Bigmart Sales Prediction Analysis and Regression

Dataset Information

The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store.

Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales.

Variable	Description
Item_Identifier	Unique product ID
Item_Weight	Weight of product
Item_Fat_Content	Whether the product is low fat or not
Item_Visibility	The % of total display area of all products in a store allocated to the particular product
Item_Type	The category to which the product belongs
Item_MRP	Maximum Retail Price (list price) of the product
Outlet_Identifier	Unique store ID
Outlet_Establishment_Year	The year in which store was established
Outlet_Size	The size of the store in terms of ground area covered
Outlet_Location_Type	The type of city in which the store is located
Outlet_Type	Whether the outlet is just a grocery store or some sort of supermarket
Item_Outlet_Sales	Sales of the product in the particulat store. This is the outcome variable to be predicted.

Import modules

In [48]:

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

Loading the dataset

In [3]:

```
#loading the dataset in pandas dataframe
df = pd.read_csv('Train.csv')

#check first five rows of the dataset
df.head()
```

Out[3]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Ou
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	

In [4]:

```
# statistical info
df.describe()
```

Out[4]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sale
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.00000
mean	12.857645	0.066132	140.992782	1997.831867	2181.28891
std	4.643456	0.051598	62.275067	8.371760	1706.49961
min	4.555000	0.000000	31.290000	1985.000000	33.29000
25%	8.773750	0.026989	93.826500	1987.000000	834.24740
50%	12.600000	0.053931	143.012800	1999.000000	1794.33100
75%	16.850000	0.094585	185.643700	2004.000000	3101.29640
max	21.350000	0.328391	266.888400	2009.000000	13086.96480

In [5]:

```
# datatype of attributes
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]:

```
# check unique values in dataset
df.apply(lambda x: len(x.unique()))
```

Out[6]:

```
Item_Identifier              1559
Item_Weight                   416
Item_Fat_Content                5
Item_Visibility              7880
Item_Type                      16
Item_MRP                     5938
Outlet_Identifier              10
Outlet_Establishment_Year       9
Outlet_Size                     4
Outlet_Location_Type            3
Outlet_Type                     4
Item_Outlet_Sales            3493
dtype: int64
```

Preprocessing the dataset

In [7]:

```
# check for null values
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]:

```
# check for categorical attributes
cat_col = []
for x in df.dtypes.index:
    if df.dtypes[x] == 'object':
        cat_col.append(x)
cat_col
```

Out[8]:

```
['Item_Identifier',
 'Item_Fat_Content',
 'Item_Type',
 'Outlet_Identifier',
 'Outlet_Size',
 'Outlet_Location_Type',
 'Outlet_Type']
```

In [9]:

```
cat_col.remove('Item_Identifier')
cat_col.remove('Outlet_Identifier')
cat_col
```

Out[9]:

```
['Item_Fat_Content',
 'Item_Type',
 'Outlet_Size',
 'Outlet_Location_Type',
 'Outlet_Type']
```

In [10]:

```
# print the categorical columns
for col in cat_col:
    print(col)
    print(df[col].value_counts())
    print()
```

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

Outlet_Size
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64

Outlet_Location_Type
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: Outlet_Location_Type, dtype: int64

Outlet_Type
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: Outlet_Type, dtype: int64
```

In [11]:

```
# fill the missing values
item_weight_mean = df.pivot_table(values = "Item_Weight", index = '
item_weight_mean
```

Out[11]:

Item_Identifier	Item_Weight
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

1555 rows × 1 columns

In [12]:

```
miss_bool = df['Item_Weight'].isnull()
miss_bool
```

Out[12]:

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

In [20]:

```
for i, item in enumerate(df['Item_Identifier']):
    if miss_bool[i]:
        if item in item_weight_mean:
            df['Item_Weight'][i] = item_weight_mean.loc[item]['Item
        else:
            df['Item_Weight'][i] = np.mean(df['Item_Weight'])
```

In [16]:
```
df['Item_Weight'].isnull().sum()
```

Out[16]:
```
0
```

In [17]:
```
#using Pivot_table
outlet_size_mode = df.pivot_table(values='Outlet_Size', columns='Ou
outlet_size_mode
```

Out[17]:

Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
Outlet_Size	Small	Small	Medium	Medium

In [18]:
```
miss_bool = df['Outlet_Size'].isnull()
df.loc[miss_bool, 'Outlet_Size'] = df.loc[miss_bool, 'Outlet_Type']
```

In [19]:
```
df['Outlet_Size'].isnull().sum()
```

Out[19]:
```
0
```

In [21]:
```
sum(df['Item_Visibility']==0)
```

Out[21]:
```
526
```

In [22]:
```
# replace zeros with mean
df.loc[:, 'Item_Visibility'].replace([0], [df['Item_Visibility'].me
```

In [23]:
```
sum(df['Item_Visibility']==0)
```

Out[23]:
```
0
```

In [24]:
```
# combine item fat content
df['Item_Fat_Content'] = df['Item_Fat_Content'].replace({'LF':'Low
df['Item_Fat_Content'].value_counts()
```

Out[24]:
```
Low Fat    5517
Regular    3006
Name: Item_Fat_Content, dtype: int64
```

Creation of New Attributes

In [25]:
```
df['New_Item_Type'] = df['Item_Identifier'].apply(lambda x: x[:2])
df['New_Item_Type']
```

Out[25]:
```
0       FD
1       DR
2       FD
3       FD
4       NC
        ..
8518    FD
8519    FD
8520    NC
8521    FD
8522    DR
Name: New_Item_Type, Length: 8523, dtype: object
```

In [26]:
```
df['New_Item_Type'] = df['New_Item_Type'].map({'FD':'Food', 'NC':'N
df['New_Item_Type'].value_counts()
```

Out[26]:
```
Food              6125
Non-Consumable    1599
Drinks             799
Name: New_Item_Type, dtype: int64
```

In [27]:
```
df.loc[df['New_Item_Type']=='Non-Consumable', 'Item_Fat_Content'] =
df['Item_Fat_Content'].value_counts()
```

Out[27]:
```
Low Fat       3918
Regular       3006
Non-Edible    1599
Name: Item_Fat_Content, dtype: int64
```

In [28]:
```
# create small values for establishment year
df['Outlet_Years'] = 2013 - df['Outlet_Establishment_Year']
```

In [29]:
```
df['Outlet_Years']
```

Out[29]:
```
0       14
1        4
2       14
3       15
4       26
        ..
8518    26
8519    11
8520     9
8521     4
8522    16
Name: Outlet_Years, Length: 8523, dtype: int64
```

In [30]:
```
df.head()
```

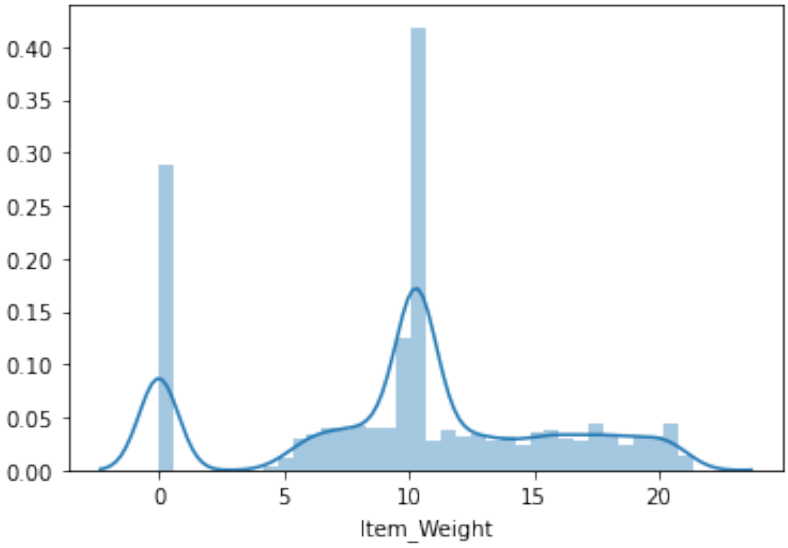
Out[30]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Ou
0	FDA15	9.30000	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92000	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50000	Low Fat	0.016760	Meat	141.6180	
3	FDX07	10.65059	Regular	0.066132	Fruits and Vegetables	182.0950	
4	NCD19	8.93000	Non-Edible	0.066132	Household	53.8614	

Exploratory Data Analysis

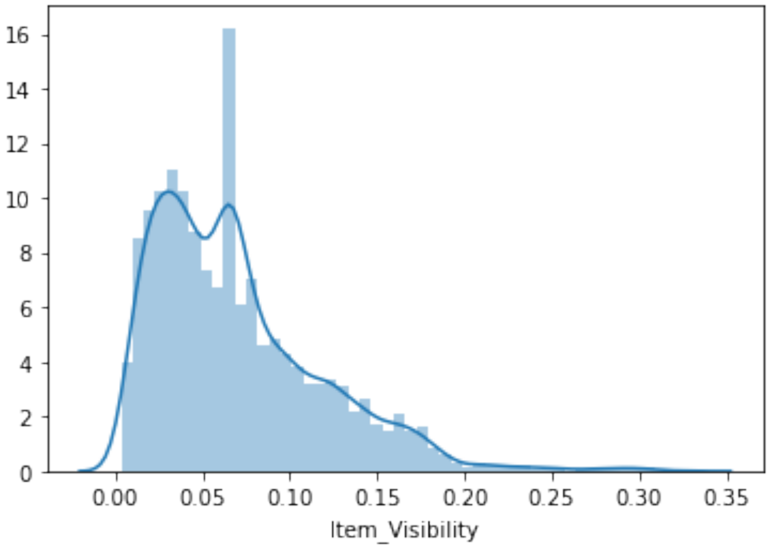
In [31]:
```
sns.distplot(df['Item_Weight'])
```

Out[31]:
```
<AxesSubplot:xlabel='Item_Weight'>
```



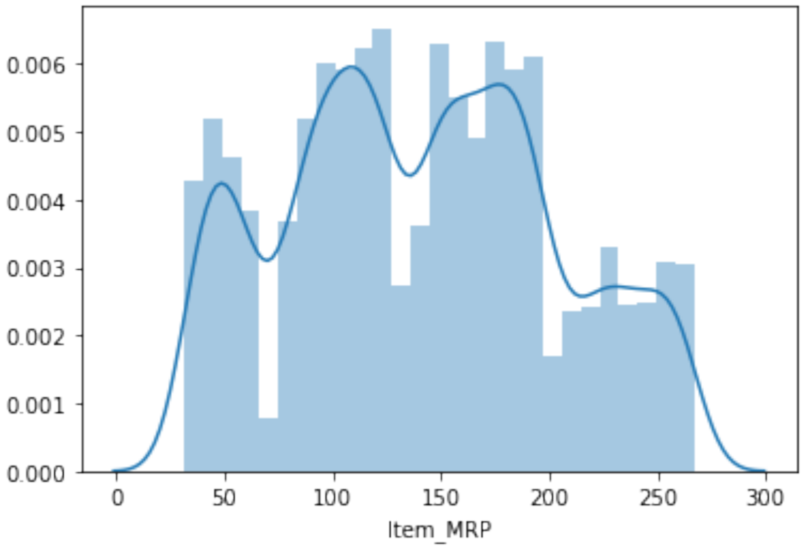
In [32]:
```
sns.distplot(df['Item_Visibility'])
```

Out[32]:
```
<AxesSubplot:xlabel='Item_Visibility'>
```



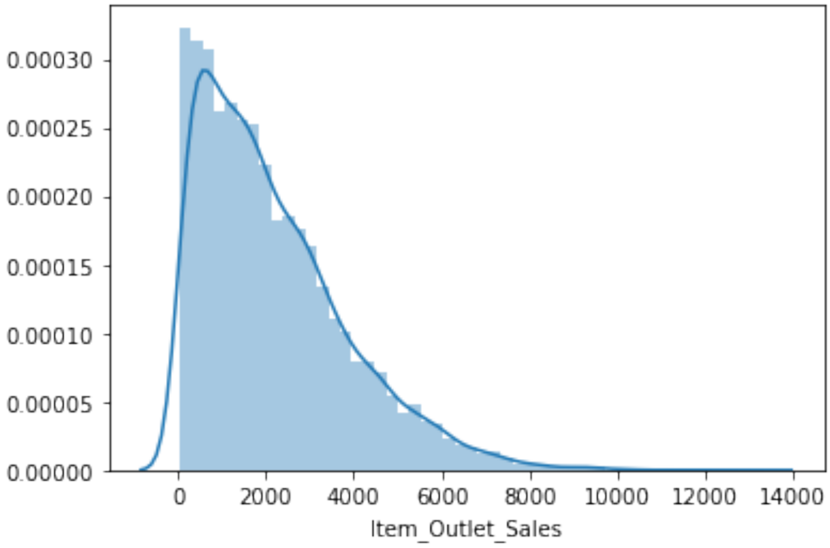
In [33]:
```
sns.distplot(df['Item_MRP'])
```

Out[33]:
```
<AxesSubplot:xlabel='Item_MRP'>
```



In [34]:
```
sns.distplot(df['Item_Outlet_Sales'])
```

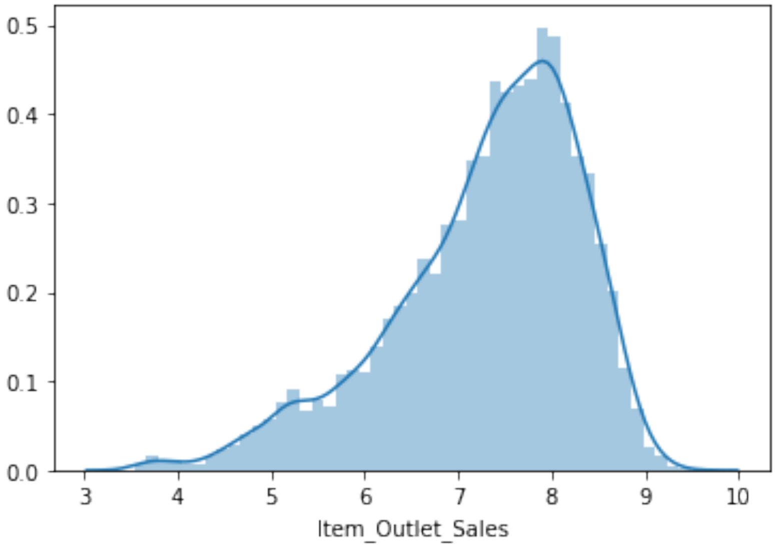
Out[34]:
```
<AxesSubplot:xlabel='Item_Outlet_Sales'>
```



In [35]:
```
# log transformation
df['Item_Outlet_Sales'] = np.log(1+df['Item_Outlet_Sales'])
```

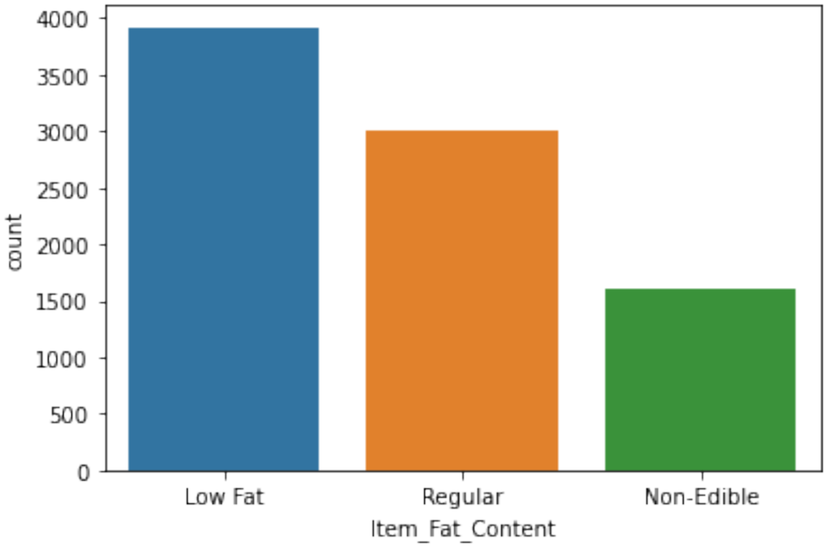
In [36]:
```
sns.distplot(df['Item_Outlet_Sales'])
```

Out[36]:
```
<AxesSubplot:xlabel='Item_Outlet_Sales'>
```



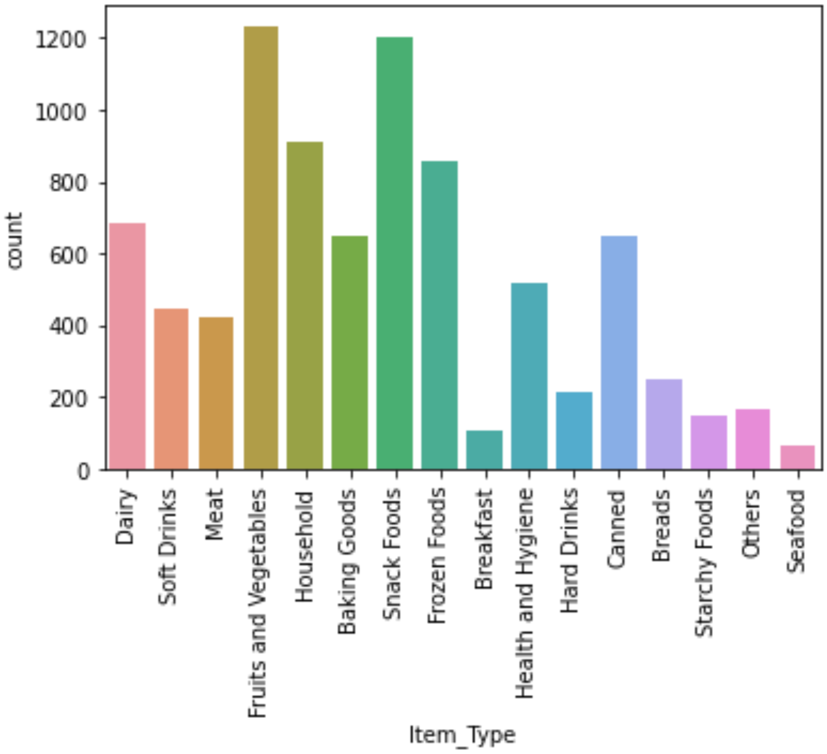
In [37]:
```
sns.countplot(df["Item_Fat_Content"])
```

Out[37]:
```
<AxesSubplot:xlabel='Item_Fat_Content', ylabel='count'>
```



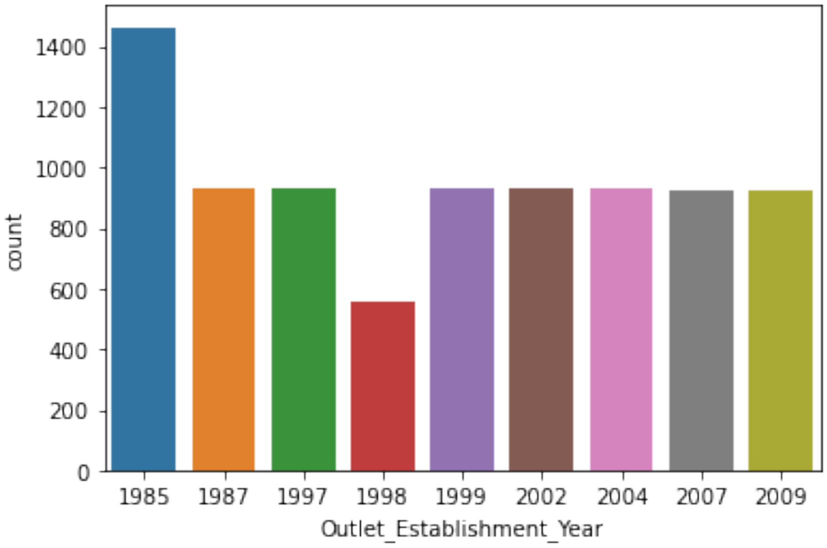
```
In [57]: # plt.figure(figsize=(15,5))
         l = list(df['Item_Type'].unique())
         chart = sns.countplot(df["Item_Type"])
         chart.set_xticklabels(labels=l, rotation=90)
```

```
Out[57]: [Text(0, 0, 'Dairy'),
          Text(1, 0, 'Soft Drinks'),
          Text(2, 0, 'Meat'),
          Text(3, 0, 'Fruits and Vegetables'),
          Text(4, 0, 'Household'),
          Text(5, 0, 'Baking Goods'),
          Text(6, 0, 'Snack Foods'),
          Text(7, 0, 'Frozen Foods'),
          Text(8, 0, 'Breakfast'),
          Text(9, 0, 'Health and Hygiene'),
          Text(10, 0, 'Hard Drinks'),
          Text(11, 0, 'Canned'),
          Text(12, 0, 'Breads'),
          Text(13, 0, 'Starchy Foods'),
          Text(14, 0, 'Others'),
          Text(15, 0, 'Seafood')]
```



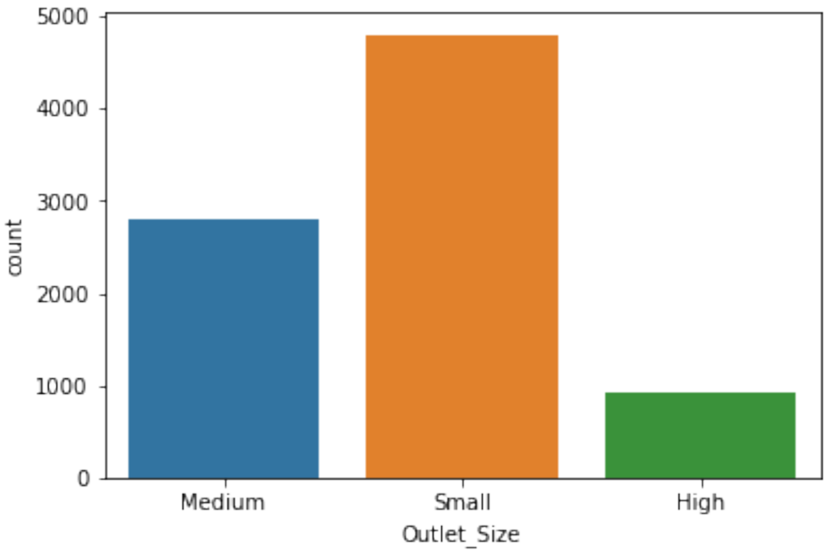
In [58]:
```
sns.countplot(df['Outlet_Establishment_Year'])
```

Out[58]:
```
<AxesSubplot:xlabel='Outlet_Establishment_Year', ylabel='count'>
```



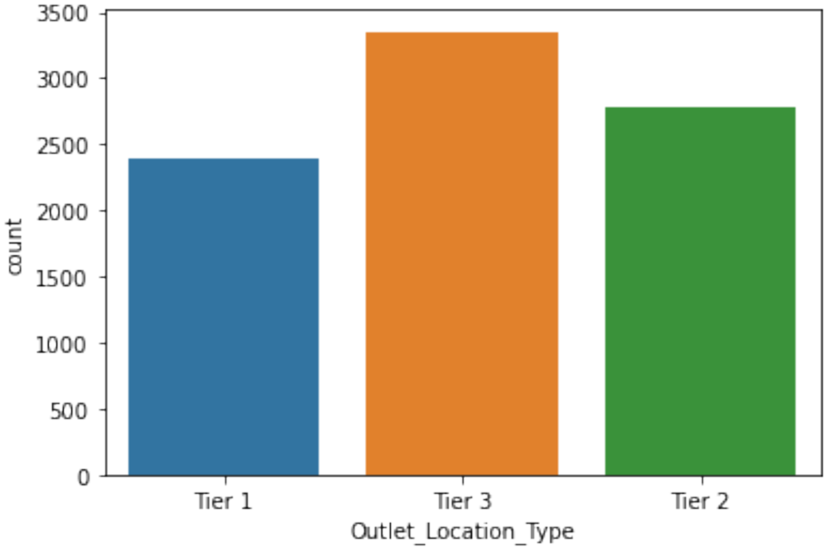
In [59]:
```
sns.countplot(df['Outlet_Size'])
```

Out[59]:
```
<AxesSubplot:xlabel='Outlet_Size', ylabel='count'>
```



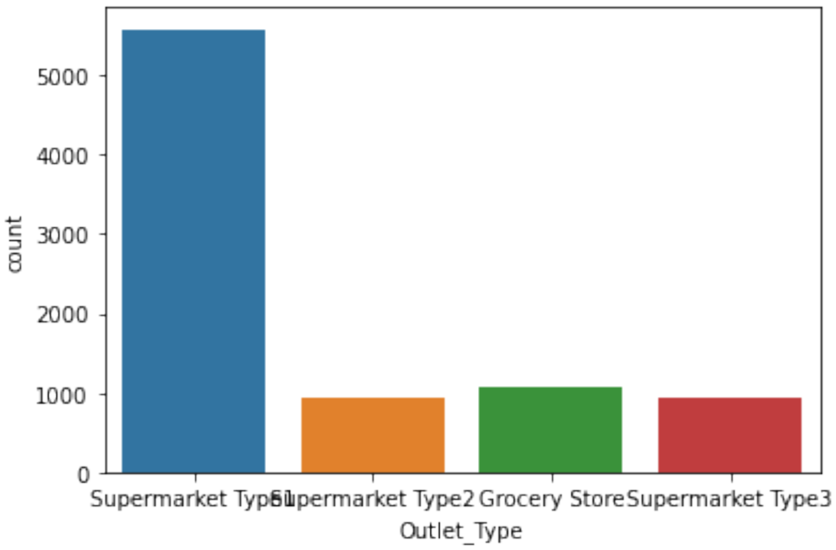
```
In [60]: sns.countplot(df['Outlet_Location_Type'])
```

```
Out[60]: <AxesSubplot:xlabel='Outlet_Location_Type', ylabel='count'>
```



```
In [61]: sns.countplot(df['Outlet_Type'])
```

```
Out[61]: <AxesSubplot:xlabel='Outlet_Type', ylabel='count'>
```



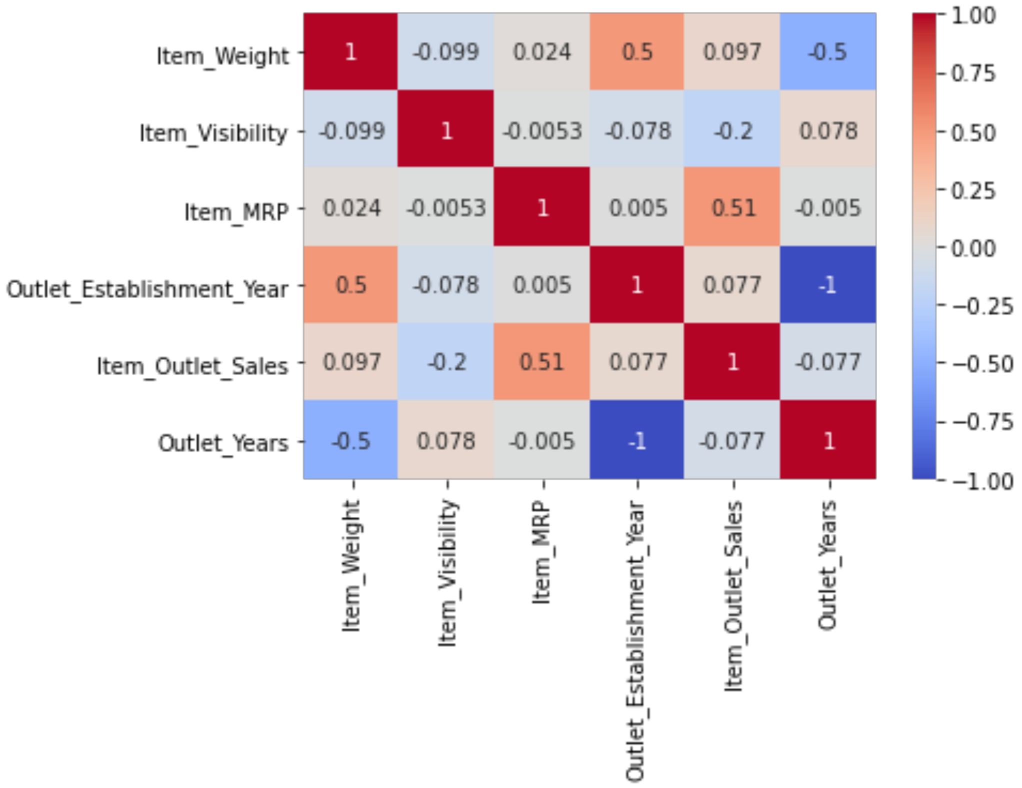
Coorelation Matrix

In [62]:

```
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[62]:

```
<AxesSubplot:>
```



In [63]:

```
df.head()
```

Out[63]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Ou
0	FDA15	9.30000	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92000	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50000	Low Fat	0.016760	Meat	141.6180	
3	FDX07	10.65059	Regular	0.066132	Fruits and Vegetables	182.0950	
4	NCD19	8.93000	Non-Edible	0.066132	Household	53.8614	

Label Encoding

In [64]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Outlet'] = le.fit_transform(df['Outlet_Identifier'])
cat_col = ['Item_Fat_Content', 'Item_Type', 'Outlet_Size', 'Outlet_
for col in cat_col:
    df[col] = le.fit_transform(df[col])
```

Onehot Encoding

In [65]:

```
df = pd.get_dummies(df, columns=['Item_Fat_Content', 'Outlet_Size',
df.head()
```

Out[65]:

	Item_Identifier	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outle
0	FDA15	9.30000	0.016047	4	249.8092	OUT049	
1	DRC01	5.92000	0.019278	14	48.2692	OUT018	
2	FDN15	17.50000	0.016760	10	141.6180	OUT049	
3	FDX07	10.65059	0.066132	6	182.0950	OUT010	
4	NCD19	8.93000	0.066132	9	53.8614	OUT013	

5 rows × 26 columns

Input Split

In [66]:

```
#spliting the dataset
X = df.drop(columns=['Outlet_Establishment_Year', 'Item_Identifier'
y = df['Item_Outlet_Sales']
```

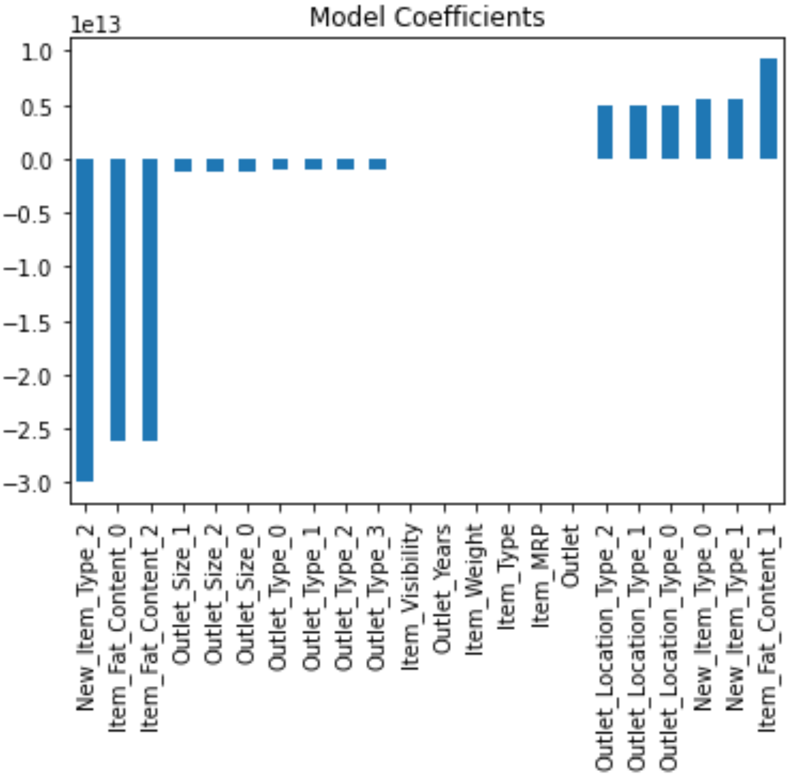
Model Training

In [77]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
def train(model, X, y):
    # train the model
    model.fit(X, y)

    # predict the training set
    pred = model.predict(X)

    # perform cross-validation
    cv_score = cross_val_score(model, X, y, scoring='neg_mean_squar
    cv_score = np.abs(np.mean(cv_score))

    print("Model Report")
    print("MSE:",mean_squared_error(y,pred))
    print("CV Score:", cv_score)
```

In [78]:
```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
model = LinearRegression(normalize=True)
train(model, X, y)
coef = pd.Series(model.coef_, X.columns).sort_values()
coef.plot(kind='bar', title="Model Coefficients")
```

```
Model Report
MSE: 0.2882074727068356
CV Score: 0.2892534032155648
```

Out[78]:
```
<AxesSubplot:title={'center':'Model Coefficients'}>
```



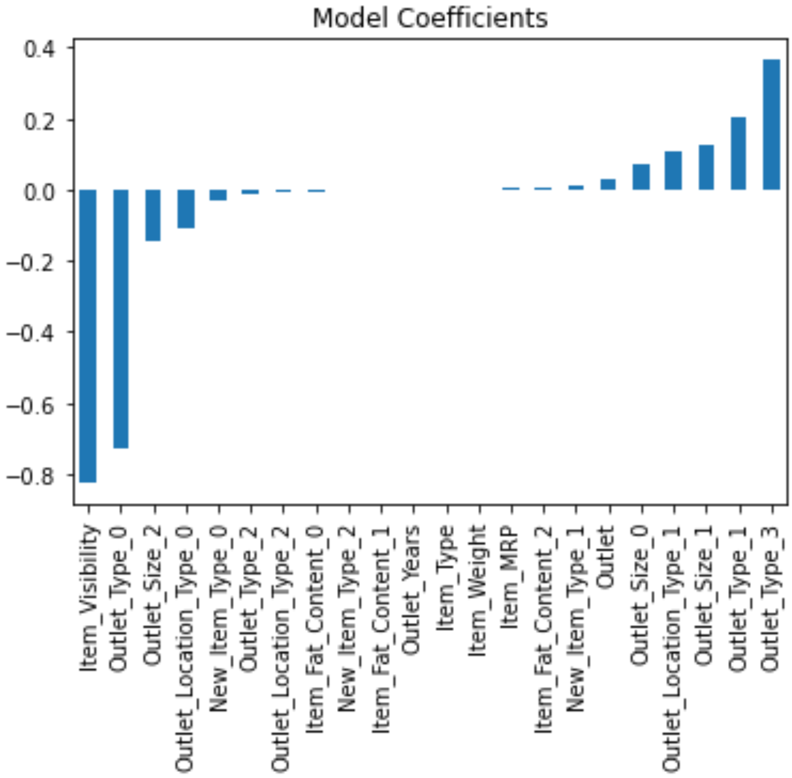
In [79]:

```
model = Ridge(normalize=True)
train(model, X, y)
coef = pd.Series(model.coef_, X.columns).sort_values()
coef.plot(kind='bar', title="Model Coefficients")
```

```
Model Report
MSE: 0.4281166030057884
CV Score: 0.42901802361866037
```

Out[79]:

```
<AxesSubplot:title={'center':'Model Coefficients'}>
```



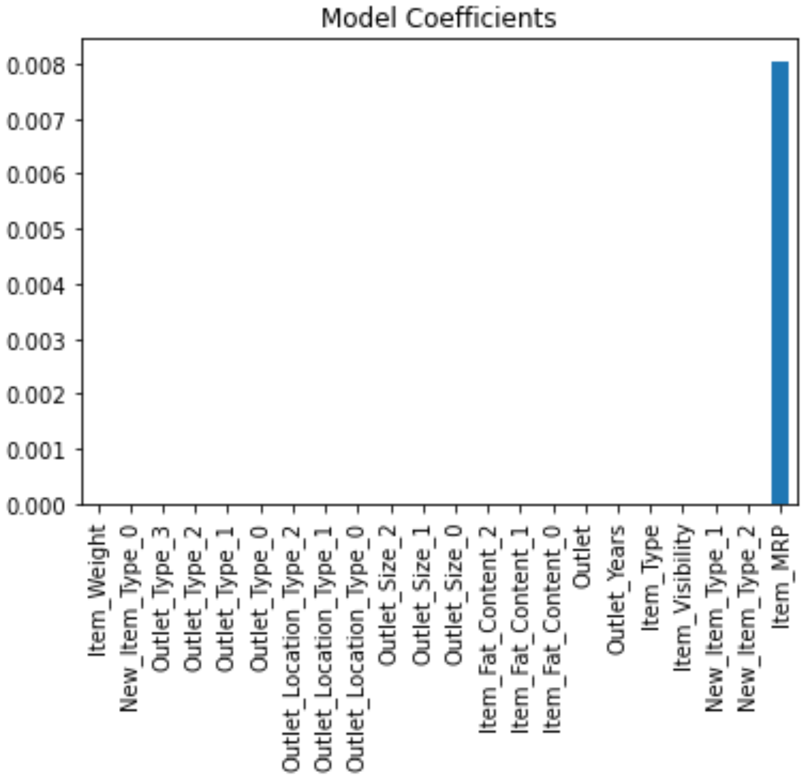
In [81]:

```
model = Lasso()
train(model, X, y)
coef = pd.Series(model.coef_, X.columns).sort_values()
coef.plot(kind='bar', title="Model Coefficients")
```

```
Model Report
MSE: 0.7628688679102086
CV Score: 0.7630789166281843
```

Out[81]:

```
<AxesSubplot:title={'center':'Model Coefficients'}>
```



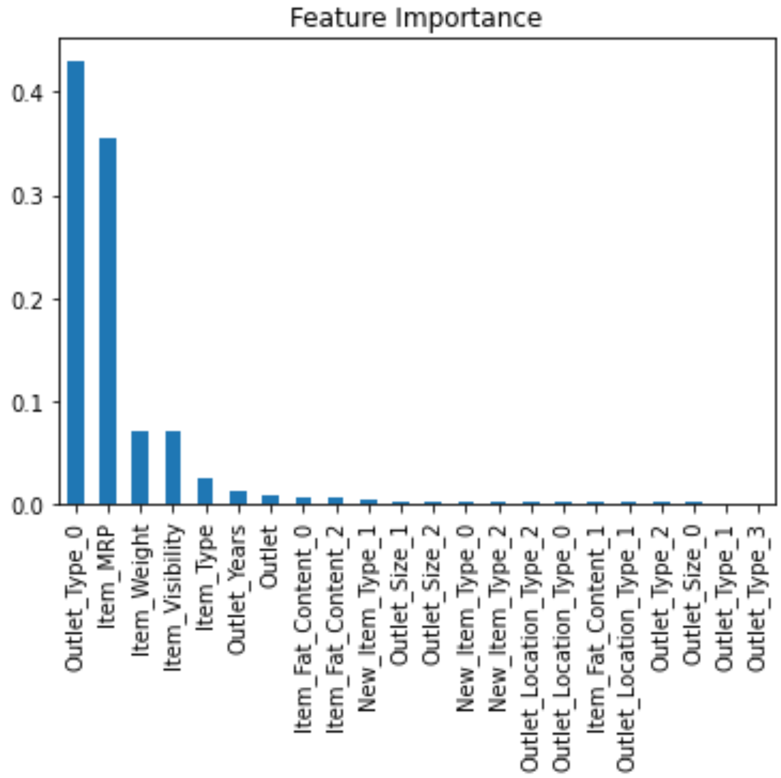
In [83]:

```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
train(model, X, y)
coef = pd.Series(model.feature_importances_, X.columns).sort_values
coef.plot(kind='bar', title="Feature Importance")
```

```
Model Report
MSE: 2.7767015319289398e-34
CV Score: 0.5684822896100131
```

Out[83]:

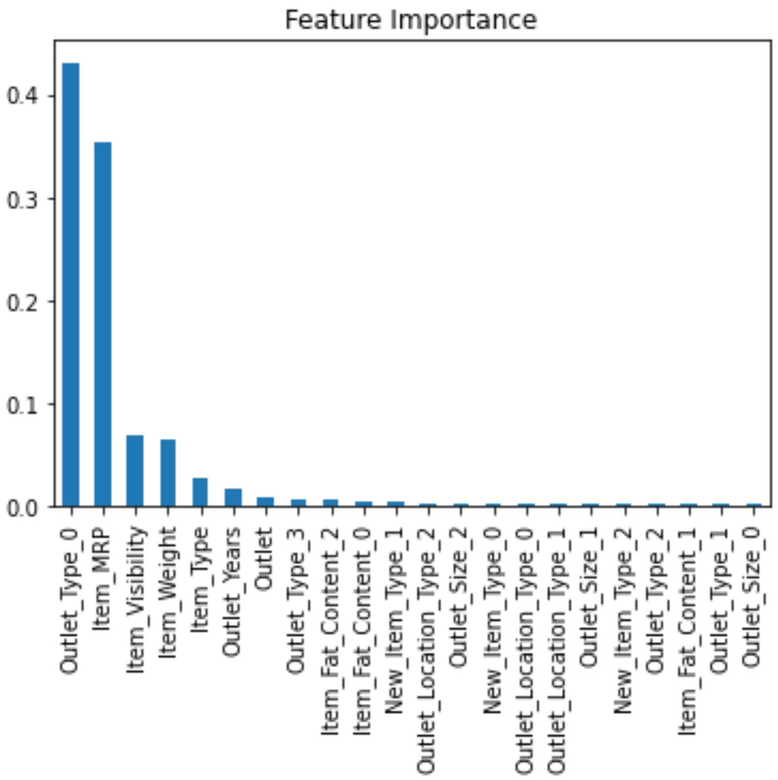
```
<AxesSubplot:title={'center':'Feature Importance'}>
```



In [84]:
```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
train(model, X, y)
coef = pd.Series(model.feature_importances_, X.columns).sort_values
coef.plot(kind='bar', title="Feature Importance")
```

```
Model Report
MSE: 0.041912334066605064
CV Score: 0.3066473050963578
```

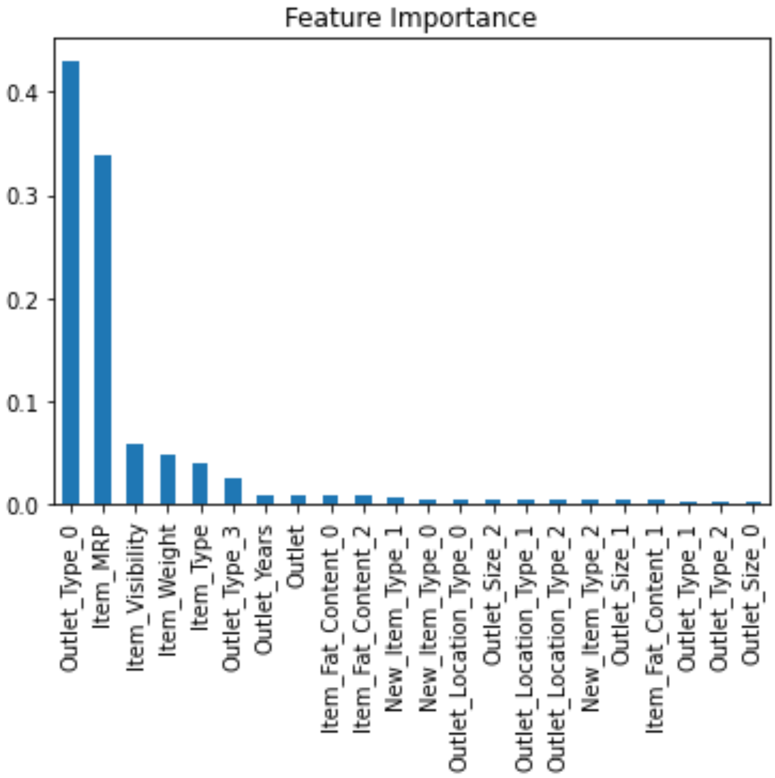
Out[84]:
```
<AxesSubplot:title={'center':'Feature Importance'}>
```



In [85]:
```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
train(model, X, y)
coef = pd.Series(model.feature_importances_, X.columns).sort_values
coef.plot(kind='bar', title="Feature Importance")
```

```
Model Report
MSE: 1.0398099340049763e-28
CV Score: 0.3295418996240995
```

Out[85]:
```
<AxesSubplot:title={'center':'Feature Importance'}>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: