Big Mart Sales Prediction(Regression Model)

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:

```
m = r'C:\Users\Pratik Sonawane\Downloads\Bigmart_sales.csv'
df = pd.read_csv(m)
```

In [4]:

```
df.head()
```

Out[4]:

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

In [6]:

```
df.shape
```

Out[6]:

```
(8523, 12)
```

In [7]:

```
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 [82]:

```
cat_col = df3.select_dtypes(include = ['object']).columns.tolist()
num_col = df3.select_dtypes(exclude = ['object']).columns.tolist()
```

In [9]:

```
for col in cat_col:
    print(df[col].value_counts())
    print('---------------------------------------------')
```

```
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_Fat_Content
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: count, dtype: int64
---------------------------------------------
Item_Type
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: count, dtype: int64
---------------------------------------------
```

```
Outlet_Identifier
OUT027    935
OUT013    932
OUT049    930
OUT046    930
OUT035    930
OUT045    929
OUT018    928
OUT017    926
OUT010    555
OUT019    528
Name: count, dtype: int64
---------------------------------------------
Outlet_Size
Medium    2793
Small     2388
High       932
Name: count, dtype: int64
---------------------------------------------
Outlet_Location_Type
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: count, dtype: int64
---------------------------------------------
Outlet_Type
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: count, dtype: int64
---------------------------------------------
```

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
0
```

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

Out[11]:
```
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 [12]:
```
df.groupby('Item_Identifier')['Item_Weight'].mean()
```

Out[12]:
```
Item_Identifier
DRA12    11.600
DRA24    19.350
DRA59     8.270
DRB01     7.390
DRB13     6.115
          ...
NCZ30     6.590
NCZ41    19.850
NCZ42    10.500
NCZ53     9.600
NCZ54    14.650
Name: Item_Weight, Length: 1559, dtype: float64
```

In [13]:
```
df1 = df.copy()
```

In [38]:
```
def fillIW(df2):
    item_avg_w = df2.groupby('Item_Identifier')['Item_Weight'].transform('mean')
    df2['Item_Weight'].fillna(item_avg_w,inplace=True)
    return df2
```

In [40]:
```
df2 = fillIW(df2.copy())
```

In [41]:
```
df2.isnull().sum()
```

Out[41]:
```
Item_Identifier                 0
Item_Weight                     4
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 [42]:
```
df2[df2['Item_Weight'].isnull()]
```

Out[42]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
927	FDN52	NaN	Regular	0.130933	Frozen Foods	86.9198	OUT027	1985	Medium
1922	FDK57	NaN	Low Fat	0.079904	Snack Foods	120.0440	OUT027	1985	Medium
4187	FDE52	NaN	Regular	0.029742	Dairy	88.9514	OUT027	1985	Medium
5022	FDQ60	NaN	Regular	0.191501	Baking Goods	121.2098	OUT019	1985	Small

the above 4 had unique item_identifier hence mean cannot be calculated so we are dropping it

In [44]:

```
df3= df2.copy()
```

In [50]:

```
def fillos(df3):
        mode_outlet_size = df3.groupby('Outlet_Type')['Outlet_Size'] \
                         .transform(lambda x: x.mode()[0] if x.mode().any() else pd.NA)
#If the group's Outlet_Size series has any mode (most frequent value)
#(x.mode().any()), it extracts the first mode using x.mode()[0]
        df3['Outlet_Size'] = df3['Outlet_Size'].fillna(mode_outlet_size)
        df3['Outlet_Size'] = df3['Outlet_Size'].fillna(df3['Outlet_Size'].mode()[0])

        return df3

df3 = fillos(df3.copy())

print(df3['Outlet_Size'].isnull().sum())
```

```
0
```

In [53]:

```
df3.groupby('Outlet_Type')['Outlet_Size'].value_counts()
```

Out[53]:

```
Outlet_Type        Outlet_Size
Grocery Store      Small          1083
Supermarket Type1  Small          3715
                   High            932
                   Medium          930
Supermarket Type2  Medium          928
Supermarket Type3  Medium          935
Name: count, dtype: int64
```

```
In [54]: df3.isnull().sum()
```

```
Out[54]: Item_Identifier              0
         Item_Weight                  4
         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 [56]: df3 = df3.dropna()
```

```
In [57]: df3.isnull().sum()
```

```
Out[57]: 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 [58]:

```
df3.describe()
```

Out[58]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	8519.000000	8519.000000	8519.000000	8519.000000	8519.000000
mean	12.875420	0.066112	141.010019	1997.837892	2181.188779
std	4.646098	0.051586	62.283594	8.369105	1706.511093
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.785000	0.026983	93.844900	1987.000000	834.247400
50%	12.650000	0.053925	143.047000	1999.000000	1794.331000
75%	16.850000	0.094558	185.676600	2004.000000	3100.630600
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

min Item_Visibility is 0 we need to replace it with mean

In [61]:

```
# (.loc) to access the entire column (indicated by :), ensuring we only modify the intended column
df3.loc[:,'Item_Visibility'].replace([0],[df3['Item_Visibility'].mean()],inplace=True)
```

In [62]:

```
df3.describe()
```

Out[62]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	8519.000000	8519.000000	8519.000000	8519.000000	8519.000000
mean	12.875420	0.070194	141.010019	1997.837892	2181.188779
std	4.646098	0.048729	62.283594	8.369105	1706.511093
min	4.555000	0.003575	31.290000	1985.000000	33.290000
25%	8.785000	0.033085	93.844900	1987.000000	834.247400
50%	12.650000	0.062511	143.047000	1999.000000	1794.331000
75%	16.850000	0.094558	185.676600	2004.000000	3100.630600
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

In [63]:

```
df3['Item_Fat_Content'].value_counts()
```

Out[63]:

```
Item_Fat_Content
Low Fat     5088
Regular     2886
LF           316
reg          117
low fat      112
Name: count, dtype: int64
```

In [66]:

```
df3['Item_Fat_Content']=df3['Item_Fat_Content'].replace({'LF':'Low Fat','reg':'Regular','low fat':'Low Fat'})
df3['Item_Fat_Content'].value_counts()
```

Out[66]:

```
Item_Fat_Content
Low Fat     5516
Regular     3003
Name: count, dtype: int64
```

In [68]:

```
df3['Item_Identifier']
```

Out[68]:

```
0       FDA15
1       DRC01
2       FDN15
3       FDX07
4       NCD19
        ...
8518    FDF22
8519    FDS36
8520    NCJ29
8521    FDN46
8522    DRG01
Name: Item_Identifier, Length: 8519, dtype: object
```

```
In [74]: df3['New_item_Types']= df3['Item_Identifier'].apply(lambda x : x[:2])
         df3['New_item_Types']
```

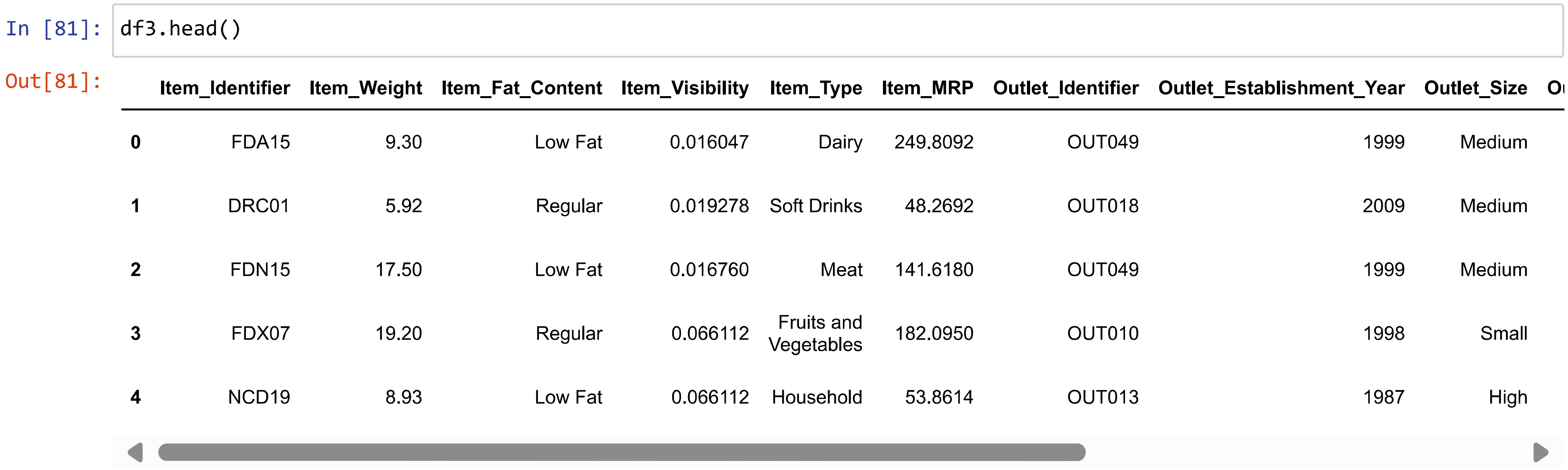
```
Out[74]: 0       FD
         1       DR
         2       FD
         3       FD
         4       NC
                 ..
         8518    FD
         8519    FD
         8520    NC
         8521    FD
         8522    DR
         Name: New_item_Types, Length: 8519, dtype: object
```

```
In [75]: df3['New_item_Types'] = df3['New_item_Types'].map({'FD':'Food','NC':'Non Consumable','DR':'Drinks'})
         df3['New_item_Types']
```

```
Out[75]: 0                 Food
         1               Drinks
         2                 Food
         3                 Food
         4       Non Consumable
                      ...
         8518              Food
         8519              Food
         8520    Non Consumable
         8521              Food
         8522            Drinks
         Name: New_item_Types, Length: 8519, dtype: object
```

```
In [76]:  df3['New_item_Types'].value_counts()
```

```
Out[76]: New_item_Types
         Food              6121
         Non Consumable    1599
         Drinks             799
         Name: count, dtype: int64
```



In [81]:

```
df3.head()
```

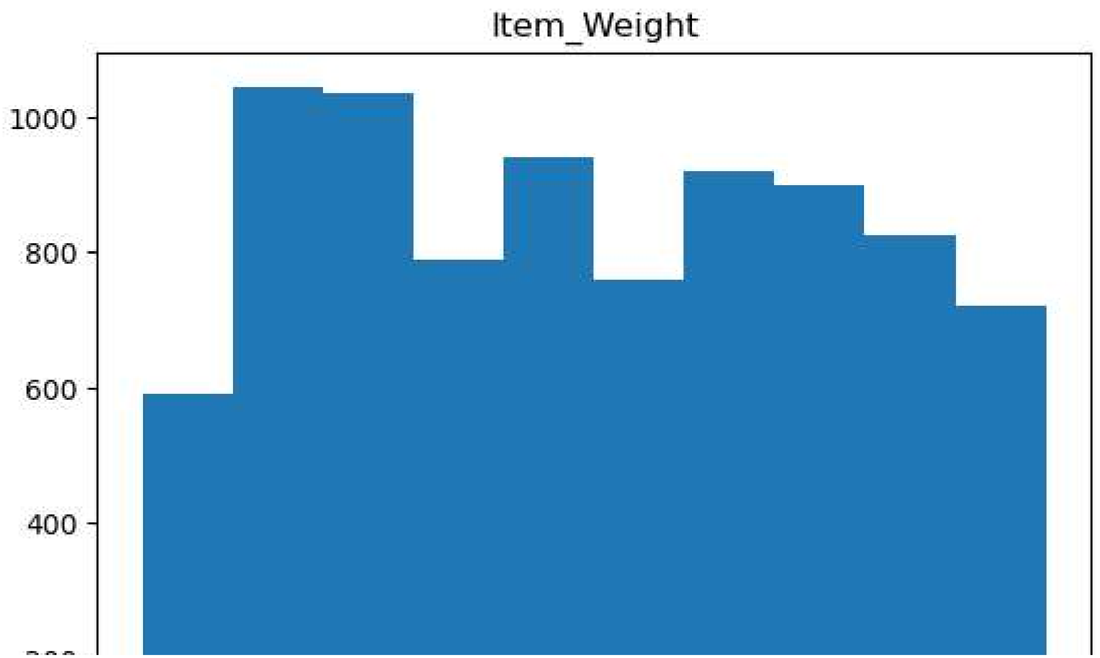
Out[81]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	O
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	0.066112	Fruits and Vegetables	182.0950	OUT010	1998	Small	
4	NCD19	8.93	Low Fat	0.066112	Household	53.8614	OUT013	1987	High	

EDA

In [83]:

```
for col in num_col:
    plt.hist(df3[col])
    plt.title(col)
    plt.show()
```



num_col is not normally distributed

In [85]:
```
df3.head()
```

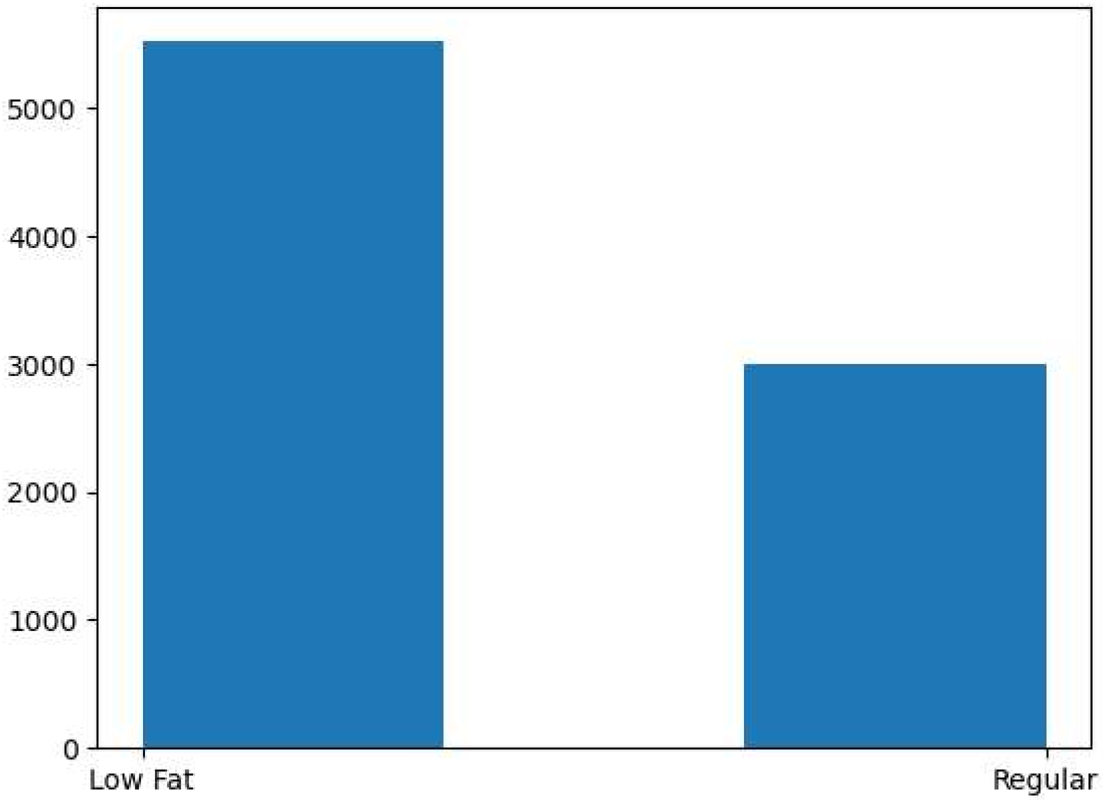
Out[85]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	O
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	0.066112	Fruits and Vegetables	182.0950	OUT010	1998	Small	
4	NCD19	8.93	Low Fat	0.066112	Household	53.8614	OUT013	1987	High	

In [136]:
```
df3=df3.drop(columns = ['Item_Identifier'])
```

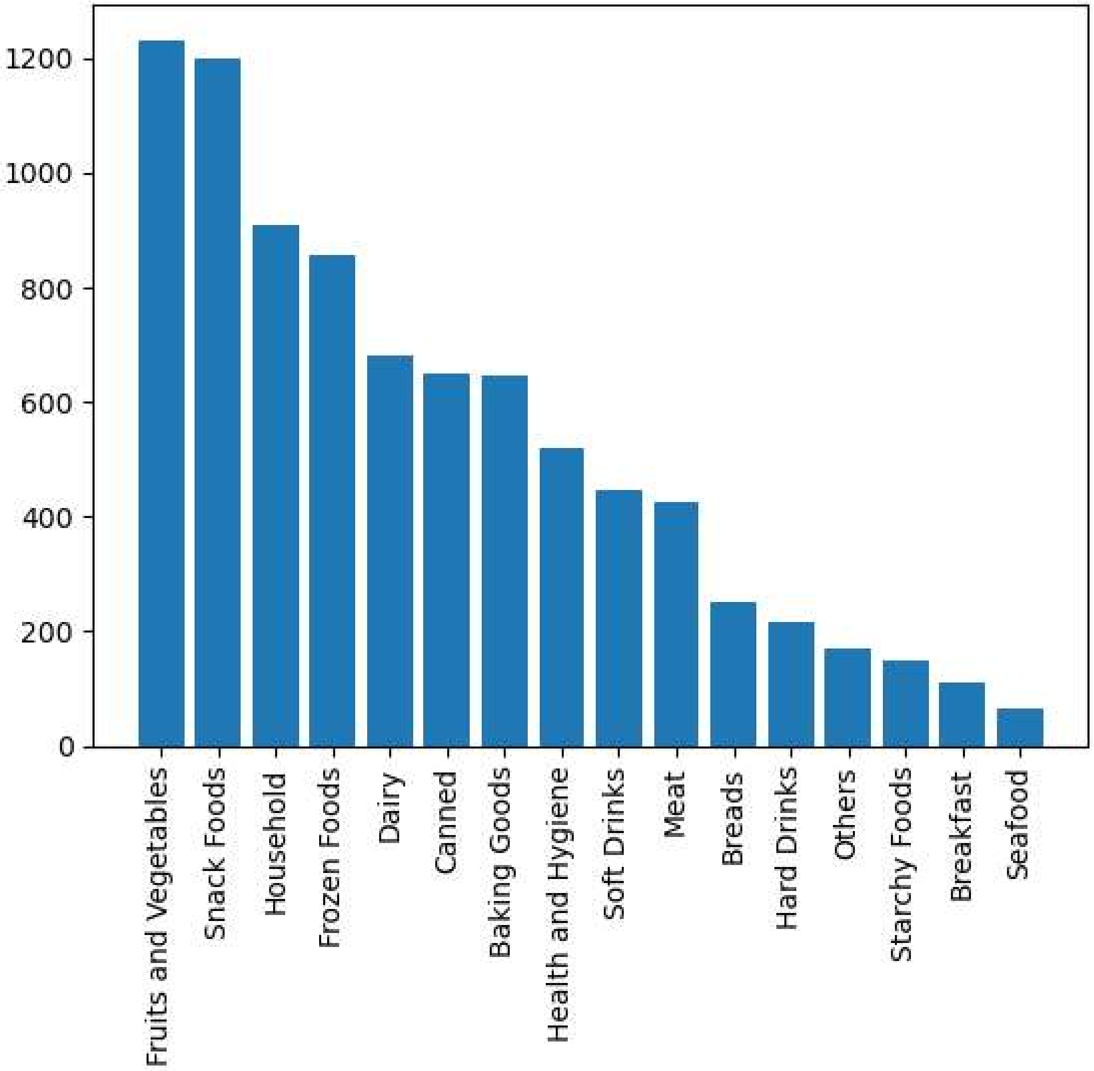
```
In [93]: plt.hist(df3['Item_Fat_Content'],bins = 3)
```

```
Out[93]: (array([5516.,    0., 3003.]),
          array([0.        , 0.33333333, 0.66666667, 1.        ]),
          <BarContainer object of 3 artists>)
```



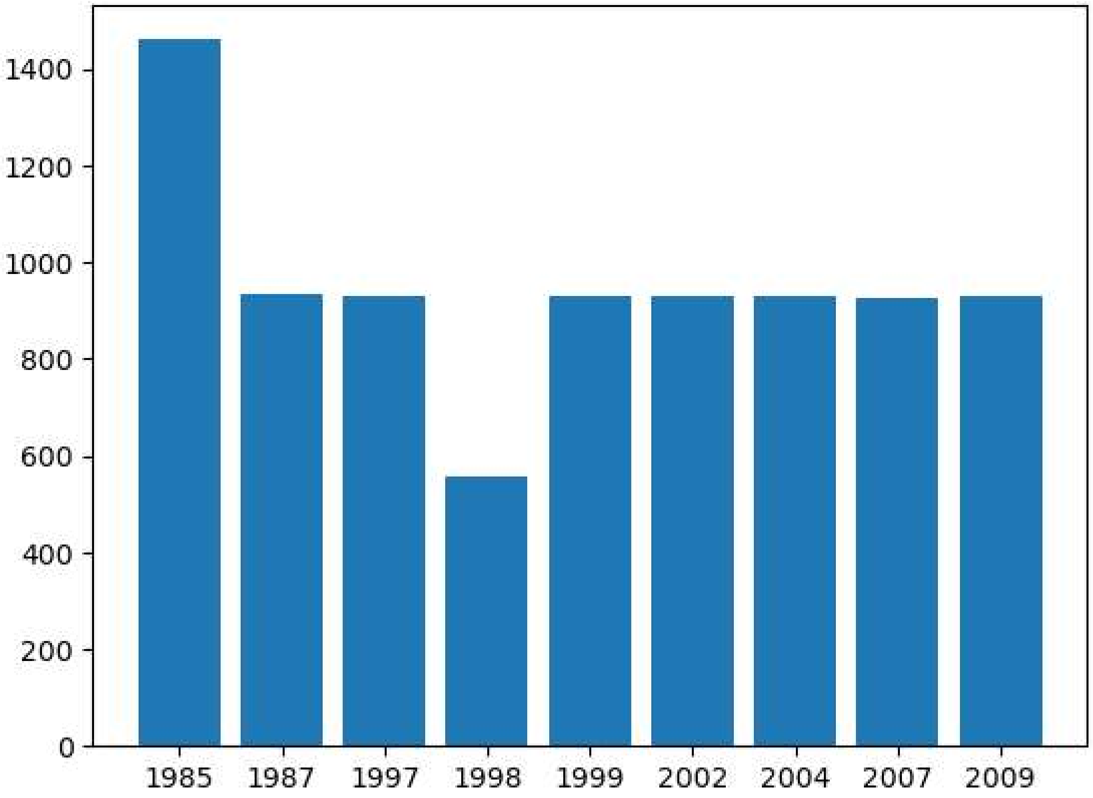
```
In [120]: count = df3['Item_Type'].value_counts()
          plt.bar(count.index,count.values)
          plt.xticks(rotation = 'vertical')
```

```
Out[120]: ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15],
           [Text(0, 0, 'Fruits and Vegetables'),
            Text(1, 0, 'Snack Foods'),
            Text(2, 0, 'Household'),
            Text(3, 0, 'Frozen Foods'),
            Text(4, 0, 'Dairy'),
            Text(5, 0, 'Canned'),
            Text(6, 0, 'Baking Goods'),
            Text(7, 0, 'Health and Hygiene'),
            Text(8, 0, 'Soft Drinks'),
            Text(9, 0, 'Meat'),
            Text(10, 0, 'Breads'),
            Text(11, 0, 'Hard Drinks'),
            Text(12, 0, 'Others'),
            Text(13, 0, 'Starchy Foods'),
            Text(14, 0, 'Breakfast'),
            Text(15, 0, 'Seafood')])
```


1200
1000
800
600
400
200
0
Fruits and Vegetables
Snack Foods
Household
Frozen Foods
Dairy
Canned
Baking Goods
Health and Hygiene
Soft Drinks
Meat
Breads
Hard Drinks
Others
Starchy Foods
Breakfast
Seafood

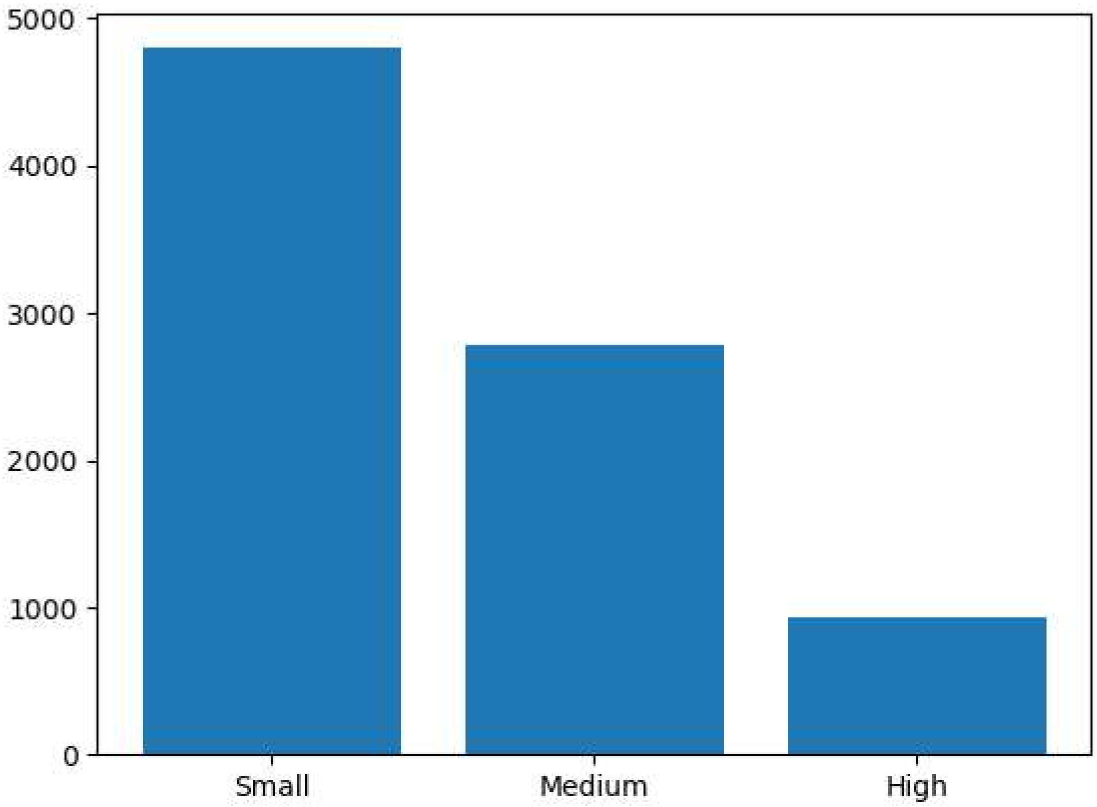
In [123]:
```
count = df3['Outlet_Establishment_Year'].value_counts().sort_index()
plt.bar(count.index.astype(str),count.values)
```

Out[123]:
```
<BarContainer object of 9 artists>
```



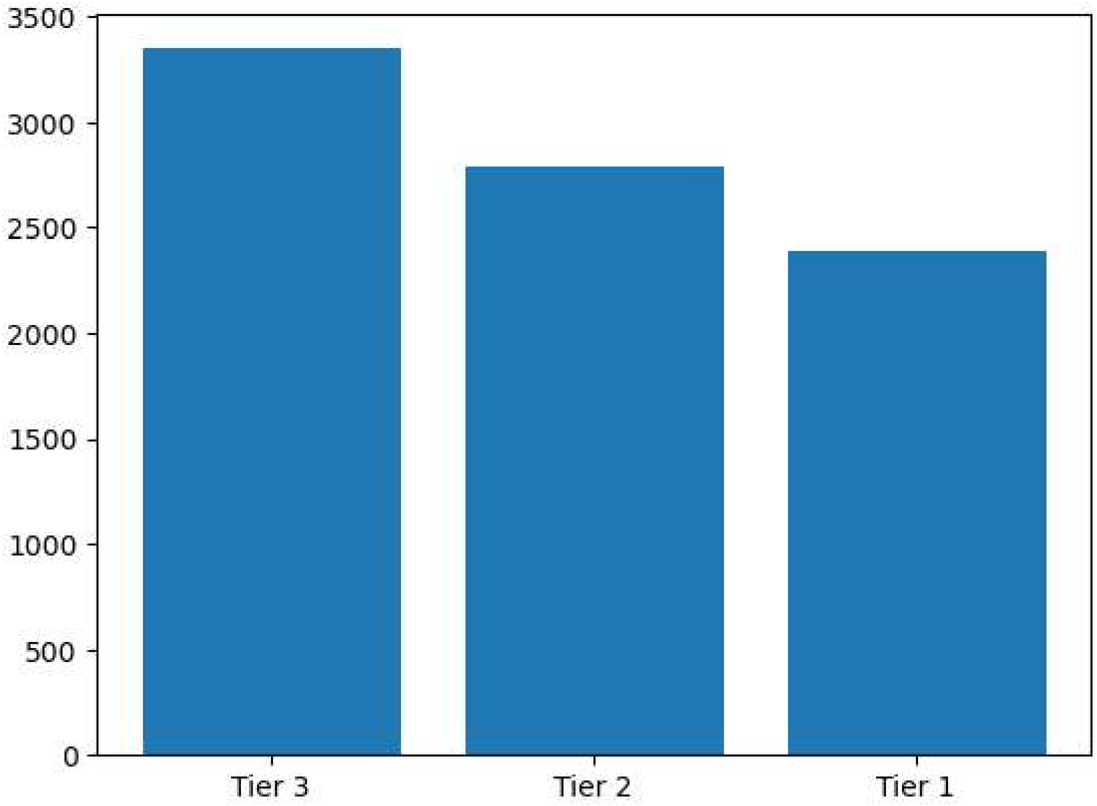
```
In [124]: count = df3['Outlet_Size'].value_counts()
          plt.bar(count.index,count.values)
```

```
Out[124]: <BarContainer object of 3 artists>
```



```
In [126]: count = df3['Outlet_Location_Type'].value_counts()
          plt.bar(count.index,count.values)

Out[126]: <BarContainer object of 3 artists>
```

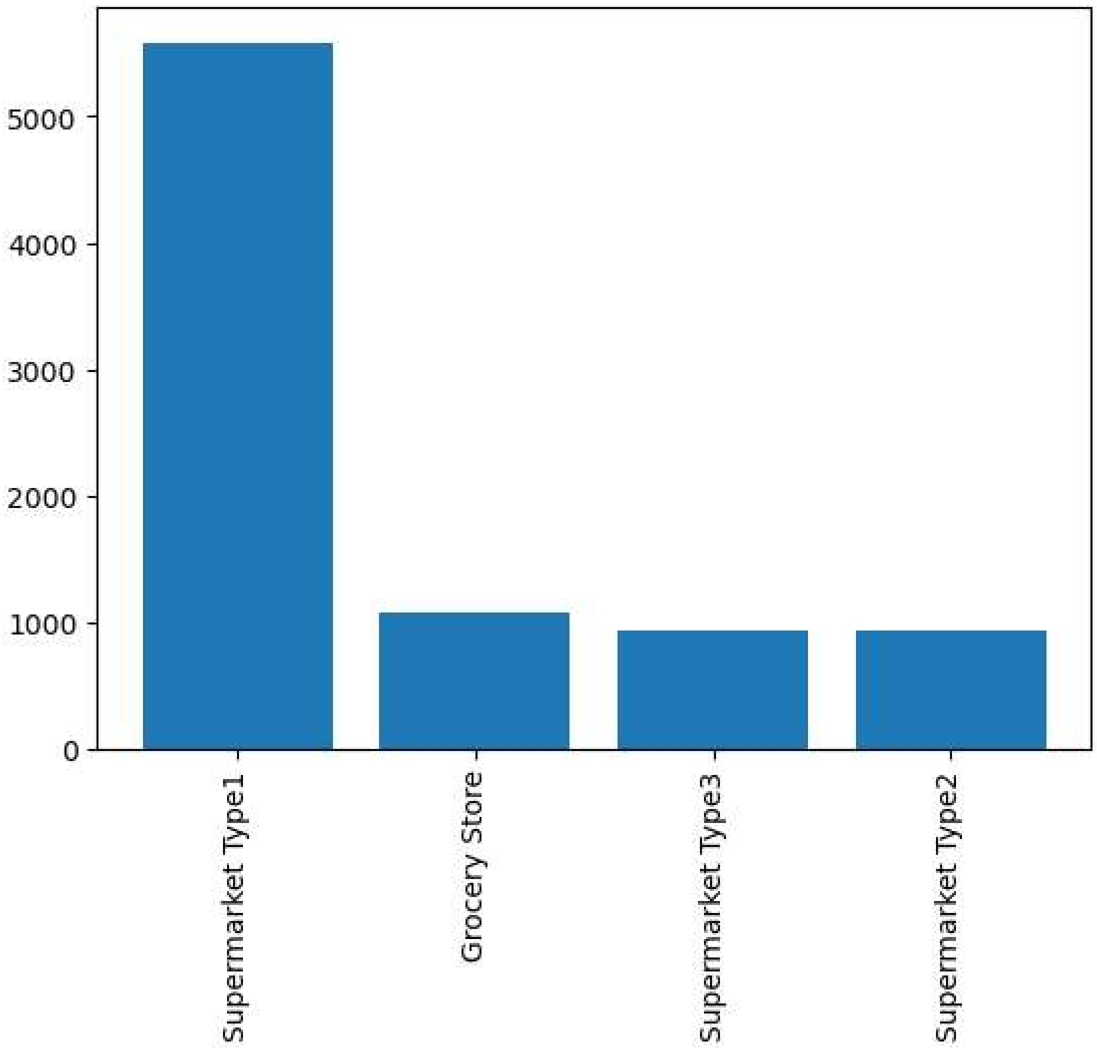


In [128]:

```
count = df3['Outlet_Type'].value_counts()
plt.bar(count.index,count.values)
plt.xticks(rotation='vertical')
```

Out[128]:

```
([0, 1, 2, 3],
 [Text(0, 0, 'Supermarket Type1'),
  Text(1, 0, 'Grocery Store'),
  Text(2, 0, 'Supermarket Type3'),
  Text(3, 0, 'Supermarket Type2')])
```



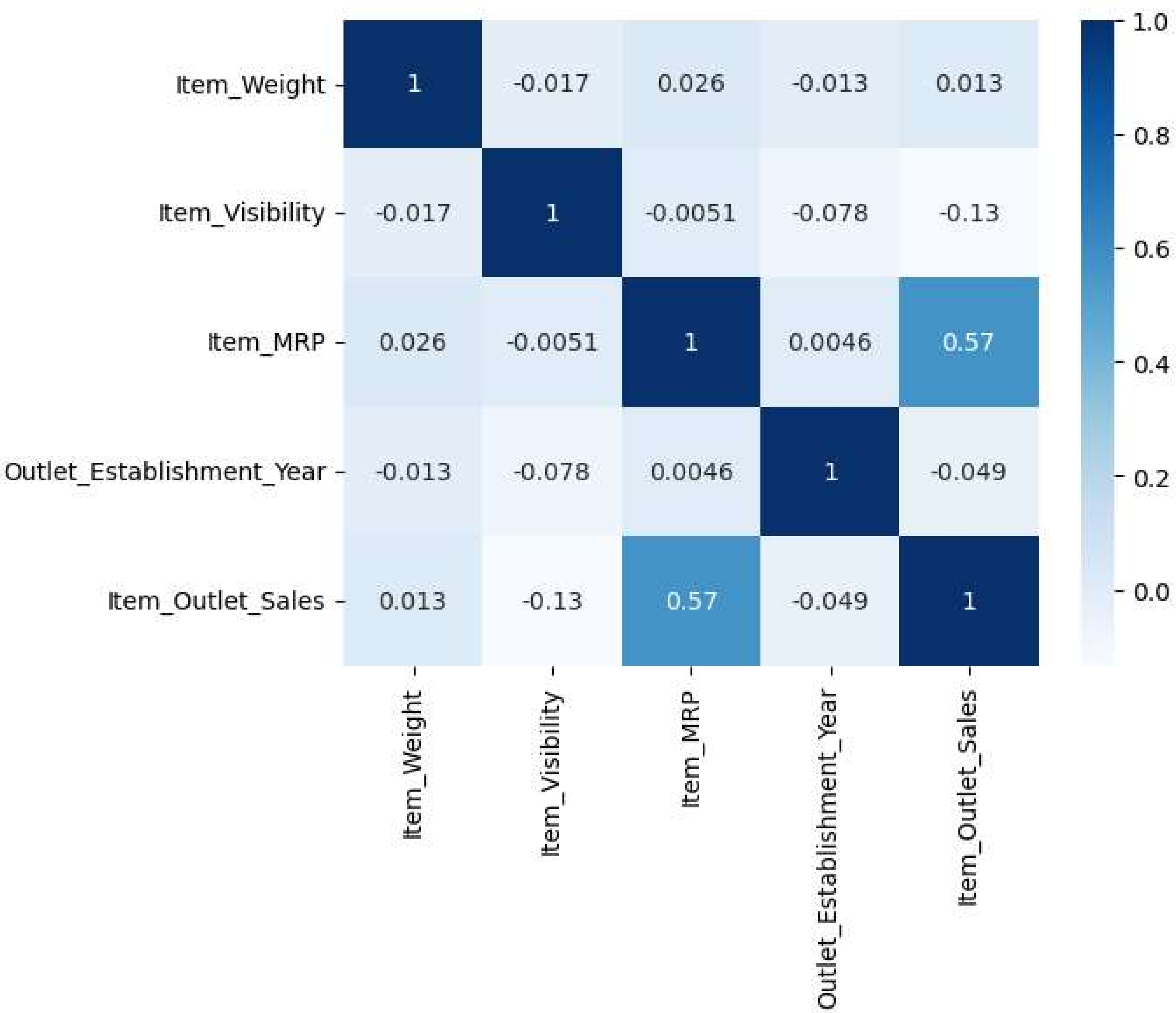
In [130]:

```
# Check corr()

corr =df3.corr(numeric_only=True)
```

```
In [133]: sns.heatmap(corr,annot=True,cmap='Blues')
```

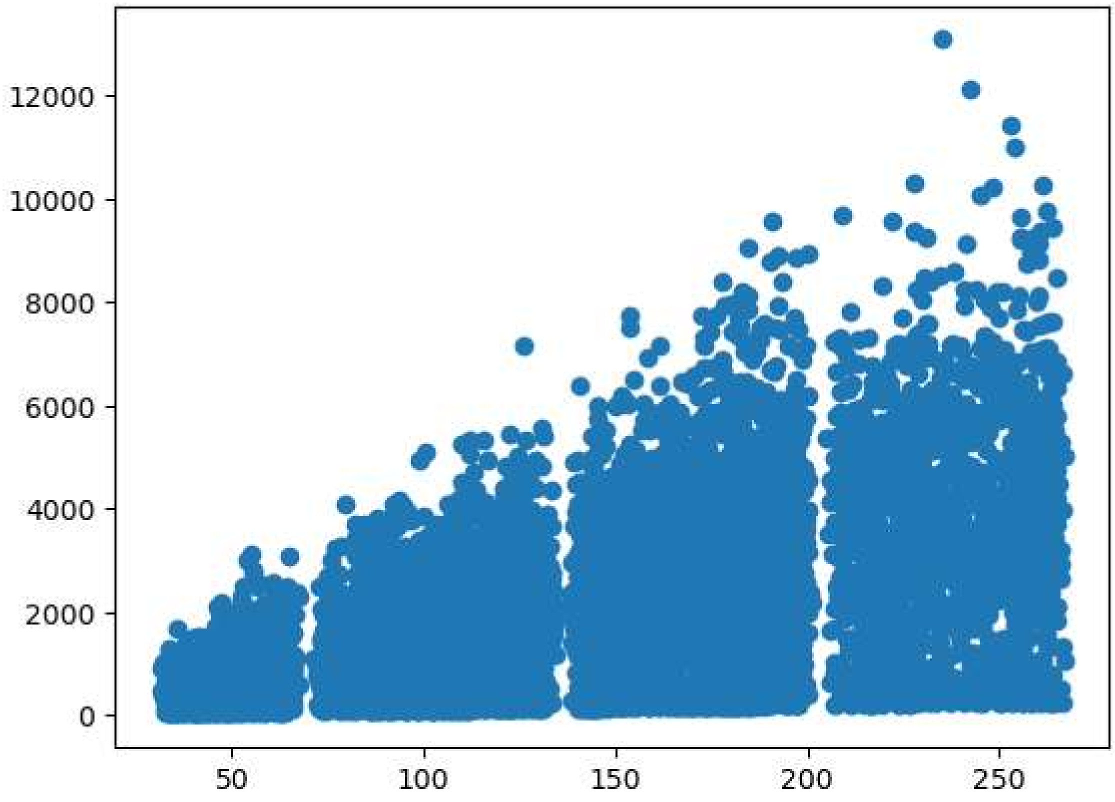
```
Out[133]: <Axes: >
```



highest corr indicated item_outlet_sales depends on item_MRP

```
In [134]: plt.scatter(df3['Item_MRP'],df3['Item_Outlet_Sales'])
```

```
Out[134]: <matplotlib.collections.PathCollection at 0x1f0ff3814d0>
```



Label Encoding

transfrom categorical columns in numerical code

In [137]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [138]:
```
df3.head(1)
```

Out[138]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_T
0	9.3	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Ti

In [139]:
```
cat_columns = ['Item_Fat_Content','Item_Type','Outlet_Identifier',
              'Outlet_Size','Outlet_Location_Type',
               'Outlet_Type','New_item_Types']
```

In [140]:
```
for col in cat_columns:
    df3[col]= le.fit_transform(df3[col])
```

In [141]:
```
df3.head()
```

Out[141]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_T
0	9.30	0	0.016047	4	249.8092	9	1999	1	
1	5.92	1	0.019278	14	48.2692	3	2009	1	
2	17.50	0	0.016760	10	141.6180	9	1999	1	
3	19.20	1	0.066112	6	182.0950	0	1998	2	
4	8.93	0	0.066112	9	53.8614	1	1987	0	

Feature Scaling

In [143]:

```
X = df3.drop(columns = ['Item_Outlet_Sales'])
y = df3['Item_Outlet_Sales']
```

In [146]:

```
X.head()
```

Out[146]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_T
0	9.30	0	0.016047	4	249.8092	9	1999	1	
1	5.92	1	0.019278	14	48.2692	3	2009	1	
2	17.50	0	0.016760	10	141.6180	9	1999	1	
3	19.20	1	0.066112	6	182.0950	0	1998	2	
4	8.93	0	0.066112	9	53.8614	1	1987	0	

In [148]:

```
fs= ['Item_Weight','Item_Visibility','Item_MRP','Outlet_Establishment_Year']
```

In [165]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X[fs])
X[fs]= scaler.transform(X[fs])
```

In [166]:
```
X.head()
```

Out[166]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_T
0	-0.769598	0	-1.111258	4	1.746938	9	0.138865	1	
1	-1.497133	1	-1.044950	14	-1.489096	3	1.333806	1	
2	0.995427	0	-1.096630	10	0.009762	9	0.138865	1	
3	1.361347	1	-0.083776	6	0.659682	0	0.019371	2	
4	-0.849240	0	-0.083776	9	-1.399305	1	-1.295064	0	

Split Data

In [167]:
```
from sklearn.model_selection import train_test_split
```

In [169]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state= 42)
X_train.shape,X_test.shape
```

Out[169]:
```
((6815, 11), (1704, 11))
```

Model

In [170]:
```
from sklearn.linear_model import LinearRegression
```

In [171]:
```
lr = LinearRegression()
```

In [172]:
```
lr.fit(X_train,y_train)
```

Out[172]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [174]: y_pred = lr.predict(X_test)
          y_pred
```

```
Out[174]: array([1941.62402977, 1876.23152175,   74.59680573, ..., 1624.25136357,
                 1246.5501686 , 2329.26045015])
```

```
In [175]: from sklearn.metrics import r2_score,mean_squared_error
```

```
In [180]: lrr2=r2_score(y_test,y_pred)
          lrr2
```

```
Out[180]: 0.5158031404223624
```

```
In [181]: lrmse=mean_squared_error(y_test,y_pred)
          lrmse
```

```
Out[181]: 1439066.043709318
```

```
In [194]: from sklearn.model_selection import cross_val_score

          lrcv_scores = cross_val_score(lr, X, y, cv=5, scoring="r2")

          lrcv_mean=lrcv_scores.mean()
```

```
In [195]: print('***********Linear Regressor***********')
          print(lrr2)
          print(lrmse)
          print(lrcv_mean)
```

```
***********Linear Regressor***********
5158000000.0
143906604370000.0
0.5014571607718897
```

In [190]:

```
from sklearn.linear_model import Lasso

lasso = Lasso()
lasso.fit(X_train, y_train)
y_pred = lasso.predict(X_test)

lasso_r2 = r2_score(y_test, y_pred)
lasso_mse = mean_squared_error(y_test, y_pred)

lasso_cv_scores = cross_val_score(lasso, X, y, cv=5, scoring="r2")
lasso_cv_mean = lasso_cv_scores.mean()
print('***********Lasso***********')
print(lasso_r2)
print(lasso_mse)
print(lasso_cv_mean)
```

```
***********Lasso***********
0.5157888225109288
1439108.5975172436
0.5015200552798098
```

In [196]:

```
from sklearn.linear_model import Ridge

ridge = Ridge()
ridge.fit(X_train, y_train)
y_pred = ridge.predict(X_test)

ridge_r2 = r2_score(y_test, y_pred)
ridge_mse = mean_squared_error(y_test, y_pred)

ridge_cv_scores = cross_val_score(ridge, X, y, cv=5, scoring="r2")
ridge_cv_mean = ridge_cv_scores.mean()

print('***********Ridge Model***********')
print(ridge_r2)
print(ridge_mse)
print(ridge_cv_mean)
```

```
***********Ridge Model***********
0.5157971131978764
1439083.9570309843
0.5014580099527018
```

```
In [198]: from sklearn.tree import DecisionTreeRegressor

          dt = DecisionTreeRegressor()
          dt.fit(X_train, y_train)
          y_pred = dt.predict(X_test)

          dt_r2 = r2_score(y_test, y_pred)
          dt_mse = mean_squared_error(y_test, y_pred)

          dt_cv_scores = cross_val_score(dt, X, y, cv=5, scoring="r2")
          dt_cv_mean = dt_cv_scores.mean()

          print('***********Decision Tree  Model***********')
          print(dt_r2)
          print(dt_mse)
          print(dt_cv_mean)
```

```
***********Decision Tree  Model***********
0.17755444432860312
2444364.205498474
0.15772872495963644
```

```
In [199]: from sklearn.ensemble import RandomForestRegressor

          rf = RandomForestRegressor()
          rf.fit(X_train, y_train)
          y_pred = rf.predict(X_test)

          rf_r2 = r2_score(y_test, y_pred)
          rf_mse = mean_squared_error(y_test, y_pred)

          rf_cv_scores = cross_val_score(rf, X, y, cv=5, scoring="r2")
          rf_cv_mean = rf_cv_scores.mean()

          print('***********Random Forest  Model***********')
          print(rf_r2)
          print(rf_mse)
          print(rf_cv_mean)
```

```
***********Random Forest  Model***********
0.5606845971979322
1305675.2974449382
0.5549706579397411
```

In [200]:

```
from sklearn.ensemble import ExtraTreesRegressor

et = ExtraTreesRegressor()
et.fit(X_train, y_train)
y_pred = et.predict(X_test)

et_r2 = r2_score(y_test, y_pred)
et_mse = mean_squared_error(y_test, y_pred)

et_cv_scores = cross_val_score(et, X, y, cv=5, scoring="r2")
et_cv_mean = et_cv_scores.mean()

print('***********Extra Tree Model***********')
print(et_r2)
print(et_mse)
print(et_cv_mean)
```

```
***********Extra Tree Model***********
0.5266262212734284
1406899.1102957558
0.5261435997075912
```

In [201]:

```
from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

knn_r2 = r2_score(y_test, y_pred)
knn_mse = mean_squared_error(y_test, y_pred)

knn_cv_scores = cross_val_score(knn, X, y, cv=5, scoring="r2")
knn_cv_mean = knn_cv_scores.mean()

print('***********KNeighbours  Model***********')
print(knn_r2)
print(knn_mse)
print(knn_cv_mean)
```

```
***********KNeighbours  Model***********
0.4875300555728299
1523095.5774683047
0.49887142862659317
```

```
In [205]: from sklearn.svm import SVR

          svr = SVR()
          svr.fit(X_train, y_train)
          y_pred = svr.predict(X_test)

          svr_r2 = r2_score(y_test, y_pred)
          svr_mse = mean_squared_error(y_test, y_pred)

          svr_cv_scores = cross_val_score(svr, X, y, cv=5, scoring="r2")
          svr_cv_mean = svr_cv_scores.mean()

          print('***********SVR Model***********')
          print(svr_r2)
          print(svr_mse)
          print(svr_cv_mean)
```

```
***********SVR Model***********
-0.004585186347650705
2985695.5927953497
-0.0028889858746814047
```

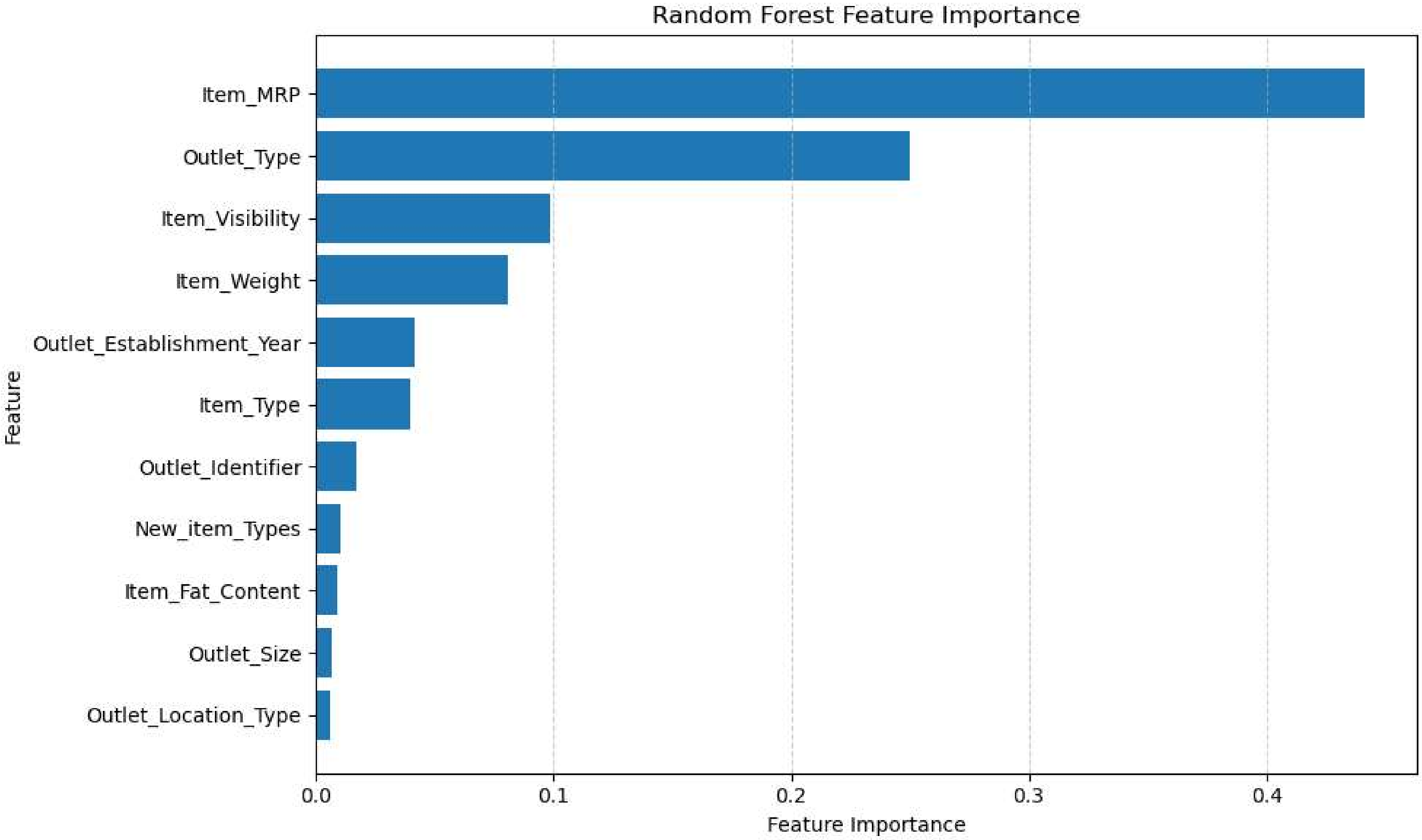
Random Forest Proved To Be Best Model

Feature Importance

In [212]:

```
feature_importances = rf.feature_importances_
feature_names = X.columns
feature_importances, feature_names = zip(*sorted(zip(feature_importances, feature_names), reverse=False))

plt.figure(figsize=(10, 6))
plt.barh(feature_names, feature_importances)
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Random Forest Feature Importance")
plt.grid(axis="x", linestyle="--", alpha=0.6)
plt.tight_layout()
plt.show()
```


Random Forest Feature Importance
Item_MRP
Outlet_Type
Item_Visibility
Item_Weight
Outlet_Establishment_Year
Item_Type
Outlet_Identifier
New_item_Types
Item_Fat_Content
Outlet_Size
Outlet_Location_Type
Feature
0.0
0.1
0.2
0.3
0.4
Feature Importance

Model Predictive System

In [213]:
```
X.columns
```

Out[213]:
```
Index(['Item_Weight', 'Item_Fat_Content', 'Item_Visibility', 'Item_Type',
       'Item_MRP', 'Outlet_Identifier', 'Outlet_Establishment_Year',
       'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type', 'New_item_Types'],
      dtype='object')
```

In [215]:
```
X.head(1)
```

Out[215]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_T
0	-0.769598	0	-1.111258	4	1.746938	9	0.138865	1	

In [223]:

```
new_item =np.array([
    float(input('Enter Item Weight =')),
    float(input("Enter Item Fat Content = ")),
    float(input("Enter Item Visibility = ")),
    int(input('Enter Item Type =')),
    float(input("Enter Item MRP = ")),
    input("Enter Outlet Identifier = "),
    int(input("Enter Outlet Establishment Year = ")),
    input("Enter Outlet Size = "),
    input("Enter Outlet Location Type = "),
    input("Enter Outlet Type = "),
    int(input("Enter New Item Types = "))])

new_item = new_item.reshape(1, -1)

predict = rf.predict(new_item)
print('Sales Prediction By Model =',predict[0])
```

```
Enter Item Weight =14.25
Enter Item Fat Content = 0.52
Enter Item Visibility = 0.12
Enter Item Type =4
Enter Item MRP = 199.99
Enter Outlet Identifier = 9
Enter Outlet Establishment Year = 1997
Enter Outlet Size = 1
Enter Outlet Location Type = 0
Enter Outlet Type = 1
Enter New Item Types = 1
Sales Prediction By Model = 4726.494225999997
```

In [220]:

```
df3.describe()
```

Out[220]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Lo
count	8519.000000	8519.000000	8519.000000	8519.000000	8519.000000	8519.000000	8519.000000	8519.000000	
mean	12.875420	0.352506	0.070194	7.227491	141.010019	4.722268	1997.837892	1.453692	
std	4.646098	0.477779	0.048729	4.209571	62.283594	2.837852	8.369105	0.683166	
min	4.555000	0.000000	0.003575	0.000000	31.290000	0.000000	1985.000000	0.000000	
25%	8.785000	0.000000	0.033085	4.000000	93.844900	2.000000	1987.000000	1.000000	
50%	12.650000	0.000000	0.062511	6.000000	143.047000	5.000000	1999.000000	2.000000	
75%	16.850000	1.000000	0.094558	10.000000	185.676600	7.000000	2004.000000	2.000000	
max	21.350000	1.000000	0.328391	15.000000	266.888400	9.000000	2009.000000	2.000000	

In [ ]: