Importing the Dependencies

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

Data Collection and Processing

```
# loading the data from csv file to Pandas DataFrame
big_mart_data = pd.read_csv('Train.csv')
```

```
# first 5 rows of the dataframe
big_mart_data.head()
```

```
  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         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
```

```
# number of data points & number of features
big_mart_data.shape
```

```
(8523, 12)
```

```
# getting some information about thye dataset
big_mart_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
```

Categorical Features:

- Item_Identifier
- Item_Fat_Content
- Item_Type
- Outlet_Identifier
- Outlet_Size
- Outlet_Location_Type
- Outlet_Type

```
# checking for missing values
big_mart_data.isnull().sum()
```

```
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
```

Handling Missing Values

Mean --> average

Mode --> more repeated value

```
# mean value of "Item_Weight" column
big_mart_data['Item_Weight'].mean()
```

```
12.857645184135976
```

```
# filling the missing values in "Item_weight column" with "Mean" value
big_mart_data['Item_Weight'].fillna(big_mart_data['Item_Weight'].mean(
), inplace=True)
```

```
# mode of "Outlet_Size" column
big_mart_data['Outlet_Size'].mode()
```

```
0    Medium
Name: Outlet_Size, dtype: object
```

```
# filling the missing values in "Outlet_Size" column with Mode
mode_of_Outlet_size = big_mart_data.pivot_table(values='Outlet_Size',
columns='Outlet_Type', aggfunc=(lambda x: x.mode()[0]))

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
```

```
miss_values = big_mart_data['Outlet_Size'].isnull()

print(miss_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
```

```
big_mart_data.loc[miss_values, 'Outlet_Size'] =
big_mart_data.loc[miss_values,'Outlet_Type'].apply(lambda x:
mode_of_Outlet_size[x])
```

```
# checking for missing values
big_mart_data.isnull().sum()
```

```
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

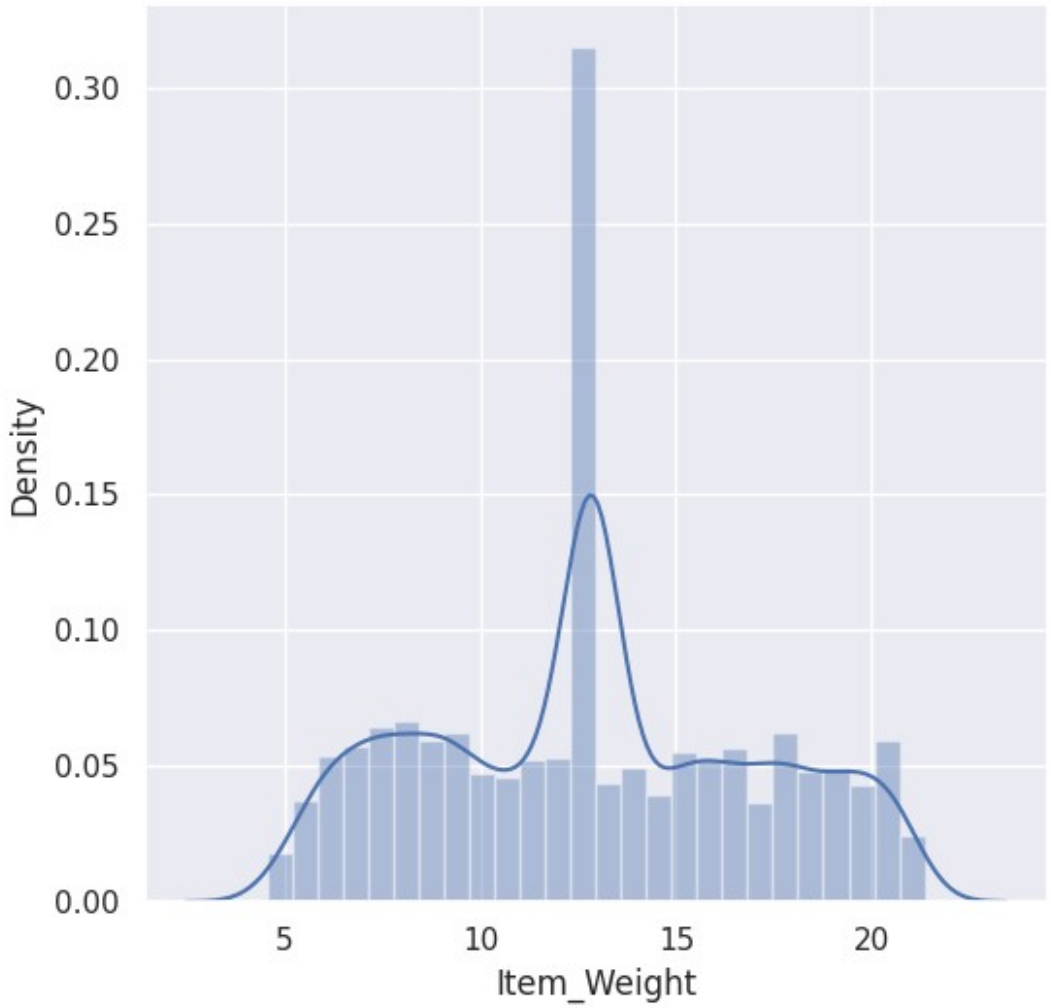
```
big_mart_data.describe()
```

```
       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

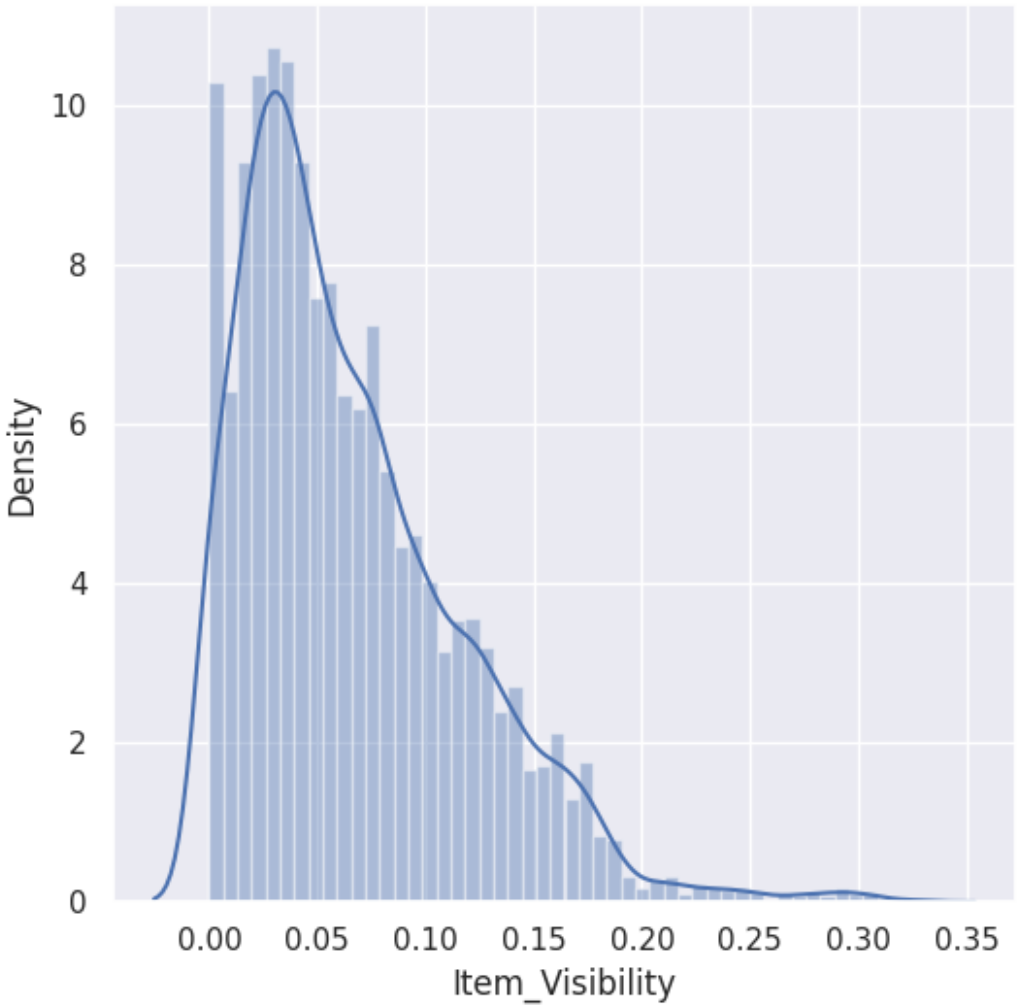
```
sns.set()

# Item_Weight distribution
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_Weight'])
plt.show()
```

```
<ipython-input-20-21151ade0b57>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(big_mart_data['Item_Weight'])
```



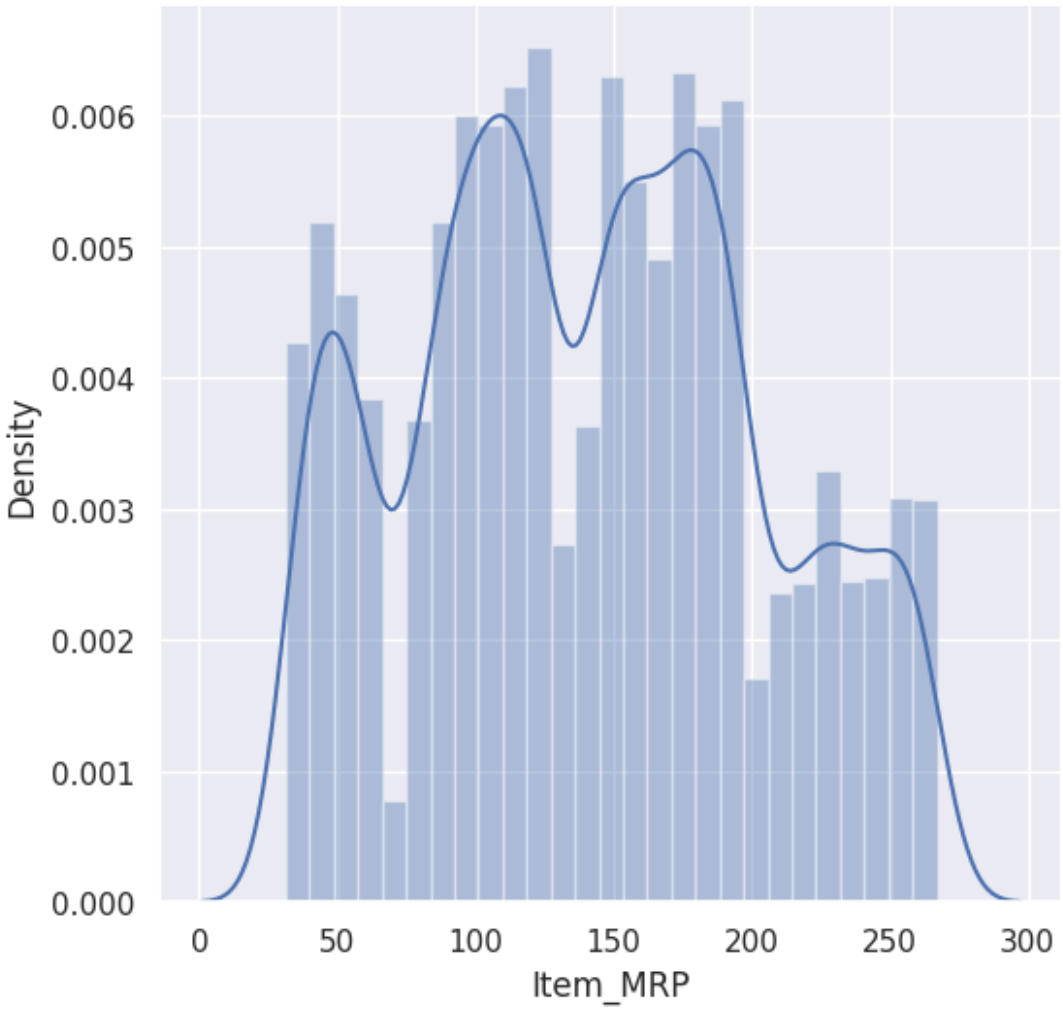

```
# Item Visibility distribution
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_Visibility'])
plt.show()
```

```
<ipython-input-21-386044597ca3>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(big_mart_data['Item_Visibility'])
```



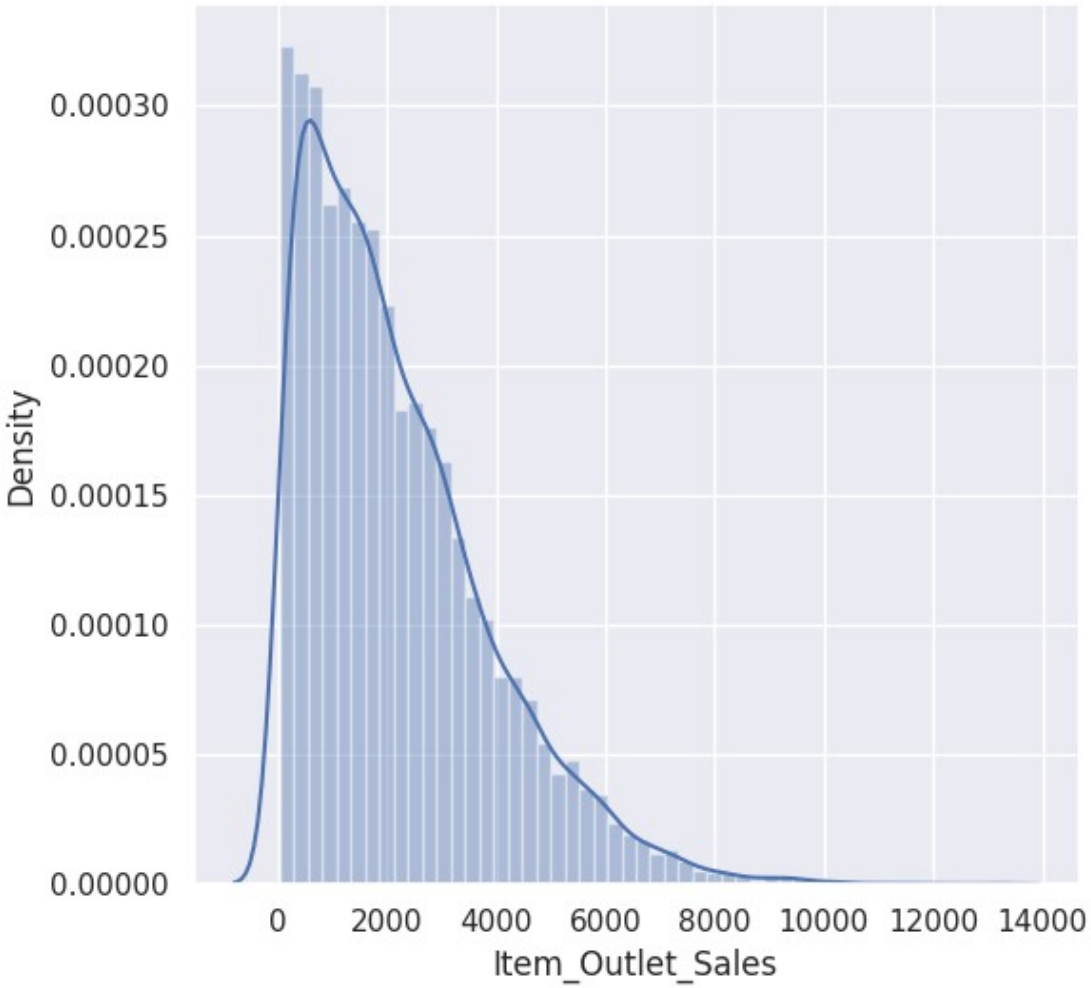

```
# Item MRP distribution
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_MRP'])
plt.show()
```

```
<ipython-input-22-0b69bf4930c1>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(big_mart_data['Item_MRP'])
```

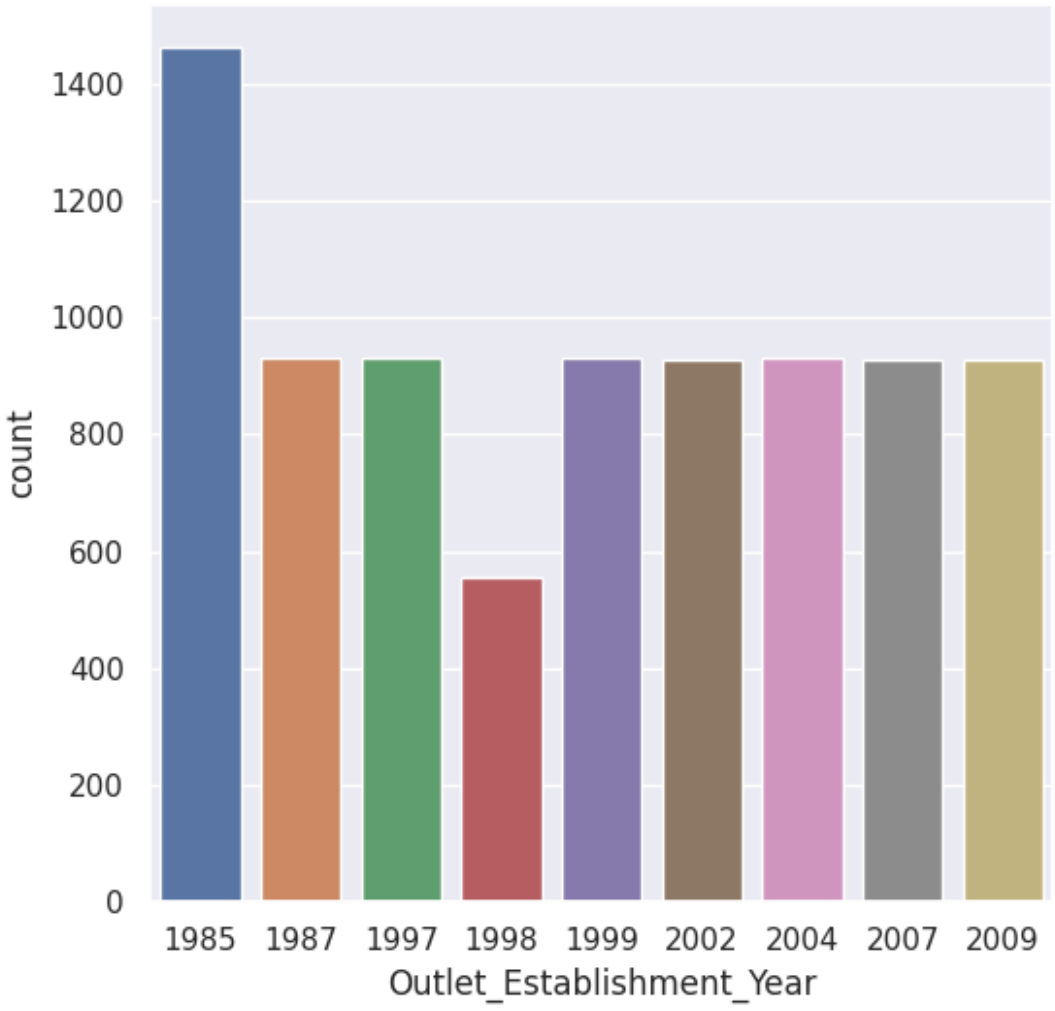



```
# Item_Outlet_Sales distribution
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_Outlet_Sales'])
plt.show()
```

```
<ipython-input-23-dedd64409ff7>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(big_mart_data['Item_Outlet_Sales'])
```

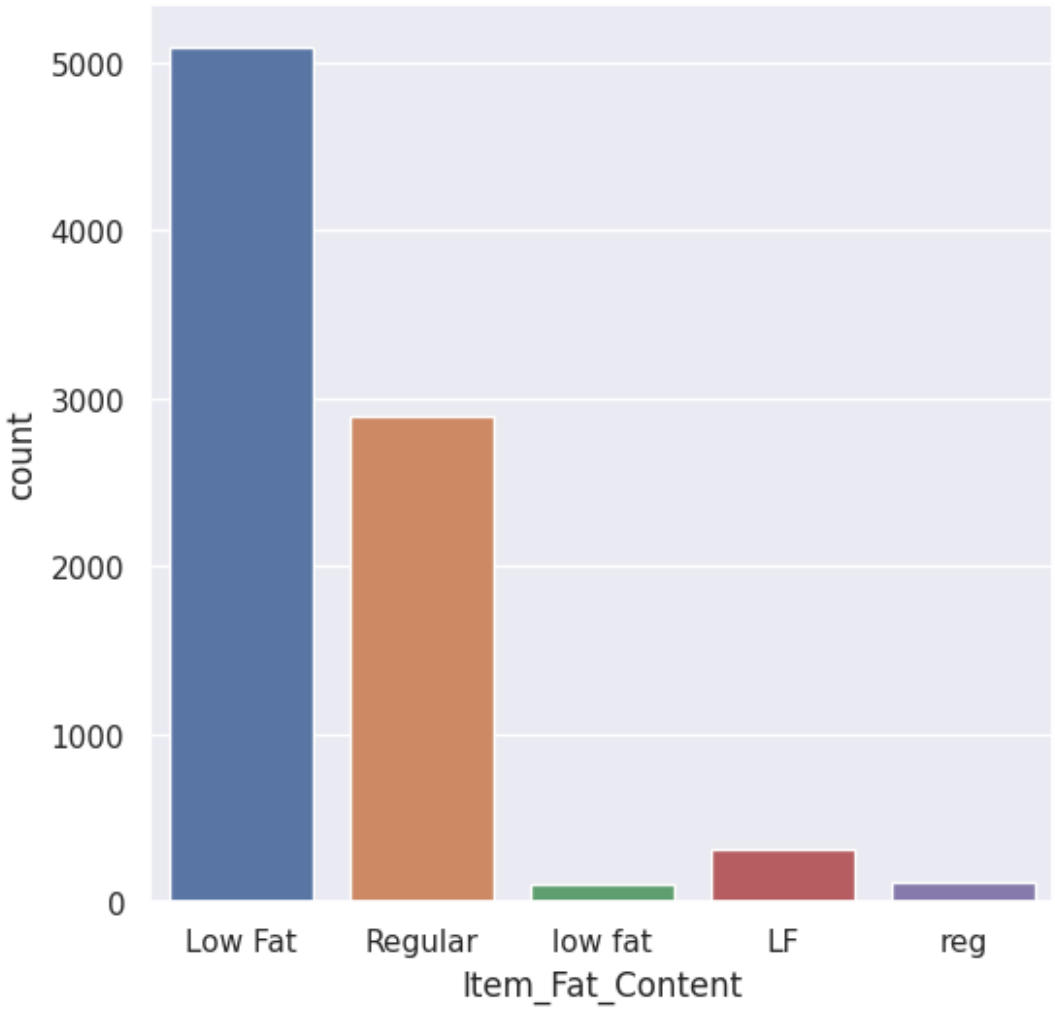



```
# Outlet_Establishment_Year column
plt.figure(figsize=(6,6))
sns.countplot(x='Outlet_Establishment_Year', data=big_mart_data)
plt.show()
```

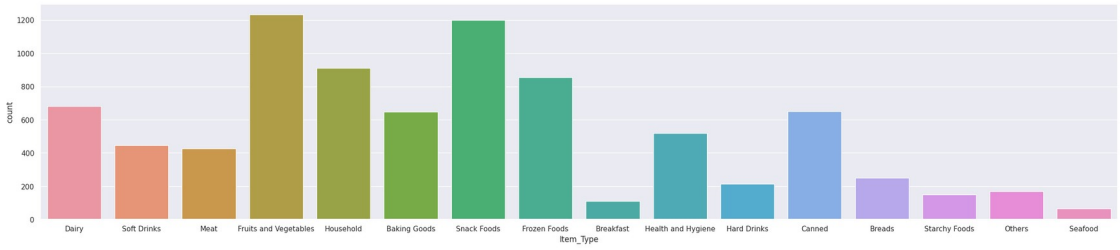


Categorical Features

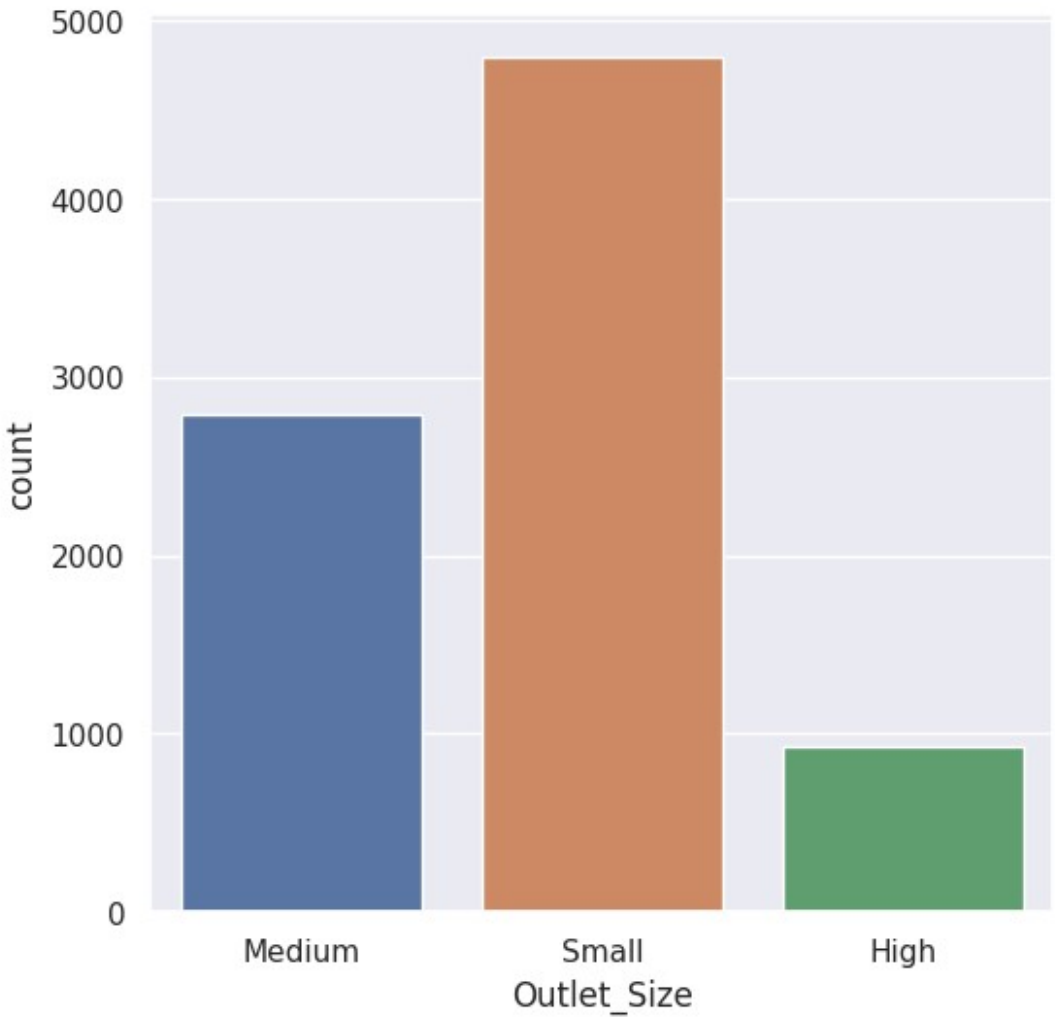
```
# Item_Fat_Content column
plt.figure(figsize=(6,6))
sns.countplot(x='Item_Fat_Content', data=big_mart_data)
plt.show()
```




```
# Item_Type column
plt.figure(figsize=(30,6))
sns.countplot(x='Item_Type', data=big_mart_data)
plt.show()
```




```
# Outlet_Size column
plt.figure(figsize=(6,6))
sns.countplot(x='Outlet_Size', data=big_mart_data)
plt.show()
```



Data Pre-Processing

```
big_mart_data.head()
```

```
  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
```

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

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

```
big_mart_data.replace({'Item_Fat_Content': {'low fat':'Low
Fat','LF':'Low Fat', 'reg':'Regular'}}, inplace=True)
```

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

```
Low Fat    5517
Regular    3006
Name: Item_Fat_Content, dtype: int64
```

Label Encoding

```
encoder = LabelEncoder()
```

```
big_mart_data['Item_Identifier'] =
encoder.fit_transform(big_mart_data['Item_Identifier'])

big_mart_data['Item_Fat_Content'] =
encoder.fit_transform(big_mart_data['Item_Fat_Content'])

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

big_mart_data['Outlet_Identifier'] =
encoder.fit_transform(big_mart_data['Outlet_Identifier'])

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

big_mart_data['Outlet_Location_Type'] =
```

```
encoder.fit_transform(big_mart_data['Outlet_Location_Type'])

big_mart_data['Outlet_Type'] =
encoder.fit_transform(big_mart_data['Outlet_Type'])

big_mart_data.head()
```

```
   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
```

Splitting features and Target

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

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]
```

```
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
```

Splitting the data into Training data & Testing Data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

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

Machine Learning Model Training

XGBoost Regressor

```
regressor = XGBRegressor()
```

```
regressor.fit(X_train, Y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
             gamma=None, gpu_id=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,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

Evaluation

```
# prediction on training data
training_data_prediction = regressor.predict(X_train)

# R squared Value
r2_train = metrics.r2_score(Y_train, training_data_prediction)

print('R Squared value = ', r2_train)
```

```
R Squared value =  0.8639680373364909
```

```
# prediction on test data
test_data_prediction = regressor.predict(X_test)

# R squared Value
r2_test = metrics.r2_score(Y_test, test_data_prediction)

print('R Squared value = ', r2_test)
```

```
R Squared value =  0.5233136709735687
```

```
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

print(mean_absolute_error(Y_test,test_data_prediction))
print(np.sqrt(mean_squared_error(Y_test,test_data_prediction)))
```

```
847.7947437393422
1213.0813291621828
```

Linear Regression

```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()

lr.fit(X_train,Y_train)
```

```
LinearRegression()
```

```
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

Y_pred_lr=lr.predict(X_test)
```

```
print(r2_score(Y_test,Y_pred_lr))
print(mean_absolute_error(Y_test,Y_pred_lr))
print(np.sqrt(mean_squared_error(Y_test,Y_pred_lr)))
```

```
0.48912987193626367
944.869172118322
1255.8241328615923
```

RandomForestRegressor

```
from sklearn.ensemble import RandomForestRegressor
rf= RandomForestRegressor(n_estimators=1000)

rf.fit(X_train,Y_train)
```

```
RandomForestRegressor(n_estimators=1000)
```

```
Y_pred_rf= rf.predict(X_test)

print(r2_score(Y_test,Y_pred_rf))
print(mean_absolute_error(Y_test,Y_pred_rf))
print(np.sqrt(mean_squared_error(Y_test,Y_pred_rf)))
```

```
0.5535269518069379
823.2235119768916
1174.008371341967
```