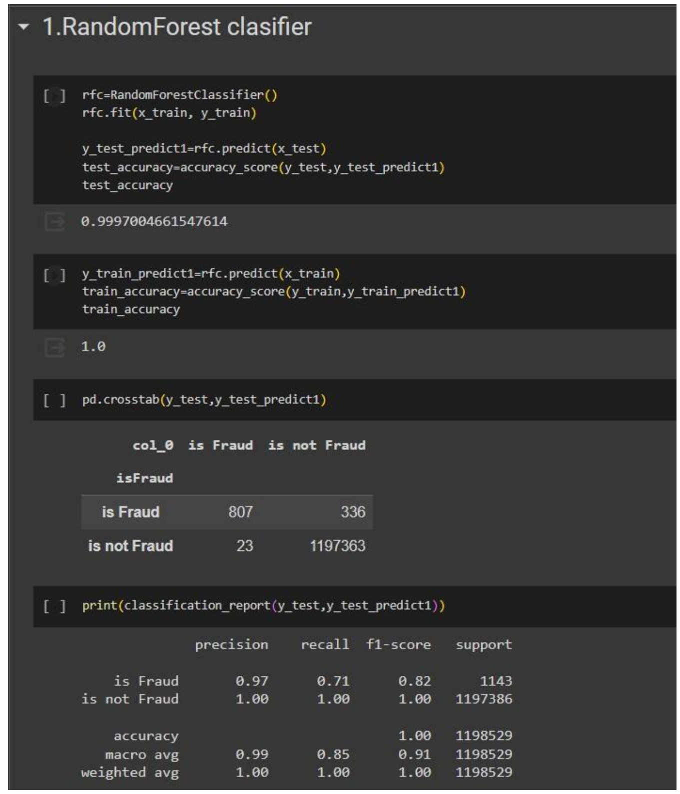
This code sets up a Flask web application for a machine learning model. It loads a pre-trained model from a saved pickle file, provides routes for different pages (about, home, predict), and handles form submission. When a user submits data on the 'predict' page, it passes the input to the model and displays the prediction on the 'result' page. The application runs in debug mode when executed as the main program, enabling web development and testing.

app.py

```python
from flask import Flask, render_template, request
import pickle
import numpy as np
import pandas as pd
model=pickle.load(open('model.pkl','rb'))
app = Flask(__name__)

@app.route("/")
def about():
    return render_template('home.html')

@app.route("/home")
def about1():
    return render_template('home.html')

@app.route("/predict")
def home1():
    return render_template('predict.html')

@app.route('/pred',methods=['POST','GET'])
def pred():
    x=[[obj for obj in request.form.values()]]
    x=np.array(x)
    output=model.predict(x)
    return render_template ("result.html",pred =str(output[0]) )

if __name__ == '__main__':
    app.run(debug=True)
```

8. PERFORMANCE TESTING

Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** Confusion Matrix – Accuracy Score- Classification Report -	

1.RandomForest clasifier

```
[ ] rfc=RandomForestClassifier()
    rfc.fit(x_train, y_train)

    y_test_predict1=rfc.predict(x_test)
    test_accuracy=accuracy_score(y_test,y_test_predict1)
    test_accuracy

    0.9997004661547614

[ ] y_train_predict1=rfc.predict(x_train)
    train_accuracy=accuracy_score(y_train,y_train_predict1)
    train_accuracy

    1.0

[ ] pd.crosstab(y_test,y_test_predict1)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	807	336
is not Fraud	23	1197363

```
[ ] print(classification_report(y_test,y_test_predict1))

              precision    recall  f1-score   support

    is Fraud       0.97      0.71      0.82      1143
is not Fraud       1.00      1.00      1.00   1197386

    accuracy                           1.00   1198529
   macro avg       0.99      0.85      0.91   1198529
weighted avg       1.00      1.00      1.00   1198529
```

2.Decision Tree classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train, y_train)

y_test_predict2=dtc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict2)
test_accuracy
```

```
0.9996912882374978
```

```
y_train_predict2=dtc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict2)
train_accuracy
```

```
1.0
```

```
pd.crosstab(y_test,y_test_predict2)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1193	245
is not Fraud	204	1496519

```
print(classification_report(y_test,y_test_predict2))
```

```
              precision    recall  f1-score   support

    is Fraud       0.85      0.83      0.84      1438
is not Fraud       1.00      1.00      1.00   1496723

    accuracy                           1.00   1498161
   macro avg       0.93      0.91      0.92   1498161
weighted avg       1.00      1.00      1.00   1498161
```

			4 Xgboost Classifier
			[ ] `import xgboost as xgb`
			`xgb1 = xgb.XGBClassifier()`
			`xgb1.fit(x_train,y_train1)`
			`y_test_predict5=xgb1.predict(x_test)`
			`test_accuracy=accuracy_score(y_test1,y_test_predict5)`
			`test_accuracy`
			0.9997904401680998
			[ ] `y_train_predict5=xgb1.predict(x_train)`
			`train_accuracy=accuracy_score(y_train1,y_train_predict5)`
			`train_accuracy`
			0.9998602933377643
			[ ] `pd.crosstab(y_test1,y_test_predict5)`
			col_0 0 1
			row_0
			0 642 172
			1 32 972623
			[ ] `print(classification_report(y_test1,y_test_predict5))`
			precision recall f1-score support
			0 0.95 0.79 0.86 814
			1 1.00 1.00 1.00 972655
			accuracy 1.00 973469
			macro avg 0.98 0.89 0.93 973469
			weighted avg 1.00 1.00 1.00 973469

2.	Tune the Model	Hyper parameter Tuning -	The accuracy for the model is high without hyper parameter tuning and the type 2 error is also very low.

9. RESULTS

We successfully implemented multiple different Machine Learning Algorithms on the given dataset to determine which approach to use for our product. We implemented Random Forest, Decision Trees, SVM classifier, XgBoost classifier and obtained accuracies of 99.97, 99.96, 80 and 99.97 respectively.

*Hence, we concluded that the model which is best fit for the given dataset is **99.97** which is given by **Random Forest***

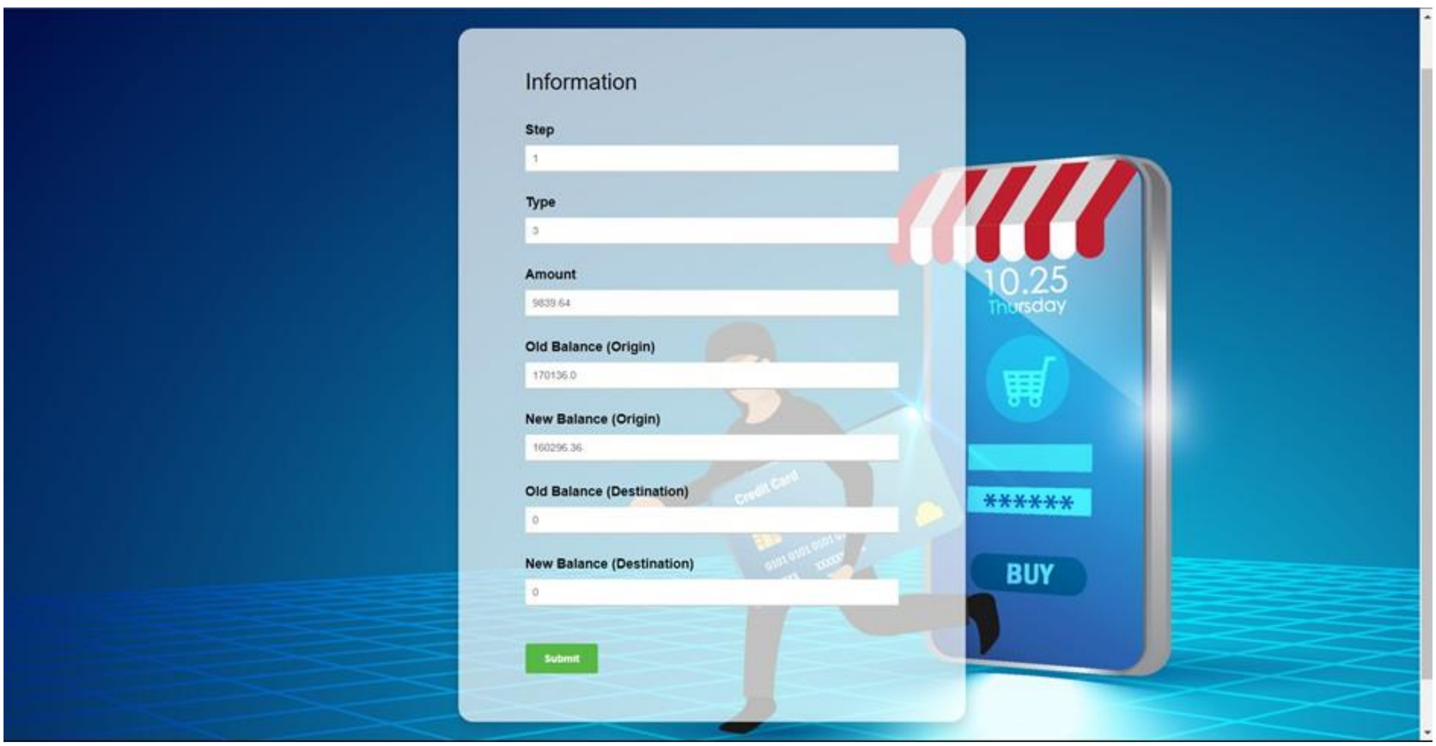
Form Page (User Input):

Form: Display a form with fields for users to input information related to a transaction for fraud detection:

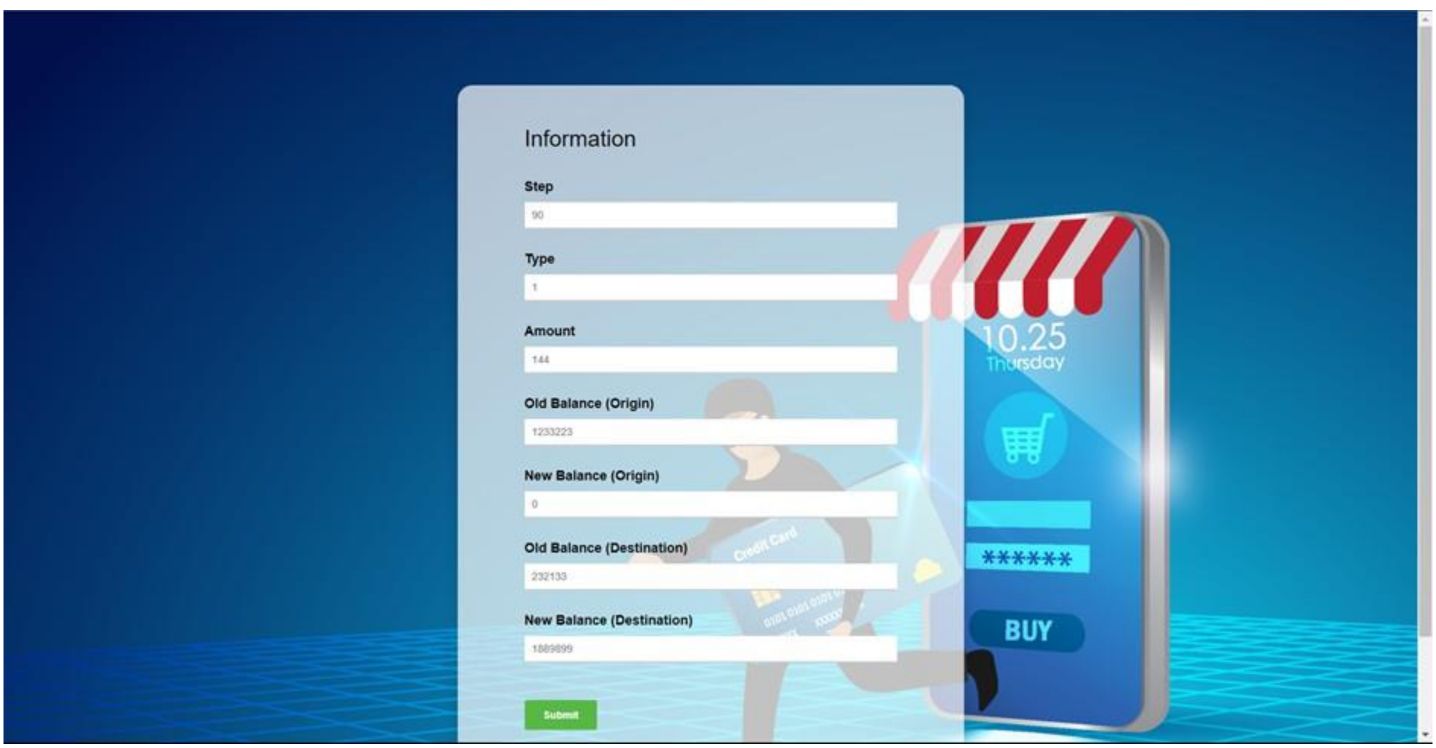
1. 'step': [Input Field]
2. 'type': [Input Field]
3. 'amount': [Input Field]
4. 'oldbalanceOrg': [Input Field]
5. 'newbalanceOrig': [Input Field]

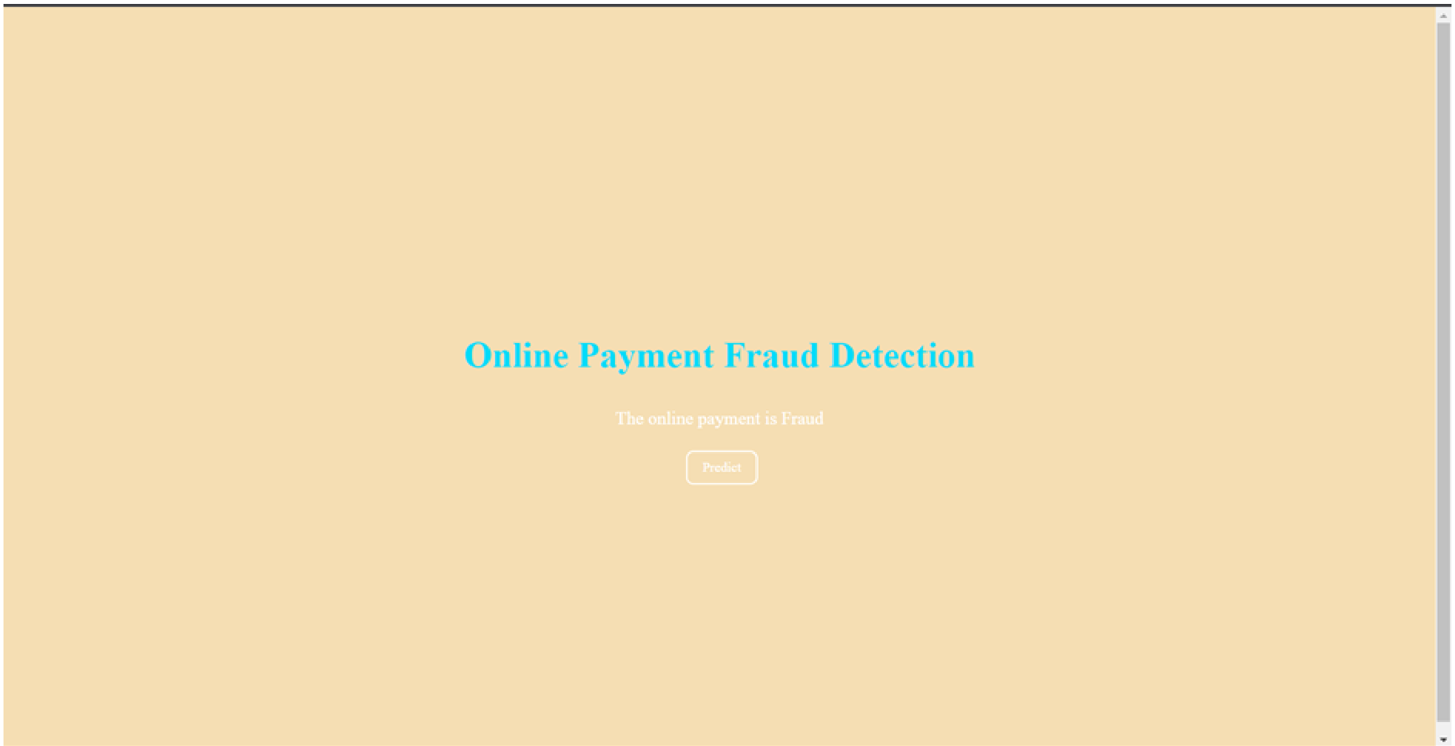
6. 'oldbalanceDest': [Input Field]
7. 'newbalanceDest': [Input Field]
8. Submit Button: Provide a "Submit" button that allows the user to submit the transaction details for fraud detection.

Result Display: After submitting the form, display the result on the same page. It may indicate whether the provided transaction information is classified as potential fraud or not. For example:
"Result: No Fraud Detected" or "Result: Possible Fraud Alert"



When the system classifies a transaction as "Not a Fraud," it means that the provided transaction details do not exhibit suspicious or fraudulent behavior.
Users can be reassured that the transaction appears legitimate, and they can proceed with confidence.





When the system classifies a transaction as "Fraud," it indicates that the provided transaction details raise suspicions of fraudulent activity.

Users should be alerted to the potential risk and advised to take immediate action to secure their accounts and prevent further damage.

10. ADVANTAGES & DISADVANTAGES

Advantages

1. Improved Security:

The primary advantage of implementing an online payment fraud detection system is enhanced security. It helps in identifying and preventing fraudulent transactions, reducing financial losses, and safeguarding the reputation of businesses.

2. Real-Time Detection:

The system can detect fraudulent activities in real-time, allowing for immediate response and mitigation, which is crucial in preventing unauthorized transactions.

3. Cost Savings:

By preventing fraudulent transactions, businesses can save money that would have otherwise been lost to chargebacks, refunds, or lost goods.

4. Customer Trust:

Effective fraud detection systems ensure that legitimate transactions are not declined, leading to higher customer satisfaction and trust.

5. Efficiency:

Automation of fraud detection processes reduces the workload on fraud analysts and minimizes the need for manual reviews of transactions.

6. Customization:

Businesses can tailor the system to their specific needs, adjusting parameters, rules, and machine learning models to adapt to changing fraud patterns.

7. Compliance:

It helps businesses comply with industry and regulatory standards, such as Payment Card Industry Data Security Standard (PCI DSS) requirements.

8. Data Analysis:

The system generates valuable data and reports, offering insights into fraud trends and patterns that can be used to strengthen security measures.

Disadvantages

1. False Positives:

One of the significant challenges is the potential for false positives, where legitimate transactions are incorrectly flagged as fraudulent, leading to customer frustration and loss of revenue.

2. Resource Intensive:

Implementing and maintaining a robust fraud detection system can be resource-intensive. It requires skilled personnel, computing resources, and continuous monitoring and updates.

3. Complexity:

These systems can be complex and challenging to set up and configure correctly, which may require a learning curve for users.

4. Cost of Implementation:

Initial setup and ongoing costs for software, hardware, and personnel can be substantial.

5. Adaptation to New Threats:

Fraudsters continuously evolve their tactics, making it a constant challenge to adapt the system to new and emerging threats.

6. Data Privacy Concerns:

Collecting and processing sensitive customer data for fraud detection can raise privacy concerns and require stringent data protection measures.

7. Downtime:

Maintenance or updates to the system may result in temporary downtime, impacting transaction processing.

8. Training:

Training staff to effectively use and manage the system is essential but can be time-consuming.

11. Conclusion

In summary this project tackles the pressing requirement, for detection of payment fraud in the fast expanding e commerce sector. Through the use of machine learning techniques, specifically real time anomaly detection our system strives to offer an effective and forward thinking approach to protecting financial transactions. By strengthening security measures this project plays a role in upholding the trustworthiness of payment systems guaranteeing a safer and more secure environment, for individuals involved in digital financial transactions.

We successfully implemented multiple different Machine Learning Algorithms on the given dataset to determine which approach to use for our product. We implemented Random Forest, Decision Trees, SVM classifier, XgBoost classifier and obtained accuracies of 99.97, 99.96, 80 and 99.97 respectively.

*Hence, we concluded that the model which is best fit for the given dataset is **99.97** which is given by **Random Forest***

(Scroll Down)

12. FUTURE SCOPE

1. Phishing Scams:

In a common phishing scam, individuals receive seemingly legitimate emails or messages that lead them to fake websites designed to steal their login credentials or credit card information. Such scams have led to unauthorized transactions and identity theft.

2. Credit Card Skimming:

Criminals install skimming devices on ATMs or card readers at gas stations and retail stores. These devices capture card details and PINs, which are then used to make unauthorized purchases or withdrawals.

3. Unauthorized Subscription Charges:

Some businesses may engage in unethical practices by signing up users for subscriptions without their knowledge or consent, resulting in recurring charges to their credit cards.

4. Seller Fraud in Online Marketplaces:

In e-commerce platforms, fraudulent sellers may list products that don't exist or send counterfeit goods after receiving payment. This leaves buyers at a loss with no way to get their money back.

5. Stolen Payment Information:

Hackers breach the security of organizations, gaining access to vast databases of credit card information. This information is then sold on the dark web or used to make unauthorized purchases.

6. Ransomware Attacks:

Ransomware attacks can encrypt a victim's data and demand a ransom to provide the decryption key. These attacks can target individuals or businesses, causing significant financial harm.

13. APPENDIX

A brief overview of the history of credit card fraud and how it has evolved over the years. This could include notable cases or changes in fraudulent techniques.

Regulatory Landscape:

An outline of key regulations and standards in the financial industry related to fraud prevention and data security. This might include compliance with Payment Card Industry Data Security Standard (PCI DSS) or other relevant regulations.

Common Types of Credit Card Fraud:

A section detailing various types of credit card fraud, such as identity theft, account takeover, or skimming. Providing examples and characteristics of each type can enhance understanding.

Impact of Credit Card Fraud:

Information on the financial impact of credit card fraud on both individuals and businesses. Discussing the costs involved in fraud prevention and recovery.

Emerging Trends and Threats:

Highlighting current trends and emerging threats in credit card fraud. This could include discussions on phishing, social engineering, or other tactics employed by fraudsters.

Customer Education and Awareness:

Discussing the importance of educating credit card users about potential fraud risks and best practices for securing their financial information.

Technological Advances in Fraud Prevention:

Briefly touching on technological advancements beyond machine learning, such as the use of biometrics, tokenization, or multi-factor authentication in enhancing overall security.

International Collaboration:

Highlighting international efforts and collaborations among financial institutions, law enforcement agencies, and cybersecurity organizations to combat global credit card fraud.

We use the following research papers and articles as inspirations for our project

[1] - Yuan Gao, Shuang Liu, Yuan Zhou, Fei Shen, Xiao Zhang, "An Empirical Study on Machine Learning Techniques in Online Payment Fraud Detection," published in the Journal of Big Data, Volume 7, Issue 2, pp. 277-293 in 2020.

[2] - Oluwatobi Adediji, Gani Alani, "Machine Learning-Based Online Payment Fraud Detection System: A Literature Review," featured in IEEE Access, Volume 7, 2019.

Real-time Credit Card Fraud Detection Using Machine Learning | IEEE Conference Publication | IEEE Xplore

Thank You

Team Members (Team LTVIP2026TMIDS82036)

1. Abhishek Lellapalli
2. Adilakshmi Kuracha
3. Allam Phaneendra
4. Anjali Noolu