

LOW LEVEL DESIGN (LLD)

BIGMART SALES PREDICTION

Revision Number : 1.0

Last Date of Revision : 27/09/2021

CONTENT

| Sr. No | Topic | Page No |
|--------|----------------------------|---------|
| 1 | Introduction | 1 |
| | 1.1 What Is LLD | 1 |
| | 1.2 Scope | 1 |
| 2 | Architecture | 2 |
| | 2.1 Model Flow | 2 |
| 3 | Architecture Description | 3 |
| | 3.1 Data Description | 3 |
| | 3.2 Export Data | 4 |
| | 3.3 Data Pre-processing | 4 |
| | 3.4 Feature Scaling | 4 |
| | 3.5 Model Building | 4 |
| | 3.6 Hyper Parameter Tuning | 4 |
| | 3.7 Model Testing | 4 |
| | 3.8 Model Dump | 5 |
| | 3.9 Cloud Setup | 5 |
| | 3.10 Data From User | 5 |
| | 3.11 Data Validation | 5 |
| | 3.12 Prediction | 5 |
| 4 | Technology Stack | 6 |
| 5 | Unit Test Case | 7 |

1. Introduction

1.1 What is Low-Level design document?

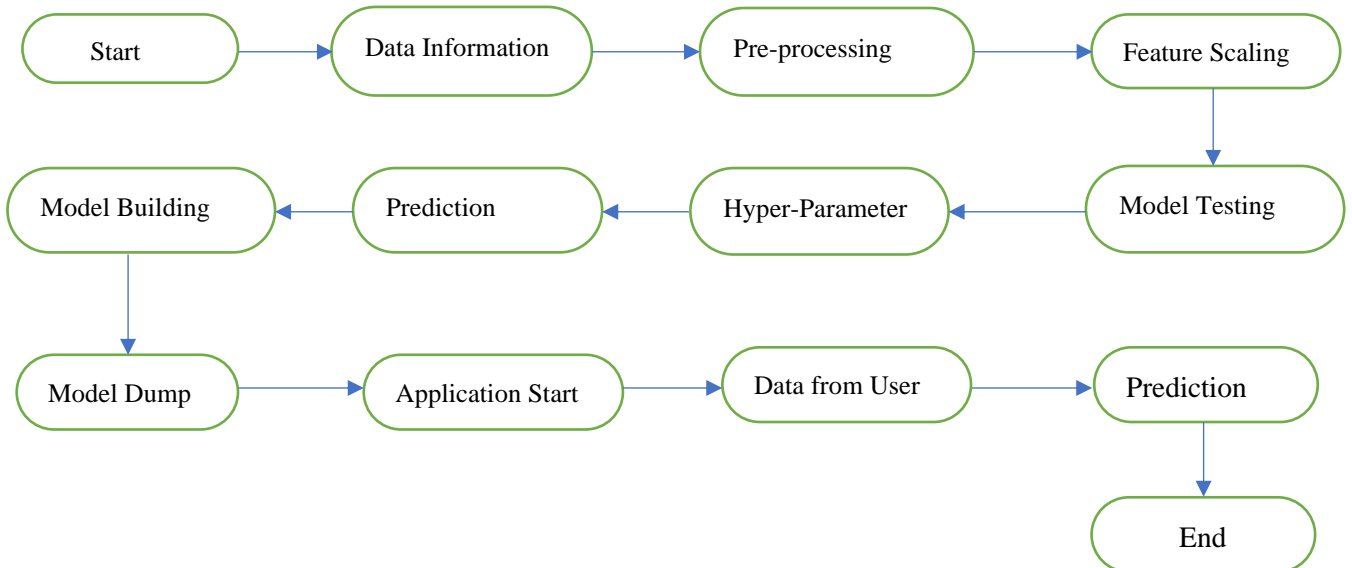
The goal of low-level design document (LLDD) is to give the internal logical design of the actual program code for Big Mart Sales Prediction. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly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

2.1 Model Flow



3. Architecture Description

3.1 Data Description

The data can be collected from the client in any file format. They perform Big Mart sales analysis using 12 different Features. The Features differ with respect to the Outlet Size, the Item Visibility, Item Type etc. The dataset comprises 8523 samples and 12 features, aiming to predict the sales. It can also be used as a multi-class classification problem if the response is rounded to the nearest integer.

The dataset contains Twelve attributes (or features, denoted by X1...X8) and two responses (or outcomes, denoted by y1 and y2). The aim is to use the eight features to predict each of the two responses.

| Indication | Names |
|------------|---------------------------|
| X1 | Item Identifier |
| X2 | Item Weight |
| X3 | Item Fat Content |
| X4 | Item Visibility |
| X5 | Item Type |
| X6 | Item MRP |
| X7 | Outlet Identifier |
| X8 | Outlet Establishment Year |
| X9 | Outlet Size |
| X10 | Outlet Location Type |
| X11 | Outlet Type |
| Y | Item Outlet Sales |

3.2 Export Data

Data Export from Database - The data in database, now we need to pull out from the database for model building.

3.3 Data Pre-Processing

In data pre-processing, we impute null values with mean and mode for respected numerical and categorical features .

3.4 Feature Scaling

The Feature Scaling technique can be used to scale down the entire feature column in one scale, so that there is not much dispersion in dataset.

3.5 Model Building

After the Encoding techniques we divide the data into train & test. The train & test data passed to the model that we are using in project i.e. Linear Regression, Elastic Net and Random Forest Regressor. Based on the score we select the model for deployment purpose. Before that we need to tune the parameter of each and every module.

3.6 Hyper Parameter Tuning:

In hyper parameter tuning we have implemented ensemble techniques like random forest regressor & regularization technique such as ridge, lasso, elastic net etc. We also done grid search cv and from that we implemented cross validation techniques for different model. From that we have chosen best parameters according to hyper parameter tuning and best score from their accuracies so we got 0.99 score for heating load and 0.96 score for cooling load for the random forest regressor model.

3.7 Model Testing

After hyper parameter tuning we put all the best parameter in our all ML model. From that we test our data & the score of the model from that we concluded the data score has been increase.

3.8 Model Dump

After comparing all accuracies and checked the score we have chosen hyper parameterized random forest regression as our best model by their results so we have dumped these model in a pickle File format with the help of pickle python module.

3.9 Cloud Setup

After model building we want to deploy the model to server. In deployment we can use different services such as Amazon Web Service (AWS), Azure Service, and Google Cloud Service (GCP). In that we deploy our model in AWS, for that 1st we have to create the environment and then after we have to create the code pipeline. After that we have to connect our pipeline with the environment.

3.10 Data from User

Here we can collect the data from the user. In which we can collect the different type of data such as Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation, Glazing Area, and Glazing Area Distribution.

3.11 Data Validation

In data validation the data from user we need to scale it down in between 0 to 1 & then we have to apply PCA so that our model can understand the data and gives a proper output.

3.12 Prediction

After entering the data and data validation when user hit the submit button we get the heating load and cooling load at the screen.

4. Technology Stack

| | |
|------------------|---|
| Front End | HTML/CSS |
| Back End | Python/Flask |
| Model | 1. Linear Regression 2.Random Forest |
| IDE | Jupyter Notebook |

5. Unit Test Case

| Test Case Description | Pre-Requisite | Expected Result |
|---|--|--|
| Verify whether the Application URL is accessible to the user | 1. Application URL should be defined | Application URL should be accessible to the user |
| Verify whether the Application loads completely for the user when the URL is accessed | 1. Application URL is accessible 2. Application is deployed | The Application should load completely for the user when the URL is accessed |
| Verify whether user is able to see input fields | 1. Application is accessible User is logged in to the application | User should be able to see input fields |
| Verify whether user is able to edit all input fields | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | User should be able to edit all input fields |
| Verify whether user gets submit button to submit the inputs | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | User should get Submit button to submit the inputs |
| Verify whether user is presented with recommended results on clicking submit | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | User should be presented with recommended results on clicking submit |
| Verify whether the recommended results are in accordance to the selections user made | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | The recommended results should be in accordance to the selections user made |