# 12_Predicting_Big_Mart_Sales_A_Machine_Learning_Approach_with_X

July 9, 2024

## 0.1 Predicting Big Mart Sales: A Machine Learning Approach with Python - Vignesh Prabhu

This project focuses on predicting sales for Big Mart stores using machine learning techniques in Python. By analyzing historical sales data, various models were developed to forecast future sales, aiding inventory management and business strategy optimization.

### Import Dependencies

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from xgboost import XGBRegressor
from sklearn import metrics
```

### Data Collection And Analysing

[2]:
```
#Load data Into Dataframe
sales_data= pd.read_csv('/content/Train.csv')
```

[3]:
```
#To Preview first 5 data's in Dataset
sales_data.head()
```

[3]:

| | Item_Identifier | Item_Weight | Item_Fat_Content | Item_Visibility | Item_Type | Item_MRP | Outlet_Identifier |
|---|---|---|---|---|---|---|---|
| 0 | FDA15 | 9.30 | Low Fat | 0.016047 | Dairy | 249.8092 | OUT049 |
| 1 | DRC01 | 5.92 | Regular | 0.019278 | Soft Drinks | 48.2692 | OUT018 |
| 2 | FDN15 | 17.50 | Low Fat | 0.016760 | Meat | 141.6180 | OUT049 |
| 3 | FDX07 | 19.20 | Regular | 0.000000 | Fruits and Vegetables | 182.0950 | OUT010 |
| 4 | NCD19 | 8.93 | Low Fat | 0.000000 | | | |

```
4              Household   53.8614            OUT013

   Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
0                       1999      Medium               Tier 1
1                       2009      Medium               Tier 3
2                       1999      Medium               Tier 1
3                       1998         NaN               Tier 3
4                       1987        High               Tier 3

         Outlet_Type  Item_Outlet_Sales
0  Supermarket Type1          3735.1380
1  Supermarket Type2           443.4228
2  Supermarket Type1          2097.2700
3      Grocery Store           732.3800
4  Supermarket Type1           994.7052
```

[5]:
```
#To Preview Last 5 Data's dataset
sales_data.tail()
```

[5]:
```
     Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
8518           FDF22        6.865          Low Fat         0.056783
8519           FDS36        8.380          Regular         0.046982
8520           NCJ29       10.600          Low Fat         0.035186
8521           FDN46        7.210          Regular         0.145221
8522           DRG01       14.800          Low Fat         0.044878

              Item_Type  Item_MRP Outlet_Identifier  \
8518        Snack Foods  214.5218            OUT013
8519       Baking Goods  108.1570            OUT045
8520  Health and Hygiene   85.1224            OUT035
8521        Snack Foods  103.1332            OUT018
8522        Soft Drinks   75.4670            OUT046

      Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
8518                      1987        High               Tier 3
8519                      2002         NaN               Tier 2
8520                      2004       Small               Tier 2
8521                      2009      Medium               Tier 3
8522                      1997       Small               Tier 1

           Outlet_Type  Item_Outlet_Sales
8518  Supermarket Type1          2778.3834
8519  Supermarket Type1           549.2850
8520  Supermarket Type1          1193.1136
8521  Supermarket Type2          1845.5976
8522  Supermarket Type1           765.6700
```

[6]:
```
#To check Number Rows and columns
sales_data.shape
```

[6]:
```
(8523, 12)
```

[9]:
```
#To Complete information
sales_data.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
```

[7]:
```
#To check Null Values
sales_data.isnull().sum()
```

[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
```

[10]:
```
#Replacing Null Values with mean Values
sales_data['Item_Weight'].fillna(sales_data['Item_Weight'].mean(), inplace=True)
```

[13]:
```
#Replacing The Missing Values in "Outlet Size" With Mode
Mode_of_Outlet_Size = sales_data.pivot_table(values='Outlet_Size',␣
 ↪columns='Outlet_Type', aggfunc=(lambda x: x.mode()[0]))
```

[14]:
```
print(Mode_of_Outlet_Size)
```

```
Outlet_Type Grocery Store Supermarket Type1 Supermarket Type2  \
Outlet_Size         Small              Small            Medium

Outlet_Type Supermarket Type3
Outlet_Size            Medium
```

[15]:
```
missing_values = sales_data['Outlet_Size'].isnull()
print(missing_values)
```

```
0       False
1       False
2       False
3        True
4       False
        …
8518    False
8519     True
8520    False
8521    False
8522    False
Name: Outlet_Size, Length: 8523, dtype: bool
```

[16]:
```
sales_data.loc[missing_values, 'Outlet_Size'] = sales_data.
 ↪loc[missing_values,'Outlet_Type'].apply(lambda x: Mode_of_Outlet_Size[x])
```

[17]:
```
sales_data.isnull().sum()
```

[17]:
```
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
```

**Data Analysis**

[18]:
```
sales_data.describe()
```

[18]:

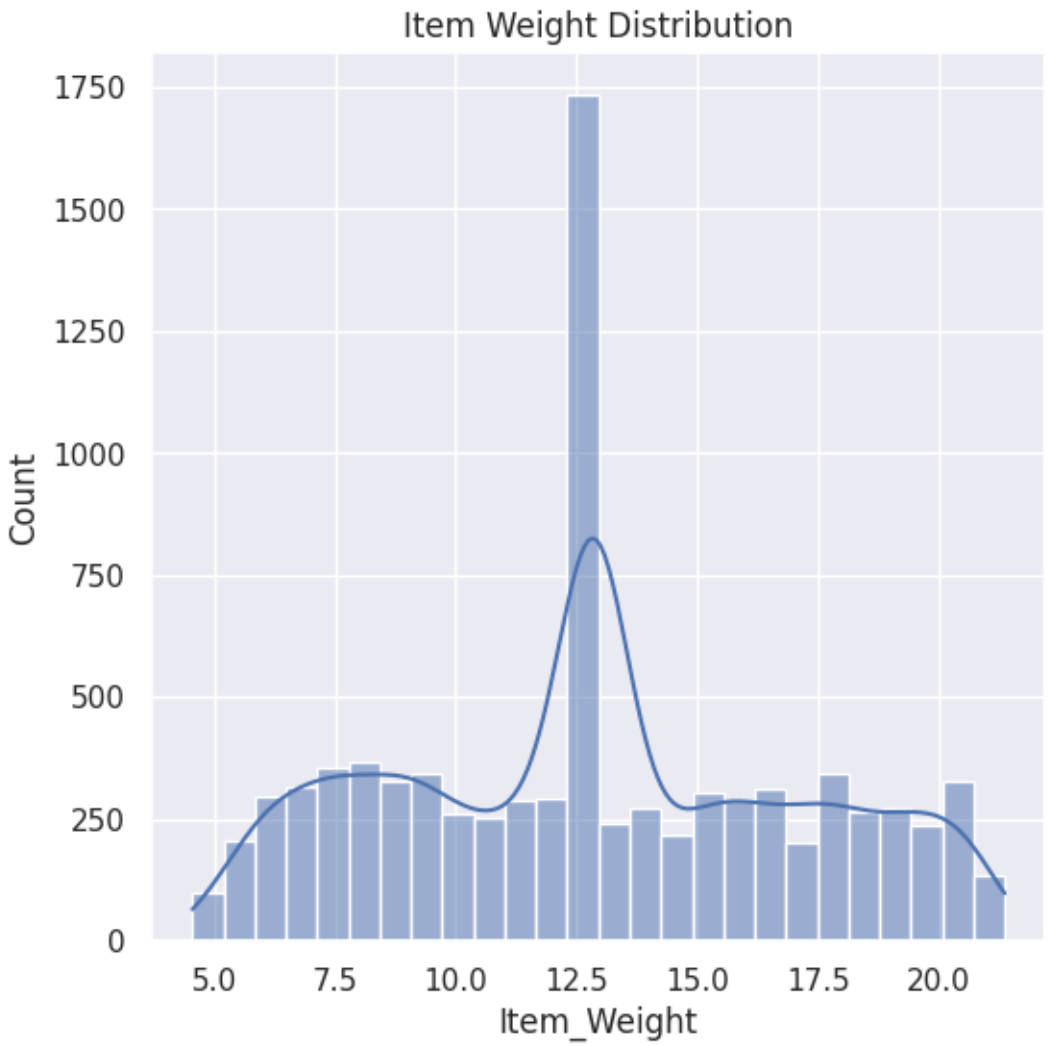
| | Item_Weight | Item_Visibility | Item_MRP | Outlet_Establishment_Year | \ |
|---|---|---|---|---|---|
| count | 8523.000000 | 8523.000000 | 8523.000000 | 8523.000000 | |
| mean | 12.857645 | 0.066132 | 140.992782 | 1997.831867 | |
| std | 4.226124 | 0.051598 | 62.275067 | 8.371760 | |
| min | 4.555000 | 0.000000 | 31.290000 | 1985.000000 | |
| 25% | 9.310000 | 0.026989 | 93.826500 | 1987.000000 | |
| 50% | 12.857645 | 0.053931 | 143.012800 | 1999.000000 | |
| 75% | 16.000000 | 0.094585 | 185.643700 | 2004.000000 | |
| max | 21.350000 | 0.328391 | 266.888400 | 2009.000000 | |

| | Item_Outlet_Sales |
|---|---|
| count | 8523.000000 |
| mean | 2181.288914 |
| std | 1706.499616 |
| min | 33.290000 |
| 25% | 834.247400 |
| 50% | 1794.331000 |
| 75% | 3101.296400 |
| max | 13086.964800 |

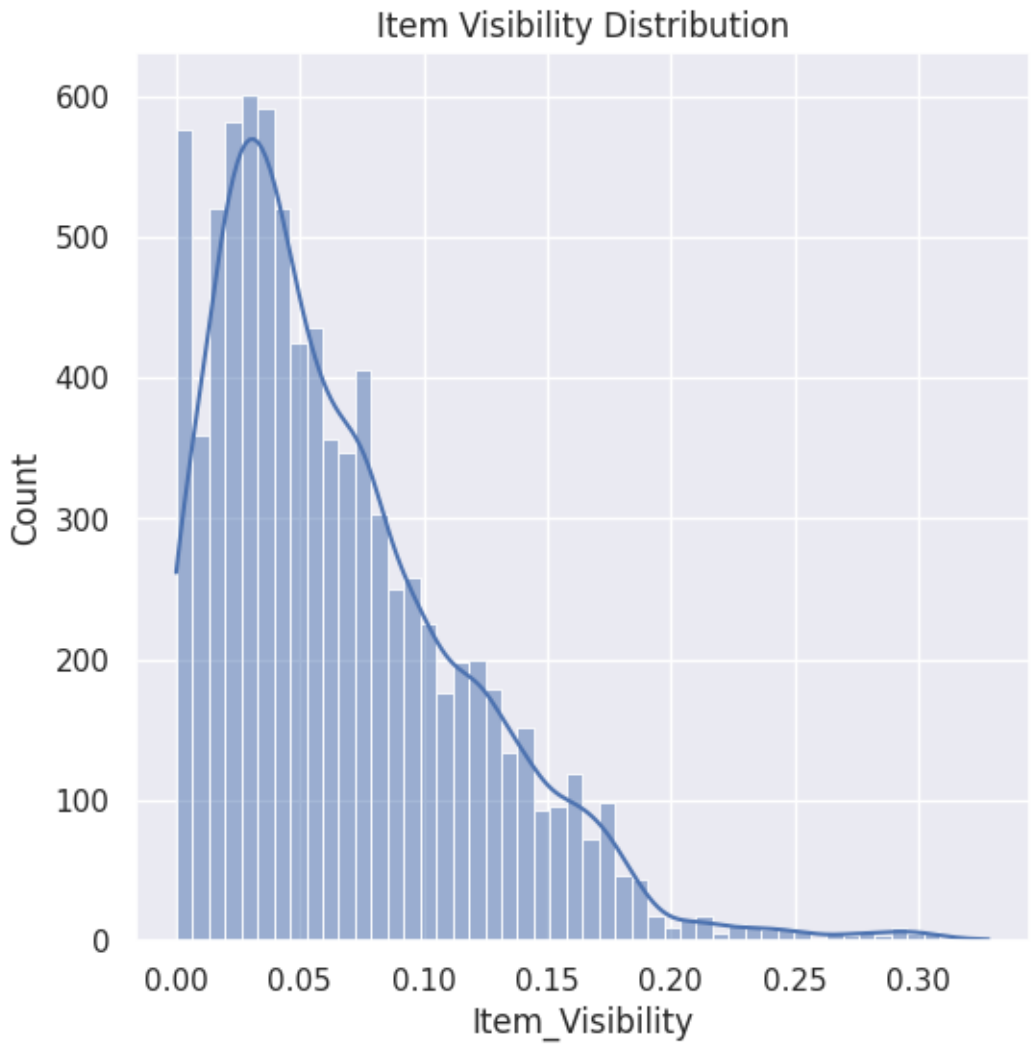
### Numerical features

[19]:
```
sns.set()
```

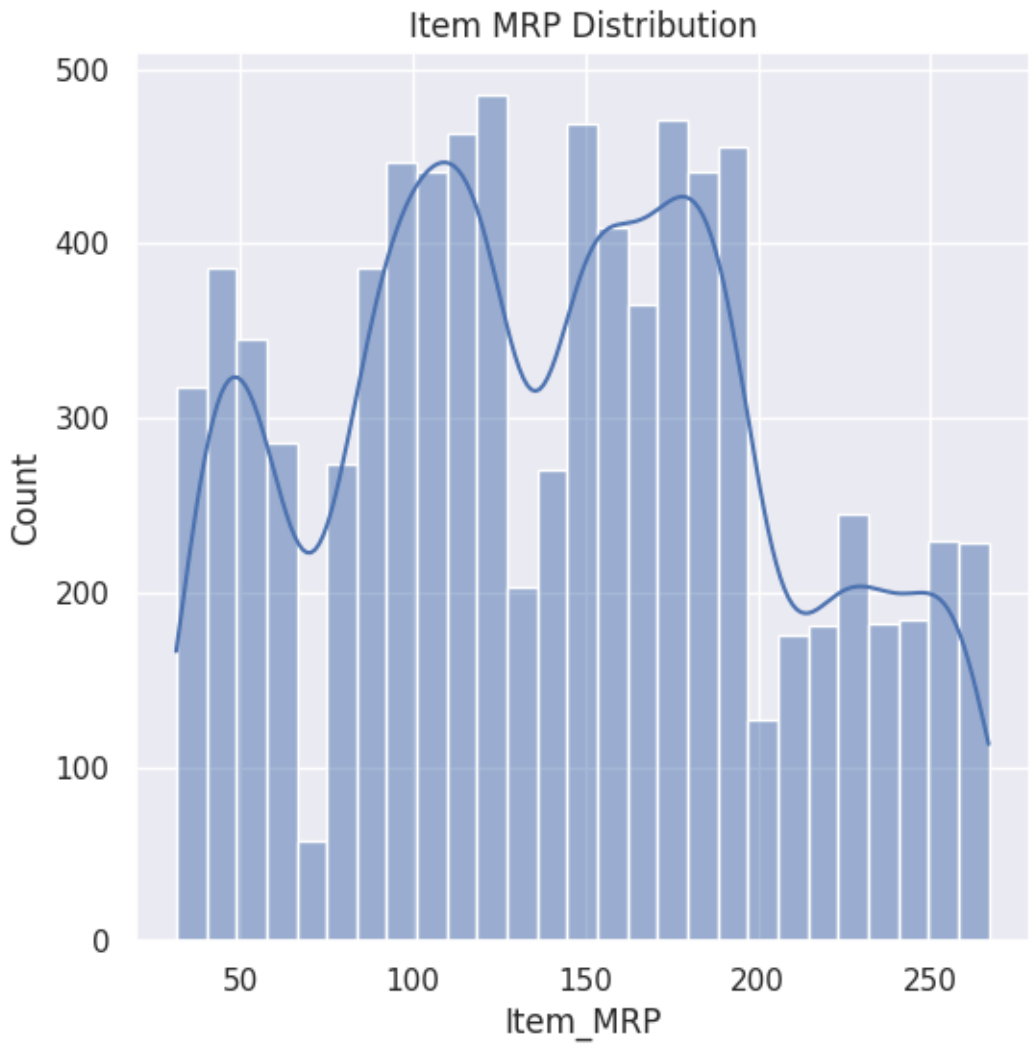
[21]:
```
#Item Weight distribution
plt.figure(figsize=(6,6))
sns.histplot(sales_data['Item_Weight'],kde=True)
plt.title('Item Weight Distribution')
plt.show()
```



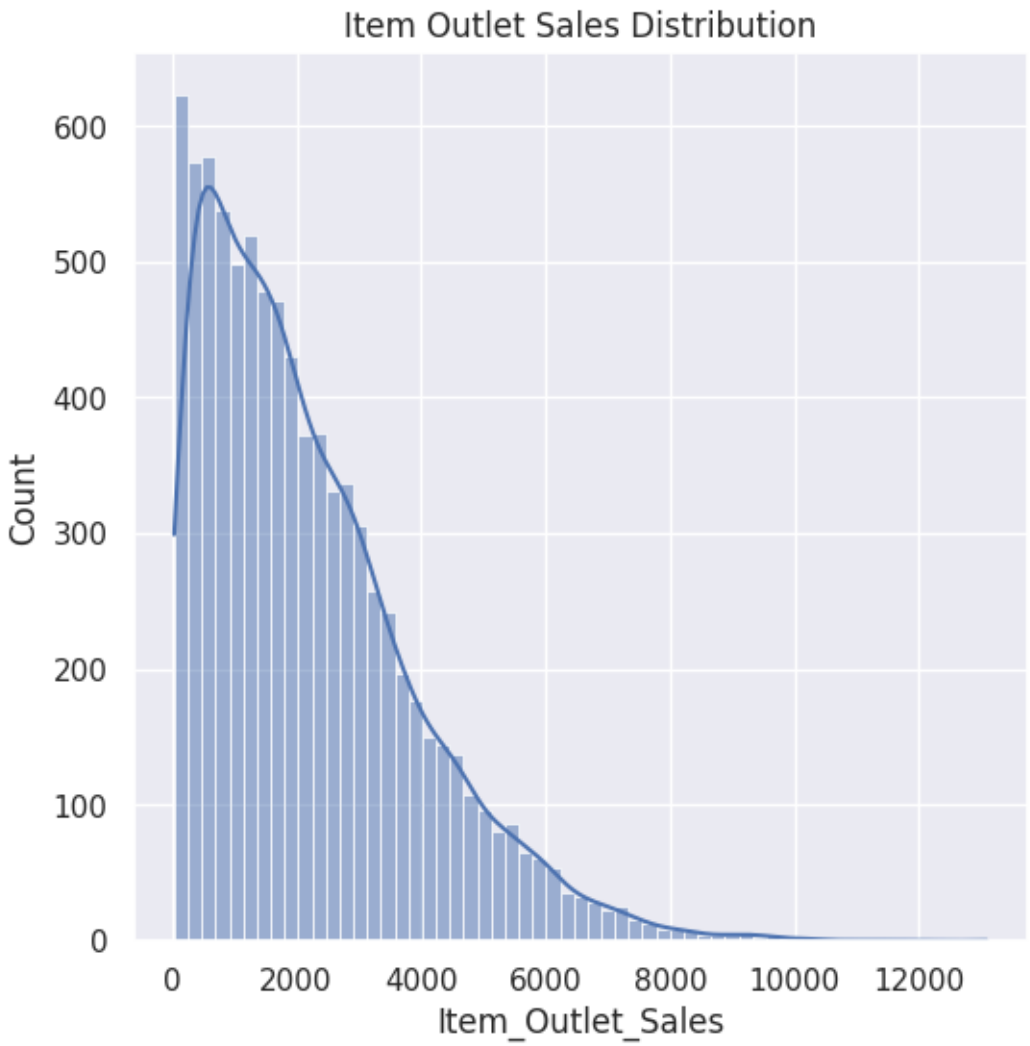

[22]:
```
#Item visiblity distribution
plt.figure(figsize=(6,6))
sns.histplot(sales_data['Item_Visibility'],kde=True)
plt.title('Item Visibility Distribution')
plt.show()
```



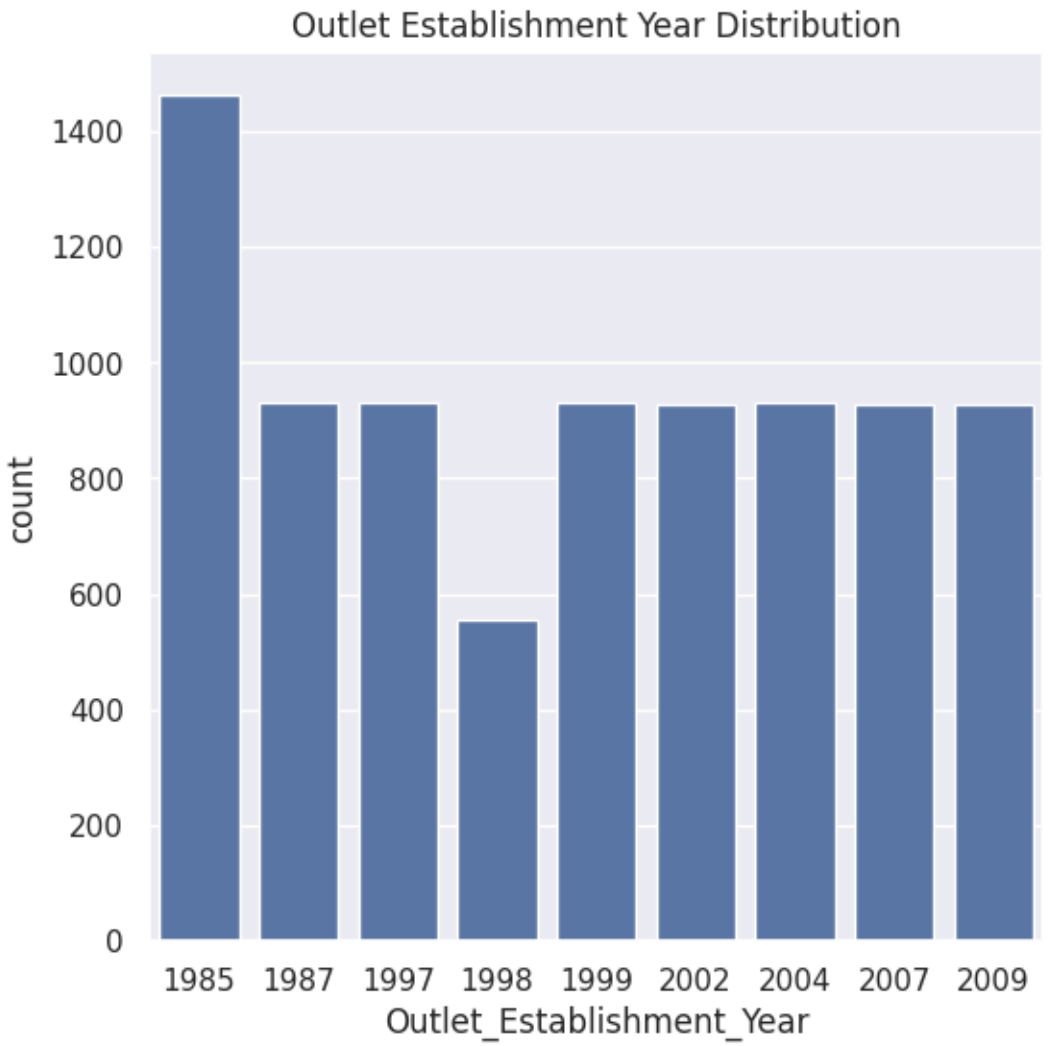

[23]:
```
#Item Mrp distribution
plt.figure(figsize=(6,6))
sns.histplot(sales_data['Item_MRP'],kde=True)
plt.title('Item MRP Distribution')
plt.show()
```




[24]:
```
#Outlet Sales Distribution
plt.figure(figsize=(6,6))
sns.histplot(sales_data['Item_Outlet_Sales'],kde=True)
plt.title('Item Outlet Sales Distribution')
plt.show()
```

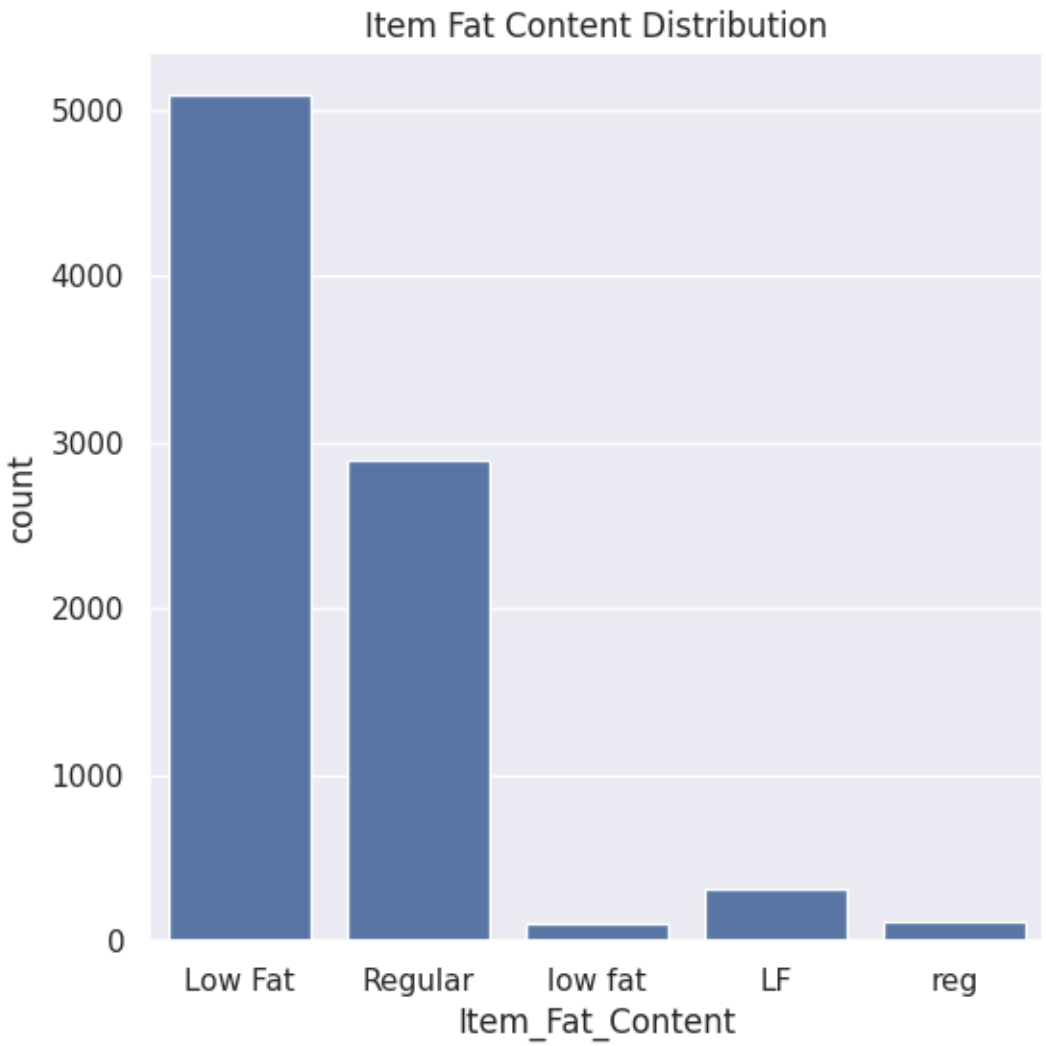



[26]:
```
#outlet establishment
plt.figure(figsize=(6,6))
sns.countplot(x='Outlet_Establishment_Year', data=sales_data)
plt.title('Outlet Establishment Year Distribution')
plt.show()
```

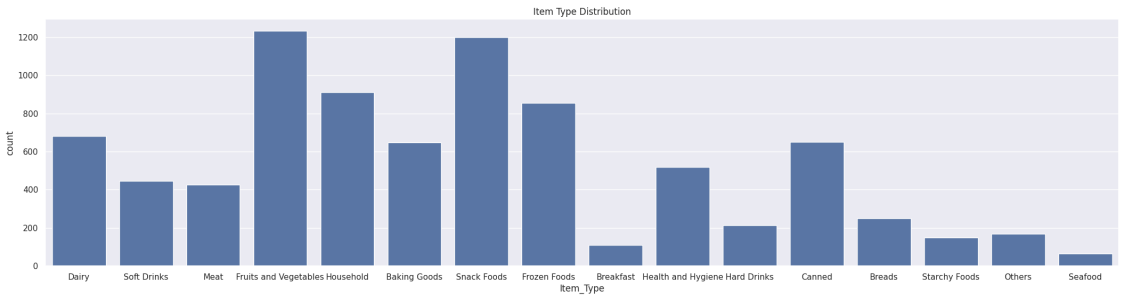



### Categorical features

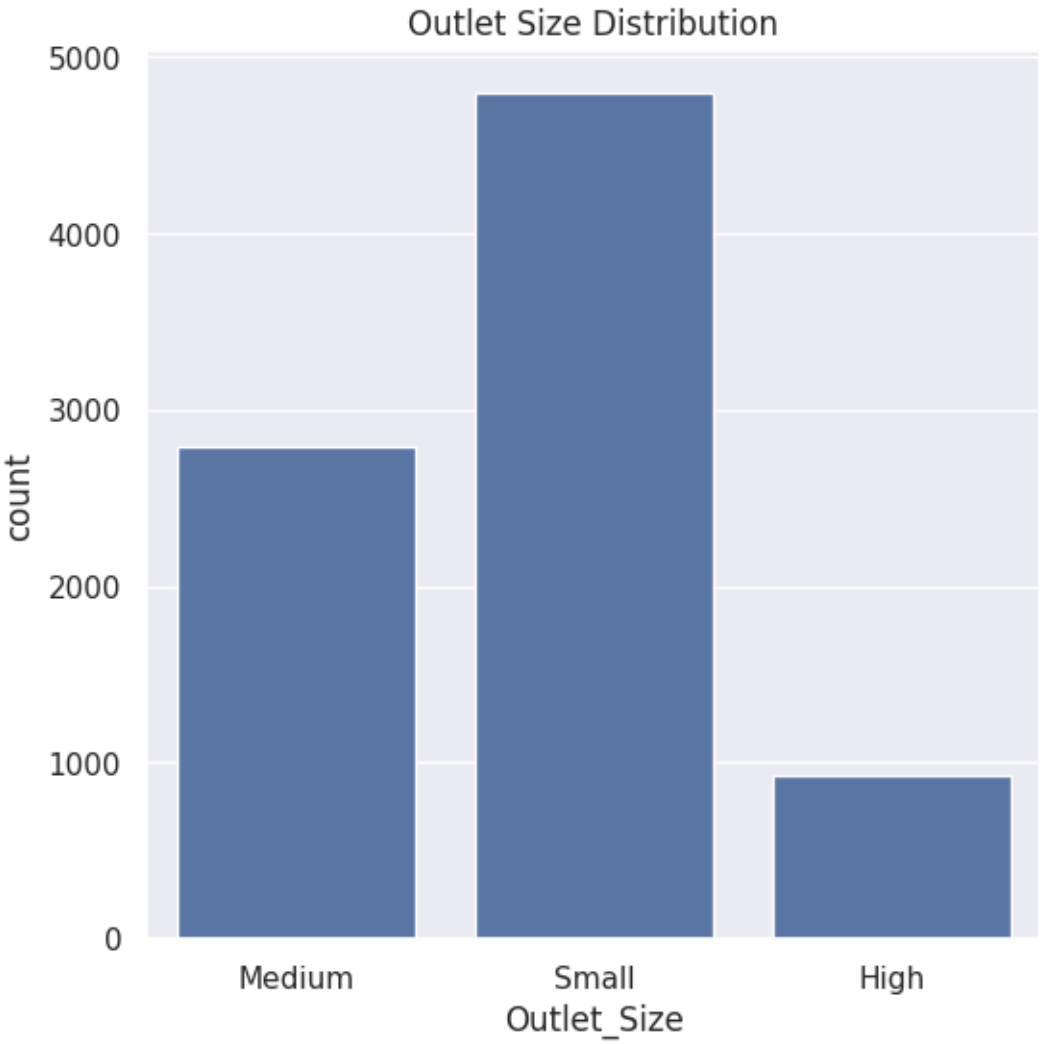
[27]:
```
#item_Fat columns
plt.figure(figsize=(6,6))
sns.countplot(x='Item_Fat_Content', data=sales_data )
plt.title('Item Fat Content Distribution')
plt.show()
```




[28]:
```
#item type Columns
plt.figure(figsize=(26,6))
sns.countplot(x='Item_Type', data=sales_data)
plt.title('Item Type Distribution')
plt.show()
```



[29]:
```
#outlet size
plt.figure(figsize=(6,6))
sns.countplot(x='Outlet_Size', data=sales_data)
plt.title('Outlet Size Distribution')
plt.show()
```




### Data PreProcessing

[30]:
```
sales_data.head()
```

[30]:

| | Item_Identifier | Item_Weight | Item_Fat_Content | Item_Visibility | \ |
|---|---|---|---|---|---|
| 0 | FDA15 | 9.30 | Low Fat | 0.016047 | |

```
1           DRC01         5.92          Regular         0.019278
2           FDN15        17.50          Low Fat         0.016760
3           FDX07        19.20          Regular         0.000000
4           NCD19         8.93          Low Fat         0.000000

              Item_Type  Item_MRP Outlet_Identifier  \
0                 Dairy  249.8092            OUT049
1           Soft Drinks   48.2692            OUT018
2                  Meat  141.6180            OUT049
3  Fruits and Vegetables  182.0950            OUT010
4             Household   53.8614            OUT013

   Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
0                       1999      Medium               Tier 1
1                       2009      Medium               Tier 3
2                       1999      Medium               Tier 1
3                       1998       Small               Tier 3
4                       1987        High               Tier 3

        Outlet_Type  Item_Outlet_Sales
0  Supermarket Type1          3735.1380
1  Supermarket Type2           443.4228
2  Supermarket Type1          2097.2700
3      Grocery Store           732.3800
4  Supermarket Type1           994.7052
```

[31]:
```
sales_data['Item_Fat_Content'].value_counts()
```

[31]:
```
Item_Fat_Content
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: count, dtype: int64
```

[32]:
```
sales_data.replace({'Item_Fat_Content': {'low fat':'Low Fat','LF':'Low Fat',␣
 ↪'reg':'Regular'}}, inplace=True)
```

[33]:
```
sales_data['Item_Fat_Content'].value_counts()
```

[33]:
```
Item_Fat_Content
Low Fat    5517
Regular    3006
Name: count, dtype: int64
```

### Label Encoding

[34]:
```
encoder = LabelEncoder()
```

[35]:
```
sales_data['Item_Identifier'] = encoder.
 ↪fit_transform(sales_data['Item_Identifier'])

sales_data['Item_Fat_Content'] = encoder.
 ↪fit_transform(sales_data['Item_Fat_Content'])

sales_data['Item_Type'] = encoder.fit_transform(sales_data['Item_Type'])

sales_data['Outlet_Identifier'] = encoder.
 ↪fit_transform(sales_data['Outlet_Identifier'])

sales_data['Outlet_Size'] = encoder.fit_transform(sales_data['Outlet_Size'])

sales_data['Outlet_Location_Type'] = encoder.
 ↪fit_transform(sales_data['Outlet_Location_Type'])

sales_data['Outlet_Type'] = encoder.fit_transform(sales_data['Outlet_Type'])
```

[36]:
```
sales_data.head()
```

[36]:

| | Item_Identifier | Item_Weight | Item_Fat_Content | Item_Visibility | Item_Type | \ |
|---|---|---|---|---|---|---|
| 0 | 156 | 9.30 | 0 | 0.016047 | 4 | |
| 1 | 8 | 5.92 | 1 | 0.019278 | 14 | |
| 2 | 662 | 17.50 | 0 | 0.016760 | 10 | |
| 3 | 1121 | 19.20 | 1 | 0.000000 | 6 | |
| 4 | 1297 | 8.93 | 0 | 0.000000 | 9 | |

| | Item_MRP | Outlet_Identifier | Outlet_Establishment_Year | Outlet_Size | \ |
|---|---|---|---|---|---|
| 0 | 249.8092 | 9 | 1999 | 1 | |
| 1 | 48.2692 | 3 | 2009 | 1 | |
| 2 | 141.6180 | 9 | 1999 | 1 | |
| 3 | 182.0950 | 0 | 1998 | 2 | |
| 4 | 53.8614 | 1 | 1987 | 0 | |

| | Outlet_Location_Type | Outlet_Type | Item_Outlet_Sales |
|---|---|---|---|
| 0 | 0 | 1 | 3735.1380 |
| 1 | 2 | 2 | 443.4228 |
| 2 | 0 | 1 | 2097.2700 |
| 3 | 2 | 0 | 732.3800 |
| 4 | 2 | 1 | 994.7052 |

### Spliting Features and Target

[37]:
```
X= sales_data.drop(columns='Item_Outlet_Sales', axis=1)
Y= sales_data['Item_Outlet_Sales']
```

[38]:
```
print(X)
```

```
      Item_Identifier  Item_Weight  Item_Fat_Content  Item_Visibility  \
0                 156        9.300                 0         0.016047
1                   8        5.920                 1         0.019278
2                 662       17.500                 0         0.016760
3                1121       19.200                 1         0.000000
4                1297        8.930                 0         0.000000
…                 …           …                   …              …
8518              370        6.865                 0         0.056783
8519              897        8.380                 1         0.046982
8520             1357       10.600                 0         0.035186
8521              681        7.210                 1         0.145221
8522               50       14.800                 0         0.044878

      Item_Type  Item_MRP  Outlet_Identifier  Outlet_Establishment_Year  \
0             4  249.8092                  9                       1999
1            14   48.2692                  3                       2009
2            10  141.6180                  9                       1999
3             6  182.0950                  0                       1998
4             9   53.8614                  1                       1987
…         …       …                   …                         …
8518         13  214.5218                  1                       1987
8519          0  108.1570                  7                       2002
8520          8   85.1224                  6                       2004
8521         13  103.1332                  3                       2009
8522         14   75.4670                  8                       1997

      Outlet_Size  Outlet_Location_Type  Outlet_Type
0               1                     0            1
1               1                     2            2
2               1                     0            1
3               2                     2            0
4               0                     2            1
…             …                     …            …
8518            0                     2            1
8519            2                     1            1
8520            2                     1            1
8521            1                     2            2
8522            2                     0            1

[8523 rows x 11 columns]
```

[39]:
```
print(Y)
```

```
0       3735.1380
1        443.4228
2       2097.2700
```

```
3        732.3800
4        994.7052
           …
8518    2778.3834
8519     549.2850
8520    1193.1136
8521    1845.5976
8522     765.6700
Name: Item_Outlet_Sales, Length: 8523, dtype: float64
```

### Spliting Train and Test Data

[40]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
↪random_state=2)
```

[41]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(8523, 11) (6818, 11) (1705, 11)
```

### Model Training

[42]:
```
Regressor = XGBRegressor()
```

[43]:
```
Regressor.fit(X_train, Y_train)
```

[43]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, …)
```

### Evaluation

[45]:
```
#Prediction on Training data
training_data_prediction = Regressor.predict(X_train)
```

[46]:
```
# R Squarred Values
r2_train = metrics.r2_score(Y_train, training_data_prediction)
print('R Squared value = ', r2_train)
```

```
R Squared value =  0.8762174618111388
```

[47]:
```
#Prediction on Test Data
test_data_prediction = Regressor.predict(X_test)
r2_test = metrics.r2_score(Y_test, test_data_prediction)
print('R Squared value = ', r2_test)
```

```
R Squared value =  0.5017253991620692
```