Low-LevelDesign

SALES PREDICTION TOOL

Written by	SACHIN SHARMA MUNENDRA KUMAR KUSHWAHA
Version	1.0
College Name	D.S. Degree college Aligarh

Index

1. Introduction

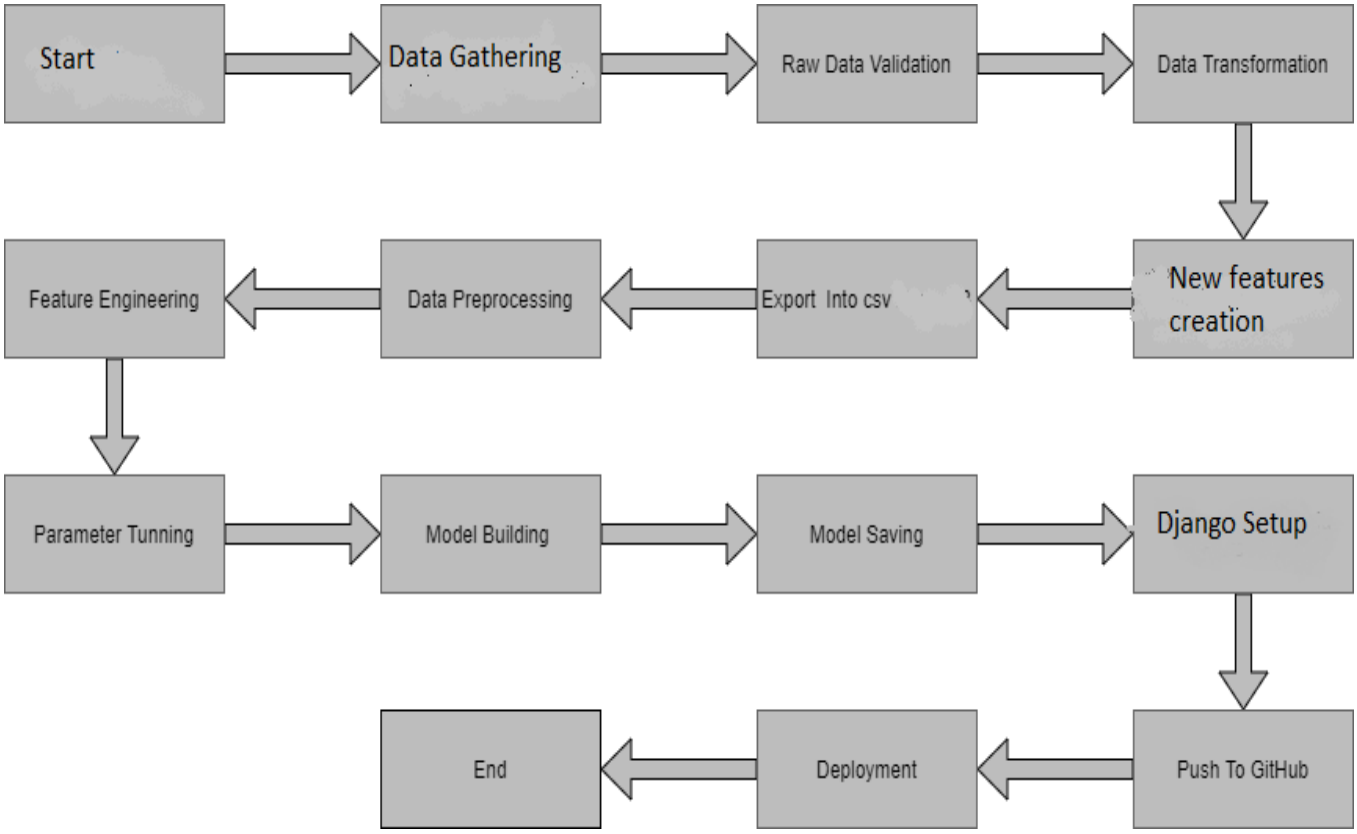
1.1 What is Low-Level Design Document.

The objective of LLD or a low-level plan report (LLDD) is to donate the inside logicaldesign oftheactual program code for ‘Stores Deals Prediction’. LLD portrays theclass graphs with themethods and relations between classes and program specs. It portrays the modules so that theprogrammercandirectlycodetheprogramfromthedocument.

1.2 Scope

Low-leveldesign(LLD)isacomponent-leveldesignprocessthatfollowsastep-by-step refinement process. This process can be used for designing data structures, required softwarearchitecture, source code, and ultimately, performance algorithms. Overall, the data organizationmaybe definedduringrequirementanalysis andthenrefinedduringdatadesignwork.

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 Raw Data Validation

After information is stacked, different sorts of approval are required some time recently we continue advance with any operation. Validations like checking for zero standard deviation for all the columns,checking for total lost values in any columns, etc. These are required since The qualities which contain these are of no utilize. It'll not play part in contributing to the deals of an thing from individual outlets. Like in the event that any trait is having zero standard deviation, it implies that's all the values are the same, its cruel is zero. This shows that either the deal is expanding or diminish that property will stay the same. Additionally, on the off chance that any quality is having full missing values, at that point there's no utilize in taking that property into an account for operation. It's pointless expanding the chances of dimensionality curse.

2.4 Data Transformation

Before sending the data into the database, data transformation is required so that data are converted into such form with which it can easily insert into the database. Here, the 'Item Weight' and "Outlet Type' attributes contain the missing values. So they are filled in both the train set as well as the test set with supported appropriate data types.

2.5 New feature generations

We can derive new item cateogory from item type and create mrp categories as mrp bin

2.6 Data Preprocessing

In information preprocessing all the forms required some time recently sending the information for demonstrate building are performed. Like, here the 'Item Visibility' properties are having a few values break even with to 0, which isn't fitting since on the off chance that an item is display within the advertise, at that point how its perceivability can be 0. So, it has been supplanted with the normal esteem of the thing perceivability of the particular 'Item Identifier' category. Modern qualities were included named ''Outlet years", where the given foundation year is subtracted from the current year. A modern "Item Type" attribute was included which fair takes the primary two characters of the Thing Identifier which shows the sorts of the things. At that point mapping of "Fat content" is done based on 'Low', 'Reg' and 'Non-edible'.

2.7 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.

2.8 Parameter Tuning

Parameters are tuned using Randomized searchCV. Four algorithms are used in this problem, Linear Regression, Gradient boost, Random Forest, and XGBoost regressor. The parameters of all these 4 algorithms are tunned and passed into the model.

2.9 Model Building

After doing all kinds of preprocessing operations mention above and performing scaling and hyperparameter tuning, the data set is passed into all four models, Linear Regression, Gradient boost, Random Forest, and XGBoost regressor. It was found that Gradient boost performs best with the smallest RMSE value i.e. 587.0 and the highest R2 score equals 0.55. So 'Gradient boost' performed well in this problem.

2.10 Model Saving

Model is saved using pickle library in `.pkl` format.

2.11Django Setup for Data Extraction

After saving the model, the API building process started using Flask. Web application creation was created here. Whatever the data user will enter and then that data will be extracted by the model to predict the prediction of sales, this is performed in this stage.

2.13 GitHub

The whole project directory will be pushed into the GitHub repository.

2.14 Deployment

The cloud environment was set up and the project was deployed from GitHub into the Heroku cloud platform.
App link- **https://salepredict.herokuapp.com/**

3. Unit Test Cases.

TestCaseDescription	Pre-Requisite	ExpectedResult
VerifywhethertheApplicationURL is accessibletotheuser	1. ApplicationURL shouldbedefined	ApplicationURLshouldbe accessibletotheuser
Verify whether the Application loadscompletely for the user when the URLisaccessed	1.Application URLisaccessible 2.Application isdeployed	The Application should loadcompletely for the user when theURLisaccessed
Verify whether a user is able to see inputfieldswhile opening the application	1.Application isaccessible 2.The user is able to see the input fields	Usersshouldbeableto seeinputfieldson loggingin
Verify whether a user is able to enter the input values.	1.Application isaccessible 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 buttontosubmitthe inputs	1.Application isaccessible 2.The user is able to see the input fields	Users should get Submit button to submittheinputs
Verify whether a user is presented with recommended resultsonclicking 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 withrecommendedresultsoncli cking submit
Verify whether a result is inaccordance withtheinput 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 inaccordance withtheinput that the user has entered