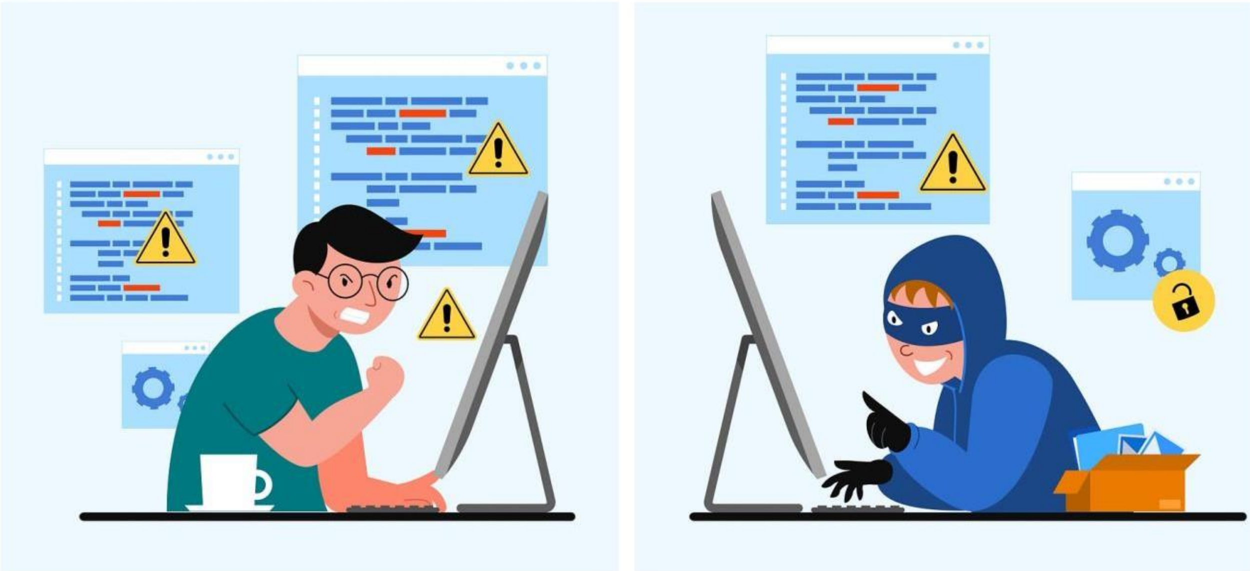
Project Report: Online Payments Fraud Detection Using ML



1. INTRODUCTION

1.1 Project Overview

The rapid growth of internet and e-commerce has led to a significant increase in online credit/debit card transactions. However, this surge in usage has also given rise to a parallel increase in fraudulent activities. Detecting fraud in online transactions is crucial for maintaining the integrity and security of financial systems. While various approaches exist for fraud detection, achieving high accuracy remains a challenge, often accompanied by specific drawbacks. This project aims to address these challenges through the implementation of advanced classification algorithms.

1.2 Purpose

The purpose of this project is to address the escalating issue of fraudulent activities in online credit/debit card transactions, a consequence of the burgeoning growth in internet and e-commerce usage. As online financial transactions become more prevalent, the need for robust fraud detection mechanisms becomes paramount to ensure the security and trustworthiness of digital financial systems.

2. LITERATURE SURVEY

2.1 Existing problem

The proliferation of internet and e-commerce has ushered in a new era of convenience and accessibility for consumers engaging in online credit/debit card transactions. However, this surge in digital transactions has come hand-in-hand with a significant and growing challenge: the increase in online payments fraud. As described in the project overview, the problem stems from the rise in the use of credit/debit cards for online transactions, leading to an uptick in fraudulent activities.

The existing problem revolves around the surge in online credit/debit card transactions leading to an increase in fraud. The limitations of traditional fraud detection methods and the challenges posed by the dynamic nature of fraud patterns underscore the need for more advanced and adaptive solutions. The project aims to contribute to the resolution of this issue by leveraging machine learning algorithms and deploying a user-friendly web application on the IBM Cloud.

2.2 References

1. Decision Analytics Journal

https://www.sciencedirect.com/science/article/pii/S2772662223000036

2. Research Gate

https://www.researchgate.net/publication/363894144_Financial_Fraud_Detection_Based_on_Machine_Learning_A_Systematic_Literature_Review

2.3 Problem Statement Definition

The exponential growth of internet and e-commerce has led to a significant surge in online credit/debit card transactions, making it a convenient and widely adopted method for financial transactions. However, this surge has also given rise to a critical challenge: the escalation of online payments fraud. The inherent vulnerabilities in the current systems for detecting fraudulent activities, coupled with the dynamic nature of fraud patterns, create a pressing need for more accurate and adaptive solutions.

3. IDEATION & PROPOSED SOLUTION

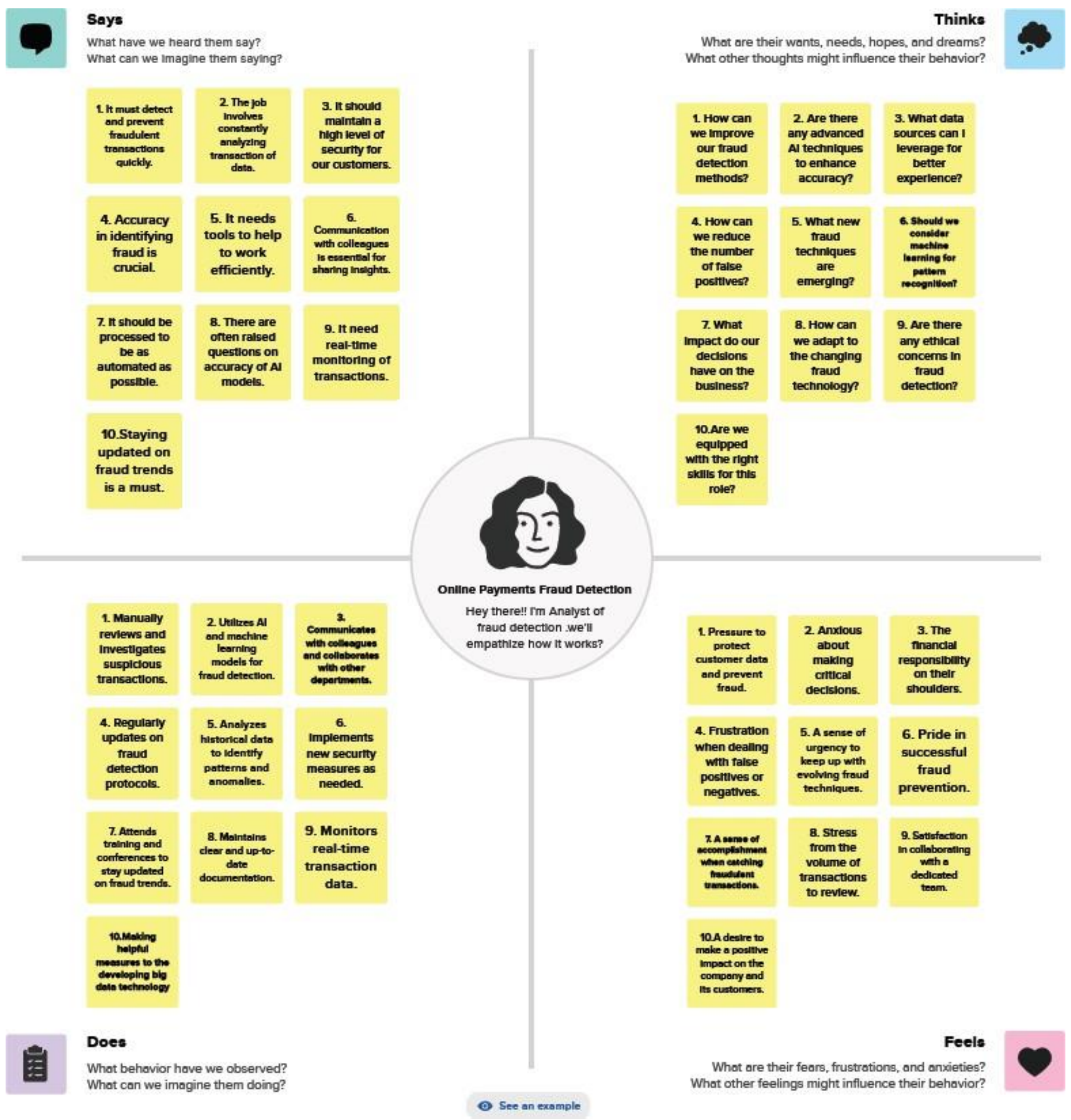
3.1 Empathy Map Canvas

In the realm of online payments fraud detection, our user, a vigilant financial professional, expresses concern about the escalating fraud rates and the limitations of current detection methods. Frustrated by the inefficiencies in existing systems, they seek a solution that adapts to the dynamic nature of fraud patterns. Eager for accuracy and real-time insights, our user envisions a more efficient and secure environment for online transactions. By adopting advanced machine learning techniques, our project aims to address these pain points, providing a user-friendly interface that empowers the user to proactively prevent fraud, ensuring trust and integrity in online financial transactions.

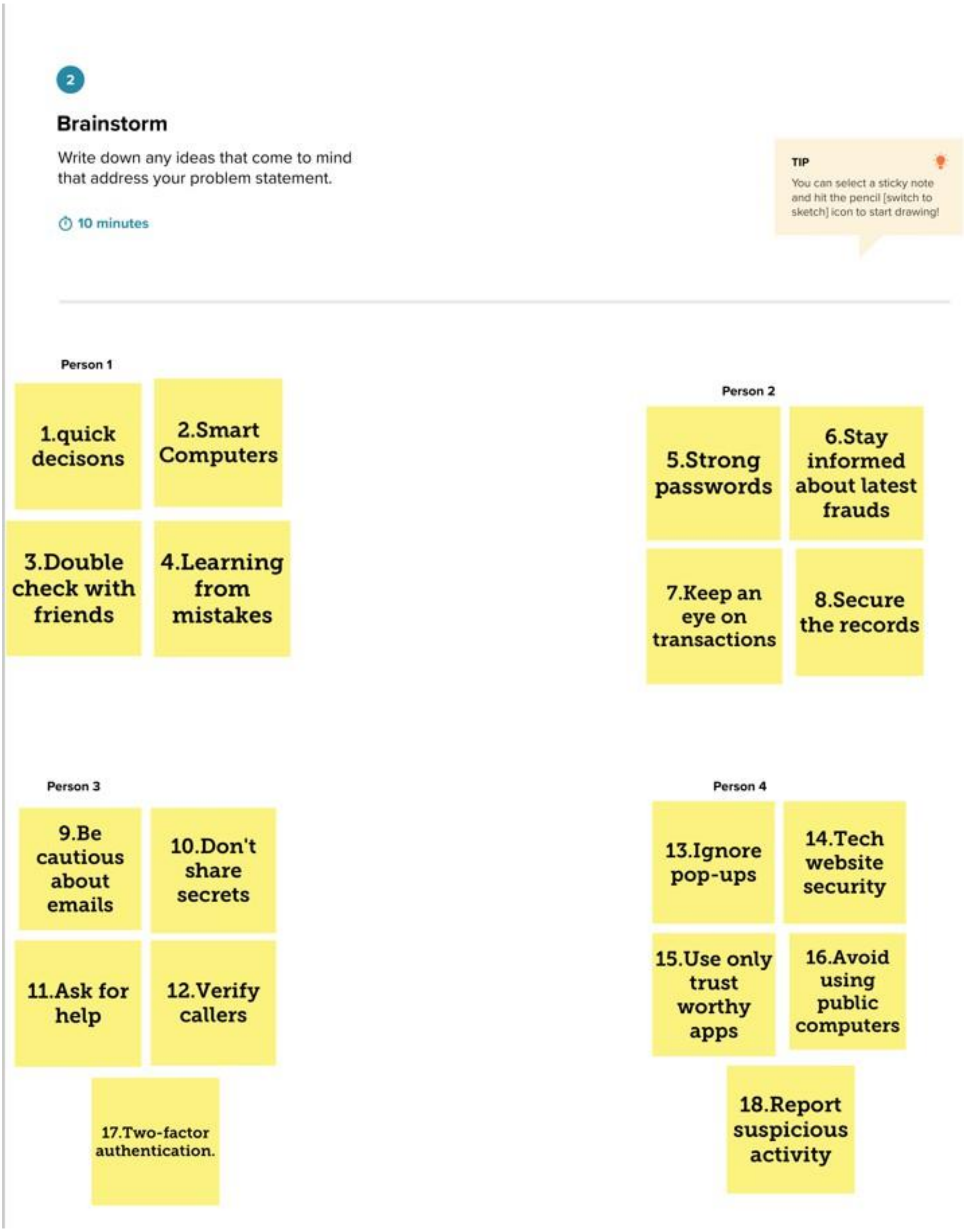
An empathy map is a template that organizes a user's behaviours and feelings to create a sense of empathy between the user and your team.

The empathy map represents a principal user and helps teams understand their motivations, concerns, and experience.

Empathy mapping is a simple yet effective that can be conducted with various users in mind, anywhere from stakeholders, individual use cases, or entire teams of people. Many teams, such as design teams, sales, product development, and customer service, can conduct it.



3.2 Ideation & Brainstorming



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 and 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.

Group 1: Security and Prevention Measures

- Strong passwords
- Two-factor authentication.
- Secure the records
- Stay informed about latest frauds
- Be cautious about emails
- Don't share secrets
- Report suspicious activity

Group 2: User Awareness and Vigilance

- Keep an eye on transactions
- Learning from mistakes
- Verify callers
- Ignore pop-ups
- check website security
- Use only trust worthy apps
- Avoid using public computers

Group 3: Communication and Support

- Double check with friends
- Ask for help
- Smart Computers
- Quick decisons

4

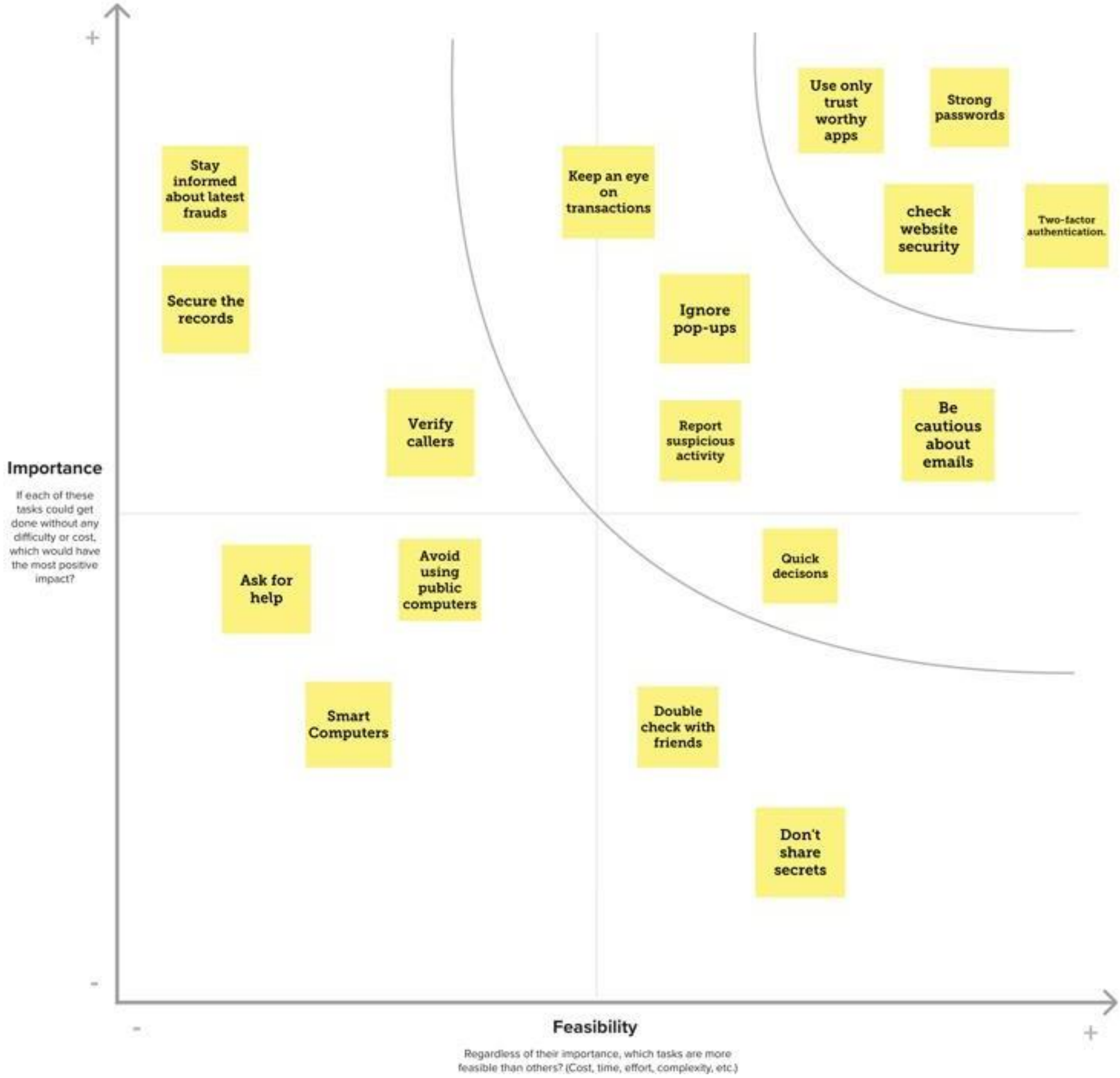
Prioritize

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 requirement

User Interface (Web Application):

1. User Authentication: Users should be able to log in securely with unique credentials.
2. Transaction Input: Provide a user-friendly interface for users to input transaction data for fraud analysis.
3. Real-time Fraud Prediction: Implement a real-time prediction mechanism to analyse transactions promptly.
4. Prediction Results Display: Display the fraud prediction results clearly, indicating the likelihood of fraud for each transaction.
5. User Alerts: Generate alerts for users when potentially fraudulent activities are detected.

Machine Learning Model:

1. Integration of Classification Algorithms: Implement Decision Tree, Random Forest, SVM, Extra Tree Classifier, and XGBoost Classifier for fraud detection.
2. Model Training: Develop a mechanism to train the machine learning model using historical transaction data.
3. Model Evaluation: Implement metrics such as accuracy, precision, recall, and F1 score for evaluating the performance of the trained models.
4. Model Saving and Loading: Enable the system to save the selected machine learning model in a pkl format and load it for real-time predictions.

Deployment and Integration:

1. Flask Integration: Integrate the machine learning model into a Flask web application for seamless interaction.
2. IBM Cloud Deployment: Deploy the web application on the IBM Cloud for scalability and accessibility.

Reporting and Logging:

1. Transaction History: Maintain a log of historical transactions and their fraud predictions for auditing purposes.
2. Performance Metrics Tracking: Implement a system for tracking and recording the performance metrics of the machine learning models over time.

Security:

1. Data Encryption: Ensure the secure transmission and storage of sensitive transaction data through encryption.
2. Access Control: Implement role-based access control to restrict system access based on user roles.

4.2 Non-Functional requirements

Performance:
1. Response Time: The system should provide real-time fraud predictions, with a response time not exceeding [X] seconds for each transaction.
2. Scalability: The system should be scalable to handle a minimum of [X] transactions per second, with the ability to scale based on increasing transaction volumes.

Reliability:
1. Availability: The system should be available 99.9% of the time, allowing for scheduled maintenance windows.
2. Fault Tolerance: The system should be designed to withstand failures gracefully, ensuring minimal impact on user experience.

Compatibility:
1. Browser Compatibility: The web application should be compatible with the latest versions of popular browsers, including Chrome, Firefox, Safari, and Edge.

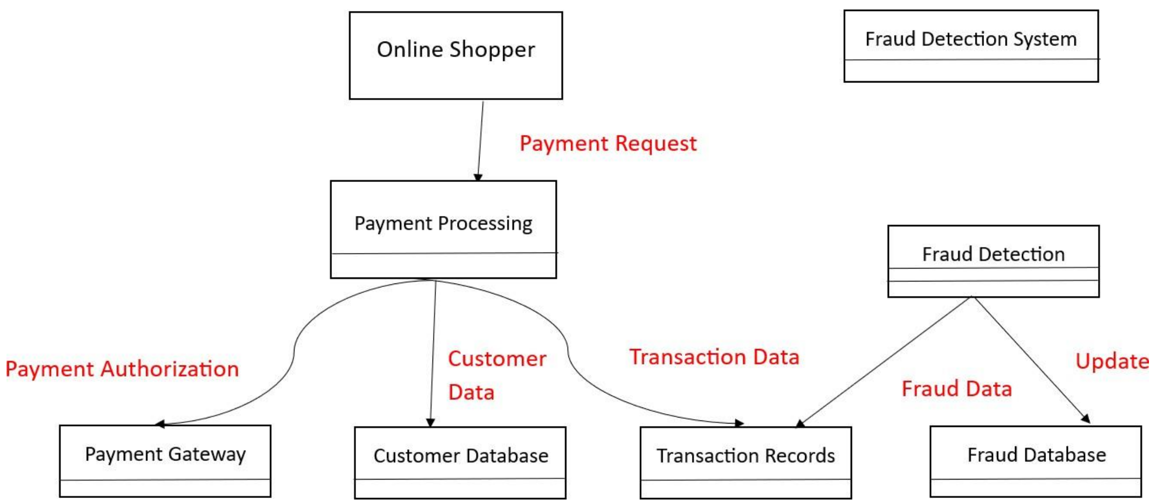
Maintainability:

Code Maintainability: The codebase should be well-documented and follow best practices to facilitate ease of maintenance and future development.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

DFD:



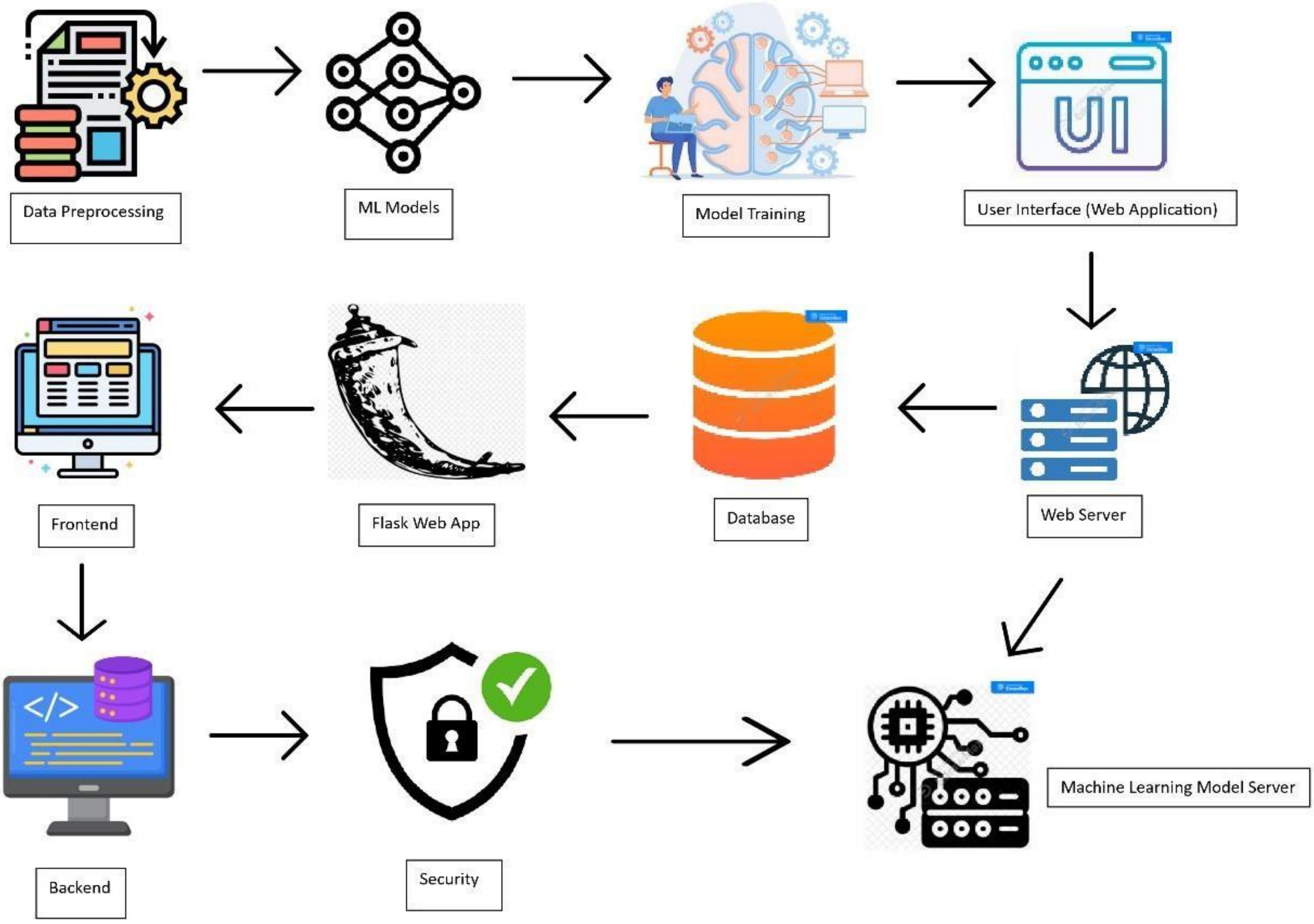
User Stories:

User Story Number	User Story / Task	Story Points	Priority
USN-1	As an API consumer, I want to integrate the Credit Card Fraud Prediction API into my application to automate fraud for a batch of transactions.	1	High
USN-2	As a user, i want my web application to be coherent and accessible.	2	High
USN-3	As an Analyst, I want feature enabling it to recognize complex fraud patterns and adapt to emerging threats.	2	Low
USN-4	As a user, I want to input transaction details into the Credit Card Fraud Detection web application to receive a prediction.	2	High
USN-5	As an API consumer I should receive meaningful error messages in case of issues, facilitating debugging and troubleshooting.	2	High
USN-6	As an Administrator, I want a configurable dashboard to monitor the performance of the fraud detection algorithms, allowing for real-time adjustments and ensuring optimal accuracy.	2	High
USN-7	As a Customer, I want a frictionless user experience, where the system dynamically adjusts security measures based on the risk level of the transaction.	2	Medium
USN-8	As a Compliance, I want comprehensive audit logs and reporting features integrated into the fraud detection system, ensuring transparency and compliance with regulatory requirements.	2	Low
USN-9	As a Software Developer, I want a documented and standardized API for integrating third-party data sources into the fraud detection system, enhancing the system's capabilities with external threat intelligence.	2	Medium

5.2 Solution Architecture

Solution architecture provides the ground for software development projects by tailoring IT solutions to specific business needs and defining their functional requirements and stages of implementation. It is comprised of many subprocesses that draw guidance from various enterprise architecture viewpoints.

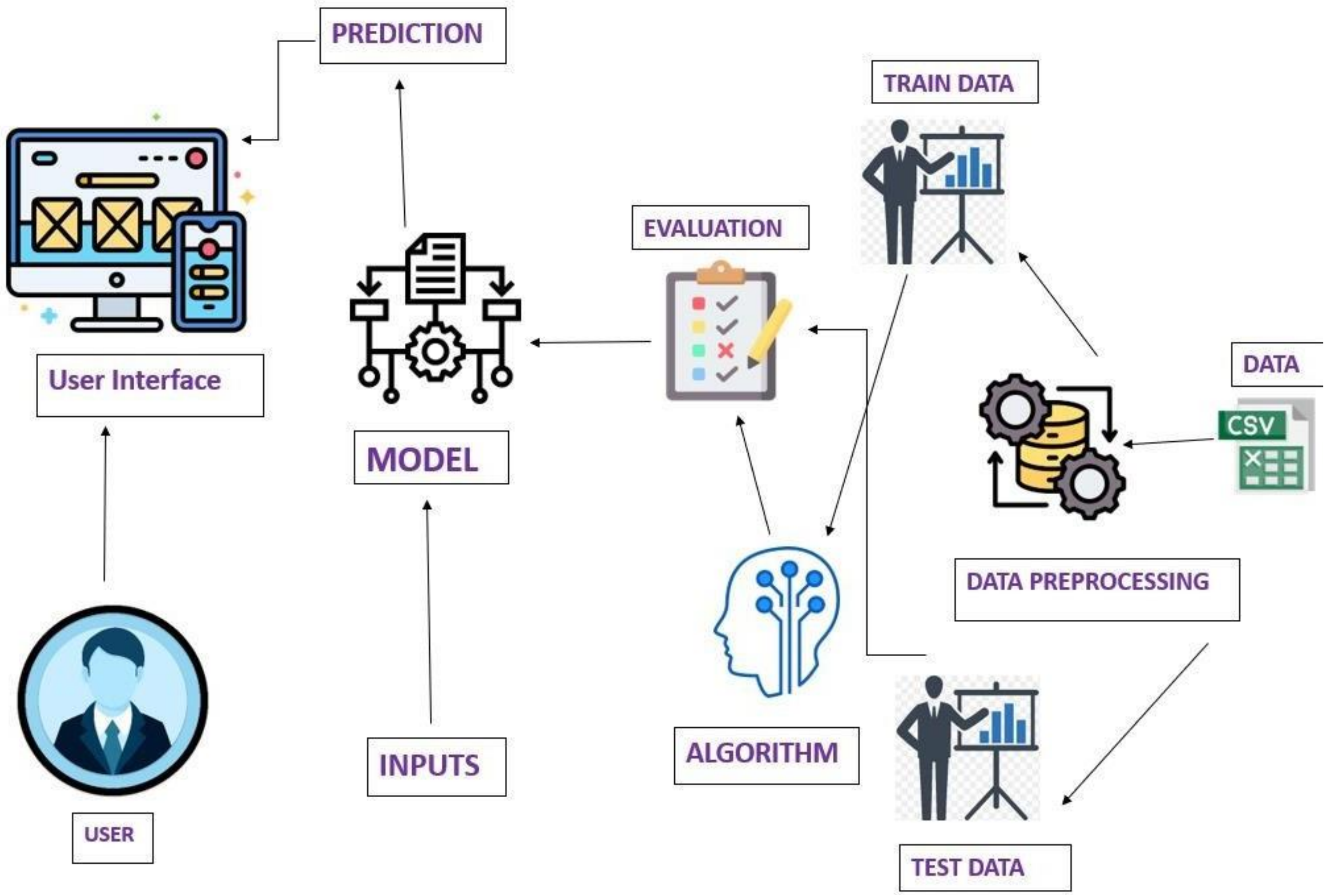
To better understand the role of solution architecture in the context of software development, you first need to think about what a solution is. Even though this might seem quite basic, it illustrates why solution architecture is one of the most important processes when re-designing your IT landscape. At its core, a solution is a way to describe an answer to a problem. In the corporate world, this means evaluating client needs or problems and addressing them with systems that replace or improve the existing system.



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

Technical Architecture (TA) is a form of IT architecture that is used to design computer systems. It involves the development of a technical blueprint with regard to the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met. Technology architecture deals with the deployment of application components on technology components. A standard set of predefined technology components is provided in order to represent servers, network, workstations.


PREDICTION
TRAIN DATA
EVALUATION
User Interface
DATA
MODEL
DATA PREPROCESSING
ALGORITHM
INPUTS
USER
TEST DATA

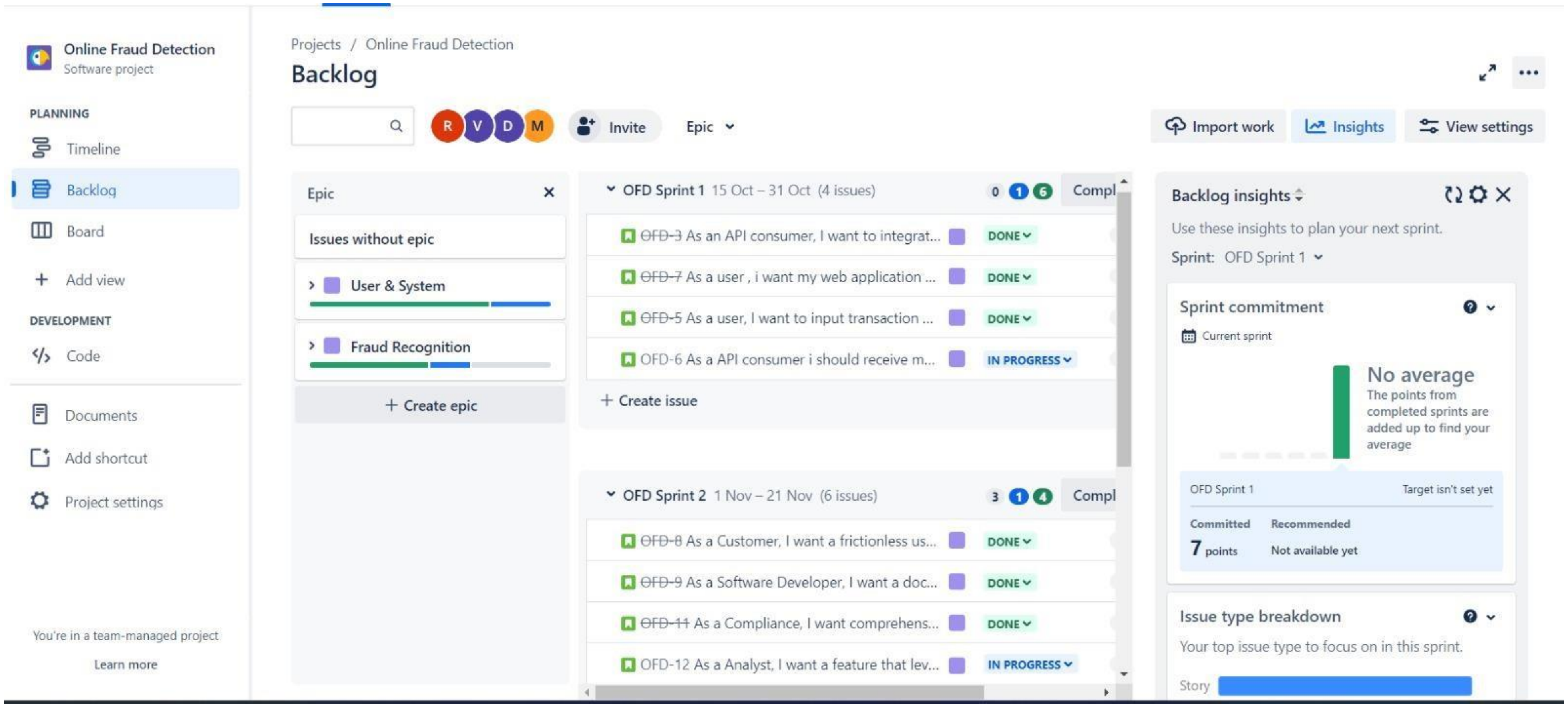
6.2 Sprint Planning & Estimation

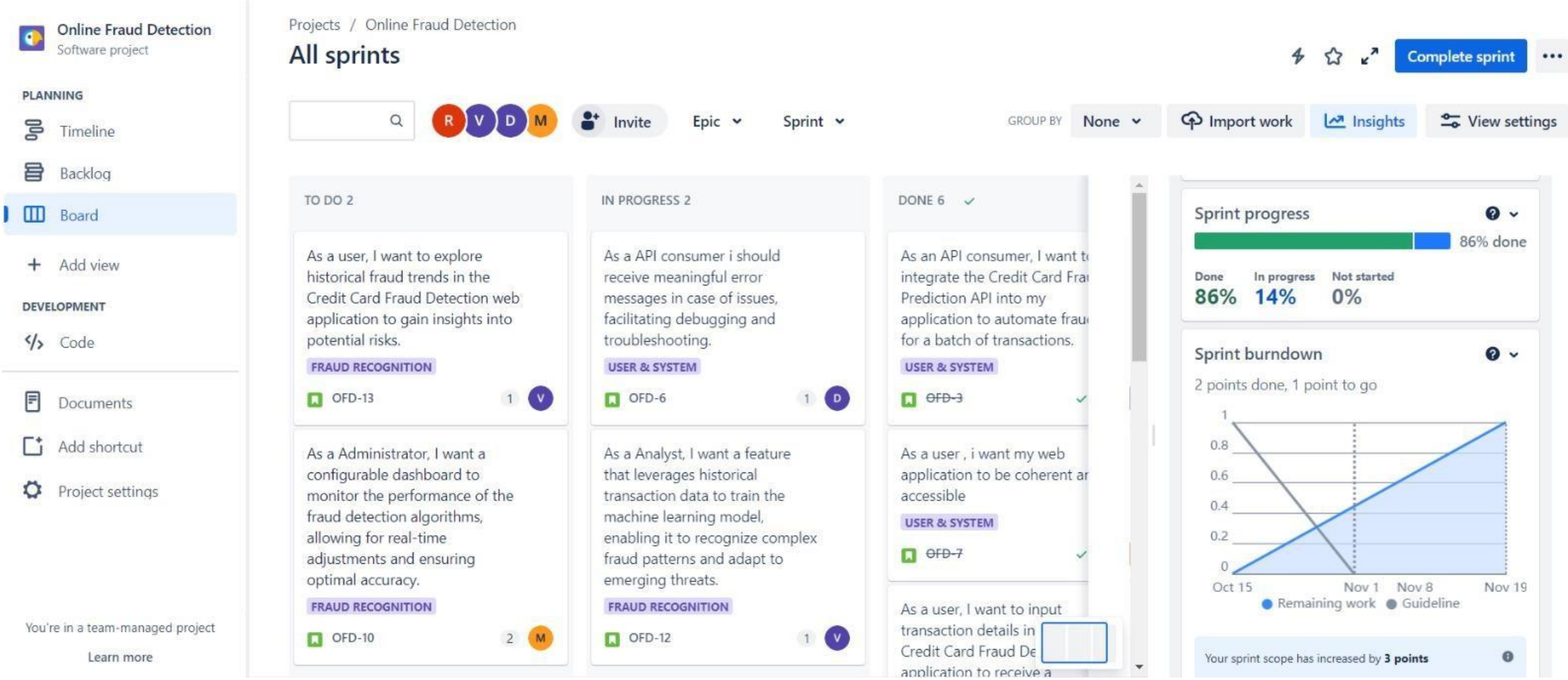
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	user & system	USN-1	As an API consumer, I want to integrate the Credit Card Fraud Prediction API into my application to automate fraud for a batch of transactions.	1	High	Vyshnavi
Sprint-1		USN-2	As a user, i want my web application to be coherent and accessible.	2	High	Rishitha
Sprint-2	Fraud recognition	USN-3	As an Analyst, I want feature enabling it to recognize complex fraud patterns and adapt to emerging threats.	2	Low	Vyshnavi
Sprint-1		USN-4	As a user, I want to input transaction details into the Credit Card Fraud Detection web application to receive a prediction.	2	High	Rishitha
Sprint-1		USN-5	As an API consumer I should receive meaningful error messages in case of issues, facilitating debugging and troubleshooting.	2	High	Gnana Sai
Sprint-2		USN-6	As an Administrator, I want a configurable dashboard to monitor the performance of the fraud detection algorithms, allowing for real-time adjustments and ensuring optimal accuracy.	2	High	Keerthana
Sprint-2		USN-7	As a Customer, I want a frictionless user experience, where the system dynamically adjusts security measures based on the risk level of the transaction.	2	Medium	Gnana Sai
Sprint-2		USN-8	As a Compliance, I want comprehensive audit logs and reporting features integrated into the fraud detection system, ensuring transparency and compliance with regulatory requirements.	2	Low	Keerthana
Sprint-2		USN-9	As a Software Developer, I want a documented and standardized API for integrating third-party data sources into the fraud detection system, enhancing the system's capabilities with external threat intelligence.	2	Medium	Rishitha

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	7	6 Days	15 Oct 2023	31 Oct 2023	17	22 Nov 2023
Sprint-2	10	6 Days	1 Nov 2023	21 Nov 2023		

Burndown Chart:




Online Fraud Detection
Software project
PLANNING
Timeline
Backlog
Board
Add view
DEVELOPMENT
Code
Documents
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Online Fraud Detection
All sprints
Complete sprint
Invite
Epic
Sprint
GROUP BY
None
Import work
Insights
View settings
TO DO 2
IN PROGRESS 2
DONE 6
As a user, I want to explore historical fraud trends in the Credit Card Fraud Detection web application to gain insights into potential risks.
FRAUD RECOGNITION
OFD-13
As a API consumer i should receive meaningful error messages in case of issues, facilitating debugging and troubleshooting.
USER & SYSTEM
OFD-6
As an API consumer, I want to integrate the Credit Card Fra
Prediction API into my application to automate frau
for a batch of transactions.
USER & SYSTEM
OFD-3
As a Administrator, I want a configurable dashboard to monitor the performance of the fraud detection algorithms, allowing for real-time adjustments and ensuring optimal accuracy.
FRAUD RECOGNITION
OFD-10
As a Analyst, I want a feature that leverages historical transaction data to train the machine learning model, enabling it to recognize complex fraud patterns and adapt to emerging threats.
FRAUD RECOGNITION
OFD-12
As a user , i want my web application to be coherent ar
accessible
USER & SYSTEM
OFD-7
As a user, I want to input transaction details in
Credit Card Fraud De
Sprint progress
86% done
Done 86%
In progress 14%
Not started 0%
Sprint burndown
2 points done, 1 point to go
Oct 15
Nov 1
Nov 8
Nov 19
Remaining work
Guideline
Your sprint scope has increased by 3 points

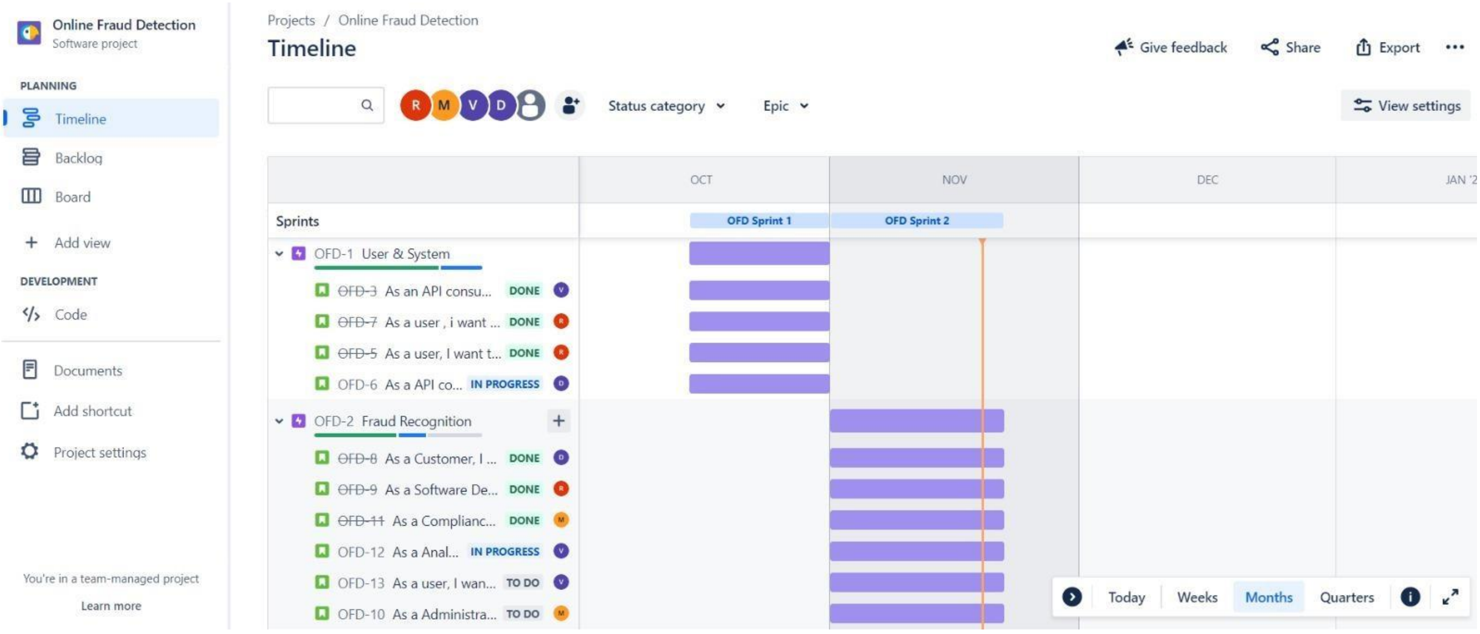
Your sprint scope has increased by **3 points**

Added	Removed	Modified
0 points	0 points	+ **3** points
5 issues	0 issues	5 issues

Epic progress

This sprint is working towards **1 epic**

OFD-1 User & System 85% done



7. CODING & SOLUTIONING

7.1 Feature 1

Supervised Learning:

The code uses a dataset with labeled examples of fraudulent and non-fraudulent transactions.

A Random Forest Classifier is trained on the labeled data, making it a supervised learning approach.

Code:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from imblearn.over_sampling import SMOTE

# Load dataset
data = pd.read_csv('creditcard.csv')

# Feature selection and preprocessing
X = data.drop(['Class'], axis=1)
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Handle class imbalance using SMOTE
oversample = SMOTE()
X_train, y_train = oversample.fit_resample(X_train, y_train)
```

```
# Train Random Forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Predictions
y_pred = clf.predict(X_test)

# Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

7.2 Feature 2

Web development:
The web page serves as UI for interacting with the ML model. The form embedded in the HTML code collects essential input features, such as gender, marital status, age, education, income, occupation, and settlement size, which are crucial for predicting customer segmentation. The styling elements ensure a visually appealing and user-friendly design, with a background image, form-container styling, and a distinctive color scheme. The form utilizes the POST method to send user inputs to the '/predict' endpoint on the server for processing. Upon submitting the form, the user-triggered prediction is processed by the back-end, and the resulting customer segmentation prediction is dynamically displayed in the designated area with the class 'prediction-text'. This integration between the HTML front-end and the back-end prediction mechanism creates a seamless user experience, allowing individuals to input their demographic information and receive real-time predictions regarding their likely segmentation.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="{{ url_for('static', filename='css/indexstyle.css') }}">
<title>ML API</title>
</head>
<body>
<form action="{{ url_for('predict')}}" method="POST">
  <input id="input-1" type="text" placeholder="Enter time" name ="time" required autofocus />
  <label for="input-1">
    <span class="label-text">TIME</span>
    <span class="nav-dot"></span>
    <div class="signup-button-trigger">Credit Card Fraud Prediction</div>
  </label>
  <input id="input-2" type="text" placeholder="Enter Amount" name="amount" required />
  <label for="input-2">
    <span class="label-text">AMOUNT</span>
    <span class="nav-dot"></span>
  </label>
  <input id="input-3" type="text" placeholder="Enter Transaction Method" name="tm" required />
  <label for="input-3">
    <span class="label-text">Transaction Method</span>
    <span class="nav-dot"></span>
  </label>
```

```
  <input id="input-4" type="text" placeholder="Transaction id" name="ti" required />
  <label for="input-4">
    <span class="label-text">Transaction id</span>
    <span class="nav-dot"></span>
  </label>
  <input id="input-5" type="text" placeholder="Enter Type Of card" name="ct" required />
  <label for="input-5">
    <span class="label-text">Card Type</span>
    <span class="nav-dot"></span>
  </label>
  <input id="input-6" type="text" placeholder="Enter Location" name="location" required />
  <label for="input-6">
    <span class="label-text">Enter Location</span>
    <span class="nav-dot"></span>
  </label>
  <input id="input-7" type="text" placeholder="Enter Bank" name="em" required />
  <label for="input-7">
    <span class="label-text">Enter Bank</span>
    <span class="nav-dot"></span>
  </label>
  <button type="submit">Predict</button>
  <p class="tip">Press Tab</p>
  <div class="signup-button">Credit card Fraud Detection</div>
</form>
</body>
</html>
```

Result.html

```
<!Doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/resultstyle.css') }}">

</head>

<!--Hey! This is the original version
of Simple CSS Waves-->
<body>
<div class="header">

<!--Content before waves-->
<div class="inner-header flex">
<!--Just the logo.. Don't mind this-->
<h1>{{ prediction }}</h1>
</div>

<!--Waves Container-->
<div>
<svg class="waves" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
viewBox="0 24 150 28" preserveAspectRatio="none" shape-rendering="auto">
<defs>
<path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-18 58 18 88 18 v44h-352z" />
</defs>
```

```
<g class="parallax">
<use xlink:href="#gentle-wave" x="48" y="0" fill="#bb1515" />
<use xlink:href="#gentle-wave" x="48" y="3" fill="#bb1515" />
<use xlink:href="#gentle-wave" x="48" y="5" fill="rgba(255,255,255,0.3)" />
<use xlink:href="#gentle-wave" x="48" y="7" fill="#bb1515" />
</g>
</svg>
</div>
<!--Waves end-->

</div>
<!--Header ends-->

<!--Content starts-->
<div class="content flex">
</div>
<!--Content ends-->
</body>
```

8. PERFORMANCE TESTING

8.1 Performance Metrics

```
In [41]: Y=d_scaled['Class']

In [42]: new_data=pd.concat([X_reduced,Y],axis=1)
         new_data.head()
         new_data.shape

Out[42]: (2492, 8)

In [43]: X_train, X_test, y_train, y_test= train_test_split(X_reduced, d_scaled['Class'], test_size = 0.30, random_state = 42)

         X_train.shape, X_test.shape

Out[43]: ((1744, 7), (748, 7))

In [44]: from sklearn.linear_model import LogisticRegression
         lr=LogisticRegression()
         lr.fit(X_train,y_train)
         y_pred_lr=lr.predict(X_test)
         y_pred_lr

Out[44]: array([0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 1., 1., 0.,
                0., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 1., 0.,
                0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
                0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 1.,
                0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                1., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
                0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 0., 1.,
                0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
                0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
```

In [45]:
```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred_lr))
```
```
[[601   2]
 [ 15 130]]
```

In [46]:
```
from sklearn.model_selection import GridSearchCV
lr_model = LogisticRegression()
lr_params = {'penalty': ['l1', 'l2'],'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
grid_lr= GridSearchCV(lr_model, param_grid = lr_params)
grid_lr.fit(X_train, y_train)

grid_lr.best_params_
```

Accuracy Score:

Out[46]:
```
{'C': 10, 'penalty': 'l2'}
```

In [47]:
```
y_pred_lr3=grid_lr.predict(X_test)
print(classification_report(y_test,y_pred_lr3))
```

```
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99       603
         1.0       0.98      0.90      0.94       145

    accuracy                           0.98       748
   macro avg       0.98      0.95      0.96       748
weighted avg       0.98      0.98      0.98       748
```

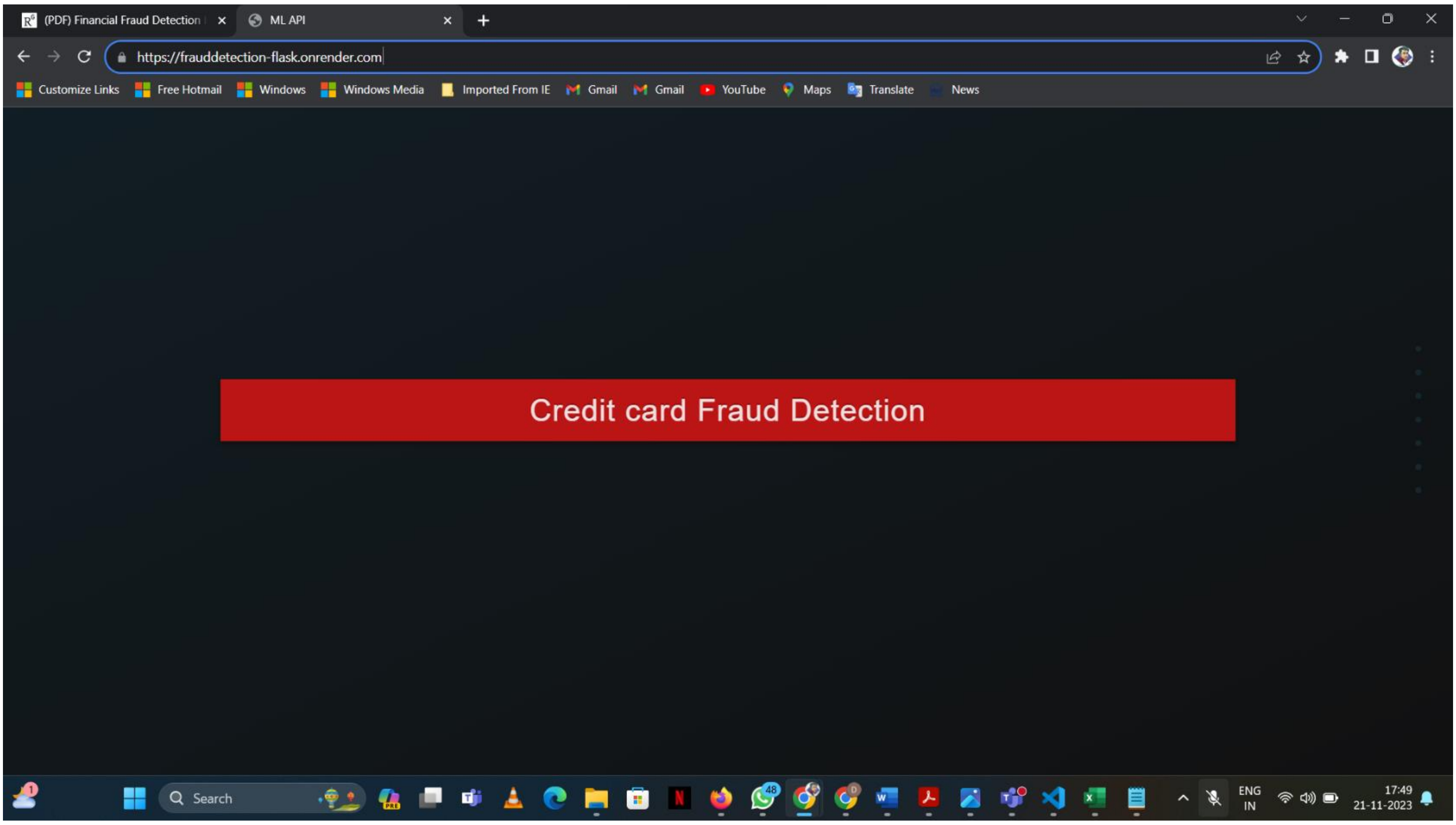
Classification Report:

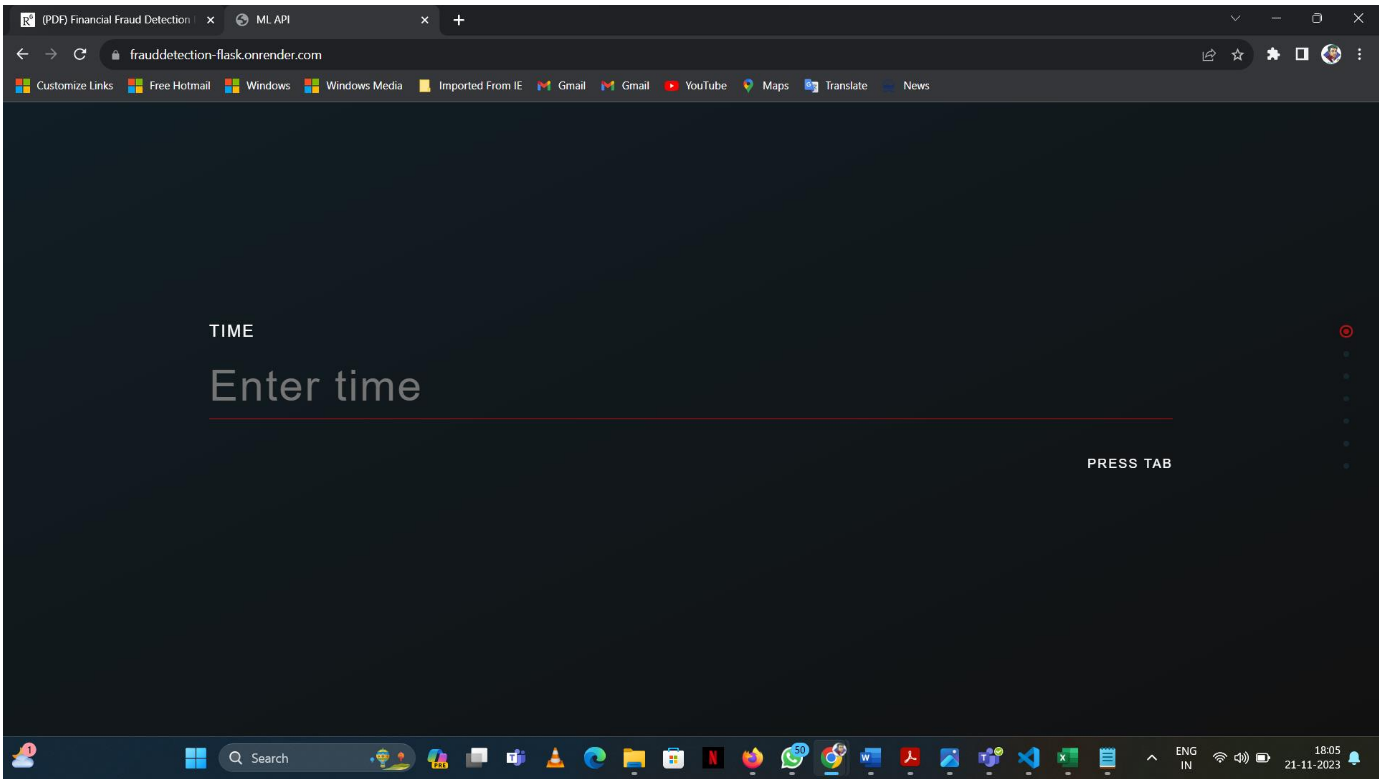
```
0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 1., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 1., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1.])
```

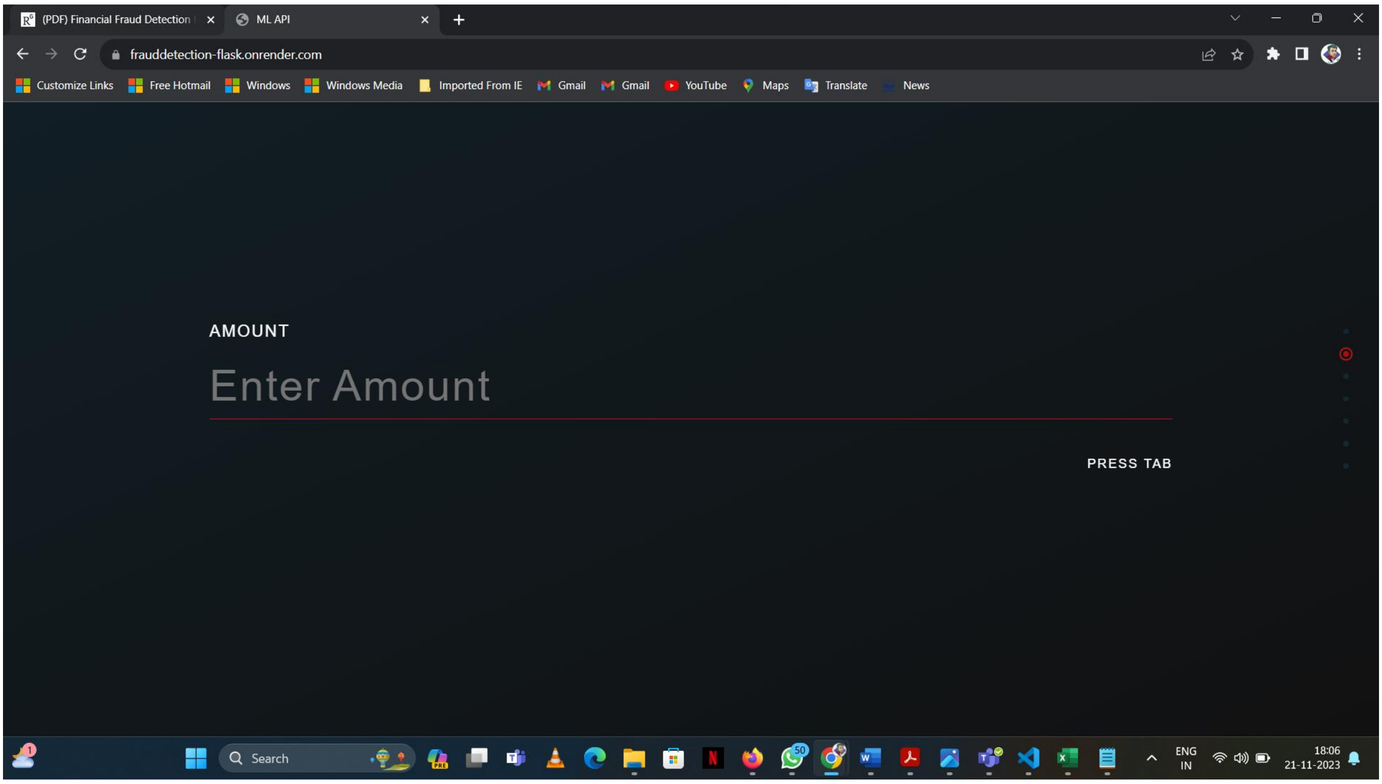
[80]:
```
type(X_test)
X_test.to_csv('testing.csv')
from sklearn.model_selection import GridSearchCV
parameters = [ {'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': [0.1, 1, 0.01, 0.0001 ,0.001]}
grid_search = GridSearchCV(estimator = svc,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           n_jobs = -1)
grid_search = grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print("Best Accuracy: {:.2f} %".format(best_accuracy*100))
print("Best Parameters:", best_parameters)

svc_param=SVC(kernel='rbf',gamma=0.01,C=100,probability=True)
svc_param.fit(X_train,y_train)
```
```
Best Accuracy: 97.13 %
Best Parameters: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
```

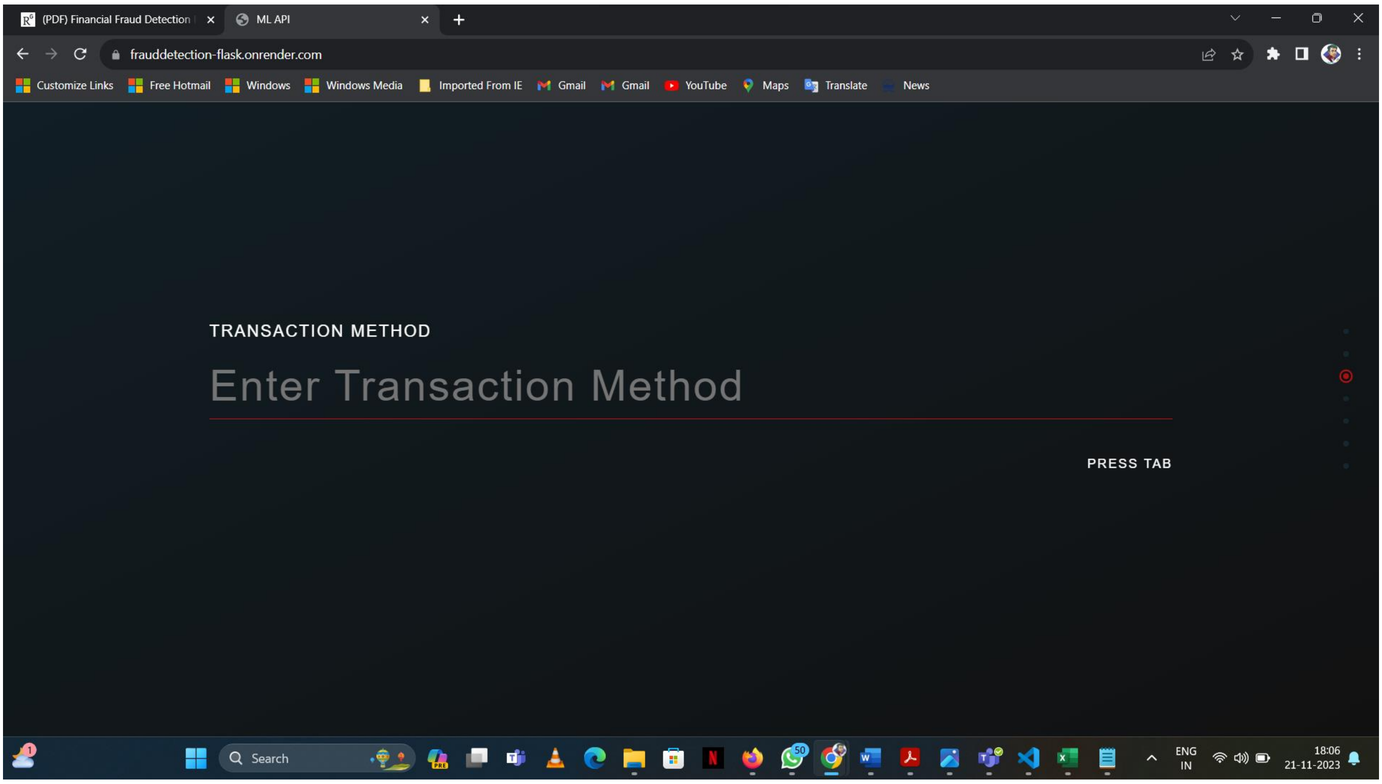
t[80]:
```
SVC
SVC(C=100, gamma=0.01, probability=True)
```

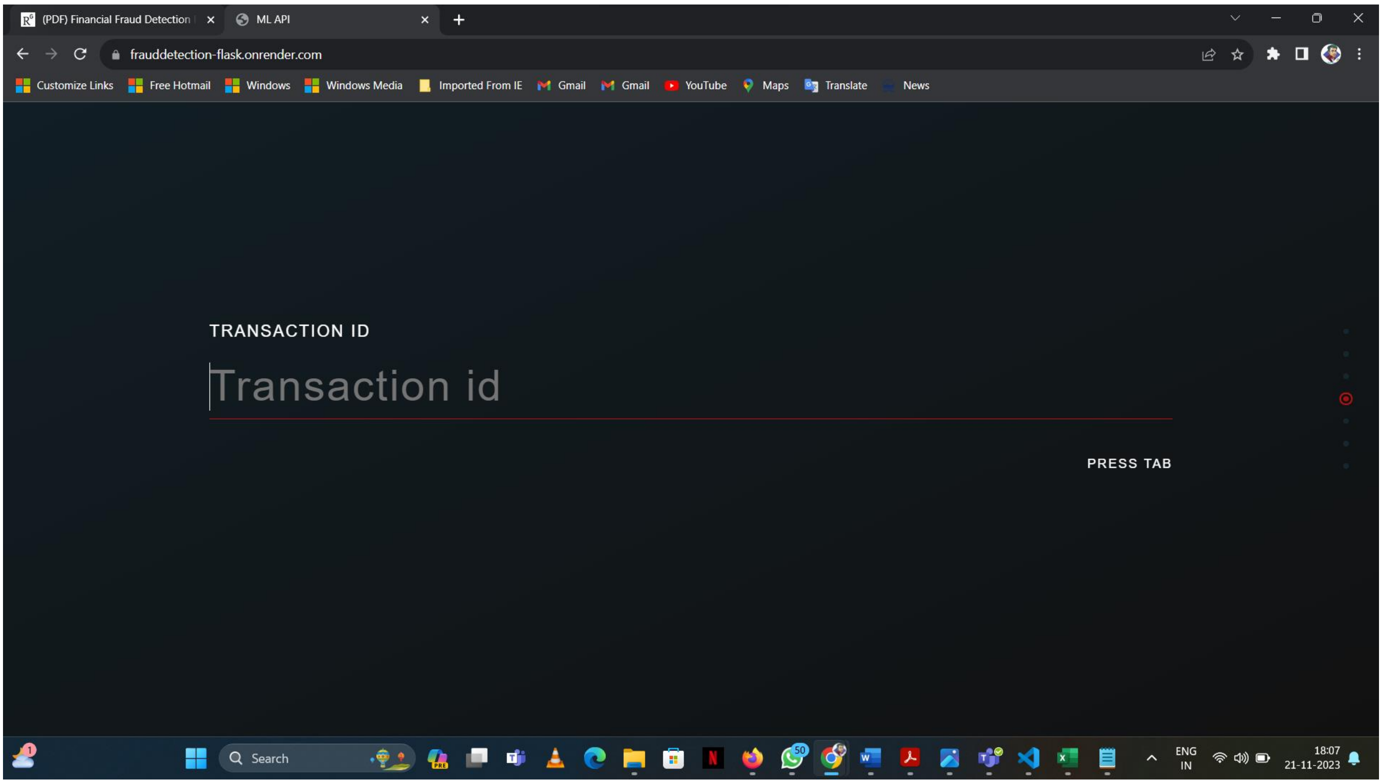
9. RESULTS
9.1 Output Screenshots

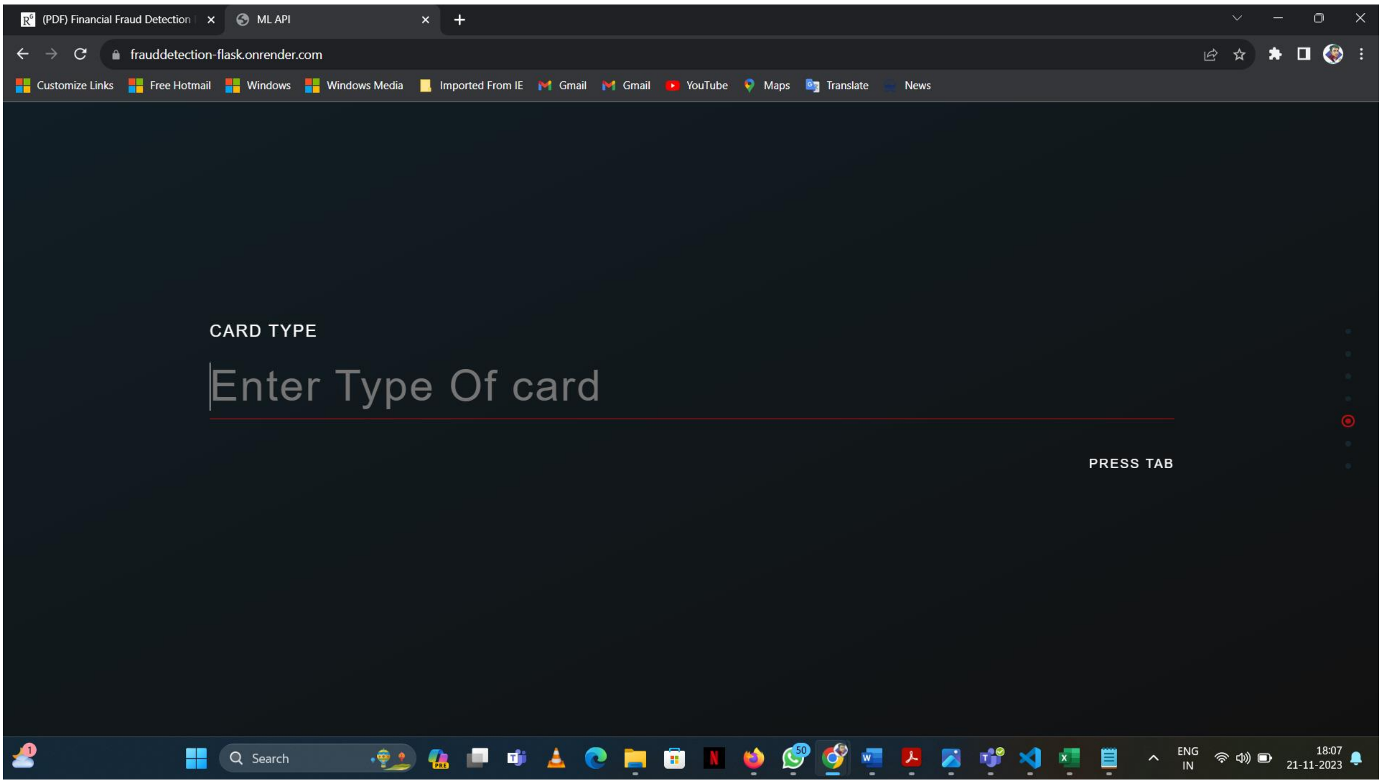


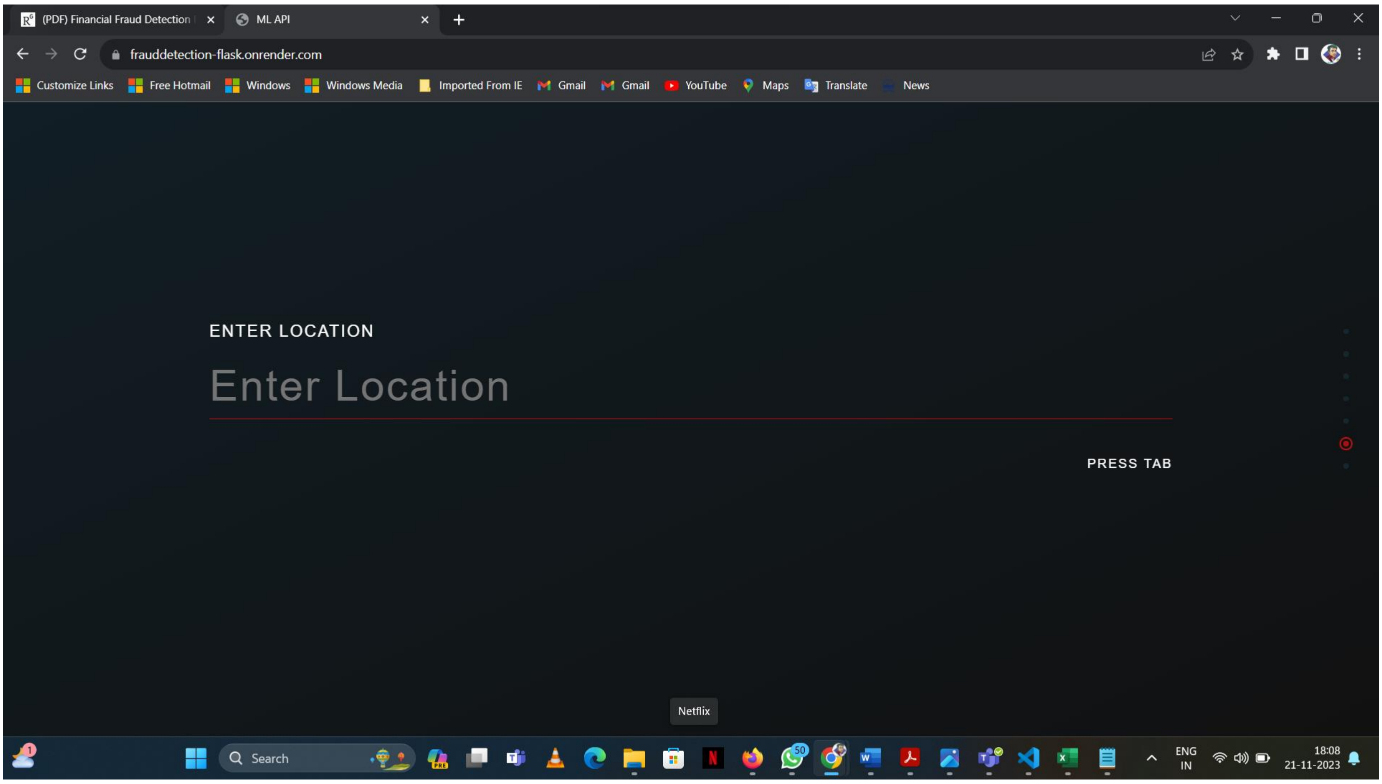


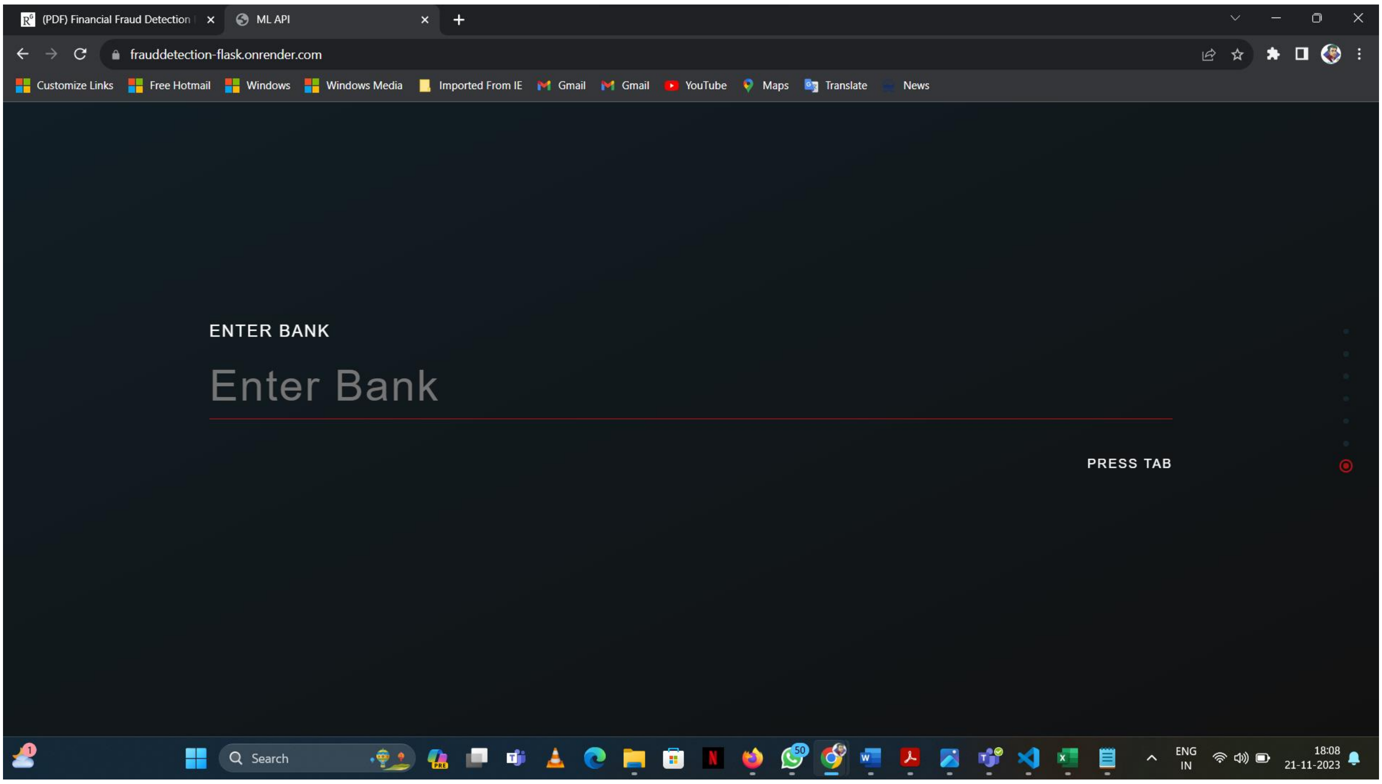

(PDF) Financial Fraud Detection
ML API
frauddetection-flask.onrender.com
Customize Links
Free Hotmail
Windows
Windows Media
Imported From IE
Gmail
Gmail
YouTube
Maps
Translate
News
AMOUNT
Enter Amount
PRESS TAB
Search
ENG
IN
18:06
21-11-2023

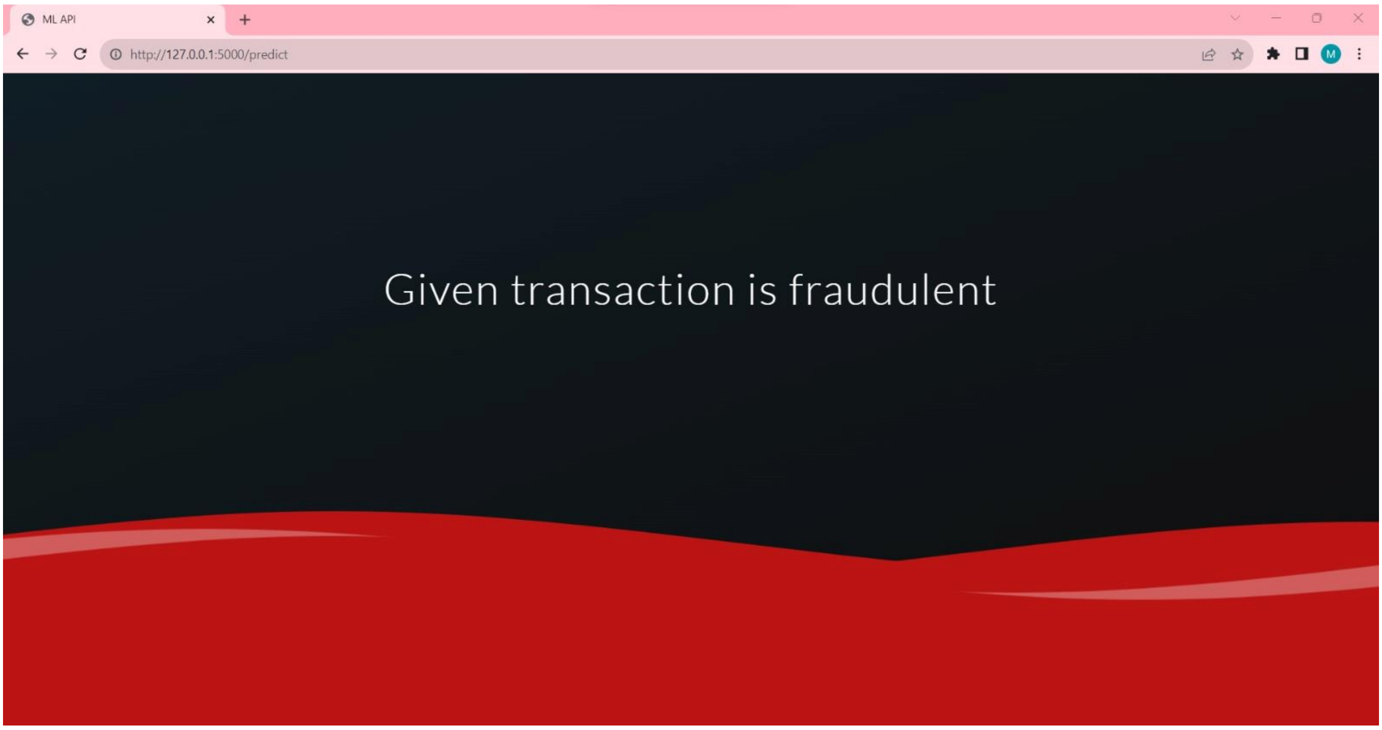

(PDF) Financial Fraud Detection
ML API
frauddetection-flask.onrender.com
Customize Links
Free Hotmail
Windows
Windows Media
Imported From IE
Gmail
Gmail
YouTube
Maps
Translate
News
TRANSACTION METHOD
Enter Transaction Method
PRESS TAB
Search
ENG
IN
18:06
21-11-2023


(PDF) Financial Fraud Detection
ML API
frauddetection-flask.onrender.com
TRANSACTION ID
Transaction id
PRESS TAB
18:07
21-11-2023


(PDF) Financial Fraud Detection
ML API
frauddetection-flask.onrender.com
CARD TYPE
Enter Type Of card
PRESS TAB
18:07
21-11-2023


(PDF) Financial Fraud Detection
ML API
frauddetection-flask.onrender.com
Customize Links
Free Hotmail
Windows
Windows Media
Imported From IE
Gmail
Gmail
YouTube
Maps
Translate
News
ENTER LOCATION
Enter Location
PRESS TAB
Netflix
Search
ENG
IN
18:08
21-11-2023


(PDF) Financial Fraud Detection
ML API
frauddetection-flask.onrender.com
Customize Links
Free Hotmail
Windows
Windows Media
Imported From IE
Gmail
Gmail
YouTube
Maps
Translate
News
ENTER BANK
Enter Bank
PRESS TAB
Search
ENG
IN
18:08
21-11-2023



10. ADVANTAGES & DISADVANTAGES

Advantages of Online Payments Fraud Detection using Machine Learning:

1. Improved Accuracy: Machine learning algorithms can analyze large datasets and detect complex patterns, leading to more accurate fraud predictions compared to traditional methods.

2.Real-time Detection: The incorporation of machine learning enables real-time fraud detection, allowing for immediate action upon identifying suspicious transactions.

3. Adaptability to Emerging Threats: Machine learning models can adapt to evolving fraud patterns, providing a dynamic and proactive approach to fraud prevention.

4. Efficiency in Handling Large Datasets: The project addresses the challenge of efficiently processing large volumes of transaction data, a common issue in online payments fraud detection.

5. User-Friendly Web Application: The web application enhances user experience by providing a user-friendly interface for inputting transaction data and receiving real-time fraud predictions.

Disadvantages:

1. Model Interpretability: Complex machine learning models, such as XGBoost, may lack interpretability, making it challenging to understand and explain the rationale behind specific fraud predictions.

2. Data Imbalance: Imbalances in the dataset, with a smaller number of fraudulent transactions, can impact the model's ability to generalize to real-world scenarios and lead to biased predictions.

3. Resource Intensive: Training and retraining machine learning models can be resource-intensive, requiring significant computational power and storage.

4. Overfitting Risk: There is a risk of overfitting, especially when dealing with intricate fraud patterns, which may result in a model that performs well on training data but poorly on new, unseen data.

11. CONCLUSION

In the rapidly evolving landscape of online transactions, the project on "Online Payments Fraud Detection using Machine Learning" stands as a pivotal initiative addressing the escalating challenges posed by fraudulent activities. The project's focus on leveraging advanced machine learning algorithms, including Decision Tree, Random Forest, SVM, Extra Tree Classifier, and XGBoost Classifier, brings forth a promising solution to enhance the accuracy and efficiency of fraud detection.

The advantages of this project are multi-faceted. By adopting machine learning, the system achieves improved accuracy in predicting fraudulent transactions, providing a real-time shield against emerging threats. The adaptability of the models to dynamic fraud patterns represents a significant stride toward proactive fraud prevention. The implementation of a user-friendly web application, coupled with integration into the Flask framework and deployment on the IBM Cloud, ensures accessibility, scalability, and a seamless user experience. In conclusion, the "Online Payments Fraud Detection using Machine Learning" project not only responds to the immediate needs of users and financial institutions but also lays the groundwork for a more secure and resilient online payment environment. As the project unfolds, ongoing efforts in monitoring, adapting to evolving threats, and user feedback will contribute to the refinement and optimization of the system, ensuring its relevance and efficacy in the ever-changing landscape of online transactions. The journey embarked upon in this project aligns with the broader mission of enhancing trust, security, and integrity in the realm of digital financial transactions.

12. FUTURE SCOPE

The future scope of the "Online Payments Fraud Detection using Machine Learning" project is promising and multifaceted. Beyond its current capabilities, the project can evolve by incorporating advanced machine learning techniques, enhancing model interpretability, and exploring real-time behavioral analysis. Continuous model evaluation, collaboration with financial institutions, and global standardization efforts will contribute to sustained effectiveness. Additionally, there is potential for user education, mobile application integration, and cross-industry applications. By staying at the forefront of technology, embracing collaboration, and adapting to emerging trends, the project can further fortify the security of online transactions, ensuring a resilient and dynamic fraud detection system for the future.

13. APPENDIX

Importing libraries and dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import random
from sklearn.utils import shuffle
d=pd.read_csv('creditcard.csv')
```

```
In [5]: d
```

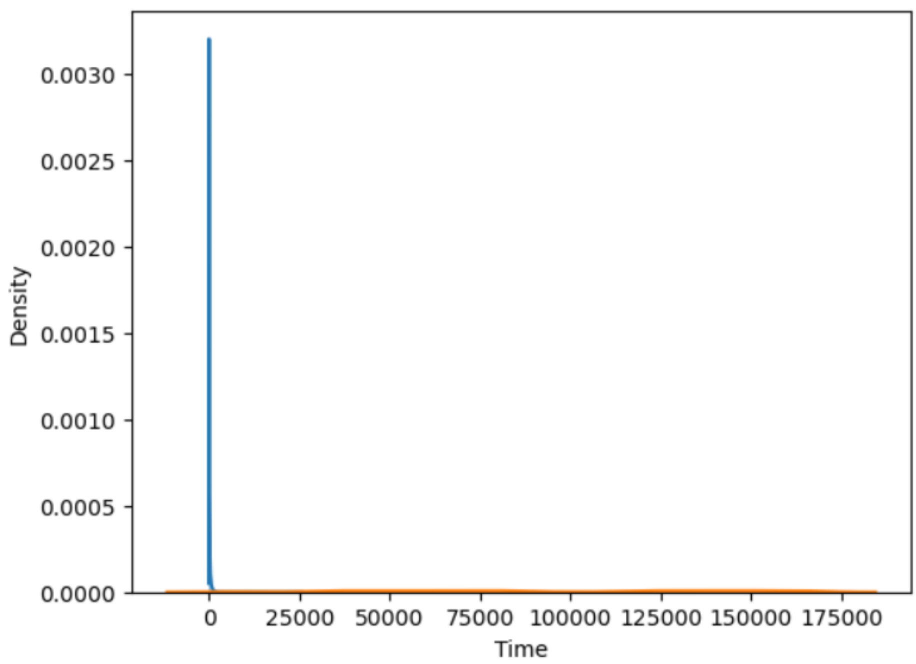
Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.06692
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.33984
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.68928
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.17557
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.14126
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.50934
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.01622
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.64013
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.12320
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.00879

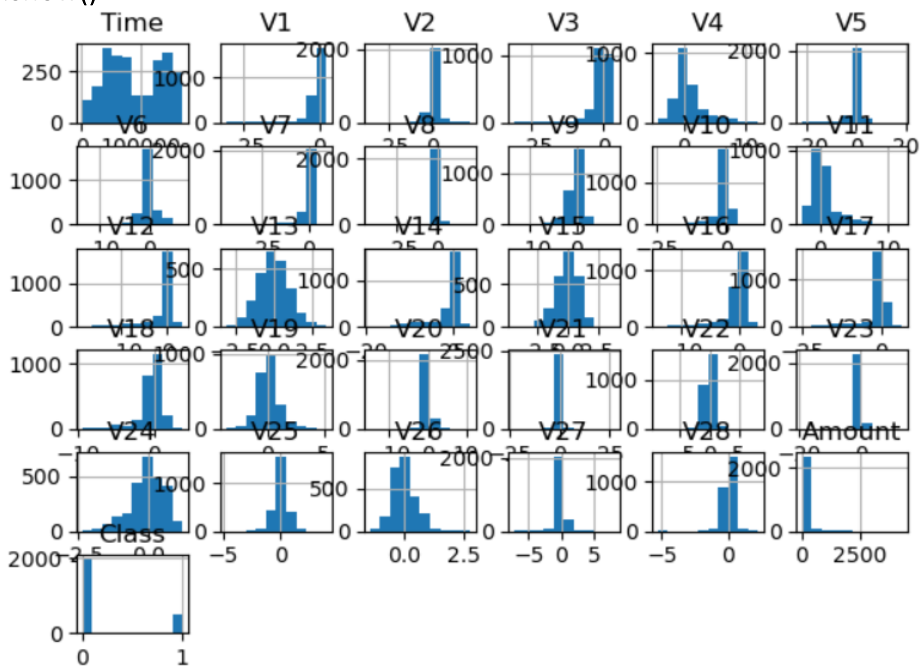
284807 rows × 31 columns

```
sns.distplot(d['Amount'])

sns.distplot(d['Time'])
```

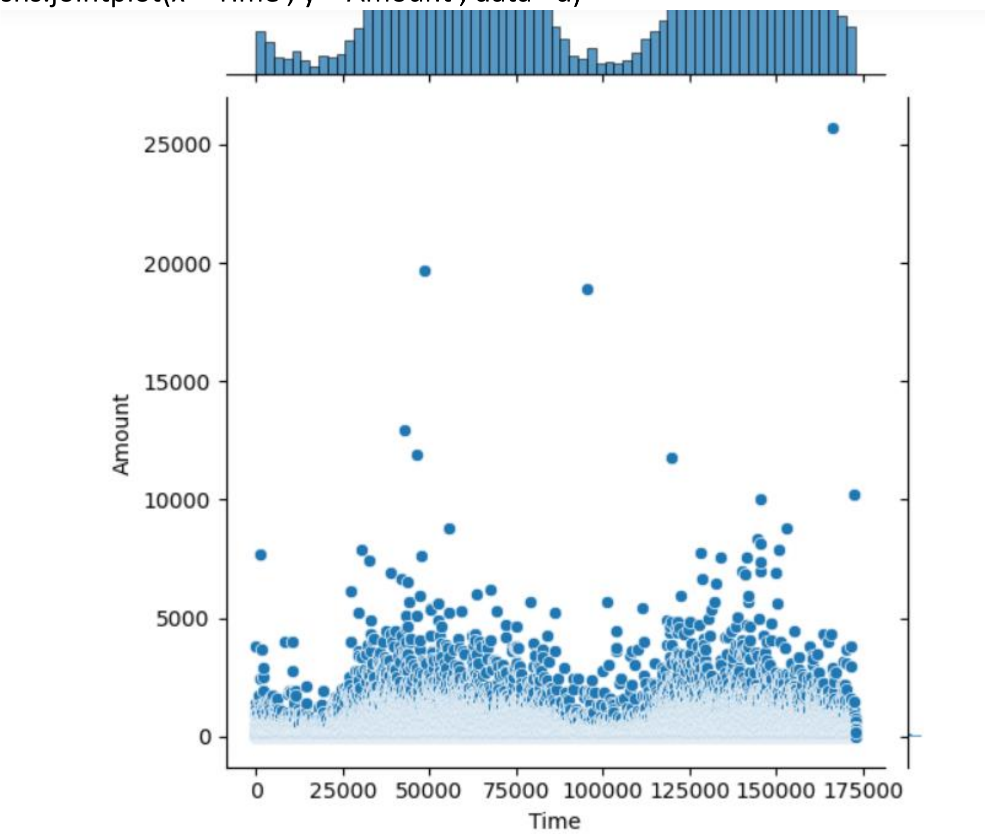
```
Out[6]: <Axes: xlabel='Time', ylabel='Density'>
```



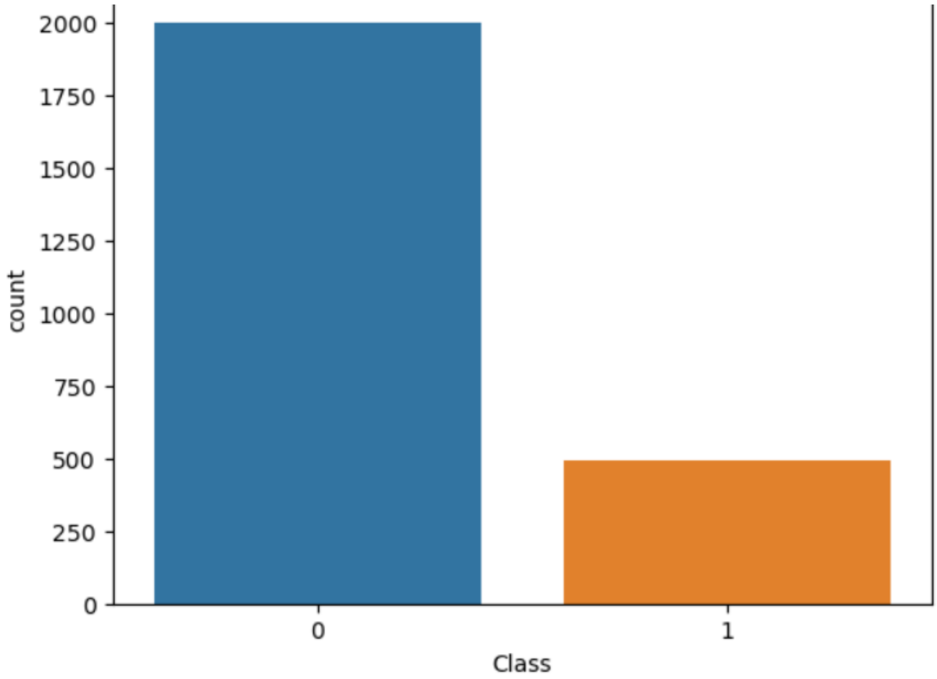
```
data.hist(figsize=(7,5))
plt.show()
```




```
sns.jointplot(x= 'Time', y= 'Amount', data= d)
```




```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'Class' is a column in your DataFrame 'df'
sns.countplot(x='Class', data=df)

# Display the plot
plt.show()
```



X=df.iloc[: ,:-1]
Y=df.iloc[: , -1]
X=pd.DataFrame(X)
X.shape---→ (2492 , 30)
Y=pd.DataFrame(Y)
Y.head()

Out[19]:

	Class
139216	0
276244	0
153639	0
109541	0
215531	0

data.describe()

Out[23]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
count	2492.000000	2492.000000	2492.000000	2492.000000	2492.000000	2492.000000	2492.000000	2492.000000	2492.000000	2492.000000	...	2492.000000
mean	91236.181782	-0.946977	0.727373	-1.383817	0.860228	-0.630729	-0.251022	-1.083121	0.074222	-0.474762	...	0.130843
std	47733.179149	3.971649	2.854086	4.409868	2.548462	2.917241	1.559170	4.018396	3.271067	1.813345	...	1.891749
min	74.000000	-30.821436	-35.616754	-31.103685	-4.345575	-22.105532	-13.360241	-43.557242	-41.044261	-13.434066	...	-22.797604
25%	50435.750000	-1.402012	-0.451572	-1.682002	-0.660915	-0.968541	-1.010898	-0.914349	-0.212450	-1.062483	...	-0.217592
50%	81924.000000	-0.298502	0.275900	-0.238059	0.278382	-0.194207	-0.395484	-0.087376	0.054171	-0.194030	...	0.020252
75%	137127.750000	1.217464	1.210635	0.799702	1.401317	0.537905	0.299020	0.488527	0.487872	0.491878	...	0.299808
max	172734.000000	2.349591	22.057729	3.664946	12.114672	18.611287	6.474115	9.303732	20.007208	7.929051	...	27.202839

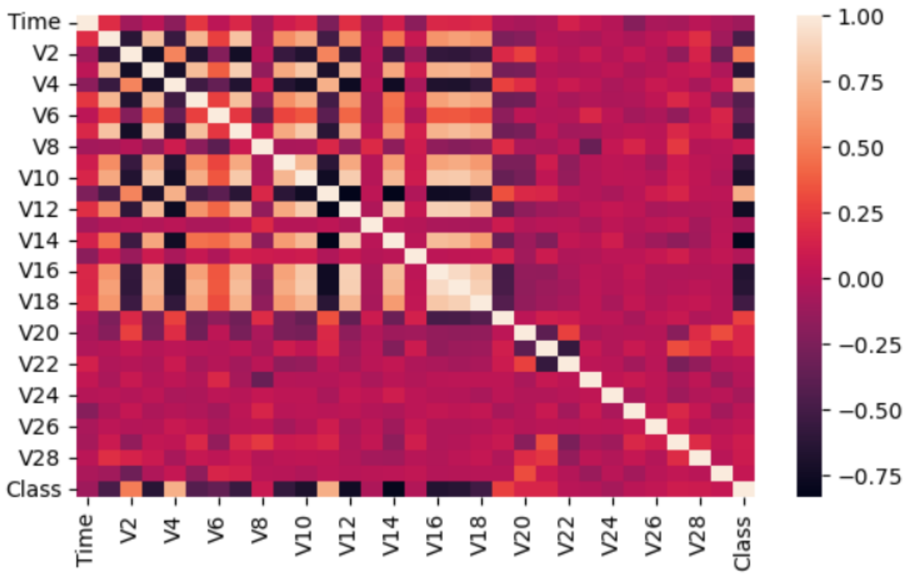
8 rows × 31 columns

data.info()

```
 1   V1      2492 non-null   float64
 2   V2      2492 non-null   float64
 3   V3      2492 non-null   float64
 4   V4      2492 non-null   float64
 5   V5      2492 non-null   float64
 6   V6      2492 non-null   float64
 7   V7      2492 non-null   float64
 8   V8      2492 non-null   float64
 9   V9      2492 non-null   float64
 10  V10     2492 non-null   float64
 11  V11     2492 non-null   float64
 12  V12     2492 non-null   float64
 13  V13     2492 non-null   float64
 14  V14     2492 non-null   float64
 15  V15     2492 non-null   float64
 16  V16     2492 non-null   float64
 17  V17     2492 non-null   float64
 18  V18     2492 non-null   float64
 19  V19     2492 non-null   float64
 20  V20     2492 non-null   float64
 21  V21     2492 non-null   float64
 22  V22     2492 non-null   float64
 23  V23     2492 non-null   float64
 24  V24     2492 non-null   float64
 25  V25     2492 non-null   float64
 26  V26     2492 non-null   float64
 27  V27     2492 non-null   float64
 28  V28     2492 non-null   float64
 29  Amount  2492 non-null   float64
 30  Class   2492 non-null   float64
dtypes: float64(31)
memory usage: 603.7 KB
```

plt.figure(figsize=(7,4))
sns.heatmap(data.corr())

```
Out[25]: <Axes: >
```



import math
import sklearn.preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score , classification_report, confusion_matrix, precision_recall_curve, f1_score, auc
X_train, X_test, y_train, y_test = train_test_split(data.drop('Class', axis=1), data['Class'], test_size=0.3, random_state=42)

Feature Scaling

cols= ['V22', 'V24', 'V25', 'V26', 'V27', 'V28']

scaler = StandardScaler()

frames= ['Time', 'Amount']

x= data[frames]
d_temp = data.drop(frames, axis=1)
temp_col=scaler.fit_transform(x)
scaled_col = pd.DataFrame(temp_col, columns=frames)

scaled_col.head()

Out[32]:

	Time	Amount
0	-0.171219	-0.385323
1	1.587073	-0.374888
2	0.177539	0.260394
3	-0.415081	-0.365256
4	1.024144	0.169170

d_scaled = pd.concat([scaled_col, d_temp], axis =1)

d_scaled.head()

Out[33]:

	Time	Amount	V1	V2	V3	V4	V5	V6	V7	V8	...	V20	V21	V22	V23
0	-0.171219	-0.385323	1.157457	0.260843	0.383889	0.574981	-0.228116	-0.395403	-0.093067	0.071054	...	-0.111829	-0.195816	-0.568083	0.172627
1	1.587073	-0.374888	-3.028582	3.234577	-3.143391	-0.756203	-1.067800	-0.591526	-1.369818	2.654259	...	-0.033207	0.488533	1.102256	0.231338
2	0.177539	0.260394	0.202219	1.077172	-1.276218	1.737891	0.890508	-1.015050	1.884316	-0.552673	...	0.331340	-0.202064	-0.154461	0.122605
3	-0.415081	-0.365256	1.273667	0.348576	0.155665	0.610139	-0.179551	-0.860330	0.100726	-0.195347	...	-0.045382	-0.305978	-0.878975	0.078359
4	1.024144	0.169170	2.041205	-1.675773	-1.346314	-1.530424	-1.056192	-0.568938	-0.701911	-0.212398	...	-0.188928	-0.306894	-0.664945	0.177506

"""# Dimensionality Reduction"""

from sklearn.decomposition import PCA

pca = PCA(n_components=7)

X_temp_reduced = pca.fit_transform(d_scaled)

pca.explained_variance_ratio_

pca.explained_variance_

```
Out[38]: array([106.69768306,  14.50512346,  10.74676857,   5.46473049,
                  4.76769311,   4.00649798,   2.61482255])
```

names=['Time','Amount','Transaction Method','Transaction Id','Location','Type of Card','Bank']

X_reduced= pd.DataFrame(X_temp_reduced,columns=names)
X_reduced.head()
names=['Time','Amount','Transaction Method','Transaction Id','Location','Type of Card','Bank']
X_reduced= pd.DataFrame(X_temp_reduced,columns=names)
X_reduced.head()

Out[40]:

	Time	Amount	Transaction Method	Transaction Id	Location	Type of Card	Bank
0	-3.860818	-0.188792	-0.051130	0.168795	-0.439317	-0.705464	0.137812
1	-1.390245	2.979890	5.342280	1.269209	0.731550	-2.011317	-0.304936
2	-3.048058	-0.643302	-1.072993	3.540518	-0.562522	0.381300	-0.249209
3	-3.876920	-0.160903	-0.087560	0.092488	-0.475221	-0.832521	0.140371
4	-4.049599	-0.005896	0.336496	-0.423122	-0.655424	-0.591550	2.789950

```
Y=d_scaled['Class']

new_data=pd.concat([X_reduced,Y],axis=1)
new_data.head()
new_data.shape

new_data.to_csv('finaldata.csv')

X_train, X_test, y_train, y_test= train_test_split(X_reduced, d_scaled['Class'], test_size = 0.30,
random_state = 42)

X_train.shape, X_test.shape
```

Out[42]: (2492, 8)

In [43]:
```
X_train, X_test, y_train, y_test= train_test_s

X_train.shape, X_test.shape
```

Out[43]: ((1744, 7), (748, 7))

In [45]:
```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred_lr))
```

```
[[601   2]
 [ 15 130]]
```

```
#Hyperparamter tuning
from sklearn.model_selection import GridSearchCV
lr_model = LogisticRegression()
lr_params = {'penalty': ['l1', 'l2'],'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
grid_lr= GridSearchCV(lr_model, param_grid = lr_params)
grid_lr.fit(X_train, y_train)

grid_lr.best_params_

y_pred_lr3=grid_lr.predict(X_test)
print(classification_report(y_test,y_pred_lr3))
```

```
Out[46]: {'C': 10, 'penalty': 'l2'}
```

```
In [47]: y_pred_lr3=grid_lr.predict(X_test)
         print(classification_report(y_test,y_pred_lr3))
```

```
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99       603
         1.0       0.98      0.90      0.94       145

    accuracy                           0.98       748
   macro avg       0.98      0.95      0.96       748
weighted avg       0.98      0.98      0.98       748
```

```
from sklearn.svm import SVC
svc=SVC(kernel='rbf')
svc.fit(X_train,y_train)
y_pred_svc=svc.predict(X_test)
y_pred_svc
print(classification_report(y_test,y_pred_svc))
```

```
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.98       603
         1.0       0.98      0.88      0.93       145

    accuracy                           0.97       748
   macro avg       0.97      0.94      0.96       748
weighted avg       0.97      0.97      0.97       748
```

```
from sklearn.model_selection import GridSearchCV
parameters = [ {'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': [0.1, 1, 0.01, 0.0001 ,0.001]}]
grid_search = GridSearchCV(estimator = svc,
                param_grid = parameters,
                scoring = 'accuracy',
                n_jobs = -1)
grid_search = grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print("Best Accuracy: {:.2f} %".format(best_accuracy*100))
print("Best Parameters:", best_parameters)

svc_param=SVC(kernel='rbf',gamma=0.01,C=100)
svc_param.fit(X_train,y_train)
y_pred_svc2=svc_param.predict(X_test)
print(classification_report(y_test,y_pred_svc2))
```

```
Best Accuracy: 97.13 %
Best Parameters: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98       603
         1.0       0.96      0.89      0.92       145

    accuracy                           0.97       748
   macro avg       0.97      0.94      0.95       748
weighted avg       0.97      0.97      0.97       748
```

```
print(confusion_matrix(y_test,y_pred_dtree))
d_tree_param=DecisionTreeClassifier()
tree_parameters={'criterion':['gini','entropy'],'max_depth':list(range(2,4,1)),
            'min_samples_leaf':list(range(5,7,1))}
grid_tree=GridSearchCV(d_tree_param,tree_parameters)
grid_tree.fit(X_train,y_train)

y_pred_dtree2=grid_tree.predict(X_test)

print(classification_report(y_test,y_pred_dtree2))


"""# Decision Tree"""

from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
y_pred_dtree=dtree.predict(X_test)
print(classification_report(y_test,y_pred_dtree))

print(confusion_matrix(y_test,y_pred_dtree))

d_tree_param=DecisionTreeClassifier()
tree_parameters={'criterion':['gini','entropy'],'max_depth':list(range(2,4,1)),
            'min_samples_leaf':list(range(5,7,1))}
grid_tree=GridSearchCV(d_tree_param,tree_parameters)
grid_tree.fit(X_train,y_train)

y_pred_dtree2=grid_tree.predict(X_test)

print(classification_report(y_test,y_pred_dtree2))



"""# Random Forest"""

from sklearn.ensemble import RandomForestClassifier
randomforest=RandomForestClassifier(n_estimators=5)
randomforest.fit(X_train,y_train)
y_pred_rf=randomforest.predict(X_test)
print(confusion_matrix(y_test,y_pred_rf))

print(classification_report(y_test,y_pred_rf))



"""# K Nearest Neighbors"""

from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
```

```
y_pred_knn=knn.predict(X_test)
y_pred_knn

print(classification_report(y_test,y_pred_knn))

print(confusion_matrix(y_test,y_pred_knn))

knn_param=KNeighborsClassifier()
knn_params={"n_neighbors": list(range(2,5,1)), 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']}
grid_knn=GridSearchCV(knn_param,param_grid=knn_params)
grid_knn.fit(X_train,y_train)
grid_knn.best_params_

knn = KNeighborsClassifier(n_neighbors=2)

knn.fit(X_train,y_train)
pred_knn2 = knn.predict(X_test)

print('WITH K=3')
print('\n')
print(confusion_matrix(y_test,pred_knn2))
print('\n')
print(classification_report(y_test,pred_knn2))
```

```
              precision    recall  f1-score   support

         0.0       0.98      0.99      0.99       603
         1.0       0.97      0.91      0.94       145

    accuracy                           0.98       748
   macro avg       0.97      0.95      0.96       748
weighted avg       0.98      0.98      0.98       748
```

```
[[593  10]
 [ 14 131]]
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98       603
         1.0       0.93      0.90      0.92       145

    accuracy                           0.97       748
   macro avg       0.95      0.94      0.95       748
weighted avg       0.97      0.97      0.97       748
```

```
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98       603
         1.0       0.96      0.88      0.92       145

    accuracy                           0.97       748
   macro avg       0.96      0.94      0.95       748
weighted avg       0.97      0.97      0.97       748
```

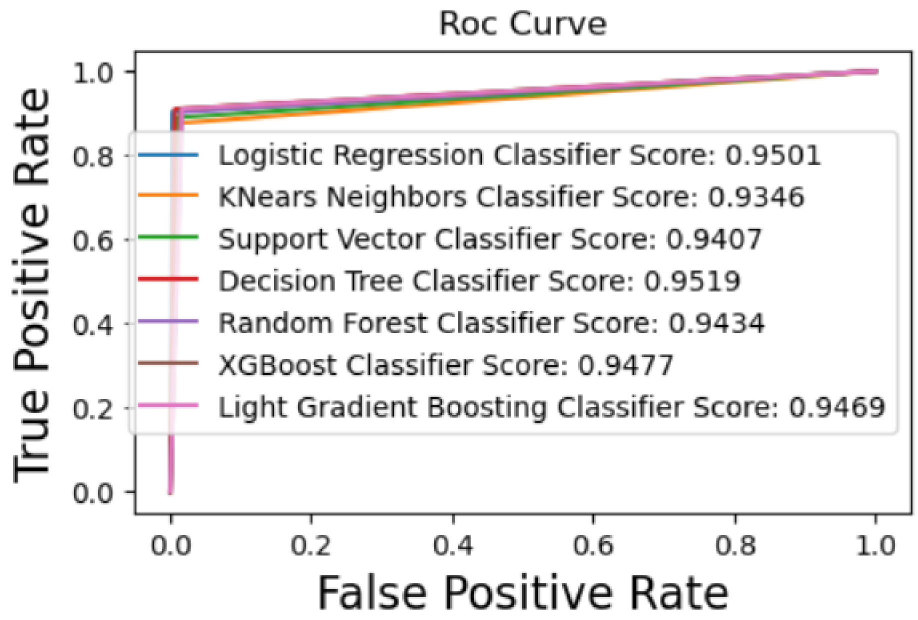
```
: print(confusion_matrix(y_test,y_pred_knn))
```

```
[[597   6]
 [ 17 128]]
```

```
"""# XGBoost"""

from xgboost import XGBClassifier
xgb=XGBClassifier()
xgb.fit(X_train,y_train)
y_pred_xg=xgb.predict(X_test)
print(classification_report(y_test,y_pred_xg))
              precision    recall  f1-score   support

         0.0       0.98      0.99      0.98       603
         1.0       0.94      0.91      0.92       145

    accuracy                           0.97       748
   macro avg       0.96      0.95      0.95       748
weighted avg       0.97      0.97      0.97       748

"""# LGB"""

import lightgbm as lgb

lgb_train = lgb.Dataset(X_train, y_train, free_raw_data= False)

lgb_test = lgb.Dataset(X_test, y_test, reference=lgb_train, free_raw_data= False)

parameters = {'num_leaves': 2**8,
        'learning_rate': 0.1,
        'is_unbalance': True,
        'min_split_gain': 0.1,
        'min_child_weight': 1,
        'reg_lambda': 1,
        'subsample': 1,
        'objective':'binary',
        #'device': 'gpu', # comment this line if you are not using GPU
        'task': 'train'
        }
num_rounds = 300

lgb_train = lgb.Dataset(X_train, y_train)

lgb_test = lgb.Dataset(X_test, y_test)

clf = lgb.train(parameters, lgb_train, num_boost_round=num_rounds)

y_prob = clf.predict(X_test)
y_pred = sklearn.preprocessing.binarize(np.reshape(y_prob, (-1,1)), threshold= 0.5)

accuracy_score(y_test, y_pred)

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

```
[LightGBM] [Warning] Found whitespace in feature_names, replace with underlines
[LightGBM] [Info] Number of positive: 347, number of negative: 1397
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000669 seconds
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1785
[LightGBM] [Info] Number of data points in the train set: 1744, number of used features: 7
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.198968 -> initscore=-1.392758
[LightGBM] [Info] Start training from score -1.392758
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
"""# ROC"""

from sklearn.metrics import roc_curve,roc_auc_score
lg_fpr,lg_tpr,lg_threshold=roc_curve(y_test,y_pred_lr3)
svc_fpr,svc_tpr,svc_threshold=roc_curve(y_test,y_pred_svc2)
dtree_fpr,dtree_tpr,dtree_threshold=roc_curve(y_test,y_pred_dtree2)
rf_fpr,rf_tpr,rf_threshold=roc_curve(y_test,y_pred_rf)
knn_fpr,knn_tpr,rf_threshold=roc_curve(y_test,pred_knn2)
xg_fpr,xg_tpr,xg_threshold=roc_curve(y_test,y_pred_xg)
lgb_fpr,lgb_tpr,lgb_threshold=roc_curve(y_test,y_pred)

plt.figure(figsize=(15,10))
plt.title("Roc Curve")
plt.plot(lg_fpr,lg_tpr, label='Logistic Regression Classifier Score: {:.4f}'.format(roc_auc_score(y_test,
y_pred_lr3)))
plt.plot(knn_fpr,knn_tpr, label='KNears Neighbors Classifier Score: {:.4f}'.format(roc_auc_score(y_test,
pred_knn2)))
plt.plot(svc_fpr, svc_tpr, label='Support Vector Classifier Score: {:.4f}'.format(roc_auc_score(y_test,
y_pred_svc2)))
plt.plot(dtree_fpr, dtree_tpr, label='Decision Tree Classifier Score: {:.4f}'.format(roc_auc_score(y_test,
y_pred_dtree2)))
plt.plot(rf_fpr,rf_tpr, label='Random Forest Classifier Score: {:.4f}'.format(roc_auc_score(y_test,
y_pred_rf)))
plt.plot(xg_fpr,xg_tpr, label='XGBoost Classifier Score: {:.4f}'.format(roc_auc_score(y_test, y_pred_xg)))
plt.plot(lgb_fpr,lgb_tpr, label='Light Gradient Boosting Classifier Score:
{:.4f}'.format(roc_auc_score(y_test, y_pred)))
plt.xlabel('False Positive Rate', fontsize=16)
plt.ylabel('True Positive Rate', fontsize=16)
plt.legend()
plt.show()
```



Github Link:

Github Link:
https://github.com/smartinternz02/SI-GuidedProject-611930-1699173347

Output (Website) Link:
https://frauddetection-flask.onrender.com/