avacado

June 18, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df=pd.read_csv("C:/Users/Ravi/Downloads/MLR/Datasets_MLR/Avacado_Price.csv")
```

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   AveragePrice  18249 non-null  float64
 1   Total_Volume  18249 non-null  float64
 2   tot_ava1      18249 non-null  float64
 3   tot_ava2      18249 non-null  float64
 4   tot_ava3      18249 non-null  float64
 5   Total_Bags    18249 non-null  float64
 6   Small_Bags    18249 non-null  float64
 7   Large_Bags    18249 non-null  float64
 8   Xlarge_Bags   18249 non-null  float64
 9   type          18249 non-null  object 
 10  year          18249 non-null  int64  
 11  region        18249 non-null  object 
dtypes: float64(9), int64(1), object(2)
memory usage: 1.7+ MB
```

[4]:
```
duplicated=df.duplicated()
```

[5]:
```
sum(duplicated)
```

[5]:
```
0
```

[6]:
```
df.isna().sum()
```

[6]:

```
AveragePrice    0
Total_Volume    0
tot_ava1        0
tot_ava2        0
tot_ava3        0
Total_Bags      0
Small_Bags      0
Large_Bags      0
Xlarge_Bags     0
type            0
year            0
region          0
dtype: int64
```

[7]:

```
df.isna().sum()
```

[7]:

```
AveragePrice    0
Total_Volume    0
tot_ava1        0
tot_ava2        0
tot_ava3        0
Total_Bags      0
Small_Bags      0
Large_Bags      0
Xlarge_Bags     0
type            0
year            0
region          0
dtype: int64
```

[8]:

```
df.shape
```

[8]:

```
(18249, 12)
```

[9]:

```
df.head()
```

[9]:

```
   AveragePrice  Total_Volume  tot_ava1   tot_ava2  tot_ava3  Total_Bags  \
0          1.33      64236.62   1036.74   54454.85     48.16     8696.87
1          1.35      54876.98    674.28   44638.81     58.33     9505.56
2          0.93     118220.22    794.70  109149.67    130.50     8145.35
3          1.08      78992.15   1132.00   71976.41     72.58     5811.16
4          1.28      51039.60    941.48   43838.39     75.78     6183.95

   Small_Bags  Large_Bags  Xlarge_Bags          type  year  region
0     8603.62       93.25          0.0  conventional  2015  Albany
1     9408.07       97.49          0.0  conventional  2015  Albany
2     8042.21      103.14          0.0  conventional  2015  Albany
3     5677.40      133.76          0.0  conventional  2015  Albany
```

```
4     5986.26      197.69          0.0  conventional  2015  Albany
```

[10]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

[11]:

```
df.type=le.fit_transform(df.type)
```

[12]:

```
df.region=le.fit_transform(df.region)
```

[13]:

```
df.describe()
```

[13]:

```
       AveragePrice  Total_Volume      tot_ava1      tot_ava2      tot_ava3  \
count  18249.000000  1.824900e+04  1.824900e+04  1.824900e+04  1.824900e+04
mean       1.405978  8.506440e+05  2.930084e+05  2.951546e+05  2.283974e+04
std        0.402677  3.453545e+06  1.264989e+06  1.204120e+06  1.074641e+05
min        0.440000  8.456000e+01  0.000000e+00  0.000000e+00  0.000000e+00
25%        1.100000  1.083858e+04  8.540700e+02  3.008780e+03  0.000000e+00
50%        1.370000  1.073768e+05  8.645300e+03  2.906102e+04  1.849900e+02
75%        1.660000  4.329623e+05  1.110202e+05  1.502069e+05  6.243420e+03
max        3.250000  6.250565e+07  2.274362e+07  2.047057e+07  2.546439e+06

         Total_Bags    Small_Bags    Large_Bags    Xlarge_Bags          type  \
count  1.824900e+04  1.824900e+04  1.824900e+04   18249.000000  18249.000000
mean   2.396392e+05  1.821947e+05  5.433809e+04    3106.426507      0.499918
std    9.862424e+05  7.461785e+05  2.439660e+05   17692.894652      0.500014
min    0.000000e+00  0.000000e+00  0.000000e+00       0.000000      0.000000
25%    5.088640e+03  2.849420e+03  1.274700e+02       0.000000      0.000000
50%    3.974383e+04  2.636282e+04  2.647710e+03       0.000000      0.000000
75%    1.107834e+05  8.333767e+04  2.202925e+04     132.500000      1.000000
max    1.937313e+07  1.338459e+07  5.719097e+06  551693.650000      1.000000

               year        region
count  18249.000000  18249.000000
mean    2016.147899     26.495644
std        0.939938     15.583788
min     2015.000000      0.000000
25%     2015.000000     13.000000
50%     2016.000000     26.000000
75%     2017.000000     40.000000
max     2018.000000     53.000000
```

[75]:

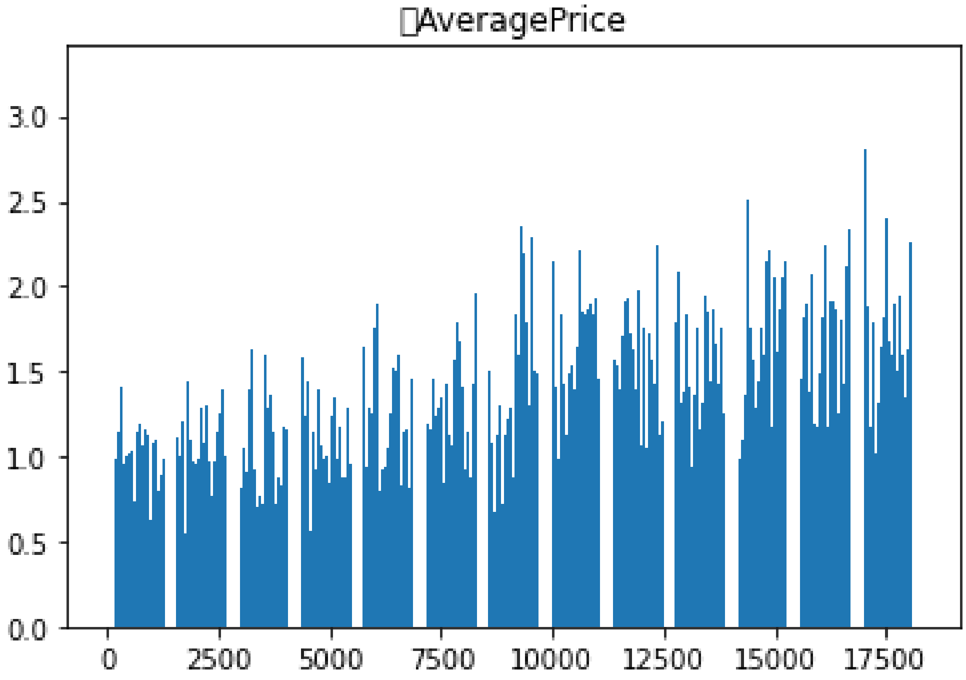
```
plt.bar(height=df.AveragePrice,x=np.arange(1,18250,1));plt.
 ↪title("        AveragePrice")
```

[75]:

```
Text(0.5, 1.0, '\tAveragePrice')
```

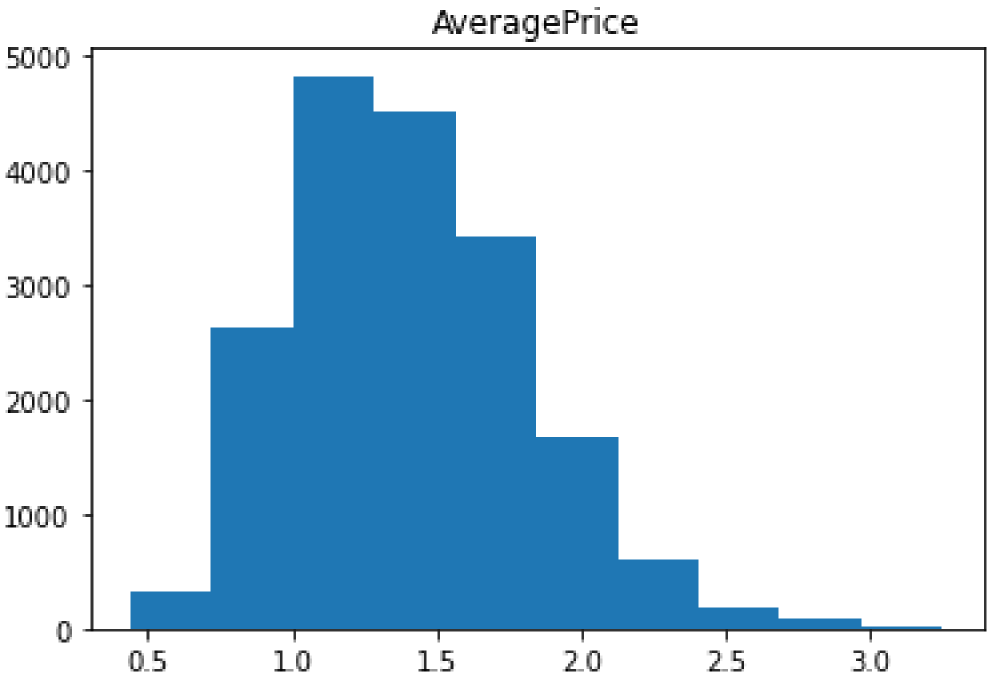
```
C:\Users\Ravi\anaconda3\lib\site-
packages\matplotlib\backends\backend_agg.py:240: RuntimeWarning: Glyph 9 missing
```

```
from current font.
  font.set_text(s, 0.0, flags=flags)
C:\Users\Ravi\anaconda3\lib\site-
packages\matplotlib\backends\backend_agg.py:203: RuntimeWarning: Glyph 9 missing
from current font.
  font.set_text(s, 0, flags=flags)
```



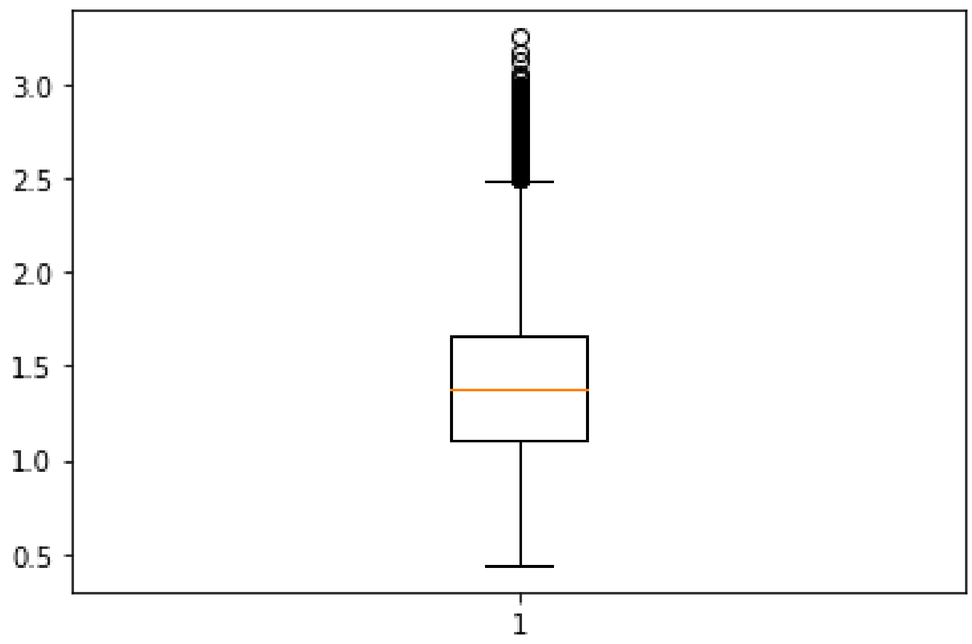
[76]:
```
plt.hist(df.AveragePrice);plt.title("AveragePrice")
```

[76]:
```
Text(0.5, 1.0, 'AveragePrice')
```



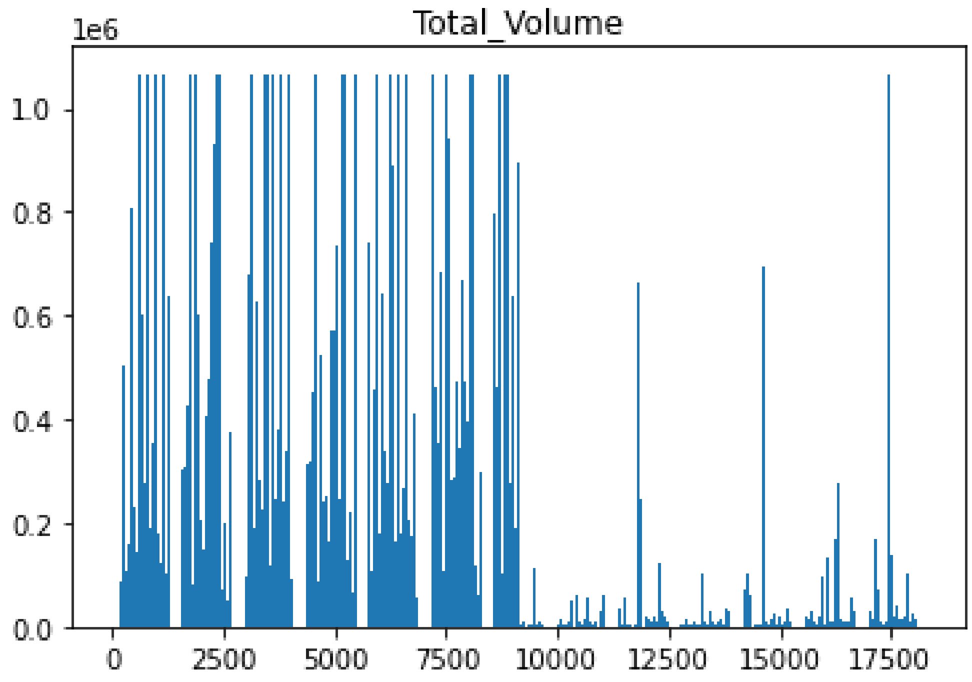
```
[145]: plt.boxplot(df.AveragePrice)
```

```
[145]: {'whiskers': [<matplotlib.lines.Line2D at 0x12e50d424f0>,
         <matplotlib.lines.Line2D at 0x12e50cb8820>],
        'caps': [<matplotlib.lines.Line2D at 0x12e50cc4760>,
         <matplotlib.lines.Line2D at 0x12e509faa60>],
        'boxes': [<matplotlib.lines.Line2D at 0x12e4641f610>],
        'medians': [<matplotlib.lines.Line2D at 0x12e50d658e0>],
        'fliers': [<matplotlib.lines.Line2D at 0x12e50d65bb0>],
        'means': []}
```



```
[77]: plt.bar(height=df.Total_Volume,x=np.arange(1,18250,1));plt.title("Total_Volume")
```

```
[77]: Text(0.5, 1.0, 'Total_Volume')
```



[146]:

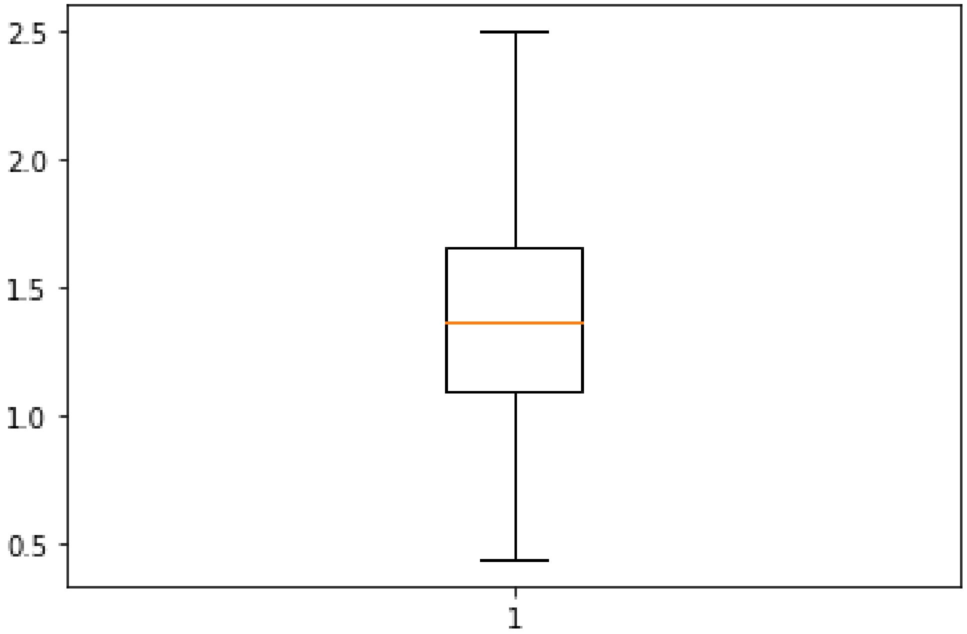
```
IQR=df.AveragePrice.quantile(0.75)-df.AveragePrice.quantile(0.25)
lower_limit=df.Total_Volume.quantile(0.25)-(IQR*1.5)
upper_limit=df.Total_Volume.quantile(0.75)-(IQR*1.5)

outliers=np.where(df.AveragePrice>upper_limit,True,np.where(df.
 ↪AveragePrice<lower_limit,True,False))
from feature_engine.outliers import Winsorizer
winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
 ↪gaussian for mean and std
                          tail='both', # cap left, right or both tails
                          fold=1.5,
                           variables=['AveragePrice'])
```

[147]:

```
df_t=winsor.fit_transform(df[["AveragePrice"]])
plt.boxplot(df_t)
```

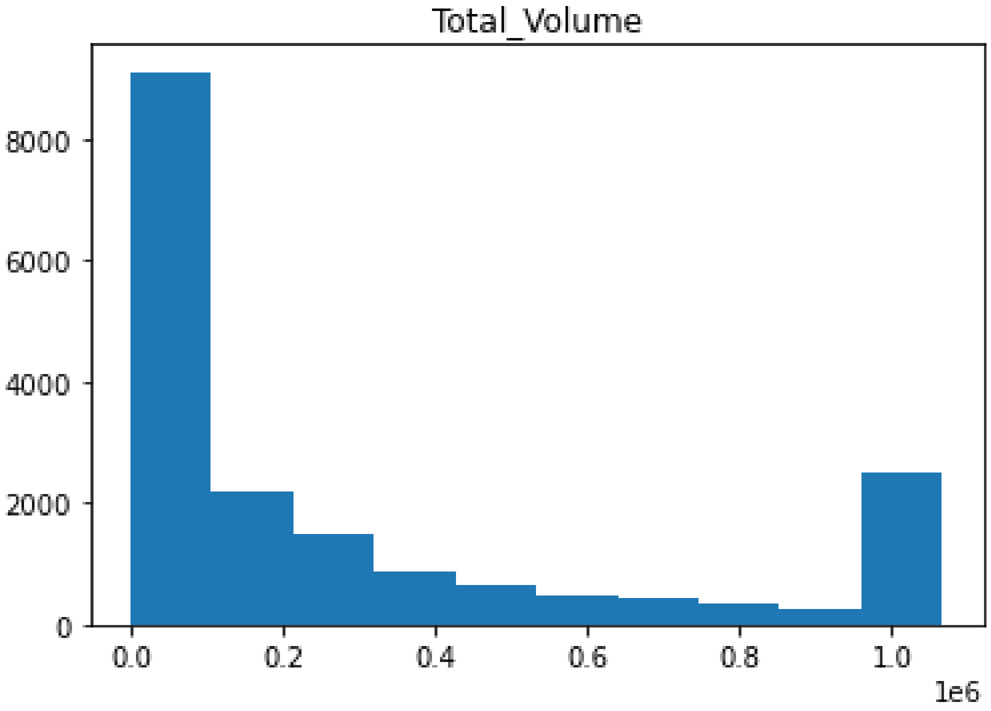
[147]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x12e50d8f820>,
  <matplotlib.lines.Line2D at 0x12e50d8fbb0>],
 'caps': [<matplotlib.lines.Line2D at 0x12e50d8ff40>,
  <matplotlib.lines.Line2D at 0x12e50dbb310>],
 'boxes': [<matplotlib.lines.Line2D at 0x12e50d8f490>],
 'medians': [<matplotlib.lines.Line2D at 0x12e50dbb6a0>],
 'fliers': [<matplotlib.lines.Line2D at 0x12e50dbba30>],
 'means': []}
```



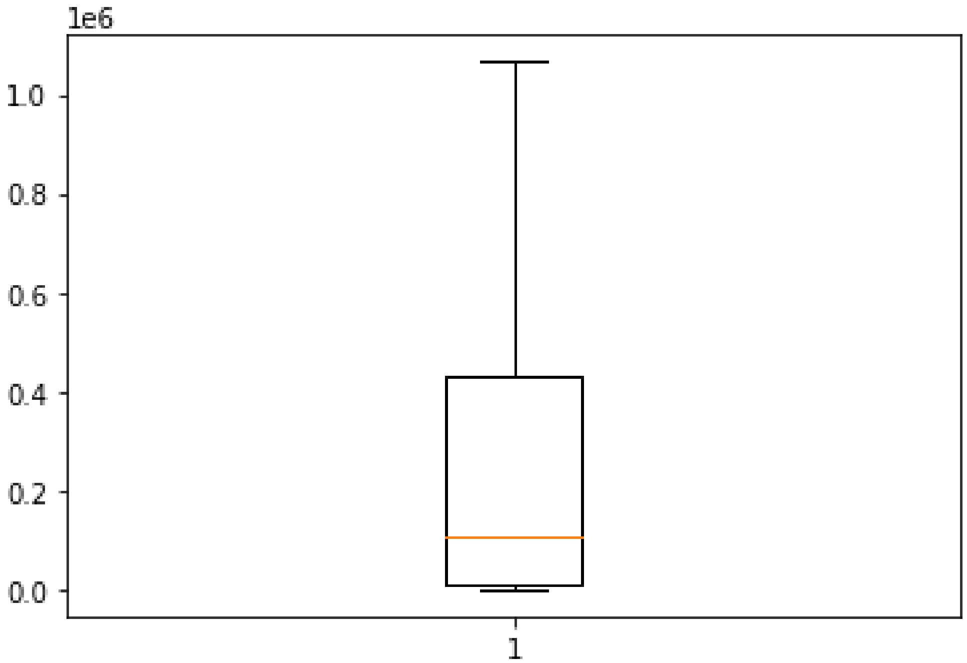
```
[78]: plt.hist(df.Total_Volume);plt.title("Total_Volume")
```

```
[78]: Text(0.5, 1.0, 'Total_Volume')
```



```
[79]: plt.boxplot(df.Total_Volume)
```

```
[79]: {'whiskers': [<matplotlib.lines.Line2D at 0x12ddc0f3130>,
        <matplotlib.lines.Line2D at 0x12ddc0f34c0>],
       'caps': [<matplotlib.lines.Line2D at 0x12ddc0f3850>,
        <matplotlib.lines.Line2D at 0x12ddc0f3be0>],
       'boxes': [<matplotlib.lines.Line2D at 0x12ddc0e6d60>],
       'medians': [<matplotlib.lines.Line2D at 0x12ddc0f3f70>],
       'fliers': [<matplotlib.lines.Line2D at 0x12ddc0fd340>],
       'means': []}
```



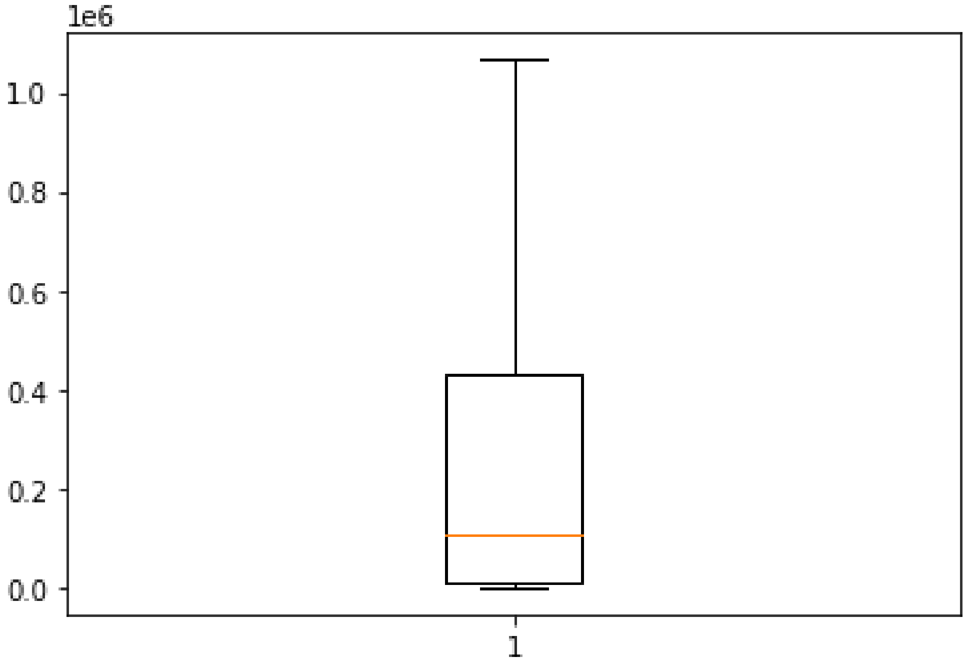
```
[80]: IQR=df.Total_Volume.quantile(0.75)-df.Total_Volume.quantile(0.25)
      lower_limit=df.Total_Volume.quantile(0.25)-(IQR*1.5)
      upper_limit=df.Total_Volume.quantile(0.75)-(IQR*1.5)
```

```
[81]: outliers=np.where(df.Total_Volume>upper_limit,True,np.where(df.
       ↪Total_Volume<lower_limit,True,False))
      from feature_engine.outliers import Winsorizer
      winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
       ↪gaussian for mean and std
                                tail='both', # cap left, right or both tails
                                fold=1.5,
                                 variables=['Total_Volume'])
```

```
[82]: df_t=winsor.fit_transform(df[["Total_Volume"]])
```

```
[83]: plt.boxplot(df_t)
```

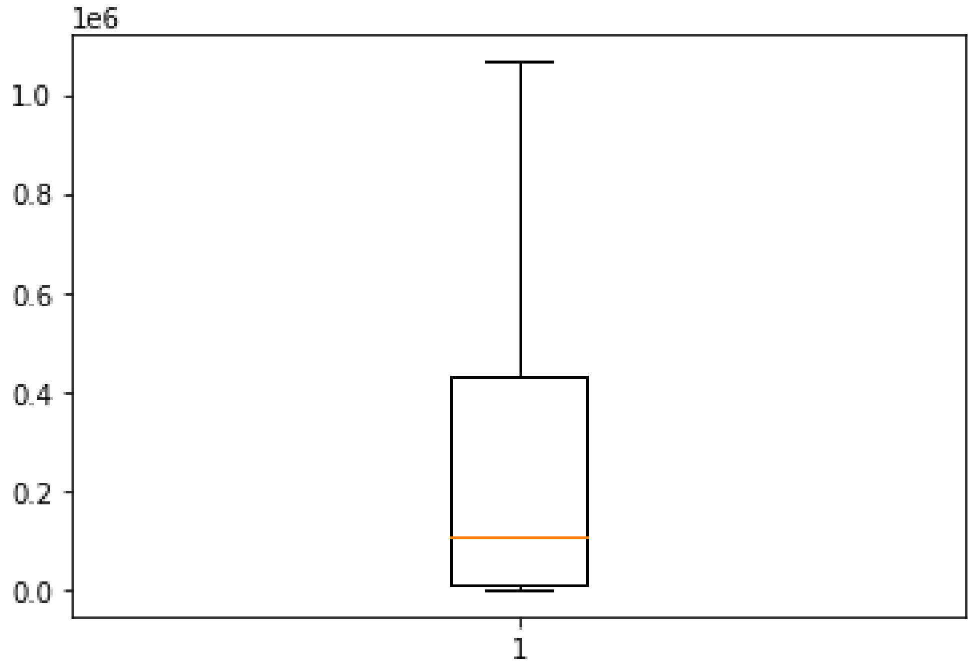
```
[83]: {'whiskers': [<matplotlib.lines.Line2D at 0x12ddc15c190>,
        <matplotlib.lines.Line2D at 0x12ddc15c520>],
       'caps': [<matplotlib.lines.Line2D at 0x12ddc15c8b0>,
        <matplotlib.lines.Line2D at 0x12ddc15cc40>],
       'boxes': [<matplotlib.lines.Line2D at 0x12ddc14cdc0>],
       'medians': [<matplotlib.lines.Line2D at 0x12ddc15cfd0>],
       'fliers': [<matplotlib.lines.Line2D at 0x12ddc1653a0>],
       'means': []}
```



```
[84]: df.Total_Volume=df_t
```

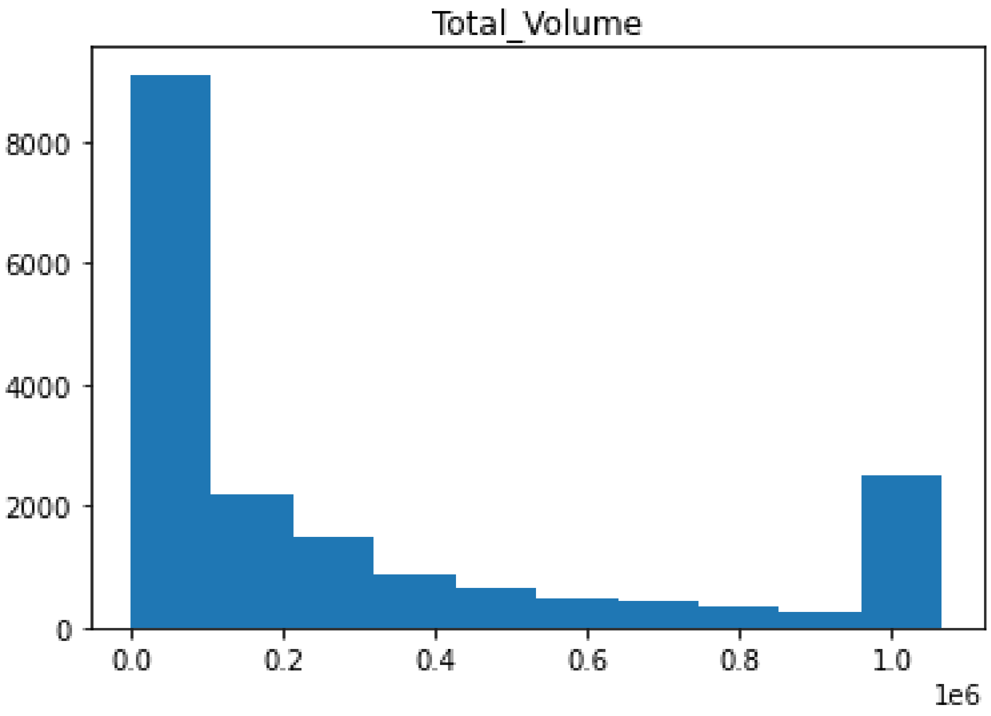
```
[85]: plt.boxplot(df.Total_Volume)
```

```
[85]: {'whiskers': [<matplotlib.lines.Line2D at 0x12ddc1c6100>,
        <matplotlib.lines.Line2D at 0x12ddc1c6490>],
       'caps': [<matplotlib.lines.Line2D at 0x12ddc1c6820>,
        <matplotlib.lines.Line2D at 0x12ddc1c6bb0>],
       'boxes': [<matplotlib.lines.Line2D at 0x12ddc1b7d30>],
       'medians': [<matplotlib.lines.Line2D at 0x12ddc1c6f40>],
       'fliers': [<matplotlib.lines.Line2D at 0x12ddc1d0310>],
       'means': []}
```



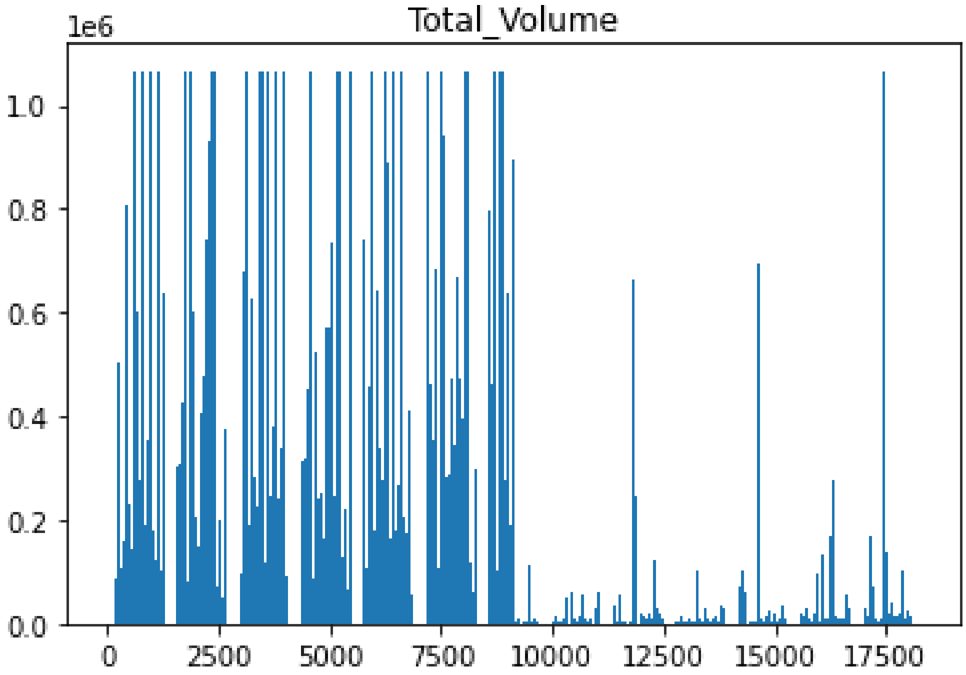
```
[86]: plt.hist(df.Total_Volume);plt.title("Total_Volume")
```

```
[86]: Text(0.5, 1.0, 'Total_Volume')
```



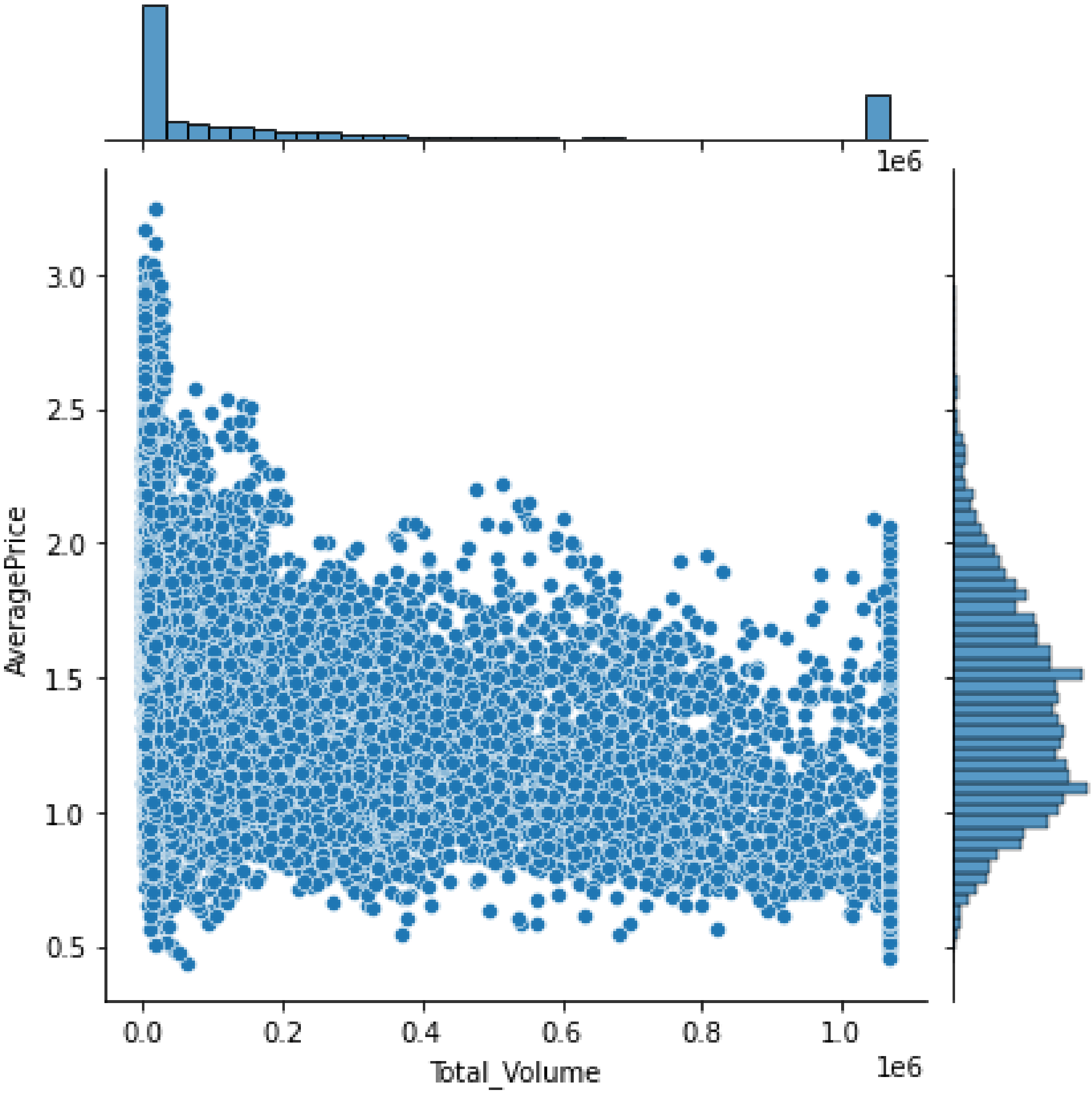
```
[87]: plt.bar(height=df.Total_Volume,x=np.arange(1,18250,1));plt.title("Total_Volume")
```

```
[87]: Text(0.5, 1.0, 'Total_Volume')
```



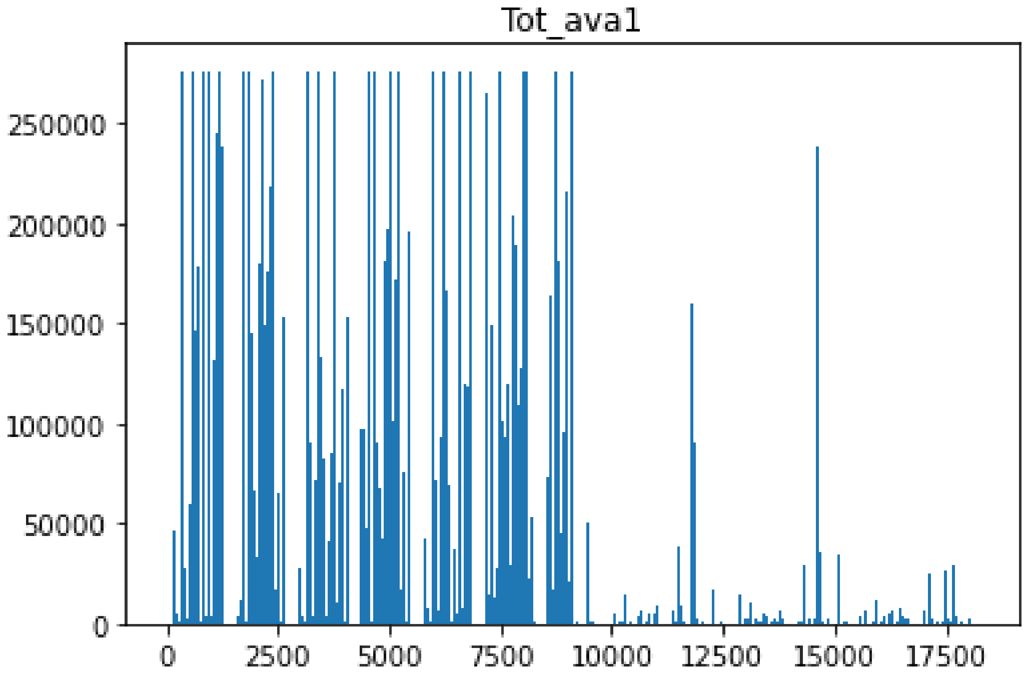
```
[88]: sns.jointplot(x=df.Total_Volume,y=df.AveragePrice)
```

```
[88]: <seaborn.axisgrid.JointGrid at 0x12ddc24a160>
```



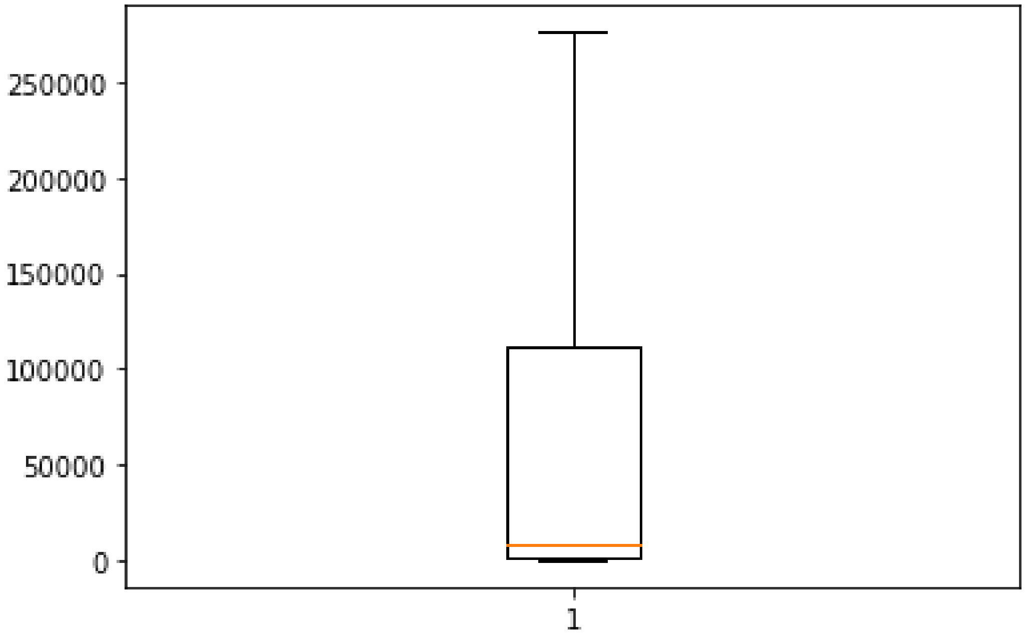
```
[89]: plt.bar(height=df.tot_ava1,x=np.arange(1,18250,1));plt.title("Tot_ava1")
```

```
[89]: Text(0.5, 1.0, 'Tot_ava1')
```



[90]:
```
plt.boxplot(df.tot_ava1)
```

[90]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x12df70b8a30>,
  <matplotlib.lines.Line2D at 0x12df70b8dc0>],
 'caps': [<matplotlib.lines.Line2D at 0x12df70c7190>,
  <matplotlib.lines.Line2D at 0x12df70c7520>],
 'boxes': [<matplotlib.lines.Line2D at 0x12df70b86a0>],
 'medians': [<matplotlib.lines.Line2D at 0x12df70c78b0>],
 'fliers': [<matplotlib.lines.Line2D at 0x12df70c7c40>],
 'means': []}
```



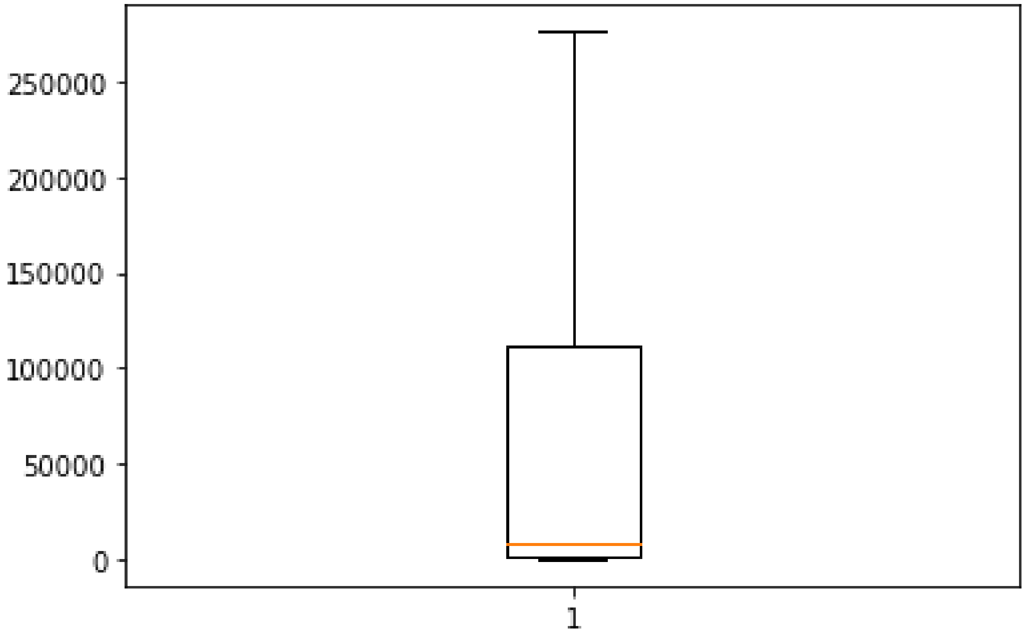
```
[91]: IQR=df.tot_ava1.quantile(0.75)-df.tot_ava1.quantile(0.25)
      lower_limit=df.tot_ava1.quantile(0.25)-(IQR*1.5)
      upper_limit=df.tot_ava1.quantile(0.75)-(IQR*1.5)


      outliers=np.where(df.tot_ava1>upper_limit,True,np.where(df.
       ↪tot_ava1<lower_limit,True,False))
      from feature_engine.outliers import Winsorizer
      winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
       ↪gaussian for mean and std
                                tail='both', # cap left, right or both tails
                                fold=1.5,
                                 variables=['tot_ava1'])
```

```
[92]: df_t=winsor.fit_transform(df[["tot_ava1"]])
```

```
[93]: df.tot_ava1=df_t
```

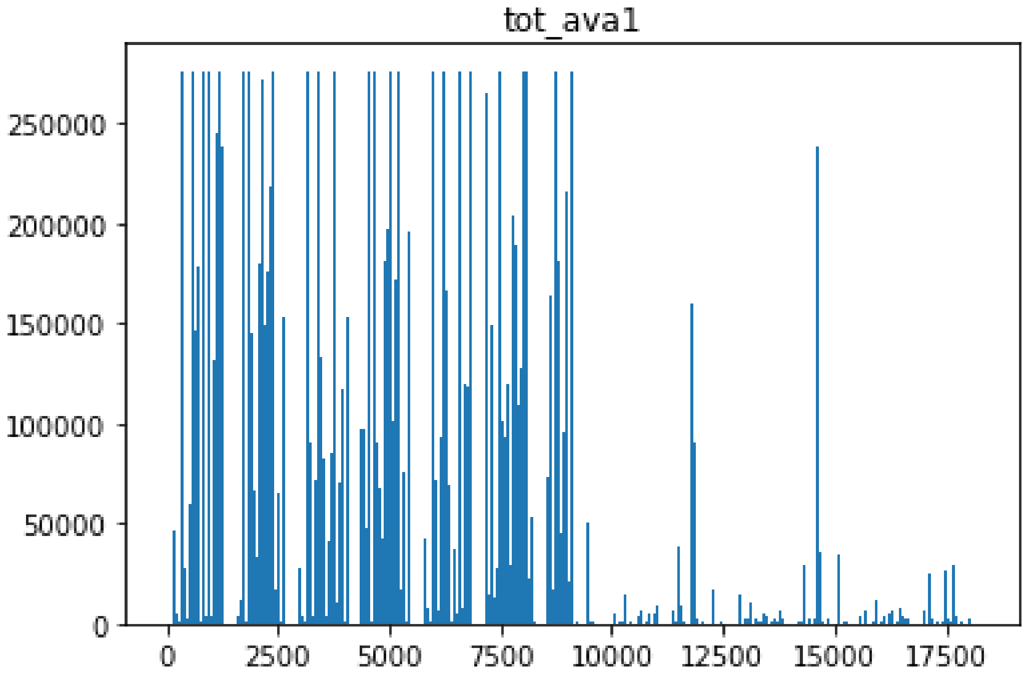
```
[94]: plt.boxplot(df.tot_ava1)
```

```
[94]: {'whiskers': [<matplotlib.lines.Line2D at 0x12df6723760>,
        <matplotlib.lines.Line2D at 0x12df6723af0>],
       'caps': [<matplotlib.lines.Line2D at 0x12df6723e80>,
        <matplotlib.lines.Line2D at 0x12df672d250>],
       'boxes': [<matplotlib.lines.Line2D at 0x12df67233d0>],
       'medians': [<matplotlib.lines.Line2D at 0x12df672d5e0>],
       'fliers': [<matplotlib.lines.Line2D at 0x12df672d970>],
       'means': []}
```



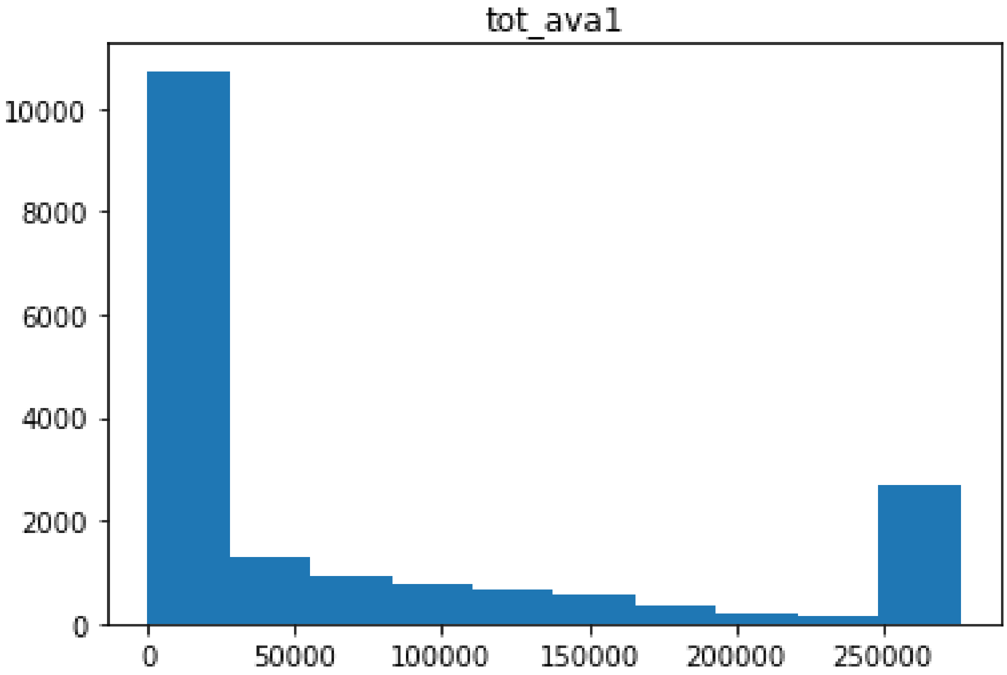
[95]:
```
plt.bar(height=df.tot_ava1,x=np.arange(1,18250,1));plt.title("tot_ava1")
```

[95]:
```
Text(0.5, 1.0, 'tot_ava1')
```



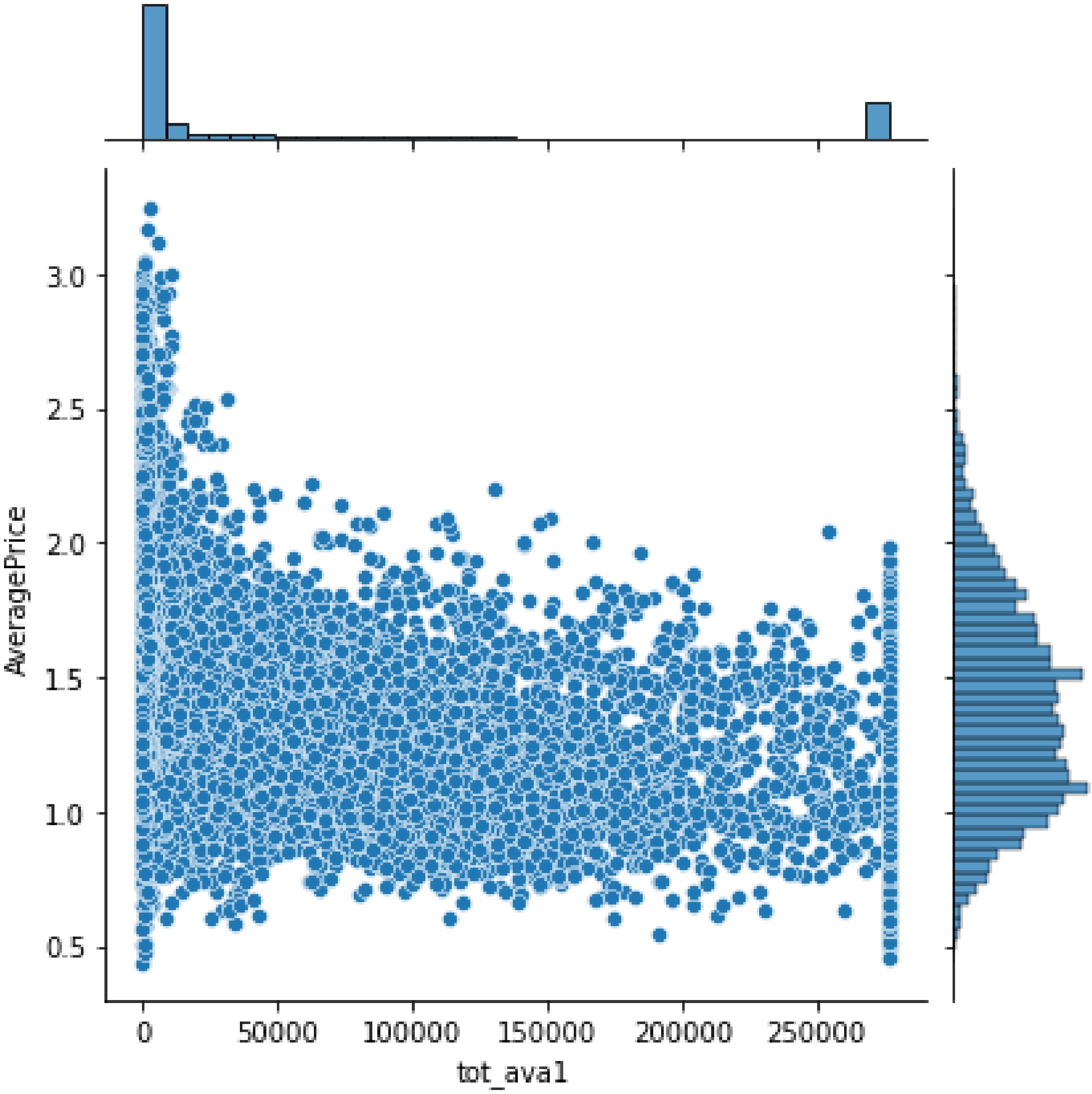
```
[96]: plt.hist(df.tot_ava1);plt.title("tot_ava1")
```

```
[96]: Text(0.5, 1.0, 'tot_ava1')
```



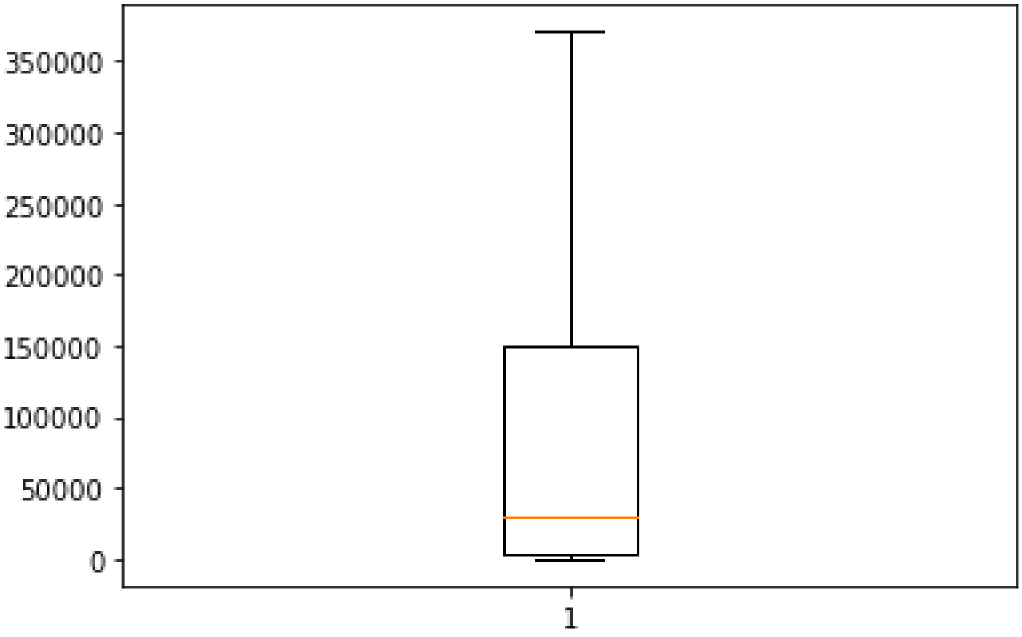
[97]:
```
sns.jointplot(x=df.tot_ava1,y=df.AveragePrice)
```

[97]:
```
<seaborn.axisgrid.JointGrid at 0x12de9557040>
```



[98]:
```
plt.boxplot(df.tot_ava2)
```

[98]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x12e03b13a90>,
  <matplotlib.lines.Line2D at 0x12e03b13e20>],
 'caps': [<matplotlib.lines.Line2D at 0x12e03b221f0>,
  <matplotlib.lines.Line2D at 0x12e03b22580>],
 'boxes': [<matplotlib.lines.Line2D at 0x12e03b13700>],
 'medians': [<matplotlib.lines.Line2D at 0x12e03b22910>],
 'fliers': [<matplotlib.lines.Line2D at 0x12e03b22ca0>],
 'means': []}
```



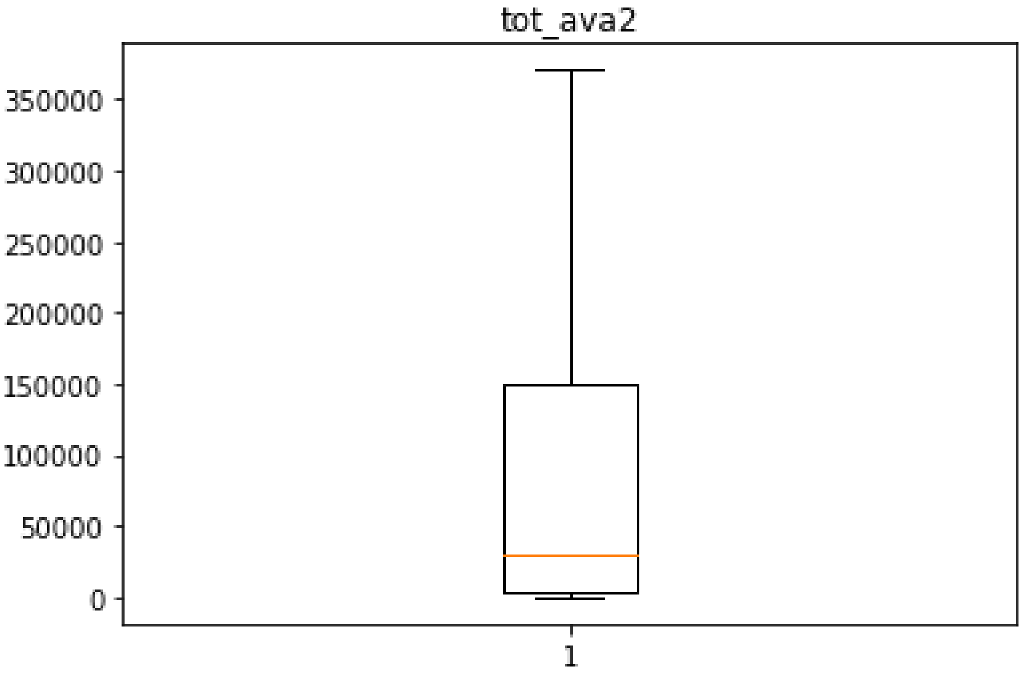
```
[99]: IQR=df.tot_ava2.quantile(0.75)-df.tot_ava2.quantile(0.25)
lower_limit=df.tot_ava2.quantile(0.25)-(IQR*1.5)
upper_limit=df.tot_ava2.quantile(0.75)-(IQR*1.5)


outliers=np.where(df.tot_ava2>upper_limit,True,np.where(df.
 ↪tot_ava2<lower_limit,True,False))
from feature_engine.outliers import Winsorizer
winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
 ↪gaussian for mean and std
                          tail='both', # cap left, right or both tails
                          fold=1.5,
                           variables=['tot_ava2'])
```

```
[100]: df_t=winsor.fit_transform(df[["tot_ava2"]])
```

```
[101]: df.tot_ava2=df_t
```

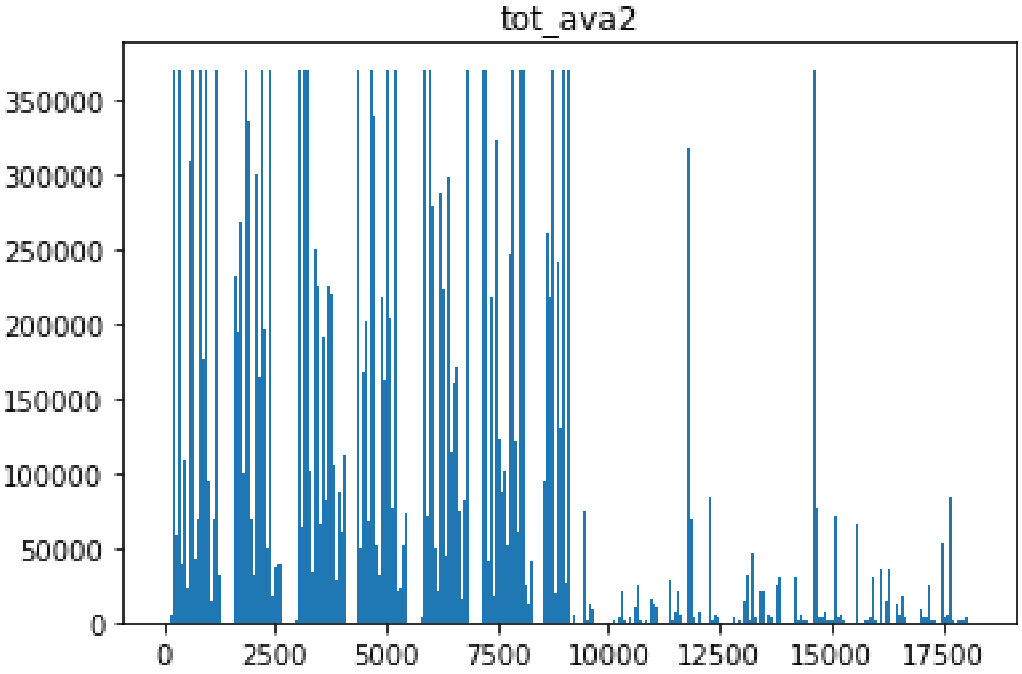
```
[102]: plt.boxplot(df.tot_ava2);plt.title("tot_ava2")
```

```
[102]: Text(0.5, 1.0, 'tot_ava2')
```



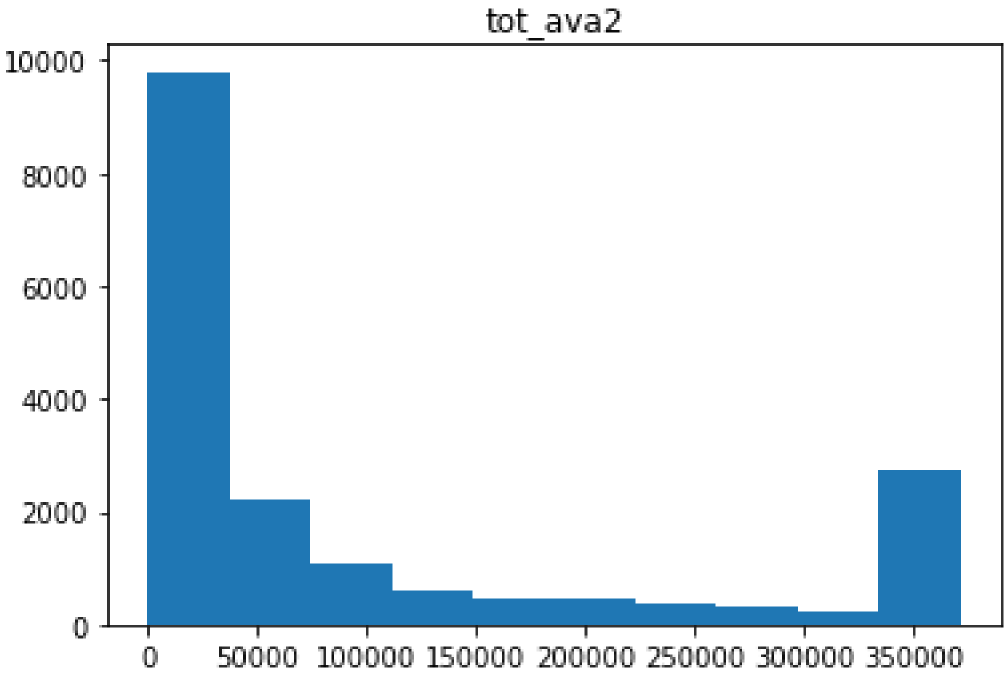
```
[103]: plt.bar(height=df.tot_ava2,x=np.arange(1,18250,1));plt.title("tot_ava2")
```

```
[103]: Text(0.5, 1.0, 'tot_ava2')
```



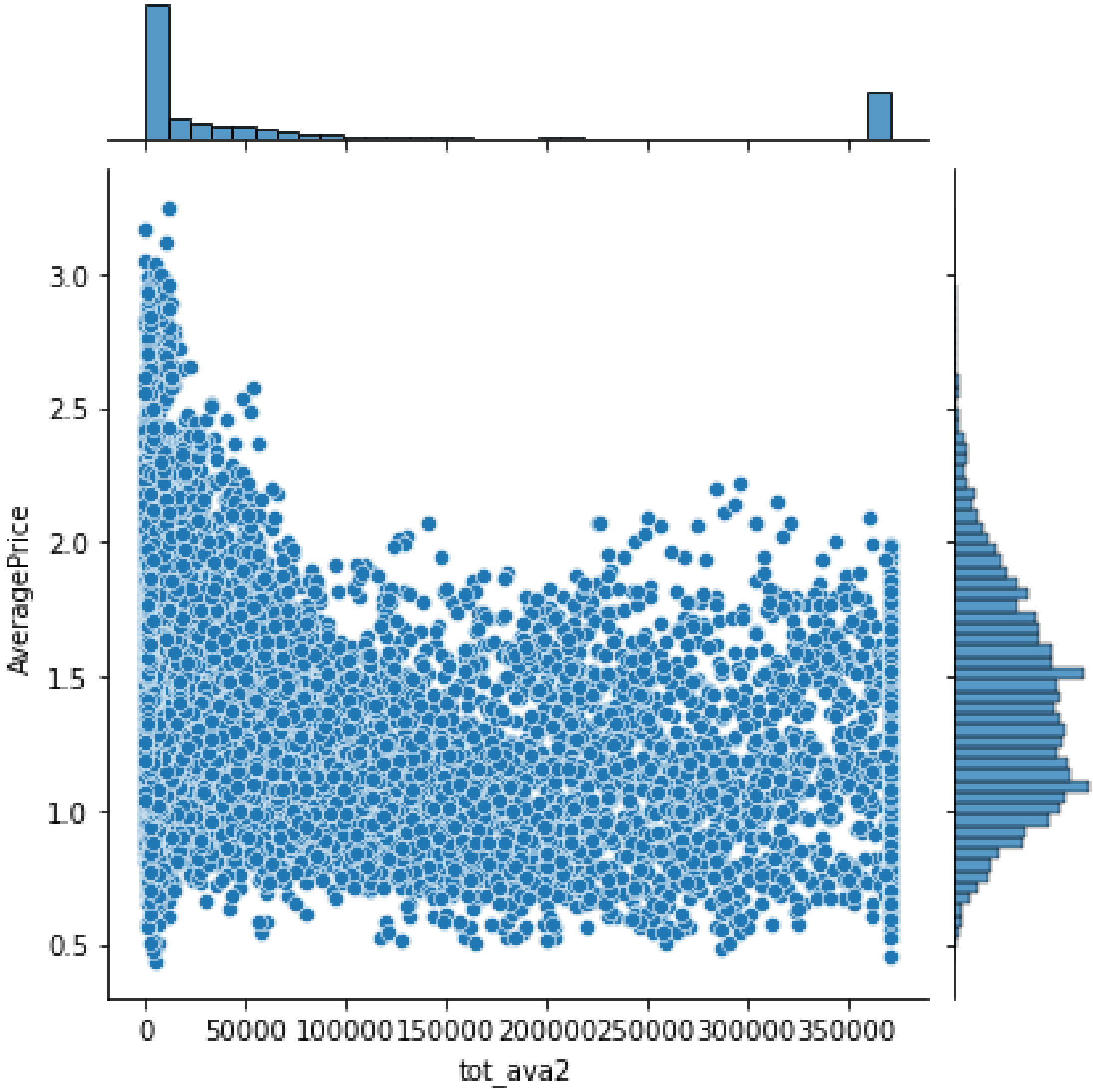
```
[104]: plt.hist(df.tot_ava2);plt.title("tot_ava2")
```

```
[104]: Text(0.5, 1.0, 'tot_ava2')
```



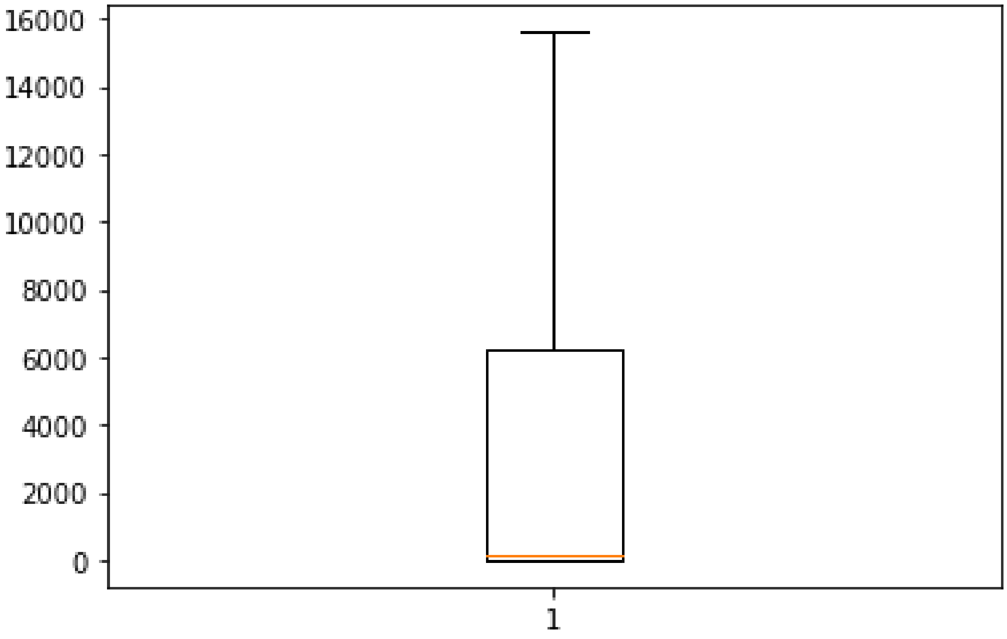
```
[105]: sns.jointplot(x=df.tot_ava2,y=df.AveragePrice)
```

```
[105]: <seaborn.axisgrid.JointGrid at 0x12e03b37850>
```



```
[106]: plt.boxplot(df.tot_ava3)
```

```
[106]: {'whiskers': [<matplotlib.lines.Line2D at 0x12e10f91e80>,
         <matplotlib.lines.Line2D at 0x12e10fa0250>],
        'caps': [<matplotlib.lines.Line2D at 0x12e10fa05e0>,
         <matplotlib.lines.Line2D at 0x12e10fa0970>],
        'boxes': [<matplotlib.lines.Line2D at 0x12e10f91af0>],
        'medians': [<matplotlib.lines.Line2D at 0x12e10fa0d00>],
        'fliers': [<matplotlib.lines.Line2D at 0x12e10fad0d0>],
        'means': []}
```



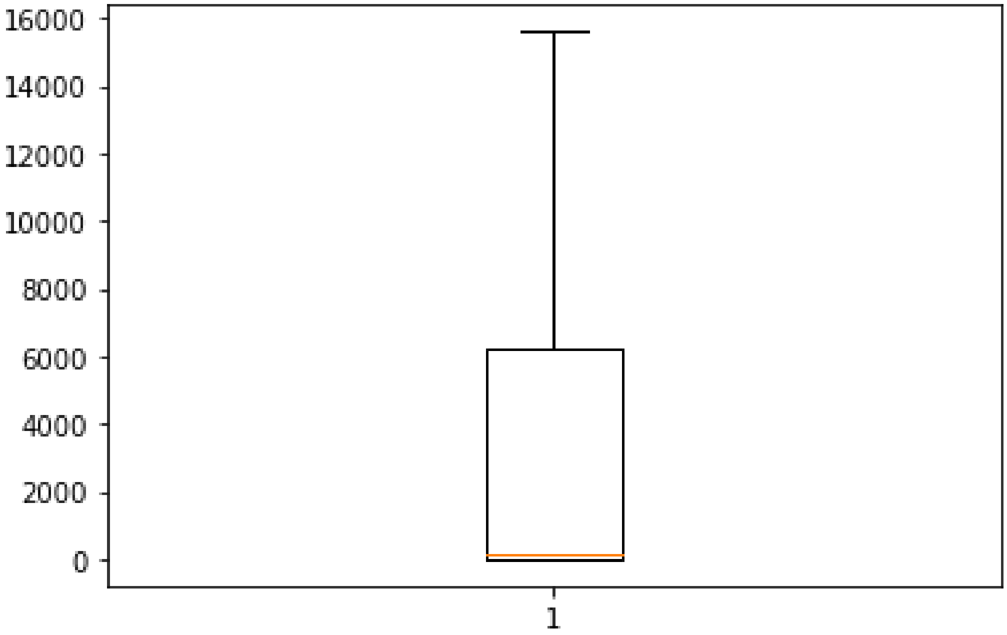
[107]:
```
IQR=df.tot_ava3.quantile(0.75)-df.tot_ava3.quantile(0.25)
lower_limit=df.tot_ava3.quantile(0.25)-(IQR*1.5)
upper_limit=df.tot_ava3.quantile(0.75)-(IQR*1.5)


outliers=np.where(df.tot_ava3>upper_limit,True,np.where(df.
 ↪tot_ava3<lower_limit,True,False))
from feature_engine.outliers import Winsorizer
winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
 ↪gaussian for mean and std
                          tail='both', # cap left, right or both tails
                          fold=1.5,
                           variables=['tot_ava3'])
```

[108]:
```
df_t=winsor.fit_transform(df[["tot_ava3"]])
```

[109]:
```
df.tot_ava3=df_t
```

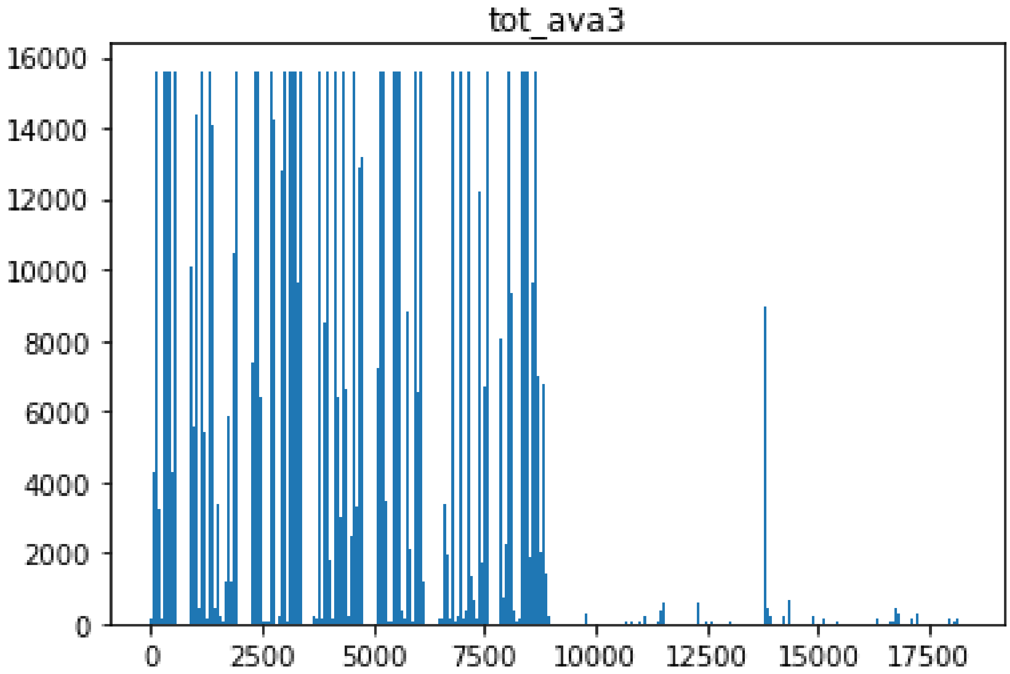
[110]:
```
plt.boxplot(df.tot_ava3)
```

[110]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x12e11003c10>,
  <matplotlib.lines.Line2D at 0x12e11003fa0>],
 'caps': [<matplotlib.lines.Line2D at 0x12e11010370>,
  <matplotlib.lines.Line2D at 0x12e11010700>],
 'boxes': [<matplotlib.lines.Line2D at 0x12e11003880>],
 'medians': [<matplotlib.lines.Line2D at 0x12e11010a90>],
 'fliers': [<matplotlib.lines.Line2D at 0x12e11010e20>],
 'means': []}
```



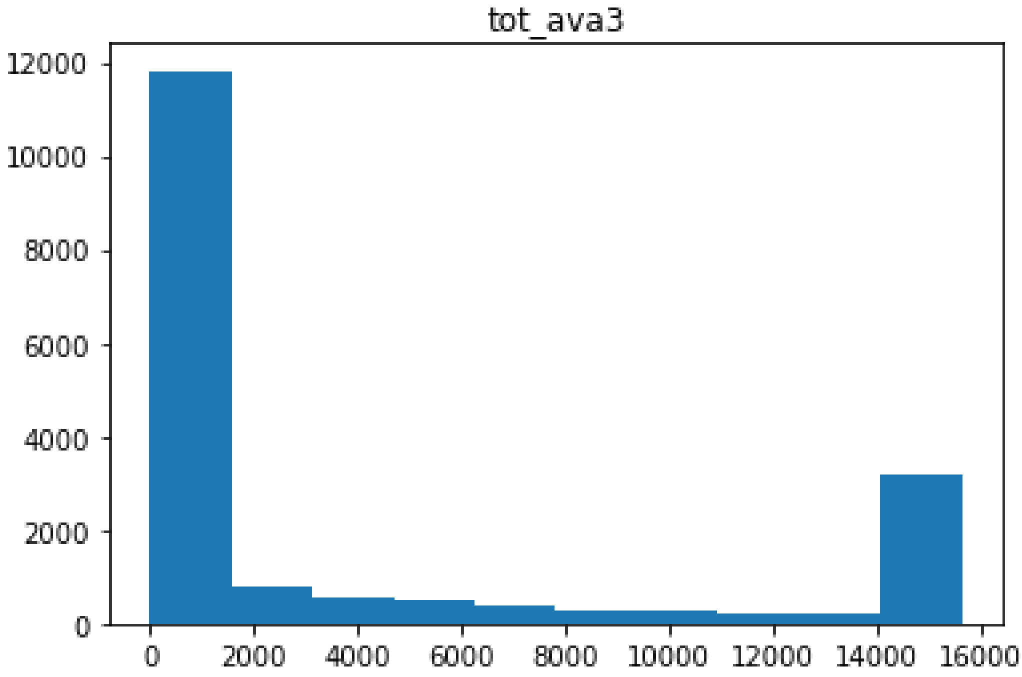
```
[111]: plt.bar(height=df.tot_ava3,x=np.arange(1,18250,1));plt.title("tot_ava3")
```

```
[111]: Text(0.5, 1.0, 'tot_ava3')
```



```
[112]: plt.hist(df.tot_ava3);plt.title("tot_ava3")
```

```
[112]: Text(0.5, 1.0, 'tot_ava3')
```

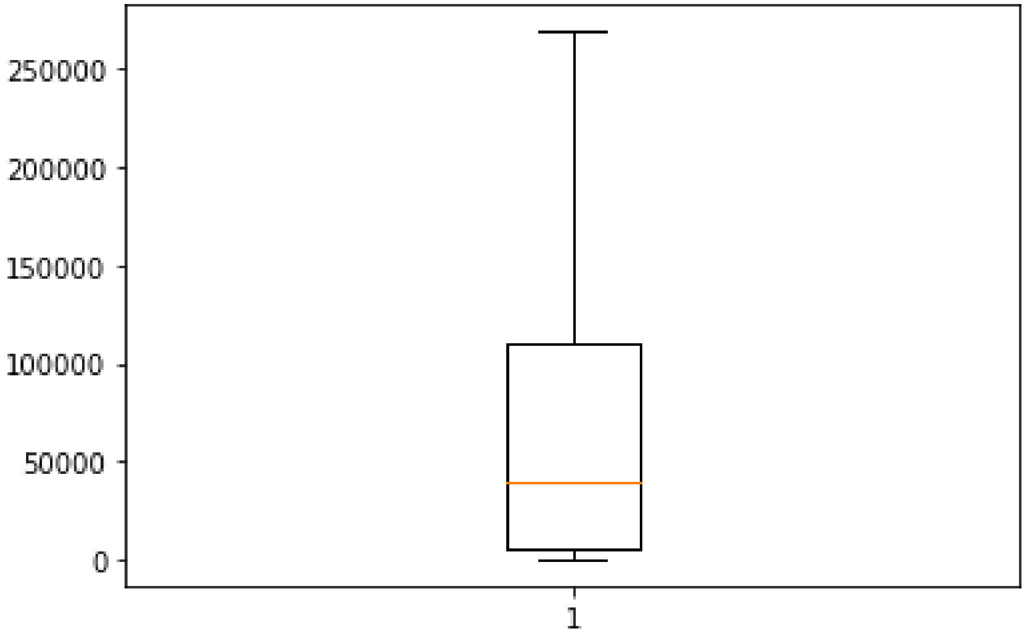


[113]:

```
plt.boxplot(df.Total_Bags)
```

[113]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x12e1ec4abe0>,
  <matplotlib.lines.Line2D at 0x12e1ec4af70>],
 'caps': [<matplotlib.lines.Line2D at 0x12e1e259340>,
  <matplotlib.lines.Line2D at 0x12e1e2596d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x12e1ec4a820>],
 'medians': [<matplotlib.lines.Line2D at 0x12e1e259a60>],
 'fliers': [<matplotlib.lines.Line2D at 0x12e1e259e20>],
 'means': []}
```



[114]:

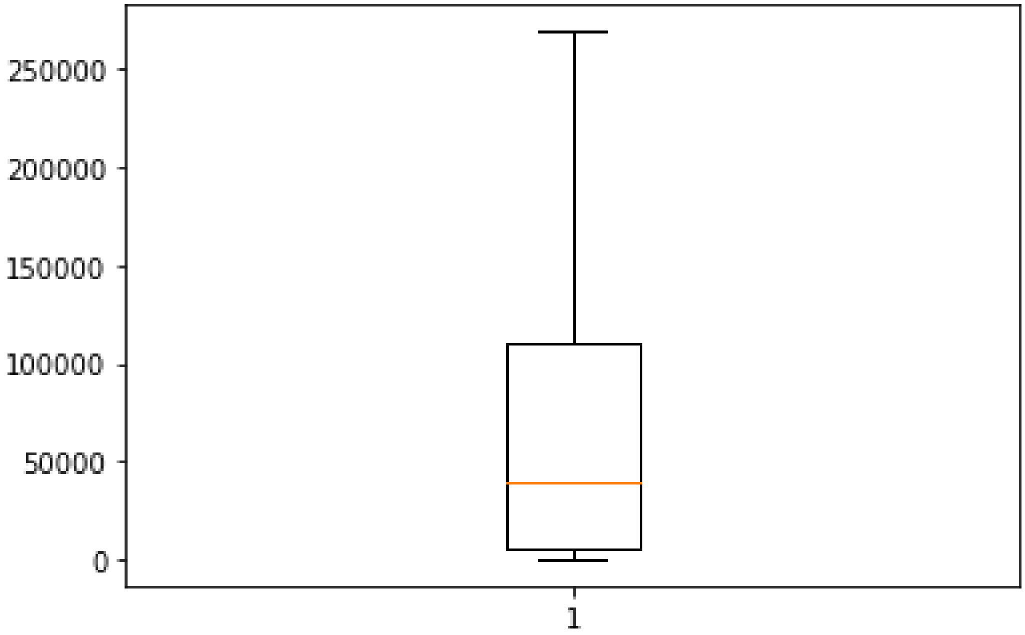
```
IQR=df.Total_Bags.quantile(0.75)-df.Total_Bags.quantile(0.25)
lower_limit=df.Total_Bags.quantile(0.25)-(IQR*1.5)
upper_limit=df.Total_Bags.quantile(0.75)-(IQR*1.5)


outliers=np.where(df.Total_Bags>upper_limit,True,np.where(df.
 ↪Total_Bags<lower_limit,True,False))
from feature_engine.outliers import Winsorizer
winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
 ↪gaussian for mean and std
                          tail='both', # cap left, right or both tails
                          fold=1.5,
                           variables=['Total_Bags'])
```

```
[115]: df_t=winsor.fit_transform(df[["Total_Bags"]])
```

```
[116]: df.Total_Bags=df_t
```

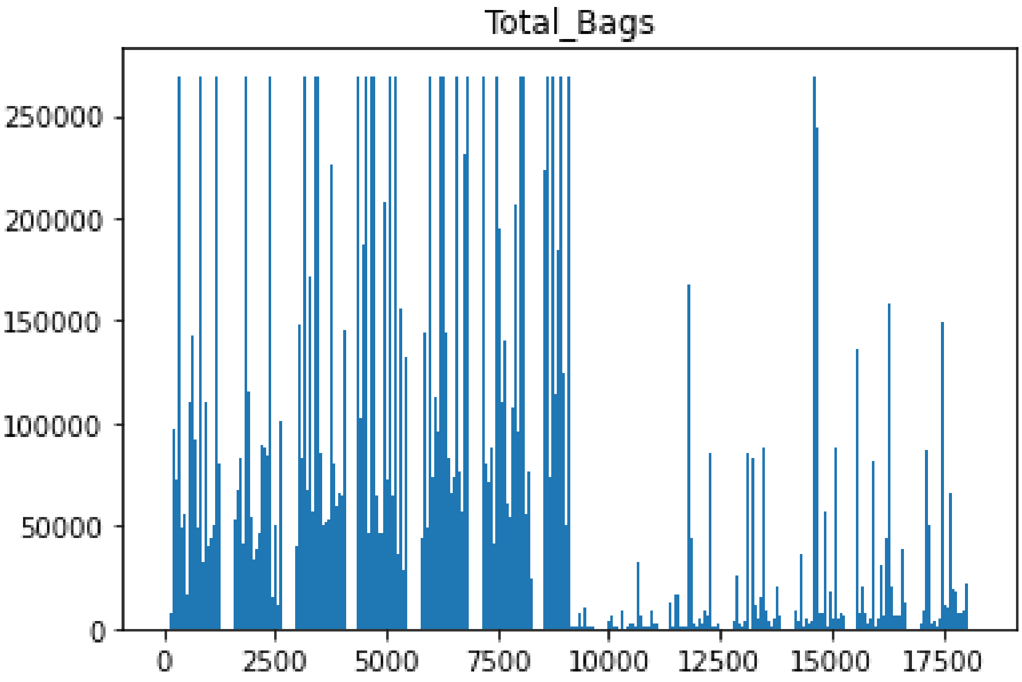
```
[117]: plt.boxplot(df.Total_Bags)
```

```
[117]: {'whiskers': [<matplotlib.lines.Line2D at 0x12e1e2b4910>,
         <matplotlib.lines.Line2D at 0x12e1e2b4ca0>],
        'caps': [<matplotlib.lines.Line2D at 0x12e1e2c2070>,
         <matplotlib.lines.Line2D at 0x12e1e2c2400>],
        'boxes': [<matplotlib.lines.Line2D at 0x12e1e2b4580>],
        'medians': [<matplotlib.lines.Line2D at 0x12e1e2c2790>],
        'fliers': [<matplotlib.lines.Line2D at 0x12e1e2c2b20>],
        'means': []}
```



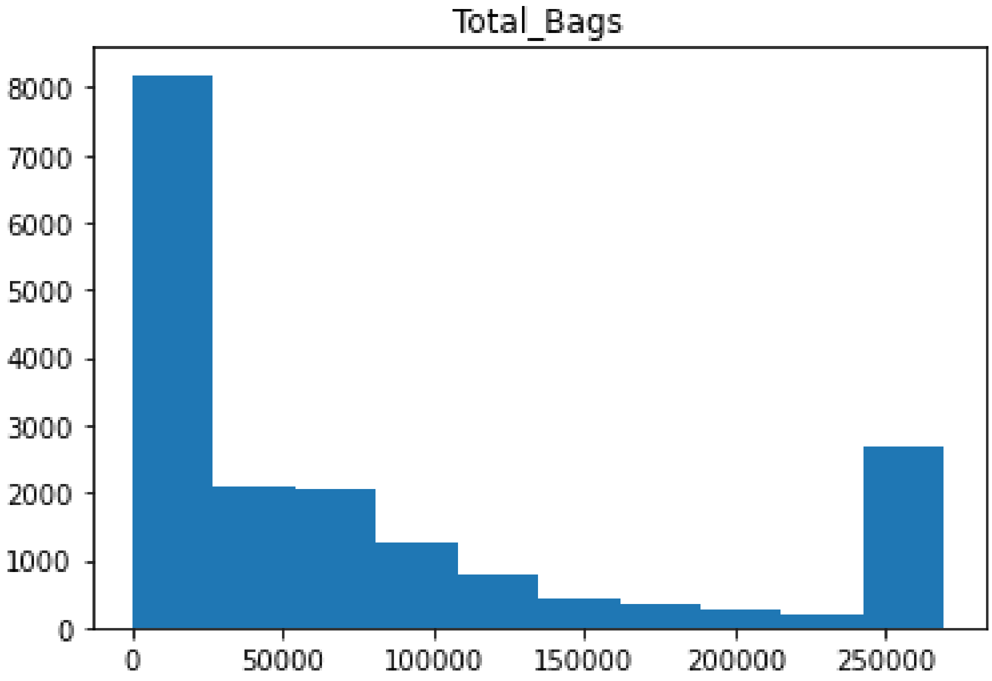
```
[118]: plt.bar(height=df.Total_Bags,x=np.arange(1,18250,1));plt.title("Total_Bags")
```

```
[118]: Text(0.5, 1.0, 'Total_Bags')
```



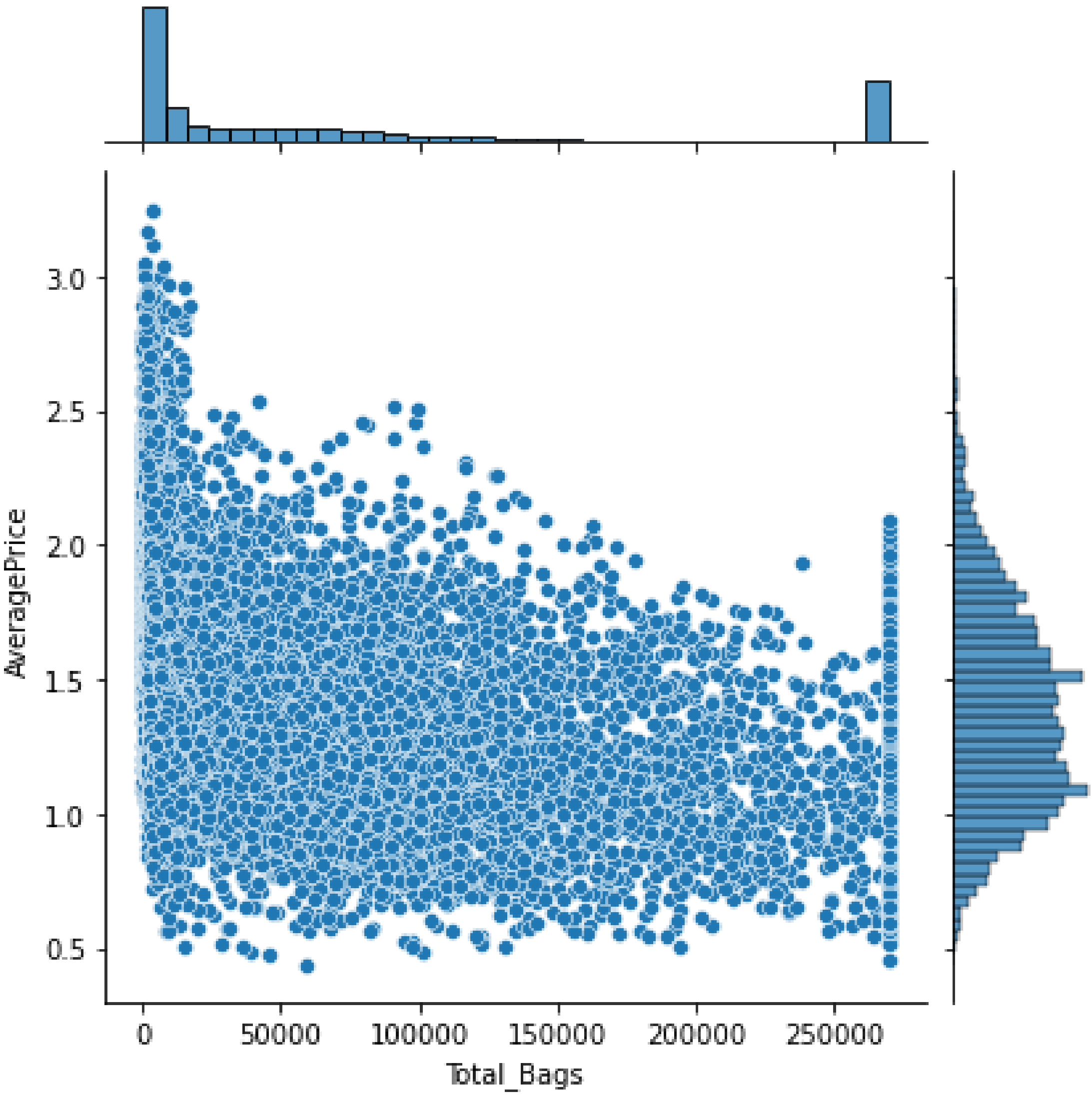
```
[119]: plt.hist(df.Total_Bags);plt.title("Total_Bags")
```

```
[119]: Text(0.5, 1.0, 'Total_Bags')
```



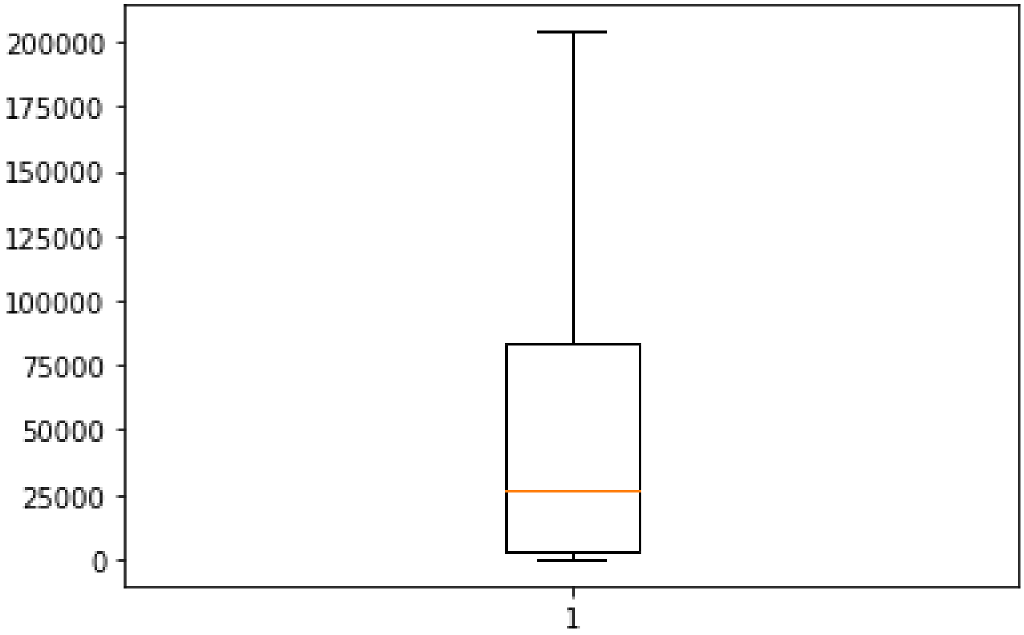
```
[120]: sns.jointplot(x=df.Total_Bags,y=df.AveragePrice)
```

```
[120]: <seaborn.axisgrid.JointGrid at 0x12e2b4b32b0>
```



```
[121]: plt.boxplot(df.Small_Bags)
```

```
[121]: {'whiskers': [<matplotlib.lines.Line2D at 0x12e2b698940>,
          <matplotlib.lines.Line2D at 0x12e2b698cd0>],
         'caps': [<matplotlib.lines.Line2D at 0x12e2b6a60a0>,
          <matplotlib.lines.Line2D at 0x12e2b6a6430>],
         'boxes': [<matplotlib.lines.Line2D at 0x12e2b6985b0>],
         'medians': [<matplotlib.lines.Line2D at 0x12e2b6a67c0>],
         'fliers': [<matplotlib.lines.Line2D at 0x12e2b6a6b50>],
         'means': []}
```



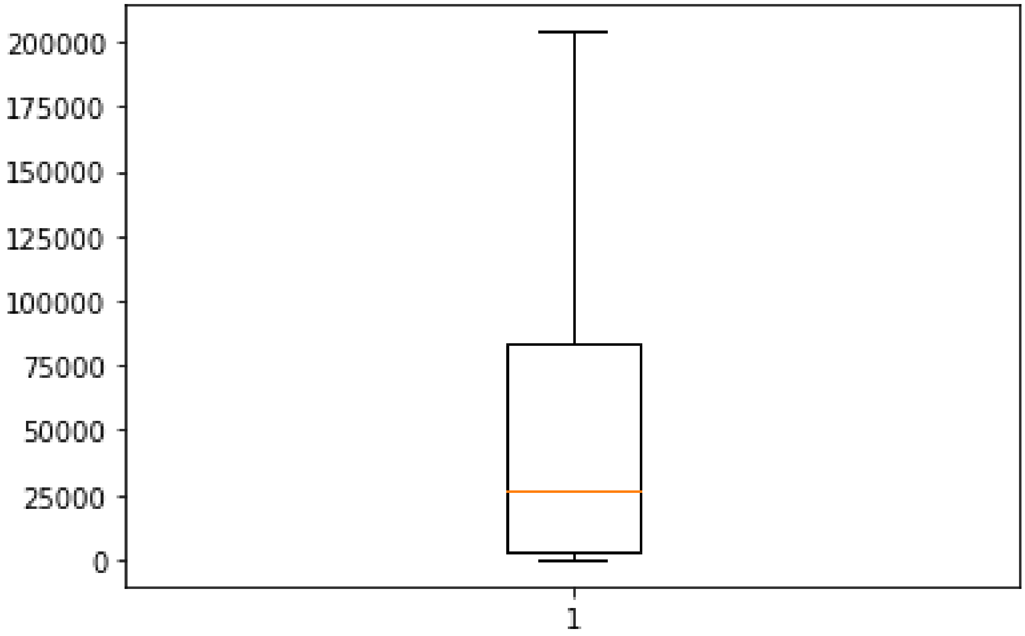
[122]:
```
IQR=df.Small_Bags.quantile(0.75)-df.Small_Bags.quantile(0.25)
lower_limit=df.Small_Bags.quantile(0.25)-(IQR*1.5)
upper_limit=df.Small_Bags.quantile(0.75)-(IQR*1.5)


outliers=np.where(df.Small_Bags>upper_limit,True,np.where(df.
 ↪Small_Bags<lower_limit,True,False))
from feature_engine.outliers import Winsorizer
winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
 ↪gaussian for mean and std
                          tail='both', # cap left, right or both tails
                          fold=1.5,
                           variables=['Small_Bags'])
```

[123]:
```
df_t=winsor.fit_transform(df[["Small_Bags"]])
```

[124]:
```
df.Small_Bags=df_t
```

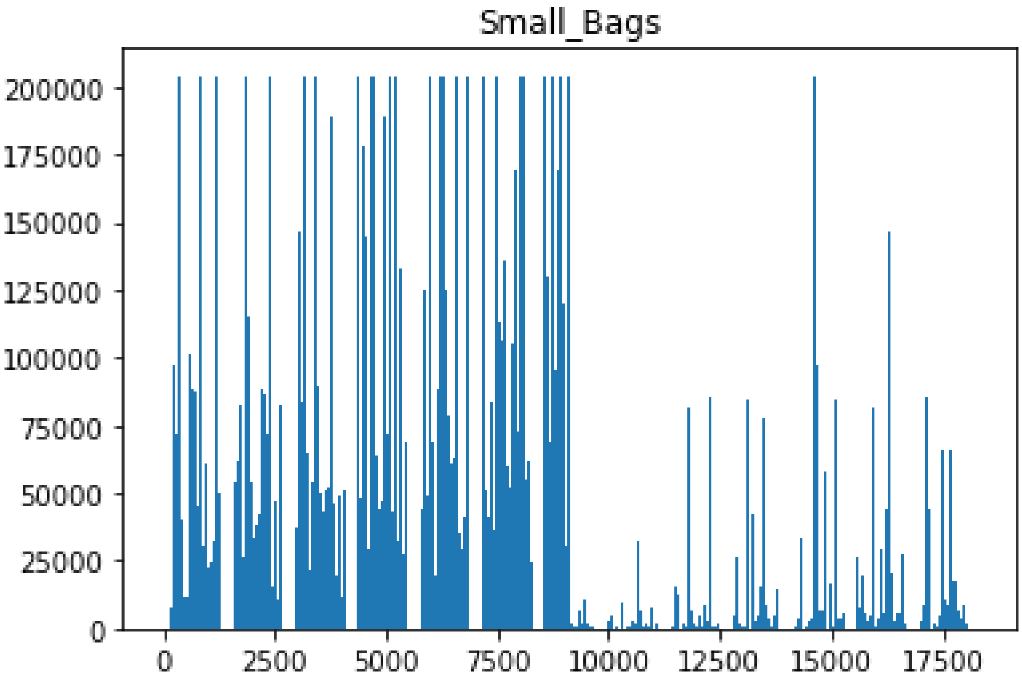
[125]:
```
plt.boxplot(df.Small_Bags)
```

```
[125]: {'whiskers': [<matplotlib.lines.Line2D at 0x12e2b707820>,
         <matplotlib.lines.Line2D at 0x12e2b707bb0>],
        'caps': [<matplotlib.lines.Line2D at 0x12e2b707f40>,
         <matplotlib.lines.Line2D at 0x12e2b712310>],
        'boxes': [<matplotlib.lines.Line2D at 0x12e2b707490>],
        'medians': [<matplotlib.lines.Line2D at 0x12e2b7126a0>],
        'fliers': [<matplotlib.lines.Line2D at 0x12e2b712a30>],
        'means': []}
```



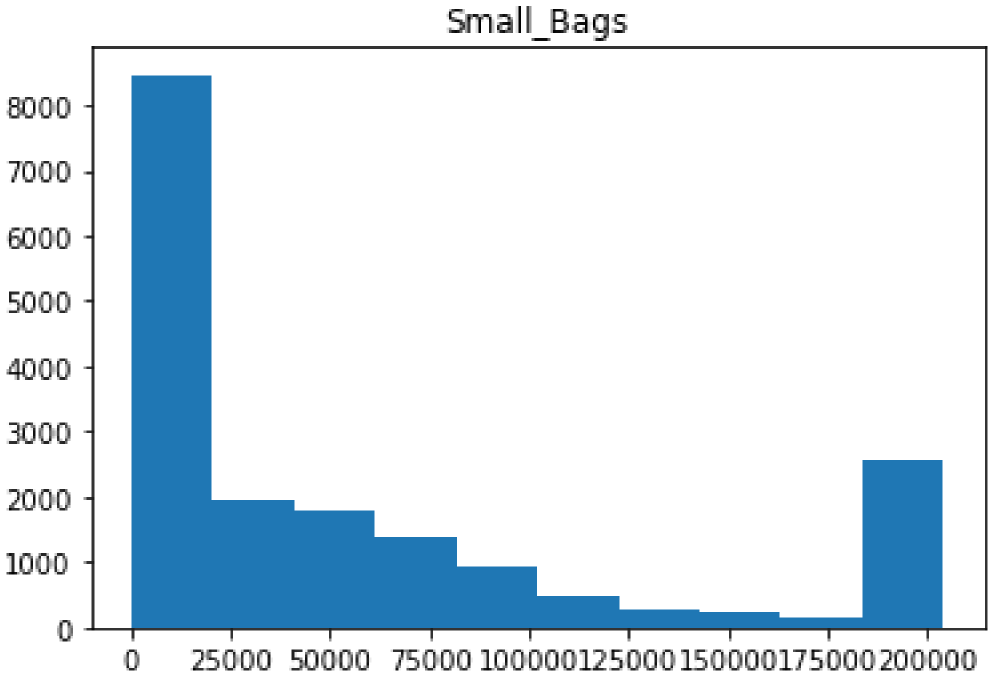
```
[126]: plt.bar(height=df.Small_Bags,x=np.arange(1,18250,1));plt.title("Small_Bags")
```

```
[126]: Text(0.5, 1.0, 'Small_Bags')
```



[127]:
```
plt.hist(df.Small_Bags);plt.title("Small_Bags")
```

[127]:
```
Text(0.5, 1.0, 'Small_Bags')
```

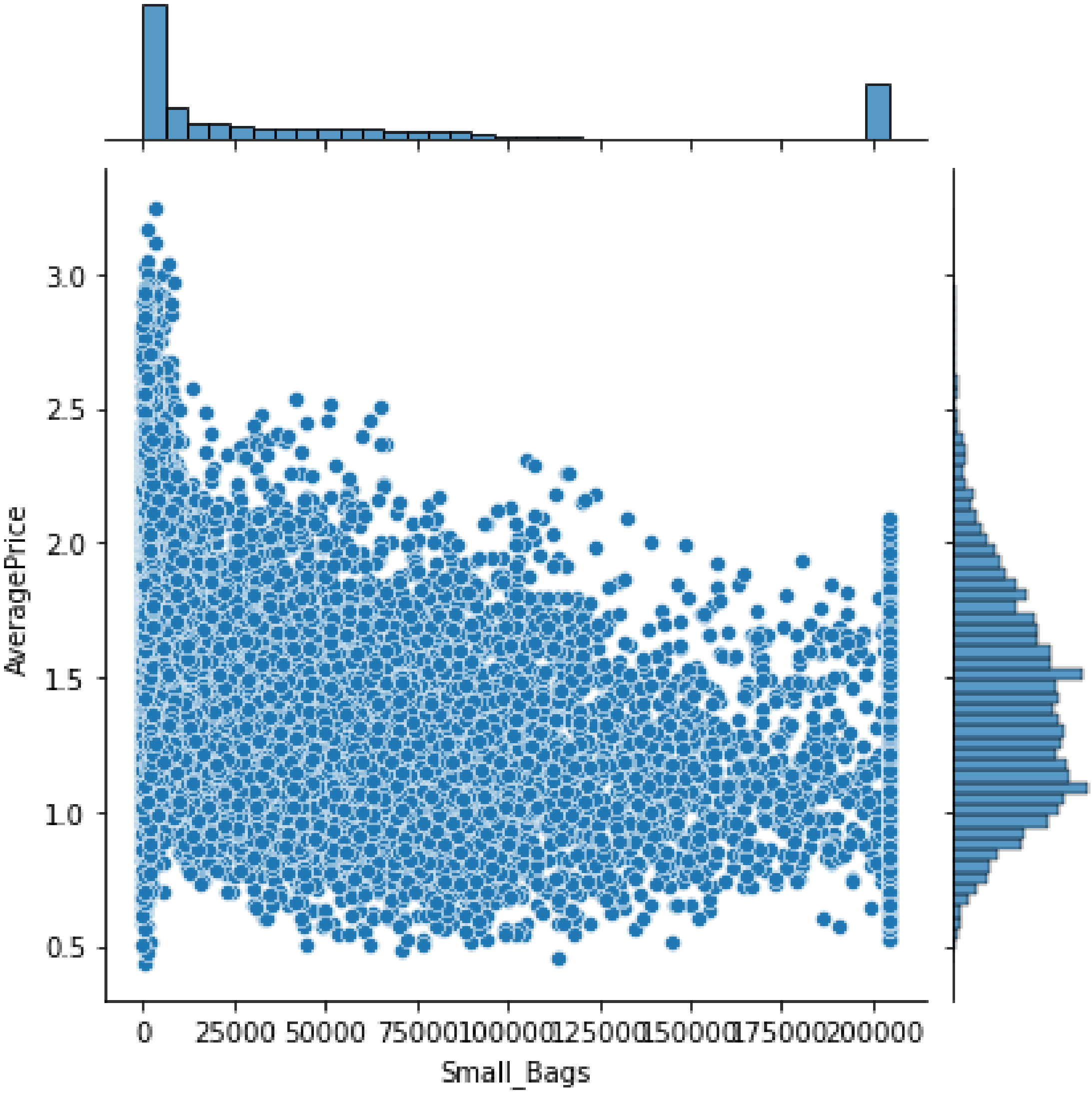


[128]:

```
sns.jointplot(x=df.Small_Bags,y=df.AveragePrice)
```

[128]:

```
<seaborn.axisgrid.JointGrid at 0x12e392988b0>
```

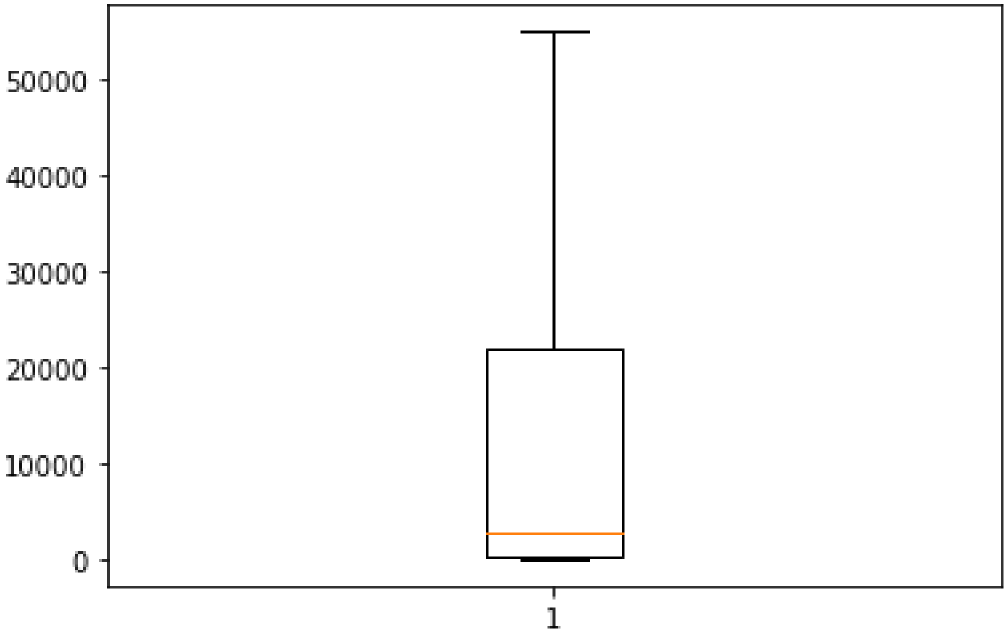


[129]:

```
plt.boxplot(df.Large_Bags)
```

[129]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x12e38afcaf0>,
  <matplotlib.lines.Line2D at 0x12e38afce80>],
 'caps': [<matplotlib.lines.Line2D at 0x12e38b0a250>,
  <matplotlib.lines.Line2D at 0x12e38b0a5e0>],
 'boxes': [<matplotlib.lines.Line2D at 0x12e38afc760>],
 'medians': [<matplotlib.lines.Line2D at 0x12e38b0a970>],
 'fliers': [<matplotlib.lines.Line2D at 0x12e38b0ad00>],
 'means': []}
```



[130]:

```
IQR=df.Large_Bags.quantile(0.75)-df.Large_Bags.quantile(0.25)
lower_limit=df.Large_Bags.quantile(0.25)-(IQR*1.5)
upper_limit=df.Large_Bags.quantile(0.75)-(IQR*1.5)


outliers=np.where(df.Large_Bags>upper_limit,True,np.where(df.
 ↪Large_Bags<lower_limit,True,False))
from feature_engine.outliers import Winsorizer
winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
 ↪gaussian for mean and std
                          tail='both', # cap left, right or both tails
                          fold=1.5,
                           variables=['Large_Bags'])
```

[131]:

```
df_t=winsor.fit_transform(df[["Large_Bags"]])
```

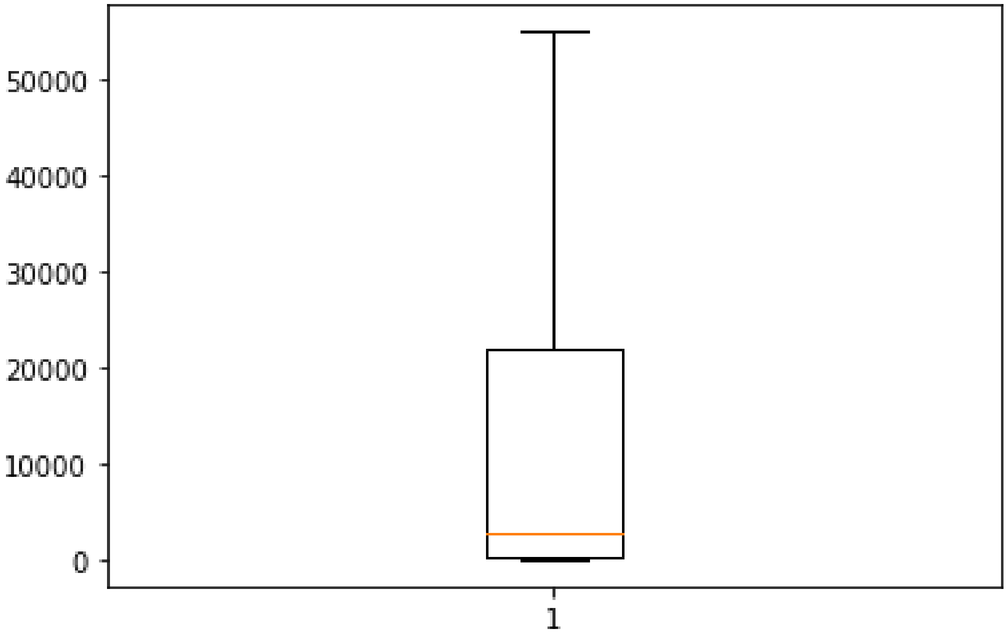
[132]:

```
df.Large_Bags=df_t
```

[133]:

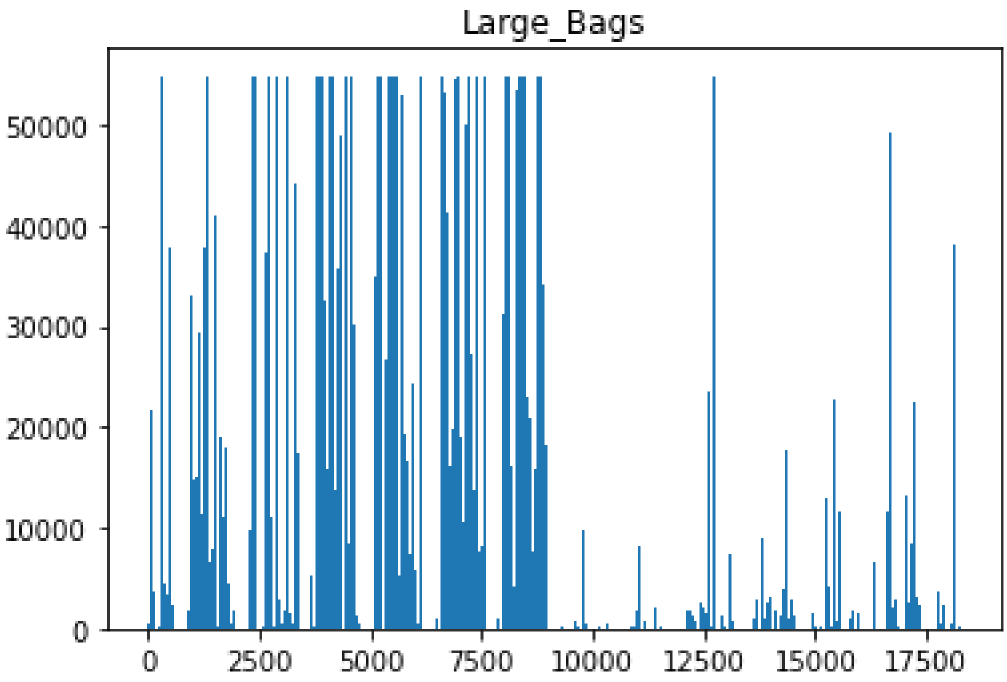
```
plt.boxplot(df.Large_Bags)
```

```
[133]: {'whiskers': [<matplotlib.lines.Line2D at 0x12e38b67760>,
         <matplotlib.lines.Line2D at 0x12e38b67af0>],
        'caps': [<matplotlib.lines.Line2D at 0x12e38b67e80>,
         <matplotlib.lines.Line2D at 0x12e38b72250>],
        'boxes': [<matplotlib.lines.Line2D at 0x12e38b673d0>],
        'medians': [<matplotlib.lines.Line2D at 0x12e38b725e0>],
        'fliers': [<matplotlib.lines.Line2D at 0x12e38b72970>],
        'means': []}
```

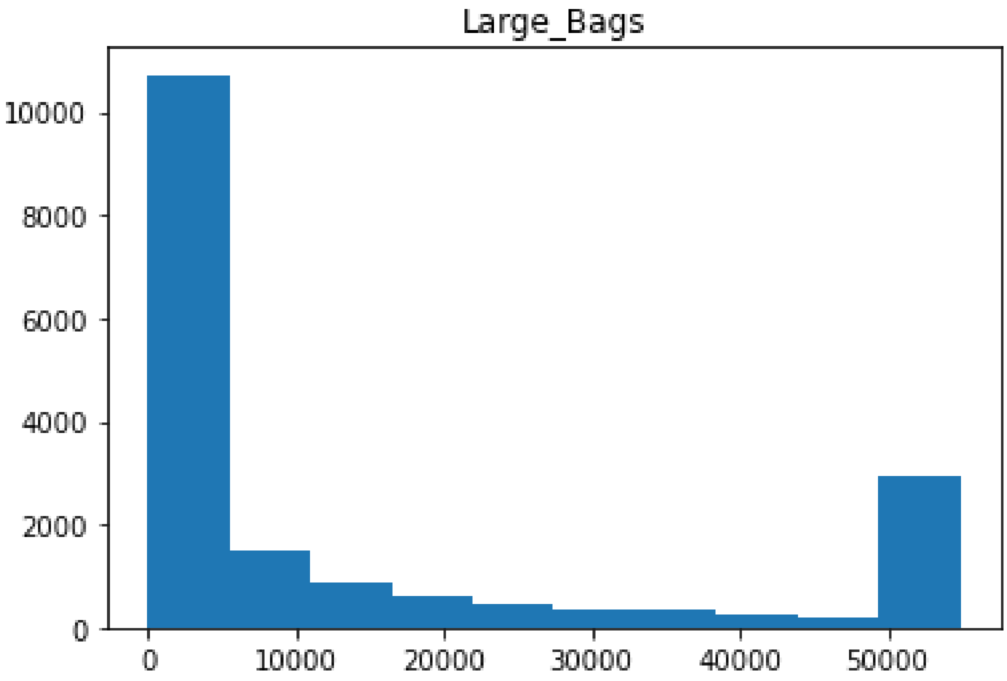


```
[134]: plt.bar(height=df.Large_Bags,x=np.arange(1,18250,1));plt.title("Large_Bags")
```

```
[134]: Text(0.5, 1.0, 'Large_Bags')
```

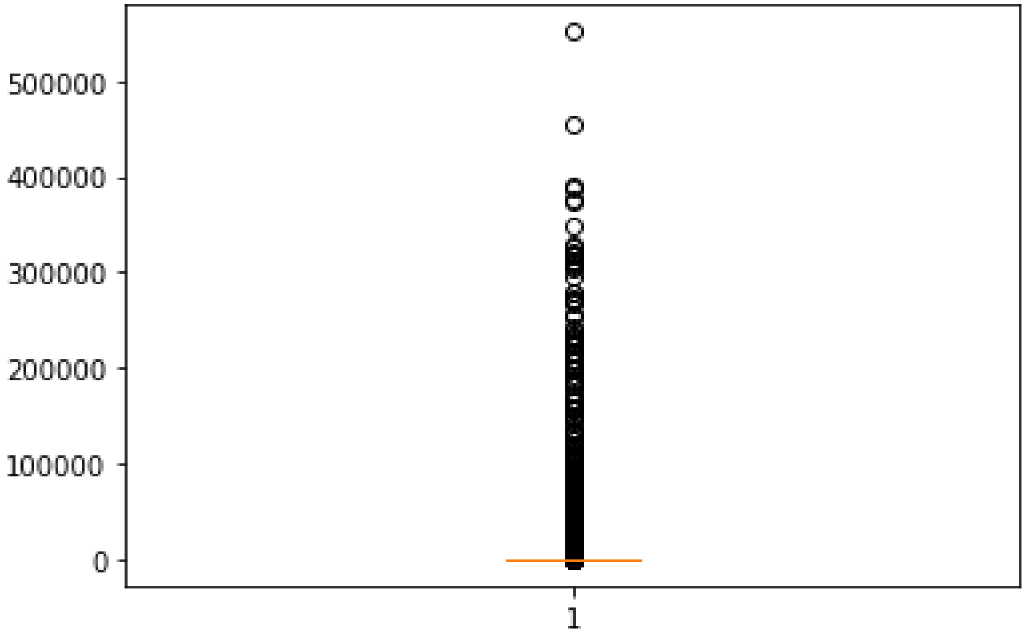


```
[135]: plt.hist(df.Large_Bags);plt.title("Large_Bags")

[135]: Text(0.5, 1.0, 'Large_Bags')
```



[136]:
```
plt.boxplot(df.Xlarge_Bags)
```

[136]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x12e4775bcd0>,
  <matplotlib.lines.Line2D at 0x12e45d980a0>],
 'caps': [<matplotlib.lines.Line2D at 0x12e45d98430>,
  <matplotlib.lines.Line2D at 0x12e45d987c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x12e4775b910>],
 'medians': [<matplotlib.lines.Line2D at 0x12e45d98b50>],
 'fliers': [<matplotlib.lines.Line2D at 0x12e45d98ee0>],
 'means': []}
```



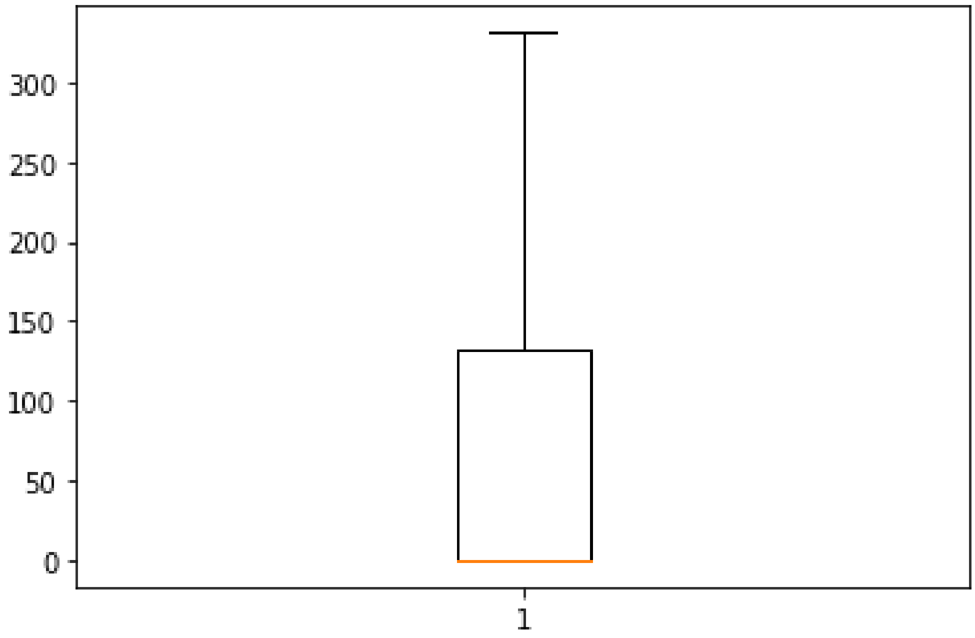
[137]:
```
IQR=df.Large_Bags.quantile(0.75)-df.Xlarge_Bags.quantile(0.25)
lower_limit=df.Xlarge_Bags.quantile(0.25)-(IQR*1.5)
upper_limit=df.Xlarge_Bags.quantile(0.75)-(IQR*1.5)


outliers=np.where(df.Xlarge_Bags>upper_limit,True,np.where(df.
 ↪Xlarge_Bags<lower_limit,True,False))
from feature_engine.outliers import Winsorizer
winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or␣
 ↪gaussian for mean and std
                          tail='both', # cap left, right or both tails
                          fold=1.5,
                           variables=['Xlarge_Bags'])
```

```
[138]: df_t=winsor.fit_transform(df[["Xlarge_Bags"]])
```

```
[139]: df.Xlarge_Bags=df_t
```

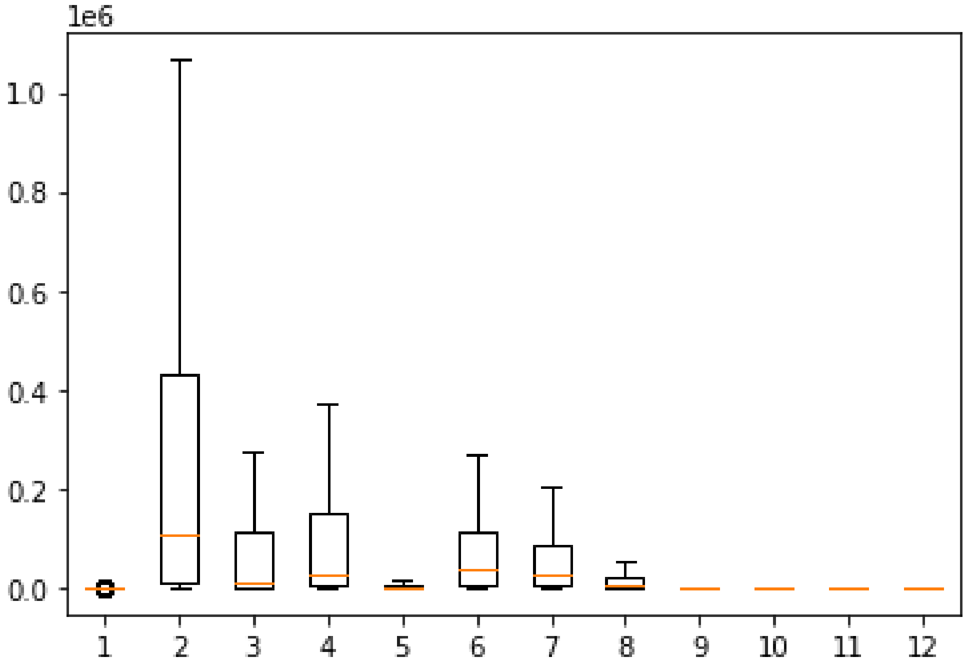
```
[140]: plt.boxplot(df.Xlarge_Bags)
```

```
[140]: {'whiskers': [<matplotlib.lines.Line2D at 0x12e45df2af0>,
          <matplotlib.lines.Line2D at 0x12e45df2e80>],
         'caps': [<matplotlib.lines.Line2D at 0x12e45dff250>,
          <matplotlib.lines.Line2D at 0x12e45dff5e0>],
         'boxes': [<matplotlib.lines.Line2D at 0x12e45df2760>],
         'medians': [<matplotlib.lines.Line2D at 0x12e45dff970>],
         'fliers': [<matplotlib.lines.Line2D at 0x12e45dffd00>],
         'means': []}
```

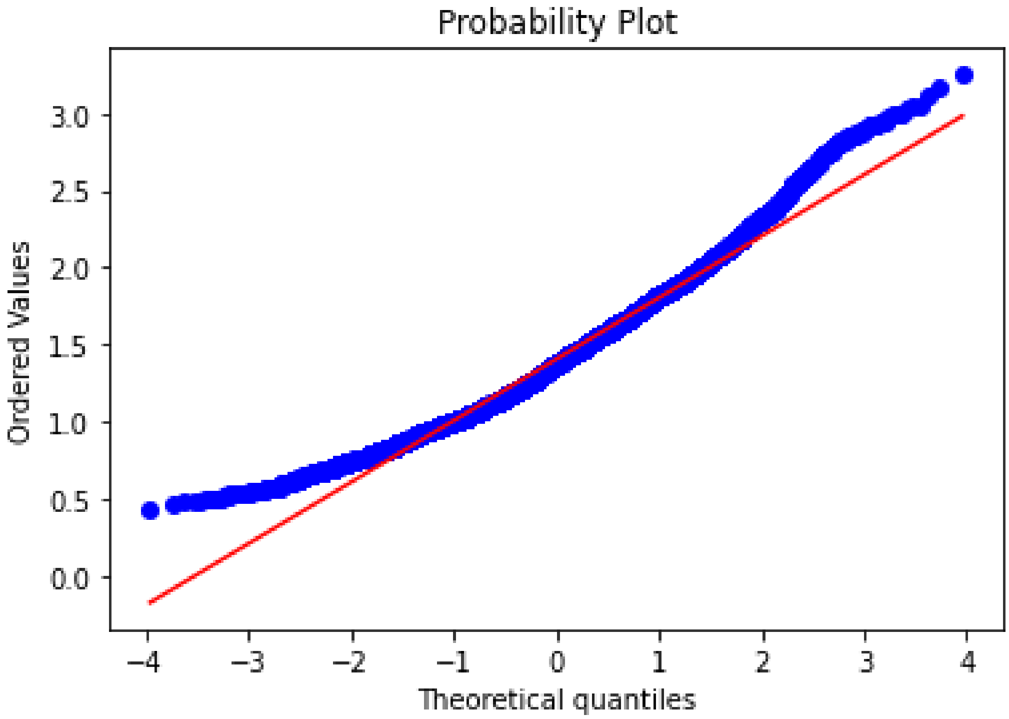


```
[148]: plt.boxplot(df)
```

```
[148]: {'whiskers': [<matplotlib.lines.Line2D at 0x12e50df9340>,
          <matplotlib.lines.Line2D at 0x12e50df96d0>,
          <matplotlib.lines.Line2D at 0x12e50e02c70>,
          <matplotlib.lines.Line2D at 0x12e50e0e040>,
          <matplotlib.lines.Line2D at 0x12e51f485e0>,
          <matplotlib.lines.Line2D at 0x12e51f48970>,
          <matplotlib.lines.Line2D at 0x12e51f56f10>,
          <matplotlib.lines.Line2D at 0x12e51f612e0>,
          <matplotlib.lines.Line2D at 0x12e51f6c880>,
          <matplotlib.lines.Line2D at 0x12e51f6cc10>,
          <matplotlib.lines.Line2D at 0x12e51f821f0>,
          <matplotlib.lines.Line2D at 0x12e51f82580>,
          <matplotlib.lines.Line2D at 0x12e51f8eb20>,
          <matplotlib.lines.Line2D at 0x12e51f8eeb0>,
          <matplotlib.lines.Line2D at 0x12e51fa4490>,
          <matplotlib.lines.Line2D at 0x12e51fa4820>,
          <matplotlib.lines.Line2D at 0x12e51faedc0>,
          <matplotlib.lines.Line2D at 0x12e51fba190>,
          <matplotlib.lines.Line2D at 0x12e51fc5730>,
          <matplotlib.lines.Line2D at 0x12e51fc5ac0>,
          <matplotlib.lines.Line2D at 0x12e51fdc0a0>,
          <matplotlib.lines.Line2D at 0x12e51fdc430>,
          <matplotlib.lines.Line2D at 0x12e51fe69d0>,
          <matplotlib.lines.Line2D at 0x12e51fe6d60>],
         'caps': [<matplotlib.lines.Line2D at 0x12e50df9a60>,
          <matplotlib.lines.Line2D at 0x12e50df9df0>,
          <matplotlib.lines.Line2D at 0x12e50e0e3d0>,
          <matplotlib.lines.Line2D at 0x12e50e0e760>,
          <matplotlib.lines.Line2D at 0x12e51f48d00>,
          <matplotlib.lines.Line2D at 0x12e51f560d0>,
          <matplotlib.lines.Line2D at 0x12e51f61670>,
          <matplotlib.lines.Line2D at 0x12e51f61a00>,
          <matplotlib.lines.Line2D at 0x12e51f6cfa0>,
          <matplotlib.lines.Line2D at 0x12e51f76370>,
          <matplotlib.lines.Line2D at 0x12e51f82910>,
          <matplotlib.lines.Line2D at 0x12e51f82ca0>,
          <matplotlib.lines.Line2D at 0x12e51f97280>,
          <matplotlib.lines.Line2D at 0x12e51f97610>,
          <matplotlib.lines.Line2D at 0x12e51fa4bb0>,
          <matplotlib.lines.Line2D at 0x12e51fa4f40>,
          <matplotlib.lines.Line2D at 0x12e51fba520>,
          <matplotlib.lines.Line2D at 0x12e51fba8b0>,
          <matplotlib.lines.Line2D at 0x12e51fc5e50>,
          <matplotlib.lines.Line2D at 0x12e51fd0220>,
          <matplotlib.lines.Line2D at 0x12e51fdc7c0>,
          <matplotlib.lines.Line2D at 0x12e51fdcb50>,
          <matplotlib.lines.Line2D at 0x12e51ff2130>,
          <matplotlib.lines.Line2D at 0x12e51ff24c0>],
         'boxes': [<matplotlib.lines.Line2D at 0x12e50de6f70>,
          <matplotlib.lines.Line2D at 0x12e50e028e0>,
          <matplotlib.lines.Line2D at 0x12e51f48250>,
          <matplotlib.lines.Line2D at 0x12e51f56b80>,
          <matplotlib.lines.Line2D at 0x12e51f6c4f0>,
          <matplotlib.lines.Line2D at 0x12e51f76e20>,
          <matplotlib.lines.Line2D at 0x12e51f8e790>,
          <matplotlib.lines.Line2D at 0x12e51fa4100>,
          <matplotlib.lines.Line2D at 0x12e51faea30>,
          <matplotlib.lines.Line2D at 0x12e51fc53a0>,
          <matplotlib.lines.Line2D at 0x12e51fd0cd0>,
          <matplotlib.lines.Line2D at 0x12e51fe6640>],
         'medians': [<matplotlib.lines.Line2D at 0x12e50e021c0>,
          <matplotlib.lines.Line2D at 0x12e50e0eaf0>,
          <matplotlib.lines.Line2D at 0x12e51f56460>,
          <matplotlib.lines.Line2D at 0x12e51f61d90>,
          <matplotlib.lines.Line2D at 0x12e51f76700>,
          <matplotlib.lines.Line2D at 0x12e51f8e070>,
          <matplotlib.lines.Line2D at 0x12e51f979a0>,
          <matplotlib.lines.Line2D at 0x12e51fae310>,
          <matplotlib.lines.Line2D at 0x12e51fbac40>,
          <matplotlib.lines.Line2D at 0x12e51fd05b0>,
          <matplotlib.lines.Line2D at 0x12e51fdcee0>,
          <matplotlib.lines.Line2D at 0x12e51ff26a0>],
         'fliers': [<matplotlib.lines.Line2D at 0x12e50e02550>,
          <matplotlib.lines.Line2D at 0x12e50e0ee80>,
          <matplotlib.lines.Line2D at 0x12e51f567f0>,
          <matplotlib.lines.Line2D at 0x12e51f6c160>,
          <matplotlib.lines.Line2D at 0x12e51f76a90>,
          <matplotlib.lines.Line2D at 0x12e51f8e400>,
          <matplotlib.lines.Line2D at 0x12e51f97d30>,
          <matplotlib.lines.Line2D at 0x12e51fae6a0>,
          <matplotlib.lines.Line2D at 0x12e51fbafd0>,
          <matplotlib.lines.Line2D at 0x12e51fd0940>,
          <matplotlib.lines.Line2D at 0x12e51fe62b0>,
          <matplotlib.lines.Line2D at 0x12e51ff2a30>],
         'means': []}
```

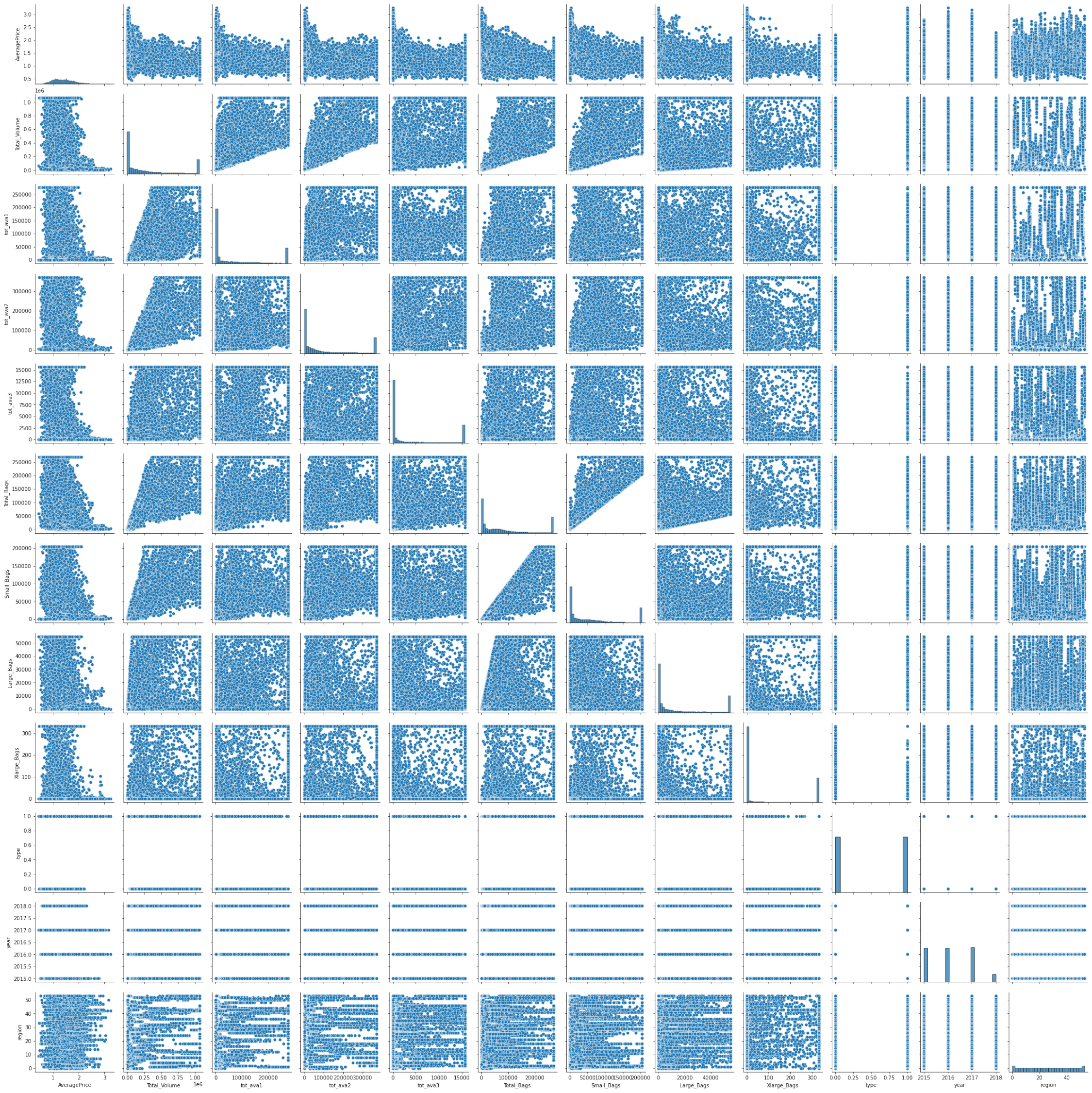


[142]:
```
from scipy import stats
import pylab
stats.probplot(df.AveragePrice, dist = "norm", plot = pylab)
plt.show()
```



```
[143]: sns.pairplot(df.iloc[:, :])
```

```
[143]: <seaborn.axisgrid.PairGrid at 0x12e45f78d00>
```



```
[144]: df.corr()
```

```
[144]:                AveragePrice  Total_Volume  tot_ava1  tot_ava2  tot_ava3  \
       AveragePrice      1.000000     -0.503153 -0.523474 -0.412565 -0.465699
       Total_Volume     -0.503153      1.000000  0.862914  0.918696  0.735324
       tot_ava1         -0.523474      0.862914  1.000000  0.679998  0.646679
       tot_ava2         -0.412565      0.918696  0.679998  1.000000  0.716079
       tot_ava3         -0.465699      0.735324  0.646679  0.716079  1.000000
       Total_Bags       -0.486599      0.926100  0.791529  0.809947  0.663865
       Small_Bags       -0.449332      0.911307  0.759106  0.827511  0.677017
       Large_Bags       -0.424658      0.706783  0.677472  0.561624  0.435210
       Xlarge_Bags      -0.352337      0.584318  0.536757  0.553875  0.559618
       type              0.615845     -0.655466 -0.627013 -0.627230 -0.610211
```

```
year              0.093197      0.038736  0.017481 -0.007602 -0.083827
region           -0.011716      0.106552  0.172991  0.049120  0.038485

              Total_Bags  Small_Bags  Large_Bags  Xlarge_Bags      type  \
AveragePrice   -0.486599   -0.449332   -0.424658    -0.352337  0.615845
Total_Volume    0.926100    0.911307    0.706783     0.584318 -0.655466
tot_ava1        0.791529    0.759106    0.677472     0.536757 -0.627013
tot_ava2        0.809947    0.827511    0.561624     0.553875 -0.627230
tot_ava3        0.663865    0.677017    0.435210     0.559618 -0.610211
Total_Bags      1.000000    0.961362    0.775343     0.599925 -0.623950
Small_Bags      0.961362    1.000000    0.632182     0.600454 -0.620843
Large_Bags      0.775343    0.632182    1.000000     0.429142 -0.464445
Xlarge_Bags     0.599925    0.600454    0.429142     1.000000 -0.592029
type           -0.623950   -0.620843   -0.464445    -0.592029  1.000000
year            0.146330    0.128440    0.144863     0.163137 -0.000032
region          0.117088    0.112723    0.153494     0.005992 -0.000280

                  year    region
AveragePrice  0.093197 -0.011716
Total_Volume  0.038736  0.106552
tot_ava1      0.017481  0.172991
tot_ava2     -0.007602  0.049120
tot_ava3     -0.083827  0.038485
Total_Bags    0.146330  0.117088
Small_Bags    0.128440  0.112723
Large_Bags    0.144863  0.153494
Xlarge_Bags   0.163137  0.005992
type         -0.000032 -0.000280
year          1.000000 -0.000055
region       -0.000055  1.000000
```

[149]:

```
import statsmodels.formula.api as smf
```

[150]:

```
model1=smf.ols("AveragePrice ~ Total_Volume + tot_ava1 + tot_ava2 + tot_ava3  +␣
 ↪Small_Bags +Large_Bags+ Xlarge_Bags+type+year+region ",data=df).fit()
```

[151]:

```
model1.summary()
```

[151]:

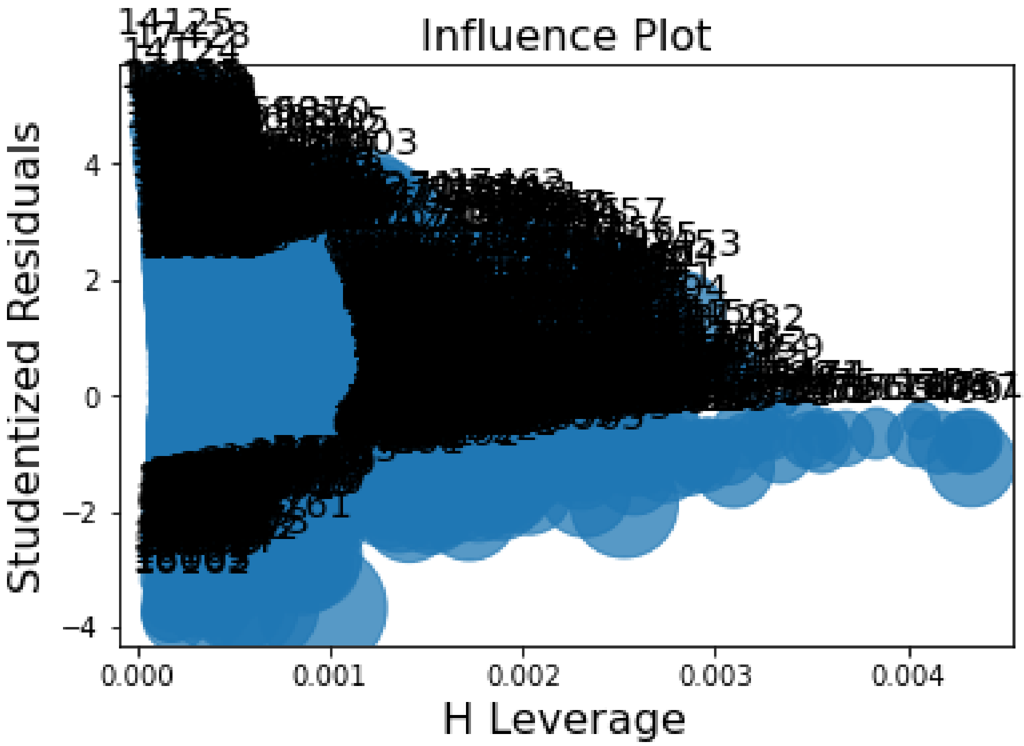
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           AveragePrice   R-squared:                       0.448
Model:                            OLS   Adj. R-squared:                  0.448
Method:                 Least Squares   F-statistic:                     1480.
Date:                Mon, 23 Jan 2023   Prob (F-statistic):               0.00
Time:                        17:23:14   Log-Likelihood:                -3872.0
```

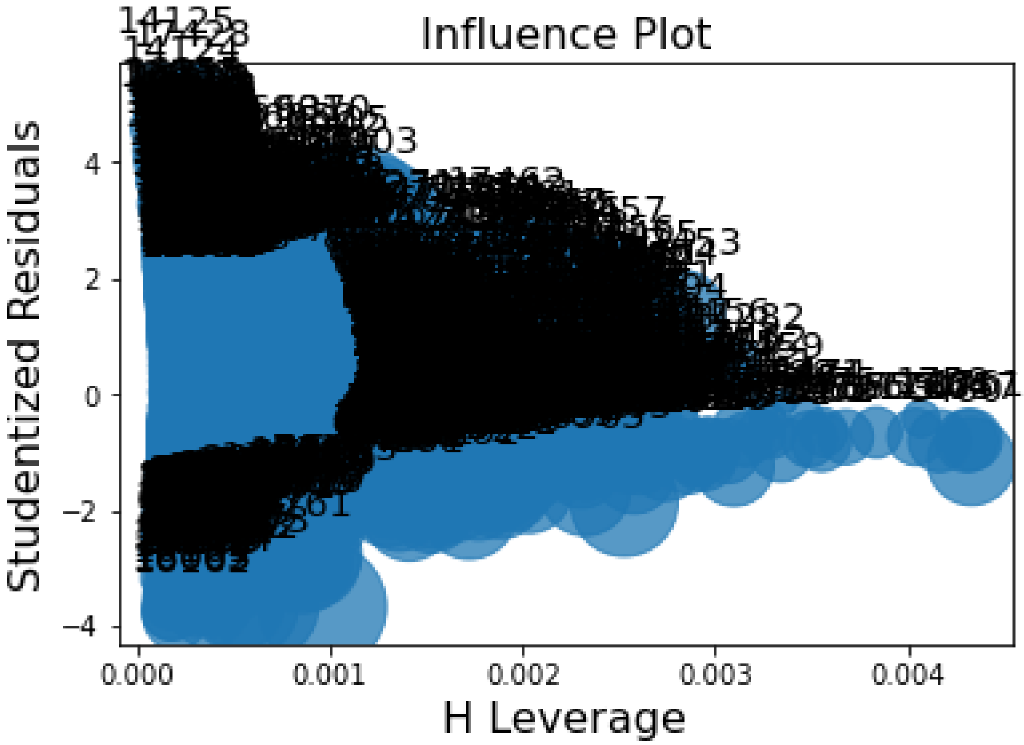
```
No. Observations:               18249   AIC:                             7766.
Df Residuals:                   18238   BIC:                             7852.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      -87.8127      5.126    -17.133      0.000     -97.859     -77.766
Total_Volume -4.499e-07   3.55e-08    -12.691      0.000   -5.19e-07    -3.8e-07
tot_ava1     -1.405e-07   5.82e-08     -2.414      0.016   -2.54e-07   -2.64e-08
tot_ava2      1.132e-06    5.6e-08     20.226      0.000    1.02e-06    1.24e-06
tot_ava3     -7.481e-06   5.94e-07    -12.596      0.000   -8.64e-06   -6.32e-06
Small_Bags    2.932e-07   8.35e-08      3.510      0.000    1.29e-07    4.57e-07
Large_Bags    -2.31e-06   1.68e-07    -13.763      0.000   -2.64e-06   -1.98e-06
Xlarge_Bags      0.0002   2.23e-05      8.440      0.000       0.000       0.000
type             0.4073      0.007     62.074      0.000       0.394       0.420
year             0.0442      0.003     17.372      0.000       0.039       0.049
region           0.0009      0.000      6.333      0.000       0.001       0.001
============================================================================
Omnibus:                     1059.748   Durbin-Watson:                   0.325
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1889.275
Skew:                           0.448   Prob(JB):                         0.00
Kurtosis:                       4.297   Cond. No.                     1.19e+09
============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.19e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[152]:
```python
import statsmodels.api as sm
```

[153]:
```python
sm.graphics.influence_plot(model1)
```

[153]:


Influence Plot
Studentized Residuals
H Leverage
4
2
0
−2
−4
0.000
0.001
0.002
0.003
0.004
14125


Influence Plot
Studentized Residuals
H Leverage
4
2
0
−2
−4
0.000
0.001
0.002
0.003
0.004
14125

```
[ ]:
```

```
[87]: res_vol=smf.ols("Total_Volume ~  tot_ava1 + tot_ava2 + tot_ava3 + Total_Bags +␣
       ↪Small_Bags +Large_Bags+ Xlarge_Bags+type+year+region ",data=df).fit().
       ↪rsquared
```

```
[88]: res_vol = 1/(1 - res_vol)
```

```
[89]: res_vol
```

```
[89]: 39.77116230349727
```

```
[90]: res_ava1=smf.ols("tot_ava1 ~  Total_Volume + tot_ava2 + tot_ava3 + Total_Bags +␣
       ↪Small_Bags +Large_Bags+ Xlarge_Bags+type+year+region ",data=df).fit().
       ↪rsquared
      res_ava1 = 1/(1 - res_ava1)
```

```
[91]: res_ava1
```

```
[91]: 7.129231122219007
```

```
[92]: res_ava2=smf.ols("tot_ava2 ~  Total_Volume + tot_ava1 + tot_ava3 + Total_Bags +␣
       ↪Small_Bags +Large_Bags+ Xlarge_Bags+type+year+region ",data=df).fit().
       ↪rsquared
      res_ava2 = 1/(1 - res_ava2)
```

```
[93]: res_ava2
```

```
[93]: 12.281073160797487
```

```
[94]: res_ava3=smf.ols("tot_ava3 ~  Total_Volume + tot_ava1 + tot_ava2 + Total_Bags +␣
       ↪Small_Bags +Large_Bags+ Xlarge_Bags+type+year+region ",data=df).fit().
       ↪rsquared
      res_ava3 = 1/(1 - res_ava3)
```

```
[95]: res_ava3
```

```
[95]: 2.5860951329231647
```

```
[96]: res_Total_Bags=smf.ols("Total_Bags ~  Total_Volume + tot_ava1 + tot_ava2 +␣
       ↪tot_ava3 + Small_Bags +Large_Bags+ Xlarge_Bags+type+year+region ",data=df).
       ↪fit().rsquared
      res_Total_Bags = 1/(1 - res_Total_Bags)
```

```
[97]: res_Total_Bags
```

[97]:
```
39.88789871182136
```

[101]:
```
res_Small_Bags=smf.ols("Total_Bags ~  Total_Volume + tot_ava1 + tot_ava2 +␣
 ↪tot_ava3 + Total_Bags +Large_Bags+ Xlarge_Bags+type+year+region ",data=df).
 ↪fit().rsquared
res_Small_Bags = 1/(1 - res_Small_Bags)
```

```
C:\Users\Ravi\AppData\Local\Temp\ipykernel_12936\662951263.py:2: RuntimeWarning:
divide by zero encountered in double_scalars
  res_Small_Bags = 1/(1 - res_Small_Bags)
```

[102]:
```
res_Small_Bags
```

[102]:
```
inf
```

[103]:
```
res_Large_Bags=smf.ols("Large_Bags ~  Total_Volume + tot_ava1 + tot_ava2 +␣
 ↪tot_ava3 + Total_Bags +Small_Bags+ Xlarge_Bags+type+year+region ",data=df).
 ↪fit().rsquared
res_Large_Bags = 1/(1 - res_Large_Bags)
```

[104]:
```
res_Large_Bags
```

[104]:
```
4.693747595611235
```

[105]:
```
res_Xlarge_Bags=smf.ols("Xlarge_Bags ~  Total_Volume + tot_ava1 + tot_ava2 +␣
 ↪tot_ava3 + Total_Bags +Small_Bags+ Large_Bags+type+year+region ",data=df).
 ↪fit().rsquared
res_Xlarge_Bags = 1/(1 - res_Xlarge_Bags)
```

[106]:
```
res_Xlarge_Bags
```

[106]:
```
1.9457175245752574
```

[107]:
```
res_type=smf.ols("type ~  Total_Volume + tot_ava1 + tot_ava2 + tot_ava3 +␣
 ↪Total_Bags +Small_Bags+ Large_Bags+Xlarge_Bags+year+region ",data=df).fit().
 ↪rsquared
res_type = 1/(1 - res_type)
```

[108]:
```
res_type
```

[108]:
```
2.197432470456866
```

[109]:
```
res_year=smf.ols("year ~  Total_Volume + tot_ava1 + tot_ava2 + tot_ava3 +␣
 ↪Total_Bags +Small_Bags+ Large_Bags+Xlarge_Bags+type+region ",data=df).fit().
 ↪rsquared
res_year = 1/(1 - res_year)
```

[110]:
```
res_year
```

```
[110]: 1.1677948547468082
```

```
[111]: res_region=smf.ols("region ~  Total_Volume + tot_ava1 + tot_ava2 + tot_ava3 +␣
        ↪Total_Bags +Small_Bags+ Large_Bags+Xlarge_Bags+type+year ",data=df).fit().
        ↪rsquared
       res_region = 1/(1 - res_region)
```

```
[112]: res_region
```

```
[112]: 1.0767857251331638
```

```
[113]: df1 = {'Variables':['region','Total_Volume', 'tot_ava1',␣
        ↪'tot_ava2','tot_ava3','Total_Bags','Small_Bags','Large_Bags','Xlarge_Bags','type','year'],␣
        ↪'VIF':[res_region,res_vol, res_ava1,␣
        ↪res_ava2,res_ava3,res_Total_Bags,res_Small_Bags,res_Large_Bags,res_Xlarge_Bags,res_type,res
       Vif_frame = pd.DataFrame(df1)
```

```
[114]: Vif_frame
```

```
[114]:       Variables        VIF
       0         region   1.076786
       1   Total_Volume  39.771162
       2       tot_ava1   7.129231
       3       tot_ava2  12.281073
       4       tot_ava3   2.586095
       5     Total_Bags  39.887899
       6     Small_Bags        inf
       7     Large_Bags   4.693748
       8    Xlarge_Bags   1.945718
       9           type   2.197432
       10          year   1.167795
```

```
[158]: model2=smf.ols("AveragePrice ~   tot_ava1  + tot_ava3 + Small_Bags +Large_Bags+␣
        ↪Xlarge_Bags+type+year+region ",data=df).fit()
```

```
[159]: model2.summary()
```

```
[159]: <class 'statsmodels.iolib.summary.Summary'>
       """
                                   OLS Regression Results
       ==============================================================================
       Dep. Variable:           AveragePrice   R-squared:                       0.434
       Model:                            OLS   Adj. R-squared:                  0.434
       Method:                 Least Squares   F-statistic:                     1751.
       Date:                Mon, 23 Jan 2023   Prob (F-statistic):               0.00
       Time:                        17:39:58   Log-Likelihood:                -4093.6
       No. Observations:               18249   AIC:                             8205.
       Df Residuals:                   18240   BIC:                             8276.
       Df Model:                           8
       Covariance Type:            nonrobust
       ==============================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
       ------------------------------------------------------------------------------
       Intercept    -76.7944      5.137    -14.948      0.000     -86.864     -66.725
       tot_ava1   -6.904e-07    4.1e-08    -16.835      0.000   -7.71e-07    -6.1e-07
       tot_ava3   -5.266e-06   5.74e-07     -9.178      0.000   -6.39e-06   -4.14e-06
       Small_Bags  3.227e-07   5.81e-08      5.553      0.000    2.09e-07    4.37e-07
       Large_Bags -2.808e-06   1.58e-07    -17.768      0.000   -3.12e-06    -2.5e-06
       Xlarge_Bags    0.0002   2.25e-05      9.950      0.000       0.000       0.000
       type           0.3832      0.007     58.600      0.000       0.370       0.396
       year           0.0387      0.003     15.192      0.000       0.034       0.044
       region         0.0009      0.000      6.232      0.000       0.001       0.001
       ==============================================================================
       Omnibus:                      986.868   Durbin-Watson:                   0.314
       Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1685.127
       Skew:                           0.436   Prob(JB):                         0.00
       Kurtosis:                       4.207   Cond. No.                     3.37e+08
       ==============================================================================

       Notes:
       [1] Standard Errors assume that the covariance matrix of the errors is correctly
       specified.
       [2] The condition number is large, 3.37e+08. This might indicate that there are
       strong multicollinearity or other numerical problems.
       """
```

[160]:

```
pre1=model2.predict(df)
```

[161]:

```
res1 = df.AveragePrice - pre1
res_sqr1 = res1 * res1
mse1 = np.mean(res_sqr1)
rmse1 = np.sqrt(mse1)
rmse1
```

[161]:

```
0.30281947789261016
```

[162]:

```
model3=smf.ols("AveragePrice ~   np.log(tot_ava1  + tot_ava3 + Small_Bags␣
 ↪+Large_Bags+ Xlarge_Bags+type+year+region) ",data=df).fit()
```

[163]:

```
model3.summary()
```

[163]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           AveragePrice   R-squared:                       0.379
Model:                            OLS   Adj. R-squared:                  0.379
Method:                 Least Squares   F-statistic:                 1.113e+04
Date:                Mon, 23 Jan 2023   Prob (F-statistic):               0.00
Time:                        17:40:44   Log-Likelihood:                -4948.3
No. Observations:               18249   AIC:                             9901.
Df Residuals:                   18247   BIC:                             9916.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
------------------------------------------------------------------------------
Intercept
2.9384      0.015    199.710      0.000       2.910       2.967
np.log(tot_ava1 + tot_ava3 + Small_Bags + Large_Bags + Xlarge_Bags + type + year
+ region)    -0.1422      0.001   -105.507      0.000      -0.145      -0.140
==============================================================================
Omnibus:                     1311.280   Durbin-Watson:                   0.252
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1798.659
Skew:                           0.624   Prob(JB):                         0.00
Kurtosis:                       3.900   Cond. No.                         69.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[164]:
```
pre2=model3.predict(df)

res2 = df.AveragePrice - pre2
res_sqr2 = res2 * res2
mse2 = np.mean(res_sqr2)
rmse2 = np.sqrt(mse2)
rmse2
```

[164]:
```
0.31733969488554886
```

[165]:
```
model4=smf.ols("np.log(AveragePrice) ~   (tot_ava1  + tot_ava3 + Small_Bags␣
 ↪+Large_Bags+ Xlarge_Bags+type+year+region) ",data=df).fit()
```

[166]:
```
model4.summary()
```

[166]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

```
                            OLS Regression Results
================================================================================
Dep. Variable:     np.log(AveragePrice)   R-squared:                       0.457
Model:                              OLS   Adj. R-squared:                  0.456
Method:                   Least Squares   F-statistic:                     1916.
Date:                  Mon, 23 Jan 2023   Prob (F-statistic):               0.00
Time:                          17:49:03   Log-Likelihood:                 2282.1
No. Observations:                 18249   AIC:                            -4546.
Df Residuals:                     18240   BIC:                            -4476.
Df Model:                             8
Covariance Type:              nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept    -50.0463      3.622    -13.816      0.000     -57.147     -42.946
tot_ava1   -6.849e-07   2.89e-08    -23.685      0.000    -7.42e-07   -6.28e-07
tot_ava3   -5.098e-06   4.05e-07    -12.601      0.000    -5.89e-06   -4.31e-06
Small_Bags   3.84e-07    4.1e-08      9.370      0.000     3.04e-07    4.64e-07
Large_Bags  -2.12e-06   1.11e-07    -19.030      0.000    -2.34e-06    -1.9e-06
Xlarge_Bags    0.0002   1.58e-05     11.829      0.000       0.000       0.000
type           0.2590      0.005     56.170      0.000       0.250       0.268
year           0.0249      0.002     13.873      0.000       0.021       0.028
region         0.0007      0.000      6.542      0.000       0.000       0.001
==============================================================================
Omnibus:                      613.278   Durbin-Watson:                   0.344
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              925.240
Skew:                          -0.331   Prob(JB):                    1.22e-201
Kurtosis:                       3.882   Cond. No.                     3.37e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.37e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[ ]:

```
[167]: pre3=model4.predict(df)

       res3 = df.AveragePrice - pre3
       res_sqr3 = res3 * res3
       mse3 = np.mean(res_sqr3)
       rmse3 = np.sqrt(mse3)
       rmse3
```

[167]:
```
1.1493230854794034
```

[137]:
```
data = {"MODEL":pd.Series(["model2", "Log model", "Exp model"]), "RMSE":pd.
 ↪Series([rmse1, rmse2, rmse3])}
table_rmse = pd.DataFrame(data)
table_rmse
```

[137]:
```
       MODEL      RMSE
0     model2  0.302819
1  Log model  0.317340
2  Exp model  1.149323
```

[168]:
```
final_model=smf.ols("(AveragePrice) ~   (tot_ava1  + tot_ava3 + Small_Bags␣
 ↪+Large_Bags+ Xlarge_Bags+type+year+region) ",data=df).fit()
```

[169]:
```
final_model.summary()
```

[169]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           AveragePrice   R-squared:                       0.434
Model:                            OLS   Adj. R-squared:                  0.434
Method:                 Least Squares   F-statistic:                     1751.
Date:                Mon, 23 Jan 2023   Prob (F-statistic):               0.00
Time:                        17:52:20   Log-Likelihood:                -4093.6
No. Observations:               18249   AIC:                             8205.
Df Residuals:                   18240   BIC:                             8276.
Df Model:                           8
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept     -76.7944      5.137    -14.948      0.000     -86.864     -66.725
tot_ava1    -6.904e-07    4.1e-08    -16.835      0.000   -7.71e-07    -6.1e-07
tot_ava3    -5.266e-06   5.74e-07     -9.178      0.000   -6.39e-06   -4.14e-06
Small_Bags   3.227e-07   5.81e-08      5.553      0.000    2.09e-07    4.37e-07
Large_Bags  -2.808e-06   1.58e-07    -17.768      0.000   -3.12e-06    -2.5e-06
Xlarge_Bags     0.0002   2.25e-05      9.950      0.000       0.000       0.000
type            0.3832      0.007     58.600      0.000       0.370       0.396
year            0.0387      0.003     15.192      0.000       0.034       0.044
region          0.0009      0.000      6.232      0.000       0.001       0.001
==============================================================================
Omnibus:                      986.868   Durbin-Watson:                   0.314
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1685.127
Skew:                           0.436   Prob(JB):                         0.00
Kurtosis:                       4.207   Cond. No.                     3.37e+08
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.37e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

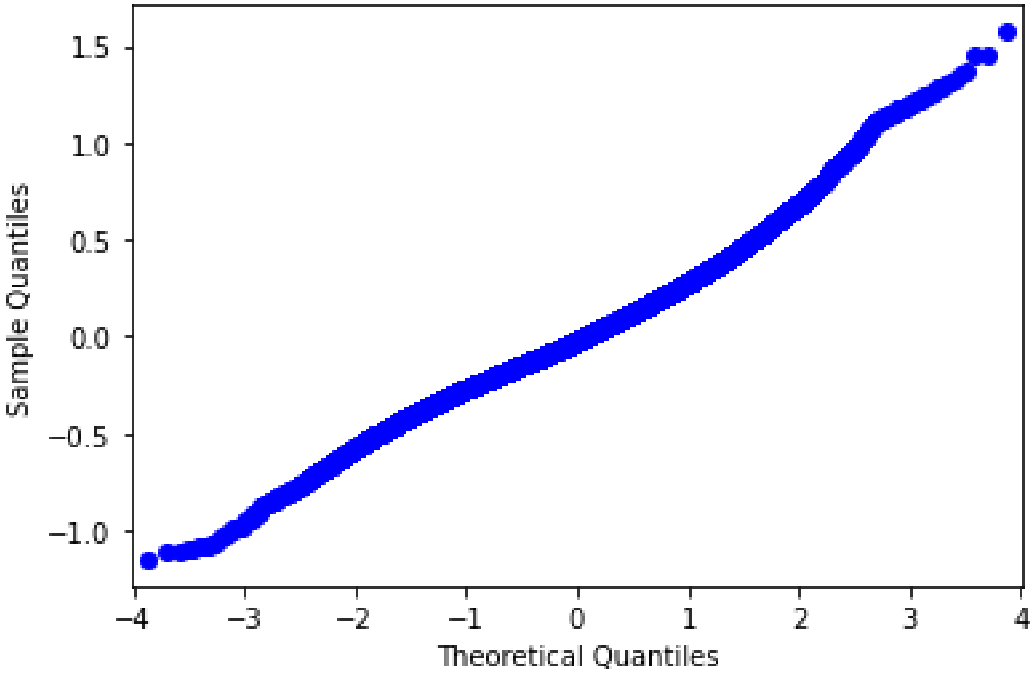
pre_final=final_model.predict(df)

```
[170]: pre_final
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_18108\2097908561.py in <module>
----> 1 pre_final

NameError: name 'pre_final' is not defined
```

```
[171]: res = final_model.resid
       sm.qqplot(res)
       plt.show()
```

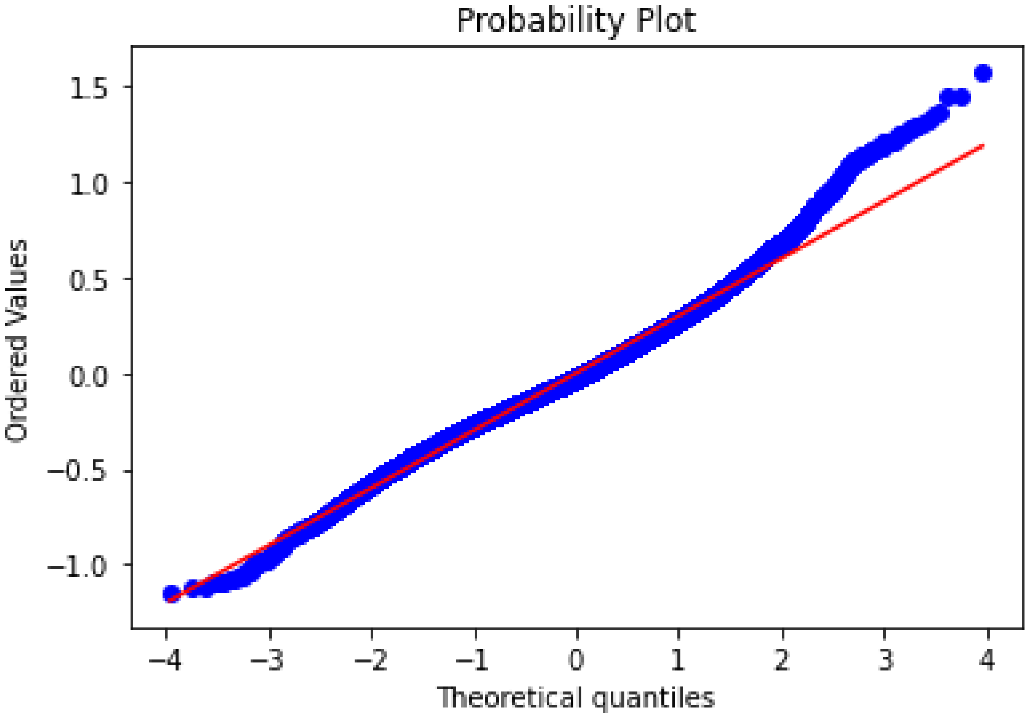
```
C:\Users\Ravi\anaconda3\lib\site-packages\statsmodels\graphics\gofplots.py:993:
UserWarning: marker is redundantly defined by the 'marker' keyword argument and
the fmt string "bo" (-> marker='o'). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```



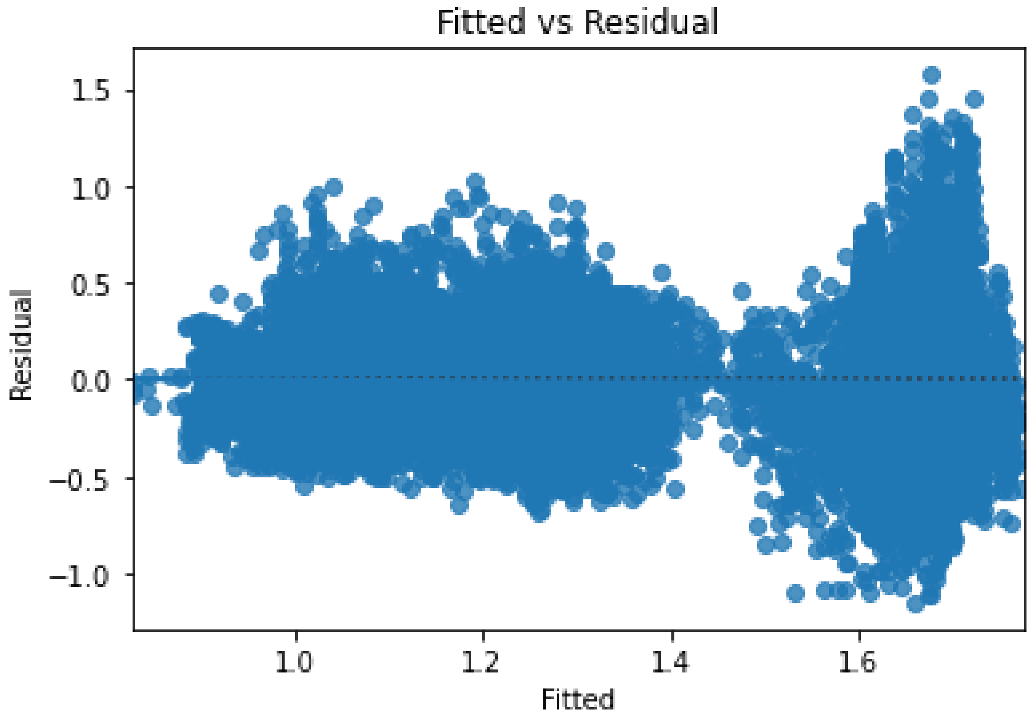
[144]:
```
pre_final
```

[144]:
```
0        1.218806
1        1.219250
2        1.218331
3        1.217554
4        1.217588
           …
18244    1.767211
18245    1.766770
18246    1.763862
18247    1.764120
18248    1.766234
Length: 18249, dtype: float64
```

[172]:
```
# Q-Q plot
stats.probplot(res, dist = "norm", plot = pylab)
plt.show()
```

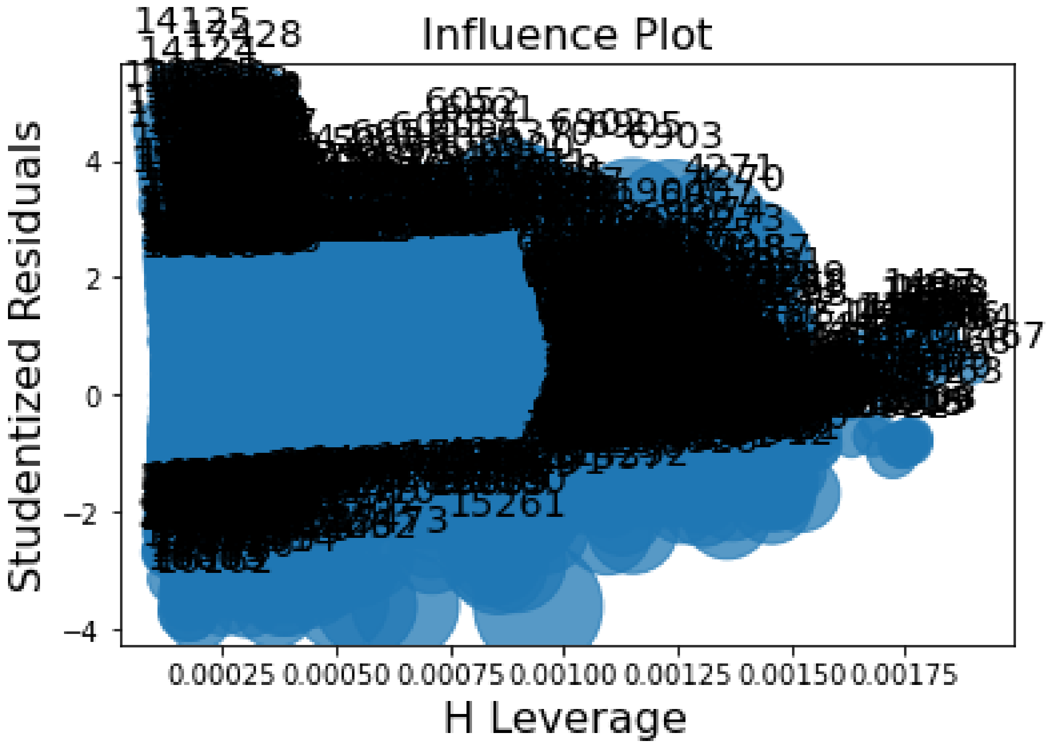


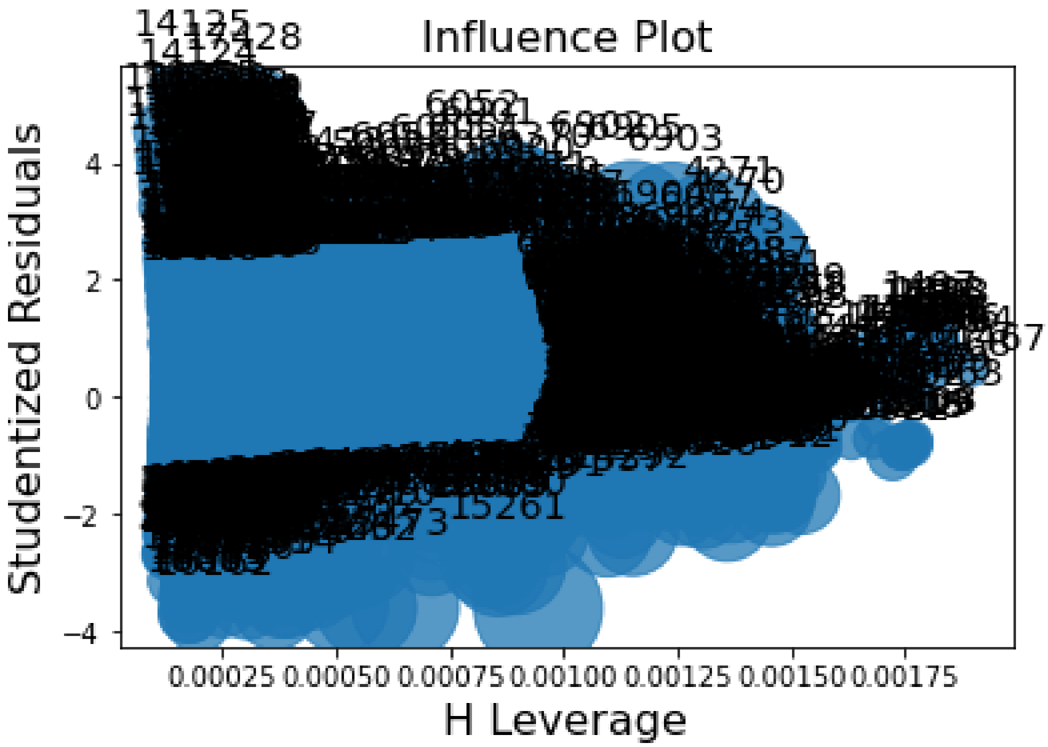
[146]:
```
sns.residplot(x = pred, y = df.AveragePrice, lowess = True)
plt.xlabel('Fitted')
plt.ylabel('Residual')
plt.title('Fitted vs Residual')
plt.show()
```



```
[122]: sm.graphics.influence_plot(final_model)
```

[122]:





[173]:
```
from sklearn.model_selection import train_test_split
df_train, df_test = train_test_split(df, test_size = 0.2,random_state=0) # 20%␣
 ↪test data
```

[174]:
```
model_train = smf.ols("AveragePrice ~  tot_ava1 +   tot_ava3 + Total_Bags +␣
 ↪Small_Bags +Large_Bags+ Xlarge_Bags+type+year+region", data = df_train).fit()
```

[175]:
```
test_pred = model_train.predict(df_test)
```

[176]:
```
# test residual values
test_resid = test_pred - df_test.AveragePrice
```

[177]:
```
# RMSE value for test data
test_rmse = np.sqrt(np.mean(test_resid * test_resid))
test_rmse
```

[177]:
```
0.3025386373302603
```

[178]:
```
# train_data prediction
train_pred = model_train.predict(df_train)
```

[179]:

```
# train residual values
train_resid  = train_pred - df_train.AveragePrice
```

[180]:

```
# RMSE value for train data
train_rmse = np.sqrt(np.mean(train_resid * train_resid))
train_rmse
```

[180]:

```
0.3015786165917111
```

[181]:

```
train_rmse
```

[181]:

```
0.3015786165917111
```

[ ]:

[ ]:

[ ]: