Department of Computer Science &Technology

2022-2023

Mini Project Report

On

"BIG MARKET SALES PREDICITION"

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

(DATASCIENCE)

By

Yamasani Venkata Akhil Teja Reddy-20R21A6760

UNDER THE GUIDANCE OF

G. Swapna

(Assistant Professor)

CERTIFICATE

This is to certify that the project entitled "BIGMARKETSALES PREDICITION" has been submitted by Yamasani Venkata Akhil Teja Reddy-(20R21A6760) in the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering (Data Science) from MLR Institute of Technology, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

Internal Guide **Head of the Department**

External Examiner

DECLARATION

I hereby declare that the project entitled "BIG MARKET SALES PREDICITION" is the work done during the period from AUG 2022 to DEC 2022 and is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor of technology in Computer Science and Engineering (Data Science) from MLR Institute of Technology, Hyderabad. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Yamasani Venkata Akhil Teja Reddy-20R21A6760

ACKNOWLEDGEMENT

There are many people who helped me directly and indirectly to complete my project successfully.I would like to take this opportunity to thank one and all.

First of all I would like to express my deep gratitude towards my internal guide **Mrs.G.Swapna Asst.Prof, Department** of CSE(AIMl) for his support in the completion of my dissertation. I wish to express my sincere thanks to **Mr. Kashi Sai Prasad HOD, Department of CSE (Ai-Ml)** and also to principal **Dr.K. Srinivasa Rao** for providing the facilities to complete the dissertation.

I would like to thank all our faculty, coordinators and friends for their help and constructive criticism during the project period. Finally, I am very much indebted to our parents for their moral support andencouragement to achieve goals.

Yamasani Venkata Akhil Teja Reddy-20R21A6760

ABSTRACT

The aim of the project is to predict the sales of Big Markets. The Big Market sales dateset also consists of certain attributes for each product and store. This model helps Big Market understand the properties of products and stores that play an important role in increasing their overall sales. We perform exploratory data analysis, data preprocessing and feature engineering on the dateset. The required outlets are shown as graphs in different ways and software used is jupyter notebook libraries like numpy, pandas, seaborn,etc are imported. We compare, correlate and predict the behavior of the outlet columns.

KEYWORDS: Exploratory data analysis, data pre-processing, feature engineering, jupyter notebook, DataLore, Numpy, pandas, seaborn, Linear Regression, Ridge Regression Randomforest Regressor, Decision tree, K-means clustering.

LIST OF FIGURES

INDEX

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

1. 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.

2. 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.

3. Due to the cheap availability of computing and storage, it has become possible to use sophisticated machine learning algorithms for this purpose. In this paper, we are providing prediction for the sales data of big mart in a number of big mart stores across various location types which is based on the outlets being established and their sales.

MOST COMMON MACHINE LEARNING ALGORITHMS:

1.K Means Clustering Algorithm (Unsupervised Learning - Clustering)

2.Linear Regression (Supervised Learning/Regression).

3.Decision Trees

4.Random Forests

5. Data Lore

1.2 PURPOSE OF THE PROJECT

The purpose of this project is to predict the model using visualization and machine learning algorithms.By this we can predict the sales of the particular outlets using different algorithms like K-means clustering, Random forest regressor, Decision tree.

1.3 MOTIVATION

We see everyday purchase of products in big markets like dmarts,reliance,smart.we need to calculate the outlet sales of each Product in big markets.We consider different types of 9 outlets and calculate the outlet sales of product.so we get the genuine report of the sales of the particular product.

CHAPTER 2
LITERATURE SURVEY

We conducted a through literature survey by reviewing existing systems for predicting the outlet sales. Research papers, journals and publications have also been referred in order to prepare this survey.

2.1 EXISTING SYSTEM

Sales projections offer guidance on how a company should manage its staff, cash flow, and resources. This is a crucial prerequisite for business planning and decision-making. It enables companies to efficiently create their business plans. Only the variables Item MRP, Outlet Identifier, Outlet Establishment Year, Outlet Size, Outlet Location Type, and Outlet Type are relevant at a significant level according to the model description, and simpler models along with proper data cleaning perform well for the regression. Linear regression, Ridge regression, Random forest,and decision trees are examples of learning algorithms that are suitable for sales forecast.Sales forecasting is made simpler with the Random Forest, and the ideal number of trees is carefully specified. A set number of decision trees are integrated in the tree-based algorithm known as Random Forest to create a potent prediction model. The general linear model was found to yield better outcomes, as determined by the RMSE values, when employing the random forest and principal component analysis methodologies. The artificial intelligence paradigm that the Decision Tree technique belongs to produces a tree with the most important function and following nodes in the root node of the tree with features of lower ranking.

2.2 LIMITIONS OF THE EXISTING SYSTEM

- An existing system uses only analysis libraries like numpy,pandas,etc.
- In existing system there is no prediction of sales.

CHAPTER 3

PROPOSED SYSTEM

3.1 PROPOSED SYSTEM

We will explore the problem in following stages:

- Data Exploration — looking at categorical and continuous feature summaries and making inferences about the data.
- Data Cleaning — imputing missing values in the data and checking for outliers.
- Feature Engineering — modifying existing variables and creating new ones for analysis.
- Model Building — making predictive models on the data.

3.2 OBJECTIVES OF PROPOSED SYSTEMS

The objectives of the proposed system are:

- To collect data sets containing various store details regarding the information of the product sales and to perform pre-processing on the data set.
- To predict the accuracy of the model ,we are using machine learning algorithms.

3.3SYSTEM REQUIREMENTS

3.3.1 SOFTWARE REQUIREMENTS

1. Jupyter notebook
2. Python programming language
3. Google chrome/Microsoft edge

3.3.2 HARDWARE REQUIREMENTS

1. Ram – 4GB
2. Hard Disk
3. 64 bit Processor
4. intel i5 core

3.3.3 FUNCTIONAL REQUIREMENTS

Numpy

Large, multi-dimensional arrays and matrices are supported by Numpy, a library for the Python programming language, along with a substantial number of high-level mathematical operations that may be performed on these arrays.

Pandas

For the purpose of manipulating and analysing data, the Python programming language has a software package called pandas. It includes specific data structures and procedures for working with time series and mathematical tables. It is free software distributed under the BSD license's three clauses. Python's Pandas package is used to manipulate data sets.

It offers tools for data exploration, cleaning, analysis, and manipulation.

Seaborn

One outstanding Python module for visualising graphical statistical graphing is Seaborn. To make the production of various statistical charts in Python more visually appealing, Seaborn offers a variety of colour palettes and elegant default styles.

Matplotlib

For the Python programming language and its NumPy numerical mathematics extension, Matplotlib is a graphing library. For integrating charts into programmes utilising all-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK, it offers an object-oriented API.

Plotly

An interactive, open-source plotting toolkit for Python, plotly provides over 40 different chart types for a variety of statistical, financial, geographic, scientific, and three-dimensional use-cases. Plotly offers scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST as well as online graphing, analytics, and statistics capabilities for individuals and groups.

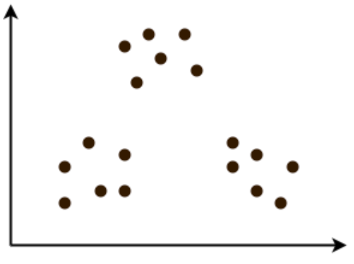
3.4 CONCEPTS USED IN PROPOSED SYSTEM

3.4.1 DATA PREPROCESSING

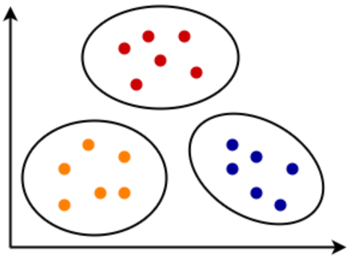
Pre-processing is a data mining technique used to turn the raw data into a format that is both practical and effective.

3.4.2 K- MEANS CLUSTERING

K-means clustering is the most common partitioning algorithm. K-means reassigns each data in the dataset to only one of the new clusters formed. A record or data point is assigned to the nearest cluster using a measure of distance or similarity.



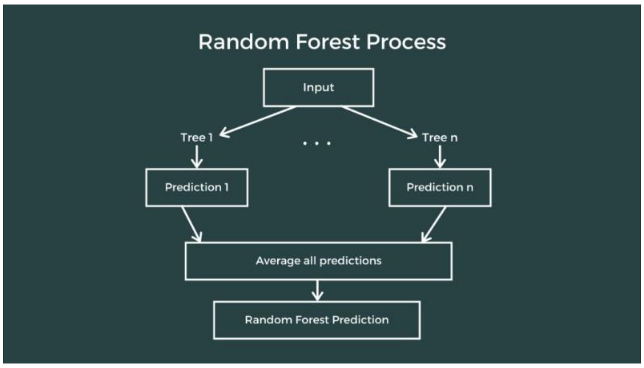
Before K-Means



After K-Means

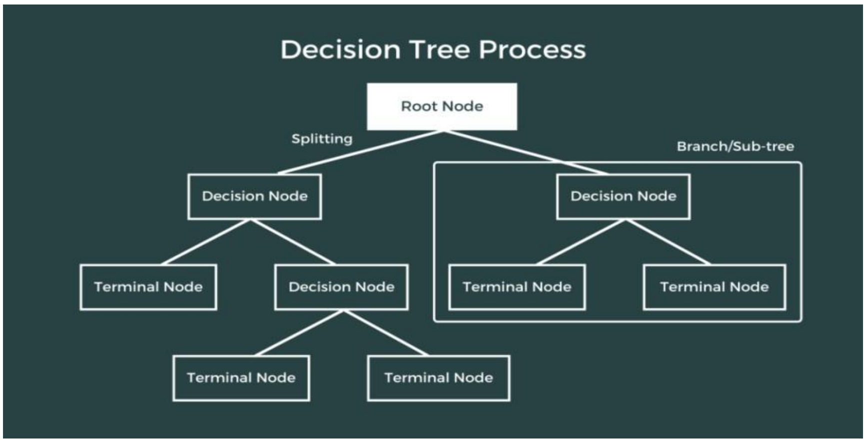
3.4.3 RANDOM FOREST REGRESSOR

A supervised learning technique called Random Forest Regression leverages the ensemble learning approach for regression. The ensemble learning method combines predictions from various machine learning algorithms to provide predictions that are more accurate than those from a single model.



3.4.4 DECISION TREE REGRESSOR

Decision tree regression trains a model in the form of a tree to predict data in the future and generate useful continuous output by observing the properties of an item. Continuous output denotes the absence of discrete output, i.e., output that is not only represented by a discrete, well-known set of numbers or values**.**



3.5 DATA SETS USED IN PROPOSED SYSTEM

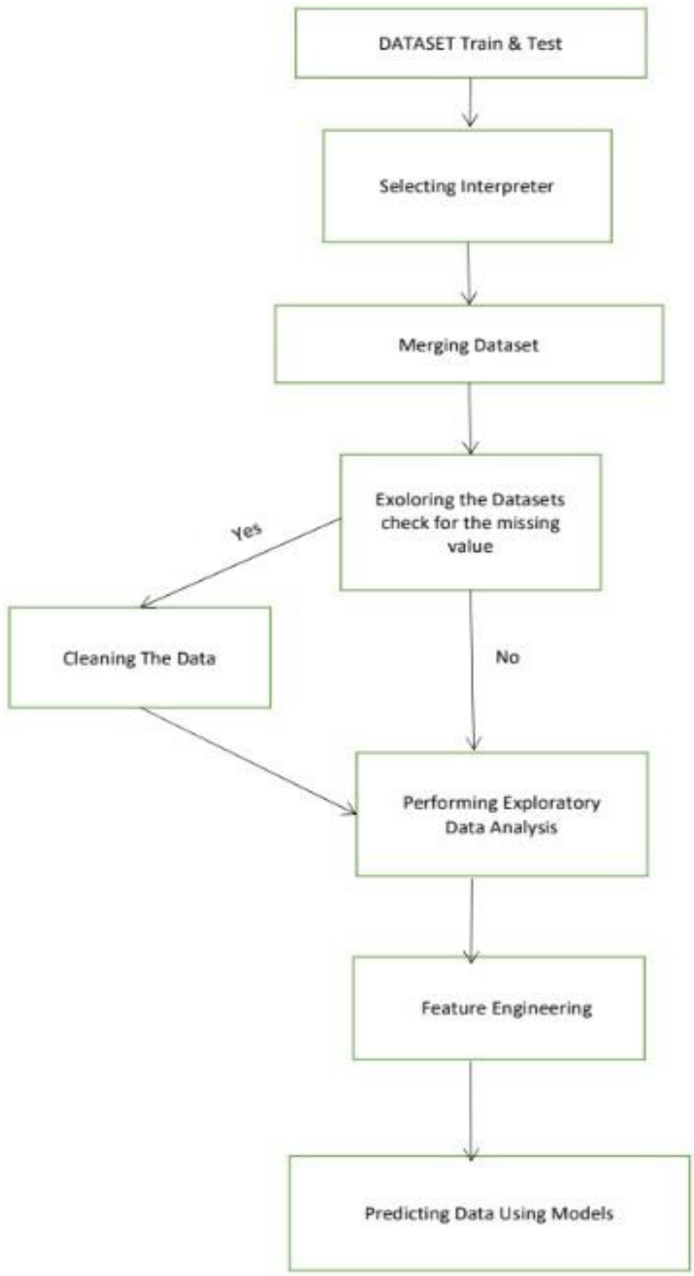
1. "C:\Users\tdiks\Downloads\Test (1).csv"

2. "C:\Users\tdiks\Downloads\Train (1).csv"

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture



CHAPTER 5

IMPLEMENTATION

5.1 SOURCE CODE

STEP 1: Import numpy, pandas,seaborn,warnings,ploty and matplotlib libraries.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

STEP 2: Read two ".csv" datasets

```
data_train=pd.read_csv('Train.csv')
data_train.head()
```

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

```
data_test=pd.read_csv('Test.csv')
data_test.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	NaN	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	

STEP 3: Merge both the datasets

```
#merging two datasets into one
data=pd.concat([data_test,data_train])
data
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locat
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	NaN	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	
...	...	...	...	...	...	...	...	...	...	
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	High	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	NaN	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	Small	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	Medium	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	OUT046	1997	Small	

STEP 4: Check rows, columns, datatypes and null values

```
#Checking rows and columns
data.shape
```

```
(14204, 12)
```

```
#checking datatypes
data.dtypes
```

```
#checking null values
data.isnull().sum()
```

STEP 5: Obtaining statistical values for the given dataset

```
#Checking avaerage of every numerical column
data.describe()
```

	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

Data cleaning

```
#since 75% of the weights are 16.75 items, null values be replaced with 16.75
data['Item_Weight'].fillna(16.75,inplace=True)
data.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	NaN	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	
4	FDY38	16.750	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	

```
#checking outlet size details
print(data['Outlet_Size'].value_counts())
print(data['Outlet_Location_Type'].value_counts())
```

```
Medium    4655
Small     3980
High      1553
Name: Outlet_Size, dtype: int64
Tier 3    5583
Tier 2    4641
Tier 1    3980
Name: Outlet_Location_Type, dtype: int64
```

```
#checking relation between outlet size and location
#data.groupby('Outlet_Size')['Outlet_Location_Type'].value_counts()
pd.crosstab(index=data['Outlet_Size'],columns=data['Outlet_Location_Type'])
```

Outlet_Location_Type	Tier 1	Tier 2	Tier 3
Outlet_Size			
High	0	0	1553
Medium	1550	0	3105
Small	2430	1550	0

```
#replaceing null values with medium because, medium range outlets can gain more advantage in every tier areas
data['Outlet_Size'].fillna('Medium',inplace=True)
data
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locat
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	Medium	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	Medium	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	Medium	
4	FDY38	16.750	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	
...	...	...	...	...	...	...	...	...	...	
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	High	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	Medium	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	Small	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	Medium	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	OUT046	1997	Small	

<u>STEP-6:</u> Separating training data and testing datasets

```
#seperating test dataset
data_test=data[data['Item_Outlet_Sales'].isnull()]
data_test.drop('Item_Outlet_Sales',axis=1,inplace=True)
data_test
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locat
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	
1	FDW14	8.300	Regular	0.038428	Dairy	87.3198	OUT01[illegible]	2007	Medium	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	Medium	
4	FDY38	16.750	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	
5	FDH56	9.800	Regular	0.063817	Fruits and Vegetables	117.1492	OUT046	1997	Small	
...	...	...	...	...	...	...	...	...	...	...
5676	FDB58	10.500	Regular	0.013496	Snack Foods	141.3154	OUT046	1997	Small	
5677	FDD47	7.600	Regular	0.142991	Starchy Foods	169.1448	OUT018	2009	Medium	
5678	NCO17	10.000	Low Fat	0.073529	Health and Hygiene	118.7440	OUT045	2002	Medium	
5679	FDJ26	15.300	Regular	0.000000	Canned	214.6218	OUT017	2007	Medium	
5680	FDU37	9.500	Regular	0.104720	Canned	79.7960	OUT045	2002	Medium	

<u>STEP-7:</u> TRAINING DATASET

```
#extracting train dataset which will further be classified into train and test split
data_train=data[data['Item_Outlet_Sales'].notnull()]
data_train
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locat
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Medium	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	
...	...	...	...	...	...	...	...	...	...	...
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	High	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	Medium	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	Small	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	Medium	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	OUT046	1997	Small	

CHAPTER 6

RESULTS

6.1 Performing Exploratory Data Analysis

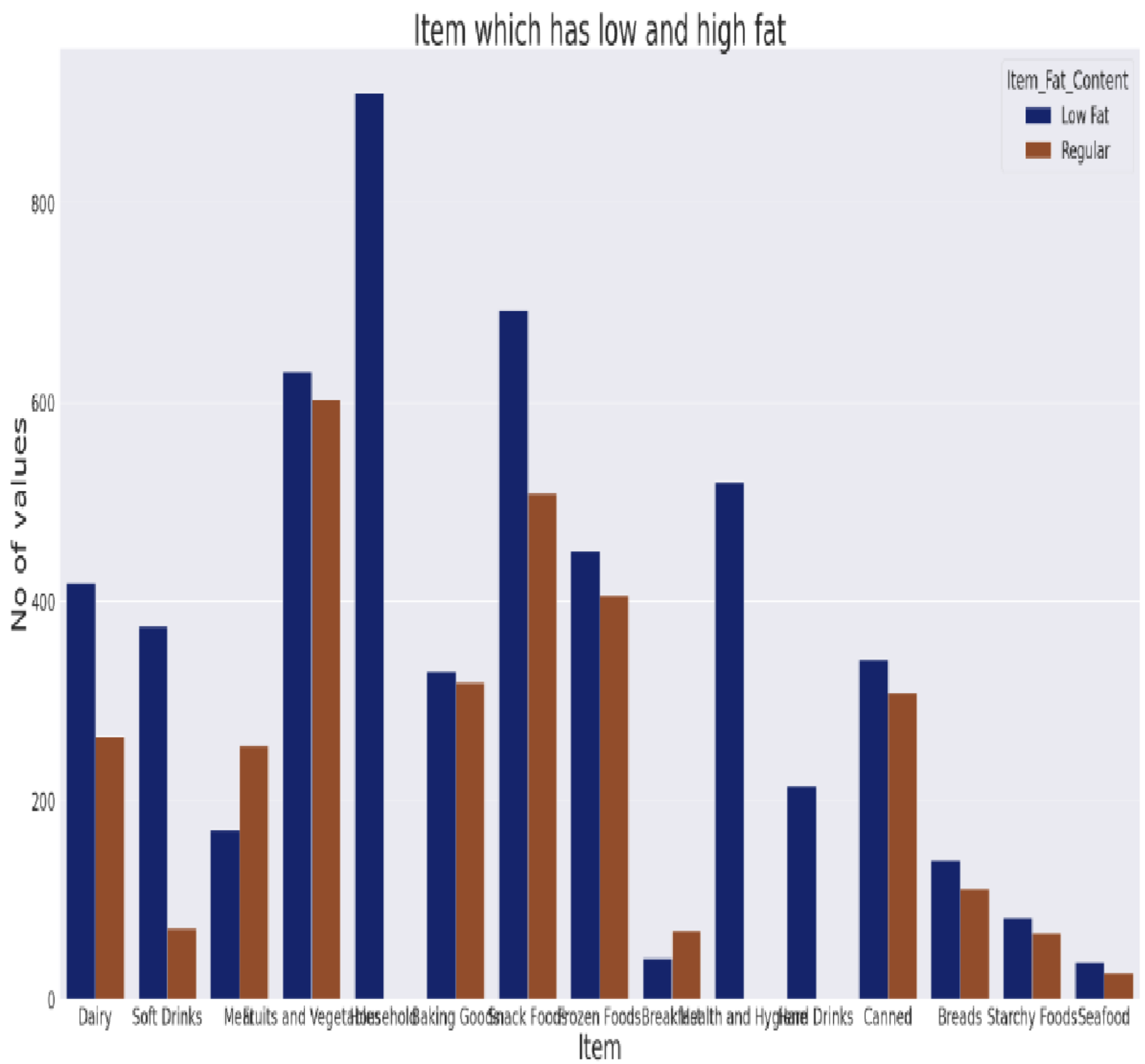
STEP-8:

Here the items containing low fats and high fats are classified into bar graphs

Using blue and brown colour.

```
#checking item type based on their fat content
plt.figure(figsize=(50,25))
#sns.set_context('poster', font_scale = 2)
sns.set_style('darkgrid')
sns.set(font_scale=3)
sns.countplot(x=data_train['Item_Type'],hue=data_train['Item_Fat_Content'],palette='dark')
plt.xlabel('Item',fontsize=50)
plt.ylabel('No of values',fontsize=50)
plt.title('Item which has low and high fat',fontsize=60)
```

```
Text(0.5, 1.0, 'Item which has low and high fat')
```



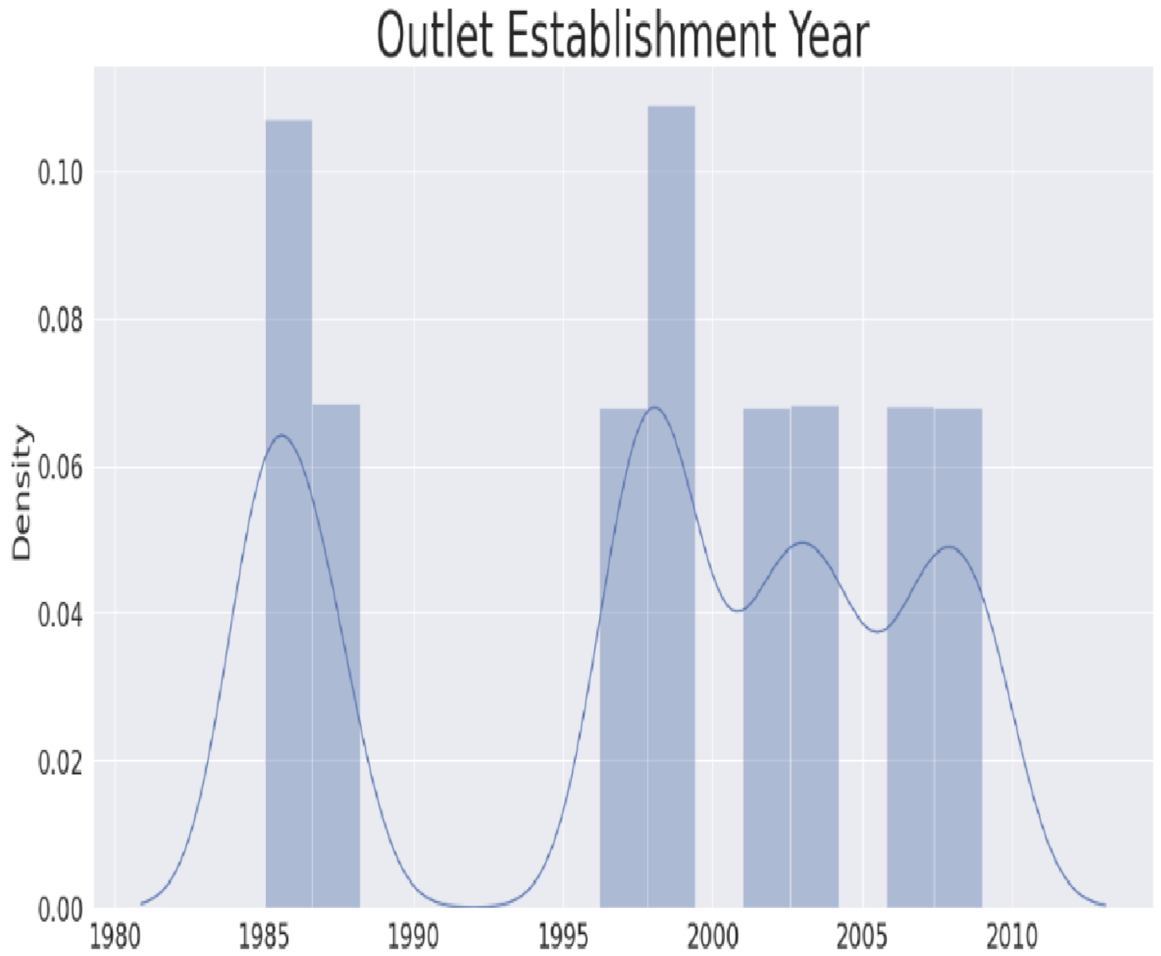
```
data1=pd.crosstab(data_train['Item_Type'],columns=data_train['Item_Fat_Content'])
data1
```

Item_Fat_Content	Low Fat	Regular
Item_Type		
Baking Goods	329	319
Breads	140	111
Breakfast	41	69
Canned	341	308
Dairy	418	264
Frozen Foods	450	406
Fruits and Vegetables	630	602
Hard Drinks	214	0
Health and Hygiene	520	0
Household	910	0
Meat	170	255
Seafood	37	27
Snack Foods	692	508
Soft Drinks	374	71
Starchy Foods	82	66

<u>Checking if the outlets of the yearly sales are in profit or loss. The sales of the overall are measured for ever five years. Using this graph ,We can easily identify about how that year sales ended.</u>

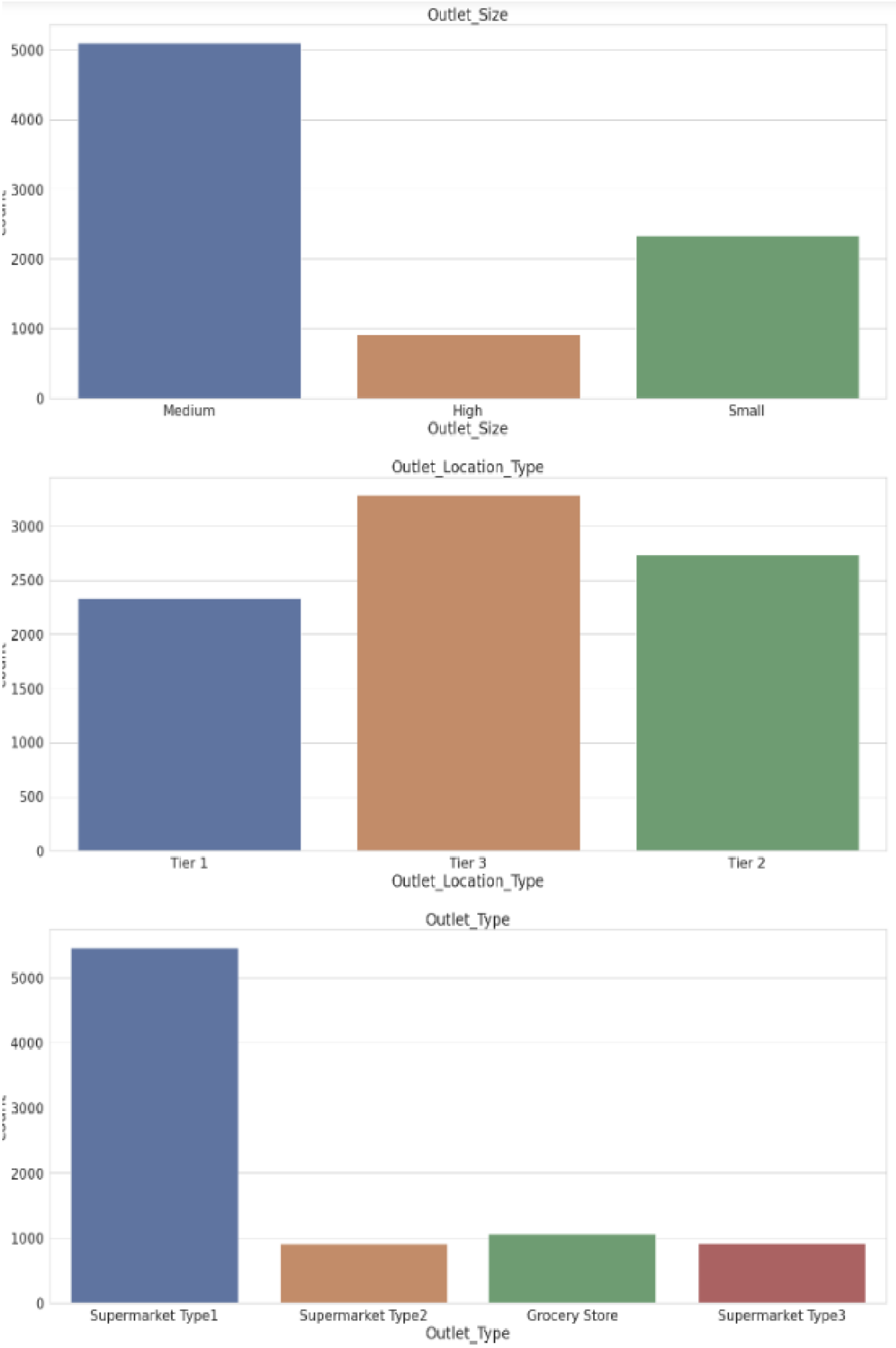
```
#checking if outlets are increasing or decreasing
plt.figure(figsize=(20,10))
sns.set_style('whitegrid')
sns.set(font_scale=2)
sns.distplot(x=data_train['Outlet_Establishment_Year'],kde=True,bins=15)
plt.title('Outlet Establishment Year',fontsize=40)
```

```
Text(0.5, 1.0, 'Outlet Establishment Year')
```



```
#checking outlets tier, size and type of goods they sell
def visual(x1,x2):
    plt.figure(figsize=(25,10))
    sns.set_style('whitegrid')
    sns.countplot(x=x1,palette='deep')
    plt.title(x2)
```

```
type=['Outlet_Size','Outlet_Location_Type','Outlet_Type']
for i in type:
    visual(data_train[i],i)
```


Outlet_Size
5000
4000
3000
2000
1000
0
Medium
High
Small
Outlet_Size
Outlet_Location_Type
3000
2500
2000
1500
1000
500
0
Tier 1
Tier 3
Tier 2
Outlet_Location_Type
Outlet_Type
5000
4000
3000
2000
1000
0
Supermarket Type1
Supermarket Type2
Grocery Store
Supermarket Type3
Outlet_Type

STEP-9: Finding MRP and outlet sales of small, medium and large outlet sizes:

```
#Finding item mrp and outlet sales of each outlet size
def out_size(x1,x2,x3):
    data_outlet_size=data_train[data_train['Outlet_Size']==x1]
    b=data_outlet_size[x2].sum()
    c=data_outlet_size[x3].sum()
    return b,c
```

```
type=['Small','Medium','High']
for i in type:
    a=out_size(i,'Item_MRP','Item_Outlet_Sales')
    print("The Total Item MRP and Item outlet sales of",i,"outlet size is",a)
```

```
The Total Item MRP and Item outlet sales of Small outlet size is (331440.9526, 4482101.6832)
The Total Item MRP and Item outlet sales of Medium outlet size is (718100.8468, 11676080.670200001)
The Total Item MRP and Item outlet sales of High outlet size is (129687.7898, 2107425.4474)
```

Creating data frame to compare outlet sizes:

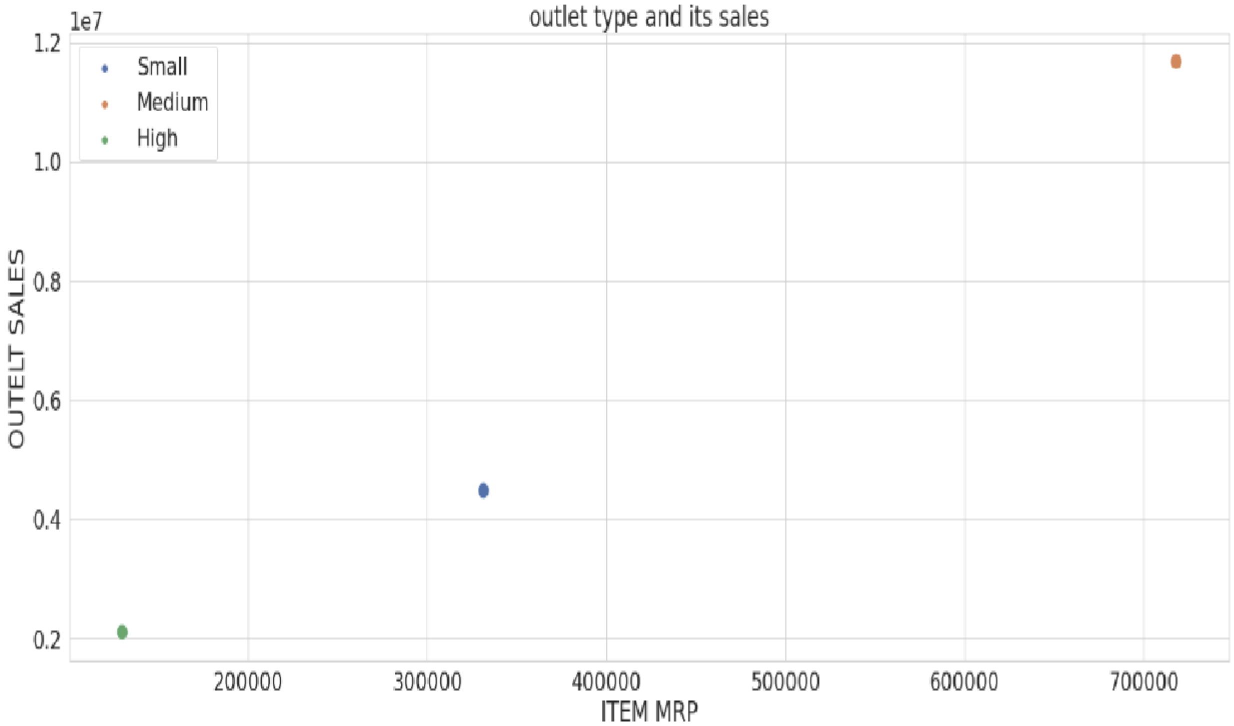
```
#Creating a small dataframe to compare outlet size sales
data_sales={"Outlet size":['Small','Medium','High'],"ITEM mrp":[331440.9526,718100.8468,129687.7898],"Outlet sales":[4482101.6832
data_outlet_size1=pd.DataFrame(data_sales,index=[0,1,2])
data_outlet_size1
```

	Outlet size	ITEM mrp	Outlet sales
0	Small	331440.9526	4.482102e+06
1	Medium	718100.8468	1.167608e+07
2	High	129687.7898	2.107425e+06

Creating visualization for data

```
#Visualizing the above data
plt.figure(figsize=(25,10))
sns.set_style('whitegrid')
sns.scatterplot(x=data_sales['ITEM mrp'],y=data_sales['Outlet sales'],hue=data_sales['Outlet size'],s=200)
plt.xlabel('ITEM MRP')
plt.ylabel('OUTELT SALES')
plt.title('outlet type and its sales')
```

```
Text(0.5, 1.0, 'outlet type and its sales')
```



STEP-10: Finding item MRP and outlet sales of each and every outlet location type and creating data frames to compare all the locations outlet sales

```
#Finding item mrp and outlet sales of each outlet location type
def out_loc_size(x1,x2,x3):
    data_location_type=data_train[data_train['Outlet_Location_Type']==x1]
    b=data_location_type[x2].sum()
    c=data_location_type[x3].sum()
    return b,c
```

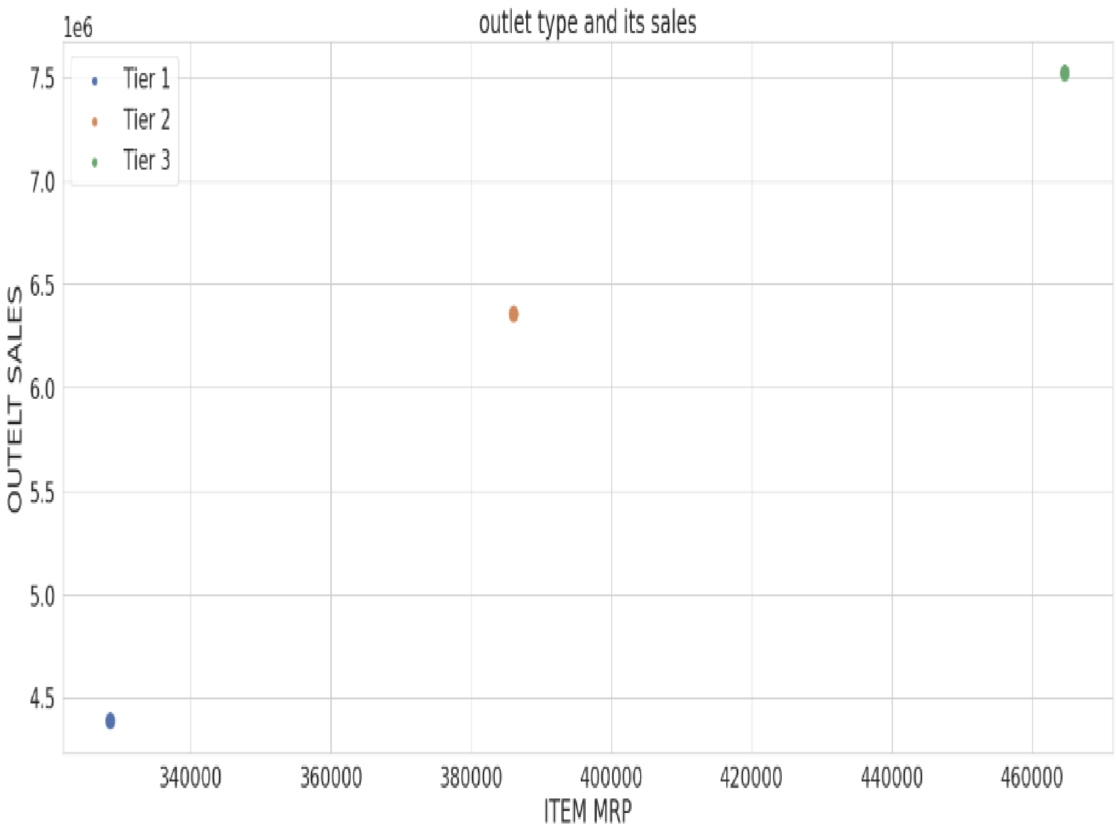
```
type=['Tier 1','Tier 2','Tier 3']
for i in type:
    a=out_loc_size(i,'Item_MRP','Item_Outlet_Sales')
    print("The Total Item MRP and Item outlet sales of",i,"outlet size is",a)
```

```
The Total Item MRP and Item outlet sales of Tier 1 outlet size is (328560.7724, 4388223.883199999)
The Total Item MRP and Item outlet sales of Tier 2 outlet size is (386056.6244, 6356712.184)
The Total Item MRP and Item outlet sales of Tier 3 outlet size is (464612.1924, 7520671.7336)
```

STEP -11: Creating Visualization

```
#Visualizing the above data
plt.figure(figsize=(25,10))
sns.set_style('whitegrid')
sns.scatterplot(x=data_outlet_location['ITEM mrp'],y=data_outlet_location['Outlet sales'],hue=data_outlet_location['Outlet size']
plt.xlabel('ITEM MRP')
plt.ylabel('OUTELT SALES')
plt.title('outlet type and its sales')
```

```
Text(0.5, 1.0, 'outlet type and its sales')
```



Finding items MRP and outlet sales of each outlet type, creating small data frames to compare each outlet sales and creating visualization for the data.

```
#finding item mrp and outlet sales for each outlet type
def out_type(x1,x2,x3):
    data_outlet_type=data_train[data_train['Outlet_Type']==x1]
    b=data_outlet_type[x2].sum()
    c=data_outlet_type[x3].sum()
    return b,c
```

```
type=['Supermarket Type1','Supermarket Type2','Grocery Store','Supermarket Type3']
for i in type:
    a=out_type(i,'Item_MRP','Item_Outlet_Sales')
    print("The Total Item MRP and Item outlet sales of",i,"outlet size is",a)
```

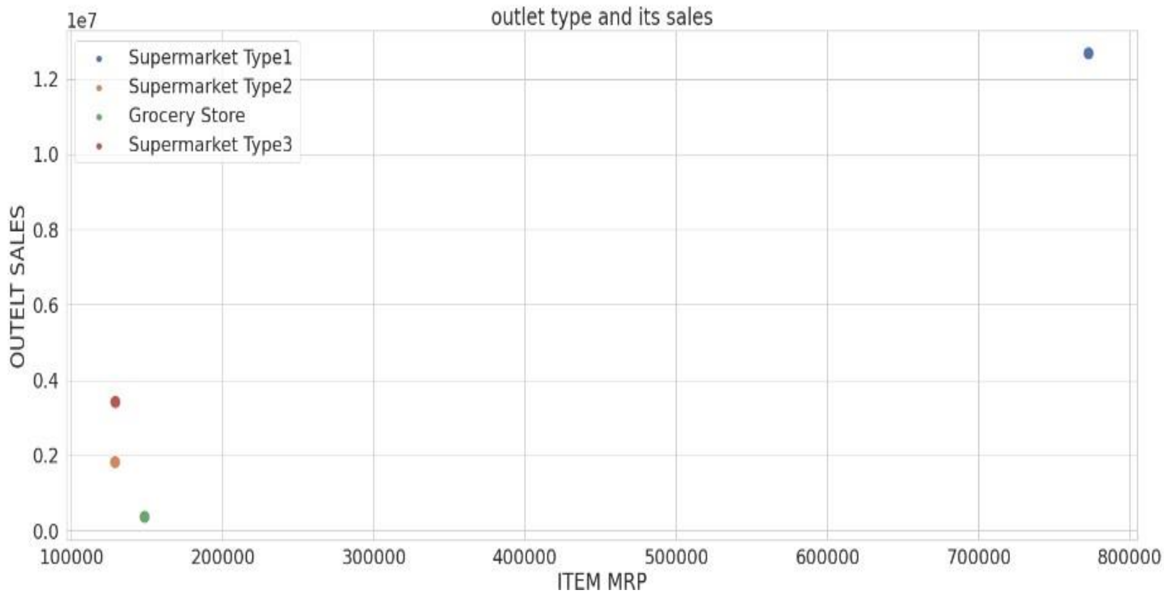
```
The Total Item MRP and Item outlet sales of Supermarket Type1 outlet size is (772723.213, 12677189.5346)
The Total Item MRP and Item outlet sales of Supermarket Type2 outlet size is (128921.5056, 1814750.4202)
The Total Item MRP and Item outlet sales of Grocery Store outlet size is (148471.8818, 360255.72459999996)
The Total Item MRP and Item outlet sales of Supermarket Type3 outlet size is (129112.98879999999, 3413412.1213999996)
```

```
#Creating a small dataframe to compare outlet size sales
data_sales={"Outlet size":['Supermarket Type1','Supermarket Type2','Grocery Store','Supermarket Type3'],"ITEM mrp":[772723.213,12
data_outlet_size1=pd.DataFrame(data_sales,index=[0,1,2,3])
data_outlet_size1
```

	Outlet size	ITEM mrp	Outlet sales
0	Supermarket Type1	772723.2130	1.267719e+07
1	Supermarket Type2	128921.5056	1.814750e+06
2	Grocery Store	148471.8818	3.602557e+05
3	Supermarket Type3	129112.9887	3.413412e+06

```
#Visualizing the above data
plt.figure(figsize=(25,10))
sns.set_style('whitegrid')
sns.scatterplot(x=data_sales['ITEM mrp'],y=data_sales['Outlet sales'],hue=data_sales['Outlet size'],s=200)
plt.xlabel('ITEM MRP')
plt.ylabel('OUTELT SALES')
plt.title('outlet type and its sales')
```

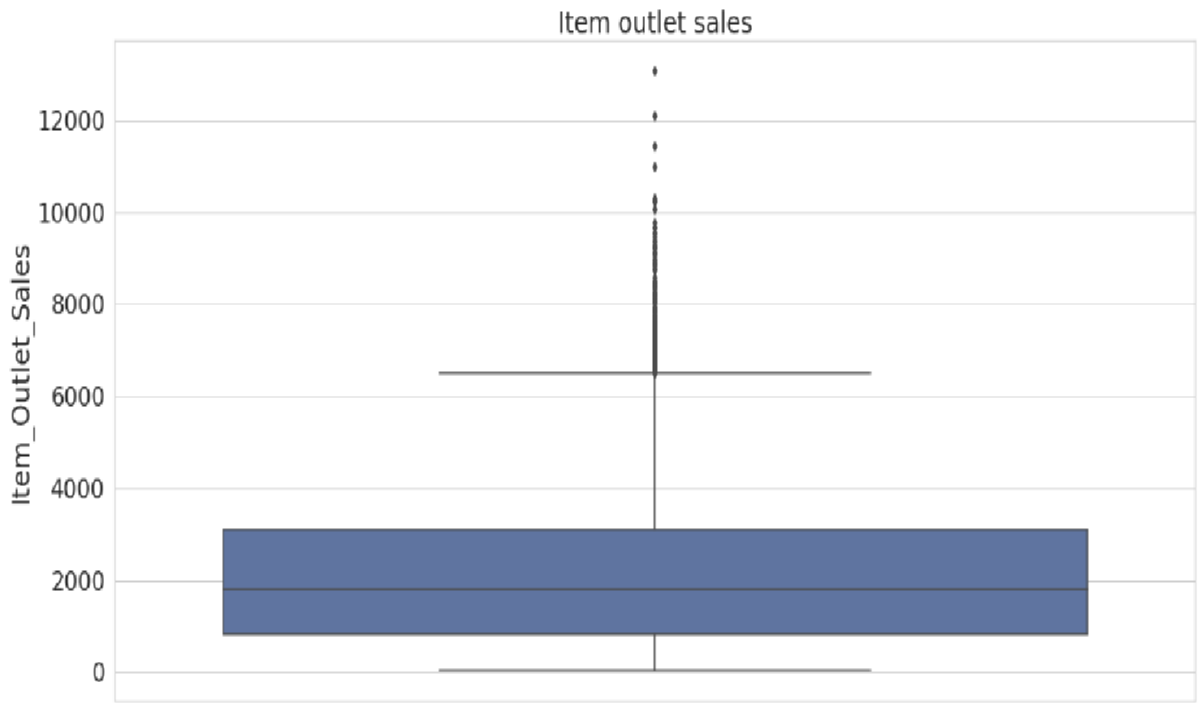
```
Text(0.5, 1.0, 'outlet type and its sales')
```



STEP-12: Creating statistics for item outliers

```
#Finding statistics for the item_outliers
plt.figure(figsize=(20,10))
sns.boxplot(y=data_train['Item_Outlet_Sales'])
plt.title('Item outlet sales')
```

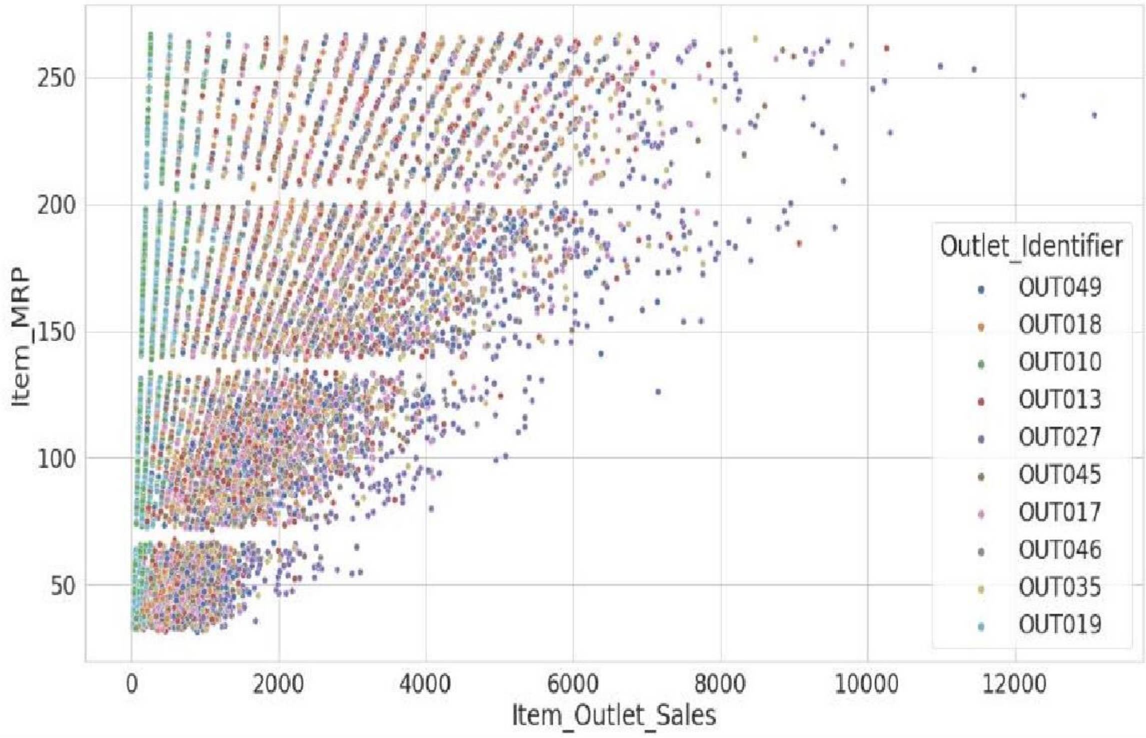
```
Text(0.5, 1.0, 'Item outlet sales')
```



7

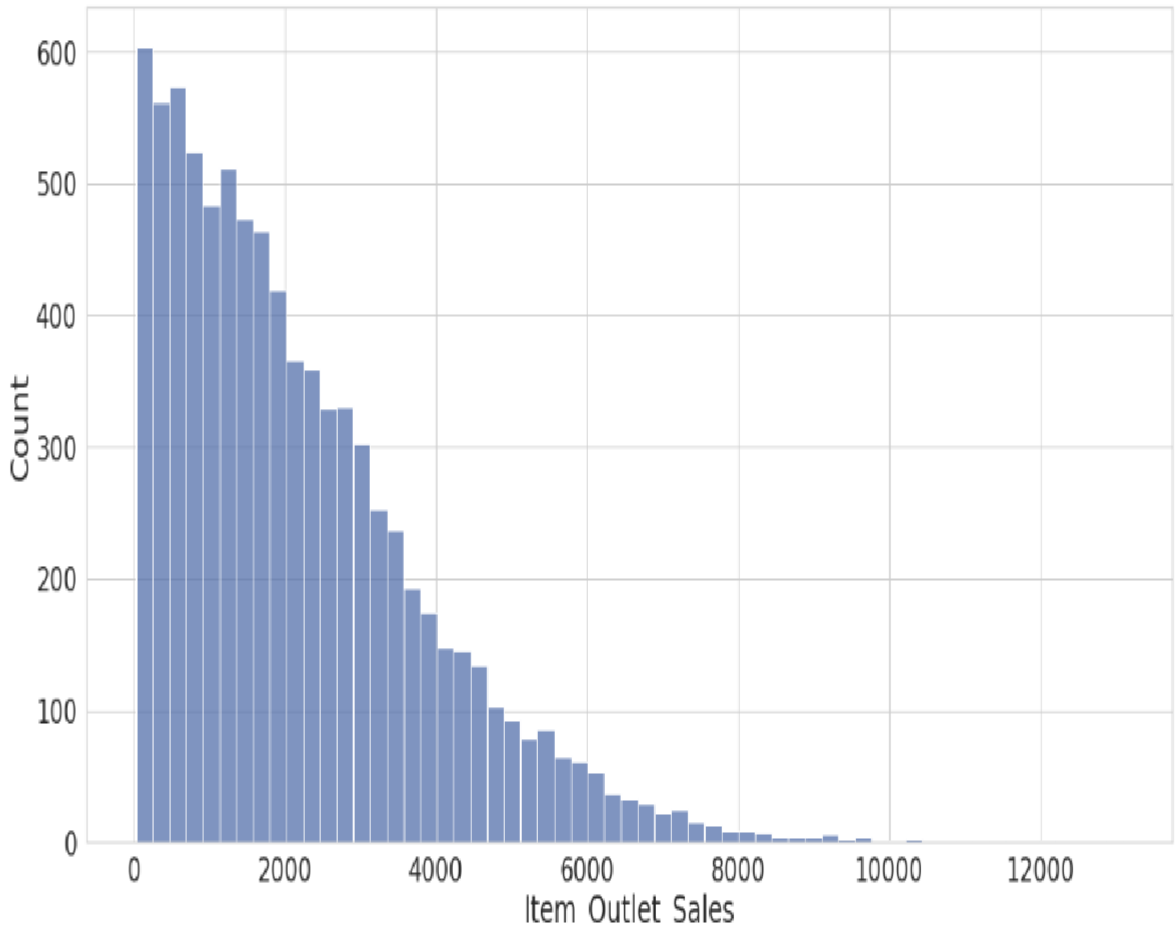
```
plt.figure(figsize=(20,10))
sns.scatterplot(y=data_train['Item_MRP'],x=data_train['Item_Outlet_Sales'],hue=data_train['Outlet_Identifier'])
```

<AxesSubplot:xlabel='Item_Outlet_Sales', ylabel='Item_MRP'>




```
plt.figure(figsize=(20,10))
sns.histplot(data_train['Item_Outlet_Sales'])
```

<AxesSubplot:xlabel='Item_Outlet_Sales', ylabel='Count'>



STEP-13:Finding outlet sales for each outlet

```
data_outlet_identifier=data_train['Outlet_Identifier'].unique()
data_outlet_identifier
```

```
#creating a function for finding sales for each outlet
def out_iden_sales(x1):
    data_out_sales=data_train[data_train['Outlet_Identifier']==x1]
    return data_out_sales['Item_Outlet_Sales'].sum()
```

```
#creating a dataframe by retiriving all the outlets present
data_outlet_identifier=pd.DataFrame()
data_outlet_identifier['Outlet']=data_train['Outlet_Identifier'].unique()
data_outlet_identifier
```

	Outlet
0	OUT049
1	OUT018
2	OUT010
3	OUT013
4	OUT027
5	OUT045
6	OUT017
7	OUT046
8	OUT035
9	OUT019

```
#a new dataframe created, where each outlet and their sales are represented
data_outlet_identifier = data_outlet_identifier.assign(Outlet_Sales=c)
data_outlet_identifier.sort_values('Outlet',inplace=True,ascending=True)
data_outlet_identifier
```

	Outlet	Outlet_Sales
2	OUT010	1.850837e+05
3	OUT013	2.107425e+06
6	OUT017	2.130077e+06
1	OUT018	1.814750e+06
9	OUT019	1.751720e+05
4	OUT027	3.413412e+06
8	OUT035	2.230075e+06
5	OUT045	1.996560e+06
7	OUT046	2.076855e+06
0	OUT049	2.136197e+06

```
#creating function for finding mrp sales for each outlet
def out_iden_mrp(x1):
    data_out_sales=data_train[data_train['Outlet_Identifier']==x1]
    return data_out_sales['Item_MRP'].sum()
```

```
#creating a dataframe by retiriving all the outlets present
data_outlet_mrp_price=pd.DataFrame()
data_outlet_mrp_price['Outlet']=data_train['Outlet_Identifier'].unique()
data_outlet_mrp_price
```

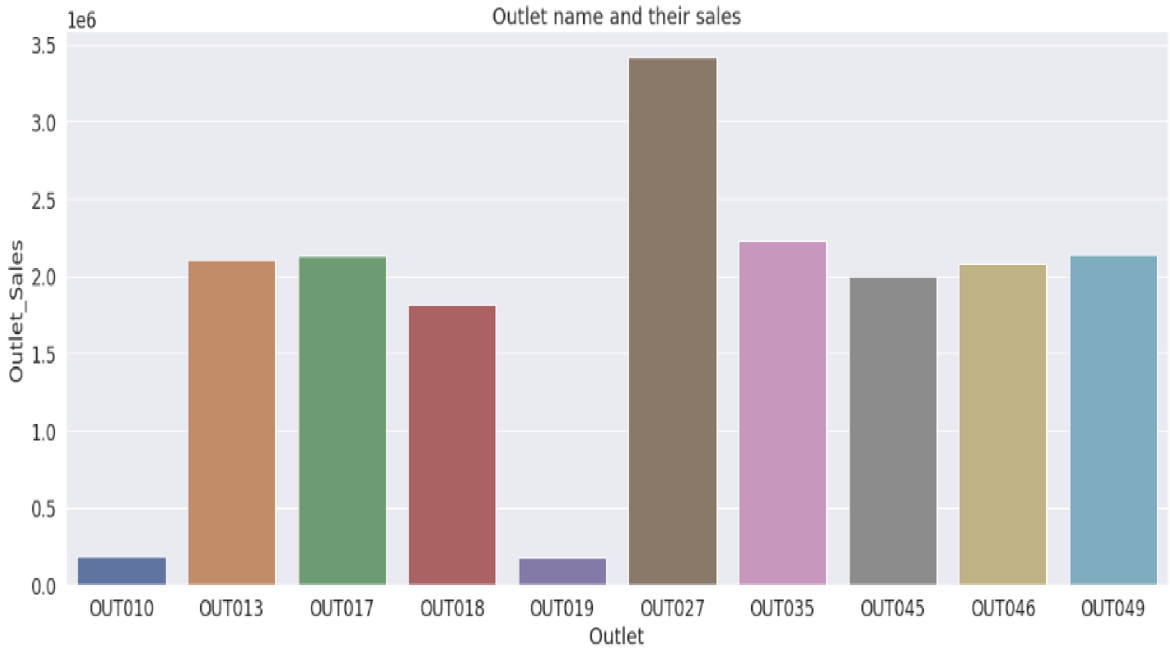
	Outlet
0	OUT049
1	OUT018
2	OUT010
3	OUT013
4	OUT027
5	OUT045
6	OUT017
7	OUT046
8	OUT035
9	OUT019

```
#a new column created, where each outlet and their item mrp are represented
data_outlet_mrp_price = data_outlet_mrp_price.assign(Outlet_Item_Mrp=c)
data_outlet_mrp_price.sort_values('Outlet',inplace=True,ascending=True)
data_outlet_mrp_price
```

	Outlet	Outlet_Item_Mrp
2	OUT010	76889.9082
3	OUT013	129687.7898
6	OUT017	126783.6672
1	OUT018	128921.5056
9	OUT019	71581.9736
4	OUT027	129112.9888
8	OUT035	130867.5112
5	OUT045	128405.4460
7	OUT046	128991.4678
0	OUT049	127987.3310

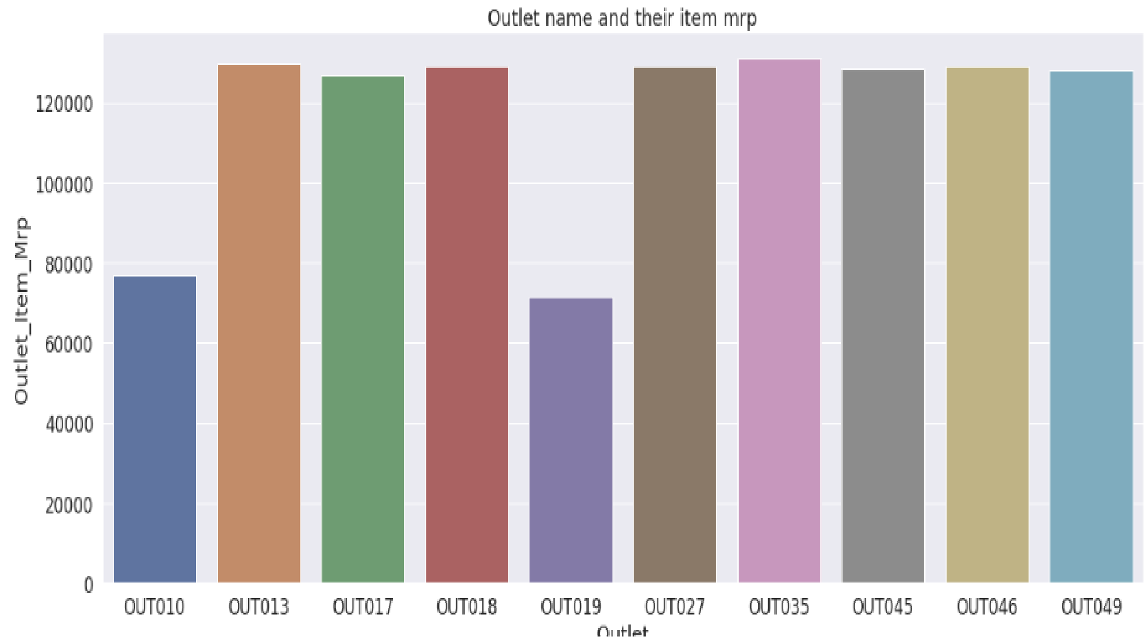
```
#Visualizing using bar graph determing each outlet and their sales for each outlet
plt.figure(figsize=(20,8))
sns.set(font_scale=1.5)
sns.barplot(x=data_outlet_identifier['Outlet'],y=data_outlet_identifier['Outlet_Sales'])
plt.title('Outlet name and their sales')
```

Text(0.5, 1.0, 'Outlet name and their sales')




```
#Visualizing a bar graph and determining their purchased item mrp
plt.figure(figsize=(20,8))
sns.set(font_scale=1.5)
sns.barplot(x=data_outlet_mrp_price['Outlet'],y=data_outlet_mrp_price['Outlet_Item_Mrp'])
plt.title('Outlet name and their item mrp')
```

Text(0.5, 1.0, 'Outlet name and their item mrp')



6.2 PREDICTING THE SALES

STEP-13: Gathering data from above analyzation

```
#Gethering Required data for prediction
data_train
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locat
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Medium	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	
...	...	...	...	...	...	...	...	...	...	
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	High	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	Medium	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	Small	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	Medium	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	OUT046	1997	Small	

```
#creating dummmy variables for each column of training dataset
data_dummies=pd.get_dummies(data_train)
data_dummies
```

	Item_Weight	Item_Visibility	Item_MRP	Item_Outlet_Sales	Outlet_Identifier_OUT010	Outlet_Identifier_OUT013	Outlet_Identifier_OUT017	Outlet_Identifier_OU
0	9.300	0.016047	249.8092	3735.1380	0	0	0	
1	5.920	0.019278	48.2692	443.4228	0	0	0	
2	17.500	0.016760	141.6180	2097.2700	0	0	0	
3	19.200	0.000000	182.0950	732.3800	1	0	0	
4	8.930	0.000000	53.8614	994.7052	0	1	0	
...	...	...	...	...	...	...	...	
8518	6.865	0.056783	214.5218	2778.3834	0	1	0	
8519	8.380	0.046982	108.1570	549.2850	0	0	0	
8520	10.600	0.035186	85.1224	1193.1136	0	0	0	
8521	7.210	0.145221	103.1332	1845.5976	0	0	0	
8522	14.800	0.044878	75.4670	765.6700	0	0	0	

8354 rows × 24 columns

```
#creating dummmy variables for each feture for test dataset
data_dummies_test=pd.get_dummies(data_test)
data_dummies_test
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Identifier_OUT010	Outlet_Identifier_OUT013	Outlet_Identifier_OUT017	Outlet_Identifier_OUT018	Outlet_Identi
0	20.750	0.007565	107.8622	0	0	0	0	
1	8.300	0.038428	87.3198	0	0	1	0	
3	7.315	0.015388	155.0340	0	0	1	0	
4	16.750	0.118599	234.2300	0	0	0	0	
5	9.800	0.063817	117.1492	0	0	0	0	
...	...	...	...	...	...	...	...	
5676	10.500	0.013496	141.3154	0	0	0	0	
5677	7.600	0.142991	169.1448	0	0	0	1	
5678	10.000	0.073529	118.7440	0	0	0	0	
5679	15.300	0.000000	214.6218	0	0	1	0	

Checking for co relation in both the feature datasets

```
#checking co-relation between two features
data_dummies.corr()
#there are only four numerical features, converting ategorical varibales into numerical variables for better accuracy
```

	Item_Weight	Item_Visibility	Item_MRP	Item_Outlet_Sales	Outlet_Identifier_OUT010	Outlet_Identifier_OUT013	Outlet_Identifier_OUT017
Item_Weight	1.000000	0.018258	0.022737	0.041204	-0.037942	-0.040673	-0.053992
Item_Visibility	0.018258	1.000000	-0.003758	-0.130900	0.180263	-0.041564	-0.033198
Item_MRP	0.022737	-0.003758	1.000000	0.567495	-0.000318	0.002382	-0.010291
Item_Outlet_Sales	0.041204	-0.130900	0.567495	1.000000	-0.285126	0.023427	0.031526
Outlet_Identifier_OUT010	-0.037942	0.180263	-0.000318	-0.285126	1.000000	-0.092709	-0.092367
Outlet_Identifier_OUT013	-0.040673	-0.041564	0.002382	0.023427	-0.092709	1.000000	-0.122698
Outlet_Identifier_OUT017	-0.053992	-0.033198	-0.010291	0.031526	-0.092367	-0.122698	1.000000
Outlet_Identifier_OUT018	-0.049780	-0.033792	0.004631	-0.038330	-0.092253	-0.122547	-0.122095
Outlet_Identifier_OUT019	0.184648	0.213253	-0.004403	-0.275012	-0.067433	-0.089575	-0.089246
Outlet_Identifier_OUT027	0.254482	-0.053358	-0.004612	0.313263	-0.092936	-0.123453	-0.122999
Outlet_Identifier_OUT035	-0.054571	-0.031518	0.011379	0.051915	-0.092595	-0.123000	-0.122547
Outlet_Identifier_OUT045	-0.068882	-0.038648	0.000575	0.002039	-0.092310	-0.122622	-0.122171
Outlet_Identifier_OUT046	-0.050724	-0.038283	0.005062	0.020578	-0.092253	-0.122547	-0.122095
Outlet_Identifier_OUT049	-0.048872	-0.036149	-0.005474	0.031382	-0.092538	-0.122925	-0.122472
Outlet_Size_High	-0.040673	-0.041564	0.002382	0.023427	-0.092709	1.000000	-0.122698
Outlet_Size_Medium	0.002629	-0.033564	-0.009880	0.073792	0.210755	-0.439889	0.278930
Outlet_Size_Small	0.025469	0.065419	0.009077	-0.096499	-0.164446	-0.218445	-0.217641
Outlet_Location_Type_Tier 1	0.029455	0.062217	-0.002648	-0.110811	-0.164397	-0.218380	-0.217576
Outlet_Location_Type_Tier 2	-0.117879	-0.068666	0.001118	0.056828	-0.184210	-0.244699	0.501423
Outlet_Location_Type_Tier 3	0.086155	0.008816	0.001357	0.047169	0.327837	0.435489	-0.281747
Outlet_Type_Grocery Store	0.104960	0.287742	-0.003412	-0.410206	0.694497	-0.133490	-0.132999
Outlet_Type_Supermarket Type1	-0.208333	-0.143861	0.002390	0.105540	-0.363828	0.254814	0.253876
Outlet_Type_Supermarket Type2	-0.049780	-0.033792	0.004631	-0.038330	-0.092253	-0.122547	-0.122095
Outlet_Type_Supermarket Type3	0.254482	-0.053358	-0.004612	0.313263	-0.092936	-0.123453	-0.122999

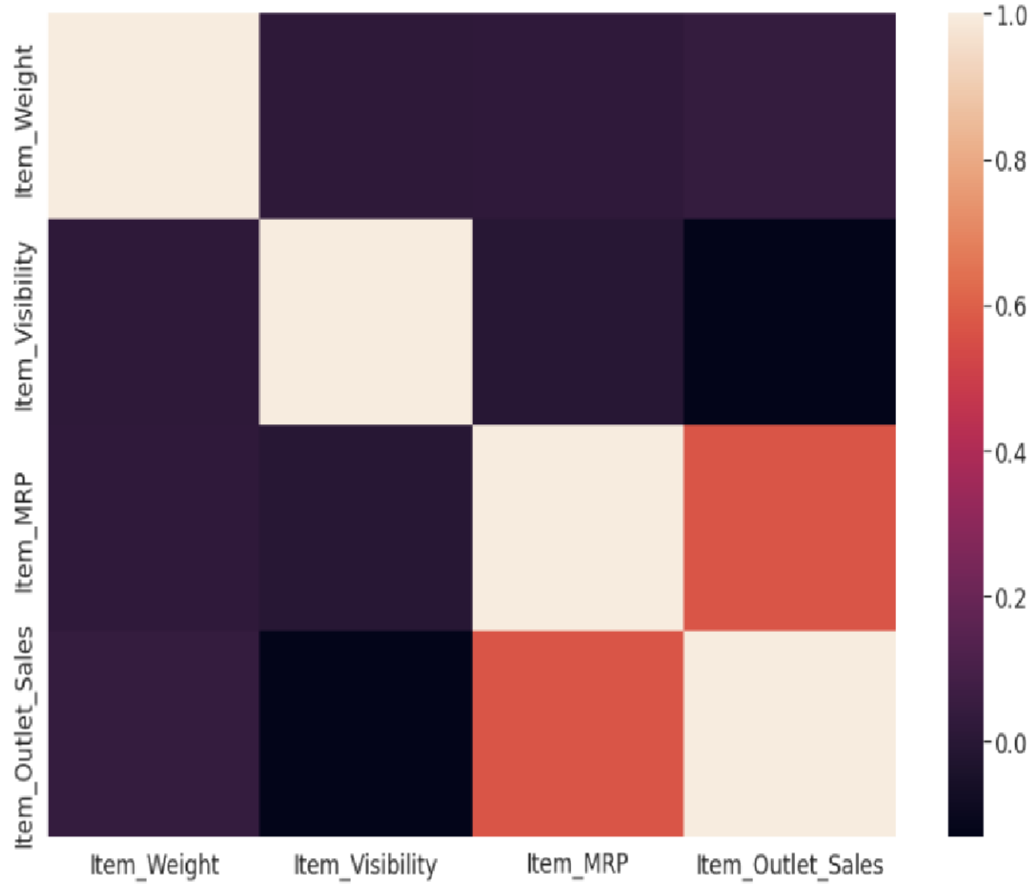
```
data_train.corr()
```

	Item_Weight	Item_Visibility	Item_MRP	Item_Outlet_Sales
Item_Weight	1.000000	0.018258	0.022737	0.041204
Item_Visibility	0.018258	1.000000	-0.003758	-0.130900
Item_MRP	0.022737	-0.003758	1.000000	0.567495
Item_Outlet_Sales	0.041204	-0.130900	0.567495	1.000000

STEP-14: Creating visualizations for co-relations from both the datasets

```
#visualizing co-relations for better understandings
plt.figure(figsize=(15,10))
sns.heatmap(data_train.corr())
```

<AxesSubplot:>



STEP-15: Separating train and validation dataset for prediction

```
#seperating train and validation dataset for prediction
y=pd.DataFrame()
y['Item_Outlet_Sales']=data_dummies['Item_Outlet_Sales']
y['Item_Outlet_Sales']=np.log(y['Item_Outlet_Sales'])
y
```

	Item_Outlet_Sales
0	8.225540
1	6.094524
2	7.648392
3	6.596300
4	6.902446
...	...
8518	7.929625
8519	6.308617
8520	7.084322
8521	7.520558
8522	6.640751

8354 rows × 1 columns

```
x=data_dummies
x.drop('Item_Outlet_Sales',axis=1,inplace=True)
x['Item_Weight']=np.log(x['Item_Weight'])
#x['Item_Visibility']=np.log(x['Item_Visibility'])
x['Item_MRP']=np.log(x['Item_MRP'])
x
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Identifier_OUT010	Outlet_Identifier_OUT013	Outlet_Identifier_OUT017	Outlet_Identifier_OUT018	Outlet_Identi
0	2.230014	0.016047	5.520697	0	0	0	0	
1	1.778336	0.019278	3.876794	0	0	0	1	
2	2.862201	0.016760	4.953133	0	0	0	0	
3	2.954910	0.000000	5.204529	1	0	0	0	
4	2.189416	0.000000	3.986414	0	1	0	0	
...	...	...	...	...	...	...	...	
8518	1.926436	0.056783	5.368411	0	1	0	0	
8519	2.125848	0.046982	4.683584	0	0	0	0	
8520	2.360854	0.035186	4.444090	0	0	0	0	
8521	1.975469	0.145221	4.636021	0	0	0	1	
8522	2.694627	0.044878	4.323695	0	0	0	0	

8354 rows × 23 columns

STEP-16: importing all the necessary packages for prediction:sklearn.linear_model, sklearn.model_selection, sklearn.metrics Import LinearRegression, train_test_split, r2 score, mean_squared_error

```
#importing necesarry packages for prediction
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error
```

```
#dividing the dataset into train and valid dataset
train_x,valid_x,train_y,valid_y=train_test_split(x,y,test_size=0.2,random_state=42)
print(train_x.shape,valid_x.shape,train_y.shape,valid_y.shape)
```

```
(6683, 23) (1671, 23) (6683, 1) (1671, 1)
```

STEP-17: prediction using linear regression

```
#prediction using linear regression
lgr=LinearRegression(fit_intercept=True)
data_fit=lgr.fit(train_x,train_y)
data_predict=data_fit.predict(valid_x)
data_predict
```

```
print(lgr.score(train_x,train_y),lgr.score(valid_x,valid_y))
data_test_predict=data_fit.predict(data_dummies_test)
data_test_predict
```

```
0.7394126317716299 0.7467181757939096
```

```
#cal rmse value
rmse=np.sqrt(mean_squared_error(valid_y,data_predict))
rmse
```

0.5203778301834706

```
r1_line_test=data_fit.score(valid_x,valid_y)
r1_line_train=data_fit.score(train_x,train_y)
print(r1_line_test,r1_line_train)
```

0.7467181757939096 0.7394126317716299

STEP-18:Applying Ridge Regression

```
from sklearn.linear_model import Ridge,RidgeCV,Lasso,LassoCV
#Increasing the accruacy using ridge regression
ridreg=Ridge(alpha=20)
ridreg.fit(train_x,train_y)
#Checking the performance score
rid_train_score=ridreg.score(train_x,train_y)
rid_test_score=ridreg.score(valid_x,valid_y)
print(rid_train_score,rid_test_score)
#There no improvement of accuracy using ridge regression
```

0.7392901792610166 0.7461915350305004

6.3 RESULTS VERIFICATION

6.3.1 Prediction using Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
```

```
rf=RandomForestRegressor(n_estimators=100,max_depth=100,min_samples_leaf=4,max_features='auto',min_samples_split=12,random_state=
rf_model=rf.fit(train_x,train_y)
datas_predict=rf.predict(valid_x)
print(datas_predict)
print(rf.score(train_x,train_y),rf.score(valid_x,valid_y))
```

```
[6.42515297 6.79878135 6.54378853 ... 8.79672477 7.62110645 7.27868272]
0.8516788177216642 0.7300217911780228
```

```
data_predicted=pd.DataFrame()
data_predicted['Actual']=valid_y
data_predicted['Predicted']=datas_predict
data_predicted
```

	Actual	Predicted
3604	7.284891	6.425153
8406	6.907121	6.798781
4675	6.563025	6.543789
2853	5.336237	6.390895
5484	7.071969	7.118768
...	...	...
762	7.142843	7.374468
5502	8.269139	8.021465
7829	8.572273	8.796725
814	7.654090	7.621106
88	6.494971	7.278683

1671 rows × 2 columns

```
data_test_predict=rf.predict(data_dummies_test)
data_test_predicted=pd.DataFrame()
data_test_predicted['Tested Data']=data_test_predict
data_test_predicted
```

	Tested Data
0	8.288508
1	8.297053
2	8.317836
3	8.556229
4	8.314162
...	...
5565	8.251154
5566	8.294221
5567	8.152707
5568	8.259522
5569	8.138782

5570 rows × 1 columns

```
rf1_mse=mean_squared_error(valid_y,datas_predict)
rf1_rmse=np.sqrt(rf1_mse)
print(rf1_rmse)
```

```
0.5372558202407977
```

```
# calculating r squared value
rf1_line_test=rf_model.score(valid_x,valid_y)
rf1_line_train=rf_model.score(train_x,train_y)
print(rf1_line_test,rf1_line_train)
```

```
0.7300217911780228 0.8516788177216642
```

6.3.2 Prediction using Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor
dr=DecisionTreeRegressor(min_samples_leaf=10,max_features='auto',max_depth=100,random_state=42)
dr_model=dr.fit(train_x,train_y)
dr_predicted=dr_model.predict(valid_x)
dr_predicted
```

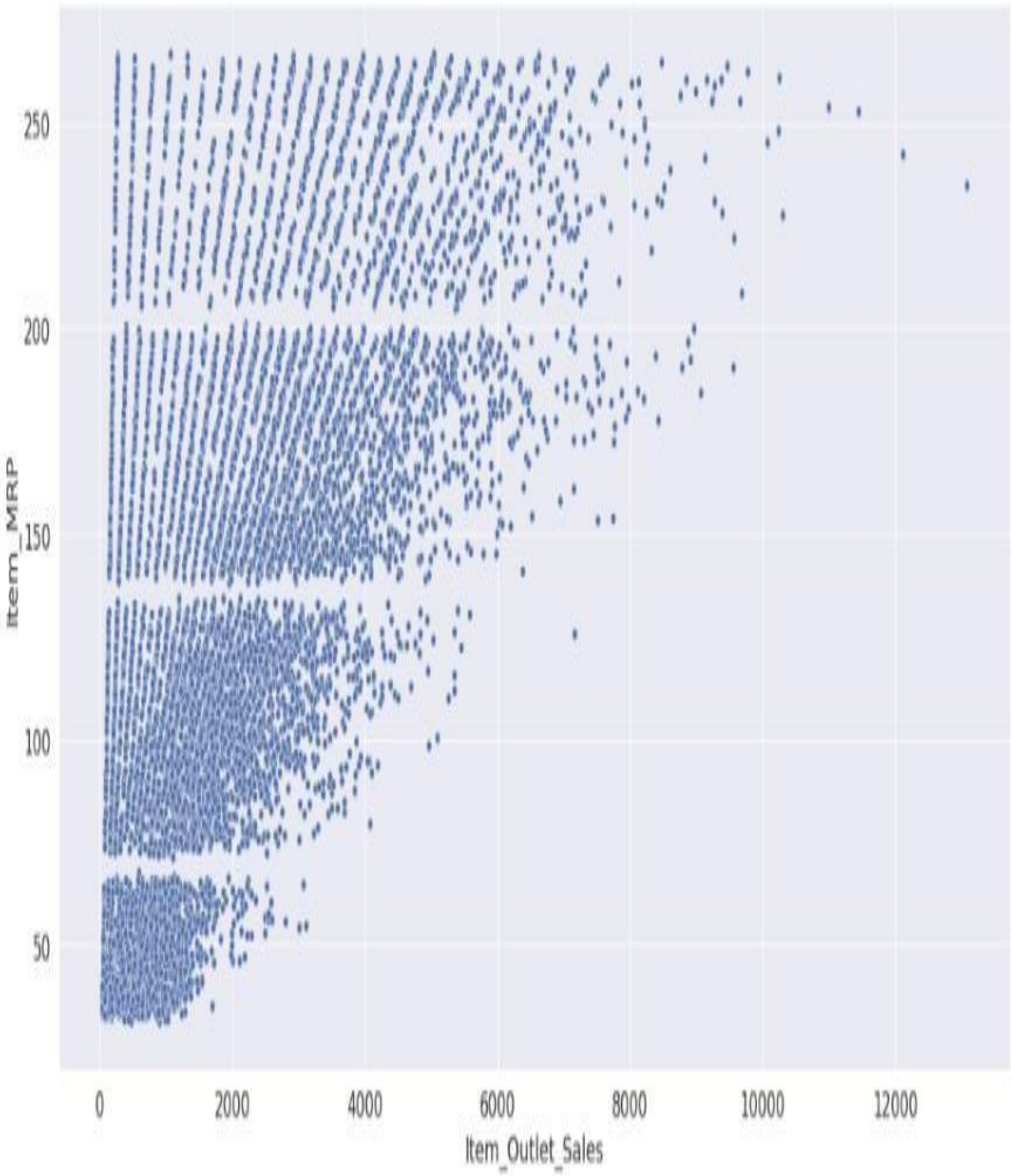
```
print(dr.score(train_x,train_y),dr.score(valid_x,valid_y))
rf1_mse=mean_squared_error(valid_y,dr_predicted)
rf1_rmse=np.sqrt(rf1_mse)
print(rf1_rmse)
# calculating r squared value
dr1_line_test=dr_model.score(valid_x,valid_y)
dr1_line_train=dr_model.score(train_x,train_y)
print(dr1_line_test,dr1_line_train)
```

```
0.8059222982636214 0.6927246629905192
0.5731662701790979
0.6927246629905192 0.8059222982636214
```

6.3.3 Performing prediction Clustering using K-means algorithm

```
plt.figure(figsize=(20,10))
sns.scatterplot(data_cluster['Item_Outlet_Sales'],data_cluster['Item_MRP'])
```

```
<AxesSubplot:xlabel='Item_Outlet_Sales', ylabel='Item_MRP'>
```



```python
# clsutering only required data for easy understanding
data_cluster=pd.DataFrame()
data_cluster['Item_Outlet_Sales']=data_train['Item_Outlet_Sales']
data_cluster['Item_MRP']=data_train['Item_MRP']
data_cluster
```

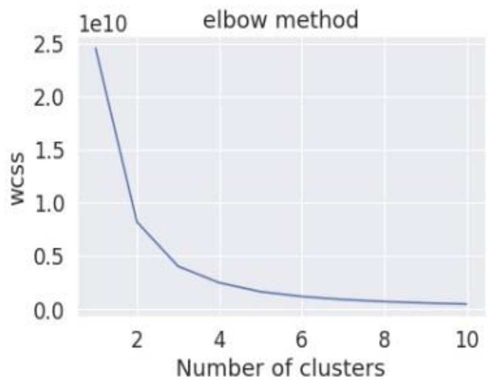
	Item_Outlet_Sales	Item_MRP
0	3735.1380	249.8092
1	443.4228	48.2692
2	2097.2700	141.6180
3	732.3800	182.0950
4	994.7052	53.8614
...	...	...
8518	2778.3834	214.5218
8519	549.2850	108.1570
8520	1193.1136	85.1224
8521	1845.5976	103.1332
8522	765.6700	75.4670

8354 rows × 2 columns

```python
# Finding The clusters by elbow method
wcss=[]
for i in range(1, 11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(data_cluster)
    wcss.append(kmeans.inertia_)

print(wcss)
plt.plot(range(1, 11), wcss)
plt.title('elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('wcss')
plt.show()
#No of clusters can considered as 2 or 4
```

```
[24494062598.58085, 8192600933.899679, 4021276156.9185743, 2467469059.5932336, 1627818078.629932, 1178407784.1802788, 88650972
7.9930694, 699868334.4311244, 562312323.6835834, 457144442.5747323]
```



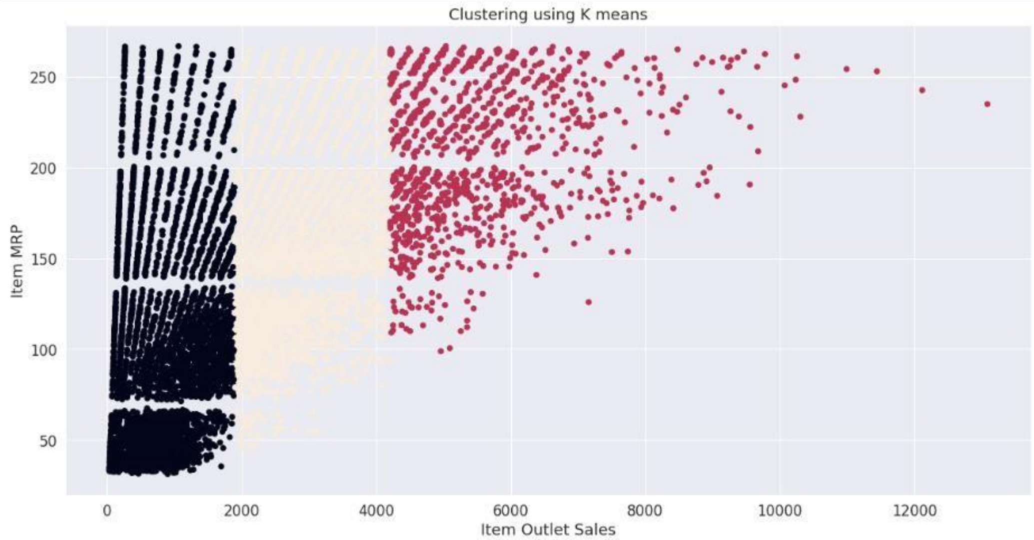
```python
kmeans = KMeans(3)
kmeans.fit(data_cluster)
identified_clusters = kmeans.fit_predict(data_cluster)
data_cluster['clusters']=identified_clusters
data_cluster
```

	Item_Outlet_Sales	Item_MRP	clusters
0	3735.1380	249.8092	2
1	443.4228	48.2692	0
2	2097.2700	141.6180	2
3	732.3800	182.0950	0
4	994.7052	53.8614	0
...	...	...	...
8518	2778.3834	214.5218	2
8519	549.2850	108.1570	0
8520	1193.1136	85.1224	0
8521	1845.5976	103.1332	0
8522	765.6700	75.4670	0

8354 rows × 3 columns

```python
plt.figure(figsize=(20,10))
sns.set_palette("bright")
sns.set_style('darkgrid')
plt.scatter(data_cluster['Item_Outlet_Sales'],data_cluster['Item_MRP'], c=identified_clusters)
plt.xlabel('Item Outlet Sales')
plt.ylabel('Item MRP')
plt.title('Clustering using K means')
#centers = kmeans.cluster_centers_
#plt.scatter(centers[:, 0], centers[:, 1], c='blue', s=100, alpha=0.9);
plt.show()
```


```python
plt.figure(figsize=(20,10))
sns.set_palette("bright")
sns.set_style('darkgrid')
plt.scatter(data_cluster['Item_Outlet_Sales'],data_cluster['Item_MRP'], c=identified_clusters)
plt.xlabel('Item Outlet Sales')
plt.ylabel('Item MRP')
plt.title('Clustering using K means')
#centers = kmeans.cluster_centers_
#plt.scatter(centers[:, 0], centers[:, 1], c='blue', s=100, alpha=0.9);
plt.show()
```

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

Most of the shopping malls / shopping centers plan to attract the customers to the store and make profit to the maximum extent by them. Once the customers enter the store then they are attracted then definitely they shop more by the special offers and obtain the desired items which are available in the favorable cost and satisfy them.

If the products as per the needs of the customers are provided then it can make maximum profit the retailers can also make the changes in the operations, objectives of the store that cause loss and efficient methods can be applied to gain more profit and sales by observing the history of data the existing stores a clear idea of sales can be known like seasonality trend and randomness.

From the above project, we have predicted the item outlets sales using different algorithms like Decision Tree Regression, K-means clustering, Random Forest Regression and displayed outputs in form of tables, small data frames and various plots of data visualization.

7.2 FUTURE SCOPE

To increase the originality and success of this sales prediction, many instances parameters and other elements can be used. The project can be further expanded in a web-based application utilising flask. Accuracy, which plays a vital part in prediction-based systems, can be considerably boosted as the number of parameters employed is raised. so that we can accurately forecast the sales of their outlets based on the most recent market data. Future scope can be expanded so that anyone can easily add the necessary information and calculate their outlet sales using our model. Our model is performing well, with an accuracy rate of about 80%. Tuning the parameters can help to boost this even more.

7.3 REFERENCES

- https://www.geeksforgeeks.org/random-forest-regression-in-python/
- https://seaborn.pydata.org/tutorial.html
- https://pandas.pydata.org/docs/
- https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
- https://stackoverflow.com/
- https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/
- https://www.geeksforgeeks.org/decision-tree/
- https://towardsdatascience.com/5-effective-ways-to-improve-the-accuracy-of-your-machine-learning-models-f1ea1f2b5d65
- https://www.analyticsvidhya.com/blog/2021/04/intuition-behind-correlation-definition-and-its-types/