Low Level Design

Store Sales Prediction

Written By	Shivansh Jayara, Subharata Nath, Anubhav Srivastav, Rohan Sinha
Document Version	1.0
Last Revised Date	06-Sep-2021

Document Control

Change Record:

Version	Date	Author	Comments
1.0	06-Sep-2021	Shivansh Jayara Subharata Nath Anubhav Srivastav Rohan Sinha	Introduction Architecture description Unit test cases

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction 1
1.1. What is Low-Level design document? 1
1.2. Scope 1
2. Architecture 2
3. Architecture Description 3
3.1. Data Description 3
3.2. Data Gathering 4
3.3. Raw Data Validation 4
3.4. Data Transformation 4
3.5. Data Insertion into Database 4
3.7. Data Preprocessing 4
3.8. Feature Engineering 5
3.9. Parameter Tuning 5
3.10. Model Building 5
3.11. Model Saving 5
3.12. Flask Setup for Data Extraction 5
3.13. GitHub 5
3.14. Deployment 5
4. Unit Test Cases 6

1. Introduction

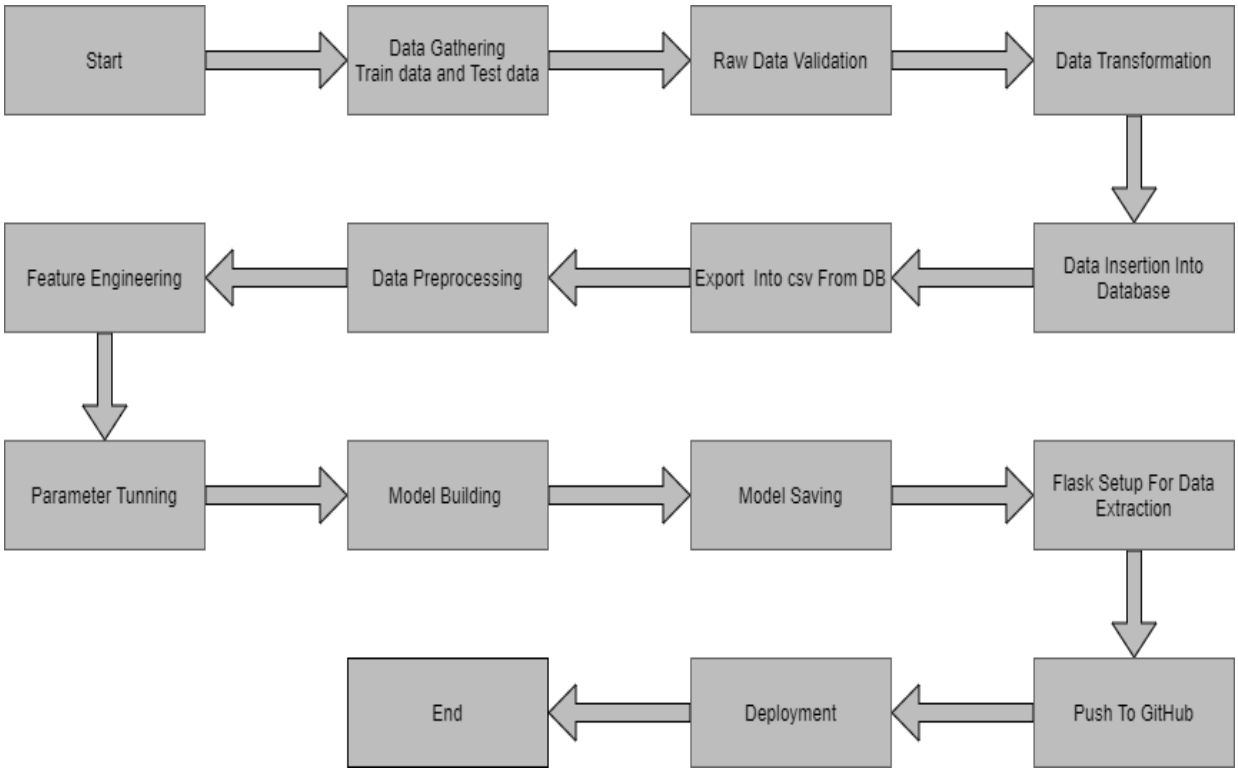
1.1. What is Low-Level design document?

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-bystep 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 Description

Given is the variable name, variable type, the measurement unit, and a brief description. The concrete compressive strength is the regression problem. The order of this listing corresponds to the order of numerals along the rows of the database.

Name	Data Type	Measurement
Item_Identifier	String	Unique product ID
Item_Weight	Float	Weight of product
Item_Fat_Content	String	Whether the product is low fat or not
Item_Visibility	Float	The % of a total display area of all products in a store allocated to the particular product
Item_Type	String	The category to which the product belongs
Item_MRP	Float	Maximum Retail Price (list price) of the product
Outlet_Identifier	String	Unique store ID
Outlet_Establishment_Year	Integer	The year in which the store was established
Outlet_Size	String	The size of the store in terms of ground area covered
Outlet_Location_Type	String	The type of city in which the store is located
Outlet_Type	String	Whether the outlet is just a grocery store or some sort of supermarket
Item_Outlet_Sales	Float	Sales of the product in the particular store. This is the outcome variable to be predicted.

3.2 Data Gathering

Data source: https://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data

Train and Test data are stored in .csv format.

3.3 Raw Data Validation

After data is loaded, various types of validation are required before we proceed further with any operation. Validations like checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because The attributes which contain these are of no use. It will not play role in contributing to the sales of an item from respective outlets.

Like if any attribute is having zero standard deviation, it means that's all the values are the same, its mean is zero. This indicates that either the sale is increasing or decrease that attribute will remain the same. Similarly, if any attribute is having full missing values, then there is no use in taking that attribute into an account for operation. It's unnecessary increasing the chances of dimensionality curse.

3.4 Data Transformation

Before sending the data into the database, data transformation is required so that data are converted into such form with which it can easily insert into the database. Here, the 'Item Weight' and "Outlet Type' attributes contain the missing values. So they are filled in both the train set as well as the test set with supported appropriate data types.

3.5 Database Insertion

Both train and test data set are inserted into the database. Here MongoDB database is used to store the data set. Separate collections were created for both train and test sets.

3.6 Export as `CSV` from Database

From the database both the train and test data set are exported into the local system and stored into CSV files. Now this CSV file will have proceeded for further processing.

3.7 Data Preprocessing

In data preprocessing all the processes required before sending the data for model building are performed. Like, here the 'Item Visibility' attributes are having some values equal to 0, which is not appropriate because if an item is present in the market, then how its visibility can be 0. So, it has been replaced with the average value of the item visibility of the respective 'Item Identifier' category. New attributes were added named ''Outlet years", where the given establishment year is subtracted from the current year. A new "Item Type" attribute was added which just takes the first two characters of the Item Identifier which

indicates the types of the items. Then mapping of “Fat content” is done based on ‘Low’, ‘Reg’ and ‘Non-edible’.

3.8 Feature Engineering

After preprocessing it was found that some of the attributes are not important to the item sales for the particular outlet. So those attributes are removed. Even one hot encoding is also performed to convert the categorical features into numerical features.

3.9 Parameter Tuning

Parameters are tuned using Randomized searchCV. Four algorithms are used in this problem, Linear Regression, Gradient boost, Random Forest, and XGBoost regressor. The parameters of all these 4 algorithms are tunned and passed into the model.

3.10 Model Building

After doing all kinds of preprocessing operations mention above and performing scaling and hyperparameter tuning, the data set is passed into all four models, Linear Regression, Gradient boost, Random Forest, and XGBoost regressor. It was found that Gradient boost performs best with the smallest RMSE value i.e. 587.0 and the highest R2 score equals 0.55. So ‘Gradient boost’ performed well in this problem.

3.11 Model Saving

Model is saved using pickle library in `.pkl` format.

3.12 Flask Setup for Data Extraction

After saving the model, the API building process started using Flask. Web application creation was created here. Whatever the data user will enter and then that data will be extracted by the model to predict the prediction of sales, this is performed in this stage.

3.13 GitHub

The whole project directory will be pushed into the GitHub repository.

3.14 Deployment

The cloud environment was set up and the project was deployed from GitHub into the Heroku cloud platform.

App link- **https://salespredictionapp.herokuapp.com/**

4. Unit Test Cases.

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 URLis accessed	1.Application URLis accessible 2.Application is deployed	The Application should load completely for the user when theURL is accessed
Verify whether a user is able to see inputfields while opening the application	1.Application is accessible 2.The user is able to see the input fields	Users should be able to see inputfields on logging in
Verify whether a user is able to enter the input values.	1.Application is accessible 2. The user is able to see the input fields	The user should be able to fill the input field
Verify whether a user gets predict button to submit the inputs	1.Application is accessible 2.The user is able to see the input fields	Users should get Submit button to submit the inputs
Verify whether a user is presented with recommended results on clicking submit	1. Application is accessible 2. The user is able to see the input fields. 3. The user is able to see the submit button	Users should be presented with recommended results on clicking submit
Verify whether a result is in accordance with the input that the user has entered	1.Application is accessible 2.The user is able to see the input fields. 3.The user is able to see the submit button	The result should be in accordance with the input that the user has entered