Problem Statement

The Big Mart Sales dataset on Kaggle aims to create a predictive model using machine learning techniques for forecasting product sales in the Big Mart retail chain. The goal is to predict continuous sales figures for various products across different stores based on historical sales data and relevant features. This model will aid Big Mart in optimizing inventory management, stock replenishment strategies, and store performance by providing accurate sales forecasts.

Library:-

To initiate our analysis, we imported essential libraries for data manipulation and visualization. Subsequently, the Big Mart Sales dataset was loaded seamlessly into our environment using Pandas, a Python library for data manipulation.

In [99]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

Load DataSet:-

In [100]:
```
df_train=pd.read_csv("Train.csv")
df_test=pd.read_csv("Test.csv")
```

In [101]:
```
df_train
```

Out[101]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outl
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	
...	...	...	...	...	...	...	
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	

8523 rows × 12 columns

In [102]:
```
df_test
```

Out[102]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outle
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	
...	...	...	...	...	...	...	
5676	FDB58	10.500	Regular	0.013496	Snack Foods	141.3154	
5677	FDD47	7.600	Regular	0.142991	Starchy Foods	169.1448	
5678	NCO17	10.000	Low Fat	0.073529	Health and Hygiene	118.7440	
5679	FDJ26	15.300	Regular	0.000000	Canned	214.6218	
5680	FDU37	9.500	Regular	0.104720	Canned	79.7960	

5681 rows × 11 columns

Manally EDA

In [103]:
```
df_train.shape
```

Out[103]: (8523, 12)

In [104]:
```
df_test.shape
```

Out[104]: (5681, 11)

In [105]:
```
df_train.isnull().sum()
```

Out[105]:
```
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 [106]:
```
df_test.isnull().sum()
```

Out[106]:
```
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 [107]:
```
df_train.info()
```

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

In [108]:
```
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 [109]:
```
df_train.describe()
```

Out[109]:

	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 [110]:
```
df_test.describe()
```

Out[110]:

	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 [111]:
```
for i in df_train.columns:
    print(i)
    print(df_train[i].value_counts())
    print("**********\n")
```

```
Item_Identifier
Item_Identifier
FDW13    10
FDG33    10
NCY18     9
FDD38     9
DRE49     9
         ..
FDY43     1
FDQ60     1
FDO33     1
DRF48     1
FDC23     1
Name: count, Length: 1559, dtype: int64
**********

Item_Weight
Item_Weight
12.150    86
```

As I seen- The dataset consists of both numerical and categorical variables

We have two column which have missing value

-Item_Weight

-Outlet_Size

Item_Weight is numerical column so we fill it with Mean Imputation

In [112]:
```
df_train['Item_Weight'].describe()
```

Out[112]:
```
count    7060.000000
mean       12.857645
std         4.643456
min         4.555000
25%         8.773750
50%        12.600000
75%        16.850000
max        21.350000
Name: Item_Weight, dtype: float64
```

In [113]:
```
df_train['Item_Weight'].fillna(df_train['Item_Weight'].mean(),inplace=True)
df_test['Item_Weight'].fillna(df_test['Item_Weight'].mean(),inplace=True)
```

In [114]:
```
df_train.isnull().sum()
```

Out[114]:
```
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 [115]:
```
df_train['Item_Weight'].describe()
```

Out[115]:
```
count    8523.000000
mean       12.857645
std         4.226124
min         4.555000
25%         9.310000
50%        12.857645
75%        16.000000
max        21.350000
Name: Item_Weight, dtype: float64
```

Outlet_Size is catagorical column so we fill it with Mode Imputation

In [116]:
```
df_train['Outlet_Size'].value_counts()
```

Out[116]:
```
Outlet_Size
Medium    2793
Small     2388
High       932
Name: count, dtype: int64
```

In [117]:
```
df_train['Outlet_Size'].mode()
```

Out[117]:
```
0    Medium
Name: Outlet_Size, dtype: object
```

In [118]:
```
df_train['Outlet_Size'].fillna(df_train['Outlet_Size'].mode()[0],inplace=True
df_test['Outlet_Size'].fillna(df_test['Outlet_Size'].mode()[0],inplace=True)
```

In [119]:
```
df_train.isnull().sum()
```

Out[119]:
```
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 [120]:
```
df_test.isnull().sum()
```

Out[120]:
```
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
```

Selecting features based on general requirements

In [121]:
```
df_train.drop(['Item_Identifier','Outlet_Identifier'],axis=1,inplace=True)
df_test.drop(['Item_Identifier','Outlet_Identifier'],axis=1,inplace=True)
```

In [122]:
```
df_train
```

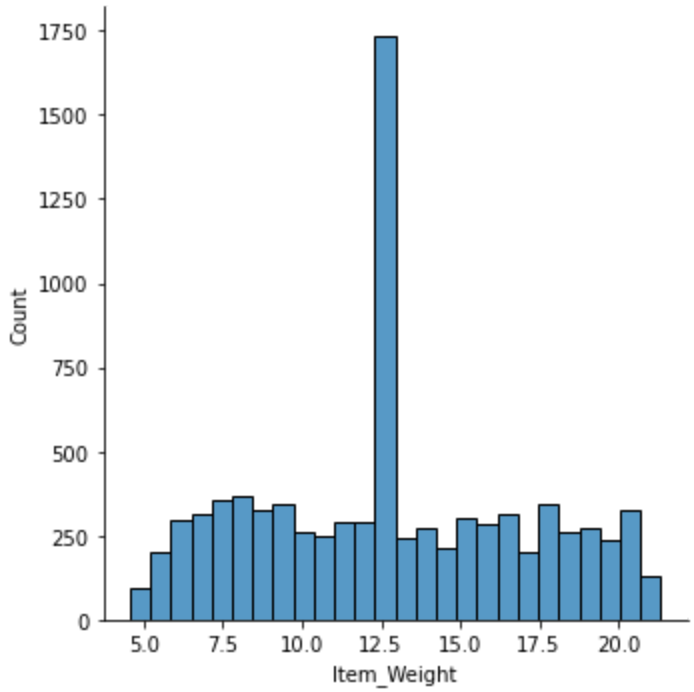
Out[122]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment
0	9.300	Low Fat	0.016047	Dairy	249.8092	
1	5.920	Regular	0.019278	Soft Drinks	48.2692	
2	17.500	Low Fat	0.016760	Meat	141.6180	
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	
4	8.930	Low Fat	0.000000	Household	53.8614	
...	...	...	...	...	...	
8518	6.865	Low Fat	0.056783	Snack Foods	214.5218	
8519	8.380	Regular	0.046982	Baking Goods	108.1570	
8520	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	
8521	7.210	Regular	0.145221	Snack Foods	103.1332	
8522	14.800	Low Fat	0.044878	Soft Drinks	75.4670	

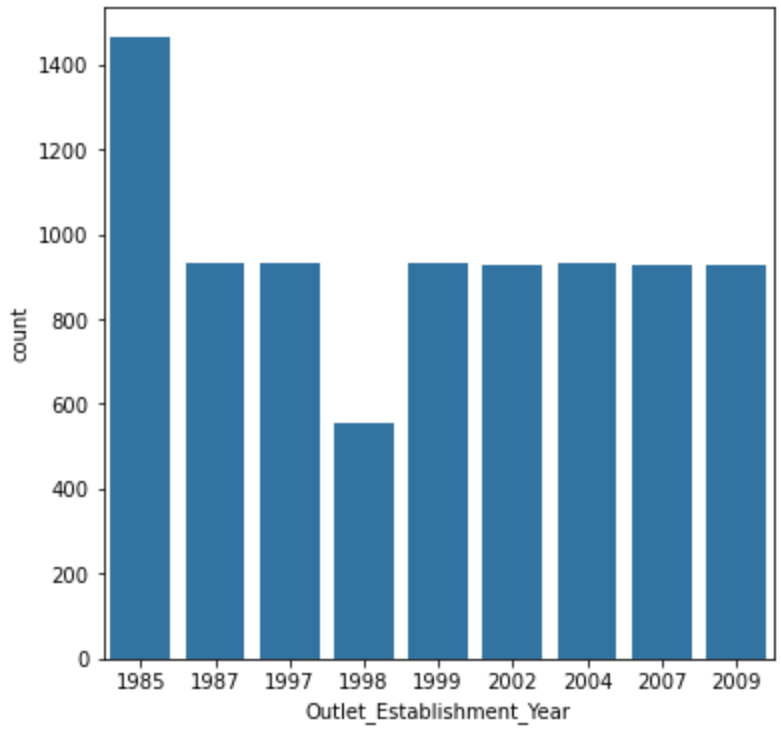
8523 rows × 10 columns

In [126]:
```
plt.figure(figsize=(6,6))
sns.displot(df_train['Item_Weight'])
plt.show()
```

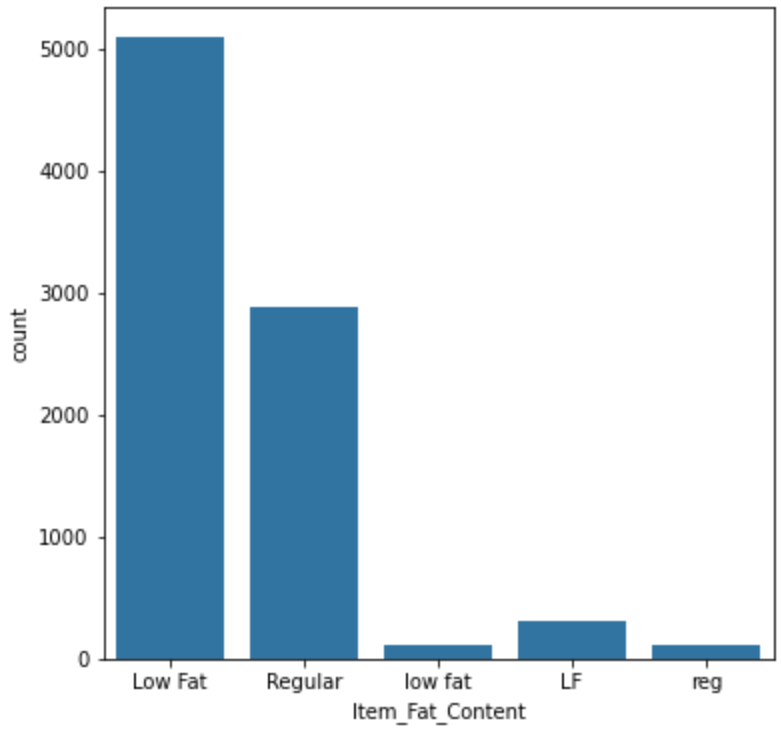
```
<Figure size 432x432 with 0 Axes>
```



In [127]:
```
plt.figure(figsize=(6,6))
sns.countplot(x='Outlet_Establishment_Year', data=df_train)
plt.show()
```



In [128]:
```
plt.figure(figsize=(6,6))
sns.countplot(x='Item_Fat_Content', data=df_train)
plt.show()
```



EDA using Pandas Profiling

In [123]:
```
from pandas_profiling import ProfileReport
```

In [124]:
```
profile = ProfileReport(df_train, title="Pandas Profiling Report")
```

In [125]:
```
profile
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Out[125]:

In [ ]:
```

```

Preprocessing Task before Model Building

In [131]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [132]:
```
df_train['Item_Fat_Content']= le.fit_transform(df_train['Item_Fat_Content'])
df_train['Item_Type']= le.fit_transform(df_train['Item_Type'])
df_train['Outlet_Size']= le.fit_transform(df_train['Outlet_Size'])
df_train['Outlet_Location_Type']= le.fit_transform(df_train['Outlet_Location_
df_train['Outlet_Type']= le.fit_transform(df_train['Outlet_Type'])
```

In [133]:
```
df_train
```

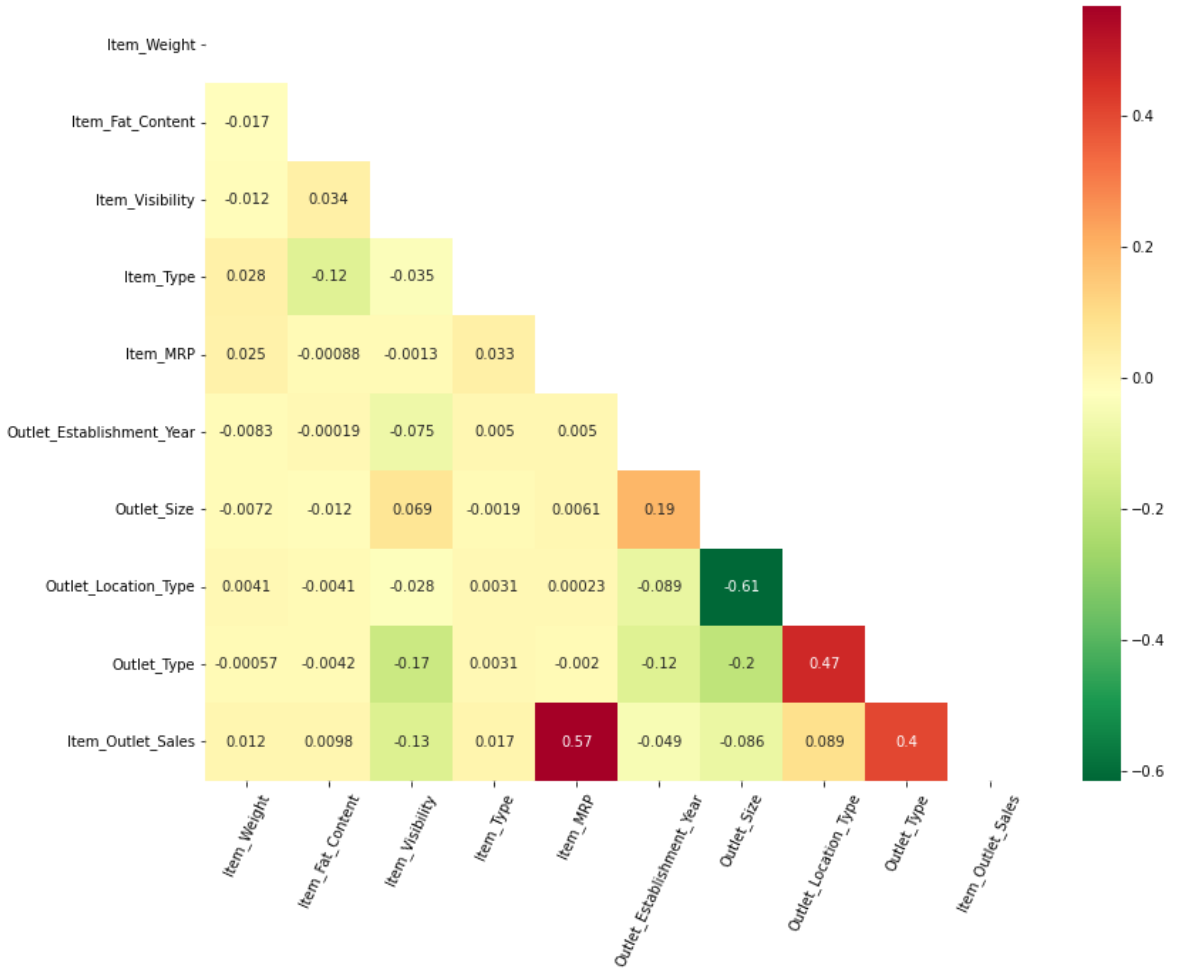
Out[133]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment
0	9.300	1	0.016047	4	249.8092	
1	5.920	2	0.019278	14	48.2692	
2	17.500	1	0.016760	10	141.6180	
3	19.200	2	0.000000	6	182.0950	
4	8.930	1	0.000000	9	53.8614	
...	...	...	...	...	...	
8518	6.865	1	0.056783	13	214.5218	
8519	8.380	2	0.046982	0	108.1570	
8520	10.600	1	0.035186	8	85.1224	
8521	7.210	2	0.145221	13	103.1332	
8522	14.800	1	0.044878	14	75.4670	

8523 rows × 10 columns

In [149]:
```
### check colleration for all columns
train_corr = df_train.corr()
mask = np.triu(np.ones_like(train_corr,dtype=bool))

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



2) Splitting our data into train and test

In [134]:
```
X=df_train.drop('Item_Outlet_Sales',axis=1)
Y=df_train['Item_Outlet_Sales']
```

In [135]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state=101,tes
```

In [136]:
```
X.describe()
```

Out[136]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establish
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8
mean	12.857645	1.369354	0.066132	7.226681	140.992782	1
std	4.226124	0.644810	0.051598	4.209990	62.275067	
min	4.555000	0.000000	0.000000	0.000000	31.290000	1
25%	9.310000	1.000000	0.026989	4.000000	93.826500	1
50%	12.857645	1.000000	0.053931	6.000000	143.012800	1
75%	16.000000	2.000000	0.094585	10.000000	185.643700	2
max	21.350000	4.000000	0.328391	15.000000	266.888400	2

Data Standardization

In [137]:
```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
```

In [138]:
```
X_train_std= sc.fit_transform(X_train)
```

```
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
```

In [139]:
```
X_test_std= sc.transform(X_test)
```

```
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
```

In [140]:
```
X_train_std
```

Out[140]:
```
array([[ 1.52290029, -0.57382672,  0.68469729, ..., -1.95699503,
         1.08786619, -0.25964107],
       [-1.23985603, -0.57382672, -0.09514748, ..., -0.28872895,
        -0.13870429, -0.25964107],
       [ 1.54667616,  0.97378032, -0.00838589, ..., -0.28872895,
        -0.13870429, -0.25964107],
       ...,
       [-0.08197107, -0.57382672, -0.9191623 , ...,  1.37953713,
        -1.36527477, -0.25964107],
       [-0.74888428,  0.97378032,  1.21363058, ..., -0.28872895,
        -0.13870429, -0.25964107],
       [ 0.67885683, -0.57382672,  1.83915356, ..., -0.28872895,
         1.08786619,  0.98524841]])
```

In [141]:
```
X_test_std
```

Out[141]:
```
array([[-0.43860915, -0.57382672, -0.21609255, ..., -0.28872895,
         1.08786619,  0.98524841],
       [ 1.22570189, -0.57382672, -0.52943461, ..., -1.95699503,
         1.08786619, -0.25964107],
       [-1.21845775,  0.97378032,  0.16277342, ...,  1.37953713,
        -1.36527477, -0.25964107],
       ...,
       [ 0.65508096, -0.57382672,  0.87824237, ..., -0.28872895,
         1.08786619, -1.50453056],
       [ 1.01171904, -0.57382672, -1.28409256, ..., -0.28872895,
         1.08786619,  0.98524841],
       [-1.56558548,  0.97378032, -1.09265374, ..., -0.28872895,
        -0.13870429, -0.25964107]])
```

In [142]:
```
Y_train
```

Out[142]:
```
3684     163.7868
1935    1607.2412
5142    1510.0344
4978    1784.3440
2299    3558.0352
           ...
599     5502.8370
5695    1436.7964
8006    2167.8448
1361    2700.4848
1547     829.5868
Name: Item_Outlet_Sales, Length: 6818, dtype: float64
```

In [143]:
```
Y_test
```

Out[143]:
```
8179     904.8222
8355    2795.6942
3411    1947.4650
7089     872.8638
6954    2450.1440
          ...
1317    1721.0930
4996     914.8092
531      370.1848
3891    1358.2320
6629    2418.1856
Name: Item_Outlet_Sales, Length: 1705, dtype: float64
```

Model Building

In [145]:
```
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [146]:
```
models = [LinearRegression, Lasso, Ridge, SVR, DecisionTreeRegressor, RandomF
mae_scores = []
mse_scores = []
rmse_scores = []
r2_scores = []

for model in models:
    regressor = model().fit(X_train, Y_train)
    Y_pred = regressor.predict(X_test)

    mae_scores.append(mean_absolute_error(Y_test, Y_pred))
    mse_scores.append(mean_squared_error(Y_test, Y_pred))
    rmse_scores.append(mean_squared_error(Y_test, Y_pred, squared=False))
    r2_scores.append(r2_score(Y_test, Y_pred))
```

```
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
C:\Users\91825\anaconda3\lib\site-packages\sklearn\utils\validation.py:571:
FutureWarning: is_sparse is deprecated and will be removed in a future versi
on. Check `isinstance(dtype, pd.SparseDtype)` instead.
  array.dtypes.apply(is_sparse).any()):
```

In [147]:
```
regression_metrics_df = pd.DataFrame({
    "Model": ["Linear Regression", "Lasso", "Ridge", "SVR", "Decision Tree Re
    "Mean Absolute Error": mae_scores,
    "Mean Squared Error": mse_scores,
    "Root Mean Squared Error": rmse_scores,
    "R-squared (R2)": r2_scores
})

regression_metrics_df.set_index('Model', inplace=True)
regression_metrics_df
```

Out[147]:

Model	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	R-squared (R2)
Linear Regression	880.999907	1.351270e+06	1162.441266	0.504188
Lasso	880.719550	1.350921e+06	1162.291338	0.504315
Ridge	880.908361	1.351126e+06	1162.379400	0.504240
SVR	1275.347120	2.803243e+06	1674.288864	-0.028575
Decision Tree Regressor	1050.811248	2.274064e+06	1508.000139	0.165593
Random Forest Regressor	783.523890	1.239731e+06	1113.432061	0.545114

Done by -> MD Shaukat Ali , From NIT Dgp

In [ ]:
```
```