```
import pandas as pd
import numpy as np
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
%matplotlib inline
```

```
df1= pd.read_csv("/content/Train1.csv")
print(df1)
```

```
     Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
0              FDA15        9.300          Low Fat         0.016047   
1              DRC01        5.920          Regular         0.019278   
2              FDN15       17.500          Low Fat         0.016760   
3              FDX07       19.200          Regular         0.000000   
4              NCD19        8.930          Low Fat         0.000000   
...              ...          ...              ...              ...   
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  \
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   
...                     ...       ...               ...   
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  \
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   
...                         ...         ...                  ...   
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
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
...                 ...                ...
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

[8523 rows x 12 columns]
```

```
df1.sample(5)# print random sample 5 rows
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_
8138	FDV01	19.20	Regular	0.085123	Canned	155.4
3057	FDX25	16.70	Low Fat	0.102036	Canned	180.9
999	NCP06	NaN	Low Fat	0.039056	Household	152.3
3210	FDH34	8.63	Low Fat	0.031144	Snack Foods	183.9
2281	FDH34	8.63	Low Fat	0.031271	Snack Foods	186.0

```
df1.shape # no of rows and columns
```

```
(8523, 12)
```

```
df1.ndim # there are 2 dimensions attributes and instances.
```

```
2
```

```
df1.size#rows product columns
```

```
102276
```

```
df1.columns#to know columns
```

```
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'],
      dtype='object')
```

```
df1.info()# data description about columns
```

```
<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
```

```
#Here we are preprocessing data for 1000 rows
df1=df1.head(1000)# print 1000 rows
print(df1)
```

```
    Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
0             FDA15        9.300          Low Fat         0.016047
1             DRC01        5.920          Regular         0.019278
2             FDN15       17.500          Low Fat         0.016760
3             FDX07       19.200          Regular         0.000000
4             NCD19        8.930          Low Fat         0.000000
..              ...          ...              ...              ...
995           FDO34       17.700          Low Fat         0.050112
996           NCL30       18.100          Low Fat         0.048931
997           FDK28        5.695          Low Fat         0.065961
998           DRJ39       20.250          Low Fat         0.036319
999           NCP06          NaN          Low Fat         0.039056

                 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
..                     ...       ...               ...
995            Snack Foods  165.9816            OUT010
```

```
996              Household  127.3336           OUT035
997           Frozen Foods  259.2646           OUT017
998                  Dairy  219.3482           OUT035
999              Household  152.3366           OUT027

     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
..                         ...         ...                  ...
995                       1998         NaN               Tier 3
996                       2004       Small               Tier 2
997                       2007         NaN               Tier 2
998                       2004       Small               Tier 2
999                       1985      Medium               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
..                 ...                ...
995      Grocery Store           167.7816
996  Supermarket Type1          1150.5024
997  Supermarket Type1          9275.9256
998  Supermarket Type1          5038.1086
999  Supermarket Type3          2115.9124

[1000 rows x 12 columns]
```

```
#here we are finding the item weight mean
df1['Item_Weight'].mean()
```

```
13.032137592137593
```

```
df1['Item_Weight'].fillna(df1['Item_Weight'].mean(),inplace=True)
```

```
df1.isnull().sum()# here we are finding the total null values
```

```
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                  284
Outlet_Location_Type           0
Outlet_Type                    0
```

```
Item_Outlet_Sales              0
dtype: int64
```

```
df1.describe()# statistical summary report
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outle
count	1000.000000	1000.000000	1000.000000	1000.000000	1000
mean	13.032138	0.066694	138.100303	1997.389000	2190
std	4.261831	0.052238	62.152665	8.417989	1758
min	4.610000	0.000000	31.290000	1985.000000	33
25%	9.395000	0.026658	90.055400	1987.000000	807
50%	13.032138	0.054691	140.099600	1999.000000	1757
75%	16.350000	0.095443	182.662750	2004.000000	3087
max	21.350000	0.328391	265.222600	2009.000000	9275

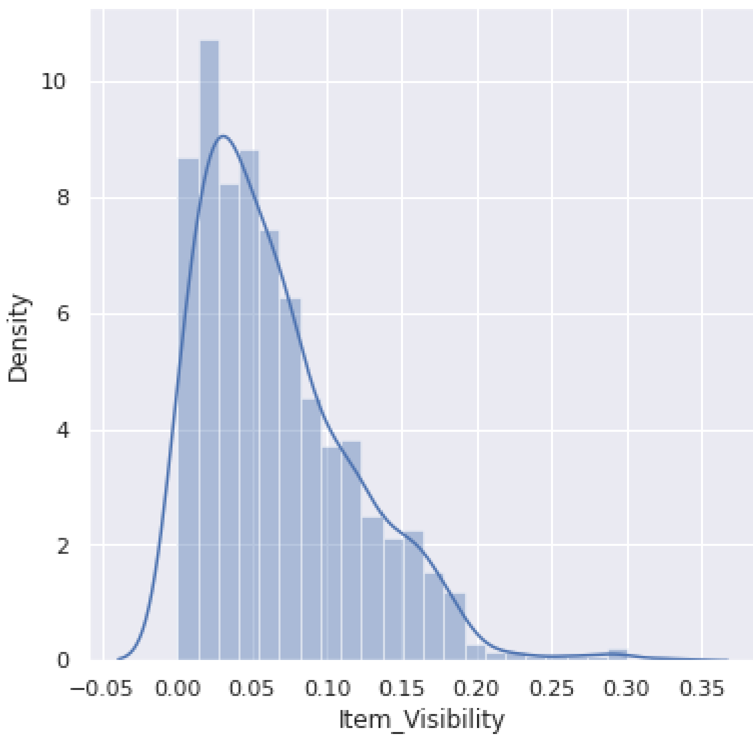
```
sns.set()
```

```
#
plt.figure(figsize=(6,6))
sns.distplot(df1['Item_Weight'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
```


```
plt.figure(figsize=(6,6))
sns.distplot(df1['Item_Visibility'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
```




```
df1.head() # starting 5 rows will be print
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950

```
#count of item fat content
df1['Item_Fat_Content'].value_counts()
```

```
Low Fat    622
Regular    328
LF          26
low fat     13
reg         11
Name: Item_Fat_Content, dtype: int64
```

```
# here we are replacing the lowfat into LowFat ,LF converted to Low FAt and reg into Regular.
df1.replace({'Item_Fat_Content':{'low fat':'Low Fat','LF':'Low Fat','reg':'Regular'}},inplace
```

```
#Total fat conetnt count
df1['Item_Fat_Content'].value_counts()
```

```
Low Fat    661
Regular    339
Name: Item_Fat_Content, dtype: int64
```

```
df1.count()#  total count of each and every column
```

```
Item_Identifier              1000
Item_Weight                  1000
Item_Fat_Content             1000
Item_Visibility              1000
Item_Type                    1000
Item_MRP                     1000
Outlet_Identifier            1000
Outlet_Establishment_Year    1000
Outlet_Size                   716
Outlet_Location_Type         1000
Outlet_Type                  1000
Item_Outlet_Sales            1000
dtype: int64
```

```
df1.duplicated()# identify duplicate values if duplicate values are there then it will return
```

```
0      False
1      False
2      False
3      False
4      False
       ...
995    False
996    False
997    False
998    False
999    False
Length: 1000, dtype: bool
```

```
df1.corr()# variable is related to another variable
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_
Item_Weight	1.000000	-0.017213	0.047787	-0.00
Item_Visibility	-0.017213	1.000000	0.012226	-0.11
Item_MRP	0.047787	0.012226	1.000000	0.05
Outlet_Establishment_Year	-0.009247	-0.114948	0.055565	1.00
Item_Outlet_Sales	0.011355	-0.093142	0.581664	-0.02

```
df1.loc[:6:]# up to 6 rows all columns
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614
5	FDP36	10.395	Regular	0.000000	Baking Goods	51.4008
6	FDO10	13.650	Regular	0.012741	Snack Foods	57.6588

```
# Here we are finding the total null values
df1.isnull()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_M
0	False	False	False	False	False	Fal
1	False	False	False	False	False	Fal
2	False	False	False	False	False	Fal
3	False	False	False	False	False	Fal
4	False	False	False	False	False	Fal
...	...	...	...	...	...	
995	False	False	False	False	False	Fal
996	False	False	False	False	False	Fal
997	False	False	False	False	False	Fal
998	False	False	False	False	False	Fal

```
df1.isnull().sum() # we are checking the percentage of missing values in our data.
```

```
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                  284
Outlet_Location_Type           0
Outlet_Type                    0
Item_Outlet_Sales              0
dtype: int64
```

```
df1.isna().all()#from all values having null values or not
```

```
Item_Identifier              False
Item_Weight                  False
Item_Fat_Content             False
Item_Visibility              False
Item_Type                    False
Item_MRP                     False
Outlet_Identifier            False
Outlet_Establishment_Year    False
Outlet_Size                  False
Outlet_Location_Type         False
Outlet_Type                  False
Item_Outlet_Sales            False
dtype: bool
```

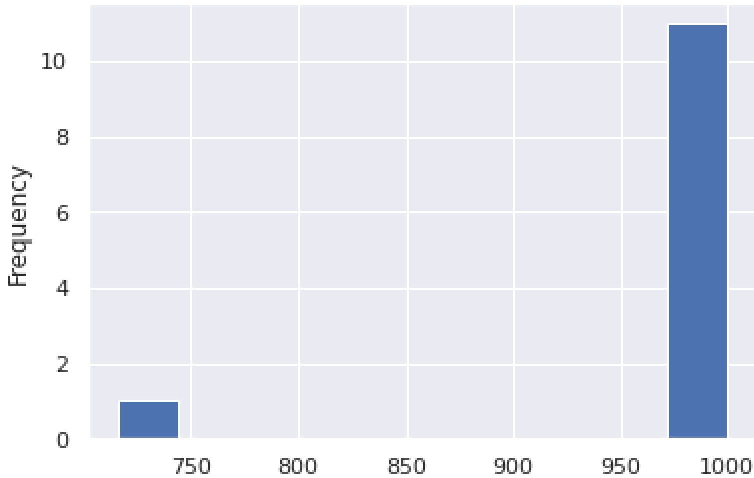
```
print(df1.shape)#drop the columns which having null values
```

```
ds=df1.dropna(axis=1)
ds.shape
```

```
(1000, 12)
(1000, 11)
```

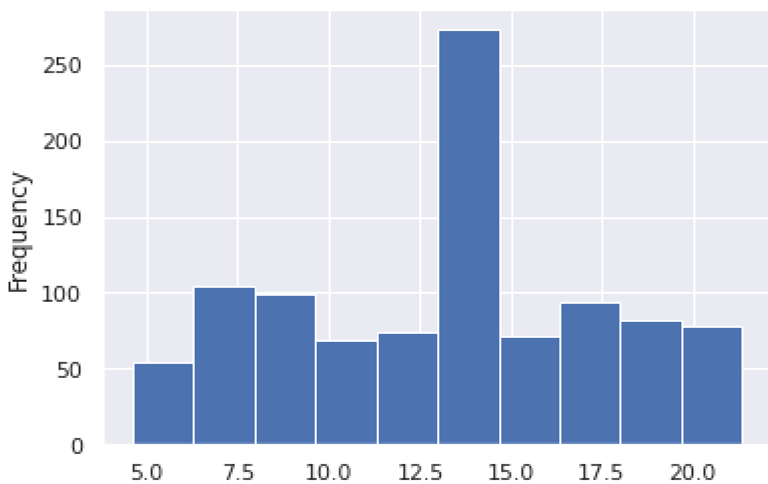
```
df1.count().plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814f434990>
```



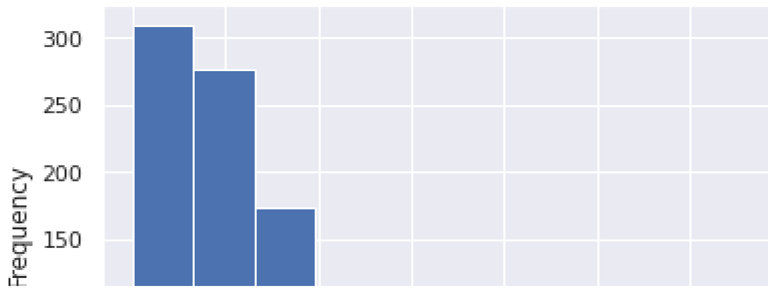

```
df1.Item_Weight.plot.hist()#here we are using histogram for knowing the item weights
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814f3bb510>
```



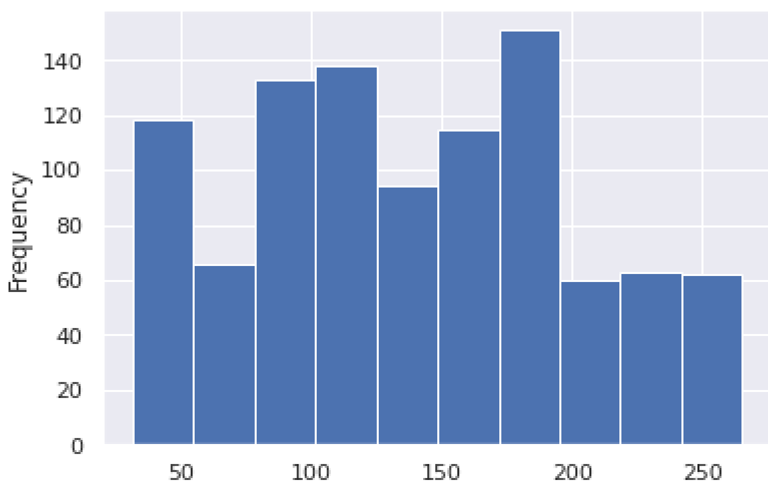

```
#here we plotting histogram for item visibilty that means the product is available percentage
df1.Item_Visibility.plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814f34f510>



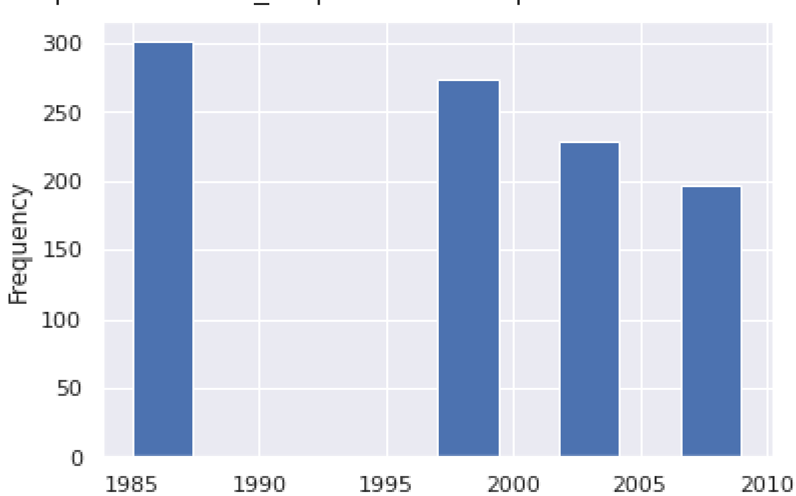

```
#here we are able to see the market retail prices of the items and by observing that we are a
df1.Item_MRP.plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814f2ca610>



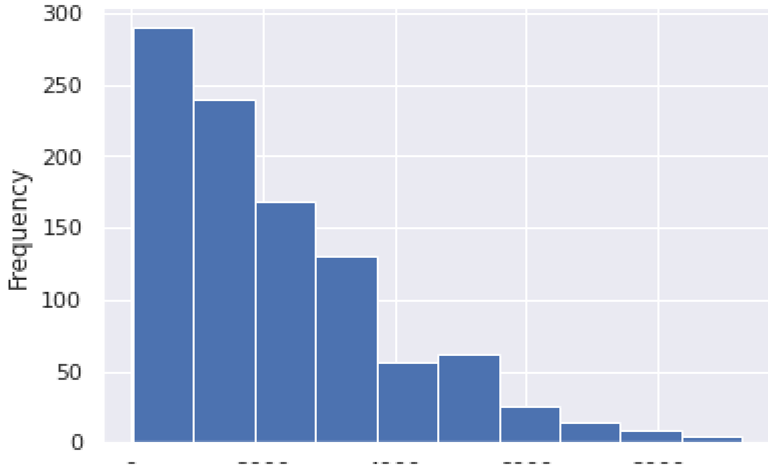

```
#here we are able to know about the store establishment year.
df1.Outlet_Establishment_Year.plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814f26e110>



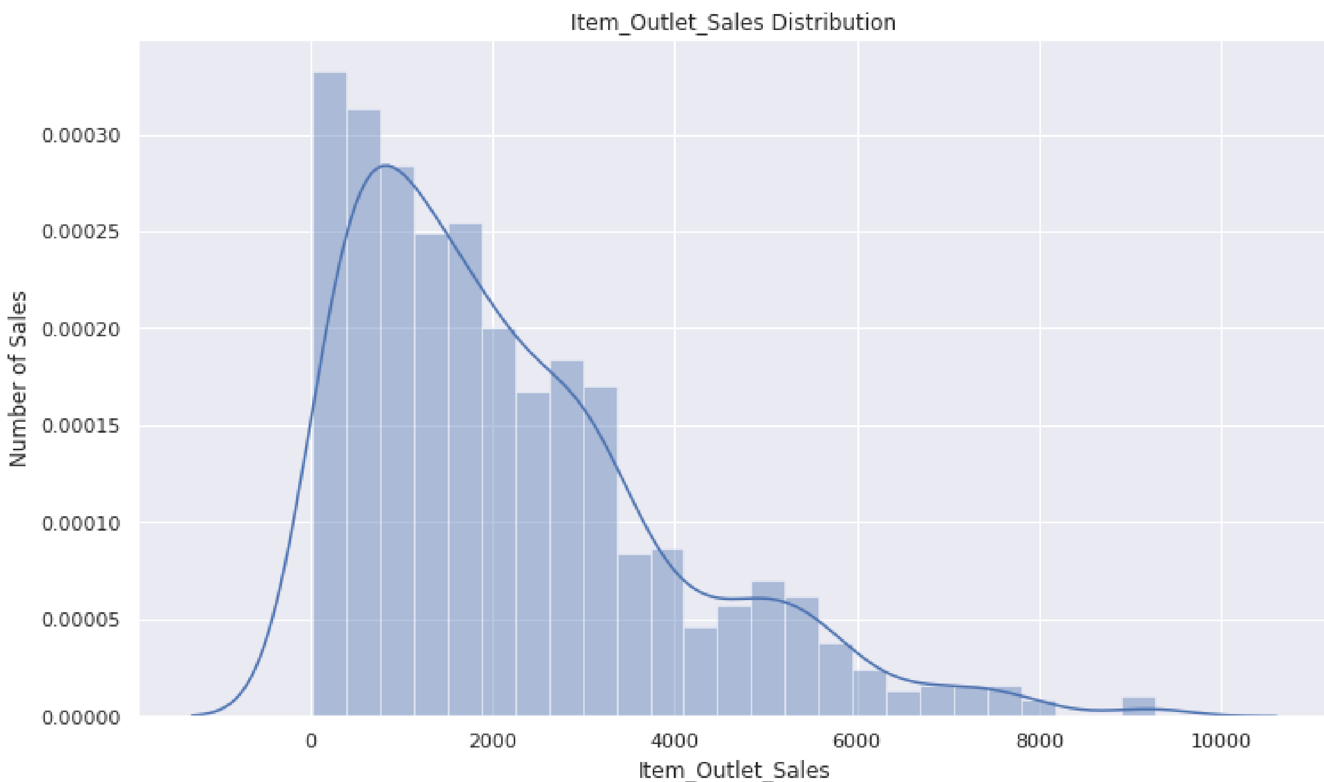

```
#Here we are able to know the sales of product and this is our target variable.
df1.Item_Outlet_Sales.plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814f172110>
```




```
#we can see that our target variable is skewed towards the right,therefore we have to normali
plt.figure(figsize=(12,7))
sns.distplot(df1.Item_Outlet_Sales, bins = 25)
plt.xlabel("Item_Outlet_Sales")
plt.ylabel("Number of Sales")
plt.title("Item_Outlet_Sales Distribution")
```

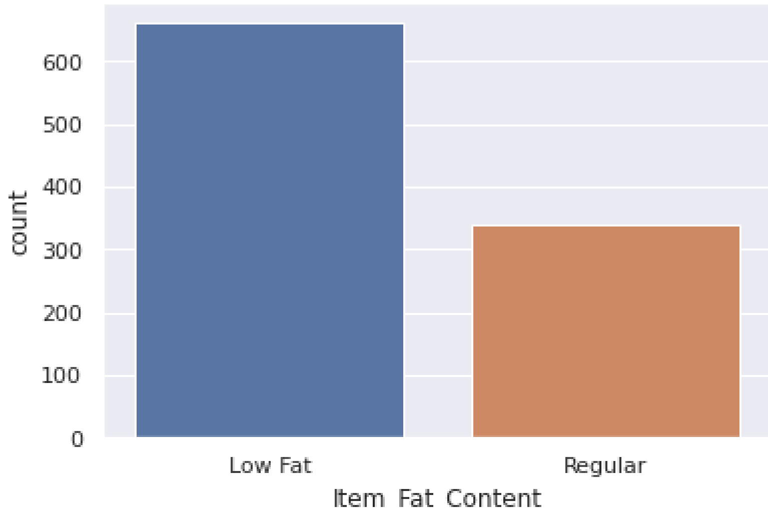
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
Text(0.5, 1.0, 'Item_Outlet_Sales\xa0Distribution')
```




```
#There are two possible type "low fat" or "regular"
```

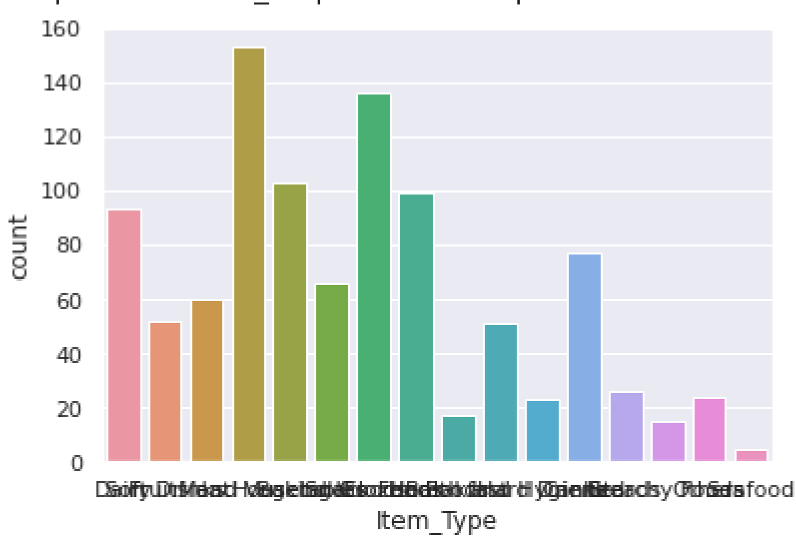
```
sns.countplot(df1.Item_Fat_Content)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass tl
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f814f1bd050>
```



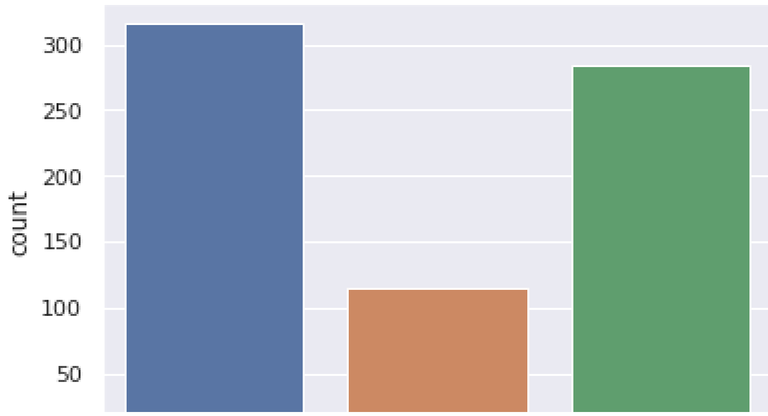

```
#we have 16 different types of unique values and it is high number for categorical variable.t
sns.countplot(df1.Item_Type)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass tl
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f814f050490>
```



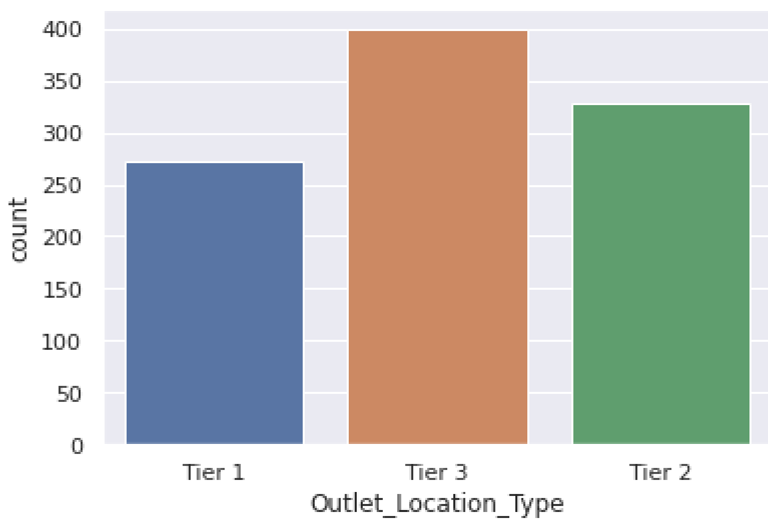

```
#There seems to be less number of stores with size equals to "high".it will be very intrestin
sns.countplot(df1.Outlet_Size)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass tl
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f814ef9d8d0>
```



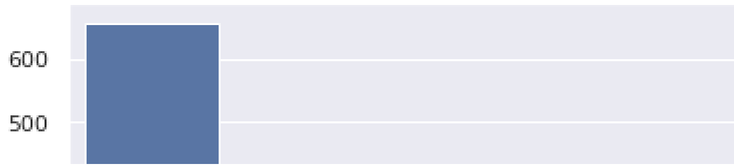

```
#from above graph we can see that bigmart is a brand of medium and small size city compare to
sns.countplot(df1.Outlet_Location_Type)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass tl
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f814ef68610>
```



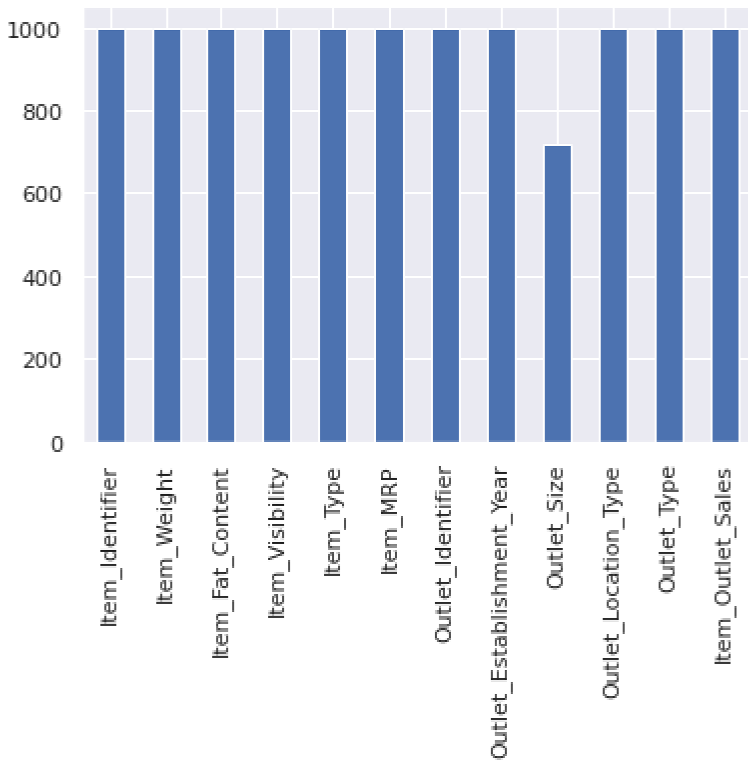

```
#there seems like supermart type2 grocery store and supermarket type3 all have low numbers of
sns.countplot(df1.Outlet_Type)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f814eec1fd0>
```



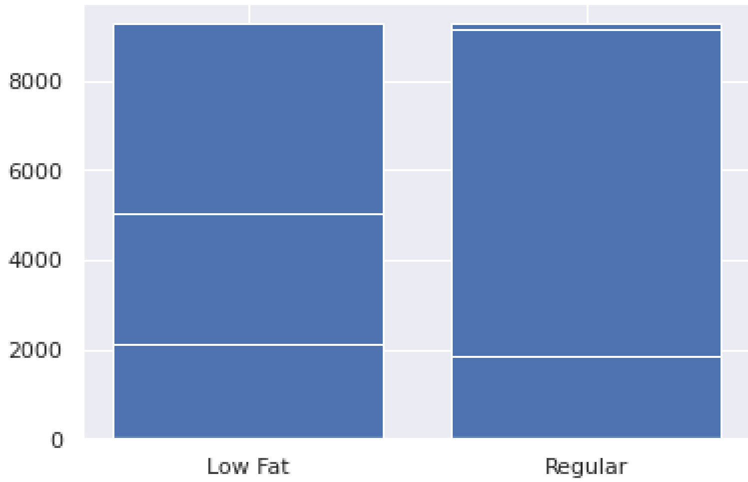

```
df1.count().plot.bar()# bar chart for total data set
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814ee41910>
```

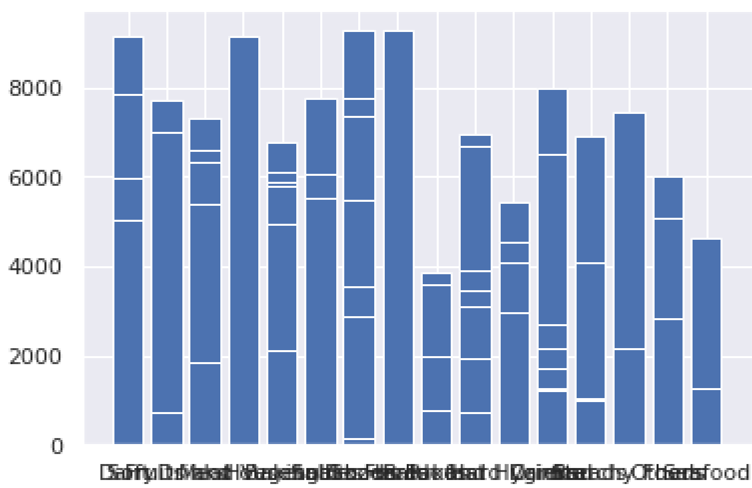



```
#here we are able to know the relation between target variable and item identifier
import matplotlib.pyplot as plt
plt.bar(df1.Item_Identifier,df1.Item_Outlet_Sales)
plt.show()
```

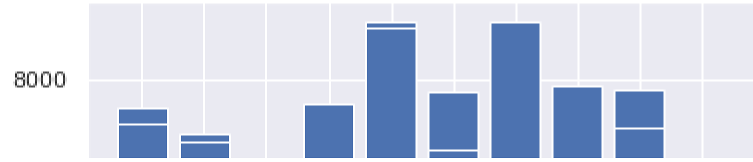
```
#Low fat products seem to be higher sales than the regular products
plt.bar(df1.Item_Fat_Content,df1.Item_Outlet_Sales)
plt.show()
```



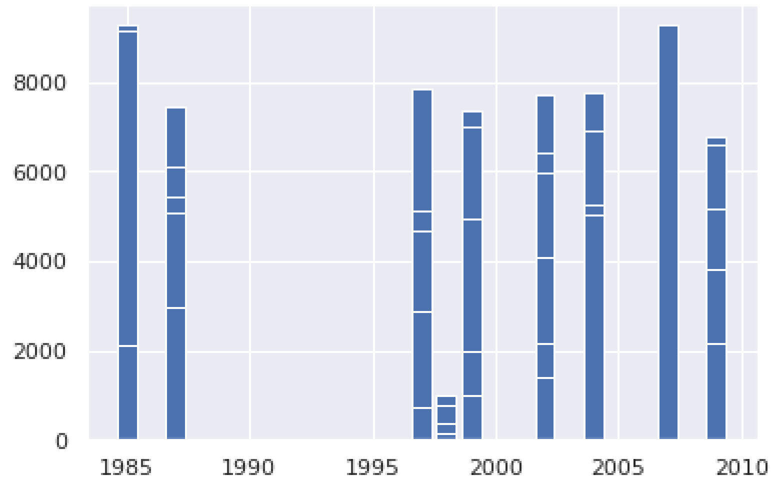

```
plt.bar(df1.Item_Type,df1.Item_Outlet_Sales)
plt.show()
```




```
#out of 10- two groceries stores 6 supermarket type1, one supermarket Type2 and 1 supermarket
plt.bar(df1.Outlet_Identifier,df1.Item_Outlet_Sales)
plt.show()
```

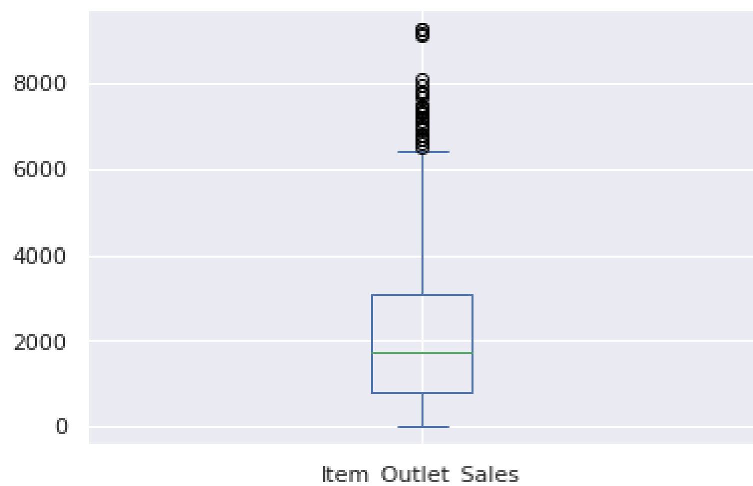



```
#There seems to be no appreciable meaning between year of store establishment and sales for t
plt.bar(df1.Outlet_Establishment_Year,df1.Item_Outlet_Sales)
plt.show()
```



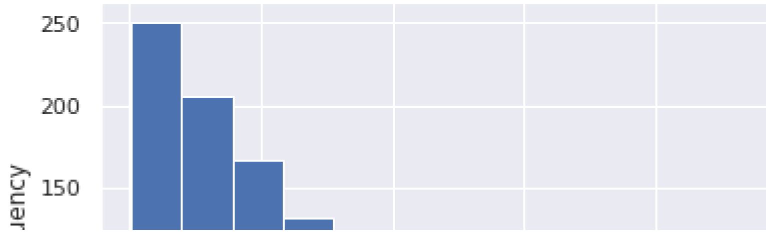

```
df1.Item_Outlet_Sales.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814cfde3d0>
```



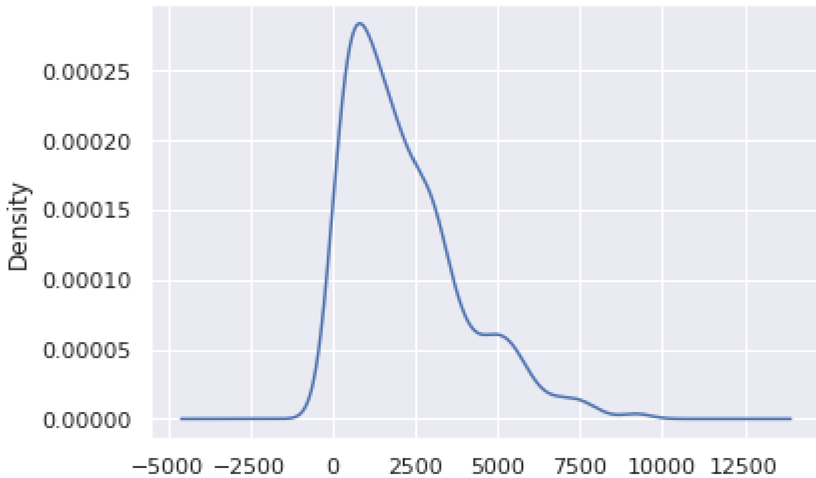

```
df1.Item_Outlet_Sales.plot.hist(bins=12)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814d290e50>
```



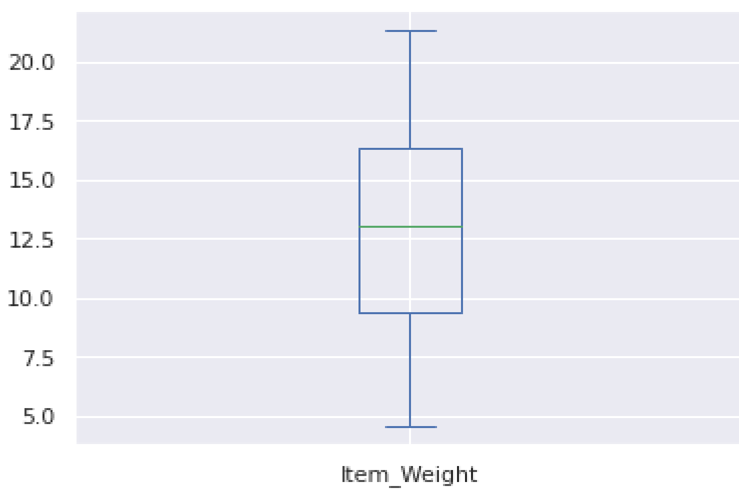

```
df1.Item_Outlet_Sales.plot.kde()#kernal density estimation
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814d458f10>
```



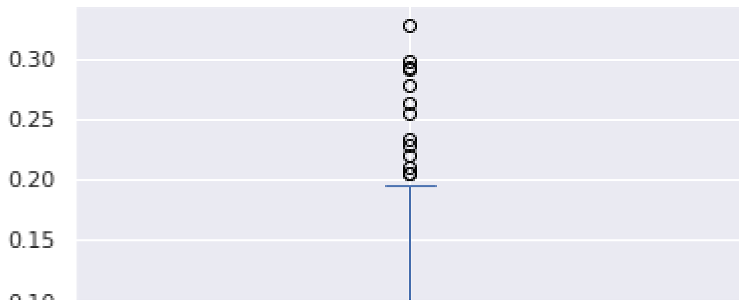

```
df1.Item_Weight.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814e3db990>
```



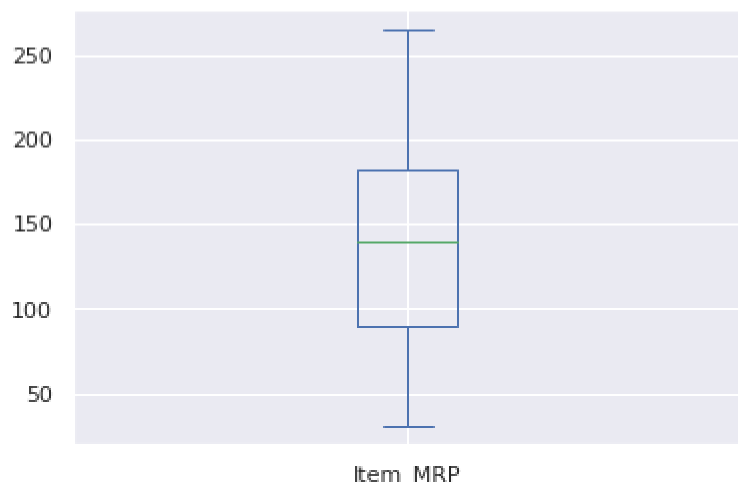

```
df1.Item_Visibility.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814db2c850>
```



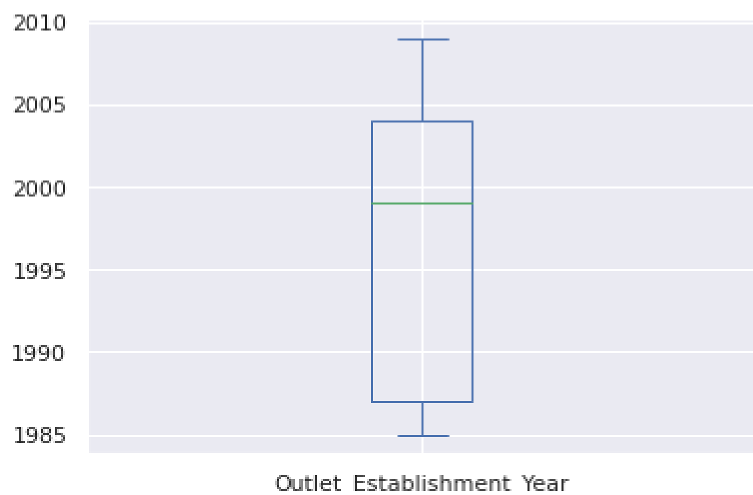

```
df1.Item_MRP.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814ce5ba50>
```



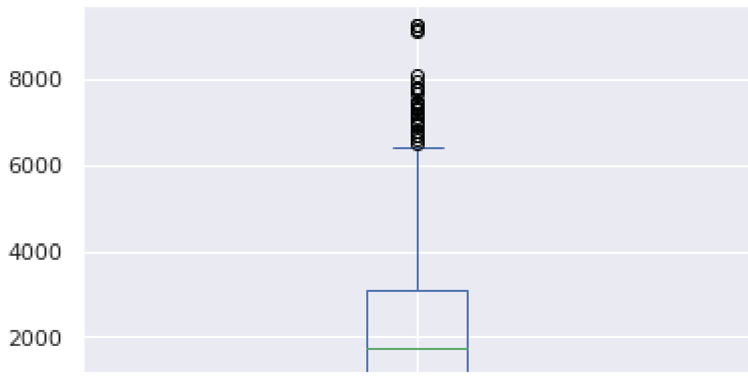

```
df1.Outlet_Establishment_Year.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814e4ce710>
```



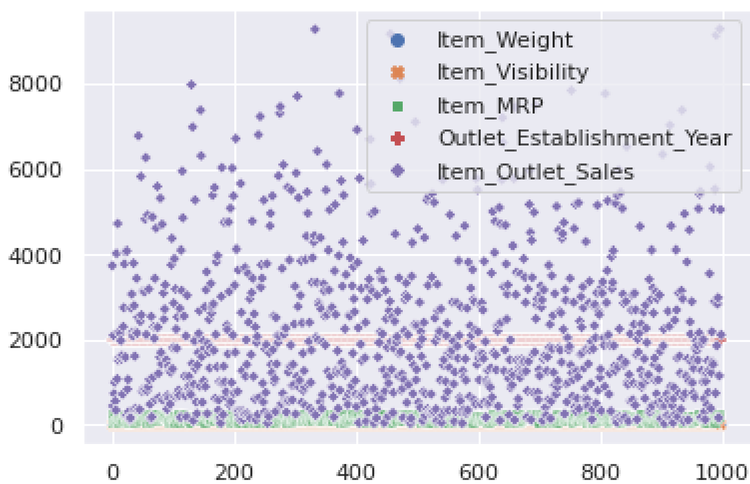

```
df1.Item_Outlet_Sales.plot.box()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814f567750>



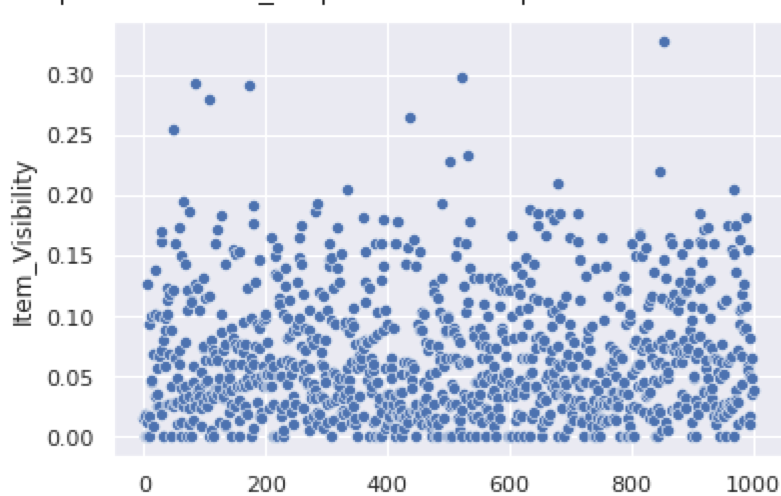

```
import seaborn as sns
sns.scatterplot(data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814f4a8750>



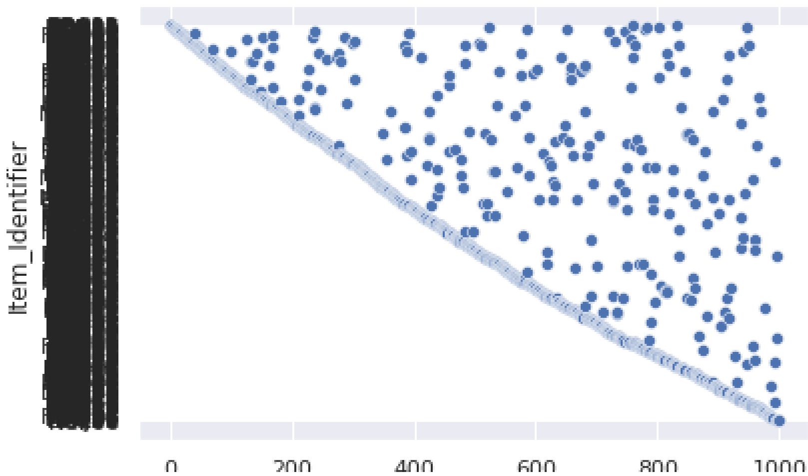

```
sns.scatterplot(data=df1.Item_Visibility)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814dbae5d0>



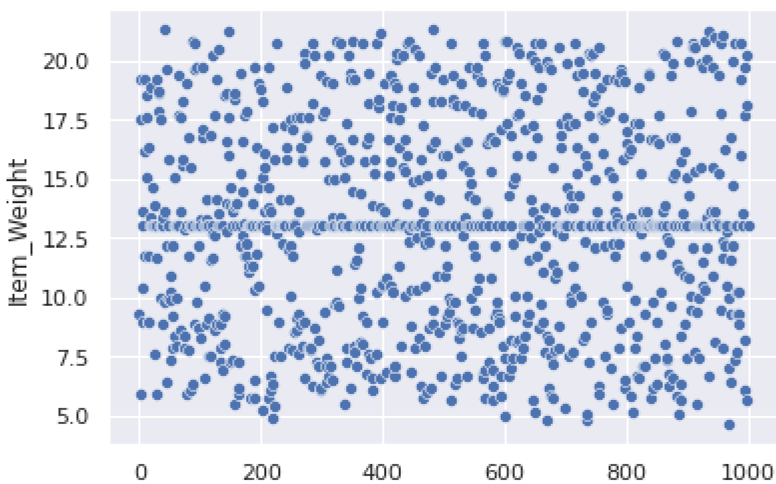

```
sns.scatterplot(data=df1.Item_Identifier)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814ed87350>
```



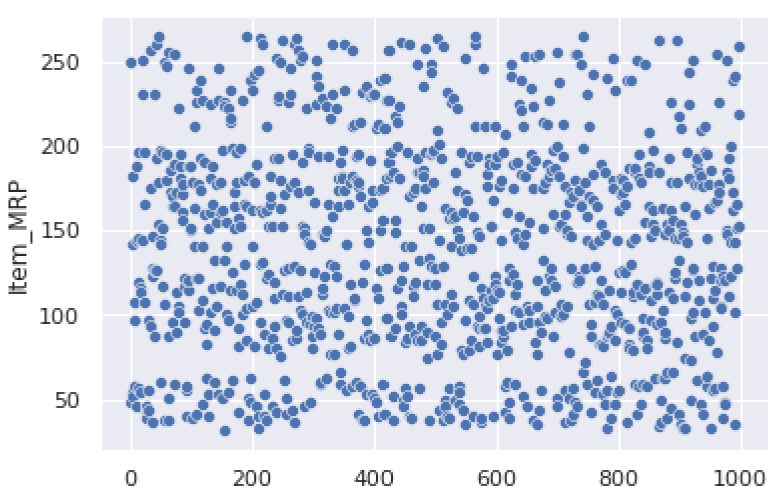

```
sns.scatterplot(data=df1.Item_Weight)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814ce36090>
```



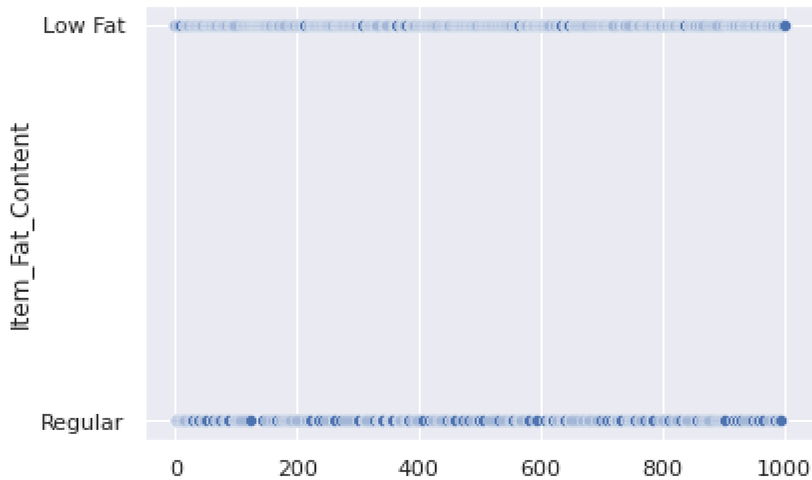

```
sns.scatterplot(data=df1.Item_MRP)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814d99a3d0>
```




```
sns.scatterplot(data=df1.Item_Fat_Content)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814ce36d90>
```

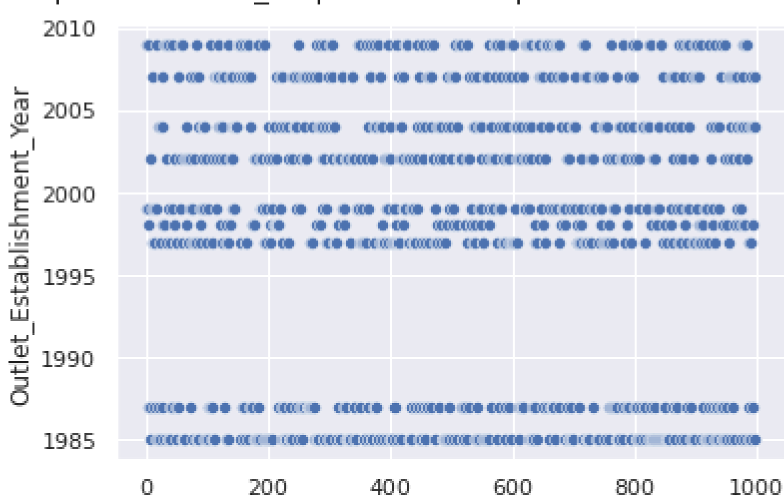



```
sns.scatterplot(data=df1.Item_Type)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814d0aad90>
```

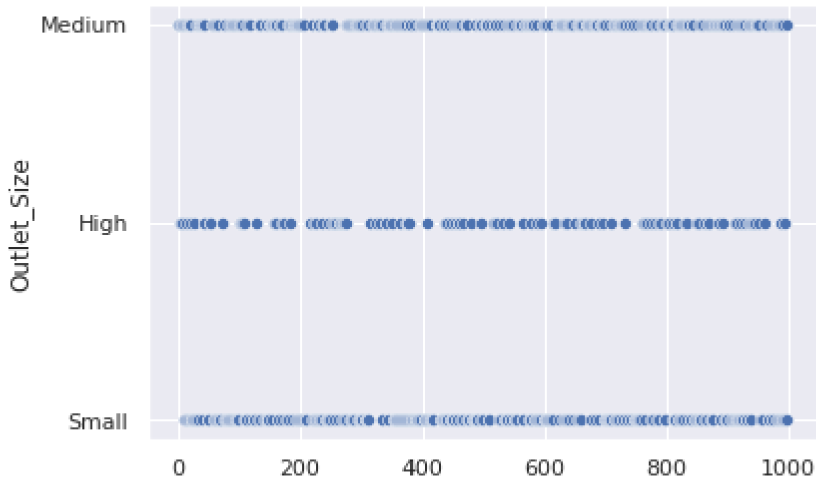

```
sns.scatterplot(data=df1.Outlet_Establishment_Year)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814e1a3090>
```



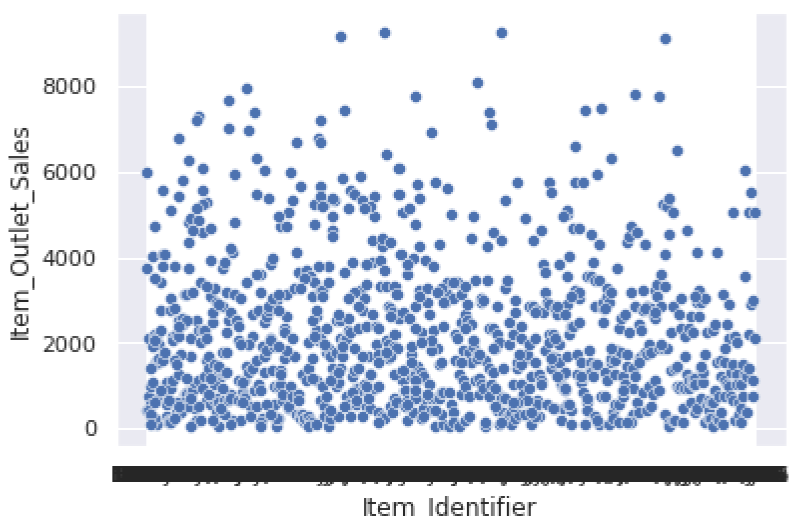
```
sns.scatterplot(data=df1.Outlet_Size)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814d657bd0>



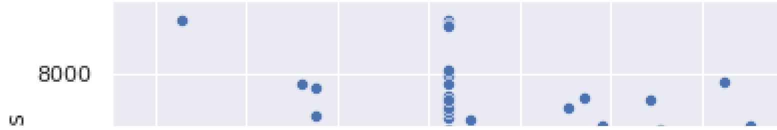

```
#here we are able to know the relation between identifier and outlet sales.
sns.scatterplot(x='Item_Identifier',y='Item_Outlet_Sales',data=df1,palette='blues')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814e1829d0>



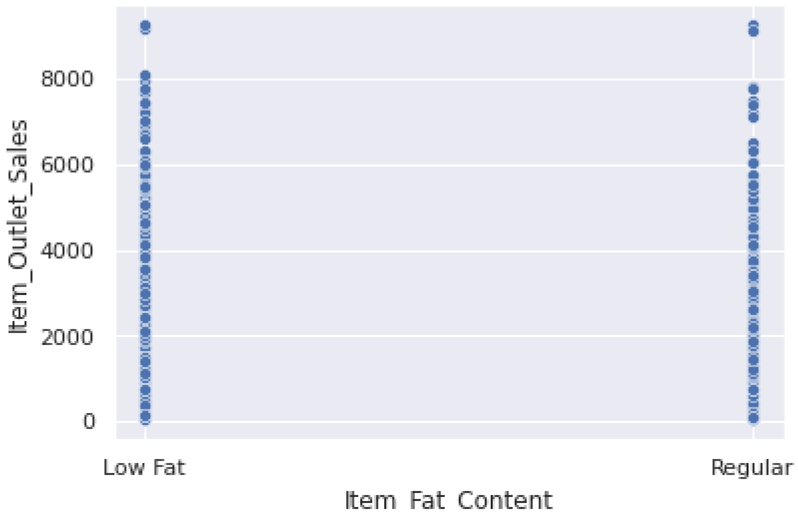

```
#Item weight and have a low correlation with our target variable.
sns.scatterplot(x='Item_Weight',y='Item_Outlet_Sales',data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814e1bf090>



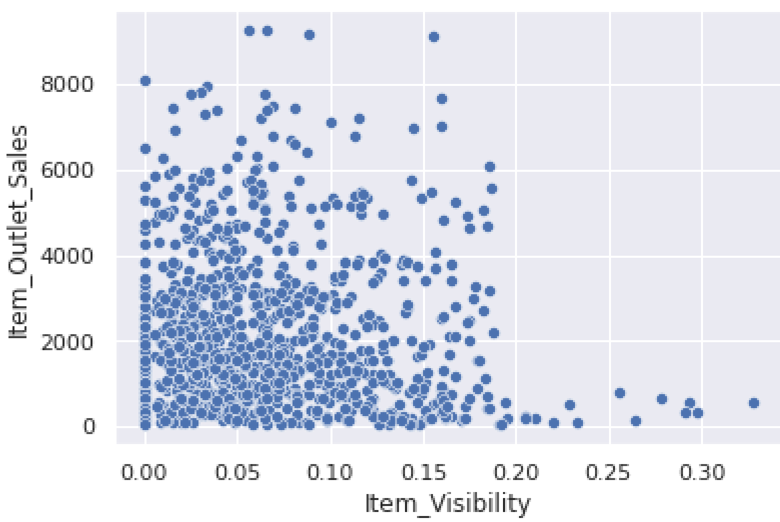

```
#it shows that there is a relation between sales and fat content.
sns.scatterplot(x='Item_Fat_Content',y='Item_Outlet_Sales',data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814d2e7d50>



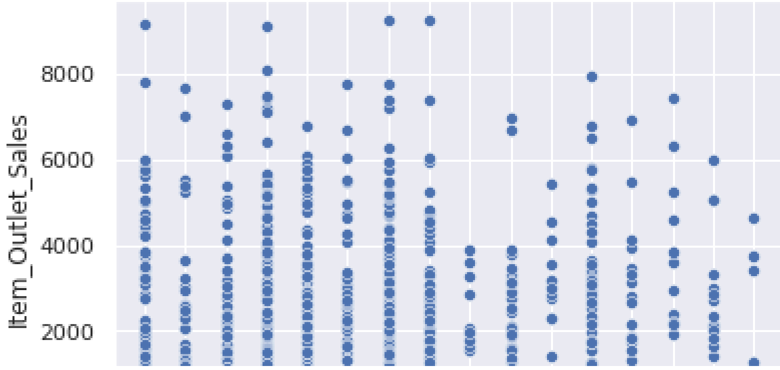

```
#it shows that visibility of product in stores and that is for sales if the visibility of ite
sns.scatterplot(x='Item_Visibility',y='Item_Outlet_Sales',data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814df85bd0>



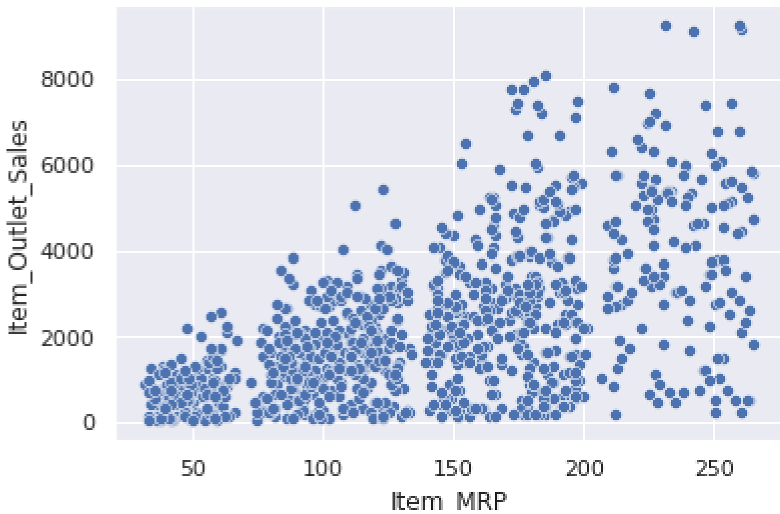

```
sns.scatterplot(x='Item_Type',y='Item_Outlet_Sales',data=df1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814f47eed0>
```



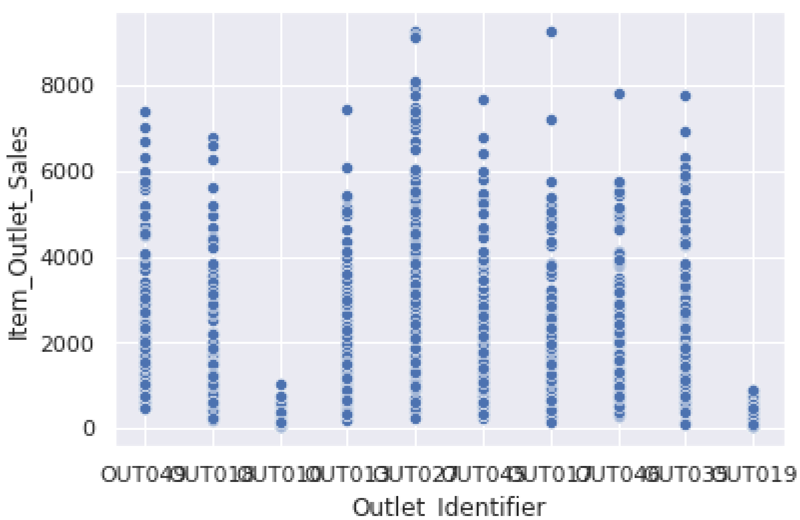

```
sns.scatterplot(x='Item_MRP',y='Item_Outlet_Sales',data=df1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814f238450>
```



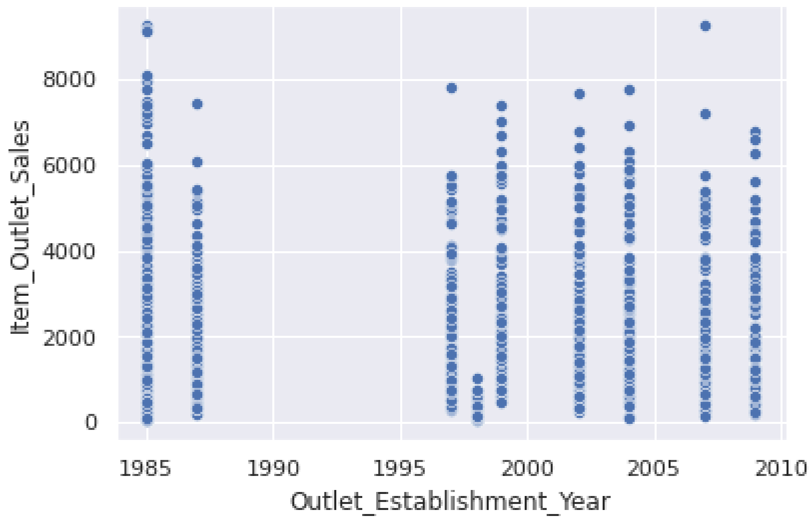

```
sns.scatterplot(x='Outlet_Identifier',y='Item_Outlet_Sales',data=df1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814e9d71d0>
```



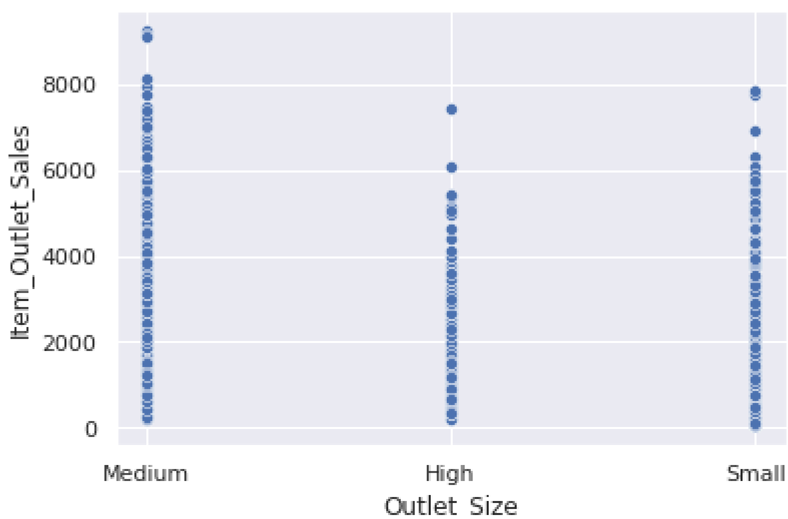

```
sns.scatterplot(x='Outlet_Establishment_Year',y='Item_Outlet_Sales',data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814eb28b10>




```
sns.scatterplot(x='Outlet_Size',y='Item_Outlet_Sales',data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814dfed850>

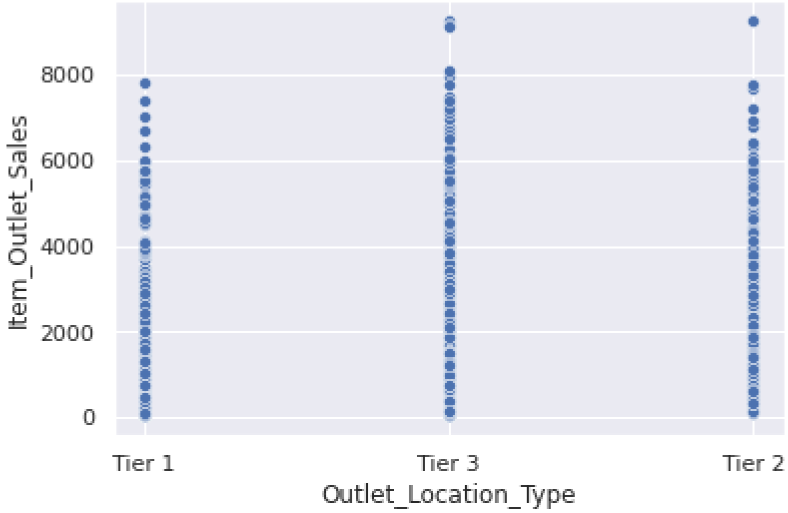



```
sns.scatterplot(x='Outlet_Size',y='Item_Outlet_Sales',data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814e4d7c10>


```
sns.scatterplot(x='Outlet_Location_Type',y='Item_Outlet_Sales',data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814ed26390>

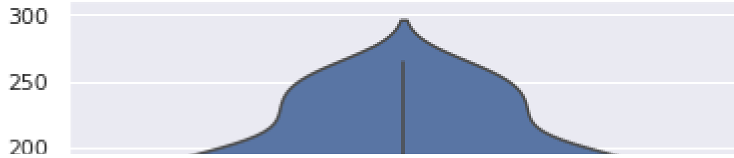



```
sns.scatterplot(x='Outlet_Type',y='Item_Outlet_Sales',data=df1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814e87abd0>

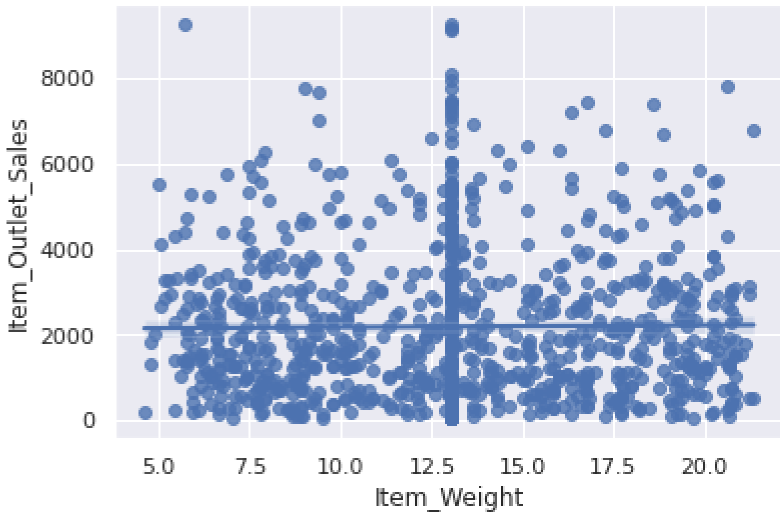

```
sns.violinplot(data=df1.Item_MRP)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814e784050>
```



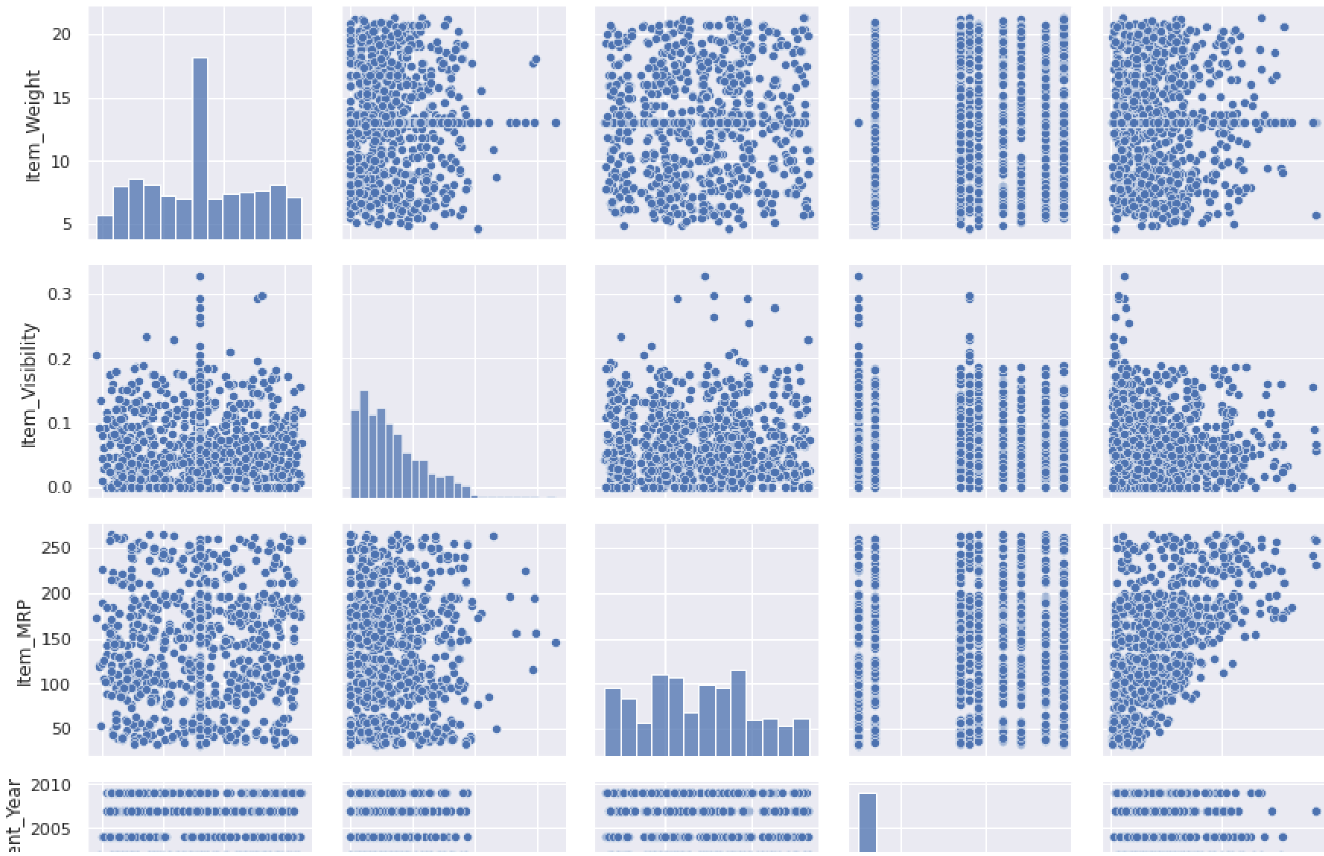

```
sns.regplot(x=df1.Item_Weight,y=df1.Item_Outlet_Sales)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814e70c850>
```



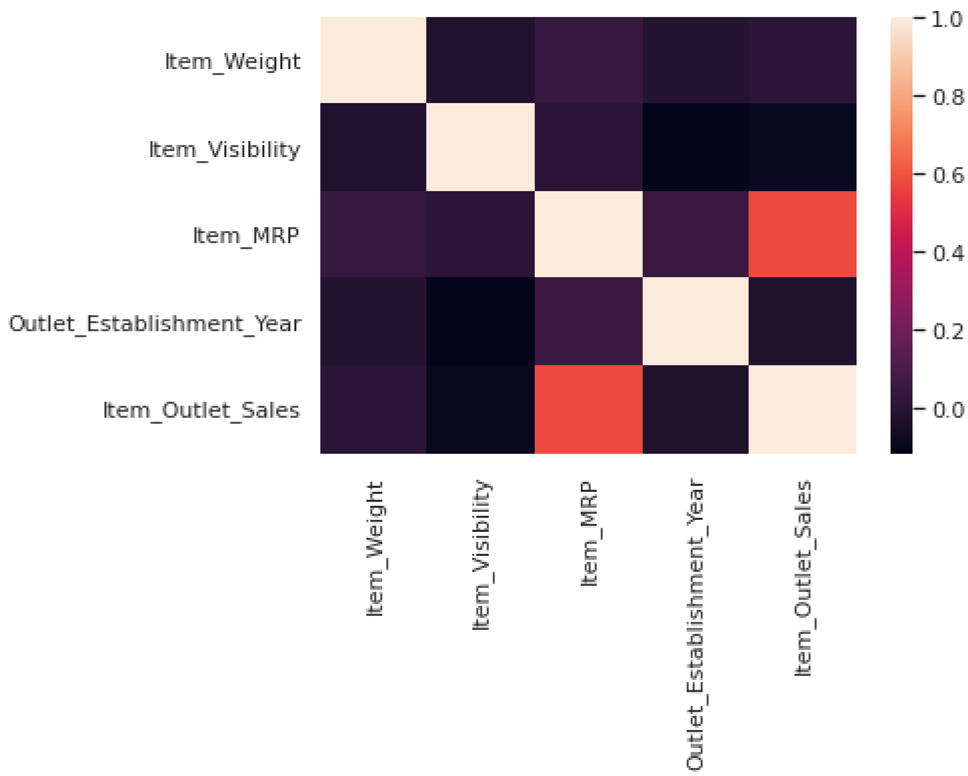

```
sns.pairplot(df1)
```

<seaborn.axisgrid.PairGrid at 0x7f814ed2d790>



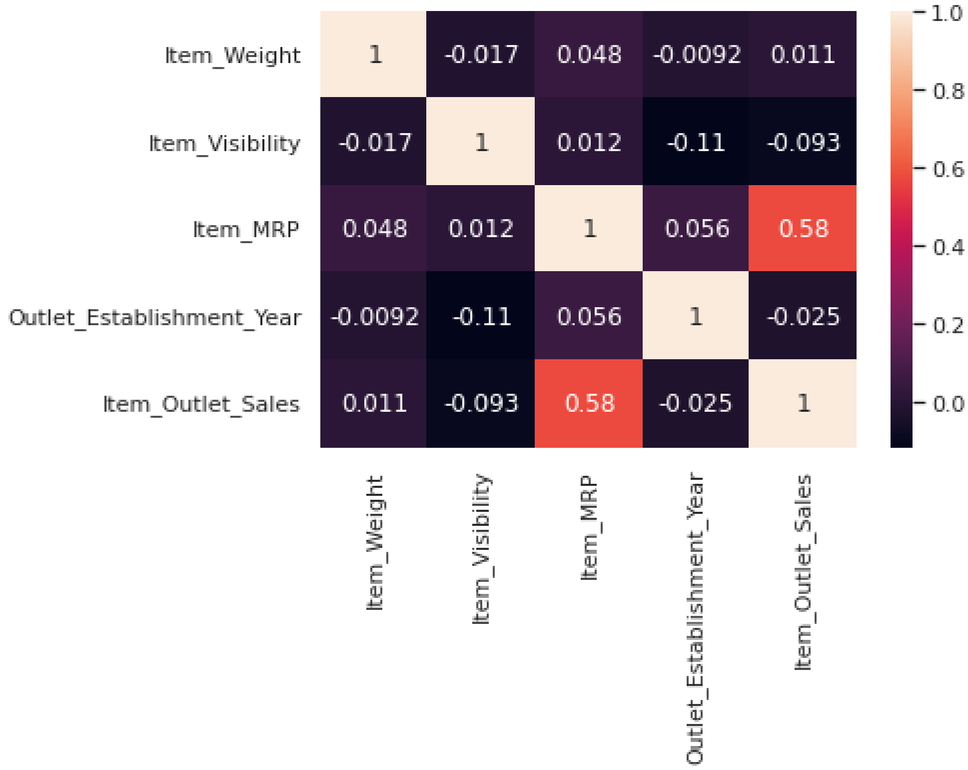

```
sns.heatmap(df1.corr())
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f814c9999d0>



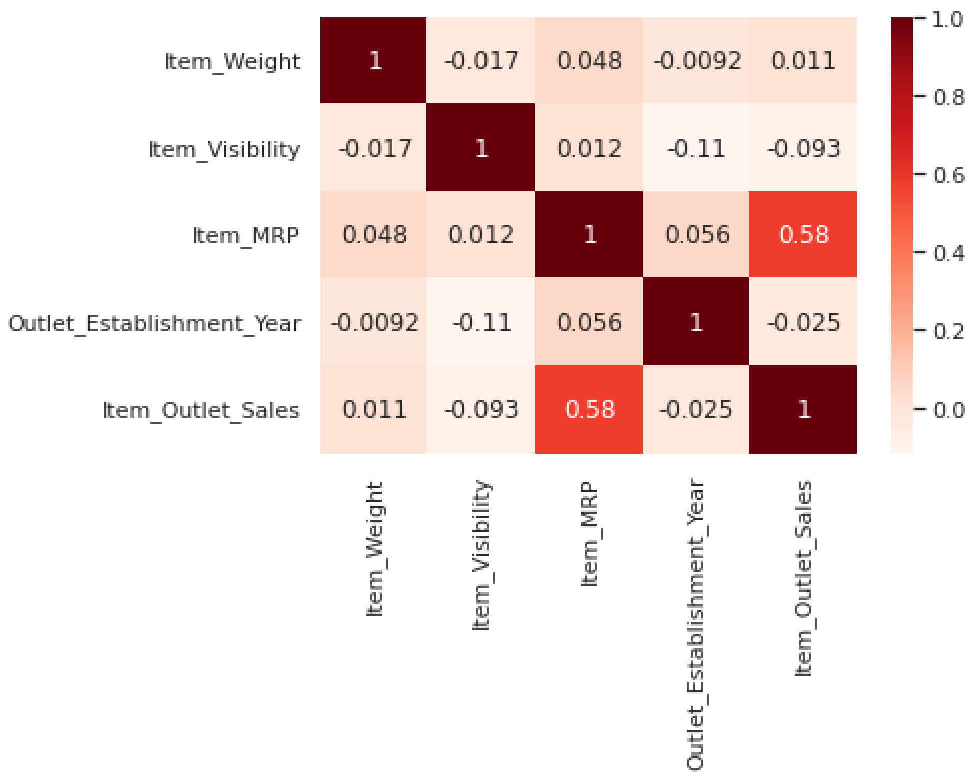

```
sns.heatmap(df1.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814c86acd0>
```




```
sns.heatmap(df1.corr(),annot=True,cmap='Reds')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f814c78f210>
```




```
df1.drop(['Item_Fat_Content'],axis=1)
```

	Item_Identifier	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Identif
0	FDA15	9.300000	0.016047	Dairy	249.8092	OUT
1	DRC01	5.920000	0.019278	Soft Drinks	48.2692	OUT
2	FDN15	17.500000	0.016760	Meat	141.6180	OUT
3	FDX07	19.200000	0.000000	Fruits and Vegetables	182.0950	OUT
4	NCD19	8.930000	0.000000	Household	53.8614	OUT
...	...	...	...	...	...	
995	FDO34	17.700000	0.050112	Snack Foods	165.9816	OUT
996	NCL30	18.100000	0.048931	Household	127.3336	OUT
997	FDK28	5.695000	0.065961	Frozen Foods	259.2646	OUT
998	DRJ39	20.250000	0.036319	Dairy	219.3482	OUT
999	NCP06	13.032138	0.039056	Household	152.3366	OUT

1000 rows × 11 columns

```
#data cleaning
df1.apply(lambda x: sum(x.isnull()))
```

```
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                  284
Outlet_Location_Type           0
Outlet_Type                    0
Item_Outlet_Sales              0
dtype: int64
```

```
df1.Item_Outlet_Sales = df1.Item_Outlet_Sales.fillna(df1.Item_Outlet_Sales.mean())
```

```
df1['Outlet_Size'].value_counts()
```

```
Medium    316
Small     285
High      115
Name: Outlet_Size, dtype: int64
```

```
df1.Outlet_Size = df1.Outlet_Size.fillna('Medium')
```

```
df1.apply(lambda x: sum(x.isnull()))
```

```
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
```

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
encoder = LabelEncoder()
```

```
df1['Item_Identifier']=encoder.fit_transform(df1['Item_Identifier'])

df1['Item_Fat_Content']=encoder.fit_transform(df1['Item_Fat_Content'])

df1['Item_Type']=encoder.fit_transform(df1['Item_Type'])

df1['Outlet_Identifier']=encoder.fit_transform(df1['Outlet_Identifier'])

df1['Outlet_Size']=encoder.fit_transform(df1['Outlet_Size'].astype(str))

df1['Outlet_Location_Type']=encoder.fit_transform(df1['Outlet_Location_Type'])

df1['Outlet_Type']=encoder.fit_transform(df1['Outlet_Type'])
```

```
df1.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP
0	84	9.30	0	0.016047	4	249.8092
1	3	5.92	1	0.019278	14	48.2692
2	325	17.50	0	0.016760	10	141.6180
3	543	19.20	1	0.000000	6	182.0950
4	638	8.93	0	0.000000	9	53.8614

```
X=df1.drop(columns='Item_Outlet_Sales',axis=1)
Y=df1['Item_Outlet_Sales']
```

```
print(X)
```

```
     Item_Identifier  Item_Weight  Item_Fat_Content  Item_Visibility  \
0                 84     9.300000                 0         0.016047   
1                  3     5.920000                 1         0.019278   
2                325    17.500000                 0         0.016760   
3                543    19.200000                 1         0.000000   
4                638     8.930000                 0         0.000000   
..               ...          ...               ...              ...   
995              354    17.700000                 0         0.050112   
996              677    18.100000                 0         0.048931   
997              271     5.695000                 0         0.065961   
998               49    20.250000                 0         0.036319   
999              705    13.032138                 0         0.039056   

     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   
..         ...       ...                ...                        ...   
995         13  165.9816                  0                       1998   
996          9  127.3336                  6                       2004   
997          5  259.2646                  2                       2007   
998          4  219.3482                  6                       2004   
999          9  152.3366                  5                       1985   

     Outlet_Size  Outlet_Location_Type  Outlet_Type  
0              1                     0            1  
1              1                     2            2  
2              1                     0            1  
3              1                     2            0  
```

```
4            0                     2            1
..         ...                   ...          ...
995          1                     2            0
996          2                     1            1
997          1                     1            1
998          2                     1            1
999          1                     2            3

[1000 rows x 11 columns]
```

```
print(Y)
```

```
0      3735.1380
1       443.4228
2      2097.2700
3       732.3800
4       994.7052
         ...
995     167.7816
996    1150.5024
997    9275.9256
998    5038.1086
999    2115.9124
Name: Item_Outlet_Sales, Length: 1000, dtype: float64
```

Model Evaluation

```
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)
```

```
(1000, 11) (800, 11) (200, 11)
```

```
regressor= XGBRegressor( )
```

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

```
[15:07:28] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
XGBRegressor()
```

```
training_data_prediction =regressor.predict(X_train)
```

```
r2_train =metrics.r2_score(y_train,training_data_prediction)
```

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

```
R Squared value= 0.7646926501420891
```

```
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= 0.6282989950899187
```

```
from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor(random_state=4)

rfr.fit(X_train,y_train)
```

```
RandomForestRegressor(random_state=4)
```

```
y_pred = rfr.predict(X_test)

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

y_pred
```

```
array([1957.099126, 4174.53271 , 1022.728722, 1184.59136 ,  741.994152,
       3381.072218, 4472.384998,  330.962522,  892.451636, 1715.886444,
       1054.800308, 1399.098804, 1508.776038,  740.13657 , 2101.451224,
       4980.823168, 2308.16215 , 1315.334506, 2774.541734, 1028.747554,
       1570.32259 , 2976.651982, 5169.73726 , 2172.791694, 5229.39294 ,
       4257.884212, 2465.896828, 2679.698524, 2977.204596, 1797.553472,
        948.079226, 2115.199994,  157.06222 ,  873.216674, 2188.977292,
       4118.512298, 4239.308392, 2308.088912,  833.588258, 5211.323128,
        273.71038 , 4951.741024, 2155.174626, 2733.80809 , 3491.774784,
       1994.86996 , 1972.39921 ,  875.247364,  970.70311 ,  441.518612,
        847.203868, 2488.620582, 3841.812476, 1410.736988,  556.40906 ,
       3222.20568 , 1866.556984, 5880.598604, 3241.24756 , 1481.584766,
        249.794844, 2735.033162, 5580.023194, 3646.167146, 1369.310912,
       3088.865914,  831.897126, 3576.517808, 2370.860536, 1121.167252,
       3527.781248, 4191.34416 , 1811.04258 ,  487.06599 ,  325.596174,
       1164.55078 , 2547.71699 , 2424.650518, 5555.195512, 1748.417432,
        981.735416, 1142.972202, 2526.431364, 3992.010298, 1059.028138,
       2996.446216,  123.585796, 3143.288406, 5421.456266,  396.657008,
       2890.291064, 1732.91095 ,  907.172474,   96.60758 , 1913.5092  ,
       2335.16034 ,  317.127198, 2371.21341 , 1915.712998, 2790.394432,
       2016.535092, 1193.473132, 2001.121822, 5152.113534, 2296.131144,
       3014.522686, 2578.949668, 2144.395324,  145.983308, 2562.36459 ,
```

```
       4487.871506,  477.039042, 2768.276556, 1590.49633 , 2334.953942,
        702.91835 , 2051.636068, 3438.25778 , 5862.255814, 1710.580018,
        494.629478, 1985.94824 ,  861.059166, 3979.206964, 2468.32034 ,
       3852.498566, 2968.529222, 3830.493876, 2151.83231 ,  323.166004,
        512.885714, 2280.471528, 2758.262924,  787.967642, 1572.572994,
       2114.108082, 1994.277398, 3593.142834, 3446.753388, 2029.711274,
       3640.188262, 1903.981602, 2699.146542, 2307.855882, 1292.377722,
       2926.517242, 2569.801576,  963.91195 ,  156.389762, 1161.1552  ,
       3260.429258, 2577.491566, 2811.27392 , 1627.61468 , 1153.09902 ,
       3935.430614, 5327.525202, 5617.740764, 1688.042688, 2095.945058,
       5114.722206, 5285.373404, 3055.82226 , 5419.159256,  441.45869 ,
        103.625112, 2128.416124,  273.783618, 2105.552552, 1984.103974,
        842.96938 , 1548.331216, 2640.16332 , 3413.796288, 2539.114854,
        851.085482, 1672.25657 ,  779.258978,  754.091738,  797.12905 ,
       2303.36839 , 2948.987992, 2872.547494,  683.317198, 2420.775562,
       2438.8254  , 1247.689226, 1777.306494,  966.94134 ,  761.848308,
       3093.10706 , 1718.316614, 1153.937928,  132.234538, 1776.634036,
       1145.895064, 2118.555626,  917.292634, 2933.761146, 6083.867344])
```

```
a=mean_squared_error(y_test,y_pred)
```

```
a
```

```
1168307.1217097656
```

```
b=mean_absolute_error(y_test,y_pred)
```

```
b
```

```
778.91029854
```

```
c=r2_score(y_test,y_pred)
```

```
c
```

```
0.6228633380148202
```

```
total=len(y_test)
```

0s completed at 8:37 PM