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.

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 purpose of online payments fraud detection using machine learning is to:

1. Prevent financial losses: Identify and block fraudulent transactions in real-time, minimizing financial losses for individuals and businesses.
2. Enhance security: Improve the overall security of online payment systems, protecting sensitive information and preventing unauthorized access.
3. Reduce false positives: Minimize the number of legitimate transactions incorrectly flagged as fraudulent, reducing unnecessary transaction declines and user friction.
4. Improve customer experience: Provide a seamless and secure payment experience, building trust and confidence in online transactions.
5. Comply with regulations: Meet regulatory requirements and industry standards for fraud detection and prevention, avoiding potential penalties and reputational damage.
6. Stay ahead of fraudsters: Continuously adapt and improve fraud detection capabilities to stay ahead of evolving fraud tactics and patterns.
7. Optimize risk management: Make data-driven decisions on risk management, balancing fraud prevention with transaction approval rates.
8. Protect brand reputation: Prevent fraudulent activity from damaging your brand's reputation and undermining customer trust.

By achieving these goals, online payments fraud detection using machine learning helps create a safer and more reliable digital payment ecosystem.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

1. Data quality issues: Noisy, incomplete, or biased data can negatively impact model performance and accuracy.

2. Class imbalance: Fraudulent transactions are often a small percentage of overall transactions, making it challenging for models to detect fraud without generating excessive false positives.

3. Evolving fraud patterns: Fraudsters continuously adapt and change tactics, requiring models to be frequently updated and retrained.

4. Model interpretability: Complex machine learning models can be difficult to interpret, making it challenging to understand why a transaction was flagged as fraudulent.

5. False positives and negatives: Incorrectly flagging legitimate transactions as fraudulent (false positives) or missing actual fraudulent transactions (false negatives) can lead to user frustration and financial losses.

6. Scalability: High transaction volumes can be challenging for models to process in real-time, leading to delays or errors.

7. Regulatory compliance: Models must comply with regulatory requirements and industry standards, such as GDPR and PCI-DSS.

8. Explainability and transparency: Models must provide clear explanations for their decisions, ensuring transparency and trust in the fraud detection process.

9. Human-in-the-loop: Models may require human intervention to review flagged transactions, leading to increased operational costs and potential bottlenecks.

10. Adversarial attacks: Fraudsters may attempt to manipulate models through adversarial attacks, requiring robust defenses to maintain model integrity.

2.2 PROPOSED SOLUTION

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

1. Hybrid approach: Combine machine learning algorithms with rule-based systems to leverage the strengths of both.

2. Ensemble learning: Use multiple models to improve detection accuracy and reduce false positives/negatives.

3. Anomaly detection: Identify unusual patterns and behavior to detect potential fraud.

4. Real-time processing: Utilize streaming data processing and real-time analytics to detect fraud as it occurs.

5. Autoencoders: Use autoencoders to identify abnormal transaction patterns.

6. Graph-based methods: Analyze transaction graphs to detect suspicious patterns and relationships.

7. Explainable AI: Implement techniques like SHAP or LIME to provide transparent and interpretable fraud detection.

8. Continuous learning: Regularly update and retrain models to adapt to evolving fraud patterns.

9. Human-in-the-loop: Implement active learning to involve human expertise in the fraud detection process.

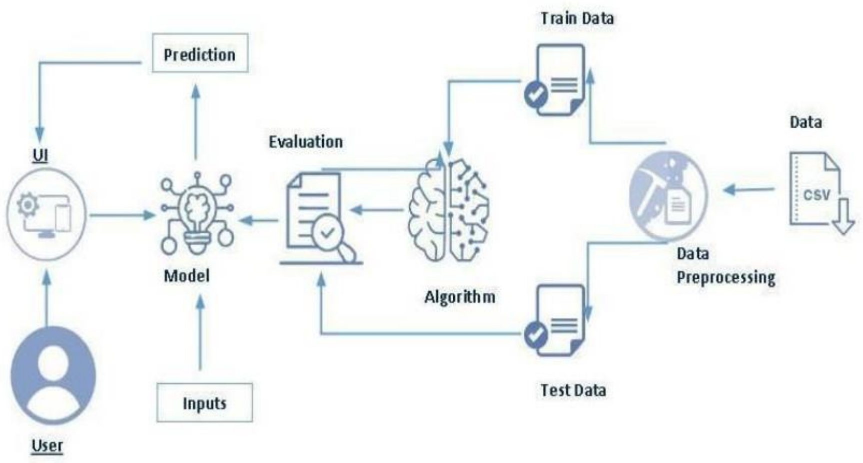
10. Multi-modal analysis: Combine machine learning with other fraud detection methods, such as device fingerprinting and behavioral analysis.

11. Cloud-based solutions: Leverage cloud-based infrastructure to handle large volumes of transactions and scale fraud detection capabilities.

12. 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:

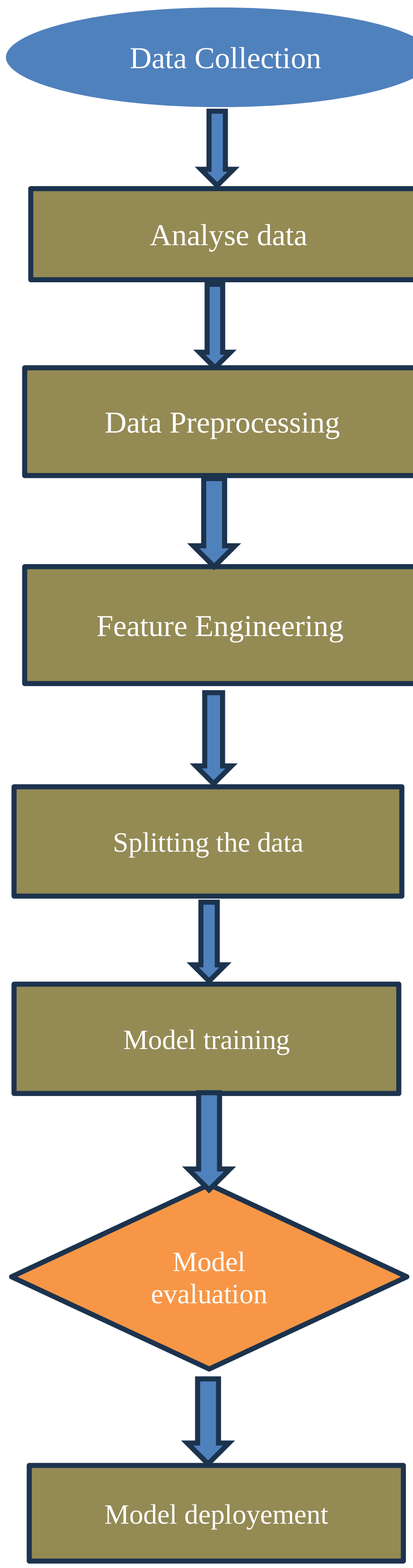
- Data collection
    - Collect the dataset or create the dataset
- Data pre-processing
    - Removing unnecessary columns
    - Checking for null values
- Visualizing and analyzing data
- Univariate analysis
- Bivariate analysis
- Descriptive analysis
- Model building
    - Handling categorical values
    - Dividing data into train and test sets
    - Import the model building libraries
    - Comparing the accuracy of various models
    - Hyperparameter tuning of the selected model

    - Evaluating the performance of models
    - Save the model
- Application Building
    - Create an HTML file
    - Build python code

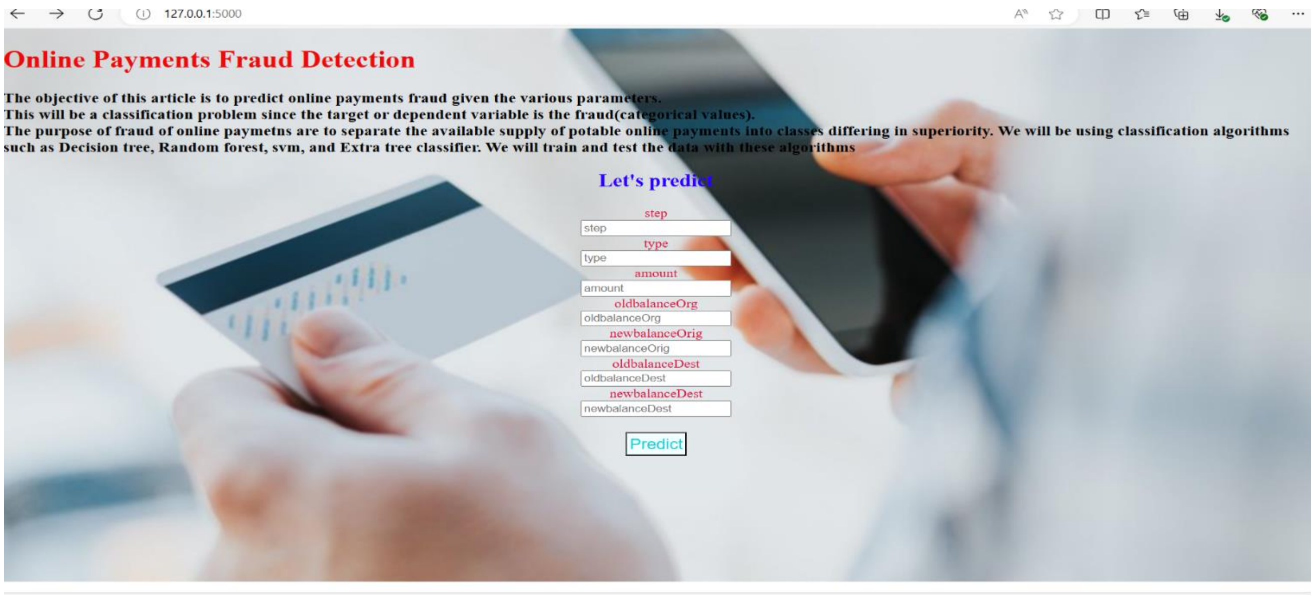
4.EXPERIMENTAL INVESTIGATION

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

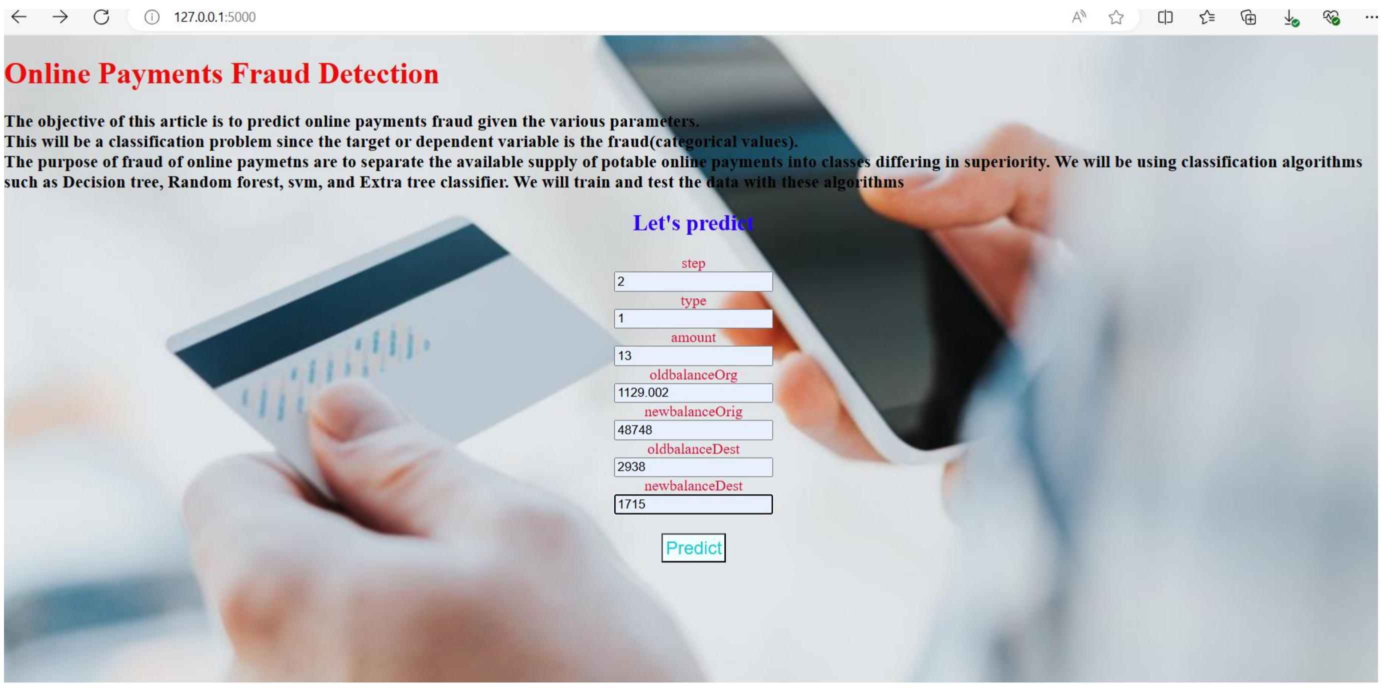

127.0.0.1:5000

Online Payments Fraud Detection

The objective of this article is to predict online payments fraud given the various parameters.
This will be a classification problem since the target or dependent variable is the fraud(categorical values).
The purpose of fraud of online paymetns are to separate the available supply of potable online payments into classes differing in superiority. We will be using classification algorithms such as Decision tree, Random forest, svm, and Extra tree classifier. We will train and test the data with these algorithms

Let's predict

step: step
type: type
amount: amount
oldbalanceOrg: oldbalanceOrg
newbalanceOrig: newbalanceOrig
oldbalanceDest: oldbalanceDest
newbalanceDest: newbalanceDest

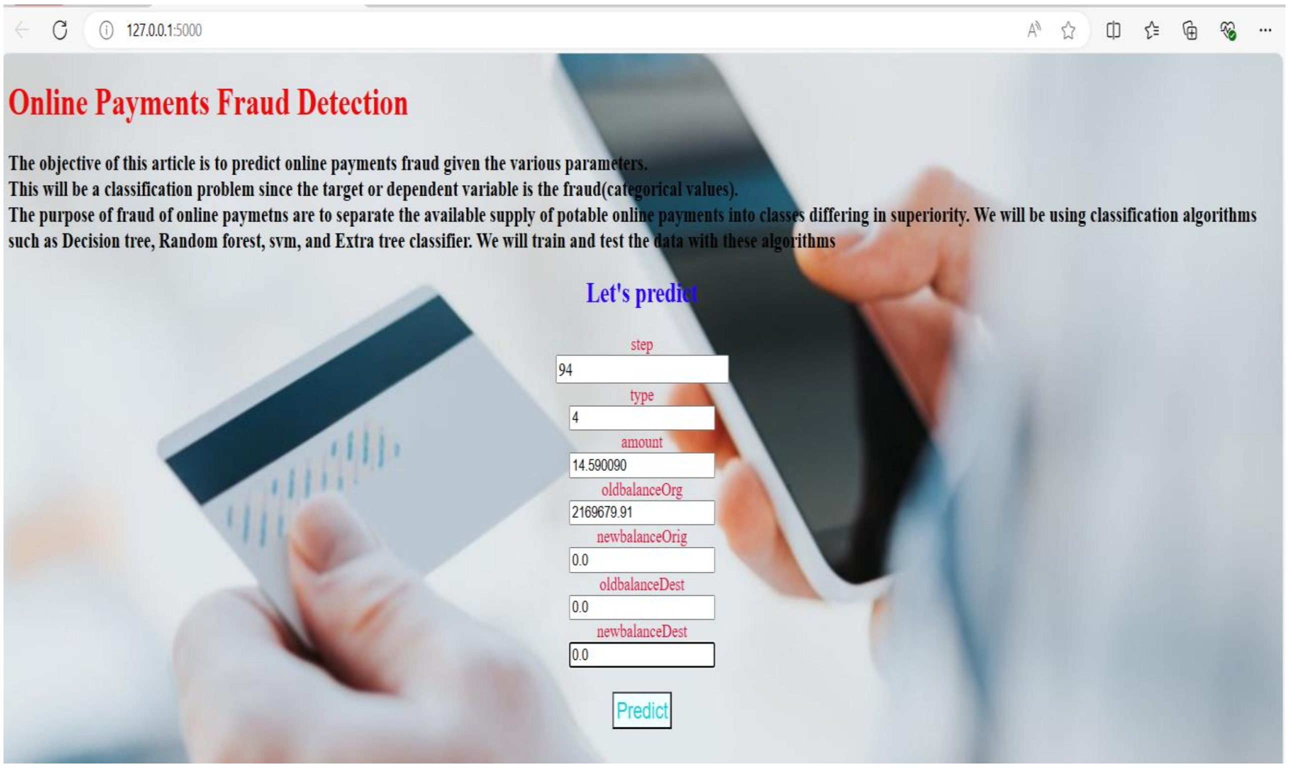

127.0.0.1:5000

Online Payments Fraud Detection

The objective of this article is to predict online payments fraud given the various parameters.
This will be a classification problem since the target or dependent variable is the fraud(categorical values).
The purpose of fraud of online paymetns are to separate the available supply of potable online payments into classes differing in superiority. We will be using classification algorithms such as Decision tree, Random forest, svm, and Extra tree classifier. We will train and test the data with these algorithms

Let's predict

step: 2
type: 1
amount: 13
oldbalanceOrg: 1129.002
newbalanceOrig: 48748
oldbalanceDest: 2938
newbalanceDest: 1715



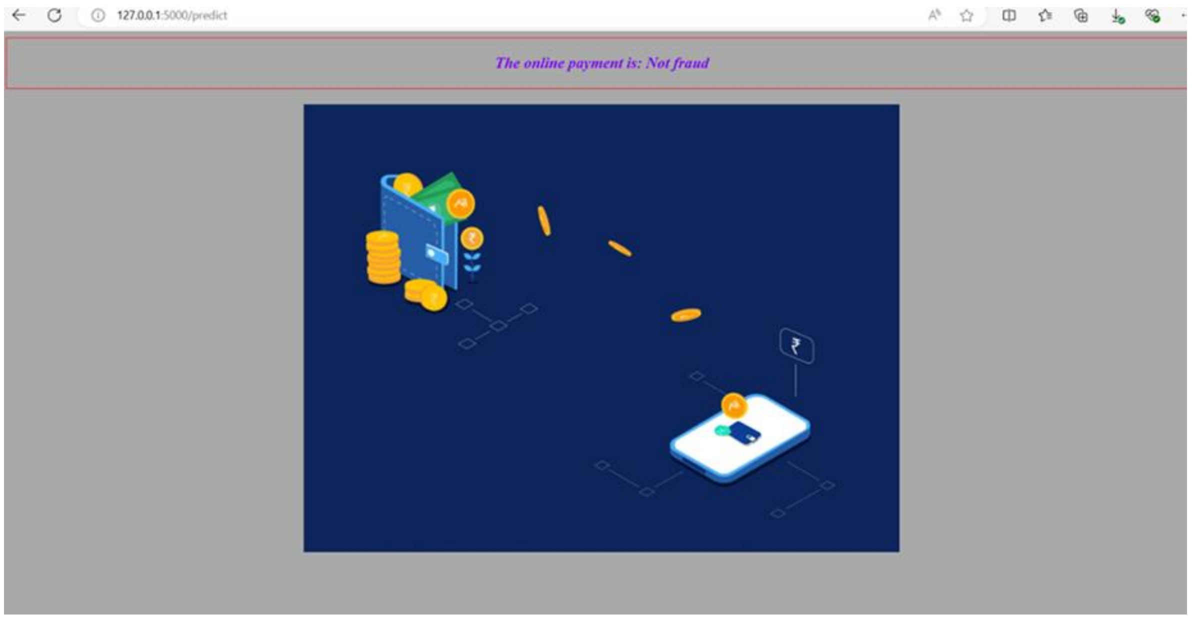
127.0.0.1:5000

Online Payments Fraud Detection

The objective of this article is to predict online payments fraud given the various parameters.
This will be a classification problem since the target or dependent variable is the fraud(categorical values).
The purpose of fraud of online paymetns are to separate the available supply of potable online payments into classes differing in superiority. We will be using classification algorithms such as Decision tree, Random forest, svm, and Extra tree classifier. We will train and test the data with these algorithms

Let's predict

step	94
type	4
amount	14.590090
oldbalanceOrg	2169679.91
newbalanceOrig	0.0
oldbalanceDest	0.0
newbalanceDest	0.0



127.0.0.1:5000/predict

The online payment is: Not fraud

6.ADVANTAGES AND DISADVANTAGES

Advantages :

1. *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.
2. *Adaptability*: Machine learning models can adapt and learn from new data, continuously improving their ability to detect evolving fraud patterns effectively.
3. *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.
4. *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.
5. *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.
6. *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:

1. *Complexity*: Implementing machine learning models for fraud detection can be complex and require specialized knowledge and resources, which may pose challenges for some organizations.
2. *Data Quality*: Machine learning algorithms heavily rely on the quality of data they are trained on. If the data is incomplete, biased, or inaccurate, it can lead to suboptimal fraud detection outcomes.
3. *Overfitting*: Machine learning models may overfit the training data, meaning they perform well on the training data but struggle to generalize to new, unseen data, potentially leading to false positives or negatives.
4. *Interpretability*: Some machine learning algorithms, like deep learning models, are often considered as "black boxes," making it difficult to interpret how they arrive at a particular fraud detection decision, which can be a challenge for regulatory compliance.
5. *Cost*: Developing and maintaining machine learning models for fraud detection can be costly, requiring investment in infrastructure, expertise, and ongoing monitoring and updates.
6. *Security Risks*: While machine learning can enhance security, it can also introduce new security risks. Fraudsters may attempt to manipulate the machine learning models or introduce adversarial attacks to deceive the system.

7.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.
- Invoice fraud detection: Machine learning can analyze invoices and related documentation to identify discrepancies that may indicate fraud.
- Loyalty program fraud detection: Machine learning can monitor customer behavior within loyalty programs to identify and flag potential fraud or abuse.

8.CONCLUSION

Online payments fraud detection using machine learning has shown promising results in identifying and preventing fraudulent transactions. By leveraging machine learning algorithms and techniques, such as supervised learning, unsupervised learning, and deep learning, models can be trained to detect fraudulent patterns and anomalies in transaction data.
The key takeaways from this approach are:
1. Improved accuracy: Machine learning models can achieve high accuracy in detecting fraudulent transactions, reducing false positives and false negatives.
2. Real-time detection: Machine learning models can be deployed in real-time, enabling instant detection and prevention of fraudulent transactions.
3. Adaptive learning: Machine learning models can learn from new data and adapt to evolving fraud patterns, improving their detection capabilities over time.
4. Reduced manual review: Machine learning models can automate the fraud detection process, reducing the need for manual review and minimizing the risk of human error.
5. Enhanced customer experience: By detecting and preventing fraudulent transactions in real-time, machine learning models can help prevent financial losses and enhance the overall customer experience.

However, it's important to note that machine learning models are not without their challenges and limitations. Some of the key challenges include:
1. Data quality: Machine learning models require high-quality data to learn and detect fraudulent patterns. Poor data quality can lead to biased or inaccurate models.
2. Data imbalance: Fraudulent transactions are often a small percentage of overall transactions, making it challenging to train models that can detect rare events.
3. Model interpretability: Machine learning models can be complex and difficult to interpret, making it challenging to understand the reasoning behind their predictions.
4. Model drift: Machine learning models can drift over time, requiring continuous updating and retraining to maintain their detection capabilities.
Overall, machine learning has the potential to revolutionize online payments fraud detection, but it's important to address the challenges and limitations associated with its adoption.

9.FUTURE SCOPE

The future scope of online payments fraud detection using machine learning is promising, with several potential developments and applications:

1. Real-time fraud detection: Machine learning models will become even more accurate and efficient, enabling real-time fraud detection and prevention.
2. Explainable AI: Models will be designed to provide transparent and interpretable explanations for their fraud detection decisions.
3. Graph neural networks: Graph neural networks will be used to analyze transaction relationships and detect complex fraud patterns.
4. Multimodal fraud detection: Models will combine multiple data sources, such as text, images, and transaction data, to detect fraud.
5. Adversarial attacks: Researchers will develop techniques to detect and prevent adversarial attacks on fraud detection models.
6. Continuous learning: Models will learn from new data and adapt to evolving fraud patterns, improving their detection capabilities over time.
7. Fraud prevention: Machine learning models will be used to prevent fraud before it occurs, rather than just detecting it after the fact.
8. Customer segmentation: Models will be used to segment customers based on their fraud risk, enabling targeted fraud prevention strategies.
9. Collaboration and data sharing: Industry-wide collaboration and data sharing will become more prevalent, enabling the development of more accurate fraud detection models.
10. Regulatory compliance: Machine learning models will be designed to comply with regulatory requirements, such as GDPR and CCPA.

These advancements will lead to even more effective fraud detection and prevention, reducing financial losses and enhancing the overall security of online payments.

10.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

10.APPENDIX

Model building :

1)Dataset

2)Google colab and VS code Application Building

 1. HTML file (Index file, Predict file , submit file)

2. Models in pickle format

SOURCE CODE:

<u>INDEX.HTML</u>

```
<!DOCTYPE html>
<html>
<head>
   <title>home Image Page</title>
   <style>
      body, html {
         height: 100%;
         margin: 0;
      }

      .bg-image {
         /* The image used */
         background-image: url('2.jpg');

         /* Full height */
         height: 100%;

         /* Center and scale the image nicely */
         background-position: center;
         background-repeat: no-repeat;
         background-size: cover;

         /* Set up the opacity */
         opacity: 0.9;
      }

      .content {
         position: absolute;
         top: 0;
         left: 0;
         width: 100%;
```

```
            height: 100%;
            background: rgba(0, 0, 0, 0); /* Transparent background to see the image behind */
        }

        h1{
            text-align: center;
            /* Center the text */
            margin: 10% 5%;
            color: rgb(33, 8, 254); /* Text color */
            font-size: 50px;
        }
        p {
            margin: 0 20px; /* Leaves space on the left and right */
            font-size: 30px; /* Increases the font size */
        }
        .top-right-button {
            position: absolute;
            top: 160px;
            right: 100px;
        }

        button {
            padding: 10px 20px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>

<div class="bg-image"></div>

<div class="content">
    <h1>Online Payments Fraud Detection</h1>
    <p>The objective of this article is to predict online payments fraud given the various parameters. This will be a classification problem since the target or dependent variable is the fraud (categorical values). The purpose of fraud of online payments are to separate the available supply of potable online payments into classes differing in superiority. We will be using classification algorithms such as Decision tree, Random forest, SVM, and Extra tree classifier. We will train and test the data with these algorithms.</p>
    <div class="top-right-button">
        <a href="predict.html"><button style="font-size: 20px;">Predict</button></a>
        <a href="home.html"><button style="font-size: 20px;">Home</button></a>
    </div>
</div>

</body>
</html>
```

PREDICT.HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Form with Background Image</title>
    <style>
        body, html {
            height: 100%;
            margin: 0;
            font-family: Arial, sans-serif;
        }

        .bg-image {
            background-image: url('2.jpg');
            filter: opacity(0.9);
            height: 100%;
            background-position: center;
            background-repeat: no-repeat;
            background-size: cover;
            position: relative;
        }

        .form-container {
            position: absolute;
            left: 50px;
            top: 50%;
            transform: translateY(-50%);
        }

        label {
            margin-top: 10px;
            display: block;
        }

        input[type="text"] {
            margin: 5px 0 20px 0;
            padding: 10px;
            width: calc(100% - 22px); /* Adjust input width considering padding */
            display: block;
        }

        button {
            padding: 10px 20px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        button:hover {
            background-color: #0056b3;
        }
        .top-right-buttons {
        position: absolute;
        top: 20px;
        right: 20px;
    }

    .top-right-buttons a {
        padding: 10px 20px;
        background-color: #007bff;
        color: white;
        text-decoration: none;
        border-radius: 5px;
        margin-left: 10px; /* Space between buttons */
    }

    .top-right-buttons a:hover {
        background-color: #0056b3;
    }

    </style>
</head>
<body>

<div class="bg-image">
    <div class="form-container">
        <form>
            <label for="Step">Step</label>
            <input type="text" id="Step" name="Step">

            <label for="Type">Step</label>
            <input type="text" id="Type" name="Type">

            <label for="Amount">Amount</label>
            <input type="text" id="Amount" name="Amount">

            <label for=" oldbalanceOrig  ">old balance Orig</label>
            <input type="text" id="oldbalanceOrig" name="oldbalanceOrig">

            <label for="newbalanceOrig">New Balance Orig</label>
            <input type="text" id="newbalanceOrig" name="newbalanceOrig">

            <label for="OldbalanceDest">OldbalanceDest</label>
            <input type="text" id="OldbalanceDest" name="OldbalanceDest">

            <label for="NewbalanceDest">NewbalanceDest</label>
            <input type="text" id="NewbalanceDest" name="NewbalanceDest">

            <a href="submit.html" style="display: inline-block; padding: 10px 20px;background-color:
#007bff;  color: white; text-decoration: none; border-radius: 5px;">Submit</a>
        </form>
    </div>
</div>
<div class="top-right-buttons">
    <a href="home.html">Home</a>
</div>

</body>
</html>
```

SUBMIT.HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Background Image Page</title>
    <style>
        body, html {
            height: 100%;
            margin: 0;
        }

        .bg-image {
            /* The image used */
            background-image: url('2.jpg');

            /* Full height */
            height: 100%;

            /* Center and scale the image nicely */
            background-position: center;
            background-repeat: no-repeat;
            background-size: cover;

            /* Set up the opacity */
            opacity: 0.6;
        }

        .content {
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            background: rgba(0, 0, 0, 0); /* Transparent background to see the image behind */
        }

        h1{
            text-align: center;
            /* Center the text */
            margin: 10% 5%;
            color: rgb(33, 8, 254); /* Text color */
            font-size: 50px;
        }
        p {
            margin: 0 20px; /* Leaves space on the left and right */
            font-size: 30px; /* Increases the font size */
        }
        .top-right-button {
            position: absolute;
            top: 160px;
            right: 100px;
        }
        .top-right-button2{
            position: absolute;
            top: 160px;
            right: 220px;
        }

        button {
            padding: 10px 20px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        button:hover {
```

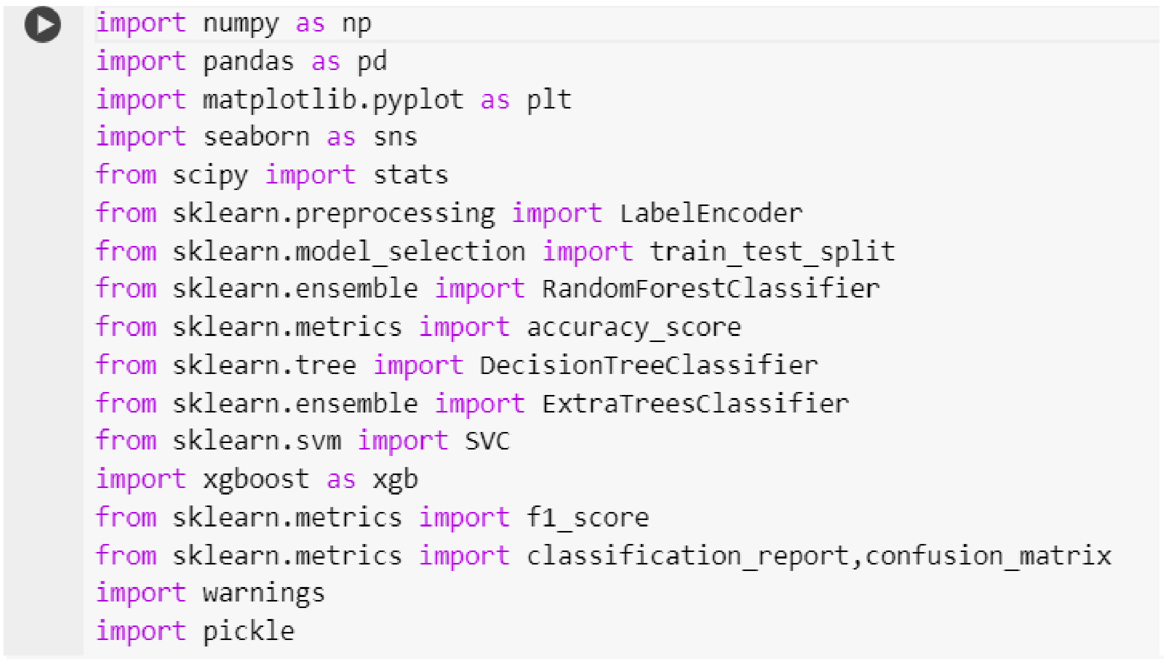
```
            background-color: #0056b3;
        }
    </style>
</head>
<body>

<div class="bg-image"></div>

<div class="content">
    <h1>Online Payments Fraud Detection</h1>
    <p>The predicted fraud of online payment is.</p>
    <div class="top-right-button">
        <a href="predict.html"><button style="font-size: 20px;">Predict</button></a>
    </div>
    <div class="top-right-button2">
        <a href="home.html"><button style="font-size: 20px;">Home</button></a>
    </div>
</div>

</body>
</html>
```

APP.PY

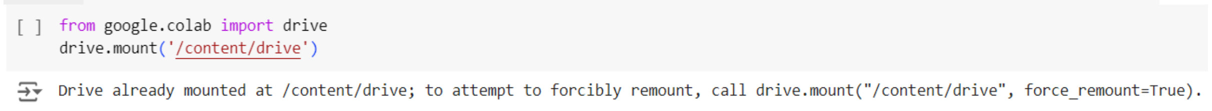
```
import numpy as np
import pickle
import pandas as pd
from flask import Flask, render_template, request

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','oldbalanceOrig','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)
```

Model building

Results

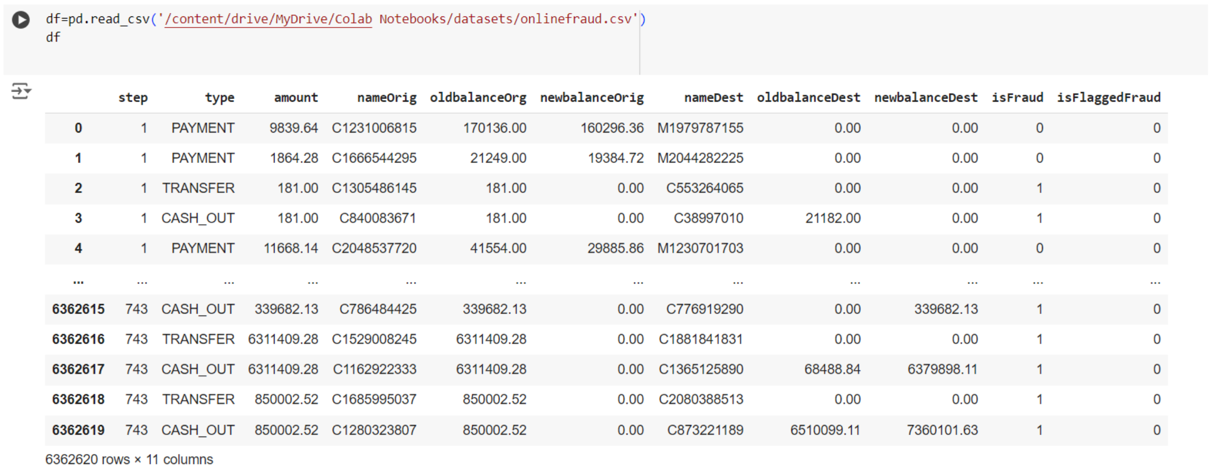


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import 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
```



```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

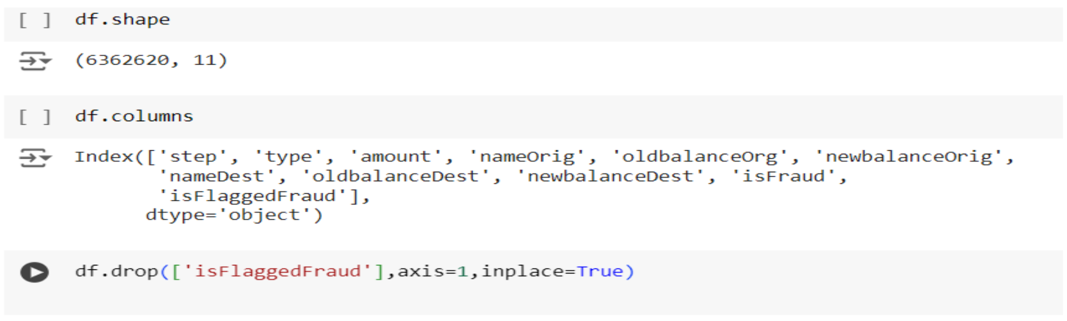
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```



```
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



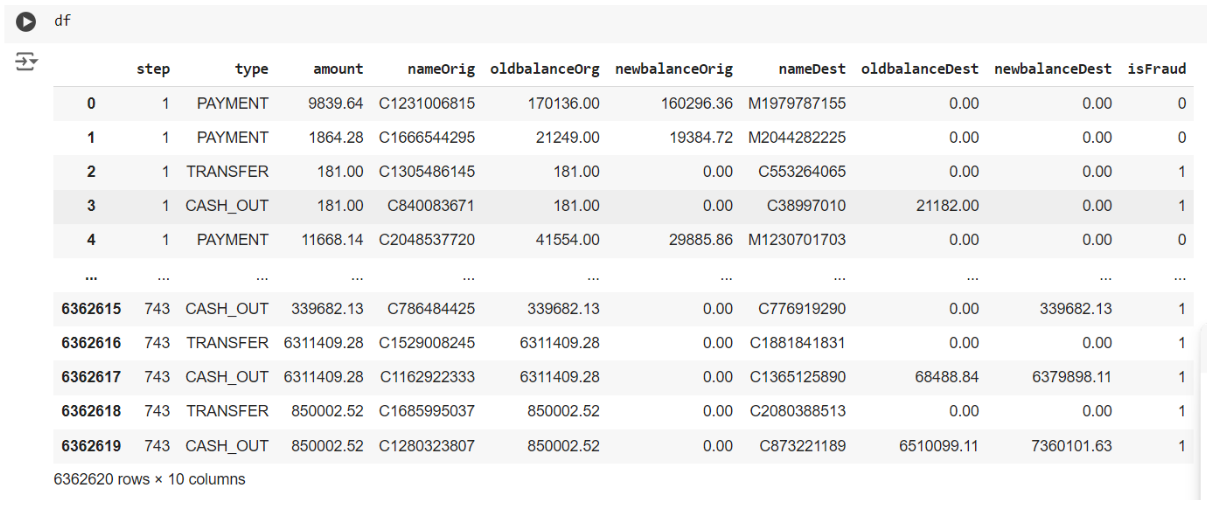
```
[ ] df.shape
```

```
(6362620, 11)
```

```
[ ] df.columns
```

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

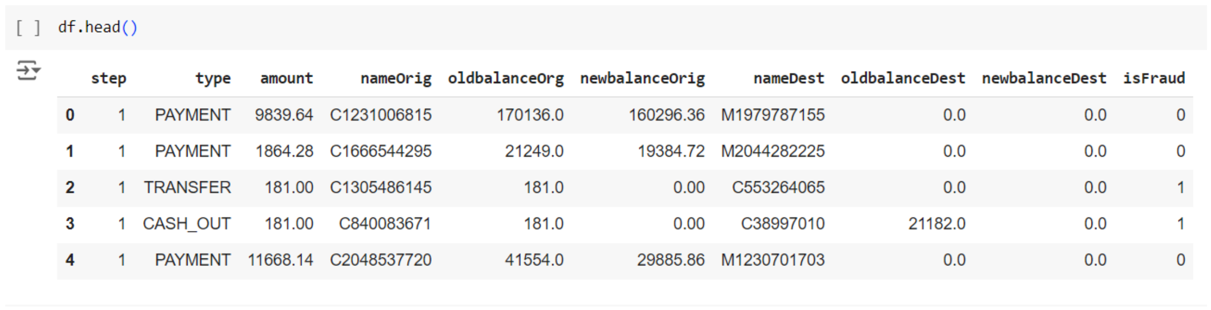
```
df.drop(['isFlaggedFraud'],axis=1,inplace=True)
```



```
df
```

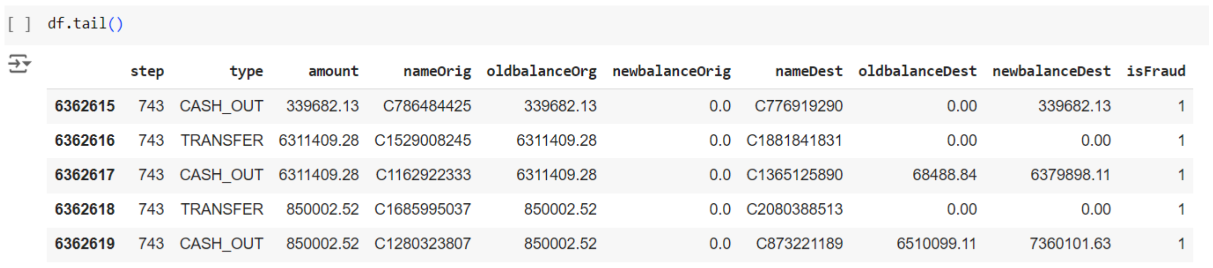
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065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

6362620 rows × 10 columns



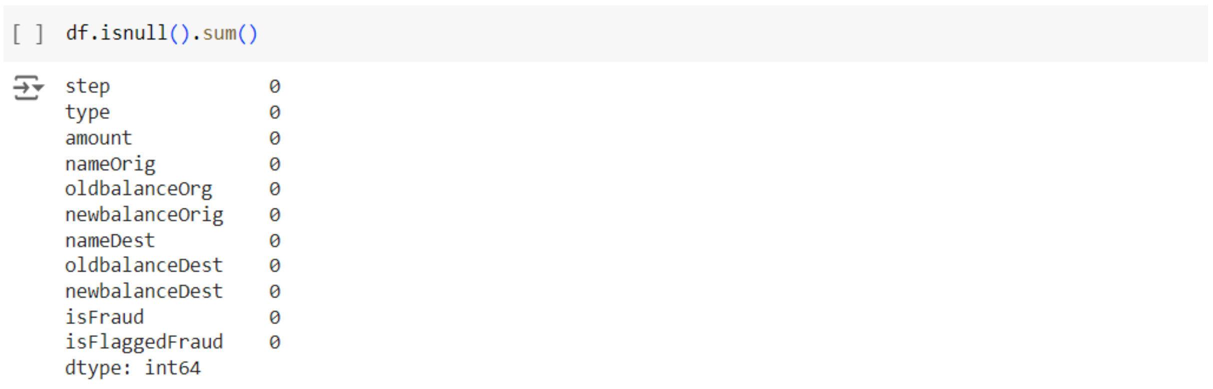
```
[ ] df.head()
```

	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



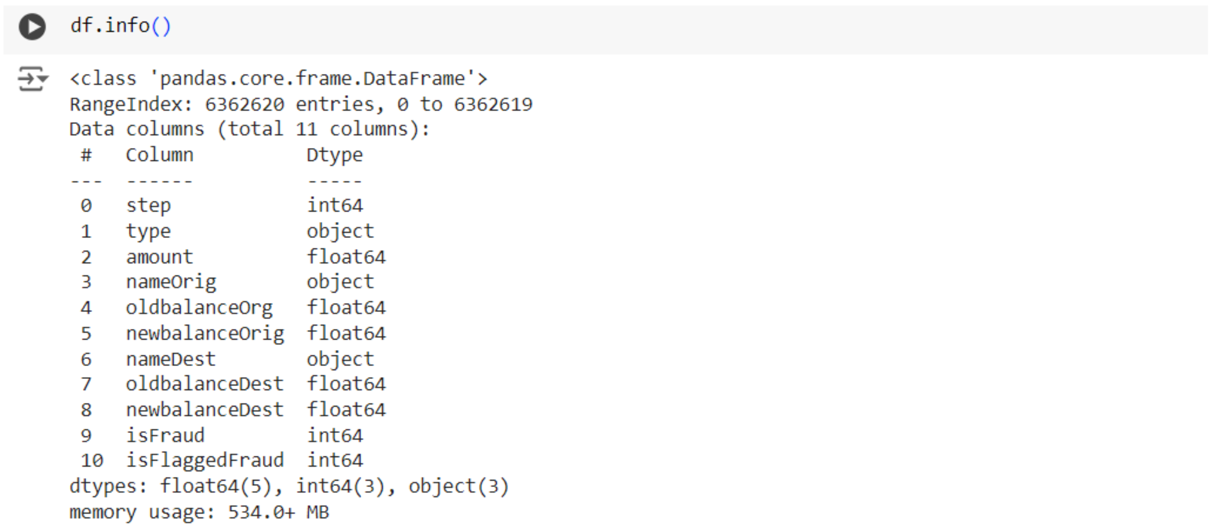
```
[ ] df.tail()
```

	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



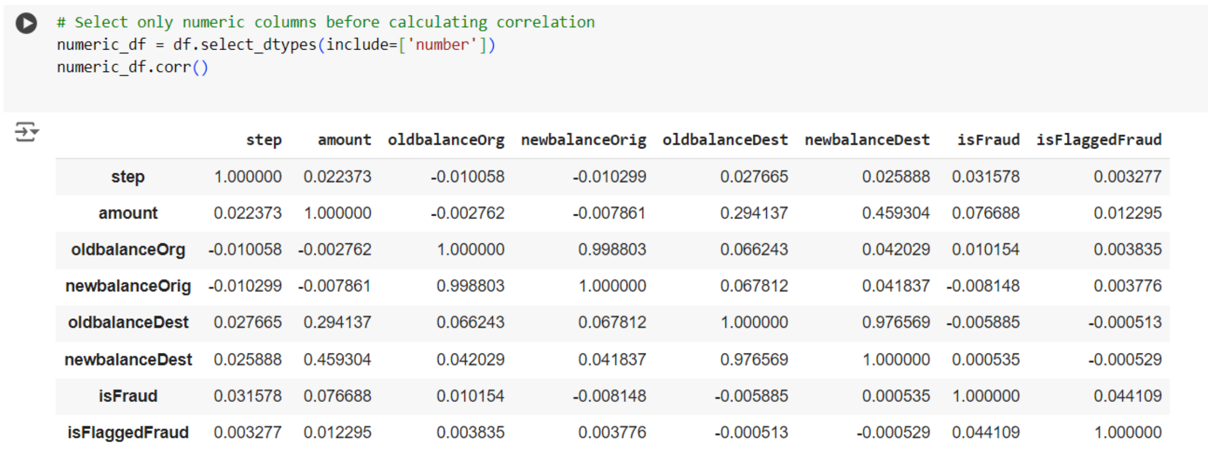
```
[ ] df.isnull().sum()
```

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



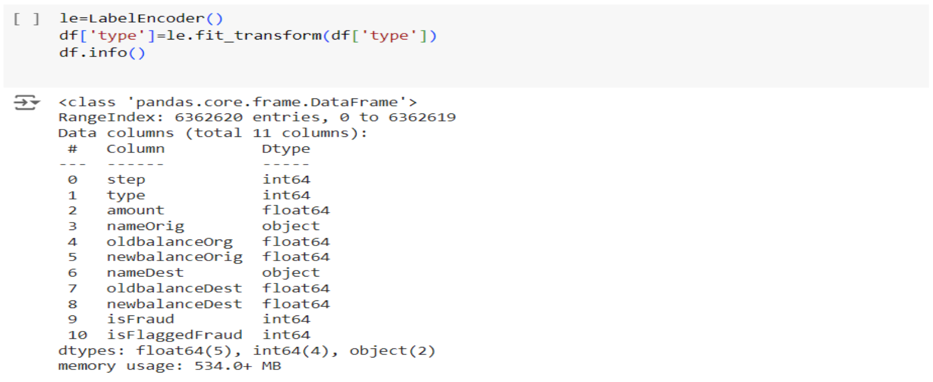
```
df.info()
```

```
<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
```



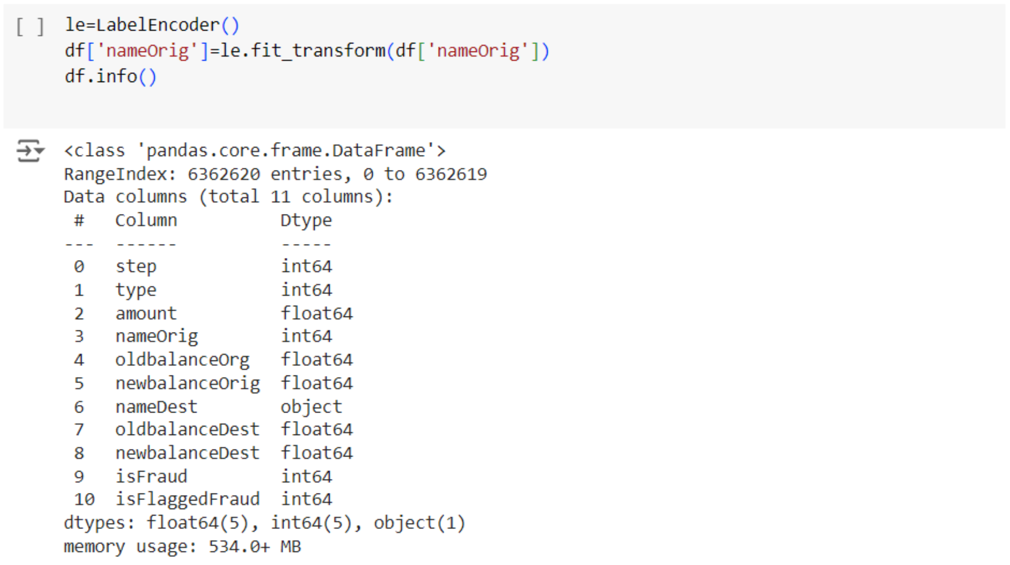
```
# Select only numeric columns before calculating correlation
numeric_df = df.select_dtypes(include=['number'])
numeric_df.corr()
```

	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



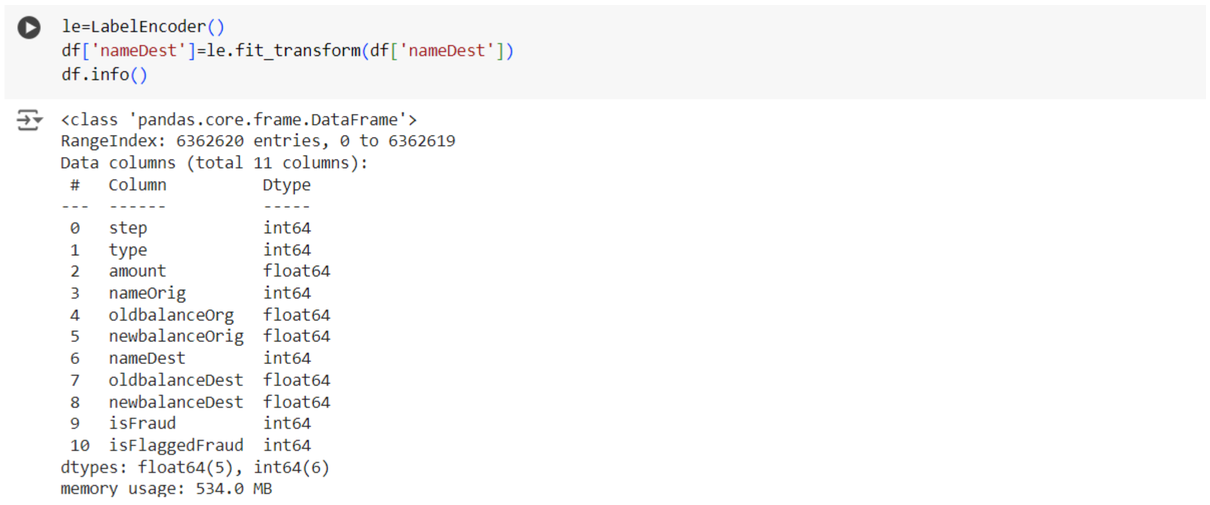
```
le=LabelEncoder()
df['type']=le.fit_transform(df['type'])
df.info()
```

```
<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
```



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

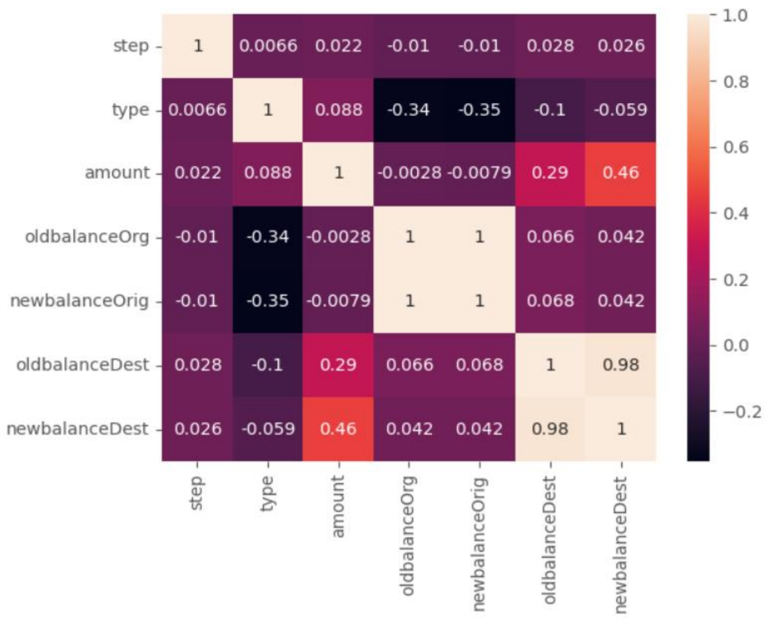
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype  
---  ------          -----  
 0   step            int64  
 1   type            int64  
 2   amount          float64
 3   nameOrig        int64  
 4   oldbalanceOrg   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['nameDest']=le.fit_transform(df['nameDest'])
df.info()
```

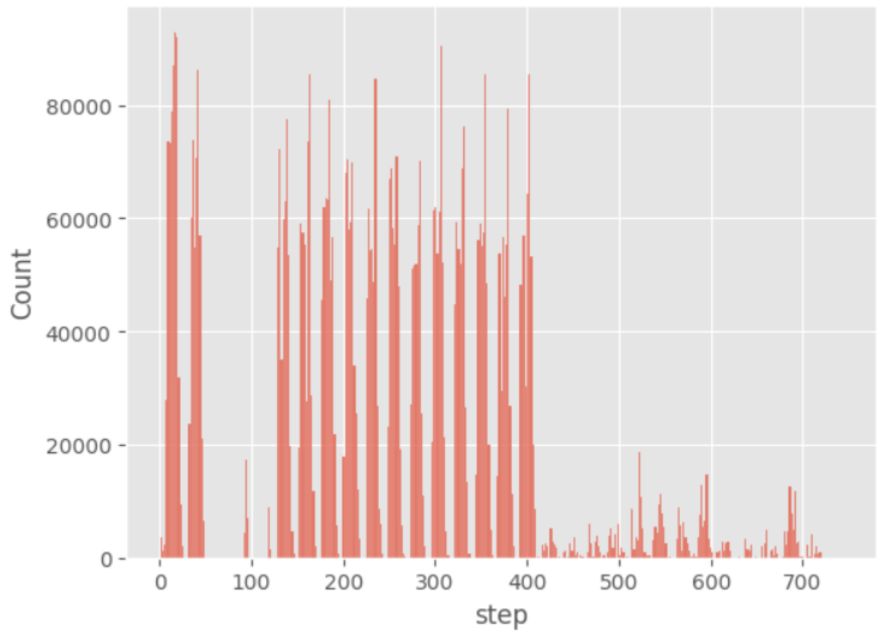
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype  
---  ------          -----  
 0   step            int64  
 1   type            int64  
 2   amount          float64
 3   nameOrig        int64  
 4   oldbalanceOrg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

Heap Map



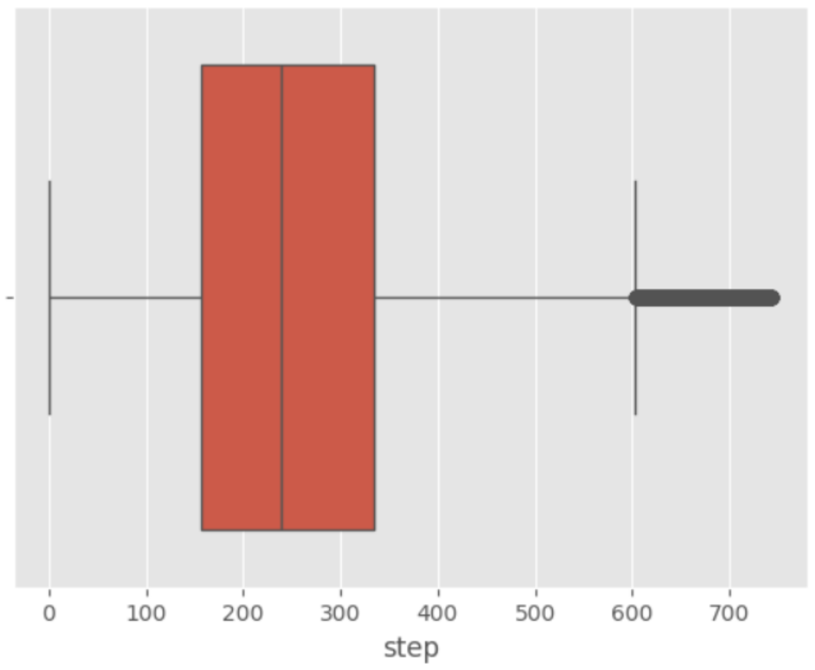
Histogram Plot

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



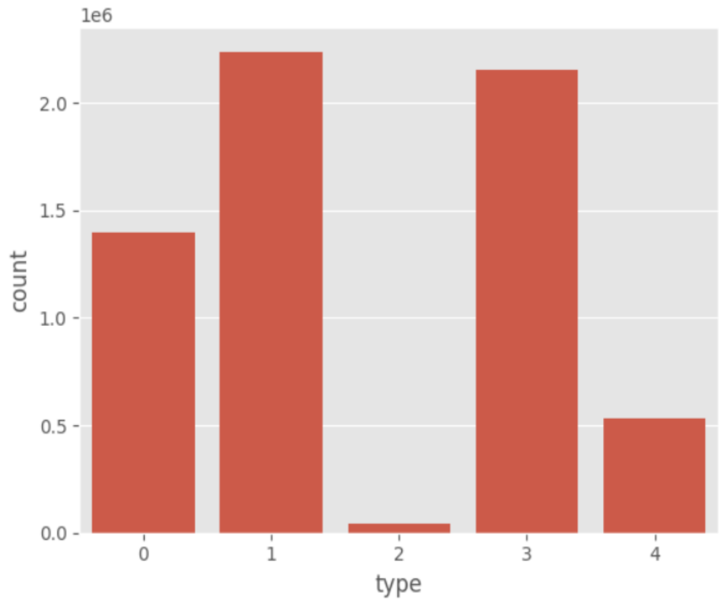
Box Plot

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


0
100
200
300
400
500
600
700
step

Count Plot

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

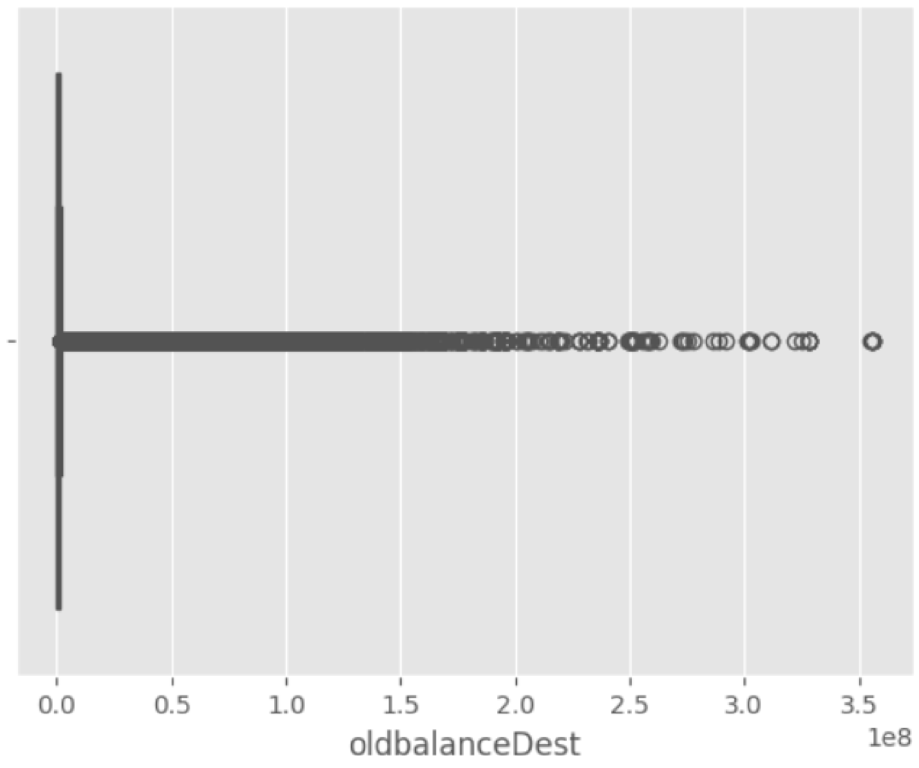

1e6
2.0
1.5
1.0
0.5
0.0
count
0
1
2
3
4
type

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

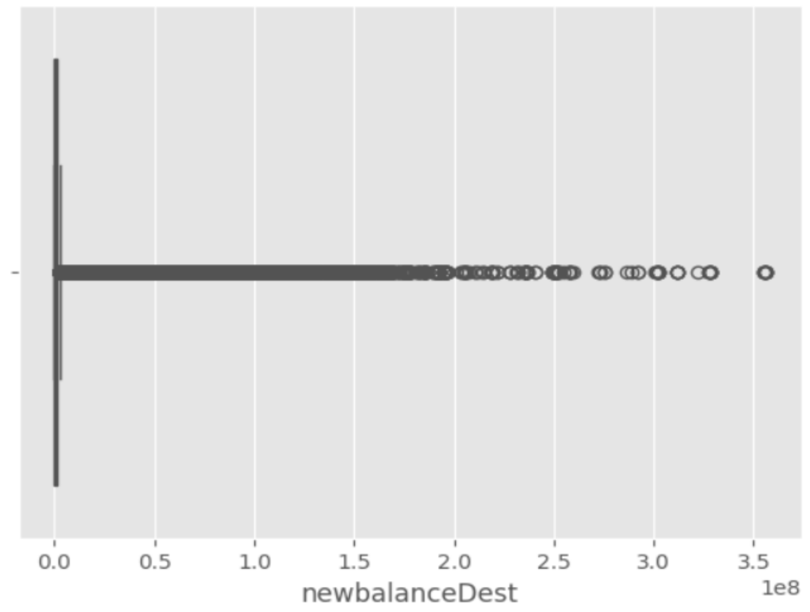
```
nameDest
84652      113
567820     109
472721     105
320660     102
349730     101
          ...
1095075      1
939730       1
1445164      1
1774945      1
319713       1
Name: count, Length: 2722362, dtype: int64
```

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

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

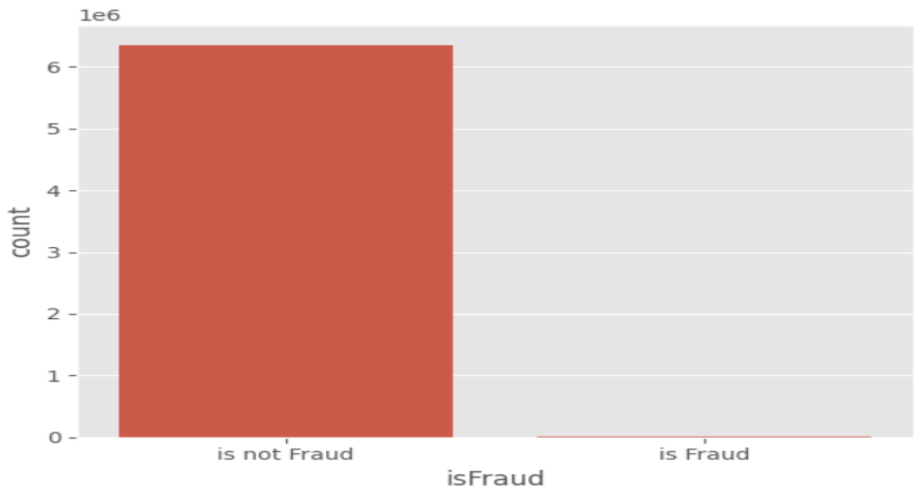


```
<Axes: xlabel='newbalanceDest'>
```



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

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



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

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

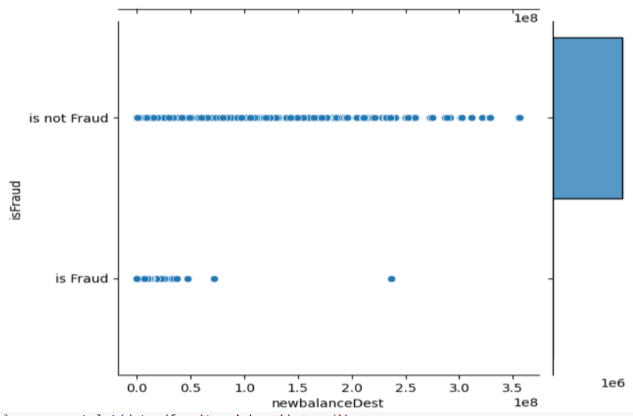
```
[ ]  df.loc[df['isFraud']==0,'isFraud']='is not Fraud'
     df.loc[df['isFraud']==1,'isFraud']='is Fraud'
```

```
[ ] df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	3	9839.64	757869	170136.00	160296.36	1662094	0.00	0.00	is not Fraud	0
1	1	3	1864.28	2188998	21249.00	19384.72	1733924	0.00	0.00	is not Fraud	0
2	1	4	181.00	1002156	181.00	0.00	439685	0.00	0.00	is Fraud	0
3	1	1	181.00	5828262	181.00	0.00	391696	21182.00	0.00	is Fraud	0
4	1	3	11668.14	3445981	41554.00	29885.86	828919	0.00	0.00	is not Fraud	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	1	339682.13	5651847	339682.13	0.00	505863	0.00	339682.13	is Fraud	0
6362616	743	4	6311409.28	1737278	6311409.28	0.00	260949	0.00	0.00	is Fraud	0
6362617	743	1	6311409.28	533958	6311409.28	0.00	108224	68488.84	6379898.11	is Fraud	0
6362618	743	4	850002.52	2252932	850002.52	0.00	319713	0.00	0.00	is Fraud	0
6362619	743	1	850002.52	919229	850002.52	0.00	534595	6510099.11	7360101.63	is Fraud	0

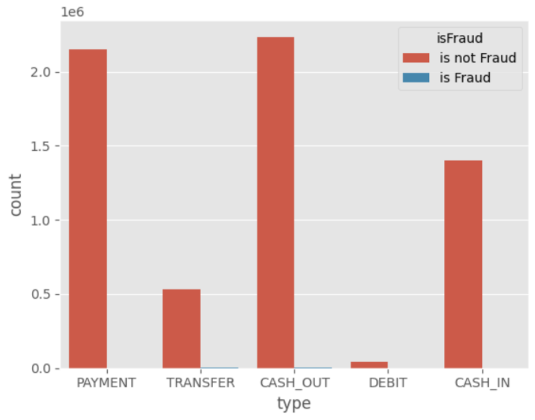
6362620 rows × 11 columns

Joint Plot



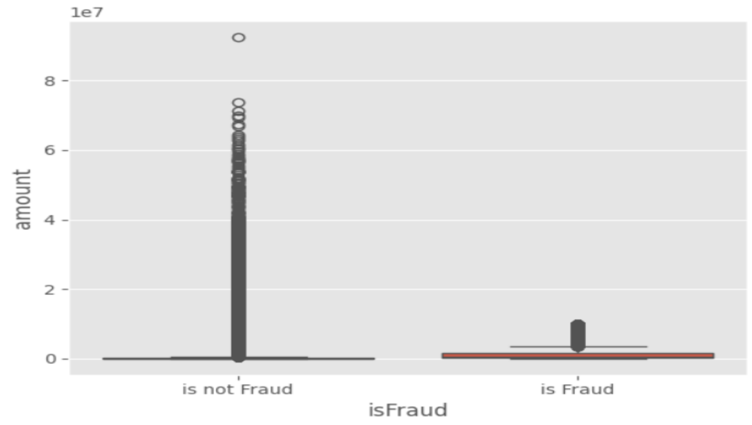
```
[ ] sns.countplot(data=df,x='type',hue='isFraud')
```

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



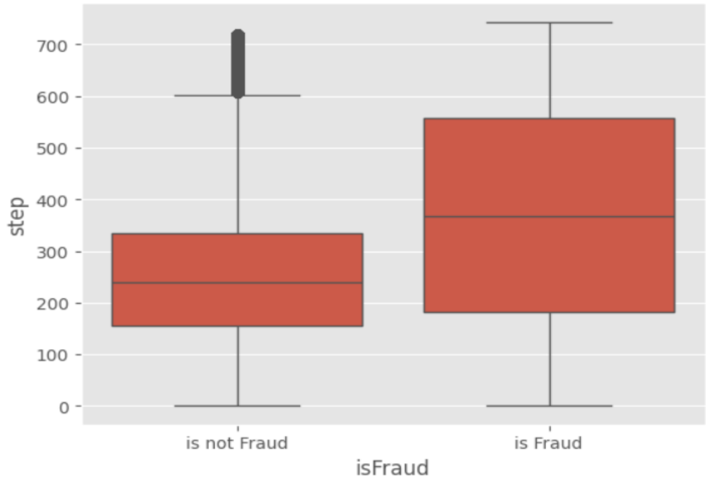
```
sns.boxplot(data=df,x='isFraud',y='amount')
```

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



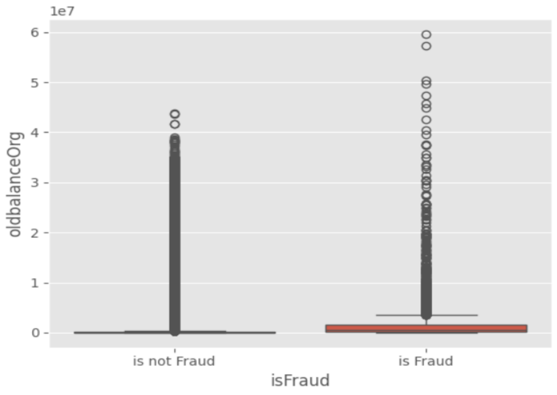
```
sns.boxplot(data=df,x='isFraud',y='step')
```

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



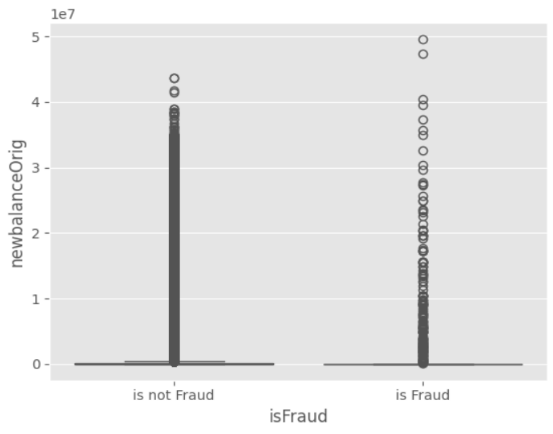
```
[ ] sns.boxplot(data=df,x='isFraud',y='oldbalanceOrg')
```

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



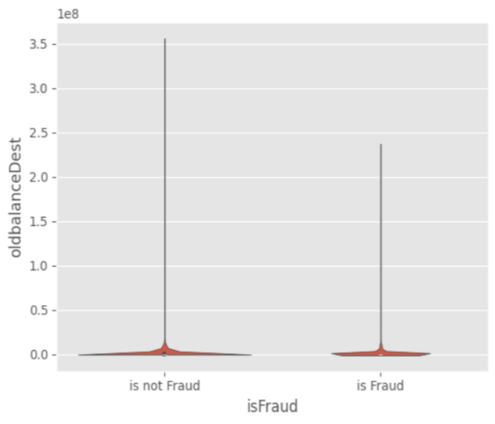
```
sns.boxplot(data=df,x='isFraud',y='newbalanceOrig')
```

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



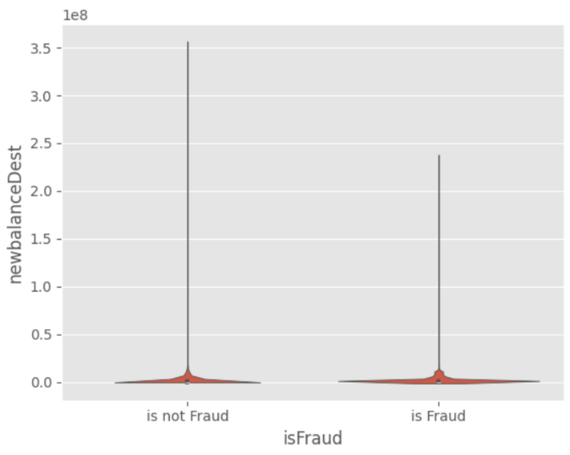
```
sns.violinplot(data=df,x='isFraud',y='oldbalanceDest')
```

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



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

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

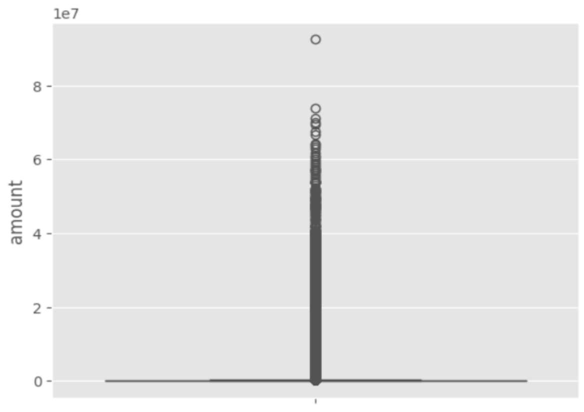


```
[ ] df.describe(include='all')
```

	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

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

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



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

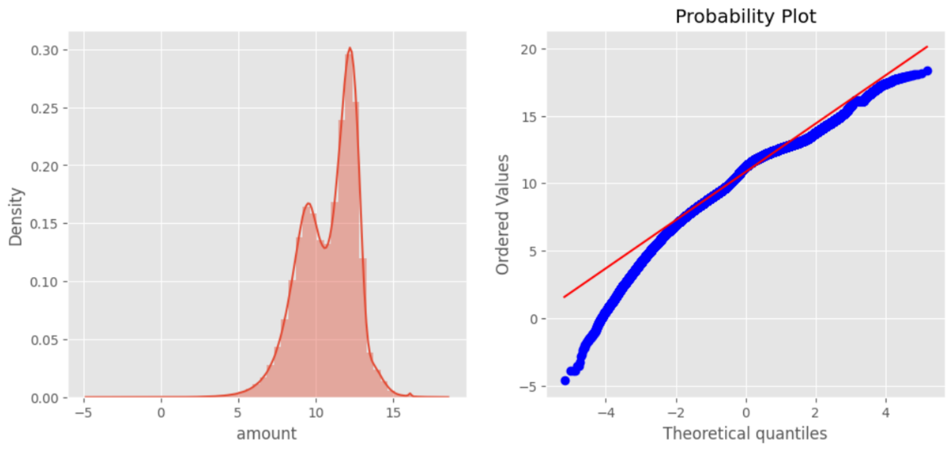
```
ModeResult(mode=10000000.0, count=3207)
179861.90354913071
```

```
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
```

```
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)
```

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



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

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

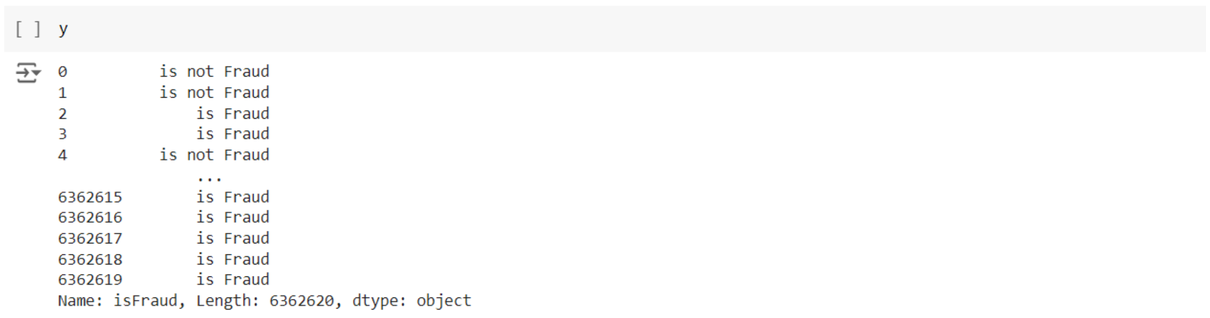
```
type
1    2237500
3    2151495
0    1399284
4     532909
2      41432
Name: count, dtype: int64
```

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

```
x
```

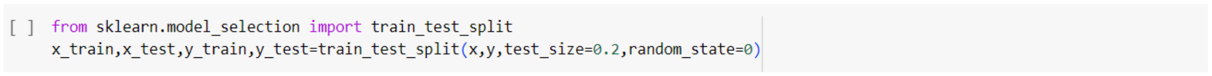
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFlaggedFraud
0	1	3	9839.64	757869	170136.00	160296.36	1662094	0.00	0.00	0
1	1	3	1864.28	2188998	21249.00	19384.72	1733924	0.00	0.00	0
2	1	4	181.00	1002156	181.00	0.00	439685	0.00	0.00	0
3	1	1	181.00	5828262	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

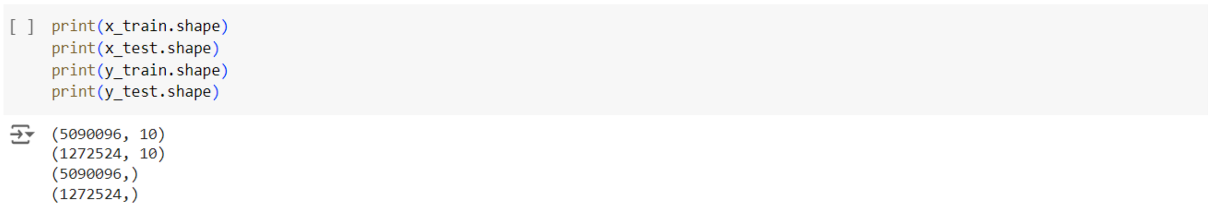


```
[ ] y
```

```
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
```

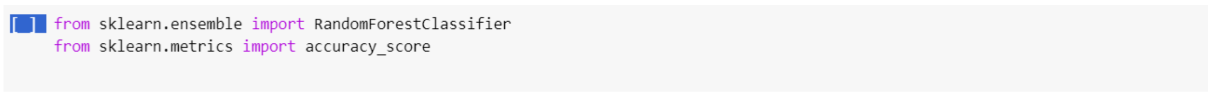


```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

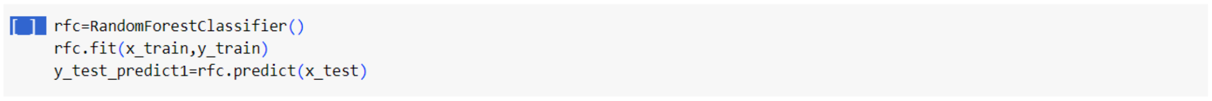


```
[ ] print(x_train.shape)
    print(x_test.shape)
    print(y_train.shape)
    print(y_test.shape)
```

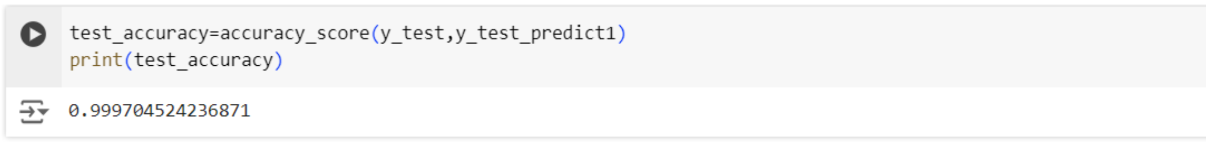
```
(5090096, 10)
(1272524, 10)
(5090096,)
(1272524,)
```



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

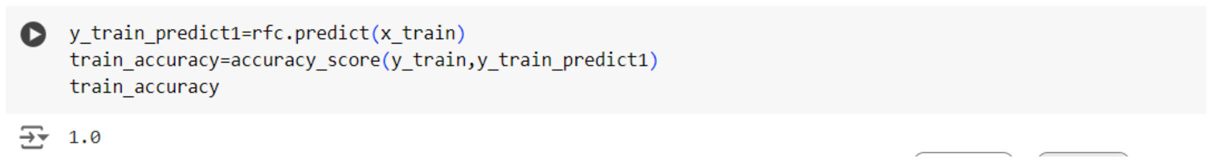


```
[ ] rfc=RandomForestClassifier()
    rfc.fit(x_train,y_train)
    y_test_predict1=rfc.predict(x_test)
```



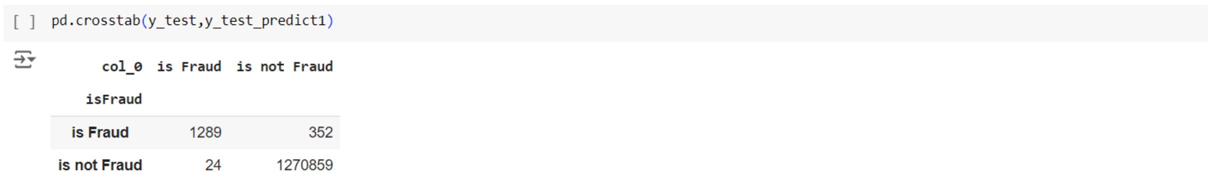
```
test_accuracy=accuracy_score(y_test,y_test_predict1)
print(test_accuracy)
```

```
0.999704524236871
```



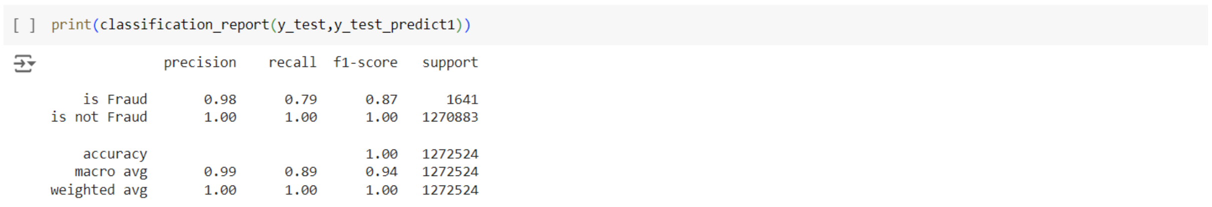
```
y_train_predict1=rfc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict1)
train_accuracy
```

```
1.0
```



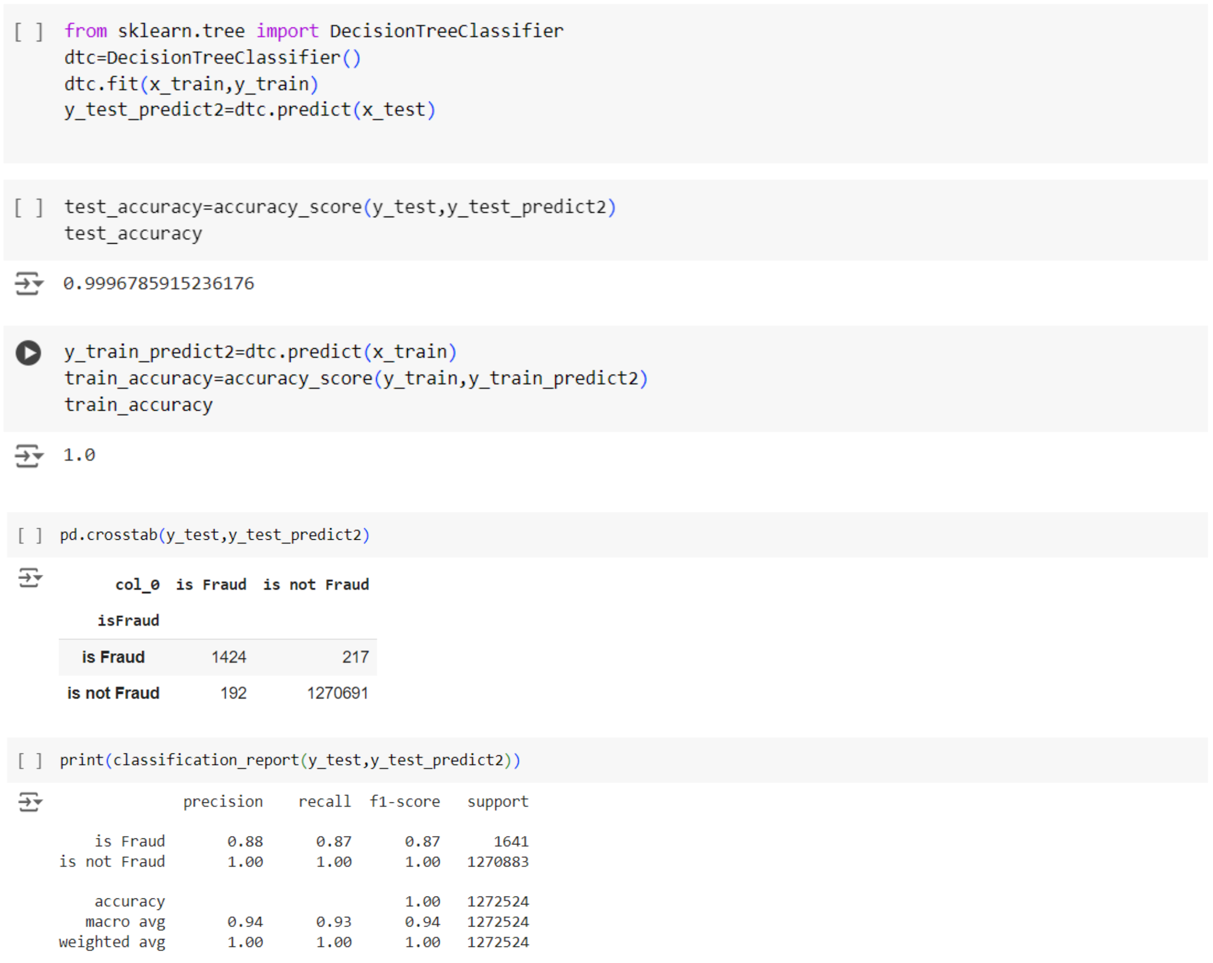
```
[ ] pd.crosstab(y_test,y_test_predict1)
```

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
```



```
[ ] 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
```

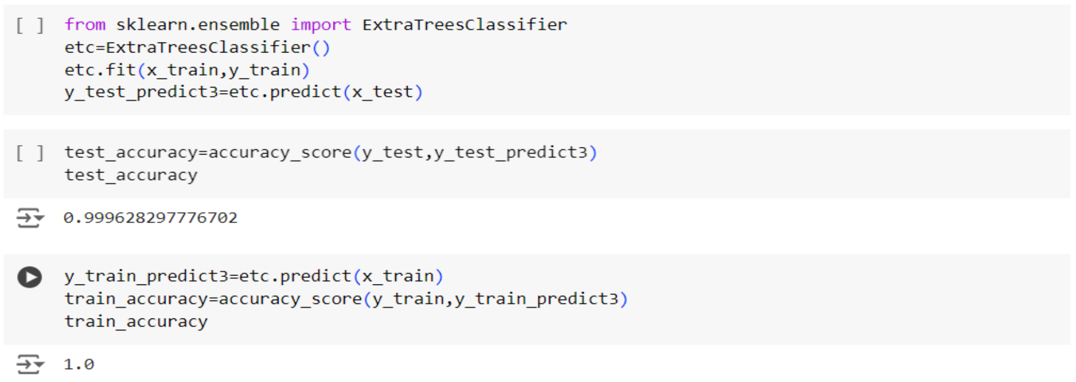
```
1.0
```

```
[ ] 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
```



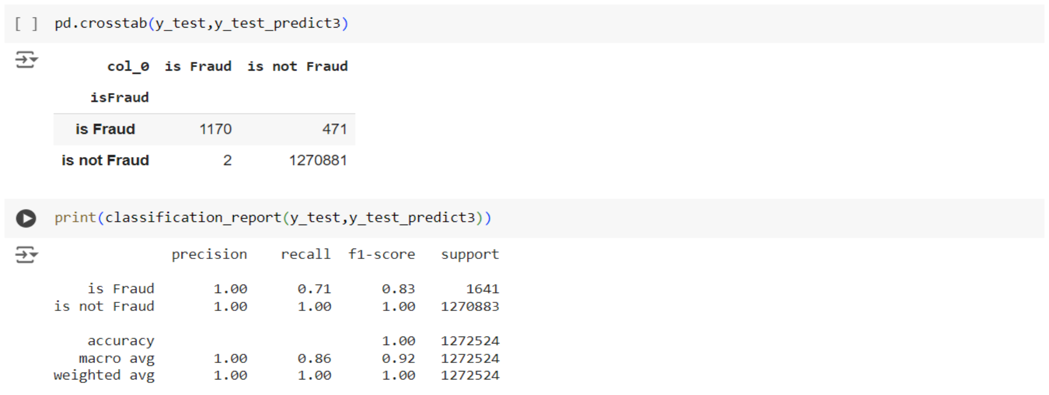
```
[ ] from sklearn.ensemble import ExtraTreesClassifier
    etc=ExtraTreesClassifier()
    etc.fit(x_train,y_train)
    y_test_predict3=etc.predict(x_test)
```

```
[ ] 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
```

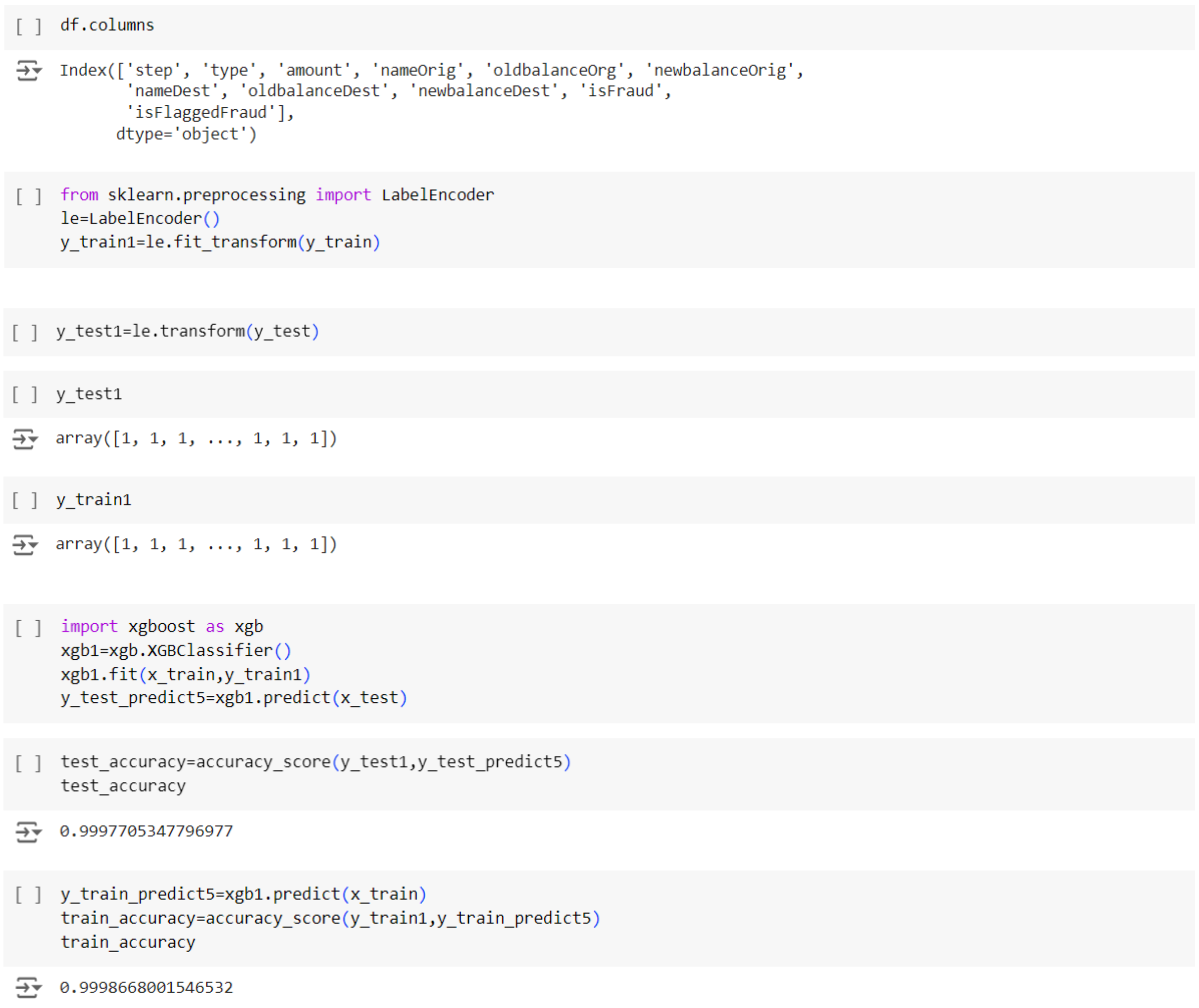


```
[ ] pd.crosstab(y_test,y_test_predict3)
```

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
```



```
[ ] 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])
```

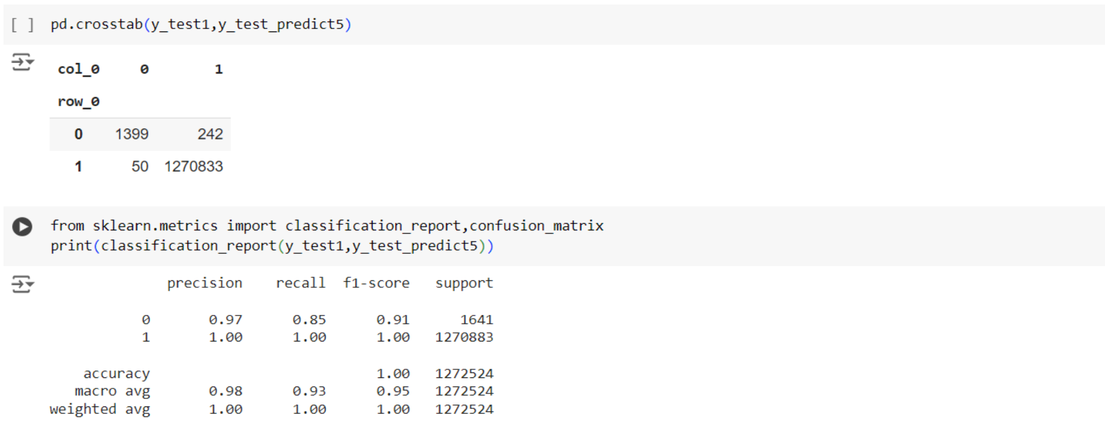
```
[ ] import xgboost as xgb
    xgb1=xgb.XGBClassifier()
    xgb1.fit(x_train,y_train1)
    y_test_predict5=xgb1.predict(x_test)
```

```
[ ] 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
```

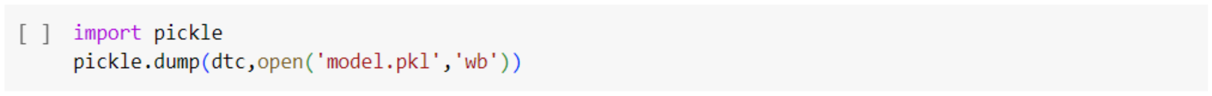


```
[ ] pd.crosstab(y_test1,y_test_predict5)
```

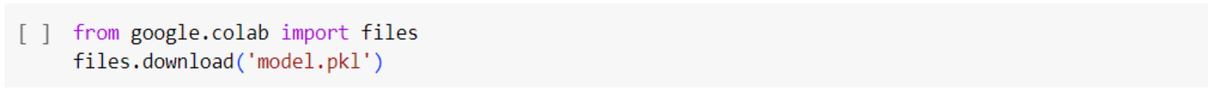
col_0	0	1
row_0		
0	1399	242
1	50	1270833

```
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
```



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



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