LOW LEVEL DESIGN

Store Sales Prediction

Written By	Manu Vats
Document Version	0.1
Last Revised Date	18 July 2022

Document Control

Change Record:

Version	Date	Author	Comments
0.1	18 July 2022	Manu Vats	

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

Introduction

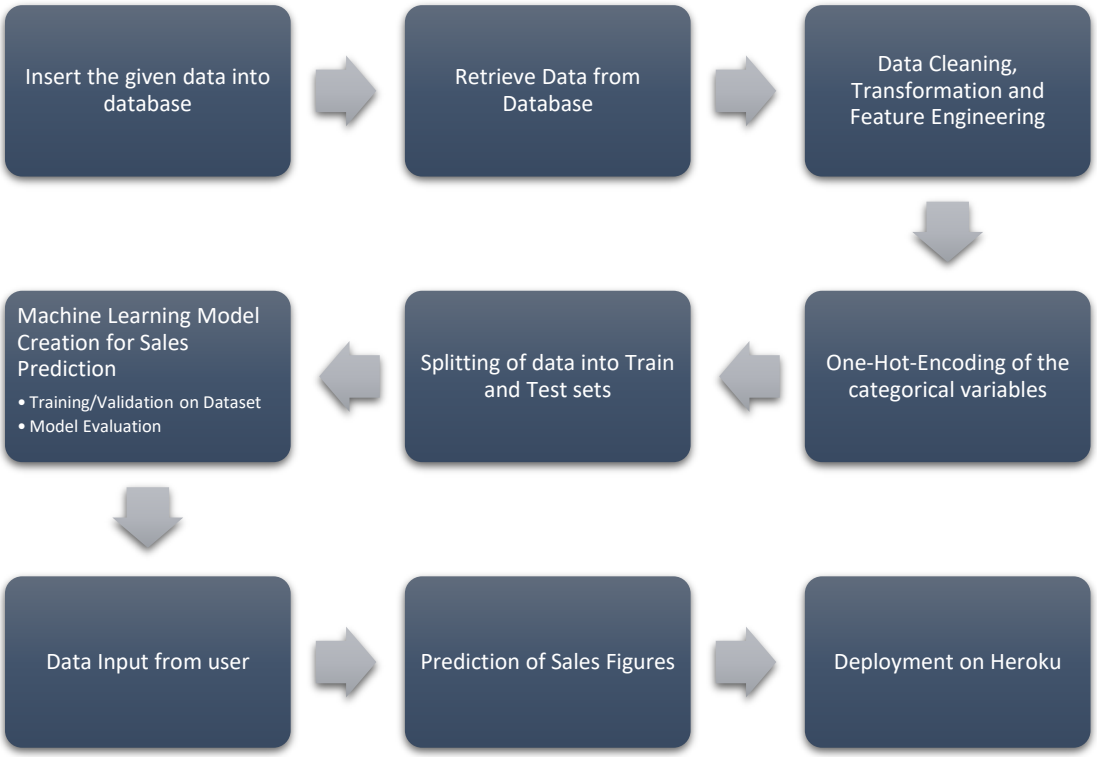
What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Store Sales Prediction 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-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

Architecture



Architecture Description

Data Description

The dataset is divided into Train and Test datasets with Train set containing around 8500 records and Test set containing around 5600 records. Train data set has both input and output variable(s). We need to predict the sales for test data set.

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

EDA

Exploratory Data Analysis is done to depict and understand the characteristics of the data before we build a prediction model. With the help of EDA, we can perform missing values imputation and appropriate data transformation to make the data better fit for model building and get higher accuracy of results.

Data Cleaning, Transformation and Feature Engineering

After performing the EDA, we can perform the necessary data cleaning, transformation and feature engineering. This includes null values imputation, standardization of the dataset.

One-Hot-Encoding

In this step, all the categorical variables are one-hot-encoded so that our Machine Learning Model can treat them as numerical variables to predict the outcome.

Machine Learning model Creation

After clusters are created, we will find the best model for each cluster. For each cluster, algorithms will be passed with the best parameters derived from Grid-Search. We will calculate the RMS scores for models and select the model with the best score. Similarly, the models will be selected for each cluster. All the models for every cluster will be saved for use in Recommendation.

Prediction of Sales Figures

After Model building, it is fed on the Test dataset to predict the sales figures.

Data from User

Here we will collect item details data from user such as item weight, type, visibility, MRP and also outlet details.

Deployment

We will be deploying the model to Heroku Cloud Platform.

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 on logging in	1. Application is accessible 2. Application is deployed	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application is accessible 2. Application is deployed	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible 2. Application is deployed	User should get Submit button to submit the inputs
Verify whether user is presented with predicted results on clicking submit	1. Application is accessible 2. Application is deployed	User should be presented with predicted results on clicking submit

	to the application 3. User is logged in to the application	
Verify whether KPIs modify as per the user inputs for the user's health	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	KPIs should modify as per the user inputs for the user's health
Verify whether the KPIs indicate details of the suggested recipe	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The KPIs should indicate details of the suggested recipe