In [15]:
```
import numpy as np
import pandas as pd
```

In [16]:
```
train = pd.read_csv(r'D:\DS\ML\bigmart_train.csv')
test = pd.read_csv(r'D:\DS\ML\bigmart_test.csv')
```

In [17]:
```
train.head()
```

Out[17]:

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

In [18]:
```
test.head()
```

Out[18]:

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

In [19]:
```
train.shape
```

Out[19]:
```
(8523, 12)
```

In [20]:
```
test.shape
```

Out[20]:
```
(5681, 11)
```

In [21]:
```
train.isna().sum()
```

Out[21]:
```
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 [22]:
```
test.isna().sum()
```

Out[22]:
```
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
```

Removing missing values and correcting erroneous data

Here since different product categories can have different median weights we group them by categories and find the medium weights for each categories and use it fill in the missing values

In [23]:
```
median_weights =  train.groupby(['Item_Type'],as_index=False).agg({'Item_Weight': np.median})
median_weights
```

Out[23]:

	Item_Type	Item_Weight
0	Baking Goods	11.650
1	Breads	10.600
2	Breakfast	10.695
3	Canned	12.150
4	Dairy	13.350
5	Frozen Foods	12.850
6	Fruits and Vegetables	13.100
7	Hard Drinks	10.100
8	Health and Hygiene	12.150
9	Household	13.150
10	Meat	12.350
11	Others	14.500
12	Seafood	11.650
13	Snack Foods	13.150
14	Soft Drinks	11.800
15	Starchy Foods	13.175

In [24]:
```
median_weights = pd.DataFrame(data=median_weights.values, columns =['Item_Type','Avg_Item_Weight'])
median_weights
```

Out[24]:

	Item_Type	Avg_Item_Weight
0	Baking Goods	11.65
1	Breads	10.6
2	Breakfast	10.695
3	Canned	12.15
4	Dairy	13.35
5	Frozen Foods	12.85
6	Fruits and Vegetables	13.1
7	Hard Drinks	10.1
8	Health and Hygiene	12.15
9	Household	13.15
10	Meat	12.35
11	Others	14.5
12	Seafood	11.65
13	Snack Foods	13.15
14	Soft Drinks	11.8
15	Starchy Foods	13.175

In [25]:
```
train = pd.merge(left=train, right = median_weights, on = ['Item_Type'], how='left')
```

In [26]:
```
train.head()
```

Out[26]:

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

In [27]:
```
train.columns
```

Out[27]:
```
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', 'Avg_Item_Weight'],
      dtype='object')
```

In [28]:
```
train['Item_Weight'] = train.apply(lambda x: x[12] if pd.isna(x[1]) else x[1], axis=1)
```

In [29]:
```
train.isna().sum()
```

Out[29]:
```
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
Avg_Item_Weight                 0
dtype: int64
```

For outlet size we use the mode to fill in the missing values

In [30]:
```
train['Outlet_Size'].unique()
```

Out[30]:
```
array(['Medium', nan, 'High', 'Small'], dtype=object)
```

In [31]:
```
train['Outlet_Size'].value_counts()
```

Out[31]:
```
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64
```

In [32]:
```
train['Outlet_Size'] = train['Outlet_Size'].apply(lambda x: 'Medium' if pd.isna(x) else x)
```

Correcting erroneous data

In [33]:
```
train['Item_Fat_Content'].unique()
```

Out[33]:
```
array(['Low Fat', 'Regular', 'low fat', 'LF', 'reg'], dtype=object)
```

In [34]:
```
train['Item_Fat_Content'] = train['Item_Fat_Content'].map({'Low Fat': 'Low Fat','Regular': 'Regular','low fat': 'Low Fat','LF': 'Low Fat','reg': 'Regular'})
```

In [35]:
```
train['Item_Fat_Content'].unique()
```

Out[35]:
```
array(['Low Fat', 'Regular'], dtype=object)
```

In [36]:
```
train.drop(['Item_Identifier','Outlet_Identifier','Avg_Item_Weight'],axis=1, inplace=True)
```

In [37]:
```
train['Outlet_Type'].unique()
```

Out[37]:
```
array(['Supermarket Type1', 'Supermarket Type2', 'Grocery Store',
       'Supermarket Type3'], dtype=object)
```

In [38]:
```
train = pd.get_dummies(columns=['Item_Fat_Content','Item_Type','Outlet_Size','Outlet_Location_Type','Outlet_Type'],
                       drop_first=True,data=train)
```

In [39]:
```
train.head()
```

Out[39]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales	Item_Fat_Content_Regular	Item_Type_Breads	Item_Type
0	9.30	0.016047	249.8092	1999	3735.1380	0	0	
1	5.92	0.019278	48.2692	2009	443.4228	1	0	
2	17.50	0.016760	141.6180	1999	2097.2700	0	0	
3	19.20	0.000000	182.0950	1998	732.3800	1	0	
4	8.93	0.000000	53.8614	1987	994.7052	0	0	

5 rows × 28 columns

Since the establishment is not an important data here, we use it to find the age of the stores which can be useful.

In [40]:
```
train['Outlet_Age'] = train['Outlet_Establishment_Year'].apply(lambda x: 2021-x)
```

In [41]:
```
train.drop(['Outlet_Establishment_Year'],axis=1, inplace=True)
```

In [42]:
```
train.head()
```

Out[42]:

	Item_Weight	Item_Visibility	Item_MRP	Item_Outlet_Sales	Item_Fat_Content_Regular	Item_Type_Breads	Item_Type_Breakfast	Item_Type_Canned	Item_Type_Dair
0	9.30	0.016047	249.8092	3735.1380	0	0	0	0	
1	5.92	0.019278	48.2692	443.4228	1	0	0	0	
2	17.50	0.016760	141.6180	2097.2700	0	0	0	0	
3	19.20	0.000000	182.0950	732.3800	1	0	0	0	
4	8.93	0.000000	53.8614	994.7052	0	0	0	0	

5 rows × 28 columns

In [43]:
```
X_train = train.drop(['Item_Outlet_Sales'], axis=1)
y_train = train['Item_Outlet_Sales']
```

In [44]:
```
X_train.head()
```

Out[44]:

	Item_Weight	Item_Visibility	Item_MRP	Item_Fat_Content_Regular	Item_Type_Breads	Item_Type_Breakfast	Item_Type_Canned	Item_Type_Dair
0	9.30	0.016047	249.8092	0	0	0	0	
1	5.92	0.019278	48.2692	1	0	0	0	
2	17.50	0.016760	141.6180	0	0	0	0	
3	19.20	0.000000	182.0950	1	0	0	0	
4	8.93	0.000000	53.8614	0	0	0	0	

5 rows × 27 columns

In [45]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [46]:
```
X_train = scaler.fit_transform(X_train)
```

In [47]:
```
from sklearn.linear_model import LinearRegression
```

In [48]:
```
bm_model = LinearRegression()
bm_model.fit(X_train,y_train)
```

Out[48]:
```
LinearRegression()
```

In [49]:
```
y_pred = bm_model.predict(X_train)
```

In [50]:
```
from sklearn.metrics import r2_score, mean_squared_error
```

In [51]:
```
r2_score(y_pred=y_pred, y_true=y_train)
```

Out[51]:
```
0.5634665404828418
```