Low Level Design (LLD)

Store Sales Prediction

Written By:

Mohamed Afaque Mulla

Version: 1.0

Date: 16-02-2023

Index

1. Introduction

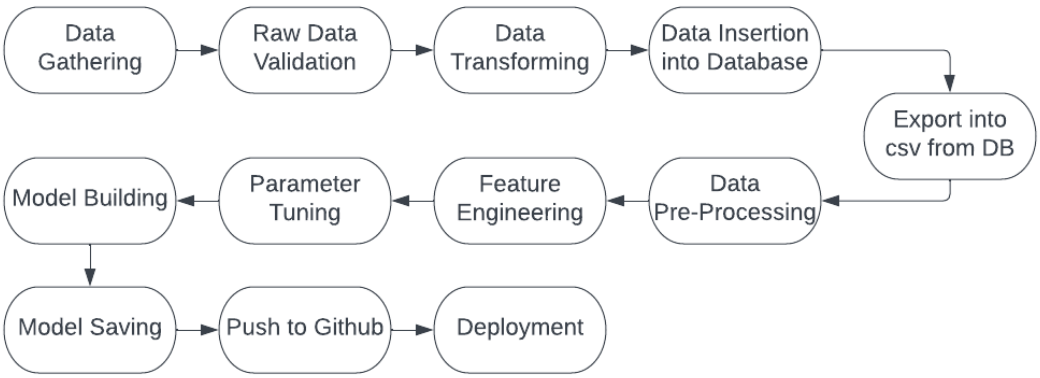
1.1 What is Low Level Design Document

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 Forecast." 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 (LLD) is an iterative refinement method that focuses on component-level design. This technique can be used for building data structures, required software architecture, source code, and eventually, performance algorithms. Overall, during requirement analysis, the data organisation may be created, and then refined, during data design work.

2. Architecture



2.1 Data Description

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.

Attributes	d_types	Description
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/datasets/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. They are necessary because the qualities 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. According 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.

2.4 Data Transformation

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.

2.6 Data Pre-processing

All of the steps necessary before transmitting the data for model development are completed in data pre-processing. 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.

2.7 Feature Engineering

It was discovered after Pre-processing that certain of the attributes are not crucial to the item sales for the specific retailer. Hence, their qualities are dropped. To turn the categorical data into numerical features, even one hot encoding is carried out.

2.8 Parameter Tuning

Randomized search CV 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 adjusted and sent to the model.

2.9 Model Building

After executing all kinds of Pre-processing operations indicated above and doing scaling and hyperparameter tweaking, the data set is passed into the models, Lasso Regression and Random Forest Regressor and Gradient Boosting Regressor. It was determined that Gradient Boosting Regressor performs best with the maximum Accuracy value 65%. As a result, the " Gradient Boosting Regressor " did well in this problem.

2.10 Model Saving

Model is saved in ".sav" format using the pickle library. So as to use the model in further applications.