Low-Level Design Document

Big Mart Sales Prediction

Written By	Vijaykumar Rewate
Document Version	0.1
Last Revised Date	25.5.2023

LOW LEVEL DESIGN (LLD)

Contents

9.3.3. Model Evaluation

9.4. Data Integration and User Input

9.4.1. Data Collection

9.4.2. Data Validation

9.5. Sales Prediction and Output

9.6. Deployment

9.6.1. Dockerization

9.6.2. FastAPI Implementation

9.6.3. Exposing Train Route

9.6.4. Exposing Predict Route

9.7 Unit Test Cases

1. Introduction

1.1. What is a Low-Level Design Document?

A Low-Level Design (LLD) document provides detailed information and specifications on how the system will be implemented at a granular level. It includes specific details about the architecture, modules, algorithms, and components of the system.

1.2. Scope

The scope of this document is to outline the low-level design details for the Big Mart Store Sales Prediction project. It covers the data transformation, model building, data integration, deployment, and unit testing aspects of the system.

1.3 Constraints

- The system should be implemented using the specified technology stack, including Python, FastAPI, Docker, and MongoDB.
- The design should adhere to the predefined dataset structure and the defined machine learning algorithm (Gradient Boosting Regressor).
- The system should be scalable to handle a large volume of data and concurrent user requests.
- The deployment should be done using Docker for easy containerization and portability.

1.4 Risks

- The accuracy of the sales prediction model might be affected by factors such as data quality, feature selection, and model training techniques.
- The system performance might be impacted by large dataset size, complex computations, and resource limitations.
- There is a risk of potential data security vulnerabilities, such as unauthorized access to the database or sensitive user information.

1.5 Out of Scope

- The document does not cover the detailed implementation of the machine learning algorithms or training/validation techniques.
- The document does not include the implementation of frontend user interfaces or graphical visualizations.

2. Technical Specifications

2.1 Predicting Sales

The sales prediction will be performed using the Gradient Boosting Regressor algorithm. The model will be trained on the provided dataset and used to make predictions for the test data.

2.2 Logging

The system will implement logging to capture relevant events, errors, and information. The logging framework will be used to record system activities, aiding in debugging and troubleshooting.

2.3 Database

The system will utilize MongoDB as the database management system. It will be used to store the dataset, trained models, and user input data. The database will provide efficient data storage and retrieval capabilities.

3. Technology Stack

The Big Mart Store Sales Prediction system will be implemented using the following technology stack:

- Python: Programming language for developing the system components.
- FastAPI: Web framework for building the API endpoints and handling user requests.
- Docker: Containerization tool for packaging and deploying the system in a consistent and portable manner.
- MongoDB: Database management system for storing and accessing data.

4. Proposed Solution

The proposed solution for the Big Mart Store Sales Prediction system includes the following components:

- Data Preprocessing: Handling missing values, data transformations, and encoding categorical variables.
- Model Training: Training the Gradient Boosting Regressor model on the provided dataset.
- User Input/Output Handling: Receiving user input, performing necessary data transformations, and providing sales predictions as output.
- Logging and Error Handling: Capturing system events and errors, and handling exceptions gracefully.
- Deployment: Containerizing the system using Docker for easy deployment and scalability.

5. Model Training/Validation Workflow

The model training/validation workflow involves the following steps:

- Data preprocessing: Checking data types, handling missing values, and performing feature encoding.
- Splitting the dataset into training and validation sets.
- Training the Gradient Boosting Regressor model using the training set.
- Evaluating the model's performance on the validation set and tuning hyperparameters if necessary.

6. User I/O Workflow

- The user I/O workflow includes the following steps:
- Receiving user input through the API endpoint.
- Validating and preprocessing the input data.
- Passing the preprocessed data to the trained model for sales prediction.
- Generating the sales prediction output.
- Formatting and returning the prediction to the user through the API response.

7. Exceptional Scenarios

The system should handle exceptional scenarios and errors gracefully. Some of the exceptional scenarios to consider include:

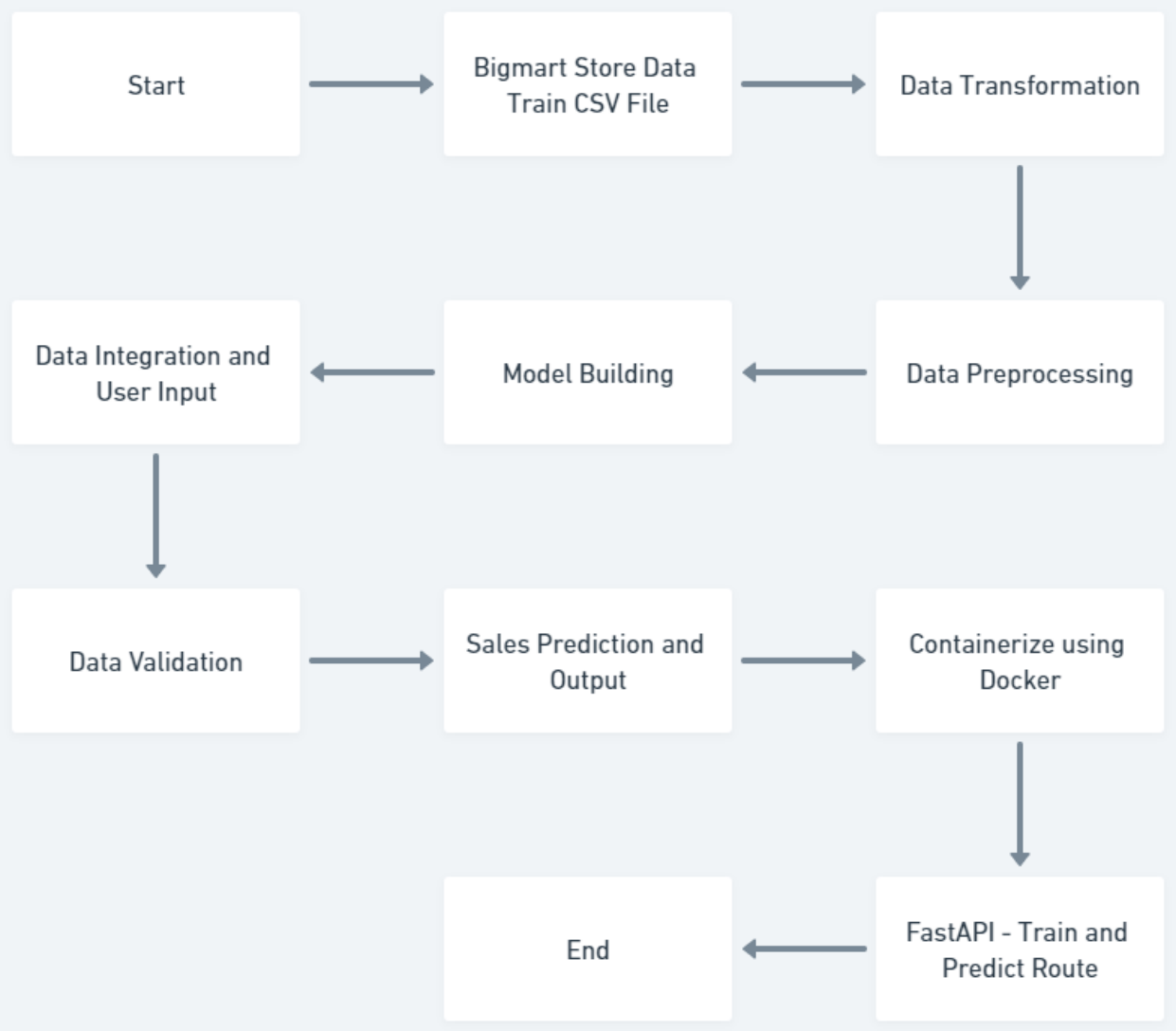
- Invalid or missing input data provided by the user.
- Connection failures to the database or external services.
- Errors during data preprocessing, model training, or prediction generation.
- Handling excessive load or concurrent user requests.

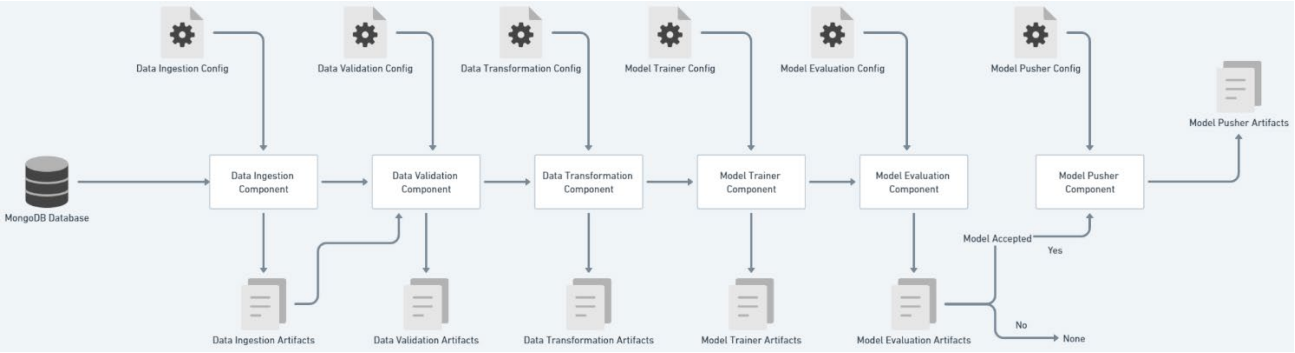
The system should incorporate appropriate error handling mechanisms, such as:

- Input data validation and error messages for invalid inputs.
- Logging and logging levels to capture errors and events for debugging purposes.
- Graceful handling of exceptions to prevent system crashes and provide informative error messages to users.
- The error handling should ensure that users are provided with meaningful feedback and the system can recover from errors without compromising its functionality.

8. Architecture

The architecture of the Big Mart Store Sales Prediction system follows a client-server model. The client interacts with the server via RESTful API endpoints exposed by the FastAPI framework.





9. Architecture Description

- The architecture follows a client-server model, where the client interacts with the server through RESTful API endpoints. The server is implemented using the FastAPI framework, which provides a high-performance and easy-to-use interface for building web APIs.
- The client application sends requests to the server, which processes the requests and returns the corresponding responses. The server performs the necessary data processing steps, including data exploration, data cleaning, feature engineering, and model building.
- The data required for prediction is stored in a MongoDB database. The server retrieves the data from the database and performs any required preprocessing, such as imputing missing values and encoding categorical features.
- The machine learning model, specifically the Gradient Boosting Regressor, is trained using the processed data. The trained model is then used to make predictions on new input data.
- The server exposes endpoints for training the model and making predictions. The training endpoint accepts the training data, performs model training, and saves the trained model for future use. The prediction endpoint accepts new input data and returns the predicted sales values.
- The entire system is containerized using Docker, allowing for easy deployment and scalability. The Docker containers can be deployed on a cloud platform or on-premises infrastructure.
- Overall, the architecture of the Big Mart Store Sales Prediction system ensures efficient data processing, model training, and prediction, providing accurate sales forecasts for the Big Mart stores.

9.1. Data Description

The dataset consists of the following features:

- Item_Identifier: Unique product ID
- Item_Weight: Weight of the product
- Item_Fat_Content: Whether the product is low fat or not
- Item_Visibility: The percentage of total display area allocated to the product
- Item_Type: The category to which the product belongs
- Item_MRP: Maximum Retail Price (list price) of the product
- Outlet_Identifier: Unique store ID
- Outlet_Establishment_Year: The year in which the store was established
- Outlet_Size: The size of the store in terms of ground area covered
- Outlet_Location_Type: The type of city in which the store is located
- Outlet_Type: Whether the outlet is a grocery store or a supermarket
- Item_Outlet_Sales: Sales of the product in the particular store (outcome variable)

9.2. Data Transformation

9.2.1. Checking Data Types

- Verify the data types of each feature in the dataset to ensure consistency and correctness.

9.2.2. Handling Missing Values

- Impute missing values in numerical columns (Item_Weight) using the Simple Imputer with the mean strategy.
- Impute missing values in categorical columns (Outlet_Size) using the Simple Imputer with the most frequent strategy.

9.2.3. Encoding Categorical Columns

- Encode categorical columns (Item_Fat_Content, Item_Type, Outlet_Identifier, Outlet_Size, Outlet_Location_Type, Outlet_Type) using One Hot Encoding and Ordinal Encoding techniques.

9.2.4. Feature Scaling

- Apply Quantile Transformer to scale the numerical features (Item_Visibility, Item_MRP).

9.3. Model Building

9.3.1. Model Selection

- Utilize the Gradient Boosting Regressor as the machine learning algorithm for the Big Mart Store Sales Prediction model.
- This algorithm is chosen for its ability to handle complex relationships and handle both numerical and categorical features effectively.

9.3.2. Model Training

- Train the Gradient Boosting Regressor model using the pre-processed data.
- Perform hyperparameter tuning to optimize the model's performance.

9.3.3. Model Evaluation

- Evaluate the performance of the trained model using appropriate metrics such as mean squared error (MSE), root mean squared error (RMSE), and R-squared score.

9.4. Data Integration and User Input

9.4.1. Data Collection

- Collect user input data for making sales predictions.
- Ensure the input data matches the required format and includes all necessary fields.

9.4.2. Data Validation

- Validate the user input data to ensure its correctness
- Check if the input data contains all the required fields.
- Validate the data types and formats of the input fields.
- Perform any additional checks or validations specific to the project requirements.

9.5. Sales Prediction and Output

- Utilize the trained Gradient Boosting Regressor model to make sales predictions based on the user input data.
- Apply the necessary transformations and preprocessing steps to the user input data before making predictions.
- Generate the predicted sales values for the given input data.

9.6. Deployment

9.6.1. Dockerization

- Containerize the Big Mart Store Sales Prediction system using Docker.
- Create a Docker file that specifies the necessary dependencies and configurations for running the system.

9.6.2. FastAPI Implementation

- Implement the FastAPI framework to create the API endpoints for the system.
- Define the routes and request handling functions for the train and predict operations.

9.6.3. Exposing Train Route

- Implement the train route to handle the training of the Gradient Boosting Regressor model.
- Specify the input parameters required for the training process.
- Save the trained model for future use.

9.6.4. Exposing Predict Route

- Implement the predict route to handle the sales prediction based on user input data.
- Specify the input parameters required for making predictions.
- Return the predicted sales values as the output.

9.7 Unit Test Cases

- Design and execute unit test cases to ensure the functionality and accuracy of the system components.
- Test the data transformation, model training, and prediction processes.
- Verify that the API endpoints and routes are working correctly.