BIG MART SALES PREDICTION USING MACHINE LEARNING

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

By

Khushi Agarwal(181487)

Under the supervision of

Dr. Ekta Gandotra

Assistant Professor (SG)

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **"Big Mart Sales Prediction Using Machine Learning"** in partial fulfillment of the requirements for the award of the degree of B.Tech in CSE and submitted to the Department of CSE, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by Khushi Agarwal (181487) during the period from January 2022 to May 2022 under the supervision of Dr. Ekta Gandotra Assistant Professor (SG), Department of CSE, Jaypee University of Information Technology, Waknaghat.

Khushi

Khushi Agarwal (181487)

The above statement made is correct to the best of our knowledge.

Ekta

Dr. Ekta Gandotra

Assistant Professor(S.G)

Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

ACKNOWLEDGEMENT

Firstly, We express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

We are really grateful and wish our profound indebtedness to Supervisor Dr. Ekta Gandotra, Assistant Professor, Department of CSE Jaypee University of Information Technology, Wakhnaghat . Deep Knowledge & keen interest of my supervisor in the field of ML to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to Dr. Ekta Gandotra Department of CSE, for her kind help to finish this project.

We would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, We might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

TABLE OF CONTENTS

CONTENT
PAGE NO.

CHAPTER 1: INTRODUCTION

CHAPTER 2: LITERATURE SURVEY

CHAPTER 3: SYSTEM DEVELOPMENT

CHAPTER 4: PERFORMANCE ANALYSIS

CHAPTER 5: CONCLUSIONS

LIST OF ABBREVIATIONS

ABBREVIATION	WORD
Ml	MACHINE LEARNING
EDA	EXPLORATORY DATA ANALYSIS
FIG	FIGURE
DESCR	DESCRIPTION
RF	RANDOM FOREST
LR	LINEAR REGRESSION
CSE	COMPUTER SCIENCE AND ENGINEERING

LIST OF FIGURES

LIST OF GRAPHS

LIST OF TABLE

ABSTRACT

Nowadays many shopping malls keep track of individual item sales data in order to forecast future client demand and adjust inventory management. In order to be ahead of the competition and earn more profit one needs to create a model which will help to predict and find out the sales of the various product present in the particular store.So to predict out the sales for the big mart one need to use the very important tool i.e. Machine Learning (ML). ML is that field of computer science which gives machines ie computers the ability to learn without doing any type of programming.Using the concepts of machine and basics of data science one can build a model which can help to predict the sales of the big mart.Because of increasing competition among various shopping complex one needs to have some predictive model which could help to gain some useful insights so as to maximize the profit and be ahead of the competitors.

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **"Big Mart Sales Prediction Using ML"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **CSE** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2021 to December 2021 under the supervision of **Dr. Ekta Gandotra** (Assistant Professor(SG)).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Khushi

(Student Signature)
Student Name: Khushi Agarwal
Roll no: 181487

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Ekta

(Supervisor Signature)
Supervisor Name : Dr. Ekta Gandotra
Designation: Assistant Professor
Department name: CSE

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

The daily competition between different malls as well as big malls is becoming more and more intense because of the rapid rise of international supermarkets and online shoppings. Every mall or mart tries to provide personal and short-term donations or benefits to attract more and more customers on a daily basis, such as the sales price of everything which is usually predicted to be managed through different ways such as corporate asset management, logistics, and transportation service, etc. Current machine learning algorithms that are very complex and provide strategies for predicting or predicting long-term demand for a company's sales, which now also help in overcoming budget and computer programs.

In this report, we basically discuss the subject of specifying a large mart sale or predicting an item for a customer's future need in a few supermarkets in various locations and products that support the previous record. Various ML algorithms such as linear regression, random forest, etc. are used to predict sales volume. As we know, good marketing is probably the lifeblood of all organizations, so sales forecasting now plays an important role in any shopping mall. It is always helpful to predict the best, and develop business strategies about useful markets and to improve market knowledge. Regular sales forecasting research can help in-depth analysis of pre-existing conditions and conditions and then, assumptions are often used in terms of customer acquisition, lack of funding, and strength before setting budgets and marketing plans for the coming year.

In other words, sales forecasts are predicted on existing services of the past. In-depth knowledge of the past is required to develop and enhance market opportunities no matter what the circumstances, especially the external environment, which allows to prepare for the future needs of the business. Extensive research is ongoing in the retailer's domain to predict long-term sales demand. An important and effective method used to predict the sale of a mathematical method, also called the conventional method, but these methods take more time to predict sales. And these methods could not manage indirect data so to overcome these problems in traditional methods the machine learning techniques used. ML methods can handle not only indirect data but also large data sets well.

1.2 PROBLEM STATEMENT

Due to increasing competition many malls and bigmart are trying their best to stay ahead in competition.In order to find out what are the various factors which affect the sales of bigmart and what strategies one needs to employ in order to gain more profit one need to have some model on which they can rely .So a predictive model can be made which could help to gain useful information and increase profit.

1.3 OBJECTIVES

Objectives of these project are:

a) Predicting future sales from a given dataset.

b) To understand the key features that are responsible for the sale of a particular product.

c) Find the best algorithm that will predict sales with the greatest accuracy.

1.4 METHODOLOGY

Figure 1.1 represents the steps of building a model. Following are the steps which one needs to follow while creating a model.

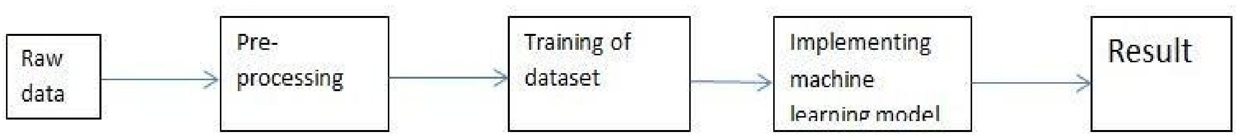


Fig 1.1: Process of building a model.

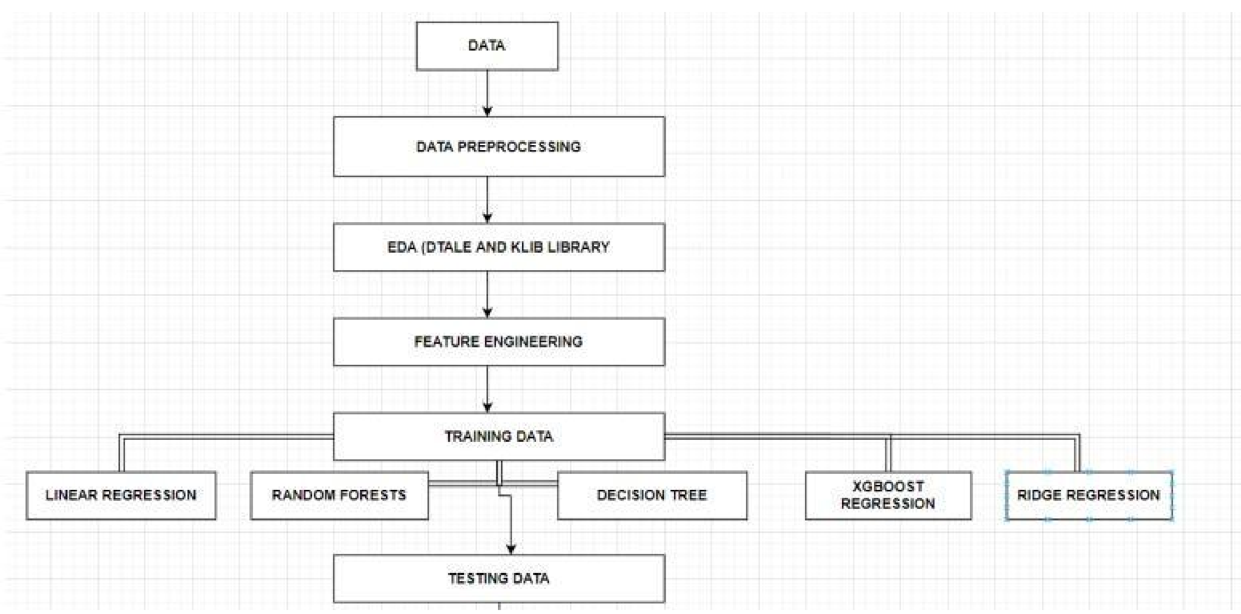


Fig 1.2:Working procedure of proposed model

1. Data collection- The step of every project is to collect the data.

   We collected our data from the Kaggle whose link is given below-

   https://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data/code

2. Data preprocessing-In this step we basically clean our dataset for example check for any missing value in the dataset , if present then handle the missing values. In our dataset attributes like Item Weight and Outlet Size had the missing value.

3. EDA-This part is considered as one of the most important parts when it comes to data analysis.To gain important insights of our data one must need to do exploratory data analysis.Here in our project we used two libraries i.e. klib and dtale library.

4. Tested various algorithms-Then various algorithms like simple LR, xgboost algorithm were applied in order to find out which algorithm can be used to predict the sales.

5. Building the model -After completing all the previous phases which are mentioned above, now our dataset is ready for further phases that is to build the model.
   Once we built the model now it is ready to be used as a predictive model to forecast sales of Big Mart.

6. Web deployment-Finally once the prediction can be made for making it more user friendly we have used web development.

.

CHAPTER 2

LITERATURE SURVEY

2.1. LITERATURE SURVEY

Kadam,et.al [1] have suggested when the prediction for the sales for bigmart was done using the algorithm like random forest and LR for prediction analysis it gave lesser accuracy.So to overcome this problem we can use another algorithm which is XG boost algorithm which not only gives better accuracy but also is more efficient.

Makridakis, et.al [2] have suggested predicting methods and applications containing Data Lack and short life cycles. So some data like historical data, consumer-focused markets face uncertain needs, which can be an accurate predictor of outcome.

C. M. Wu , et.al [3] have suggested comparison of Different ML Algorithms for Multiple Regression on Black Friday Sales Data used the concept of neural network to compare the various different algorithms.Using neural network as the concept which is very complex and less efficient concluded that we should use much simpler algorithm for the prediction purpose.

Das, et.al [4] have suggested in the prediction of retail sales of footwear which used recurrent Neural Networks and feed forward used the neural network to predict the sales.Using neural network for predicting the sales which is not an efficient method so XGboost algorithm can be used.

S. Cheriyan, et.al [5] have suggested in the study they implemented three ML algorithms on the given dataset and the models for evaluating the performance. Based upon the testing the algorithm which gave maximum accuracy was chosen for the prediction which was found to be a gradient boosting algorithm.

A. Krishna, et.al[6] have suggested that both the normal regression and boosting algorithms were implemented and found out that boosting algorithms have better results than the regular algorithms.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 ALGORITHMS EMPLOYED

3.1.1 LINEAR REGRESSION (LR)

As we know Regression can be termed as a parametric technique which means we can predict a continuous or dependent variable on the basis of a provided datasets of independent variables.

The Equation of simple LR is:

$Y = \beta o + \beta 1X + \in$------------ (1)

where,

Y : It is basically the variable which we used as a predicted value.

X : It is a variable(s) which is used for making a prediction.

βo : It is said to be a prediction value when X=0.

$\beta 1$: when there is a change in X value by 1 unit then Y value is also changed. It can also be said as slope term $\in$

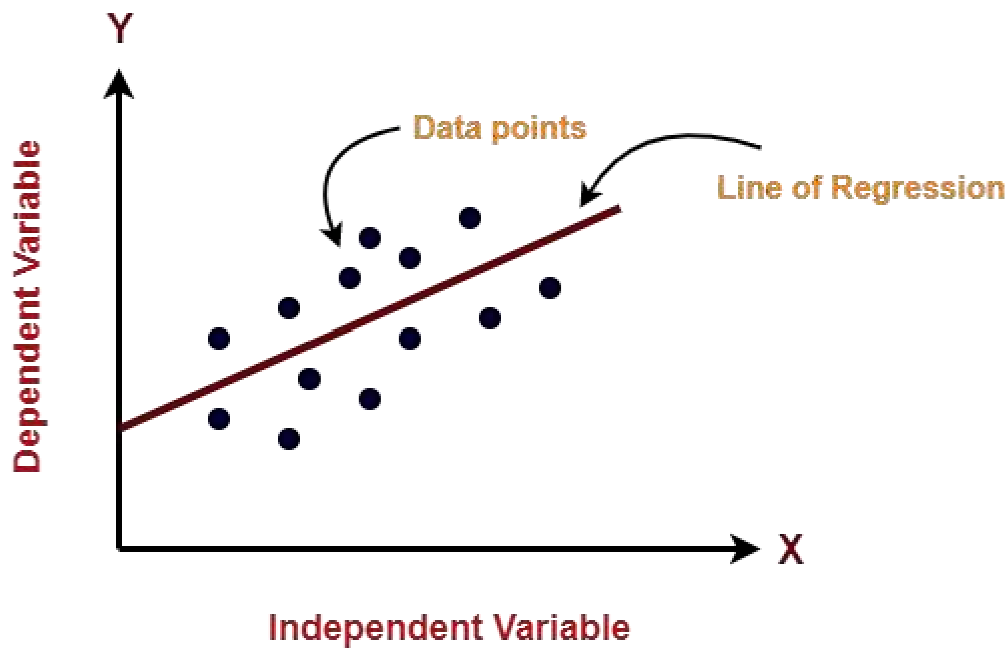


Fig 3.1 Given figure represent line of regression

3.1.2 RANDOM FOREST REGRESSION

Random Forest is a tree-based bootstrapping algorithm based on that tree that includes a certain number of decision trees to build a powerful predictive model. Individual learners, a set of random lines and a randomly selected few variables often create a tree of choice. The final prediction may be the function of all predictions made by each learner. In the event of a regression. The final prediction may be the meaning of all the predictions.

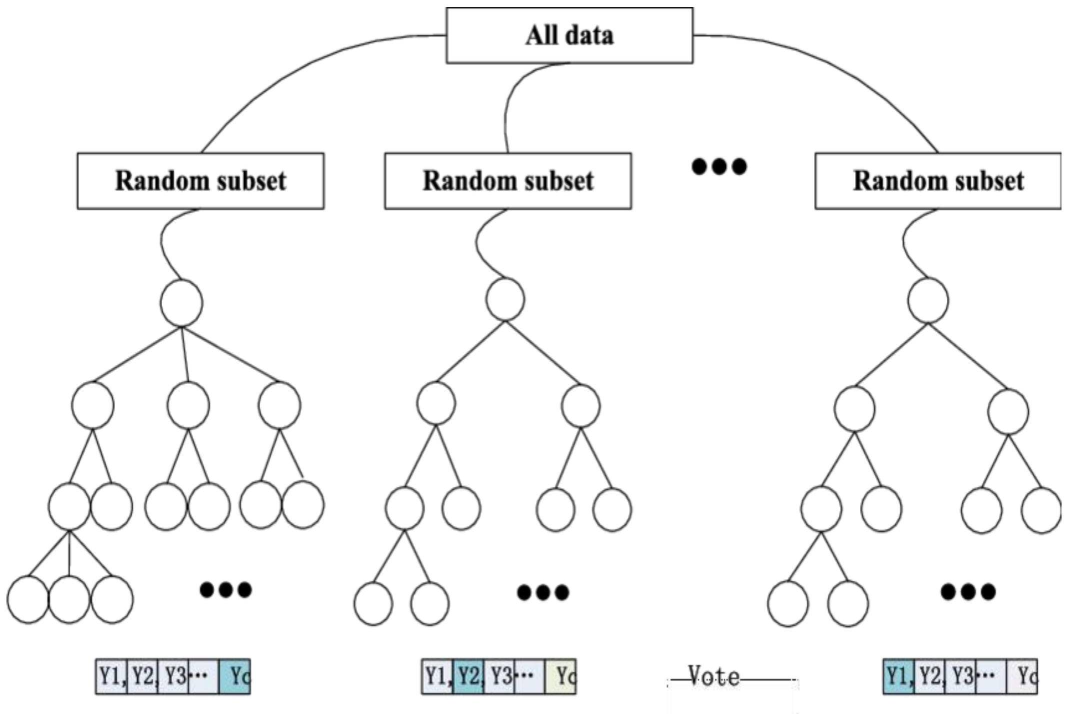


Fig 3.2 : Flowchart of Random Forest Regression

HYPER PARAMETER TUNING

In ML, optimization of the hyperparameter or problem solving by selecting the correct set of parameters for the learning algorithm. To control the learning process a hyperparameter parameter value is used.In contrast, the values of some parameters are calculated.

The same type of ML model may require different types of weights, learning scales or constraints in order to make different data and information patterns more general. The steps are also called hyperparameters and must be used for the model to solve the ML problem.

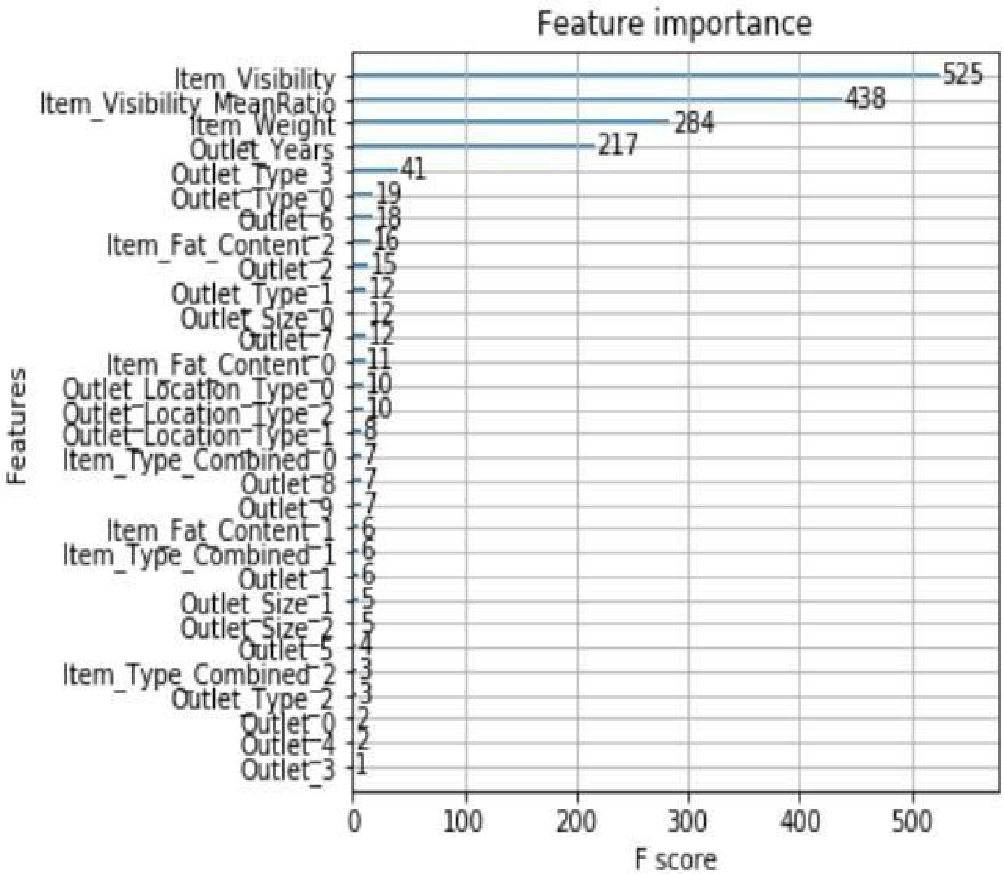


Fig 3.3: Relationship between Feature Importance and their F score in Hyper parameter tuning

XGBOOST REGRESSION

XGBoost stands for eXtreme Gradient Boosting. The implementation of an algorithm designed for the efficient operation of computer time and memory resources. Boosting is a sequential process based on the principle of the ensemble. This includes a collection of lower learners as well improves the accuracy of forecasts.No model prices n heavy for any minute t, based on the results of the previous t-speed. Well-calculated results are given less weight, and the wrong ones are weighed down. With this algorithm system

The XGBoost model uses stepwise, ridge regression internally, automatically selecting features as well as deleting multicollinearity.

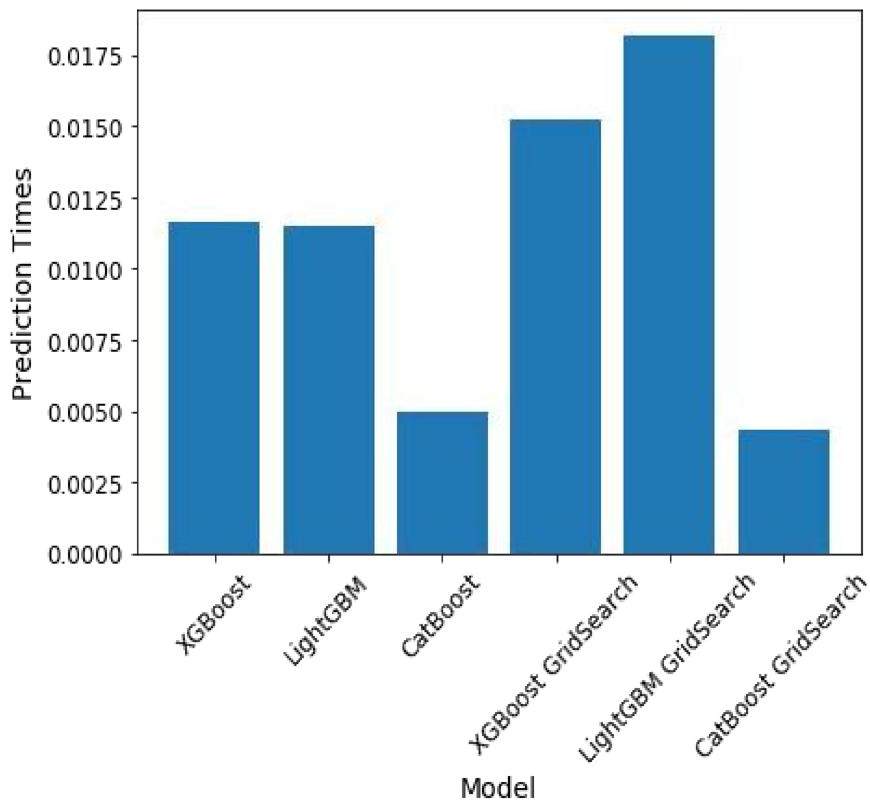


Fig 3.4 : Represents the types of XGBoost regression

3.2 PHASE OF MODEL

3.2.1 DATA AND ITS PREPROCESSING

In our work, we have used the 2013 Big Mart sales data as a database. Where the data set contains 12 features such as Item Fat, Item Type, MRP Item, Output Type, Object Appearance, Object Weight, Outlet Indicator, Outlet Size, Outlet Year of Establishment, Type of Exit, Exit Identity, and Sales. In these different aspects of responding to the Item Outlet Sales features as well, the other features are also used as the predictive variables. Our dataset has in total 8523 products in various regions and cities. The data set is also based on product level and store-level considerations . Where store level includes features such as city, population density, store capacity, location, etc. and product-level speculation involves factors such as product, ad, etc. After all considerations, a data set is finally created, then the data set is split into two parts that are tested and trained in a ratio of 80:20.

Variable	Description
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 particulat store. This is the outcome variable to be predicted.

Fig 3.5: Depicting the features of the dataset

```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

Fig 3.6 : How libraries, train and test datasets are imported.

```
df_train.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3

Fig 3.7 Head function representing first five dataset

Item_Visibility has a value = 0 as values which have no meaning, Item_Identifier is a character string with some specific code used by the bigmart and Outlet_Size contains some missing values as well.

```
[ ] df_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item_Identifier            8523 non-null   object 
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object 
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object 
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object 
 7   Outlet_Establishment_Year  8523 non-null   int64  
 8   Outlet_Size                6113 non-null   object 
 9   Outlet_Location_Type       8523 non-null   object 
 10  Outlet_Type                8523 non-null   object 
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

Fig 3.8 : Description of dataset using info() method

In figure 3.8 we can clearly see that there are in total 12 features out of which Numeric data count is 5 and Categorical data count is 7.

```
[ ] df_train.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

Fig 3.9 : Description of dataset using describe() method

In figure 3. Item_Visibility feature has a minimum value of 0.00 and Item_weight has count of 7060.

3.2.2 HANDLING MISSING VALUES

While analyzing the dataset we come across some missing values in the dataset.In order to check for the missing value we have the following code-

```
df_train.isnull().sum()
```

```
Item_Identifier                 0
Item_Weight                  1463
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  2410
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64
```

```
[5] df_test.isnull().sum()
```

```
Item_Identifier                 0
Item_Weight                   976
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  1606
Outlet_Location_Type            0
Outlet_Type                     0
dtype: int64
```

Fig 3.10 Depicts the number of missing value

From the above Fig 3.10 we can clearly see that column names item_weight and outlet_size have 976 and 1606 missing values respectively.

In order to handle these missing values we have different approaches for e.g. dropping the rows having missing value or filling the missing value with suitable values using different methods. Looking at our dataset we have 8523 rows so dropping would not be a better option as it would lead to decrease the prediction accuracy.

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

Fig 3.11 Datatype of various features of dataset

Since item_weight is a numerical feature, filling its missing value using the average imputation method.

```
[8] df_train['Item_Weight'].fillna(df_train['Item_Weight'].mean(),inplace=True)
    df_test['Item_Weight'].fillna(df_test['Item_Weight'].mean(),inplace=True)
```

```
[9] df_train.isnull().sum()

    Item_Identifier                 0
    Item_Weight                     0
    Item_Fat_Content                0
    Item_Visibility                 0
    Item_Type                       0
    Item_MRP                        0
    Outlet_Identifier               0
    Outlet_Establishment_Year       0
    Outlet_Size                  2410
    Outlet_Location_Type            0
    Outlet_Type                     0
    Item_Outlet_Sales               0
    dtype: int64
```

Fig 3.12 Missing value in outlet_size column = 2410

```
df_train['Outlet_Size'].fillna(df_train['Outlet_Size'].mode()[0],inplace=True)
df_test['Outlet_Size'].fillna(df_test['Outlet_Size'].mode()[0],inplace=True)
```

Fig 3.13: Filling Values in Outlet_Size.

Outlet size is a categorical feature so filling the value using the mode imputation method

So finally -

```
df_train.isnull().sum()
```

```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64
```

Fig 3.14 : Now there are no missing values in the item_weight and Outer_size columns.

3.2.3 EDA

a) EDA WITH DTALE LIBRARY

D-Tale is a Flask and React-based powerful tool which is used to analyze and visualize pandas' data structure seamlessly.

D-Tale also supports objects like Data Frame, Series, etc.

```
[ ] import dtale
```

```
[ ] #dtale.show(df_train)
    import dtale.app as dtale_app
    dtale_app.USE_COLAB = True
    dtale.show(df_train)
```

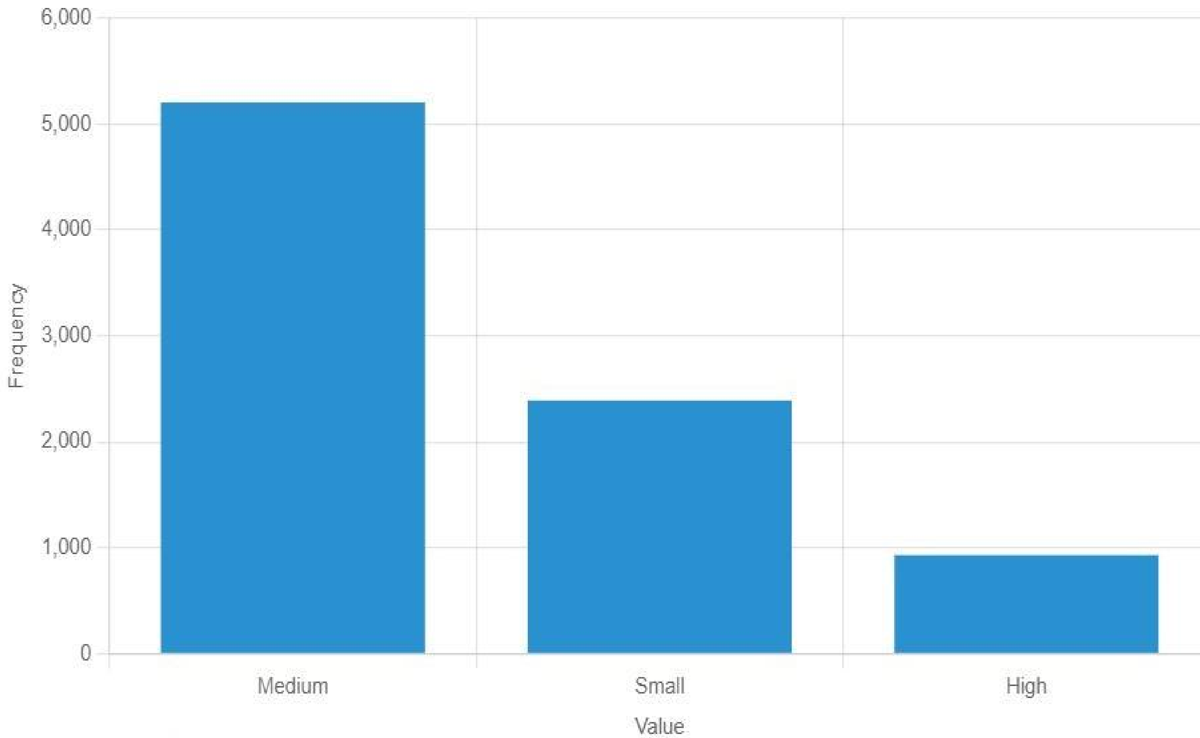
https://rehxjuyyoz-496ff2e9c6d22116-40000-colab.googleusercontent.com/dtale/main/2

Fig 3.15: Represents how to import dtale library and display the table

10 8523	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	9.30	Low Fat	0.02	Dairy	249.81	1999	Medium	Tier 1	Supermarket Type1	3735.14
1	5.92	Regular	0.02	Soft Drinks	48.27	2009	Medium	Tier 3	Supermarket Type2	443.42
2	17.50	Low Fat	0.02	Meat	141.62	1999	Medium	Tier 1	Supermarket Type1	2097.27
3	19.20	Regular	0.00	Fruits and Vegetables	182.10	1998	nan	Tier 3	Grocery Store	732.38
4	8.93	Low Fat	0.00	Household	53.86	1987	High	Tier 3	Supermarket Type1	994.71
5	10.40	Regular	0.00	Baking Goods	51.40	2009	Medium	Tier 3	Supermarket Type2	556.61
6	13.65	Regular	0.01	Snack Foods	57.66	1987	High	Tier 3	Supermarket Type1	343.55
7	12.86	Low Fat	0.13	Snack Foods	107.76	1985	Medium	Tier 3	Supermarket Type3	4022.76
8	16.20	Regular	0.02	Frozen Foods	96.97	2002	nan	Tier 2	Supermarket Type1	1076.60
9	19.20	Regular	0.09	Frozen Foods	187.82	2007	nan	Tier 2	Supermarket Type1	4710.54
10	11.80	Low Fat	0.00	Fruits and Vegetables	45.54	1999	Medium	Tier 1	Supermarket Type1	1516.03
11	18.50	Regular	0.05	Dairy	144.11	1997	Small	Tier 1	Supermarket Type1	2187.15
12	15.10	Regular	0.10	Fruits and Vegetables	145.48	1999	Medium	Tier 1	Supermarket Type1	1589.26
13	17.60	Regular	0.05	Snack Foods	119.68	1997	Small	Tier 1	Supermarket Type1	2145.21
14	16.35	Low Fat	0.07	Fruits and Vegetables	196.44	1987	High	Tier 3	Supermarket Type1	1977.43
15	9.00	Regular	0.07	Breakfast	56.36	1997	Small	Tier 1	Supermarket Type1	1547.32
16	11.80	Low Fat	0.01	Health and Hygiene	115.35	2009	Medium	Tier 3	Supermarket Type2	1621.89
17	9.00	Regular	0.07	Breakfast	54.36	1999	Medium	Tier 1	Supermarket Type1	718.40
18	12.86	Low Fat	0.03	Hard Drinks	113.28	1985	Medium	Tier 3	Supermarket Type3	2303.67
19	13.35	Low Fat	0.10	Dairy	230.54	2004	Small	Tier 2	Supermarket Type1	2748.42
20	18.85	Regular	0.14	Snack Foods	250.87	1987	High	Tier 3	Supermarket Type1	3775.09

21	12.86	Regular	0.04	Baking Goods	144.54	1985	Medium	Tier 3	Supermarket Type3	4064.04
22	14.60	Low Fat	0.03	Household	196.51	2004	Small	Tier 2	Supermarket Type1	1587.27
23	12.86	Low Fat	0.06	Baking Goods	107.69	1985	Small	Tier 1	Grocery Store	214.39
24	13.85	Regular	0.03	Frozen Foods	165.02	1997	Small	Tier 1	Supermarket Type1	4078.03
25	13.00	Low Fat	0.10	Household	45.91	2007	nan	Tier 2	Supermarket Type1	838.91
26	7.65	Regular	0.07	Snack Foods	42.31	2004	Small	Tier 2	Supermarket Type1	1065.28
27	11.65	low fat	0.02	Hard Drinks	39.12	1987	High	Tier 3	Supermarket Type1	308.93
28	5.93	Regular	0.16	Dairy	45.51	1998	nan	Tier 3	Grocery Store	178.43
29	12.86	Regular	0.07	Canned	43.65	1985	Small	Tier 1	Grocery Store	125.84
30	19.25	Low Fat	0.17	Dairy	55.80	1998	nan	Tier 3	Grocery Store	163.79
31	18.60	Low Fat	0.08	Health and Hygiene	96.44	2009	Medium	Tier 3	Supermarket Type2	2741.76
32	18.70	Low Fat	0.00	Snack Foods	256.67	2009	Medium	Tier 3	Supermarket Type2	3068.01
33	17.85	Low Fat	0.00	Breads	93.14	2002	nan	Tier 2	Supermarket Type1	2174.50
34	17.50	Low Fat	0.10	Soft Drinks	174.87	1997	Small	Tier 1	Supermarket Type1	2085.29
35	10.00	Low Fat	0.09	Health and Hygiene	146.71	1999	Medium	Tier 1	Supermarket Type1	3791.07
36	12.86	Regular	0.06	Fruits and Vegetables	128.07	1985	Medium	Tier 3	Supermarket Type3	2797.69
37	8.85	Regular	0.11	Soft Drinks	122.54	2009	Medium	Tier 3	Supermarket Type2	1609.90
38	12.86	Regular	0.12	Snack Foods	36.99	1985	Medium	Tier 3	Supermarket Type3	388.16
39	12.86	Low Fat	0.03	Snack Foods	87.62	1985	Medium	Tier 3	Supermarket Type3	2180.50
40	13.35	Low Fat	0.10	Dairy	230.64	1997	Small	Tier 1	Supermarket Type1	3435.53
41	9.80	Low Fat	0.03	Meat	126.00	1987	High	Tier 3	Supermarket Type1	2150.53
42	13.60	Low Fat	0.12	Snack Foods	192.91	1999	Medium	Tier 1	Supermarket Type1	2527.38
43	21.35	Low Fat	0.07	Canned	259.93	2009	Medium	Tier 3	Supermarket Type2	6768.52
44	12.15	Regular	0.04	Canned	126.50	1987	High	Tier 3	Supermarket Type1	373.51
45	6.42	LF	0.09	Dairy	178.10	1998	nan	Tier 3	Grocery Store	358.20
46	19.60	Low Fat	0.00	Health and Hygiene	153.30	2002	nan	Tier 2	Supermarket Type1	2428.84

Fig 3.16: The Dtale Window



Unique Row Values:

Medium (5203), Small (2388), High (932)

Fig 3.17: Frequency of values in the column name Outlet_Size

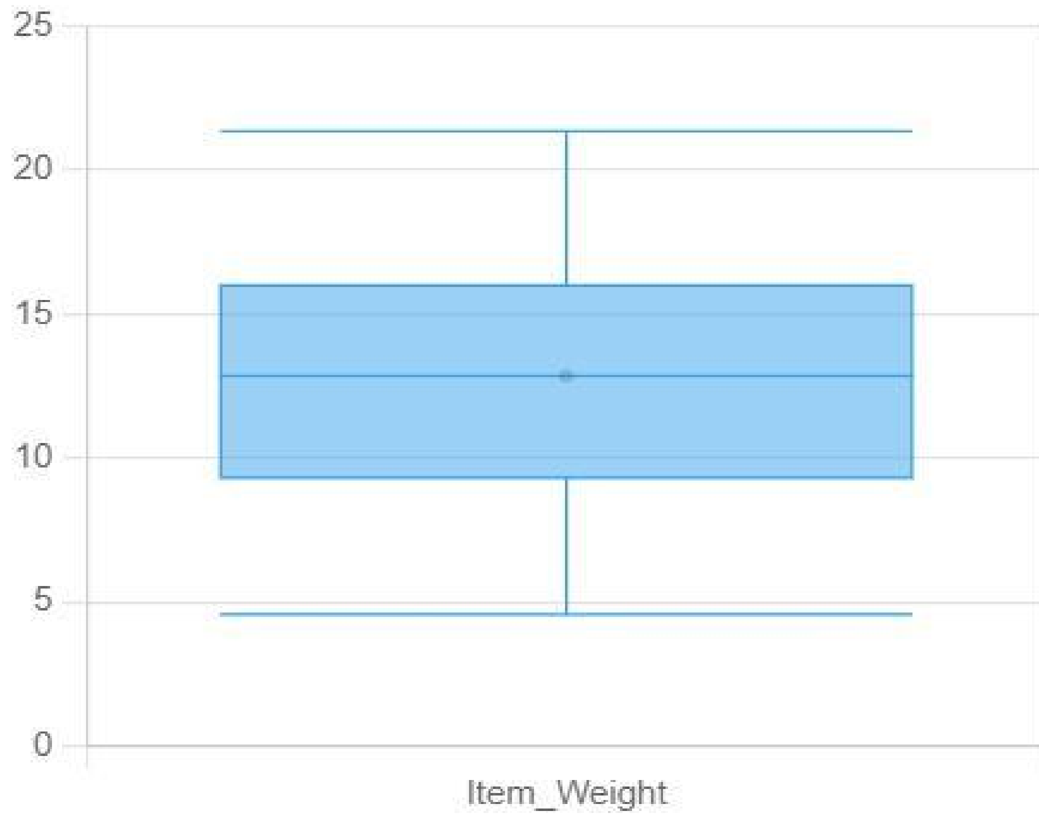


Fig 3.18 : This figure represents the Item_Weight value range

b) EDA USING KLIB LIBRARY

Klib is a python library which is used for importing, cleaning, analyzing and preprocessing the data.

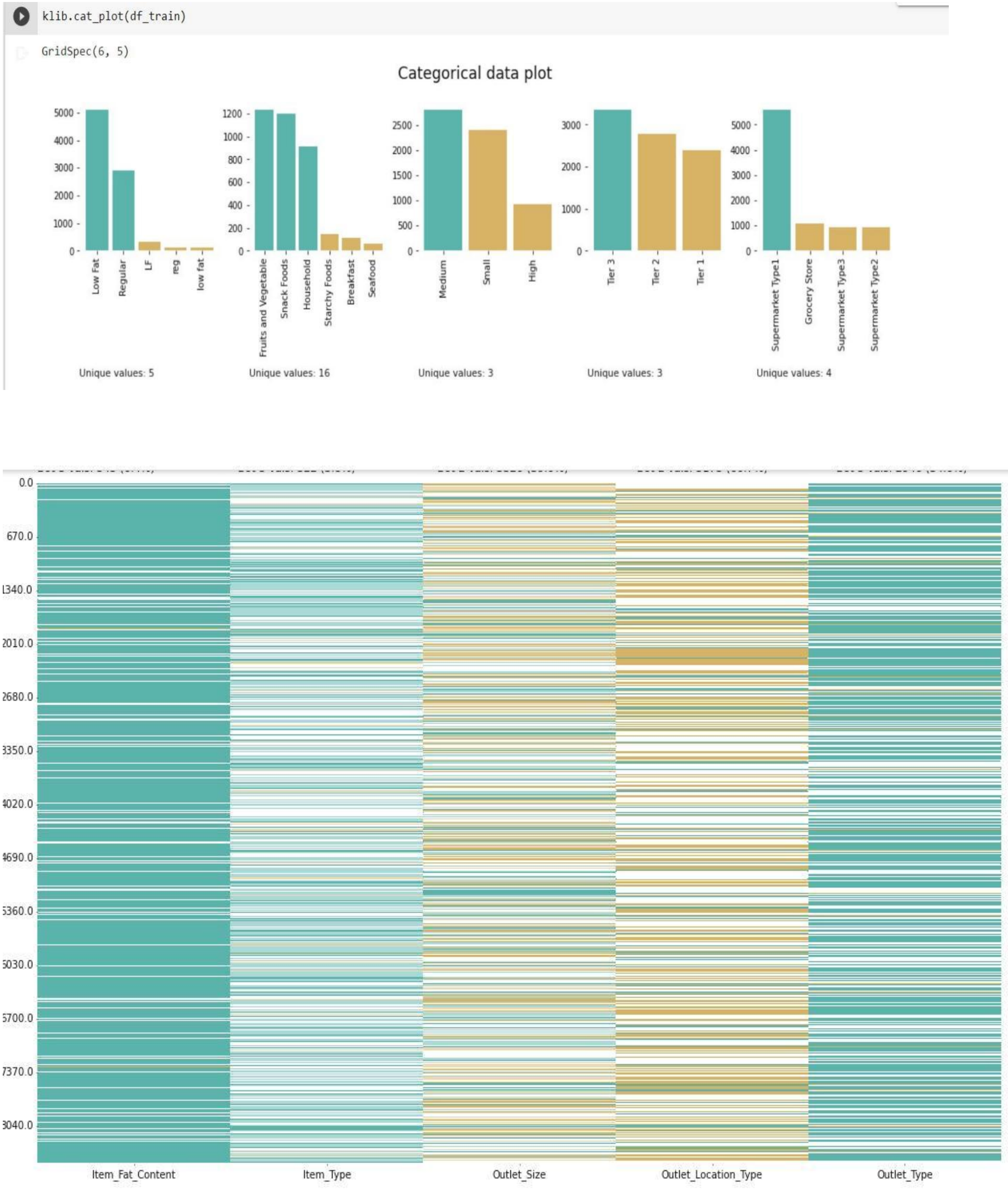


Fig 3.19 : Categorical data plot of all variables present in dataset using Klib Library

```
klib.corr_plot(df_train) # returns a color-encoded heatmap, ideal for correlations
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f31545a4350>

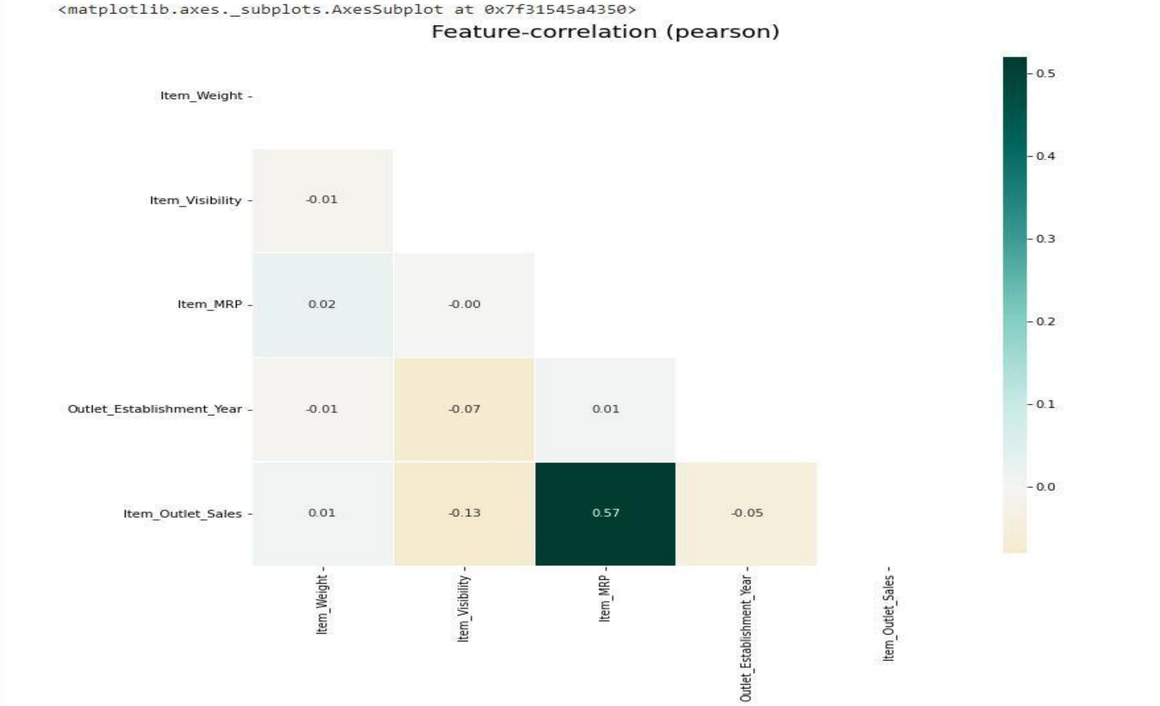


Fig 3.20 : Feature- correlation using klib Library

```
klib.corr_mat(df_train) # returns a color-encoded correlation matrix
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
Item_Weight	1.00	-0.01	0.02	-0.01	0.01
Item_Visibility	-0.01	1.00	-0.00	-0.07	-0.13
Item_MRP	0.02	-0.00	1.00	0.01	0.57
Outlet_Establishment_Year	-0.01	-0.07	0.01	1.00	-0.05
Item_Outlet_Sales	0.01	-0.13	0.57	-0.05	1.00

Fig 3.21 :Color- encoded correlation matrix.

```
klib.dist_plot(df_train) # returns a distribution plot for every numeric feature
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3163176290>
```

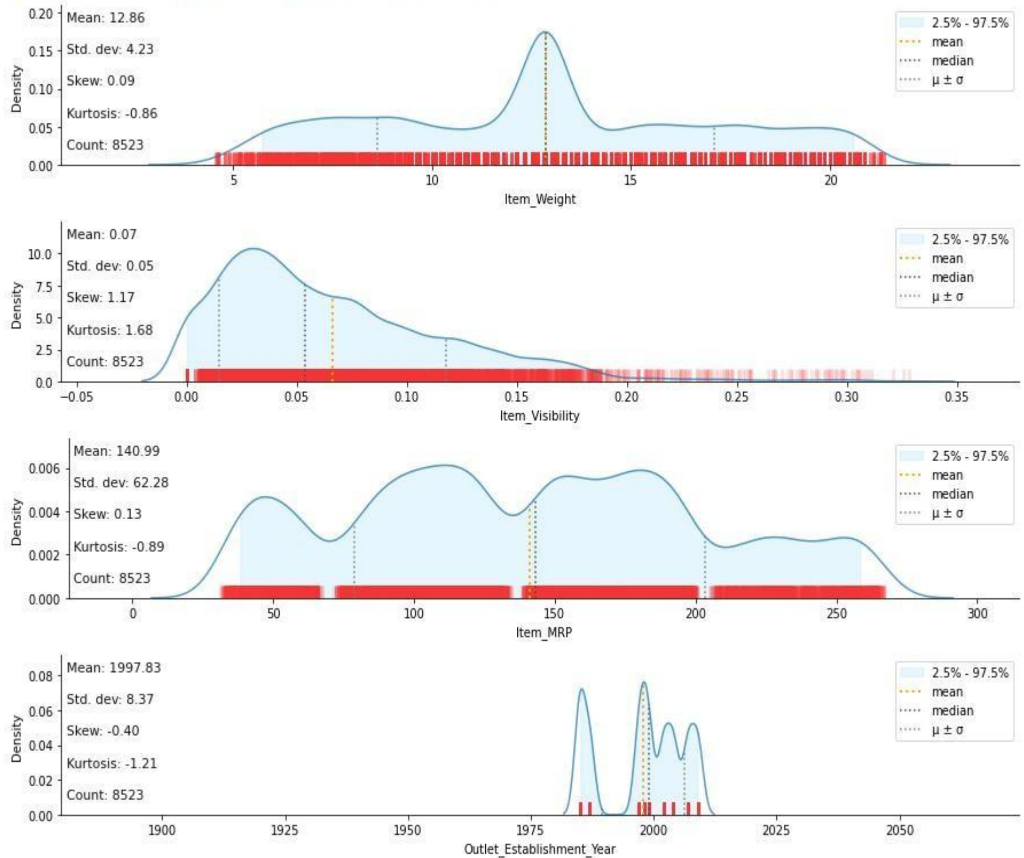


Fig 3.22: Distribution plot for every numeric feature.

c) **EDA WITH SEABORN LIBRARY-** Seaborn is a data visualization library built on top of matplotlib

EDA using seaborn library

```
[ ]  sns.heatmap(df_train.corr(),annot=True)
     plt.show()
```

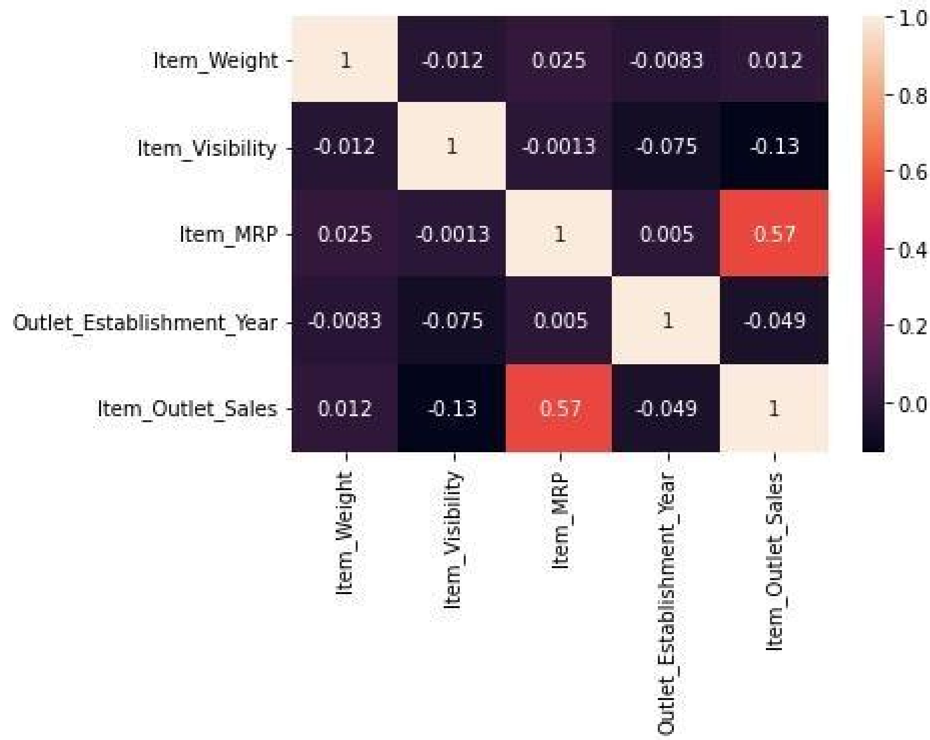


Fig 3.23 : Correlation between different features

From the figure 3.23 we can clearly see that item_visibility attribute has the lowest correlation with the other target variables and Item_MRP has strong positive correlation with target variables i.e. 0.57.

3.2.4 DATA CLEANING USING KLIB LIBRARY

Data cleaning is basically the process where the corrupt recordset, tables or databases are detected and then corrected by replacing, modifying, or deleting the dirty or coarse data.

```
[ ] # klib.clean - functions for cleaning datasets
    klib.data_cleaning(df_train) # performs datacleaning (drop duplicates & empty rows/cols, adjust dtypes,...)

    Shape of cleaned data: (8523, 10)Remaining NAs: 2410

    Changes:
    Dropped rows: 0
         of which 0 duplicates. (Rows: [])
    Dropped columns: 0
         of which 0 single valued.     Columns: []
    Dropped missing values: 0
    Reduced memory by at least: 0.46 MB (-70.77%)
```

Fig 3.24 :Cleaning the data using klib library

[105]

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type	item_outlet_sal
0	9.300000	Low Fat	0.016047	Dairy	249.809204	1999	Medium	Tier 1	Supermarket Type1	3735.1379
1	5.920000	Regular	0.019278	Soft Drinks	48.269199	2009	Medium	Tier 3	Supermarket Type2	443.4227
2	17.500000	Low Fat	0.016760	Meat	141.617996	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	19.200001	Regular	0.000000	Fruits and Vegetables	182.095001	1998	NaN	Tier 3	Grocery Store	732.3800
4	8.930000	Low Fat	0.000000	Household	53.861401	1987	High	Tier 3	Supermarket Type1	994.7052
...	...	...	...	...	...	...	...	...	...	
8518	6.865000	Low Fat	0.056783	Snack Foods	214.521805	1987	High	Tier 3	Supermarket Type1	2778.3833
8519	8.380000	Regular	0.046982	Baking Goods	108.156998	2002	NaN	Tier 2	Supermarket Type1	549.2849
8520	10.600000	Low Fat	0.035186	Health and Hygiene	85.122398	2004	Small	Tier 2	Supermarket Type1	1193.1136
8521	7.210000	Regular	0.145221	Snack Foods	103.133202	2009	Medium	Tier 3	Supermarket Type2	1845.5976
8522	14.800000	Low Fat	0.044878	Soft Drinks	75.467003	1997	Small	Tier 1	Supermarket Type1	765.6699

8523 rows × 10 columns

```
[107] df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   item_weight                8523 non-null   float64
 1   item_fat_content           8523 non-null   object
 2   item_visibility            8523 non-null   float64
 3   item_type                  8523 non-null   object
 4   item_mrp                   8523 non-null   float64
 5   outlet_establishment_year  8523 non-null   int64
 6   outlet_size                6113 non-null   object
 7   outlet_location_type       8523 non-null   object
 8   outlet_type                8523 non-null   object
 9   item_outlet_sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(5)
memory usage: 666.0+ KB
```

Fig 3.26 : Represents the 12 features of the dataset ie numerical and categorical

```
df_train=klib.convert_datatypes(df_train) # converts existing to more efficient dtypes, also called inside data_cleaning()
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   item_weight                8523 non-null   float32
 1   item_fat_content           8523 non-null   category
 2   item_visibility            8523 non-null   float32
 3   item_type                  8523 non-null   category
 4   item_mrp                   8523 non-null   float32
 5   outlet_establishment_year  8523 non-null   int16
 6   outlet_size                6113 non-null   category
 7   outlet_location_type       8523 non-null   category
 8   outlet_type                8523 non-null   category
 9   item_outlet_sales          8523 non-null   float32
dtypes: category(5), float32(4), int16(1)
memory usage: 192.9 KB
```

Fig 3.27 : Converting to more efficient data types using convert_datatypes function

3.2.5 FEATURE ENGINEERING

Feature Engineering is a way of using domain data to understand how to build mechanical operations learning algorithms. When feature engineering is done properly, the ability to predict ML algorithms are developed by creating useful raw data features that simplify the ML process. Feature engineering including correction of incorrect values. In the device database, object visibility has a small value of 0 which is unacceptable, because the object must be accessible to all, and so it is replaced by the mean of the column.

1) Label Encoding

```
[ ]  from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()
```

```
df_train['item_fat_content']= le.fit_transform(df_train['item_fat_content'])
df_train['item_type']= le.fit_transform(df_train['item_type'])
df_train['outlet_size']= le.fit_transform(df_train['outlet_size'])
df_train['outlet_location_type']= le.fit_transform(df_train['outlet_location_type'])
df_train['outlet_type']= le.fit_transform(df_train['outlet_type'])
```

```
df_train
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type	item_outlet_sal
0	9.300000	1	0.016047	4	249.809204	1999	1	0	1	3735.1379
1	5.920000	2	0.019278	14	48.269199	2009	1	2	2	443.4227
2	17.500000	1	0.016760	10	141.617996	1999	1	0	1	2097.2700
3	19.200001	2	0.000000	6	182.095001	1998	3	2	0	732.3800
4	8.930000	1	0.000000	9	53.861401	1987	0	2	1	994.7052
...	...	...	...	...	...	...	...	...	...	
8518	6.865000	1	0.056783	13	214.521805	1987	0	2	1	2778.3833
8519	8.380000	2	0.046982	0	108.156998	2002	3	1	1	549.2849
8520	10.600000	1	0.035186	8	85.122398	2004	2	1	1	1193.1136
8521	7.210000	2	0.145221	13	103.133202	2009	1	2	2	1845.5976
8522	14.800000	1	0.044878	14	75.467003	1997	2	0	1	765.6699

8523 rows × 10 columns

Fig 3.28 : Label Encoding Code

2) Splitting our data into train and test

```
[ ] X=df_train.drop('item_outlet_sales',axis=1)
```

```
[ ] Y=df_train['item_outlet_sales']
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state=101, test_size=0.2)
```

Fig 3.29 : Splitting of data into train and test data set.

3) Standarization

```
[ ] X.describe()
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.858088	1.369354	0.066132	7.226681	140.992767	1997.831867	1.736360	1.112871	1.201220
std	4.226130	0.644810	0.051598	4.209990	62.275051	8.371760	0.989181	0.812757	0.796459
min	4.555000	0.000000	0.000000	0.000000	31.290001	1985.000000	0.000000	0.000000	0.000000
25%	9.310000	1.000000	0.026989	4.000000	93.826500	1987.000000	1.000000	0.000000	1.000000
50%	12.857645	1.000000	0.053931	6.000000	143.012802	1999.000000	2.000000	1.000000	1.000000
75%	16.000000	2.000000	0.094585	10.000000	185.643700	2004.000000	3.000000	2.000000	1.000000
max	21.350000	4.000000	0.328391	15.000000	266.888397	2009.000000	3.000000	2.000000	3.000000

Fig 3.30 : Standardization of dataset

```
[ ] X_train_std

array([[ 1.52290023, -0.57382672,  0.68469731, ..., -1.7570342 ,
         1.08786619, -0.25964107],
       [-1.239856  , -0.57382672, -0.09514746, ...,  1.28755895,
        -0.13870429, -0.25964107],
       [ 1.54667619,  0.97378032, -0.0083859 , ...,  1.28755895,
        -0.13870429, -0.25964107],
       ...,
       [-0.08197109, -0.57382672, -0.91916229, ...,  0.27269457,
        -1.36527477, -0.25964107],
       [-0.74888436,  0.97378032,  1.21363045, ...,  1.28755895,
        -0.13870429, -0.25964107],
       [ 0.67885675, -0.57382672,  1.83915361, ..., -0.74216982,
         1.08786619,  0.98524841]])
```

```
X_test_std

array([[-0.44354743, -0.56892467, -0.19860257, ..., -0.75373281,
         1.1067281 ,  1.07886076],
       [ 1.18274465, -0.56892467, -0.51369355, ..., -1.74994556,
         1.1067281 , -0.22387125],
       [-1.20558148,  0.99561817,  0.18237795, ...,  0.24247994,
        -1.38596862, -0.22387125],
       ...,
       [ 0.62515889, -0.56892467,  0.90184011, ...,  1.23869268,
         1.1067281 , -1.52660325],
       [ 0.97365005, -0.56892467, -1.27256347, ..., -0.75373281,
         1.1067281 ,  1.07886076],
       [-1.54477944,  0.99561817, -1.08005617, ...,  1.23869268,
        -0.13962026, -0.22387125]])
```

Fig 3.31 X_train_std array and X_test_std array

```
[ ] Y_train

3684     163.786804
1935    1607.241211
5142    1510.034424
4978    1784.343994
2299    3558.035156
            ...
599     5502.836914
5695    1436.796387
8006    2167.844727
1361    2700.484863
1547     829.586792
Name: item_outlet_sales, Length: 6818, dtype: float32
```

```
[ ] Y_test

8179     904.822205
8355    2795.694092
3411    1947.464966
7089     872.863770
6954    2450.144043
            ...
1317    1721.093018
4996     914.809204
531      370.184814
3891    1358.232056
6629    2418.185547
Name: item_outlet_sales, Length: 1705, dtype: float32
```

Fig 3.32 Y_train array and Y_test array

In figures 3.33 and 3.34 we just split the train and test data into X_train_std , Y_train, X_test_std and Y_test.

3.2.6 MODEL BUILDING

Now the dataset is ready to fit a model after performing Data Preprocessing and Feature Transformation. The training set is fed into the algorithm in order to learn how to predict values. Testing data is given as input after Model Building a target variable to predict. The models are built using:

a) LR
b) RF Regression
c) Hyper Parameter Tuning
d) XGBoost Regression
e) Decision Tree
f) Ridge Regression

Model building

1. Linear regression

```
[ ]  from sklearn.linear_model import LinearRegression
     lr= LinearRegression()
```

```
[ ]  lr.fit(X_train_std,Y_train)

     LinearRegression()
```

```
lr.predict(X_test_std)
```

```
array([2110.22755889, 2147.54273582, 1241.33075705, ..., 1253.40857534,
       2425.63758608, 2378.49866902])
```

```
Y_pred_lr=lr.predict(X_test_std)
```

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
print(r2_score(Y_test,Y_pred_lr))
print(mean_absolute_error(Y_test,Y_pred_lr))
print(np.sqrt(mean_squared_error(Y_test,Y_pred_lr)))
```

```
0.5020054027842016
885.7810693115644
1164.996528679539
```

Fig 3.35: Value of R^2 in Linear Regression = 0.50

2) RANDOM FOREST REGRESSION

```
[ ] from sklearn.ensemble import RandomForestRegressor
    rf= RandomForestRegressor(n_estimators=1000)
```

```
[ ] rf.fit(X_train_std,Y_train)
```

```
RandomForestRegressor(n_estimators=1000)
```

```
[ ] Y_pred_rf= rf.predict(X_test_std)
```

```
[ ] print(r2_score(Y_test,Y_pred_rf))
    print(mean_absolute_error(Y_test,Y_pred_rf))
    print(np.sqrt(mean_squared_error(Y_test,Y_pred_rf)))
```

```
0.5509957886177873
777.7411339180328
1106.209854454175
```

Fig 3.36: Value of R^2 in Random Forest Regression = 0.55

Hyperparameter Tuning

```
[ ] from sklearn.model_selection import RepeatedStratifiedKFold
    from sklearn.model_selection import GridSearchCV

    # define models and parameters
    model = RandomForestRegressor()
    n_estimators = [10, 100, 1000]
    max_depth=range(1,31)
    min_samples_leaf=np.linspace(0.1, 1.0)
    max_features=["auto", "sqrt", "log2"]
    min_samples_split=np.linspace(0.1, 1.0, 10)

    # define grid search
    grid = dict(n_estimators=n_estimators)

    #cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=101)

    grid_search_forest = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1,
                                scoring='r2',error_score=0,verbose=2,cv=2)

    grid_search_forest.fit(X_train_std, Y_train)

    # summarize results
    print(f"Best: {grid_search_forest.best_score_:.3f} using {grid_search_forest.best_params_}")
    means = grid_search_forest.cv_results_['mean_test_score']
    stds = grid_search_forest.cv_results_['std_test_score']
    params = grid_search_forest.cv_results_['params']

    for mean, stdev, param in zip(means, stds, params):
        print(f"{mean:.3f} ({stdev:.3f}) with: {param}")
```

```
Fitting 2 folds for each of 3 candidates, totalling 6 fits
Best: 0.551 using {'n_estimators': 1000}
0.509 (0.007) with: {'n_estimators': 10}
0.546 (0.006) with: {'n_estimators': 100}
0.551 (0.006) with: {'n_estimators': 1000}
```

```
grid_search_forest.best_params_
```

```
{'n_estimators': 1000}
```

```
Y_pred_rf_grid=grid_search_forest.predict(X_test_std)
```

```
r2_score(Y_test,Y_pred_rf_grid)
```

```
0.5506742023512964
```

```
grid_search_forest.best_params_
```

```
{'n_estimators': 1000}
```

```
Y_pred_rf_grid=grid_search_forest.predict(X_test_std)
```

```
r2_score(Y_test,Y_pred_rf_grid)
```

```
0.5506742023512964
```

Fig 3.37: Value of R2 = 0.55

4) XGBOOST REGRESSION

```
[ ] from xgboost import XGBRegressor
    from sklearn import metrics
```

```
[ ] regressor = XGBRegressor()
```

```
[ ] regressor.fit(X_train, Y_train)

    [13:25:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
    XGBRegressor()
```

```
[ ] # prediction on training data
    training_data_prediction = regressor.predict(X_train)
    # prediction on test data
    test_data_prediction = regressor.predict(X_test)
```

```
[ ] # R squared Value
    r2_train = metrics.r2_score(Y_train, training_data_prediction)
    r2_test = metrics.r2_score(Y_test, test_data_prediction)
```

```
[ ] print('R Squared value = ', r2_train)
    print('R Squared value = ', r2_test)

    R Squared value =  0.635441553503312
    R Squared value =  0.5977658125516876
```

Fig 3.38: Value of R^2 in XGBoost Regression = 0.63

3) DECISION TREE

```
[ ] from sklearn.tree import DecisionTreeRegressor
    dr = DecisionTreeRegressor()
```

```
[ ] dr.fit(X_train_std,Y_train)

    DecisionTreeRegressor()
```

```
[ ] X_test.head()
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type
8179	11.0	1	0.055163	8	100.3358	2009	1	2	2
8355	18.0	1	0.038979	13	148.6418	1987	0	2	1
3411	7.72	2	0.074731	1	77.598602	1997	2	0	1
7089	20.700001	1	0.049035	6	39.9506	2007	3	1	1
6954	7.55	1	0.027225	3	152.934006	2002	3	1	1

```
Y_pred_dr= dr.predict(X_test_std)
```

+ Code + Text

```
[ ] from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
[ ] from sklearn import metrics
    value=r2_score(Y_test,Y_pred_dr)
    print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred_dr))
    print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred_dr)))
    print('MSE:', metrics.mean_squared_error(Y_test,Y_pred_dr))
    print('R2:',(value))

    MAE: 1051.4840754399902
    RMSE: 1500.682614470758
    MSE: 2252048.3093747897
    R2: 0.17367085570430163
```

Fig 3.39: Value of R^2 in Decision Tree = 0.17

5) RIDGE REGRESSION

```
from sklearn.linear_model import Ridge
```

```
[ ] rr = Ridge()
```

```
[ ] rr.fit(X_train_std,Y_train)
```

```
Ridge()
```

```
[ ] X_test.head()
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type
8179	11.0	1	0.055163	8	100.3358	2009	1	2	2
8355	18.0	1	0.038979	13	148.6418	1987	0	2	1
3411	7.72	2	0.074731	1	77.598602	1997	2	0	1
7089	20.700001	1	0.049035	6	39.9506	2007	3	1	1
6954	7.55	1	0.027225	3	152.934006	2002	3	1	1

```
[ ] Y_pred_rr= rr.predict(X_test_std)
```

```
[ ] from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
[ ] from sklearn import metrics
    value=r2_score(Y_test,Y_pred_rr)
    print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred_rr))
    print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred_rr)))
    print('MSE:', metrics.mean_squared_error(Y_test,Y_pred_rr))
    print('R2:',(value))
```

```
MAE: 893.098025301489
RMSE: 1177.0331952260133
MSE: 1385407.1426639582
R2: 0.4916617490250573
```

Fig 3.40: Value of R^2 in Ridge Regression = 0.4916

CHAPTER 4

4 . PERFORMANCE ANALYSIS

For the purpose of performance analysis we can go and look for the R^2 value of the different algorithm performed and check for which algorithm gives us the best performance

LR

```
value=r2_score(Y_test,Y_pred_lr)
print(mean_absolute_error(Y_test,Y_pred_lr))
print(np.sqrt(mean_squared_error(Y_test,Y_pred_lr)))
print('R2:',(value)*100)
```

```
893.1098219165341
1177.0425587900395
R2: 49.165366048734604
```

Fig 4.1 Performance of Linear Regression

RF regression

```
[ ] value = r2_score(Y_test,Y_pred_rf)
    print(mean_absolute_error(Y_test,Y_pred_rf))
    print(np.sqrt(mean_squared_error(Y_test,Y_pred_rf)))
    print('R2:',(value)*100)
```

```
776.9585233226115
1106.2543507541138
R2: 55.095966630735546
```

Fig 4.2 :Performance of Random Forest Regression

Hyper parameter tuning

```
Y_pred_rf_grid=grid_search_forest.predict(X_test_std)
```

```
value= r2_score(Y_test,Y_pred_rf_grid)
print('R2:', (value)*100)
```

```
R2: 54.92565801082437
```

Fig 4.3: Performance of Hyper Tuning Parameter

Decision Tree

```
from sklearn import metrics
value=r2_score(Y_test,Y_pred_dr)
print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred_dr))
print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred_dr)))
print('MSE:', metrics.mean_squared_error(Y_test,Y_pred_dr))
print('R2:',(value)*100)
```

```
MAE: 1057.1568508002765
RMSE: 1508.469906849536
MSE: 2275481.459870648
R2: 16.507268425656818
```

Fig 4.4: Performance of Decision Tree

XGBoost Regression

```
from sklearn import metrics
value=r2_score(Y_test,Y_pred_regressor)
print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred_regressor))
print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred_regressor)))
print('MSE:', metrics.mean_squared_error(Y_test,Y_pred_regressor))
print('R2:',(value)*100)
```

```
MAE: 742.4536
RMSE: 1047.2457
MSE: 1096723.6
R2: 59.75864575943323
```

Fig 4.5: Performance of XgBoost Regression

Ridge Regression

```
[ ]  from sklearn import metrics
     value=r2_score(Y_test,Y_pred_rr)
     print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred_rr))
     print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred_rr)))
     print('MSE:', metrics.mean_squared_error(Y_test,Y_pred_rr))
     print('R2:',(value))

     MAE: 893.098025301489
     RMSE: 1177.0331952260133
     MSE: 1385407.1426639582
     R2: 0.4916617490250573
```

Fig 4.6: Performance of Ridge Regression

TABLE 4.1 : Algorithms Performance

ALGORITHM	R2	RMSE	MSE
Linear Regression	49.165	1177.04	1385429.18
Random Forest Regression	55.09	1105	12222736.57
Decision Tree	16.50	1508.46	2275481.45
XGBoost Regression	59.75	1047	1096723.67
Ridge Regression	49.166	117.03	1385407.14

To forecast BigMart's revenue, simple to advanced ML algorithms have been implemented, such as LR, Decision Tree, RF regression and XGBoost.

From the above table, we conclude that the XGBoost algorithm is more efficient and gives accurate and fast results.

PERFORMANCE ANALYSIS USING GRAPHS

RMSE AND MSE VALUES

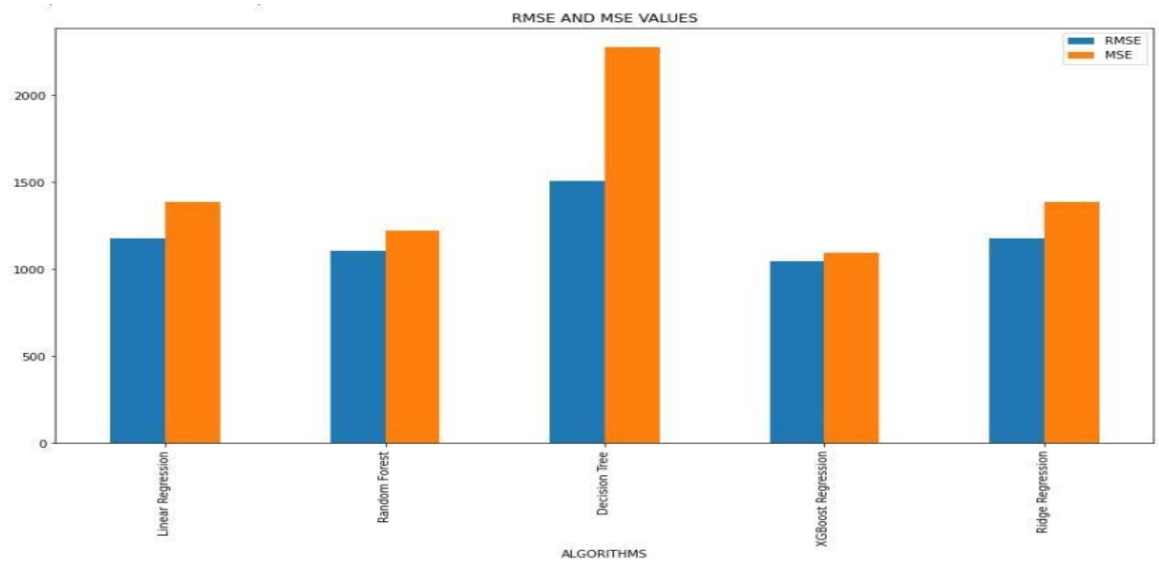


Fig 4.7:Comparison of RMSE and MSE values for ML Algorithms used

Figure 4.7 shows the comparative analysis of RMSE and MSE values. RMSE is the squared root of MSE and MSE is calculated by the squared difference between the original and predicted values in the data set. In this experiment Decision tree has the highest RMSE and MSE value and XgBoost Regression has the lowest RMSE and MSE value.

R^2 AND MAE VALUES

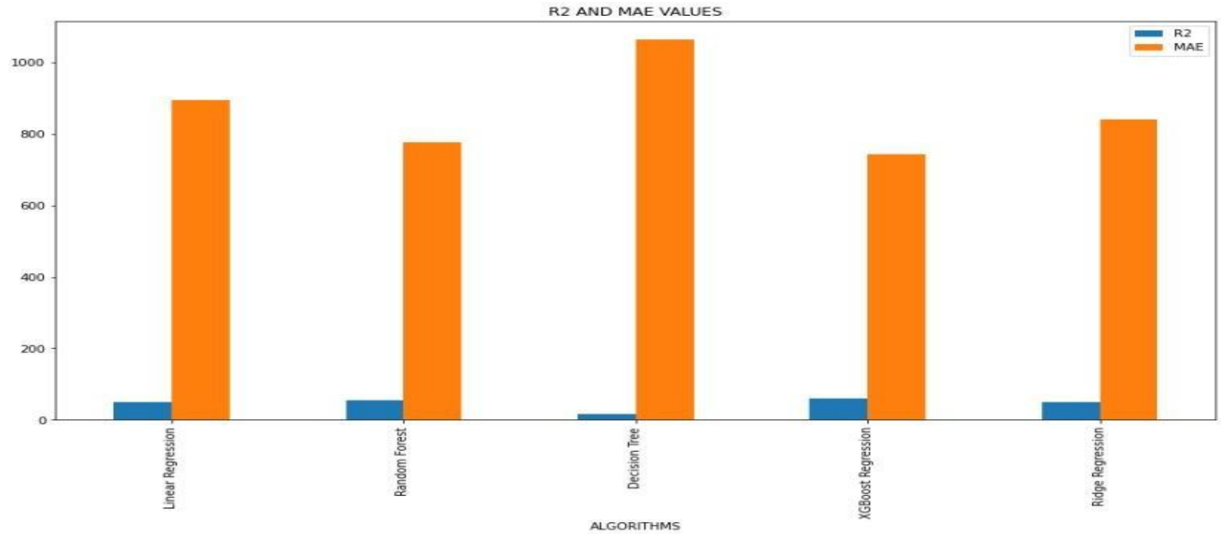


Fig 4.8:Comparison of R^2 and MAE values for ML Algorithms used

Figure 4.8 shows the comparative analysis of R^2 and MAE values. MAE is calculated by the average of the absolute difference between the actual and predicted values in the dataset and R^2 is calculated by the sum of the residuals squared, and the total sum of squares is the sum of all the data's deviations from the mean. In this experiment Decision tree has the highest MAE value whereas XgBoost has the lowest and in case of R^2 XgBoost has the highest value whereas Decision tree has the lowest value.

It has been observed that increased efficiency is observed with XGBoost algorithms with lower RMSE, MSE and MAE rating and higher R^2 rating

CHAPTER 5

5 . CONCLUSIONS

5.1 CONCLUSION

So from this project we conclude that a smart sales forecasting program is required to manage vast volumes of knowledge for business organizations.

The Algorithms which are presented in this report , LR, RF regression, Decision tree and XGBoost regression provide an effective method for data sharing as well as decision-making and also provide new approaches that are used for better identifying consumer needs and formulate marketing plans that are going to be implemented.

The outcomes of ML algorithms which are done in this project will help us to pick the foremost suitable demand prediction algorithm and with the aid of which BigMart will prepare its marketing campaigns.

5.2 FUTURE SCOPE

The future scope of this project is that this project can further collaborate with any other devices which are supported with an in-built intelligence by virtue of the Internet of Things (I0T) which makes it more feasible to use.

Multiple instances parameters and various factors are also make this sales prediction project more

innovative and successful.

The most important term for any prediction-based system that is accuracy, is often significantly increased

because of the increase in the number of parameters.

6. REFERENCES

1. Beheshti-Kashi, S., Karimi, H.R., Thoben, K.D., Lutjen, M., Teucke, M.: A survey on retail sales forecasting and prediction in fashion markets. Systems Science &Control Engineering 3(1), (2015), pp.154–161

2. Bose, I., Mahapatra, R.K.: Business data mining ML perspective. Information & management 39(3),(2001), pp. 211–225

3.Mitchell, T. M. ML and data mining. Communications of the ACM, 42(11) , (1999), pp. 30-36.

4. Das, P., Chaudhury, S.: Prediction of retail sales of footwear using feedforward and recurrent neural networks. Neural Computing and Applications 16(4-5),(2007), pp. 491–502

5. Punam, K., Pamula, R., Jain, P.K.: A two-level statistical model for big mart sales prediction. In: 2018 International Conference on Computing, Power and Communication Technologies (GUCON), IEEE (2018). pp. 617–620.