Low-Level Design

Stores Sales Prediction

Written By	Shaik Imran Fazil Bandi Lokesh O Dinesh Kumar Shaik Chetansha
Version	1.0
Date	07-02-2023

Document Change Control Record

Version	Date	Author	Comments

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Index

Content	Page No
1. Introduction	2
1.1 What is Low-Level Design Document	2
1.2 Scope	2
Architecture	3
2. Architecture Description	4
2.1 Data Description	4
2.2 Data Gathering	4
2.3 Raw Data Validation	4
2.4 Data Transformation	5
2.5 New Feature Generation	5
2.6 Data Preprocessing	5
2.7 Feature Engineering	5
2.8 Parameter Tuning	5
2.9 Model Building	5
2.10 Model Saving	6
2.11 Django Setup for Data Extraction	6
2.12 GitHub	6
2.13 Deployment	6
3. Unit Test Cases	6

1. Introduction

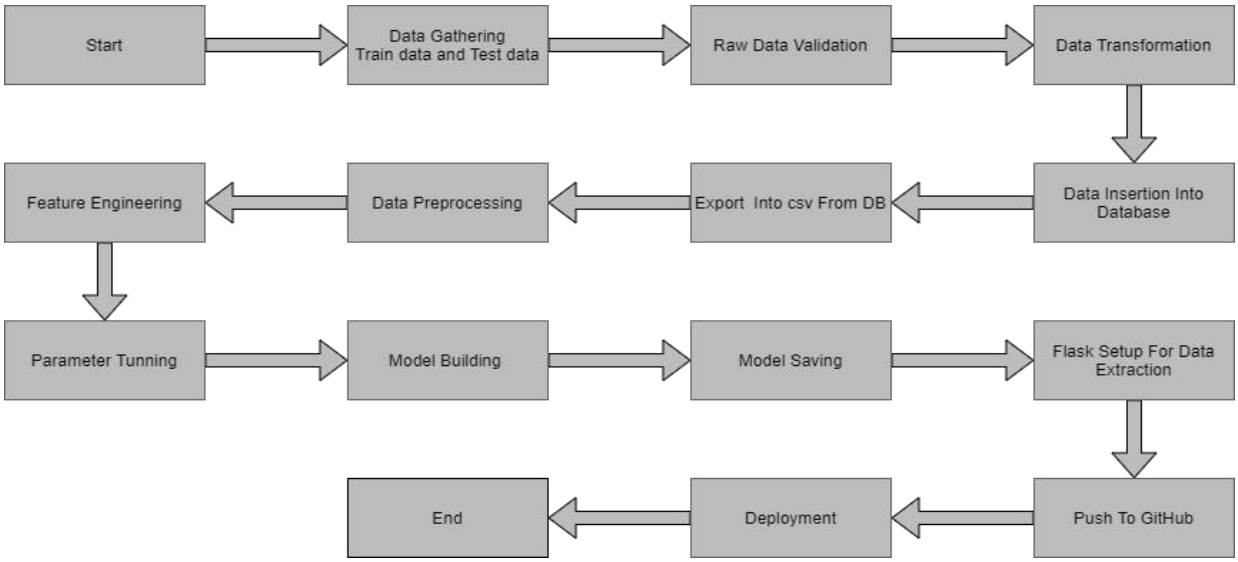
1.1 Low-Level Design Document definition.

A low-level design document (LLDD), often known as an LLD, aims to provide the internal logical structure of the actual programme code for "Stores Sales Prediction." LLD explains class diagrams that show the relationships and methods between classes and programme specifications. In order for the programmer to create the programme directly from the document, it describes the modules.

1.2 Scope

Low-level design (LLD) is an iterative refinement method that focuses on component-level design. Data structures, necessary software architecture, source code, and finally performance algorithms can all be designed using this method. Overall, during requirement analysis, the data organisation may be created, and then refined, during data design work.

Architecture :



2. Architecture Description

2.1 Describe the data

The name, type, measurement unit, and a brief explanation of the variable are all provided. The regression issue is the compressive strength of concrete. The order of the numbers along the database's rows matches the order of this listing.

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.

2.2 Data Collection

Data source: https://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data
Train and Test data are stored in .csv format.

2.3 Raw Data Validation

Before moving on with any operation, multiple sorts of validation must be performed on the loaded data. Validations include ensuring that all of the columns have a standard deviation of zero and ensuring that no columns have any complete missing data. These are necessary because the attributes that include them are useless. It won't have any impact on how much a product sells at the relevant stores.
For example, if an attribute has zero standard deviation, all of its values are the same and the attribute's mean is zero. This suggests that regardless of whether sales are up or down, the attribute will remain the same. Similar to this, it serves no purpose to take any property into account when operating if all of its values are missing. Increasing the likelihood of the dimensionality curse is needless.

2.4Transformation of Data

Data transformation is necessary before transferring the data to the database so that it can be transformed into a format that makes database insertion simple. The two properties in this case, "Item Weight" and "Outlet Type," have the missing data. So they are filled out with supported relevant data types in both the train set and the test set.

2.5new generations of features

We can construct new mrp categories as mrp bins by deriving new item categories from item types.

2.6 Preprocessing of Data

All of the steps necessary before transmitting the data for model construction are completed during data preprocessing. For instance, some of the "Item Visibility" characteristics in this instance have values of 0, which is inappropriate given that if an item is available on the market, how can its visibility be 0? In its place, the average value of item visibility for the relevant "Item Identifier" category has been used. A new characteristic called "Outlet years" was added, which subtracts the current year from the supplied establishment year. The first two characters of the Item Identifier, which represent the types of the items, are now part of a new "Item Type" attribute. Next, "Fat content" is mapped based on "Low," "Reg," and "Non-edible."

2.7 Aspect Engineering

It was discovered after preprocessing that certain of the attributes are not crucial to the

item sales for the specific retailer. Therefore, their qualities are dropped. To turn the categorical data into numerical features, even one hot encoding is carried out.

2.8 Setting parameters

Randomized searchCV is used to fine-tune the parameters. In order to solve this issue, Linear Regression and Random Forest are utilised. These two algorithms' parameters are tuned and supplied to the model.

2.9 Building a model

The data set is passed into 2 models, Linear Regression and Random Forest regressor, after completing the various preparation processes mentioned above, scaling, and hyperparameter tuning. With the lowest RMSE value, or 781.64, and the greatest R2 score, or 0.55, it was discovered that the Random Forest Regressor performs best. As a result, the "Random forest regressor" did well in this problem.

2.10 Saving models

Model is saved in ".sav" format using the pickle library.

2.11 Setting up Django for Data Extraction

After saving the model, Flask was used to begin creating the API. Here, the building of web applications was created. This stage involves taking whatever information the user enters and using the model to extract it in order to predict sales.

2.13 GitHub

The GitHub repository will be updated with the entire project directory.

2.14 Deployment

The project was uploaded from GitHub to the Heroku cloud platform after the cloud environment had been set up.

Link to the app: https://bigmartsalesprediction.herokuapp

3. Cases for unit tests.

Test Case Description	Pre-Requisite	Expected Result
Check to see if the user can reach the application's URL.	1.Application URL needs to be specified.	The user should be able to visit the application's URL
When the URL is reached, check to see if the Application loads completely for the user.	1. The application's URL is reachable 1.The application is launched	When the URL is reached, the Application should entirely load for the user.

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