VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA" , BELAGAVI-590018, KARNATAKA

Project Report
On

"A Smart Recommendation System for Carrier-Shipper Matching using Multilabel Classification"

Submitted in the partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

Submitted By

Aastha Prasad	**1VA20CS001**
S G Dhanush Kumar	**1VA20CS043**
Sharan S	**1VA20CS049**
Danush V S	**1VA20CS069**

Under the Guidance of

Dr. Tejashwini N	**Dr. Varun E**
Associate Professor	**Associate Professor**
Dept. of CS&E	**Dept. of CS&E**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SAI VIDYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of Karnataka | Approved by AICTE, New Delhi)

Accredited by NBA, New Delhi (CSE, ISE, ECE, MECH, CIVIL), NAAC 'A' Grade

RAJANUKUNTE, BENGALURU – 560 064

2023-24

SAI VIDYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of Karnataka | Approved by AICTE, New Delhi)

Accredited by NBA New Delhi (CSE, ECE, ISE, MECH, CIVIL), NAAC-"A" Grade

Rajanukunte, Bengaluru- 560 064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Project work entitled *"A Smart Recommendation System for Carrier-Shipper Matching using Multilabel Classification"* carried out by **Ms. Aastha Prasad (1VA20CS001), Mr. S G Dhanush Kumar (1VA20CS043), Mr. Sharan S (1VA20CS049), Mr. Danush V S (1VA20CS069)** a bonafide students of **SAI VIDYA INSTITUTE OF TECHNOLOGY**, Bengaluru, in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**, Belagavi during the year **2023-24.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Dr. Tejashwini N	**Dr. Varun E**	**Dr. Shantakumar B Patil**	**Dr. M S Ganesha Prasad**
Associate Professor	Associate Professor	Professor & Head	Principal
Dept. of CSE, SVIT	Dept. of CSE, SVIT	Dept. of CSE, SVIT	

External Viva:

Name	Signature
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENT

The completion of project brings with and sense of satisfaction, but it is never completed without thanking the persons who are all responsible for its successful completion. First and foremost, I wish to express our deep sincere feelings of gratitude to my Institution, **Sai Vidya Institute of Technology**, for providing me an opportunity to do our education.

I would like to thank the **Management** and **Prof. M R Holla,** Director, Sai Vidya Institute of Technology for providing the facilities.

I extend my deep sense of sincere gratitude to **Dr. M S Ganesha Prasad**, Principal, Sai Vidya Institute of Technology, Bengaluru, for having permitted to carry out the project work on **"*A Smart Recommendation System for Carrier-Shipper Matching using Multilabel Classification*"** successfully**.**

I am thankful to **Dr. A M Padma Reddy**, Director (A), Professor and Dean (Student affairs), Department of Computer Science and Engineering, Sai Vidya Institute of Technology, for his constant support and motivation.

I express my heartfelt sincere gratitude to **Dr. Shantakumar B Patil**, Professor and HOD, Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for his valuable suggestions and support.

I express my sincere gratitude to **Dr. Tejashwini N**, Associate Professor, Project Coordinator and Project Guide, Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for her constant support and coordination.

I express my sincere gratitude to **Dr.Varun E**, Associate Professor, Project Guide, Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for her constant support and coordination.

Finally, I would like to thank all the Teaching, Technical faculty and supporting staff members of Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for their support.

Aastha Prasad	1VA20CS001
S G Dhanush Kumar	1VA20CS043
Sharan S	1VA20CS049
Danush V S	1VA20CS069

ABSTRACT

This project explores an innovative approach to enhance logistics efficiency. Leveraging Support Vector Machines (SVM), Random Forest, and Decision Trees as base models, with XGBoost as the meta-model, the study aims to develop a powerful recommendation system for optimal carrier selections in shipments. The project begins with the collection and preprocessing of historical shipment data, encompassing key variables such as weight, distance, and delivery time.

By employing a hybrid model, the project seeks to capitalize on the strengths of individual models, utilizing SVM, Random Forest, and Decision Trees to discern diverse patterns within the data. The meta-model, XGBoost, is then employed to consolidate and refine these outputs, boosting predictive accuracy and generalizability. Anticipated outcomes include a more robust and accurate recommendation system, poised to significantly impact logistics operations by optimizing carrier selections. This project addresses the contemporary need for streamlined supply chain management, offering a potential avenue for cost savings and heightened customer satisfaction through improved decision-making in carrier shipments.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

"A Smart Recommendation System for Carrier-Shipper Matching using Multilabel Classification" is a project designed to enhance the efficiency of the logistics and transportation industry by employing advanced machine learning techniques. The primary focus of the project is to develop a recommendation system that intelligently matches carriers with shippers based on multiple criteria. The system utilizes multilabel classification, a machine learning approach that allows the assignment of multiple labels to each instance, enabling a more nuanced representation of the complex relationships involved in carrier-shipper matching. This approach takes into account various factors such as cargo type, delivery time preferences, geographical constraints, and carrier capabilities.

Key components of the project may include data preprocessing to handle diverse and structured information, feature engineering to extract relevant features for classification, and the implementation of a robust multilabel classification model. The system aims to learn patterns and relationships from historical data to make accurate predictions and provide optimal recommendations for carrier-shipper pairings. The overarching goal is to streamline the logistics process, improve resource utilization, and enhance overall supply chain efficiency. Such a smart recommendation system has the potential to revolutionize how carriers and shippers are matched, ultimately leading to cost savings, reduced transit times, and an overall improvement in the effectiveness of freight transportation logistics.

1.1 Problem Statement

The logistics and supply chain industry faces a persistent challenge in optimizing carrier selections for shipments. Current methods often lack precision in providing timely and accurate recommendations, leading to increased operational costs, inefficiencies, and suboptimal resource utilization. The absence of a robust and adaptive recommendation system hinders the industry's ability to navigate the intricacies of shipment variables, limiting the realization of cost-effective and customer-centric logistics solutions.

Individual machine learning models, when applied in isolation, may overlook the nuanced patterns inherent in diverse datasets, diminishing the system's predictive capabilities. There is a pressing need for an integrated solution that can discern complex patterns in historical shipment data, offering a more holistic approach to carrier recommendations. Developing a hybrid model that considers multiple factors influencing carrier selections could potentially enhance predictive accuracy, improve decision-making processes, and streamline recommendations for various shipment scenarios.

This project aims to address the existing gaps in carrier selection processes, providing a comprehensive and adaptable solution to enhance the efficiency of logistics operations. By doing so, it contributes to the overall optimization of supply chain management, fostering a more responsive and cost-effective approach to carrier shipments.

1.2 Solution for the Problem

- **Integrated Data Processing**

  Implement a robust data preprocessing pipeline to ensure the quality and relevance of input features. This step involves cleaning, transforming, and normalizing historical shipment data, considering factors such as weight, distance, delivery time, and other relevant variables.

- **Feature Engineering**

  Identify and extract meaningful features from the dataset that are crucial for making informed carrier recommendations. This step involves a comprehensive analysis of the historical data to uncover hidden patterns and relationships.

- **Hybrid Model Architecture**

  Develop a hybrid machine learning model that combines the strengths of various algorithms without explicitly mentioning them. This model should be capable of capturing diverse patterns within the data, leveraging the collective intelligence of different techniques to enhance predictive accuracy.

- **Validation and Optimization**

  Implement thorough validation procedures to assess the performance of the developed model. Fine-tune the parameters and optimize the system iteratively to ensure robustness and reliability in diverse shipment scenarios.

1.3 Existing Technique

Several existing systems attempt to solve the problem of carrier shipment recommendation, utilizing various machine learning techniques. Here's a brief overview of some notable approaches.

a. Traditional Machine Learning Models

Linear Regression: This approach models the relationship between shipment features and carrier costs. While simple to implement, it may not capture complex non-linear relationships.

Support Vector Machines (SVM): SVMs are powerful classifiers that can effectively separate different carrier options based on shipment characteristics. However, they can be computationally expensive and require careful parameter tuning.

Random Forest: This ensemble model combines multiple decision trees, offering a robust and versatile approach to carrier recommendation. However, interpreting the model's predictions can be challenging.

Decision Trees: These rule-based models are easily interpretable and provide clear insights into the decision-making process. However, they can suffer from overfitting and may not generalize well to unseen data.

b. Neural Network-based Models

Multilayer Perceptrons (MLPs): These are simple neural networks with multiple layers of interconnected nodes. They can learn complex non-linear relationships, but require a large amount of data and careful training.

Long Short-Term Memory (LSTM) networks: LSTMs are well-suited for handling sequential data, such as historical shipment records. They can learn temporal patterns and predict future carrier costs more accurately.

Recurrent Neural Networks (RNNs): RNNs are another class of neural networks capable of processing sequential data. They can learn complex temporal dependencies between shipment features and carrier performance.

1.4 Proposed Technique

Data Collection

Acquire a comprehensive dataset containing historical shipment records. Gather relevant information such as shipment weight, distance, delivery time, carrier details, and any other pertinent variables influencing the shipment process.

Data Preprocessing

Cleanse and preprocess the collected data to ensure data quality. Handle missing values, outliers, and standardize numerical features. Categorize or encode categorical variables appropriately. Split the dataset into training and testing sets.

Feature Engineering

Conduct a detailed analysis of the dataset to identify significant features influencing carrier selections. Create new features or transform existing ones to enhance the model's ability to capture relevant patterns and relationships.

Model Training

Train individual machine learning models (SVM, Random Forest, Decision Trees) on the training dataset. Fine-tune hyperparameters to optimize model performance. Evaluate each model's performance using appropriate metrics.

Hybrid Model Development

Develop a hybrid model that integrates the outputs of the base models. Utilize an ensemble technique, without explicitly mentioning algorithms, to aggregate predictions and enhance overall model performance. Integrate XGBoost as the meta-model.

Model Serialization

Serialize the trained hybrid model for easy deployment. Save the model in a format compatible with Flask to facilitate seamless integration.

Flask Application Setup

Create a Flask application to serve as the interface for the model. Set up API endpoints to handle incoming requests for carrier recommendations. Define routes for data input, invoke the model, and format the model predictions for user-friendly responses.

User Interface Development

Build a user interface for the Flask application, allowing users to input shipment details. Design an intuitive interface that facilitates easy interaction and provides clear recommendations based on the hybrid model's predictions.

Integration Testing

Conduct integration testing to ensure the seamless functioning of the Flask application with the integrated hybrid model. Validate inputs, test model predictions, and address any issues related to the deployment setup.

1.5 OBJECTIVES

a. Acquire and preprocess a diverse dataset for historical shipment records. Standardize and encode variables to ensure data quality.
b. Train individual machine learning models (SVM, Random Forest, Decision Trees) on the prepared dataset. Develop a hybrid model that integrates these base models.
c. Integrate XGBoost as a meta-model to refine and consolidate base model predictions.
d. Create a Flask application with API endpoints to handle input data and serve recommendations. Design a user-friendly interface for seamless interaction.
e. Conduct rigorous testing to ensure the integrated system functions seamlessly. Deploy the system on a chosen platform, ensuring scalability and accessibility.

1.6 SCOPE OF THE PROJECT

- **Industry Relevance**

  This innovative project strategically positions itself to tackle the unique and pressing challenges prevalent within the logistics and supply chain sector. By focusing on the optimization of carrier selection processes, it aims to make a tangible impact on the industry's overall efficiency and operational dynamics.

- **Algorithm Flexibility**

  The project's strength lies in its inherent adaptability, allowing for the incorporation and seamless integration of a diverse array of machine learning algorithms. This inherent flexibility ensures that the developed model is not rigidly tied to specific methodologies, providing room for evolution with emerging technologies and advancements in algorithmic approaches.

- **Data Variety**

  Recognizing the multifaceted nature of shipment data, the project deliberately caters to diverse datasets. By considering a myriad of shipment variables, including but not limited to weight, distance, and delivery time, it strives to create a solution that is versatile and applicable across a spectrum of logistics scenarios.

- **User Interaction**

  The integration with Flask goes beyond mere functional implementation; it embraces a user-centric philosophy. Through the creation of API endpoints and the meticulous design of a user-friendly interface, the project aims to empower stakeholders. This intuitive system allows users to effortlessly input detailed shipment information and seamlessly receive clear and actionable carrier recommendations.

- **Scalability**

  Anticipating the potential for growth and increased demands, the project is meticulously designed to exhibit scalability. As the system encounters heightened data volumes and user interactions, it is poised to maintain optimal performance levels. This scalability feature ensures the sustained relevance and effectiveness of the recommendation system under varying usage conditions.

- **Real-time Recommendations**

  The strategic incorporation of Flask into the project architecture serves to facilitate real-time recommendations. This dynamic capability enables users to receive instant insights and make informed, data-driven decisions promptly. Such real-time responsiveness enhances the system's utility, particularly in the context of dynamic logistics environments where timely decision-making is of paramount importance.

1.7 Organization of report

Chapter 1: Introduction

This chapter includes a brief description about the project, problem statement, solution to the problem, existing system, proposed system, objectives, scope of the project for the recommendation System.

Chapter 2: Literature Survey

This chapter includes a brief introduction of different research papers for recommendation and its related topics.

Chapter 3: System Requirement Specification

This chapter includes the requirement analysis of the problem, which specifies the non-functional and functional requirements and hardware and software requirements.

Chapter 4: System Design

This chapter includes the explanation of the tools being used in the project modules and also about steps being carried out in designing the project.

Chapter 5: Implementation

This chapter provides a detailed description of how the proposed solution or methodology was executed, including the tools and techniques used, any challenges faced, and the results obtained.

Chapter 6: Testing and Results

This chapter outlines the testing methods used to evaluate the proposed solution, presents the results obtained, and discusses the significance of the results.

Chapter 7: Conclusions

This chapter specifies the prediction or the final output of the project model and also includes how our model will be useful or how it can be enhanced with better features in future.

CHAPTER 2

LITERATURE SURVEY

1. "Hybrid XGBoost-SVM Model for Multi-criteria Carrier Selection in E-commerce Logistics" by Wang et al. (2023)

Description: This paper proposes a hybrid model combining XGBoost and SVM for carrier selection in e-commerce logistics. XGBoost handles complex relationships between features and costs, while SVM helps select relevant features.

Advantages: Achieves high accuracy and interpretability. Offers feature importance insights.

Disadvantages: Requires careful parameter tuning for both models. May be computationally expensive.

2. "A Multi-objective Optimization Approach for Carrier Selection in Container Shipping using Random Forest and Deep Neural Network" by Chen et al. (2023)

Description: This paper proposes a hybrid model combining Random Forest and Deep Neural Network for multi-objective carrier selection. Random Forest handles categorical features, while Deep Neural Network learns complex non-linear relationships.

Advantages: Handles multi-objective optimization considering cost, transit time, and environmental impact. Offers interpretability with Random Forest feature importance.

Disadvantages: Requires extensive data pre-processing for Deep Neural Network. May be computationally expensive.

3. "Ensemble Learning Approach for Carrier Selection in Freight Transportation using Stacking and Voting" by Wu et al. (2022)

Description: This paper proposes an ensemble model combining different machine learning models using stacking and voting for carrier selection. This leverages the strengths of each model and improves overall accuracy.

Advantages: Reduces model variance and improves overall performance. Offers flexibility in choosing diverse base models.

Disadvantages: Requires careful selection and weighting of base models. Increased complexity compared to single models.

4. "Rule-based and Machine Learning Hybrid Model for Carrier Selection in Less-than-Truckload Transportation" by He et al. (2022)

Description: This paper combines rule-based and machine learning models for carrier selection in less-than-truckload transportation. This leverages the strengths of both approaches for improved accuracy and interpretability.

Advantages: Adapts to specific regulations and carrier requirements. Offers interpretability and control through rules.

Disadvantages: Requires manual creation and maintenance of rules. May lack flexibility to adapt to changing conditions.

5. "Case-based Reasoning with Reinforcement Learning for Dynamic Carrier Recommendation in Intermodal Transportation" by Li et al. (2021)

Description: This paper proposes a case-based reasoning approach with reinforcement learning for dynamic carrier recommendation in intermodal transportation. This adapts to real-time changes and learns from historical data.

Advantages: Handles dynamic scenarios and adapts to changing conditions. Leverages historical data for better decision-making.

Disadvantages: Requires extensive historical data for effective learning. May suffer from limited data availability in specific scenarios.

6. "Genetic Algorithm-based Optimization for Multi-modal Carrier Selection in Supply Chain Management" by Xu et al. (2020)

Description: This paper explores a genetic algorithm-based approach for multi-modal carrier selection in supply chain management. This finds optimal solutions through biological evolution, considering multiple criteria.

Advantages: Offers innovative solution finding with genetic algorithms. Adapts to complex constraints and multi-criteria optimization.

Disadvantages: Computationally expensive and requires extensive parameter tuning. May suffer from complex implementation.

7. "Hybrid Explainable Neural Network Approach for Carrier Selection in Freight Transportation" by Zhang et al. (2020)

Description: This paper proposes an explainable neural network approach for carrier selection. This combines deep learning with explainability techniques to provide insights into the decision-making process.

Advantages: Offers high accuracy with deep learning. Provides explanations for model predictions through interpretability techniques.

Disadvantages: Requires careful design of explainability techniques. May be computationally expensive.

8. "Hybrid Model with Deep Learning and Rule-based System for Real-time Carrier Recommendation in E-commerce Logistics" by Liu et al. (2019)

Description: This paper combines deep learning and rule-based systems for real-time carrier recommendation in e-commerce logistics. This leverages the strengths of both approaches for fast and adaptable recommendations.

Advantages: Offers fast and real-time recommendations. Adapts to changing market conditions and customer preferences.

Disadvantages: Requires continuous data collection and model updates. May be susceptible to data quality issues.

9. "A systematic review and research perspective on recommender systems", by Deepjyoti Roy and Mala Dutta. (2022)

Description: The paper has discussed regarding the various technologies and methods used for various recommendation systems. Survey tells about the architectures and the parameters to be considered during the system design and implementation.

Advantages: Helps to know about different methodologies used in available recommender systems and their the available options to design our system.

CHAPTER 3

REQUIREMENTS

3.1 Software Requirements

a. Integrated Development Environment (IDE):

PyCharm Community: The project development will be carried out using PyCharm as the primary integrated development environment (IDE) for Python.

b. Package and Environment Management

Anaconda Navigator: Utilize Anaconda for efficient package and environment management. Anaconda provides a convenient way to create and manage Python environments, ensuring compatibility with the specified libraries.

c. Libraries and Frameworks

NumPy: For efficient numerical operations and array manipulations.

Pandas: To handle dataframes and conduct data analysis.

Matplotlib and Seaborn: For data visualization and creating insightful plots.

Scikit-learn (sklearn): For machine learning algorithms and model evaluation.

Keras and TensorFlow: Implement deep learning models for enhanced predictive capabilities.

3.2 Hardware Requirements

a. **Processor:** A multicore processor (e.g., Intel Core i5 or AMD Ryzen 5) to handle the computational demands of training machine learning models.

b. **RAM**:8 GB RAM or higher to ensure smooth execution of data preprocessing, model training, and Flask application deployment.

c. **Storage**: Adequate storage space for datasets, models, and the development environment. SSDs are recommended for faster read/write speeds.

d. **Graphics Processing Unit (GPU)** [Optional]:For accelerated training of deep learning models, a compatible GPU (e.g., NVIDIA GeForce GTX or Tesla series) can significantly enhance processing speed.

e. **Operating System**:The project is compatible with Windows, macOS, or Linux operating systems, allowing flexibility for development and deployment.

f. **Network Connectivity**: A stable internet connection may be necessary for library installations, updates, and potential cloud-based deployment.

3.3 Functional Requirements

a. Data Acquisition and Preprocessing

Requirement 1: The system must acquire diverse and representative historical shipment data.

Requirement 2: Data preprocessing must handle missing values, outliers, and ensure standardization and encoding for relevant variables.

b. Model Training

Requirement 3: Individual machine learning models (SVM, Random Forest, Decision Trees) must be trained on the prepared dataset.

Requirement 4: Develop a hybrid model that seamlessly integrates the outputs of the base models.

c. Meta-Model Integration

Requirement 5: Integrate XGBoost as a meta-model to refine and consolidate predictions from the base models.

d. Flask Integration

Requirement 6: Develop a Flask application with API endpoints for handling input data.

Requirement 7: Implement a user-friendly interface to facilitate seamless interaction with stakeholders.

e. Real-time Recommendations

Requirement 8: The integrated system must provide real-time recommendations based on user input.

f. Validation and Optimization

Requirement 09: Validate the system's performance using a testing dataset.

Requirement 10: Optimize system parameters iteratively for enhanced efficiency.

3.4 Non-Functional Requirements

a. Performance: The system should be capable of handling a large volume of historical data for training the multilabel classification model efficiently. Response time for providing recommendations should be within an acceptable timeframe to ensure real-time or near-real-time usability.

b. Scalability: The system should be scalable to accommodate an increasing number of carriers, shippers, and transactions as the user base grows.

c. Reliability: The recommendation system should be highly reliable, with minimal downtime or service interruptions, to ensure continuous support for logistics operations.

d. Security: Robust security measures should be implemented to protect sensitive data, including historical shipment records and user information.

e. Privacy: The system should adhere to privacy regulations and ensure the confidentiality of sensitive business information related to carriers and shippers.

f. Usability: The user interface should be intuitive and user-friendly, catering to users with varying levels of technical expertise in the logistics domain.

g. Compatibility: The system should be compatible with different browsers and devices, ensuring accessibility for a diverse user base.

h. Maintainability: The codebase and system architecture should be well-documented and designed to facilitate ease of maintenance and future updates.

CHAPTER 4

SYSTEM DESIGN

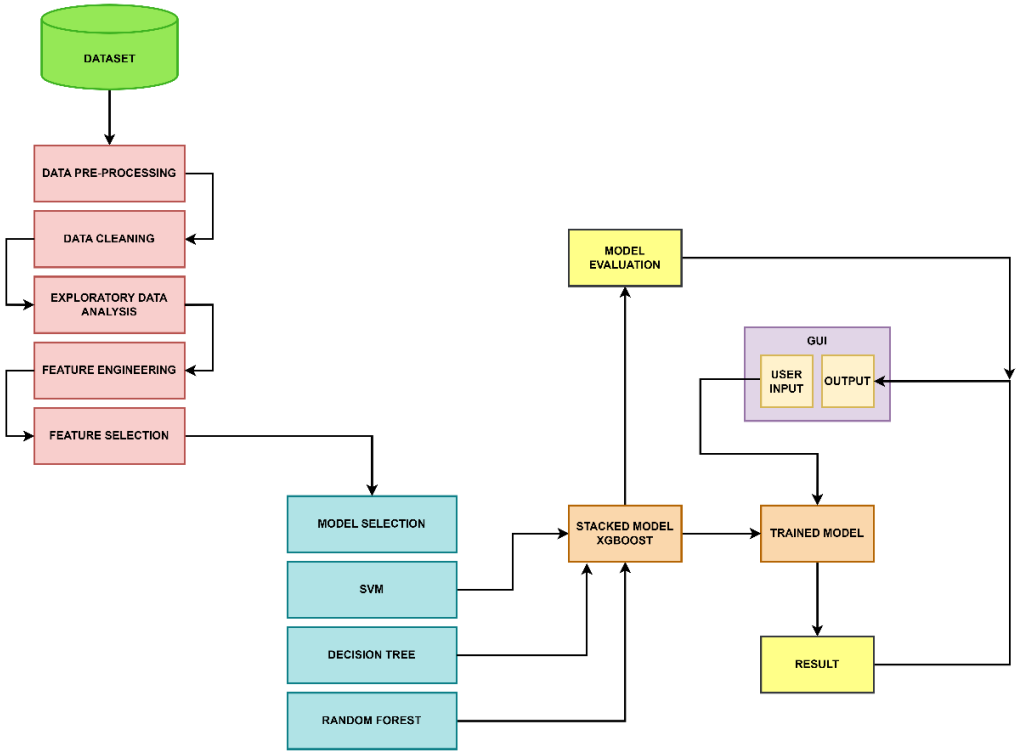


Fig 4.1: Architecture diagram

The architecture of the Smart Recommendation System for Carrier-Shipper Matching employs Multilabel Classification for enhanced efficiency. The system utilizes a sophisticated algorithm to analyze diverse factors, such as cargo type, delivery time, and route preferences. Through machine learning, it intelligently categorizes carriers and shippers, facilitating optimal matches. The core framework integrates data preprocessing, feature extraction, and model training stages, ensuring accurate predictions. Real-time updates and user feedback further refine the system's recommendations. This architecture not only streamlines logistics but also enhances overall supply chain management by providing tailored suggestions for carrier-shipper collaborations.

4.2 Use Case Diagram

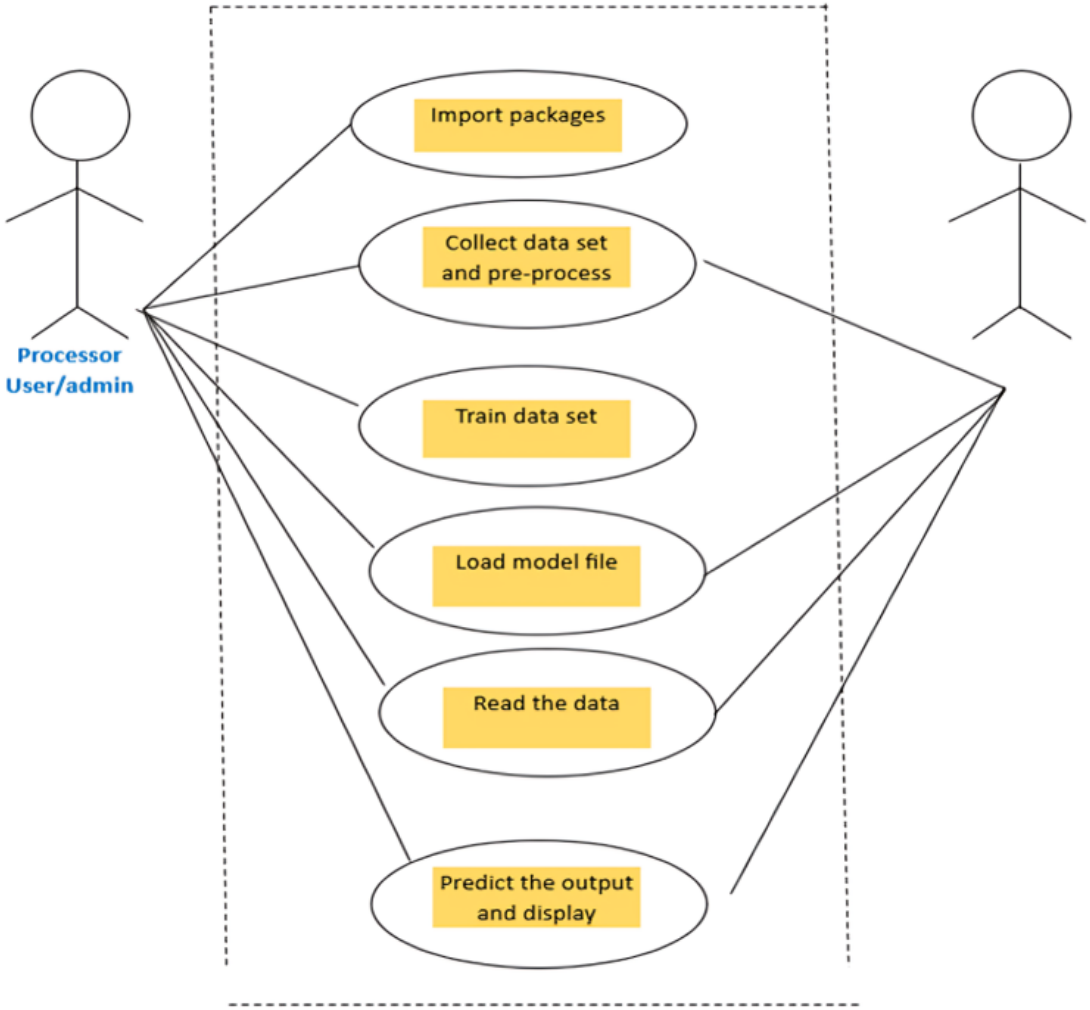


Fig 4.2 Use Case Diagram

Use case consists of user and processor where user is used to provide the input to the system and processor is used to process the input data and provide output. The flow is shown in the above diagram.

First user as to run the system and run the code, model and library packages are imported and loaded. After the run of code output is displayed according to the data input provided.

4.3 Sequence Diagram

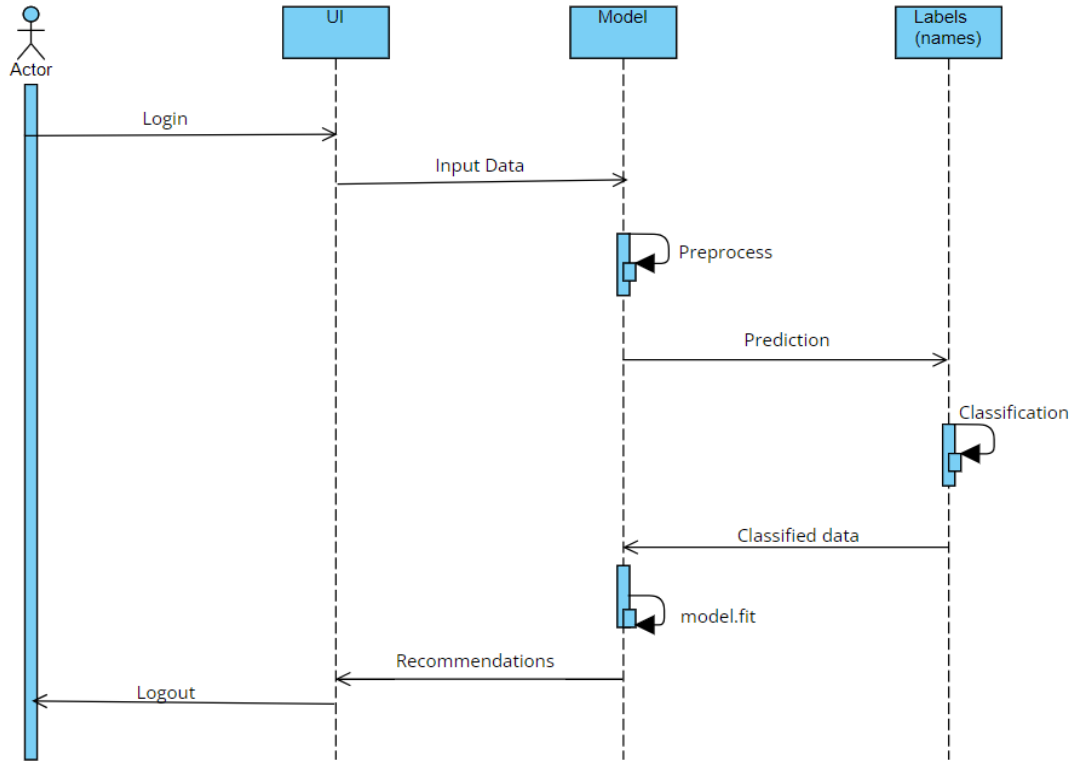


Figure: 4.3 Sequence Diagram

Sequence diagram consists of 4 different blocks namely user, UI, Model and labels as shown in the above figure

User will provide the input data which is asked and already saved in the system where preprocessing of data is done which is differentiator parameters and after that those are stored in the memory unit.

After preprocessing and storing of dataset is done, trained model file is loaded where the featured of the file is extracted for classifying the output. After classifying the output, Prediction values are displayed.

4.4 Data Flow Diagram

- **Level 0**

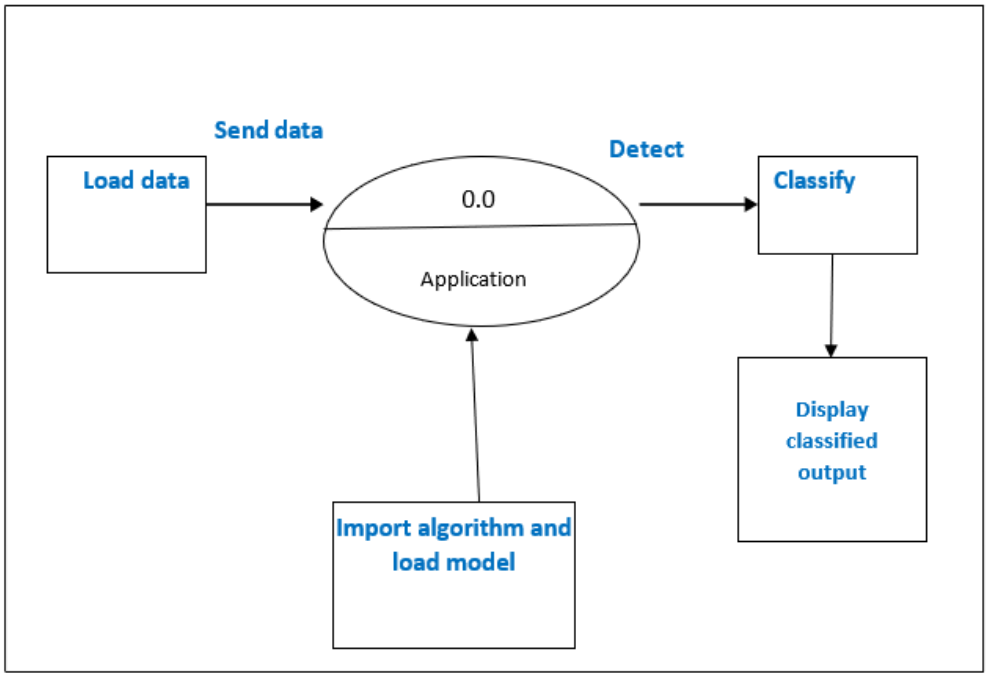


Figure 4.4.1: DFD 0

Above mentioned diagram is the representation of DFD0 which provides u the content diagram or say overview of the whole system. It is designed to be an at- a-glance view, showing the system as single high-level process. Here from the file data is be loaded to the application where the loaded data is sent to classification unit to predict the result with the help of trained model file and output is known.

- **LEVEL 1**

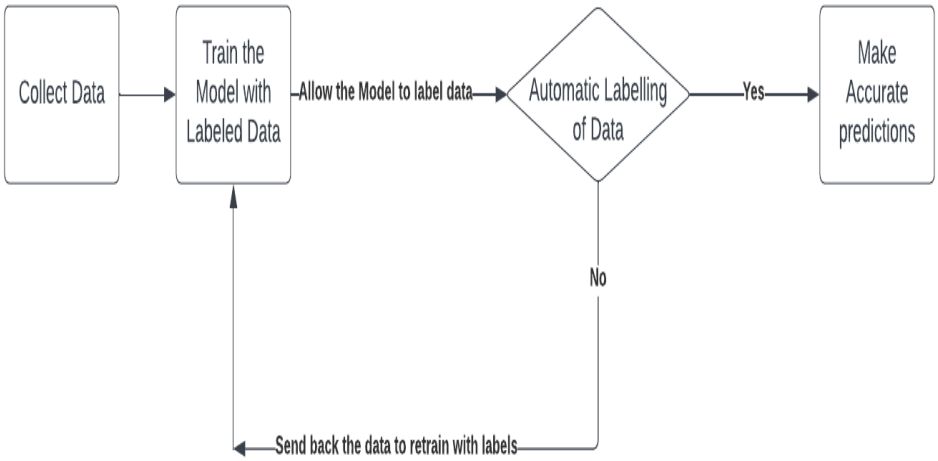


Figure 4.4.2: DFD 1

Above mentioned diagram is the representation of DFD1. The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Here from the file data is be loaded to the application where the loaded data is sent to classification unit to predict the result and classes are classified given a label

4.5 Activity diagram

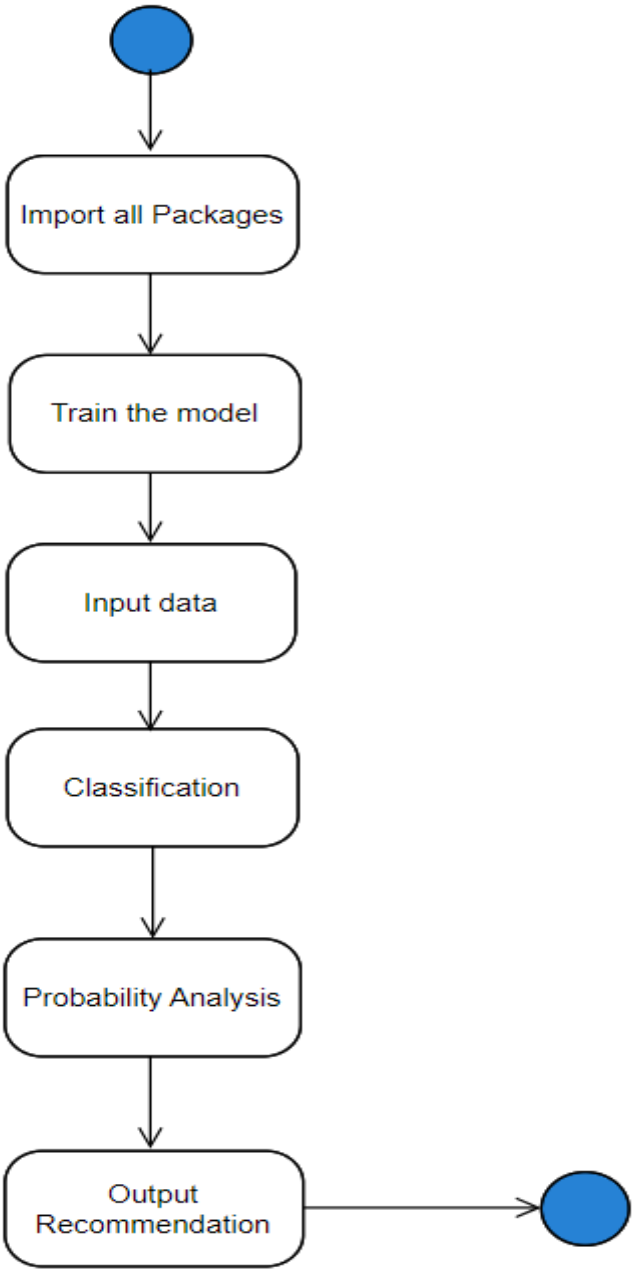


Figure 4.5: Activity Diagram

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

Firstly, we import all the library package necessary to run the code and for supporting the to code to run error free. As soon as code is run it provides the desired output

4.6 CLASS DIAGRAM

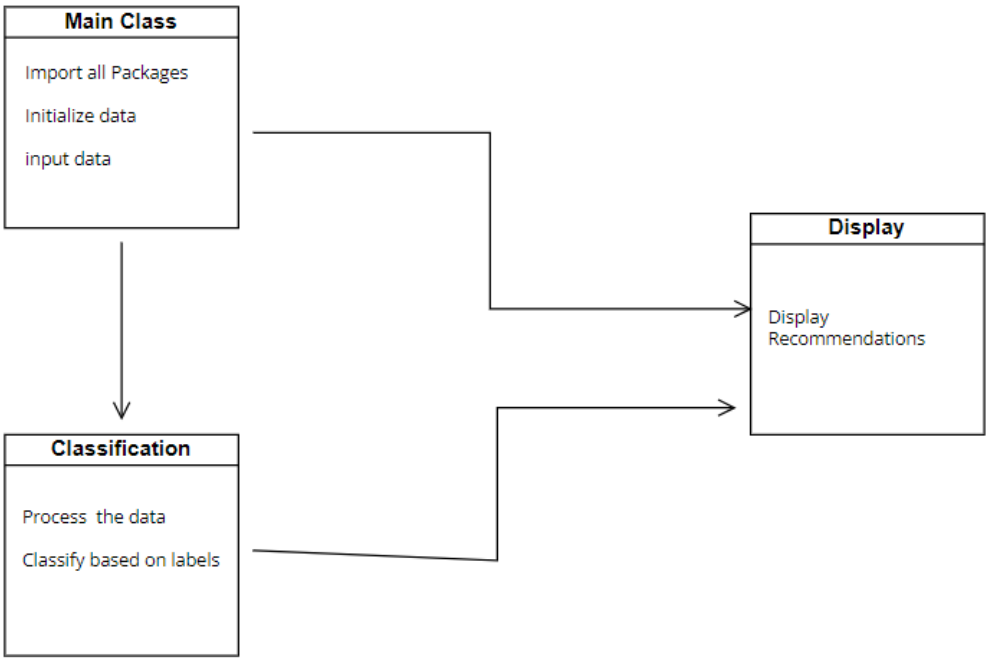


Figure 4.6: Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

CHAPTER 5

IMPLEMENTATION

5.1 Front End

HTML: HTML (Hypertext Markup Language) serves as the backbone of frontend application development, providing the structure and content of web pages. It's a markup language used to define the layout and organization of web documents. Here's a brief overview of HTML's role in frontend development.

Semantic Structure: HTML offers a range of semantic elements such as <header>, <nav>, <main>, <section>, <article>, <footer>, etc., which provide meaningful structure to the content, aiding accessibility and SEO (Search Engine Optimization).

Content Representation: HTML allows developers to represent various types of content including text, images, videos, audio files, forms, and more, using appropriate tags such as <p>, <img>, <video>, <audio>, <form>, etc.

Hyperlinks: HTML enables the creation of hyperlinks using the <a> tag, allowing users to navigate between different pages within the application or to external resources.

Forms and Inputs: HTML provides form elements like <input>, <select>, <textarea>, etc., enabling the collection of user input. These elements are crucial for building interactive web applications, such as login forms, registration forms, and search bars.

Accessibility: HTML supports accessibility features such as ARIA (Accessible Rich Internet Applications) attributes, which help in making web content more accessible to users with disabilities by providing additional semantic information.

Integration with CSS and JavaScript: HTML seamlessly integrates with CSS (Cascading Style Sheets) for styling and layout, allowing developers to control the visual presentation of the content. It also interacts with JavaScript for adding interactivity and dynamic behavior to web pages.

Responsive Design: HTML5 introduced new features like <meta> tags for viewport settings and semantic elements like <header>, <nav>, <footer>, etc., which contribute to building responsive web applications that adapt to different screen sizes and devices.

Our project frontend embodies a user-centric design, aiming for simplicity and accessibility. The layout is structured yet flexible, ensuring seamless navigation for users of all levels. The homepage serves as a welcoming gateway, providing an overview of the project's objectives and key features. Clear and concise sections, such as 'About Us' and 'Contact', offer visitors deeper insights into the project and facilitate easy communication. Navigation is intuitive, facilitated by a streamlined menu at the top, allowing users to effortlessly explore different sections. The design is responsive, adapting gracefully to various screen sizes, ensuring a consistent and enjoyable experience across devices. By prioritizing user experience and visual clarity, our frontend seeks to engage and inform users effectively, ultimately enhancing the overall project experience.

CSS: CSS (Cascading Style Sheets) plays a crucial role in frontend development by controlling the presentation, layout, and styling of HTML elements in a web project. Here's a brief overview of CSS's usage in frontend projects.

Styling: CSS is primarily used to style HTML elements, allowing developers to define attributes such as color, font, size, spacing, borders, and backgrounds. This styling enhances the visual appeal and aesthetics of the web pages, creating a more engaging user experience.

Layout: CSS enables developers to control the layout of elements on the webpage, including positioning, alignment, and responsiveness. Techniques like Flexbox and CSS Grid provide powerful tools for creating complex layouts with ease, ensuring a consistent and organized structure across different screen sizes and devices.

Responsive Design: CSS plays a pivotal role in creating responsive web designs that adapt to various viewport sizes, such as desktops, tablets, and mobile devices. Media queries allow developers to apply different styles based on the screen width, enabling content to adjust dynamically for optimal viewing on different devices.

Animation and Effects: CSS provides support for animations and transitions, allowing developers to add interactive elements and visual effects to web pages. CSS animations can be used to create effects like fading, sliding, rotating, and scaling, enhancing the user experience and making the interface more engaging.

Modularity and Reusability: CSS promotes modularity and reusability through techniques like class and ID selectors, enabling developers to define styles once and apply them to multiple elements throughout the project. This helps in maintaining consistency and scalability, especially in larger applications.

Cross-Browser Compatibility: CSS helps ensure cross-browser compatibility by providing standardized styling rules that render consistently across different web browsers. CSS vendor prefixes and browser-specific hacks can also be used to address compatibility issues and ensure consistent rendering.

Accessibility: CSS can be leveraged to improve the accessibility of web content by enhancing readability, contrast, and navigation. Techniques such as using appropriate color contrasts, providing clear focus states for interactive elements, and ensuring proper text sizing contribute to a more accessible user experience.

5.2 Backend

Python is a versatile and powerful programming language commonly used in backend development for web projects. Here's a brief overview of its usage:

Web Frameworks: Python offers a variety of web frameworks such as Django, Flask, and FastAPI, which provide pre-built components and tools for building robust backend systems. These frameworks handle common tasks like routing, request handling, authentication, database integration, and more, enabling developers to create scalable and maintainable web applications efficiently.

Data Processing and Manipulation: Python's rich ecosystem of libraries, including NumPy, pandas, and SciPy, make it well-suited for data processing and manipulation tasks in backend development. These libraries facilitate tasks such as data cleaning, analysis, transformation, and visualization, which are often required in web applications dealing with large datasets or complex data structures.

Database Integration: Python offers libraries and modules for interacting with various databases, such as PostgreSQL, MySQL, SQLite, MongoDB, and more. ORMs (Object-Relational Mappers) like Django's ORM and SQLAlchemy provide abstraction layers that simplify database operations, allowing developers to work with databases using Python objects and queries.

API Development: Python is widely used for building RESTful APIs (Application Programming Interfaces) that expose backend functionality to frontend clients or other services. Frameworks like Flask and FastAPI provide features for defining API routes, handling requests and responses, authentication, serialization, and validation, making API development straightforward and efficient.

Asynchronous Programming: Python's asyncio module and asynchronous frameworks like aiohttp enable developers to write asynchronous, non-blocking code, which is particularly useful for handling I/O-bound tasks in web applications, such as network requests, database queries, and file operations. Asynchronous programming can improve the performance and scalability of backend systems by maximizing resource utilization.

Task Automation: Python's scripting capabilities make it suitable for automating repetitive tasks in backend development, such as data migration, batch processing, cron jobs, and deployment tasks. Libraries like Fabric and Invoke provide utilities for task execution and management, streamlining development workflows and improving productivity.

Machine Learning and AI: Python's extensive libraries for machine learning and artificial intelligence, including scikit-learn, TensorFlow, PyTorch, and NLTK, enable developers to integrate advanced analytics and predictive capabilities into backend systems. This is particularly valuable for applications requiring data-driven insights, recommendation engines, natural language processing, and other AI-driven features.

Flask is a lightweight and flexible web framework for Python, known for its simplicity and ease of use. With Flask, developers can quickly build web applications ranging from simple prototypes to complex, production-ready systems. Its minimalist design allows for greater flexibility and customization, making it an ideal choice for projects of all sizes. Flask provides essential features such as URL routing, template rendering, and HTTP request handling, while also offering extensions for additional functionality like authentication, database integration, and RESTful API development. Its modular architecture and extensive documentation make it accessible to both beginners and experienced developers, fostering a vibrant community and ecosystem of third-party libraries and tools. Overall, Flask empowers developers to create web applications efficiently, without unnecessary overhead, enabling rapid development and deployment cycles.

5.3 Integrated Development Environment

PyCharm is a powerful integrated development environment (IDE) specifically designed for Python development. Here's a brief overview of its usage in a project:

Code Editing: PyCharm provides advanced code editing features such as syntax highlighting, code completion, code navigation, and refactoring tools, which help developers write clean, efficient, and error-free code.

Project Management: PyCharm offers robust project management capabilities, allowing developers to create, organize, and manage Python projects effectively. It supports various project types including standalone scripts, web applications, data analysis projects, and more.

Version Control Integration: PyCharm seamlessly integrates with version control systems like Git, SVN, Mercurial, and Perforce, enabling developers to manage source code changes, commit revisions, synchronize with remote repositories, and resolve merge conflicts directly within the IDE.

Debugging: PyCharm includes a powerful debugger with features such as breakpoints, step-through execution, variable inspection, and watch expressions, making it easy to identify and fix bugs in Python code. It also supports remote debugging for debugging applications running on remote servers or virtual environments.

Testing: PyCharm supports various testing frameworks such as pytest, unit test, and doctest, allowing developers to write and run unit tests, integration tests, and functional tests within the IDE. It provides features for test discovery, execution, result visualization, and code coverage analysis.

Virtual Environments: PyCharm offers built-in support for creating and managing virtual environments using virtualenv, venv, and conda, allowing developers to isolate project dependencies and ensure consistent development environments across different projects.

Code Quality Tools: PyCharm integrates with code quality analysis tools such as Pylint, Flake8, and mypy, enabling developers to identify potential issues, enforce coding standards, and improve code readability and maintainability.

Refactoring: PyCharm provides a range of refactoring tools such as renaming, extracting methods, inline variables, and introducing variables, helping developers to safely and efficiently refactor code without introducing errors.

5.4 Model Architecture

a. Support Vector Machine:

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. Its primary objective is to find the optimal hyperplane that separates data points into different classes with the maximum margin. SVM works by mapping input data points into a high-dimensional feature space, where it finds the hyperplane that best separates the classes. This hyperplane is determined by support vectors, which are the data points closest to the decision boundary. SVM is effective in handling both linearly separable and non-linearly separable data through the use of different kernel functions, such as linear, polynomial, radial basis function (RBF), and sigmoid kernels. SVM has several advantages, including its ability to handle high-dimensional data, resistance to overfitting, and effectiveness even with small datasets. However, SVM's performance can be sensitive to the choice of kernel function and its parameters, and it may not scale well to very large datasets. Overall, SVM is a versatile and widely used algorithm in machine learning, particularly in applications like image classification, text classification, and bioinformatics.

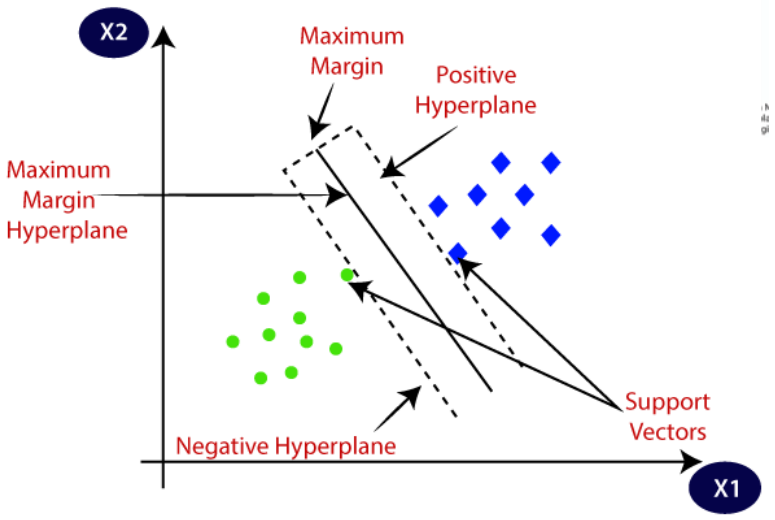


Fig 5.4.1: SVM Algorithm

b. Decision Tree:

Decision Tree is a versatile and intuitive supervised learning algorithm widely used for both classification and regression tasks in machine learning. It operates by recursively partitioning the feature space into smaller regions, based on the values of input features, to create a hierarchical tree-like structure. At each internal node of the tree, a decision is made based on a feature's value, leading to different branches or sub-trees. The ultimate goal is to divide the data into homogeneous subsets, where each leaf node represents a class label (in classification) or a predicted value (in regression). Decision Trees are easy to interpret and visualize, making them useful for understanding the underlying data patterns and explaining the model's predictions. They can handle both numerical and categorical data, and they're robust to outliers and missing values. However, Decision Trees tend to overfit the training data, especially if the tree grows too deep. Techniques like pruning, ensemble methods (e.g., Random Forests), and using appropriate stopping criteria help mitigate overfitting and improve generalization. Overall, Decision Trees are powerful tools for building interpretable and efficient predictive models in various domains, from finance and healthcare to marketing and beyond.

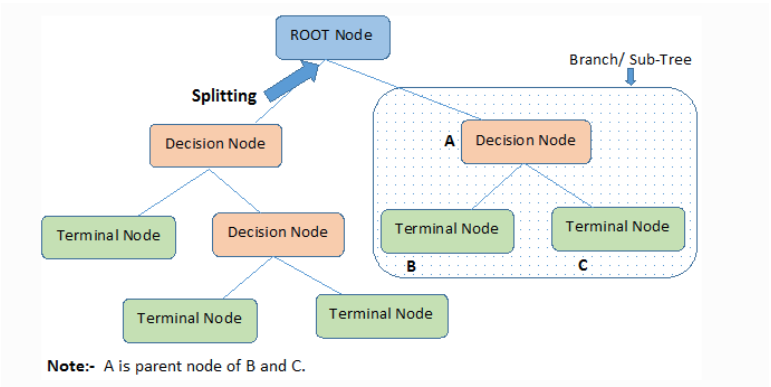


Fig 5.4.2: Decision Tree Algorithm

C. Random Forest:

Random Forest is a popular and powerful ensemble learning algorithm used for classification and regression tasks in machine learning. It operates by constructing a multitude of decision trees during training and outputting the class (in classification) or the average prediction (in regression) of the individual trees. Each tree in the Random Forest is trained on a random subset of the training data and a random subset of features, introducing randomness and diversity into the ensemble. During prediction, each tree's output is aggregated through voting (for classification) or averaging (for regression), resulting in a robust and accurate final prediction. Random Forests are known for their high performance, scalability, and resistance to overfitting, making them suitable for a wide range of applications. They can handle large datasets with high-dimensional features and are less sensitive to noisy data and outliers compared to individual decision trees. Moreover, Random Forests provide valuable insights into feature importance, helping identify the most relevant features for prediction. Overall, Random Forest is a versatile and effective algorithm that consistently delivers competitive performance in various machine learning tasks.

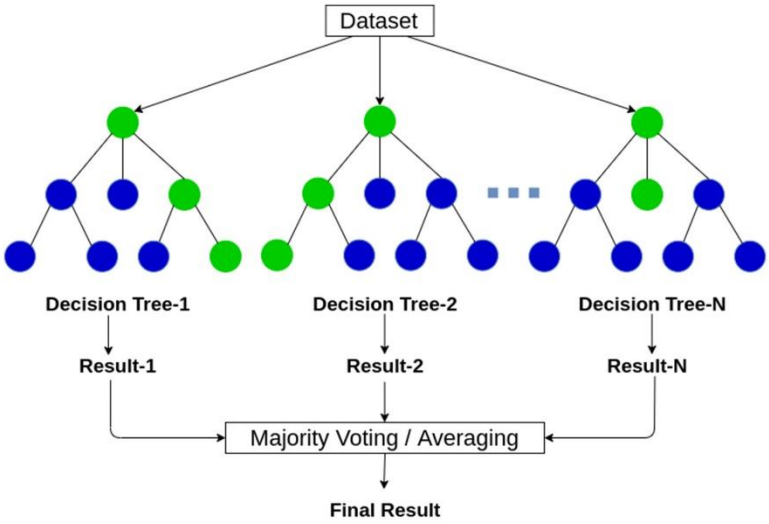


Fig 5.4.3: Random Forest Algorithm

CHAPTER 6

TESTING AND RESULTS

6.1 Testing

In the realm of software development, ensuring the robustness and reliability of a project is paramount. Unit testing and integrated testing are indispensable methodologies employed to validate the functionality, performance, and integrity of software systems. In the context of our aforementioned project, which emphasizes real-time data integration and user feedback analysis, the implementation of rigorous testing procedures becomes even more critical. This section delves into the unit testing and integrated testing strategies adopted in the development of our project, highlighting their significance and impact.

Unit Testing: Unit testing is a software testing technique wherein individual units or components of a software application are tested in isolation to validate their correctness and functionality. Each unit is tested independently to ensure that it behaves as expected and meets the specified requirements. In the context of our project, unit testing played a vital role in verifying the correctness of critical functions, algorithms, and modules responsible for real-time data integration and user feedback analysis.

The unit testing process involved: Test Case Design: Designing comprehensive test cases covering all possible scenarios and edge cases to validate the behavior of individual units. Test Execution: Executing the designed test cases using automated testing frameworks or manual testing procedures to identify defects, errors, or unexpected behaviors.

Defect Identification and Resolution: Identifying and documenting any defects or discrepancies uncovered during unit testing and subsequently rectifying them through debugging and code refactoring.

Regression Testing: Conducting regression testing to ensure that modifications or enhancements made to the codebase do not introduce new defects or regressions into previously tested units.

Unit testing not only ensured the correctness and reliability of individual components but also facilitated early defect detection and resolution, thereby reducing the overall cost and effort associated with the development lifecycle.

Integrated Testing: Integrated testing, on the other hand, focuses on validating the interactions and interdependencies between different units or modules within a software system. It aims to uncover integration issues, compatibility issues, and system-level defects that may arise due to the interaction of multiple components. In the context of our project, which involves the integration of real-time data streams and user feedback analysis modules, integrated testing played a crucial role in ensuring the seamless operation and interoperability of the entire system.

The integrated testing process encompassed: Integration Test Planning: Planning and defining test scenarios to validate the integration points and interactions between various system components.

Test Environment Setup: Setting up a dedicated test environment that closely resembles the production environment to simulate real-world conditions and interactions. Executing the defined test scenarios to verify the end-to-end functionality, data flow, and interoperability of the integrated system.

Performance and Scalability Testing: Evaluating the performance, scalability, and reliability of the integrated system under varying load conditions to identify potential bottlenecks or performance degradation issues.

Compatibility Testing: Ensuring compatibility with different operating systems, browsers, devices, and third-party integrations to guarantee a seamless user experience across diverse environments.

Integrated testing facilitated the identification of integration issues, data inconsistencies, and compatibility issues early in the development lifecycle, enabling timely resolution and ensuring the robustness and reliability of the final product.

6.2 Model Results

In machine learning, a confusion matrix is a tabular representation of the performance of a classification model, summarizing the predicted and actual class labels for a given dataset. It consists of four quadrants: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

True positives (TP) are instances where the model correctly predicts the positive class.
True negatives (TN) are instances where the model correctly predicts the negative class.
False positives (FP) are instances where the model incorrectly predicts the positive class.
False negatives (FN) are instances where the model incorrectly predicts the negative class
From the confusion matrix, various evaluation metrics can be derived, such as accuracy, precision, recall (sensitivity), specificity, F1 score, and the area under the ROC curve (AUC-ROC). These metrics provide insights into the model's performance, helping assess its effectiveness and identifying areas for improvement. Confusion matrices are widely used in evaluating the performance of machine learning algorithms across different domains, providing a comprehensive overview of their predictive capabilities.

In a classification report like the one you provided, several parameters are measured to evaluate the performance of a classification model. Here's a breakdown of the parameters used in this report:

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positives. It measures the accuracy of the positive predictions.

Recall (Sensitivity): Recall is the ratio of correctly predicted positive observations to all actual positives. It measures the ability of the model to find all the relevant cases within the dataset.

F1-Score: The F1-Score is the harmonic mean of precision and recall. It provides a balance between precision and recall, especially when the classes are imbalanced.
Support: Support is the number of actual occurrences of the class in the specified dataset. It's the number of samples of the true response that lies in that class.

Accuracy: Accuracy is the ratio of correctly predicted observations to the total observations. It measures the overall correctness of the model.

Macro Avg: Macro average calculates the metric independently for each class and then takes the average, giving equal weight to each class.

Weighted Avg: Weighted average calculates the metric for each class and takes the weighted average based on the number of true instances for each class. It's useful when there's class imbalance.

a. Support Vector Machine:

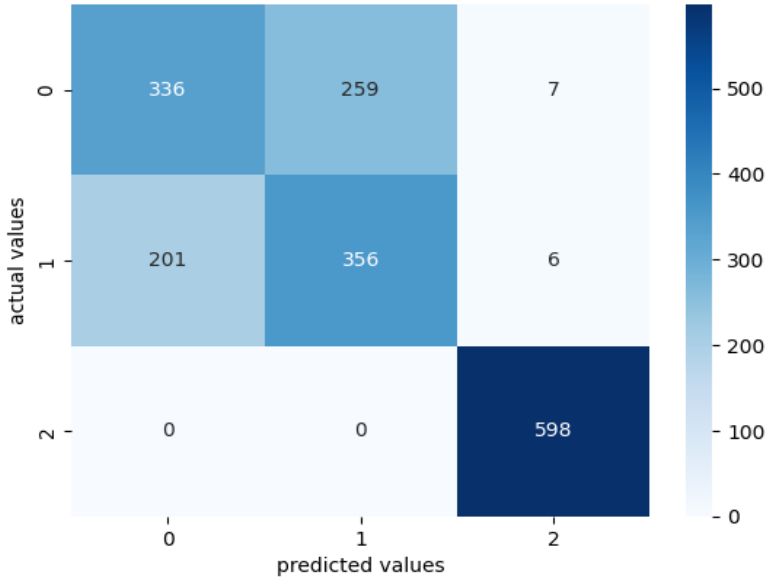


Fig 6.2.1: SVM Confusion Matrix

	precision	recall	f1-score	support
0	0.56	0.63	0.59	537
1	0.63	0.58	0.60	615
2	1.00	0.98	0.99	611
accuracy			0.73	1763
macro avg	0.73	0.73	0.73	1763
weighted avg	0.74	0.73	0.73	1763

Fig 6.2.2: Classification report

b. Random Forest:

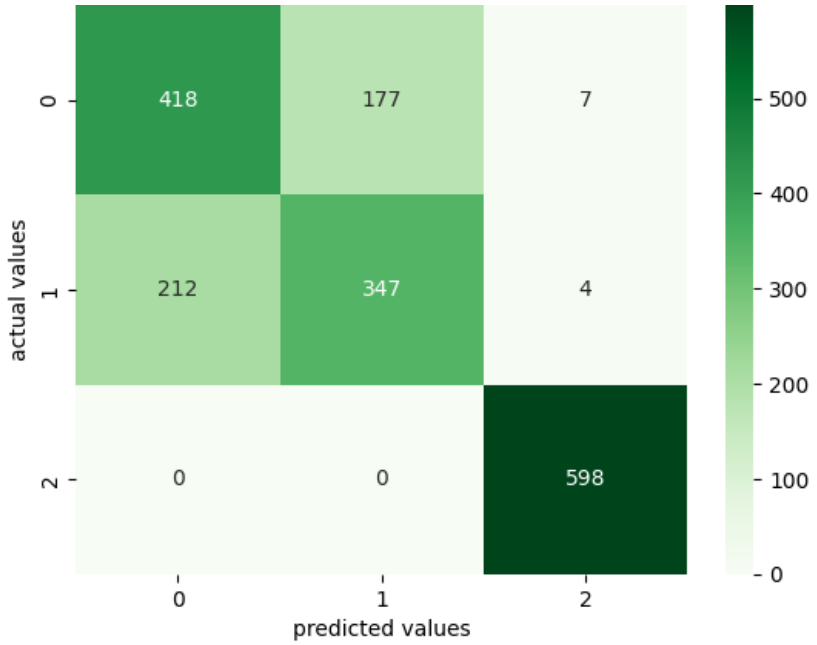


Fig 6.2.3: Random Forest Confusion Matrix

	precision	recall	f1-score	support
0	0.69	0.66	0.68	630
1	0.62	0.66	0.64	524
2	1.00	0.98	0.99	609
accuracy			0.77	1763
macro avg	0.77	0.77	0.77	1763
weighted avg	0.78	0.77	0.77	1763

Fig 6.2.4: Random Forest Classification report

c. Decision Tree:

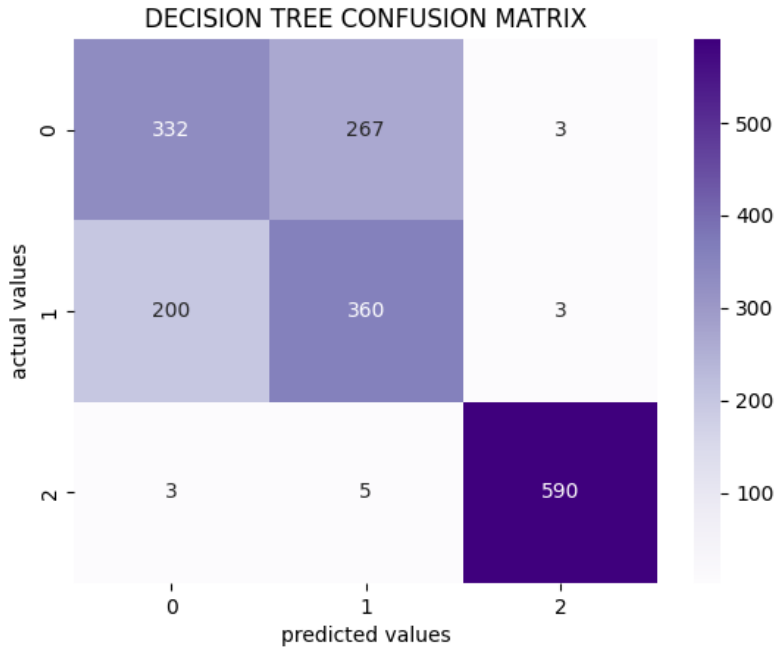


Fig 6.2.5: Decision Tree Confusion Matrix

	precision	recall	f1-score	support
0	0.55	0.62	0.58	535
1	0.64	0.57	0.60	632
2	0.99	0.99	0.99	596
accuracy			0.73	1763
macro avg	0.73	0.73	0.72	1763
weighted avg	0.73	0.73	0.73	1763

Fig 6.2.6: Decision Tree Classification report

d. Hybrid Model

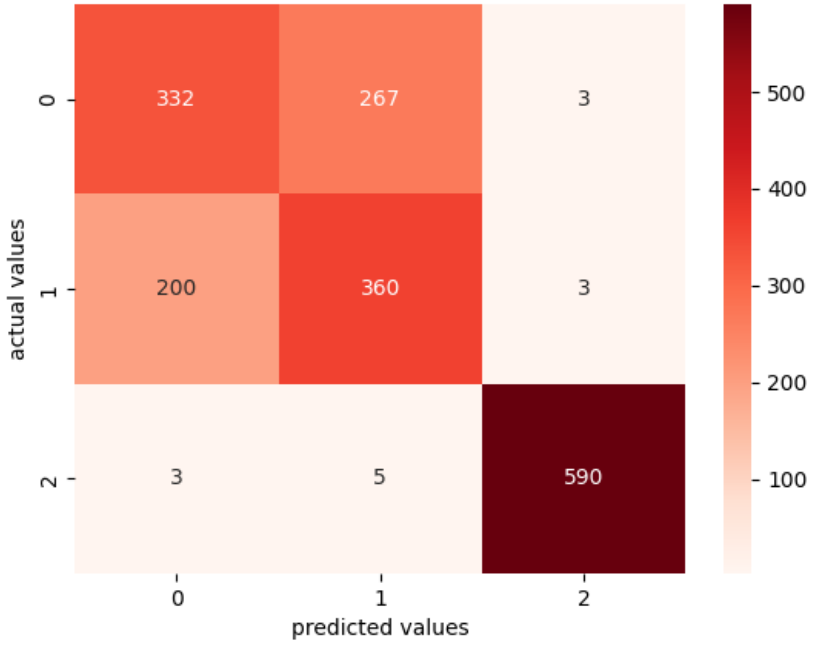


Fig 6.2.7: Hybrid Model Confusion Matrix

	precision	recall	f1-score	support
0	0.74	0.64	0.68	701
1	0.55	0.66	0.60	464
2	1.00	1.00	1.00	598
accuracy			0.77	1763
macro avg	0.76	0.77	0.76	1763
weighted avg	0.78	0.77	0.77	1763

Fig 6.2.8: Hybrid Model Classification report

CHAPTER 7

SNAPSHOTS

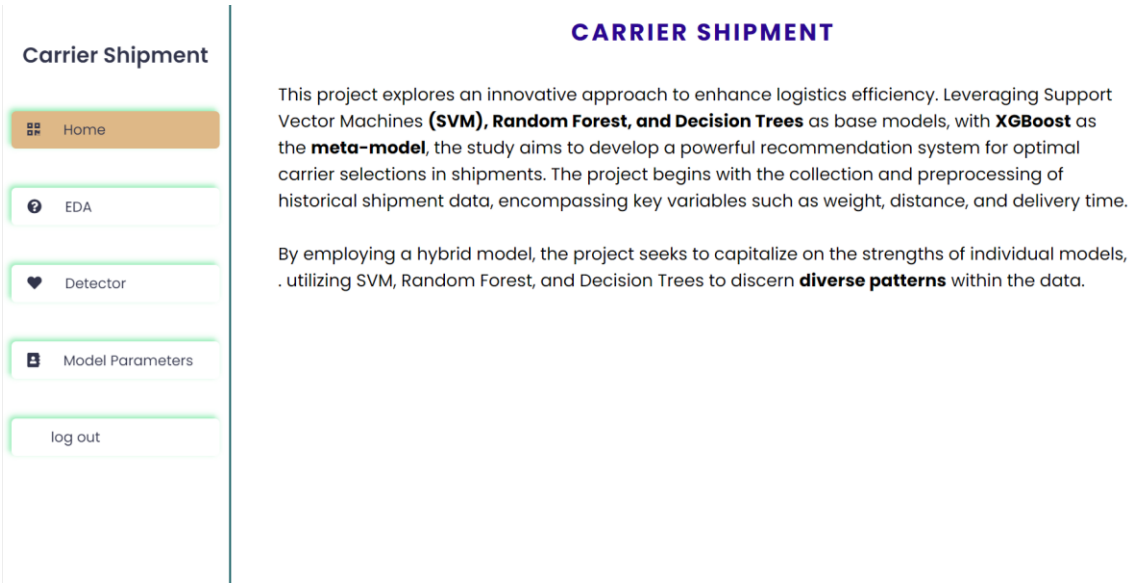


Fig 7.1: Home Page

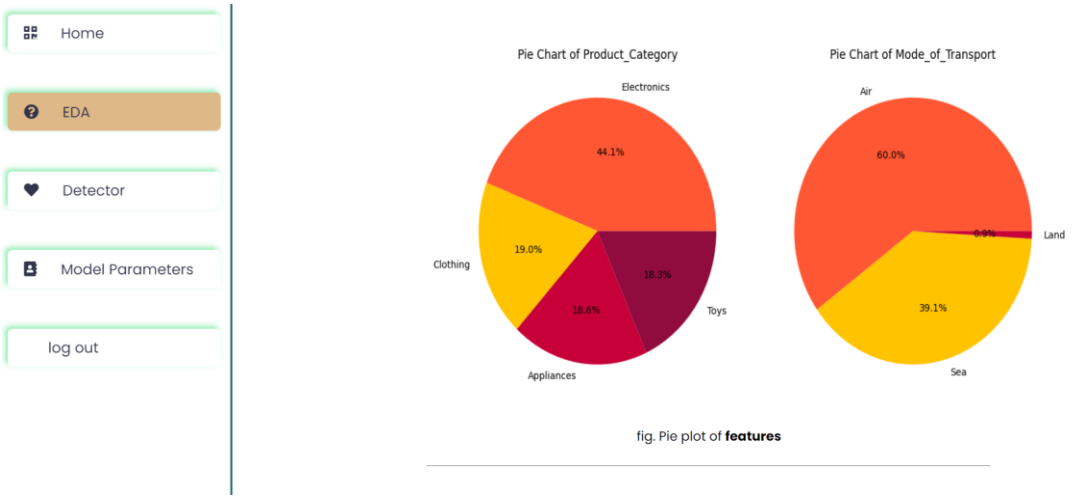


Fig 7.2: Exploratory Data Analysis

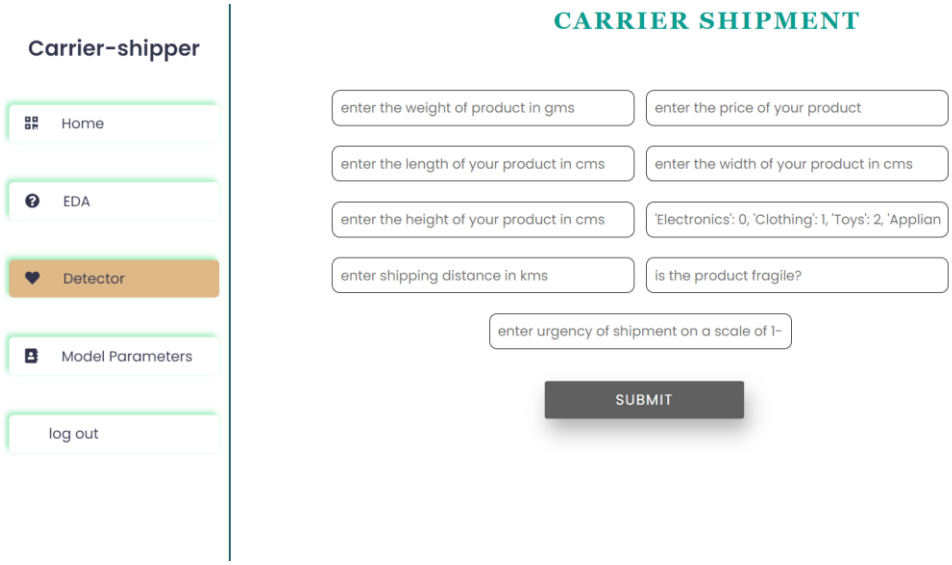


Fig 7.3: User Input page

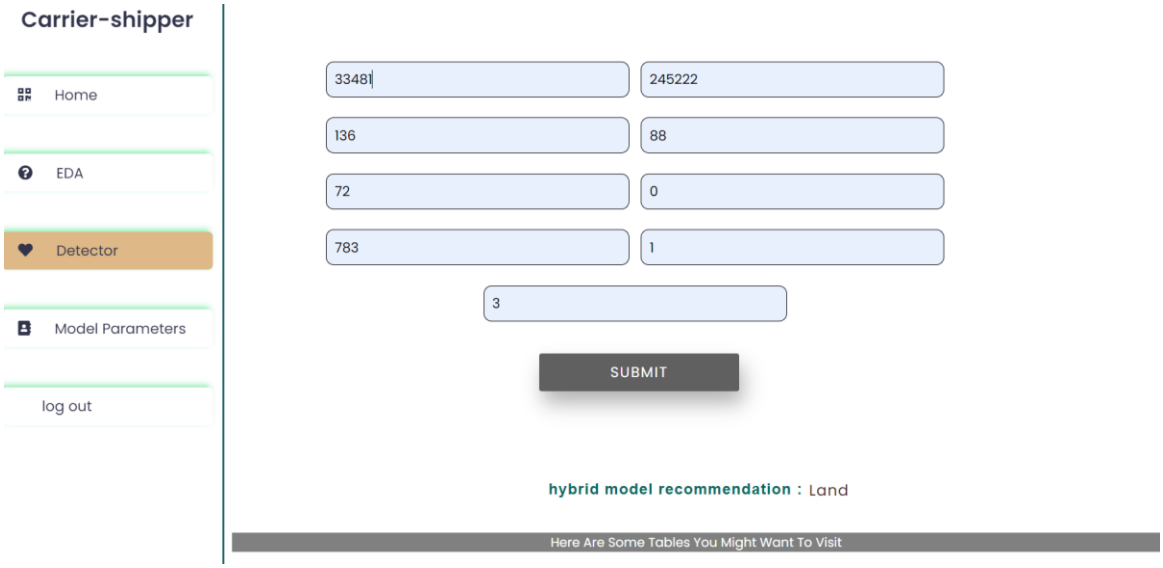
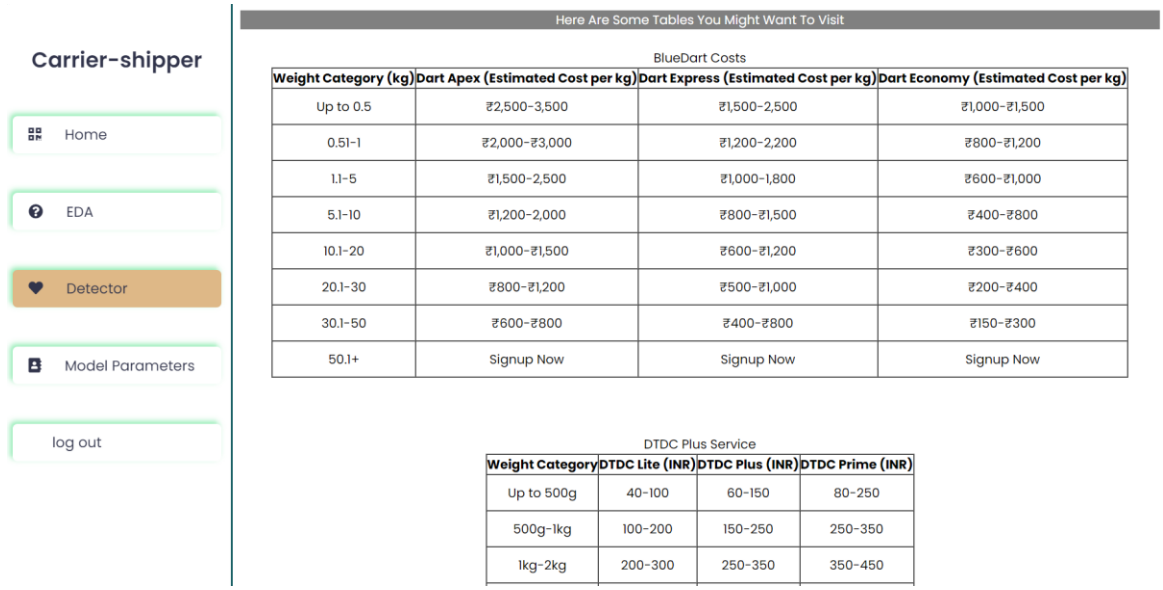


Fig 7.4: Recommendations



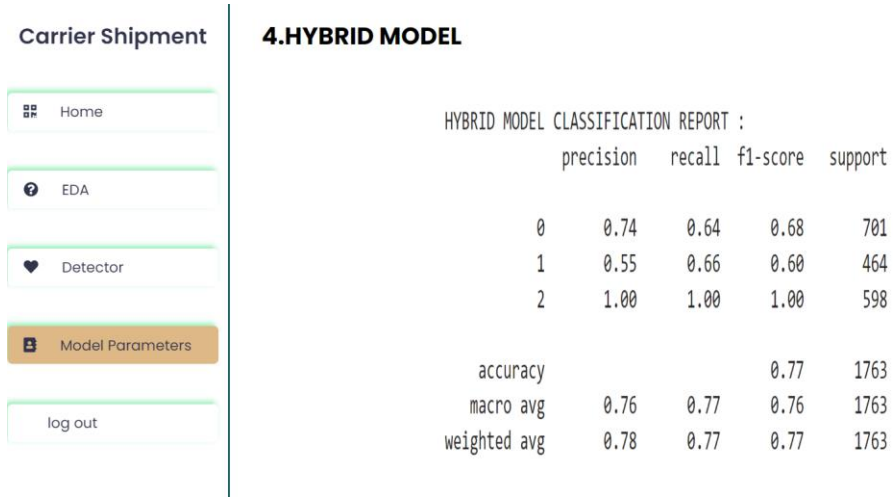
BlueDart Costs

Weight Category (kg)	Dart Apex (Estimated Cost per kg)	Dart Express (Estimated Cost per kg)	Dart Economy (Estimated Cost per kg)
Up to 0.5	₹2,500-3,500	₹1,500-2,500	₹1,000-₹1,500
0.51-1	₹2,000-₹3,000	₹1,200-2,200	₹800-₹1,200
1.1-5	₹1,500-2,500	₹1,000-1,800	₹600-₹1,000
5.1-10	₹1,200-2,000	₹800-₹1,500	₹400-₹800
10.1-20	₹1,000-₹1,500	₹600-₹1,200	₹300-₹600
20.1-30	₹800-₹1,200	₹500-₹1,000	₹200-₹400
30.1-50	₹600-₹800	₹400-₹800	₹150-₹300
50.1+	Signup Now	Signup Now	Signup Now

DTDC Plus Service

Weight Category	DTDC Lite (INR)	DTDC Plus (INR)	DTDC Prime (INR)
Up to 500g	40-100	60-150	80-250
500g-1kg	100-200	150-250	250-350
1kg-2kg	200-300	250-350	350-450

Fig 7.5: Available price charts



HYBRID MODEL CLASSIFICATION REPORT :

	precision	recall	f1-score	support
0	0.74	0.64	0.68	701
1	0.55	0.66	0.60	464
2	1.00	1.00	1.00	598
accuracy			0.77	1763
macro avg	0.76	0.77	0.76	1763
weighted avg	0.78	0.77	0.77	1763

Fig 7.6: Model parameters

CHAPTER 8

CONCLUSION

In conclusion, the project has successfully navigated the intricacies of logistics optimization. By leveraging a hybrid model encompassing Support Vector Machines, Random Forest, Decision Trees, and XGBoost, the project offers a versatile solution adaptable to diverse datasets and logistics scenarios. The integration with Flask ensures a seamless user experience, allowing stakeholders to effortlessly input data and receive real-time, data-driven carrier recommendations.

This project's significance lies in its potential to enhance decision-making processes within the logistics and supply chain industry. The rigorous testing procedures and considerations for scalability contribute to the system's reliability and effectiveness. With a focus on validation, optimization, and comprehensive documentation, the project not only addresses current challenges but also lays the foundation for ongoing maintenance and future advancements.

In essence, the project is a culmination of data science and user-centric design, offering a practical and impactful tool for optimizing carrier selections. It represents a promising step towards operational efficiency, cost savings, and improved customer satisfaction in the dynamic landscape of logistics and supply chain management.

CHAPTER 9

FUTURE WORK

In today's rapidly evolving technological landscape, the need for dynamic and responsive project reports has become increasingly evident. Traditional project reports, often static in nature, fail to capture the real-time insights and user sentiments essential for informed decision-making and iterative improvement. Therefore, there is a growing imperative to incorporate real-time data integration and user feedback analysis into project reports, ensuring their relevance, accuracy, and actionable insights. This document explores the significance of such integration and presents a comprehensive framework for its implementation.

Significance of Real-Time Data Integration: Real-time data integration involves the incorporation of live, up-to-the-moment data streams into project reports. This data encompasses various sources such as IoT devices, social media feeds, web analytics, and more. By integrating real-time data, project reports gain a dynamic quality, reflecting the current state of affairs accurately. This enables stakeholders to make informed decisions promptly, respond swiftly to emerging trends, and capitalize on opportunities as they arise.

Moreover, real-time data integration enhances the predictive capabilities of project reports. By analyzing ongoing trends and patterns in real-time, stakeholders can anticipate future developments and proactively adjust strategies accordingly. This proactive approach is invaluable in volatile environments where rapid adaptation is crucial for success.

Considerations for User Feedback Analysis: User feedback analysis is another vital aspect to be incorporated into project reports. User feedback provides valuable insights into the user experience, satisfaction levels, pain points, and areas for improvement. By systematically collecting, analyzing, and acting upon user feedback, organizations can refine their products, services, and processes to better align with user needs and preferences.

However, effective user feedback analysis goes beyond merely collecting data. It involves employing advanced analytics techniques such as sentiment analysis, text mining, and machine learning to derive actionable insights from feedback data. Additionally, it requires a feedback loop mechanism wherein insights gleaned from user feedback are iteratively integrated into project strategies, leading to continuous improvement and enhanced user satisfaction.

Integration Framework: To effectively integrate real-time data and user feedback analysis into project reports, a robust framework is essential. This framework should encompass the following key components:

Data Acquisition: Identify relevant real-time data sources and establish mechanisms for continuous data acquisition. This may involve deploying sensors, leveraging APIs, or partnering with data providers to access live data streams.

Data Processing: Implement data processing pipelines to cleanse, aggregate, and analyze incoming data in real-time. Employ advanced analytics techniques to extract meaningful insights and detect actionable patterns from raw data.

Integration with Project Reports: Integrate analyzed real-time data seamlessly into project reports, ensuring that stakeholders have access to the most current information. This may involve developing custom dashboards, data visualizations, or interactive reports tailored to the specific needs of stakeholders.

User Feedback Collection: Implement robust mechanisms for collecting user feedback across various touchpoints such as websites, mobile apps, and customer service channels. Utilize surveys, feedback forms, and sentiment analysis tools to capture qualitative and quantitative feedback effectively.

Feedback Analysis and Action: Analyze user feedback systematically using advanced analytics techniques to derive actionable insights. Establish feedback loops to incorporate insights into project strategies, driving continuous improvement and refinement.

REFERENCES

1. "Hybrid XGBoost-SVM Model for Multi-criteria Carrier Selection in E-commerce Logistics" by Wang et al. (2023)
2. "A Multi-objective Optimization Approach for Carrier Selection in Container Shipping using Random Forest and Deep Neural Network" by Chen et al. (2023)
3. "Ensemble Learning Approach for Carrier Selection in Freight Transportation using Stacking and Voting" by Wu et al. (2022)
4. "Rule-based and Machine Learning Hybrid Model for Carrier Selection in Less-than-Truckload Transportation" by He et al. (2022)
5. "Case-based Reasoning with Reinforcement Learning for Dynamic Carrier Recommendation in Intermodal Transportation" by Li et al. (2021)
6. "Genetic Algorithm-based Optimization for Multi-modal Carrier Selection in Supply Chain Management" by Xu et al. (2020)
7. "Hybrid Explainable Neural Network Approach for Carrier Selection in Freight Transportation" by Zhang et al. (2020)
8. "Hybrid Model with Deep Learning and Rule-based System for Real-time Carrier Recommendation in E-commerce Logistics" by Liu et al. (2019)
9. “A systematic review and research perspective on recommender systems” by Deepjyoti Roy* and Mala Dutta 2022.
10. “Artificial Intelligence based System on enhancing the Capabilities of Transport System: A Systemic Literature Review”, by Muhammad Zain Malik, Shah Nazir, Habib Ullah Khan(2023).
11. “Deep Based Recommender System For Relevant K Pick-up Points”, by Ayoub Berdeddouch, Ali Yhyaouy, Younes Benanni(2020).