Low Level Design

Store Sales Prediction

|  |  |
| --- | --- |
| Written By | Suhail Fazal |
| Document Version | 1.0 |
| Last Revised Date | 20-Feb-2022 |

**Document Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
|  |  |  |  |

**Approval Status:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

Contents

1. [Introduction 1](#_bookmark0)
   1. [What is Low-Level design document? 1](#_bookmark1)
   2. [Scope 1](#_bookmark2)
2. [Architecture 2](#_bookmark3)
3. [Architecture Description 3](#_bookmark4)
   1. [Data Description 3](#_TOC_250012)
   2. [Data Gathering 4](#_TOC_250011)
   3. [Raw Data Validation 4](#_TOC_250010)
   4. [Data Transformation 4](#_TOC_250009)
   5. Data Insertion into Database 4
   6. [Data Preprocessing 4](#_TOC_250008)
   7. [Feature Engineering 5](#_TOC_250007)
   8. [Parameter Tuning 5](#_TOC_250006)
   9. [Model Building 5](#_TOC_250005)
   10. [Model Saving 5](#_TOC_250004)
   11. [Flask Setup for Data Extraction 5](#_TOC_250003)
   12. [GitHub 5](#_TOC_250002)
   13. [Deployment 5](#_TOC_250001)
4. [Unit Test Cases 6](#_TOC_250000)

# Introduction

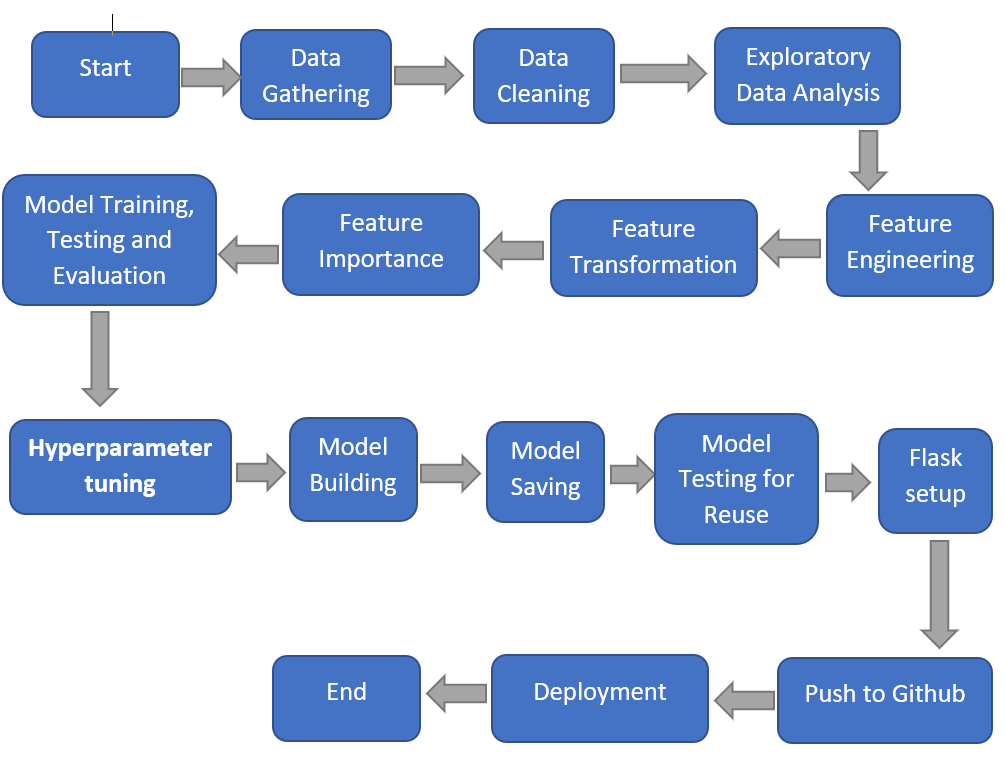
## What is Low-Level design document?

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.

## Scope

Low-level design (LLD) is a component-level design process that follows a step-bystep [refinement](https://en.wikipedia.org/wiki/Refinement_(computing)) 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

# Architecture



# Architecture Description

* 1. **Data Description**

Variable names, their types and unit of measurement is given below.

Big Mart’s data of their 10 stores situated at different locations with each store having 1559 different products. Using all the observations it is inferred what role certain properties of an item play and how they affect their sales. 8523 rows and 12 columns/variables in train data.

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Measurement |
| Item\_Identifier | String | Unique product ID |
| Item\_Weight | Float | Weight of product |
| Item\_Fat\_Content | String | Low Fat or Regular |
| 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. |

* 1. **Data Gathering**

Train and Test data was provided at INeuron Portal in .csv format.

**3.3** **Raw Data Validation and Data Cleaning**

* The Data might contain Repeated/Duplicate entries, removing them.
* Data Validity check, Item Visibility has 0 as minimum value, all items need to have some visibility.
* Missing values imputation of the products will give better output than deleting them.
* Outliers – Outliers can be valid or a data entry mistake, treating them with transformations to convert skewed data to normal distribution.

**3.4** **Exploratory** **Data Analysis**

* Uni-variate analysis = Checking how each variable is distributed.
* Bi-variate/Multi-variate analysis = how variables are related with each other and with output variable, which is Item Outlet Sales in this case.
* Pivots – to check variables and their contribution/importance on the output variable.
* Visualization and Data Insights – Visualizing data makes it much easier to interpret the data.

**3.5 Feature Engineering**

* Feature Creation: Creating new features from “Outlet Establishment Year” as “Outlet Year” subtract it with 2013 to get the age of the outlet when data was collected.
* New Item Type attribute to be created after extracting first 2 characters from the “Item Identifier” feature. FC: Food, DR: Drinks, and NC: Non-Consumables.
* Mapping and combining: Item Fat Content
  + 'LF', 'low fat’ to “Low Fat”
  + 'reg' to “Regular”

**3.6** **Feature Transformation**

* Outlier handling – Log Transformation.
* Dropping “Outlet Establishment Year”, “Item Identifier” and “Outlet Identifier”.
* Label Encoding – converting text to integer (Problem is they are ranked alphabetically)
* One Hot Encoding – encoding labels by creating new variables for each label, Dummy variable trap is handled by dropping first encoded variable.
* Feature Scaling - scaling down the data of all the numerical variables to bring them into similar scale. Ex: Item MRP has 245 and Item Visibility is 0.25 to bring them all into similar scaling to fit Gradient Descent based ML algorithms such as Linear Regression and Logistic Regression.

### **3.7 Feature Importance**

### using Extra Trees Regressor to plot features having more impact on the outlet sales.

### **3.8 Model Training and Testing**

### Trained the model on Linear Regression, Ridge and Lasso Regression, and Random Forest Regressor. Best model with highest Prediction score (R Squared) and Lowest Error (Root Mean Squared Error) is selected.

### **3.9 Model Evaluation**

### Predict and Evaluate the model on validation dataset.

**4.0 Hyperparameter tuning**

### tuning parameters to get the best score and best parameters combinations using Randomized Search Cross Validation.

**4.1 Model Building**

Building the model with suggested parameters from Hyperparameter tuned model, testing and evaluating the model. Tuned Random Forest Regressor got highest accuracy and lowest error score. With 71.56% R Squared, and RMSE of 0.5499.

**4.2 Saving the model**

Model is saved in pickle format as pkl.

**4.3 Model Testing for Reuse**

Predicting from the saved pickle file to validate if it’s working.

**4.4 Flask Setup**

Web application was created using Flask, which takes user inputs and passes it to the model to predict sales.

**4.5 Push to GitHub -** Project Directory will be pushed to Github.

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

App link-Store Sales Prediction - INeuron

1. **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 a user is able to see input fields while opening the application | 1. Application is accessible 2. The user is able to see the input fields | Users should be able to see input fields 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 |