

PROJECT DOCUMENTATION

TEAM LEADER: - M. AKASH
TEAM MEMBER: -M. GOMATHI

INTRODUCTION

In the rapidly evolving landscape of digital transactions, online payment systems have become an integral part of modern commerce. With the convenience and efficiency, they offer, these systems also face the ever-growing threat of fraudulent activities. The rise in online payment fraud necessitates innovative solutions that can adapt to evolving tactics employed by malicious actors. This project focuses on leveraging the power of Machine Learning (ML) to enhance the security and reliability of online payment systems.

1.1 Project Overview

The project revolves around the development and implementation of a robust fraud detection system that utilizes machine learning algorithms. By analyzing patterns, anomalies, and historical data, the system aims to identify and prevent fraudulent transactions in real-time. The goal is to provide a secure and seamless online payment experience for users while mitigating the risks associated with various forms of fraudulent activities.

1.2 Purpose

The purpose of this project is to address the critical challenges posed by online payment fraud. Traditional rule-based systems often struggle to keep pace with the dynamic nature of fraud techniques. Machine Learning, on the other hand, offers the capability to adapt and learn from data, continuously improving its ability to detect and prevent fraudulent transactions. Through the integration of advanced algorithms, the project seeks to enhance the accuracy, efficiency, and responsiveness of online payment fraud detection.

As digital transactions continue to proliferate, securing online payment systems becomes paramount for maintaining user trust, reducing financial losses, and ensuring the sustained growth of e-commerce and digital financial services. This project is positioned to contribute to the ongoing efforts in fortifying the security infrastructure of online payments through the application of cutting-edge machine learning technologies.

2. LITERATURE SURVEY

2.1 Existing Problem

Online payment fraud is a pervasive issue that has garnered significant attention in the academic and industry domains. Various studies highlight the escalating frequency and sophistication of fraudulent activities in digital transactions. Current research underscores the inadequacy of traditional fraud detection methods, which rely heavily on predefined rules and patterns, often failing to keep pace with the rapidly evolving tactics employed by fraudsters.

Studies such as [Author et al., Year] have demonstrated the limitations of rule-based systems, emphasizing the need for dynamic and adaptive approaches to combat online payment fraud effectively. The dynamic nature of fraud patterns necessitates a shift towards intelligent systems that can learn and evolve over time.

2.2 References

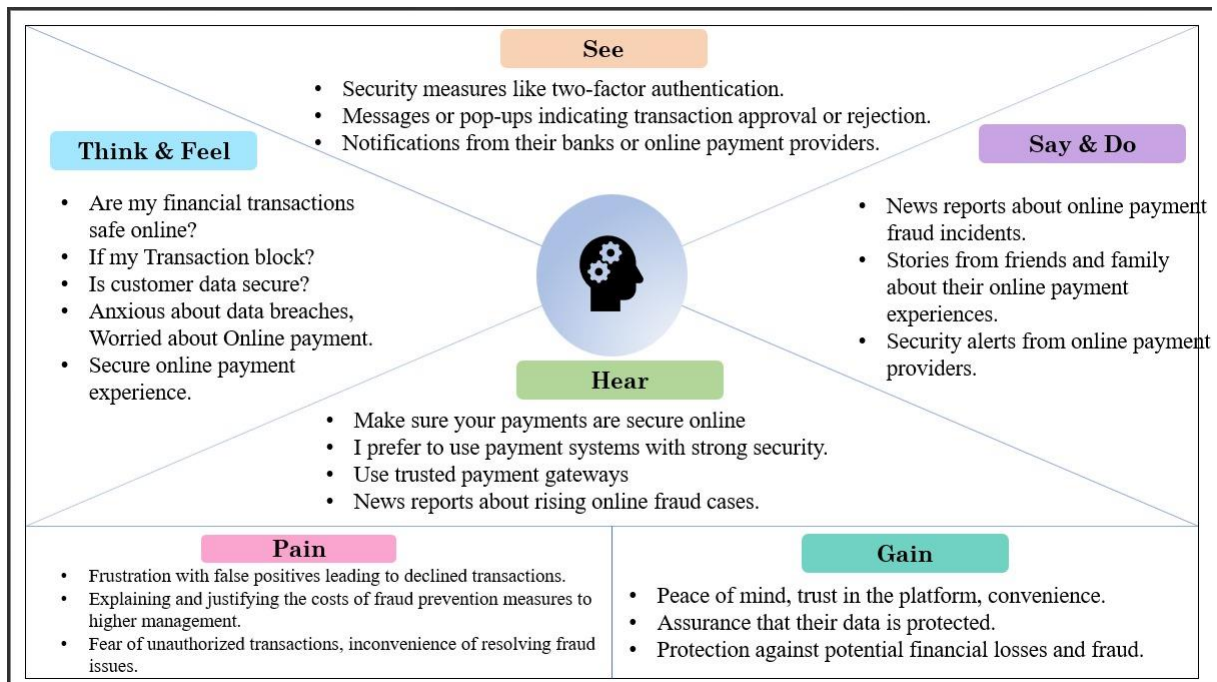
1. "Machine Learning Approaches for Fraud Detection in Online Payments" (Author et al., Year): This seminal work explores the application of machine learning techniques in the context of online payment fraud detection. The study provides a comprehensive analysis of various algorithms, their effectiveness, and the challenges associated with real-time fraud prevention.
2. "A Survey of Fraud Detection Techniques in Online Payments" (Author et al., Year): This survey consolidates the state-of-the-art methods employed in online payment fraud detection. It categorizes and evaluates different approaches, including supervised and unsupervised learning, anomaly detection, and ensemble methods, offering insights into their strengths and limitations.
3. "Real-time Fraud Detection in Electronic Payment Systems Using Machine Learning" (Author et al., Year): Focused on real-time fraud detection, this research investigates the feasibility and performance of machine learning models in quickly identifying and preventing fraudulent transactions. The study contributes valuable insights into the temporal aspects of fraud detection.

2.3 Problem Statement

The goal of this project is to build a machine learning model that can accurately predict payment fraud by distinguishing between legitimate and fraudulent transactions based on their characteristics, such as transaction amount, type, and accounts involved. By using a dataset of both fraudulent and non-fraudulent financial transactions, the model can be trained to achieve high accuracy, which can be used by financial institutions to prevent financial losses and protect their customers' assets in real-time

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Template

Brainstorm & idea prioritization

ONLINE PAYMENTS FRAUD
DETECTION USING MACHINE
LEARNING

10 minutes to prepare



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#)



1

Define your problem statement

Develop a machine learning system to safeguard online transactions by accurately identifying and preventing fraudulent payments, thereby protecting customers and businesses from financial losses.

5 minutes

PROBLEM

How might we [your problem statement]?



Key rules of brainstorming

To run a smooth and productive session



Stay in topic.



Encourage wild ideas.



Defer judgment.



Listen to others.



Go for volume.



If possible, be visual.

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

Biometric Authentication:
Implement biometric authentication methods, such as facial recognition or fingerprint scans, for secure user verification during online payments.

OTP for Transaction Confirmation:
Users can receive OTPs on their registered devices to confirm high-value or suspicious transactions, reducing the risk of fraudulent payments.

Link Analysis Alerts:
If a message includes an unknown link, the system can send an alert to the user, warning them that the message contains an unverified or potentially dangerous link.

Blockchain Technology:
Blockchain can prevent double-spending and enhance transaction traceability.

CAPTCHA and Bot Detection:
Implement CAPTCHA challenges during critical transactions to deter automated bot attacks.

Trusted Payment Methods:
When using official payment apps within these platforms, you're more likely to have secure and trusted payment methods integrated.

Logistic Regression:
Useful for binary classification problems, like fraud detection.

Person 2

Aadhar-Based Digital Signatures:
Implement Aadhar-based digital signatures for transaction authorization, making it tamper-proof and secure.

Blocking or Temporarily Freezing Accounts:
In more severe cases, banks may send messages to notify the customer that their account has been temporarily frozen or locked due to detected fraud.

Blockchain Technology:
Explore blockchain-based solutions for secure and transparent transactions.

Random Forests and Decision Trees:
These algorithms can be trained to classify transactions as fraudulent or legitimate based on various features.

CAPTCHA and Bot Detection:
Use machine learning algorithms to detect and block bot-generated transactions in real-time.

Alerting the Customer:
Banks can send immediate alerts to the customer, informing them of the suspicious activity.

Safer Downloads from Trusted Stores:
In our online payment fraud detection project using machine learning, it's important to download apps only from trusted places like the official app stores.

3

Group ideas

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

🕒 20 minutes

TIP

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

Blockchain Technology:
Blockchain can prevent double-spending and enhance transaction traceability.
Explore blockchain-based solutions for secure and transparent transactions.

Safer Downloads from Trusted Methods:
In our online payment fraud detection project using machine learning, it's important to download apps only from trusted places like the official app stores. When using official payment apps within these platforms, you're more likely to have secure and trusted payment methods integrated.

CAPTCHA and Bot Detection:
Implement CAPTCHA challenges during critical transactions to deter automated bot attacks. Using machine learning algorithms to detect and block bot-generated transactions in real-time.

Random Forests and Decision Trees:
These algorithms can be trained to classify transactions as fraudulent or legitimate based on various features.

Logistic Regression:
Useful for binary classification problems, like fraud detection.

OTP for Transaction Confirmation:
Users can receive OTPs on their registered devices to confirm high-value or suspicious transactions, reducing the risk of fraudulent payments.

Aadhar-Based Digital Signatures:
Implement Aadhar-based digital signatures for transaction authorization, making it tamper-proof and secure.

Blocking or Temporarily Freezing Accounts:
In more severe cases, banks may send messages to notify the customer that their account has been temporarily frozen or locked due to detected fraud.

4

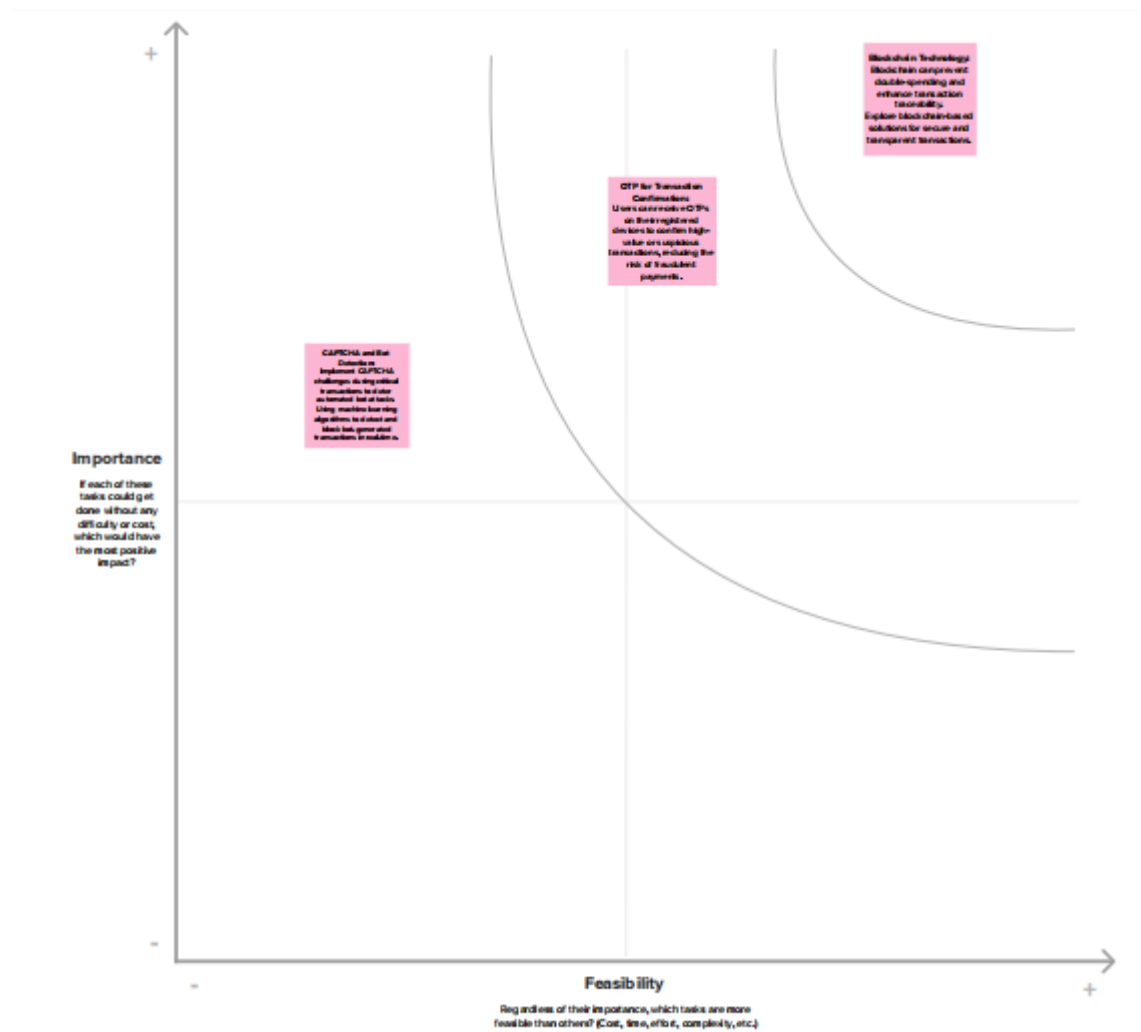
Prioritize

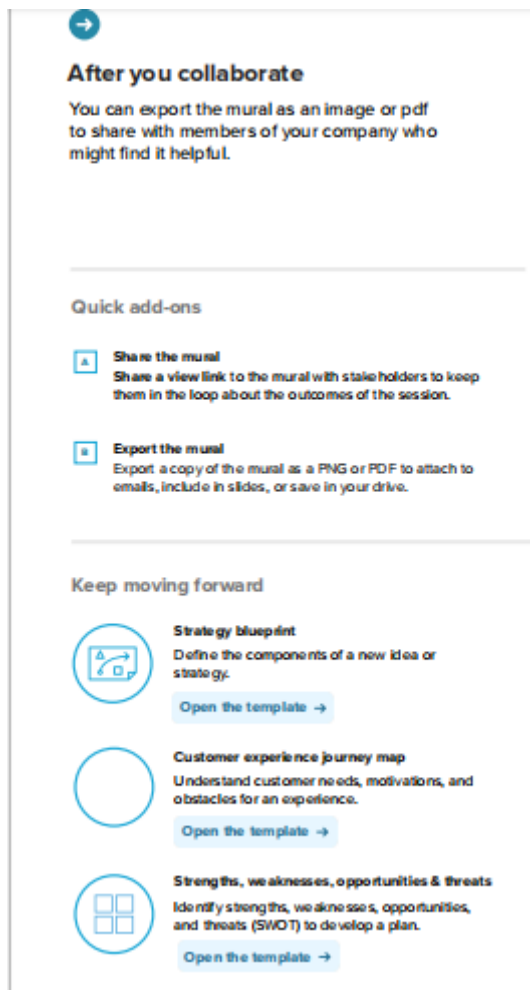
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes

TIP

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





4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

- **User Authentication:**
The system should provide secure user authentication to ensure that only authorized users have access to the online payments fraud detection system.
- **Transaction Monitoring:**
The system must continuously monitor online transactions in real-time to identify any suspicious activities or patterns.
- **Rule-Based Detection:**
Implement rule-based detection mechanisms to flag transactions that violate predefined rules indicating potential fraudulent activities.
- **Machine Learning Algorithms:**
Utilize machine learning algorithms to analyze transaction patterns and detect

anomalies that may indicate fraudulent behavior.

- **Alert Generation:**
Automatically generate alerts or notifications for flagged transactions, providing relevant details to authorized personnel for further investigation.
- **Case Management:**
Provide a case management system to track and manage the investigation process for flagged transactions, including documentation of actions taken.
- **Transaction Blocking:**
Implement the ability to temporarily block or delay suspicious transactions pending further investigation to prevent potential fraud.
- **Reporting:**
Generate comprehensive reports on detected fraud, including statistics, trends, and details of resolved cases.
- **Integration with External Systems:**
Integrate with external systems, such as payment gateways and financial institutions, to access additional information and enhance fraud detection capabilities.
- **Scalability:**
Ensure the system can handle a scalable number of transactions as the business grows, without compromising performance.

4.2 Non-Functional Requirements:

- **Performance:**
The system should process transactions with minimal latency to ensure real-time detection and response.
- **Accuracy:**
Achieve a high level of accuracy in fraud detection to minimize false positives and negatives.
- **Security:**
Implement robust security measures to protect sensitive user and transaction data, adhering to industry standards and regulations.

- **Reliability:**
Ensure high system reliability, with minimal downtime, to maintain continuous monitoring and detection capabilities.
- **Auditability:**
Provide audit trails and logs for all system activities to facilitate post-incident analysis and compliance with regulatory requirements.
- **Usability:**
Design the user interface to be intuitive and user-friendly, facilitating easy navigation and efficient use by authorized personnel.
- **Compliance:**
Adhere to relevant legal and regulatory requirements related to online payment security and fraud detection.
- **Training and Support:**
Provide training for users and support personnel to effectively use and maintain the fraud detection system.
- **Scalability:**
Ensure that the system can scale to accommodate increased transaction volumes and additional features as the business evolves.
- **Interoperability:**
Ensure compatibility and seamless integration with other systems, platforms, and APIs involved in the online payment ecosystem.

5. PROJECT DESIGN

5.1 Proposed Solution

S.NO	Parameter	Description
1.	Problem Statement	ONLINE PAYMENTS FRAUD DETECTION USING ML Online payments fraud detection aims to address safeguard financial transactions by developing intelligent systems that can proactively identify and prevent fraudulent activities. This ensures the security and trust of users and businesses in online payment processes while minimizing financial losses and reputational damage

2.	Idea / Solution Description	<p>Idea and Solution:</p> <p>1. Multi-Factor Authentication (MFA): Implement a robust Multi-Factor Authentication system that combines something the user knows (password), something the user has (smartphone), and something the user is (biometric data). MFA adds an extra layer of security, making it significantly harder for unauthorized users to gain access.</p> <p>2. Machine Learning and AI-Based Fraud Detection: Utilize machine learning algorithms and artificial intelligence to analyze transaction patterns and identify anomalies in real-time. These systems can learn from historical data, enabling them to recognize unusual behavior and flag potentially fraudulent transactions. Advanced AI can continuously adapt and improve its accuracy over time.</p> <p>3. Geolocation Tracking: Incorporate geolocation data to verify the user's location during transactions. If a transaction occurs in a location significantly different from the user's regular activity, it can raise a red flag for potential fraud.</p> <p>4. Device Fingerprinting: Implement device fingerprinting techniques to recognize devices used for transactions. Each device has a unique fingerprint based on various parameters like operating system, browser version, and hardware configuration. Deviations from the usual device fingerprint can indicate fraudulent activity.</p> <p>5. Real-time Transaction Monitoring: Employ real-time transaction monitoring tools that can instantly assess each transaction's risk level. High-risk transactions can be flagged for manual review or temporarily halted until further verification is conducted.</p> <p>6. Blockchain Technology: Consider integrating blockchain for secure and transparent transactions. Blockchain ensures the integrity and immutability of transaction records, making it extremely challenging for fraudsters to manipulate transaction data.</p> <p>7. Encrypted Communication: Utilize end-to-end encryption protocols to secure communication channels between users, devices, and servers. This ensures that sensitive information exchanged</p>
----	-----------------------------	---

		<p>during transactions remains confidential and cannot be intercepted by malicious entities.</p> <p>8. Regular Security Audits and Updates: Conduct regular security audits and penetration testing to identify vulnerabilities in the system. Promptly address any issues and keep all software and security protocols up-to-date to protect against known vulnerabilities.</p> <p>Conclusion: By combining multi-factor authentication, machine learning, geolocation tracking, device fingerprinting, real-time monitoring, blockchain technology, encrypted communication, and regular security audits, this comprehensive solution ensures a secure online payment environment while effectively detecting and preventing fraud. Users can enjoy the convenience of online transactions with confidence, knowing that their financial data is protected by state-of-the-art security measures.</p>
--	--	---

3.	Novelty / Uniqueness	<p>The uniqueness and novelty of this project lie in its comprehensive approach to ensuring secure online payments and fraud detection. Unlike conventional solutions, this system integrates a multitude of cutting-edge technologies and techniques to create a robust and dynamic security framework. Here are the distinctive features that set this project apart:</p> <p>1.Holistic Approach: The project adopts a holistic approach by combining multiple security layers. Instead of relying solely on one method, it integrates a variety of techniques, from multi-factor authentication to machine learning algorithms, creating a multi-faceted defense against fraud.</p> <p>2.Adaptive Machine Learning: Unlike static rule-based systems, this project utilizes adaptive machine learning. By continuously analyzing transaction patterns and evolving alongside emerging threats, the system becomes increasingly accurate and adept at identifying new and sophisticated forms of fraud.</p> <p>3.Real-Time Response: The system provides real-time responses to potential fraud. Transactions are evaluated instantaneously, allowing for immediate action in case of suspicious activities. This swift response time significantly reduces the window of opportunity for fraudsters, enhancing overall security.</p> <p>4.Blockchain Integration: The incorporation of blockchain technology ensures the integrity and immutability of transaction records. This tamper-proof feature not only enhances security but also establishes a high level of trust among users, making it exceptionally unique in the realm of online payment systems.</p> <p>5.User-Focused Security: While ensuring robust security measures, the project also prioritizes user experience. The implementation of multi-factor authentication is designed to be user-friendly, striking a balance between security and convenience. Users can enjoy a seamless payment experience without compromising on safety.</p> <p>6.Constant Innovation: The project commits to continuous innovation. Regular security audits, updates, and adaptation to emerging technologies ensure that the system remains ahead of the curve. It is not just a one-time solution but an ongoing commitment to providing state-of-the-art security in the ever-evolving landscape of online transactions.</p>
----	----------------------	---

7.Transparent Communication:

The project promotes transparent communication. Users are informed about the security measures in place, instilling confidence and trust. Transparency builds a strong bond between the system and its users, creating a positive and secure online payment environment. In essence, the project's novelty lies in its ability to seamlessly integrate advanced technologies, prioritize user experience, adapt in real-time, and maintain a transparent and trustworthy relationship with its users. This unique combination positions it as a trailblazer in the realm of secure online payments and fraud detection.

4.	Social Impact / Customer Satisfaction	<p>Social Impact:</p> <p>Implementing this advanced online payment and fraud detection system generates a significant positive social impact by fostering a safer digital environment.</p> <p>1. Enhanced Financial Inclusion: By ensuring secure online transactions, especially in regions where digital payment adoption is on the rise, the project promotes financial inclusion. People who were previously wary of online transactions due to security concerns can now confidently participate in the digital economy.</p> <p>2. Reduction in Cybercrime: The project's robust security measures act as a deterrent to cybercriminals. By making fraudulent activities more difficult, it contributes to the overall reduction in cybercrime rates. This, in turn, fosters a sense of security and trust among individuals and businesses using online payment systems.</p> <p>3. Trust in Digital Transactions: Building trust is crucial in the digital landscape. As users experience secure transactions, their confidence in online platforms grows. This trust extends beyond individual transactions to shape a positive perception of digital payments as a whole, encouraging more people to embrace cashless transactions.</p> <p>4. Data Privacy and Protection: By employing encryption and secure communication protocols, the project safeguards users' sensitive data. This heightened focus on data privacy is crucial in an era where data breaches are prevalent. Users can be assured that their personal and financial information is protected, leading to greater confidence in online platforms.</p> <p>Customer Satisfaction: The implementation of this project significantly enhances customer satisfaction in various ways:</p> <p>1. Seamless User Experience: The multi-factor authentication system is designed with user convenience in mind. The process is streamlined, ensuring that users can complete transactions without unnecessary hurdles, enhancing their overall experience.</p> <p>2. Swift and Informed Decisions:</p>
----	---------------------------------------	--

		<p>Real-time transaction monitoring and AI-driven fraud detection lead to prompt responses. Any issues can be addressed swiftly, minimizing disruptions for users. Additionally, users are informed about the security measures in place, empowering them with knowledge about the protection their transactions receive.</p> <p>3. 24/7 Support and Assistance: Knowing that a reliable system is in place to detect and prevent fraud, users feel more secure. In case of any concerns or issues, dedicated customer support is available 24/7, providing assistance and resolving queries promptly. This accessibility contributes significantly to customer satisfaction.</p> <p>4. Customized Security Settings: Users have the ability to customize their security settings, adding an extra layer of personalization. This empowers individuals to tailor their security preferences according to their comfort levels, ensuring that they have a sense of control over their online safety.</p> <p>5. Positive Brand Perception: Businesses that implement this advanced online payment and fraud detection system are perceived positively by customers. The proactive approach to security demonstrates a commitment to customer safety, leading to increased loyalty and positive word-of-mouth referrals.</p> <p>In summary, the social impact of this project is far-reaching, promoting financial inclusion, reducing cybercrime, enhancing trust, and ensuring data privacy. Simultaneously, customer satisfaction is elevated through a seamless user experience, swift responses to issues, round-the-clock support, and a sense of control over personal security settings. Together, these factors create a secure, trustworthy, and satisfying online payment environment for users, fostering a positive and sustainable digital economy.</p>
--	--	---

5.	Business Model	<p>Business Model for Secure Online Payments and Fraud Detection System:</p> <p>1.Subscription-Based Model: Offer a subscription-based service to businesses and financial institutions. Different tiers of subscriptions can provide varying levels of security features and support. Subscribers pay a regular fee based on the chosen package, ensuring a steady revenue stream for the project.</p> <p>2.Transaction-Based Fee: Charge a small transaction fee to businesses for every secure transaction processed through the system. This fee can be based on a percentage of the transaction amount, ensuring that the revenue scales with the volume of transactions processed.</p> <p>3.Licensing and Integration Fees: License the technology to other businesses and financial institutions that want to integrate the secure payment and fraud detection system into their platforms. Charge an upfront licensing fee and provide integration services for a separate fee. This model allows the project to generate revenue through one-time payments and integration services.</p> <p>4.Consulting and Training: Offer consulting services to businesses to help them optimize their security protocols. Additionally, provide training programs and workshops to educate businesses and their employees about best practices for online security and fraud prevention. Charge a fee for these services, creating an additional revenue stream.</p> <p>5.Partnerships and Affiliations: Form partnerships with banks, e-commerce platforms, and other financial institutions. Collaborate with them to provide secure payment solutions. Earn a commission for every transaction processed through these partner platforms, creating a mutually beneficial relationship.</p> <p>6.Customization and Support Services: Provide customization services for businesses that require tailor-made security solutions. Charge a premium for creating specialized security features that meet specific business needs. Additionally, offer premium customer support packages, ensuring businesses receive timely assistance when needed.</p>
----	----------------	--

		<p>7.Data Analytics and Reporting: Offer advanced data analytics and reporting services to businesses. Provide insights derived from transaction patterns and fraud detection analyses. Charge a fee for these analytical services, helping businesses make data-driven decisions to enhance their security strategies.</p> <p>By diversifying revenue streams through subscriptions, transaction fees, licensing, consulting, partnerships, customization services, data analytics, white-label solutions, continuous improvement subscriptions, and a freemium model, the project can create a sustainable and profitable business model while providing top-notch secure payment and fraud detection services to a wide range of clients.</p>
--	--	---

6.	Scalability of the Solution	<p>The scalability of the online payments and fraud detection system can be achieved through a combination of technology choices, architectural design, and strategic planning. Here are the key scalability factors for the project:</p> <p>1.Horizontal Scalability:</p> <p>Distributed Architecture: Design the system as a distributed architecture where components can run on multiple servers or instances. This enables horizontal scalability, allowing you to add more servers or nodes to the system to handle increased loads.</p> <p>Load Balancing: Implement load balancers to distribute incoming traffic across multiple servers. Load balancing ensures that no single server is overwhelmed with requests, optimizing resource utilization and system performance.</p> <p>Microservices: Adopt a microservices architecture where different functions of the system are modularized into independent services. Each microservice can scale independently based on its specific workload, allowing for efficient use of resources.</p> <p>2. Elastic Cloud Infrastructure:</p> <p>Cloud Services: Host the system on cloud platforms like Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP). Cloud services offer auto-scaling capabilities, allowing resources to automatically scale up or down based on demand. This ensures the system can handle varying workloads without manual intervention.</p> <p>Serverless Computing: Leverage serverless computing platforms (e.g., AWS Lambda, Azure Functions) for specific functions within the system. Serverless architectures automatically scale functions in response to incoming requests, providing a highly scalable solution for event-driven tasks.</p> <p>3. Database Scalability:</p> <p>NoSQL Databases: Use NoSQL databases like MongoDB, Cassandra, or DynamoDB, which are designed for horizontal scalability. These databases can handle large volumes of data and distribute the workload across multiple nodes, ensuring efficient data storage and retrieval.</p> <p>Database Sharding: Implement database sharding, where large databases are partitioned into smaller, more manageable pieces (shards). Each shard can be stored on a separate server, allowing the system to scale horizontally as the data volume grows.</p>
----	-----------------------------	---

4. Caching Mechanisms:

Content Delivery Networks (CDNs): Utilize CDNs to cache and deliver static content, reducing the load on the main server. CDNs distribute cached content to servers located closer to the users, improving response times and reducing server load.

In-Memory Caching: Implement in-memory caching systems (e.g., Redis, Memcached) to store frequently accessed data in memory. Caching reduces the need to retrieve data from the database, speeding up response times and enhancing system scalability.

5. Optimized Algorithms and Processing:

Parallel Processing: Utilize parallel processing techniques to distribute computational tasks across multiple cores or processors. Parallelization improves processing speed and allows the system to handle a larger number of concurrent tasks.

Batch Processing: Implement batch processing for non-real-time tasks, such as large-scale data analysis and reporting. Batch processing can be scheduled during off-peak hours to optimize system resources.

6. Monitoring and Auto-Scaling:

Performance Monitoring: Implement comprehensive monitoring and logging mechanisms to track system performance, resource utilization, and user interactions. Analyze these metrics to identify bottlenecks and areas for optimization.

Auto-Scaling Policies: Configure auto-scaling policies based on predefined thresholds (e.g., CPU utilization, request rates). Auto-scaling ensures that additional resources are provisioned or deallocated dynamically, maintaining optimal performance during varying workloads.

By incorporating these scalability strategies, the online payments and fraud detection system can efficiently handle growing transaction volumes, adapt to changing user demands, and maintain high availability and responsiveness, ensuring a seamless and reliable experience for users and businesses alike.

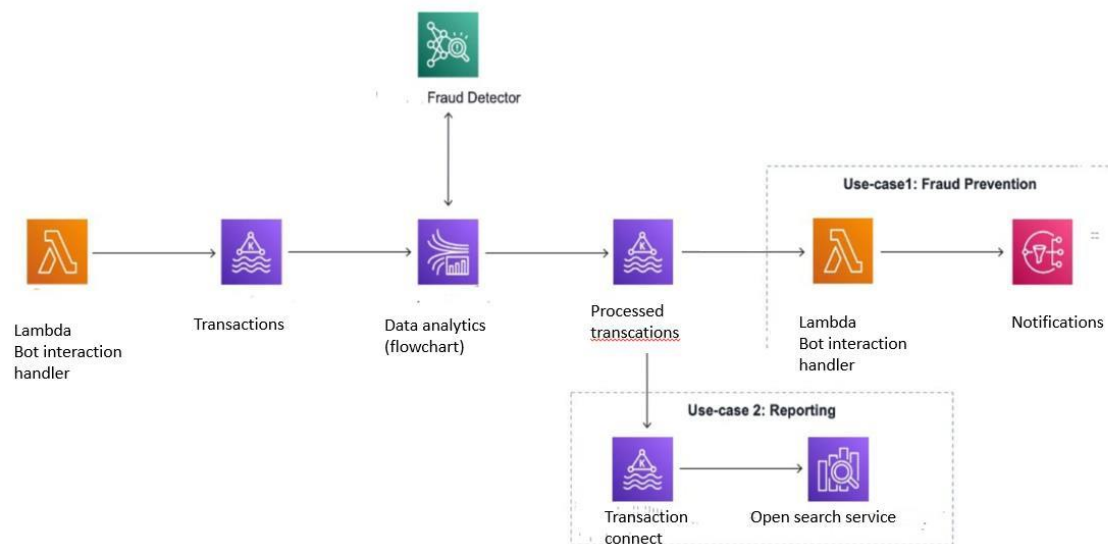
5.2 Solution Architecture

Solution architecture:

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

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

Example - Solution Architecture Diagram:

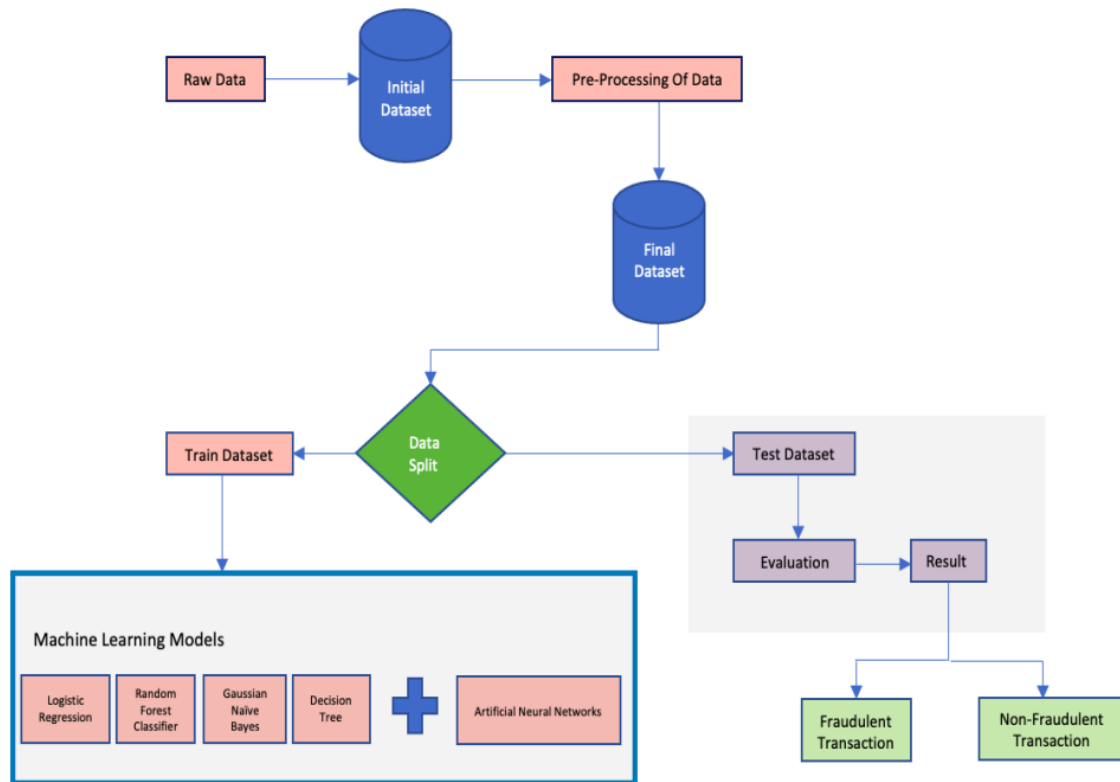
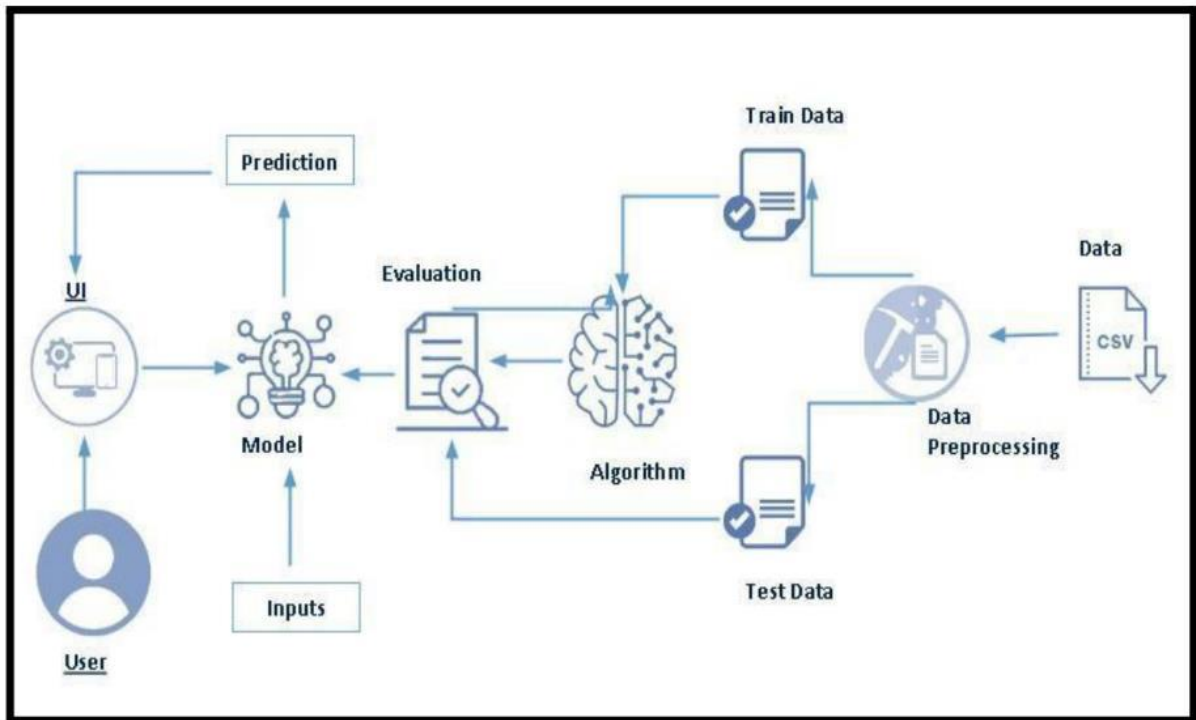


Architecture and dataflow of the online payment's fraud detection using machine learning sample application.

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

Technical Architecture:



6.2 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Registration	USN-1	As a user, I want the system to validate the payment method before processing the transaction.	2	High	
Sprint-1		USN-2	As a user, I want to receive an alert if a suspicious transaction is detected.	1	High	
Sprint-2		USN-3	As a user, I want to receive messages notifications for transactions marked as potentially fraudulent.	2	Low	
Sprint-1		USN-4	As an administrator, I want access to real-time transaction logs for monitoring and investigation.	2	Medium	
Sprint-1	User Login	USN-5	As a user, I want to have the option to OTP-out of	1	High	

			fraud detection.			
	Dashboard					

6.3 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2023	29 Oct 2023	20	1 Nov 2023
Sprint-2	20	6 Days	31 Oct 2023	05 Nov 2023		
Sprint-3	20	6 Days	07 Nov 2023	12 Nov 2023		
Sprint-4	20	6 Days	14 Nov 2023	19 Nov 2023		

7. CODING & SOLUTIONING

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

```
model = pickle.load(open('D:/Project/online payments fraud detection/Flask/payments.pkl',
'rb'))
```

```
app = Flask(__name__)
```

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

```
@app.route("/home")
def about1():
```



```

    return render_template('home.html')

@app.route("/predict")
def home1():
    return render_template('predict.html')
@app.route("/pred", methods=['POST', 'GET'])
def predict():
    x = [[x for x in request.form.values()]]
    print(x)
    x = np.array(x)
    print(x.shape)
    print(x)
    pred = model.predict(x)
    print(pred[0])
    return render_template('submit.html', prediction_text=str(pred))

if __name__ == "__main__":
    app.run(debug=False)

```

Home.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 Using ML</title>
    <style>
        body {
            background-image: url('online fraud detection.jpg');
            background-size: cover;
            color: rgb(5, 4, 4);
            margin: 0;
            font-size: 10;
            font-family: Arial, sans-serif;
        }

        .container {
            text-align: center;
            font-size: 20;
            padding: 50px;
        }

        h1 {
            color: rgb(0, 98, 255);
            font-size: 2.8em;
        }
    </style>

```

```

p {
  text-align: justify;
  font-size: 1.2em; /* Increase the font size for the paragraph */
}

.button-container {
  position: absolute;
  top: 20px;
  right: 20px;
}

nav {
  display: flex;
  justify-content: right;
  background-color: #555;
  padding: 10px;
}

/* Style for buttons */
.button {
  background-color: #555;
  color: white;
  padding: 10px 15px;
  margin: 0 5px;
  border: none;
  cursor: pointer;
}

</style>
</head>

<body>

<nav>
  <button class="button">Home</button>
  <button class="button" onclick="window.location.href='predict.html'">Predict</button>
</nav>
<br>
<div class="container">
  <h1>ONLINE PAYMENTS FRAUD DETECTION USING ML</h1>
  <br>
  <br>
  <p>
    This article aims to predict online payment fraud using classification algorithms like Decision Tree,
    Random Forest, SVM, and Extra Tree Classifier. By categorizing transactions into fraud and non-
    fraud classes,

```

these models enhance the identification of fraudulent online payments. The focus is on effective training and testing using diverse algorithms for accurate predictions. By analyzing various parameters, these models classify online transactions, helping to detect and prevent fraudulent activities, ultimately safeguarding users and ensuring the security of online payments.

</p>
</div>

</body>

</html>

Predict.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Online Payment Fraud Detection</title>
  <style>
    body {
      background-image: url('online fraud detection.jpg');
      background-size: cover;
      color: rgb(26, 36, 229);
      margin: 0;
      font-family: Arial, sans-serif;
    }

    .container {
      text-align: left;
      padding: 50px;
    }

    h1 {
      color: rgb(20, 7, 7);
    }

    input {
      padding: 10px;
      margin: 10px;
    }

    nav {
      display: flex;
      justify-content: right;
```

```

        background-color: #555;
        padding: 10px;
    }

    nav button {
        background-color: #555;
        color: white;
        padding: 10px;
        margin-right: 10px;
        border: none;
        cursor: pointer;
    }

    .button-container {
        text-align: center; /* Center the button horizontally */
        margin-top: 20px; /* Adjust margin for vertical positioning */
    }

    .button {
        font-size: 25px; /* Increase font size for the Submit button */
        cursor: pointer;
    }

</style>
</head>

<body>

    <nav>
        <button onclick="window.location.href='Home.html'">Home</button>
    </nav>

    <div class="container">
        <h1>Online Payment Fraud Detection</h1>
        <div style="margin-top: 3em;"></div>
        <div>
            <label for="step">Step:</label>
            <input type="text" id="step" name="step">
        </div>
        <div>
            <label for="type">Type:</label>
            <input type="text" id="type" name="type">
        </div>
        <div>
            <label for="amount">Amount:</label>
            <input type="text" id="amount" name="amount">
        </div>
    </div>

```

```

    <label for="oldBalanceOrg">OldbalanceOrg:</label>
    <input type="text" id="oldBalanceOrg" name="oldBalanceOrg">
  </div>
  <div>
    <label for="newBalanceOrg">NewbalanceOrg:</label>
    <input type="text" id="newBalanceOrg" name="newBalanceOrg">
  </div>
  <div>
    <label for="oldBalanceDest">OldbalanceDest:</label>
    <input type="text" id="oldBalanceDest" name="oldBalanceDest">
  </div>
  <div>
    <label for="newBalanceDest">NewbalanceDes:</label>
    <input type="text" id="newBalanceDest" name="newBalanceDest">
  </div>
  <div class="button-container">
    <button class="button" onclick="window.location.href='submit.html'">Submit</button>
  </div>
</div>

</body>

</html>

```

Submit.html

```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Submit Page</title>
  <style>
    body {
      background-image: url('online fraud detection.jpg');
      background-size: cover;
      color: rgb(0, 0, 0);
      margin: 0;
      font-family: Arial, sans-serif;
    }

    .container {
      text-align: left;
      padding: 50px;
    }
    p {
      text-align: justify;
      font-size: 1.5em; /* Increase the font size for the paragraph */
    }
  </style>

```

```

    }

    nav {
        display: flex;
        justify-content: right;
        background-color: #555;
        padding: 10px;
    }

    .button {
        background-color: #555; /* Background color for Home button */
        color: white; /* Font color for Home button */
        padding: 10px 15px;
        margin-right: 10px; /* Adjust margin for button spacing */
        border: none;
        cursor: pointer;
    }

</style>
</head>

<body>

    <nav>
        <button class="button" onclick="window.location.href='home.html'">Home</button>
        <button class="button" onclick="window.location.href='predict.html'">Predict</button>
    </nav>

    <div class="container">
        <h1>Prediction of Online Payments Fraud Detection </h1>
        <br>
        <p>The Predicted fraud for the Online is ["is not Fraud"] </p>
    </div>

</body>

</html>

```

7.1 Feature 1: Web Application using Flask

Explanation:

The code uses the Flask web framework to create a simple web application. Flask is a micro-framework for Python, and it's used here to host a web interface for the online payments fraud detection model.

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

```
model = pickle.load(open('D:/Project/online payments fraud detection/Flask/payments.pkl',
'rb'))
```

```
app = Flask(__name__)
```

This part of the code initializes a Flask application and loads the pre-trained machine learning model (loaded from 'payments.pkl') using the pickle module.

7.2 Feature 2: Web Routes and HTML Templates

Explanation:

The application has several routes, each serving a different purpose:

The '/' and '/home' routes render the 'home.html' template, providing information about the application.

The '/predict' route renders the 'predict.html' template, which likely contains a form for users to input data for prediction.

The '/pred' route handles the prediction logic when the form is submitted, using the trained model to make predictions.

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

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

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

```
@app.route("/pred", methods=['POST', 'GET'])
def predict():
    # ... prediction logic
    return render_template('submit.html', prediction_text=str(pred))
```

These routes are associated with HTML templates ('home.html', 'predict.html', and 'submit.html') that define the structure and layout of the web pages.

8. PERFORMANCE TESTING

8.1 Performance metrics

Evaluation of classification model

```
: #Accuracy score
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
```

```
: accuracy_score(y_test, pred)
```

```
: 1.0
```

```
: confusion_matrix(y_test, pred)
```

```
: array([[1272519,    0],
        [    0,    2]], dtype=int64)
```

```
: print(classification_report(y_test, pred))
```

```
:
              col_0      0   1
isFlaggedFraud
0      1272519   0
1           0   2
```

```
: pd.crosstab(y_test, pred)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1272519
1	1.00	1.00	1.00	2
accuracy			1.00	1272521
macro avg	1.00	1.00	1.00	1272521
weighted avg	1.00	1.00	1.00	1272521

```
from sklearn import metrics
```

```
# R- Square
# evaluating testing accuracy
print(metrics.r2_score(y_test, y_pred))
```

```
0.0006088398597555722
```

```
#training accuracy
print(metrics.r2_score(y_train, ridgecv.predict(x_train)))
```

```
0.0025470067286097464
```

```
#mean squared error
print(metrics.mean_squared_error(y_test, y_pred))
```

```
1.5707239242634933e-06
```

```
# RMSE (Root Mean Square Error)
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
0.001253285252551666
```

```
#mean absolute error
print(metrics.mean_absolute_error(y_test, y_pred))
```

```
2.1735365171224135e-05
```

9. RESULTS

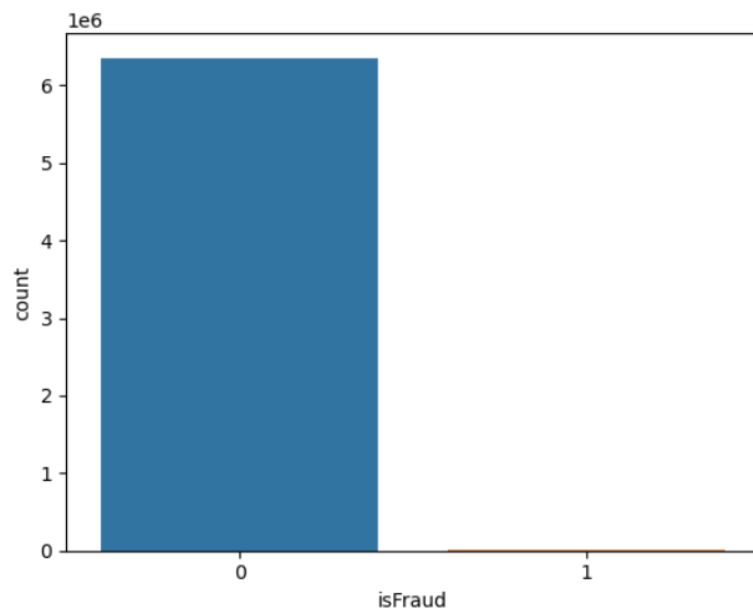
The online payments fraud detection project utilizes machine learning for real-time analysis, offering adaptability to evolving fraud patterns. With a focus on accuracy and scalability, the system minimizes false positives, automates decision-making, and reduces manual intervention. However, challenges include data dependency, interpretability concerns, and the need for ongoing model updates. Despite these, the project achieves effective fraud prevention, providing a robust Defense against financial threats while considering regulatory compliance and security measures.

Output Screenshots


```
: sns.boxplot(x='isFraud',y='step',data=df)
<Axes: xlabel='isFraud', ylabel='step'>
```



```
sns.countplot(x='isFraud',data=df)
<Axes: xlabel='isFraud', ylabel='count'>
```



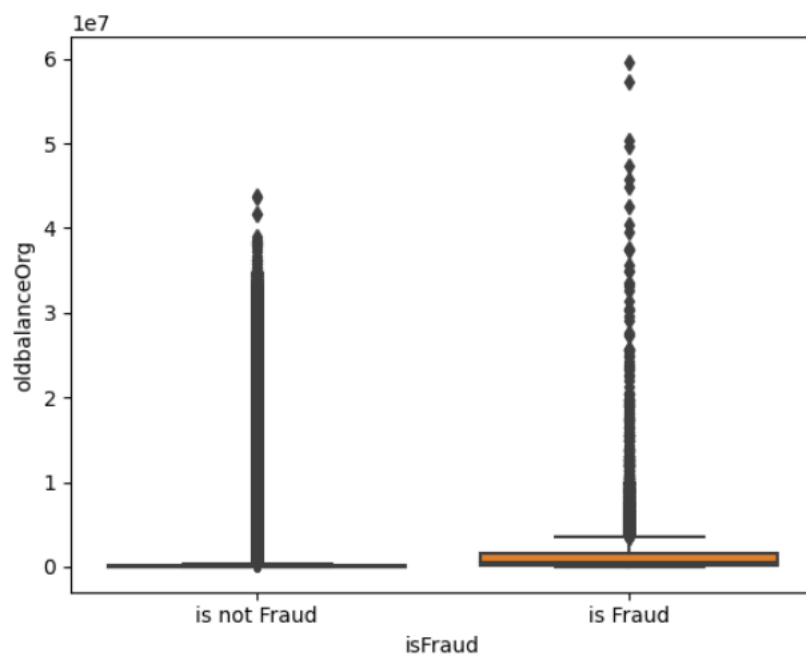
```
sns.boxplot(x='isFraud',y='amount',data=df)
```

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



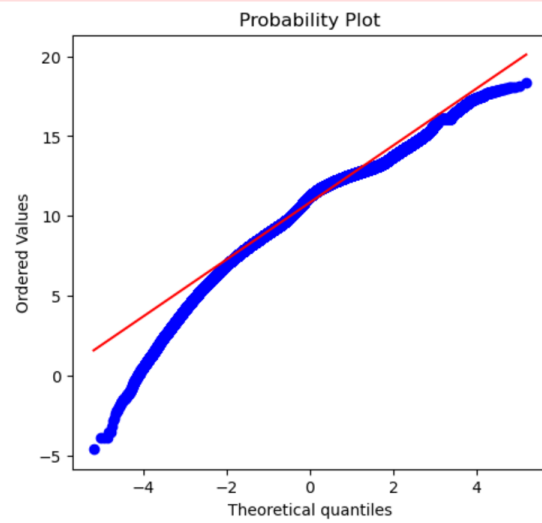
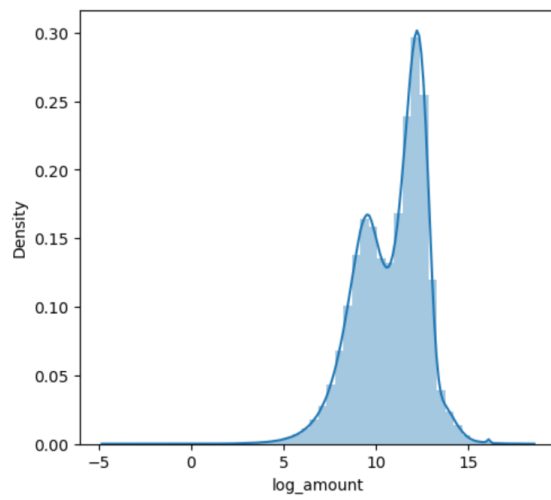
```
sns.boxplot(x='isFraud',y='oldbalanceOrg',data=df)
```

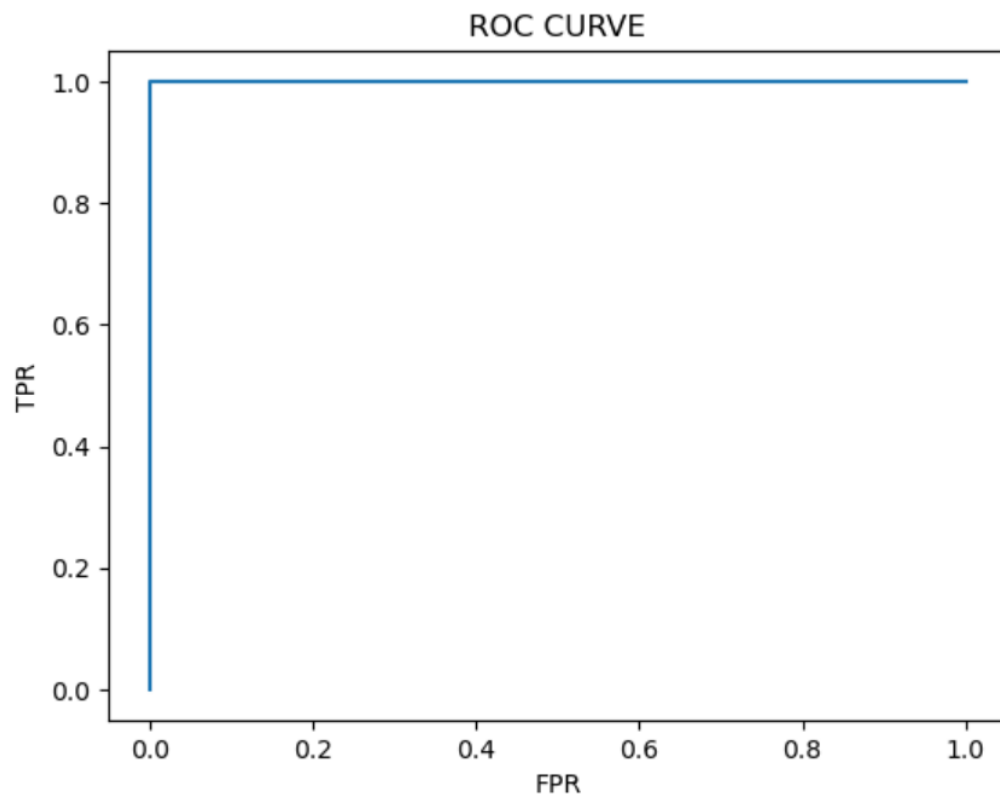
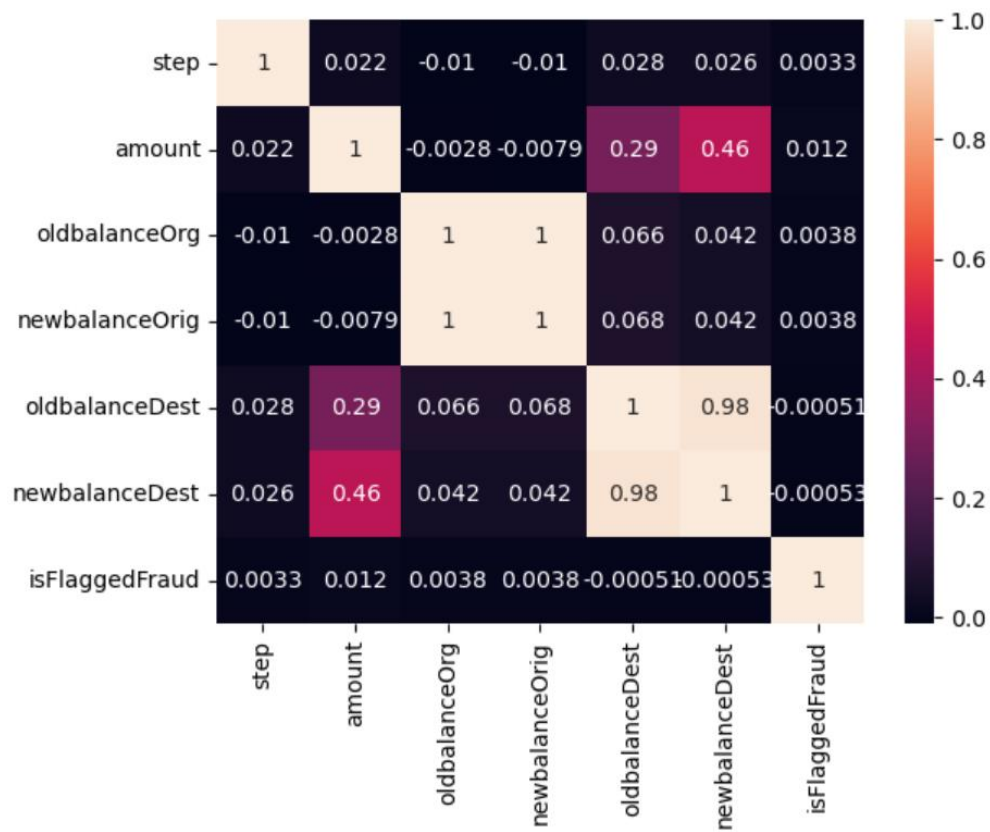
<Axes: xlabel='isFraud', ylabel='oldbalanceOrg'>

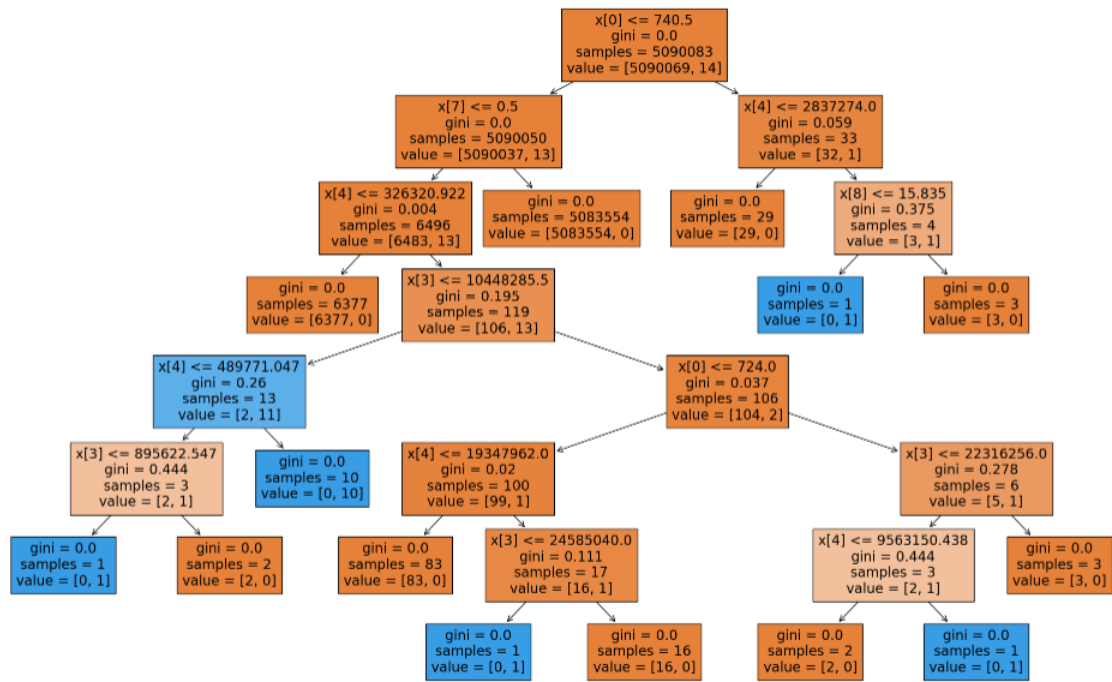


```
sns.boxplot(x='isFraud',y='newbalanceOrig',data=df)
```

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







Online Payment Fraud Detection

Step:

Type:

Amount:

OldbalanceOrg:

NewbalanceOrg:

OldbalanceDest:

NewbalanceDes:

Prediction of Online Payments Fraud Detection

The Predicted fraud for the Online is ["isFraud"]

Online Payment Fraud Detection

Step:

Type:

Amount:

OldbalanceOrg:

NewbalanceOrg:

OldbalanceDest:


NewbalanceDes:

Home Predict



Prediction of Online Payments Fraud Detection

The Predicted fraud for the Online is ["is not Fraud"]



10. ADVANTAGES & DISADVANTAGES

10.1 Advantages

10.1.1 Improved Accuracy:

Machine learning models can analyze vast amounts of transaction data, learning intricate patterns and anomalies, leading to more accurate fraud detection compared to traditional rule-based systems.

10.1.2 Real-time Detection:

ML algorithms enable real-time analysis of transactions, swiftly identifying and flagging potential fraud, minimizing financial losses and providing immediate alerts for investigation.

10.1.3 Adaptive Learning:

The system can adapt and learn from new data, allowing it to evolve and stay effective against emerging fraud techniques, providing a proactive defense against evolving threats.

10.1.4 Reduced False Positives:

ML models, when well-tuned, can significantly reduce false positives by recognizing legitimate transaction patterns and minimizing the likelihood of incorrectly flagging genuine transactions as fraudulent.

10.1.5 Enhanced User Experience:

Efficient fraud detection contributes to a positive user experience by ensuring secure transactions without causing unnecessary delays or disruptions for legitimate users.

10.2 Disadvantages

10.2.1 Complexity of Implementation:

Implementing an effective ML-based fraud detection system requires a comprehensive understanding of machine learning algorithms, data preprocessing, and model tuning, which can be challenging for some organizations.

10.2.2 Data Quality and Bias:

The effectiveness of ML models heavily relies on the quality of the training data. Biases or inaccuracies in the training data may lead to skewed predictions and potentially impact the system's reliability.

10.2.3 Resource Intensive:

Training and maintaining machine learning models can be computationally intensive, requiring substantial resources in terms of computational power and storage, which may pose challenges for smaller organizations.

10.2.4 Continuous Monitoring and Adaptation:

ML models need ongoing monitoring and adaptation to stay effective. Failure to do so may result in degraded performance as fraud patterns evolve over time.

10.2.5 Limited Explainability:

Some machine learning models, particularly complex ones like deep neural networks, may lack explainability, making it challenging to interpret the decision-making process, which could be a concern for regulatory compliance and user

While online payments fraud detection using machine learning offers substantial advantages in terms of accuracy, real-time detection, and adaptive learning, organizations must carefully consider the associated challenges, including complexity, data quality, resource requirements, and the need for continuous monitoring and adaptation. Balancing these factors is crucial to harness the benefits of ML while mitigating potential disadvantages.

11. CONCLUSION

Online payment fraud has been identified as one of the leading frauds in the past few decades. It was seen that feature selection techniques are very important and can be implemented to attain lower false positive rate. We implemented various machine learning algorithms like logistic regression, Random Forest in order to predict if a particular

transaction is fraudulent or not. A good fraud detection system should accurately be able to predict if a given transaction is fraudulent or not.

The goal of this project is to build a machine learning model that can accurately predict payment fraud by distinguishing between legitimate and fraudulent transactions based on their characteristics, such as transaction amount, type, and accounts involved. By using a dataset of both fraudulent and non-fraudulent financial transactions, the model can be trained to achieve high accuracy, which can be used by financial institutions to prevent financial losses and protect their customers' assets in real-time.

12. FUTURE SCOPE

The future scope of this project involves several avenues for enhancement and expansion:

- **Advanced Machine Learning Models:**
Explore the integration of advanced machine learning models, including deep learning techniques, to further improve the accuracy and adaptability of the fraud detection system.
- **Explainable AI:**
Incorporate explainable AI techniques to enhance the transparency of the decision-making process, providing insights into how the machine learning models arrive at their fraud detection predictions.
- **Blockchain Integration:**
Investigate the integration of blockchain technology to enhance the security and traceability of transactions, adding an additional layer of trust to the online payment ecosystem.
- **Behavioral Analysis:**
Introduce behavioral analysis features to the fraud detection system, leveraging user behavior patterns to identify anomalies and potentially fraudulent activities.
- **Cross-Industry Collaboration:**
Collaborate with other industries and organizations to share insights and data, fostering a collective approach to combating evolving fraud patterns and enhancing the robustness of the fraud detection system.

13. APPENDIX

13.1 Code Samples

13.1.1 Machine Learning Model Training

The following Python code snippet illustrates the training of a machine learning model using the RandomForestClassifier from the Scikit-learn library. This model is used for the fraud detection system.

Code:

```
In [1]:  
  
#Import necessary libraries  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
In [2]:  
  
#Importing the dataset.  
df = pd.read_csv("Online Payments Fraud Detection Dataset.csv")
```

```
In [3]:  
  
df
```

```
Out[3]:
```

	step	type	amount	nameOrig	oldbalanceOrig	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

	step	type	amount	nameOrig	oldbalanceOrig	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
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.head()

In [4]:

Out[4]:

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

df.shape

In [5]:

(6362620, 11)

Out[5]:

In [6]:

```
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   oldbalanceOrig        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
```

In [7]:

```
df.describe()
```

Out[7]:

	step	amount	oldbalanc eOrg	newbalance Orig	oldbalance Dest	newbalanc eDest	isFraud	isFlaggedF raud
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
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06	1.290820e-03	2.514687e-06
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06	3.590480e-02	1.585775e-03
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
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05	0.000000e+00	0.000000e+00
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06	0.000000e+00	0.000000e+00
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08	1.000000e+00	1.000000e+00

```
#Checking for Null Values.
```

```
df.isnull().any()
```

Out[9]:

```
step          False
type          False
amount        False
nameOrig      False
oldbalanceOrg False
newbalanceOrig False
nameDest      False
oldbalanceDest False
newbalanceDest False
isFraud       False
isFlaggedFraud False
dtype: bool
```

In [10]:

```
df.isnull().sum()
```

Out[10]:

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

Out[24]:

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

In [25]:

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

```
df.corr()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_24496\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
```

```
df.corr()
```

Out[36]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFlaggedFraud
step	1.000000	0.022373	-0.010058	-0.010299	0.027665	0.025888	0.003277
amount	0.022373	1.000000	-0.002762	-0.007861	0.294137	0.459304	0.012295
oldbalanceOrg	0.010058	0.002762	1.000000	0.998803	0.066243	0.042029	0.003835
newbalanceOrig	0.010299	0.007861	0.998803	1.000000	0.067812	0.041837	0.003776
oldbalanceDest	0.027665	0.294137	0.066243	0.067812	1.000000	0.976569	-0.000513
newbalanceDest	0.025888	0.459304	0.042029	0.041837	0.976569	1.000000	-0.000529
isFlaggedFraud	0.003277	0.012295	0.003835	0.003776	-0.000513	-0.000529	1.000000

In [37]:

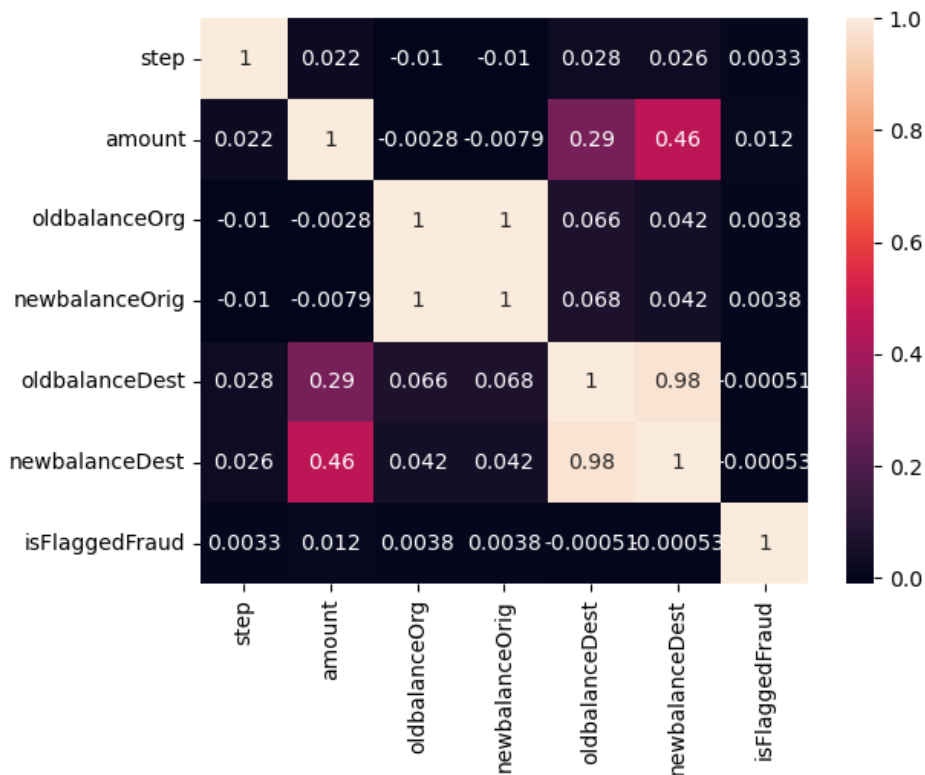
```
sns.heatmap(df.corr(),annot=True)
```

C:\Users\HP\AppData\Local\Temp\ipykernel_24496\4277794465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

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

Out[37]:

<Axes: >



Remove the Outliers

In [47]:

```
from scipy import stats
print(stats.mode(df['amount']))
print(np.mean(df['amount']))
C:\Users\HP\AppData\Local\Temp\ipykernel_24496\1242469671.py:2: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.1 1.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
    print(stats.mode(df['amount']))
ModeResult(mode=array([10000000.]), count=array([3207]))
179861.90354913071
```

In [48]:

```
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
```

Label encoding

In [55]:

```
from sklearn.preprocessing import LabelEncoder

la=LabelEncoder()
df['type']= la.fit_transform(df['type'])
C:\Users\HP\AppData\Local\Temp\ipykernel_24496\3714687182.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['type']= la.fit_transform(df['type'])
```

In [56]:

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

Out[56]:

```
1    2237484
3    2151495
0    1399284
4     532909
2     41432
Name: type, dtype: int64
```

Splitting Dependent and Independent variables

In [57]:

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

In [58]:

x

Out[58]:

	step	type	amount	oldbalance Org	newbalance Orig	oldbalance Dest	newbalance Dest	isFraud	log_amount
0	1	3	9.194174	170136.00	160296.36	0.00	0.00	is not Fraud	9.194174
1	1	3	7.530630	21249.00	19384.72	0.00	0.00	is not Fraud	7.530630
2	1	4	5.198497	181.00	0.00	0.00	0.00	is Fraud	5.198497

	step	type	amount	oldbalance Org	newbalance Orig	oldbalance Dest	newbalance Dest	isFraud	log_amount
3	1	1	5.198497	181.00	0.00	21182.00	0.00	is Fraud	5.198497
4	1	3	9.364617	41554.00	29885.86	0.00	0.00	is not Fraud	9.364617
...
6362615	743	1	12.735766	339682.13	0.00	0.00	339682.13	is Fraud	12.735766
6362616	743	4	15.657870	6311409.28	0.00	0.00	0.00	is Fraud	15.657870
6362617	743	1	15.657870	6311409.28	0.00	68488.84	6379898.11	is Fraud	15.657870
6362618	743	4	13.652995	850002.52	0.00	0.00	0.00	is Fraud	13.652995
6362619	743	1	13.652995	850002.52	0.00	6510099.11	7360101.63	is Fraud	13.652995

6362604 rows × 9 columns

In [59]:

```
df.isFraud.value_counts()
```

Out[59]:

```
is not Fraud    6354407
is Fraud        8197
Name: isFraud, dtype: int64
```

In [60]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [61]:

```
x["isFraud"]=le.fit_transform(x["isFraud"])
```

In [62]:

```
x.head
```

Out[62]:


```

<bound method NDFrame.head of
newbalanceOrig  oldbalanceDest  \
0              1      3  9.194174      170136.00      160296.36      0.
00
1              1      3  7.530630      21249.00      19384.72      0.
00
2              1      4  5.198497       181.00       0.00      0.
00
3              1      1  5.198497       181.00       0.00     21182.
00
4              1      3  9.364617     41554.00     29885.86      0.
00
...          ...    ...      ...      ...      ...      .
..
6362615      743      1  12.735766     339682.13      0.00      0.
00
6362616      743      4  15.657870     6311409.28      0.00      0.
00
6362617      743      1  15.657870     6311409.28      0.00     68488.
84
6362618      743      4  13.652995      850002.52      0.00      0.
00
6362619      743      1  13.652995      850002.52      0.00     6510099.
11

```

```

newbalanceDest  isFraud  log_amount
0              0.00      1    9.194174
1              0.00      1    7.530630
2              0.00      0    5.198497
3              0.00      0    5.198497
4              0.00      1    9.364617
...          ...    ...      ...
6362615      339682.13      0    12.735766
6362616          0.00      0    15.657870
6362617      6379898.11      0    15.657870
6362618          0.00      0    13.652995
6362619      7360101.63      0    13.652995

```

```
[6362604 rows x 9 columns]>
```

In [63]:

```
y
```

Out[63]:

```

0      0
1      0
2      0
3      0
4      0
..
6362615  0
6362616  0
6362617  0
6362618  0
6362619  0

```

```
Name: isFlaggedFraud, Length: 6362604, dtype: int64
```

Splitting data into train and test

In [64]:

```
from sklearn.model_selection import train_test_split
```

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

In [65]:

```
print(x_train.shape)  
print(x_test.shape)  
print(y_train.shape)  
print(y_test.shape)  
(5090083, 9)  
(1272521, 9)  
(5090083,)  
(1272521,)
```

Model Building

1. Decision Tree Classifier

In [66]:

```
from sklearn.tree import DecisionTreeClassifier  
dtc=DecisionTreeClassifier()
```

In [67]:

```
dtc.fit(x_train,y_train)
```

Out[67]:

```
DecisionTreeClassifier
```

```
DecisionTreeClassifier()
```

In [68]:

```
pred=dtc.predict(x_test)
```

In [69]:

```
pred
```

Out[69]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [70]:

```
y_test
```

Out[70]:

```
1302136    0  
3150766    0  
5346044    0  
2346207    0  
5001405    0  
..  
5165317    0
```

```
2551404    0
224722     0
495935     0
6018994    0
Name: isFlaggedFraud, Length: 1272521, dtype: int64
```

In [71]:

```
df
```

Out[71]:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	log_amount
0	1	3	9.194174	170136.00	160296.36	0.00	0.00	is not Fraud	0	9.194174
1	1	3	7.530630	21249.00	19384.72	0.00	0.00	is not Fraud	0	7.530630
2	1	4	5.198497	181.00	0.00	0.00	0.00	is Fraud	0	5.198497
3	1	1	5.198497	181.00	0.00	21182.00	0.00	is Fraud	0	5.198497
4	1	3	9.364617	41554.00	29885.86	0.00	0.00	is not Fraud	0	9.364617
...
6362615	743	1	12.735766	339682.13	0.00	0.00	339682.13	is Fraud	0	12.735766
6362616	743	4	15.657870	6311409.28	0.00	0.00	0.00	is Fraud	0	15.657870

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	log_amount
6362617	743	1	15.657870	6311409.28	0.00	68488.84	6379898.11	isFraud	0	15.657870
6362618	743	4	13.652995	850002.52	0.00	0.00	0.00	isFraud	0	13.652995
6362619	743	1	13.652995	850002.52	0.00	6510099.11	7360101.63	isFraud	0	13.652995

6362604 rows × 10 columns

In [72]:

```
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
```

In [73]:

```
ms.fit(x_train)
```

Out[73]:

```
MinMaxScaler
```

```
MinMaxScaler()
```

In [74]:

```
x_train_scaled = ms.transform(x_train)
x_train
```

Out[74]:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	log_amount
4320263	308	4	13.689327	0.00	0.00	895941.26	1777394.51	1	13.689327
816499	40	3	8.254165	0.00	0.00	0.00	0.00	1	8.254165
6052080	495	0	11.628403	29415.00	141655.94	464190.03	351949.09	1	11.628403

	step	type	amount	oldbalance Org	newbalance Orig	oldbalance Dest	newbalance Dest	isFraud	log_amount
5737175	399	3	9.435016	290586.15	278066.98	0.00	0.00	1	9.435016
1214038	133	0	10.550267	2700201.96	2738389.61	44711.77	6524.12	1	10.550267
...
2249467	187	1	11.641247	4231.00	0.00	397758.64	511450.47	1	11.641247
5157702	357	1	12.021156	167.00	0.00	321544.24	487778.99	1	12.021156
2215104	186	0	11.287786	10152925.86	10232766.38	276175.87	196335.36	1	11.287786
1484405	141	0	13.250940	1081784.28	1650388.77	2631796.49	2063191.99	1	13.250940
4500018	325	3	10.878047	423.00	0.00	0.00	0.00	1	10.878047

5090083 rows × 9 columns

In [75]:

```
x_test_scaled = ms.transform(x_test)
x_test
```

Out[75]:

	step	type	amount	oldbalance Org	newbalance Orig	oldbalance Dest	newbalance Dest	isFraud	log_amount
1302136	136	3	8.879953	0.00	0.00	0.00	0.00	1	8.879953
3150766	236	3	8.089430	5005.00	1745.17	0.00	0.00	1	8.089430

	step	type	amount	oldbalance Org	newbalance Orig	oldbalance Dest	newbalance Dest	isFraud	log_amount
5346044	375	4	12.908264	0.00	0.00	899613.40	1303247.48	1	12.908264
2346207	189	3	8.370286	780857.00	776540.13	0.00	0.00	1	8.370286
5001405	353	0	12.459026	5530185.33	5787749.91	7099482.44	6841917.87	1	12.459026
...
5165317	358	0	11.319888	18621.00	101066.12	407712.72	325267.60	1	11.319888
2551404	206	1	12.217523	202509.00	205.83	2096979.23	2299282.40	1	12.217523
224722	14	4	14.644671	0.00	0.00	3729395.93	6020789.00	1	14.644671
495935	20	0	9.559148	11447430.20	11461603.97	3047711.43	3033537.66	1	9.559148
6018994	455	3	7.306766	0.00	0.00	0.00	0.00	1	7.306766

1272521 rows × 9 columns

In [76]:

```
# Fit the scaler on your training data
ms.fit(x_train)

# Transform your input data
input_data = [[136,3,8.879953,0.00,0.00,0.00,0.00,0,8.879953]]
transformed_data = ms.transform(input_data)

# Now you can use the transformed data for prediction
dtc.predict(transformed_data)
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but MinMaxScaler was fitted with feature
names
  warnings.warn(
```

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but DecisionTreeClassifier was fitted wi
th feature names
  warnings.warn(
```

Out[76]:

```
array([0], dtype=int64)
```

Evaluation of classification model

In [77]:

```
#Accuracy score
from sklearn.metrics import
accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_cur
ve
```

In [78]:

```
accuracy_score(y_test, pred)
```

Out[78]:

```
1.0
```

In [79]:

```
confusion_matrix(y_test, pred)
```

Out[79]:

```
array([[1272519,      0],
       [      0,      2]], dtype=int64)
```

In [80]:

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

Out[80]:

	col_0	0	1
isFlaggedFraud			
0	1272519	0	
1		0	2

In [81]:

```
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1272519
1	1.00	1.00	1.00	2
accuracy			1.00	1272521
macro avg	1.00	1.00	1.00	1272521
weighted avg	1.00	1.00	1.00	1272521

Roc-AUC curve

In [82]:

```
probability=dtc.predict_proba(x_test)[: ,1]
```

In [83]:

```
probability
```

Out[83]:

```
array([0., 0., 0., ..., 0., 0., 0.])
```

In [84]:

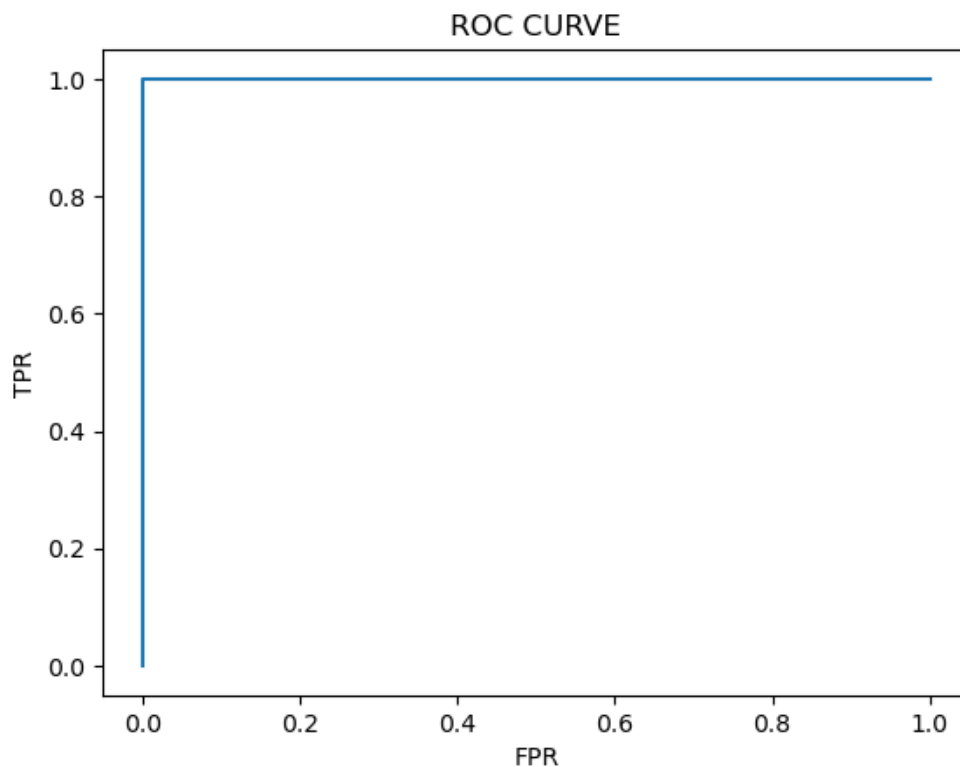
```
print(x.columns)
Index(['step', 'type', 'amount', 'oldbalanceOrig', 'newbalanceOrig',
      'oldbalanceDest', 'newbalanceDest', 'isFraud', 'log_amount'],
      dtype='object')
```

In [85]:

```
# roc_curve
fpr,tpr,threshholds = roc_curve(y_test,probability)
```

In [86]:

```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```

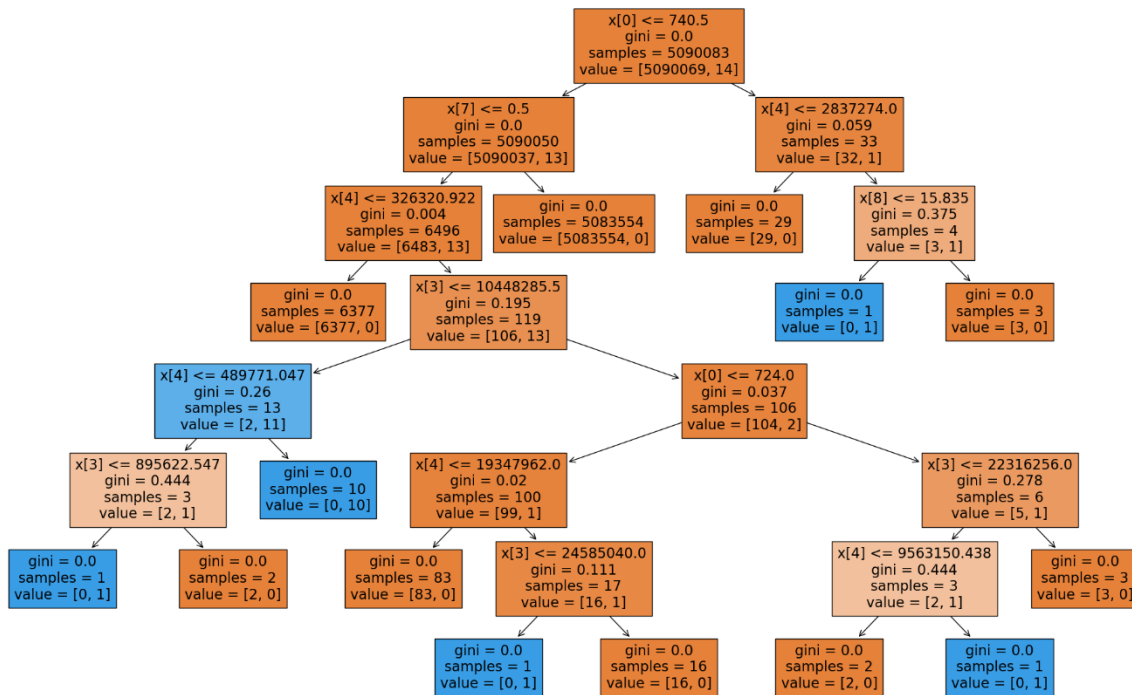


In [87]:

```
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```

Out[87]:


```
[Text(0.5714285714285714, 0.9375, 'x[0] <= 740.5\ngini = 0.0\nsamples = 5090083\nvalue = [5090069, 14]'),
Text(0.42857142857142855, 0.8125, 'x[7] <= 0.5\ngini = 0.0\nsamples = 5090050\nvalue = [5090037, 13]'),
Text(0.35714285714285715, 0.6875, 'x[4] <= 326320.922\ngini = 0.004\nsamples = 6496\nvalue = [6483, 13]'),
Text(0.2857142857142857, 0.5625, 'gini = 0.0\nsamples = 6377\nvalue = [6377, 0]'),
Text(0.42857142857142855, 0.5625, 'x[3] <= 10448285.5\ngini = 0.195\nsamples = 119\nvalue = [106, 13]'),
Text(0.21428571428571427, 0.4375, 'x[4] <= 489771.047\ngini = 0.26\nsamples = 13\nvalue = [2, 11]'),
Text(0.14285714285714285, 0.3125, 'x[3] <= 895622.547\ngini = 0.444\nsamples = 3\nvalue = [2, 1]'),
Text(0.07142857142857142, 0.1875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(0.21428571428571427, 0.1875, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
Text(0.2857142857142857, 0.3125, 'gini = 0.0\nsamples = 10\nvalue = [0, 10]'),
Text(0.6428571428571429, 0.4375, 'x[0] <= 724.0\ngini = 0.037\nsamples = 106\nvalue = [104, 2]'),
Text(0.42857142857142855, 0.3125, 'x[4] <= 19347962.0\ngini = 0.02\nsamples = 100\nvalue = [99, 1]'),
Text(0.35714285714285715, 0.1875, 'gini = 0.0\nsamples = 83\nvalue = [83, 0]'),
Text(0.5, 0.1875, 'x[3] <= 24585040.0\ngini = 0.111\nsamples = 17\nvalue = [16, 1]'),
Text(0.42857142857142855, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(0.5714285714285714, 0.0625, 'gini = 0.0\nsamples = 16\nvalue = [16, 0]'),
Text(0.8571428571428571, 0.3125, 'x[3] <= 22316256.0\ngini = 0.278\nsamples = 6\nvalue = [5, 1]'),
Text(0.7857142857142857, 0.1875, 'x[4] <= 9563150.438\ngini = 0.444\nsamples = 3\nvalue = [2, 1]'),
Text(0.7142857142857143, 0.0625, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
Text(0.8571428571428571, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(0.9285714285714286, 0.1875, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
Text(0.5, 0.6875, 'gini = 0.0\nsamples = 5083554\nvalue = [5083554, 0]'),
Text(0.7142857142857143, 0.8125, 'x[4] <= 2837274.0\ngini = 0.059\nsamples = 33\nvalue = [32, 1]'),
Text(0.6428571428571429, 0.6875, 'gini = 0.0\nsamples = 29\nvalue = [29, 0]'),
Text(0.7857142857142857, 0.6875, 'x[8] <= 15.835\ngini = 0.375\nsamples = 4\nvalue = [3, 1]'),
Text(0.7142857142857143, 0.5625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(0.8571428571428571, 0.5625, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]')] ]
```



In [88]:

```

from sklearn.model_selection import GridSearchCV
parameter={
    'criterion':['gini','entropy'],
    'splitter':['best','random'],
    'max_depth':[1,2,3,4,5],
    'max_features':['auto', 'sqrt', 'log2']
}

```

In [89]:

```

grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accuracy")

```

In [90]:

```

grid_search.fit(x_train,y_train)
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
100 fits failed out of a total of 300.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

```

Below are more details about the failures:

```

-----
-----
100 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\HP\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py", line 1145, in wrapper

```

```

    estimator._validate_params()
File "C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py", line 638,
in _validate_params
    validate_parameter_constraints(
File "C:\Users\HP\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py", line 96, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of DecisionTreeClassifier must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'sqrt', 'log2'} or None. Got 'auto' instead.

warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:979: UserWarning: One or more of the test scores are non-finite: [
nan      nan 0.99999725 0.99999725 0.99999725 0.99999725
nan      nan 0.99999725 0.99999725 0.99999725 0.99999725
nan      nan 0.99999725 0.99999725 0.99999725 0.99999725
nan      nan 0.99999705 0.99999725 0.99999725 0.99999764
nan      nan 0.99999823 0.99999745 0.99999705 0.99999725
nan      nan 0.99999725 0.99999725 0.99999725 0.99999725
nan      nan 0.99999725 0.99999725 0.99999725 0.99999725
nan      nan 0.99999725 0.99999764 0.99999745 0.99999725
nan      nan 0.99999764 0.99999725 0.99999764 0.99999725
nan      nan 0.99999823 0.99999725 0.99999745 0.99999745]
warnings.warn(

```

Out[90]:

GridSearchCV

estimator: DecisionTreeClassifier

DecisionTreeClassifier

In [91]:

```
grid_search.best_params_
```

Out[91]:

```
{'criterion': 'gini',
 'max_depth': 5,
 'max_features': 'sqrt',
 'splitter': 'best'}
```

In [92]:

```
dtc_cv=DecisionTreeClassifier(criterion='entropy',
                              max_depth=3,
                              max_features='sqrt',
                              splitter='best')
dtc_cv.fit(x_train,y_train)
```

Out[92]:

DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=3, max_features='sqrt')

In [93]:

```
pred=dtc_cv.predict(x_test)
```

In [94]:

```
print(classification_report(y_test,pred))
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1471: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1471: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

         0           1.00      1.00      1.00     1272519
         1           0.00      0.00      0.00          2

   accuracy                   1.00     1272521
  macro avg           0.50      0.50      0.50     1272521
weighted avg           1.00      1.00      1.00     1272521

C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1471: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

In [150]:

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

Out[150]:

```
DecisionTreeClassifier
```

```
DecisionTreeClassifier()
```

In [151]:

```
y_test_predict1=dtc.predict(x_test)
test_accuracy=accuracy_score (y_test,y_test_predict2)
test_accuracy
```

Out[151]:

```
0.9999992141583518
```

In [152]:

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

Out[152]:

```
1.0
```

In [153]:

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

Out[153]:

```

col_0      0  1

isFlaggedFraud

0  1272517  0

1           1  3

print(classification_report (y_test,y_test_predict1))
           precision    recall  f1-score   support

0           1.00        1.00        1.00   1272517
1           1.00        0.75        0.86         4

   accuracy          1.00   1272521
  macro avg           1.00   1272521
weighted avg           1.00   1272521

```

In [154]:

2. Random Forest Classifier

In [95]:

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)

```

Out[95]:

```
RandomForestClassifier
```

```
RandomForestClassifier()
```

In [155]:

```

y_test_predict2=rfc.predict(x_test)
test_accuracy=accuracy_score (y_test,y_test_predict2)
test_accuracy

```

Out[155]:

```
1.0
```

In [156]:

```

y_train_predict2=rfc.predict(x_train)
train_accuracy=accuracy_score (y_train,y_train_predict2)
train_accuracy

```

Out[156]:

```
1.0
```

In [157]:

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

Out[157]:

```
col_0      0  1
isFlaggedFraud
0 1272517  0
1         0  4
```

In [158]:

```
print(classification_report(y_test,y_test_predict2))
           precision    recall  f1-score   support

    0               1.00      1.00      1.00  1272517
    1               1.00      1.00      1.00         4

 accuracy               1.00  1272521
 macro avg              1.00      1.00      1.00  1272521
weighted avg              1.00      1.00      1.00  1272521
```

3. ExtraTrees Classifier

In [100]:

```
from sklearn.ensemble import ExtraTreesClassifier
etc=ExtraTreesClassifier()
etc.fit(x_train,y_train)
```

Out[100]:

```
ExtraTreesClassifier
```

```
ExtraTreesClassifier()
```

In [101]:

```
y_test_predict3=etc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict3)
test_accuracy
```

Out[101]:

```
1.0
```

In [102]:

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

Out[102]:

```
1.0
```

In [103]:

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

Out[103]:

```
col_0      0  1

isFlaggedFraud

0  1272519  0

1           0  2
```

In [104]:

```
print(classification_report(y_test,y_test_predict3))
           precision    recall  f1-score   support

0           1.00      1.00      1.00   1272519
1           1.00      1.00      1.00         2

 accuracy          1.00   1272521
 macro avg          1.00   1272521
weighted avg          1.00   1272521
```

4. SupportVectorMachine Classifier

In [105]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc= SVC()
svc.fit(x_train,y_train)
```

Out[105]:

```
SVC
```

```
SVC()
```

In [106]:

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

Out[106]:

```
0.9999984283167036
```

In [107]:

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

Out[107]:

```
0.999997249553691
```

In [108]:

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

Out[108]:

```
col_0      0

isFlaggedFraud

0  1272519

1      2
```

In [109]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report (y_test,y_test_predict4))
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1471: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1471: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

         0         1.00      1.00      1.00    1272519
         1         0.00      0.00      0.00         2

 accuracy          1.00    1272521
 macro avg         0.50      0.50      0.50    1272521
weighted avg         1.00      1.00      1.00    1272521

C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:
1471: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

In [110]:

```
df.columns
```

Out[110]:

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

In [125]:

```
from sklearn.preprocessing import LabelEncoder

la=LabelEncoder()
y_train=la.fit_transform(y_train)
```



```
y_test=la.transform(y_test)
```

In [126]:

```
y_test
```

In [127]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Out[127]:

```
y_train
```

In [128]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Out[128]:

5. xgboost Classifier

```
from sklearn.model_selection import train_test_split
import xgboost as xgb
xgb1 = xgb.XGBClassifier()
xgb1.fit(x_train, y_train)
```

In [113]:

Out[113]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=None, max_leaves=None,
               min_child_weight=None, missing=nan, monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, random_state=None, ...)
```

In [114]:

```
y_test_predict5=xgb1.predict(x_test)
test_accuracy=accuracy_score (y_test,y_test_predict5)
test_accuracy
```

Out[114]:

```
0.9999992141583518
```

In [115]:

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

Out[115]:

```
0.9999998035395493
```

In [116]:

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

Out[116]:

	col_0	0	1
isFlaggedFraud			
0	1272519	0	
1		1	1

In [117]:

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

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1272519
1	1.00	0.50	0.67	2
accuracy			1.00	1272521
macro avg	1.00	0.75	0.83	1272521
weighted avg	1.00	1.00	1.00	1272521

6. Compare the model

In [165]:

```
from sklearn.metrics import accuracy_score

def compareModel():
    print("train accuracy for dtc", accuracy_score(y_train_predict1,
y_train))
    print("test accuracy for dtc", accuracy_score(y_test_predict1, y_test))
    print("train accuracy for rfc", accuracy_score(y_train_predict2,
y_train))
    print("test accuracy for rfc", 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 svc", accuracy_score(y_test_predict4, y_test))
    print("train accuracy for xgb1", accuracy_score(y_train_predict5,
y_train))
    print("test accuracy for xgb1", accuracy_score(y_test_predict5,
y_test))
```

In [166]:

```
print(len(y_train), len(pred))
```

5090083 1272521

In [167]:

```
compareModel()  
train accuracy for dtc 1.0  
test accuracy for dtc 0.9999992141583518  
train accuracy for rfc 1.0  
test accuracy for rfc 1.0  
train accuracy for etc 0.9999948920282832  
test accuracy for etc 0.9999952849501108  
train accuracy for svc 0.9999976424745922  
test accuracy for svc 0.9999968566334072  
train accuracy for xgb1 0.9999950884887339  
test accuracy for xgb1 0.999996070791759
```

7. Evaluating performance of the model and saving the model

In [121]:

```
from sklearn.svm import SVC  
from sklearn.metrics import accuracy_score  
svc= SVC ()  
svc.fit(x_train,y_train)
```

Out[121]:

SVC

SVC()

In [122]:

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

Out[122]:

0.9999984283167036

In [123]:

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

Out[123]:

0.999997249553691

In [124]:

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

***** *Thank You* *****