ONLINE PAYMENTS FRAUD DETECTION USING MACHINCE LEARNING

AN INDUSTRY ORIENTED MINI REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

Submitted By

NUKA THANUJA	**21UK1A6680**
YENAGANDULA SUMANTH	**22UK5A6612**
NUKA AKSHAYA	**21UK1A6679**
MOHAMMAD RIYAZ	**22UK5A6609**

Under the guidance of
Mr. T. Sanath kumar
Assistant Professor

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JUNTH,HYDRABAD,

BOLLIKUNTA,WARANGAL (T.S) –506005

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

VAAGDEVI ENGINEERING COLLEGE (WARANGAL)

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED MINI PROJECT

This is to certify that the MINI PROJECT entitled "**ONLINE PAYMENTS FRAUD DETECTION USING MACHINE LEARNING**" is being submitted by **NUKA THANUJA (21UK1A6680), YENAGANDULA SUMANTH (22UK5A6612),NUKA AKSHAYA(21UK1A6679),MOHAMMAD RIYAZ (22UK5A6609)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering (Artificial Intelligence and Machine Learning) to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023- 2024.

Project Guide	**HOD**
Mr. T. Sanath Kumar	**Dr. Rekha Gangula**
(Assistant Professor)	(Associate Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr.SYED MUSTHAK AHMED,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this Mini Project in the institute.

We extend our heartfelt thanks to **Dr.REKHA GANGULA,** Head of the Department of CSE (AI&ML), Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the Mini Project

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the Mini Project and for their support in completing the Mini Project.

We express heartfelt thanks to the guide, **T.SANATH KUMAR**, Assistant professor, Department of CSE(AI&ML) for his constant support and giving necessary guidance for completion of this Mini Project.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the Mini Project.

NUKA THANUJA (21UK1A6680)
YENAGANDULA SUMANTH (22UK5A6612)
NUKA AKSHAYA (21UK1A6679)
MOHAMMAD RIYAZ (22UK5A6609)

ABSTRACT

Online payment systems are integral to financial transactions, but they are also vulnerable to fraud, posing significant risks to both consumers and businesses. This project aims to develop a robust fraud prediction model using machine learning techniques to enhance the security of online payment systems. By leveraging a variety of algorithms, including decision trees, random forests, logistic regression, and neural networks, we analyze historical transaction data to identify patterns indicative of fraudulent activity. The model's effectiveness is evaluated based on its accuracy, precision, recall, and F1 score to ensure reliable detection of fraudulent transactions. Key features such as transaction amount, location, time, and user behavior are examined to optimize the model's performance. This research not only aims to advance the understanding of fraud detection mechanisms but also provides a scalable solution that financial institutions can deploy to protect their customers and reduce financial losses. Through rigorous testing and validation, the proposed system demonstrates its potential to significantly mitigate the risk of online payment fraud.

TABLE OF CONTENTS:-

1.INTRODUCTION

1.1 OVERVIEW

Online payments fraud detection using machine learning is a vital component of secure digital transactions. Machine learning algorithms are trained on historical data to recognize patterns and anomalies that may indicate fraudulent activity. These algorithms analyze various factors, including transaction amount and frequency, user behavior and location, device and browser information, and payment method and credentials. The goals of these algorithms are to identify high-risk transactions, flag potential fraud in real-time, reduce false positives and negatives, and continuously improve detection accuracy through ongoing training and learning. Techniques used include supervised learning, unsupervised learning, and neural networks. By leveraging machine learning, online payment systems can significantly improve their ability to detect and prevent fraud, enhancing the security and trustworthiness of digital transactions.

1.2 PURPOSE

The primary purpose of online payments fraud detection using machine learning is to identify and prevent fraudulent transactions in real-time, thereby protecting consumers, merchants, and financial institutions from financial losses. By leveraging machine learning algorithms and techniques, such as predictive modeling, anomaly detection, and clustering, online payment systems can analyze vast amounts of transactional data, recognize patterns, and detect suspicious activity that may indicate fraudulent behavior. This enables online payment systems to flag potentially fraudulent transactions for review, preventing unauthorized transactions and minimizing financial losses. Moreover, machine learning-based fraud detection systems can adapt to emerging patterns and trends in fraudulent activity, ensuring that online payment systems stay ahead of increasingly sophisticated cyber threats and fraudulent activities. Furthermore, the accuracy and efficiency of machine learning-based fraud detection systems can help reduce false positives, which can lead to unnecessary declines of legitimate transactions and damage to customer relationships.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

- Data Quality and Quantity: High-quality, labeled data is crucial for training effective machine learning models. However, obtaining such data can be difficult due to privacy concerns and the rarity of fraudulent transactions compared to legitimate ones.
- Adaptive Fraudsters: Fraudsters are constantly evolving their tactics to bypass detection systems. This means that models need to be continuously updated and retrained to keep up with new patterns of fraudulent behavior.
- False Positives: Balancing the detection of actual fraud while minimizing false positives (legitimate transactions flagged as fraudulent) is a significant challenge. High false positive rates can lead to customer dissatisfaction and increased operational costs.
- Real-Time Processing: Online payment systems require real-time or near-real-time fraud detection to prevent fraudulent transactions from being completed. This demands highly efficient algorithms and infrastructure.
- Feature Engineering: Identifying the most relevant features (variables) that can indicate fraudulent activity is a complex task. Effective feature engineering is essential for building accurate models.
- Regulatory Compliance: Ensuring that fraud detection systems comply with various regulatory requirements and standards can be challenging, especially when operating across multiple jurisdictions.
- Integration with Existing Systems: Integrating new machine learning models with existing payment systems and workflows can be technically challenging and resource-intensive.
- Behavioral Biometrics: Incorporating behavioral biometrics (e.g., typing patterns, mouse movements) to enhance fraud detection adds another layer of complexity but can significantly improve detection accuracy

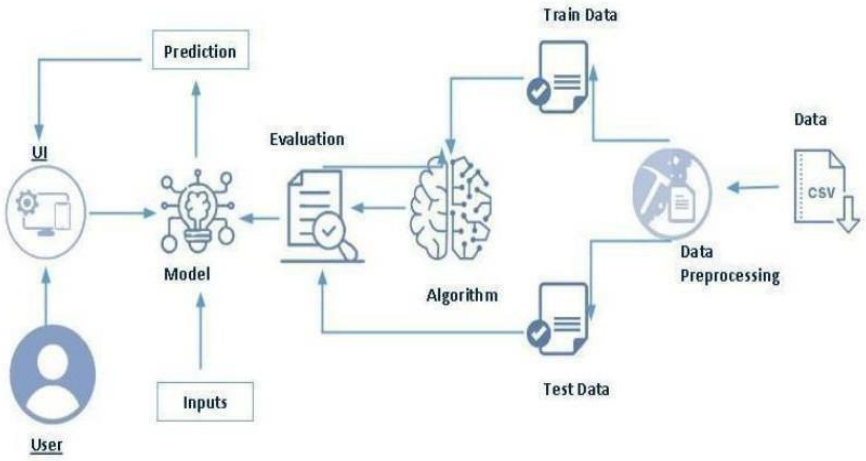
2.2 PROPOSED SOLUTION

Here are some proposed solutions for online fraud detection using machine learning:

- **Hybrid approach:** Combine machine learning algorithms with rule-based systems to leverage the strengths of both.
- **Ensemble learning:** Use multiple models to improve detection accuracy and reduce false positives/negatives.
- **Anomaly detection:** Identify unusual patterns and behavior to detect potential fraud.
- **Real-time processing:** Utilize streaming data processing and real-time analytics to detect fraud as it occurs.
- **Autoencoders**: Use autoencoders to identify abnormal transaction patterns.
- **Graph-based methods:** Analyze transaction graphs to detect suspicious patterns and relationship
- **Explainable AI:** Implement techniques like SHAP or LIME to provide transparent and interpretable fraud detection.
- **Continuous learning:** Regularly update and retrain models to adapt to evolving fraud patterns.
- **Human-in-the-loop:** Implement active learning to involve human expertise in the fraud detection process.
- **Multi-modal analysis:** Combine machine learning with other fraud detection methods, such as device fingerprinting and behavioral analysis.
- **Cloud-based solutions:** Leverage cloud-based infrastructure to handle large volumes of transactions and scale fraud detection capabilities.
- **Collaborative approaches:** Share data and insights with other organizations to improve fraud detection across industries

3.THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. SOFTWARE DESIGNING

The following is the software required to complete this project:

- **Google Colab:** Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud based Jupyter Notebook environment with access to Python libraries and hardware acceleration.
- **Dataset (CSV File):** The dataset in CSV format is essential for training and testing your predictive model. It should include historical online payments data, longitude, latitude, and other relevant features.
- **Data Preprocessing Tools**: Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.
- **Feature Selection/Drop:** Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn for custom Python code to enhance the model's efficiency.
- **Model Training Tools:** Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered.
- **Model Accuracy Evaluation:** After model training, accuracy and performance

evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict fraud or not based on historical data.

- **UI Baised on Flask Environment:**Flask, a Python web framework, will be used to develop the user interface (UI) for the system.The Flask application will provide a userfriendly platform for users to input location data or view startup success predictions, location, and recommended information.
- Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality.

4.EXPERIMENTAL INVESTIGATION

1. **Data Collection:** Gather a dataset of online payment transactions, including features such as:

   - Transaction amount and currency

   - Payment method (e.g., credit card, PayPal)

   - User location and device information

   - Time and date of transaction

   - Label (fraudulent or legitimate)

2. **Data Preprocessing:**

   - Handle missing values and outliers

   - Normalize and transform data as needed

   - Split data into training (70-80%) and testing sets (20-30%)

3. **Machine Learning Algorithms:**

   - Train and test multiple algorithms, such as:

   - Supervised learning: logistic regression, decision trees, random forest

   - Unsupervised learning: anomaly detection (e.g., One-Class SVM, Local Outlier Factor)

   - Neural networks: convolutional neural networks (CNNs), recurrent neural networks (RNNs)

4. **Evaluation Metrics:**

   - Accuracy

   - Precision

   - recall

   - F1-score

   - Receiver Operating Characteristic (ROC) curve

- Area Under the ROC Curve (AUC-ROC)

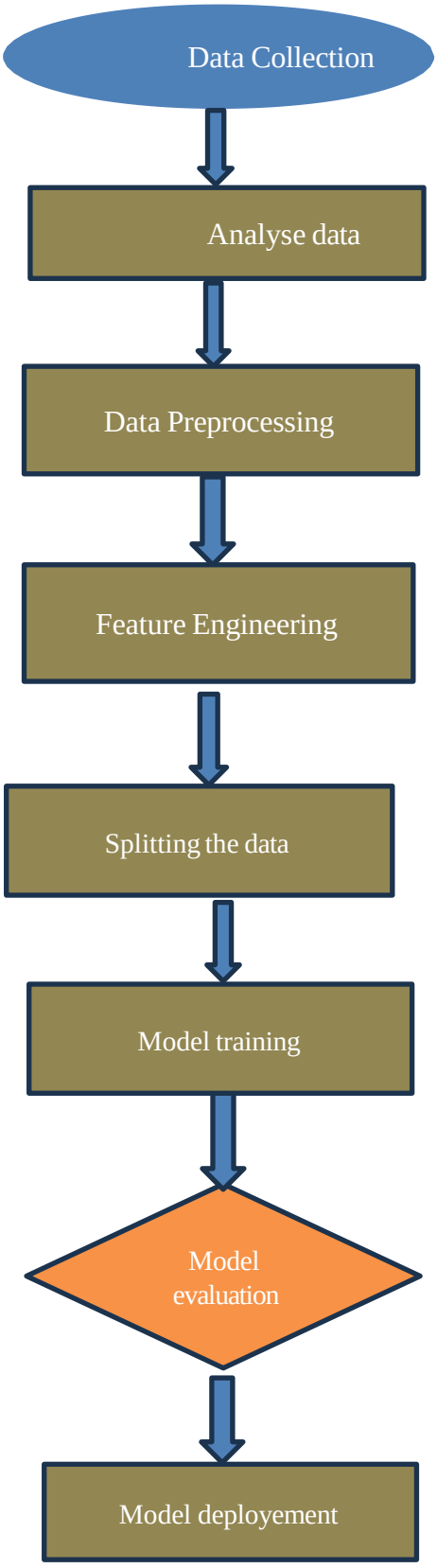
5. **Experimentation:**

   - Train and test each algorithm on the dataset
   - Tune hyperparameters for optimal performance
   - Compare results across algorithms

6. **Results and Analysis:**

   - Present results in tables and figures
   - Analyze performance metrics and discuss strengths and weaknesses of each algorithm
   - Identify the most effective algorithm(s) for online payment fraud detection

5.FLOWCHART


Data Collection
Analyse data
Data Preprocessing
Feature Engineering
Splitting the data
Model training
Model evaluation
Model deployement

6.RESULTS

Figure-6.1 – Home page of online payments fraud detection.

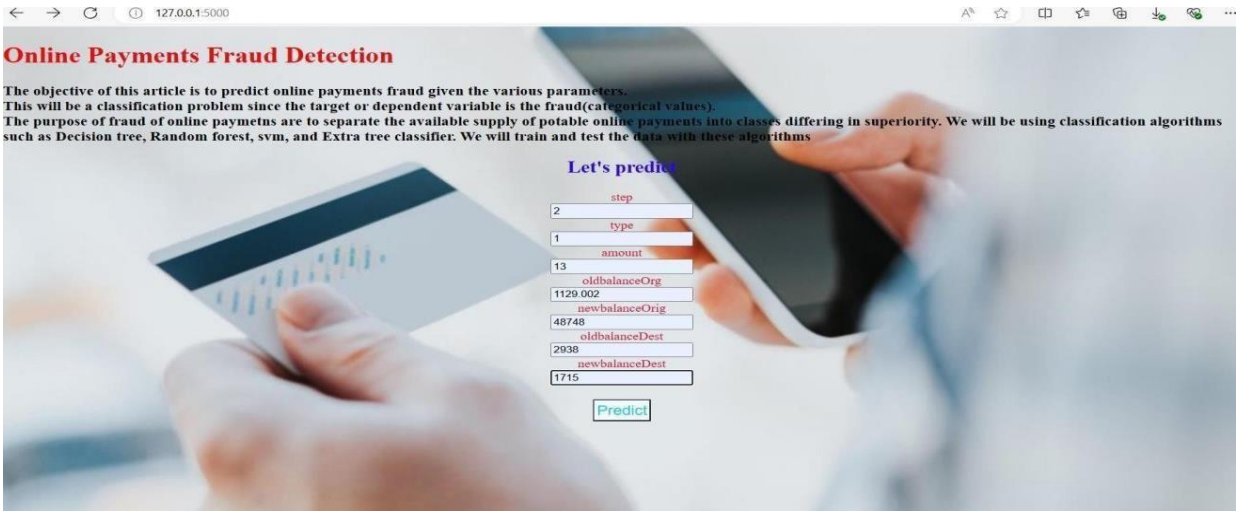


Figure 6.2 -user gives input to Predict.

figure 6.3 - predicted output.

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- Real-time Detection: Machine learning algorithms can analyze data quickly, enabling the detection of fraudulent activities in real-time, reducing the response time to potential threats.
- Adaptability: Machine learning models can adapt and learn from new data, continuously improving their ability to detect evolving fraud patterns effectively.
- Detection of Complex Patterns: Machine learning algorithms can identify intricate and subtle patterns in data that may be challenging for traditional rule-based systems to detect, increasing the accuracy of fraud detection.
- Reduced False Positives: By leveraging machine learning, the system can better distinguish between genuine transactions and fraudulent ones, reducing false positives and minimizing the inconvenience to legitimate customers.
- Continuous Improvement: Machine learning models can evolve and improve over time by learning from new data and adjusting to new fraud tactics, staying ahead of fraudsters' strategies.
- Enhanced Security: Implementing machine learning for fraud detection can enhance the overall security of online payment systems, protecting businesses and customers from financial losses and potential data breaches.

DISADVANTAGES:

- Data Dependency: Machine learning models rely heavily on large datasets to function effectively. The quality and quantity of data available can impact the performance of fraud detection systems. Limited or poor-quality data can result in inaccurate models.
- Complexity and Interpretability: Machine learning models, especially deep learning ones, can be highly complex and difficult to interpret. This lack of transparency can make it challenging to understand why certain transactions are flagged as fraudulent, which is a concern for both regulatory compliance and customer trust.
- High Costs: Developing, training, and maintaining machine learning models can be expensive. This includes costs associated with data collection, computing

power, and hiring skilled personnel.

- Scalability Issues: Implementing machine learning models that need to process vast amounts of transaction data in real-time can be technically challenging and resource-intensive.
- False Positives and Negatives: Striking the right balance between false positives (legitimate transactions incorrectly flagged as fraud) and false negatives (fraudulent transactions not detected) is challenging. Both scenarios can lead to significant issues, such as customer dissatisfaction or financiallosses.

8.APPLICATIONS

- **Risk scoring:** Machine learning models can assign risk scores to transactions or accounts based on various factors, such as transaction amount, location, frequency, and past behavior.
- **Network analysis:** Machine learning algorithms can identify unusual patterns in transactional data and chniques can analyze relationships between entities (users, accounts, or devices) and identify unusual connections or clusters that may indicate fraud.
- **Text analysis:** Machine learning algorithms can analyze unstructured text data, such as emails or social media posts, to identify patterns or keywords that may indicate fraud.
- **Identity verification:** Machine learning models can verify user-provided information, such as identification documents or facial recognition data, to prevent identity theft.
- **Adaptive learning:** Machine learning models can be retrained on new data, enabling them to stay up- to- date and detect emerging fraud patterns.
- **Credit card fraud detection:** Machine learning algorithms can analyze transaction data to identify potentially fraudulent transactions in real time.
- **Point-of-sale (POS) anomaly detection:** Machine learning can monitor POS transactions and identify unusual patterns that may indicate internal fraud or theft.
- **Device fingerprinting:** Machine learning models can analyze device-specific information to detect fraudulent activities, such as account takeovers or multiple accounts linked to a single device.
- **Behavioral biometrics:** Machine learning can analyze user behavior patterns, such as typing speed or swipe gestures, to verify the user's identity and detect anomalies that may suggest fraud.
- **Account takeover prevention:** Machine learning can monitor user login patterns and detect unusual activities that may indicate an account takeover attempt.
- **Friendly fraud detection:** Machine learning can identify patterns related to friendly fraud, in which customers make a purchase and later claim the transaction was unauthorized or they never received the product.

9.CONCLUSION

The implementation of machine learning in online payment fraud detection marks a significant advancement in safeguarding digital financial transactions. Machine learning algorithms excel in analyzing extensive datasets and identifying patterns that signify fraudulent activities, often in real time. This technological shift has resulted in more precise detection and prevention measures, thereby reducing financial losses and enhancing the security of online payments. However, the deployment of machine learning for fraud detection is not without challenges. Data dependency, the complexity of models, high operational costs, scalability issues, and the constant evolution of fraud tactics necessitate ongoing adaptation and improvement. Additionally, balancing the trade-off between false positives and false negatives, ensuring regulatory compliance, and addressing ethical concerns around data usage remain critical considerations. Despite these hurdles, the proactive and dynamic nature of machine learning makes it an indispensable tool in the continuous battle against online payment fraud. The future of fraud detection lies in the collaborative efforts of technology developers, financial institutions, and regulatory bodies to refine these systems, ensuring they remain robust, adaptive, and effective in an ever-evolving digital landscape.

10.FUTURE SCOPE

The future of online payment fraud detection using machine learning is poised for significant advancements as technology continues to evolve. As fraudsters employ increasingly sophisticated techniques, the need for more advanced and adaptive detection methods becomes critical. Future systems will likely incorporate more complex and deep learning models that can analyze vast amounts of data with greater accuracy and speed. Additionally, the integration of behavioral biometrics, such as user interaction patterns and device- specific data, will enhance the ability to detect anomalies that indicate fraudulent behavior. Innovations in real-time processing and edge computing will enable faster and more efficient fraud detection, reducing the window of opportunity for fraudulent transactions to occur. Collaborative efforts between financial institutions, technology developers, and regulatory bodies will be essential to develop standardized frameworks that ensure robust security measures while maintaining user privacy. Furthermore, advancements in explainable AI (XAI) will address the interpretability and transparency challenges of machine learning models, fostering greater trust and compliance with regulatory standards. As machine learning algorithms continue to evolve and improve, the future scope of online payment fraud detection holds the promise of creating a more secure and resilient digital payment ecosystem, capable of staying ahead of emerging threats and ensuring the safety and trust of consumers worldwide.

11.BIBILOGRAPHY

[1] Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A comparison of machine learning techniques for phishing detection. Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit, 60-69.

[2] Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. Proceedings of the World Congress on Engineering, 1-8.

[3] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. IEEE transactions on neural networks and learning systems, 29(8), 3784-3797.

[4] Jurgovsky, J., Granitzer, G., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. Expert Systems with Applications, 100, 234- 245.

[5] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. Decision Support Systems, 50(3), 602-613.

[6] Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. Statistical Science, 235-249.

[7] Wei, W., Dong, Y., Yang, N., Chao, H. C., & Zhou, Q. (2013). A hybrid approach for credit card fraud detection using rough set and decision tree. Journal of Information and Computational Science, 10(9), 2487- 2494.

[8] Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., & Adams, N. M. (2009). Transaction aggregation as a strategy for credit card fraud detection. Data Mining and Knowledge Discovery, 18(1), 30-55.

[9] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y. A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). Scarff: A scalable framework for streaming credit card fraud detection with Spark. Information Fusion, 41, 182- 194.

[10] Phua, C., Lee, V., Smith, K., & Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. arXiv preprint arXiv:1009.6119

12.APPENDIX

Model building:

1. Dataset
2. Google colab &vs code
    1. HTML file
    2. APP.PY

SOURCE CODE:

INDEX.HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width",initial-scale=1.0">
    <link rel="stylesheet" href="style.css" />
    <title>oNLINE PAYMENT FRAUD Detection</title>
    <style> body {background: url("static/2.jpg") center; height: 100%;background-
    position: center; background-size: cover; background-repeat: no-repeat; position:
    sticky;}h1 {color: rgb(236, 11, 11);}
      .btn {margin-top: 20px; padding: 3px; background-color: azure; font-size: larger;
      color: rgb(17, 208, 214); cursor: pointer;}
      form {color: crimsom;align-content: center; text-align: center;}
    </style>
  </head>
  <body>
    <h1>Online Payments Fraud Detection</h1>
  <h3><b>
The objective of this article is to predict online payments fraud given the various
parameters.<br>This will be a classification problem since the target or dependent
variable is the fraud(categorical values).<br>
```

```
</b></h3>
<h2 style="color: rgb(43, 0, 255); text-align: center">Let's predict</h2>
<div class="inputs">
<form action="{{ url_for('predict')}}" method="post">
<label>step</label><br /><input type="text" name="step"placeholder="step"/><br />
<label>type</label><br /><input type="text" name="type"placeholder="type "/><br />
<label>amount</label><br /><input type="text"
name="amount"placeholder="amount"/><br />
<label>oldbalanceOrg</label><br /><input
type="text"name="oldbalanceOrg"placeholder="oldbalanceOrg"/><br />
<label>newbalanceOrig</label><br /><input
type="text"name=”newbalanceOrig"placeholder="newbalanceOri”/><br />
<label>oldbalanceDest</label><br /><input type="text"name="oldbalanceDest"
placeholder="oldbalanceDest"/><br /><label>newbalanceDest</label><br />
<input type="text"name="newbalanceDest" placeholder="newbalanceDest"/><br />
<a href="result.html"><button class="btn" type="submit">Predict</button></a>
</form>
</div>
<br /><br />
<section>
<h3 style="color: blueviolet; text-align: center">
{{ prediction_text }}
</h3>
</section>
</body>
</html>
```

RESULT.HTML:

```
<html>
<head>
<title>Result</title><style> body {background-color: darkgrey;}.output { padding:20px;border: 1px solid red; text-align: center; color: rgb(124, 0, 241); font-style: italic;font-size: larger;}.result { display: block; margin-left: auto;
margin-right:auto; width:50%;}
</style>
    </head>
    <body>
      <h3 class="output">{{ prediction_text }}</h3>
      <img class="result" src="static/fig.jpeg" alt="prediction" width="200" />
    </body>
  </html>
```

APP.PY:

```
import numpy as np
import pickle import
pandas as pd from flask
import Flask,
render_template,
request
app=Flask( name ,static_url_path='/Flask/static')
model = pickle.load(open('C:/Users/Thanuja/Desktop/Mini project/Flask/model.pkl',
'rb'))
@app.route('/') def home() :
return render_template( 'home.html')
@app.route ('/predict',methods=["POST","GET"]) def predict():
if request.method=="POST": Step=float(request.form["Step"])
Type=float(request.form["Type"]) Amount=float(request.form["Amount"])
oldbalanceOrig=float(request.form["oldbalanceOrig"])newbalanceOrig=float(request.
form["newbalanceOrig"]) OldbalanceDest=float (request.form["OldbalanceDest"])
NewbalanceDest=float (request.form["NewbalanceDest"])
features_values=np.array([[Step,Type,Amount,oldbalanceOrig,newbalanceOrig,
OldbalanceDest,New balanceDest]])
df=pd.DataFrame(features_values,columns=['Step','Type','Amount','oldbalanceO
rig','newbalanceOrig'
,'OldbalanceDest','NewbalanceDest']) print(df) prediction=model.predict(df)
print(prediction[0]) result=prediction[0]if prediction[0]==0:
result="The Predict fraud for the online payment is ['is Fraud']"
elif prediction[0]==1: result="The Predict fraud for the online
payment is ['is not Fraud']"
text="Hence,based on calculation:"return
render_template("predict.html",prediction_text=text+str(result)) else: return
render_template("predict.html")
if name =="  main  ":  app.run(debug=True,port=5000)
```

CODE SNIPPETS:

MODEL BUILDING:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
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.ensemble import ExtraTreesClassifier
from sklearn.svm import SVC
import xgboost as xgb
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report,confusion_matrix
import warnings
import pickle
```

Figure 12.1-Importing the required libraries

```
df=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/datasets/onlinefraud.csv')
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	1	0

6362620 rows × 11 columns

Figure 12.2- loading the dataset

```
[ ] df.columns

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

Figure 12.3-total columns present in the data.

```
df.shape

(6362620, 11)
```

Figure 12.4-Provides the number of columns and rows.

```
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0

Figure 12.5- First 5 rows of the dataset.

```
[ ] df.tail()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C776919290	0.00	339682.13	1
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C1881841831	0.00	0.00	1
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C1365125890	68488.84	6379898.11	1
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C2080388513	0.00	0.00	1
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C873221189	6510099.11	7360101.63	1

Figure 12.6-Last 5 rows of the dataset.

```
[ ] df.isnull().sum()

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

Figure 12.7-Number of missing values in the dataset.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 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
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

Figure 12.8:Prints information about the dataframe.

```
# Select only numeric columns before calculating correlation
numeric_df = df.select_dtypes(include=['number'])
numeric_df.corr()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
step	1.000000	0.022373	-0.010058	-0.010299	0.027665	0.025888	0.031578	0.003277
amount	0.022373	1.000000	-0.002762	-0.007861	0.294137	0.459304	0.076688	0.012295
oldbalanceOrg	-0.010058	-0.002762	1.000000	0.998803	0.066243	0.042029	0.010154	0.003835
newbalanceOrig	-0.010299	-0.007861	0.998803	1.000000	0.067812	0.041837	-0.008148	0.003776
oldbalanceDest	0.027665	0.294137	0.066243	0.067812	1.000000	0.976569	-0.005885	-0.000513
newbalanceDest	0.025888	0.459304	0.042029	0.041837	0.976569	1.000000	0.000535	-0.000529
isFraud	0.031578	0.076688	0.010154	-0.008148	-0.005885	0.000535	1.000000	0.044109
isFlaggedFraud	0.003277	0.012295	0.003835	0.003776	-0.000513	-0.000529	0.044109	1.000000

Figure 12.9:Selecting only numeric columns before calculating correlation.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            int64
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(4), object(2)
memory usage: 534.0+ MB
```

Figure 12.10:Used to covert one data type to other type.

```
[ ]  le=LabelEncoder()
     df['nameOrig']=le.fit_transform(df['nameOrig'])
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            int64
 2   amount          float64
 3   nameOrig        int64
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(5), object(1)
memory usage: 534.0+ MB
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            int64
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(4), object(2)
memory usage: 534.0+ MB
```

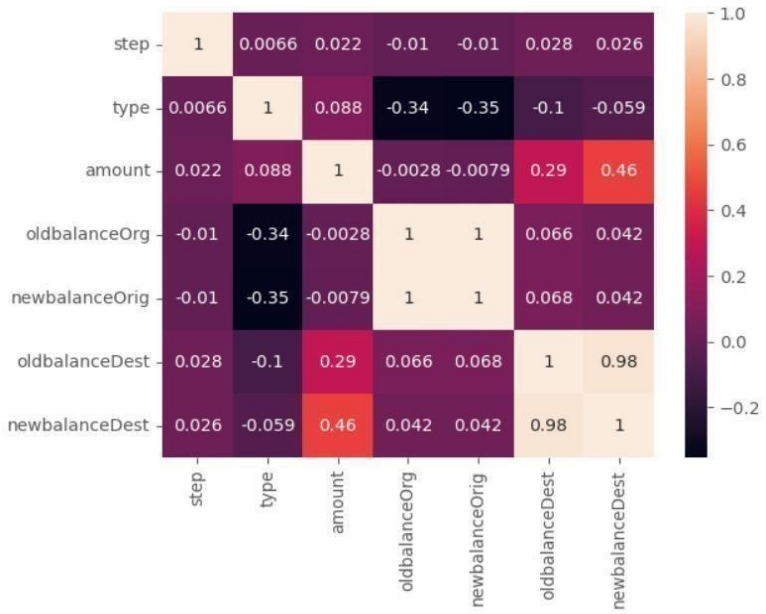


Figure 12.11 Heatmap.

Figure 12.12-Histogram plot

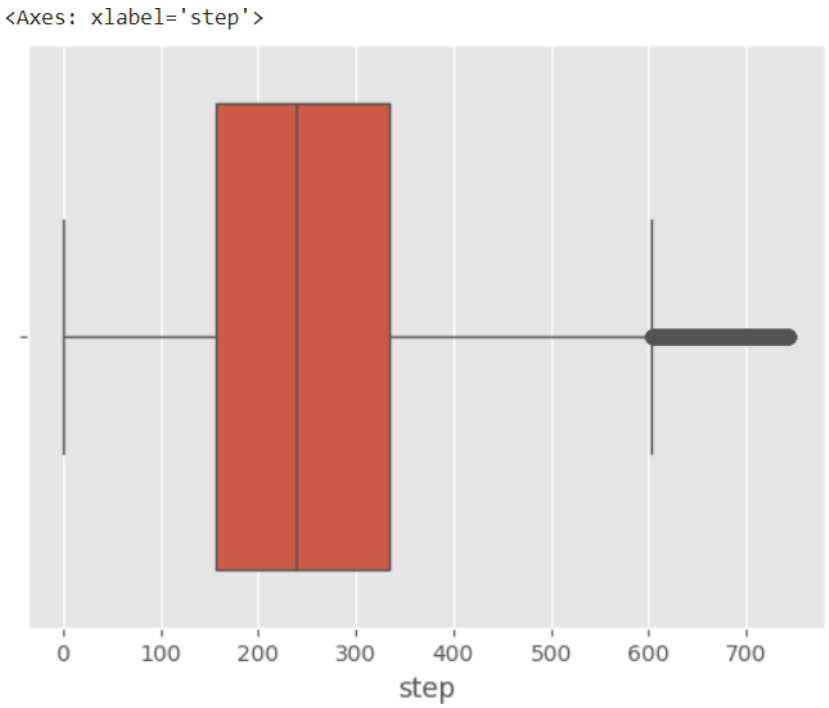


Figure 12.13-Box Plot

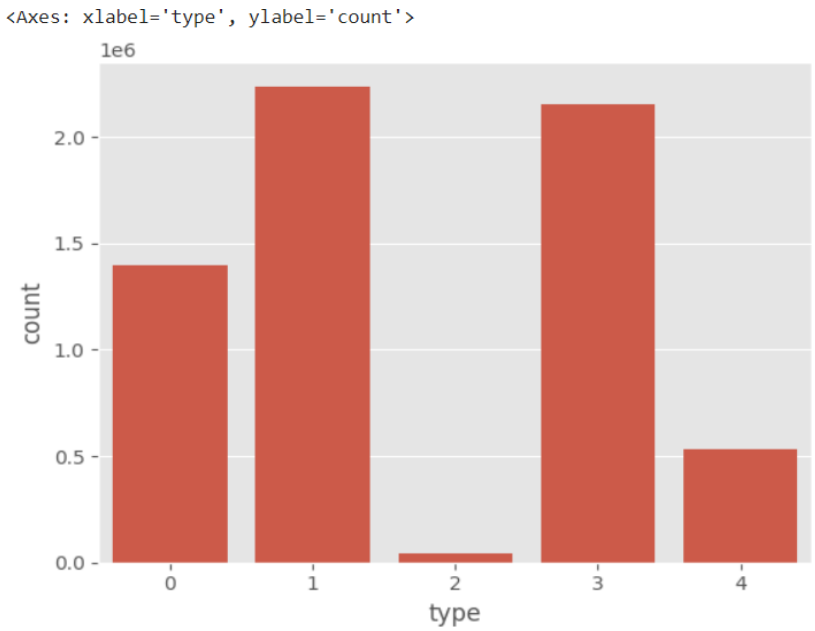


Figure 12.14-Count plot

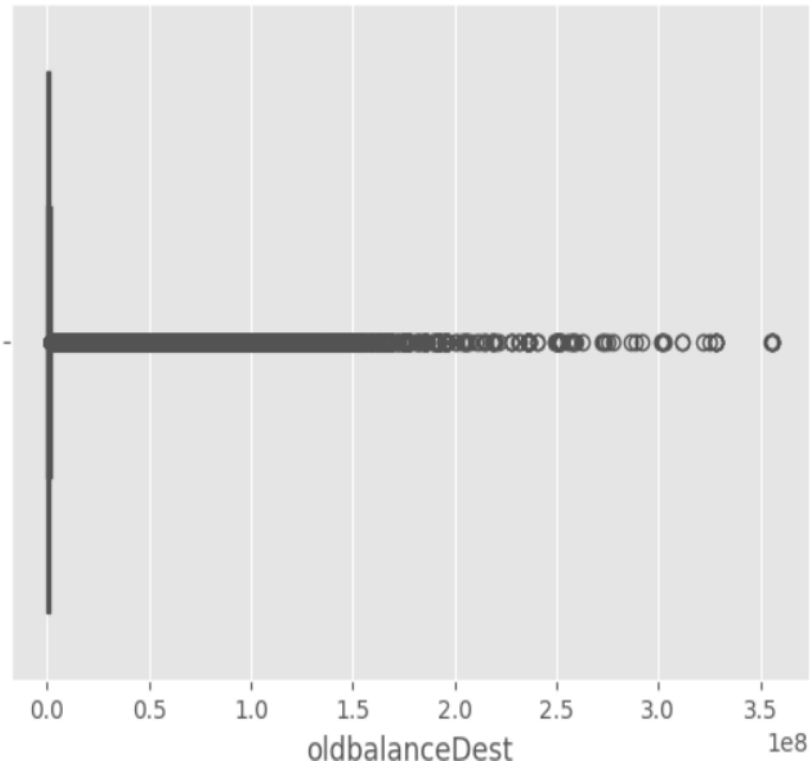
```
df['nameDest'].value_counts()
```

```
nameDest
84652      113
567820     109
472721     105
320660     102
349730     101
          ...
1095075      1
939730       1
1445164      1
1774945      1
319713       1
Name: count, Length: 2722362, dtype: int64
```

Figure 12.15 – count of unique values.

```
sns.boxplot(data=df,x='oldbalanceDest')
```

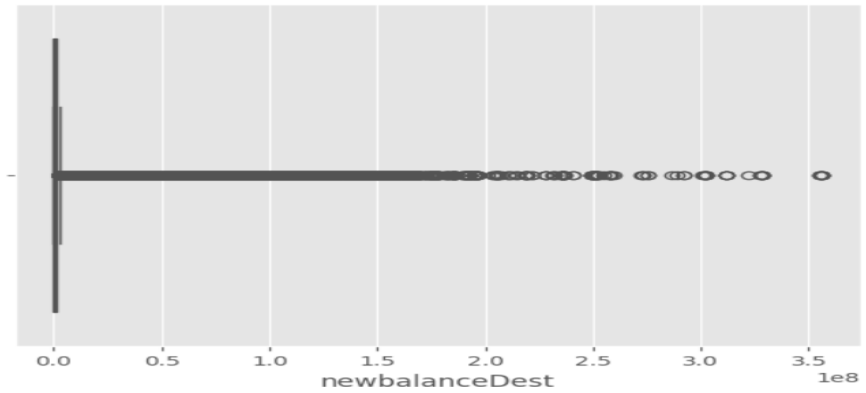
```
<Axes: xlabel='oldbalanceDest'>
```



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

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


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

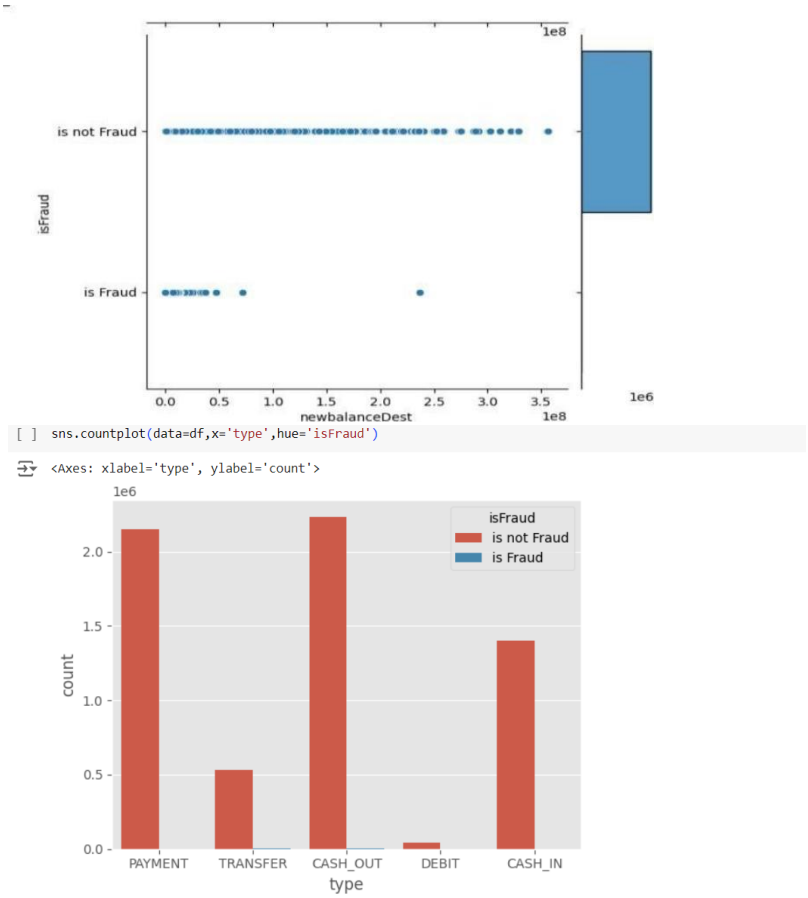


```
[ ]  df['isFraud'].value_counts()

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

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

Figure 12.16-count the number of occurances.




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

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


Figure 12.17-Joint plot

```
sns.boxplot(data=df,x='isFraud',y='amount')
```

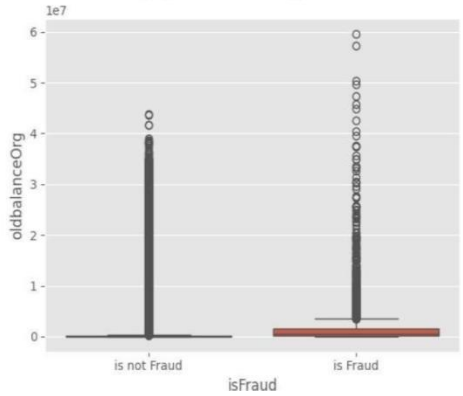
```
<Axes: xlabel='isFraud', ylabel='amount'>
```


```
sns.boxplot(data=df,x='isFraud',y='step')
```

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


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

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



```
sns.boxplot(data=df,x='isFraud',y='newbalanceOrig')
```

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

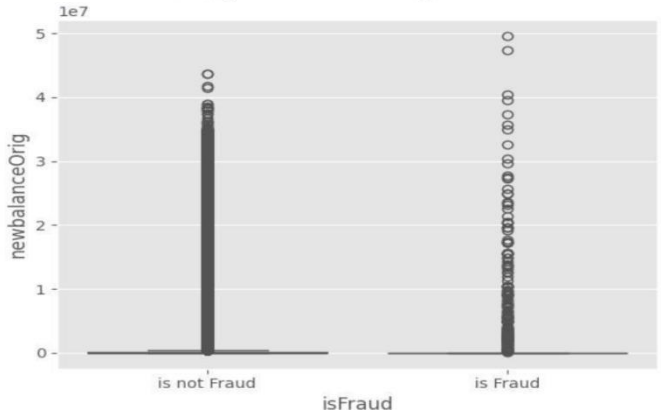


Figure 12.18: Boxplot

```
sns.violinplot(data=df,x='isFraud',y='oldbalanceDest')
```

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

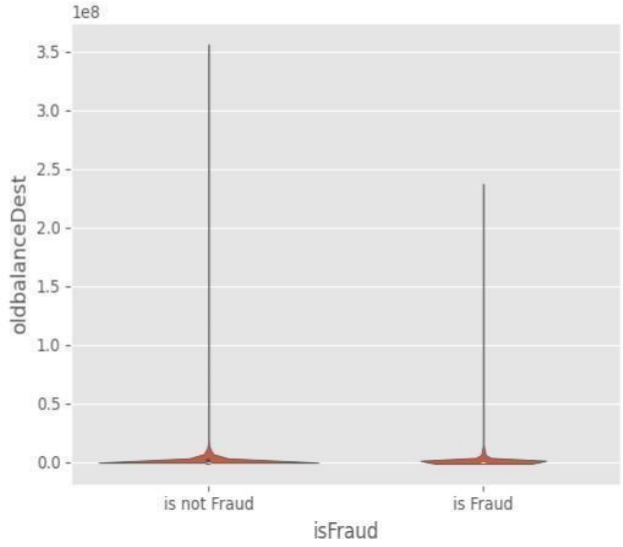


Figure 12.19-Violinplot

```
sns.violinplot(data=df,x='isFraud',y='newbalanceDest')
```

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

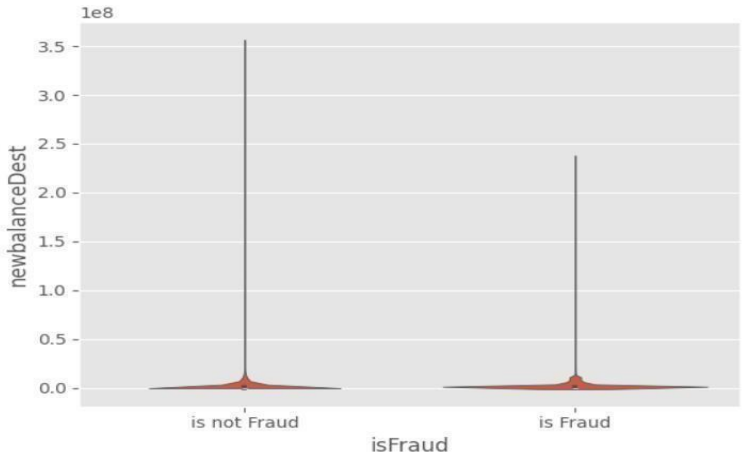


Figure 12.20-Violinplot.

```
[ ] df.describe(include='all')
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	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	6.362620e+06	6.362620e+06	6362620
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	is not Fraud
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6354407
mean	2.433972e+02	1.714150e+00	1.798619e+05	3.176678e+06	8.338831e+05	8.551137e+05	7.464270e+05	1.100702e+06	1.224996e+06	NaN
std	1.423320e+02	1.350117e+00	6.038582e+05	1.834064e+06	2.888243e+06	2.924049e+06	7.502455e+05	3.399180e+06	3.674129e+06	NaN
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	NaN
25%	1.560000e+02	1.000000e+00	1.338957e+04	1.588332e+06	0.000000e+00	0.000000e+00	2.168950e+05	0.000000e+00	0.000000e+00	NaN
50%	2.390000e+02	1.000000e+00	7.487194e+04	3.176672e+06	1.420800e+04	0.000000e+00	4.322890e+05	1.327057e+05	2.146614e+05	NaN
75%	3.350000e+02	3.000000e+00	2.087215e+05	4.765048e+06	1.073152e+05	1.442584e+05	1.132509e+06	9.430367e+05	1.111909e+06	NaN
max	7.430000e+02	4.000000e+00	9.244552e+07	6.353306e+06	5.958504e+07	4.958504e+07	2.722361e+06	3.560159e+08	3.561793e+08	NaN

Figure 12.21-Union of attributes.

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

```
<Axes: ylabel='amount'>
```

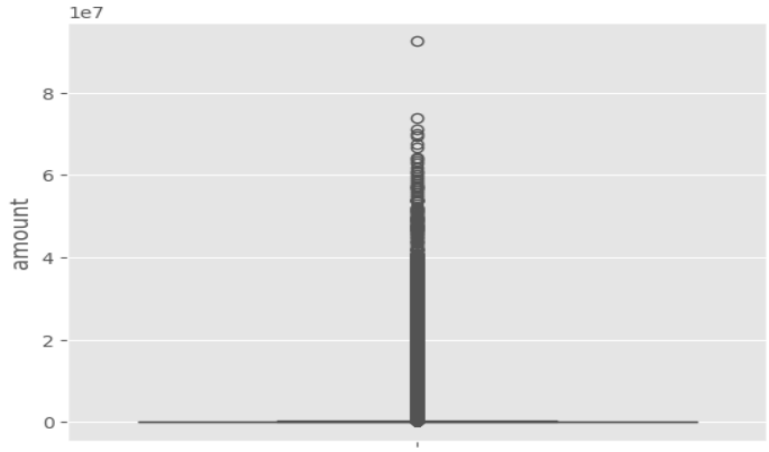


Figure 12.22 -Boxplot

```
from scipy import stats
print(stats.mode(df['amount']))
print(np.mean(df['amount']))
```

```
ModeResult(mode=10000000.0, count=3207)
179861.90354913071
```

Figure 12.23- ModeResult

```
q1=np.quantile(df['amount'],0.25)
q3=np.quantile(df['amount'],0.75)
IQR=q3-q1
upper_bound=q3+(1.5*IQR)
lower_bound=q1-(1.5*IQR)
print('q1:',q1)
print('q3:',q3)
print('IQR:',IQR)
print('Upper Bound:',upper_bound)
print('Lower Bound:',lower_bound)
print('Skewed data:',len(df[df['amount']>upper_bound]))
print('Skewed data:',len(df[df['amount']<lower_bound]))
```

```
q1: 13389.57
q3: 208721.4775
IQR: 195331.9075
Upper Bound: 501719.33875
Lower Bound: -279608.29125
Skewed data: 338078
Skewed data: 0
```

Figure 12.24-Calculates the quantile of the values.

```
def transformationPlot(feature):
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    # Handle potential infinite values
    sns.distplot(feature[np.isfinite(feature)])
    plt.subplot(1,2,2)
    stats.probplot(feature[np.isfinite(feature)], plot=plt)
```

Figure 12.25-Transformation plot code

```
positive_amounts = df['amount'][df['amount'] > 0]
transformationPlot(np.log(positive_amounts))
```

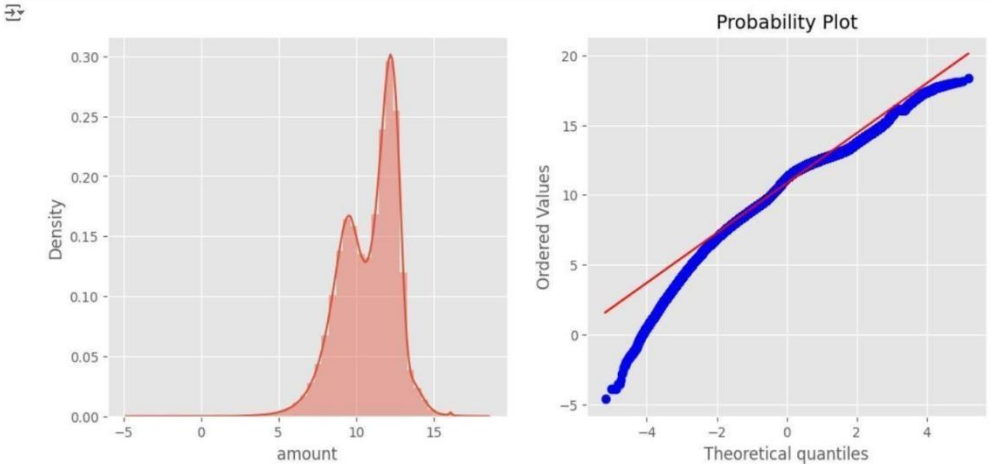


Figure 12.26-Transformation plot.

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

[ ] df['type'].value_counts()

    type
    1    2237500
    3    2151495
    0    1399284
    4     532909
    2      41432
    Name: count, dtype: int64

[ ] x=df.drop('isFraud',axis=1)
    y=df['isFraud']
```

Figure 12.27-Removes the column.

```
x
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFlaggedFraud
0	1	3	9839.64	757869	170136.00	160296.36	1662094	0.00	0.00	0
1	1	3	1864.28	2188998	21249.00	19384.72	1733924	0.00	0.00	0
2	1	4	181.00	1002156	181.00	0.00	439685	0.00	0.00	0
3	1	1	181.00	5828262	181.00	0.00	391696	21182.00	0.00	0
4	1	3	11668.14	3445981	41554.00	29885.86	828919	0.00	0.00	0
...	...	...	...	...	...	...	...	...	...	...
6362615	743	1	339682.13	5651847	339682.13	0.00	505863	0.00	339682.13	0
6362616	743	4	6311409.28	1737278	6311409.28	0.00	260949	0.00	0.00	0
6362617	743	1	6311409.28	533958	6311409.28	0.00	108224	68488.84	6379898.11	0
6362618	743	4	850002.52	2252932	850002.52	0.00	319713	0.00	0.00	0
6362619	743	1	850002.52	919229	850002.52	0.00	534595	6510099.11	7360101.63	0

6362620 rows × 10 columns

```
[ ] y
```

```
0          is not Fraud
1          is not Fraud
2              is Fraud
3              is Fraud
4          is not Fraud
                 ...
6362615        is Fraud
6362616        is Fraud
6362617        is Fraud
6362618        is Fraud
6362619        is Fraud
Name: isFraud, Length: 6362620, dtype: object
```

```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
[ ] print(x_train.shape)
    print(x_test.shape)
    print(y_train.shape)
    print(y_test.shape)
```

```
(5090096, 10)
(1272524, 10)
(5090096,)
(1272524,)
```

Figure 12.28-Split the data into train and test.

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score
```

```
[ ] rfc=RandomForestClassifier()
    rfc.fit(x_train,y_train)
    y_test_predict1=rfc.predict(x_test)
```

```
test_accuracy=accuracy_score(y_test,y_test_predict1)
print(test_accuracy)
```

```
0.999704524236871
```

```
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	1289	352
is not Fraud	24	1270859

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

```
              precision    recall  f1-score   support

    is Fraud       0.98      0.79      0.87      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.99      0.89      0.94   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Figure12.29-Random forest 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
```

```
0.9996785915236176
```

```
    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	1424	217
is not Fraud	192	1270691

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

```
              precision    recall  f1-score   support

    is Fraud       0.88      0.87      0.87      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.94      0.93      0.94   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Figure 12.30-DecisionTree Classifier Model.

```
[ ] from sklearn.ensemble import ExtraTreesClassifier
    etc=ExtraTreesClassifier()
    etc.fit(x_train,y_train)
    y_test_predict3=etc.predict(x_test)
```

```
[ ] test_accuracy=accuracy_score(y_test,y_test_predict3)
    test_accuracy
```

```
0.999628297776702
```

```
    y_train_predict3=etc.predict(x_train)
    train_accuracy=accuracy_score(y_train,y_train_predict3)
    train_accuracy
```

```
1.0
```

```
[ ] pd.crosstab(y_test,y_test_predict3)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1170	471
is not Fraud	2	1270881

```
    print(classification_report(y_test,y_test_predict3))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.71      0.83      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       1.00      0.86      0.92   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Figure 12.31 -Extratreeclassifer model.

```
[ ] df.columns
```

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

```
[ ] from sklearn.preprocessing import LabelEncoder
    le=LabelEncoder()
    y_train1=le.fit_transform(y_train)
```

```
[ ] y_test1=le.transform(y_test)
```

```
[ ] y_test1
```

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

```
[ ] y_train1
```

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

```
[ ] 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
```

```
0.9997705347796977
```

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

```
0.9998668001546532
```

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

col_0	0	1
row_0		
0	1399	242
1	50	1270833

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test1,y_test_predict5))
```

```
              precision    recall  f1-score   support

           0       0.97      0.85      0.91      1641
           1       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.98      0.93      0.95   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Figure 12.32 - xgboost model.

```
[ ] import pickle
    pickle.dump(dtc,open('model.pkl','wb'))
```

```
[ ] from google.colab import files
    files.download('model.pkl')
```

Figure 12.33-Saving the best model in pickle file.