Low Level Design (LLD)

Store Sales Prediction

|  |  |
| --- | --- |
| Written By | Pavitra, Madhav , Neet |
| Document Version | Initial LLD |
| Last Revised Date | 31-03-2022 |

**Document Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **0.1** | **08-01-2022** | **Neet**  **Pavitra**  **Madhav** | **First version of Document.** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Contents

1.Introduction.....................................................................................................................4

1.1 What is Low-Level design document........................................................................4

1.2 Scope..........................................................................................................................4

2. Architecture...................................................................................................................5

3. Architecture Description...............................................................................................6

3.1 Data Description..........................................................................................................6

3.2 Data Gathering ............................................................................................................6

3.3 Data pre-processing ....................................................................................................6

3.4 Feature Engineering.....................................................................................................6

3.5 Parameter tuning……………………………………………………………………..7

3.6 Deployment…………………………………………………………………………..7

4. Unit Test Cases .............................................................................................................8

1. Introduction

1.1. What is a Low-Level design document?

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. Post-build, each component is specified in detail or in other words it can be describe as the goal of LLD 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 as well as It is used to design internals of the individual modules identified during HLD i.e., data structures and algorithms of the modules are designed and documented.

1.2. Scope

LLD helps in providing the interface for all classes that include public and private methods, including parameters, return. It also helps in describe and justify your choice of data structures as well as the major alternatives that you considered and why you made the choice.

Design documents are created to coordinate efforts of a large team, give them a stable reference point, and describe all parts of the software and how they will operate. It assures that the product is built to meet the needs and is on par with what was agreed upon prior to the inception of the product/software.

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.

|  |  |  |
| --- | --- | --- |
| 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 percentage 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**](https://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data)

Train and Test data are stored in .csv format.

3.3. Data Pre-processing

In data pre-processing 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’

3.4. Feature Engineering

A "feature" in the context of predictive modelling is just another name for a predictor variable*.*Feature engineering is the general term for creating and manipulating predictors so that a good predictive model can be created. After pre-processing 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.



3.5. Parameter Tuning

Parameter tuning is crucial as **they control the overall behaviour of a machine learning model**. Every machine learning model will have different parameters that can be set before the learning process begins. Parameters are tuned using Randomized searchCV. The algorithm which we used is Random Forest Regressor.

3.6. Deployment

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

App link- Will appear in YouTube.

App demo link - https://youtu.be/1Gt8gyJR3TA

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 user has options to  filter the recommended results as well | 1. Application is  accessible | User should have options to filter  the recommended results as well |
| 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 |
| Verify whether user has options to  filter the recommended results as well | 1. Application is  accessible | User should have options to filter  the recommended results as well |
| 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 |