Sales_Store_final

August 6, 2023

1 1. Problem Statement

- Nowadays, shopping malls and Big Marts keep track of individual item sales data in order to forecast future client demand and adjust inventory management. In a datawarehouse, these data stores hold a significant amount of consumer information and particular item details. By mining the data store from the data warehouse, more anomalies and common patterns can be discovered.

2 2. Approach:

- The classical machine learning tasks like Data Exploration, Data Cleaning, Feature Engineering, Model Building and Model Testing.This model is built on the basis of the Linear Regression.

3 3. Objective

- Data inspection and EDA tasks suitable for this dataset - data cleaning, univariate analysis, bivariate analysis etc.
- Outlier Analysis: You must perform the Outlier Analysis on the dataset. - - However, you do have the flexibility of not removing the outliers if it suits the business needs or a lot of countries are getting removed. Hence, all you need to do is find the outliers in the dataset, and then choose whether to keep them or remove them depending on the results you get.
- Create model using both K-means and Hierarchical clustering(both single and complete linkage) on this dataset to create the clusters.
- Analyse the clusters and identify the ones which are in dire need of aid. You can analyse the clusters by comparing how these three variables - [gdpp, child_mort and income] vary for each cluster of countries to recognise and differentiate the clusters of developed countries from the clusters of under-developed countries.
- Perform visualisations on the clusters that have been formed using the features selected for building the clustering model

4 4. Data Dictionary

- Item_Identifier: Unique product ID
- Item_Weight: Weight of product
- Item_Fat_Content: Whether the product is low fat or not
- Item_Visibility: The % of total display area of all products in a store allocated to the

- particular product
- Item_Type: The category to which the product belongs
- Item_MRP: Maximum Retail Price (list price) of the product
- Outlet_Identifier: Unique store ID
- Outlet_Establishment_Year: The year in which store was established
- Outlet_Size: The size of the store in terms of ground area covered
- Outlet_Location_Type: The type of city in which the store is located
- Outlet_Type: Whether the outlet is just a grocery store or some sort of supermarket
- Item_Outlet_Sales: Sales of the product in the particulat store. This is the outcome variable to be predicted.

[147]:
```
#Data Analysis & Data wrangling
import numpy as np
import pandas as pd
import missingno as mn
from random import sample
from numpy.random import uniform
from math import isnan

#Visualization
import matplotlib.pyplot as plt
import matplotlib.style as style
import seaborn as sns
%matplotlib inline

#Plotly Libraris
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly import tools
from plotly.colors import n_colors
from plotly.subplots import make_subplots
from plotly.offline import init_notebook_mode, iplot
from plotly import tools
from IPython.display import display, HTML
init_notebook_mode(connected=True)

import warnings
warnings.filterwarnings('ignore')
```

[148]:
```
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.expand_frame_repr', False)
```

[149]:
```
df=pd.read_csv('Train.csv')
```

5 5. Data Understanding

5.0.1 5.1 Data Reading

[150]:
```
df.head()
```

[150]:
```
  Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility
Item_Type  Item_MRP Outlet_Identifier  Outlet_Establishment_Year Outlet_Size
Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales
0           FDA15         9.30          Low Fat         0.016047
Dairy  249.8092            OUT049                        1999      Medium
Tier 1  Supermarket Type1          3735.1380
1           DRC01         5.92          Regular         0.019278            Soft
Drinks   48.2692            OUT018                        2009      Medium
Tier 3  Supermarket Type2           443.4228
2           FDN15        17.50          Low Fat         0.016760
Meat  141.6180            OUT049                        1999      Medium
Tier 1  Supermarket Type1          2097.2700
3           FDX07        19.20          Regular         0.000000  Fruits and
Vegetables  182.0950            OUT010                        1998         NaN
Tier 3      Grocery Store           732.3800
4           NCD19         8.93          Low Fat         0.000000
Household   53.8614            OUT013                        1987        High
Tier 3  Supermarket Type1           994.7052
```

[151]:
```
df.tail()
```

[151]:
```
     Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility
Item_Type  Item_MRP Outlet_Identifier  Outlet_Establishment_Year Outlet_Size
Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales
8518           FDF22        6.865          Low Fat         0.056783
Snack Foods  214.5218            OUT013                        1987        High
Tier 3  Supermarket Type1          2778.3834
8519           FDS36        8.380          Regular         0.046982
Baking Goods  108.1570            OUT045                        2002         NaN
Tier 2  Supermarket Type1           549.2850
8520           NCJ29       10.600          Low Fat         0.035186  Health and
Hygiene   85.1224            OUT035                        2004       Small
Tier 2  Supermarket Type1          1193.1136
8521           FDN46        7.210          Regular         0.145221
Snack Foods  103.1332            OUT018                        2009      Medium
Tier 3  Supermarket Type2          1845.5976
8522           DRG01       14.800          Low Fat         0.044878         Soft
Drinks   75.4670            OUT046                        1997       Small
Tier 1  Supermarket Type1           765.6700
```

5.0.2 5.2 Inspecting the Data

[152]:
```python
df.info()
```

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

[153]:
```python
print('Size of the data :', df.size)
print('Dimenstion of Data :', df.shape)
```

```
Size of the data : 102276
Dimenstion of Data : (8523, 12)
```

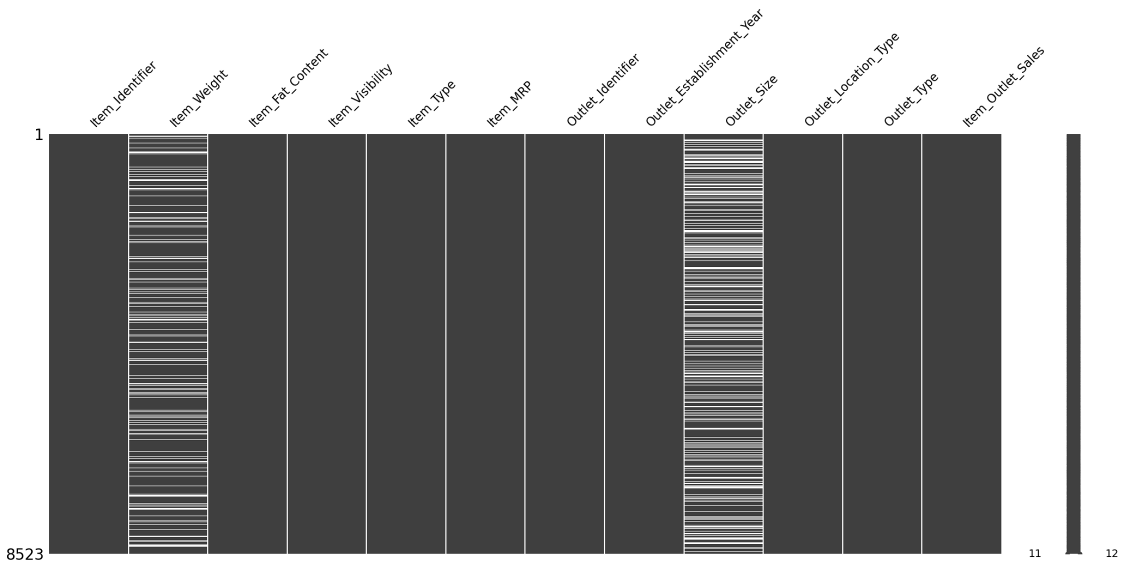
5.0.3 5.3 Checking for Null and duplicate Data’s

[154]:
```python
# checking for duplicate rows
df_duplicate=df.copy()
df_duplicate.drop_duplicates(subset=None, inplace=True)
df_duplicate.shape
```

[154]:
```
(8523, 12)
```

[155]:
```python
# checking for nul values.
mn.matrix(df)
```

[155]:
```
<Axes: >
```



5.0.4 5.4 Dealing with Missing Values

[156]:
```
# here we can see that there are some missing values in the Item_weight and␣
↪Outlet_Size
# Dealing with the Item_weight column
df.Item_Weight.value_counts(normalize=True)
```

[156]:
```
12.150    0.012181
17.600    0.011615
13.650    0.010907
11.800    0.010765
15.100    0.009632
9.300     0.009632
10.500    0.009348
16.700    0.009348
19.350    0.008924
20.700    0.008782
16.000    0.008782
9.800     0.008640
17.700    0.008499
17.750    0.008499
18.850    0.008357
15.850    0.008357
15.000    0.008357
16.750    0.008215
18.250    0.008215
19.600    0.008215
15.700    0.008074
```

9.195	0.007932
12.500	0.007932
20.200	0.007507
12.100	0.007507
12.600	0.007507
10.195	0.007507
15.600	0.007365
13.500	0.007224
11.500	0.007224
19.700	0.007082
11.600	0.007082
20.250	0.007082
12.350	0.007082
12.850	0.006941
9.600	0.006941
12.300	0.006941
9.500	0.006941
13.150	0.006941
17.850	0.006799
20.350	0.006657
14.000	0.006657
15.500	0.006657
15.200	0.006516
16.500	0.006516
16.350	0.006516
17.250	0.006374
14.500	0.006232
20.500	0.006232
19.000	0.006232
10.100	0.006232
9.000	0.006232
18.200	0.006091
10.000	0.006091
10.300	0.006091
16.200	0.006091
11.100	0.005949
13.350	0.005949
19.100	0.005807
17.500	0.005807
14.150	0.005807
16.100	0.005807
13.000	0.005666
15.350	0.005666
20.750	0.005524
19.850	0.005524
19.200	0.005524
11.650	0.005524

13.100	0.005524
18.000	0.005382
20.600	0.005382
18.700	0.005382
18.350	0.005241
18.600	0.005241
17.350	0.005241
17.100	0.005099
12.650	0.005099
10.895	0.005099
17.000	0.005099
8.895	0.004958
20.850	0.004958
14.300	0.004958
10.800	0.004958
19.500	0.004816
13.800	0.004816
9.695	0.004816
14.850	0.004816
11.300	0.004674
9.395	0.004533
20.100	0.004533
20.000	0.004533
11.350	0.004108
11.150	0.004108
16.600	0.004108
16.850	0.004108
12.800	0.003824
14.650	0.003824
8.600	0.003824
7.500	0.003683
10.695	0.003683
18.500	0.003683
11.000	0.003683
13.850	0.003541
19.250	0.003541
18.750	0.003399
21.250	0.003399
16.250	0.003399
13.600	0.003258
19.750	0.003258
14.100	0.003258
14.600	0.002975
13.300	0.002975
15.250	0.002975
9.100	0.002833
5.880	0.002691

14.800	0.002691
7.390	0.002550
10.600	0.002550
21.100	0.002408
7.720	0.002408
14.350	0.002408
10.650	0.002408
7.825	0.002266
7.235	0.002266
15.750	0.002266
10.395	0.002266
6.865	0.002266
17.200	0.002266
5.785	0.002125
7.420	0.002125
7.905	0.002125
8.710	0.002125
9.895	0.002125
8.270	0.002125
5.780	0.002125
5.175	0.002125
11.395	0.002125
6.135	0.002125
5.820	0.002125
8.420	0.002125
7.270	0.002125
7.075	0.001983
15.150	0.001983
11.850	0.001983
14.700	0.001983
7.475	0.001841
18.100	0.001841
12.000	0.001841
6.780	0.001841
8.300	0.001841
7.855	0.001700
8.930	0.001700
7.285	0.001700
7.970	0.001700
6.360	0.001700
8.365	0.001700
6.635	0.001700
7.785	0.001700
6.425	0.001700
8.180	0.001700
6.130	0.001700
6.590	0.001700

8.880	0.001700
8.850	0.001700
6.110	0.001700
5.460	0.001700
5.655	0.001700
8.510	0.001558
8.975	0.001558
10.850	0.001558
8.235	0.001558
8.630	0.001558
8.390	0.001558
6.215	0.001558
7.405	0.001558
8.395	0.001558
6.670	0.001558
7.365	0.001558
5.940	0.001416
8.210	0.001416
8.100	0.001416
7.050	0.001416
7.680	0.001416
7.550	0.001416
8.355	0.001416
6.055	0.001416
7.020	0.001416
7.750	0.001416
6.650	0.001416
8.430	0.001275
6.235	0.001275
6.920	0.001275
7.810	0.001275
8.185	0.001275
6.675	0.001275
6.615	0.001275
5.980	0.001275
6.710	0.001275
7.895	0.001275
6.825	0.001275
6.035	0.001275
8.260	0.001275
7.935	0.001275
7.630	0.001275
6.630	0.001133
6.380	0.001133
8.785	0.001133
6.385	0.001133
8.020	0.001133

6.260	0.001133
8.500	0.001133
6.115	0.001133
8.645	0.001133
5.365	0.001133
7.930	0.001133
7.725	0.001133
5.765	0.001133
6.550	0.001133
7.510	0.001133
8.575	0.001133
15.300	0.001133
7.520	0.001133
5.985	0.001133
8.050	0.001133
6.320	0.000992
8.770	0.000992
5.150	0.000992
7.975	0.000992
9.285	0.000992
5.260	0.000992
21.350	0.000992
7.670	0.000992
4.610	0.000992
8.155	0.000992
5.945	0.000992
7.435	0.000992
5.695	0.000992
8.060	0.000992
7.000	0.000992
7.035	0.000850
7.600	0.000850
8.315	0.000850
6.800	0.000850
21.000	0.000850
8.195	0.000850
5.465	0.000850
7.300	0.000850
7.350	0.000850
7.210	0.000850
6.465	0.000850
5.590	0.000850
8.380	0.000850
8.010	0.000850
6.850	0.000850
8.775	0.000850
6.030	0.000850

8.750	0.000850
6.695	0.000850
6.570	0.000850
7.220	0.000850
8.890	0.000850
6.980	0.000850
6.960	0.000850
6.445	0.000850
5.190	0.000850
5.730	0.000850
9.310	0.000850
8.520	0.000850
5.320	0.000850
5.615	0.000850
7.655	0.000850
5.325	0.000708
7.315	0.000708
5.110	0.000708
5.800	0.000708
4.635	0.000708
5.480	0.000708
7.155	0.000708
5.405	0.000708
5.440	0.000708
6.765	0.000708
6.890	0.000708
7.310	0.000708
7.145	0.000708
8.905	0.000708
6.195	0.000708
5.095	0.000708
7.325	0.000708
6.300	0.000708
5.510	0.000708
8.960	0.000708
4.880	0.000708
9.170	0.000708
6.985	0.000708
8.970	0.000708
21.200	0.000708
8.655	0.000708
9.130	0.000708
5.920	0.000708
8.985	0.000708
5.500	0.000708
6.155	0.000708
8.680	0.000708

5.925	0.000708
6.750	0.000708
4.590	0.000708
6.715	0.000708
8.695	0.000708
6.575	0.000708
6.480	0.000708
6.150	0.000708
4.785	0.000708
8.310	0.000708
8.945	0.000708
7.760	0.000708
7.945	0.000708
6.785	0.000708
4.920	0.000708
7.575	0.000708
7.470	0.000708
6.365	0.000708
8.935	0.000567
7.445	0.000567
7.485	0.000567
4.555	0.000567
7.360	0.000567
9.210	0.000567
6.420	0.000567
5.485	0.000567
6.175	0.000567
7.170	0.000567
8.615	0.000567
5.340	0.000567
7.640	0.000567
6.170	0.000567
5.905	0.000567
7.960	0.000567
6.760	0.000567
5.630	0.000567
6.655	0.000567
5.030	0.000567
7.840	0.000567
6.860	0.000567
8.325	0.000567
6.965	0.000567
6.690	0.000567
7.850	0.000567
7.090	0.000567
8.115	0.000567
7.535	0.000567

6.610	0.000567
5.635	0.000567
7.645	0.000567
14.750	0.000567
5.035	0.000567
5.860	0.000567
4.805	0.000567
6.885	0.000567
7.060	0.000567
7.865	0.000567
9.270	0.000567
4.615	0.000567
7.100	0.000567
6.095	0.000567
8.840	0.000567
9.065	0.000567
6.525	0.000567
7.590	0.000567
5.750	0.000567
6.280	0.000567
8.760	0.000567
6.305	0.000425
5.305	0.000425
8.000	0.000425
6.935	0.000425
7.710	0.000425
7.105	0.000425
6.460	0.000425
7.260	0.000425
8.275	0.000425
5.825	0.000425
9.105	0.000425
5.425	0.000425
5.845	0.000425
7.070	0.000425
9.060	0.000425
7.565	0.000425
9.035	0.000425
5.000	0.000425
6.440	0.000425
8.350	0.000425
6.905	0.000425
6.895	0.000283
6.400	0.000283
7.605	0.000283
8.670	0.000283
5.210	0.000283

```
8.485     0.000283
6.775     0.000283
7.890     0.000283
5.155     0.000283
5.885     0.000283
4.905     0.000283
7.560     0.000283
6.325     0.000283
8.800     0.000283
6.405     0.000283
5.675     0.000283
8.920     0.000283
5.735     0.000283
7.275     0.000283
7.685     0.000142
9.420     0.000142
6.520     0.000142
5.400     0.000142
Name: Item_Weight, dtype: float64
```

[157]:
```
Item_Weight_null=df.Item_Weight.isnull()
```

[158]:
```
Item_Weight_null
```

[158]:
```
0       False
1       False
2       False
3       False
4       False
        …
8518    False
8519    False
8520    False
8521    False
8522    False
Name: Item_Weight, Length: 8523, dtype: bool
```

[159]:
```
mean_value=df['Item_Weight'].mean()
mean_value
```

[159]:
```
12.857645184135976
```

[160]:
```
df['Item_Weight'].fillna(df['Item_Weight'].mean(), inplace=True )
```

[161]:
```
df.isnull().sum()
```

[161]:
```
Item_Identifier                 0
Item_Weight                     0
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  2410
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64
```

[162]:
```
# Treating Outlet_size
outlet_size_mode = df.pivot_table(values='Outlet_Size', columns='Outlet_Type',␣
 ↪aggfunc=(lambda x: x.mode()[0]))
outlet_size_mode
```

[162]:
```
Outlet_Type Grocery Store Supermarket Type1 Supermarket Type2 Supermarket Type3
Outlet_Size         Small             Small            Medium            Medium
```

[163]:
```
### Dealing with the missing values of the Outlet_Size.
### We can replace the missing values of the Outlet_Size by using the mode
miss_bool = df['Outlet_Size'].isnull() #False=>present , True =>missing
df.loc[miss_bool, 'Outlet_Size'] = df.loc[miss_bool, 'Outlet_Type'].
 ↪apply(lambda x: outlet_size_mode[x])

df['Outlet_Size'].isnull().sum()
```

[163]: 0

[164]:
```
df.isnull().sum()
```

[164]:
```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64
```

6 6. Data Standarization

6.0.1 6.1 Standarizing the values of the Item_fat_content

[165]:
```
# Standarizing the values of the Item_fat_content
df['Item_Fat_Content']=df['Item_Fat_Content'].replace({'LF':'Low Fat', 'reg':
↪'Regular', 'low fat':'Low Fat'})
```

[166]:
```
sum(df['Item_Visibility']==0)
```

[166]:
```
526
```

6.0.2 6.2 Restructuring Column Item_Visibility

[167]:
```
# there are 526 zero Visibility values
# replacing zeros with mean
df.loc[:, 'Item_Visibility'].replace([0], [df['Item_Visibility'].mean()],␣
↪inplace=True)
sum(df['Item_Visibility']==0)
```

[167]:
```
0
```

[168]:
```
df.head()
```

[168]:
```
  Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility
Item_Type  Item_MRP Outlet_Identifier  Outlet_Establishment_Year Outlet_Size
Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales
0           FDA15         9.30          Low Fat         0.016047
Dairy  249.8092            OUT049                        1999      Medium
Tier 1  Supermarket Type1         3735.1380
1           DRC01         5.92          Regular         0.019278            Soft
Drinks   48.2692            OUT018                        2009      Medium
Tier 3  Supermarket Type2          443.4228
2           FDN15        17.50          Low Fat         0.016760
Meat  141.6180            OUT049                        1999      Medium
Tier 1  Supermarket Type1         2097.2700
3           FDX07        19.20          Regular         0.066132  Fruits and
Vegetables  182.0950            OUT010                        1998       Small
Tier 3      Grocery Store          732.3800
4           NCD19         8.93          Low Fat         0.066132
Household   53.8614            OUT013                        1987        High
Tier 3  Supermarket Type1          994.7052
```

6.0.3 6.3 Creating a new column derived from the column Item_Indentifier

[169]:
```
## we can create new_item_type by extracting the data from the Item_identifier.
df['New_Item_Type'] = df['Item_Identifier'].apply(lambda x: x[:2])
df['New_Item_Type']
```

[169]:
```
0       FD
1       DR
2       FD
3       FD
4       NC
        ..
8518    FD
8519    FD
8520    NC
8521    FD
8522    DR
Name: New_Item_Type, Length: 8523, dtype: object
```

[170]:
```
# here we can compare the 'New_item_type' column with the non-consumable where␣
↪it creates the boolean value if the New_Item_type has the non_consumbale␣
↪value it will return as true else flase
# It sets the 'Item_Fat_Content' column to the value 'Non-Edible' for all the␣
↪rows that satisfy the condition mentioned in the first part. In other words,␣
↪for all rows where 'New_Item_Type' is 'Non-Consumable', the␣
↪'Item_Fat_Content' column will be updated to 'Non-Edible'.

df.loc[df['New_Item_Type']=='Non-Consumable', 'Item_Fat_Content'] = 'Non-Edible'
df['Item_Fat_Content'].value_counts()
```

[170]:
```
Low Fat    5517
Regular    3006
Name: Item_Fat_Content, dtype: int64
```

[171]:
```
df['New_Item_Type'] = df['New_Item_Type'].map({'FD':'Food', 'NC':
↪'Non-Consumable', 'DR':'Drinks'})
df['New_Item_Type'].value_counts()
```

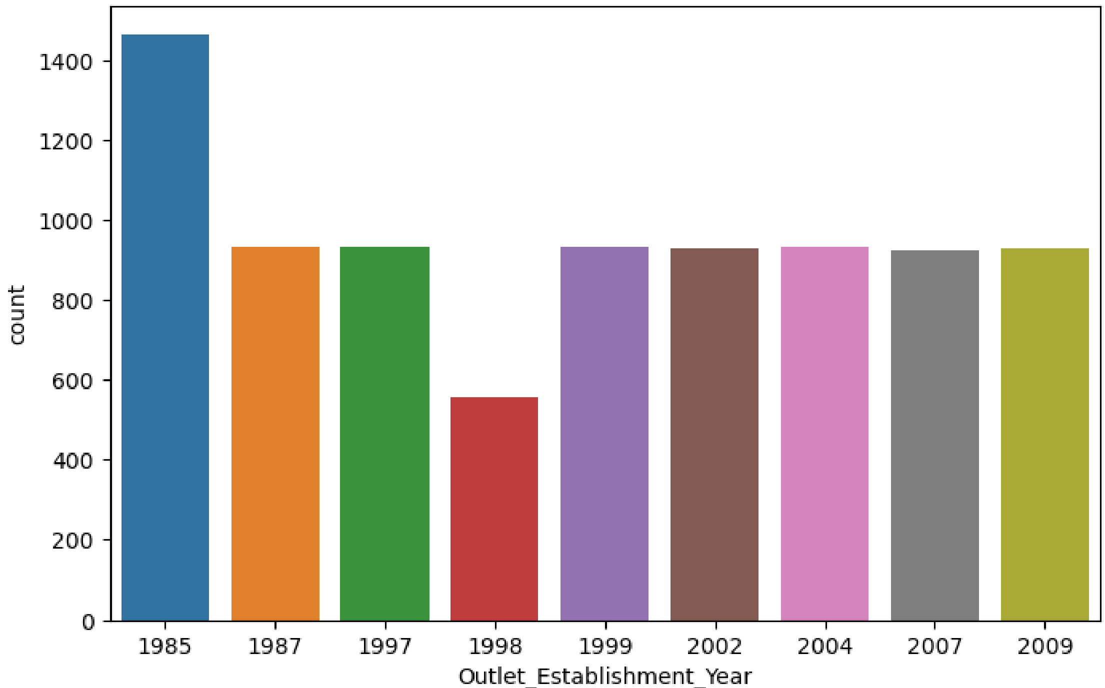
[171]:
```
Food              6125
Non-Consumable    1599
Drinks             799
Name: New_Item_Type, dtype: int64
```

7 7. Exploratory Data Analysis.

7.0.1 7.1 Outlet Establishment year

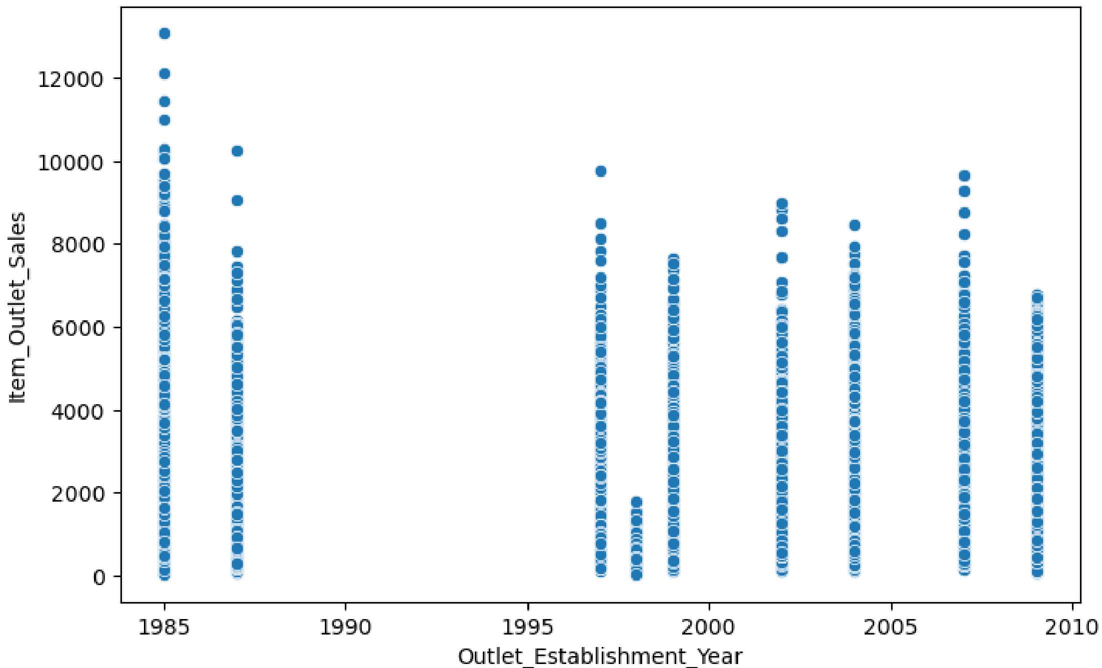
- Outlet_Establishment_Year: The year in which store was established

[172]:
```
plt.figure(figsize=(8,5))
sns.countplot(x='Outlet_Establishment_Year', data=df)
plt.show()
```



7.0.2 7.2 Outlet Establishment Year v/s Item Outlet Sale

[173]:
```
plt.figure(figsize=(8,5))
sns.scatterplot(x='Outlet_Establishment_Year',y='Item_Outlet_Sales', data=df)
plt.show()
```

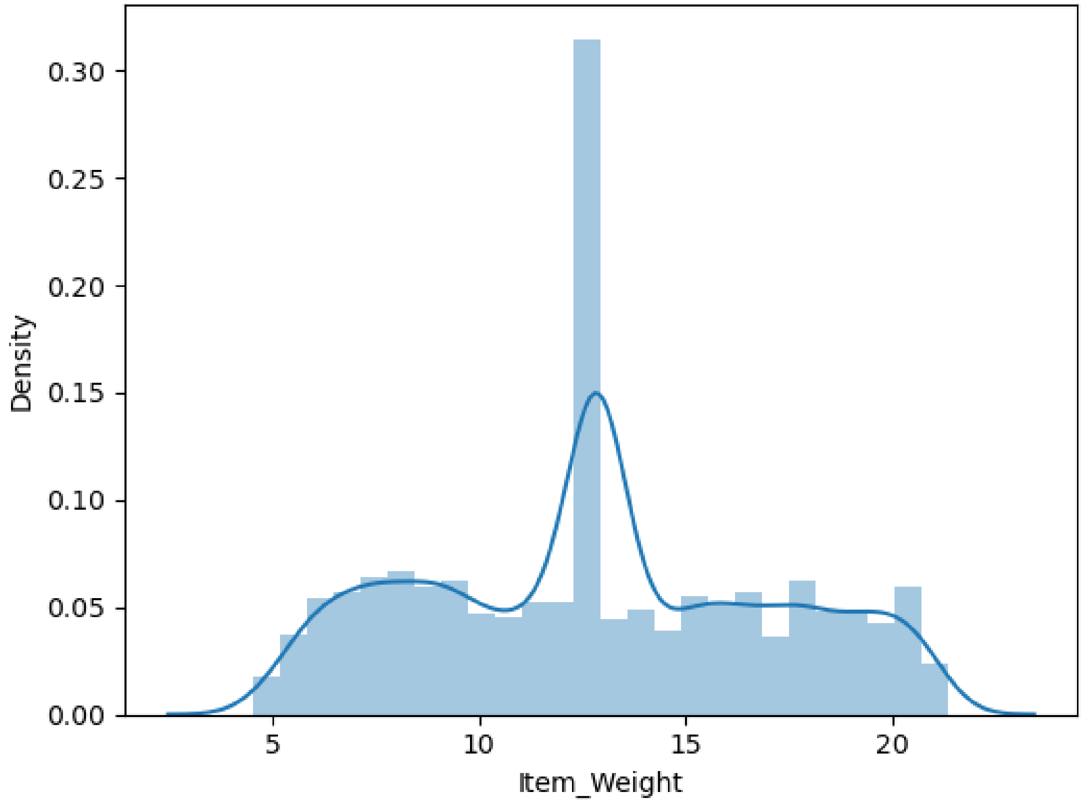


7.0.3 7.3 Item_Weight

- Item_Weight: Weight of product

[174]:
```
df['Item_Weight'] = pd.to_numeric(df['Item_Weight'], errors='coerce')

# Fill missing values (NaN) with the mean of the column
df['Item_Weight'].fillna(df['Item_Weight'].mean(), inplace=True)

# Now you can plot the distribution
sns.distplot(df['Item_Weight'])
```

[174]:
```
<Axes: xlabel='Item_Weight', ylabel='Density'>
```

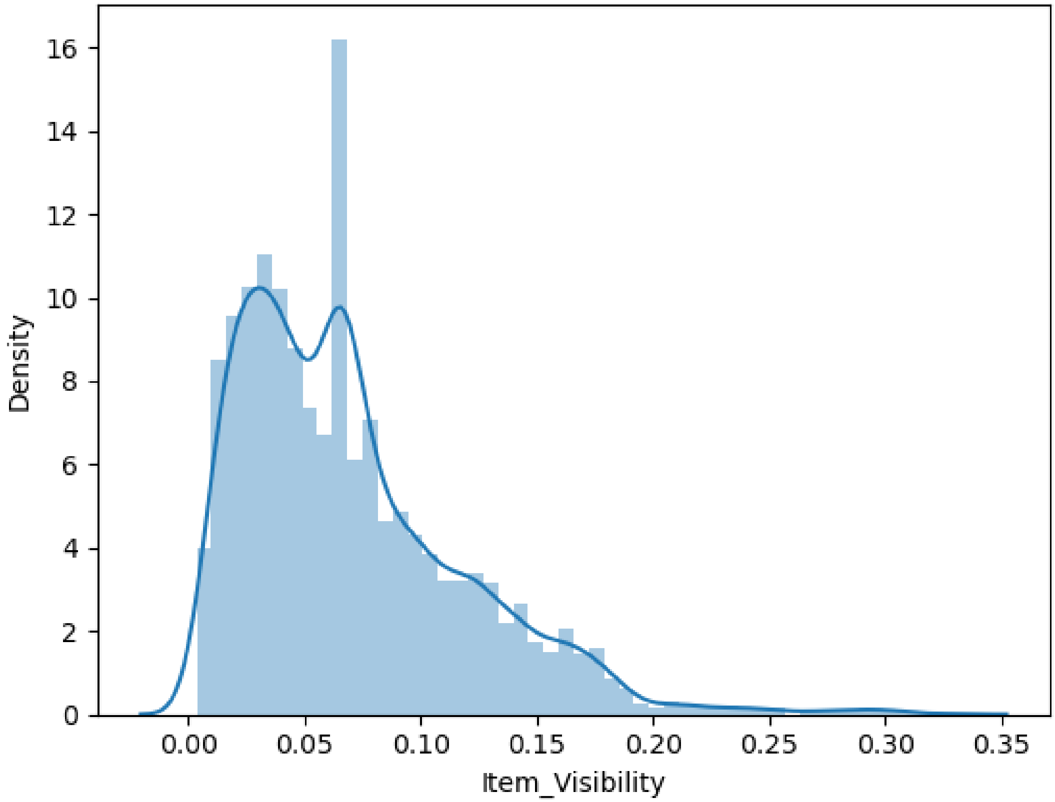


7.0.4 7.4 Item Visbility

- Item_Visibility: The % of total display area of all products in a store allocated to the

[175]:
```
sns.distplot(df['Item_Visibility'])
```

[175]: <Axes: xlabel='Item_Visibility', ylabel='Density'>

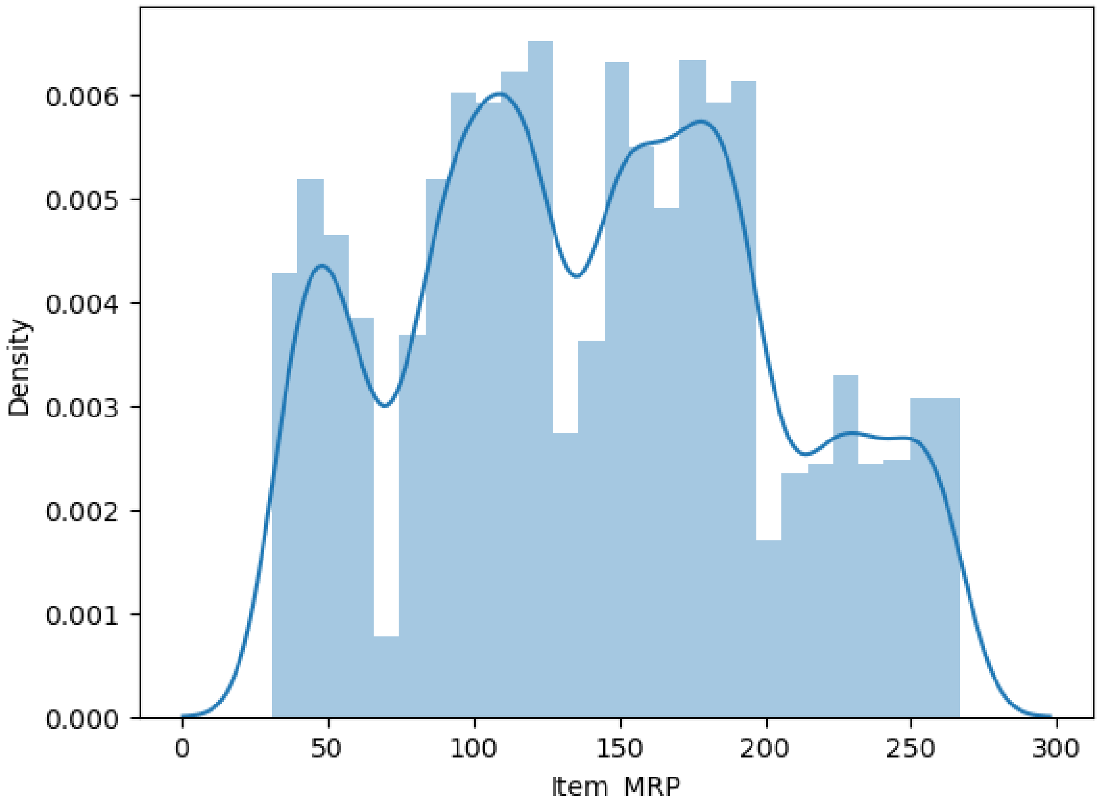


7.0.5 7.5 Item_MRP

- Maximum Retail Price (list price) of the product

[176]:
```
sns.distplot(df['Item_MRP'])
```

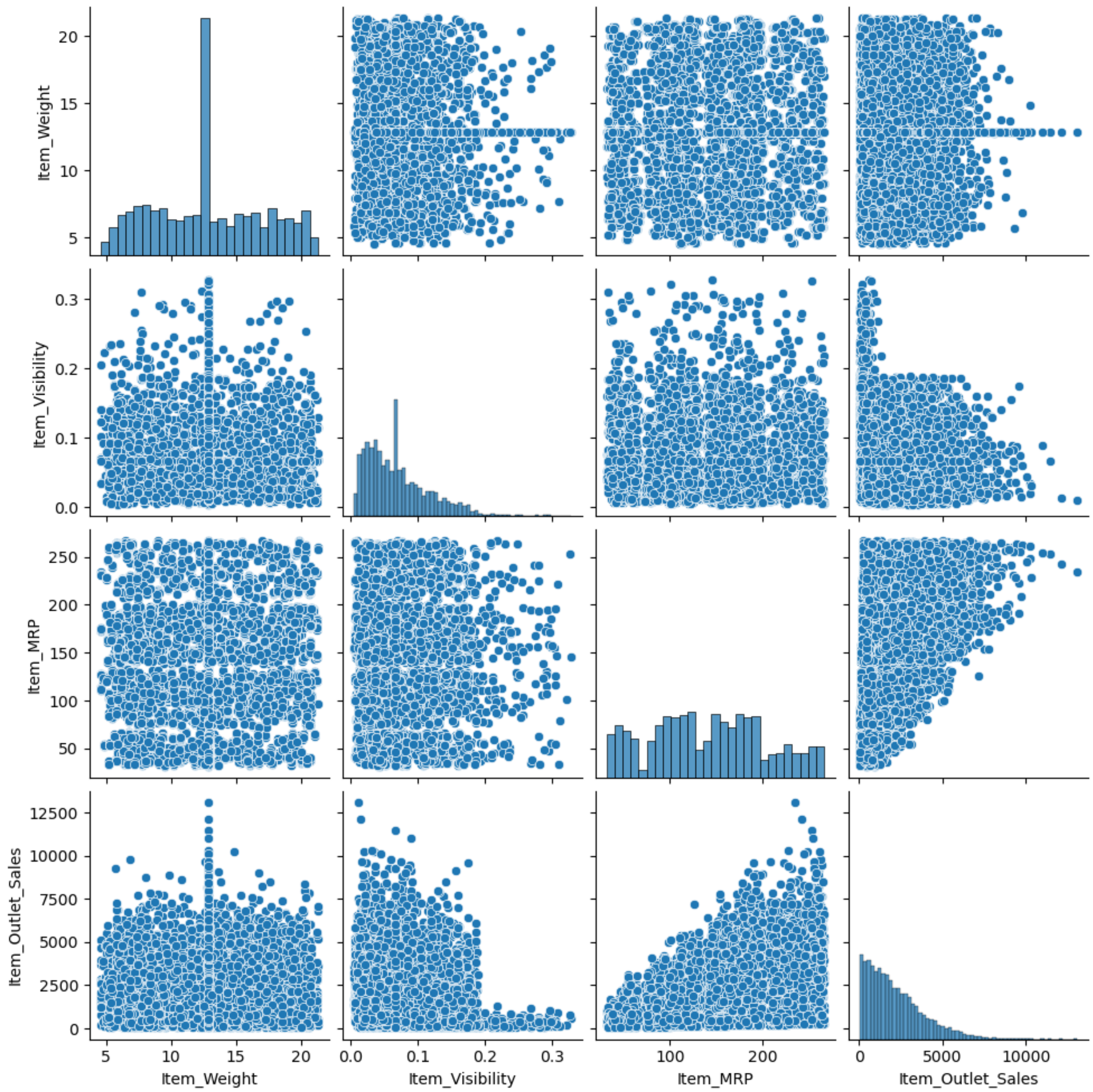
[176]: `<Axes: xlabel='Item_MRP', ylabel='Density'>`



7.0.6 7.6 Pair Plot

[177]:
```
df_numeric=df.select_dtypes(include=['float64'])
# plotting the pair plot collectively
plt.figure(figsize=(20, 10))
sns.pairplot(df_numeric)
plt.show()
```

```
<Figure size 2000x1000 with 0 Axes>
```

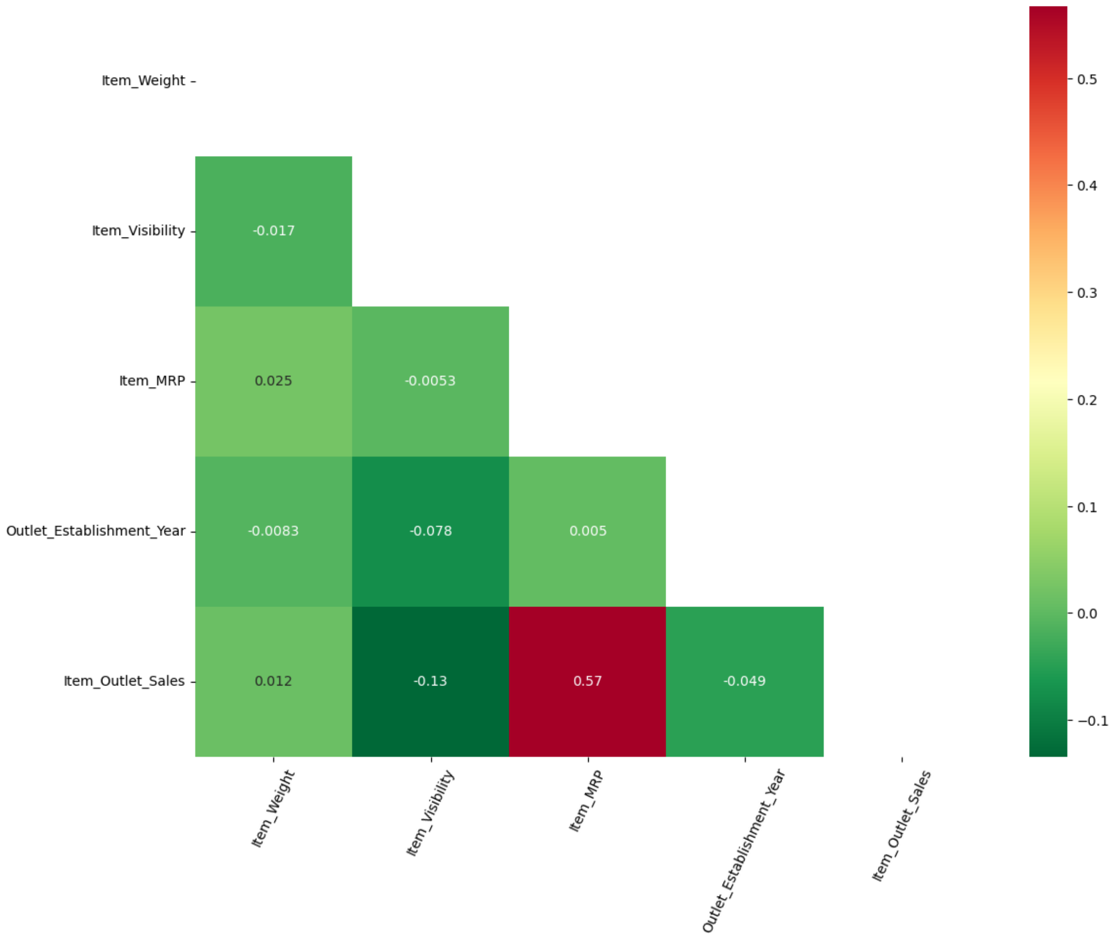


- Here we can observe that there no specific correation between the variables
- The Pair Plot reveals that the Item_MRP is shows a Positive trend with the Item Outlet.With Increasing MRP the Sales Increases.
- Item Visibility shows a negative trend such that Item having less visibility has the low sales which is quite opposite from the understanding . There we can say that visibilty has relation with the Sales of the Item.

7.0.7 7.7 Correlation Heat Map

[178]:
```
### check colleration for all columns
df_corr =df.corr()
mask = np.triu(np.ones_like(df_corr,dtype=bool))
```

```
plt.figure(figsize=(13,10))
sns.heatmap(df_corr, cmap='RdYlGn_r', mask=mask , annot=True)
plt.xticks(rotation=65)
plt.show()
```



- Here in the correlation heatmap the Item_MRP shows the highest correlation with the Item_Outlet_Sales.

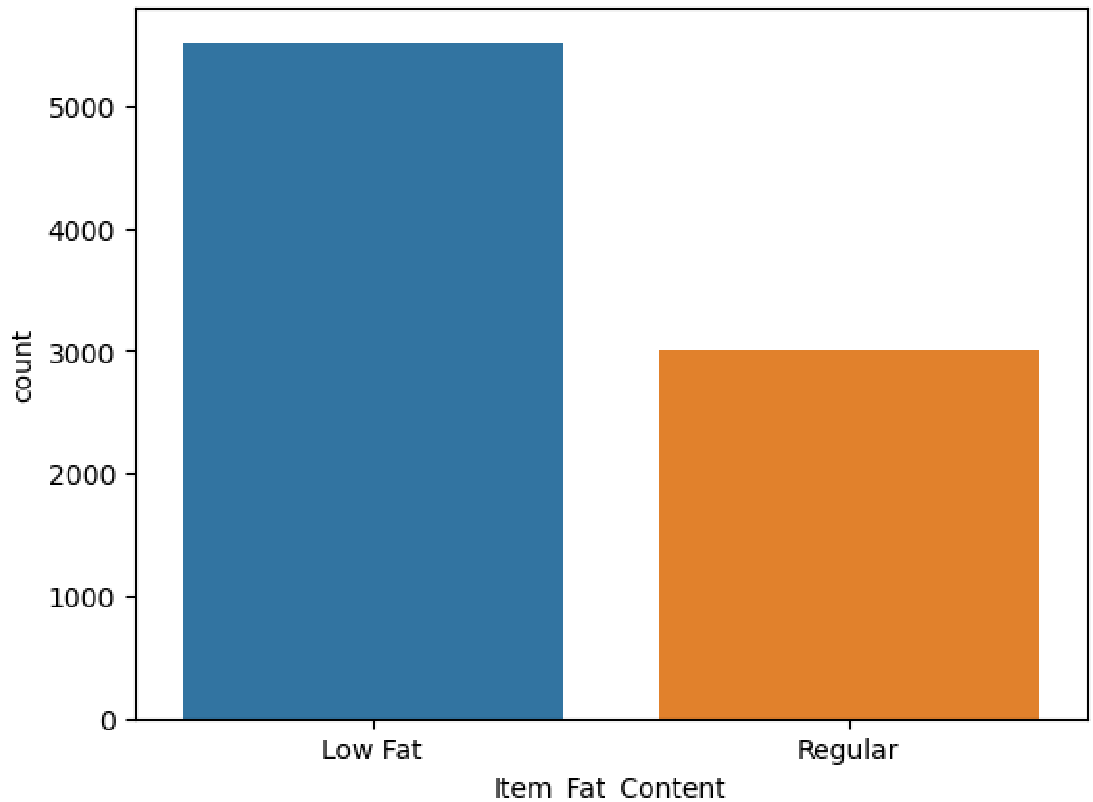
[179]:
```
# dealing with the categorical data
df_categorical=df.select_dtypes('object')
```

7.0.8 Deaing with Categorical Values .

7.0.9 7.7 Item Fat Content

- Item_Fat_Content: Whether the product is low fat or not

[180]:
```
sns.countplot(x= 'Item_Fat_Content', data=df)
plt.show()
```

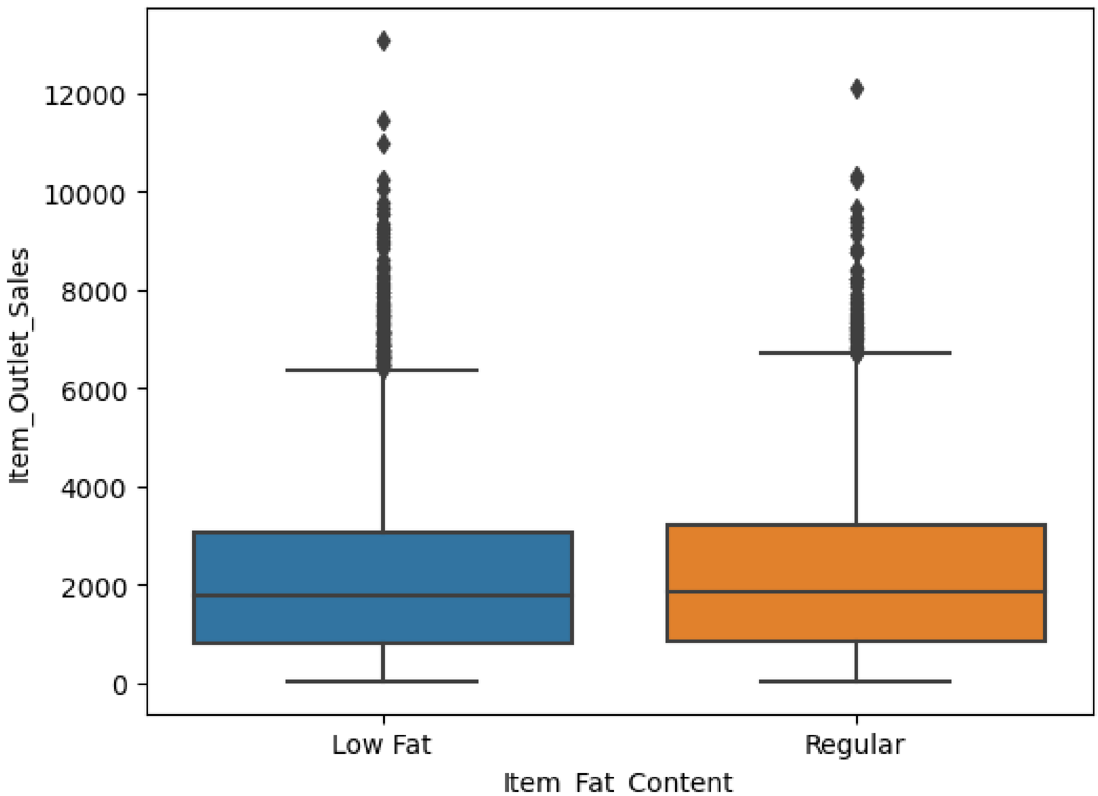


- There are more products which falls in the category of Low Fat Content Type.

7.0.10 7.7.1 Item_Fat_Content v/s Item Outlet Sales

[181]:
```
sns.boxplot(x='Item_Fat_Content',y='Item_Outlet_Sales',data=df)
```

[181]:
```
<Axes: xlabel='Item_Fat_Content', ylabel='Item_Outlet_Sales'>
```

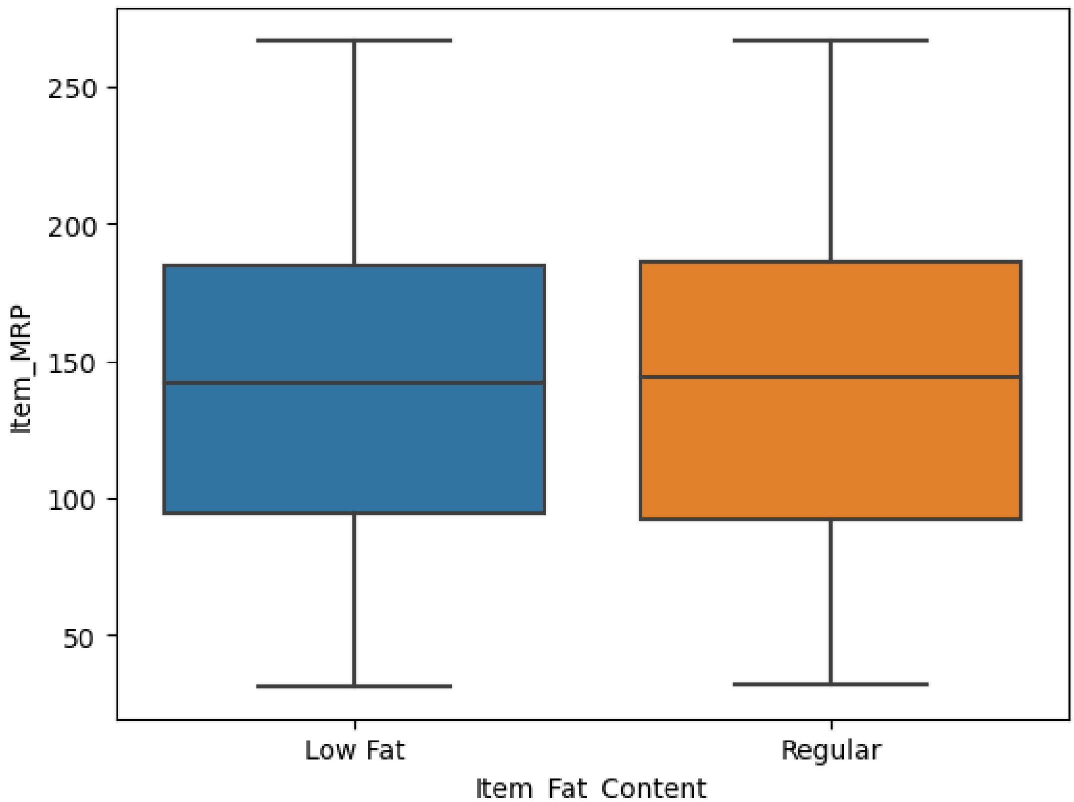


- Despite having large number of low fat content items in the store it does not justify the low fat sales in the store, As we can observe that the low fat content has the same sales as of the Regular and Non-Edible Items.

7.0.11 7.7.2 Item Fat Content v/s Item MRP

[182]:
```
sns.boxplot(x='Item_Fat_Content',y='Item_MRP',data=df)
```

[182]:
```
<Axes: xlabel='Item_Fat_Content', ylabel='Item_MRP'>
```



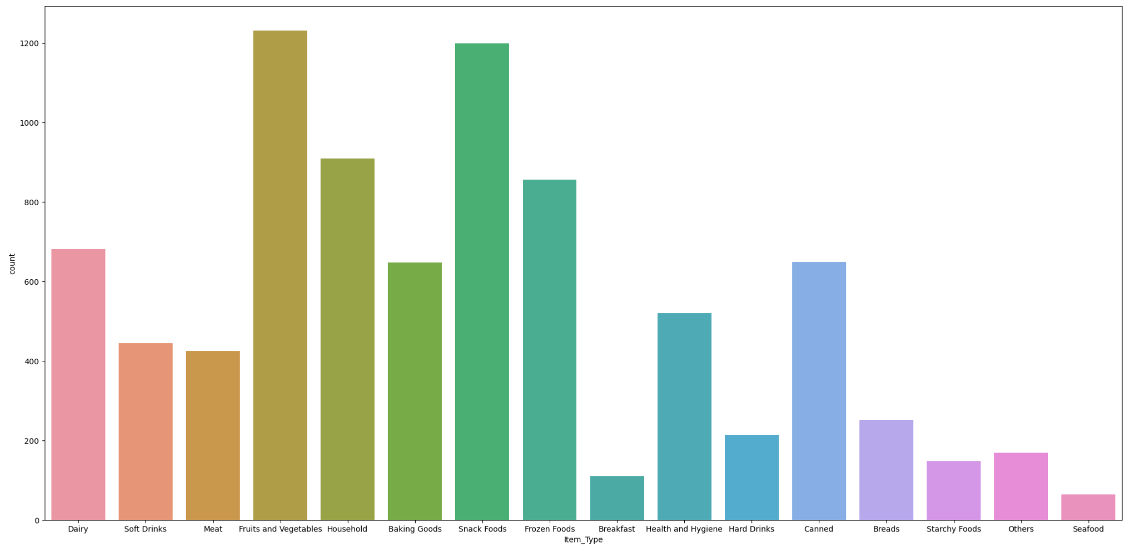
- MRP of the all three type of products are some what same.

7.0.12 7.8 Item Type

- Item_Type : The category to which the product belongs

[183]:
```
plt.figure(figsize=(25,12))
sns.countplot(x='Item_Type',data=df)
```

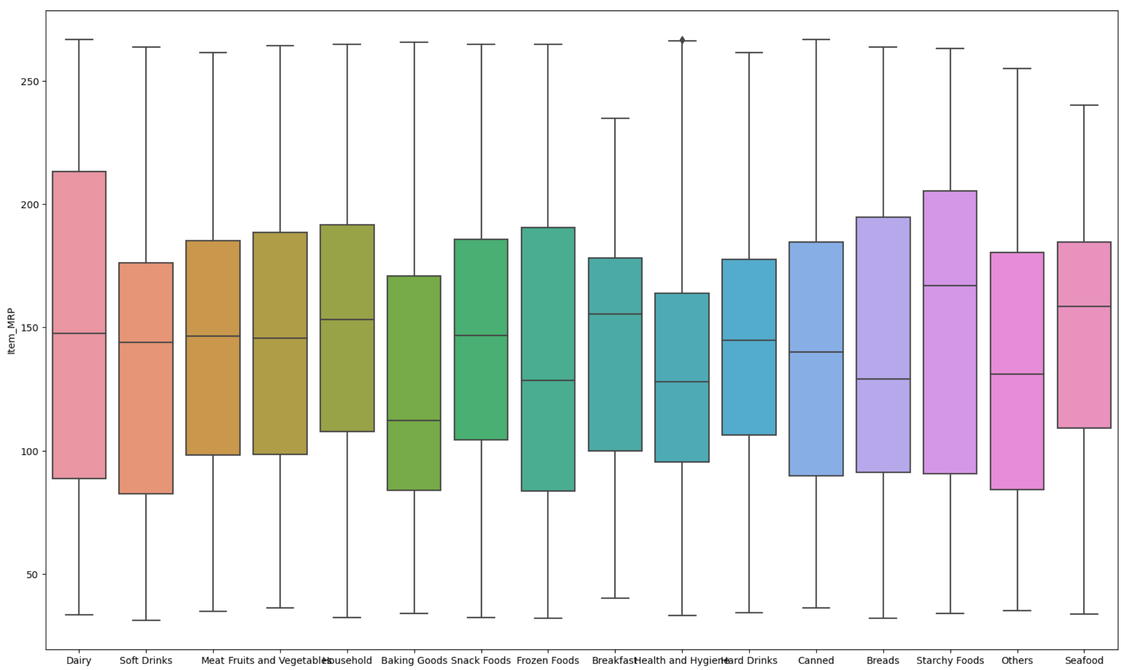
[183]: <Axes: xlabel='Item_Type', ylabel='count'>



7.0.13 7.8.1 Item_type v/s Item_MRP

[184]:
```
plt.figure(figsize=(20,12))
sns.boxplot(x='Item_Type',y='Item_MRP',data=df)
```

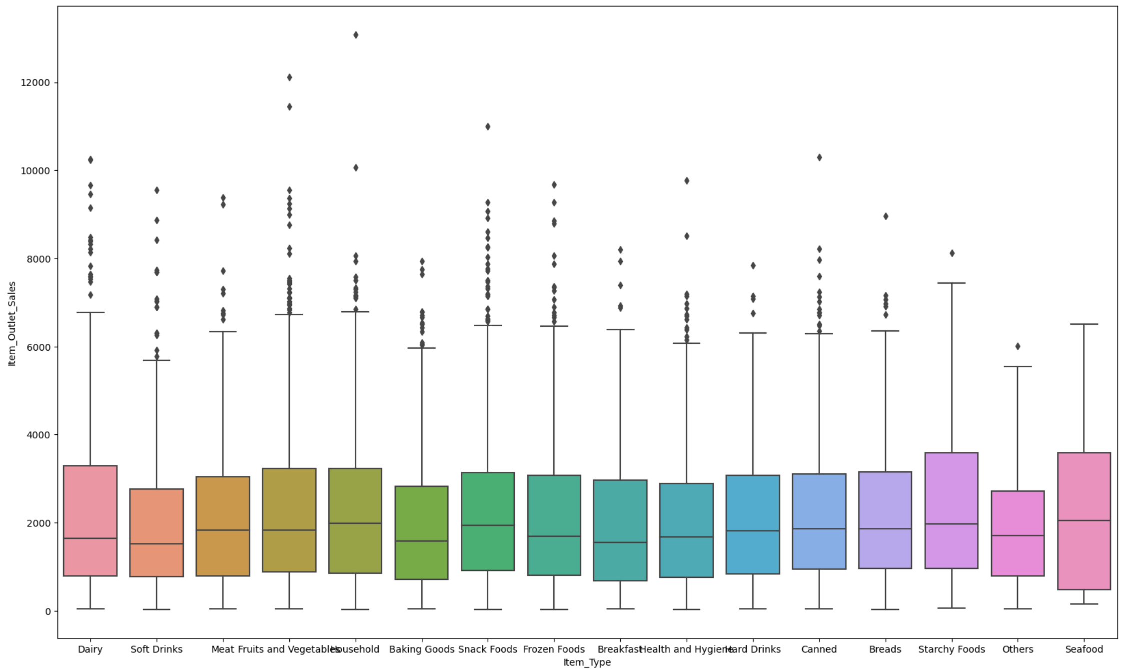
[184]: <Axes: xlabel='Item_Type', ylabel='Item_MRP'>



7.0.14 7.8.2 Item Type v/s Item Outlet Sales

[185]:
```
plt.figure(figsize=(20,12))

sns.boxplot(x='Item_Type',y='Item_Outlet_Sales',data=df)
```

[185]: <Axes: xlabel='Item_Type', ylabel='Item_Outlet_Sales'>



[ ]:
```

```

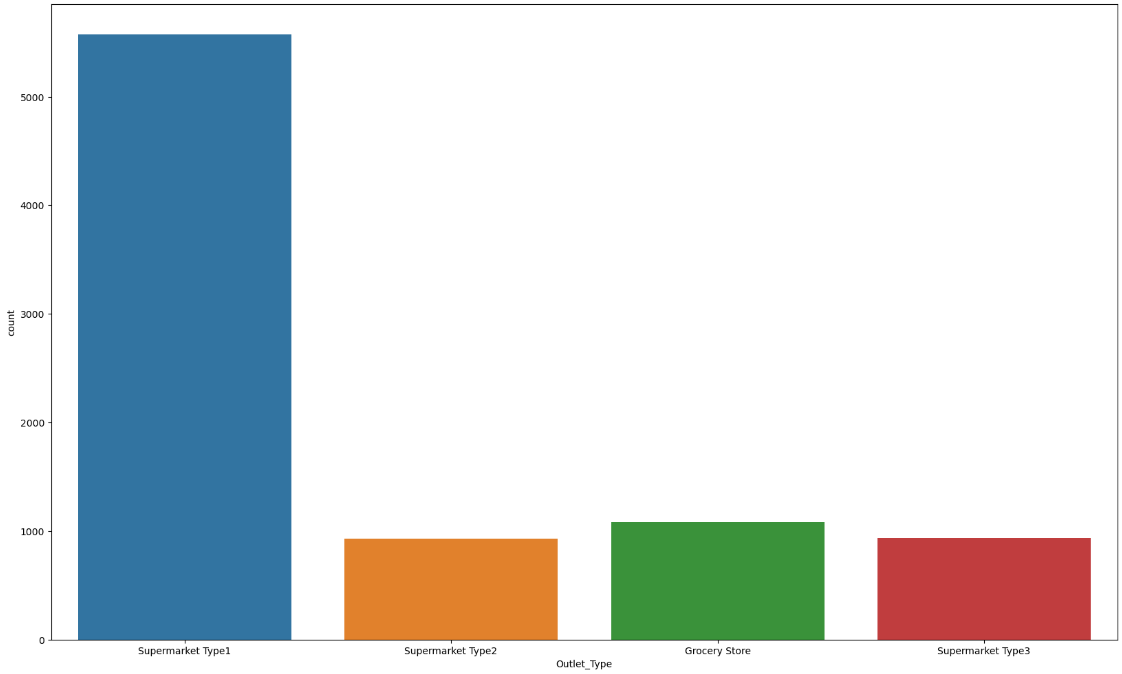
- The Item Fruits and Vegetables and Snack Food are the most

7.0.15 7.9 Outlet Type

- Outlet_Type : Whether the outlet is just a grocery store or some sort of supermarket

[186]:
```
plt.figure(figsize=(20,12))
sns.countplot(x='Outlet_Type',data=df)
```

[186]: <Axes: xlabel='Outlet_Type', ylabel='count'>

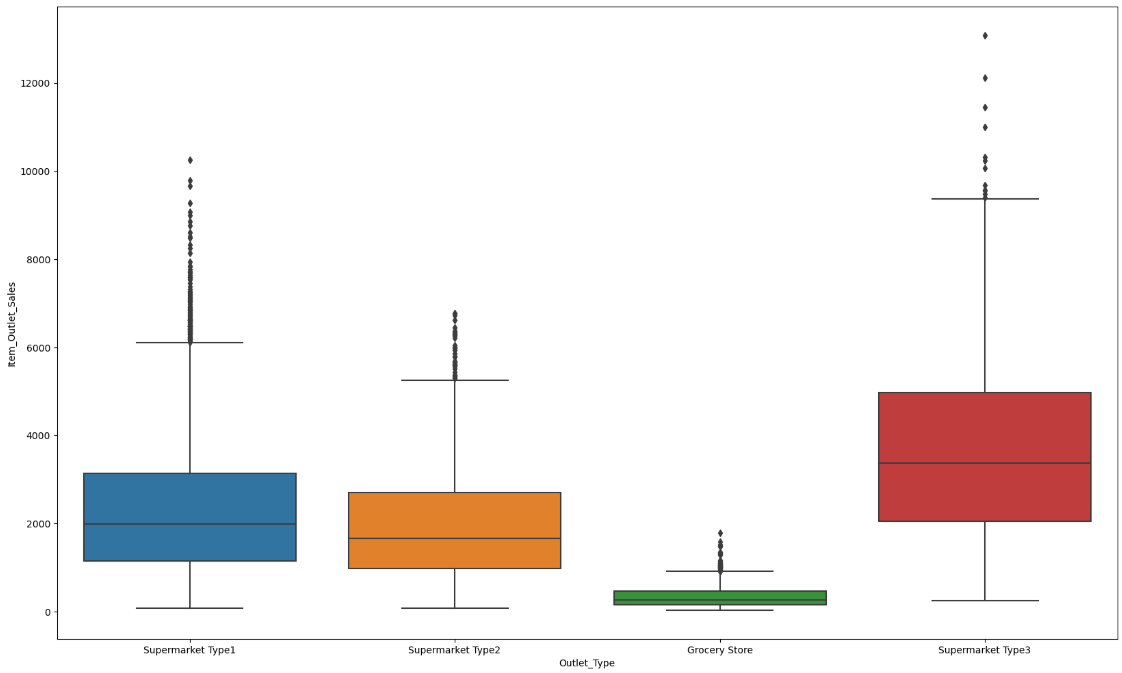


- Supermarket Type 1 is has most number of the Stores as compared to the other stores . Supermarket type 2,Type 3 and Grocery Stores have the some what same amount of the Stores.

7.0.16 7.9.1 Outlet type v/s Item_Outlet_Sales

[187]:
```
plt.figure(figsize=(20,12))
sns.boxplot(x='Outlet_Type',y='Item_Outlet_Sales',data=df)
```

[187]:
```
<Axes: xlabel='Outlet_Type', ylabel='Item_Outlet_Sales'>
```



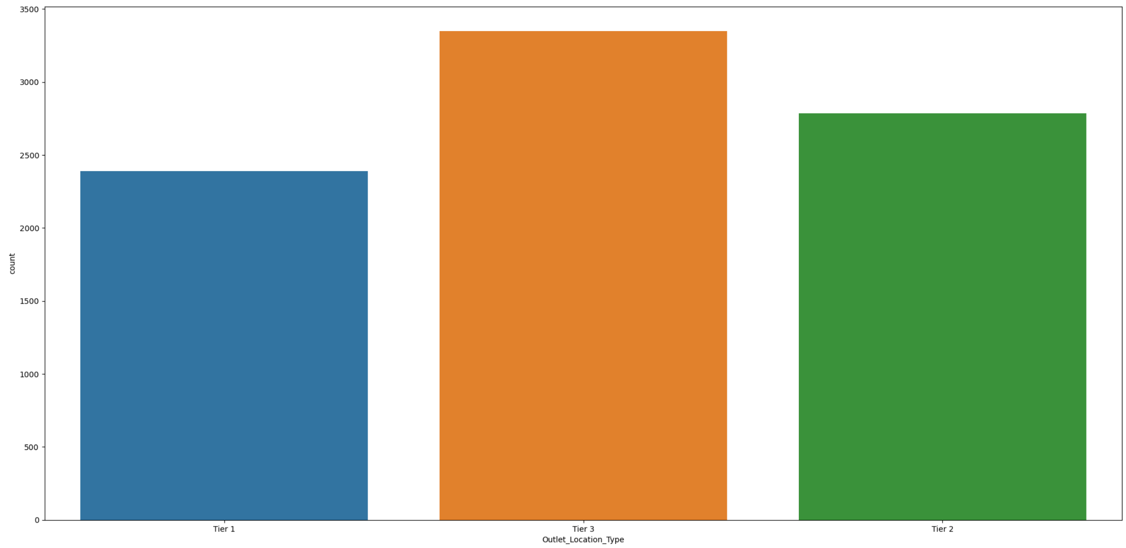
- Despite having the large numbr of stores of supermarket type 1 comparitavely there is no large amount of sales coming from the Supermarket type 1 ,
- Here even though Supermarket type 3 has the average amount of stores , the sales generated by the type 3 is greater than that of the any other Stores.
- Grocery Stores has the lowest sales generated of the any other stores.

7.1 7.10 Outlet_Location_Type

- outlet_Location_Type : The type of city in which the store is located

[188]:
```
plt.figure(figsize=(25,12))
sns.countplot(x='Outlet_Location_Type',data=df)
```

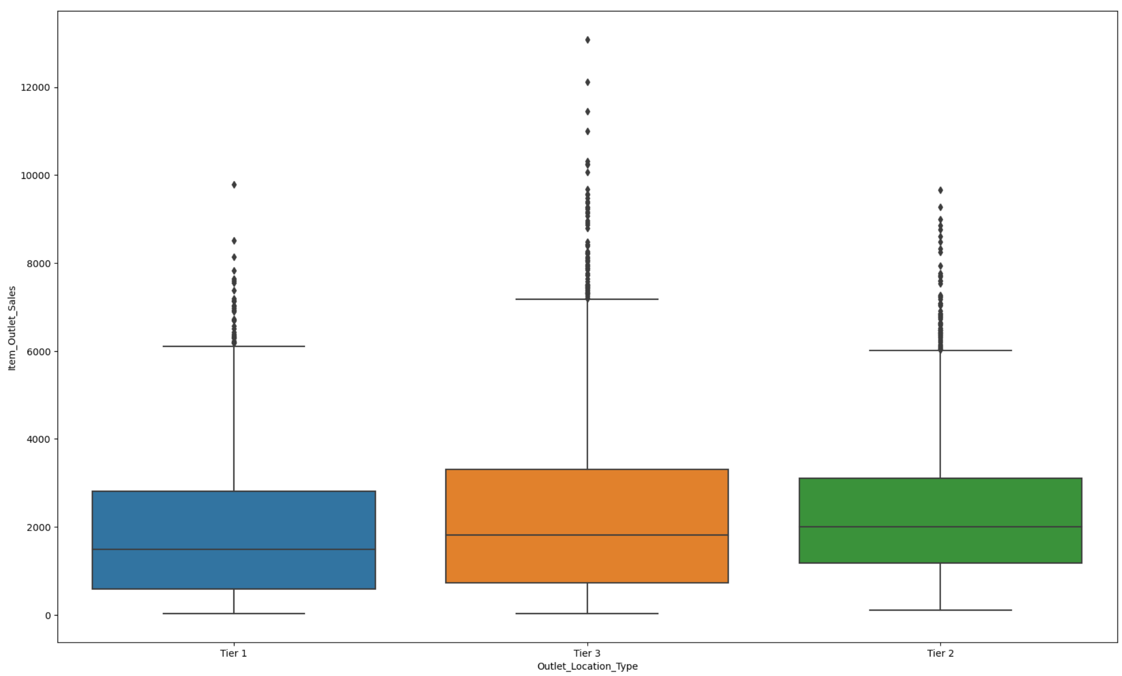
[188]:
```
<Axes: xlabel='Outlet_Location_Type', ylabel='count'>
```



7.1.1 7.10.1 Outlet_Location_Type v/s Outlet_Sales

[189]:
```
plt.figure(figsize=(20,12))
sns.boxplot(x='Outlet_Location_Type',y='Item_Outlet_Sales',data=df)
```

[189]: <Axes: xlabel='Outlet_Location_Type', ylabel='Item_Outlet_Sales'>



[190]:
```
mapping={'Tier 1':1 , 'Tier 2': 2 , 'Tier 3': 3}
df['Outlet_Location_Type']=df['Outlet_Location_Type'].map(mapping)
```

[191]:
```
df['Outlet_Location_Type']
```

[191]:
```
0       1
1       3
2       1
3       3
4       3
       ..
8518    3
8519    2
8520    2
8521    3
8522    1
Name: Outlet_Location_Type, Length: 8523, dtype: int64
```

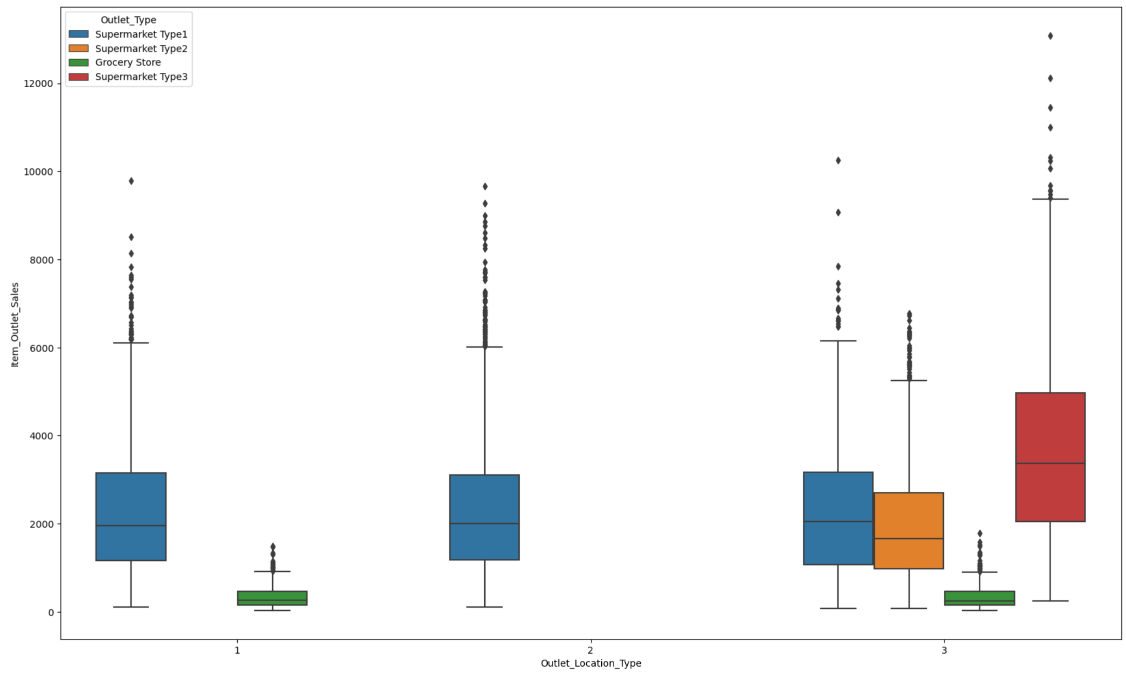
[192]:
```
df['Outlet_Location_Type'].value_counts(normalize=True)
```

[192]:
```
3    0.393054
2    0.326763
1    0.280183
Name: Outlet_Location_Type, dtype: float64
```

7.1.2 7.10.2 Outlet Location Type v/s Outlet_type

[193]:
```
plt.figure(figsize=(20,12))
sns.
 ↪boxplot(x='Outlet_Location_Type',y='Item_Outlet_Sales',data=df,hue='Outlet_Type')
```

[193]:
```
<Axes: xlabel='Outlet_Location_Type', ylabel='Item_Outlet_Sales'>
```



- Here we can observe that there is presence of the supermarket type 3 stores only in Tier 3 cities.
- In Tier 1 cities there are Super_market type 1 and Grocery Stores
- In Tier 2 cities there is only Supermarlet type 1 .

7.1.3 Insights

- From the Data visaulization we understood that Supermarket Type 3 , stores generate large amount of sales. Therefore it is necessary to increase the supermarket type 3 in numbers.
- The Supermarket_type 3 stores should be increase in rest of the tier cities ex. like Tier 1 and Tier 2 cities.
- Item_MRP strongly contiributes to the Sales of the Item.

8 8.Data Preprocessing

8.0.1 8.1 Converting categorical data into numerical data using Label_Encoder

[194]:
```
## we can convert categorical data into numerical data
import sklearn
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

[195]:
```
df['Outlet']=le.fit_transform(df['Outlet_Identifier'])
```

[196]:
```
df.columns
```

[196]:

```
Index(['Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibility',
'Item_Type', 'Item_MRP', 'Outlet_Identifier', 'Outlet_Establishment_Year',
'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type', 'Item_Outlet_Sales',
'New_Item_Type', 'Outlet'], dtype='object')
```

[197]:

```python
cat_col=['Item_Fat_Content','Item_Type','Outlet_Size','Outlet_Location_Type','Outlet_Type','Ne
```

[198]:

```python
for col in cat_col:
    df[col]=le.fit_transform(df[col])
```

8.0.2 8.2 one hot encoding

[203]:

```python
df = pd.get_dummies(df, columns=['Item_Fat_Content', 'Outlet_Size',␣
↪'Outlet_Location_Type', 'Outlet_Type', 'New_Item_Type'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[203], line 1
----> 1 df =␣
 ↪pd.get_dummies(df, columns=['Item_Fat_Content', 'Outlet_Size', 'Outlet_Location_Type', 'Ou

File ~\anaconda3\lib\site-packages\pandas\core\reshape\encoding.py:146, in␣
 ↪get_dummies(data, prefix, prefix_sep, dummy_na, columns, sparse, drop_first,␣
 ↪dtype)
    144     raise TypeError("Input must be a list-like for parameter `columns`")
    145 else:
--> 146     data_to_encode = data[columns]
    148 # validate prefixes and separator to avoid silently dropping cols
    149 def check_len(item, name):

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3813, in DataFrame.
 ↪__getitem__(self, key)
   3811     if is_iterator(key):
   3812         key = list(key)
-> 3813     indexer = self.columns._get_indexer_strict(key, "columns")[1]
   3815 # take() does not accept boolean indexers
   3816 if getattr(indexer, "dtype", None) == bool:

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:6070, in Index.
 ↪_get_indexer_strict(self, key, axis_name)
   6067 else:
   6068     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6070 self._raise_if_missing(keyarr, indexer, axis_name)
   6072 keyarr = self.take(indexer)
   6073 if isinstance(key, Index):
   6074     # GH 42790 - Preserve name from an Index
```

```
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:6130, in Index.
↪_raise_if_missing(self, key, indexer, axis_name)
   6128     if use_interval_msg:
   6129         key = list(key)
-> 6130     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6132 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
   6133 raise KeyError(f"{not_found} not in index")

KeyError: "None of [Index(['Item_Fat_Content', 'Outlet_Size',␣
↪'Outlet_Location_Type', 'Outlet_Type', 'New_Item_Type'], dtype='object')] are␣
↪in the [columns]"
```

[200]:
```
df.columns
```

[200]:
```
Index(['Item_Identifier', 'Item_Weight', 'Item_Visibility', 'Item_Type',
'Item_MRP', 'Outlet_Identifier', 'Outlet_Establishment_Year',
'Item_Outlet_Sales', 'Outlet', 'Item_Fat_Content_0', 'Item_Fat_Content_1',
'Outlet_Size_0', 'Outlet_Size_1', 'Outlet_Size_2', 'Outlet_Location_Type_0',
'Outlet_Location_Type_1', 'Outlet_Location_Type_2', 'Outlet_Type_0',
'Outlet_Type_1', 'Outlet_Type_2', 'Outlet_Type_3', 'New_Item_Type_0',
'New_Item_Type_1', 'New_Item_Type_2'], dtype='object')
```

[204]:
```
df.head()
```

[204]:
```
  Item_Identifier  Item_Weight  Item_Visibility  Item_Type  Item_MRP
Outlet_Identifier  Outlet_Establishment_Year  Item_Outlet_Sales  Outlet
Item_Fat_Content_0  Item_Fat_Content_1  Outlet_Size_0  Outlet_Size_1
Outlet_Size_2  Outlet_Location_Type_0  Outlet_Location_Type_1
Outlet_Location_Type_2  Outlet_Type_0  Outlet_Type_1  Outlet_Type_2
Outlet_Type_3  New_Item_Type_0  New_Item_Type_1  New_Item_Type_2
0           FDA15         9.30         0.016047          4  249.8092
OUT049                         1999         3735.1380       9                 1
0             0              1              0                        1
0                      0              0              1              0
0               0                 1                 0
1           DRC01         5.92         0.019278         14   48.2692
OUT018                         2009          443.4228       3                 0
1             0              1              0                        0
0                      1              0              0              1
0               1                 0                 0
2           FDN15        17.50         0.016760         10  141.6180
OUT049                         1999         2097.2700       9                 1
0             0              1              0                        1
0                      0              0              1              0
0               0                 1                 0
3           FDX07        19.20         0.066132          6  182.0950
OUT010                         1998          732.3800       0                 0
```

```
1              0              0              1                       0
0                       1              1              0              0
0                0                1                0
4           NCD19          8.93         0.066132          9    53.8614
OUT013                        1987          994.7052        1                   1
0              1              0              0                       0
0                       1              0              1              0
0                0                0                1
```

[205]:
```
df.describe()
```

[205]:
```
       Item_Weight  Item_Visibility    Item_Type      Item_MRP  \
Outlet_Establishment_Year  Item_Outlet_Sales       Outlet  Item_Fat_Content_0  \
Item_Fat_Content_1  Outlet_Size_0  Outlet_Size_1  Outlet_Size_2  \
Outlet_Location_Type_0  Outlet_Location_Type_1  Outlet_Location_Type_2  \
Outlet_Type_0  Outlet_Type_1  Outlet_Type_2  Outlet_Type_3  New_Item_Type_0  \
New_Item_Type_1  New_Item_Type_2
count  8523.000000      8523.000000  8523.000000  8523.000000
8523.000000        8523.000000  8523.000000         8523.000000
8523.000000    8523.000000    8523.000000    8523.000000               8523.000000
8523.000000              8523.000000    8523.000000    8523.000000    8523.000000
8523.000000      8523.000000       8523.000000       8523.000000
mean      12.857645         0.070213      7.226681   140.992782
1997.831867         2181.288914      4.722281            0.647307
0.352693       0.109351        0.327702        0.562947                 0.280183
0.326763                 0.393054        0.127068        0.654347       0.108882
0.109703         0.093746          0.718644         0.187610
std        4.226124         0.048742      4.209990     62.275067
8.371760         1706.499616      2.837201            0.477836              0.477836
0.312098       0.469403        0.496051                 0.449115
0.469057                 0.488457        0.333069        0.475609       0.311509
0.312538         0.291493          0.449687         0.390423
min        4.555000         0.003575      0.000000     31.290000
1985.000000           33.290000      0.000000            0.000000
0.000000       0.000000        0.000000        0.000000                 0.000000
0.000000                 0.000000        0.000000        0.000000       0.000000
0.000000         0.000000          0.000000         0.000000
25%        9.310000         0.033085      4.000000     93.826500
1987.000000          834.247400      2.000000            0.000000
0.000000       0.000000        0.000000        0.000000                 0.000000
0.000000                 0.000000        0.000000        0.000000       0.000000
0.000000         0.000000          0.000000         0.000000
50%       12.857645         0.062517      6.000000    143.012800
1999.000000         1794.331000      5.000000            1.000000
0.000000       0.000000        0.000000        1.000000                 0.000000
0.000000                 0.000000        0.000000        1.000000       0.000000
0.000000         0.000000          1.000000         0.000000
```

```
75%      16.000000         0.094585    10.000000   185.643700
2004.000000        3101.296400     7.000000            1.000000
1.000000       0.000000       1.000000       1.000000                 1.000000
1.000000                1.000000       0.000000       1.000000       0.000000
0.000000         0.000000         1.000000         0.000000
max      21.350000         0.328391    15.000000   266.888400
2009.000000       13086.964800     9.000000            1.000000
1.000000       1.000000       1.000000       1.000000                 1.000000
1.000000                1.000000       1.000000       1.000000       1.000000
1.000000         1.000000         1.000000         1.000000
```

8.1 9.Data Scaling

Using Standard scaler method

[206]:
```
X=df.drop(columns=['Outlet_Establishment_Year', 'Item_Identifier',␣
↪'Outlet_Identifier', 'Item_Outlet_Sales'])
Y = df['Item_Outlet_Sales']
```

8.1.1 9.1 Splitting the Data Set into Train and Test

[207]:
```
from sklearn.model_selection import train_test_split
df_train,df_test= train_test_split(df, test_size=0.3, random_state=100)
```

[208]:
```
df_train.shape
```

[208]:
```
(5966, 24)
```

[209]:
```
df_test.shape
```

[209]:
```
(2557, 24)
```

[210]:
```
num_var=['Item_Weight','Item_Visibility','Item_Type','Item_MRP','Outlet']
```

8.1.2 9.2 Scalling the Data using Standard Scale.

[211]:
```
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

[212]:
```
df_train[num_var]=scaler.fit_transform(df_train[num_var])
```

[213]:
```
df_train.describe()
```

[213]:
```
        Item_Weight  Item_Visibility     Item_Type       Item_MRP
Outlet_Establishment_Year  Item_Outlet_Sales        Outlet  Item_Fat_Content_0
```

```
Item_Fat_Content_1  Outlet_Size_0  Outlet_Size_1  Outlet_Size_2
Outlet_Location_Type_0  Outlet_Location_Type_1  Outlet_Location_Type_2
Outlet_Type_0  Outlet_Type_1  Outlet_Type_2  Outlet_Type_3  New_Item_Type_0
New_Item_Type_1  New_Item_Type_2
count  5.966000e+03     5.966000e+03  5.966000e+03  5.966000e+03
5966.000000        5966.000000  5.966000e+03        5966.000000
5966.000000    5966.000000    5966.000000    5966.000000             5966.000000
5966.000000             5966.000000    5966.000000    5966.000000    5966.000000
5966.000000      5966.000000      5966.000000      5966.000000
mean  -1.381545e-16    -4.525750e-17  5.478539e-17  2.381974e-18
1997.739021        2181.443146 -6.907724e-17            0.641468
0.358532       0.107107       0.327187       0.565706                0.282937
0.325846                0.391217       0.129232       0.653201       0.104425
0.113141         0.090178         0.718404         0.191418
std    1.000084e+00     1.000084e+00  1.000084e+00  1.000084e+00
8.355920        1715.972354  1.000084e+00           0.479609
0.479609       0.309275       0.469226       0.495706                0.450464
0.468730                0.488064       0.335485       0.475991       0.305837
0.316792         0.286460         0.449815         0.393450
min   -1.966442e+00    -1.382151e+00 -1.711397e+00 -1.749490e+00
1985.000000          33.290000 -1.670874e+00            0.000000
0.000000       0.000000       0.000000       0.000000                0.000000
0.000000                0.000000       0.000000       0.000000       0.000000
0.000000         0.000000         0.000000         0.000000
25%   -8.186739e-01    -7.545508e-01 -7.586424e-01 -7.551947e-01
1987.000000         820.265600 -9.665153e-01            0.000000
0.000000       0.000000       0.000000       0.000000                0.000000
0.000000                0.000000       0.000000       0.000000       0.000000
0.000000         0.000000         0.000000         0.000000
50%    2.465084e-03    -1.496658e-01 -2.822651e-01  3.000450e-02
1999.000000        1780.349200  9.002235e-02            1.000000
0.000000       0.000000       0.000000       1.000000                0.000000
0.000000                0.000000       0.000000       1.000000       0.000000
0.000000         0.000000         1.000000         0.000000
75%    7.476498e-01     5.055199e-01  6.704895e-01  7.161800e-01
2004.000000        3124.432950  7.943808e-01            1.000000
1.000000       0.000000       1.000000       1.000000                1.000000
1.000000                1.000000       0.000000       1.000000       0.000000
0.000000         0.000000         1.000000         0.000000
max    2.016360e+00     5.199650e+00  1.861433e+00  2.016113e+00
2009.000000       13086.964800  1.498739e+00            1.000000
1.000000       1.000000       1.000000       1.000000                1.000000
1.000000                1.000000       1.000000       1.000000       1.000000
1.000000         1.000000         1.000000         1.000000
```

[214]:

```python
X_train=df_train.drop(columns=['Outlet_Establishment_Year', 'Item_Identifier',␣
↪'Outlet_Identifier', 'Item_Outlet_Sales'])
y_train = df_train['Item_Outlet_Sales']
```

[215]:
```python
X_train.describe()
```

[215]:
```
        Item_Weight  Item_Visibility     Item_Type      Item_MRP        Outlet
Item_Fat_Content_0  Item_Fat_Content_1  Outlet_Size_0  Outlet_Size_1
Outlet_Size_2  Outlet_Location_Type_0  Outlet_Location_Type_1
Outlet_Location_Type_2  Outlet_Type_0  Outlet_Type_1  Outlet_Type_2
Outlet_Type_3  New_Item_Type_0  New_Item_Type_1  New_Item_Type_2
count  5.966000e+03     5.966000e+03  5.966000e+03  5.966000e+03  5.966000e+03
5966.000000         5966.000000     5966.000000     5966.000000     5966.000000
5966.000000             5966.000000             5966.000000     5966.000000
5966.000000     5966.000000     5966.000000      5966.000000      5966.000000
5966.000000
mean  -1.381545e-16    -4.525750e-17  5.478539e-17  2.381974e-18 -6.907724e-17
0.641468            0.358532        0.107107        0.327187        0.565706
0.282937                0.325846                0.391217        0.129232
0.653201        0.104425        0.113141         0.090178         0.718404
0.191418
std    1.000084e+00     1.000084e+00  1.000084e+00  1.000084e+00  1.000084e+00
0.479609            0.479609        0.309275        0.469226        0.495706
0.450464                0.468730                0.488064        0.335485
0.475991        0.305837        0.316792         0.286460         0.449815
0.393450
min   -1.966442e+00    -1.382151e+00 -1.711397e+00 -1.749490e+00 -1.670874e+00
0.000000            0.000000        0.000000        0.000000        0.000000
0.000000                0.000000                0.000000        0.000000
0.000000        0.000000        0.000000         0.000000         0.000000
0.000000
25%   -8.186739e-01    -7.545508e-01 -7.586424e-01 -7.551947e-01 -9.665153e-01
0.000000            0.000000        0.000000        0.000000        0.000000
0.000000                0.000000                0.000000        0.000000
0.000000        0.000000        0.000000         0.000000         0.000000
0.000000
50%    2.465084e-03    -1.496658e-01 -2.822651e-01  3.000450e-02  9.002235e-02
1.000000            0.000000        0.000000        0.000000        1.000000
0.000000                0.000000                0.000000        0.000000
1.000000        0.000000        0.000000         0.000000         1.000000
0.000000
75%    7.476498e-01     5.055199e-01  6.704895e-01  7.161800e-01  7.943808e-01
1.000000            1.000000        0.000000        1.000000        1.000000
1.000000                1.000000                1.000000        0.000000
1.000000        0.000000        0.000000         0.000000         1.000000
0.000000
max    2.016360e+00     5.199650e+00  1.861433e+00  2.016113e+00  1.498739e+00
```

```
1.000000             1.000000       1.000000       1.000000       1.000000
1.000000                 1.000000                 1.000000       1.000000
1.000000       1.000000       1.000000        1.000000        1.000000
1.000000
```

9 10. Data Modelling

9.0.1 10.1 Data Modellinf Using Stats Model.

[216]:
```python
import statsmodels.api as sm
X_train=sm.add_constant(X_train)
lm=sm.OLS(y_train,X_train).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.568
Model:                            OLS   Adj. R-squared:                  0.566
Method:                 Least Squares   F-statistic:                     520.5
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:37:10   Log-Likelihood:                -50398.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5950   BIC:                         1.009e+05
Df Model:                          15
Covariance Type:            nonrobust
================================================================================
==========
                             coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                    755.4359     23.135     32.653      0.000     710.083
800.789
Item_Weight                0.3909     14.675      0.027      0.979     -28.378
29.159
Item_Visibility          -12.3430     15.342     -0.805      0.421     -42.418
17.732
Item_Type                  6.2208     15.525      0.401      0.689     -24.214
36.656
Item_MRP                 976.9814     14.660     66.643      0.000     948.243
1005.720
Outlet                   -54.5400     33.686     -1.619      0.105    -120.577
11.497
Item_Fat_Content_0       336.3139     18.694     17.991      0.000     299.668
372.960
Item_Fat_Content_1       419.1221     22.161     18.913      0.000     375.679
462.565
```

```
Outlet_Size_0              205.1663     77.770      2.638      0.008      52.708
357.624
Outlet_Size_1              334.1411     44.895      7.443      0.000     246.130
422.152
Outlet_Size_2              216.1285     41.037      5.267      0.000     135.681
296.576
Outlet_Location_Type_0     287.6876     45.065      6.384      0.000     199.344
376.032
Outlet_Location_Type_1     282.2001     49.929      5.652      0.000     184.322
380.078
Outlet_Location_Type_2     185.5483     55.002      3.373      0.001      77.725
293.372
Outlet_Type_0            -1526.5406     62.373    -24.474      0.000   -1648.815
-1404.266
Outlet_Type_1              446.3002     57.473      7.765      0.000     333.633
558.967
Outlet_Type_2               29.9547     57.539      0.521      0.603     -82.843
142.753
Outlet_Type_3             1805.7217     58.688     30.768      0.000    1690.671
1920.772
New_Item_Type_0            237.4033     38.038      6.241      0.000     162.836
311.971
New_Item_Type_1            262.3451     25.053     10.472      0.000     213.233
311.457
New_Item_Type_2            255.6875     30.183      8.471      0.000     196.519
314.856
==============================================================================
Omnibus:                      685.184   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1695.293
Skew:                           0.665   Prob(JB):                         0.00
Kurtosis:                       5.247   Cond. No.                     5.32e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 7.16e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

- By modelling the X_train set we got an r_score of 56.7
- we can minimize the feature by using the RFE model

9.0.2 10.2 RFE

[217]:
```
# Create a LinearRegression model
lm = LinearRegression()

# RFE with 15 features
```

```
rfe1 = RFE(estimator=lm, n_features_to_select=15)

# Fit with 15 features
rfe1.fit(X_train, y_train)
```

[217]: RFE(estimator=LinearRegression(), n_features_to_select=15)

[218]:
```
print(rfe1.support_)
print(rfe1.ranking_)
```

```
[False False False False  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True False False]
[7 6 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 5]
```

[219]:
```
col=X_train.columns[rfe1.support_]
```

[220]:
```
col
```

[220]:
```
Index(['Item_MRP', 'Outlet', 'Item_Fat_Content_0', 'Item_Fat_Content_1',
'Outlet_Size_0', 'Outlet_Size_1', 'Outlet_Size_2', 'Outlet_Location_Type_0',
'Outlet_Location_Type_1', 'Outlet_Location_Type_2', 'Outlet_Type_0',
'Outlet_Type_1', 'Outlet_Type_2', 'Outlet_Type_3', 'New_Item_Type_0'],
dtype='object')
```

[221]:
```
print(X_train.columns[~rfe1.support_])
```

```
Index(['const', 'Item_Weight', 'Item_Visibility', 'Item_Type',
'New_Item_Type_1', 'New_Item_Type_2'], dtype='object')
```

[222]:
```
X_train_rfe.shape
```

[222]: (5966, 12)

[223]:
```
X_train_rfe=X_train[col]
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     710.1
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:39:05   Log-Likelihood:                -50398.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5954   BIC:                         1.009e+05
Df Model:                          11
Covariance Type:            nonrobust
```

```
================================================================================
==========
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                 862.5695     25.140     34.310      0.000     813.285
911.854
Item_MRP              977.3627     14.642     66.749      0.000     948.658
1006.067
Outlet                -54.5830     33.677     -1.621      0.105    -120.601
11.435
Item_Fat_Content_0    390.0565     19.223     20.291      0.000     352.372
427.741
Item_Fat_Content_1    472.5130     20.651     22.881      0.000     432.030
512.996
Outlet_Size_0         240.6030     78.592      3.061      0.002      86.534
394.671
Outlet_Size_1         370.3660     44.297      8.361      0.000     283.527
457.205
Outlet_Size_2         251.6006     40.388      6.230      0.000     172.426
330.775
Outlet_Location_Type_0   323.1645     45.415      7.116      0.000     234.135
412.194
Outlet_Location_Type_1   318.0149     50.623      6.282      0.000     218.775
417.254
Outlet_Location_Type_2   221.3901     54.282      4.079      0.000     114.978
327.802
Outlet_Type_0        -1507.3223     61.688    -24.434      0.000   -1628.254
-1386.391
Outlet_Type_1         475.9901     56.634      8.405      0.000     364.967
587.013
Outlet_Type_2          59.0594     57.887      1.020      0.308     -54.420
172.538
Outlet_Type_3        1834.8424     58.935     31.134      0.000    1719.309
1950.376
New_Item_Type_0       -17.5658     51.929     -0.338      0.735    -119.365
84.234
==============================================================================
Omnibus:                      686.505   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1703.166
Skew:                           0.665   Prob(JB):                         0.00
Kurtosis:                       5.254   Cond. No.                     4.50e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
[2] The smallest eigenvalue is 8.38e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

- By using RFE the R_score obtained is 0.567 which is not different from the R_Score we obtained.
- Therefore we can go ahead with RFE selected features.

9.0.3 10.3 Manual Feature Elimination

- Here there some Features with High P_values. These Features need to eliminated
- We can eliminated the Features by comparing the VIF values and P_values

9.0.4 10.4 VIF

[224]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in␣
 ↪range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[224]:

	Features	VIF
3	Item_Fat_Content_0	inf
4	Item_Fat_Content_1	inf
5	Outlet_Size_0	inf
6	Outlet_Size_1	inf
7	Outlet_Size_2	inf
8	Outlet_Location_Type_0	inf
9	Outlet_Location_Type_1	inf
10	Outlet_Location_Type_2	inf
11	Outlet_Type_0	inf
12	Outlet_Type_1	inf
13	Outlet_Type_2	inf
14	Outlet_Type_3	inf
2	Outlet	5.30
15	New_Item_Type_0	1.03
1	Item_MRP	1.00
0	const	0.00

- We need to compare the P_values and VIF values
- Feature having P_vlaue > 0.05 and VIF value > 5 needs to be eliminated.

[225]:
```
X_train_rfe.drop('New_Item_Type_0', axis = 1, inplace = True)
```

[226]:
```
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     781.3
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:44:38   Log-Likelihood:                -50398.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5955   BIC:                         1.009e+05
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
==========
                             coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                    862.1058     25.101     34.345      0.000     812.899
911.313
Item_MRP                 977.5777     14.627     66.832      0.000     948.903
1006.253
Outlet                   -54.6343     33.674     -1.622      0.105    -120.647
11.378
Item_Fat_Content_0       388.9075     18.920     20.556      0.000     351.818
425.997
Item_Fat_Content_1       473.1984     20.549     23.027      0.000     432.914
513.483
Outlet_Size_0            240.7171     78.585      3.063      0.002      86.661
394.773
Outlet_Size_1            369.9768     44.279      8.356      0.000     283.174
456.780
Outlet_Size_2            251.4119     40.381      6.226      0.000     172.251
330.573
Outlet_Location_Type_0   323.2046     45.411      7.117      0.000     234.182
412.227
Outlet_Location_Type_1   317.8016     50.615      6.279      0.000     218.577
417.026
Outlet_Location_Type_2   221.0997     54.271      4.074      0.000     114.709
327.490
Outlet_Type_0          -1507.7846     61.669    -24.450      0.000   -1628.678
-1386.891
Outlet_Type_1            475.6676     56.622      8.401      0.000     364.669
586.667
Outlet_Type_2             59.2193     57.880      1.023      0.306     -54.247
172.686
Outlet_Type_3           1835.0035     58.928     31.140      0.000    1719.483
1950.524
==============================================================================
```

```
Omnibus:                      685.956   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1701.565
Skew:                           0.665   Prob(JB):                         0.00
Kurtosis:                       5.253   Cond. No.                     3.61e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 1.3e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[227]:
```
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in␣
 ↪range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[227]:

	Features	VIF
3	Item_Fat_Content_0	inf
4	Item_Fat_Content_1	inf
5	Outlet_Size_0	inf
6	Outlet_Size_1	inf
7	Outlet_Size_2	inf
8	Outlet_Location_Type_0	inf
9	Outlet_Location_Type_1	inf
10	Outlet_Location_Type_2	inf
11	Outlet_Type_0	inf
12	Outlet_Type_1	inf
13	Outlet_Type_2	inf
14	Outlet_Type_3	inf
2	Outlet	5.3
1	Item_MRP	1.0
0	const	0.0

[228]:
```
X_train_rfe.drop('Outlet_Type_2', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     781.3
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
```

```
Time:                        20:45:22   Log-Likelihood:                -50398.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5955   BIC:                         1.009e+05
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
==========
                            coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                   889.4378     46.812     19.000      0.000     797.670
981.206
Item_MRP                977.5777     14.627     66.832      0.000     948.903
1006.253
Outlet                  -54.6343     33.674     -1.622      0.105    -120.647
11.378
Item_Fat_Content_0      402.5735     27.269     14.763      0.000     349.116
456.031
Item_Fat_Content_1      486.8644     28.588     17.030      0.000     430.822
542.907
Outlet_Size_0           249.8277     84.136      2.969      0.003      84.891
414.764
Outlet_Size_1           379.0875     39.164      9.679      0.000     302.311
455.864
Outlet_Size_2           260.5226     40.923      6.366      0.000     180.300
340.746
Outlet_Location_Type_0   332.3152     50.110      6.632      0.000     234.081
430.550
Outlet_Location_Type_1   326.9122     54.457      6.003      0.000     220.157
433.667
Outlet_Location_Type_2   230.2104     51.120      4.503      0.000     129.997
330.424
Outlet_Type_0         -1567.0039     98.528    -15.904      0.000   -1760.154
-1373.854
Outlet_Type_1           416.4482    105.772      3.937      0.000     209.097
623.799
Outlet_Type_3          1775.7842     67.085     26.471      0.000    1644.273
1907.295
==============================================================================
Omnibus:                      685.956   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1701.565
Skew:                           0.665   Prob(JB):                         0.00
Kurtosis:                       5.253   Cond. No.                     3.67e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

```
specified.
[2] The smallest eigenvalue is 1.26e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[229]:
```python
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in␣
 ↪range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[229]:

	Features	VIF
3	Item_Fat_Content_0	inf
4	Item_Fat_Content_1	inf
5	Outlet_Size_0	inf
6	Outlet_Size_1	inf
7	Outlet_Size_2	inf
8	Outlet_Location_Type_0	inf
9	Outlet_Location_Type_1	inf
10	Outlet_Location_Type_2	inf
12	Outlet_Type_1	11.85
2	Outlet	5.30
11	Outlet_Type_0	5.11
13	Outlet_Type_3	2.11
1	Item_MRP	1.00
0	const	0.00

[230]:
```python
X_train_rfe.drop('Outlet', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     867.5
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:45:53   Log-Likelihood:                -50399.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5956   BIC:                         1.009e+05
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
==========
                            coef    std err          t      P>|t|      [0.025
0.975]
```

```
------------------------------------------------------------------------------------
----------
const                    904.8766     45.841     19.740      0.000     815.012
994.741
Item_MRP                 977.5654     14.629     66.822      0.000     948.886
1006.244
Item_Fat_Content_0       410.2350     26.861     15.273      0.000     357.578
462.892
Item_Fat_Content_1       494.6416     28.187     17.549      0.000     439.385
549.898
Outlet_Size_0            299.8722     78.289      3.830      0.000     146.398
453.346
Outlet_Size_1            352.1653     35.480      9.926      0.000     282.611
421.719
Outlet_Size_2            252.8392     40.653      6.219      0.000     173.144
332.534
Outlet_Location_Type_0   292.6989     43.765      6.688      0.000     206.904
378.494
Outlet_Location_Type_1   344.6241     53.359      6.459      0.000     240.022
449.226
Outlet_Location_Type_2   267.5536     45.652      5.861      0.000     178.060
357.047
Outlet_Type_0          -1528.5210     95.643    -15.981      0.000   -1716.017
-1341.025
Outlet_Type_1            377.9640    103.092      3.666      0.000     175.866
580.062
Outlet_Type_3           1737.3028     62.761     27.681      0.000    1614.269
1860.337
==============================================================================
Omnibus:                      687.072   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1705.812
Skew:                           0.666   Prob(JB):                         0.00
Kurtosis:                       5.256   Cond. No.                     3.37e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 1.48e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[231]:
```
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in␣
 ↪range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
```

```
vif
```

[231]:

```
                   Features    VIF
2        Item_Fat_Content_0    inf
3        Item_Fat_Content_1    inf
4             Outlet_Size_0    inf
5             Outlet_Size_1    inf
6             Outlet_Size_2    inf
7    Outlet_Location_Type_0    inf
8    Outlet_Location_Type_1    inf
9    Outlet_Location_Type_2    inf
11            Outlet_Type_1  11.26
10            Outlet_Type_0   4.81
12            Outlet_Type_3   1.85
1                  Item_MRP   1.00
0                     const   0.00
```

[232]:

```
X_train_rfe.drop('Outlet_Size_2', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     867.5
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:46:38   Log-Likelihood:                -50399.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5956   BIC:                         1.009e+05
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
==========
                             coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                   1042.7889     48.475     21.512      0.000     947.761
1137.817
Item_MRP                 977.5654     14.629     66.822      0.000     948.886
1006.244
Item_Fat_Content_0       479.1912     27.972     17.131      0.000     424.355
534.027
Item_Fat_Content_1       563.5978     29.288     19.243      0.000     506.183
621.013
Outlet_Size_0             47.0330    102.760      0.458      0.647    -154.413
```

```
248.479
Outlet_Size_1              99.3260     62.331      1.594      0.111     -22.865
221.517
Outlet_Location_Type_0    338.6697     44.170      7.667      0.000     252.081
425.259
Outlet_Location_Type_1    390.5949     49.257      7.930      0.000     294.033
487.157
Outlet_Location_Type_2    313.5244     50.519      6.206      0.000     214.489
412.559
Outlet_Type_0           -1528.5210     95.643    -15.981      0.000   -1716.017
-1341.025
Outlet_Type_1             377.9640    103.092      3.666      0.000     175.866
580.062
Outlet_Type_3            1737.3028     62.761     27.681      0.000    1614.269
1860.337
==============================================================================
Omnibus:                      687.072   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1705.812
Skew:                           0.666   Prob(JB):                         0.00
Kurtosis:                       5.256   Cond. No.                     1.96e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 3.86e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[233]:

```
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[233]:

	Features	VIF
2	Item_Fat_Content_0	inf
3	Item_Fat_Content_1	inf
6	Outlet_Location_Type_0	inf
7	Outlet_Location_Type_1	inf
8	Outlet_Location_Type_2	inf
10	Outlet_Type_1	11.26
9	Outlet_Type_0	4.81
4	Outlet_Size_0	4.72
5	Outlet_Size_1	4.00
11	Outlet_Type_3	1.85

```
1                  Item_MRP   1.00
0                     const   0.00
```

[234]:

```
X_train_rfe.drop('Outlet_Size_0', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     976.1
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:47:14   Log-Likelihood:                -50399.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5957   BIC:                         1.009e+05
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
==========
                             coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                   1035.1714     45.525     22.738      0.000     945.925
1124.417
Item_MRP                 977.5168     14.628     66.825      0.000     948.840
1006.193
Item_Fat_Content_0       475.4759     26.767     17.764      0.000     423.003
527.948
Item_Fat_Content_1       559.6955     28.018     19.977      0.000     504.771
614.620
Outlet_Size_1             90.7102     59.416      1.527      0.127     -25.767
207.188
Outlet_Location_Type_0   329.1609     38.977      8.445      0.000     252.753
405.569
Outlet_Location_Type_1   372.4707     29.294     12.715      0.000     315.044
429.897
Outlet_Location_Type_2   333.5398     25.292     13.188      0.000     283.958
383.121
Outlet_Type_0          -1522.8152     94.821    -16.060      0.000   -1708.699
-1336.931
Outlet_Type_1            407.4875     80.413      5.067      0.000     249.848
565.127
Outlet_Type_3           1737.3018     62.757     27.683      0.000    1614.276
1860.328
==============================================================================
```

```
Omnibus:                      687.006   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1706.153
Skew:                           0.666   Prob(JB):                         0.00
Kurtosis:                       5.256   Cond. No.                     2.19e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 3.07e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[235]:

```python
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in␣
 ↪range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[235]:

	Features	VIF
2	Item_Fat_Content_0	inf
3	Item_Fat_Content_1	inf
5	Outlet_Location_Type_0	inf
6	Outlet_Location_Type_1	inf
7	Outlet_Location_Type_2	inf
9	Outlet_Type_1	6.85
8	Outlet_Type_0	4.73
4	Outlet_Size_1	3.63
10	Outlet_Type_3	1.85
1	Item_MRP	1.00
0	const	0.00

[236]:

```python
X_train_rfe.drop('Item_Fat_Content_0', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     976.1
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:48:04   Log-Likelihood:                -50399.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5957   BIC:                         1.009e+05
Df Model:                           8
```

```
Covariance Type:            nonrobust
================================================================================
==========
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                   1391.7784     63.076     22.065      0.000    1268.127
1515.430
Item_MRP                 977.5168     14.628     66.825      0.000     948.840
1006.193
Item_Fat_Content_1        84.2196     30.502      2.761      0.006      24.425
144.014
Outlet_Size_1             90.7102     59.416      1.527      0.127     -25.767
207.188
Outlet_Location_Type_0   448.0299     43.775     10.235      0.000     362.215
533.845
Outlet_Location_Type_1   491.3397     32.103     15.305      0.000     428.406
554.273
Outlet_Location_Type_2   452.4088     26.560     17.033      0.000     400.341
504.476
Outlet_Type_0          -1522.8152     94.821    -16.060      0.000   -1708.699
-1336.931
Outlet_Type_1            407.4875     80.413      5.067      0.000     249.848
565.127
Outlet_Type_3           1737.3018     62.757     27.683      0.000    1614.276
1860.328
==============================================================================
Omnibus:                      687.006   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1706.153
Skew:                           0.666   Prob(JB):                         0.00
Kurtosis:                       5.256   Cond. No.                     1.16e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 9.2e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[237]:
```python
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[237]:

```
                  Features   VIF
4  Outlet_Location_Type_0   inf
5  Outlet_Location_Type_1   inf
6  Outlet_Location_Type_2   inf
8           Outlet_Type_1  6.85
7           Outlet_Type_0  4.73
3           Outlet_Size_1  3.63
9           Outlet_Type_3  1.85
1                Item_MRP  1.00
2       Item_Fat_Content_1  1.00
0                   const  0.00
```

[238]:

```
X_train_rfe.drop('Outlet_Location_Type_1', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     976.1
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:49:01   Log-Likelihood:                -50399.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5957   BIC:                         1.009e+05
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
==========
                             coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                   1883.1181     85.100     22.128      0.000    1716.292
2049.944
Item_MRP                 977.5168     14.628     66.825      0.000     948.840
1006.193
Item_Fat_Content_1        84.2196     30.502      2.761      0.006      24.425
144.014
Outlet_Size_1             90.7102     59.416      1.527      0.127     -25.767
207.188
Outlet_Location_Type_0   -43.3098     47.282     -0.916      0.360    -136.000
49.380
Outlet_Location_Type_2   -38.9309     47.705     -0.816      0.414    -132.449
54.587
Outlet_Type_0          -1522.8152     94.821    -16.060      0.000   -1708.699
-1336.931
```

```
Outlet_Type_1           407.4875     80.413      5.067      0.000     249.848
565.127
Outlet_Type_3          1737.3018     62.757     27.683      0.000    1614.276
1860.328
==============================================================================
Omnibus:                      687.006   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1706.153
Skew:                           0.666   Prob(JB):                         0.00
Kurtosis:                       5.256   Cond. No.                         14.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[239]:

```python
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[239]:

	Features	VIF
0	const	33.86
7	Outlet_Type_1	6.85
6	Outlet_Type_0	4.73
3	Outlet_Size_1	3.63
5	Outlet_Location_Type_2	2.53
4	Outlet_Location_Type_0	2.12
8	Outlet_Type_3	1.85
1	Item_MRP	1.00
2	Item_Fat_Content_1	1.00

[240]:

```python
X_train_rfe.drop('Outlet_Location_Type_2', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     1115.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        20:49:44   Log-Likelihood:                -50400.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5958   BIC:                         1.009e+05
```

```
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
==========
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                1850.5481     75.159     24.622      0.000    1703.210
1997.886
Item_MRP              977.4816     14.628     66.824      0.000     948.806
1006.157
Item_Fat_Content_1     83.9377     30.499      2.752      0.006      24.149
143.727
Outlet_Size_1          84.4511     58.918      1.433      0.152     -31.049
199.951
Outlet_Location_Type_0   -25.8239     42.147     -0.613      0.540    -108.448
56.800
Outlet_Type_0        -1518.6726     94.683    -16.040      0.000   -1704.285
-1333.060
Outlet_Type_1         428.9295     75.998      5.644      0.000     279.946
577.913
Outlet_Type_3        1737.3002     62.755     27.684      0.000    1614.278
1860.323
==============================================================================
Omnibus:                      687.361   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1707.529
Skew:                           0.666   Prob(JB):                         0.00
Kurtosis:                       5.257   Cond. No.                         13.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[241]:
```python
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in␣
 ↪range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[241]:

	Features	VIF
0	const	26.41
6	Outlet_Type_1	6.12
5	Outlet_Type_0	4.72

```
3          Outlet_Size_1   3.57
7          Outlet_Type_3   1.85
4  Outlet_Location_Type_0   1.69
1               Item_MRP   1.00
2      Item_Fat_Content_1   1.00
```

[243]:
```python
X_train_rfe.drop('Outlet_Location_Type_0', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     1301.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        21:03:17   Log-Likelihood:                -50400.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5959   BIC:                         1.009e+05
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
const               1871.0881     67.265     27.817      0.000    1739.225
2002.951
Item_MRP             977.4403     14.627     66.826      0.000     948.767
1006.114
Item_Fat_Content_1    84.0081     30.497      2.755      0.006      24.223
143.793
Outlet_Size_1         63.8843     48.418      1.319      0.187     -31.032
158.801
Outlet_Type_0      -1551.7651     77.763    -19.955      0.000   -1704.208
-1399.322
Outlet_Type_1        403.1090     63.240      6.374      0.000     279.136
527.082
Outlet_Type_3       1737.3006     62.752     27.685      0.000    1614.285
1860.317
==============================================================================
Omnibus:                      687.427   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1709.667
Skew:                           0.666   Prob(JB):                         0.00
Kurtosis:                       5.260   Cond. No.                         11.2
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[244]:
```python
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in␣
 ↪range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[244]:

	Features	VIF
0	const	21.16
5	Outlet_Type_1	4.24
4	Outlet_Type_0	3.18
3	Outlet_Size_1	2.41
6	Outlet_Type_3	1.85
1	Item_MRP	1.00
2	Item_Fat_Content_1	1.00

[245]:
```python
X_train_rfe.drop('Outlet_Size_1', axis = 1, inplace = True)
X_train_rfe=sm.add_constant(X_train_rfe)
lm=sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Item_Outlet_Sales   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     1561.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        21:03:50   Log-Likelihood:                -50401.
No. Observations:                5966   AIC:                         1.008e+05
Df Residuals:                    5960   BIC:                         1.009e+05
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
const               1935.0916     46.602     41.524      0.000    1843.734
2026.449
Item_MRP             977.5507     14.627     66.830      0.000     948.876
1006.226
```

```
Item_Fat_Content_1     83.6891     30.498      2.744      0.006      23.902
143.476
Outlet_Type_0      -1615.6529     60.852    -26.551      0.000   -1734.945
-1496.361
Outlet_Type_1        349.9395     48.742      7.179      0.000     254.388
445.491
Outlet_Type_3       1737.2986     62.756     27.684      0.000    1614.275
1860.322
==============================================================================
Omnibus:                      685.068   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1702.058
Skew:                           0.664   Prob(JB):                         0.00
Kurtosis:                       5.255   Cond. No.                         8.48
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[246]:

```python
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[246]:

	Features	VIF
0	const	10.15
4	Outlet_Type_1	2.52
3	Outlet_Type_0	1.95
5	Outlet_Type_3	1.85
1	Item_MRP	1.00
2	Item_Fat_Content_1	1.00

9.1 11 . Applying the trained model on the Test set

[247]:

```python
X_train=X_train_rfe
```

[249]:

```python
X_train
```

[249]:

```
      const  Item_MRP  Item_Fat_Content_1  Outlet_Type_0  Outlet_Type_1
Outlet_Type_3
4122    1.0 -0.215801                   0              0              1
0
6590    1.0  0.227862                   0              0              1
0
```

```
5460     1.0  0.125566                   0              0              0
1
4541     1.0 -0.259415                   0              0              1
0
4186     1.0 -0.897946                   0              0              1
0
…     …       …                 …            …            …
…
350      1.0 -0.588531                   1              0              0
0
79       1.0  1.306773                   0              0              1
0
8039     1.0  1.875508                   0              0              1
0
6936     1.0  1.082016                   0              0              0
1
5640     1.0  1.273686                   1              1              0
0

[5966 rows x 6 columns]
```

[250]:
```
df_test[num_var]=scaler.transform(df_test[num_var])
```

[251]:
```
df_test
```

[251]:
```
     Item_Identifier  Item_Weight  Item_Visibility  Item_Type  Item_MRP
Outlet_Identifier  Outlet_Establishment_Year  Item_Outlet_Sales    Outlet
Item_Fat_Content_0  Item_Fat_Content_1  Outlet_Size_0  Outlet_Size_1
Outlet_Size_2  Outlet_Location_Type_0  Outlet_Location_Type_1
Outlet_Location_Type_2  Outlet_Type_0  Outlet_Type_1  Outlet_Type_2
Outlet_Type_3  New_Item_Type_0  New_Item_Type_1  New_Item_Type_2
3454           FDD14     1.862218         2.060522  -0.996831  0.682597
OUT013                      1987          4426.2384 -1.318695
1                 0              1              0              0
0                   0                       1              0              1
0              0               0                1               0
3386           FDT37     0.308937        -0.721049  -0.996831  1.846756
OUT017                      2007          4845.0266 -0.966515
1                 0              0              0              1
0                   1                       0              0              1
0              0               0                1               0
235            DRM47     0.002465        -0.553074  -0.044076  0.832295
OUT027                      1985          2293.0152  0.090022
1                 0              0              1              0
0                   0                       1              0              0
0              1               1                0               0
7201           NCM53     1.399791        -0.371468   0.194112 -0.523533
```

```
OUT017                      2007         1065.2800 -0.966515
1                   0              0              0              1
0                       1                      0              0              1
0             0                0                0                1
7782          FDS31      0.059938         -0.540250  -0.282265  0.631491
OUT046                      1997         2345.6134  1.146560
0                   1              0              0              1
1                       0                      0              0              1
0             0                0                1                0
…             …          …              …        …        …
…                        …              …        …                 …
…           …            …            …                     …
…                    …            …            …            …
…             …             …              …
7168          FDR22      1.542076         -0.812243   1.385055 -0.460724
OUT010                      1998          223.7088 -1.670874
0                   1              0              0              1
0                       0                      1              1              0
0             0                0                1                0
1325          FDN56     -1.751828          0.762358  -0.282265  0.054758
OUT035                      2004          288.9572  0.442202
0                   1              0              0              1
0                       1                      0              0              1
0             0                0                1                0
2079          NCY18     -1.319044         -0.375486   0.432301  0.544686
OUT010                      1998          525.3162 -1.670874
1                   0              0              0              1
0                       0                      1              1              0
0             0                0                0                1
6552          FDA07     -1.256201         -0.813541  -0.282265 -0.279116
OUT045                      2002         2082.6224  0.794381
0                   1              0              0              1
0                       1                      0              0              1
0             0                0                1                0
7125          FDW09      0.190366         -0.915934   1.385055 -0.992202
OUT017                      2007          713.0718 -0.966515
0                   1              0              0              1
0                       1                      0              0              1
0             0                0                1                0

[2557 rows x 24 columns]
```

9.1.1 11.1 Dividing the Test Set

[252]:
```
X_test=df_test.drop(columns=['Outlet_Establishment_Year', 'Item_Identifier',␣
↪'Outlet_Identifier', 'Item_Outlet_Sales'])
y_test = df_test['Item_Outlet_Sales']
```

[253]:
```
X_test_new=sm.add_constant(X_test)
X_test_new_1=X_test_new[X_train.columns]
```

[254]:
```
X_test_new_1
```

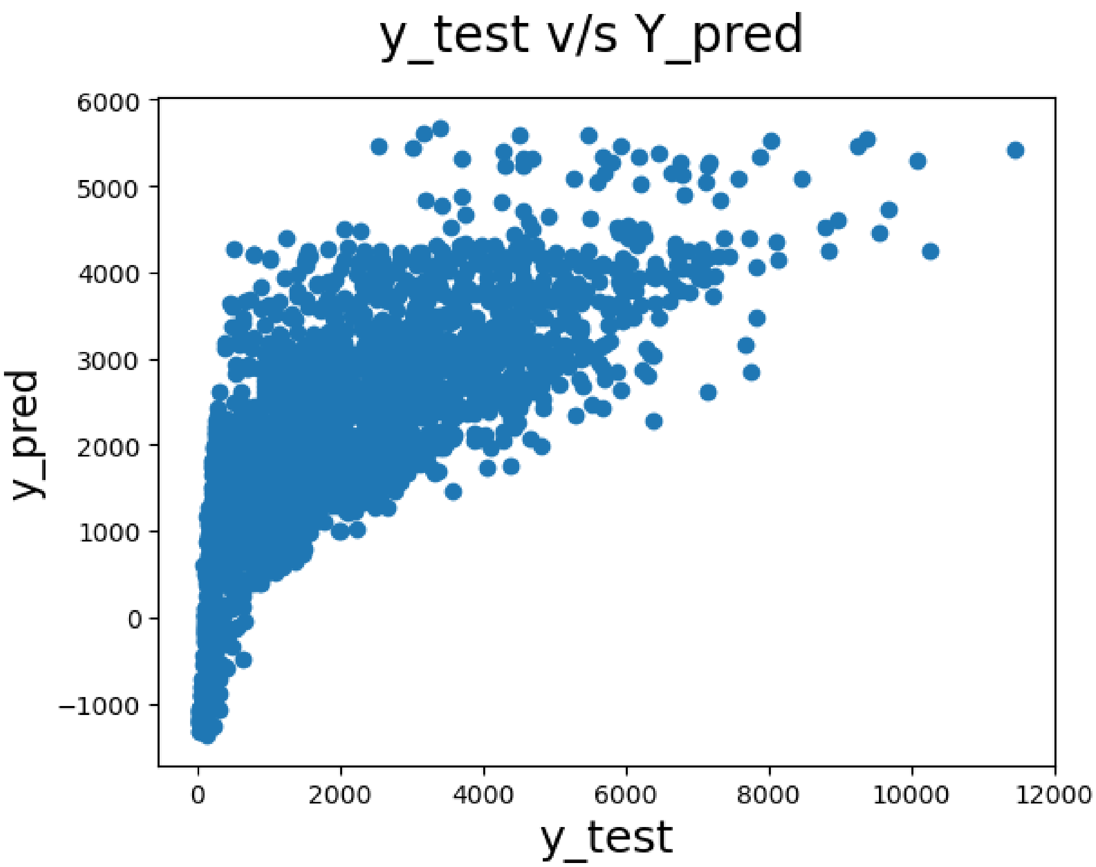
[254]:
```
      const   Item_MRP  Item_Fat_Content_1  Outlet_Type_0  Outlet_Type_1
Outlet_Type_3
3454    1.0  0.682597                   0              0              1
0
3386    1.0  1.846756                   0              0              1
0
235     1.0  0.832295                   0              0              0
1
7201    1.0 -0.523533                   0              0              1
0
7782    1.0  0.631491                   1              0              1
0
…      …        …                  …            …             …
…
7168    1.0 -0.460724                   1              1              0
0
1325    1.0  0.054758                   1              0              1
0
2079    1.0  0.544686                   0              1              0
0
6552    1.0 -0.279116                   1              0              1
0
7125    1.0 -0.992202                   1              0              1
0

[2557 rows x 6 columns]
```

9.1.2 11.2 Predicting the model.

[255]:
```
y_pred=lm.predict(X_test_new_1)
```

[256]:
```
#model evalution
fig=plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test v/s Y_pred',fontsize=20)
plt.xlabel('y_test',fontsize=18)
plt.ylabel('y_pred',fontsize=16)
```

[256]: Text(0, 0.5, 'y_pred')



9.1.3 11.3 R2_score

[257]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

[257]: 0.550232923858538

- Thus, for the model with 6 variables, the r-squared on training and test data is about 56.7% and 55.02% respectively. The adjusted r-squared on the train set is about is about 56.7%.

[258]:
```
print(X_train.columns)
```

```
Index(['const', 'Item_MRP', 'Item_Fat_Content_1', 'Outlet_Type_0',
'Outlet_Type_1', 'Outlet_Type_3'], dtype='object')
```

[259]:
```
cols=['Item_MRP', 'Item_Fat_Content_1', 'Outlet_Type_0', 'Outlet_Type_1',␣
↪'Outlet_Type_3']
```

[261]:
```
cols
```

[261]:
```
['Item_MRP',
 'Item_Fat_Content_1',
 'Outlet_Type_0',
 'Outlet_Type_1',
 'Outlet_Type_3']
```

[262]:
```
plt.figure(figsize=(15,10))

# Heatmap
sns.heatmap(df[cols].corr(), cmap="YlGnBu", annot=True)
plt.show()
```

9.2 12 Summary

By following equation we can predict the Outlet Sales of the Store

- Outlet_Sales=977.55 $\times$ Item_MRP + 83.68 $\times$ Item_Fat_Content_2 - 1615.65 $\times$ Outlet_Type_0 + 349.93 Outlet_Type_1 + 1737.29 $\times$ Outlet_Type_3

Following are the varibales that cotributed in predicting the sales of the store

- Item_MRP
- Item_Fat_Content_2
- Outlet_Type_0
- Outlet_Type_1
- Outlet_Type_3

9.3 13. Result

- built a solution that is able to predict the sales of thedifferent stores of Big Mart according to the provided dataset.

[ ]:
```

```