Low Level Design (LLD)

Store Sales Prediction

Written By	Siddhartha Borgohain
Document Version	Initial LLD -V1.0
Last Revised Date	28-07-2022

Document Control

Change Record:

Version	Date	Author	Comments
0.1	18-05-2022	Siddhartha	First version of Document finished.

Reviews:

Version	Date	Reviewer	Comments
0.1	28-07-2022	Siddhartha	Document content updated.

Approval Status :

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

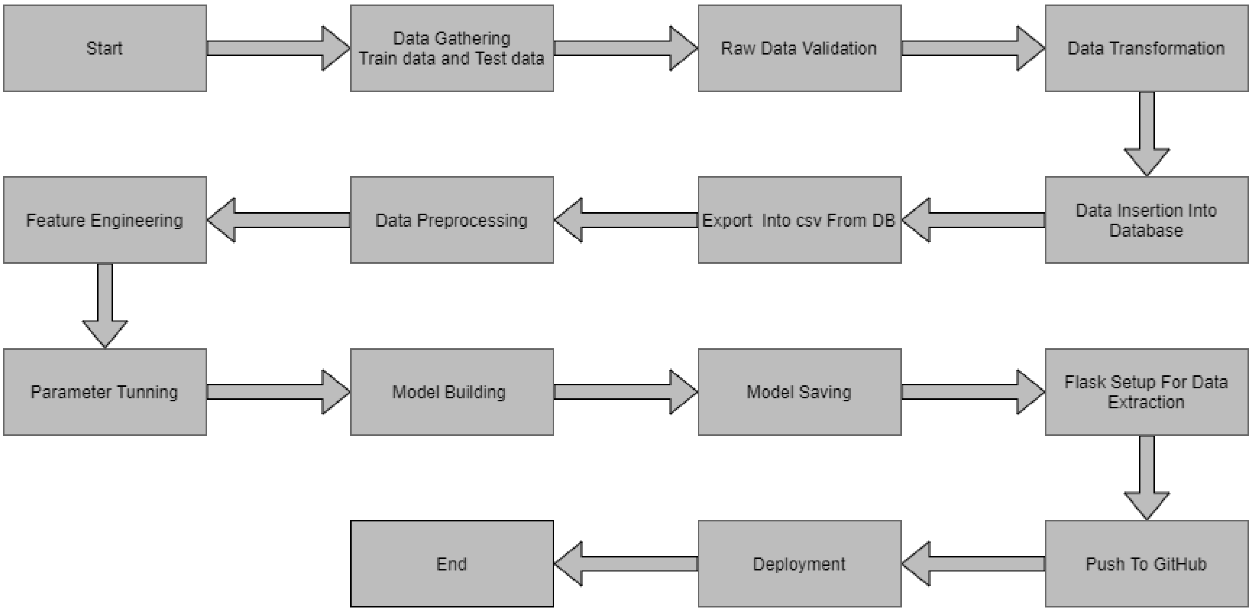
1.1. What is a Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for store 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. Architecture Description

2.1. 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.

2.2 Data Gathering

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

2.3 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 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’

2.4 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.

2.5.Parameter Tuning

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

2.6 Model Building

After doing all kinds of preprocessing operations mentioned above and performing scaling and hyperparameter tuning, the data set is passed into all four models, Linear Regression, Gradient boost and Random Forest. It was found that Gradient boost performs best with the smallest MAE value i.e. 737.3 and the highest score equals 0.60. So ‘Random Forest’ performed well in this problem.

2.7 Deployment

The cloud environment was set up and the project was deployed from GitHub into the google cloud platform.
App link- https://github.com/siddharthaborgohain/Store-Sales-Prediction

3. 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 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 should be able to see input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible	User should get Submit button to submit the inputs
Verify whether the predicted results are in accordance to the selections user made	1. Application is accessible	The predicted results should be in accordance to the selections user made