In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import r2_score
from sklearn.model_selection  import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=pd.read_csv(r'https://raw.githubusercontent.com/dsrscientist/bigdatamart_rep/master/bigdatamart_Train.csv')
```

In [3]:
```
df
```

Out[3]:

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

8523 rows × 12 columns

In [4]:
```
df.describe()
```

Out[4]:

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

In [5]:
```
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
```

In [6]:
```
df.columns
```

Out[6]:
```
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'],
      dtype='object')
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Item_Identifier               0
Item_Weight                1463
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                2410
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```

In [8]:
```
df['Item_Weight'].fillna(df['Item_Weight'].mean(),inplace=True)
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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
```

In [10]:
```
df['Outlet_Size'].fillna(df['Outlet_Size'].mode()[0],inplace=True)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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
```

In [12]:
```
df.shape
```

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

In [13]:
```
df.Item_Fat_Content.value_counts()
```

Out[13]:
```
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64
```

In [14]:
```
df.Item_Type.value_counts()
```

Out[14]:
```
Fruits and Vegetables    1232
Snack Foods              1200
Household                 910
Frozen Foods              856
Dairy                     682
Canned                    649
Baking Goods              648
Health and Hygiene        520
Soft Drinks               445
Meat                      425
Breads                    251
Hard Drinks               214
Others                    169
Starchy Foods             148
Breakfast                 110
Seafood                    64
Name: Item_Type, dtype: int64
```

In [15]:
```
df.Outlet_Size.value_counts()
```

Out[15]:
```
Medium    5203
Small     2388
High       932
Name: Outlet_Size, dtype: int64
```

In [16]:
```
df.Outlet_Location_Type.value_counts()
```

Out[16]:
```
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: Outlet_Location_Type, dtype: int64
```

In [17]:
```
df.Outlet_Type.value_counts()
```

Out[17]:
```
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
```

```
Name: Outlet_Type, dtype: int64
```

In [18]:
```
df['Outlet_Establishment_Year'].unique()
```

Out[18]:
```
array([1999, 2009, 1998, 1987, 1985, 2002, 2007, 1997, 2004], dtype=int64)
```

In [19]:
```
df['Item_Fat_Content'].replace(['LF','Low Fat','low fat', 'reg','Regular'],['Low Fat','Low Fat','Low Fat','Regula
df
```

Out[19]:

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

8523 rows × 12 columns

In [21]:
```
df['Outlet_Age']= df['Outlet_Establishment_Year'].apply(lambda year: 2021 - year)
df
```

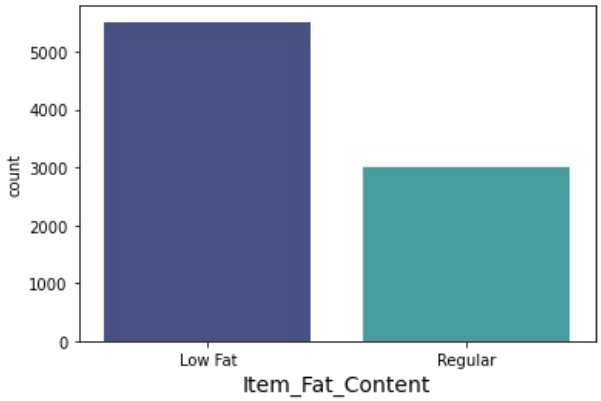
Out[21]:

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

8523 rows × 13 columns

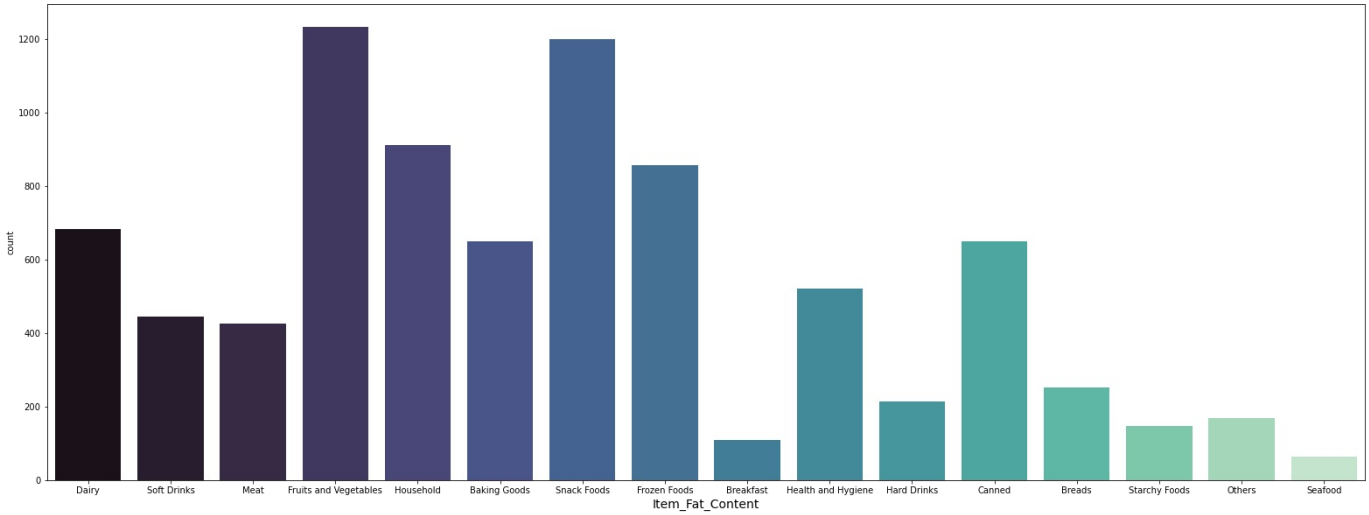
In [22]:
```
plt.figure(figsize=(6,4))
```

```
sns.countplot(x='Item_Fat_Content', data=df ,palette='mako')
plt.xlabel('Item_Fat_Content', fontsize=14)
plt.show()
```



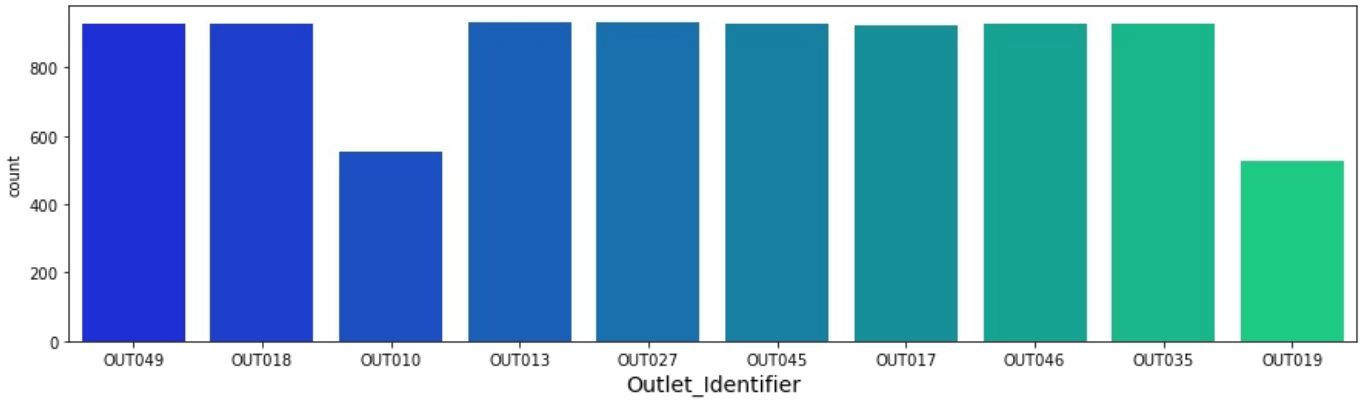
In [23]:

```
plt.figure(figsize=(27,10))
sns.countplot(x='Item_Type', data=df ,palette='mako')
plt.xlabel('Item_Fat_Content', fontsize=14)
plt.show()
```



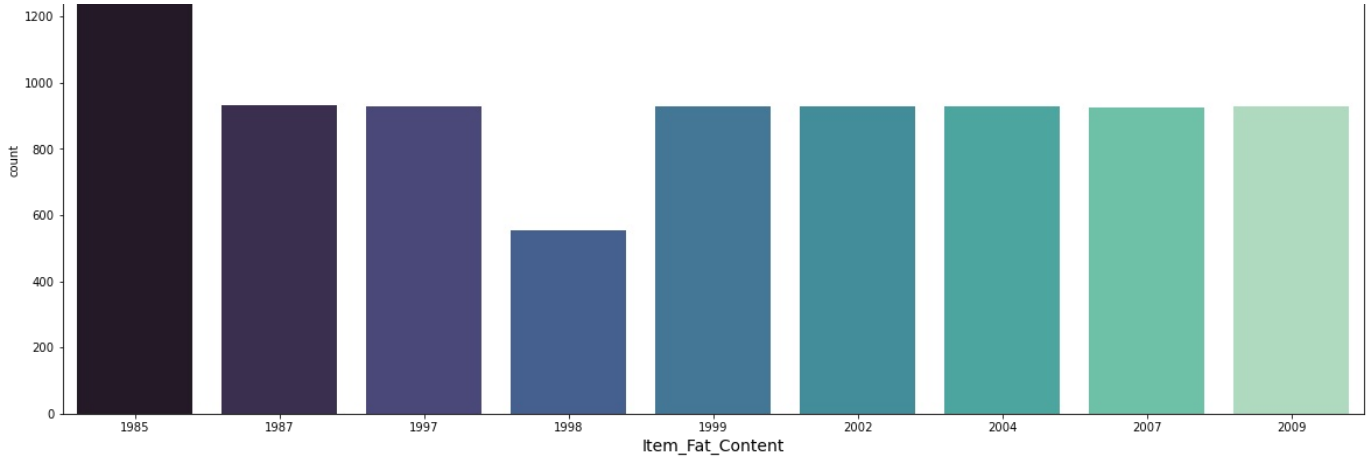
In [24]:

```
plt.figure(figsize=(15,4))
sns.countplot(x='Outlet_Identifier', data=df ,palette='winter')
plt.xlabel('Outlet_Identifier', fontsize=14)
plt.show()
```



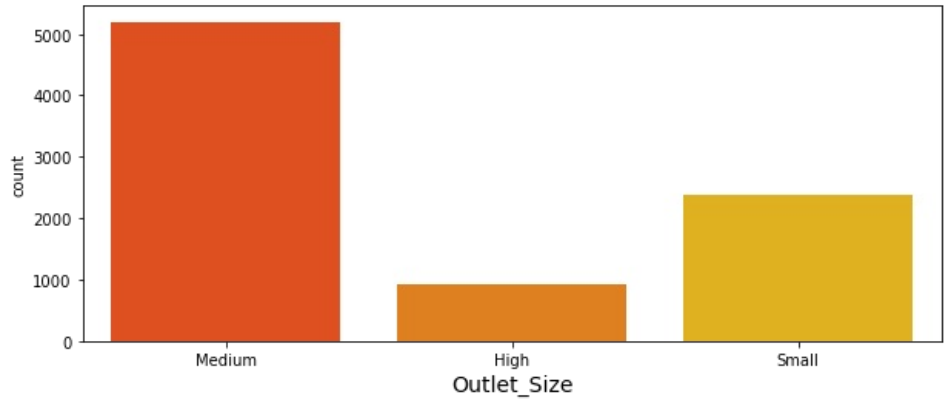
In [25]:

```
plt.figure(figsize=(20,8))
sns.countplot(x='Outlet_Establishment_Year', data=df ,palette='mako')
plt.xlabel('Item_Fat_Content', fontsize=14)
plt.show()
```



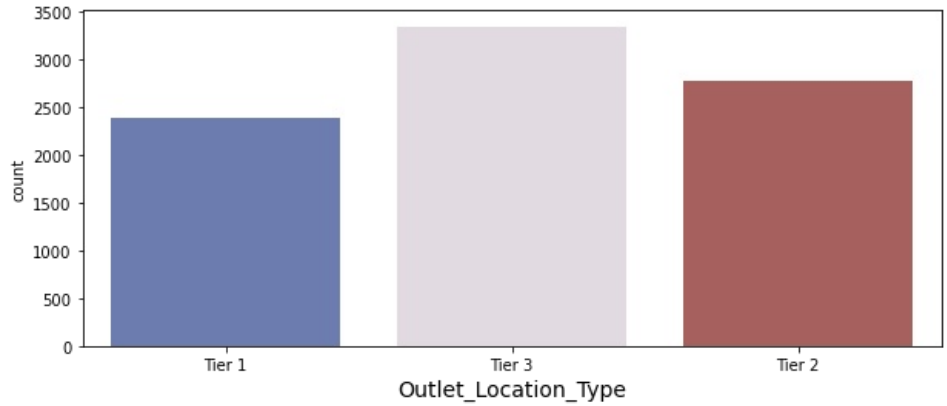
In [26]:

```
plt.figure(figsize=(10,4))
sns.countplot(x='Outlet_Size', data=df ,palette='autumn')
plt.xlabel('Outlet_Size', fontsize=14)
plt.show()
```



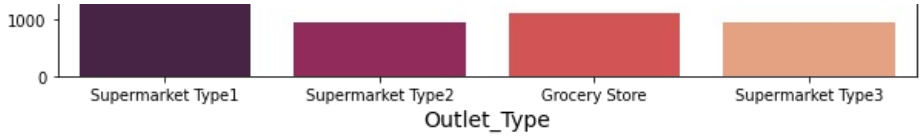
In [27]:

```
plt.figure(figsize=(10,4))
sns.countplot(x='Outlet_Location_Type', data=df ,palette='twilight_shifted')
plt.xlabel('Outlet_Location_Type', fontsize=14)
plt.show()
```



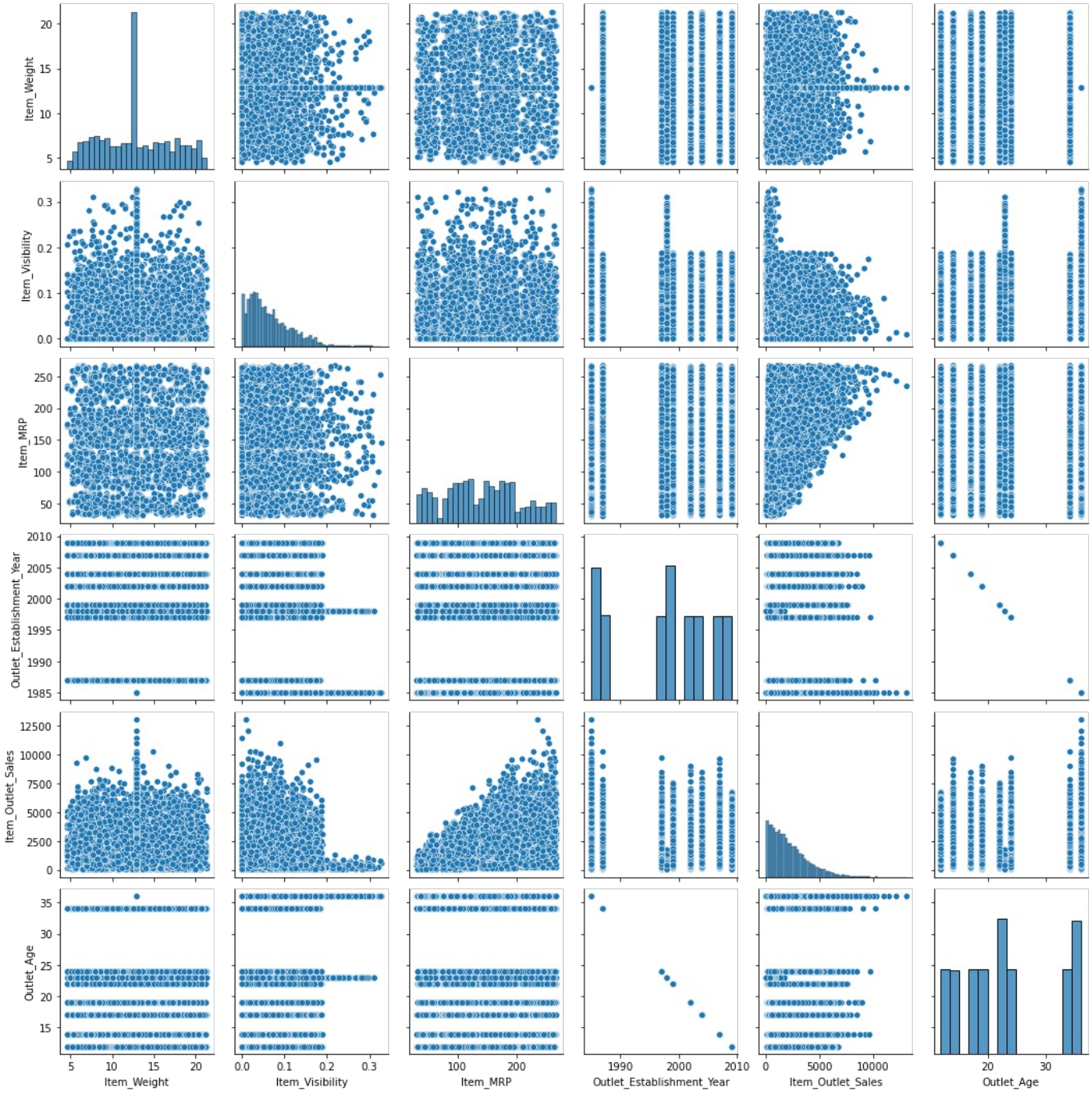
In [28]:

```
plt.figure(figsize=(10,4))
sns.countplot(x='Outlet_Type', data=df ,palette='rocket')
plt.xlabel('Outlet_Type', fontsize=14)
plt.show()
```

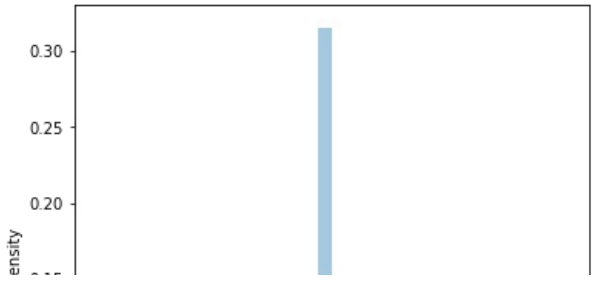


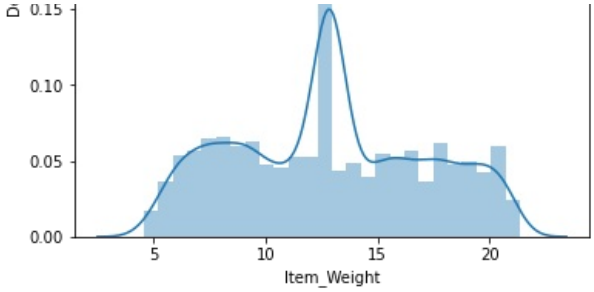
In [29]:
```
sns.pairplot(df)
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x1a3f2b53250>

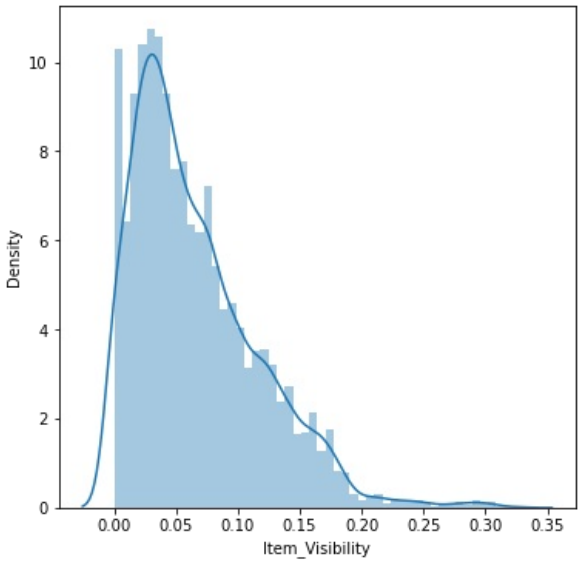


In [30]:
```
plt.figure(figsize=(6,6))
sns.distplot(df["Item_Weight"])
plt.show()
```

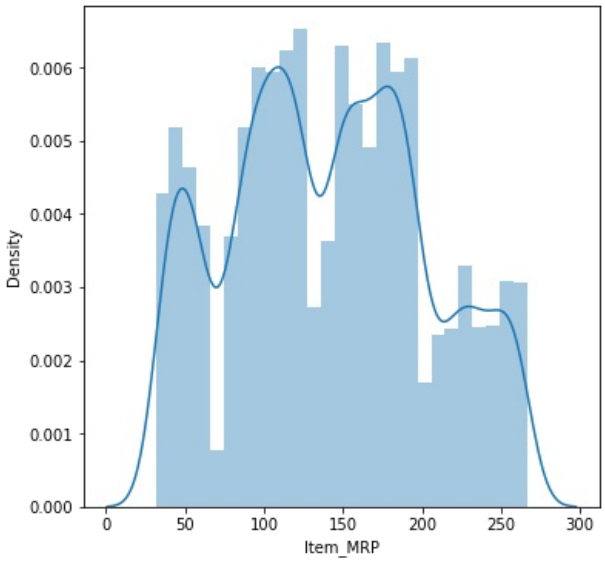




In [31]:
```
plt.figure(figsize=(6,6))
sns.distplot(df["Item_Visibility"])
plt.show()
```



In [32]:
```
plt.figure(figsize=(6,6))
sns.distplot(df["Item_MRP"])
plt.show()
```

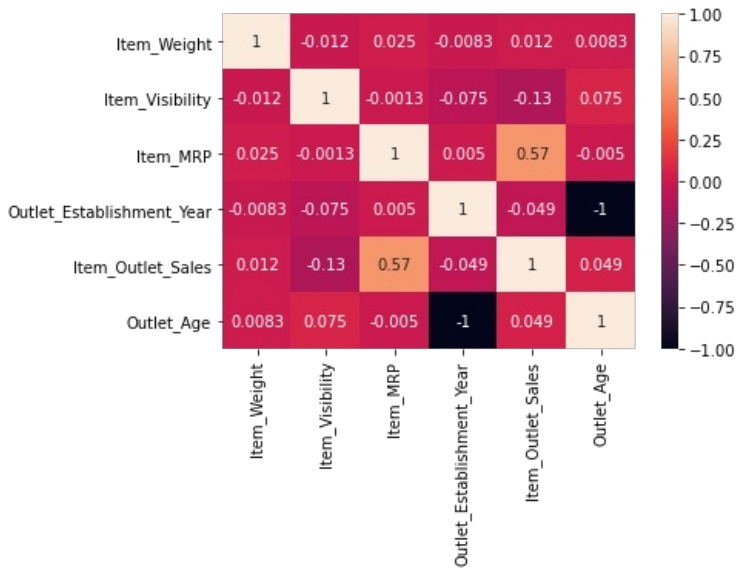


In [33]:
```
df.skew()
```

Out[33]:
```
Item_Weight                  0.090561
Item_Visibility              1.167091
Item_MRP                     0.127202
Outlet_Establishment_Year   -0.396641
Item_Outlet_Sales            1.177531
Outlet_Age                   0.396641
dtype: float64
```

In [34]:
```
df_corr=df.corr()
sns.heatmap(df_corr,annot=True)
plt.show
```

Out[34]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [35]:
```
df
```

Out[35]:

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

8523 rows × 13 columns

In [42]:
```
le = LabelEncoder()
Label = ['Item_Fat_Content','Outlet_Size','Outlet_Location_Type','Outlet_Type']

for i in Label:
    df[i] = le.fit_transform(df[i])

df.head()
```

Out[42]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	Outl
0	9.30	0	0.016047	Dairy	249.8092	1	0	1	3735.1380	
1	5.92	1	0.019278	Soft Drinks	48.2692	1	2	2	443.4228	
2	17.50	0	0.016760	Meat	141.6180	1	0	1	2097.2700	

3	19.20	1	0.000000	Fruits and Vegetables	182.0950	1	2	0	732.3800
4	8.93	0	0.000000	Household	53.8614	0	2	1	994.7052

In [43]:
```
df['Outlet_Type'].unique()
```

Out[43]:
```
array([1, 2, 0, 3])
```

In [44]:
```
cols = ['Item_Type']
```

In [45]:
```
OH_encoder = OneHotEncoder(handle_unknown='ignore', sparse=False)
tr_oh = pd.DataFrame(OH_encoder.fit_transform(df[cols])).astype('int64')
```

In [46]:
```
tr_oh.columns = OH_encoder.get_feature_names(cols)
```

In [47]:
```
tr_oh.index =df.index
```

In [48]:
```
tr_fe = pd.concat([df, tr_oh], axis=1)
```

In [49]:
```
tr_fe
```

Out[49]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	9.300	0	0.016047	Dairy	249.8092	1	0	1	3735.1380
1	5.920	1	0.019278	Soft Drinks	48.2692	1	2	2	443.4228
2	17.500	0	0.016760	Meat	141.6180	1	0	1	2097.2700
3	19.200	1	0.000000	Fruits and Vegetables	182.0950	1	2	0	732.3800
4	8.930	0	0.000000	Household	53.8614	0	2	1	994.7052
...	...	...	...	...	...	...	...	...	...
8518	6.865	0	0.056783	Snack Foods	214.5218	0	2	1	2778.3834
8519	8.380	1	0.046982	Baking Goods	108.1570	1	1	1	549.2850
8520	10.600	0	0.035186	Health and Hygiene	85.1224	2	1	1	1193.1136
8521	7.210	1	0.145221	Snack Foods	103.1332	1	2	2	1845.5976
8522	14.800	0	0.044878	Soft Drinks	75.4670	2	0	1	765.6700

8523 rows × 26 columns

In [50]:
```
tr_fe
```

Out[50]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	9.300	0	0.016047	Dairy	249.8092	1	0	1	3735.1380
1	5.920	1	0.019278	Soft Drinks	48.2692	1	2	2	443.4228
2	17.500	0	0.016760	Meat	141.6180	1	0	1	2097.2700
3	19.200	1	0.000000	Fruits and Vegetables	182.0950	1	2	0	732.3800
4	8.930	0	0.000000	Household	53.8614	0	2	1	994.7052
...	...	...	...	...	...	...	...	...	...
8518	6.865	0	0.056783	Snack Foods	214.5218	0	2	1	2778.3834
8519	8.380	1	0.046982	Baking Goods	108.1570	1	1	1	549.2850
8520	10.600	0	0.035186	Health and Hygiene	85.1224	2	1	1	1193.1136

8521	7.210	1	0.145221	Snack Foods	103.1332	1	2	2	1845.5976
8522	14.800	0	0.044878	Soft Drinks	75.4670	2	0	1	765.6700

8523 rows × 26 columns

In [51]:
```
tr_fe.drop(columns='Item_Type',inplace=True)
```

In [52]:
```
tr_fe.head()
```

Out[52]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	Outlet_Age	Iten
0	9.30	0	0.016047	249.8092	1	0	1	3735.1380	22	
1	5.92	1	0.019278	48.2692	1	2	2	443.4228	12	
2	17.50	0	0.016760	141.6180	1	0	1	2097.2700	22	
3	19.20	1	0.000000	182.0950	1	2	0	732.3800	23	
4	8.93	0	0.000000	53.8614	0	2	1	994.7052	34	

5 rows × 25 columns

In [53]:
```
X = tr_fe.drop(columns='Item_Outlet_Sales')
Y = tr_fe['Item_Outlet_Sales']
```

In [54]:
```
scalar=StandardScaler()
X_scaled=scalar.fit_transform(X)
```

In [55]:
```
lm=LinearRegression()
```

In [56]:
```
lm.fit(X,Y)
```

Out[56]:
```
LinearRegression()
```

In [57]:
```
print(lm.intercept_)
```

```
-224.08404157766563
```

In [58]:
```
print(lm.coef_)
```

```
[-7.91389288e-01  4.50345384e+01 -1.50241352e+03  1.55973987e+01
 -3.33653271e+02 -4.22623406e+02  9.97674873e+02 -1.23881646e+00
  9.22805650e-01  1.93767379e-01 -1.54339189e+01  2.92667509e+01
 -4.47135289e+01 -2.42558633e+01  2.87273631e+01  7.19471470e+00
 -2.70569118e+01 -3.91086816e+01 -3.51517447e+01 -7.93571486e+01
  1.61979672e+02 -1.16247599e+01 -1.54637764e+01  6.38812605e+01]
```

In [59]:
```
import statsmodels.formula.api as smf
```

In [60]:
```
sm=smf.ols(formula='Y~X',data=df).fit()
```

In [61]:
```
sm.summary()
```

Out[61]:

OLS Regression Results

Dep. Variable:	Y	**R-squared:**	0.508
Model:	OLS	**Adj. R-squared:**	0.507
Method:	Least Squares	**F-statistic:**	382.1

Date:	Thu, 03 Mar 2022	**Prob (F-statistic):**	0.00
Time:	16:58:19	**Log-Likelihood:**	-72497.
No. Observations:	8523	**AIC:**	1.450e+05
Df Residuals:	8499	**BIC:**	1.452e+05
Df Model:	23		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-210.9026	80.198	-2.630	0.009	-368.111	-53.694
X[0]	-0.7914	3.093	-0.256	0.798	-6.854	5.271
X[1]	45.0345	29.958	1.503	0.133	-13.691	103.760
X[2]	-1502.4135	259.940	-5.780	0.000	-2011.958	-992.869
X[3]	15.5974	0.210	74.350	0.000	15.186	16.009
X[4]	-333.6533	28.312	-11.785	0.000	-389.152	-278.154
X[5]	-422.6234	22.831	-18.511	0.000	-467.378	-377.869
X[6]	997.6749	19.109	52.209	0.000	960.216	1035.134
X[7]	-1.2388	1.604	-0.772	0.440	-4.383	1.905
X[8]	-12.2586	48.005	-0.255	0.798	-106.360	81.843
X[9]	-12.9876	73.415	-0.177	0.860	-156.900	130.925
X[10]	-28.6153	109.352	-0.262	0.794	-242.971	185.741
X[11]	16.0853	47.945	0.335	0.737	-77.898	110.069
X[12]	-57.8949	47.023	-1.231	0.218	-150.072	34.282
X[13]	-37.4373	42.746	-0.876	0.381	-121.231	46.356
X[14]	15.5459	37.317	0.417	0.677	-57.605	88.697
X[15]	-5.9867	79.531	-0.075	0.940	-161.887	149.913
X[16]	-40.2383	53.482	-0.752	0.452	-145.077	64.600
X[17]	-52.2901	42.792	-1.222	0.222	-136.173	31.593
X[18]	-48.3332	58.212	-0.830	0.406	-162.443	65.777
X[19]	-92.5386	89.083	-1.039	0.299	-267.162	82.085
X[20]	148.7983	141.927	1.048	0.294	-129.414	427.010
X[21]	-24.8062	37.441	-0.663	0.508	-98.199	48.587
X[22]	-28.6452	56.439	-0.508	0.612	-139.279	81.989
X[23]	50.6998	94.455	0.537	0.591	-134.455	235.855

Omnibus:	830.933	**Durbin-Watson:**	2.012
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1713.566
Skew:	0.630	**Prob(JB):**	0.00
Kurtosis:	4.799	**Cond. No.**	9.63e+17

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.24e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [62]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X_scaled,Y,test_size=0.25,random_state=21)
```

In [63]:
```
lm.fit(X_train,Y_train)
```

Out[63]:
```
LinearRegression()
```

In [64]:
```
Y_pred=lm.predict(X_test)
Y_pred_train=lm.predict(X_train)
```

In [65]:
```
lm.score(X_train,Y_train)
```

Out[65]: 0.5026479029216013

In [66]:
```
lm.score(X_test,Y_test)
```

Out[66]: 0.5240024154454632

In [67]:
```
r2=r2_score(Y_test,Y_pred)
r2
```

Out[67]: 0.5240024154454632

In [68]:
```
scores = cross_val_score(lm, X_train, Y_train, scoring='r2', cv=5)
scores
```

Out[68]: array([0.48237845, 0.49792406, 0.48861545, 0.5060224 , 0.50995276])

In [69]:
```
scores = cross_val_score(lm, X_train, Y_train, scoring='r2', cv=5).mean()
scores
```

Out[69]: 0.496978624404148

In [70]:
```
LR_MAE = round(MAE(Y_test, Y_pred),2)
LR_MSE = round(MSE(Y_test, Y_pred),2)
```

In [71]:
```
print(f" Mean Absolute Error: {LR_MAE}\n")
print(f" Mean Squared Error: {LR_MSE}\n")
```

```
 Mean Absolute Error: 889.29

 Mean Squared Error: 1406258.94
```

lasso

In [72]:
```
lasso=Lasso()
```

In [73]:
```
parameters={
    'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]
}
lasso_regressor=GridSearchCV(lasso,parameters,scoring='r2',cv=5)
```

In [74]:
```
lasso_regressor.fit(X_train,Y_train)
```

Out[74]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10,
                                   20, 30, 35, 40, 45, 50, 55, 100]},
             scoring='r2')
```

In [75]:
```
lasso_regressor.best_params_
```

Out[75]: {'alpha': 10}

In [76]:
```
print(lasso_regressor.best_score_)
```

```
0.49869362789897254
```

In [77]:
```
lasso_regressor.score(X_test,Y_test)
```

Out[77]:
```
0.5238536530163884
```

In [78]:
```
LS=Lasso(alpha=10)
```

In [79]:
```
LS.fit(X_train,Y_train)
```

Out[79]:
```
Lasso(alpha=10)
```

In [80]:
```
y_predict = LS.predict(X_test)
```

In [81]:
```
LS_MAE = round(MAE(Y_test, y_predict),2)
LS_MSE = round(MSE(Y_test, y_predict),2)
LS_R_2 = round(r2_score(Y_test, y_predict),4)
```

In [82]:
```
print(f" Mean Absolute Error: {LS_MAE}\n")
print(f" Mean Squared Error: {LS_MSE}\n")
print(f" R^2 Score: {LS_R_2}\n")
```

```
Mean Absolute Error: 889.27

Mean Squared Error: 1406698.43

R^2 Score: 0.5239
```

random forest

In [83]:
```
RFR= RandomForestRegressor(n_estimators=200,max_depth=5, min_samples_leaf=100,n_jobs=4,random_state=101)
```

In [84]:
```
RFR.fit(X_train, Y_train)
```

Out[84]:
```
RandomForestRegressor(max_depth=5, min_samples_leaf=100, n_estimators=200,
                      n_jobs=4, random_state=101)
```

In [85]:
```
y_predict = RFR.predict(X_test)
```

In [86]:
```
RFR_MAE= round(MAE(Y_test, y_predict),2)
RFR_MSE = round(MSE(Y_test, y_predict),2)
RFR_R_2 = round(r2_score(Y_test, y_predict),4)
```

In [87]:
```
print(f" Mean Absolute Error: {RFR_MAE}\n")
print(f" Mean Squared Error: {RFR_MSE}\n")
print(f" R^2 Score: {RFR_R_2}\n")
```

```
Mean Absolute Error: 756.49

Mean Squared Error: 1160426.4

R^2 Score: 0.6072
```

Test Dataset

In [88]:
```
df_test=pd.read_csv(r'https://raw.githubusercontent.com/dsrscientist/bigdatamart_rep/master/bigdatamart_Test.csv'
```

In [89]:
```
df_test
```

Out[89]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Si
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Mediu
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	N
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	N
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	N
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Mediu
...	...	...	...	...	...	...	...	...	
5676	FDB58	10.500	Regular	0.013496	Snack Foods	141.3154	OUT046	1997	Sm
5677	FDD47	7.600	Regular	0.142991	Starchy Foods	169.1448	OUT018	2009	Mediu
5678	NCO17	10.000	Low Fat	0.073529	Health and Hygiene	118.7440	OUT045	2002	N
5679	FDJ26	15.300	Regular	0.000000	Canned	214.6218	OUT017	2007	N
5680	FDU37	9.500	Regular	0.104720	Canned	79.7960	OUT045	2002	N

5681 rows × 11 columns

In [90]:
```
df_test.describe()
```

Out[90]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year
count	4705.000000	5681.000000	5681.000000	5681.000000
mean	12.695633	0.065684	141.023273	1997.828903
std	4.664849	0.051252	61.809091	8.372256
min	4.555000	0.000000	31.990000	1985.000000
25%	8.645000	0.027047	94.412000	1987.000000
50%	12.500000	0.054154	141.415400	1999.000000
75%	16.700000	0.093463	186.026600	2004.000000
max	21.350000	0.323637	266.588400	2009.000000

In [91]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5681 entries, 0 to 5680
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            5681 non-null   object
 1   Item_Weight                4705 non-null   float64
 2   Item_Fat_Content           5681 non-null   object
 3   Item_Visibility            5681 non-null   float64
 4   Item_Type                  5681 non-null   object
 5   Item_MRP                   5681 non-null   float64
 6   Outlet_Identifier          5681 non-null   object
 7   Outlet_Establishment_Year  5681 non-null   int64
 8   Outlet_Size                4075 non-null   object
 9   Outlet_Location_Type       5681 non-null   object
 10  Outlet_Type                5681 non-null   object
dtypes: float64(3), int64(1), object(7)
memory usage: 488.3+ KB
```

In [92]:
```
df_test.columns
```

Out[92]:
```
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'],
      dtype='object')
```

In [93]:
```
df_test.isnull().sum()
```

Out[93]:
```
Item_Identifier               0
Item_Weight                 976
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                1606
Outlet_Location_Type          0
Outlet_Type                   0
dtype: int64
```

In [94]:
```
df_test['Item_Weight'].fillna(df_test['Item_Weight'].mean(),inplace=True)
```

In [95]:
```
df_test.isnull().sum()
```

Out[95]:
```
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                1606
Outlet_Location_Type          0
Outlet_Type                   0
dtype: int64
```

In [96]:
```
df_test['Outlet_Size'].fillna(df_test['Outlet_Size'].mode()[0],inplace=True)
```

In [97]:
```
df_test.isnull().sum()
```

Out[97]:
```
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
dtype: int64
```

In [98]:
```
df_test.shape
```

Out[98]:
```
(5681, 11)
```

In [99]:
```
df_test['Item_Fat_Content'].replace(['LF','Low Fat','low fat', 'reg','Regular'],['Low Fat','Low Fat','Low Fat','F
```

In [100…
```
df_test
```

Out[100…

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Si
0	FDW58	20.750000	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Mediu
1	FDW14	8.300000	Regular	0.038428	Dairy	87.3198	OUT017	2007	Mediu
2	NCN55	14.600000	Low Fat	0.099575	Others	241.7538	OUT010	1998	Mediu
3	FDQ58	7.315000	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	Mediu
4	FDY38	12.695633	Regular	0.118599	Dairy	234.2300	OUT027	1985	Mediu
...	...	...	...	...	...	...	...	...	
5676	FDB58	10.500000	Regular	0.013496	Snack Foods	141.3154	OUT046	1997	Sm
5677	FDD47	7.600000	Regular	0.142991	Starchy Foods	169.1448	OUT018	2009	Mediu
5678	NCO17	10.000000	Low Fat	0.073529	Health and Hygiene	118.7440	OUT045	2002	Mediu
5679	FDJ26	15.300000	Regular	0.000000	Canned	214.6218	OUT017	2007	Mediu
5680	FDU37	9.500000	Regular	0.104720	Canned	79.7960	OUT045	2002	Mediu

5681 rows × 11 columns

In [101…
```
df_test['Outlet_Age']= df_test['Outlet_Establishment_Year'].apply(lambda year: 2021 - year)
```

In [102…
```
df_test
```

Out[102…

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Si
0	FDW58	20.750000	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Mediu
1	FDW14	8.300000	Regular	0.038428	Dairy	87.3198	OUT017	2007	Mediu
2	NCN55	14.600000	Low Fat	0.099575	Others	241.7538	OUT010	1998	Mediu
3	FDQ58	7.315000	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	Mediu
4	FDY38	12.695633	Regular	0.118599	Dairy	234.2300	OUT027	1985	Mediu
...	...	...	...	...	...	...	...	...	
5676	FDB58	10.500000	Regular	0.013496	Snack Foods	141.3154	OUT046	1997	Sm
5677	FDD47	7.600000	Regular	0.142991	Starchy Foods	169.1448	OUT018	2009	Mediu
5678	NCO17	10.000000	Low Fat	0.073529	Health and Hygiene	118.7440	OUT045	2002	Mediu
5679	FDJ26	15.300000	Regular	0.000000	Canned	214.6218	OUT017	2007	Mediu
5680	FDU37	9.500000	Regular	0.104720	Canned	79.7960	OUT045	2002	Mediu

5681 rows × 12 columns

In [103…
```
df_test.drop(columns=['Item_Identifier','Outlet_Identifier','Outlet_Establishment_Year'],inplace=True)
```

In [104…
```
df_test
```

Out[104…

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Outlet_Age
0	20.750000	Low Fat	0.007565	Snack Foods	107.8622	Medium	Tier 1	Supermarket Type1	22
1	8.300000	Regular	0.038428	Dairy	87.3198	Medium	Tier 2	Supermarket Type1	14

2	14.600000	Low Fat	0.099575	Others	241.7538	Medium	Tier 3	Grocery Store	23
3	7.315000	Low Fat	0.015388	Snack Foods	155.0340	Medium	Tier 2	Supermarket Type1	14
4	12.695633	Regular	0.118599	Dairy	234.2300	Medium	Tier 3	Supermarket Type3	36
...	...	...	...	...	...	...	...	...	...
5676	10.500000	Regular	0.013496	Snack Foods	141.3154	Small	Tier 1	Supermarket Type1	24
5677	7.600000	Regular	0.142991	Starchy Foods	169.1448	Medium	Tier 3	Supermarket Type2	12
5678	10.000000	Low Fat	0.073529	Health and Hygiene	118.7440	Medium	Tier 2	Supermarket Type1	19
5679	15.300000	Regular	0.000000	Canned	214.6218	Medium	Tier 2	Supermarket Type1	14
5680	9.500000	Regular	0.104720	Canned	79.7960	Medium	Tier 2	Supermarket Type1	19

5681 rows × 9 columns

In [105…
```
le = LabelEncoder()
Label = ['Item_Fat_Content','Outlet_Size','Outlet_Location_Type','Outlet_Type']

for i in Label:
    df_test[i] = le.fit_transform(df_test[i])

df_test.head()
```

Out[105…

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Outlet_Age
0	20.750000	0	0.007565	Snack Foods	107.8622	1	0	1	22
1	8.300000	1	0.038428	Dairy	87.3198	1	1	1	14
2	14.600000	0	0.099575	Others	241.7538	1	2	0	23
3	7.315000	0	0.015388	Snack Foods	155.0340	1	1	1	14
4	12.695633	1	0.118599	Dairy	234.2300	1	2	3	36

In [106…
```
cols = ['Item_Type']
```

In [107…
```
OH_encoder = OneHotEncoder(handle_unknown='ignore', sparse=False)
te_oh = pd.DataFrame(OH_encoder.fit_transform(df_test[cols])).astype('int64')
```

In [108…
```
te_oh.columns = OH_encoder.get_feature_names(cols)
```

In [109…
```
te_oh.index =df_test.index
```

In [110…
```
te_fe = pd.concat([df_test, te_oh], axis=1)
```

In [111…
```
te_fe
```

Out[111…

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Outlet_Age	Item_Ty
0	20.750000	0	0.007565	Snack Foods	107.8622	1	0	1	22	
1	8.300000	1	0.038428	Dairy	87.3198	1	1	1	14	
2	14.600000	0	0.099575	Others	241.7538	1	2	0	23	
3	7.315000	0	0.015388	Snack Foods	155.0340	1	1	1	14	
4	12.695633	1	0.118599	Dairy	234.2300	1	2	3	36	
...	...	...	...	...	...	...	...	...	...	
5676	10.500000	1	0.013496	Snack Foods	141.3154	2	0	1	24	
5677	7.600000	1	0.142991	Starchy Foods	169.1448	1	2	2	12	

5678	10.000000	0	0.073529	Health and Hygiene	118.7440	1	1	1	19
5679	15.300000	1	0.000000	Canned	214.6218	1	1	1	14
5680	9.500000	1	0.104720	Canned	79.7960	1	1	1	19

5681 rows × 25 columns

In [113…
```
te_fe.drop(columns='Item_Type',inplace=True)
```

In [114…
```
te_fe.head()
```

Out[114…

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Outlet_Age	Item_Type_Baking Goods	Iter
0	20.750000	0	0.007565	107.8622	1	0	1	22	0	
1	8.300000	1	0.038428	87.3198	1	1	1	14	0	
2	14.600000	0	0.099575	241.7538	1	2	0	23	0	
3	7.315000	0	0.015388	155.0340	1	1	1	14	0	
4	12.695633	1	0.118599	234.2300	1	2	3	36	0	

5 rows × 24 columns

Prediction on Test Dataset Using Random Forest

In [115…
```
y_pred_test=RFR.predict(te_fe)
```

In [116…
```
df3=pd.DataFrame(y_pred_test)
df3
```

Out[116…

	0
0	4846.098031
1	4804.532675
2	4806.205605
3	4806.205605
4	5672.172024
...	...
5676	4844.425101
5677	5672.172024
5678	4806.205605
5679	4804.532675
5680	4804.532675

5681 rows × 1 columns

In [117…
```
df1_test=pd.concat([df_test,df3],axis=1)
```

In [118…
```
df1_test
```

Out[118…

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Outlet_Age	
0	20.750000	0	0.007565	Snack Foods	107.8622	1	0	1	22	4846.09
1	8.300000	1	0.038428	Dairy	87.3198	1	1	1	14	4804.53
2	14.600000	0	0.099575	Others	241.7538	1	2	0	23	4806.20
3	7.315000	0	0.015388	Snack Foods	155.0340	1	1	1	14	4806.20
4	12.695633	1	0.118599	Dairy	234.2300	1	2	3	36	5672.17

...	...	...	...	...	...	...	...	...	...	
5676	10.500000	1	0.013496	Snack Foods	141.3154	2	0	1	24	4844.42
5677	7.600000	1	0.142991	Starchy Foods	169.1448	1	2	2	12	5672.17
5678	10.000000	0	0.073529	Health and Hygiene	118.7440	1	1	1	19	4806.20
5679	15.300000	1	0.000000	Canned	214.6218	1	1	1	14	4804.53
5680	9.500000	1	0.104720	Canned	79.7960	1	1	1	19	4804.53

5681 rows × 10 columns

In [119…
```
df1_test.rename(columns={0:"Predicted_Sales"},inplace=True)
```

In [120…
```
df1_test
```

Out[120…

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Outlet_Age	Predicte
0	20.750000	0	0.007565	Snack Foods	107.8622	1	0	1	22	484
1	8.300000	1	0.038428	Dairy	87.3198	1	1	1	14	480
2	14.600000	0	0.099575	Others	241.7538	1	2	0	23	480
3	7.315000	0	0.015388	Snack Foods	155.0340	1	1	1	14	480
4	12.695633	1	0.118599	Dairy	234.2300	1	2	3	36	567
...	...	...	...	...	...	...	...	...	...	
5676	10.500000	1	0.013496	Snack Foods	141.3154	2	0	1	24	484
5677	7.600000	1	0.142991	Starchy Foods	169.1448	1	2	2	12	567
5678	10.000000	0	0.073529	Health and Hygiene	118.7440	1	1	1	19	480
5679	15.300000	1	0.000000	Canned	214.6218	1	1	1	14	480
5680	9.500000	1	0.104720	Canned	79.7960	1	1	1	19	480

5681 rows × 10 columns

In [121…
```
md=[lm,Lasso,RFR]
import pickle
filename="Big_mart_Dataset"
pickle.dump(md,open(filename,"wb"))
```

Conclusion:

1.Conducted 3 models on Big_Mart dataset namely, Linear Regression, Lasso and Random Forest 2.It comes to a conclusion that Random Forest model is providing best score for Big_Mart dataset 3.r2_score=0.6072 4.Prediction is Done On Test Dataset using random forest Model

In [ ]:
```
```