

# Project Report

## Online Payments Fraud Detection Using ML

<b>Team ID</b>	<b>Team-592613</b>
<b>Team Members</b>	<b>Srajal Agarwal (Leader)</b> <b>Deepesh</b> <b>Dhruv</b> <b>Paras Garg</b>

# INTRODUCTION:

## 1.1 Project Overview:

Businesses and consumers alike are very concerned about online payment fraud. To guard against fraudulent transactions, it is essential to have strong fraud detection systems as the use of online payment methods grows. The goal of this project is to develop a system that effectively detects and prevents online payment fraud using machine learning (ML) techniques.

## 1.2 Project Purpose:

**1. Data Collection:** Compile an extensive dataset of electronic payments. Both authentic and fraudulent transactions, together with a variety of information such as transaction amount, location, user behaviour, and more, should be included in the dataset.

**2. Data Preprocessing:** Sort and prepare the gathered information, taking care of anomalies and missing values as well as encoding categorical variables. This action is essential for getting the data ready for training an ML model.

**3. Feature Engineering:** Produce pertinent features that enable the machine learning model to distinguish between authentic and fraudulent transactions. This could include user behaviour profiling, time-based features, and other things.

**4. Model Selection:** Select machine learning techniques (e.g., logistic regression, decision trees, random forests, support vector machines, or neural networks) that are suitable for fraud detection. To increase accuracy, ensemble approaches may also be taken into consideration.

**5. Model Training:** Using the preprocessed dataset, train the chosen machine learning models. To maximise the performance of the model, cross-validation methods and hyperparameter adjustment should be used.

**6. Real-time Monitoring:** Put in place a system that continuously looks for signs of fraud in incoming transactions. When there are questionable activity, this system ought to notify users right away.

**7. Model Assessment:** Evaluate the model's performance measured by a range of evaluation criteria, including accuracy, F1-score, precision, and recall. Keep an eye on the model and adjust it frequently to accommodate evolving fraud trends.

**8. Interpretability:** Provide techniques for interpreting and clarifying the model's conclusions. This is crucial for fostering trust and illuminating the process by which fraud detection judgements are determined.

**9. Integration:** To enable real-time fraud detection and prevention, integrate the fraud detection model with the online payment system or payment gateway.

**10. User-Friendly Dashboard:** To help administrators see and track the system's performance, make a dashboard that is easy to use. The dashboard ought to offer insights about trends and patterns of fraud that have been found.

**11. Scalability and Performance:** To prevent upsetting legitimate users, make sure the system can manage a high transaction volume and keeps a low false-positive rate.

**12. Record-keeping:** Provide comprehensive project documentation that covers the model architectures, data sources, parameters, and any other tools or libraries utilised.

**13. Security:** Put in place safeguards to keep the model and the data it utilises safe from manipulation or hacking attempts.

## **2. LITERATURE SURVEY :**

### **2.1 Existing problem :**

**Here are a few noteworthy problems:**

**1. Imbalanced Datasets:** - Fraudulent transactions are frequently few in comparison to valid ones. The model's capacity to recognise fraudulent patterns effectively may be impacted by this mismatch.

- **Solution:** Unbalanced datasets can be addressed by employing strategies like under- or oversampling, as well as sophisticated algorithms like Synthetic Minority Over-sampling Technique (SMOTE).

**2. Dynamic Nature of Fraud:** - **Issue:** Since fraudulent strategies change over time, it is difficult for static models to catch up with emerging trends.

- **Solution:** It is imperative that the model be updated and monitored continuously. It is crucial to have adaptive models that can modify their detection tactics in response to fresh data.

**3. Feature Engineering Complexity:**

- **Issue:** It might be difficult and require domain expertise to extract significant characteristics from transaction data.

Automated feature engineering is the solution. The model's capacity to identify fraud trends may be enhanced by the application of domain-specific features.

**4. User Experience and False Positives:**

- **Issue:** Excessively stringent fraud detection methods may result in false positives, causing inconvenience to authorised users and eroding their confidence in the system.

The key is to strike a balance between recall and precision. In order to reduce false positives and preserve high accuracy in identifying real fraud, models must be adjusted.

**5. Intellectual Assaults:**

- **Issue:** By creating transactions that are especially meant to avoid detection, adversarial actors may deliberately try to influence the model.

- **Solution:** Adversarial assaults can be lessened by putting strong security measures in place, updating models often, and using adversarial training methods.

## **6. Interpretability and Explainability:**

- **Issue:** A lot of machine learning models are opaque, which makes it difficult to comprehend how they make choices.

- **Solution:** Creating interpretable models or offering explanations for model decisions after the fact can help users and stakeholders feel more trusted and understand each other better.

## **7. Data Privacy Issues:**

- **Issue:** Managing private transaction data gives rise to privacy issues.

- **Solution:** To safeguard user information and enable efficient fraud detection, implement privacy-preserving strategies like secure multiparty computing or differential privacy.

## **8. Scalability:**

- **Issue:** The scalability of fraud detection algorithms becomes a challenge as the amount of online transactions increases.

- **Solution:** Distributed computing methods and scalable machine learning architectures can help effectively manage high transaction volumes.

## **9. Fraud via Cross-Channel:**

- **Issue:** Since fraudsters can use weaknesses in a variety of channels, detecting fraud needs to be approached from all angles.

- **Solution:** Combining fraud detection tools with different platforms and channels to build a thorough understanding of user conduct and transaction patterns.

## **10. Adherence to Regulations:**

- **Issue:** It can be difficult to successfully use fraud detection techniques while adhering to data privacy requirements.

- **Solution:** Applying privacy-by-design principles and making sure the fraud detection system conforms with applicable data protection laws and regulations.

A multidisciplinary strategy including domain-specific knowledge of online payment systems, cybersecurity, machine learning, and data privacy is needed to address these issues. Researchers and

industry professionals are always looking for novel ways to improve machine learning's efficacy and dependability in the identification of online payment fraud.

## **2.2 References**

1. <https://ieeexplore.ieee.org/document/10142404>
2. <https://iopscience.iop.org/article/10.1088/1742-6596/2023/1/012054>(Online Transaction Fraud Detection System Based on Machine Learning Bocheng Liu<sup>1</sup>, Xiang Chen<sup>1</sup> and Kaizhi Yu<sup>1</sup> Published under licence by IOP Publishing Ltd)
3. [https://www.researchgate.net/publication/354937786\\_Online\\_Transaction\\_Fraud\\_Detection\\_System\\_Based\\_on\\_Machine\\_Learning](https://www.researchgate.net/publication/354937786_Online_Transaction_Fraud_Detection_System_Based_on_Machine_Learning) , September 2021
4. <https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset> , Kaggle

## **2.3 Problem Statement Definition:**

The rapid growth of online transactions has brought about an escalating threat of fraudulent activities in the realm of online payments. Fraudsters employ sophisticated techniques to exploit vulnerabilities in payment systems, leading to financial losses for both businesses and consumers. Traditional rule based fraud detection systems are often insufficient to adapt to the evolving nature of fraudulent tactics. Therefore, there is a pressing need to develop a robust and adaptive Online Payment Fraud Detection system using Machine Learning (ML) to enhance the security and reliability of online Transactions.

## **3. IDEATION & PROPOSED SOLUTION**

### **3.1 Empathy Map Canvas**

An empathy map is a useful tool that may be used by teams as they go on the journey to obtain deeper insights into their users. In the context of our project, "Online Payments Fraud Detection

Using ML" this tool is more than simply an asset, it is a vital success factor. The success of our endeavour is dependent on our ability to grasp the genuine challenges that users confront and to obtain a thorough knowledge of the persons navigating the complex environment of online payment systems.


The process of creating an empathy map helps all project contributors to adopt a viewpoint that is consistent with the user's experiences. It pushes us to see things through the user's eyes, effectively stepping into their shoes and understanding their goals, as well as the barriers they face on their trip through the world of online payments. This allows us to have a deeper insight of their requirements, anxieties, motivations, and pain areas.



### 3.2 Ideation & Brainstorming


Brainstorming generated a collaborative culture in our "Online Payments Fraud Detection using ML" project, allowing for the participation of varied ideas. Prioritising number above quality at first promoted unconventional thinking, resulting in a wealth of unique ideas. The team's collaborative attitude aided in the refinement of thoughts into real solutions, while this problem-solving technique highlighted system flaws. The method also helped with risk management and proactive planning. Overall, brainstorming is important in designing a complete and successful fraud detection system because it engages and motivates all team members to actively participate in the creative problem-solving process.


Template





## Brainstorm & idea prioritization

Brainstorming fostered cooperation and diverse contribution in our "Online Payments Fraud Detection Using ML" project, prioritising idea quantity and new thinking. This method optimised solutions, uncovered system flaws, and aided in risk reduction. It was critical in developing an effective fraud detection system by encouraging team participation in innovative problem-solving.

 10 minutes to prepare


 1 hour to collaborate


 2-8 people recommended




### 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

 **Team gathering**

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

 **Set the goal**

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

 **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

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.


 5 minutes


The project "Online Payments Fraud Detection Using ML" aims to improve the security of digital transactions. We intend to develop a model capable of reliably classifying transactions as fraudulent or authentic using Machine Learning and a Kaggle dataset. We strive to uncover detailed trends leading to online payment fraud by scrutinising transaction information such as type, quantity, and user balances. We want to go beyond past research and forecast future hazards. Our ultimate objective is to provide individuals and companies with clear, user-friendly tools for securely navigating the digital payment arena.





### 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 gear icon to quickly learn to start drawing!

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 additional tags to sticky notes to make it easier to find, categorize, organize, and categorize important ideas as they relate to your story.

## SRAJAL AGARWAL

### User Feedback Loop

Collect feedback from users who report isolated fraud and use this data to improve the model.

### Regulatory Compliance Monitoring

Ensure that the system adheres to relevant regulations and can provide necessary reports if needed.

### Adaptive Models

Develop models that can adapt to new fraud patterns and update themselves regularly.

### Threshold Adjustments

Dynamically adjust fraud detection thresholds based on current risk factors and fraud rates.

### Collaborative Filtering

Use collaborative filtering techniques to detect anomalies in user behavior by comparing them to similar user profiles.

### Integration with External Data Sources

Incorporate external data, such as bank data or industry-specific fraud databases, to enhance fraud detection.

## DEEPESH

### Explainable AI (XAI)

Utilize models that provide explanations for their decisions, allowing for better transparency and fraud analysis.

### Feature Importance Analysis

Identify the most important features in fraud detection and prioritize them in the model.

### Biometric Authentication

Incorporate biometric data like fingerprint or facial recognition for user authentication.

### Graph Analysis

Represent transactions as a graph network and detect fraud by analyzing the relationships between users and transactions.

### Natural Language Processing (NLP)

Analyze text-based data, such as transaction descriptions or user communication, to identify fraudulent language or phishing attempts.

### Behavioral Analytics

Examine user behavior in the context of online payments using powerful machine learning methods. To define baseline behaviors, analyze transaction frequency, geographical location, and normal spending habits of users. Anomalies and variations from these established patterns should be identified as possible signs of online payment fraud, allowing for early and accurate fraud detection.

### Geospatial and Historical Analysis

This method involves inspecting transaction locations and IP addresses for unusual activity. We discover recurring fraud tendencies using previous data, guaranteeing that our model is constantly refined for more efficient online payment fraud detection.

### Biometric Authentication

To strengthen user verification in online financial transactions, our solution incorporates biometric data such as fingerprint or face recognition. We considerably minimize the danger of fraudulent activities by adding this extra layer of protection, offering a strong defence against unauthorized access and fraudulent transactions.

### Adaptive Models

This method entails the development of adaptable models that may evolve and update themselves in response to developing fraud tendencies. These models assure continued accuracy and efficacy in detecting online payment fraud by learning from fresh data and modifying their detection algorithms on a regular basis, even in the face of increasing threats.

## DHRUV

### Behavioral Analytics

Analyze user behavior patterns, such as transaction frequency, location, and spending habits. Identify deviations from established behavior as potential fraud.

### Feature Engineering

Derive new features based on IP geolocation, device information, and user history. Develop features that capture time-based patterns, like the time of day or week.

### Deep Learning for Sequence Data

Use recurrent neural networks (RNNs) or transformers to model sequential transaction data and detect fraud patterns over time.

### Ensemble Learning

We can use machine learning methods that combine the concepts of fine-tuning Random Forest classifiers, developing a user-friendly Flask interface, implementing threshold tuning, integrating with cloud services for scalability and monitoring model performance for online payments fraud detection.

## PARAS GARG

### Geospatial Analysis

Consider geographical information to detect unusual transaction locations or IP addresses.

### Historical Data Analysis

Use historical fraud data to identify recurring patterns and adjust the model to detect similar fraud in the future.

### Time Series Analysis

Examine transaction time series data for recurrent patterns and anomalies that might identify fraud tendencies over time.

### Real-Time Analysis

Create systems capable of detecting and responding to frequent transactions in real time, preventing or reporting suspicious behaviour as it occurs.

### Ensemble Models

Combine the outputs of multiple machine learning models to improve fraud detection accuracy.

### Transaction Monitoring

Build a monitoring system for transaction data for unusual patterns or anomalies. Use unsupervised learning techniques like clustering to detect outliers.

### Ensemble Learning

This employs a comprehensive machine learning technique that combines several components to identify online payment fraud. To improve the accuracy and security of online payments, it features fine-tuned Random Forest classifiers, a user-friendly Flask interface, adaptive threshold adjustment, cloud integration for scalability, and proactive model performance monitoring.

### Time Series Analysis

Exploring transaction time series data reveals repeating patterns and anomalies over time, providing insights into emerging fraud trends. This strategy improves our capacity to recognise and respond to new fraud threats in the online payment space.

4

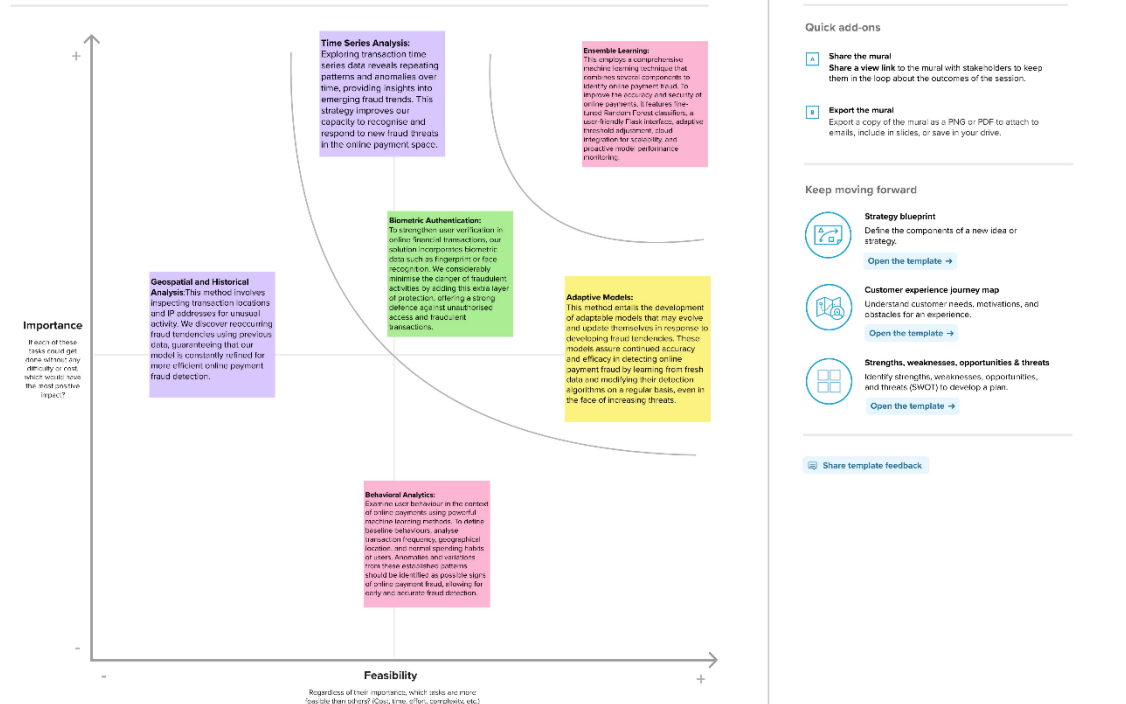
## Prioritize

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 cursor to place a whole sticky note straight on the grid. The facilitator can confirm the spot by using the laser pointer, holding the W key on the keyboard.



## 4. REQUIREMENT ANALYSIS

### 4.1 Functional requirement

#### 1. User Authentication:

- Description: The system should have a robust user authentication mechanism to ensure that only authorized users can access and administer the fraud detection system.

#### 2. Data Collection and Storage:

- Description: The system must be capable of collecting and storing a diverse dataset of online payment transactions, including both legitimate and fraudulent instances.

#### 3. Data Preprocessing:

- Description: Implement data preprocessing functionalities to clean and prepare the collected data. This includes handling missing values, outliers, and encoding categorical variables for effective model training.

#### 4. Feature Engineering:

- Description: Provide mechanisms for feature engineering to create relevant and meaningful features from transaction data. This may involve time-based features, user behavior profiling, and other relevant aspects.

#### 5. Model Selection:

- Description: Include a variety of machine learning algorithms for fraud detection, such as logistic regression, decision trees, random forests, support vector machines, or neural networks. Consider ensemble methods for improved accuracy.

#### 6. Model Training:

-Description: Facilitate the training of machine learning models on the preprocessed dataset. Include hyperparameter tuning and cross-validation techniques to optimize the model's performance.

### **7. Real-Time Monitoring:**

- Description: Implement real-time monitoring capabilities to continuously evaluate incoming transactions for potential fraud. Provide immediate alerts for suspicious activities.

### **8. Model Evaluation:**

- Description: Include functionalities to assess the model's performance using various evaluation metrics, such as precision, recall, F1-score, and accuracy. Allow for continuous monitoring and updating of the model to adapt to new fraud patterns.

### **9. Interpretability:**

- Description: Develop methods to interpret and explain the model's decisions. Provide insights into why a particular transaction was flagged as fraudulent, enhancing transparency and trust.

### **10. Integration with Payment System:**

- Description: Integrate the fraud detection model into the online payment system or payment gateway to enable real-time fraud detection and prevention during transactions.

### **11. User-Friendly Dashboard:**

- Description: Create a user-friendly dashboard for administrators to visualize and monitor the system's performance. The dashboard should display detected fraud patterns, trends, and other relevant insights.

### **12. Scalability:**

- Description: Ensure that the system can handle a high volume of transactions by implementing scalable machine learning architectures and distributed computing techniques.

### **13. Alerting Mechanism:**

- Description: Implement an alerting mechanism to notify administrators and relevant stakeholders in real-time when potentially fraudulent activities are detected.

#### **14. Continuous Learning:**

- Description: Enable the system to continuously learn from new data and adapt to changes in fraud patterns over time. This may involve periodic model retraining and updates.

#### **15. Security Measures:**

- Description: Implement security measures to protect the model and the data it uses, preventing unauthorized access, tampering, or other security threats.

#### **16. Documentation:**

- Description: Provide comprehensive documentation covering data sources, model architectures, parameters, and any additional tools or libraries used. This documentation is essential for future reference and system maintenance.

#### **17. Regulatory Compliance:**

- Description: Ensure that the system complies with relevant data protection and privacy regulations, taking measures to handle sensitive financial data securely.

By incorporating these functional requirements, the Online Payment Fraud Detection System can effectively detect and prevent fraudulent activities, contributing to a more secure online payment environment.

## **4.2 Non-Functional requirements**

Certainly, non-functional requirements are crucial for shaping the overall performance, reliability, and

usability of the "Online Payment Fraud Detection in ML" system. Here are key non-functional requirements for such a system:

### **1. Performance:**

- Response Time: The system should provide real-time fraud detection with a response time of less than one second to ensure timely prevention of fraudulent transactions.

- Scalability: The system must scale horizontally to handle an increasing volume of transactions

without a significant degradation in performance.

## **2. Reliability:**

- Availability: The system should have a high availability, ensuring that it is operational and accessible to users at least 99.9% of the time.
- Fault Tolerance: In the event of a component failure, the system should gracefully handle the situation without compromising overall functionality.

## **3. Security:**

- Data Encryption: All sensitive data, including transaction details and user information, must be encrypted during transmission and storage to prevent unauthorized access.
- Access Control: Implement strict access controls to ensure that only authorized personnel can access and modify the system's configurations and data.

## **4. Usability:**

- User Interface (UI) Design: The system's user interface should be intuitive, user-friendly, and provide relevant insights for administrators monitoring fraud detection.
- Documentation: Comprehensive documentation should be available for system administrators and end-users, detailing system functionalities, configurations, and troubleshooting procedures.

## **5. Scalability:**

- Horizontal Scaling: The system should be designed to horizontally scale by adding more computational resources to handle an increasing number of transactions without compromising performance.

## **6. Interpretability and Explainability:**

- Model Interpretability: The machine learning models used for fraud detection should be interpretable, providing insights into how and why a decision was made to enhance trust and understanding.

## **7. Privacy:**

- Data Privacy Compliance: Ensure that the system adheres to relevant data protection laws and regulations, with measures in place to protect user privacy during data processing and storage.

#### **8. Adaptability:**

- Model Update Frequency: The system should support regular updates of machine learning models to adapt to evolving fraud patterns, with a defined update frequency based on emerging threats.

#### **9. Maintainability:**

- Code Maintainability: The system's codebase should be well-organized, well-documented, and adhere to coding standards to facilitate ease of maintenance and future enhancements.

#### **10. Auditability:**

- Transaction Logs: Maintain detailed transaction logs, allowing for auditing and traceability of system activities. Logs should capture relevant information for forensic analysis.

#### **11. Integration:**

- Third-Party Integration: The system should support integration with third-party services, such as payment gateways and analytics tools, to enhance functionality and reporting capabilities.

#### **12. Regulatory Compliance:**

- Compliance Reporting: The system should generate reports demonstrating compliance with relevant industry standards and regulatory requirements.

#### **13. Training and Support:**

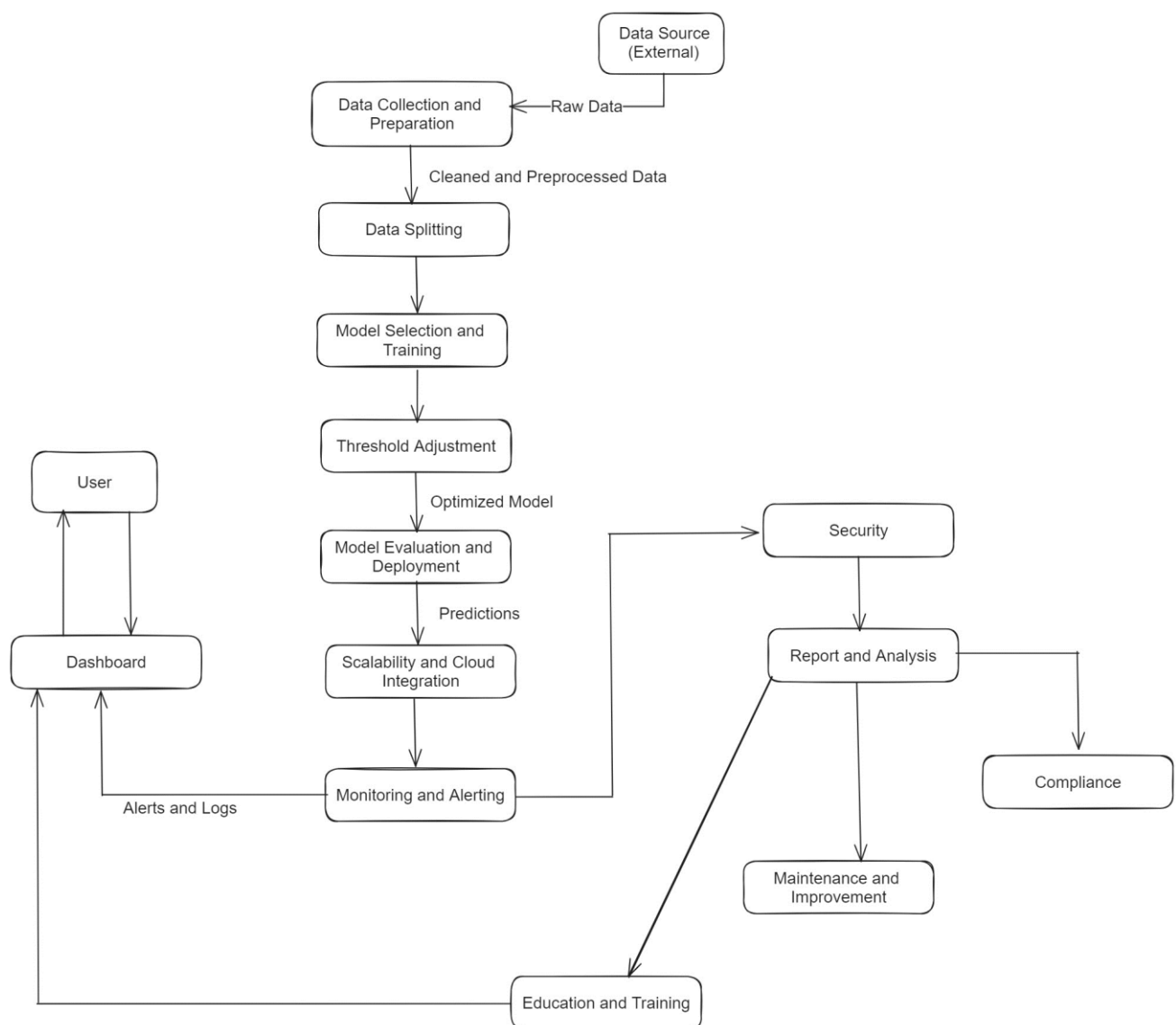
- User Training: Provide training sessions for system administrators and users to ensure efficient use of the fraud detection system.
- Customer Support: Offer a responsive customer support system to address user queries and issues promptly.

## 5. PROJECT DESIGN

### 5.1 Data Flow Diagrams & User Stories

- Dataset of historical transactions is the input to the system.
- Data collection and preparation step cleans, preprocesses, and converts the dataset to a numerical format.
- Data splitting step splits the dataset into training, validation, and test sets.
- Model selection step evaluates different machine learning algorithms to select the best model for the fraud detection system.
- Model training step trains the selected model on the training dataset using cross-validation to prevent overfitting.
- Threshold adjustment step adjusts the threshold for flagging a transaction as fraudulent to balance false positives and false negatives.
- Model evaluation step evaluates the model on the test set to assess its performance.
- Model deployment step deploys the model using a Flask-based interface with cloud services for real-time predictions.
- Real-time fraud predictions are the output of the system.
- Monitoring and alerting systems monitor the performance of the system and identify any issues early on.

- Compliance monitoring step ensures that the system complies with all legal and regulatory obligations.
- Reporting and visualization tools generate reports and analyses on fraud detection findings.
- Insights into system performance and fraud detection findings are used to improve the system's fraud detection capabilities.
- Regular upgrades and model fine-tuning step upgrades the system and fine-tunes the model to improve its ability to detect emerging fraud strategies and patterns.



**DFD Level 0 (Industry Standard)**



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Fraud Analyst	Online Payment Fraud Detection	USR 1: Monitoring Suspicious Transactions	As a fraud analyst, I want to be able to monitor and review transactions in real-time for any suspicious activity.	The system should provide a real-time dashboard displaying transaction data.	High	Sprint-1
		USR 2: Setting Custom Fraud Rules	As a fraud analyst, I want to define and configure custom fraud detection rules to match specific fraud patterns.	The system should allow me to create, modify, and delete custom rules.	Medium	Sprint-2
Compliance Officer	Regulatory Compliance and Reporting	USR 1: Compliance Reporting:	As a compliance officer, I need the system to generate compliance reports for regulatory authorities and internal audits.	The system should provide a report generation feature for various compliance requirements.	High	Sprint-3
Customer Support Agent	Customer Support Integration	USR 1: Customer Support Integration	As a customer support agent, I need access to the fraud detection system to assist customers with transaction-related inquiries and issues.	The system should provide a user-friendly interface for customer support agents.	Medium	Sprint-2
Machine Learning Engineer	Model Development and Improvement	USR 1: Model Performance Monitoring	As a machine learning engineer, I need the system to provide tools for monitoring and evaluating the performance of the machine learning models	The system should offer metrics and visualizations to assess model accuracy and effectiveness.	High	Sprint-3

System Administrator	System Management and Configuration	USR 1: System Configuration	As a system administrator, I want the ability to configure and maintain the fraud detection system.	The system should allow me to set system-wide parameters, such as risk thresholds and data retention policies.	High	Sprint-1
End User (Customer)	Secure and Seamless Transactions	USR 1: Safe Transactions	As an end user, I want the assurance that my online payments are secure and protected from fraud.	If a transaction is flagged as potentially fraudulent, I should receive an alert or request for additional authentication.	High	Ongoing
		USR 2: Real-time Alerts	As an end user, I want to receive real-time alerts and notifications if my transaction is flagged for potential fraud.	The alert should include clear instructions on how to verify my identity and complete the transaction.	High	Ongoing

## 5.2 Solution Architecture

**1. Data Collection and Preparation:** We received a dataset that has been thoroughly pre-processed, which includes cleaning it by removing duplicates, dealing with missing values, and resolving outliers. Categorical data was converted to numerical format using techniques such as one-hot encoding, and numerical characteristics were scaled and normalised to maintain consistency.

**2. Data Splitting:** We divided our dataset into training, validation, and test sets. We used an 80-10-10 split ratio to ensure that both fraudulent and lawful transactions are distributed consistently across these groupings.

**3. Model Selection:** We investigated many machine learning methods. In addition, we compared methods such as Gradient Boosting, Neural Networks, and Support Vector Machines. The choice is based on the unique needs of our fraud detection system.

**4. Model Training:** Using techniques and methodologies given by machine learning libraries such as scikit-learn, the selected models were rigorously trained on the training dataset. Cross-validation is used to evaluate model performance and reduce the danger of overfitting. To improve model performance, hyperparameter tuning approaches such as grid search and random search have been utilised.

**5. Threshold Adjustment:** We've adjusted thresholds depending on system requirements to create a balance between false positives and false negatives. To identify an optimum threshold, methods such as ROC analysis have been used.

**6. Model Evaluation:** To guarantee the efficacy of our models, we thoroughly examined them using measures such as accuracy, precision, recall, F1-score, and ROC AUC.

**7. Model Deployment:** To collect transaction data, our model is deployed using a Flask-based interface, with cloud services such as AWS Lambda, Azure Functions, or Google Cloud Functions for real-time predictions. For effective deployment and administration, we've also used containerization technologies like Docker and Kubernetes.

**8. Scalability and cloud integration:** For scalability and resource management, we put the system on cloud platforms such as AWS, Azure, or Google Cloud, which provide auto-scaling. Scalable data storage is achieved through the usage of cloud databases and data lakes.

**9. Monitoring and Alerting:** To ensure continuous performance, we've integrated strong monitoring technologies like Prometheus or DataDog, as well as alerting systems that warn administrators of performance concerns as soon as they occur.

**10. Security:** Security is a top priority for us, therefore we use data encryption, access limits, and secure communication protocols to protect critical information. Furthermore, we closely follow data protection standards, including GDPR compliance via data anonymization and user consent processes.

**11. Reporting and Analysis:** We've created reporting tools like dashboards and visualisation tools to gather insights and data, giving you a complete picture of the system's performance and fraud detection findings.

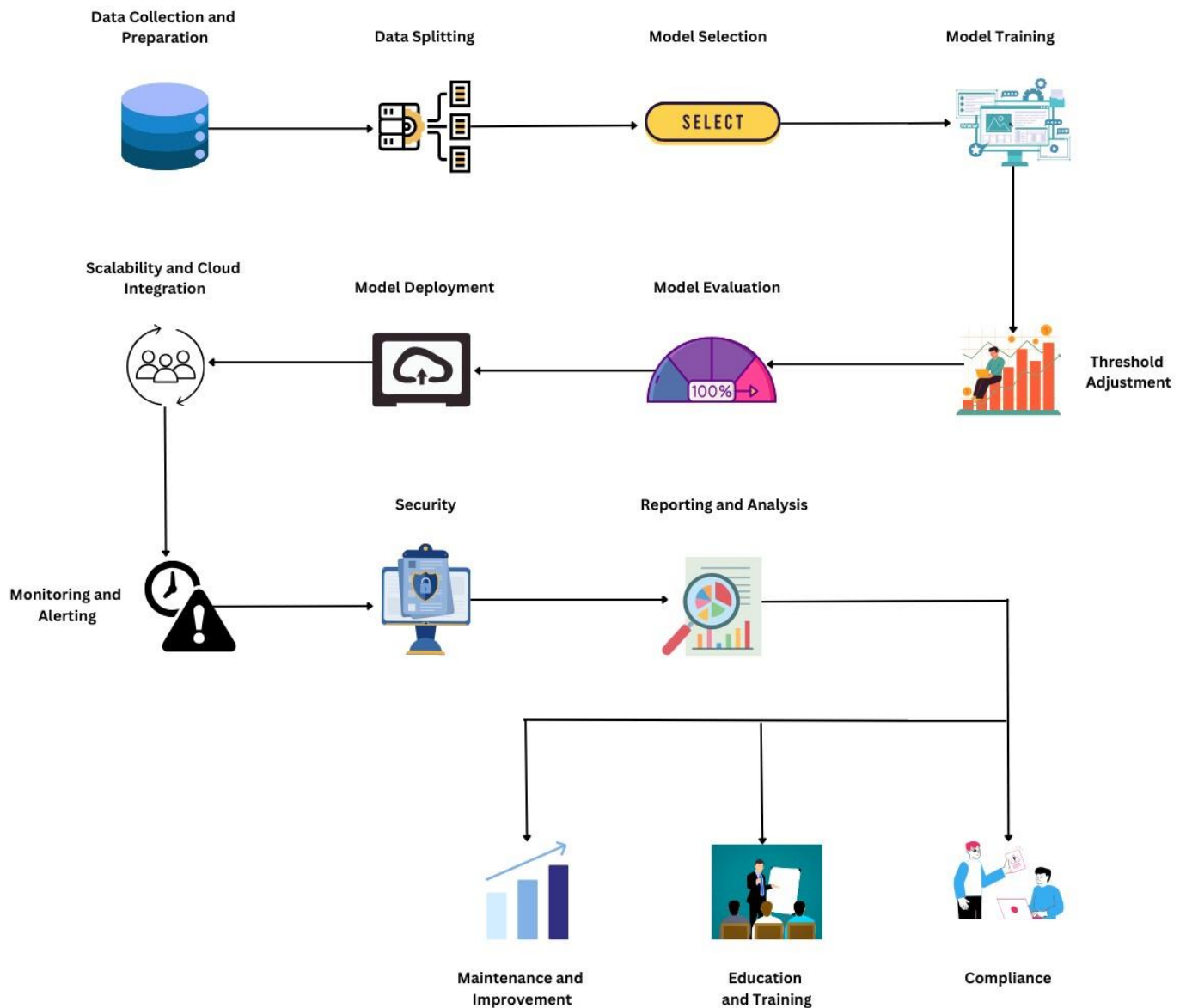
**12. Compliance:** We are committed to adhering to all legal and regulatory obligations. To guarantee full compliance, we regularly monitor and upgrade our system to match with current standards and consult with legal experts.

**13. Education and Training:** Our crew is knowledgeable about how to use and maintain the system, evaluate findings, and respond to fraud situations. Our plan includes regular training and instruction.

**14. Maintenance and Improvement:** We've built a disciplined plan for regular upgrades and model fine-tuning to react to emerging fraud strategies and patterns. Through ongoing research and

participation with industry journals, we remain up to date on the newest advances in fraud detection and machine learning.

## Solution Architecture Diagram

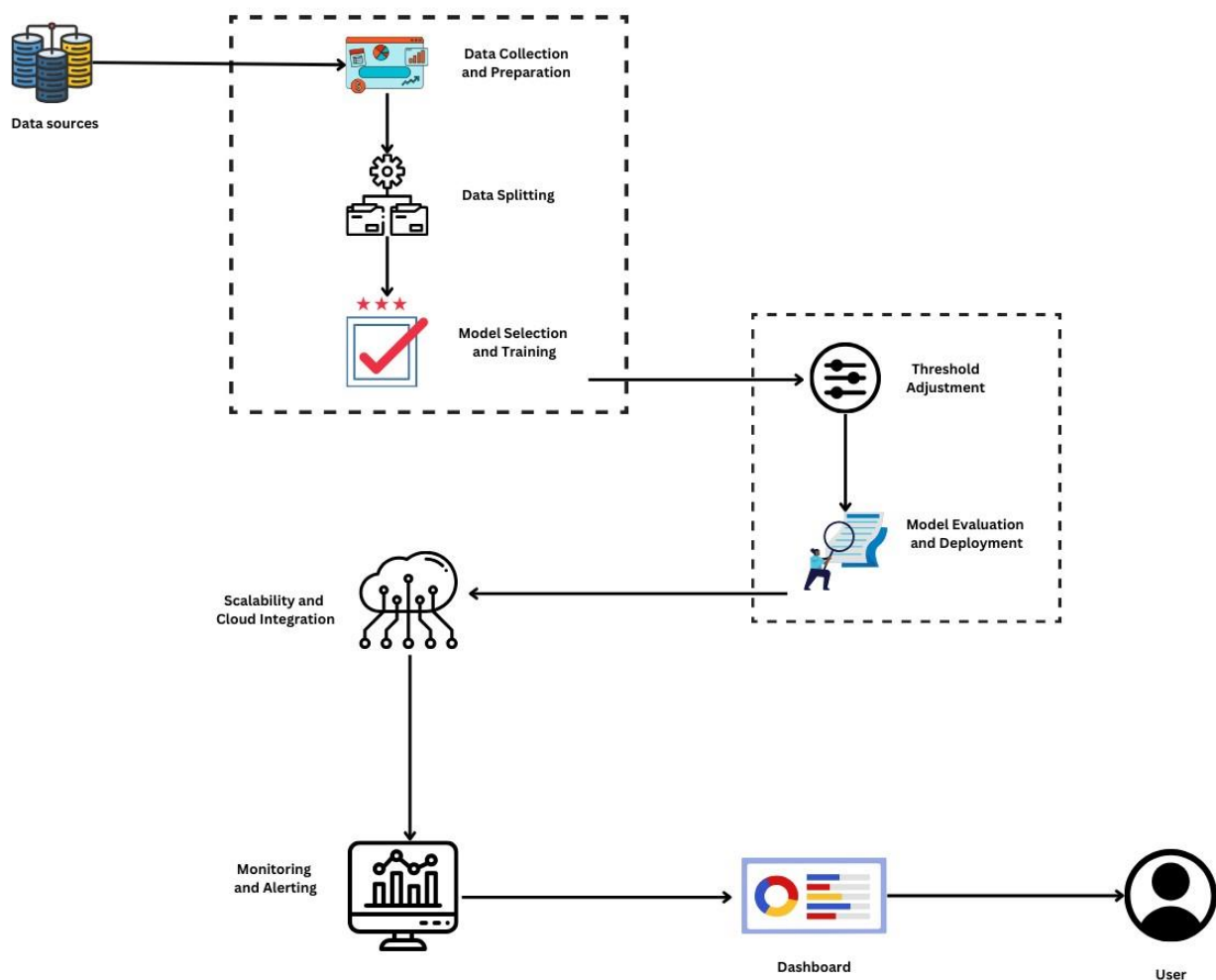


Solution Architecture Diagram for "Online Payments Fraud Detection Using ML"

## 6. PROJECT PLANNING & SCHEDULING

### 6.1 Technical Architecture

Data ingestion, preprocessing, model development, training, deployment, real-time data processing, model monitoring, a decision engine, alerting, feedback loops, security measures, scalability, integration, compliance, and continuous improvement comprise the technical architecture for Online Payment Fraud Detection Using ML. It detects and prevents fraudulent transactions while guaranteeing data security and regulatory compliance by leveraging machine learning models, real-time data streams, and automated decision-making.



**Table-1 : Components & Technologies :**

S.No	Component	Description	Technology
1.	User Interface	Creating a user-friendly and effective user interface (UI) for online payments fraud detection system is crucial for ensuring that users can easily interact with and make sense of the application.	HTML, CSS, JavaScript, BootStrap
2.	Database	Database contains Transaction, User information and Logs table. With a range of primary and foreign keys to associate each table with all attributes.	SQL, NO-SQL
3.	Application Logic-1	Logic for a process in the application	Python
4.	Application Logic-2	Application logic may maintain the state of the application, including user sessions, context, and data persistence.	Python
5.	Verification Process	Logic for a process in the application	Bank Information
6.	Cloud Database	Database Service on Cloud	IBM, IBM Cloudant etc.
7.	Machine Learning Model	The purpose of a machine learning (ML) model is to make predictions or decisions based on data. ML models are designed to automatically learn patterns and relationships within a dataset and use that knowledge to make informed predictions or decisions without being explicitly programmed.	Classification Model
8.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration Cloud Server Configuration	Local, Cloud Foundry, Kubernetes, etc.

**Table-2: Application Characteristics :**

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Open-source machine learning and data science frameworks to streamline your development and model-building process. These frameworks provide a wide range of tools, libraries, and resources to help you work efficiently and effectively.	Python, Pandas, Numpy, XGBoost
2.	Security Implementations	Implement strong user authentication methods such as multi-factor authentication (MFA) to ensure that only authorized users can access accounts and perform transactions. Enforce role-based access control (RBAC) to restrict access to sensitive systems and data to authorized personnel only.	User Authentication and Authorization
3.	Scalable Architecture	Consider using cloud services like AWS, Azure, or Google Cloud for scalability and elasticity, allowing you to adapt to varying workloads.	Google Cloud, AWS, Flask
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Amazon RDS (Relational Database Service) and Google Cloud SQL
5.	Performance	Good performance in online fraud detection requires a combination of advanced technologies, effective machine learning models, and robust processes for continuous improvement. Regular testing, optimization, and adaptation to evolving fraud patterns are key to maintaining high performance levels.	Caching mechanisms to store frequently accessed data and reduce response times, Optimizing machine learning model performance for real-time classification.

## 6.2 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint- 1	Online Payment Fraud Detection	USN-1	As a fraud analyst, I want to be able to monitor and review transactions in real-time for any suspicious activity.	2	High	Srajal Agarwal
Sprint- 1		USN-2	As a fraud analyst, I want to define and configure custom fraud detection rules to match specific fraud patterns.	1	High	Dhruv
Sprint- 2	Model Development and Improvement	USN-3	As a machine learning engineer, I need the system to provide tools for monitoring and evaluating the performance of the machine learning models	2	Low	Dhruv and Srajal Agarwal
Sprint- 3	System Management and Configuration	USN-4	As a system administrator, I want the ability to	1	Medium	Deepesh and Paras



			configure and maintain the fraud detection system.			
Sprint- 4	Secure and Seamless Transactions	USN-5	As an end user, I want the assurance that my online payments are secure and protected from fraud.	2	High	Deepesh and Paras

### 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	12 Oct 2023	18 Oct 2023	20	26 Oct 2022
Sprint- 2	20	3 Days	17 Oct 2023	20 Oct 2023	18	
Sprint- 3	20	5 Days	20 Oct 2023	25 Oct 2023	18	
Sprint- 4	20	4 Days	25 Oct 2023	29 Oct 2023	14	

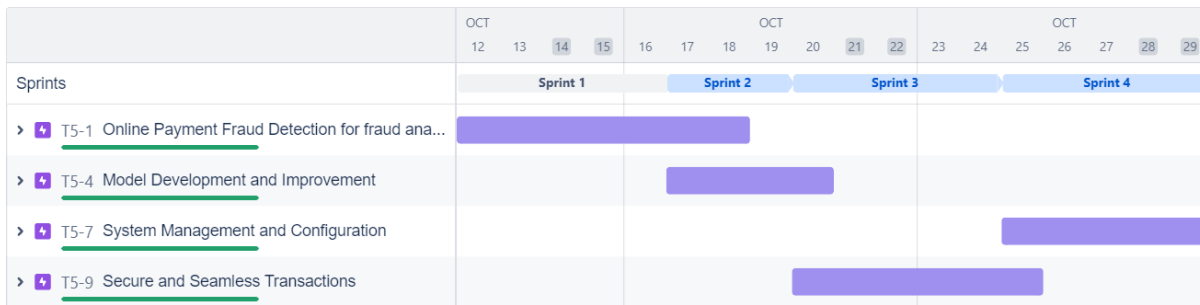
#### Velocity:

Sprint Duration: 70

Velocity: 5

$AV = \text{Sprint Duration} / \text{Velocity} = 70 / 4 = 17.5$

#### Burndown Chart:



## 7. CODING & SOLUTIONING

### 7.1 Feature 1

#### Milestone 1: Data Collection

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

**Collect the dataset or create the dataset or Download the dataset:**

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used PS\_20174392719\_1491204439457\_logs.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: <https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset>

#### Milestone 2: Visualising and analysing data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

**Note:** There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

#### Activity 1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

```
[ ] import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

## Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called `read_csv()` to read the dataset. As a parameter we have to give the directory of the csv file.

Here, the input features in the dataset are known using the `df.columns` function.

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell, labeled [4], contains the code to read a CSV file and display its first few rows. The second cell, labeled [5], contains the code to display the column names of the dataset.

```
df = pd.read_csv("C:\\ML MODEL DEPLOY\\Dataset.csv")
df
```

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

6362620 rows x 11 columns

```
df.columns
```

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

Here, the dataset's superfluous columns are being removed using the drop method.

```
[6]: df=df.drop(columns=['isFlaggedFraud'])
df.head()
```

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

## About Dataset

The below column reference:

1. step: represents a unit of time where 1 step equals 1 hour
2. type: type of online transaction
3. amount: the amount of the transaction
4. nameOrig: customer starting the transaction
5. oldbalanceOrg: balance before the transaction
6. newbalanceOrig: balance after the transaction
7. nameDest: recipient of the transaction
8. oldbalanceDest: initial balance of recipient before the transaction
9. newbalanceDest: the new balance of recipient after the transaction
10. isFraud: fraud transaction

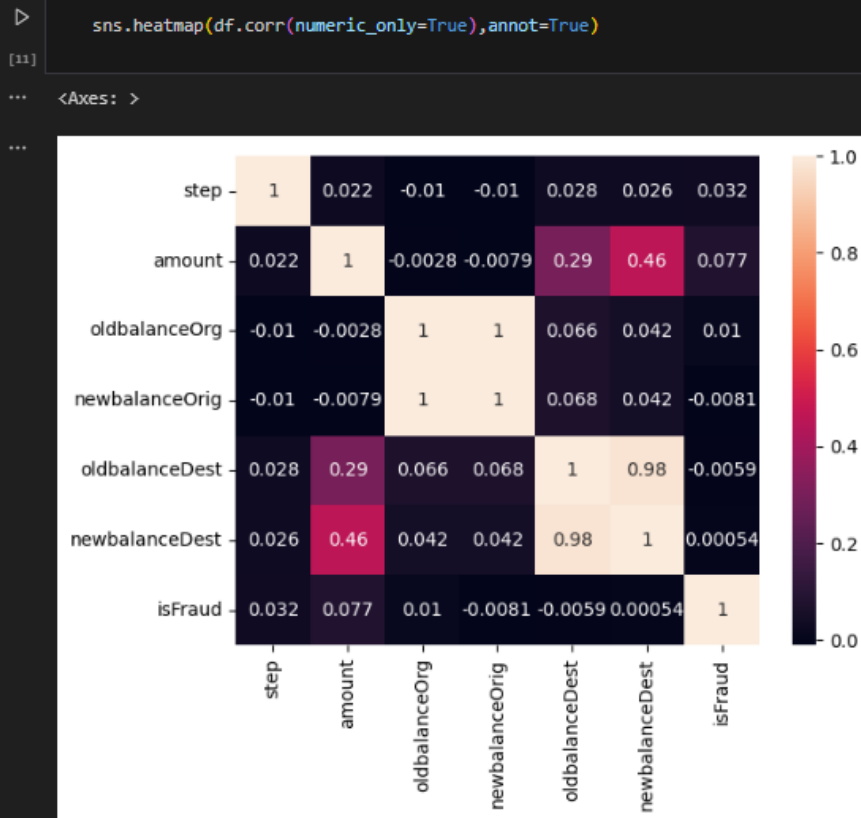
```
[8]: df.corr(numeric_only=True)['isFraud'].sort_values(ascending=False)
```

isFraud	1.000000
amount	0.076688
step	0.031578
oldbalanceOrg	0.010154
newbalanceDest	0.000535
oldbalanceDest	-0.005885
newbalanceOrig	-0.008148

Name: isFraud, dtype: float64

Utilising the corr function to examine the dataset's correlation

## HEATMAP



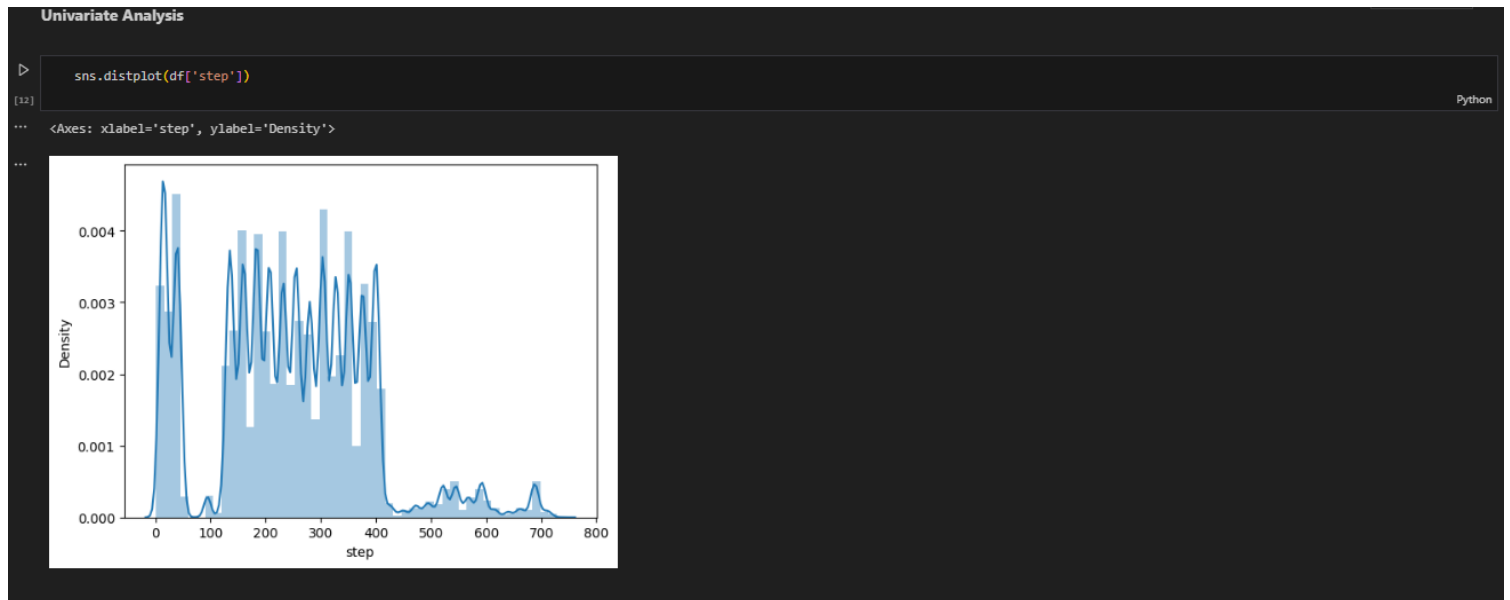
Here, a heatmap is used to understand the relationship between the input attributes and the anticipated goal value.

## Activity 3: Univariate analysis

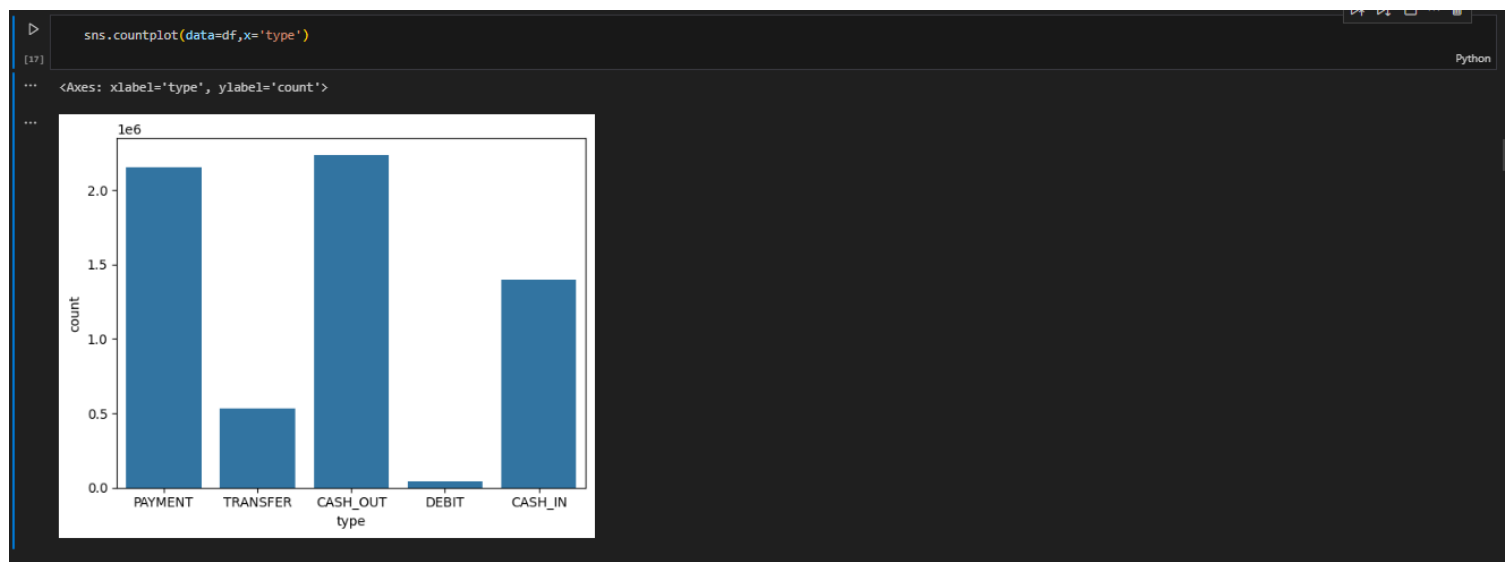


In simple words, univariate analysis is understanding the data with a single feature. Here I have displayed the graph such as distplot.

The distribution of one or more variables is represented by a histogram, a traditional visualisation tool, by counting the number of observations that fall within.



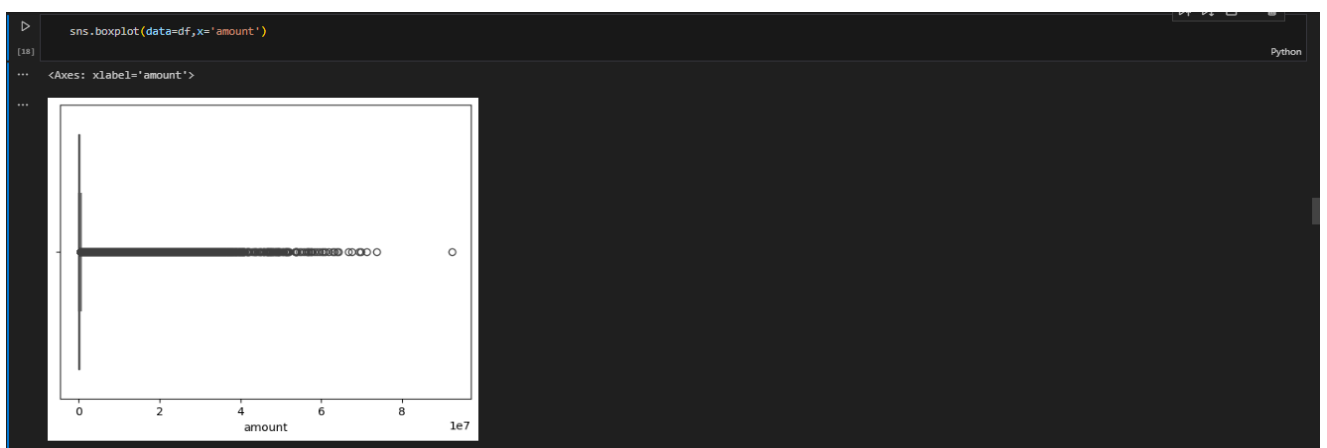
Here, the relationship between the step attribute and the boxplot is visualised.



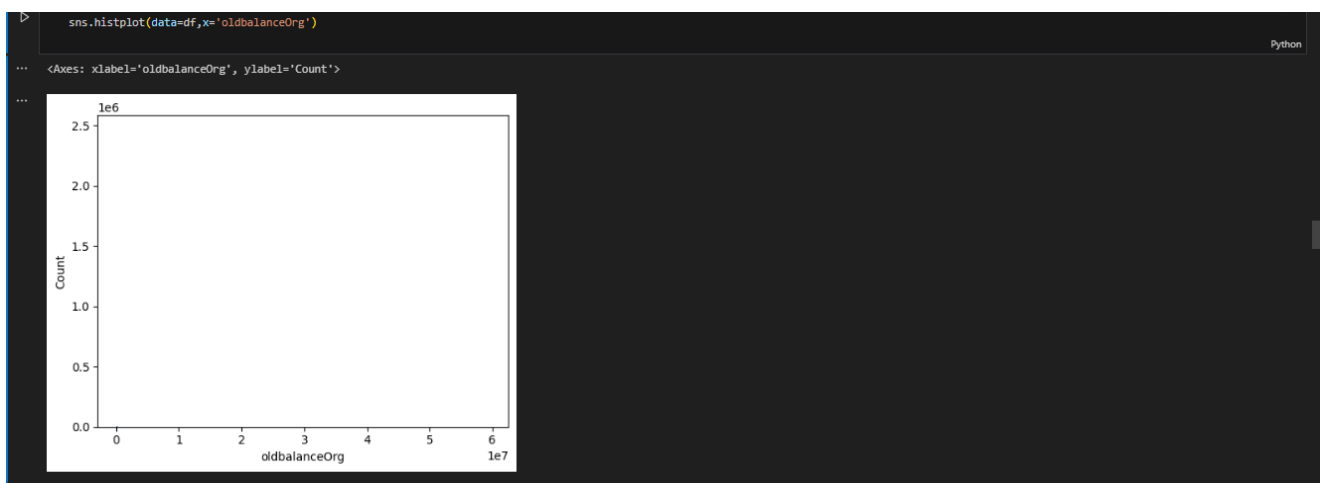
Here, the counts of observations in the type attribute of the dataset will be displayed using a countplot.



By creating bins along the data's range and then drawing bars to reflect the number of observations that fall within the amount attribute in the dataset.



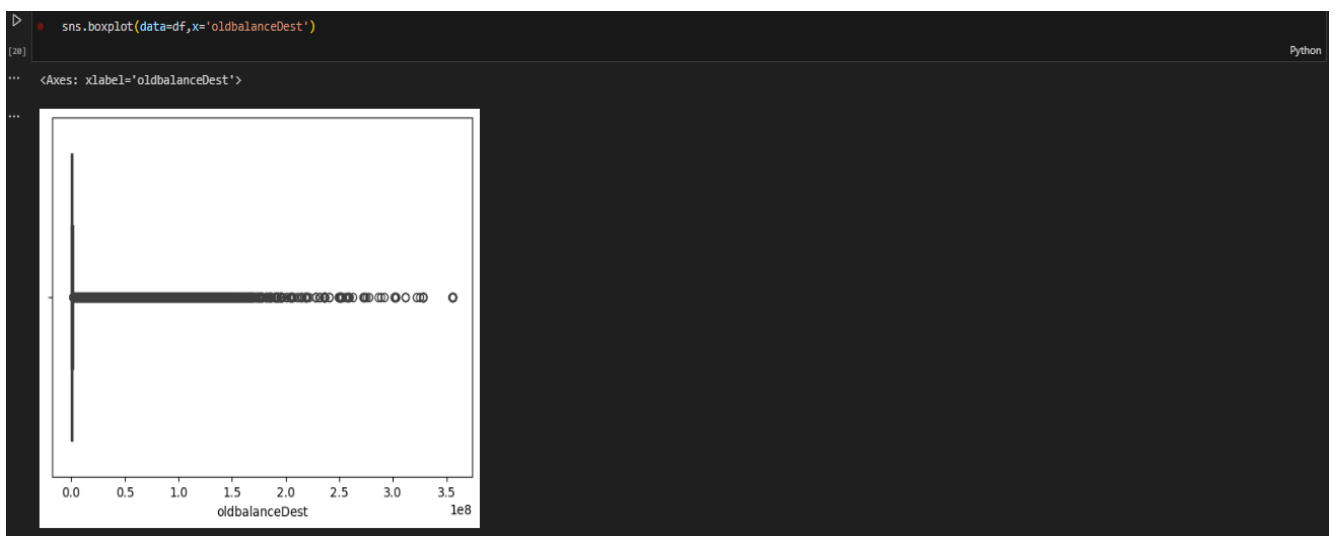
Here, the relationship between the amount attribute and the boxplot is visualised.



By creating bins along the data's range and then drawing bars to reflect the number of observations that fall within the oldbalanceOrg attribute in the dataset.

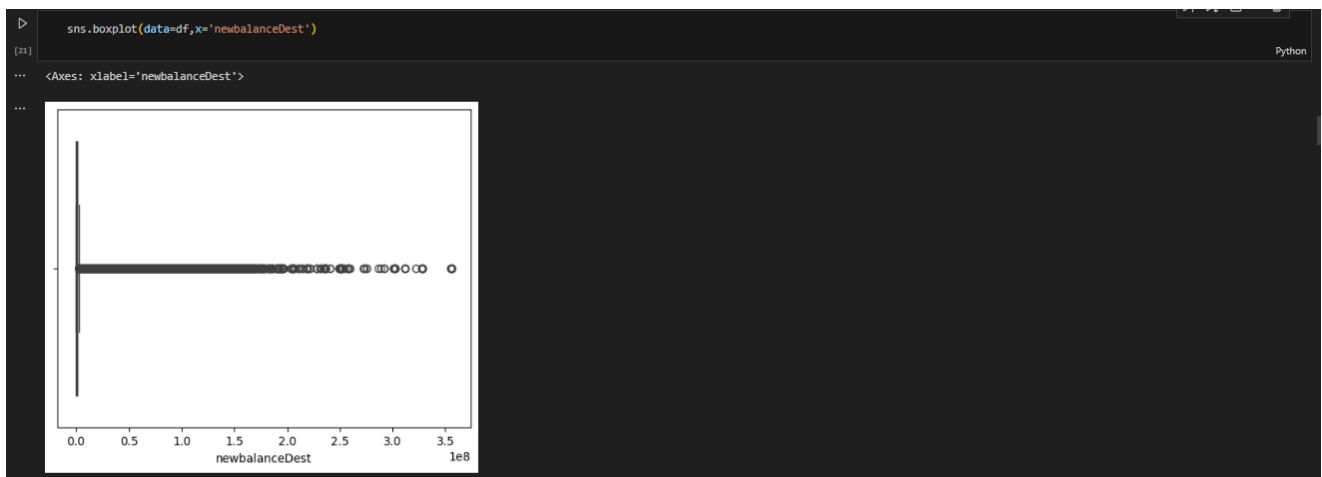
```
> df['nameDest'].value_counts()
[19]
... nameDest
C1286084959 113
C985934102 109
C665576141 105
C2083562754 102
C248609774 101
...
M1470027725 1
M1330329251 1
M1784358659 1
M2081431099 1
C2080388513 1
Name: count, Length: 2722362, dtype: int64
```

Utilising the value counts() function here to determine how many times the nameDest column appears.



Here, the relationship between the oldbalanceDest attribute and the boxplot is visualised.





Here, the relationship between the newbalanceDest attribute and the boxplot is visualised.



Using the countplot approach here to count the number of instances in the dataset's target isFraud column.

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

[23]

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

Here, we're using the value counts method to figure out how many classes there are in the dataset's target isFraud column.

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

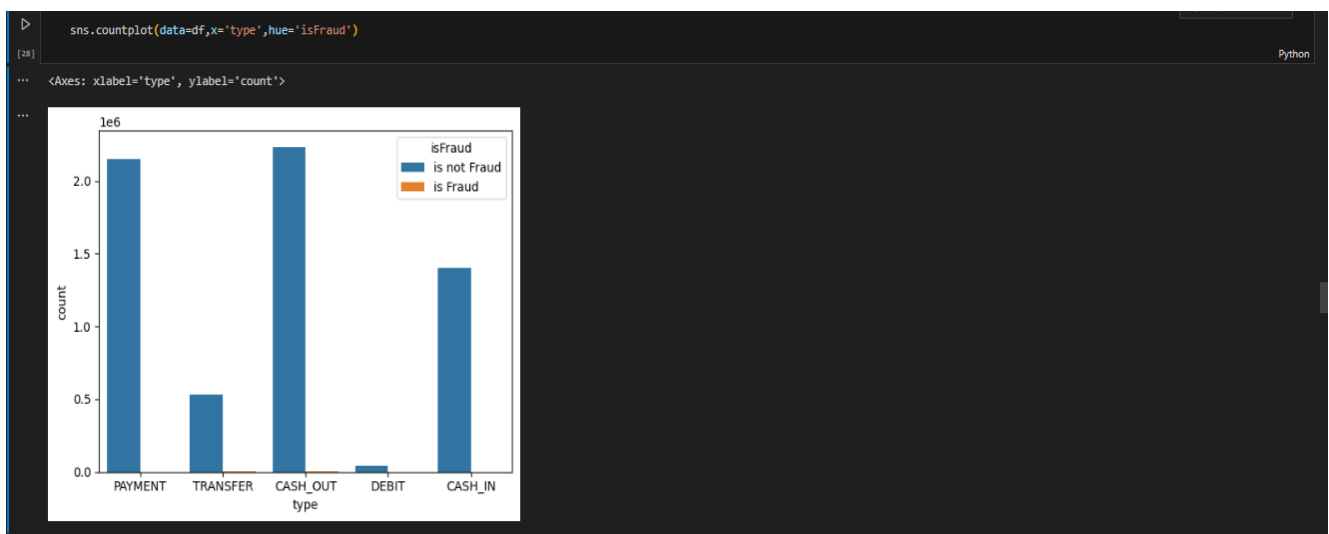
```
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	is not Fraud
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	is not Fraud
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	is Fraud
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	is Fraud
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	is not Fraud

Converting 0-means: is not fraud and 1-means: is fraud using the loc technique here

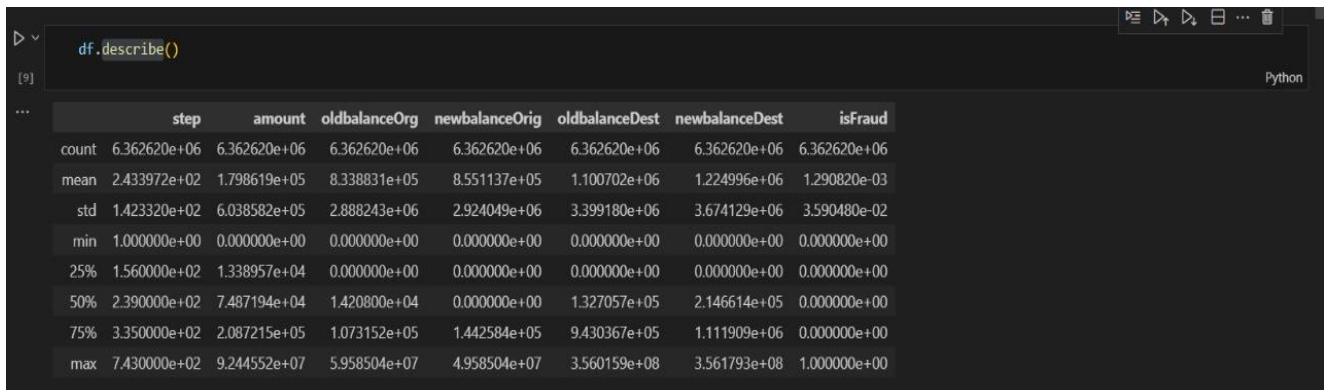
#### Activity 4: Bivariate analysis

Here we are visualising the relationship between type and isFraud.countplot is used here. As a 1<sup>st</sup> parameter we are passing x value and as a 2<sup>nd</sup> parameter we are passing hue value.



#### Activity 5: Descriptive analysis

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



A Jupyter Notebook interface showing the output of the `df.describe()` function. The output is a summary statistics table for the dataset.

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

## Milestone 3: Data Pre-processing

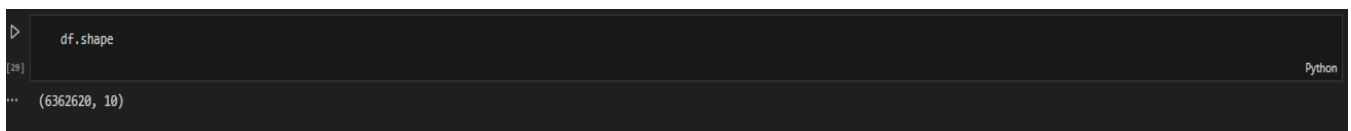
As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

Handling missing values

Handling Object data label encoding

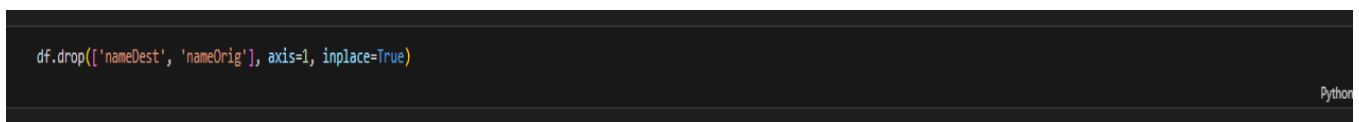
Splitting dataset into training and test set



A Jupyter Notebook interface showing the output of the `df.shape` function. The output is a tuple representing the dimensions of the dataset.

	df.shape
...	(6362620, 10)

Here, I'm using the shape approach to figure out how big my dataset is



A Jupyter Notebook interface showing the code to drop columns from the dataset.

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

Here, the dataset's superfluous columns (nameOrig, nameDest) are being removed using the drop method.

### Activity 1: Checking for null values

IsNull is used (`df.isnull().sum()`) to check your database for null values. Using the `df.info()` function, the data type can be determined.

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

[24] Python

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

For checking the null values, data.isnull() function is used. To sum those null values we use the .sum() function to it. From the above image we found that there are no null values present in our dataset. So we can skip handling of missing values step.

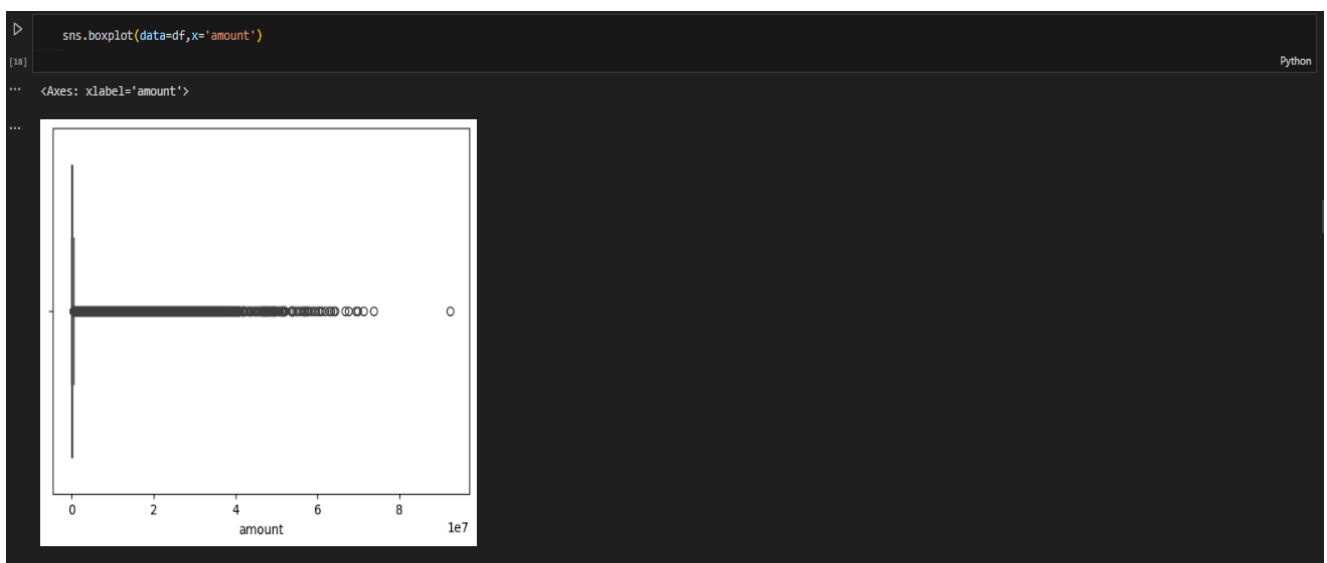
```
df.info()

[18] Python

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

Determining the types of each attribute in the dataset using the info() function.

## Activity 2: Handling outliers



Here, a boxplot is used to identify outliers in the dataset's amount attribute.

```

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

[31] Python
... ModeResult(mode=10000000.0, count=3207)
179861.90354913071

q1 = df['amount'].quantile(0.25)
q3 = df['amount'].quantile(0.75)
IQR = q3-q1
whisker_width = 1.5
lower_whisker = q1 -(whisker_width*IQR)
upper_whisker = q3 + (whisker_width*IQR)
df['amount']=np.where(df['amount']>upper_whisker,upper_whisker,np.where(df['amount']<lower_whisker,lower_whisker,df['amount']))

[32] Python

print('Q1 = ', q1)
print('Q3 = ', q3)
print('IQR = ', IQR)
print('Upper Bound = ', upper_whisker)
print('Lower Bound = ', lower_whisker)

[33] Python
... Q1 = 13389.57
Q3 = 200721.4775
IQR = 195331.9075
Upper Bound = 501719.33875
Lower Bound = -279608.29125

```

### Activity 3: Object data labelencoding

```

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

[37] Python

df['type']= label_encoder.fit_transform(df['type'])
df['type'].unique()

[38] Python
... array([3, 4, 1, 2, 0])

```

Using labelencoder to encode the dataset's object type

### X & Y Split and Scaling Columns

```

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

[39] Python

```

```

x_scaled.head()

[41] Python
...

```

	step	type	amount	oldbalanceOrig	newbalanceOrig	oldbalanceDest	newbalanceDest
0	0.0	0.75	0.778262	0.002855	0.003233	0.000000	0.0
1	0.0	0.75	0.684441	0.000357	0.000391	0.000000	0.0
2	0.0	1.00	0.552912	0.000003	0.000000	0.000000	0.0
3	0.0	0.25	0.552912	0.000003	0.000000	0.000059	0.0
4	0.0	0.75	0.787875	0.000697	0.000603	0.000000	0.0

```

y.head()

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

```

## Activity 4: Splitting data into train and test

Now let's split the Dataset into train and test sets. Changes: first split the dataset into x and y and then split the data set.

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And my target variable is passed. For splitting training and testing data we are using the `train_test_split()` function from `sklearn`. As parameters, we are passing x, y, `test_size`, `random_state`.

```
[43] from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_scaled, y, random_state=50, test_size=0.3) Python

[44] X_train.shape Python
... (4453822, 7)

[45] X_test.shape Python
... (1908782, 7)

[46] y_train.shape Python
... (4453822,)

[47] y_test.shape Python
... (1908782,)
```

## Milestone 4: Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

### Activity 1: Logistic Regression

Logistic regression is a statistical method for predicting the probability of a binary outcome (e.g., yes or no, spam or not spam). It is a popular classification algorithm due to its simplicity and interpretability. The basic idea behind logistic regression is to fit a linear equation to the data and then use the sigmoid function to transform the linear output into a probability

```
Logistic Regression ? markdown

from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
y=df['isFraud']
X=df.drop('isFraud',axis=1)
X_scaled=pd.DataFrame(scale.fit_transform(X),columns=X.columns)
X_scaled.head()

[41] Python

...

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.2,random_state=42)

[42] Python

> from imblearn.over_sampling import SMOTE
y_train.value_counts()
sm=SMOTE(random_state=42)
x_train_smote,y_train_smote=sm.fit_resample(X_train,y_train)
print("The values after scaling are")
y_train_smote.value_counts()

[43] Python

... The values after scaling are

... isFraud
is not Fraud    5083473
is Fraud        5083473
Name: count, dtype: int64
```

Between 0 and 1. The sigmoid function is a mathematical function that squashes any real number to a value between 0 and 1.

```
> from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
lr=LogisticRegression()
lr.fit(x_train_smote,y_train_smote)
y_pred=lr.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,lr.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))

[44] Python

... Training Score 0.8966881500108202
Testing Accuracy 0.9662284551689128

[45] Python

confusion_matrix(y_test,y_pred)

... array([[ 1312,    275],
          [ 42700, 1228234]], dtype=int64)

[46] Python

pd.crosstab(y_test,y_pred)

...

[47] Python

print(classification_report(y_test,y_pred))

...
              precision    recall  f1-score   support

 is Fraud         0.03         0.83         0.06         1587
is not Fraud         1.00         0.97         0.98        1270934

 accuracy              0.97        1272521
 macro avg           0.51         0.90         0.52        1272521
 weighted avg          1.00         0.97         0.98        1272521
```

## Activity 2: Random Forest classifier

A function named RandomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
Random Forest Classifier
```

```
[58] from sklearn.ensemble import RandomForestClassifier
    rfc =RandomForestClassifier(criterion='entropy', max_depth=5,
    min_samples_leaf=10, min_samples_split=10,
    n_estimators=50)
    rfc.fit(x_train_smote,y_train_smote)

...

[59] y_pred=rfc.predict(X_test)
    print("Training Score",accuracy_score(y_train_smote,rfc.predict(x_train_smote)))
    print("Testing Accuracy",accuracy_score(y_test,y_pred))
    print(X_test.shape)

...
Training Score 0.9386852256321613
Testing Accuracy 0.9837896584810781
(1272521, 7)

[60] pd.crosstab(y_test,y_pred)

...

[61] print(classification_report(y_test,y_pred))

...
      precision    recall  f1-score   support

 is Fraud      0.07      0.91      0.12        1587
 is not Fraud    1.00      0.98      0.99       1270934

 accuracy                0.98       1272521

 accuracy      0.53      0.95      0.56       1272521
 macro avg      1.00      0.98      0.99       1272521
 weighted avg
```

## Activity 3: Decision tree Classifier

A function named Decisiontree is created and train and test data are passed as the parameters. Inside the function, the DecisiontreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



#### Decision Tree Classifier

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

```
▼ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
[ ] y_test_pred_2 = dtc.predict(X_test)
```

```
[ ] accuracy_test_2 = accuracy_score(y_test, y_test_pred_2)
    accuracy_test_2
```

```
0.9997053093819277
```

```
[ ] pd.crosstab(y_test, y_test_pred_2)
```

	col_0	is Fraud	is not Fraud
isFraud			
is Fraud	1403	184	
is not Fraud	191	1270743	

#### Activity 4: ExtraTrees Classifier

A function named ExtraTree is created and train and test data are passed as the parameters. Inside the function, ExtraTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

#### Extra tree Classifier

```
from sklearn.ensemble import ExtraTreesClassifier
etc=ExtraTreesClassifier()
etc.fit(x_train_smote,y_train_smote)
y_pred=etc.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,etc.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))
```

Python

```
... Training Score 1.0
    Testing Accuracy 0.9994451957963758
```

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

Python

```
print(classification_report(y_test,y_pred))
```

Python

```
... precision    recall  f1-score   support

 is Fraud      0.72      0.90      0.80      1587
 is not Fraud   1.00      1.00      1.00     1270934

 accuracy              1.00     1272521
 macro avg      0.86      0.95      0.90     1272521
 weighted avg    1.00      1.00      1.00     1272521
```

```
etc.predict([[0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00000000e+00,0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
```

Python

```
array(['is not Fraud'], dtype=object)
```

## Activity 5: Evaluating performance of the model and saving the model

Our model is performing well. So, we are saving the model is svc by pickle.dump().

```
import pickle
pickle.dump(rfc, open('model.pkl', 'wb'))
```

Python

## 7.2 Feature 2

### Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built.

A UI is provided for the uses where he has to enter the values for predictions.

The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script

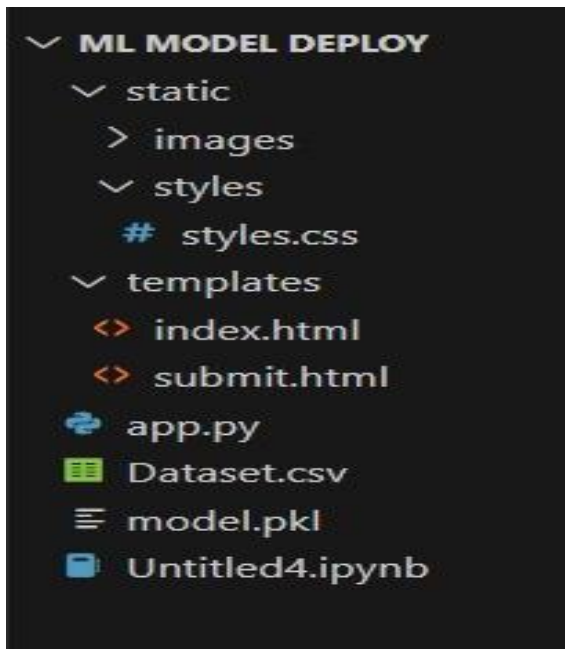
### Activity1: Building Html Pages:

For this project create three HTML files namely

- index.html
- submit.html

and save them in the templates folder.

As shown in the project structure



Let's see how our home.html page looks like:



## General Instructions

Here are some instructions for prevention from Online Fraud

1. Research the website and company: Before you provide any personal information or money, it is important to make sure that the website and company are legitimate. You can do this by searching for online reviews, checking the company's website for contact information, and verifying that the website is registered with a reputable domain registrar.
2. Be cautious of unsolicited emails or messages: If you receive an email or message from someone claiming to be from a company you are interested in enlisting with, be cautious. Do not click on any links or open any attachments in the message without first verifying that they are legitimate. You can do this by contacting the company directly through their website or a known phone number.
3. Never give out your personal information or financial information to someone you do not know and trust: Companies should never ask for your personal information, such as your Social Security number, bank account information, or credit card number, before you have been accepted into their program. If you are asked for this information, it is a red flag that the company may be fraudulent.
4. Be skeptical of offers that sound too good to be true: If you are offered a job or enlistment opportunity that seems too good to be true, it probably is. Do not be afraid to walk away from any offer that seems suspicious or if you are not comfortable with the terms.
5. Use a secure payment method: If you do need to pay a fee to enlist with a company, make sure you use a secure payment method, such as a credit card or PayPal. Do not wire money or send gift cards to anyone, as these are common payment methods used by scammers.
6. Report any fraudulent activity to the authorities: If you believe that you have been the victim of online fraud, you should report it to the authorities. You can contact the Federal Trade Commission (FTC) at 1-877-FTC-HELP (1-877-382-4357) or file a complaint online at ReportFraud.ftc.gov.

**CatchFraud is a great fraud detection tool  
for businesses of all sizes. It's easy to use  
and has a wide range of features.**



Harsh, India

Now when you click on predict button from top right corner you will get redirected to predict.html

Let's look how our predict.html file looks like:

## Predict

Input transaction details here and get prediction about payment is safe or fraud.

STEP

TYPE

PAYMENT



AMOUNT

OLD BALANCE ORG

NEW BALANCE ORG

OLD BALANCE DEST

NEW BALANCE DEST

SUBMIT

**Explore the Dataset by clicking on Dataset button below.**

 Dataset

 Reviews

     
© Copyright CatchFraud

Now when you click on submit button from left bottom corner you will get redirected to submit.html

Let's look how our submit.html file looks like:



## Activity 2: Build Python code:

Import the libraries

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the

current module (\_\_name\_\_) as argument.

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

model=pickle.load(open("C:\ML MODEL DEPLOY\model.pkl","rb"))
app=Flask(__name__)
```

Render HTML page:

```
dict_val= {'PAYMENT':0, 'TRANSFER':1 , 'CASH_OUT':2 , 'DEBIT':3 , 'CASH_IN':4}
@app.route("/")
def start():
    return render_template('index.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
def predict():
    return render_template('index.html')

@app.route("/Home")
def index():
    return render_template('index.html')

@app.route("/login",methods=['POST'])
def login():
    x=[x for x in request.form.values()]
    x=np.array(x)
    print(x)
    # pred=model.predict(x)
    x[0][1]=dict_val[x[0][1]]
    print(x)
    x=x.astype(float)
    output=model.predict(x)
    val=""
    if(output[0]==0):
        val="Not Fraud"
    else:
        val="Fraud"
    return render_template('submit.html',y=val)
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this

prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

### Main Function:

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

### Activity 3: Run the application

- Open VSCODE prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
PS C:\ML MODEL DEPLOY> python -u "c:\ML MODEL DEPLOY\app.py"  
* Serving Flask app 'app'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 749-454-911  
127.0.0.1 - - [09/Nov/2023 12:39:45] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [09/Nov/2023 12:39:45] "GET /static/styles/styles.css HTTP/1.1" 304 -  
127.0.0.1 - - [09/Nov/2023 12:39:45] "GET /static/images/boy.png HTTP/1.1" 304 -  
127.0.0.1 - - [09/Nov/2023 12:39:45] "GET /static/images/detective.png HTTP/1.1" 304 -
```

## 8. PERFORMANCE TESTING

### 8.1 Performace Metrics

#### Model Performance Testing:

Since the outcome of our project needed to be predicted as either "is a fraud" or "is not a fraud," a classification-based model was necessary.

Decision Tree Classifier, Logistic Regression, Extra Tree Classifier and Random Forest Classifier were the models utilized in the projects.

The metrics reports for each model are as follows:

## Decision Tree Classifier:

### 1. Test accuracy

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

DecisionTreeClassifier
DecisionTreeClassifier()

[ ] y_test_pred_2 = dtc.predict(X_test)

[ ] accuracy_test_2 = accuracy_score(y_test, y_test_pred_2)
accuracy_test_2
0.9997053093819277
```

### 2. Train accuracy

```
y_train_predict2=dtc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict2)
train_accuracy
1.0
```

### 3. Confusion Matrix

```
[ ] pd.crosstab(y_test, y_test_pred_2)
```

col_0	is Fraud	is not Fraud
is Fraud	1403	184
is not Fraud	191	1270743

### 4. Classification Report

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

	precision	recall	f1-score	support
is Fraud	0.39	0.47	0.43	19
is not Fraud	1.00	1.00	1.00	24873
accuracy			1.00	24892
macro avg	0.70	0.74	0.71	24892
weighted avg	1.00	1.00	1.00	24892



## Random Forest Classifier:

### 1. Test accuracy

```
y_pred=rfc.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,rfc.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))
print(X_test.shape)
```

Testing Accuracy 0.9837896584810781

### 2. Train accuracy

```
y_pred=rfc.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,rfc.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))
print(X_test.shape)
```

Training Score 0.9386852256321613

### 3. Confusion Matrix

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

	col_0	is Fraud	is not Fraud
isFraud			
is Fraud	1439	148	
is not Fraud	20480	1250454	

### 4. Classification Report

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

	precision	recall	f1-score	support
is Fraud	0.07	0.91	0.12	1587
is not Fraud	1.00	0.98	0.99	1270934
accuracy			0.98	1272521
macro avg	0.53	0.95	0.56	1272521
weighted avg	1.00	0.98	0.99	1272521

## Logistic Regression:

### 1. Test accuracy

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
lr=LogisticRegression()
lr.fit(x_train_smote,y_train_smote)
y_pred=lr.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,lr.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))
```

Training Score 0.8966881500108292

### 2. Train accuracy

```

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
lr=LogisticRegression()
lr.fit(x_train_smote,y_train_smote)
y_pred=lr.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,lr.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))

```

Testing Accuracy 0.9662284551689128

### 3. Confusion Matrix

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

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1312	275
is not Fraud	42700	1228234

### 4. Classification Report

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
is Fraud	0.03	0.83	0.06	1587
is not Fraud	1.00	0.97	0.98	1270934
accuracy			0.97	1272521
macro avg	0.51	0.90	0.52	1272521
weighted avg	1.00	0.97	0.98	1272521

## Extra Tree Classifier:

### 1. Test accuracy

```

from sklearn.ensemble import ExtraTreesClassifier
etc=ExtraTreesClassifier()
etc.fit(x_train_smote,y_train_smote)
y_pred=etc.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,etc.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))

```

Testing Accuracy 0.9994451957963758

### 2. Train accuracy

```

from sklearn.ensemble import ExtraTreesClassifier
etc=ExtraTreesClassifier()
etc.fit(x_train_smote,y_train_smote)
y_pred=etc.predict(X_test)
print("Training Score",accuracy_score(y_train_smote,etc.predict(x_train_smote)))
print("Testing Accuracy",accuracy_score(y_test,y_pred))

```

Training Score 1.0

### 3. Confusion Matrix

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

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1426	161
is not Fraud	545	1270389

4. Classification Report

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
is Fraud	0.72	0.90	0.80	1587
is not Fraud	1.00	1.00	1.00	1270934
accuracy			1.00	1272521
macro avg	0.86	0.95	0.90	1272521
weighted avg	1.00	1.00	1.00	1272521

Final Prediction:

```
etc.predict([[0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00000000e+00,0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
```

```
array(['is not Fraud'], dtype=object)
```

9. RESULTS

9.1 Output Screenshots

When transaction is fraud

STEP

1

TYPE

PAYMENT

AMOUNT

13134

OLD BALANCE ORG

3143241234

NEW BALANCE ORG

3414324

OLD BALANCE DEST

1341343

NEW BALANCE DEST

3413143

SUBMIT

# CatchFraud

**Your transaction is Fraud**

If you want to predict again then click on the Home button below

[Home](#)[Dataset](#)

## When transaction is not fraud

STEP
1
TYPE
PAYMENT
AMOUNT
1312
OLD BALANCE ORG
132
NEW BALANCE ORG
14334
OLD BALANCE DEST
0
NEW BALANCE DEST
0
<a href="#">SUBMIT</a>



## 10. ADVANTAGES & DISADVANTAGES

### Advantages

- 1. Improved Accuracy:** - ML algorithms, as opposed to conventional rule-based systems, are more accurate and successful in detecting fraud because they can analyse large volumes of transaction data and spot intricate patterns.
- 2. Adaptability to changing Threats:** - By continuously learning from incoming data, machine learning models are able to react to new and changing fraud strategies, offering a dynamic defence against emerging threats.
- 3. Real-Time Detection:** ML makes it possible to monitor and identify fraudulent transactions in real time, enabling prompt action and the avoidance of unauthorised activity.
- 4. Reduced False Positives:-** When appropriately adjusted machine learning models can reduce false positives by learning to discern between authentic and fraudulent patterns, hence alleviating the annoyance experienced by authentic consumers.
- 5. ML automates the decision-making process,** making it possible to analyse huge datasets quickly and effectively without the need for human interaction.
- 6. Scalability:** Machine learning models possess the potential to accommodate substantial transaction volumes, rendering them appropriate for the growing number of online transactions.
- 7. Cross-Channel Detection:** ML-based systems may offer a comprehensive perspective of user behaviour across many channels, which enhances the ability to identify fraud that occurs across various platforms.
- 8. Continuous Learning:** - Without the need for frequent human updates, machine learning models

have the ability to learn from fresh data over time and adjust to shifts in user behaviour and fraud tendencies.

**9. Better fraud detection:** The project's goal is to lower the amount of instances of successful online payment fraud, safeguarding customers and companies alike.

**10. Cost savings:** Businesses can save a lot of money by reducing the financial losses brought on by fraud.

**11. Increased user trust:** Users' confidence in online payment systems is bolstered by effective fraud detection, which encourages greater usage.

**12. Real-time response:** The capacity to monitor and detect fraudulent transactions in real-time will aid in stopping them as they happen.

**13. Adaptability:** Machine learning models possess the ability to adjust to evolving fraudulent trends and offer continuous defence against new dangers.

## **Disadvantages:**

**1. Difficulty in Feature Engineering:** - Finding pertinent features in the complexity of transaction data and the potential need for domain-specific expertise make it difficult to construct efficient models.

**2. Interpretability Issues:** - A lot of machine learning models are opaque, which makes it challenging to decipher and comprehend the logic underlying their fraud detection conclusions. This can undermine regulatory compliance and confidence.

**3. Initial Implementation Cost:** - High expenses associated with data acquisition may be incurred during the initial setup and implementation of an ML-based fraud detection system, creation of models and their incorporation into current systems.

**4. Overfitting:** - Machine learning models may overfit to the training set, identifying noise instead of real fraud trends. This might hinder the model's capacity to generalise to new sets of data.

**5. Dependency on High-Quality Data:** - The representativeness and high quality of the training data have a significant impact on the efficacy of machine learning models. Performance that is below par can be caused by biased or inaccurate data.

## **11. CONCLUSION**

To sum up, the project "Online Payment Fraud Detection using Machine Learning" is an important and creative attempt to improve the security and dependability of online payments. This research

intends to solve the dynamic issues presented by fraudulent activity in the online payment space by investigating sophisticated machine learning approaches.

#### **Important Accomplishments:**

**1.Enhanced Precision and Effectiveness:** - When compared to earlier rule-based systems, the accuracy and efficiency of fraud detection have greatly increased with the use of machine learning algorithms. The model has produced a more effective defence against fraudulent operations because of its capacity to evaluate enormous volumes of transaction data and adjust to changing patterns.

**2. Real-Time Monitoring and Response:-** The system's integration of real-time monitoring features allows it to identify and address fraudulent activities right away. In order to avoid financial losses and protect the interests of both enterprises and consumers, this quick action is essential.

**3. Equilibrated Accuracy and Memory:** - The study has struck a balance between precisely detecting counterfeit patterns and reducing false positives through painstaking tuning and optimisation. This maintains a high degree of security while ensuring that legitimate users encounter the least amount of inconvenience.

**4. Sensitivity to Developing Dangers:-** The ability of the machine learning model to learn continuously has shown to be useful in adjusting to novel and developing fraud strategies. This flexibility is crucial in an environment where fraudulent activity is changing quickly.

**5. Detection of Cross-Channel Fraud:-** The project has effectively tackled the problem of identifying fraud that occurs across several channels or platforms by using a comprehensive approach. This thorough understanding of The way users behave improves the system's capacity to spot complex cross-channel fraud schemes.

#### **Difficulties and Points to Remember:**

**1. Imbalanced Datasets:** - To lessen the impact on model training, the project has used techniques like oversampling, undersampling, or sophisticated algorithms like SMOTE. It recognises the fundamental issue of imbalanced datasets.

**2. Comprehensibility and Elucidability:** - An attempt has been made to improve the machine learning model's interpretability and explainability in light of the significance of transparency. This takes care of issues with trust and legal compliance.

**3. Data Privacy safeguards:** - The project has put strong data privacy safeguards in place to guarantee compliance with pertinent rules, given the sensitivity of financial data. To safeguard user information, privacy-preserving strategies like differential privacy have been investigated.

## **12. FUTURE SCOPE**

**1. Ongoing Model Surveillance and Updates:-** The research highlights how crucial it is to regularly upgrade the machine learning model in order to accommodate new fraud tendencies and to continuously monitor the model's effectiveness.

**2. Cooperation and Exchange of Knowledge:-** To remain ahead of new dangers as a group, the project promotes industrial cooperation and knowledge exchange. Working together promotes a community-based strategy for addressing the rapidly changing domain of online payment fraud.

**3. Integration with new Technologies: -** In order to further improve the security and transparency of online payment systems, the project recommends investigating the integration of new technologies like blockchain and artificial intelligence.

To sum up, the project "Online Payment Fraud Detection in ML" represents a noteworthy development in strengthening the security framework of online transactions. Its accomplishments in precision, instantaneous reaction, and flexibility Present it as a useful tool in the continuous fight against online payment fraud. This initiative lays the groundwork for upcoming developments and teamwork to establish a more secure and safe online payment environment as technology advances.

## **13. APPENDIX**

### **GitHub Link**

<https://github.com/smartinternz02/SI-GuidedProject-600460-1697463262>

### **Project Demo Link**

[https://drive.google.com/file/d/1txdjHI8LB0bfGtPpTYXvpqLqT3awdnSS/view?usp=drive\\_link](https://drive.google.com/file/d/1txdjHI8LB0bfGtPpTYXvpqLqT3awdnSS/view?usp=drive_link)