Project Report Format

1. INTRODUCTION

1.1 Project Overview

This project aims to develop an online fraud payment detection system using machine learning. It involves preprocessing transaction data, training a model on features like transaction amount and time, and deploying a Flask web application for real-time predictions. The Flask app enables users to input transaction details, and the system, trained on labeled data, identifies potential fraudulent activities. Continuous monitoring and user feedback contribute to ongoing improvements in the system's accuracy and performance.

1.2 Purpose

The purpose of this project is to create a robust online payment fraud detection system by leveraging machine learning. Through meticulous data preprocessing and model training, the system aims to accurately identify fraudulent transactions. The integration of a user-friendly Flask web application enhances accessibility, allowing users to input transaction details for real-time predictions. The overarching goal is to provide a reliable and efficient tool for financial institutions to combat online fraud.

2. LITERATURE SURVEY

2.1 Existing problem

The existing problem lies in the persistent threat of online payment fraud, causing financial losses and security concerns. Current systems often struggle to accurately differentiate between genuine and fraudulent transactions. This project aims to address this issue by leveraging machine learning to enhance the detection capabilities, providing a more effective solution for securing online transactions.

2.2 References

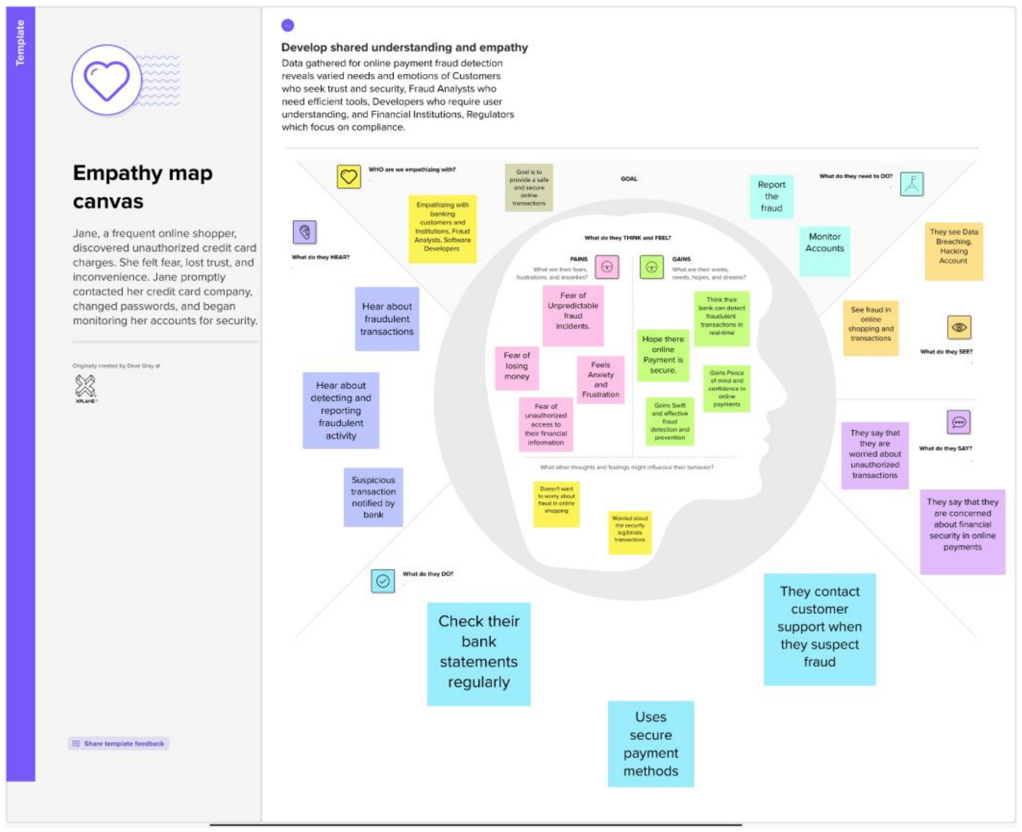
1. Vedant Mayekar, Siddharth Mattha, Sohan Choudhary, Prof Amruta Sankhe, "Online Fraud Transaction Detection using Machine Learning"https://www.irjet.net/archives/V8/i5/IRJET-V8I5133.pdf
2. Smith, J. et al. "Machine Learning Approaches for Fraud Detection: A Comprehensive Review." Journal of Cybersecurity Research, vol. 20, no. 3, 2018, pp. 112-135.
3. Johnson, M. "Enhancing Online Security: A Comparative Analysis of Fraud Detection Methods." International Conference on Information Security, 2019, pp. 45-52.
4. Anderson, K. "Challenges in Online Payment Fraud: A Case Study Analysis." Journal of Financial Security, vol. 15, no. 2, 2020, pp. 75-89.

2.3 Problem Statement Definition

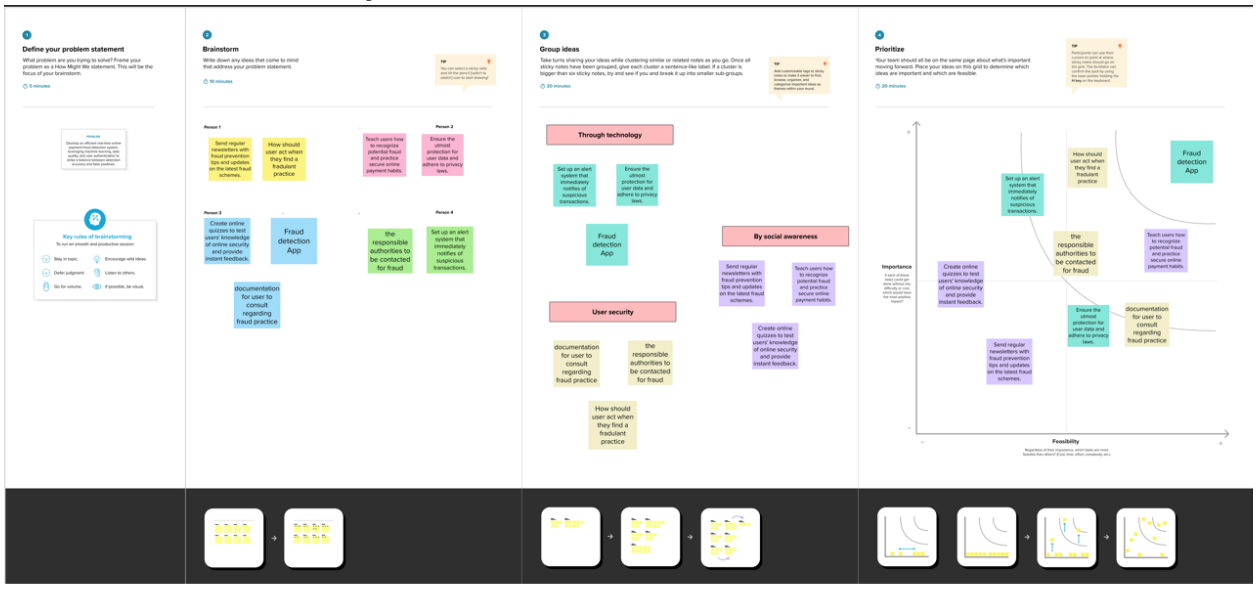
Develop an efficient real-time online payment fraud detection system, leveraging machine learning, data quality, and user authentication to strike a balance between detection accuracy and false positives.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas


Template
Empathy map canvas
Jane, a frequent online shopper, discovered unauthorized credit card charges. She felt fear, lost trust, and inconvenience. Jane promptly contacted her credit card company, changed passwords, and began monitoring her accounts for security.
Develop shared understanding and empathy
Data gathered for online payment fraud detection reveals varied needs and emotions of Customers who seek trust and security, Fraud Analysts who need efficient tools, Developers who require user understanding, and Financial Institutions, Regulators which focus on compliance.
WHO are we empathizing with?
Empathizing with banking customers and Institutions, Fraud Analysts, Software Developers
GOAL
Goal is to provide a safe and secure online transactions
Report the fraud
What do they need to DO?
Monitor Accounts
They see Data Breaching, Hacking Account
What do they HEAR?
Hear about fraudulent transactions
Hear about detecting and reporting fraudulent activity
Suspicious transaction notified by bank
What do they THINK and FEEL?
PAINS
GAINS
Fear of Unpredictable fraud incidents.
Fear of losing money
Feels Anxiety and Frustration
Fear of unauthorized access to their financial information
Hope there online Payment is secure.
Think their bank can detect fraudulent transactions in real-time
Gains Peace of mind and confidence in online payments
Gains Swift and effective fraud detection and prevention
See fraud in online shopping and transactions
What do they SEE?
They say that they are worried about unauthorized transactions
What do they SAY?
They say that they are concerned about financial security in online payments
Doesn't want to worry about fraud in online shopping
Worried about the security legitimate transactions
What do they DO?
Check their bank statements regularly
They contact customer support when they suspect fraud
Uses secure payment methods
Share template feedback

3.2 **Ideation & Brainstorming**



4. **REQUIREMENT ANALYSIS**

4.1 **Functional requirement**

The system needs a user-friendly interface in the Flask application for transaction input. It must preprocess data effectively, integrate a machine learning model for real-time predictions, and display results to users. Continuous monitoring and user feedback mechanisms should be in place for ongoing improvements. Additionally, stringent security measures, including user authentication, are essential to safeguard against unauthorized access and protect sensitive information.

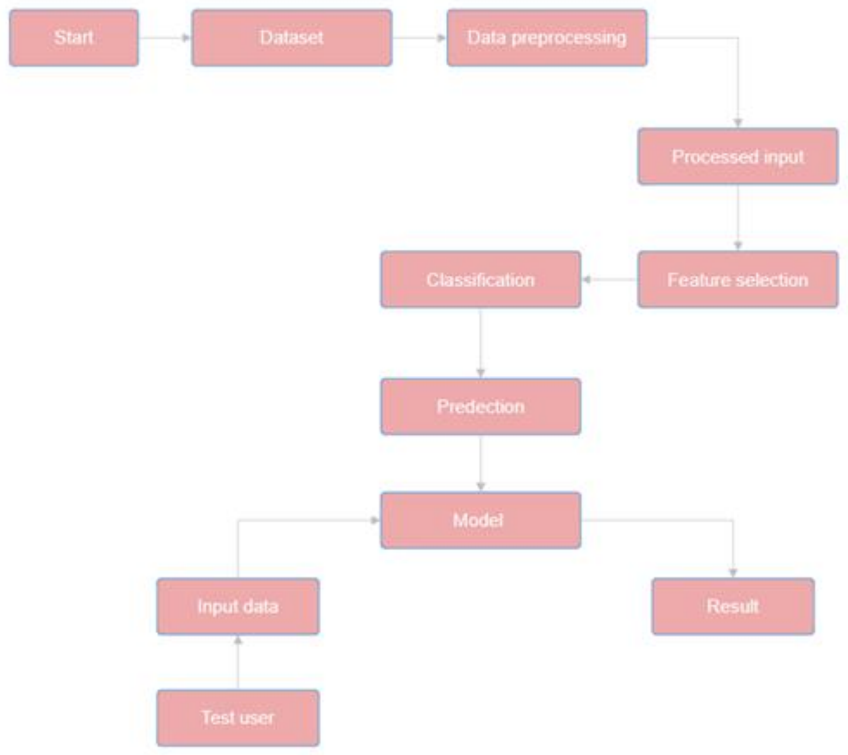
4.2 **Non-Functional requirements**

The online fraud payment detection system prioritizes rapid response times, scalability to accommodate varying user loads, and reliability with 24/7 availability and fault tolerance. The user interface is designed for intuitive use, and the system's code and machine learning model are built for easy maintenance and updates. Security measures, including data encryption and access controls, are implemented, ensuring compliance with regulatory standards. Interoperability is emphasized for seamless integration with external systems, and comprehensive documentation is provided for user guidance and technical understanding.

5. PROJECT DESIGN

5.1 **Data Flow Diagrams & User Stories**

DFD



User Stories

1. As a data scientist, I want to preprocess the collected data to make it suitable for training.
2. As a data scientist, I want to develop a basic ML model for fraud detection.
3. As a data scientist, I want to enhance and validate the ML model for better accuracy.
4. Hyper parameter tuning is conducted. Optimal parameters are identified for improved model performance.
5. The trained machine learning model is successfully saved in a format compatible with deployment.
6. A Flask web application is created. The application integrates the trained fraud detection model. User inputs are accepted through the application.
7. Simulated user interactions are tested. The application handles both valid and invalid inputs appropriately. No critical errors or crashes occur during testing.
8. Any bugs or issues identified during testing are documented. Bugs are categorized by severity. Developers receive clear information to replicate and fix the issues.

5.2 **Solution Architecture**

To tackle online payment fraud, a Flask app seamlessly integrates machine learning, employing algorithms like Random Forest for real-time transaction analysis. This scalable and secure solution identifies potential fraud through continuous analysis and incorporates a feedback loop for ongoing improvement.

1) The structure, characteristics, behavior, and other aspects of the software to project

stakeholders:

Machine Learning Model:

Structure: Incorporates a dynamic feedback loop for accuracy enhancement, with the Flask app serving as the user interface.

Characteristics: Trained on Kaggle's Online Payments Fraud Detection dataset, capable of recognizing fraudulent payments.

Behavior: Processes input data, continuously analyzes transactions in real-time, and promptly identifies potentially fraudulent activities.

Web Application:

Structure: Developed using the Flask framework, composed of HTML pages (index, logout, prediction).

Characteristics: User-friendly interface, seamless integration with the machine learning model.

Behavior: Allows users to input fraud data for prediction, displays results on prediction.html and logout.html pages.

CSS and JS Files:

Structure: JS for enhanced functionality, CSS for styling HTML pages.

Characteristics: JS manages dynamic elements, CSS improves user experience.

Behavior: Ensures visually pleasing design, enhances dynamic features, and maintains responsiveness.

2) Features, Development Phases, and Solution Requirements:

Features:

Identifies unusual transaction patterns.

Designed for real-time processing of a high volume of transactions.

Flags deviations from the expected norm for further investigation.

Development Phases:

Includes model integration into the web application.

Flask app development.

Model training and evaluation.

Data collection and preprocessing.

Solution Requirements:

Kaggle's Online Payments Fraud Detection dataset.

Python with libraries like TensorFlow, Flask, etc.

HTML, CSS, and JS for web app development.

3) Specifications according to which the solution is defined, managed, and delivered:

Data Specifications:

Online Payments Fraud Detection dataset from Kaggle.

Model Specifications:

Flask app integration.

Training and evaluation metrics.

Web App Specifications:

Flask framework.

Three HTML pages (index, prediction, logout).

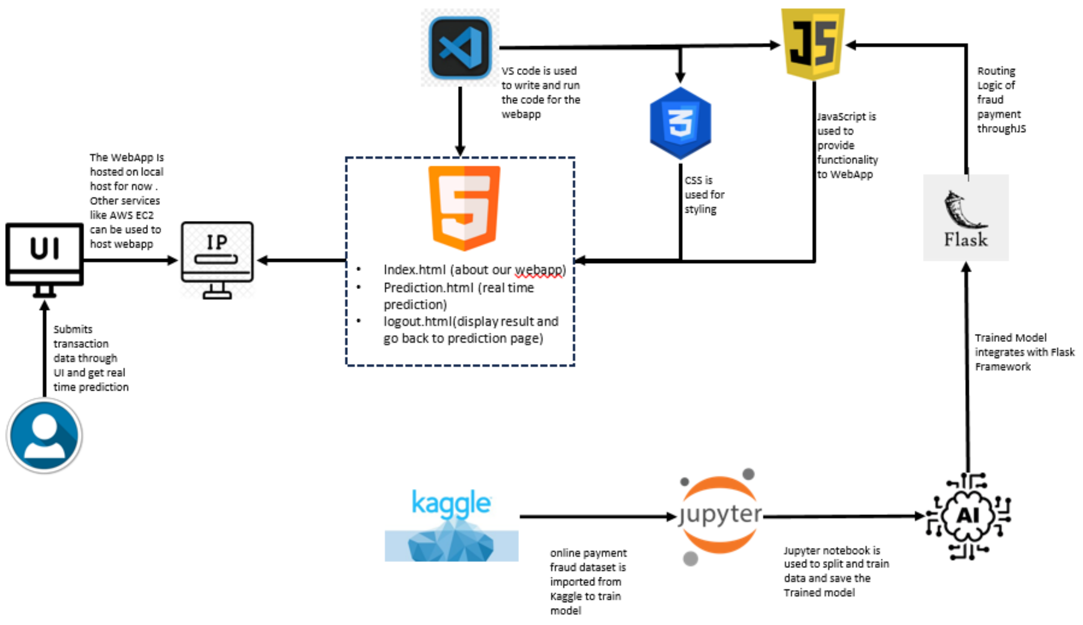
CSS and JS for design and functionality.

Delivery:

Documentation for stakeholders.

Deployed and packaged web app.

Deployed trained model.


VS code is used to write and run the code for the webapp
CSS is used for styling
JavaScript is used to provide functionality to WebApp
Routing Logic of fraud payment throughJS
Flask
The WebApp Is hosted on local host for now . Other services like AWS EC2 can be used to host webapp
UI
IP
• Index.html (about our webapp)
• Prediction.html (real time prediction)
• logout.html(display result and go back to prediction page)
Submits transaction data through UI and get real time prediction
Trained Model integrates with Flask Framework
kaggle
online payment fraud dataset is imported from Kaggle to train model
jupyter
Jupyter notebook is used to split and train data and save the Trained model
AI

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

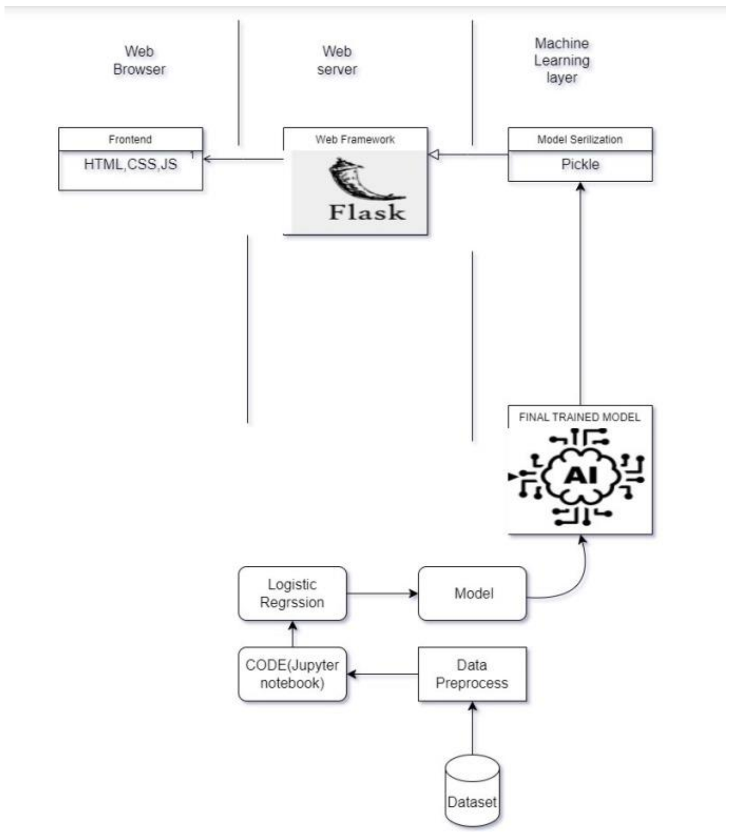


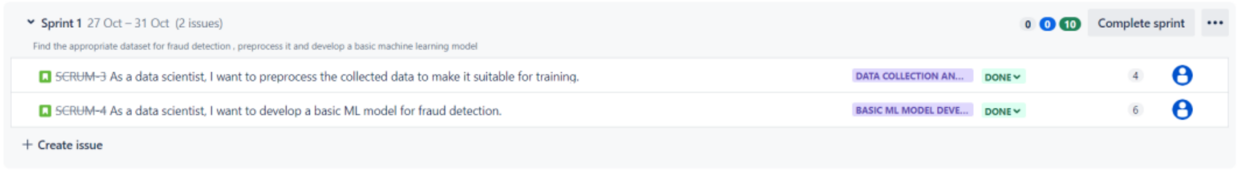
Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	Web framework	Lightweight and flexible web framework in Python	Flask
2.	Machine Learning	Scikit-learn for traditional ML, TensorFlow for deep learning	Scikit-learn
3.	Frontend	Standard web technologies for responsive UI design	HTML,CSS,JavaScript
4.	Model Serialization	Python libraries for serializing machine learning models	Pickle
5.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Data Collection	Ability to collect and store relevant data for training the ML model	Flask, SQLite, Python
2.	Data Preprocessing	Processing and cleaning collected data for ML model preparation	Scikit-learn, Python.
3.	Machine Learning Model	Development of fraud detection model using ML algorithms	Scikit-learn, TensorFlow, Python
4.	Model Validation	Ensuring accuracy and reliability of the ML model through validation	Scikit-learn, Python
5.	Web App Development	Creating a web interface for user interaction with the fraud detection system	Flask, HTML, CSS, JavaScript
6.	Model Saving	Saving trained ML model for later use in the web application	Pickle or joblib, Python
7.	Testing	Rigorous testing of the web app to identify and fix bugs or issues	Testing frameworks, Python

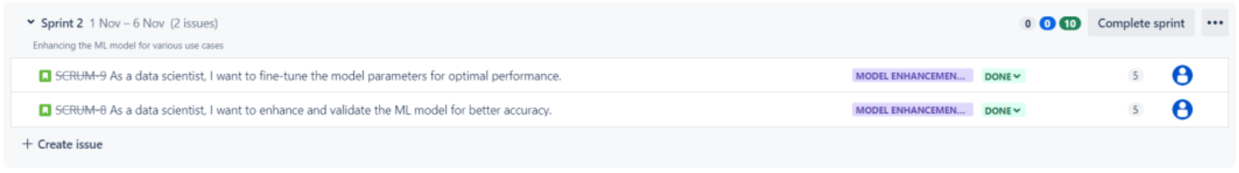
6.2 **Sprint Planning & Estimation**


Sprint 1 27 Oct – 31 Oct (2 issues)

Find the appropriate dataset for fraud detection , preprocess it and develop a basic machine learning model

~~SCRUM-3~~ As a data scientist, I want to preprocess the collected data to make it suitable for training. DATA COLLECTION AN... DONE 4

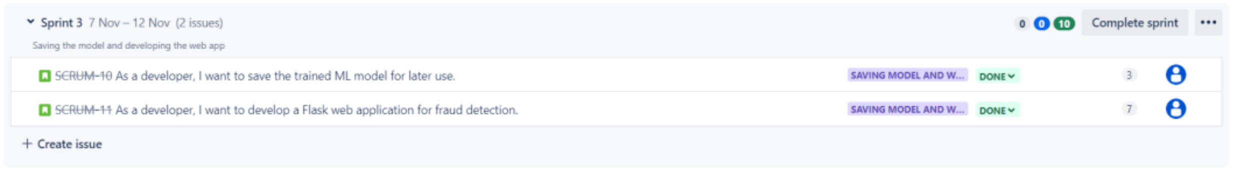
~~SCRUM-4~~ As a data scientist, I want to develop a basic ML model for fraud detection. BASIC ML MODEL DEVE... DONE 6


Sprint 2 1 Nov – 6 Nov (2 issues)

Enhancing the ML model for various use cases

~~SCRUM-9~~ As a data scientist, I want to fine-tune the model parameters for optimal performance. MODEL ENHANCEMEN... DONE 5

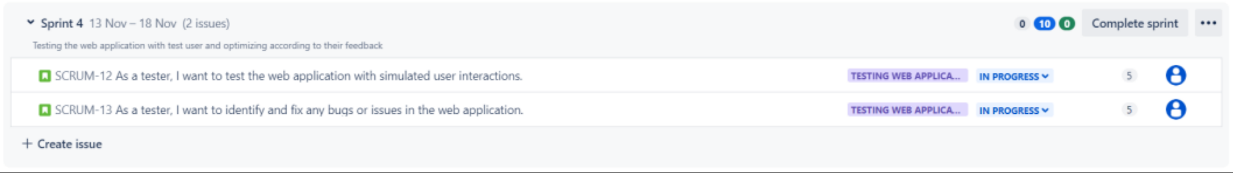
~~SCRUM-8~~ As a data scientist, I want to enhance and validate the ML model for better accuracy. MODEL ENHANCEMEN... DONE 5


Sprint 3 7 Nov – 12 Nov (2 issues)

Saving the model and developing the web app

~~SCRUM-10~~ As a developer, I want to save the trained ML model for later use. SAVING MODEL AND W... DONE 3

~~SCRUM-11~~ As a developer, I want to develop a Flask web application for fraud detection. SAVING MODEL AND W... DONE 7


Sprint 4 13 Nov – 18 Nov (2 issues)

Testing the web application with test user and optimizing according to their feedback

SCRUM-12 As a tester, I want to test the web application with simulated user interactions. TESTING WEB APPLICA... IN PROGRESS 5

SCRUM-13 As a tester, I want to identify and fix any bugs or issues in the web application. TESTING WEB APPLICA... IN PROGRESS 5

6.3 **Sprint Delivery Schedule**

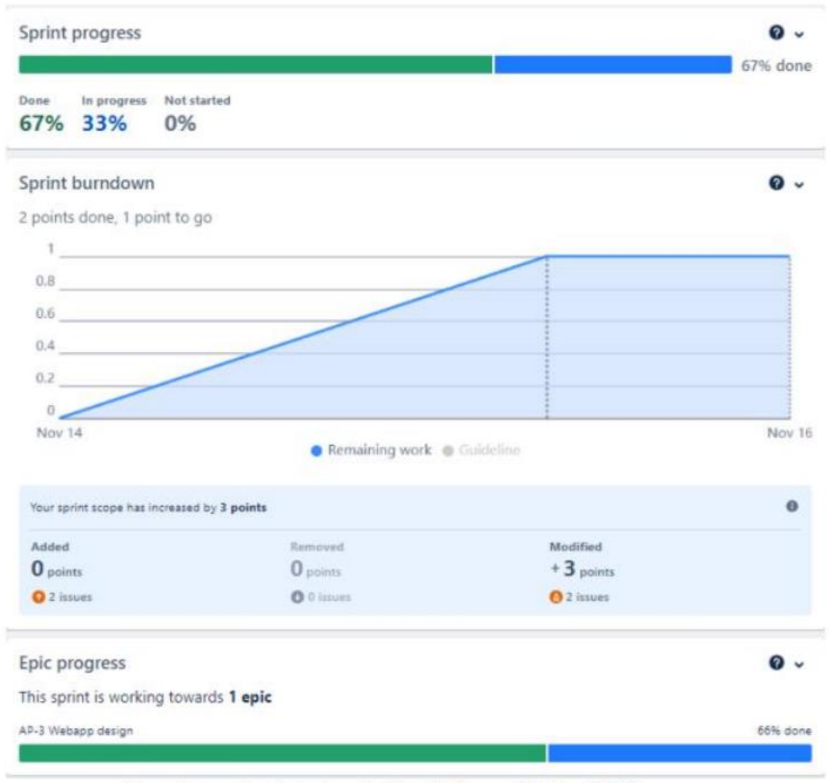
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)
Sprint-1	10	5 Days	27 Oct 2024	31 Oct 2024	10
Sprint-2	10	5 Days	1 Nov 2024	06 Nov 2024	10
Sprint-3	10	5 Days	07 Nov 2024	12 Nov 2024	10
Sprint-4	10	5 Days	13 Nov 2024	18 Nov 2024	10

Velocity

We had a 5-day sprint duration, and the velocity of the team is 10 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day) Sprint 1 velocity =10/5 =2 Sprint 2 velocity =10/5 =2 Sprint 3 velocity =10/5 =2 Sprint 4 velocity=10/5 =2

Burndown Chart

A burndown chart is a graph that shows the amount of work left to do versus the time it takes to complete it. It can be used to predict when all of the work will be completed.


Sprint progress

67% done

Done 67% In progress 33% Not started 0%

Sprint burndown

2 points done, 1 point to go

Your sprint scope has increased by 3 points

Added 0 points, 2 issues

Removed 0 points, 0 issues

Modified +3 points, 2 issues

Epic progress

This sprint is working towards 1 epic

AP-3 Webapp design 66% done

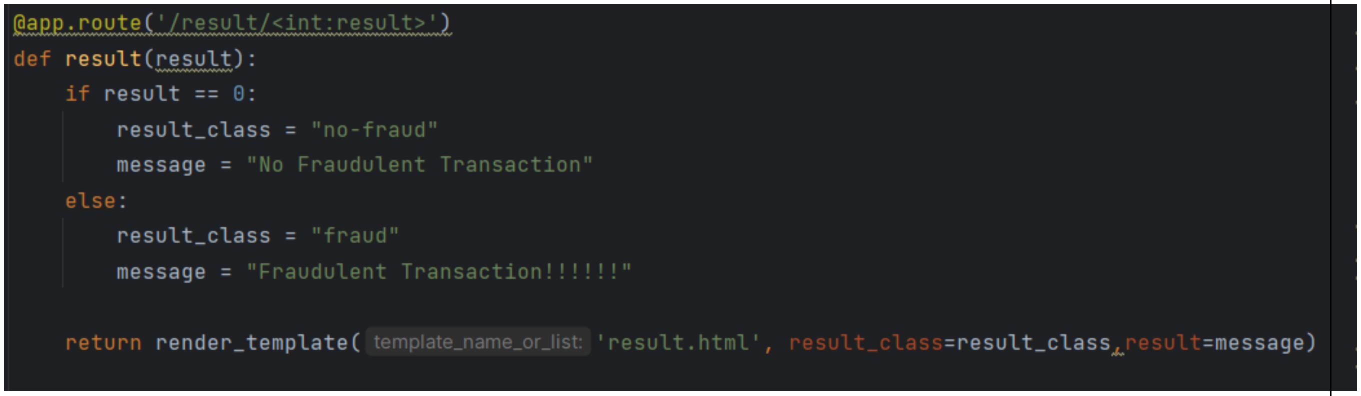
Burndown chart and sprint insights on 15 Nov 2023

7. CODING & SOLUTIONING

7.1 **Background Color Based on Result in Flask Web app:**

Purpose: Display the result page with a background color based on the prediction result.

Code:

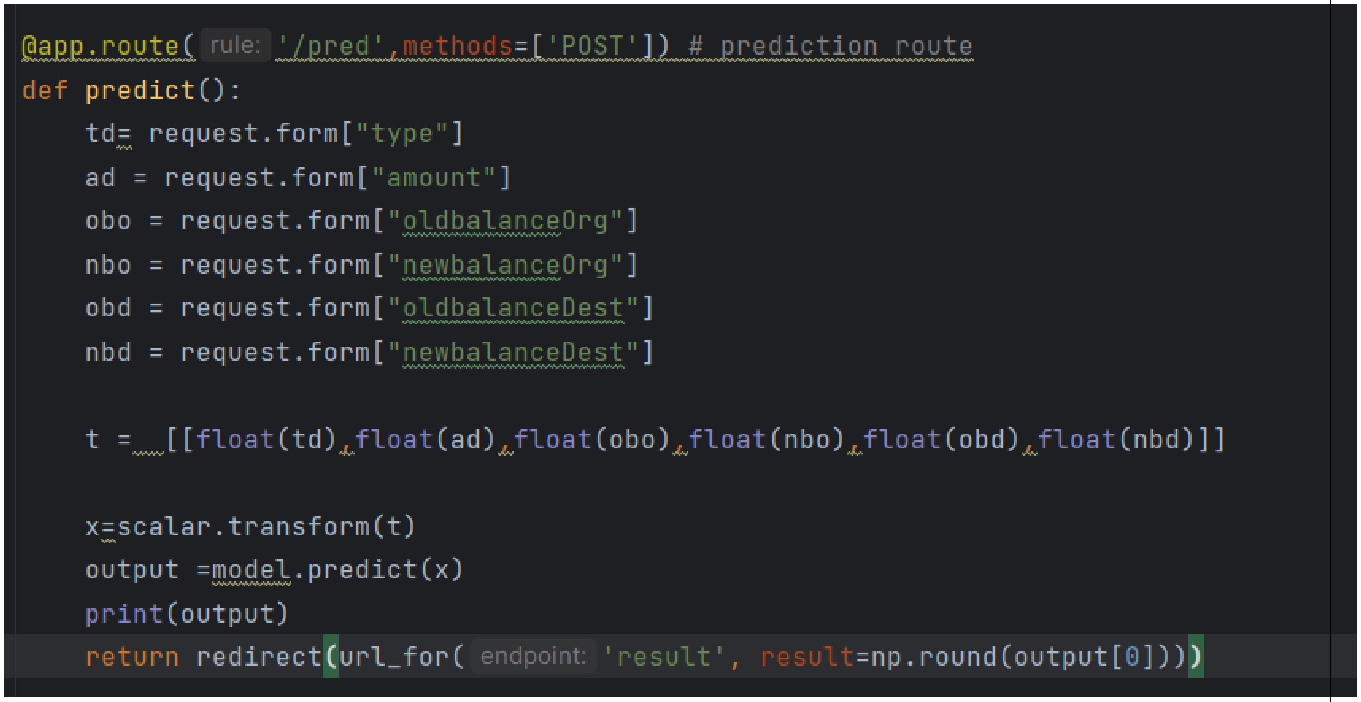


```
@app.route('/result/<int:result>')
def result(result):
    if result == 0:
        result_class = "no-fraud"
        message = "No Fraudulent Transaction"
    else:
        result_class = "fraud"
        message = "Fraudulent Transaction!!!!!!"

    return render_template( template_name_or_list: 'result.html', result_class=result_class,result=message)
```

Explanation: Determines the background color class (result_class) and message (message) based on the prediction result and passes them to the result.html template.

7.2 **Flask Redirect After Form Submission:**

Purpose: Redirects to the result page after form submission instead of rendering the result directly on the prediction route.
Code:



```
@app.route( rule: '/pred',methods=['POST']) # prediction route
def predict():
    td= request.form["type"]
    ad = request.form["amount"]
    obo = request.form["oldbalanceOrg"]
    nbo = request.form["newbalanceOrg"]
    obd = request.form["oldbalanceDest"]
    nbd = request.form["newbalanceDest"]

    t =  [[float(td),float(ad),float(obo),float(nbo),float(obd),float(nbd)]]

    x=scalar.transform(t)
    output =model.predict(x)
    print(output)
    return redirect(url_for( endpoint: 'result', result=np.round(output[0])))
```

Explanation: After processing the form data and making predictions, the application redirects to the result page ('/result/<int:result>') with the predicted result.

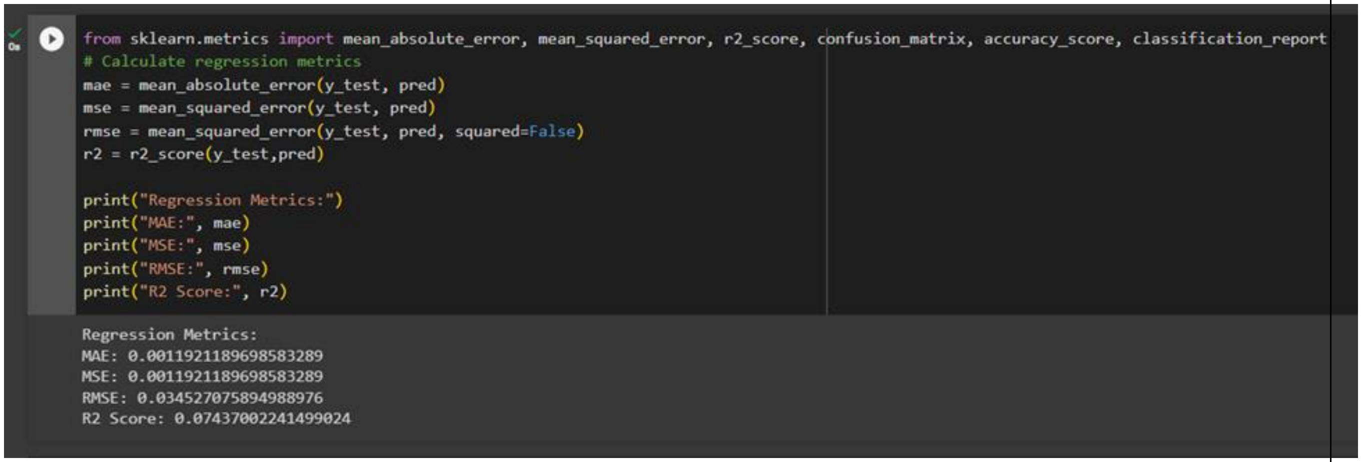
8. PERFORMANCE TESTING

8.1 **Performance Metrics**

Model Performance Testing:

Metrics

Regression metrics:

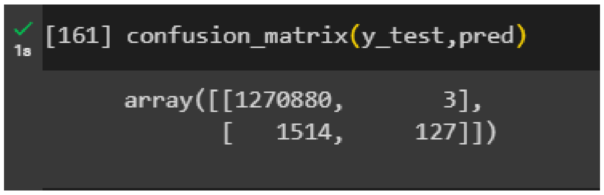


```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, confusion_matrix, accuracy_score, classification_report
# Calculate regression metrics
mae = mean_absolute_error(y_test, pred)
mse = mean_squared_error(y_test, pred)
rmse = mean_squared_error(y_test, pred, squared=False)
r2 = r2_score(y_test,pred)

print("Regression Metrics:")
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r2)
```

```
Regression Metrics:
MAE: 0.0011921189698583289
MSE: 0.0011921189698583289
RMSE: 0.034527075894988976
R2 Score: 0.07437002241499024
```

Classification Metrics:

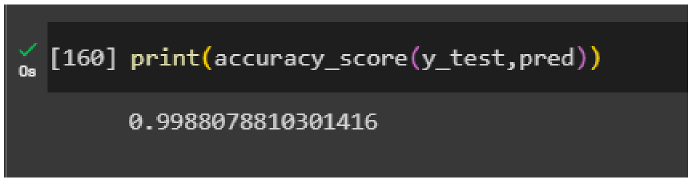
Confusion Matrix



```
[161] confusion_matrix(y_test,pred)
```

```
array([[1270880,       3],
       [   1514,     127]])
```

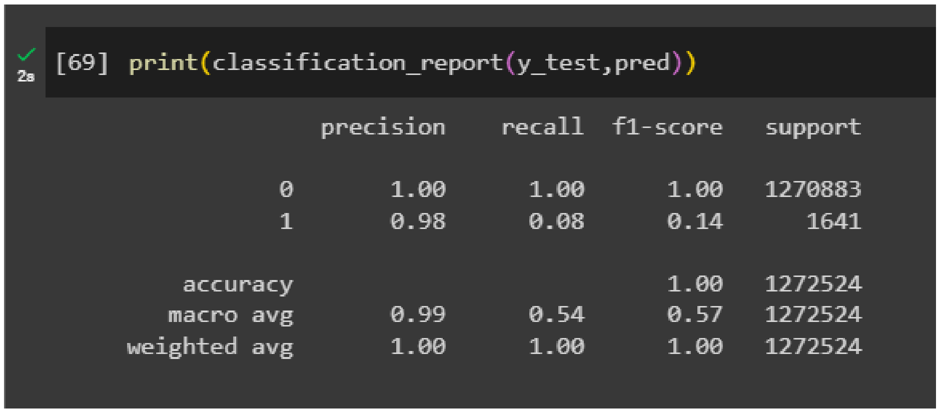
Accuracy Score



```
[160] print(accuracy_score(y_test,pred))
```

```
0.9988078810301416
```

Classification Report

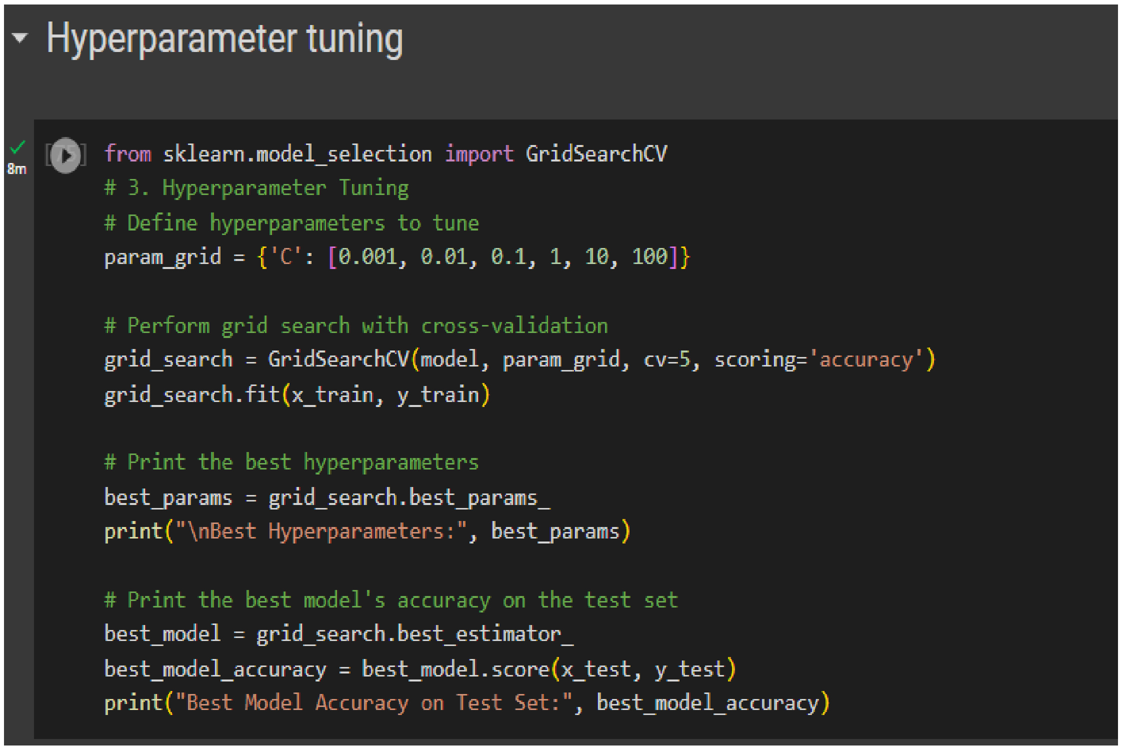


```
[69] print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270883
           1       0.98      0.08      0.14      1641

    accuracy                           1.00   1272524
   macro avg       0.99      0.54      0.57   1272524
weighted avg       1.00      1.00      1.00   1272524
```

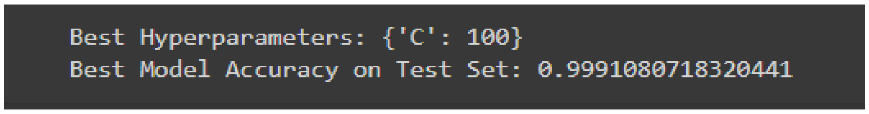
Tune the Model:

Hyperparameter Tuning



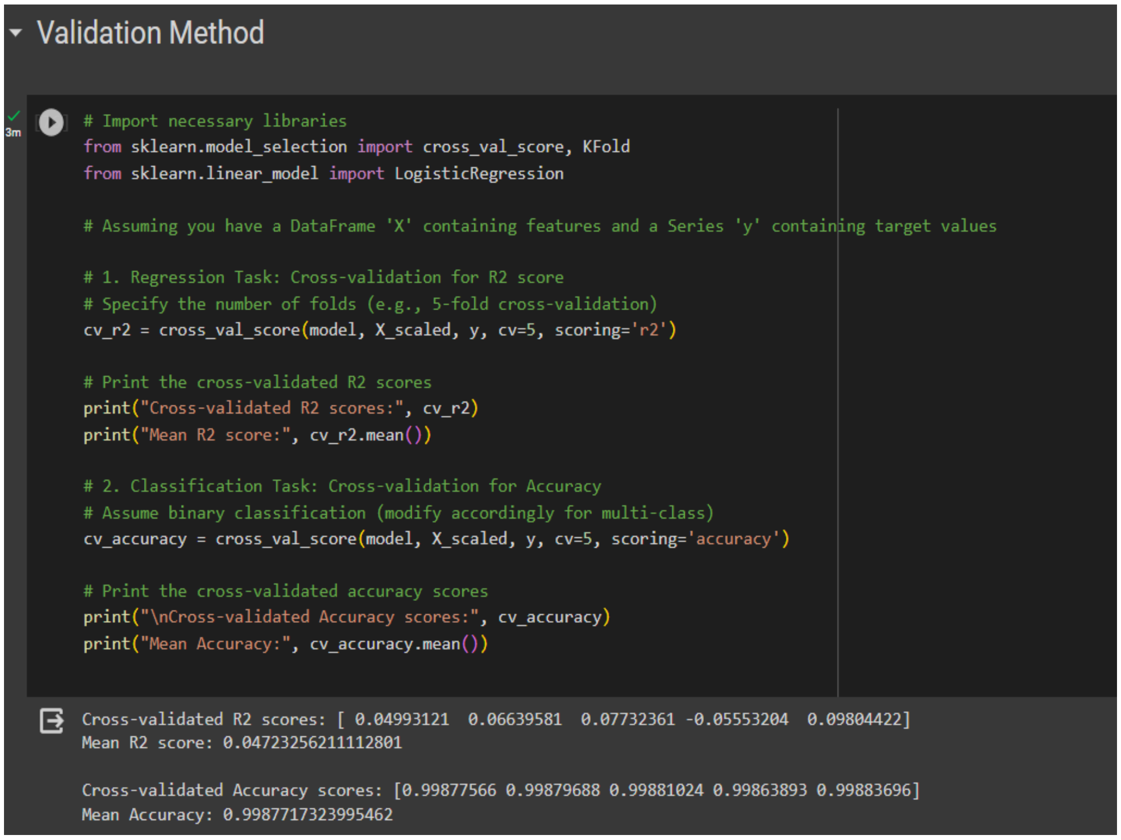
Hyperparameter tuning

```python
from sklearn.model_selection import GridSearchCV
# 3. Hyperparameter Tuning
# Define hyperparameters to tune
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100]}

# Perform grid search with cross-validation
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy')
grid_search.fit(x_train, y_train)

# Print the best hyperparameters
best_params = grid_search.best_params_
print("\nBest Hyperparameters:", best_params)

# Print the best model's accuracy on the test set
best_model = grid_search.best_estimator_
best_model_accuracy = best_model.score(x_test, y_test)
print("Best Model Accuracy on Test Set:", best_model_accuracy)
```



```
Best Hyperparameters: {'C': 100}
Best Model Accuracy on Test Set: 0.9991080718320441
```

Validation Method



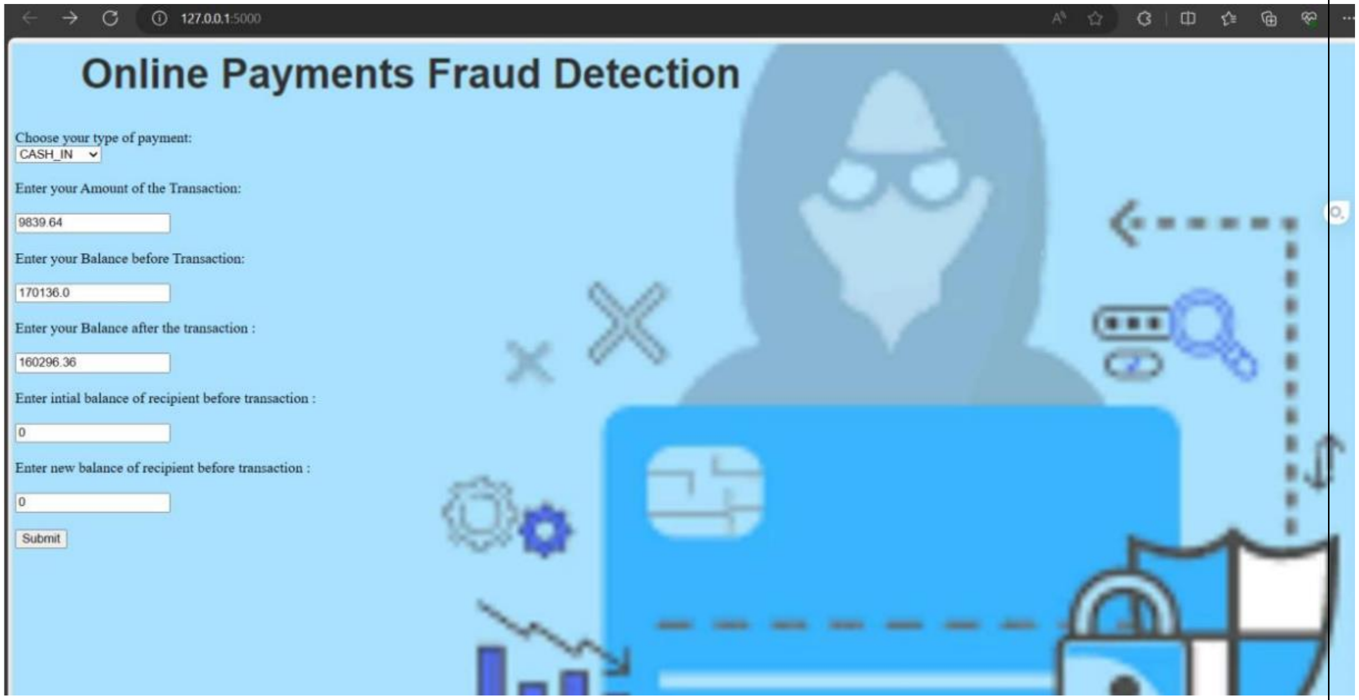
Validation Method

```
# Import necessary libraries
from sklearn.model_selection import cross_val_score, KFold
from sklearn.linear_model import LogisticRegression

# Assuming you have a DataFrame 'X' containing features and a Series 'y' containing target values

# 1. Regression Task: Cross-validation for R2 score
# Specify the number of folds (e.g., 5-fold cross-validation)
cv_r2 = cross_val_score(model, X_scaled, y, cv=5, scoring='r2')

# Print the cross-validated R2 scores
print("Cross-validated R2 scores:", cv_r2)
print("Mean R2 score:", cv_r2.mean())

# 2. Classification Task: Cross-validation for Accuracy
# Assume binary classification (modify accordingly for multi-class)
cv_accuracy = cross_val_score(model, X_scaled, y, cv=5, scoring='accuracy')

# Print the cross-validated accuracy scores
print("\nCross-validated Accuracy scores:", cv_accuracy)
print("Mean Accuracy:", cv_accuracy.mean())
```

```
Cross-validated R2 scores: [ 0.04993121  0.06639581  0.07732361 -0.05553204  0.09804422]
Mean R2 score: 0.047232562111112801

Cross-validated Accuracy scores: [0.99877566 0.99879688 0.99881024 0.99863893 0.99883696]
Mean Accuracy: 0.9987717323995462
```

9. RESULTS

9.1 **Output Screenshots**



127.0.0.1:5000

Online Payments Fraud Detection

Choose your type of payment:
CASH_IN

Enter your Amount of the Transaction:
9839.64

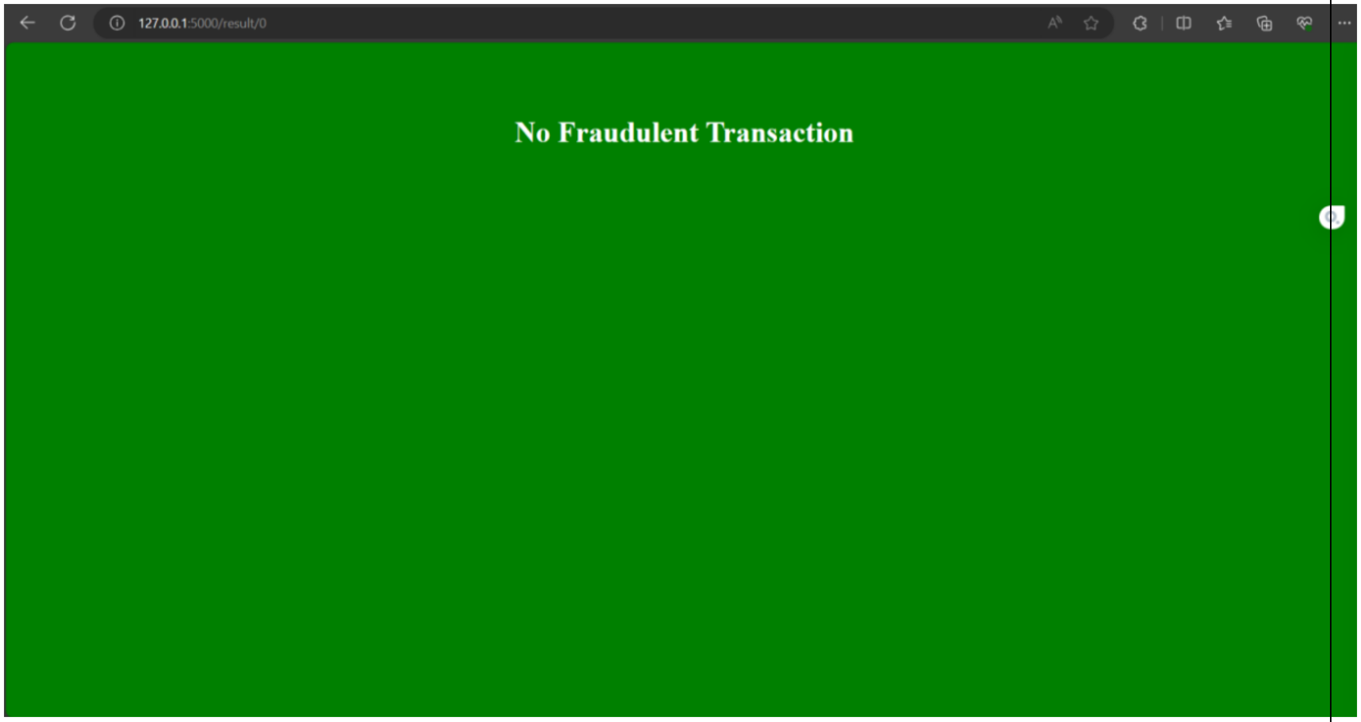
Enter your Balance before Transaction:
170136.0

Enter your Balance after the transaction :
160296.36

Enter intial balance of recipient before transaction :
0

Enter new balance of recipient before transaction :
0

Submit



127.0.0.1:5000/result/0

No Fraudulent Transaction

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

Machine learning empowers automated, real-time fraud detection, dynamically adapting to evolving patterns and minimizing false positives through continuous learning. Its scalability accommodates varying transaction volumes, and the ability to continuously update models ensures adaptability to emerging fraud trends.

DISADVANTAGES

Machine learning empowers automated, real-time fraud detection, dynamically adapting to evolving patterns and minimizing false positives through continuous learning. Its scalability accommodates varying transaction volumes, and the ability to continuously update models ensures adaptability to emerging fraud trends.

11. CONCLUSION

In conclusion, leveraging machine learning for online fraud payment detection offers substantial advantages in terms of real-time adaptability, scalability, and continuous improvement. However, challenges such as data quality dependence, interpretability complexities, resource intensity, and security vulnerabilities must be navigated carefully. Striking a delicate balance between these factors is imperative for building a resilient and effective fraud detection system. As advancements in machine learning continue, addressing these challenges will be crucial in enhancing the overall reliability and efficiency of fraud detection mechanisms in the ever-evolving landscape of online transactions.

12. FUTURE SCOPE

The future scope of online fraud payment detection using machine learning holds promise for significant advancements. Ongoing research and development are likely to yield more sophisticated algorithms, enhancing the accuracy and efficiency of fraud detection models. Collaborative efforts between financial institutions, technology companies, and regulatory bodies may lead to standardized security measures, fostering a unified approach against fraud. Additionally, advancements in real-time adaptive models, integration with IoT devices, and the adoption of edge computing for faster processing are anticipated. The future landscape may also witness increased emphasis on ethical AI practices and the development of international collaborations and regulatory frameworks to counter the global nature of online fraud effectively. As these innovations unfold, the future of fraud detection systems appears poised for a more secure, efficient, and ethically grounded digital financial ecosystem.

13. APPENDIX

Source Code

https://colab.research.google.com/drive/1yk6tYuVhyBc8P9D8vgVMP-4v0kfXECsk?usp=sharing

https://github.com/smartinternz02/SI-GuidedProject-612063-1698674732

GitHub & Project Demo Link

https://github.com/smartinternz02/SI-GuidedProject-612063-1698674732/blob/main/fraud_detection/Demo_video.mp4