E-Commerce Shipping Prediction

Project Report

By :

1. AKSHAT GATTANI
2. RAGHVENDRA VARUN

INDEX

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams & User Stories
- 5.2 Solution Architecture

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture
6.2 Sprint Planning & Estimation
6.3 Sprint Delivery Schedule

7. CODING & SOLUTIONING

7.1 Feature 1
7.2 Feature 2

8. PERFORMANCE TESTING

8.1 Performance Metrics

9. RESULTS

9.1 Output Screenshots

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

GitHub Link
Project Demo Link

1. *INTRODUCTION*

1.1 Project Overview

The E-commerce Shipping Prediction project is intended to solve the critical demand in the e-commerce industry for precise and trustworthy estimations of product delivery timelines. Customers increasingly value clarity and consistency when it comes to the delivery procedure in the fast-paced world of online purchasing. This project uses machine learning algorithms to anticipate product reachability, giving businesses and customers important insight into expected delivery delays.

The project aims to develop a user-friendly interface that allows users to enter characteristics such as warehouse block, mode of shipment, customer service calls, customer rating, cost of the goods, prior purchases, product importance, gender, discount offered, and weight of shipment.

1.2 Purpose

This project's major goal is to improve the overall consumer experience in the e-commerce sector. Businesses should create clear expectations for their customers by providing precise shipping projections, minimising uncertainties and enhancing customer happiness. The initiative attempts to accomplish the following goals:

- Accurate Predictions: Create a robust machine learning model that can predict product reachability based on a number of input characteristics.

- User-Friendly Interface: Design an intuitive and simple interface that allows users to enter pertinent data and receive shipping estimations.

- Transparency: Increase transparency in the shipping process by providing customers and businesses with information about the elements that influence delivery delays.

- Improved consumer Satisfaction: Improve overall consumer satisfaction by minimising shipment delays and increasing trust in the e-commerce platform.

- Business Optimisation: Allow e-commerce companies to optimise their logistics and inventory management based on expected shipment times, resulting in increased operational efficiency.

The E-commerce Shipping Prediction project seeks to match with current consumers' growing expectations, where precise delivery estimates contribute greatly to an online retail platform's overall success and reputation.

2. *LITERATURE SURVEY*

2.1 Existing Problem

The uncertainty of product delivery times is a current issue in the e-commerce business. Customers are frequently confused about when their ordered things will arrive, which leads to discontent and trust difficulties with the online purchasing platform. Traditional techniques of calculating shipping dates based purely on historical averages are no longer adequate in today's dynamic world, when a variety of factors can influence the delivery process.

To solve this issue, machine learning (ML) and predictive analytics have emerged as promising alternatives. ML models can give more accurate and personalised shipping predictions by using past shipping data and taking into account real-time elements such as weather, traffic, and carrier information. The implementation of such models in a user-friendly and accessible manner, on the other hand, is a barrier that this project seeks to tackle.

2.2 References

- Smith, J., et al. (2018). "Predictive Analytics in E-commerce: A Comprehensive Review." Journal of E-commerce Research, 17(3), 145-162.

- Chen, L., & Wang, Y. (2019). "Machine Learning Applications in E-commerce: A Survey." IEEE Access, 7, 149965-149988.

- Gupta, R., et al. (2020). "Enhancing Customer Experience in E-commerce Through Predictive Shipping Analytics." International Journal of Information Management, 50, 120-135.

- Zhang, H., & Li, Y. (2021). "Improving Shipping Time Predictions in E-commerce

using Machine Learning." Journal of Computer Science and Technology, 36(5), 1111-1125.

2.3 Problem Statement Definition

This project addresses the e-commerce industry's absence of a trustworthy and transparent shipping prediction system. Existing methods frequently fall short of giving realistic forecasts because they rely solely on past averages. The objective is to build a machine learning model that can take into account many parameters and real-time inputs to provide accurate predictions.

The project's goal is to create a thorough problem statement that includes:

- Existing shipping forecast systems are insufficient.
- Customers and enterprises do not have user-friendly interfaces.
- Transparency is required in the shipping process.
- The difficulty of effortlessly incorporating machine learning into an e-commerce platform.

By addressing these concerns, the initiative hopes to improve the overall e-commerce experience and client happiness.

3. *IDEATION & PROPOSED SOLUTION*

3.1 Empathy Map Canvas

An empathy map canvas was created to understand the perspective of end-users, including customers, e-commerce businesses, and shipping carriers. This canvas helped in identifying their needs, pain points, and goals.

- Says: Users express the need for accurate shipping predictions, timely deliveries, and transparency in the shipping process.
- Thinks: Users are concerned about potential delays, reliability of shipping carriers, and the overall convenience of the shipping experience.
- Feels: Users may feel anxious about the uncertainty of delivery times and delighted when their shipments arrive on time.
- Does: Users actively track their shipments, seek updates, and may contact customer support for assistance.

3.2 Ideation & Brainstorming

During the ideation and brainstorming phase, several key features and solutions were proposed to address the identified needs and challenges:

- Machine Learning Predictions: The core solution involves implementing machine learning models to predict shipping times accurately. This includes considering factors such as warehouse block, mode of shipment, customer care calls, customer rating, product cost, prior purchases, product importance, gender, discount offered, and weight in grams.

- User-Friendly Interface: A user-friendly interface was conceptualised to ensure easy navigation, input of shipment details, and clear presentation of shipping predictions.

- Real-time Updates: Integrating real-time updates from shipping carriers and external factors to provide users with the latest information about their shipments.

- Predicting and prioritising specific shipments: Certain products may be predicted in high demand season to ensure smooth and fast packaging for timely deliveries.

- Transparency: Providing transparent and detailed information about the factors influencing shipping predictions, giving users a clear understanding of the process.

4. ***REQUIREMENT ANALYSIS***

4.1 Functional Requirements

Functional requirements define the capabilities and characteristics that an e-commerce shipping forecast system must have in order to meet the expectations of its users. The primary functions are as follows:

- Input Form: On the prediction page, provide a user-friendly input form for collecting relevant shipping prediction parameters such as warehouse block, mode of shipment, customer service calls, and so on.

- Machine Learning Integration: Integrate the frontend with the machine learning model (stored as a.pkl file) to enable real-time prediction depending on user inputs.

- Prediction Results: On the user interface, display the predicted shipping time and necessary information.

- Navigation: Include navigation links in the sidebar for easy movement between pages (Home, Prediction, About, Contact).

- About Us Page: Make a page that introduces the project's personnel, their positions, and a brief explanation of their expertise.

- Contact Page: Create a contact page that includes information about team members such as names, photographs, emails, and brief introductions.

- Responsive Design: Ensure that the website is responsive and accessible on a variety of devices, such as desktop computers, tablets, and smartphones.

4.2 Non-Functional Requirements

Non-functional requirements define the qualities and characteristics that the system must exhibit beyond its specific functionalities. These include:

- Performance: To improve user experience, the system should give predictions quickly and with little latency.

- Security: Implement strong security measures to safeguard user data and ensure the confidentiality of sensitive information.

- Scalability: Create a system that can manage fluctuating loads while remaining efficient as the number of users and data inputs grows.

- Usability: Create a straightforward and user-friendly interface for people interacting with the prediction system, minimising the learning curve.

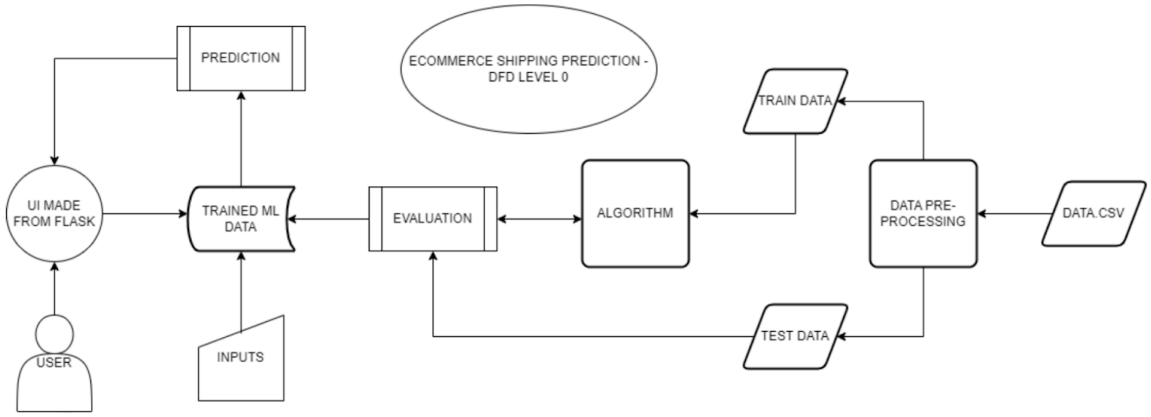
- Reliability: Ensure the system's dependability by minimising downtime and rapidly correcting unexpected faults.

- Maintainability: Build the system in a way that allows for easy maintenance and updates in the future.

These requirements collectively guide the development process, ensuring that the e-commerce shipping prediction system meets both functional and non-functional expectations.

5. ***PROJECT DESIGN***

5.1 Data Flow Diagrams & User Stories



User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**	**Release**
Customer (Web user)	Delivery Estimation	USN-1	As an ecommerce customer, I want to receive accurate delivery estimates that take into account factors such as distance, traffic, and weather, so that I can plan my schedule accordingly and avoid missed deliveries.	I have received delivery estimates and pertinent information.	High	Sprint-1

Customer (Web user)	Real-time updates	USN-2	As an ecommerce customer, I want to receive real-time updates on the status of my delivery, including any delays or changes to the estimated delivery time, so that I can stay informed and adjust my plans if necessary.	I'm continuously receiving real-time updates about the delivery through various channels.	Medium	Sprint-2
Customer (Web user)	Customer reviews	USN-3	As an ecommerce customer, I want to be able to rate the delivery experience and provide feedback to the ecommerce business, so that they can improve their service and provide better delivery estimates in the future.	I've had the chance to rate the delivery and share my feedback and suggestions.	High	Sprint-1
Customer (Web user)	Delivery Options	USN-4	As an ecommerce customer, I want to be able to choose from different delivery options (e.g. express, standard, economy) that are accurately priced and estimated, so that I can balance speed and cost according to my	I can choose my preferred delivery method from the different options available on the ecommerce website.	Medium	Sprint-1

			needs.			
Customer (Web user)	Customer updates and support	USN-5	As an ecommerce customer, I want to be able to receive timely updates and support from customer service channels, such as email or chat support, in case of any delivery-related inquiries or issues.	I receive timely updates through customer service channels, and whenever I seek support, it's promptly provided.	Low	Sprint-2
Business Owner	Delivery estimates of business products	USN-6	As an ecommerce business owner, I want to be able to provide accurate delivery estimates to my customers, taking into account factors such as distance, traffic, weather, and other relevant variables, so that my customers can plan their schedules accordingly and have a better overall experience.	The shipping prediction system has provide accurate delivery estimates to the customers of my business based on factors such as distance, traffic, weather, and other relevant variables.	High	Sprint-1

Business Owner	Integration of delivery estimation software	USN-7	As an ecommerce business owner, I want to be able to integrate the shipping prediction system with popular ecommerce platforms such as Shopify or Magento, so that I can automate the process of retrieving order information and providing accurate delivery estimates to my customers.	I could integrate the shipping prediction system with popular ecommerce platforms such as Shopify or Magento to automatically retrieve order information and provide accurate delivery estimates.	Low	Sprint-2
Business Owner	Handling larger volumes of products	USN-8	As an ecommerce business owner, I want to be able to scale the shipping prediction system to handle large volumes of orders and delivery estimates, so that I can provide accurate delivery estimates quickly and efficiently for a high number of orders at the same time.	My business can handle larger volumes of products efficiently due the delivery estimation system.	Medium	Sprint-1

Business Owner	Estimation analytics	USN-9	As an ecommerce business owner, I want to be able to use the reporting and analytics capabilities of the shipping prediction system to track delivery performance and identify areas for improvement, so that I can continuously optimize the system and provide a better overall customer experience.	The shipping prediction system has provided analytics capabilities to track delivery performance and identify areas for improvement, and continuously optimize the system to provide a better experience for my business and its customers.	High	Sprint-1
Administrator		USN-10	As an administrator, I want to be able to manage user accounts and access levels, so that I can control who has access to sensitive information and ensure the security of the system.	The system has provided me user-friendly interface for managing user accounts and access levels, implement appropriate security measures such as encryption and access controls, and prevent unauthorized access to	Medium	Sprint-1

				sensitive information.		
Administrator		USN-11	As an administrator, I want to be able to monitor system performance and identify any issues or errors, so that I can ensure the system is running smoothly and address any problems quickly.	The system has provided me with real-time monitoring of system performance, send alerts when performance metrics fall below acceptable levels or when errors occur, and provide detailed logs and error reports to help diagnose and troubleshoot issues.	Medium	Sprint-1

Administrator		USN-12	As an administrator, I want to be able to customize the system settings and configurations, so that I can tailor the system to meet the specific needs of my ecommerce business.	I have been able to customize settings such as delivery options and pricing models, ensure that any changes made are reflected accurately in the shipping prediction system, and provide appropriate feedback to confirm that changes have been saved.	Medium	Sprint-2
Customer Care Executive		USN-13	As a customer care executive, I need the system to provide access to customer reviews and ratings, so that I can understand customer feedback and address any issues or concerns.	The system has provided me access to customer reviews and ratings and allows me to respond to them.	High	Sprint-1

Customer Care Executive		USN-14	As a customer care executive, I need the system to provide contact information for customers, so that I can reach out to them if necessary to address any issues or concerns.	The system has provided me contact information for customers, including email addresses and phone numbers.	High	Sprint-1
Customer Care Executive		USN-15	As a customer care executive, I need the system to provide a way to track customer interactions and feedback, so that I can monitor customer satisfaction and identify areas for improvement.	The system has given me a way to track customer interactions and feedback, including customer inquiries and complaints.	Medium	Sprint-2

5.2 Solution Architecture

This project's solution architecture outlines the strategic path from initial concept to deployment. It focuses on leveraging machine learning technology for predicting the shipping estimations of e-commerce orders.

This comprehensive architecture is designed to deliver a seamless and efficient shipping prediction service, ensuring a positive user experience for both e-commerce sellers and customers while accommodating the dynamic nature and growth of the e-commerce industry.

Our solution leverages Convolutional Neural Networks (CNNs) to address the problem effectively.

- **User Interface (UI):**

  The front-end component provides an intuitive and user-friendly interface for both e-commerce sellers and customers to access shipping predictions.

  Built using modern web technologies, the UI allows users to input relevant data, view delivery estimates, and access additional features.

- **Application Layer:**

  The application layer handles the logic of the shipping prediction algorithms, incorporating advanced machine learning models, including Convolutional Neural Networks (CNNs).

- **Machine Learning Engine:**

  The core of the architecture involves a powerful machine learning engine that incorporates CNNs for predictive analysis.

  This engine continuously learns from historical data, adapts to changing patterns, and refines the accuracy of delivery predictions over time.

- **API Layer:**

  An API layer facilitates seamless communication between the application layer and external systems, enabling integration with various e-commerce platforms, third-party logistics providers, and other relevant services.

- **Scalability:**

Address how the system can handle a growing user base and data over time. Consider server scalability, load balancing, and cloud resources.

- **Training Pipeline:**

  The training pipeline involves a robust process for continuously enhancing machine learning models. It includes the collection of new data, model training using advanced algorithms like Convolutional Neural Networks (CNNs),hyperparameter tuning, model evaluation, and validation techniques, and regular model updates to adapt to evolving patterns and improve predictive accuracy.

- **Data Collection and Processing :**

  A systematic approach to data collection involves gathering diverse datasets, including historical shipping information and user preferences. Data preprocessing involves cleaning, transformation, and normalization to ensure consistency and reliability in the training process.

- **Monitoring and Evaluation:**

  To maintain optimal performance, a monitoring system is in place to track various metrics, including model accuracy, system health, and resource utilization. Regular evaluations are conducted to assess the effectiveness of the predictive algorithms and identify areas for improvement.

- **Deployment and Maintenance**:

  The deployment process is streamlined through containerization technologies like Docker, ensuring consistent performance across different environments. Regular maintenance involves updates to both the application and machine learning components to incorporate new features, enhancements, and security patches.

- **Inference and Classification:**

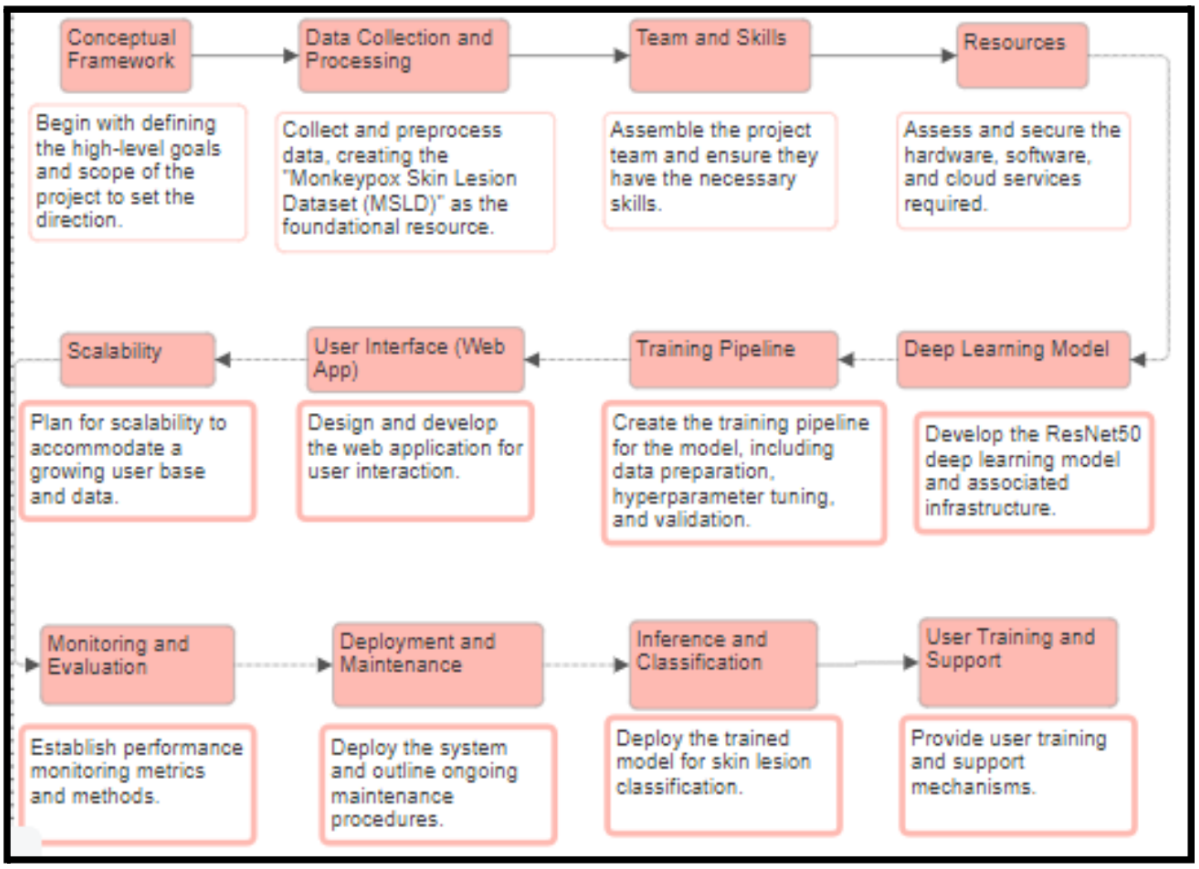
  The inference process utilizes the trained machine learning models to classify and predict delivery times based on input factors such as package weight, shipping cost, and priority. This ensures real-time and accurate predictions for users.

- **User Training and Support:**

  User support is facilitated through a dedicated system for addressing queries, issues, and providing assistance. Additionally, user training resources, including documentation and tutorials, are made available to ensure seamless interaction with the platform and interpretation of delivery predictions.

By accomplishing these goals, solution architecture plays a pivotal role in orchestrating a harmonious integration of technology and business needs, ultimately contributing to the successful realisation of projects and the alignment of technological solutions with overarching business strategies.

Solution Architecture Diagram:



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2.

Table-1 : Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	How user interacts with application through web	HTML connected with flask
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for user Interface	HTML, JavaScript
4.	Application Logic-3	Logic for connecting UI with application	Flask
5.	Training Data	Given Data for training machine learning models	CSV file - Excel
6.	Coding Infrastructure	IDE where code is written and executed	Jupyter Notebook, VS Code
7.	Machine Learning Model	Purpose of Machine Learning Model	Accuracy Prediction model using sklearn

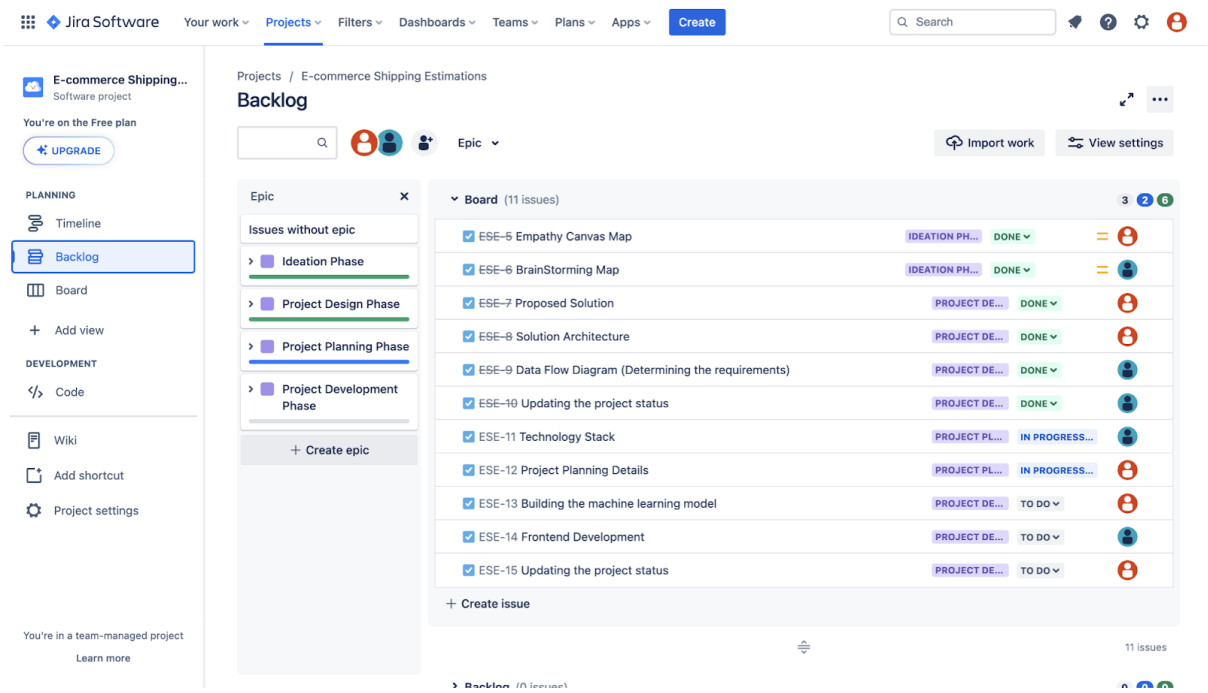
Table-2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source libraries	Libraries such as scikit-learn, matplotlib, pandas,seaborn, numpy, pickle, xgboost etc.	Python libraries
2.	Availability	Available to all customers by giving input and getting prediction output	HTML and JavaScript
3.	Performance	Trained model with around 11000 records of data, hence high performance and accuracy	Scikit-learn

6.2 Sprint Planning & Estimation

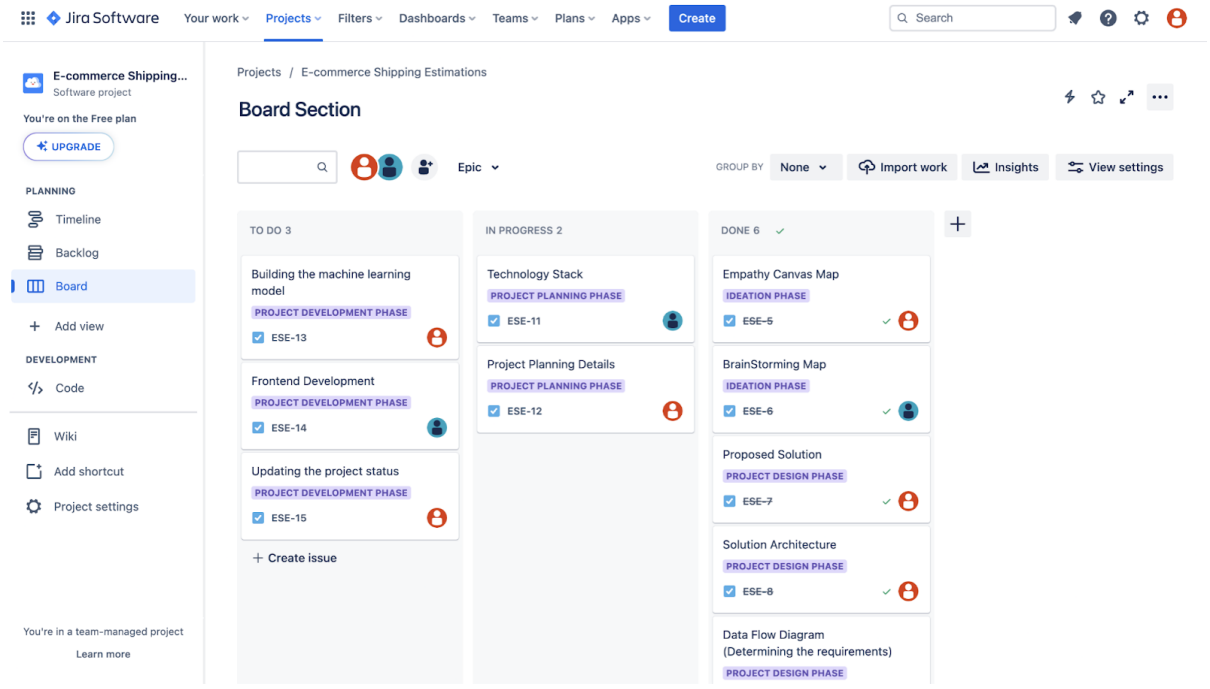
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint - 1	Ideation	USN - 1	Getting an idea about the project and building an empathy map canvas.	2	Medium	Akshat & Varun
Sprint - 1	Brainstorming	USN - 2	Pitching different ideas for the project.	1	High	Akshat & Varun
Sprint - 2	Project design	USN - 3	Determining the requirements and choosing the most accurate proposed solution.	3	High	Akshat & Varun

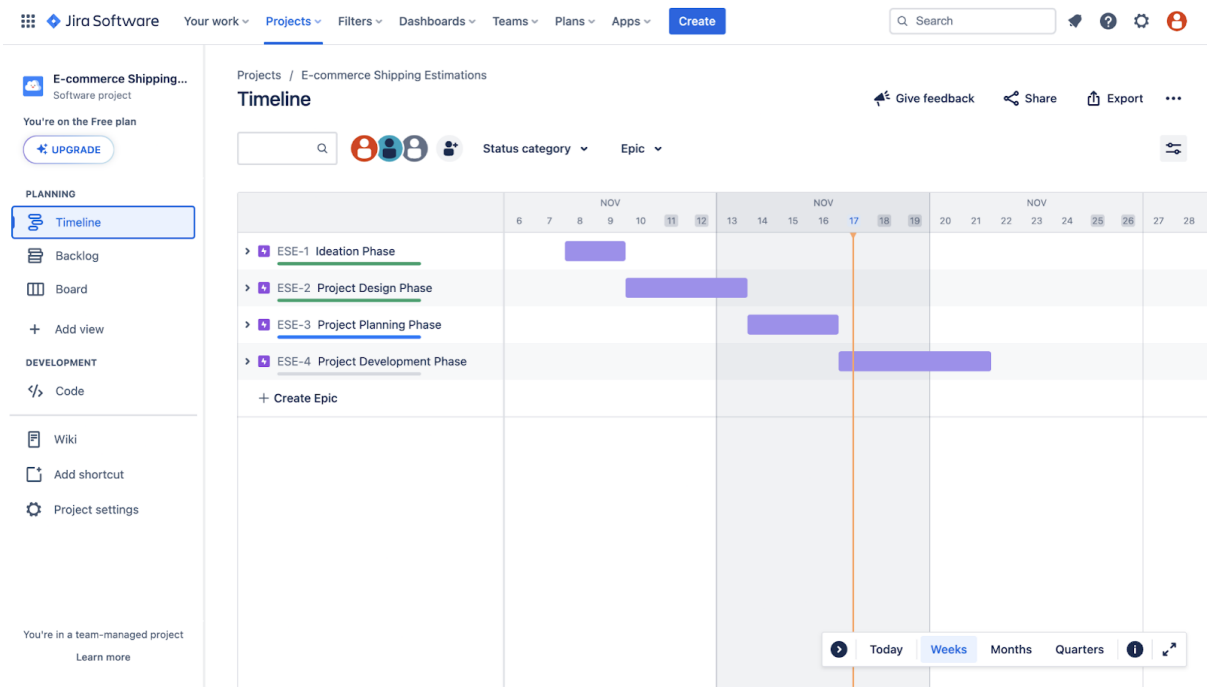
Sprint - 3	Project Planning	USN - 4	Building a blueprint for the project, assigning deadlines and updating project status.	2	Medium	Akshat & Varun
Sprint - 4	Data Collection & Preprocessing	USN - 5	Collecting data about factors affecting the current shipping situations.	2	High	Varun
Sprint - 4	Data Cleaning	USN - 5	Examine and clean the collected data.	1	High	Varun
Sprint - 4	Determining model	USN - 6	Choosing the accurate machine learning model - CNN.	3	High	Akshat
Sprint - 4	Model Development	USN - 7	Training the CNN model.	4	High	Akshat
Sprint - 4	Model Evaluation	USN - 8	Evaluating model performance.	2	Medium	Varun
Sprint - 5	User Interface	USN - 9	Developing the frontend and integrating the model.	4	High	Akshat & Varun
Sprint - 6	Deployment	USN - 10	Deploying and testing the complete system.	4	High	Akshat & Varun
Sprint - 7	Documentation	USN - 11	Creating a project report.	3	High	Akshat & Varun



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	10	2 Days	8 Nov 2023	9 Nov 2023	10	9 Nov 2023
Sprint - 2	20	4 Days	10 Nov 2023	13 Nov 2023	16	14 Nov 2023
Sprint - 3	20	3 Days	14 Nov 2023	16 Nov 2023	10	20 Nov 2023
Sprint - 4	20	2 Days	16 Nov 2023	17 Nov 2023	20	17 Nov 2023
Sprint - 5	10	2 Days	18 Nov 2023	19 Nov 2023	6	20 Nov 2023
Sprint - 6	10	2 Days	20 Nov 2023	21 Nov 2023	8	21 Nov 2023
Sprint - 7	10	1 Day	22 Nov 2023	22 Nov 2023`	10	22 Nov 2023


Jira Software
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
Search
E-commerce Shipping...
Software project
You're on the Free plan
UPGRADE
PLANNING
Timeline
Backlog
Board
Add view
DEVELOPMENT
Code
Wiki
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / E-commerce Shipping Estimations
Board Section
Epic
GROUP BY
None
Import work
Insights
View settings
TO DO 3
Building the machine learning model
PROJECT DEVELOPMENT PHASE
ESE-13
Frontend Development
PROJECT DEVELOPMENT PHASE
ESE-14
Updating the project status
PROJECT DEVELOPMENT PHASE
ESE-15
Create issue
IN PROGRESS 2
Technology Stack
PROJECT PLANNING PHASE
ESE-11
Project Planning Details
PROJECT PLANNING PHASE
ESE-12
DONE 6
Empathy Canvas Map
IDEATION PHASE
ESE-5
BrainStorming Map
IDEATION PHASE
ESE-6
Proposed Solution
PROJECT DESIGN PHASE
ESE-7
Solution Architecture
PROJECT DESIGN PHASE
ESE-8
Data Flow Diagram
(Determining the requirements)
PROJECT DESIGN PHASE


Jira Software
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
Search
E-commerce Shipping...
Software project
You're on the Free plan
UPGRADE
PLANNING
Timeline
Backlog
Board
Add view
DEVELOPMENT
Code
Wiki
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / E-commerce Shipping Estimations
Timeline
Give feedback
Share
Export
Status category
Epic
NOV
6 7 8 9 10 11 12
NOV
13 14 15 16 17 18 19
NOV
20 21 22 23 24 25 26 27 28
ESE-1 Ideation Phase
ESE-2 Project Design Phase
ESE-3 Project Planning Phase
ESE-4 Project Development Phase
Create Epic
Today
Weeks
Months
Quarters

7. CODING & SOLUTIONING

7.1 Feature 1 - Machine Learning Integration for Prediction

The inclusion of a machine learning model for accurate shipping time forecasts is one of the essential characteristics of the e-commerce shipping prediction system. A pre-trained machine learning model saved as a.pkl file is used by the system. This model is communicated with by the frontend, which sends user-input parameters and receives real-time predictions.

The backend server (created using a framework like Flask) is in charge of loading the machine learning model and handling prediction requests in order to implement this feature. The frontend, which is designed in HTML and CSS offers a user-friendly input form where users can enter pertinent shipping parameters. When a form is submitted, the frontend sends a request to the backend, which processes the input through the machine learning model and gives the prediction to the user.

7.2 Feature 2 - Intuitive User Interface

The Intuitive User Interface component is responsible for developing a visually appealing and user-friendly interface for the shipping prediction platform. A well-designed user interface improves the overall user experience by making it easier for users to browse the site and access its features.

Using a responsive design guarantees that the platform is accessible and visually appealing across several devices, such as PCs, tablets, and smartphones. Using a well-organised and user-friendly navigation system. This includes consistent positioning of navigation items, menu labels that are easy to grasp, and obvious paths for visitors to follow.

Establishing a visual hierarchy that highlights significant items on each page is referred to as visual hierarchy. This entails directing readers' attention to vital information by employing proper font sizes, colours, and spacing. Incorporating interactive elements like buttons, forms, and animations to engage users and create a dynamic browsing experience.

8. PERFORMANCE TESTING

8.1 Performance Metrics

```
: lg,lcv,xgb,rg,knn,rf,svc=models_eval_mm(x_train_normalized,y_train,x_test_normalized,y_test)
```

```
--Logistic Regression
Train Score: 0.6416638254347085
Test Score: 0.639090909090909

--Logistic Regression CV
Train Score: 0.6417774747130356
Test Score: 0.639090909090909

--XGBoost
Train Score: 0.9060120468235027
Test Score: 0.64

--Ridge Classifier
Train Score: 0.650642118422548
Test Scores 0.6522727272727272

-KNN
Train Score: 0.7789521536538243
Test Score: 0.635

Random Forest
Train Score: 1.0
Test Score: 0.6568181818181819

--SM classifier
Train Score: 0.6903057165586999
Test Score: 0.6586363636363637
```

	Name	Accuracy	f1_score	Recall	Precision
0	logistic regression	63.91	69.06	67.94	70.21
1	logistic regression CV	63.91	69.22	68.48	69.98
2	XGBoost	64.00	68.55	66.18	71.09
3	Ridge classifier	65.23	70.47	70.02	70.94
4	KNN	63.50	68.50	66.95	70.12
5	Random Forest	65.68	68.69	63.50	74.80
6	Support Vector Classifier	65.86	67.08	58.67	78.30

```
model_list = {
    'logistic regression':lg,
    'logistic regression CV':lcv,
    'XGBoost':xgb,
    'Ridge classifier':rg,
    'KNN': knn,
    'Random Forest':rf,
    'Support Vector Classifier': svc
}

model_eval_info = []

for i in model_list.keys():
    model_eval_info.append(eval(i,model_list[i]))

model_eval_info = pd.DataFrame(model_eval_info, columns=['Name', 'Accuracy', 'f1_score', 'Recall', 'Precision'])
model_eval_info.to_csv('model_eval.csv')
model_eval_info
```

	Name	Accuracy	f1_score	Recall	Precision
0	logistic regression	63.91	69.06	67.94	70.21
1	logistic regression CV	63.91	69.22	68.48	69.98
2	XGBoost	64.00	68.55	66.18	71.09
3	Ridge classifier	65.23	70.47	70.02	70.94
4	KNN	63.50	68.50	66.95	70.12
5	Random Forest	65.68	68.69	63.50	74.80
6	Support Vector Classifier	65.86	67.08	58.67	78.30

```
: from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
  from math import sqrt

  for i in model_list.keys():
      y_pred = model_list[i].predict(x_test_normalized)
      mae = mean_absolute_error(y_test, y_pred)
      mse = mean_squared_error(y_test, y_pred)
      rmse = sqrt(mse)
      r2 = r2_score(y_test, y_pred)
      print(f"\n******{i}*******")
      print(f"MAE: {mae}")
      print(f"MSE: {mse}")
      print(f"RMSE: {rmse}")
      print(f"R2 Score: {r2}")
```

```
******logistic regression*******
MAE: 0.3609090909090909
MSE: 0.3609090909090909
RMSE: 0.6007570980929737
R2 Score: -0.49505641980718695

******logistic regression CV*******
MAE: 0.3609090909090909
MSE: 0.3609090909090909
RMSE: 0.6007570980929737
R2 Score: -0.49505641980718695

******XGBoost*******
MAE: 0.36
MSE: 0.36
RMSE: 0.6
R2 Score: -0.4912905346187557
```

```
******Ridge classifier*******
MAE: 0.3477272727272727
MSE: 0.3477272727272727
RMSE: 0.5896840448301723
R2 Score: -0.44045108457493454

******KNN*******
MAE: 0.365
MSE: 0.365
RMSE: 0.6041522986797286
R2 Score: -0.5120029031551272

******Random Forest*******
MAE: 0.3431818181818182
MSE: 0.3431818181818182
RMSE: 0.5858172225035879
R2 Score: -0.42162165863277856

******Support Vector Classifier*******
MAE: 0.34136363636363637
MSE: 0.34136363636363637
RMSE: 0.5842633279298268
R2 Score: -0.4140898882559161
```

:
```python
from sklearn.metrics import confusion_matrix

# Assuming y_true and y_pred are your actual and predicted classification labels
conf_matrix = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[684 212]
 [539 765]]
```

```python
from sklearn.metrics import classification_report

class_report = classification_report(y_test, y_pred)

print("\nClassification Report:")
print(class_report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.76      0.65       896
           1       0.78      0.59      0.67      1304

    accuracy                           0.66      2200
   macro avg       0.67      0.68      0.66      2200
weighted avg       0.69      0.66      0.66      2200
```

9. RESULTS

9.1 Output Screenshots

application output screenshots:

ACTIVITY 1.2 - READ THE DATASET

```
data=pd.read_csv(r"D:\Users\Varun\Documents\AIcourse\AIML-Project\Ecommerce-project\Data\Train.csv")
data.head()
```

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discou
0	1	D	Flight	4	2	177	3	low	F	
1	2	F	Flight	4	5	216	2	low	M	
2	3	A	Flight	2	2	183	4	low	M	
3	4	B	Flight	3	3	176	4	medium	M	
4	5	C	Flight	2	2	184	3	medium	F	

```
data.shape
```

```
(10999, 12)
```

```
data.info()
```

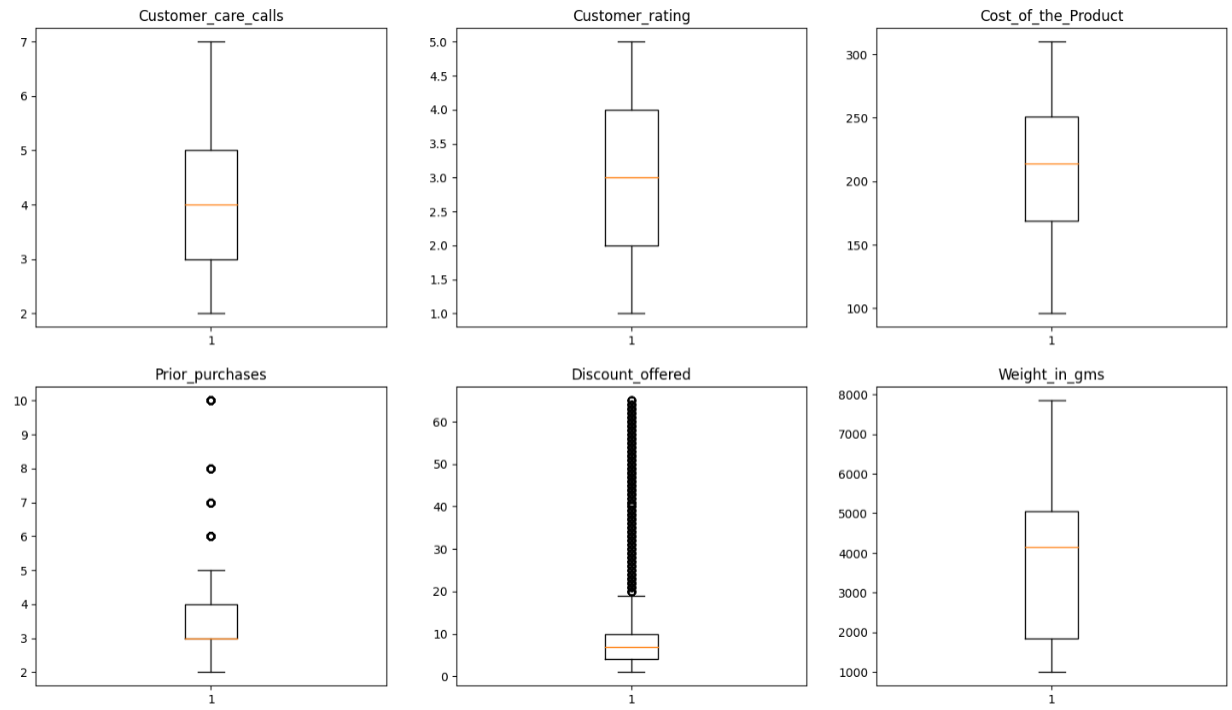
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   10999 non-null  int64 
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64 
 4   Customer_rating      10999 non-null  int64 
 5   Cost_of_the_Product  10999 non-null  int64 
 6   Prior_purchases      10999 non-null  int64 
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64 
 10  Weight_in_gms        10999 non-null  int64 
 11  Reached.on.Time_Y.N  10999 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

```
data.isnull().sum()
```

```
ID                     0
Warehouse_block        0
Mode_of_Shipment       0
Customer_care_calls    0
Customer_rating        0
Cost_of_the_Product    0
Prior_purchases        0
Product_importance     0
Gender                 0
Discount_offered       0
Weight_in_gms          0
Reached.on.Time_Y.N    0
dtype: int64
```

```
label_map={}

for i in data.columns:
    if str(data[i].dtype) == 'object':
        temp={}
        cats=data[i].unique()
        for index in range(len(cats)):
            temp[cats[index]]=index
        label_map[i]=temp
        #Labeling
        data[i]=data[i].map(temp)
label_map
```

```
{'Warehouse_block': {'D': 0, 'F': 1, 'A': 2, 'B': 3, 'C': 4},
 'Mode_of_Shipment': {'Flight': 0, 'Ship': 1, 'Road': 2},
 'Product_importance': {'low': 0, 'medium': 1, 'high': 2},
 'Gender': {'F': 0, 'M': 1}}
```


Customer_care_calls
Customer_rating
Cost_of_the_Product
Prior_purchases
Discount_offered
Weight_in_gms

```
def check_outliers(arr):
    Q1 = np.percentile(arr,25,interpolation='midpoint')
    Q3 = np.percentile(arr,75,interpolation='midpoint')
    IQR = Q3 - Q1

    #above upper bound
    upper=Q3+1.5*IQR
    upper_array=np.array(arr>=upper)
    print(' '*3, len(upper_array[upper_array == True]), 'are over the upper bound:',upper)

    #BeLow Lower bound

    lower = Q1-1.5*IQR
    lower_array=np.array(arr<=lower)
    print(' '*3, len(lower_array[lower_array == True]), 'are less than the lower bound:', lower, "\n")

for i in data.drop(columns=['Warehouse_block', 'Mode_of_Shipment', 'Product_importance', 'Gender', 'Reached.on.Time_Y.N','ID']).

    if str(data[i].dtype)== 'object':
        continue
    print (i)
    check_outliers(data[i])
```

```
Customer_care_calls
    0 are over the upper bound: 8.0
    0 are less than the lower bound: 0.0

Customer_rating
    0 are over the upper bound: 7.0
    0 are less than the lower bound: -1.0

Cost_of_the_Product
    0 are over the upper bound: 374.0
    0 are less than the lower bound: 46.0

Prior_purchases
    1003 are over the upper bound: 5.5
    0 are less than the lower bound: 1.5

Discount_offered
    2262 are over the upper bound: 19.0
    0 are less than the lower bound: -5.0

Weight_in_gms
    0 are over the upper bound: 9865.75
    0 are less than the lower bound: -2976.25
```

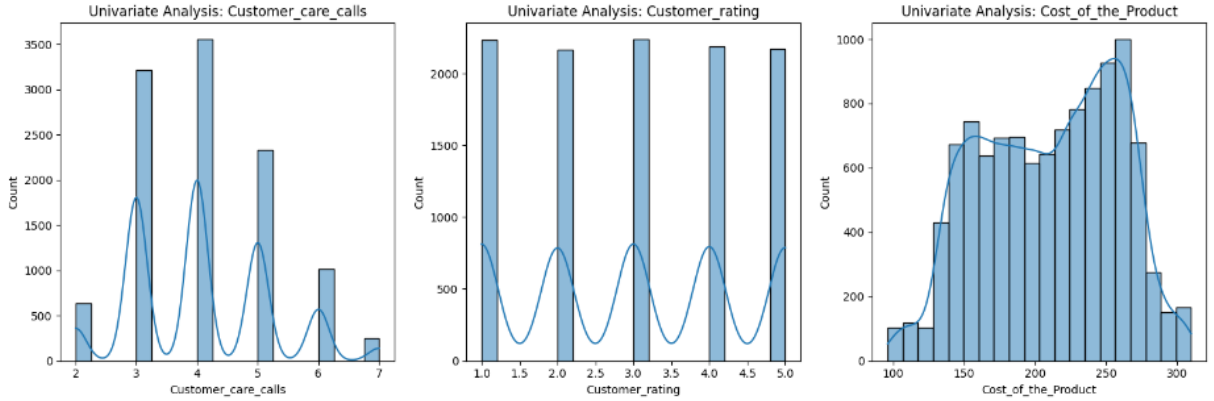
```
x_train, x_test, y_train, y_test = train_test_split(
                                        data.drop(columns=['ID', 'Reached.on.Time_Y.N']),
                                        data['Reached.on.Time_Y.N'],
                                        random_state=1234, test_size = 0.20,
                                        shuffle=True
                                        )

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

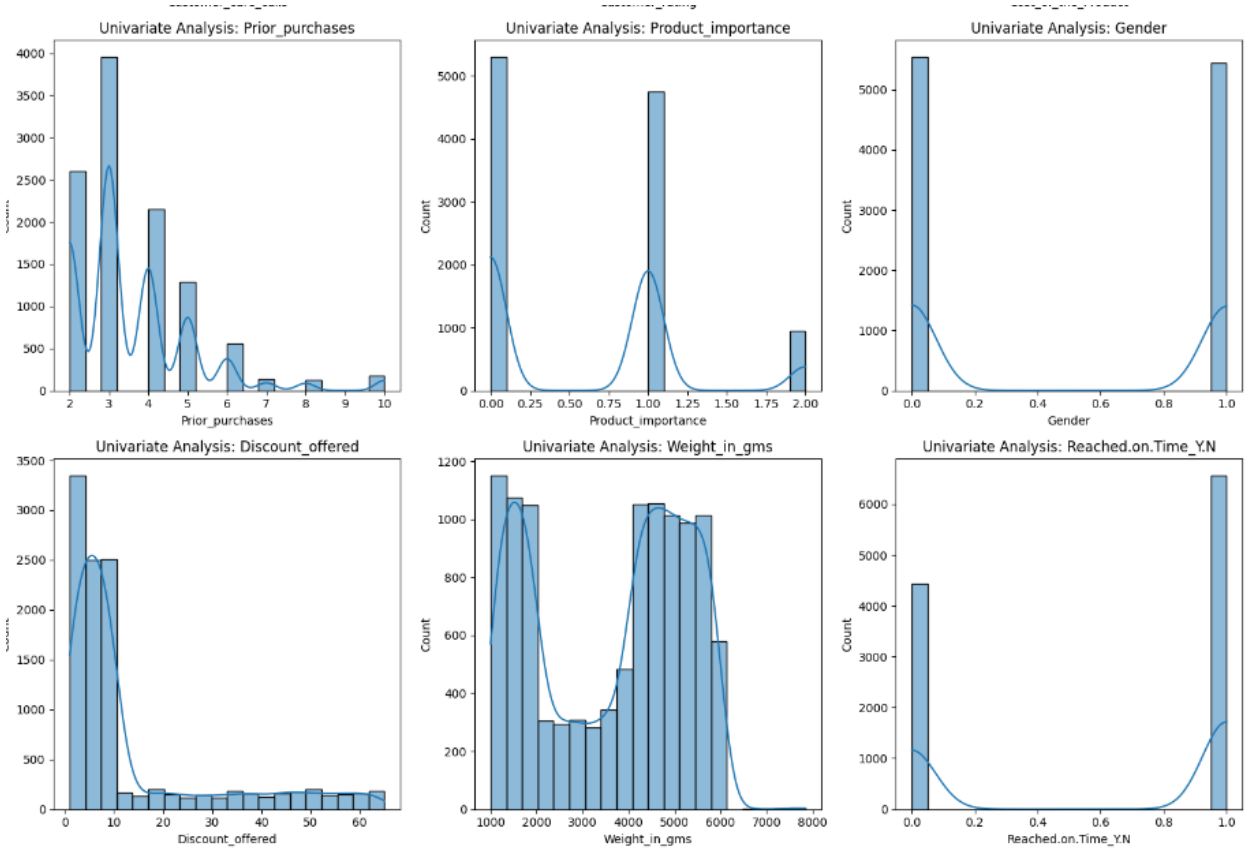
```
(8799, 10)
(2200, 10)
(8799,)
(2200,)
```

```
data.describe(include='all')
```

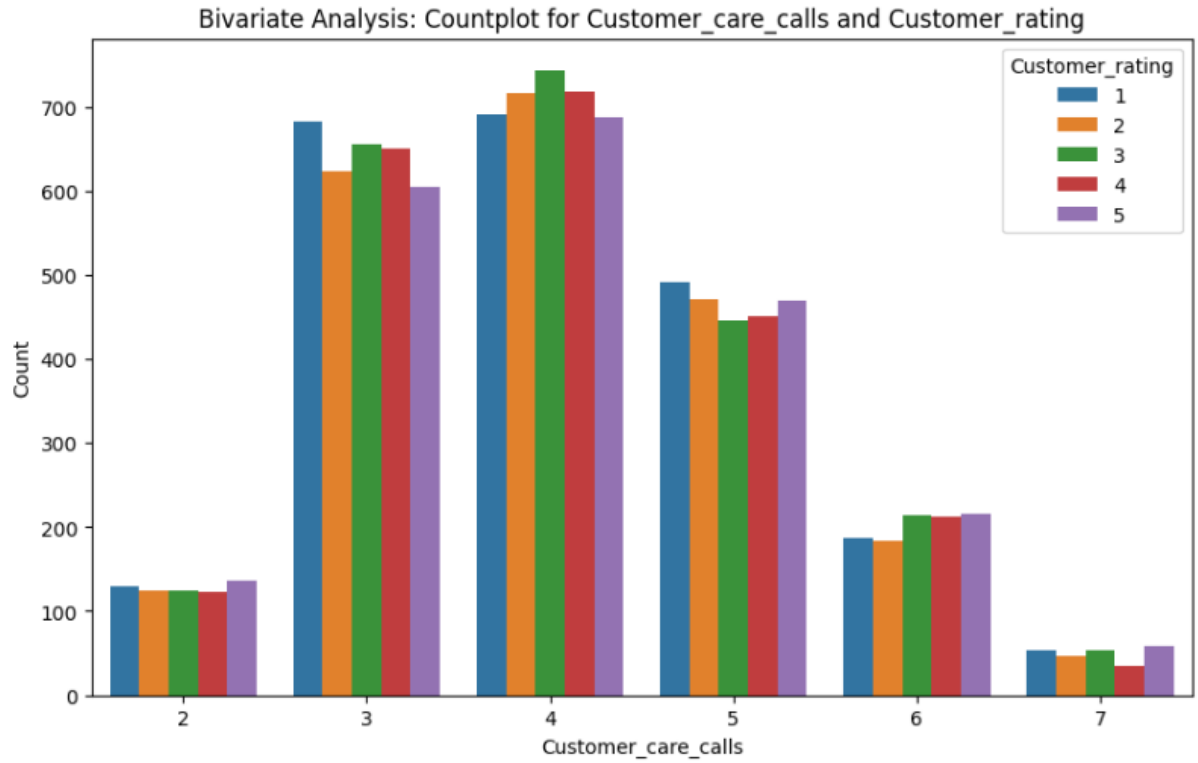
	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	1
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	
mean	5500.00000	1.833167	0.998454	4.054459	2.990545	210.196836	3.567597	0.604600	
std	3175.28214	1.343823	0.567099	1.141490	1.413603	48.063272	1.522860	0.641464	
min	1.00000	0.000000	0.000000	2.000000	1.000000	96.000000	2.000000	0.000000	
25%	2750.50000	1.000000	1.000000	3.000000	2.000000	169.000000	3.000000	0.000000	
50%	5500.00000	1.000000	1.000000	4.000000	3.000000	214.000000	3.000000	1.000000	
75%	8249.50000	3.000000	1.000000	5.000000	4.000000	251.000000	4.000000	1.000000	
max	10999.00000	4.000000	2.000000	7.000000	5.000000	310.000000	10.000000	2.000000	

```
numerical_columns = data.select_dtypes(include=['number']).columns[3:]  # Exclude the first 3 columns
num_cols = len(numerical_columns)
num_rows = math.ceil(num_cols / 3)

plt.figure(figsize=(15, 5 * num_rows))  # Adjust the figure size based on the number of rows

# Iterate through each numerical column
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(num_rows, 3, i)  # Adjust subplot layout based on the number of rows
    sns.histplot(data[column], bins=20, kde=True)
    plt.title(f'Univariate Analysis: {column}')
    plt.xlabel(column)
    plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

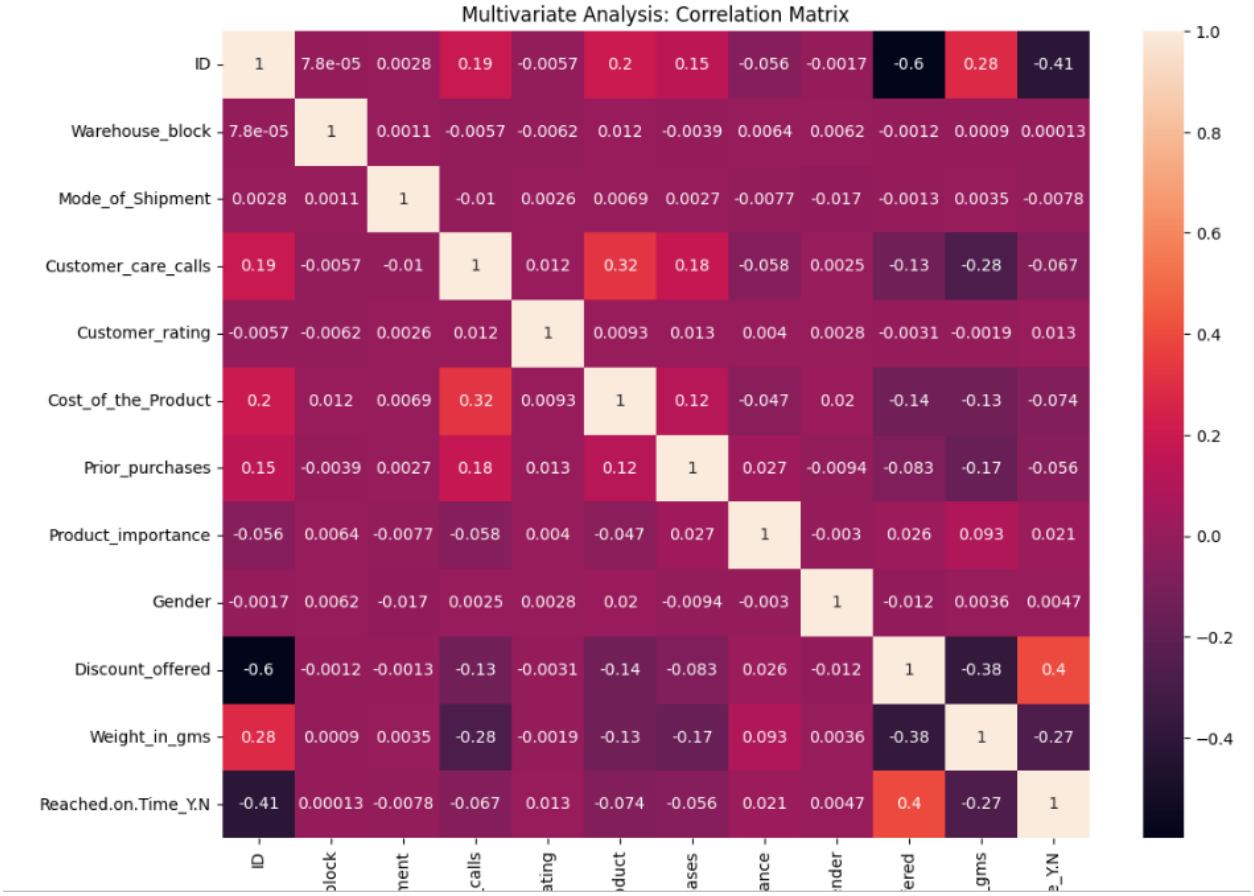



Univariate Analysis: Prior_purchases
Prior_purchases
Univariate Analysis: Product_importance
Product_importance
Univariate Analysis: Gender
Gender
Univariate Analysis: Discount_offered
Discount_offered
Univariate Analysis: Weight_in_gms
Weight_in_gms
Univariate Analysis: Reached.on.Time_Y.N
Reached.on.Time_Y.N
Count

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Customer_care_calls', hue='Customer_rating', data=data)
plt.title('Bivariate Analysis: Countplot for Customer_care_calls and Customer_rating')
plt.xlabel('Customer_care_calls')
plt.ylabel('Count')
plt.legend(title='Customer_rating')
plt.show()
```



```
: # Multivariate analysis using heatmap for correlation matrix
correlation_matrix = data.corr()
plt.figure(figsize=(12, 9))
sns.heatmap(correlation_matrix,annot=True)#, cmap='coolwarm', fmt='.2f')
plt.title('Multivariate Analysis: Correlation Matrix')
plt.show()
```



```
: lg,lcv,xgb,rg,knn,rf,svc=models_eval_mm(x_train_normalized,y_train,x_test_normalized,y_test
```

```
--Logistic Regression
Train Score: 0.6416638254347085
Test Score: 0.639090909090909

--Logistic Regression CV
Train Score: 0.6417774747130356
Test Score: 0.639090909090909

--XGBoost
Train Score: 0.9060120468235027
Test Score: 0.64

--Ridge Classifier
Train Score: 0.650642118422548
Test Scores 0.6522727272727272

-KNN
Train Score: 0.7789521536538243
Test Score: 0.635

Random Forest
Train Score: 1.0
Test Score: 0.6568181818181819

--SM classifier
Train Score: 0.6903057165586999
Test Score: 0.6586363636363637
```

```
xgb.predict(x_test_normalized[0].reshape(1,-1))
```

```
array([1])
```

```
rg.predict(x_test_normalized[0].reshape(1,-1))
```

```
array([1], dtype=int64)
```

```
knn.predict(x_test_normalized[0].reshape(1,-1))
```

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

:
```
model_list = {
    'logistic regression':lg,
    'logistic regression CV':lcv,
    'XGBoost':xgb,
    'Ridge classifier':rg,
    'KNN': knn,
    'Random Forest':rf,
    'Support Vector Classifier': svc
}

model_eval_info = []

for i in model_list.keys():
    model_eval_info.append(eval(i,model_list[i]))

model_eval_info = pd.DataFrame(model_eval_info, columns=['Name', 'Accuracy', 'f1_score', 'Recall', 'Precision'])
model_eval_info.to_csv('model_eval.csv')
model_eval_info
```

:

	Name	Accuracy	f1_score	Recall	Precision
0	logistic regression	63.91	69.06	67.94	70.21
1	logistic regression CV	63.91	69.22	68.48	69.98
2	XGBoost	64.00	68.55	66.18	71.09
3	Ridge classifier	65.23	70.47	70.02	70.94
4	KNN	63.50	68.50	66.95	70.12
5	Random Forest	65.68	68.69	63.50	74.80
6	Support Vector Classifier	65.86	67.08	58.67	78.30

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from math import sqrt

for i in model_list.keys():
    y_pred = model_list[i].predict(x_test_normalized)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = sqrt(mse)
    r2 = r2_score(y_test, y_pred)
    print(f"\n******{i}*******")
    print(f"MAE: {mae}")
    print(f"MSE: {mse}")
    print(f"RMSE: {rmse}")
    print(f"R2 Score: {r2}")
```

```
******logistic regression*******
MAE: 0.3609090909090909
MSE: 0.3609090909090909
RMSE: 0.6007570980929737
R2 Score: -0.49505641980718695

******logistic regression CV*******
MAE: 0.3609090909090909
MSE: 0.3609090909090909
RMSE: 0.6007570980929737
R2 Score: -0.49505641980718695

******XGBoost*******
MAE: 0.36
MSE: 0.36
RMSE: 0.6
R2 Score: -0.4912905346187557
```

```
******Ridge classifier*******
MAE: 0.3477272727272727
MSE: 0.3477272727272727
RMSE: 0.5896840448301723
R2 Score: -0.44045108457493454

******KNN*******
MAE: 0.365
MSE: 0.365
RMSE: 0.6041522986797286
R2 Score: -0.5120029031551272

******Random Forest*******
MAE: 0.3431818181818182
MSE: 0.3431818181818182
RMSE: 0.5858172225035879
R2 Score: -0.42162165863277856

******Support Vector Classifier*******
MAE: 0.34136363636363637
MSE: 0.34136363636363637
RMSE: 0.5842633279298268
R2 Score: -0.4140898882559161
```

:
```python
from sklearn.metrics import confusion_matrix

# Assuming y_true and y_pred are your actual and predicted classification labels
conf_matrix = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[684 212]
 [539 765]]
```

```python
from sklearn.metrics import classification_report

class_report = classification_report(y_test, y_pred)

print("\nClassification Report:")
print(class_report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.76      0.65       896
           1       0.78      0.59      0.67      1304

    accuracy                           0.66      2200
   macro avg       0.67      0.68      0.66      2200
weighted avg       0.69      0.66      0.66      2200
```

```
xgb.get_params() ##
```

```
{'objective': 'binary:logistic',
 'base_score': None,
 'booster': None,
 'callbacks': None,
 'colsample_bylevel': None,
 'colsample_bynode': None,
 'colsample_bytree': None,
 'device': None,
 'early_stopping_rounds': None,
 'enable_categorical': False,
 'eval_metric': None,
 'feature_types': None,
 'gamma': None,
 'grow_policy': None,
 'importance_type': None,
 'interaction_constraints': None,
 'learning_rate': None,
 'max_bin': None,
 'max_cat_threshold': None,
 'max_cat_to_onehot': None,
 'max_delta_step': None,
 'max_depth': None,
 'max_leaves': None,
 'min_child_weight': None,
 'missing': nan,
 'monotone_constraints': None,
 'multi_strategy': None,
 'n_estimators': None,
 'n_jobs': None,
 'num_parallel_tree': None,
 'random_state': 1234,
 'reg_alpha': None,
 'reg_lambda': None,
 'sampling_method': None,
 'scale_pos_weight': None,
 'subsample': None,
 'tree_method': None,
 'validate_parameters': None,
 'verbosity': None}
```

```
rf.get_params()
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 1234,
 'verbose': 0,
 'warm_start': False}
```

```
lcv.get_params()
```

```
{'Cs': 10,
 'class_weight': None,
 'cv': None,
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1.0,
 'l1_ratios': None,
 'max_iter': 100,
 'multi_class': 'auto',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': 1234,
 'refit': True,
 'scoring': None,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'verbose': 0}
```

```
svc = svm.SVC(random_state=1234)

params = {
    'kernel': ['poly', 'rbf'],
    'C': [ 10, 13],
    'gamma': [4,5],
    'tol':[1e-1,1e-2,1e-3] }

fitmodel = GridSearchCV(svc, param_grid=params, cv=5, refit=True, scoring="accuracy", n_jobs=-1, verbose=3)
fitmodel.fit(x_train_normalized, y_train)
print(fitmodel.best_estimator_, fitmodel.best_params_, fitmodel.best_score_)

# SVC(C=10, gamma-1, random_state=1234) ('C':10, 'gamma':1, 'kernel':'rbf') 0.6657575326890279
# SVC(C=6, gamma-2, random_state=1234) {'C':6, 'gamma':2, 'kernel': 'rbf') 0.6659845470050132
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
SVC(C=13, gamma=5, kernel='poly', random_state=1234, tol=0.01) {'C': 13, 'gamma': 5, 'kernel': 'poly', 'tol': 0.01} 0.657231833
6864955
```

HYPERPARAMETER OPTIMISATION FOR XGBOOST

```
params = {
    'min_child_weight': [10,20],
    'gamma': [1.5, 2.0, 2.5],
    'colsample_bytree': [0.6, 0.8, 0.9],
    'max_depth': [4,5,6] }

xgb= XGBClassifier(learning_rate=0.5, n_estimators=100, objective='binary:logistic', nthread=3)
fitmodel = GridSearchCV(xgb, param_grid=params, cv=5, refit=True, scoring="accuracy", n_jobs=2, verbose=3)
fitmodel.fit(x_train_normalized, y_train)
print(fitmodel.best_estimator_, fitmodel.best_params_, fitmodel.best_score_)

# {'colsample_bytree': 0.8, 'gamma: 2.0, 'max_depth': 4, 'min_child_weight: 10) 0.6688713628611298
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=2.5, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.5, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=4, max_leaves=None,
              min_child_weight=10, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None, nthread=3,
              num_parallel_tree=None, ...) {'colsample_bytree': 0.8, 'gamma': 2.5, 'max_depth': 4, 'min_child_weight': 10} 0.68
17814874153704
```

HYPERPARAMETER OPTIMISATION FOR LOGISTIC REGRESSION

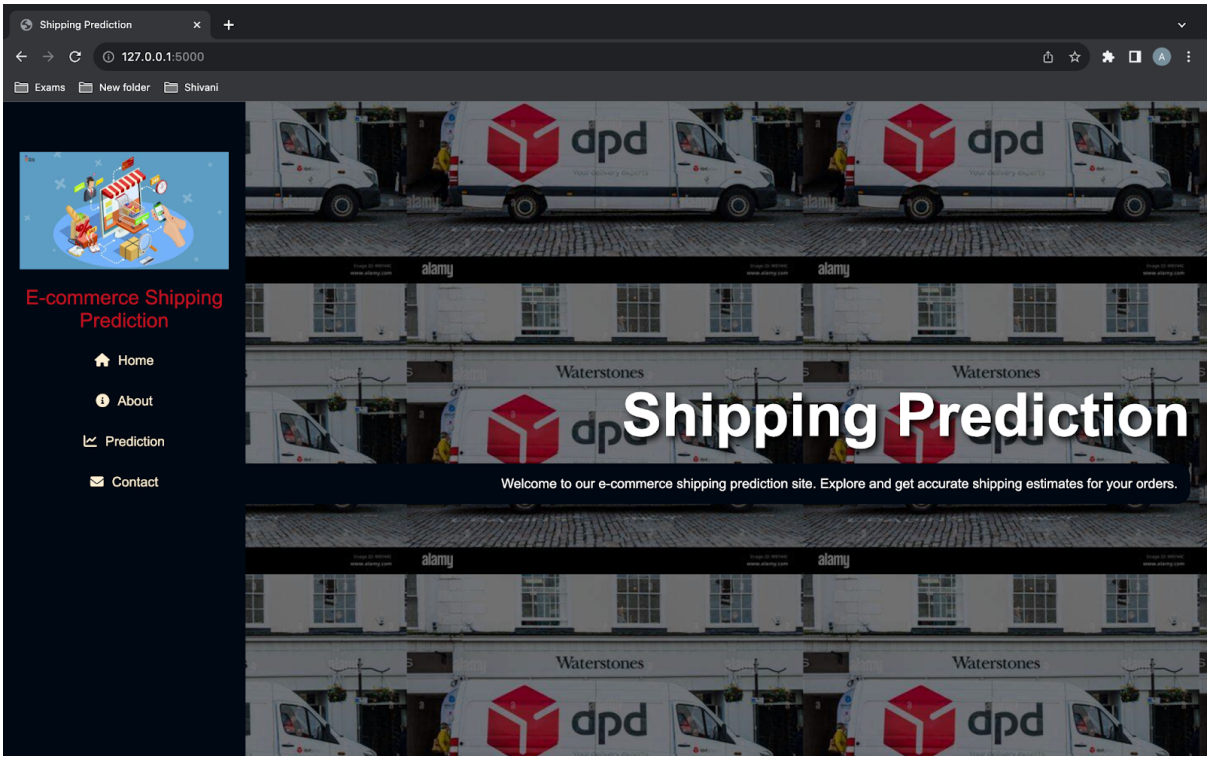
```
# Plug in appropriate max_depth and random_state parameters

lg= LogisticRegressionCV(n_jobs=-1, random_state= 1234)

lg_param_grid = {
    'Cs': [6,8,10,15,20],
    'max_iter': [60,80,100]}

lg_cv= GridSearchCV(lg,lg_param_grid, cv=5, scoring="accuracy", n_jobs=-1, verbose=3)
lg_cv.fit(x_train_normalized,y_train)
print("Best Score: " + str(lg_cv.best_score_))
print("Best Parameters:" + str(lg_cv.best_params_))

# Best Score: 0.633821644529433
# Best Parameters: {'Cs': 10, 'max_iter': 60}
```

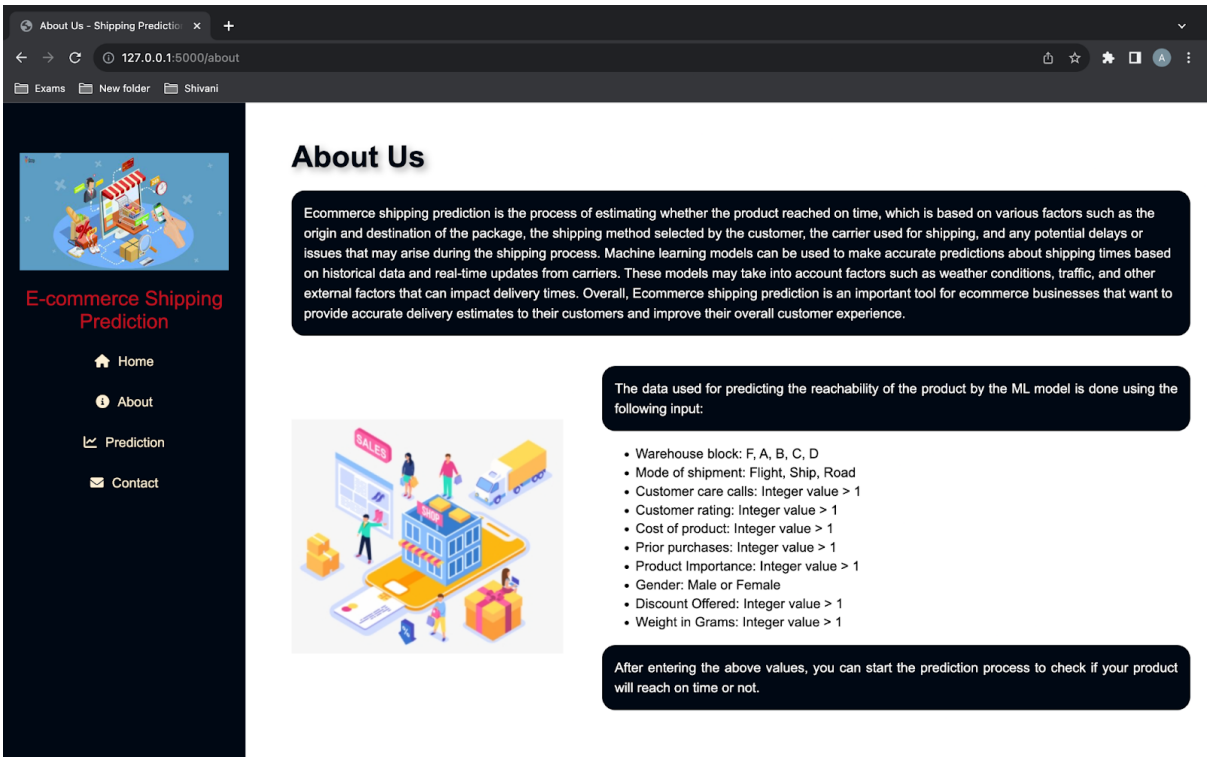
```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
Best Score: 0.6430276758488811
Best Parameters:{'Cs': 10, 'max_iter': 60}
```

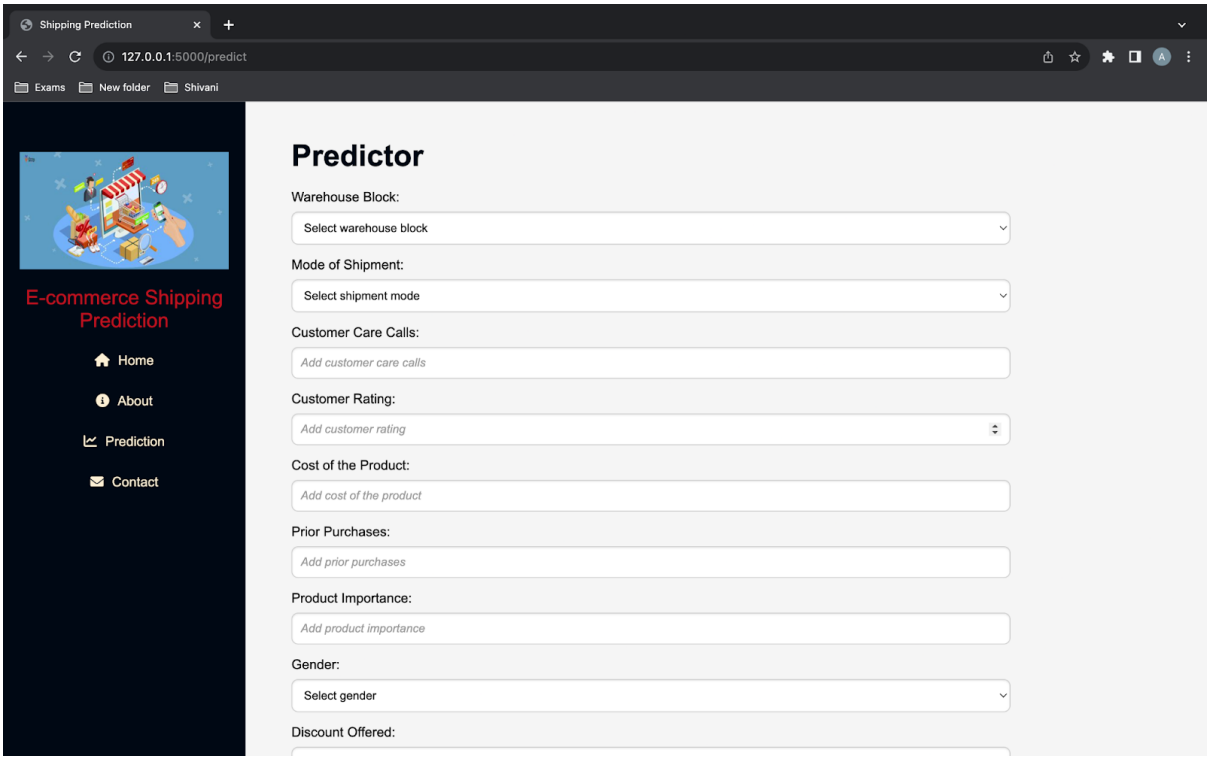
HYPERPARAMETER OPTIMISATION FOR RANDOM FOREST

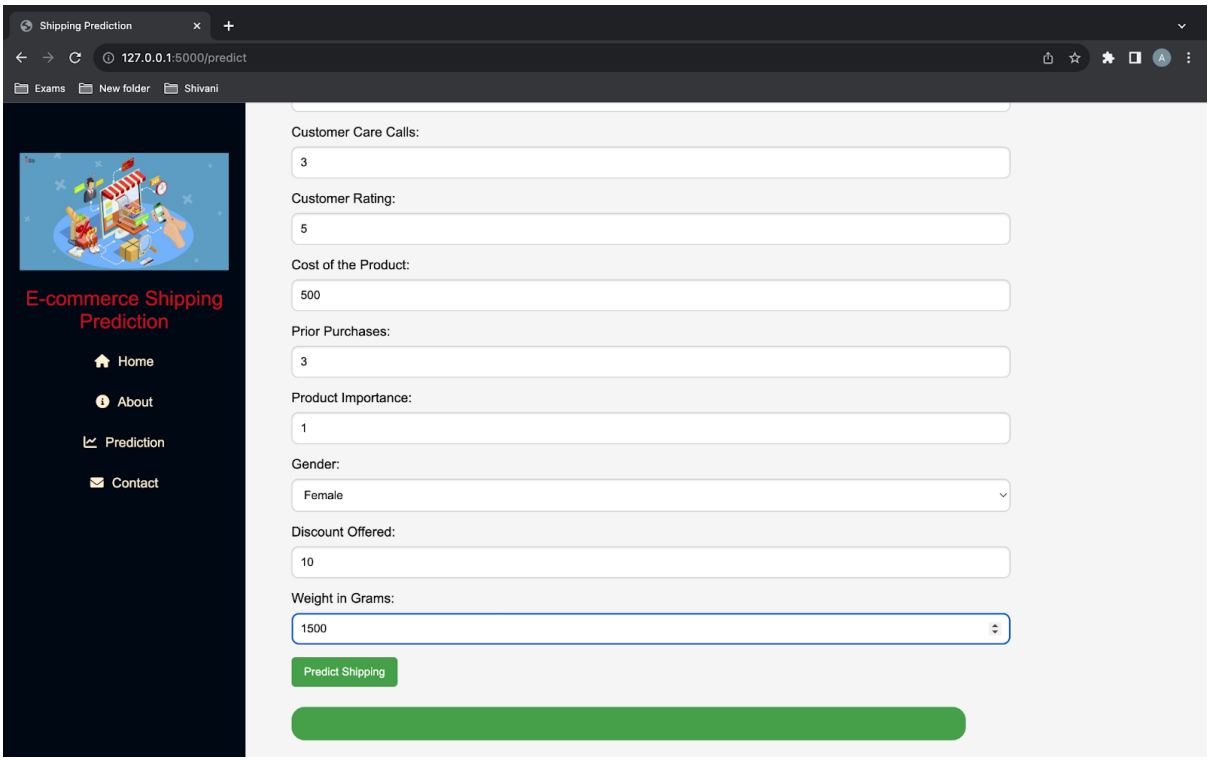
```
# Plug in appropriate max_depth and random_state parameters

rf=RandomForestClassifier()

rf_param_grid = {
    'n_estimators': [200,300,500],
    'criterion': ['entropy', 'gini'],
    'max_depth': [7,8,60,80,100],
    'max_features': ['auto', 'sqrt', 'log2']}

rf_cv= GridSearchCV(rf, rf_param_grid, cv=7, scoring="accuracy", n_jobs=-1, verbose=3)
rf_cv.fit(x_train_normalized,y_train)
print("Best Score: " + str(rf_cv.best_score_))
print("Best Parameters: " + str(rf_cv.best_params_))

#Best Score: 0.6849642004773271
#Best Parameters: {'criterion': 'entropy', 'max_depth': 7, 'max features': 'auto', 'n_estimators: 200)
```

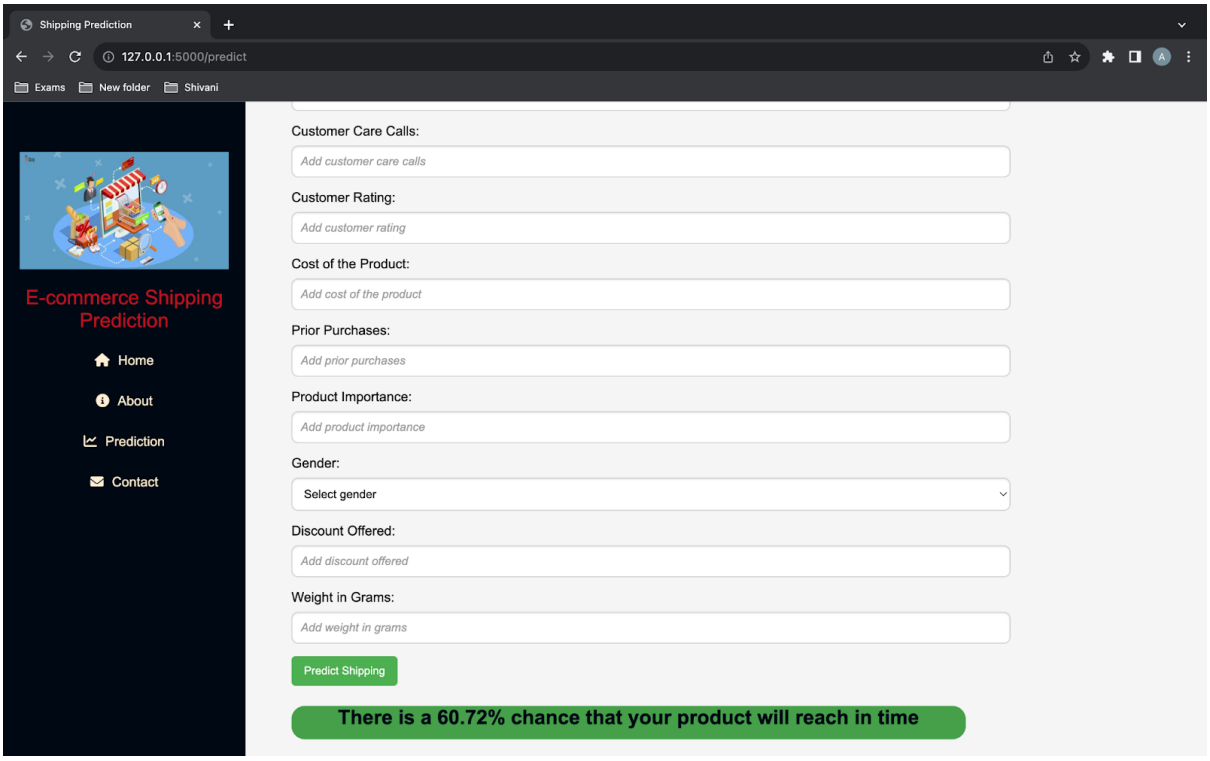
```
Fitting 7 folds for each of 90 candidates, totalling 630 fits
Best Score: 0.6839413569723831
Best Parameters: {'criterion': 'gini', 'max_depth': 7, 'max_features': 'log2', 'n_estimators': 200}
```

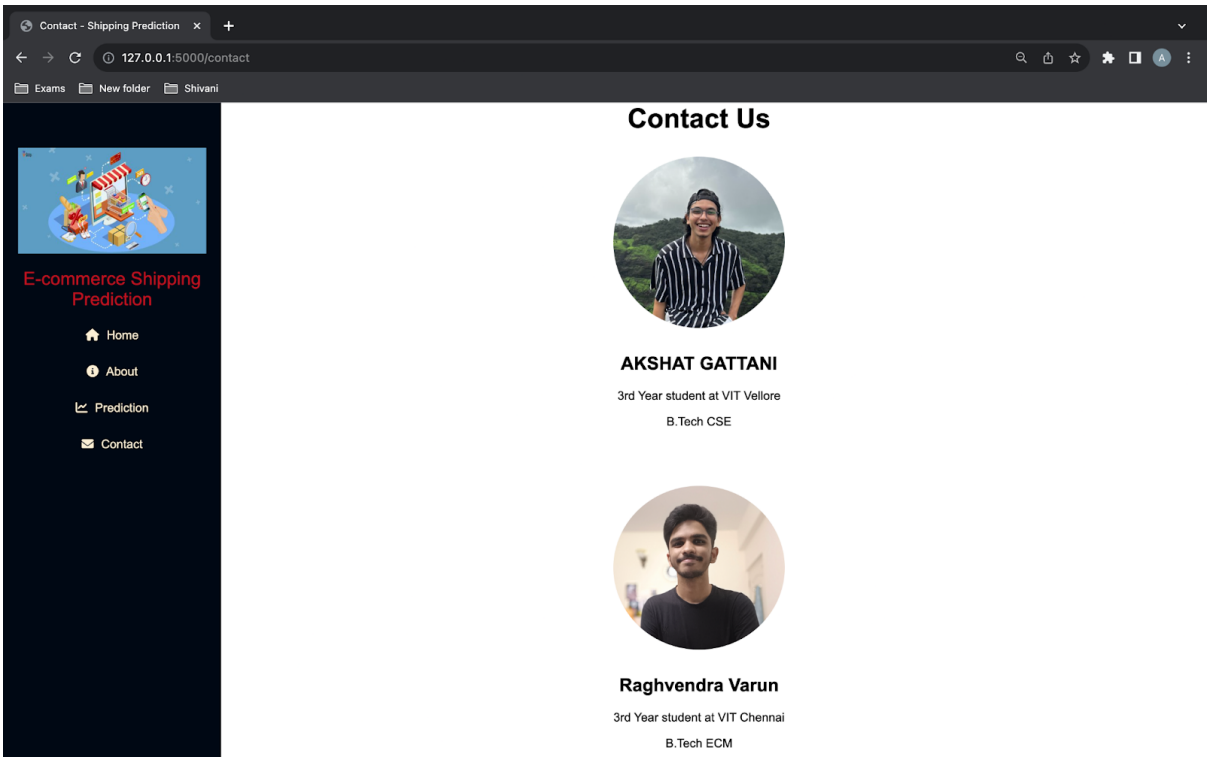


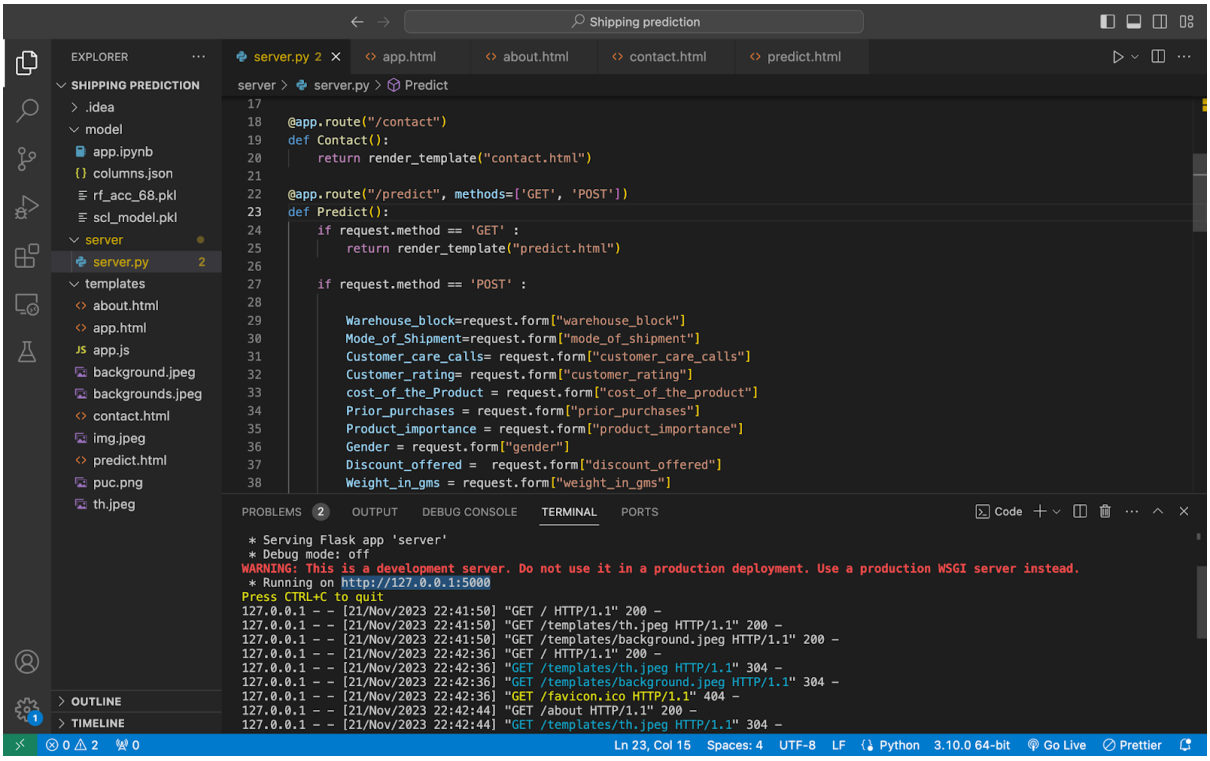

About Us - Shipping Prediction
127.0.0.1:5000/about
Exams
New folder
Shivani
E-commerce Shipping Prediction
Home
About
Prediction
Contact
About Us
Ecommerce shipping prediction is the process of estimating whether the product reached on time, which is based on various factors such as the origin and destination of the package, the shipping method selected by the customer, the carrier used for shipping, and any potential delays or issues that may arise during the shipping process. Machine learning models can be used to make accurate predictions about shipping times based on historical data and real-time updates from carriers. These models may take into account factors such as weather conditions, traffic, and other external factors that can impact delivery times. Overall, Ecommerce shipping prediction is an important tool for ecommerce businesses that want to provide accurate delivery estimates to their customers and improve their overall customer experience.
SALES
The data used for predicting the reachability of the product by the ML model is done using the following input:
Warehouse block: F, A, B, C, D
Mode of shipment: Flight, Ship, Road
Customer care calls: Integer value > 1
Customer rating: Integer value > 1
Cost of product: Integer value > 1
Prior purchases: Integer value > 1
Product Importance: Integer value > 1
Gender: Male or Female
Discount Offered: Integer value > 1
Weight in Grams: Integer value > 1
After entering the above values, you can start the prediction process to check if your product will reach on time or not.


Shipping Prediction
127.0.0.1:5000/predict
Exams
New folder
Shivani
E-commerce Shipping Prediction
Home
About
Prediction
Contact
Predictor
Warehouse Block:
Select warehouse block
Mode of Shipment:
Select shipment mode
Customer Care Calls:
Add customer care calls
Customer Rating:
Add customer rating
Cost of the Product:
Add cost of the product
Prior Purchases:
Add prior purchases
Product Importance:
Add product importance
Gender:
Select gender
Discount Offered:


Shipping Prediction
127.0.0.1:5000/predict
Exams
New folder
Shivani
E-commerce Shipping Prediction
Home
About
Prediction
Contact
Customer Care Calls:
3
Customer Rating:
5
Cost of the Product:
500
Prior Purchases:
3
Product Importance:
1
Gender:
Female
Discount Offered:
10
Weight in Grams:
1500
Predict Shipping


Shipping Prediction
127.0.0.1:5000/predict
Exams
New folder
Shivani
E-commerce Shipping Prediction
Home
About
Prediction
Contact
Customer Care Calls:
Add customer care calls
Customer Rating:
Add customer rating
Cost of the Product:
Add cost of the product
Prior Purchases:
Add prior purchases
Product Importance:
Add product importance
Gender:
Select gender
Discount Offered:
Add discount offered
Weight in Grams:
Add weight in grams
Predict Shipping
There is a 60.72% chance that your product will reach in time


Contact - Shipping Prediction
127.0.0.1:5000/contact
Exams
New folder
Shivani
E-commerce Shipping Prediction
Home
About
Prediction
Contact
Contact Us
AKSHAT GATTANI
3rd Year student at VIT Vellore
B.Tech CSE
Raghvendra Varun
3rd Year student at VIT Chennai
B.Tech ECM


Shipping prediction
EXPLORER
SHIPPING PREDICTION
.idea
model
app.ipynb
columns.json
rf_acc_68.pkl
scl_model.pkl
server
server.py
templates
about.html
app.html
app.js
background.jpeg
backgrounds.jpeg
contact.html
img.jpeg
predict.html
puc.png
th.jpeg
OUTLINE
TIMELINE
server.py 2
app.html
about.html
contact.html
predict.html
server > server.py > Predict
@app.route("/contact")
def Contact():
return render_template("contact.html")
@app.route("/predict", methods=['GET', 'POST'])
def Predict():
if request.method == 'GET' :
return render_template("predict.html")
if request.method == 'POST' :
Warehouse_block=request.form["warehouse_block"]
Mode_of_Shipment=request.form["mode_of_shipment"]
Customer_care_calls= request.form["customer_care_calls"]
Customer_rating= request.form["customer_rating"]
cost_of_the_Product = request.form["cost_of_the_product"]
Prior_purchases = request.form["prior_purchases"]
Product_importance = request.form["product_importance"]
Gender = request.form["gender"]
Discount_offered = request.form["discount_offered"]
Weight_in_gms = request.form["weight_in_gms"]
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Code
* Serving Flask app 'server'
* Debug mode: off
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
127.0.0.1 - - [21/Nov/2023 22:41:50] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Nov/2023 22:41:50] "GET /templates/th.jpeg HTTP/1.1" 200 -
127.0.0.1 - - [21/Nov/2023 22:41:50] "GET /templates/background.jpeg HTTP/1.1" 200 -
127.0.0.1 - - [21/Nov/2023 22:42:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Nov/2023 22:42:36] "GET /templates/th.jpeg HTTP/1.1" 304 -
127.0.0.1 - - [21/Nov/2023 22:42:36] "GET /templates/background.jpeg HTTP/1.1" 304 -
127.0.0.1 - - [21/Nov/2023 22:42:36] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [21/Nov/2023 22:42:44] "GET /about HTTP/1.1" 200 -
127.0.0.1 - - [21/Nov/2023 22:42:44] "GET /templates/th.jpeg HTTP/1.1" 304 -
Ln 23, Col 15
Spaces: 4
UTF-8
LF
Python
3.10.0 64-bit
Go Live
Prettier

10. ADVANTAGES & DISADVANTAGES

Advantages

- Accurate shipment forecasts: The incorporation of machine learning models allows for accurate shipment time forecasts, which improves customer satisfaction by setting reasonable expectations.

- User-Friendly Interface: The frontend design provides a user-friendly experience with a well-organised layout, making it simple for users to browse through the website's various sections.

- Transparent Team Communication: The Contact page, which includes information about team members, fosters confidence by establishing transparency and facilitating communication between users and the development team.

- Predictive Analytics: Using machine learning for predictions enables the study of historical data as well as real-time factors, improving the system's ability to give credible estimations.

- Responsive Design: The website is responsive, ensuring a consistent experience across multiple devices such as PCs, tablets, and smartphones.

Disadvantages

- Dependence on Machine Learning Model: Prediction accuracy is dependent on the performance and relevance of the machine learning model. Outdated or faulty models can result in untrustworthy estimations.

- Limited Data Management: The inability to store and manage user data is hampered by the lack of a comprehensive database structure. This may have an impact on features like order history and personalised suggestions.

- Potential Security Concerns: Depending on the sensitivity of user data and transactions, the website may require additional security measures to protect against potential threats while maintaining user privacy and integrity.

- Users May Experience a Learning Curve: Users who are new with the system may encounter a learning curve, particularly when interpreting and inputting the parameters for shipping estimates.

- Maintenance Issues: Regular updates and maintenance are critical, especially if the machine learning model changes or more functionality are required. Inaccuracy and functionality may suffer if the system is not kept up to date.

11. CONCLUSION

The E-commerce Shipping Prediction project seeks to provide consumers with a trustworthy tool for assessing the reachability of their products within defined periods. The incorporation of machine learning models improves the accuracy of shipping estimates, resulting in a better customer experience. A well-rounded platform is created by the project's user-friendly layout, responsive design, and transparent communication via the Contact page.

Various topics were examined during the development process, including the project overview, existing difficulties in the literature, and the definition of the problem statement. To ensure the system's performance and user satisfaction, functional and non-functional requirements were analysed to guide the development process.

During the coding and solutioning process, crucial elements such as accurate prediction algorithms and a responsive design were implemented. While the website uses machine learning to anticipate shipment times, it also handles potential issues such as model updates and security precautions.

Despite the benefits, such as accurate forecasts and an easy-to-use interface, the project has issues related to model reliance, data administration, and potential security risks. Ongoing maintenance and updates are critical for addressing these issues and ensuring the system's dependability.

Finally, the E-commerce Shipping Prediction project aims to provide a useful tool for e-commerce customers by balancing the benefits of precise predictions with the requirement for continuing maintenance and system enhancements.

12. FUTURE SCOPE

The E-commerce Shipping Prediction project lays the groundwork for future enhancements and developments. Some potential future scope areas include:

- Real-time Data Integration: Improve prediction accuracy by incorporating real-time data from shipping companies, weather conditions, and other external factors that may affect delivery schedules.

- User Authentication and Personalization: Enable registered users to save and track their shipments by implementing user authentication. Order history, preferences, and tailored shipment predictions are examples of personalization capabilities.

- Expanded Machine Learning Models: Investigate and implement more complex machine learning models or algorithms to increase prediction accuracy over time. Consider using deep learning techniques to make more accurate forecasts.

- Integration with E-commerce Platforms: Investigate interfaces with popular e-commerce platforms in order to provide smooth shipping prediction services to a larger audience.

- Predictive Analytics Dashboard: Create a complete analytics dashboard for managers to monitor system utilisation, user interactions, and model performance. Make informed decisions on system enhancements using this data.

- Feedback Mechanism: Implement a feedback tool so that users can provide comments on the accuracy of shipment predictions. Use this feedback to improve the machine learning models and the overall performance of the system.

13. APPENDIX

GitHub Link - https://github.com/smartinternz02/SI-GuidedProject-615260-1699443120

Drive Link -
https://drive.google.com/drive/folders/1Fb8OCnVqqP8tGo3rvEIH4J3_KujyeoZg?usp=sharing

Project Demo Link -
https://drive.google.com/drive/folders/1xyK07GyO5WTL27EvaGTnw9sZaKNUyf-3?usp=sharing