TRAINING REPORT

OF

**THREE MONTHS INDUSTRIAL TRAINING,
UNDERTAKEN**

AT

AICHUNKS PRIVATE LIMITED

IN

MACHINE LEARNING

ON

BIG MART SALES PREDICTION

**SUBMITTED IN PARTIAL FULFILLMENT OF
THE DEGREE**

OF

BE(CSE)

Under the Guidance of:
Name: Shubham Gupta
Designation: Team Lead
Department: Applied Data Science

Submitted By:
Name: KRITI SHAHI
University ID No.: 1610991466

Chitkara University, Punjab, India

CERTIFICATE

Ref. No. TRC/2020/119 **Date: 7th May' 2020**

CERTIFICATE OF TRAINING

This is to certify that **Ms. KRITI SHAHI**, a student of B. E. (*CSE*) of Chitkara University, Punjab is under the internship/industrial training from **20th February,2020** to Till Date. During this training, she is working under the guidance of Mr ROHIT SHARMA and Mr SHUBHAM GUPTA. Her overall performance during the training period is **Excellent**.

Other details related to his/her training are as below:

i. Name of the Organization: AICHUNKS Pvt. Ltd.
ii. Place of work: Chandigarh
iii. Address of the Employer: SCO 112-113, Level – I, Sector 34-A, Chandigarh- 160022
iv. Stipend (in Rs.): N/A

AICHUNKS PVT. LTD.
Rohit Sharma
DIRECTOR

(Signature)

Name ROHIT SHARMA

Designation DIRECTOR

ACKNOWLEDGEMENT

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

I express my sincere gratitude to all for providing me an opportunity to undergo Industrial Training as the part of the curriculum.

I am thankful to Mr Rohit Sharma and Mr Shubham Gupta for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.

Lastly, I would like to thank the almighty and our parents for their moral support and friends with whom we shared our day-to day experience and received lots of suggestions that improve our quality of work.

Kriti Shahi

ID No. 1610991466

TABLE OF CONTENTS

PREFACE

This Project Report has been prepared for the Industrial Training of the programme CSE in 8th Semester. I had pursued my training/internship at AIChunks Private Limited for a period of three months and made a project on BigMart Sales Prediction.

There is a lot of competition between companies and all the companies are trying their level best to improve their products with the help of forecasting by analyzing the data collected and finding insights from data. I have performed sales forecasting on the dataset of BigMart company by analysis and regression techniques. This report contains motivation behind choosing the project, its industry applications, detailed analysis of every module, coding performed and future enhancements.

This project helped me to apply my theoretical concepts learned during my industrial training.

I gained a lot of knowledge from the industrial training as well as developing a predictive model for sales forecasting. This industrial training experience will be valuable for my career.

1. PROJECT UNDERTAKEN

1.1 OBJECTIVE

With the rapid development of global malls and stores chains and the increase in the number of electronic payment customers, the competition among the rival organizations is becoming more serious day by day. Each organization is trying to attract more customers using personalized and short-time offers which makes the prediction of future volume of sales of every item an important asset in the planning and inventory management of every organization, transport service, etc. Due to the cheap availability of computing and storage, it has become possible to use sophisticated machine learning algorithms for this purpose. The objective of this project is to forecast for the sales data of big mart in a number of big mart stores across various location types which is based on the historical data of sales volume. According to the characteristics of the data, regression techniques are to be applied.

1.2 NEED TO CHOOSE THE PROJECT

Machine Learning has entered into every business and companies like Google, Amazon, Microsoft etc are developing their products as well as improving the earlier products by making use of Machine Learning. A lot of data is collected and analysed taking care of the customer behaviour which helps in enhancements of the product.

I had pursued training/ internship in Machine Learning as I wanted to apply theoretical concepts in the project. I am interested in domains like computer vision, finance, retail etc. I have already made my last project in computer vision and now in this semester, I wanted to apply the knowledge gained into another domain.

1.3 INDUSTRY APPLICATION

The industry application of this project is sales forecasting**.** Sales forecasting is a process of predicting future sales which will be beneficial for the company. Sale forecasts enable companies to make business decisions which will help them in improving the short-term and long-term performance. Past sales data and economic trends act as base for these forecasts.

Established companies are able to predict future sales based on years of past business data and this task is relatively easy as compared to the startups who don't have enough data to make a base for their forecasts. The small companies who haven't yet established themselves in the market use competitive analysis. The competitive analysis enables to analyse the situation of the competitors in the global market. The analysis data performance is completely legal and public which helps to develop strategies, save time and resources, work towards boosting their sales results and become a tough competitor in the market. Competitive analysis is being performed by every company in the world and all the companies are looking at their competitors to get insights about the customers.

The companies who are forecasting for longer term, they will get less insights and benefits. After all, the world is going to change, forecasts will need to get updated to reflect those changes. Forecast is the best guess at what's going to happen in future.

2. INTRODUCTION TO ASSIGNED JOB

The aim is to build a predictive model for sales of Big Mart Outlets. The Big Mart is an international brand & it started its journey in 2007 with free home delivery services of food and grocery. Big Mart allows you to walk away from the drudgery of grocery shopping and welcome an easy relaxed way of browsing and shopping for groceries. Discover new products and shop for all your food and grocery needs from the comfort of your home or office. No more getting stuck in traffic jams, paying for parking, standing in long queues and carrying heavy bags – get everything you need, when you need, right at your doorstep.

The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of products and outlets have been defined. The job assigned to me is to build a predictive model on the basis of the data collected by the data scientist. The train dataset contains 8523 rows of data which contains input as well as output variable. The test dataset contains 5681 rows of data having same columns as that of train dataset except Item_Outlet_Sales. The model learns from the input and output data provided by the training dataset and then values of the column Item_Outlet_Sales are to predicted by the model.

To build the model, following steps are followed -:

1. Descriptive Statistics
2. Data Exploration
3. Data Cleaning
4. Feature Engineering
5. Applying different regression algorithms

3. FEASIBILITY STUDY

The project performs a supervised learning task to boost the sales of the company by analyzing different attributes that affect the sales of the company and build a model to predict the outcome of the sales.

The future is uncertain, and the sales cannot be predicted with certainty. Sales of a company depend on various factors. The economic condition is a factor that affects the sales. The recession leads to decline in sales revenue and profits, the companies plans some strategies like cutting costs, hiring of new employers etc to make itself stable in the future. The demographics of consumers such as age, sex, education, occupation, income, etc. make a considerable impact on sales and should also be studied while building a model for sales prediction. The groups made on social website influence the behaviour of the consumers and changes their thoughts on buying a product. The changes in the advertising campaigns, promotional schemes and pricing policy can bring a significant change in the sales figure.

The proposed system does not take into account the above factors. The data of all these factors are to be collected and making some changes while analyzing and building the model will give accurate results on the basis of the factors.

4. MODULAR DESCRIPTION OF JOB

4.1 DESCRIPTIVE STATISTICS

The dataset is explored and got information about various columns in the form of summary. By performing descriptive statistics, I got information about the datatypes of columns, columns having null values and then categorized them into continuous and categorical variables, minimum, maximum, average, standard deviation as well as quantile values with respect to each numeric column. The data was visualized in the form of histograms and skewness of target variable i.e Item_Outlet_Sales was checked and applied square root transformation to target variable.

4.2 DATA EXPLORATION

Univariate analysis and Bivariate analysis are performed in the data exploration step. Univariate analysis is performed on each continuous and categorical variable. The metrics used for univariate analysis in case of continuous variables is central tendency like mean, standard deviation etc and histograms are used for visualisation purposes. In case of categorical variables, frequency distribution table which contains Count and Count% is used and bar chart are used for visualisation purpose. In bivariate analysis, relationship between numerical features and target variables is analysed in the form of scatterplots. Relationship between categorical variables and target variable is analysed in the form of boxplots. After performing analysis, observations are written down in jupyter notebook.

4.3 DATA CLEANING

There are three columns i.e Item_Weight, Outlet_Size and Item_Outlet_Sales with missing values. The missing values of Item_Weight and Outlet_Size are imputed with their mean values. The missing values of Item_Outlet_Sales belong to the test dataset so no need of dealing with them as they will be updated when applying models in the end.

4.4 FEATURE ENGINEERING

The two techniques that are performed in feature engineering are feature creation and feature transformation. There were a lot of zeroes in Item_Visibility column which is impossible so those missing values with mean are imputed. Outlet Years column is created from Outlet Establishment Year column . All items have been categorized into Food, Non Consumable and Drinks. There was need of creating non edible category in Item Fat Content so that Non Consumable category of items be put into it. One hot encoding of categorical data is performed.

4.5 MODEL BUILDING

Various algorithms like Linear Regression, Ridge Regression, Lasso Regression, Random Forest, Decision Tree and XGBoost are applied to build a predictive model. Out of all these algorithms XGBoost performs the best. A features importance is also plotted in the end.

5. DETAILED ANALYSIS OF INDIVIDUAL MODULE

5.1 DESCRIPTIVE STATISTICS

```
data.info() # Prints the summary of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14204 entries, 0 to 5680
Data columns (total 13 columns):
Item_Identifier              14204 non-null object
Item_Weight                  11765 non-null float64
Item_Fat_Content             14204 non-null object
Item_Visibility              14204 non-null float64
Item_Type                    14204 non-null object
Item_MRP                     14204 non-null float64
Outlet_Identifier            14204 non-null object
Outlet_Establishment_Year    14204 non-null int64
Outlet_Size                  10188 non-null object
Outlet_Location_Type         14204 non-null object
Outlet_Type                  14204 non-null object
Item_Outlet_Sales            8523 non-null float64
source                       14204 non-null object
dtypes: float64(4), int64(1), object(8)
memory usage: 1.5+ MB
```

The above summary includes the list of the columns with their data types and non null values of each column. It is an essential step to recognize the different types of data (numerical and categorical). Here in this dataset, eight of the variables are categorical (labelled as 'object') including the source column which has been created while the remaining five are numerical (labelled as 'int64' and 'float64').

```
data.nunique() # Unique values of each column
```

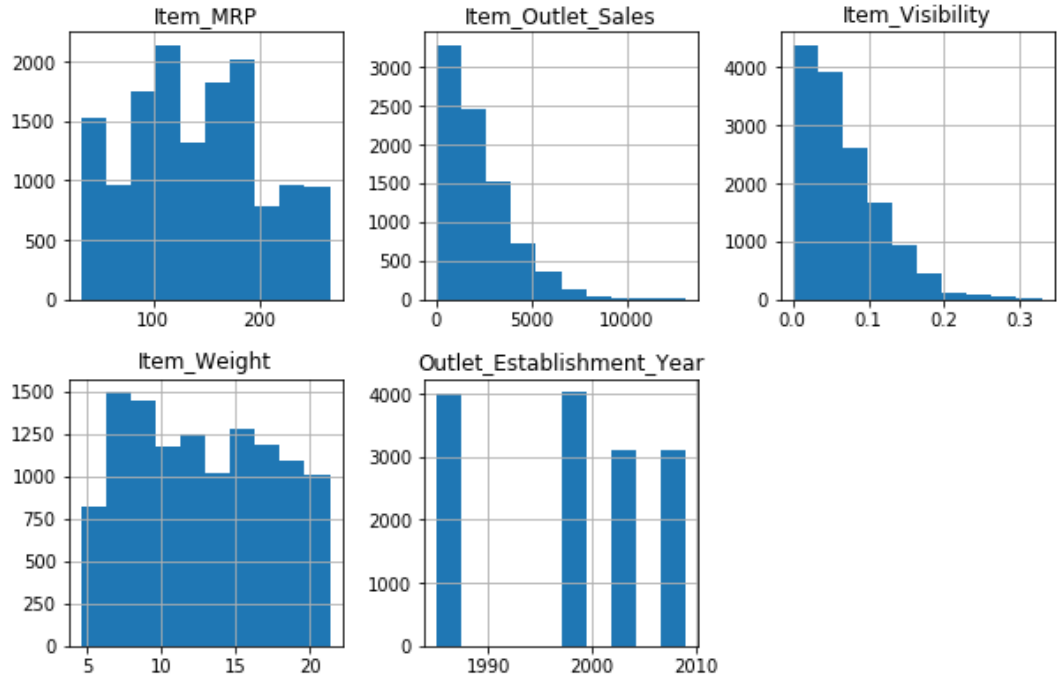
```
Item_Identifier              1559
Item_Weight                   415
Item_Fat_Content                5
Item_Visibility             13006
Item_Type                      16
Item_MRP                     8052
Outlet_Identifier              10
Outlet_Establishment_Year       9
Outlet_Size                     3
Outlet_Location_Type            3
Outlet_Type                     4
Item_Outlet_Sales            3493
source                          2
dtype: int64
```

```
data.describe() # Prints the summary statistics of numerical variables
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	11765.000000	14204.000000	14204.000000	14204.000000	8523.000000
mean	12.792854	0.065953	141.004977	1997.830681	2181.288914
std	4.652502	0.051459	62.086938	8.371664	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.710000	0.027036	94.012000	1987.000000	834.247400
50%	12.600000	0.054021	142.247000	1999.000000	1794.331000
75%	16.750000	0.094037	185.855600	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

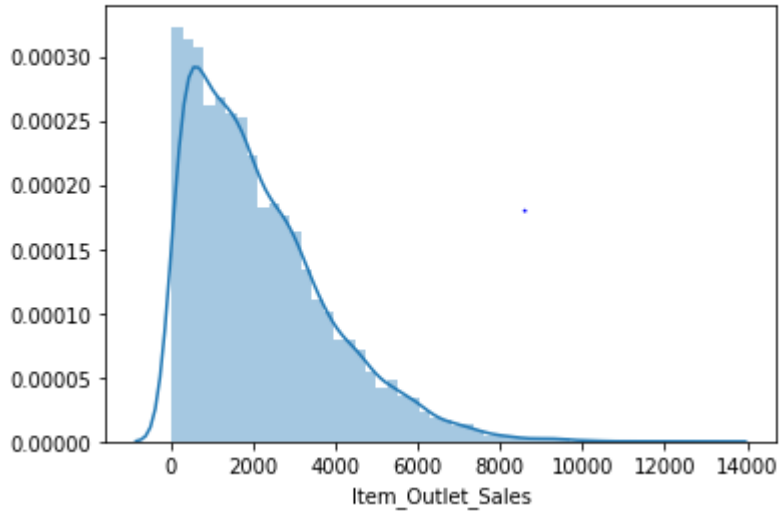
It shows us minimum, maximum, average, standard deviation as well as quantile values with respect to each numeric column.

```
# Visualizing the data
data.hist(figsize=(10,10),layout=(3,3))
plt.show()
```

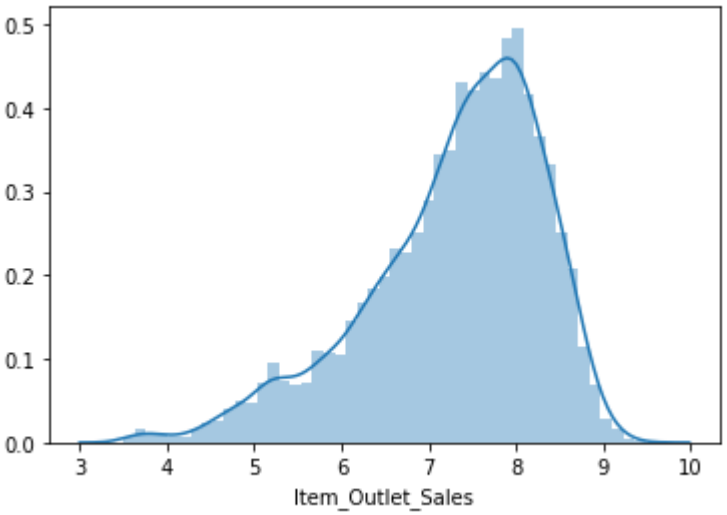


This confirms that Item_MRP, Item_Outlet_Sales, Item_Visibility, Item_Weight are numerical variables. By looking at the histogram of Outlet_Establishment_Year, one might think it is of categorical nature but it is not as it varies continuously.

```
sns.distplot(train_data['Item_Outlet_Sales']);
```



We can see that target variable has right skewed distribution with a long tail towards high values. We'll need to log transform this variable so that it becomes normally distributed. A normally distributed target variable helps in better modeling the relationship between target and independent variables.

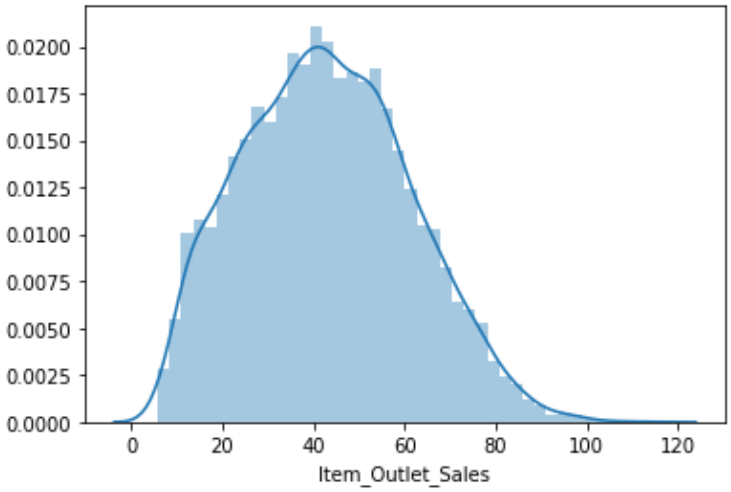


The log method did not convert the skewed distribution into normalized distribution so now applying square root transformation.

```
# Applying square root transformation to the target variable
target = np.sqrt(train_data['Item_Outlet_Sales'])
print(f"Skewness is {target.skew()}")

# Visualizing the target variable after applying square root transformation
sns.distplot(target);
```

```
Skewness is 0.23467599347099255
```



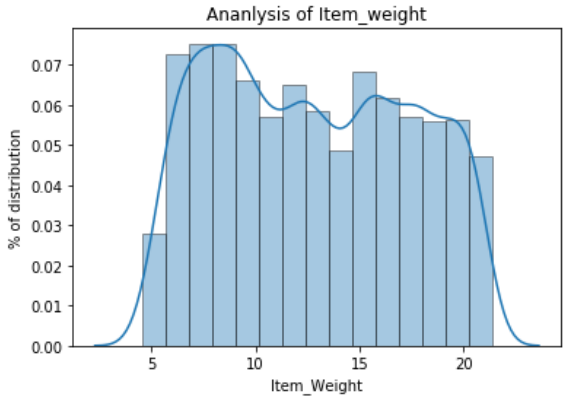
5.2 DATA EXPLORATION

Data Exploration is a technique of analysing datasets by summarizing the characterstics and visulizations to understand the dataset in a better form. Steps that are involved in building a predictive model are Univariate and Bivariate Analysis, Missing values Treatment, Variable Transformation and Variable creation.

5.2.1 Univariate Analysis

Univariate Analysis explores the variables one by one. The methods to perform univariate analysis are different for continuous and categorical variables.The metrics used for univariate analysis in case of continuous variables is central tendency like mean, standard deviation etc and histograms are used for visualisation purposes. In case of categorical variables, frequency distribution table which contains Count and Count% is used and bar chart are used for visualisation purpose.

Analysis of Item_Weight




```
# Metrics for analysis of Item_Weight
data['Item_Weight'].describe()
```

```
count    11765.000000
mean        12.792854
std          4.652502
min          4.555000
25%          8.710000
50%         12.600000
75%         16.750000
max         21.350000
Name: Item_Weight, dtype: float64
```

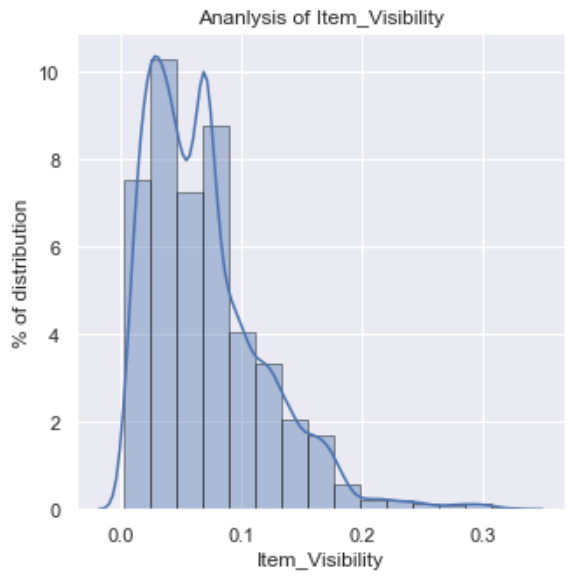
Observations

1. 65% of products have weight less than 10

2. 60% of products have weight less than 20

3. There are no outliers

Analysis of Item_Visibility



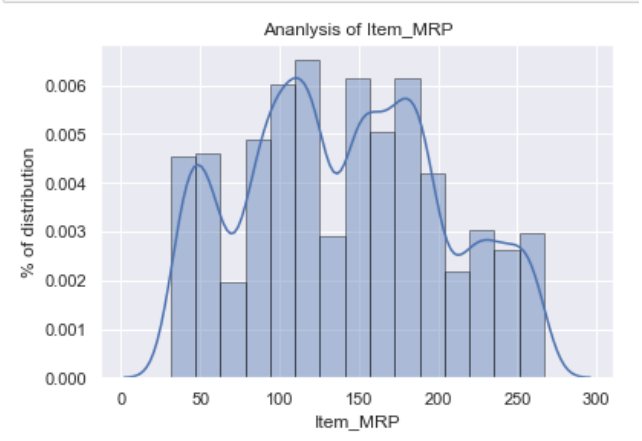

```
count    14204.000000
mean         0.065953
std          0.051459
min          0.000000
25%          0.027036
50%          0.054021
75%          0.094037
max          0.328391
Name: Item_Visibility, dtype: float64
```

Observations

1. 10% of the products have Item_Visibility 0.025

2. Less products are present of higher visibility. This is may be due to the fact of higher sales of the particular product.

Analysis of Item_MRP



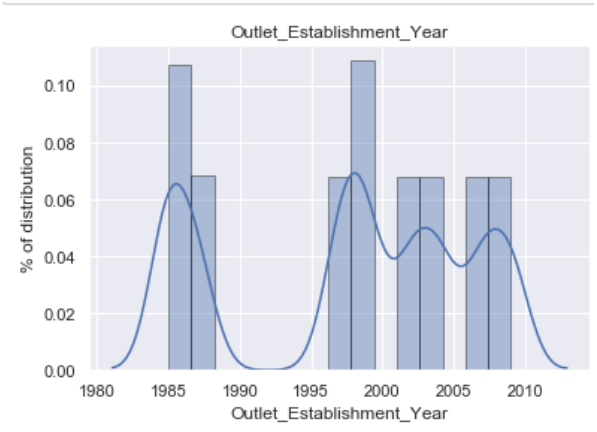

```
count    14204.000000
mean       141.004977
std         62.086938
min         31.290000
25%         94.012000
50%        142.247000
75%        185.855600
max        266.888400
Name: Item_MRP, dtype: float64
```

Observations

1. Most of MRP lies in range 100 and 180.

2. Food items having MRP greater than 200 are less.

Analysis of Outlet_Establishment_Year




```
count    14204.000000
mean      1997.830681
std          8.371664
min       1985.000000
25%       1987.000000
50%       1999.000000
75%       2004.000000
max       2009.000000
Name: Outlet_Establishment_Year, dtype: float64
```

Observations

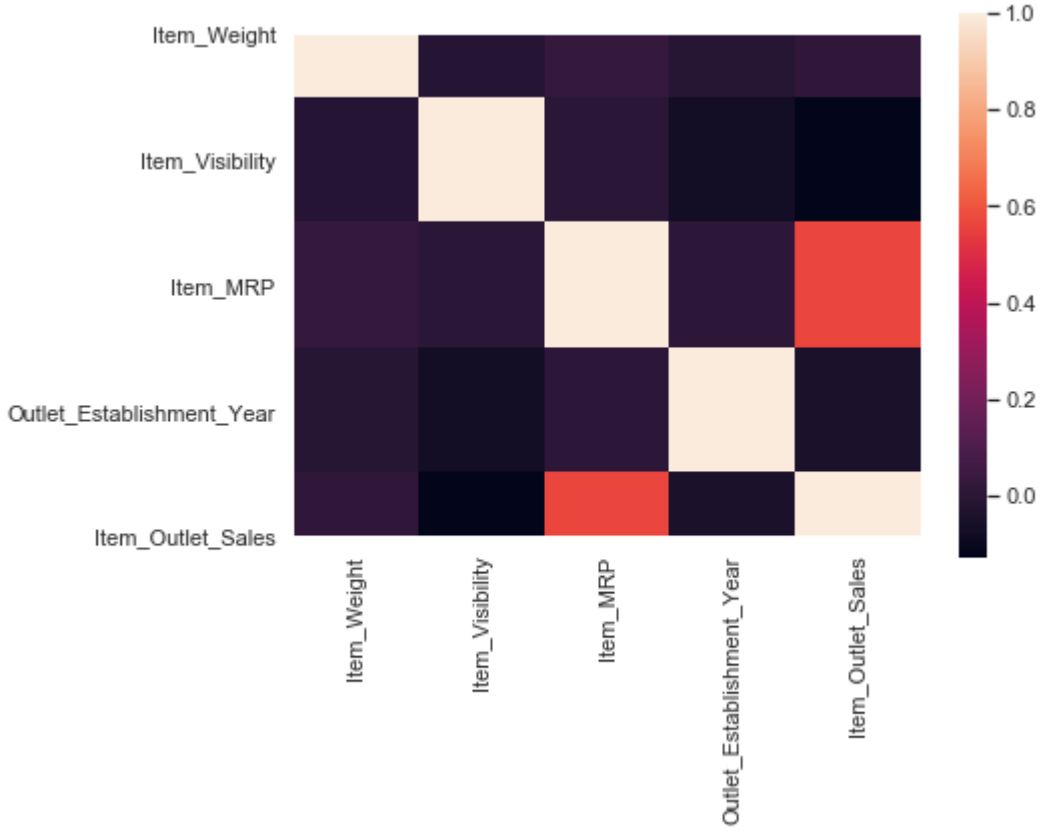
1. There are no sales from year 1990 to 1995.

2. 65% of products are sold in year 1985.

3. 70% of products are sold in year 1997.

Correlation Matrix

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.



The table below shows the correlation score with target variable

```
Item_Outlet_Sales            1.000000
Item_MRP                     0.567574
Item_Weight                  0.014123
Outlet_Establishment_Year   -0.049135
Item_Visibility             -0.128625
Name: Item_Outlet_Sales, dtype: float64
```

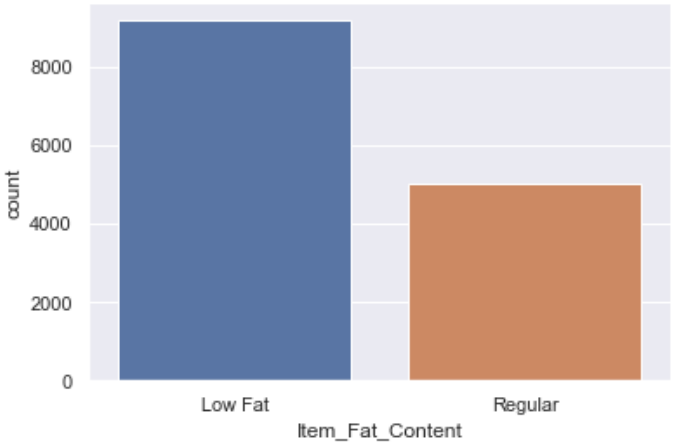
Observations

1. Item_Visibility has the least correlation with Item_Outlet_Sales according to the Correlation Matrix which means that if product is less visible then sales will be higher. But this may not be the case.

2. Item_MRP has highest correlation with Item_Outlet_Sales which means if the price of the product increases then sale of that product also increases.

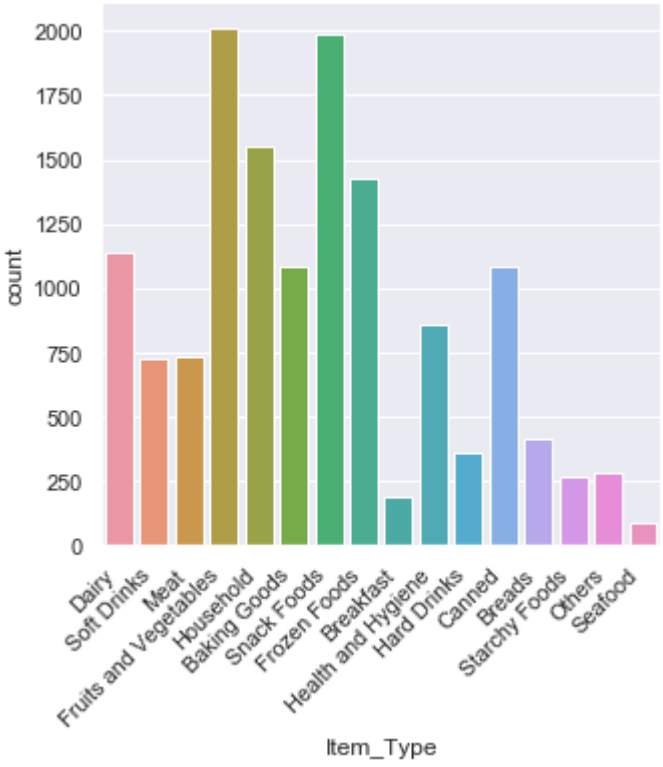
Analysis of Item_Fat_Content

The category Low Fat has been miscoded as LF and low fat category. Also, category Regular has been miscoded as reg. The category LF and low fat are replaced with that of Low Fat category and category reg with Regular.



	Count	Count%
Low Fat	9185	64.664883
Regular	5019	35.335117

Analysis of Item_Type



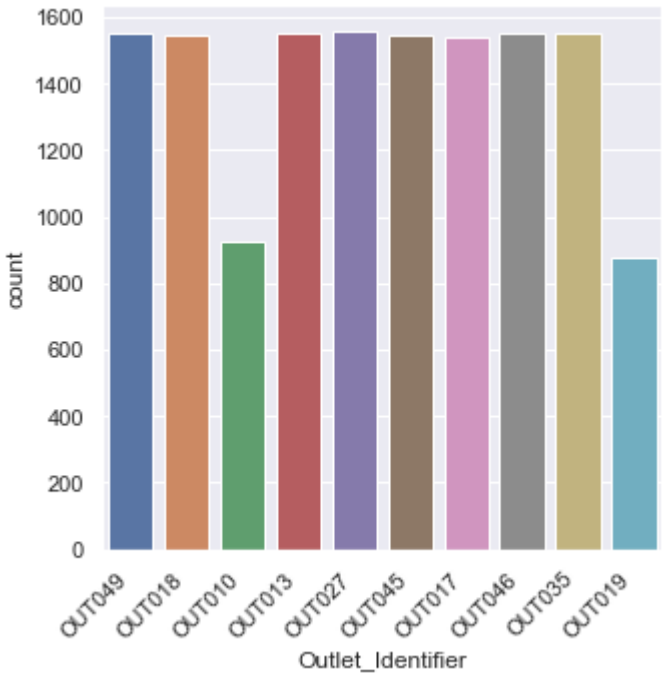
	Count	Count%
Fruits and Vegetables	2013	14.172064
Snack Foods	1989	14.003098
Household	1548	10.898338
Frozen Foods	1426	10.039426
Dairy	1136	7.997747
Baking Goods	1086	7.645734
Canned	1084	7.631653
Health and Hygiene	858	6.040552
Meat	736	5.181639
Soft Drinks	726	5.111236
Breads	416	2.928752
Hard Drinks	362	2.548578
Others	280	1.971276
Starchy Foods	269	1.893833
Breakfast	186	1.309490
Seafood	89	0.626584

Observations

1. Most of foods are from Fruits and Vegetables, Snack Foods and Baking Goods category

2. Food from the Starchy Food, Breakfast and Seafood are less.

Analysis of Outlet_Identifier



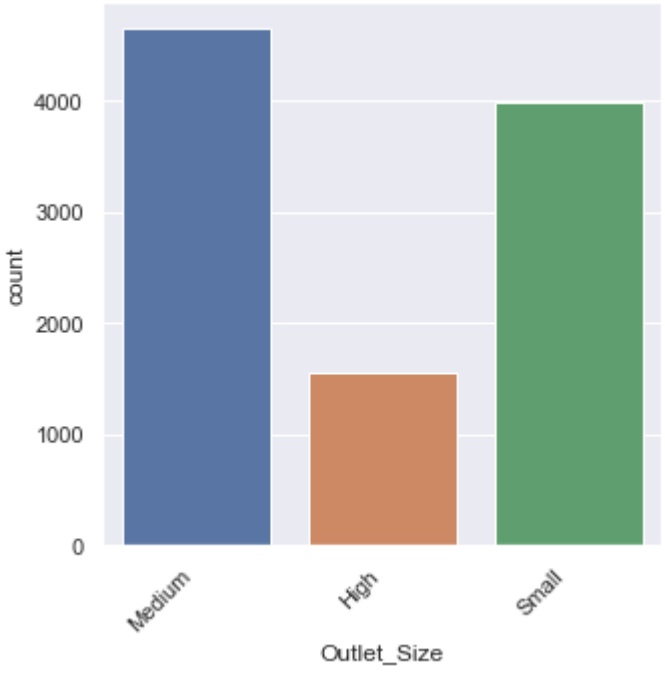
	Count	Count%
OUT027	1559	10.975781
OUT013	1553	10.933540
OUT049	1550	10.912419
OUT046	1550	10.912419
OUT035	1550	10.912419
OUT045	1548	10.898338
OUT018	1546	10.884258
OUT017	1543	10.863137
OUT010	925	6.512250
OUT019	880	6.195438

Observations

1. Number of products are same in outlets having numbers OUT49, OUT35, OUT46.

2. Highest number of products are in OUT027 and least in OUTO10

Analysis of Outlet_Size



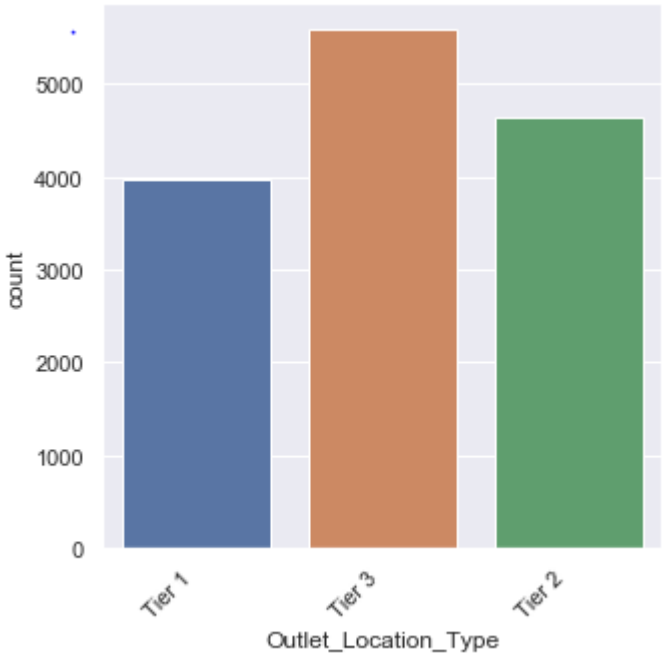
	Count	Count%
Medium	4655	32.772458
Small	3980	28.020276
High	1553	10.933540

Observations

1. Most of the outlets have medium size followed by small size.

2. Only 10% of outlets are in high category with respect to size.

Analysis of Outlet_Location_Type



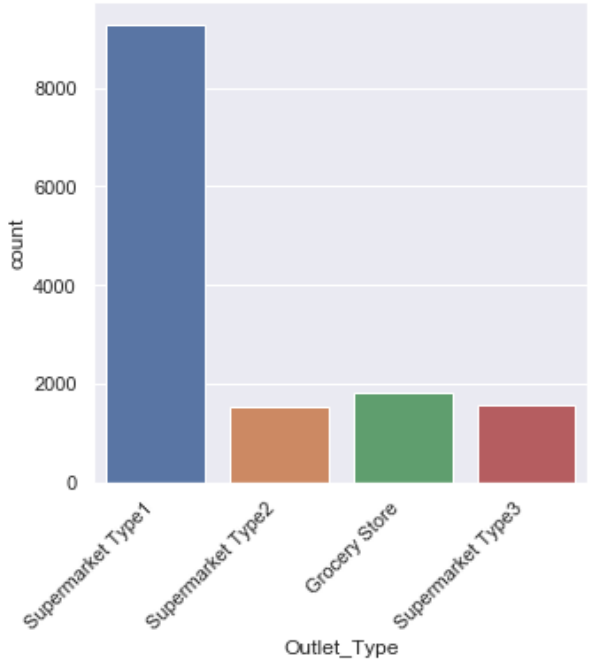
	Count	Count%
Tier 3	5583	39.305829
Tier 2	4641	32.673895
Tier 1	3980	28.020276

Observations

1. 39% of outlets are in Tier 3

2. Tier 1 has least number of outlets

Analysis of Outlet_Type



	Count	Count%
Supermarket Type1	9294	65.432273
Grocery Store	1805	12.707688
Supermarket Type3	1559	10.975781
Supermarket Type2	1546	10.884258

Observations

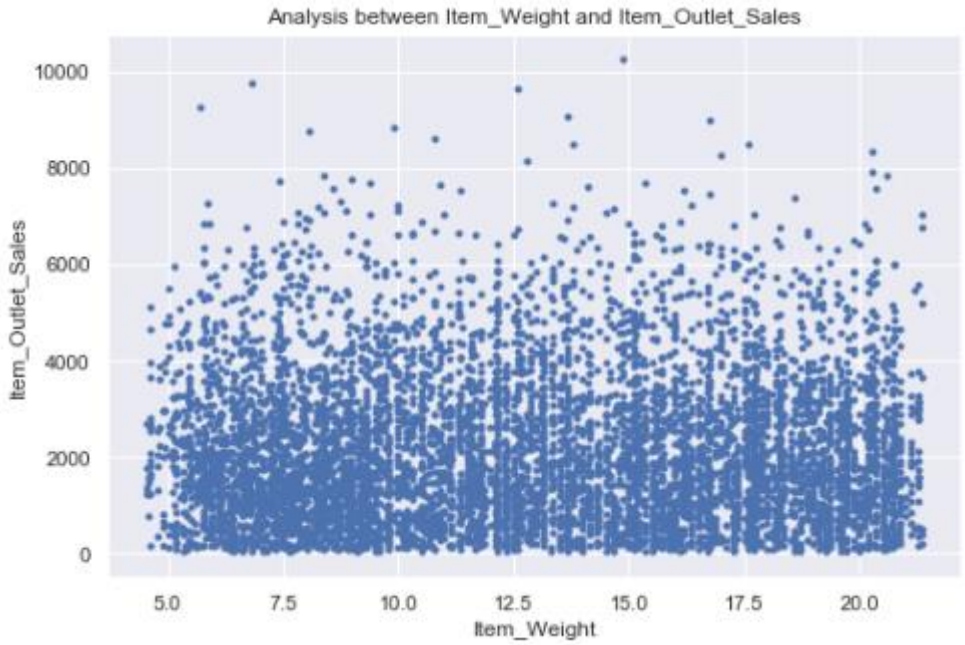
1. 65% of the outlets are of type Supermarket Type 1

2. 10% of the outlets are of type Supermarket Type 1 and Supermarket Type 3

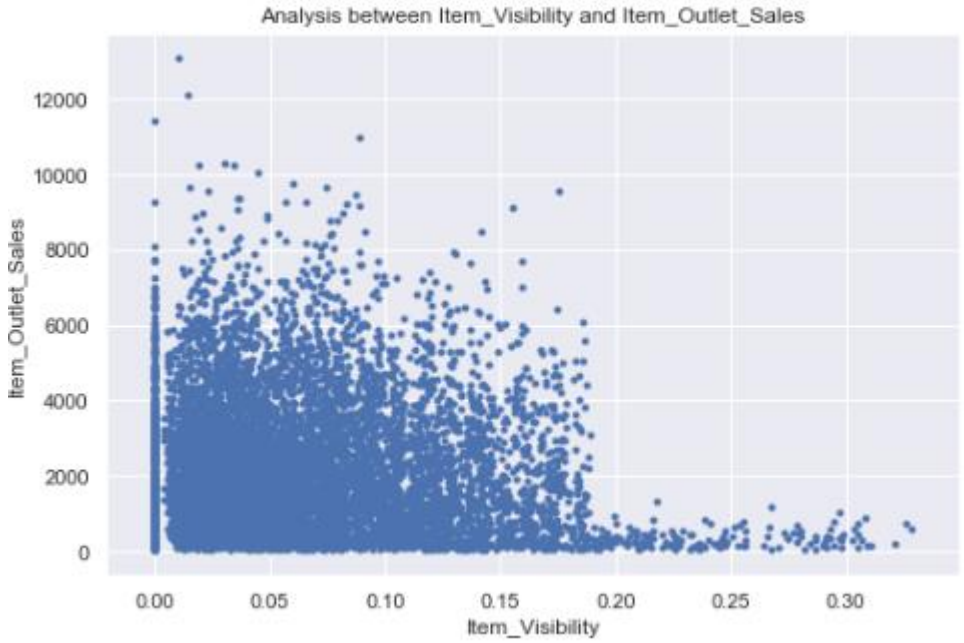
5.2.2 BIVARIATE ANALYSIS

Bi-variate Analysis finds out the relationship between two variables. These variables can be of any combination like categorical and categorical, categorical and continuous and continuous and continuous. Different methods are used to tackle these combinations during analysis process. The analysis between numerical features and target variables is understood using scatter plots. The target variable i.e Item_Outlet_Sales is a continuous variable so below relations between continuous and continuous variables are being analysed.

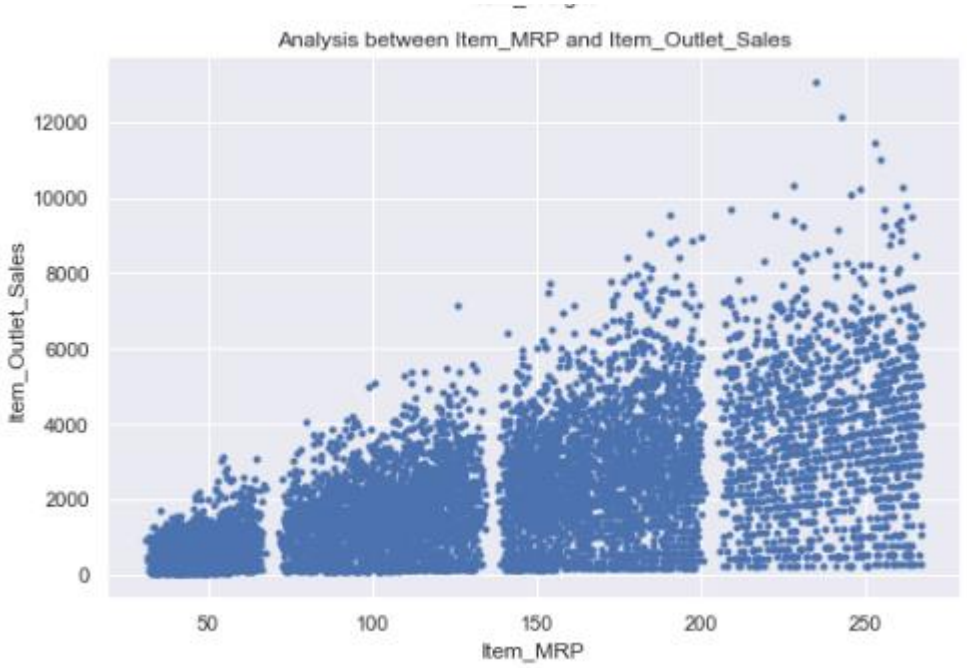
Analysis between Item_Weight and Item_Outlet_Sales



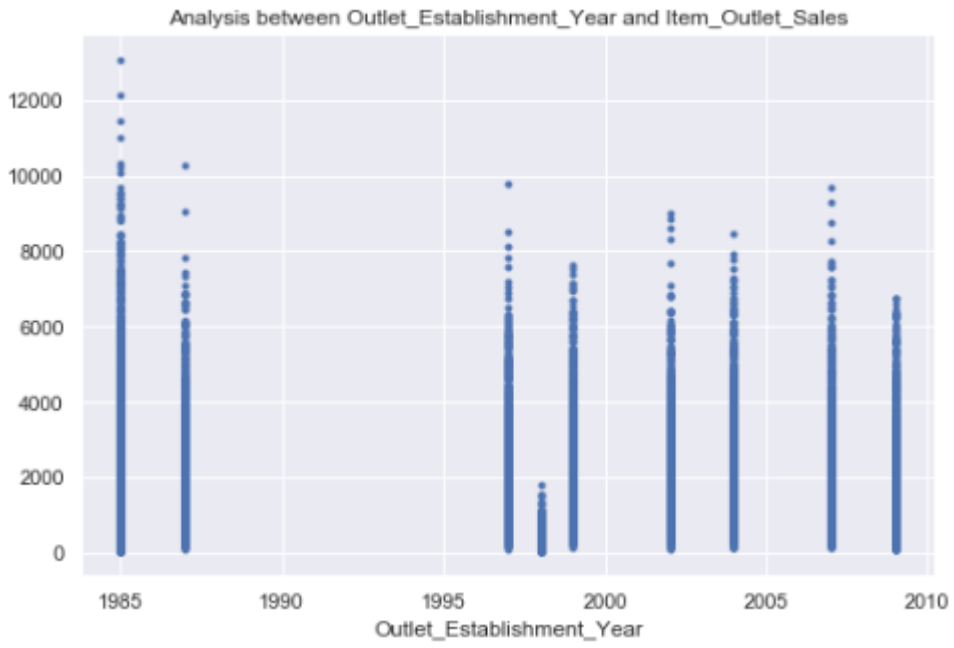
Analysis between Item_Weight and Item_Visibility



Analysis between Item_MRP and Item_Outlet_Sales



Analysis between Outlet_Establishment_Year and Item_Outlet_Sales



Observations

1. There is no pattern between Item_Weight and Outlet_Sales

2. The second plot indictates that if the product is more visible in the store less will be the sales. This plot also shows that there are large number of products with zero visibility.

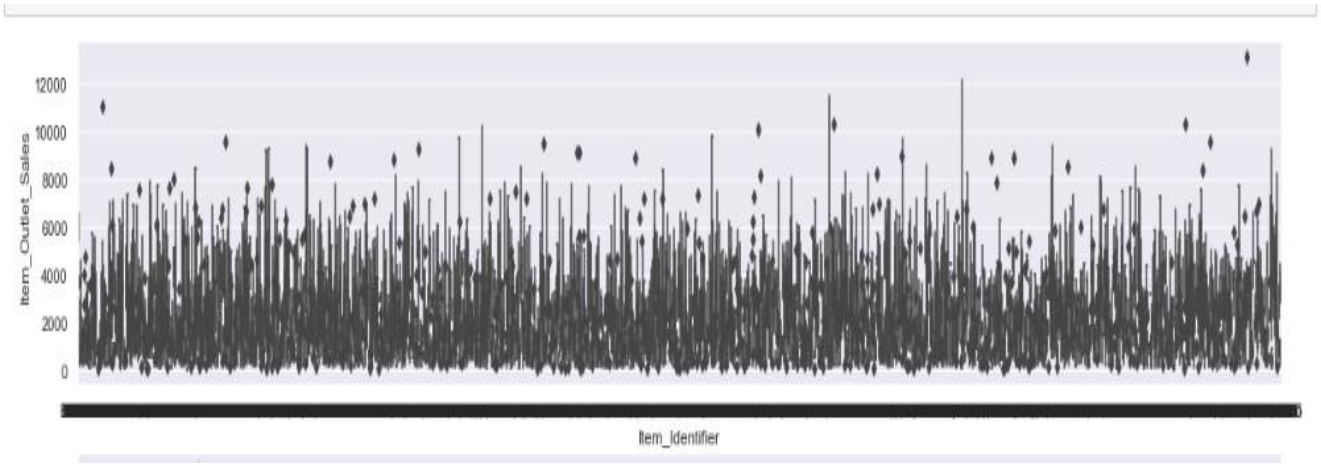
3. In third plot of Item_MRP vs Item_Outlet_Sales, there are four segments of prices that can be used in feature engineering to create a new variable.

4. In fourth plot of Outlet_Establishment_Year vs Item_Outlet_Sales, we can see that Outlet_Establishment_Year should be considered as categorical variable instead of continuous variable which will be corrected later.

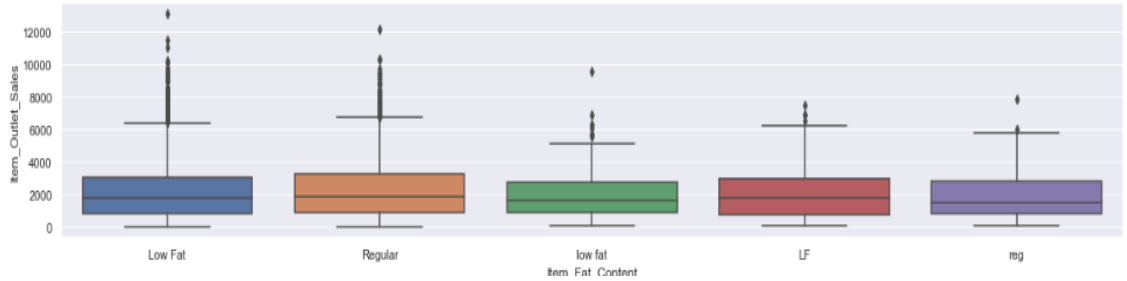
Bivariate Analysis between Categorical and Target variable

Relationships between numerical variables and categorical variables are being analysed using the box-and-whisker plot

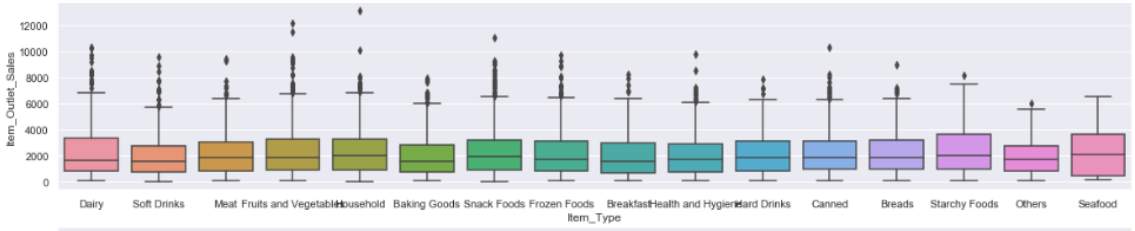
Analysis between Item_Identifier and Item_Outlet_Sales



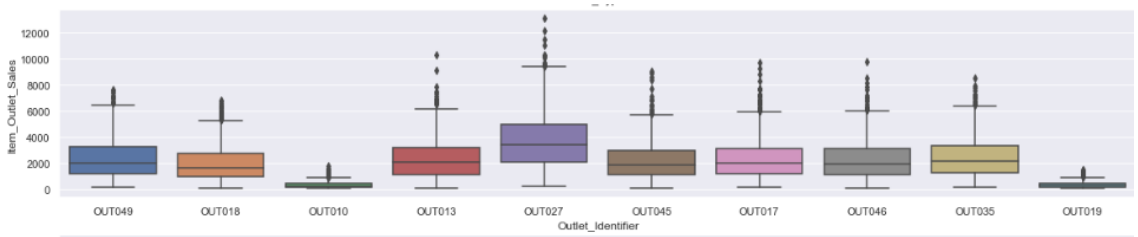
Analysis between Item_Fat_Content and Item_Outlet_Sales



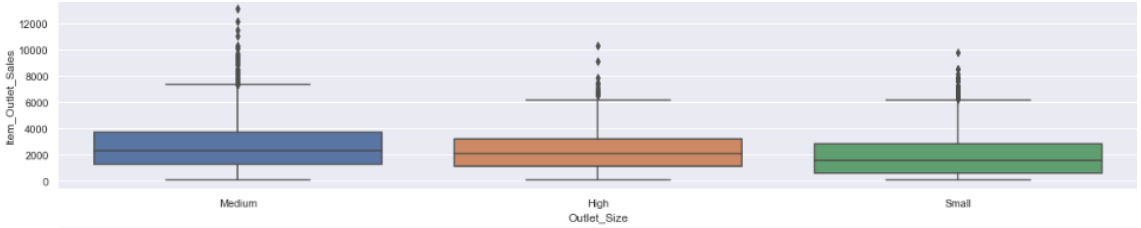
Analysis between Item_Type and Item_Outlet_Sales



Analysis between Outlet_Identifier and Item_Outlet_Sales



Analysis between Outlet_Size and Item_Outlet_Sales



Observations

1. There are no boxplots between Item_Outlet_Sales and Item_Identifier as Item_Identifier is a number.

2. Categories are to be changed in second plot which will be done later.

3. Starchy food and seafood are having more sales.

4. Sales are mostly in OUT027 and least in OUT019

5. Most of the sales are in outlets having medium size

6. Tier 3 is having highest number of sales

5.3 DATA CLEANING

Data cleaning is the process of identifying and removing (or correcting) inaccurate records from a dataset, table, or database. Steps like Imputing Missing Values and Treatment of Outliers are performed. The values of Item Weight column are imputed using mean and values of column Outlet Size are imputed using mode. Outliers have not been treated because tree algorithms like random forest, decision tree, xgboost are impervious to outliers. The missing values of Item Outlet Sales belong to the test dataset so we will be imputing the missing values of Item Weight and Outlet Size of the dataset.

```
Item_Identifier               0
Item_Weight                2439
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                4016
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales          5681
source                        0
dtype: int64
```

In the above image, the numbers written beside each column shows the number of missing values.

```
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          5681
source                        0
dtype: int64
```

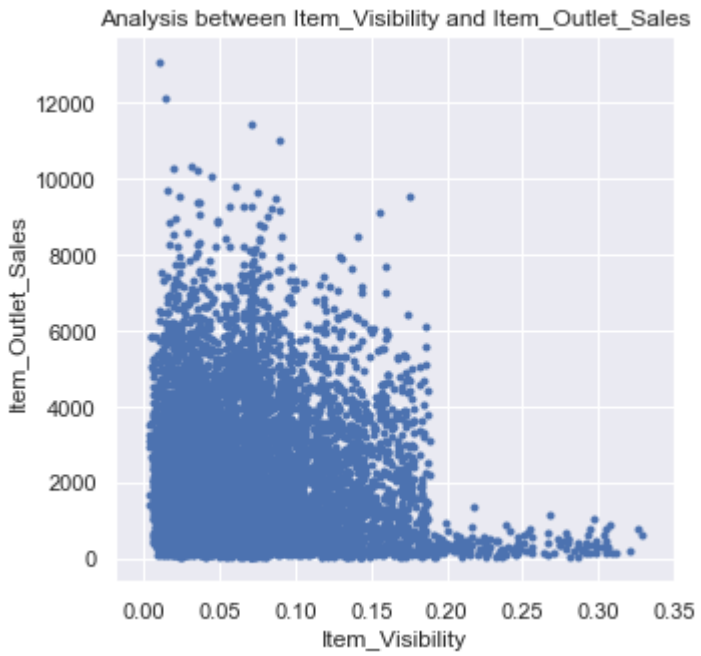
The above image shows that missing values of columns Item_Weight and Outlet_Size have been imputed and Item_Outlet_Sales is not imputed.

5.4 FEATURE ENGINEERING

Feature Engineering is an art of extracting information from existing data. The two techniques that are performed in feature engineering are feature creation and feature transformation. A new feature is to be created based on previous analysis of data. Feature transformation is about changing the feature like one hot encoding of categorical variables.

5.4.1 Modify Item Visibility Column

There were a lot of zero values when performed bivariate analysis between Item Visibility and Item Outlet Sales column. The zero values have been imputed with the help of mean and below image is the outcome of imputation performed.



5.4.2 Determine the years of operation of store

Outlet Establishment Year column contains the years like 1985, 1987 etc and the data that is being used is collected in 2013 so from this information a new feature i.e Outlet Years has been created which determines the operations of outlets of BigMart store.

```
count    14204.000000
mean        15.169319
std          8.371664
min          4.000000
25%          9.000000
50%         14.000000
75%         26.000000
max         28.000000
Name: Outlet_Years, dtype: float64
```

5.4.3 Create broad category on Item_Type

There are almost 16 categories of Item Type which might not prove to be very useful in analysis. First two letters of Item Identifier tells that categories of Item Type can be grouped together in broader categories like FD stands for Food, DR stands for Drinks and NC stands for Non Consumable. The broader category created is Item Type Combined column.

```
Food              10201
Non Consumable     2686
Drinks             1317
Name: Item_Type_Combined, dtype: int64
```

The above image shows that items have been grouped and the name of item type along with their counts.

5.4.4 Create category in Item_Fat_Content

There is need of creating a category Non Edibile for the Non Consumables category of Item_Type as Low and High Fat category would not be suitable.

5.4.5 Encoding of Categorical Data

One hot encoding is performed on all the categorical variables like Item_Fat_Content, Item_Type_Combined, Outlet_Size, Outlet_Location_Type, Outlet_Type, Outlet_Years. The categories of the columns are first converted into numbers using Label Encoder and then they are converted into dummy variables.

Outlet_Years
4
0
4
5
7

This shows the number of years of operation of the store

Outlet_Years_0	Outlet_Years_1	Outlet_Years_2	Outlet_Years_3	Outlet_Years_4	Outlet_Years_5	Outlet_Years_6	Outlet_Years_7	Outlet_Years_8
0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0

Outlet_Years has been transformed into 8 columns i.e Outlet Year 0, Outlet Year 1 and so on. The first value in Outlet Years column was four so a number one has been displayed in Outlet Year 4 and zeroes are displayed in all other columns in the first row.

5.5 MODEL BUILDING

5.5.1 Exporting Data and Reading modified files

The columns are dropped which have been already converted into features. Dataset is divided into train and test datasets using source column and all the unnecessary columns are dropped and then they are converted into modified train and test datasets. These modified files are read with the help of pandas library.

5.5.2 Linear Regression

Linear regression is an algorithm which is used to establish linear relationship between independent and dependent variable. The best fit line is obtainted using the method of Least of Squares.In this method sum of squares are taken for every data point and are added together and are minimized. Higher values are penalized more as compared to lower values..

```
R^2 Score is 0.563479254447071
MSE: 1271060.7831030414
RMSE: 1127.4133151169722
```

5.5.3 Ridge Regression

Ridge Regression is used when the data suffers from multicollinearity and solves the problem using shrinkage parameter. It uses l2 regularization and reduces the variance which helps in preventing overfitting.

```
Value of alpha is 1.0
R^2 Score : 0.5634790577522832
RMSE: 1127.413569121278
```

5.5.4 Lasso Regression

The full form of Lasso is Least Absolute Shrinkage and Selection Operator. Lasso Regression uses l1 regularization and can be used for feature selection when there are more number of features.

```
R^2 Score : 0.5634107280016212
MSE: 1271260.3183560714
RMSE: 1127.5018041475905
```

5.5.5 Decision Tree

Decision Tree is a supervised learning algorithm which works on both categorical and numerical data. The dataset is divided into smaller datasets. The branches represents the decisions nodes and the end nodes represents the result. Decision trees are called classification trees if the target variables takes the discrete values and if target variables are of the continuous values then we call it as regression trees.

```
R^2 Score : 0.60360465585572
MSE : 15831.22884300096
RMSE: 125.82221124666725
```

5.5.6 XGBoost

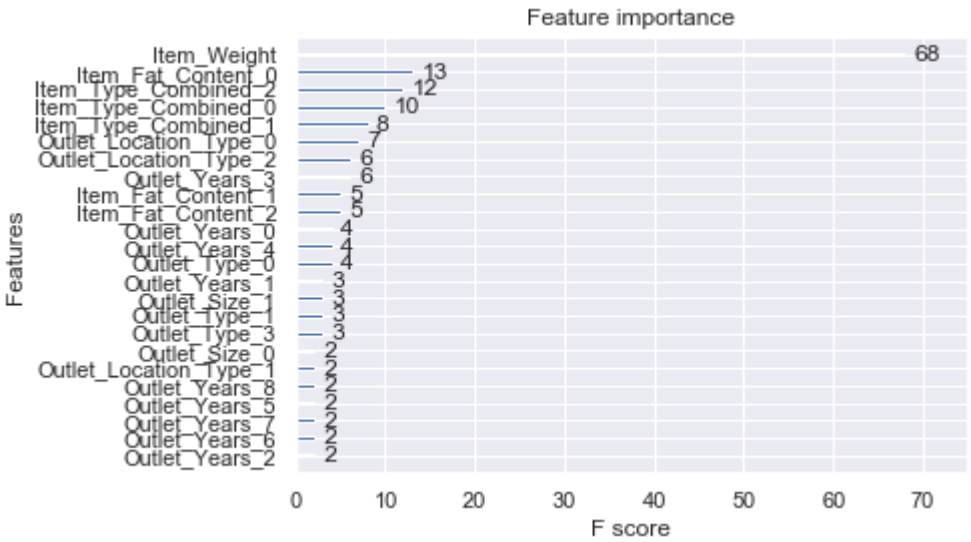
The full form of XGBoost is Extreme Gradient Boosting algorithm. This provides the importance of features after training the model The XGBoost is a gradient boosting decision tree algorithm. Gradient boosting is an approach where weak learners are added to minimize the residual loss errors.

```
R^2 Score : 0.60360465585572
MSE : 15831.22884300096
RMSE: 125.82221124666725
```

These are values obtained while applying XGBoost algorithm.

```
RMSE: 1032.2717773999998
```

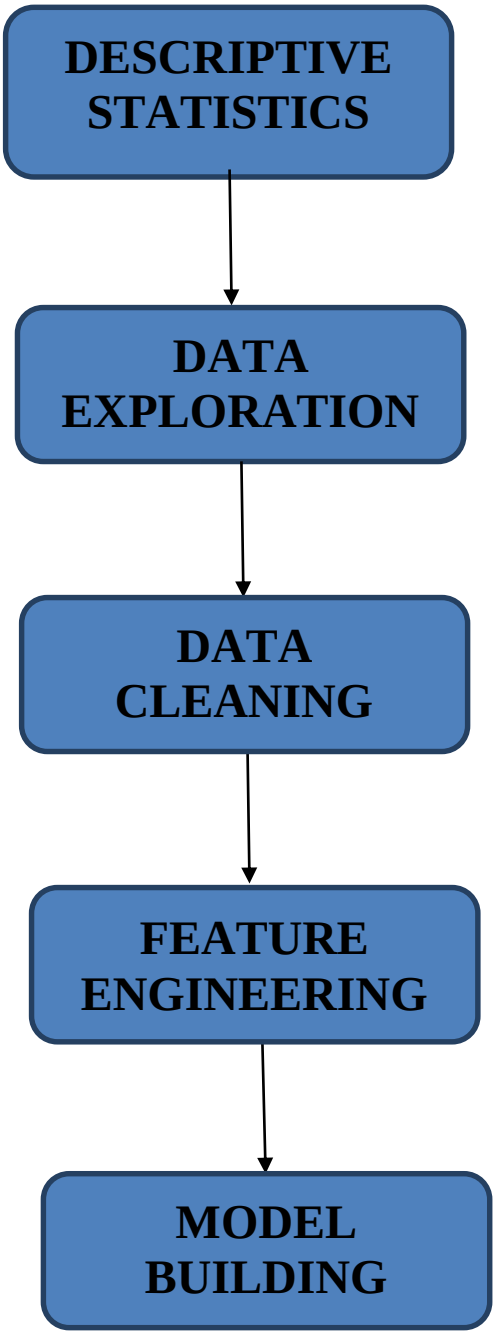
These is the value obtained while applying XGBoost algorithm with cross validation.



6. DESIGN

6.1 FLOWCHART

Steps used for building a predictive model is shown by the flowchart below



6.2 DATA DICTIONARY

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 particular store. This is the outcome variable to be predicted.

7. CODING

12.1 Importing the libraries

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import mean_squared_error

from sklearn.linear_model import LinearRegression

from sklearn.linear_model import RidgeCV

from sklearn.linear_model import Lasso

from sklearn.tree import DecisionTreeRegressor

import xgboost as xgb
```

12.2 Reading the files

```
# Read files

train_data = pd.read_csv("Train.csv")

test_data = pd.read_csv("Test.csv")

# Displays first five rows of train_data

train_data.head()

# Making a new column source in train and test datasets

train_data['source'] = 'train'

test_data['source'] = 'test'
```

```
# Combining train and test datasets

data = pd.concat([train_data, test_data],sort=False)


# Displaying shapes of datasets

print(f"Shape of train dataset is {train_data.shape}")

print(f"Shape of test dataset is {test_data.shape}")

print(f"Shape after combining of dataset is {data.shape}")
```

12.3 Descriptive Statistics

```
data.info() # Prints the summary of dataset

data.nunique() # Unique values of each column

data.describe() # Prints the summary statistics of numerical variables

# Visualizing the data

data.hist(figsize=(10,10),layout=(3,3))

plt.show()

data.describe(include="all") # Prints the summary statistics of all variables (numerical and categorical)

sns.distplot(train_data['Item_Outlet_Sales']); # Plotting the skewness of target variable

print(f"Skewness of Item Outlet Sales is {train_data['Item_Outlet_Sales'].skew()}")

# Applying log transformation to the target variable

target = np.log(train_data['Item_Outlet_Sales'])

print(f"Skewness is {target.skew()}")

# Visualizing target variable after applying log transformation

sns.distplot(target);

# Applying square root transformation to the target variable
```

```
target = np.sqrt(train_data['Item_Outlet_Sales'])

print(f"Skewness is {target.skew()}")



# Visualizing the target variable after applying square root transformation

sns.distplot(target);
```

12.4 Data Exploration

12.4.1 Univariate Analysis

Analysis of Item_Weight

```
# Plotting Histogram of Item_Weight

ax = sns.distplot(data['Item_Weight'].dropna(), hist=True, bins=15, hist_kws={'edgecolor':'black'});

ax.set(ylabel="% of distribution", title="Ananlysis of Item_weight")

# Metrics for analysis of Item_Weight

data['Item_Weight'].describe()
```

Analysis of Item_Visibility

```
# Plotting Histogram of Item_Visibility

ax = sns.distplot(data['Item_Visibility'].dropna(), hist=True, bins=15, hist_kws={'edgecolor':'black'});

ax.set(ylabel="% of distribution", title="Ananlysis of Item_Visibility");

# Metrics for analysis of Item_Visibility

data['Item_Visibility'].describe()
```

Analysis of Item_MRP

```
# Plotting Histogram of Item_MRP

ax = sns.distplot(data['Item_MRP'].dropna(), hist=True, bins=15, hist_kws={'edgecolor':'black'});

ax.set(ylabel="% of distribution", title="Ananlysis of Item_MRP");

# Metrics for analysis of Item_MRP

data['Item_MRP'].describe()
```

Analysis of Outlet_Establishment_Year

```
# Plotting Histogram of Outlet_Establishment_Year

ax = sns.distplot(data['Outlet_Establishment_Year'].dropna(), hist=True, bins=15, hist_kws={'edgecolor':'black'});

ax.set(ylabel="% of distribution", title="Outlet_Establishment_Year");

# Metrics for analysis of Outlet_Establishment_Year

data['Outlet_Establishment_Year'].describe()

# Separate numerical and categorical data

numerical_data = train_data.select_dtypes(include=[np.number])

categorical_data = train_data.select_dtypes(exclude=[np.number])

print(f"There are {numerical_data.shape[1]} columns having numerical data- {', '.join(numerical_data.columns)}")

print(f"There are {categorical_data.shape[1]} columns having categorical data - {', '.join(categorical_data.columns)}")
```

Correlation Matrix of numerical with target variable

```
corr = numerical_data.corr()

fig, ax = plt.subplots(figsize=(7,5))

sns.heatmap(corr, square=True);
```

Correlation Scores with Target Variable

```
print(corr["Item_Outlet_Sales"].sort_values(ascending=False))

print(data['Item_Fat_Content'].value_counts()) # Category along with their counts will be displayed
```

Analysis of Item_Fat_Content

```
# Replacing required categories

data['Item_Fat_Content'] = data['Item_Fat_Content'].replace({'LF':'Low Fat', 'low fat':'Low Fat', 'reg':'Regular'})

print(data['Item_Fat_Content'].value_counts())

# Plotting the category Item_Fat_Content

sns.set(style="darkgrid")

sns.countplot(data['Item_Fat_Content']);

count = data['Item_Fat_Content'].value_counts().rename('Count')

count_percent=((data['Item_Fat_Content'].value_counts()/data['Item_Fat_Content'].size) * 100).rename('Count%')

freq_table = pd.concat([count,count_percent],axis=1)

print(freq_table)
```

Analysis of Item_Type

```
# Making a frequency table

count = data['Item_Type'].value_counts().rename('Count')

count_percent = ((data['Item_Type'].value_counts()/data['Item_Fat_Content'].size) * 100).rename('Count%')

freq_table = pd.concat([count,count_percent],axis=1)

print(freq_table)

# Plotting the category Item_Type

sns.set(style="darkgrid")

plt.figure(figsize=(5,5))

sns.countplot(data['Item_Type'])

plt.xticks(rotation=45, horizontalalignment='right')

plt.show()
```

Analysis of Outlet_Identifier

```
# Making a frequency table

count = data['Outlet_Identifier'].value_counts().rename('Count')

count_percent = ((data['Outlet_Identifier'].value_counts()/data['Outlet_Identifier'].size) * 100).rename('Count%')

freq_table = pd.concat([count,count_percent],axis=1)

print(freq_table)

# Plotting the category Outlet_Identifier
```

```
sns.set(style="darkgrid")

plt.figure(figsize=(5,5))

sns.countplot(data['Outlet_Identifier'])

plt.xticks(rotation=45, horizontalalignment='right')

plt.show()
```

Analysis of Outlet_Size

```
# Making a frequency table

count = data['Outlet_Size'].value_counts().rename('Count')

count_percent = ((data['Outlet_Size'].value_counts()/data['Outlet_Size'].size) * 100).rename('Count%')

freq_table = pd.concat([count,count_percent],axis=1)

print(freq_table)

# Plotting the category Outlet_Size

sns.set(style="darkgrid")

plt.figure(figsize=(5,5))

sns.countplot(data['Outlet_Size'])

plt.xticks(rotation=45, horizontalalignment='right')

plt.show()
```

Analysis of Outlet_Location_Type

```
# Making a frequency table

count = data['Outlet_Location_Type'].value_counts().rename('Count')
```

```
count_percent = ((data['Outlet_Location_Type'].value_counts()/data['Outlet_Location_Type'].size) * 100).rename('Count%')

freq_table = pd.concat([count,count_percent],axis=1)

print(freq_table)

# Plotting the category Outlet_Location_Type

sns.set(style="darkgrid")

plt.figure(figsize=(5,5))

sns.countplot(data['Outlet_Location_Type'])

plt.xticks(rotation=45, horizontalalignment='right')

plt.show()
```

Analysis of Outlet_Type

```
# Making a frequency table

count = data['Outlet_Type'].value_counts().rename('Count')

count_percent = ((data['Outlet_Type'].value_counts()/data['Outlet_Type'].size) * 100).rename('Count%')

freq_table = pd.concat([count,count_percent],axis=1)

print(freq_table)

# Plotting the category Outlet_Type

sns.set(style="darkgrid")

plt.figure(figsize=(5,5))

sns.countplot(data['Outlet_Type'])
```

```
plt.xticks(rotation=45, horizontalalignment='right')

plt.show()
```

12.4.2 Bivariate Analysis

Analysing Relationship between Numerical Features and Target Variable

```
fig, axes = plt.subplots(2,2,figsize=(15,10))

axes[0,0].scatter(data['Item_Weight'], data['Item_Outlet_Sales'], marker='.')

axes[0,0].set(title="Analysis between Item_Weight and Item_Outlet_Sales", xlabel="Item_Weight", ylabel="Item_Outlet_Sales")

axes[0,1].scatter(data['Item_Visibility'], data['Item_Outlet_Sales'], marker='.')

axes[0,1].set(title="Analysis between Item_Visibility and Item_Outlet_Sales", xlabel="Item_Visibility", ylabel="Item_Outlet_Sales")

axes[1,0].scatter(data['Item_MRP'], data['Item_Outlet_Sales'], marker='.')

axes[1,0].set(title="Analysis between Item_MRP and Item_Outlet_Sales", xlabel="Item_MRP", ylabel="Item_Outlet_Sales")

axes[1,1].scatter(data['Outlet_Establishment_Year'], data['Item_Outlet_Sales'], marker='.')

axes[1,1].set(title="Analysis between Outlet_Establishment_Year and Item_Outlet_Sales", xlabel="Outlet_Establishment_Year", ylabel="Item_Outlet_Sales")

fig.tight_layout()

plt.show()
```

Analysing Relationship between Categorical Features and Target Variable

```
fig, axes = plt.subplots(6,1,figsize=(20,25))

for category,ax in zip(categorical_data.columns,axes.flatten()):

   if category != 'source':

      sns.boxplot(x=category, y=data["Item_Outlet_Sales"], data=categorical_data, ax=ax);
```

12.5 DATA CLEANING

12.5.1 Dealing with missing values

```
data.isnull().sum() # Count of missing values of each column in dataset

# Imputing the values of Item_Weight column

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

imputer.fit(data[['Item_Weight']])

data[['Item_Weight']] = imputer.transform(data[['Item_Weight']])

# Imputing the values of Outlet_Size column

imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')

imputer.fit(data[['Outlet_Size']])

data[['Outlet_Size']] = imputer.transform(data[['Outlet_Size']])
```

12.6 FEATURE ENGINEERING

12.6.1 Modify Item_Visibility

```
# Imputing zeroes with mean values
imputer = SimpleImputer(missing_values=0, strategy='mean')
imputer.fit(data[['Item_Visibility']])
data[['Item_Visibility']] = imputer.transform(data[['Item_Visibility']])
# Displaying scatterplot with mean values instead of zeroes
fig, ax = plt.subplots(figsize=(5,5));
plt.scatter(x=data["Item_Visibility"], y=data["Item_Outlet_Sales"], marker='.')
plt.xlabel("Item_Visibility")
plt.ylabel("Item_Outlet_Sales")
plt.title("Analysis between Item_Visibility and Item_Outlet_Sales")
plt.show()
```

12.6.2 Determine the years of operation of store

```
# Creating a new feature from Outlet_Establishment_Year
data['Outlet_Years'] = 2013 - data['Outlet_Establishment_Year']
data['Outlet_Years'].describe()
```

12.6.3 Create broad category on Item_Type

```
# Get the first two characters of Item_Identifier
data['Item_Type_Combined'] = data['Item_Identifier'].apply(lambda x: x[0:2])
# Creating broader categories
data['Item_Type_Combined'] = data['Item_Type_Combined'].map({'FD':'Food',
                                                             'NC':'Non Consumable',
                                                             'DR':'Drinks'})
```

```
# Dropping the column Item_Type

data['Item_Type_Combined'].value_counts()
```

12.6.4 Create category in Item_Fat_Content

```
data.loc[data['Item_Type_Combined']=='Non Consumable',"Item_Fat_Content"] = "Non-Edible"

data['Item_Fat_Content'].value_counts()
```

12.6.5 Encoding of Categorical Data

```
# Changing the categorical columns with numbers

categorical_cols = ['Item_Fat_Content', 'Item_Type_Combined', 'Outlet_Size',

                    'Outlet_Location_Type', 'Outlet_Type', 'Outlet_Years']

le = LabelEncoder()

for col in categorical_cols:

    data[col] = le.fit_transform(data[col])

data.head()


# One Hot Encoding of columns

data = pd.get_dummies(data, columns=categorical_cols)

data.head()
```

12.7 BUILDING MODELS

12.7.1 Exporting data and Reading modified data

```
# Dropping columns which have been already converted into another features

data.drop(['Item_Type','Outlet_Establishment_Year'], axis=1, inplace=True)

# Dividing the dataset into train and test datasets

train = data.loc[data['source']=="train"].copy()

test = data.loc[data['source']=="test"].copy()

# Dropping the columns

test.drop(['Item_Outlet_Sales','source'], axis=1, inplace=True)

train.drop('source', axis=1, inplace=True)

# Converting the datasets to .csv files

train.to_csv("train_modified.csv", index=False)

test.to_csv("test_modified.csv", index=False)

# Reading the modified files of train and test dataset

train_df = pd.read_csv("train_modified.csv")

test_df = pd.read_csv("test_modified.csv")
```

12.7.2 Applying Linear Regression

```
# Linear Regression Model

features = [col for col in train_df.columns if col not in ('Item_Outlet_Sales', 'Item_Identifier', 'Outlet_Identifier')]

target = 'Item_Outlet_Sales'

X_train = train_df[features]

y_train = train_df['Item_Outlet_Sales']

regressor = LinearRegression()

regressor.fit(X_train, y_train)
```

```
y_pred = regressor.predict(X_train)

mse = mean_squared_error(y_train,y_pred)

rmse = np.sqrt(mse)

print(f"R^2 Score is {regressor.score(X_train, y_train)}")

print(f"MSE: {mse}")

print(f"RMSE: {rmse}")
```

12.7.2 Applying RidgeCV Regression

```
# Ridge Regression Model

ridge_cv = RidgeCV(alphas=[0.001,0.1,1.0])

ridge_cv.fit(X_train,y_train)

y_pred = ridge_cv.predict(X_train)

mse = mean_squared_error(y_train,y_pred)

rmse = np.sqrt(mse)

print(f"Value of alpha is {ridge_cv.alpha_}")

print(f"R^2 Score : {ridge_cv.score(X_train,y_train)}")

print(f"RMSE: {rmse}")
```

12.7.3 Applying Lasso Regression

```
# Lasso Regression Model

lasso = Lasso()

lasso.fit(X_train,y_train)

y_pred = lasso.predict(X_train)

mse = mean_squared_error(y_train,y_pred)

rmse = np.sqrt(mse)

print(f"R^2 Score : {lasso.score(X_train,y_train)}")

print(f"MSE: {mse}")  print(f"RMSE: {rmse}")
```

12.7.4 Applying Decision Tree Regression

```
# Decision Tree Regression Model

decision_tree = DecisionTreeRegressor(max_depth=5)

decision_tree.fit(X_train, y_train)

y_pred = decision_tree.predict(X_train)

mse = mean_squared_error(y_train,y_pred)

rmse = np.sqrt(mse)

print(f"R^2 Score : {decision_tree.score(X_train,y_train)}")

print(f"MSE: {mse}")

print(f"RMSE: {rmse}")
```

12.7.5 Applying XGBoost

```
# XGBOOST Model

xg_reg = xgb.XGBRegressor(n_estimators=1000, learning_rate = 0.3)

xg_reg.fit(X_train,y_train)

y_pred = xg_reg.predict(X_train)

mse = mean_squared_error(y_train,y_pred)

rmse = np.sqrt(mse)

print(f"R^2 Score : {decision_tree.score(X_train,y_train)}")

print(f"MSE : {mse}")

print(f"RMSE: {rmse}")
```

12.7.6 XGBoost Model using cross validation

```
# XGBOOST Model using cross validation

data_dmatrix = xgb.DMatrix(data=X_train,label=y_train)

params = {"objective":"reg:squarederror",'colsample_bytree': 0.3,'learning_rate': 0.3,

            'max_depth': 5, 'alpha': 10}

cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=5,

                num_boost_round=50,early_stopping_rounds=10,metrics="rmse",
as_pandas=True, seed=100)

rmse = cv_results['train-rmse-mean'][29]

print(f"RMSE: {rmse}")
```

Features Importance Graph

```
# Plotting features importance

xg_reg = xgb.train(params=params, dtrain=data_dmatrix, num_boost_round=10)

xgb.plot_importance(xg_reg)

plt.rcParams['figure.figsize'] = [5, 5]

plt.show()
```

10. FUTURE ENHANCEMENTS

10.1 Deployment of ML Model

The Machine Learning model which is built for sales forecasting methods can be deployed using Flask. After this deployment, this model will be able to predict any new data and can be used by everyone in the world online.

10.2 Usage of Neural Networks

The artificial neural network can also be used to optimize the process of sales forecasting. A neural network takes some inputs, recognizes patterns in data and predicts the output.

10.3 Improvement in Data Collection

There are a number of parameters which affect the sales that are not taken into account like economic factors, competition, customer behaviour etc so data should be collected taking the parameters into account. This will be helpful to make a predictive model providing better accuracy.

11. CONCLUSION

The main goal of this project was to study the different steps required to build a predictive model and develop it. A considerable amount of time was taken to do the research on analysis of dataset in particular. All the libraries required like numpy, pandas, matplotlib, seaborn, sklearn etc were studied.

After studying the theoretical concepts, the focus was shifted on the practical implementation. As a result, a predictive model for sales forecasting was developed. It was a challenge, but contained a massive number of tasks which needs to be finished. The project was built from scratch as it was the best opportunity to learn Machine Learning.

I learned a lot about analysis of data and applied algorithms on the dataset. During the project, I faced a lot of challenges and solving them boosted my confidence. A lot of new experience was gained during this project.

12. BIBLIOGRAPHY

1. https://www.analyticsvidhya.com/blog/2016/02/bigmart-sales-solution-top-20/
2. https://medium.com/diogo-menezes-borges/project-1-bigmart-sale-prediction-fdc04f07dc1e