ARCHITECTURE DESIGN (AD)

Shipment Pricing Prediction System

Revision Number: 1.0

Last date of revision: 04/06/2024

Authors:

[GOBIKRISHNAN

SRIRAM

SHINY]

Document Version Control

Date	Version	Description	Authors
4th june 2024	1.0	Initial Draft	GOBIKRISHNAN SRIRAM SHINY

Contents

Abstract:

The Shipment Pricing Prediction system leverages advanced machine learning techniques to accurately forecast the cost of shipping goods based on various factors. By analyzing historical shipment data, the model considers key variables such as destination country, shipment mode, product group, sub-classification, brand, unit of measure, quantity, and item value. The system utilizes one-hot encoding to handle categorical data and processes numerical features to generate precise pricing predictions. This solution aims to assist businesses in optimizing their logistics and budgeting strategies, ensuring cost-efficiency and better decision-making in supply chain management. Additionally, the system's integration with a user-friendly interface and robust backend database allows for seamless prediction services, real-time data updates, and comprehensive monitoring and logging, thereby enhancing overall operational efficiency and transparency**.**

1. Introduction

1.1 Why this Low-Level Design Document?

Low-level documentation for the Shipment Pricing Prediction system is crucial to provide detailed insights into the implementation, ensuring clarity and precision in the system's design, functionality, and maintenance, which is essential for developers, data scientists, and stakeholders to understand, optimize, and troubleshoot the system effectively.

1.2 Scope

The system is designed to predict shipment prices based on user inputs such as destination country, shipment mode, product group, sub-classification, brand, unit of measure, quantity, and item value. The primary objective is to provide accurate shipment price predictions to users based on historical data and machine learning algorithms.

1.3 Constraints

- Limited to predicting shipment prices for a predefined set of countries, shipment modes, product groups, sub-classifications, and brands.
- The accuracy of predictions depends on the quality and quantity of the historical data.
- The system can only handle inputs that match the formats and categories used in the training data.

1.4 Risks

- Data quality and completeness can affect prediction accuracy.
- Changes in shipping costs due to external factors (e.g., fuel price fluctuations, geopolitical events) can impact the model’s effectiveness.
- Variability in data sources and any biases in the historical data can introduce errors in predictions.

1.5 Out of Scope

- Real-time shipment price updates.
- Integration with live shipping or logistics management systems.
- Predicting shipment prices for countries, shipment modes, product groups, sub-classifications, or brands not included in the training dataset.
- Handling of edge cases or exceptions not covered by the training data, such as rare or one-off shipping scenarios.

2. Technical Specifications

The dataset consists of historical shipment data, including:

Country: The destination country for the shipment.

Shipment Mode: The mode of shipment such as Air, Truck, Air Charter, or Ocean.

Product Group: The category of the product being shipped, such as ARV, HRDT, ANTM, ACT, or MRDT.

Sub Classification: Further classification of the product group, including Adult, Pediatric, HIV test, HIV test - Ancillary, Malaria, or ACT.

Brand: The brand of the product being shipped, which could be Generic, Others, or Determine.

Unit of Measure (Per Pack): The unit of measurement per pack.

Line-Item Quantity: The quantity of items in the shipment.

Line-Item Value: The total value of the line items in the shipment.

2.2 Predicting Shipment Pricing

The system features a user interface built with Tkinter, allowing users to input shipment details.

Upon form submission, the system processes the input data, one-hot encodes categorical variables, and uses the pre-trained model to predict the shipment price.

The predicted price is then presented to the user within the interface.

2.3 Logging

To implement logging for the shipment pricing prediction system, the logging module is first imported into the Python script. Following this, logging settings are configured, defining parameters such as the log level, format, and output destination, typically at the beginning of the script or within a separate configuration file. Log messages are then strategically placed throughout the code to track the execution flow, capture errors, and record significant events. Different log levels, including DEBUG, INFO, WARNING, ERROR, and CRITICAL, are utilized depending on the severity of the message. Exception handling is employed to catch errors and

log them appropriately, ensuring that exceptions are properly recorded along with relevant information. Finally, the log file, such as 'shipment_prediction.log' in this scenario, is monitored to review logged messages and gain insights into the system's behavior. By incorporating logging into the shipment pricing prediction system, users can effectively track its execution, debug issues, and maintain visibility into its operations, enhancing system reliability and maintainability.

2.4 Database

The shipment pricing prediction system utilizes an AstraDB database powered by DataStax, accessed through the AstraPy library. It establishes a connection to the database, retrieves shipment data stored in the 'shipment' table within the 'keyspp' keyspace, and loads it into a Pandas DataFrame for further analysis and model training. The database houses crucial information such as country, shipment mode, product group, sub-classification, brand, unit of measure, line item quantity, line item value, and pack price, facilitating comprehensive analysis and accurate prediction of shipment prices.

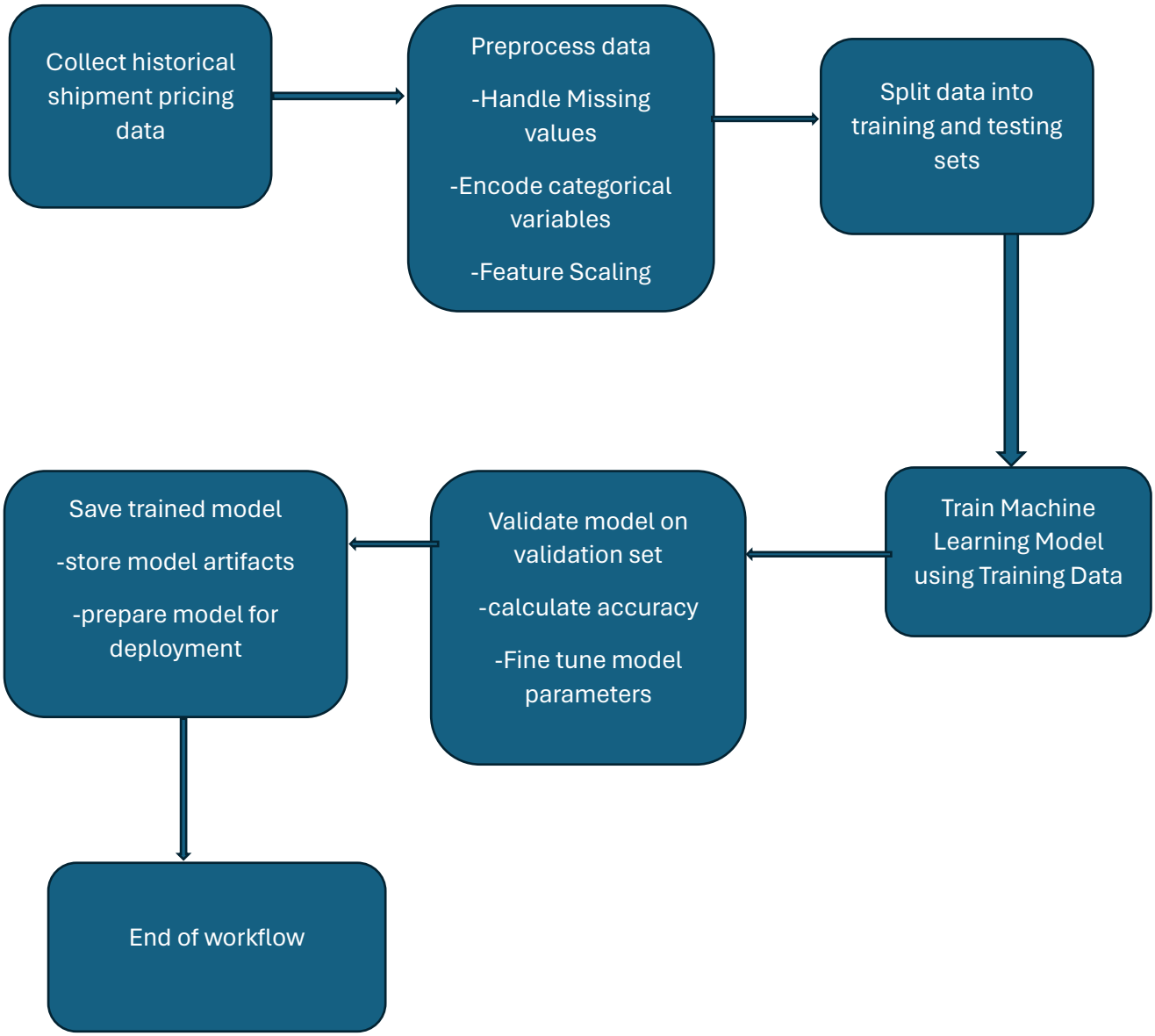
3. Technology Stack

Component	Technology
Frontend	FLASK
Database	Cassandra
Machine Learning	Scikit-learn
Deployment	Google cloud

4. Proposed Solution

The proposed solution for the shipment pricing system involves integrating Flask frame work. Flask will continue to serve as the user interface for capturing input data. With Flask, backend operations such as processing user inputs, making predictions using the pre-trained machine learning model, and serving the frontend will be handled. Flask will offer endpoints to receive user inputs, preprocess the data, execute predictions using the machine learning model, and return the predicted prices to the frontend seamlessly. The pre-trained machine learning model for predicting shipment prices will remain unchanged, integrated within the Flask backend to process user inputs and generate accurate predictions. The Cassandra database will be maintained for any required data storage or retrieval tasks. Deployment will be facilitated on Google Cloud Platform or a similar hosting service to ensure reliable and scalable application delivery. Additionally, logging and monitoring functionalities will be implemented within Flask to record relevant information about requests, errors, and application performance, ensuring smooth operation and facilitating maintenance.

5. Model Training/Validation Workflow



Collect historical shipment pricing data.

Preprocess data: handle missing values, encode categorical variables, etc.

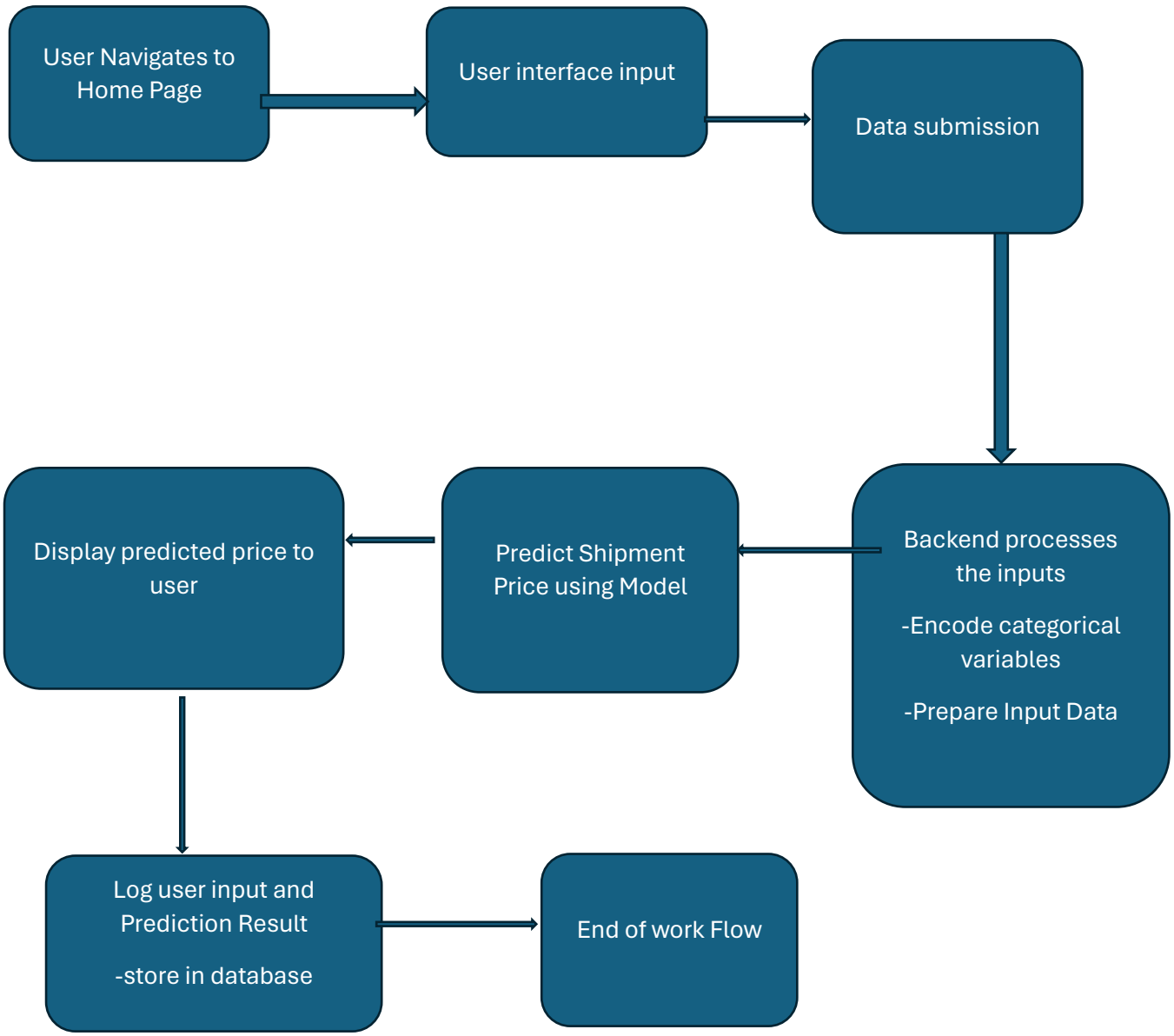
Split data into training and validation sets.

Train the machine learning model using training data.

Validate the model on the validation set to ensure accuracy.

Save the trained model to be used in the prediction system.

6. User I/O Workflow



User navigates to the home page.

User inputs country, shipment mode, product group, sub classification, brand, unit of measure

User submits the form.

Backend processes the inputs and predicts the shipment price using the machine learning model.

The predicted price is displayed to the user.

The user's inputs and prediction result are logged and stored in the database.

7. Exceptional Scenarios

Invalid Input: If the user provides invalid or incomplete input data, the system should display an error message indicating the issue and prompt the user to correct the input fields.

Model Prediction Failure: In the event of a failure during the model prediction process, the system should log the error for debugging purposes and display a generic error message to the user, informing them about the technical issue.

Database Connection Failure: If there is a failure in establishing a connection with the Cassandra database, the system should log the error and display a message to the user indicating the issue with the database connection. Additionally, the system should gracefully handle the error by providing alternative options or notifying the user when the connection is restored.

These exceptional scenarios are crucial to ensure the robustness and reliability of the Shipment Pricing Prediction System, allowing for effective error handling and user communication in various unforeseen situations.