Low Level Design

SHIPMENT PRICING PREDICTION SYSTEM

Written By	GOBIKRISHNAN, SRIRAM,SHINY
Document Version	0.3
Last **Revised** Date	04– JUNE -2024

Document Control

Change Record:

Version	Date	Author	Comments
0.1	02 – JUNE - 2024	GOBI KRISHNAN	Introduction & Architecture defined
0.2	03 – JUNE - 2024	SRIRAM	Architecture & Architecture Description appended and updated
0.3	04 – JUNE- 2024	SHINY	Unit Test Cases defined and appended

Reviews:

Version	Date	Reviewer	Comments
0.2	04 – JUNE - 2024		Document Content , Version Control and Unit Test Cases to be added

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

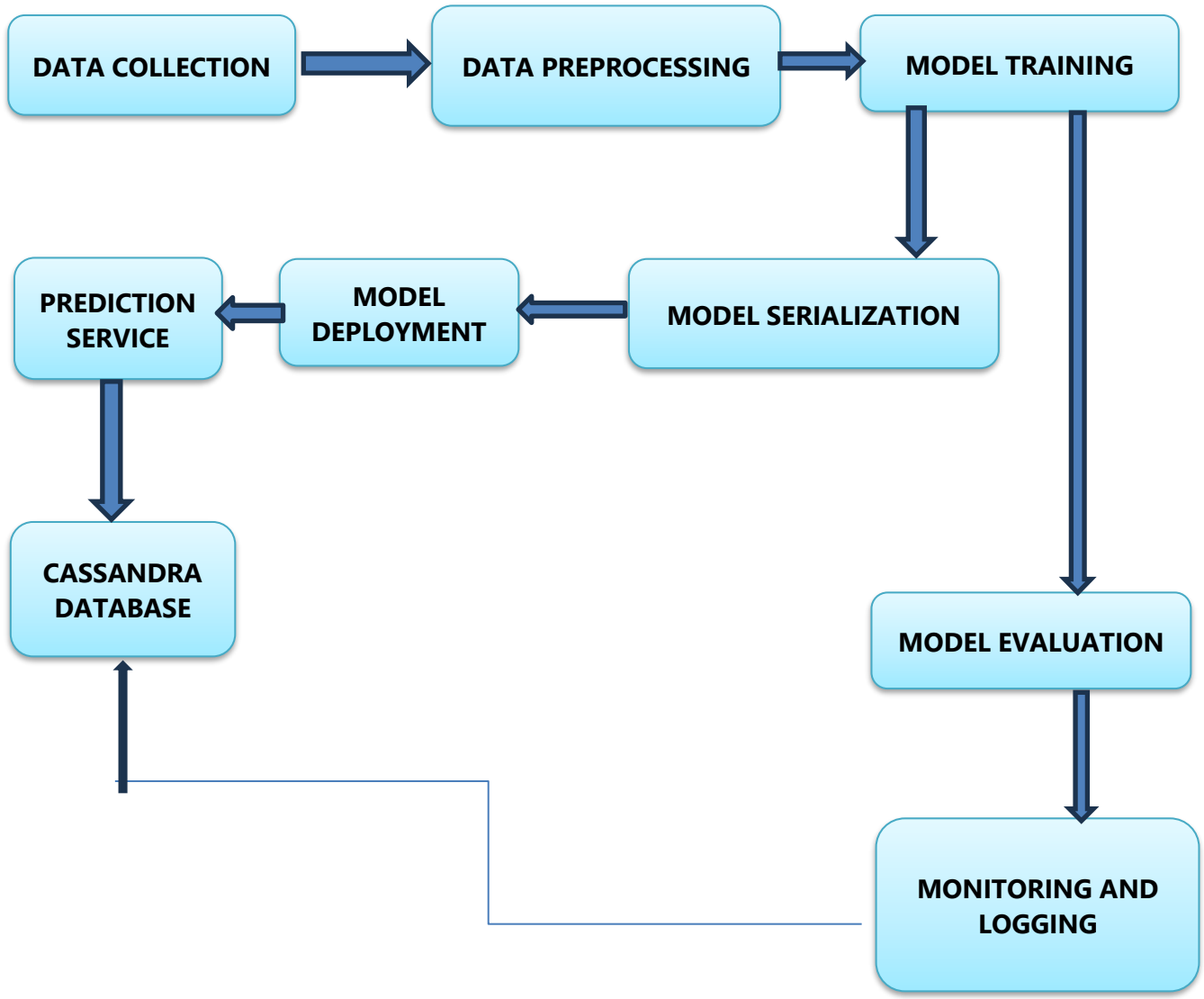
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

2. Architecture



3. Architecture Description

3.1. Data Collection

The goal of the shipment pricing prediction system is to accurately predict the cost of shipping a package based on various factors. To achieve this, a robust and comprehensive dataset is required. This dataset will consist of historical shipment records, capturing a variety of attributes that influence shipping prices..

3.2. Data Preprocessing

The data preprocessing for the shipment pricing prediction system involves several key steps to ensure the data's quality and suitability for modeling. First, the necessary columns affecting pack price are selected from the dataset. Missing values in critical columns such as 'Shipment Mode' are imputed using the most common value for the respective country. Rare categorical values in 'Dosage Form', 'Brand', and 'Country' are consolidated into an 'Others' category to reduce sparsity. Categorical variables are then one-hot encoded for model compatibility. Outliers are detected and removed using Isolation Forest to enhance data integrity. The preprocessed data is split into features (X) and target (y), and subsequently divided into training and testing sets to prepare for model training and evaluation. This comprehensive preprocessing ensures a robust and reliable dataset for predicting shipment prices.

3.3. Model Training

The model training phase for the shipment pricing prediction system involves selecting and evaluating various regression algorithms to determine the best-performing model for predicting pack prices. Initially, the preprocessed data is split into training and testing sets, with 75% of the data used for training and 25% for testing. Several regression models are considered, including Random Forest, Linear Regression, K-Nearest Neighbors, Support Vector Machine, Decision Tree, AdaBoost, XGBoost, and Polynomial Regression. Each model is trained using the training dataset, and their performance is evaluated using cross-validation. The evaluation metrics used for comparison include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score. These metrics provide a comprehensive assessment of each model's accuracy and reliability. Special handling is applied for Polynomial Regression, where the features are transformed using Polynomial Features before training. After cross-validation, the Random Forest Regressor and XGBoost Regressor are identified as top-performing models. These models are further evaluated on the testing set to confirm their predictive performance. The Random Forest Regressor shows excellent results, with low MAE, MSE, and RMSE, and a high R^2 score, both on the training and testing sets, indicating its suitability for the shipment pricing prediction task. Finally, the best-performing model, XGBoost Regressor, is saved using Joblib for deployment. This thorough model training process ensures that the selected model is accurate, reliable, and ready for real-world application in predicting shipment prices.

3.4. Model Evaluation

The model evaluation for the shipment pricing prediction system involved comparing several regression models: Random Forest, Linear Regression, K-Nearest Neighbors, Support Vector Machine, Decision Tree, AdaBoost, XGBoost, and Polynomial Regression. These models were evaluated using 5-fold cross-validation on the training set, with metrics including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score.

The Random Forest and XGBoost models demonstrated the best performance with low MAE, MSE, RMSE, and high R^2 scores. These models were further validated on the testing set, confirming their reliability and accuracy.

The final chosen model for deployment is the XGBoost Regressor due to its robust performance. The model was saved using Joblib for future use, ensuring the system's accuracy and reliability in predicting shipment prices.

The model evaluation for the shipment pricing prediction system involved comparing several regression models: Random Forest, Linear Regression, K-Nearest Neighbors, Support Vector Machine, Decision Tree, AdaBoost, XGBoost, and Polynomial Regression. These models were evaluated using 5-fold cross-validation on the training set, with metrics including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score.

The Random Forest and XGBoost models demonstrated the best performance with low MAE, MSE, RMSE, and high R^2 scores. These models were further validated on the testing set, confirming their reliability and accuracy.

The final chosen model for deployment is the XGBoost Regressor due to its robust performance. The model was saved using Joblib for future use, ensuring the system's accuracy and reliability in predicting shipment prices.

3.5. Model Serialization

To serialize the trained model for the shipment pricing prediction system, the XGBoost Regressor is first trained on the preprocessed data. After splitting the data into training and testing sets, the model is fitted using the training data. Once trained, the model is saved to disk using Joblib, which allows for efficient storage and future reuse without the need for retraining. The serialization process involves saving the model to a file named 'shipment_pricing_model.pkl'. For making predictions in the future, the model can be

deserialized from this file using Joblib, enabling consistent and reliable predictions on new data. This approach ensures that the model deployment is both efficient and effective.

3.6. Model Deployment

The shipment pricing prediction system is deployed using FLASK framework. Upon launching the application, the trained machine learning model, saved as a Joblib file, is loaded into memory. The GUI presents input fields for various shipment details, such as country, shipment mode, product group, sub-classification, brand, unit of measure per pack, line-item quantity, and line-item value. Users enter these details, and upon clicking the "Predict Price" button, the system processes the inputs, one-hot encodes categorical features, and makes a prediction using the loaded model. The predicted pack price is then displayed on the interface. This deployment enables users to conveniently access the prediction functionality, making it suitable for real-world scenarios requiring quick and accurate estimation of shipment prices.

3.7. Prediction Service

The prediction service for the shipment pricing system is designed using FLASK. It utilizes a pre-trained machine learning model to predict pack prices based on various shipment details provided by the user. Upon inputting the necessary information and clicking the "Predict Price" button, the service processes the data, applies one-hot encoding to categorical features, and generates a prediction using the loaded model. The predicted pack price is then displayed on the interface, offering users a quick and user-friendly way to obtain accurate price estimates for their shipments.

3.7 Cassandra Database

The Astra database, hosted on DataStax's managed cloud platform, likely contains shipment-related information organized in a tabular format. Each row in the "shipment" table corresponds to a specific shipment record, and each column represents a different attribute or characteristic associated with a shipment, such as country, shipment mode, product details, quantities, and prices.

By leveraging the power of Apache Cassandra, the database is capable of handling large-scale data with high availability and fault tolerance. The Python code retrieves data from this database, enabling analysis, visualization, and predictive modeling tasks to be performed using the shipment data. This integration streamlines data access and analysis, empowering users to derive valuable insights and make informed decisions based on the stored shipment information.

3.8 Monitoring and Logging

Monitoring and logging are essential components of the shipment pricing prediction system, ensuring its reliability, performance, and security. Monitoring involves real-time tracking of system health, performance metrics, and error rates, enabling proactive detection of issues. This includes

monitoring system resources, response times, and database performance. Alerts are configured to notify administrators of critical events such as downtime or high error rates. Logging, on the other hand, captures detailed information about incoming requests, prediction outcomes, errors, database access, and security-related events. By implementing robust logging using tools like Python's logging module and centralized log management systems such as ELK Stack or Splunk, the system can maintain a comprehensive record of activities for troubleshooting, auditing, and performance optimization purposes. Together, effective monitoring and logging mechanisms ensure the smooth operation and continuous improvement of the shipment pricing prediction system.

4.Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify if the prediction service URL is accessible to users	Prediction service URL should be defined	Prediction service URL should be accessible to users
Verify if the prediction service loads completely upon accessing the URL	Prediction service URL is accessible	Prediction service should load completely upon accessing the URL
Verify if users can submit valid data through the prediction service	Prediction service is accessible	Users should be able to submit valid data through the prediction service
Verify if users receive accurate fare predictions upon submitting valid input data	Prediction service is accessible	Users should receive accurate fare predictions upon submitting valid input data
Verify if users receive an error message for invalid input data	Prediction service is accessible	Users should receive an error message for invalid input data
Verify if the prediction service returns results according to user inputs	Prediction service is accessible	The prediction service should return results according to user inputs
Verify if users have options to filter the recommended results	Prediction service is accessible	Users should have options to filter the recommended results
Verify if key performance indicators (KPIs) update based on user inputs	Prediction service is accessible	Key performance indicators (KPIs) should update based on user inputs
Verify if the KPIs display details of the predicted flight fares	Prediction service is accessible	The KPIs should display details of the predicted flight fares
Verify if the prediction service handles concurrent requests gracefully	Prediction service is accessible	The prediction service should handle concurrent requests gracefully
Verify if the prediction service logs incoming requests and responses	Prediction service is accessible	The prediction service should log incoming requests and responses
Verify if the prediction service maintains response time within acceptable limits	Prediction service is accessible	The prediction service should maintain response time within acceptable limits