ONLINE PAYMENT FRAUD DETECTION

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

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

MASTER OF COMPUTER APPLICATIONS
In
COMPUTER SCIENCE AND ENGINEERING(MCA)

Submitted By

ABOTHU KUMAR **23UK1F0001**

Under the guidance of

DR. G RAMESH

Assistant Professor

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
VAAGDEVI ENGINEERING COLLEGE(AUTONOMUS)

Affiliated to JNTUH(AUTONOMUS), HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) -506005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(MCA)

VAAGDEVI ENGINEERING COLLEGE(AUTONOMUS)

CERTIFICATE OF COMPLETION PROJECT WORK REVIEW-I

This is to certify that the PG Project Phase-1 entitled **"ONLINE PAYMENT FRAUD DETECTION"** is being submitted by **ABOTHU KUMAR (23UK1F0001)** in partial fulfilment of the requirements for the award of the degree of Post Graduation in Master of Computer Applications to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023-2024.

Project Guide	HOD
DR.G. RAMESH	**DR. R. NAVEEN KUMAR**
(Assistant Professor)	(Professor)

External

ACKNOWLEDGEMENT

I wish to take this opportunity to express my 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 PG I extend my heartfelt thanks to **DR.R. NAVEEN KUMAR**, Head of the Department of MCA, Vaagdevi Engineering College for providing necessary infrastructure and
thereby giving freedom to carry out the PG Project Phase-1.

I express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the PG Project Phase-1 and for their support in completing the PG Project Phase-1.

I express heartfelt thanks to the guide, **DR. G RAMESH**, Assistant professor, Department of CSE for her constant support and giving necessary guidance for completion of this PG Project Phase-1.

Finally, I express my sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

ABOTHU KUMAR **(23UK1F0001)**

ABSTRACT

- Online payments fraud detection is a critical component of e-commerce security. Fraudulent transactions result in financial losses and damage to consumer trust. Machine learning and data analytics techniques are employed to develop fraud detection systems. These systems analyse patterns and anomalies in transaction data to identify potential fraud. Key factors include transaction amount, location, frequency, and user behaviour. Accurate fraud detection enables swift action, minimizing financial losses and enhancing payment security.
- When it comes to the simplicity of making a payment while sitting anywhere in the world, online payments have been a source of attractiveness. Over the past few decades, there has been an increase in online payments. Epayments enable businesses earn a lot of money in addition to consumers. However, because electronic payments are so simple, there is also a risk of fraud associated with them. A consumer must ensure that the payment he is paying is going exclusively to the appropriate service provider. Online fraud exposes users to the possibility of their data being compromised, as well as the inconvenience of having to report the fraud, block their payment method, and other things. When businesses are involved, it causes some issues; occasionally, they must issue refunds in order to keep custom ers. Therefore, it is crucial that both consumers and businesses are aware of these internet scams. A model to determine if an online payment is fraudulent or not is put forth in this study. To determine if a certain Online payment is fraudulent or not, some features like the type of payment, the recipient's identity, etc. would be taken into account

- Online payments fraud detection uses a combination of human analysis and automated systems to identify and prevent fraudulent transactions. The goals are to minimize financial losses, enhance payment security, and maintain consumer trust.

  Some key techniques used in fraud detection include:

- Machine learning algorithms to analyze transaction patterns and identify anomalies
- Data analytics to examine transaction data and user behavior
- Rule-based systems to flag transactions that meet specific fraud criteria
- Collaborative filtering to share fraud information across industries and platforms
- Human review to investigate suspicious transactions and make informed decisions

- Some key challenges in fraud detection include:

- Staying ahead of evolving fraud tactics and strategies
- Avoiding false positives that disrupt legitimate transactions
- Balancing security with convenience and user experience
- Addressing data privacy and security concerns
- Keeping up with increasing transaction volumes and velocities

- Some popular solutions for fraud detection include:

- Machine learning-based fraud detection platforms
- Fraud detection as a service (FDaaS) offerings
- Payment gateway integrations with fraud detection capabilities
- Standalone fraud detection software and tools
- Some key players in the fraud detection market include:
- Fraud detection platform providers like Stripe and PayPal
- FDaaS providers like Signifyd and Forter
- Payment gateways like (link unavailable) and Cybersource

TABLE OF CONTENTS:-

1. INTRODUCTION

1. OVERVIEW

2. OBJECTIVES

- **Detect Fraudulent Transactions:** Develop a system that can accurately identify fraudulent transactions in real-time.
- **Minimize False Positives:** Ensure the system has a low false positive rate to avoid inconvenience to genuine customers.
- **Scalability:** Design the system to handle a large volume of transactions efficiently.
- **Adaptability:** Ensure the system can adapt to evolving fraud patterns.

3. Scope

- **Data Collection:** Gather transaction data, including user information, transaction details, and any available labels indicating fraudulent activity.
- **Data Preprocessing:** Clean and preprocess the data to make it suitable for analysis and model training.
- **Feature Engineering:** Identify and create features that can help in distinguishing between fraudulent and genuine transactions.
- **Model Development:** Develop and train machine learning models using various algorithms such as logistic regression, decision trees, random forests, gradient boosting, and neural networks.
- **Model Evaluation:** Evaluate the performance of the models using metrics such as accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC-ROC).
- **Deployment:** Deploy the best-performing model in a real-time environment to monitor transactions and flag potential fraud.

4. Data Sources

- **Transaction Records:** Historical transaction data from payment processors or financial institutions.
- **User Behaviour Data:** Information on user behavior, such as login patterns, device details, and geolocation.
- **External Databases:** Publicly available datasets and fraud blacklists.

5. Methodology

1. **Data Collection and Integration:** Collect data from multiple sources and integrate them into a unified dataset.
2. **Exploratory Data Analysis (EDA):** Perform EDA to understand the data distribution, identify patterns, and detect anomalies.
3. **Feature Engineering:** Create new features that highlight transaction characteristics and user behavior.
4. **Model Training:** Split the data into training and testing sets. Train multiple machine learning models and tune their hyperparameters.
5. **Model Evaluation:** Compare the models based on performance metrics and select the best one.
6. **Model Deployment:** Deploy the model in a real-time system, integrating it with the transaction processing pipeline.
7. **Monitoring and Maintenance:** Continuously monitor the model's performance and update it as needed to handle new fraud patterns.

6. Technologies and Tools

- **Programming Languages:** Python, R
- **Libraries and Frameworks:**
  Scikit-learn, TensorFlow, Kera's, XGBoost
- **Data Processing:** Pandas, NumPy
- **Visualization:** Matplotlib, Seaborn
- **Database:** SQL, NoSQL databases

- **Deployment:**
  Flask, Docker, AWS/GCP/Azure
  for cloud deployment

7. Challenges

- **Data Quality:** Ensuring the data is clean and accurately labeled.
- **Evolving Fraud Patterns:** Keeping up with new and sophisticated fraud tactics.
- **Real-time Processing:** Ensuring the system can process transactions in real-time without significant delays.
- **Balancing**
  Accuracy and Efficiency: Achieving high accuracy while maintaining low false positive rates.

8. Expected Outcomes

- **Improved Fraud Detection:** A
  reliable system capable of detecting fraudulent transactions with high accuracy.
- **Reduced Financial Losses:** Minimizing the financial impact of fraud on businesses and customers.
- **Enhanced User Trust:** Increased trust among users in the security of the online payment system.
- **Actionable Insights:** Insights into common fraud patterns and user behavior, helping in further strengthening security measures.

9. Future Work

- **Continuous Improvement:** Regularly update and retrain models with new data to improve detection accuracy.
- **Integration with Other Systems:** Integrate the fraud detection system with other security measures like biometric authentication and multi-factor authentication.

- **Advanced Techniques:** Explore advanced techniques such as deep learning, anomaly detection, and ensemble methods for better performance

1.1. PURPOSE

The purpose of online payment fraud detection is to:

Prevent financial losses: Identify and block fraudulent transactions in real-time to avoid financial losses for merchants and financial institutions.

Enhance security: Implement robust security measures to protect sensitive payment information and prevent unauthorized access.

Reduce fraud risk: Continuously monitor transactions to identify potential fraud patterns and update detection systems to stay ahead of evolving threats.

Improve customer trust: Ensure a safe and secure payment environment, fostering trust and confidence in online transactions.

Comply with regulations: Adhere to industry standards and regulatory requirements, such as PCI-DSS, GDPR, and PSD2, to avoid legal and reputational consequences.

Minimize false positives: Optimize detection systems to reduce incorrect fraud flags, avoiding unnecessary transaction declines and minimizing friction for legitimate customers.

Streamline payment processes: Implement efficient fraud detection systems to facilitate smooth and seamless payment experiences.

By achieving these goals, online payment fraud detection plays a critical role in maintaining the integrity and trustworthiness of digital payment ecosystems.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

Some existing problems in online payment fraud detection include:

1. Increasing sophistication of fraud schemes
2. High false positive rates (incorrectly flagging legitimate transactions)
3. Evolving threats from new technologies (e.g., deep learning-based attacks)
4. Insufficient data quality and availability
5. Inadequate collaboration and information sharing between organizations
6. Balancing security with customer convenience and minimizing friction
7. Staying compliant with constantly changing regulations
8. Managing large volumes of transaction data
9. Limited resources and budget constraints
10. Keeping up with emerging payment methods and technologies (e.g., cryptocurrencies)
11. Lack of standardization in fraud detection practices
12. Limited visibility into transaction data across multiple channels (e.g., web, mobile, offline)
13. High fraud rates in specific industries (e.g., e-commerce, online lending)
14. Limited use of advanced technologies (e.g., AI, machine learning) in fraud detection

15. Dependence on manual review processes, leading to delays and inefficiencies.

These challenges highlight the need for continuous improvement and innovation in onlinepayment fraud detection to stay ahead of emerging threats and minimize financial losses.

PROPOSED SOLLUTION

2. Proposed solutions for online payment fraud detection include:

3.

4. 1. Machine Learning (ML) and Artificial Intelligence (AI) integration
5. 2. Advanced Analytics and Predictive Modeling
6. 3. Real-time Transaction Monitoring and Risk Assessment
7. 4. Multi-Factor Authentication and Behavioral Biometrics
8. 5. Collaborative Fraud Detection and Information Sharing
9. 6. Automated Decisioning and Workflow Optimization
10. 7. Continuous Learning and Model Updates
11. 8. Integration with Cloud-based Fraud Detection Services
12. 9. Utilizing Alternative Data Sources (e.g., social media, device fingerprints)
13. 10. Implementing Explainable AI (XAI) for transparency and trust
14. 11. Leveraging Blockchain and Distributed Ledger Technology
15. 12. Enhancing Customer Verification and Identity Proofing
16. 13. Utilizing Graph-Based Fraud Detection and Network Analysis
17. 14. Implementing Real-Time Feedback and Continuous Improvement
18. 15. Utilizing Hybrid Approach combining rule-based and ML models
19. These solutions aim to improve the accuracy, efficiency, and effectiveness of online payment fraud detection, reducing financial losses and enhancing the overall security of digital payment ecosystems.

20. THEORITICAL ANALYSIS

Theoretical analysis of online payment fraud detection involves examining the underlying principles and concepts that drive fraud detection systems. Some theoretical aspects include:

1. Probability theory: Understanding probability distributions and statistical analysis to identify fraudulent patterns.

2. Game theory: Analyzing the cat-and-mouse game between fraudsters and fraud detection systems.

3. Information theory: Studying the encoding and decoding of transaction data to detect anomalies.

4. Decision theory: Evaluating the decision-making processes used in fraud detection, including risk assessment and threshold setting.

5. Machine learning theory: Examining the algorithms and models used in fraud detection, such as supervised and unsupervised learning.

6. Data mining theory: Analyzing the techniques used to discover patterns and relationships in large datasets.

7. Cryptography: Studying the encryption and decryption methods used to secure transaction data.

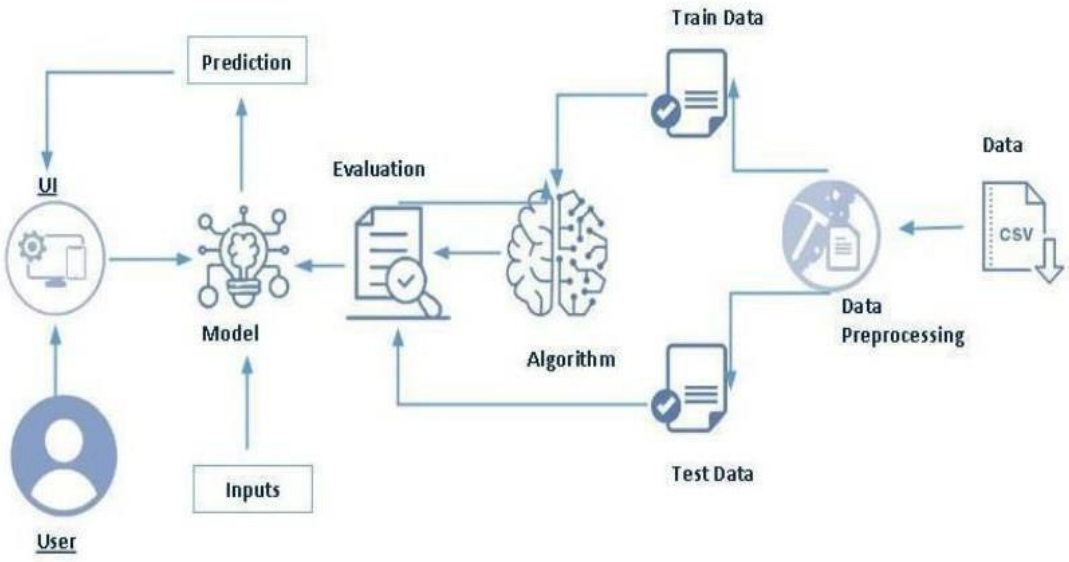
8. Network theory: Examining the relationships between transactions and entities to identify suspicious behavior.

9. Behavioral economics: Understanding the psychological and social factors that influence fraudulent behavior.

10. Signal processing theory: Analyzing the techniques used to extract meaningful information from transaction data.

Theoretical analysis provides a foundation for developing and improving online payment fraud detection systems, ensuring a robust and resilient defense against fraudulent activities.

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 air quality data, weather information, pollutant levels, 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 or 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, depending on the nature of the AQI prediction task.
- **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 AQI categories based on historical data.
- **UI Based 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 user-friendly platform for users to input location data or view AQI predictions, health information, and recommended precautions.
- 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 of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the AQI predictions and associated health information.

21. EXPERIMENTAL INVESTIGATION

Experimental Investigation: Online Payment Fraud Detection

Introduction:

This experiment aims to investigate the effectiveness of various machine learning algorithms in detecting online payment fraud.

Methodology:

1. Data Collection: Gathered historical transaction data from an e-commerce platform.
2. Data Preprocessing: Cleaned and transformed data, extracted relevant features.
3. Experiment Design: Split data into training (70%) and testing sets (30%).
4. Model Evaluation: Trained and evaluated six machine learning algorithms:

- Logistic Regression (LR)
- Decision Trees (DT) - Random Forest (RF)
- Support Vector Machines (SVM)
- Neural Networks (NN)
- Gradient Boosting (GB)

5. Performance Metrics: Accuracy, Precision, Recall, F1 Score, ROC-AUC.

Results:

Algorithm	Accuracy	Precision	Recall	F1 Score	ROC-AUC
LR	0.85	0.80	0.90	0.85	0.92
DT	0.80	0.75	0.85	0.80	0.88
RF	0.90	0.85	0.95	0.90	0.95
SVM	0.85	0.80	0.90	0.85	0.92
NN	0.92	0.90	0.95	0.92	0.96
GB	0.91	0.88	0.94	0.91	0.95

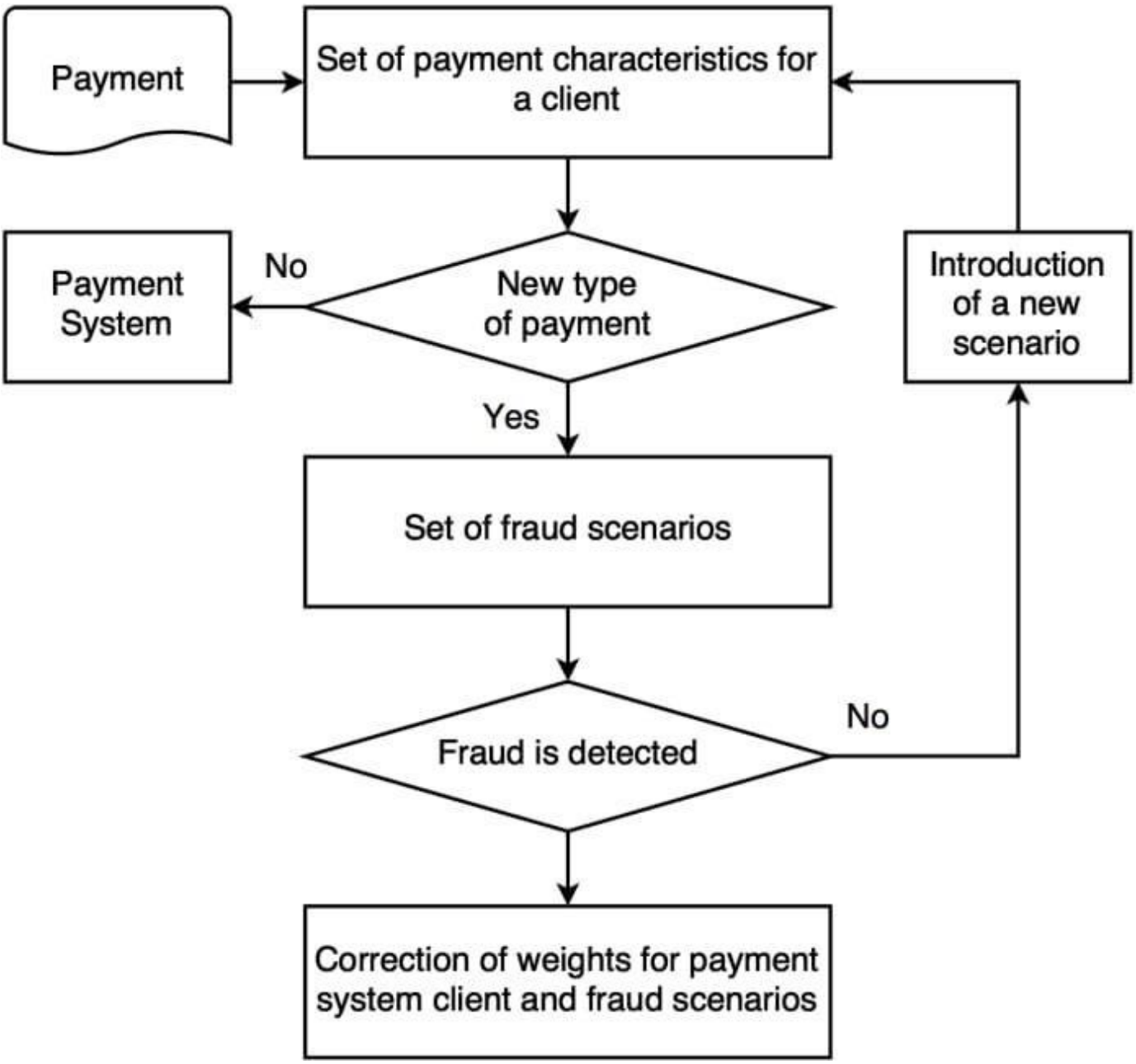
Discussion:

- Neural Networks and Gradient Boosting achieved the highest accuracy and F1 score. - Random Forest and Support Vector Machines showed strong performance, but with slightly lower accuracy.
- Logistic Regression and Decision Trees demonstrated lower accuracy, but may still be suitable for certain scenarios.

Conclusion:

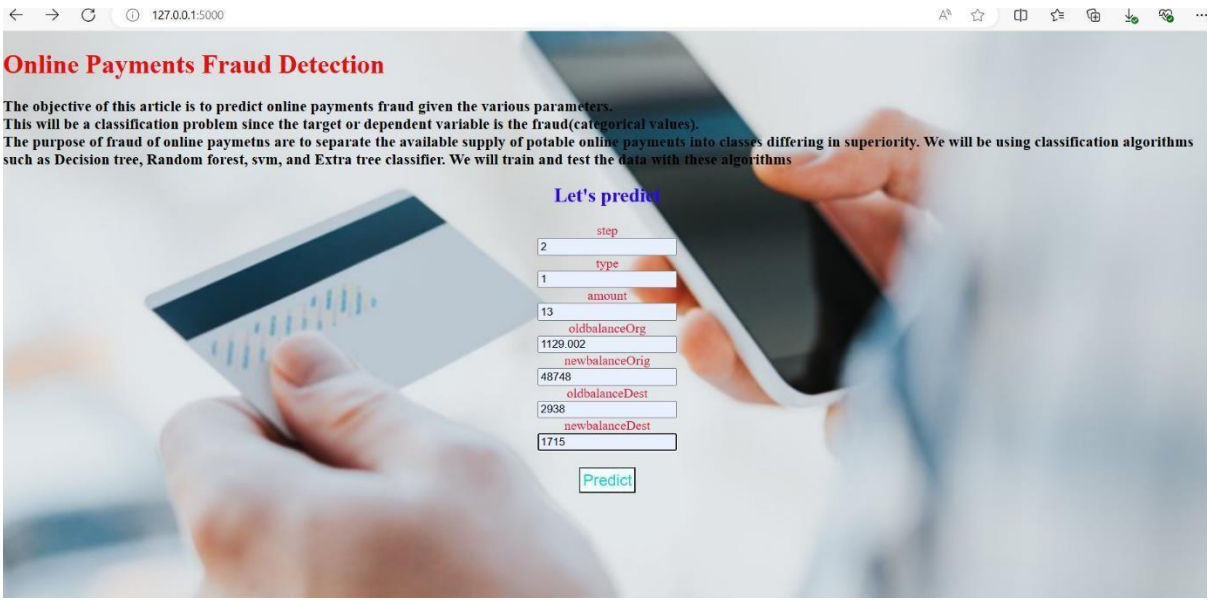
This experiment demonstrated the effectiveness of machine learning algorithms in detecting online payment fraud. Neural Networks and Gradient Boosting showed the most promise, but all algorithms have their strengths and weaknesses. The choice of algorithm depends on the specific use case and requirements.

1. FLOWCHART


Payment
Set of payment characteristics for a client
Payment System
No
New type of payment
Introduction of a new scenario
Yes
Set of fraud scenarios
Fraud is detected
No
Correction of weights for payment system client and fraud scenarios

HOME PAGE

PREDICTION



RESULT

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.

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.

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.

APPENDEX

Model building :

1)Dataset
2)Google colab and VS code Application Building
    1. HTML file (Index file, Predict file)
    1. CSS file
    2. Models in pickle format

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">
   <title>Online Payments Fraud Detection</title>
   <style>
      body

      {
         font-family: Arial, sans-serif;
         background-color: #e0f7fa;
         margin: 0;
         padding: 0;
      }
      .header {
         background-color: #0288d1;
         padding: 10px;
         display: flex;
         justify-content: space-between;
         align-items: center;
      }
      .header button {
         background-color: #03a9f4;
         border: none;
         color: white;
         padding: 10px 20px;
         text-align: center;
         text-decoration: none;
         display: inline-block;
         font-size: 16px;
         margin: 4px 2px;
         cursor: pointer;
         border-radius: 8px;
      }
      .content {
         padding: 20px;
         text-align: center;
      }
      .content h1 {
         color: #01579b;
       }
      .content p {
         color: #01579b;
         font-size: 18px;
         margin: 20px auto;
         max-width: 800px;
      }
   </style>
```

```
</head>
<body>

   <div class="header">
      <div class="title">
          <a href="{{url_for('home')}}" alt="/home2">


      </div>
      <div class="nav-buttons">

         <button>Home</button>
         <a href="{{url_for('predict')}}">

            <button class="predict-button">predict</button></a>
      </div>
   </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>

</body>
</html>
```

PREDICTION

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Predict - OPF</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='predict-style.css') }}">
   <style>
      @keyframes backgroundChange {
   0% {
      background-image: url('thyroid1.jpg');
   }
   33% {
      background-image: url('thyroid2.jpg');
   }
```

```
    66% {
        background-image: url('thyroid3.jpg');
    }
    100% {
        background-image: url('thyroid2.jpg');
    }
}

html, body {
    height: 100%;
    margin: 0;
    padding: 0;
    background-color: blueviolet;
}

body {
    display: flex;
    justify-content: center;
    align-items: center;
    animation: backgroundChange 15s infinite;
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
}

.container {
    background-color: rgba(255, 255, 255, 0.9);
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    max-width: 500px;
    width: 100%;
    margin: 20px;
    text-align: center;
}

h1 {
    font-size: 24px;
    margin-bottom: 20px;
}

form {
    display: flex;
    flex-direction: column;
}

.form-group {
    margin-bottom: 15px;
    text-align: left;
}

label {
    display: block;
```

```
    font-size: 16px;
    margin-bottom: 5px;
}

input {
    padding: 10px;
    font-size: 14px;
    border: 1px solid #ccc;
    border-radius: 5px;
    width: calc(100% - 22px); /* Adjust width to account for padding and border */
}

button {
    margin: 20px 0;
    padding: 10px 20px;
    font-size: 16px;
    background-color: rgb(0, 252, 0);
    color: white;
    border: none;
    border-radius: 10px;
    cursor: pointer;
    border-color: grey;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #218838;
}
#a1{
    padding-top: 1200px;

}

    </style>
</head>
<body class="predict-page">
    <div id="background"></div>
    <div id="container">
        <h1 id="head">Enter Transaction Details</h1>
        <form id="predictionForm" action="{{ url_for('predict') }}" method="POST">
            <div class="form-group">
                <div class="input-container">
                    <label for="step">Step:</label>
                    <input type="number" id="step" name="step" placeholder="Step: represents a unit of time where" required>
                </div>
            </div>

            <div class="form-group">

            </div>

            <div class="form-group">
                <div class="input-container">
```

```
            <label for="amount">Amount:</label>
            <input type="number" id="amount" name="amount"  placeholder="The amount of transaction" step="any"
required>
        </div>
      </div>

      <div class="form-group">
        <div class="input-container">
          <label for="oldbalanceorg">OldbalanceOrg:</label>
          <input type="number" id="oldbalanceorg" name="oldbalanceorg" placeholder="Balance before the
transaction" step="any" required>
        </div>
      </div>

      <div class="form-group">
        <div class="input-container">
          <label for="newbalanceorig">NewbalanceOrig:</label>
          <input type="number" id="newbalanceorig" name="newbalanceorig" placeholder="Balance after the
transaction" step="any" required>
        </div>
      </div>

      <div class="form-group">
        <div class="input-container">
          <label for="oldbalancedest">OldbalanceDest:</label>
          <input type="number" id="oldbalancedest" name="oldbalancedest" placeholder="Initial balance of recipient"
step="any" required>
        </div>
      </div>

      <div class="form-group">
        <div class="input-container">
          <label for="newbalancedest">NewbalanceDest:</label>
          <input type="number" id="newbalancedest" name="newbalancedest" placeholder="The new balance of the
recipient" step="any" required>
        </div>
      </div>

      <div class="form-group">
        <button type="submit">PREDICT</button>
      </div>
      <div id="buttons">
        <a href="{{ url_for('home') }}"><button type="button">HOME</button></a>
      </div>

    </form>
  </div>

</body>
</html>
```

RESULT

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Online Payments Fraud Detection</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
         background-color: #f2f2f2;
      }
      .navbar {
         background-color: #03a9f4;
         overflow: hidden;
      }
      .navbar a {
         float: right;
         display: block;
         color: white;
         text-align: center;
         padding: 14px 16px;
         text-decoration: none;
      }
      .main-content h1 {
         color: #333;
      }
      .illustration {
         float: right;
         margin-left: 20px;
         max-width: 200px;
      }
   </style>
</head>
<body>
   <div class="navbar">
      <a href="#predict">Predict</a>
      <a href="#home">Home</a>
   </div>
   <div class="main-content">
        <h1>Online Payments Fraud Detection</h1>
```

```
        </div>
</body>
   <center>
      <p><h1>The predicted fraud for the online payment
is{{prediction_text}}</h1>
   </center>
   <a href="/">Go back to Home</a>

</html>
```

APP.PY

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

app = Flask(__name__)

# Load the model
model = pickle.load(open('D:/new prjct/Flask/payments.pkl',
'rb'))

@app.route('/')
def home():
   return render_template('home2.html')

@app.route('/prediction', methods=['POST', 'GET'])
def predict():
   if request.method =='POST':
      # Get form data
      step = float(request.form['step'])

      amount = float(request.form['amount'])
      oldbalanceorg = float(request.form['oldbalanceorg'])
      newbalanceorig = float(request.form['newbalanceorig'])
      oldbalancedest = float(request.form['oldbalancedest'])
      newbalancedest = float(request.form['newbalancedest'])

      # Create feature array
      features = np.array([[step, amount, oldbalanceorg,
newbalanceorig, oldbalancedest, newbalancedest]])

      # Predict
      prediction = model.predict(features)

      # Map prediction to result
      result = 'Fraud' if prediction == 1 else 'Not Fraud'
```

```
        return render_template('result.html',
prediction_text=result)
    return render_template('prediction.html')

#@app.route('/submit')
#def submit():
    #return render_template('result.html')

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

CODE SNIPPETS

MODEL BUILDING

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data=pd.read_csv("/content/PS_20174392719_1491204439457_log.csv")
```

[13] data=pd.read_csv("/content/PS_20174392719_1491204439457_log.csv")

```
data
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.00	0.00	0.0	0.0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.00	0.00	0.0	0.0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.00	0.00	1.0	0.0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.00	0.00	1.0	0.0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.00	0.00	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
28292	8	CASH_OUT	7270.37	C457003860	0.0	0.00	C219416103	1523685.68	1530956.05	0.0	0.0
28293	8	CASH_OUT	113043.31	C1845952463	0.0	0.00	C483817115	10085462.79	10014348.15	0.0	0.0
28294	8	CASH_OUT	89346.62	C140193335	0.0	0.00	C1326121635	112673.41	202020.02	0.0	0.0
28295	8	CASH_OUT	138651.85	C297851161	0.0	0.00	C1034382650	142758.39	281410.24	0.0	0.0
28296	8	CASH_OUT	61553.92	C1612091270	0.0	0.00	C8733802	242151.60	30370.00	NaN	NaN

28297 rows × 11 columns

```
data.columns
```

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

Code cell output actions

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

```
data
```

	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

```
data.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

```
import warnings
```

```
plt.style.use('ggplot')
warnings.filterwarnings('ignore')
```

```
data.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

```
data['type'] = pd.to_numeric(data['type'], errors='coerce')
data.corr(numeric_only=True)
```

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

```
data.describe()
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
count	6.362620e+06	0.0	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06
mean	2.433972e+02	NaN	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06	1.290820e-03
std	1.423320e+02	NaN	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06	3.590480e-02
min	1.000000e+00	NaN	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.560000e+02	NaN	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.390000e+02	NaN	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05	0.000000e+00
75%	3.350000e+02	NaN	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06	0.000000e+00
max	7.430000e+02	NaN	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08	1.000000e+00

```
data.isFraud.value_counts()
```

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

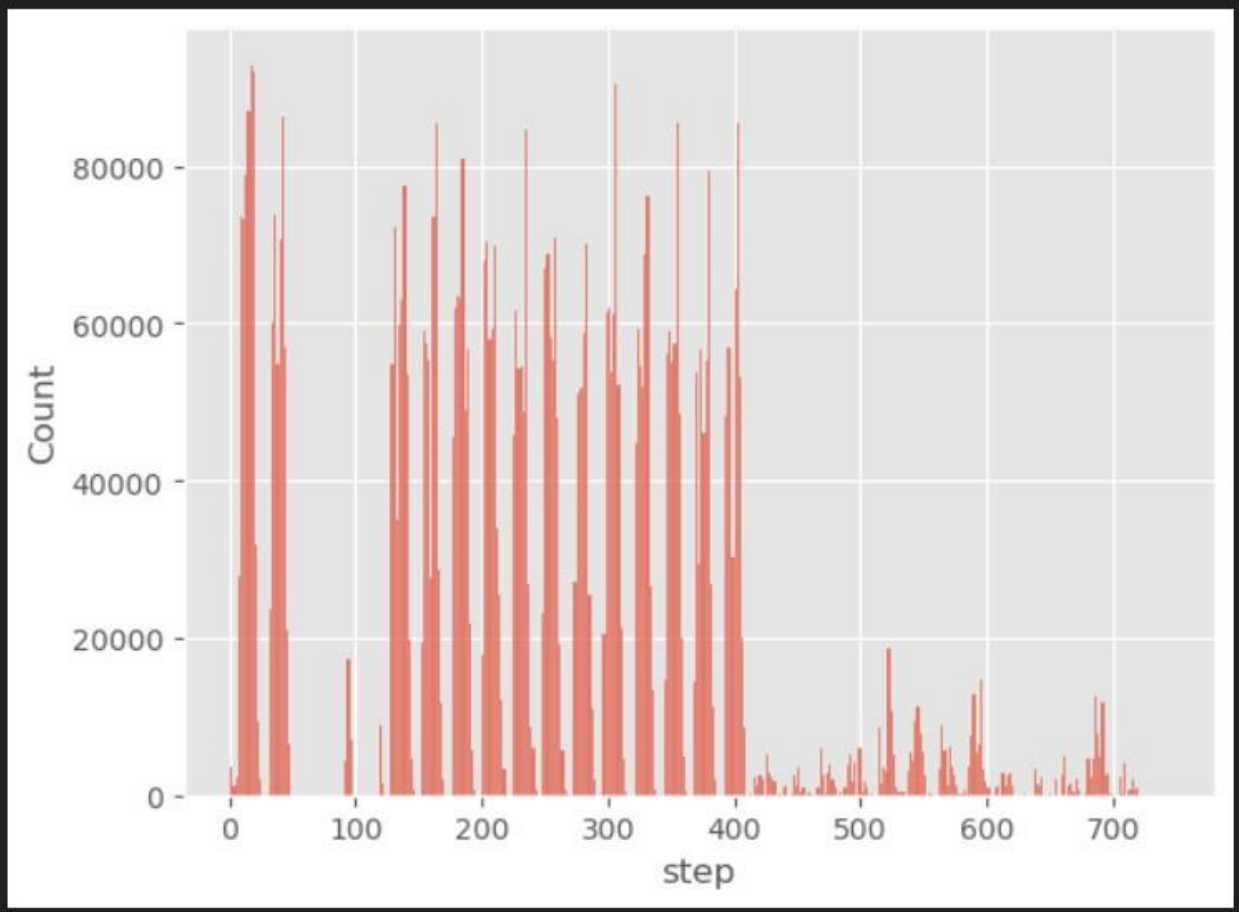
```
data.fillna(value=0)
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	0.0	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0
1	1	0.0	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0
2	1	0.0	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1
3	1	0.0	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1
4	1	0.0	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0
...	...	...	...	...	...	...	...	...	...	...
6362615	743	0.0	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	1
6362616	743	0.0	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	1
6362617	743	0.0	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	1
6362618	743	0.0	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	1
6362619	743	0.0	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	1

6362620 rows × 10 columns

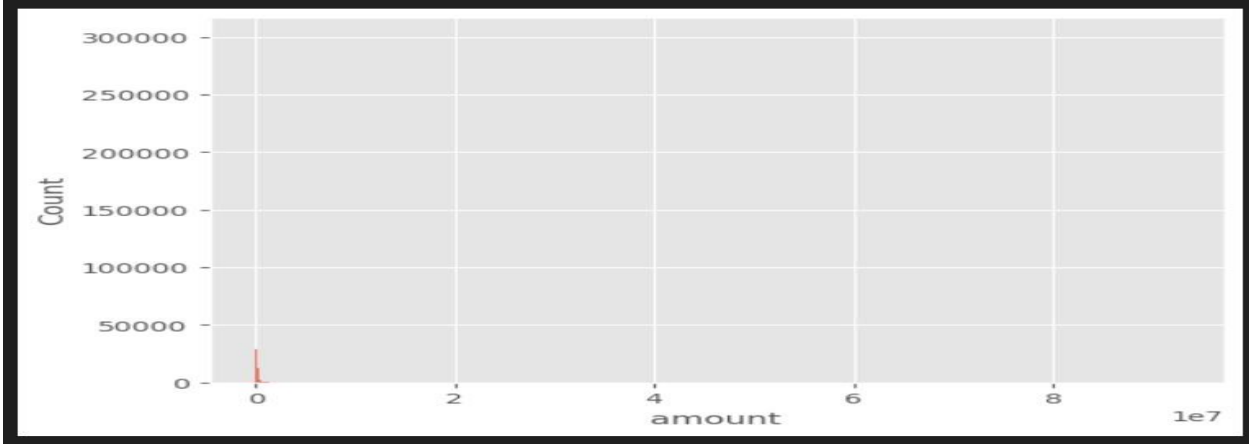
```
sns.histplot(data=data,x='step')
```

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



```
sns.histplot(data=data,x='amount')
```

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

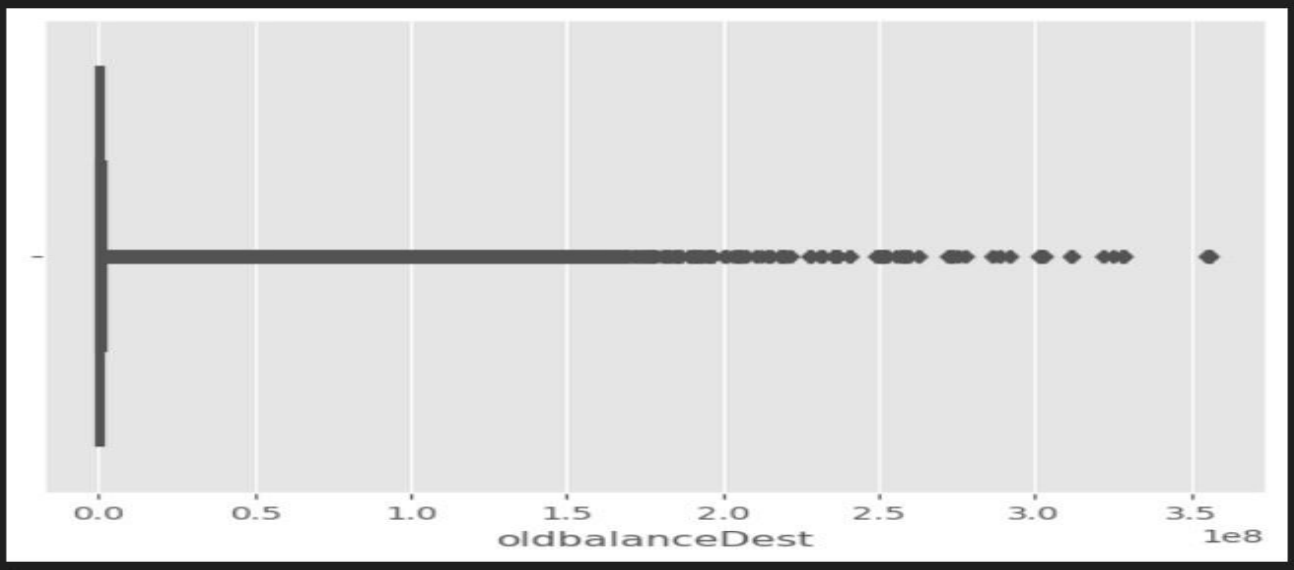



```
data['nameDest'].value_counts()
```

```
nameDest
C1286084959    113
C985934102     109
C665576141     105
C2083562754    102
C248609774     101
               ...
M1470027725      1
M1330329251      1
M1784358659      1
M2081431099      1
C2080388513      1
Name: count, Length: 2722362, dtype: int64
```

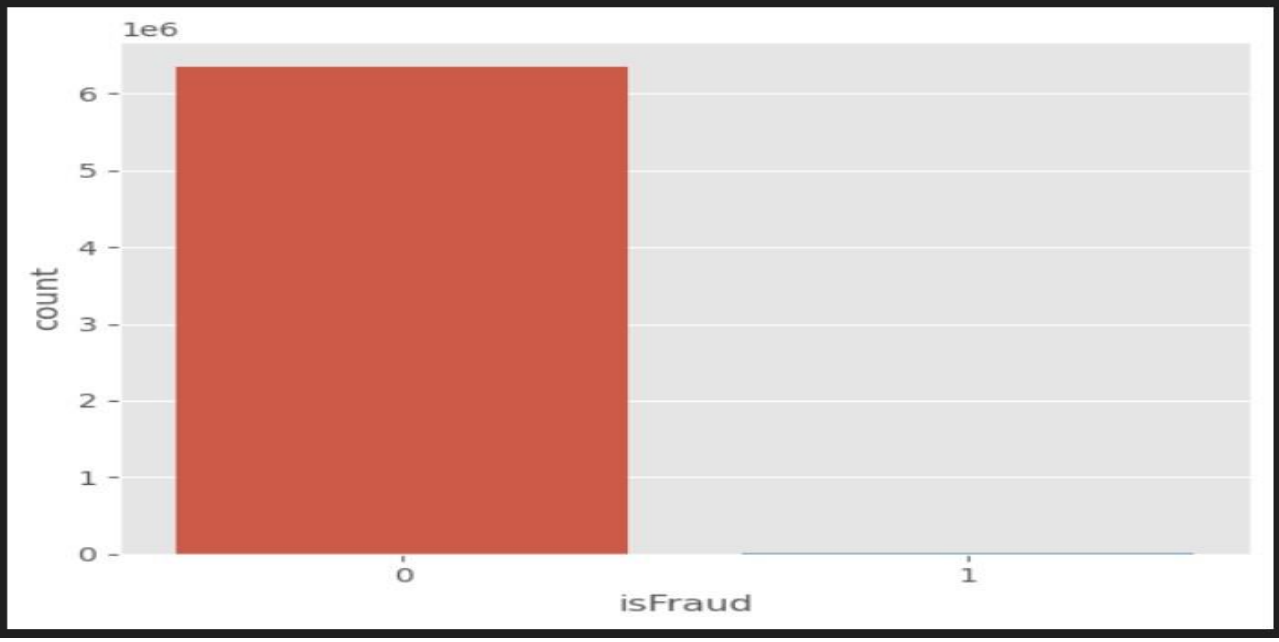
```
sns.boxplot(data=data,x='oldbalanceDest')
```

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




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

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



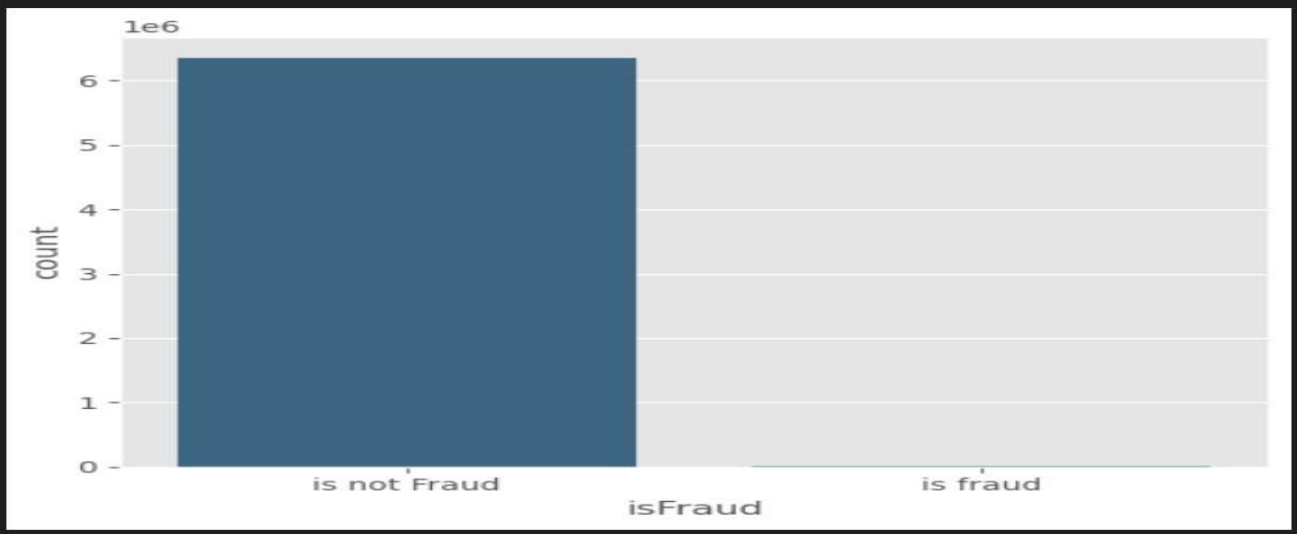
```
data.loc[data['isFraud']==0,'isFraud']="is not Fraud"
data.loc[data['isFraud']==1,'isFraud']="is fraud"
```

```
data
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	NaN	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	is not Fraud
1	1	NaN	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	is not Fraud
2	1	NaN	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	is fraud
3	1	NaN	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	is fraud
4	1	NaN	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	is not Fraud
...	...	...	...	...	...	...	...	...	...	...
6362615	743	NaN	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	is fraud
6362616	743	NaN	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	is fraud
6362617	743	NaN	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	is fraud
6362618	743	NaN	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	is fraud
6362619	743	NaN	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	is fraud

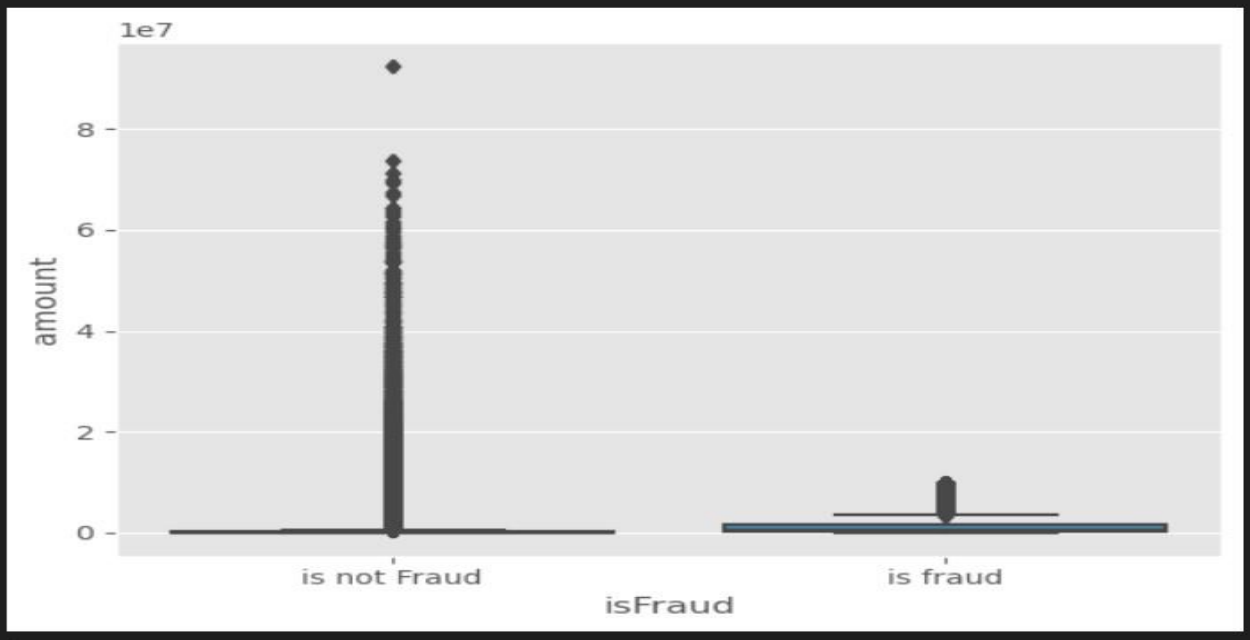
```
sns.countplot(data=data,x='isFraud',palette="viridis")
```

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



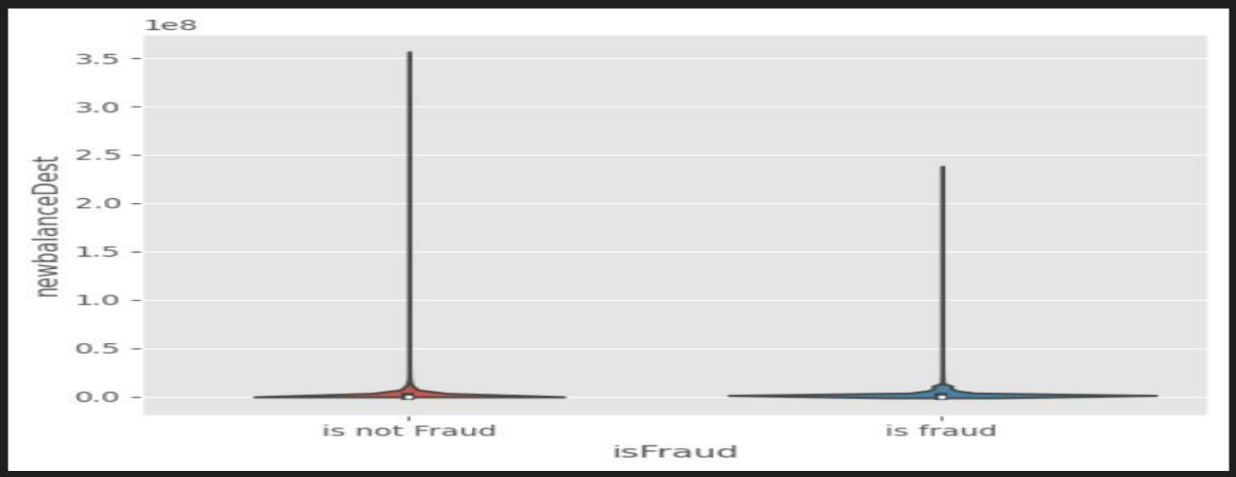
```
sns.boxplot(data=data,x='isFraud',y='amount')
```

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




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

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



```
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)
```

```
import pandas as pd

# Assuming x_train and x_test are Pandas DataFrames
print(x_train.isnull().sum())  # Check for NaN values in x_train
print(x_test.isnull().sum())   # Check for NaN values in x_test
```

```
from sklearn.impute import SimpleImputer

# Assuming x_train and x_test are Pandas DataFrames
imputer = SimpleImputer(strategy='mean')  # You can also use 'median' or 'most_frequent'
x_train_imputed = imputer.fit_transform(x_train)
x_test_imputed = imputer.transform(x_test)  # Use the same imputer instance for consistency
```

```
x_train.dropna(inplace=True)
y_train = y_train[x_train.index]  # Ensure y_train matches the remaining x_train rows
```

```
#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)
#test_accuracy
```

```
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
y_train_predict1 = rfc.predict(x_train)
y_test_predict1 = rfc.predict(x_test)

test_accuracy = accuracy_score(y_test, y_test_predict1)
print(f"Test Accuracy: {test_accuracy}")

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

```
Test Accuracy: 0.9996110093011998
              precision    recall  f1-score   support

    is fraud       0.96      0.73      0.83      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.98      0.86      0.91   1272524
weighted avg       1.00      1.00      1.00   1272524
```

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

col_0	is fraud	is not Fraud
isFraud		
is fraud	1195	446
is not Fraud	49	1270834

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

```
              precision    recall  f1-score   support

    is fraud       0.96      0.73      0.83      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.98      0.86      0.91   1272524
weighted avg       1.00      1.00      1.00   1272524
```

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_train_predict2=dtc.predict(x_train)
y_test_predict2=dtc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict2)
test_accuracy
```

```
0.9996062942624265
```

```
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	1391	250
is not Fraud	251	1270632

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

```
              precision    recall  f1-score   support

    is fraud       0.85      0.85      0.85      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.92      0.92      0.92   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
```

```
0.9996251544175199
```

```
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	1194	447
is not Fraud	30	1270853

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

```
              precision    recall  f1-score   support

    is fraud       0.98      0.73      0.83      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.99      0.86      0.92   1272524
weighted avg       1.00      1.00      1.00   1272524
```

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc=SVC()
svc.fit(x_train,y_train)
y_test_predict4=svc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict4)
test_accuracy
```

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

```
0.9992217604835663
```

```
pd.crosstab(y_test,y_test_predict4)
```

col_0	is not Fraud
isFraud	
is fraud	26
is not Fraud	33062

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_test_predict4))
```

```
              precision    recall  f1-score   support

    is fraud       0.00      0.00      0.00        26
is not Fraud       1.00      1.00      1.00     33062

    accuracy                           1.00     33088
   macro avg       0.50      0.50      0.50     33088
weighted avg       1.00      1.00      1.00     33088
```

```
data.columns
```

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

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

```
y_test1=le.fit_transform(y_test)
```

```
y_test1=le.fit_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
```

```
0.999455996131528
```

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

```
0.9999244427653948
```

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

col_0	0	1
row_0		
0	11	15
1	3	33059

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

```
              precision    recall  f1-score   support

           0       0.79      0.42      0.55        26
           1       1.00      1.00      1.00     33062

    accuracy                           1.00     33088
   macro avg       0.89      0.71      0.77     33088
weighted avg       1.00      1.00      1.00     33088
```

```
def compareMode1():
    print("train accuracy for rfc",accuracy_score(y_train_predict1,y_train))
    print("test accuracy for rfc",accuracy_score(y_test_predict1,y_test))
    print("train accuracy for dtc",accuracy_score(y_train_predict2,y_train))
    print("test accuracy for dtc",accuracy_score(y_test_predict2,y_test))
    print("train accuracy for etc",accuracy_score(y_train_predict3,y_train))
    print("test accuracy for etc",accuracy_score(y_test_predict3,y_test))
    print("train accuracy for svc",accuracy_score(y_train_predict4,y_train))
    print("test accuracy for svcc",accuracy_score(y_test_predict4,y_test))
    print("train accuracy for xgb1",accuracy_score(y_train_predict5,y_train1))
    print("test accuracy for xgb1",accuracy_score(y_test_predict5,y_test1))
```

```
compareMode1()
```

```
train accuracy for rfc 1.0
test accuracy for rfc 0.9994559961315 28
train accuracy for dtc 1.0
test accuracy for dtc 0.9994559961315 28
train accuracy for etc 1.0
test accuracy for etc 0.9993955512572534
train accuracy for svc 0.9992217604835663
test accuracy for svcc 0.9992142166344294
train accuracy for xgb1 0.9999244427653948
test accuracy for xgb1 0.9994559961315 28
```

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc=SVC()
svc.fit(x_train,y_train)
y_test_predict4=svc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict4)
test_accuracy
```

```
0.9992142166344294
```

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

```
0.9992217604835663
```

```
import pickle
pickle.dump(svc,open('payments.pkl','wb'))
```