In [1]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as pt
import seaborn as sns
import missingno as msno
from sklearn.preprocessing import StandardScaler
import sklearn.metrics as sm
ds=pd.read_csv(r"C:\Users\harini\OneDrive\Documents\city_day.csv")
print(ds.head(5))
print(ds.info())
#print(ds.isna())
```

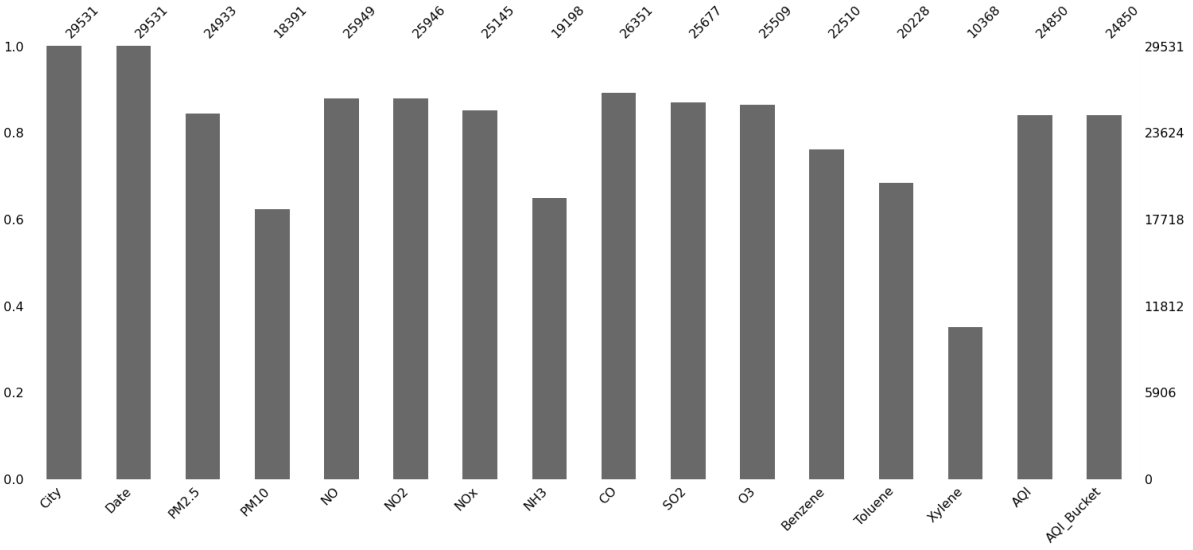
```
        City        Date  PM2.5  PM10     NO    NO2    NOx  NH3     CO    SO2  \
0  Ahmedabad  01-01-2015    NaN   NaN   0.92  18.22  17.15  NaN   0.92  27.64
1  Ahmedabad  02-01-2015    NaN   NaN   0.97  15.69  16.46  NaN   0.97  24.55
2  Ahmedabad  03-01-2015    NaN   NaN  17.40  19.30  29.70  NaN  17.40  29.07
3  Ahmedabad  04-01-2015    NaN   NaN   1.70  18.48  17.97  NaN   1.70  18.59
4  Ahmedabad  05-01-2015    NaN   NaN  22.10  21.42  37.76  NaN  22.10  39.33

       O3  Benzene  Toluene  Xylene  AQI AQI_Bucket
0  133.36     0.00     0.02    0.00  NaN        NaN
1   34.06     3.68     5.50    3.77  NaN        NaN
2   30.70     6.80    16.40    2.25  NaN        NaN
3   36.08     4.43    10.14    1.00  NaN        NaN
4   39.31     7.01    18.89    2.78  NaN        NaN
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        29531 non-null  object
 1   Date        29531 non-null  object
 2   PM2.5       24933 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25145 non-null  float64
 7   NH3         19198 non-null  float64
 8   CO          26351 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     22510 non-null  float64
 12  Toluene     20228 non-null  float64
 13  Xylene      10368 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
None
```

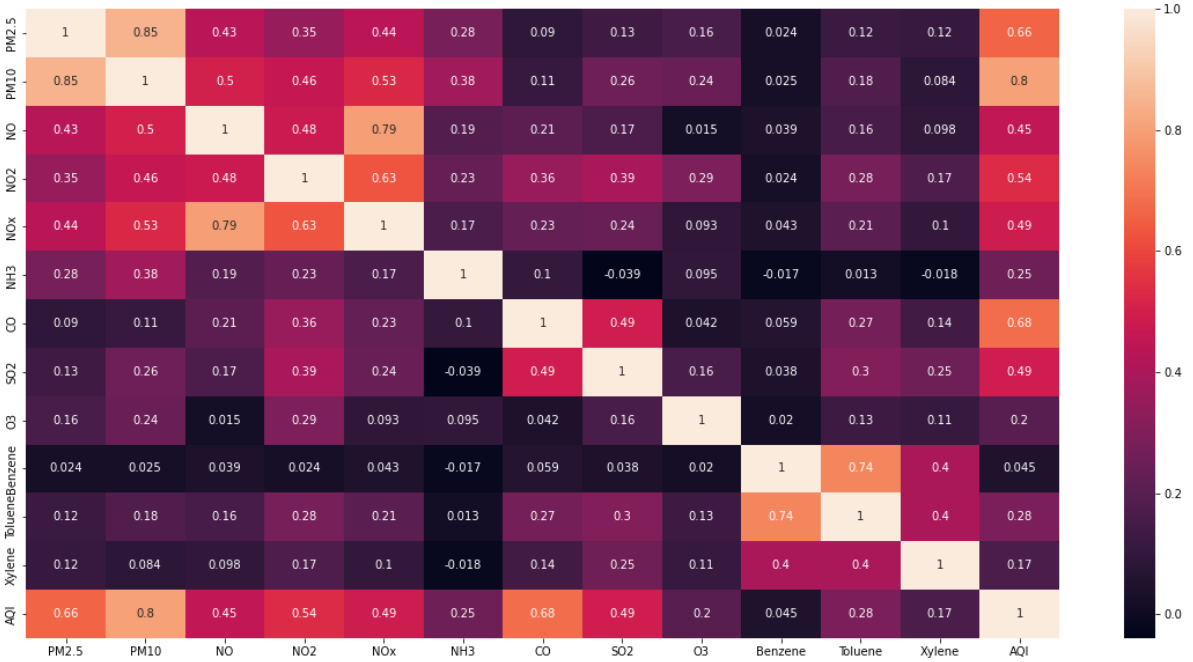
In [2]:
```
import warnings
warnings.filterwarnings("ignore")
ds['Date']=pd.to_datetime(ds['Date'],infer_datetime_format=True)
print(ds.isna().sum())
df=ds.copy()
```

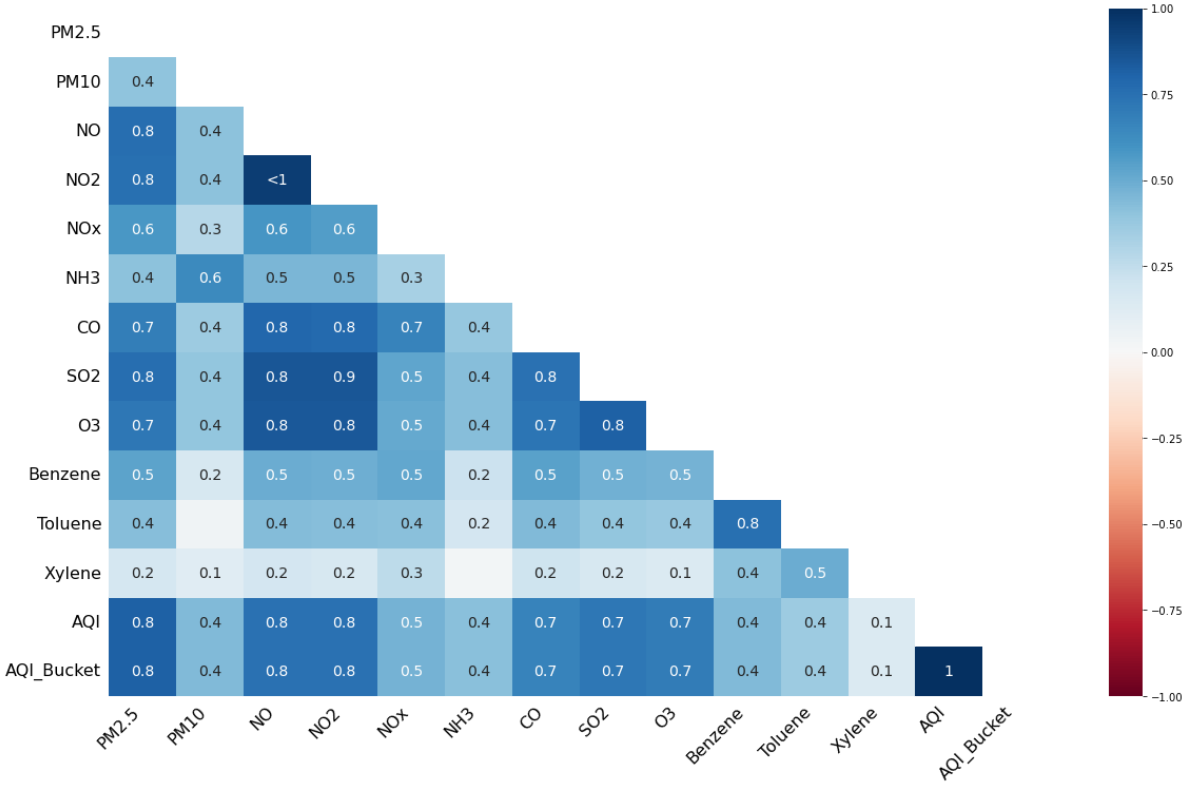
```
City              0
Date              0
PM2.5          4598
PM10          11140
NO             3582
NO2            3585
NOx            4386
NH3           10333
CO             3180
SO2            3854
O3             4022
Benzene        7021
Toluene        9303
Xylene        19163
AQI            4681
AQI_Bucket     4681
dtype: int64
```

In [3]:

```
pt.figure(figsize=(30,10))
msno.bar(ds)
pt.figure(figsize=(20,10))
sns.heatmap(ds.corr(),annot=True)
pt.show()
msno.heatmap(ds)
pt.show()
```







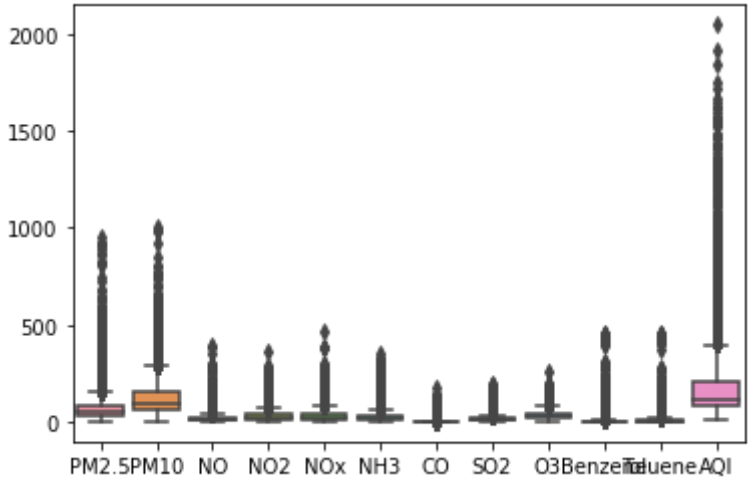
In [4]:
```
"""remove xylene attriibute as there are many null values"""
ds=df.dropna(axis=0,thresh=ds.shape[1]-13)
ds=ds.drop(labels="Xylene",axis=1)
print(ds.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27455 entries, 0 to 29530
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        27455 non-null  object
 1   Date        27455 non-null  datetime64[ns]
 2   PM2.5       24933 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25145 non-null  float64
 7   NH3         19198 non-null  float64
 8   CO          26351 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     22510 non-null  float64
 12  Toluene     20228 non-null  float64
 13  AQI         24850 non-null  float64
 14  AQI_Bucket  24850 non-null  object
dtypes: datetime64[ns](1), float64(12), object(2)
memory usage: 3.4+ MB
None
```

In [5]:
```
d=pd.DataFrame()
sns.boxplot(data=ds,orient=(90))
pt.figure(figsize=(100,80))
x=pd.DatetimeIndex(ds['Date']).year
def find_outliers_IQR(ds):
   q1=ds.quantile(0.25)
   q3=ds.quantile(0.75)
   IQR=q3-q1
   outliers = ds[((ds<(q1-1.5*IQR)) | (ds>(q3+1.5*IQR)))]
   return outliers
```

```
for i in range(2,len(ds.columns)-1):
    outliers = find_outliers_IQR(ds[ds.columns[i]])
    print("average outlier value of: ",ds.columns[i]," is ",outliers.mean())
```

```
average outlier value of:  PM2.5  is  237.51763874873913
average outlier value of:  PM10  is  381.3118259224218
average outlier value of:  NO  is  73.79220821472148
average outlier value of:  NO2  is  104.84487373737355
average outlier value of:  NOx  is  118.89449649973056
average outlier value of:  NH3  is  102.33393103448279
average outlier value of:  CO  is  15.836593939393948
average outlier value of:  SO2  is  57.32219937936394
average outlier value of:  O3  is  104.7757082748949
average outlier value of:  Benzene  is  27.186726001271396
average outlier value of:  Toluene  is  45.00981312472827
average outlier value of:  AQI  is  565.4072164948453
```




```
<Figure size 7200x5760 with 0 Axes>
```

In [6]:

```
drop_outlier = ds[(ds['AQI']>600) | (ds['PM2.5']>200) | (ds['NO']>70) |(ds['NH3']>
                  | (ds['NO2']>90) | (ds['NOx']>100) |
                  (ds['PM10']>400) |(ds['CO']>30)|(ds['SO2']>60) | (ds['O3']>125)
                  | (ds["Toluene"]>50)].index
ds=ds.drop(drop_outlier)
print(ds.describe())
city_uni=(ds.iloc[:,0]).unique()
dff=df.copy()
```

```
              PM2.5          PM10            NO           NO2           NOx  \
count  20341.000000  15827.000000  21162.000000  21195.000000  20534.000000
mean      53.105744    101.851170     12.591890     24.109085     25.595190
std       37.279587     65.734557     11.589113     16.531758     19.093261
min        0.040000      0.010000      0.020000      0.010000      0.000000
25%       26.190000     53.490000      5.060000     11.070000     11.760000
50%       43.420000     88.210000      8.940000     20.150000     21.760000
75%       68.550000    132.900000     15.970000     33.230000     34.117500
max      199.980000    399.130000     69.970000     89.440000     99.990000

               NH3            CO           SO2            O3       Benzene  \
count  16143.000000  21547.000000  20911.000000  20854.000000  18265.000000
mean      17.135633      1.371199     11.465261     33.567971      2.110920
std       11.773467      2.525758      9.863398     20.321697      3.352055
min        0.010000      0.000000      0.040000      0.010000      0.000000
25%        7.790000      0.520000      5.450000     18.530000      0.170000
50%       13.800000      0.850000      8.530000     30.050000      1.020000
75%       24.680000      1.290000     13.440000     44.837500      2.720000
max       50.000000     29.960000     59.650000    124.700000     29.970000

            Toluene           AQI
count  16283.000000  20281.000000
mean       6.221446    133.158178
std        8.517952     82.182518
min        0.000000     13.000000
25%        0.780000     76.000000
50%        2.870000    107.000000
75%        7.825000    162.000000
max       49.870000    592.000000
```

In [7]:

```
for j in range(2,14):
    b=pd.DataFrame();
    for i in range(len(city_uni)):
        a=(((ds.loc[ds["City"]==city_uni[i]]).iloc[:,j]))

        if(np.isnan(a.median()) or a.median()==0):
            a=a.fillna((ds.iloc[:,j].median()))
        else:
            a=a.fillna(a.mean())
        b=pd.concat([b,a],axis=0)
    ds[ds.columns[j]]=b;
ds.isna().sum()
```

Out[7]:

```
City             0
Date             0
PM2.5            0
PM10             0
NO               0
NO2              0
NOx              0
NH3              0
CO               0
SO2              0
O3               0
Benzene          0
Toluene          0
AQI              0
AQI_Bucket    2235
dtype: int64
```

In [8]:

```
import warnings
warnings.filterwarnings("ignore")
lrd=(ds[ds["AQI_Bucket"].isna()])
lrd1=ds[ds["AQI_Bucket"].notnull()]
```

```
x_train=lrd1.iloc[:,2:14]
y_train=lrd1.iloc[:,14]
x_test=lrd.iloc[:,2:14]
y_test=lrd.iloc[:,14]
lr=LogisticRegression()
lr.fit(x_train,y_train)
y_test1=pd.DataFrame(lr.predict(x_test))
y_test1.index=y_test.index
aqi=pd.concat([y_test1,y_train],axis=0)
aqi.sort_index()
ds["AQI_Bucket"]=aqi
aqi_uni=(ds.iloc[:,14]).unique()
for i in aqi_uni:
    print(i,len(ds.loc[ds["AQI_Bucket"]==i]))
print("the skewness is avoided here")
```

```
Poor 2723
Moderate 8906
Very Poor 1332
Severe 154
Satisfactory 8106
Good 1295
the skewness is avoided here
```

In [9]:

```
scale= StandardScaler()
x = ds.iloc[:,2:13]
nscale=x
y = ds.AQI
scaled_data = scale.fit_transform(x)
ds_f = pd.DataFrame(scaled_data)
ds.iloc[:,2:13]=scaled_data
```

In [10]:

```
a={"Severe":0,"Very Poor":1,"Poor":2,"Moderate":3,"Satisfactory":4,"Good":5}
ds["AQI_Bucket"]=ds["AQI_Bucket"].map(a)
print(ds.info())
df=ds.copy()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22516 entries, 1 to 29530
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        22516 non-null  object
 1   Date        22516 non-null  datetime64[ns]
 2   PM2.5       22516 non-null  float64
 3   PM10        22516 non-null  float64
 4   NO          22516 non-null  float64
 5   NO2         22516 non-null  float64
 6   NOx         22516 non-null  float64
 7   NH3         22516 non-null  float64
 8   CO          22516 non-null  float64
 9   SO2         22516 non-null  float64
 10  O3          22516 non-null  float64
 11  Benzene     22516 non-null  float64
 12  Toluene     22516 non-null  float64
 13  AQI         22516 non-null  float64
 14  AQI_Bucket  22516 non-null  int64
dtypes: datetime64[ns](1), float64(12), int64(1), object(1)
memory usage: 2.7+ MB
None
```

In [11]:
```
from sklearn.model_selection import train_test_split
x=ds.iloc[:,2:13]
y=ds.iloc[:,13]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [12]:
```
def calc_metr(y_test,pred):

   mae =round(sm.mean_absolute_error(y_test, pred), 3)
   mse=round(sm.mean_squared_error(y_test, pred), 3)
   rmse=round(np.sqrt(mse),3)
   r2= round(sm.r2_score(y_test, pred), 3)
   return([mae,mse,rmse,r2])

final_metric=[]
```

In [13]:
```
from sklearn import linear_model
model=linear_model.LinearRegression()
model.fit(x_train,y_train)
pred=model.predict(x_test)
lr_m=calc_metr(y_test,pred)
print("Performance metrics:",lr_m)
final_metric.append(lr_m)
```

```
Performance metrics: [23.1, 1206.233, 34.731, 0.806]
```

In [14]:
```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(x_train, y_train)
pred=model.predict(x_test)
lr_m=calc_metr(y_test,pred)
print("Performance metrics:",lr_m)
final_metric.append(lr_m)
```

```
Performance metrics: [23.1, 1206.233, 34.731, 0.806]
```

In [15]:
```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor(n_estimators=5,random_state=0)
regressor.fit(x_train,y_train)
pred2=regressor.predict(x_test)
lr_m=calc_metr(y_test,pred2)
print("The performance metrics are " ,lr_m)
print("The predicted AQI Index is :",pred2)
final_metric.append(lr_m)
```

```
The performance metrics are  [19.23, 1034.288, 32.16, 0.834]
The predicted AQI Index is : [105.2         87.4        153.6        ... 135.73558
606  99.2
 203.        ]
```

In [16]:
```
from sklearn.linear_model import Ridge
rr = Ridge(alpha=0.01)
rr.fit(x_train, y_train)
pred= rr.predict(x_test)
lr_m=calc_metr(y_test,pred)
print("Performance metrics:",lr_m)
final_metric.append(lr_m)
```

```
Performance metrics: [23.1, 1206.233, 34.731, 0.806]
```

In [17]:
```
from sklearn.linear_model import Lasso
model_lasso = Lasso(alpha=0.01)
model_lasso.fit(x_train, y_train)
pred= model_lasso.predict(x_test)
lr_m=calc_metr(y_test,pred)
```

```
#print(model_lasso.coef_)
print("Performance metrics:",lr_m)
final_metric.append(lr_m)
```

```
Performance metrics: [23.1, 1206.272, 34.731, 0.806]
```

In [18]:
```
from sklearn.linear_model import ElasticNet
model_enet = ElasticNet(alpha = 0.01)
model_enet.fit(x_train, y_train)
pred= model_enet.predict(x_test)
lr_m=calc_metr(y_test,pred)
print("Performance metrics:",lr_m)
final_metric.append(lr_m)
```

```
Performance metrics: [23.112, 1205.695, 34.723, 0.807]
```

In [19]:
```
print("-------------------------------------------------------------------------")
print("MODEL                    :  MAE        MSE       RMSE    R-Squared(R2) ")
print("-------------------------------------------------------------------------")
print("Linear Regression      :",final_metric[0][0],"   ",final_metric[0][1],"  ",
print("DecisionTree Regressor :",final_metric[1][0],"   ",final_metric[1][1],"  ",
print("RandomForest Regressor :",final_metric[2][0],"  ",final_metric[2][1],"  ",f
print("Ridge Regression       :",final_metric[3][0],"   ",final_metric[3][1],"  ",
print("Lasso Regression       :",final_metric[4][0],"   ",final_metric[4][1],"  ",
print("ElasticNet  Regression :",final_metric[5][0]," ",final_metric[5][1],"  ",fi
```

```
-------------------------------------------------------------------------
MODEL                    :  MAE        MSE       RMSE    R-Squared(R2)
-------------------------------------------------------------------------
Linear Regression      : 23.1     1206.233    34.731    0.806
DecisionTree Regressor : 23.1     1206.233    34.731    0.806
RandomForest Regressor : 19.23    1034.288    32.16     0.834
Ridge Regression       : 23.1     1206.233    34.731    0.806
Lasso Regression       : 23.1     1206.272    34.731    0.806
ElasticNet  Regression : 23.112   1205.695    34.723    0.807
```

In [20]:
```
predd=pd.DataFrame(pred2)
scaled_data = scale.fit_transform(predd)
pred_f = pd.DataFrame(scaled_data)
pred_f.index=x_test.index
a=pd.concat([x_test,pred_f],axis=1)
a=a.rename({0 : 'AQI'}, axis=1)
pr=lr.predict(a)
print("predicted AQI:  " )
print(pred2)
print("Predicted AQI_Bucket")
print(pr)
```

```
predicted AQI:
[105.2         87.4        153.6        ... 135.73558606  99.2
 203.        ]
Predicted AQI_Bucket
['Very Poor' 'Severe' 'Satisfactory' ... 'Good' 'Severe' 'Satisfactory']
```

In [21]:
```
print("AQI_Bucket counts of predicted values : ", np.unique(pr,return_counts=True)
```

```
AQI_Bucket counts of predicted values :  (array(['Good', 'Poor', 'Satisfactory',
'Severe', 'Very Poor'],
      dtype=object), array([ 903,    4, 1875, 1456,  266], dtype=int64))
```

In [ ]: