20104016

DEENA

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
```

Importing Datasets

In [2]:

```
df=pd.read_csv("madrid_2008.csv")
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.

226392 rows × 17 columns

Data Cleaning and Data Preprocessing

In [3]:

In [4]:

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3
',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25631 entries, 4 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25631 non-null  object 
 1   BEN      25631 non-null  float64
 2   CO       25631 non-null  float64
 3   EBE      25631 non-null  float64
 4   MXY      25631 non-null  float64
 5   NMHC     25631 non-null  float64
 6   NO_2     25631 non-null  float64
 7   NOx      25631 non-null  float64
 8   OXY      25631 non-null  float64
 9   O_3      25631 non-null  float64
 10  PM10     25631 non-null  float64
 11  PM25     25631 non-null  float64
 12  PXY      25631 non-null  float64
 13  SO_2     25631 non-null  float64
 14  TCH      25631 non-null  float64
 15  TOL      25631 non-null  float64
 16  station  25631 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

In [6]:
```
data=df[['CO' ,'station']]
```

Out[6]:

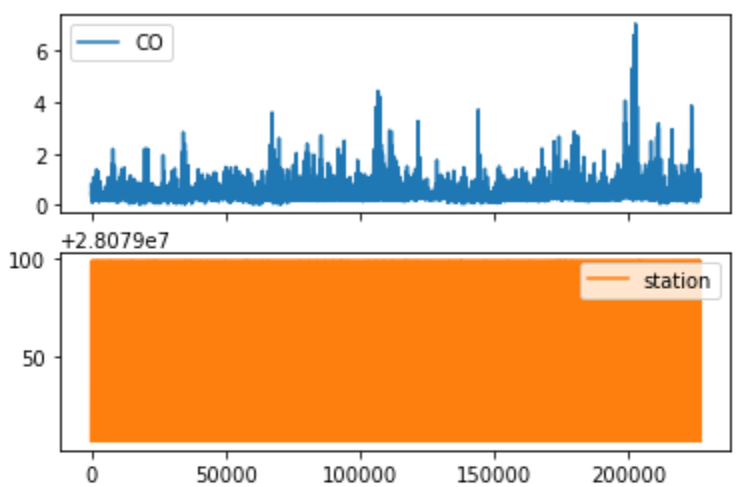
	CO	station
4	0.80	28079006
21	0.37	28079024
25	0.39	28079099
30	0.51	28079006
47	0.39	28079024
...	...	...
226362	0.35	28079024
226366	0.46	28079099
226371	0.53	28079006
226387	0.30	28079024
226391	0.36	28079099

25631 rows × 2 columns

Line chart

In [7]:
```
[illegible]
```

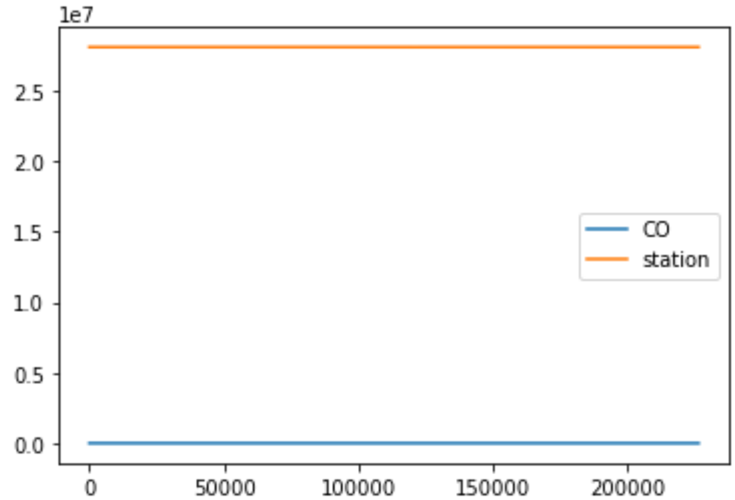
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:

Out[8]: <AxesSubplot:>

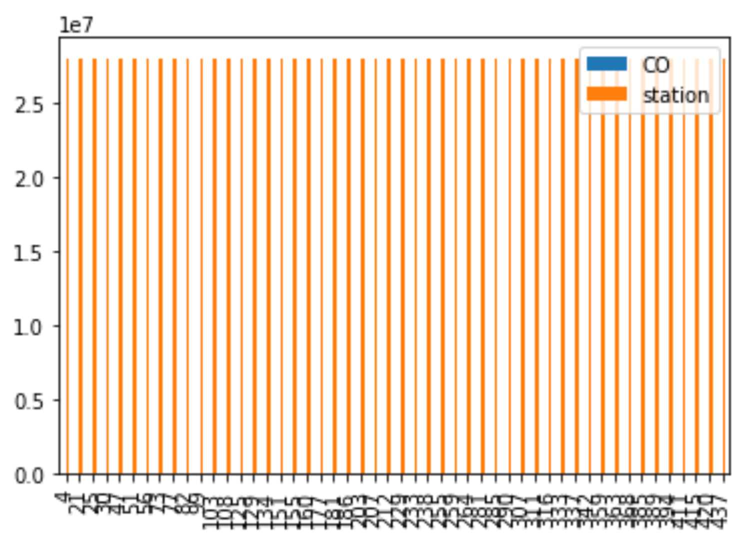


Bar chart

In [9]:

In [10]:

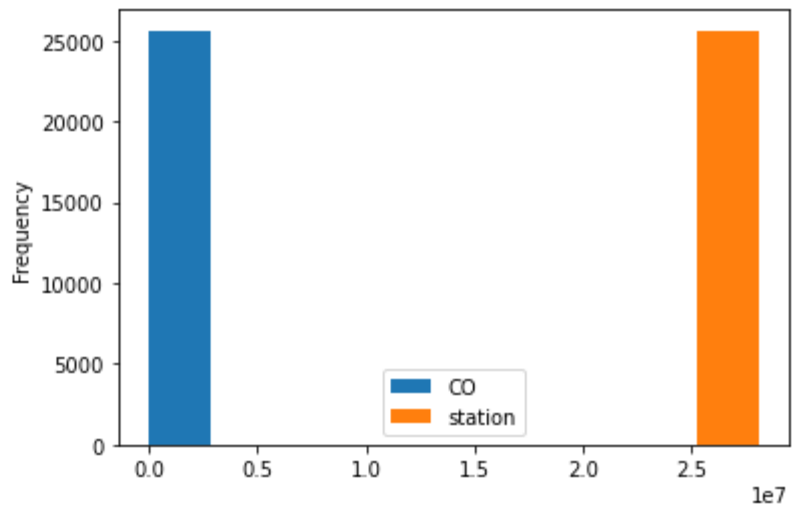
Out[10]: <AxesSubplot:>



Histogram

In [11]:

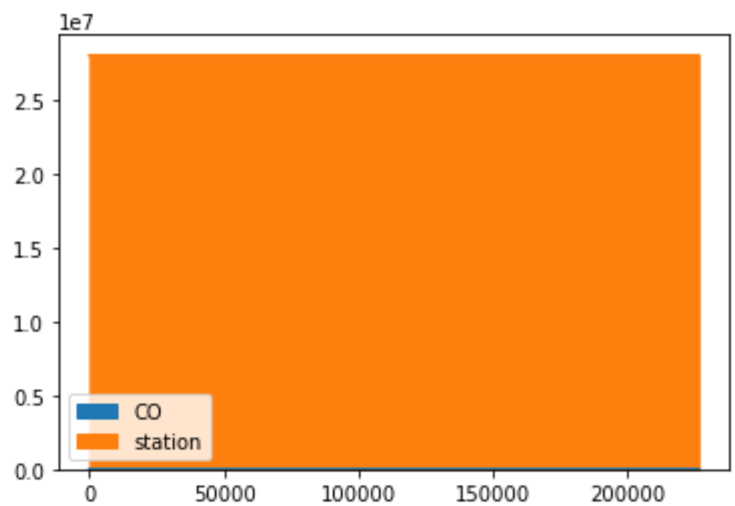
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:

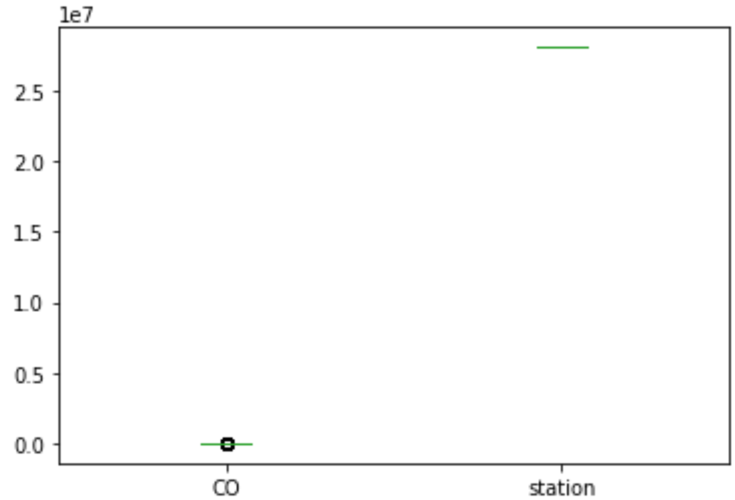
Out[12]: <AxesSubplot:>



Box chart

In [13]:
```
data.plot.box()
```

Out[13]: <AxesSubplot:>

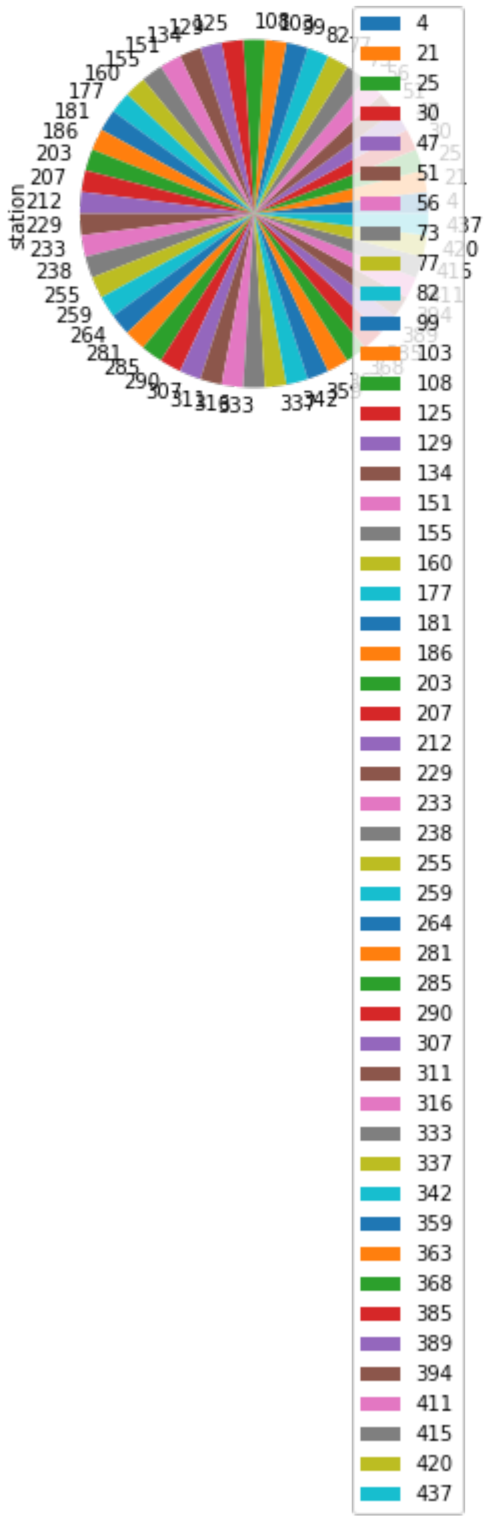


Pie chart

In [14]:

```
[illegible]
```

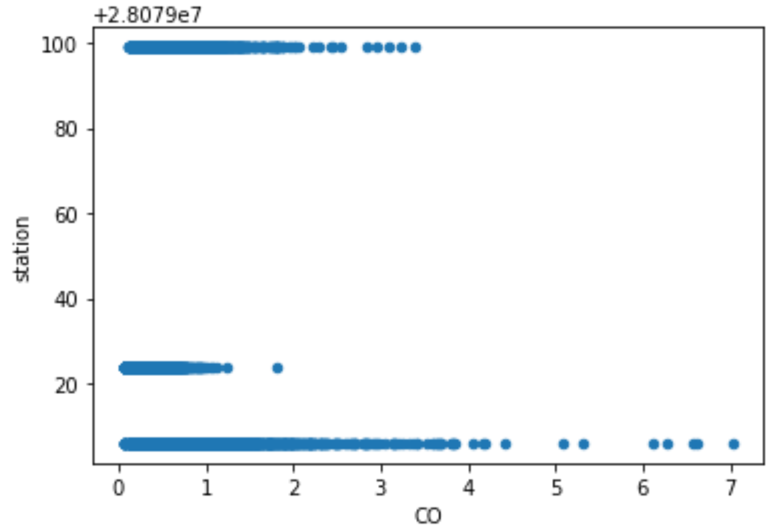
Out[14]: <AxesSubplot:ylabel='station'>



Scatter chart

In [15]:

Out[15]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [16]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25631 entries, 4 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25631 non-null  object 
 1   BEN      25631 non-null  float64
 2   CO       25631 non-null  float64
 3   EBE      25631 non-null  float64
 4   MXY      25631 non-null  float64
 5   NMHC     25631 non-null  float64
 6   NO_2     25631 non-null  float64
 7   NOx      25631 non-null  float64
 8   OXY      25631 non-null  float64
 9   O_3      25631 non-null  float64
 10  PM10     25631 non-null  float64
 11  PM25     25631 non-null  float64
 12  PXY      25631 non-null  float64
 13  SO_2     25631 non-null  float64
```

In [17]:
```
df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	256
mean	1.090541	0.440632	1.352355	2.446045	0.213323	54.225261	
std	1.146461	0.317853	1.118191	2.390023	0.123409	38.164647	1
min	0.100000	0.060000	0.170000	0.240000	0.000000	0.240000	
25%	0.430000	0.260000	0.740000	1.000000	0.130000	25.719999	
50%	0.750000	0.350000	1.000000	1.620000	0.190000	48.000000	
75%	1.320000	0.510000	1.580000	3.105000	0.270000	74.924999	1
max	27.230000	7.030000	26.740000	55.889999	1.760000	554.900024	20

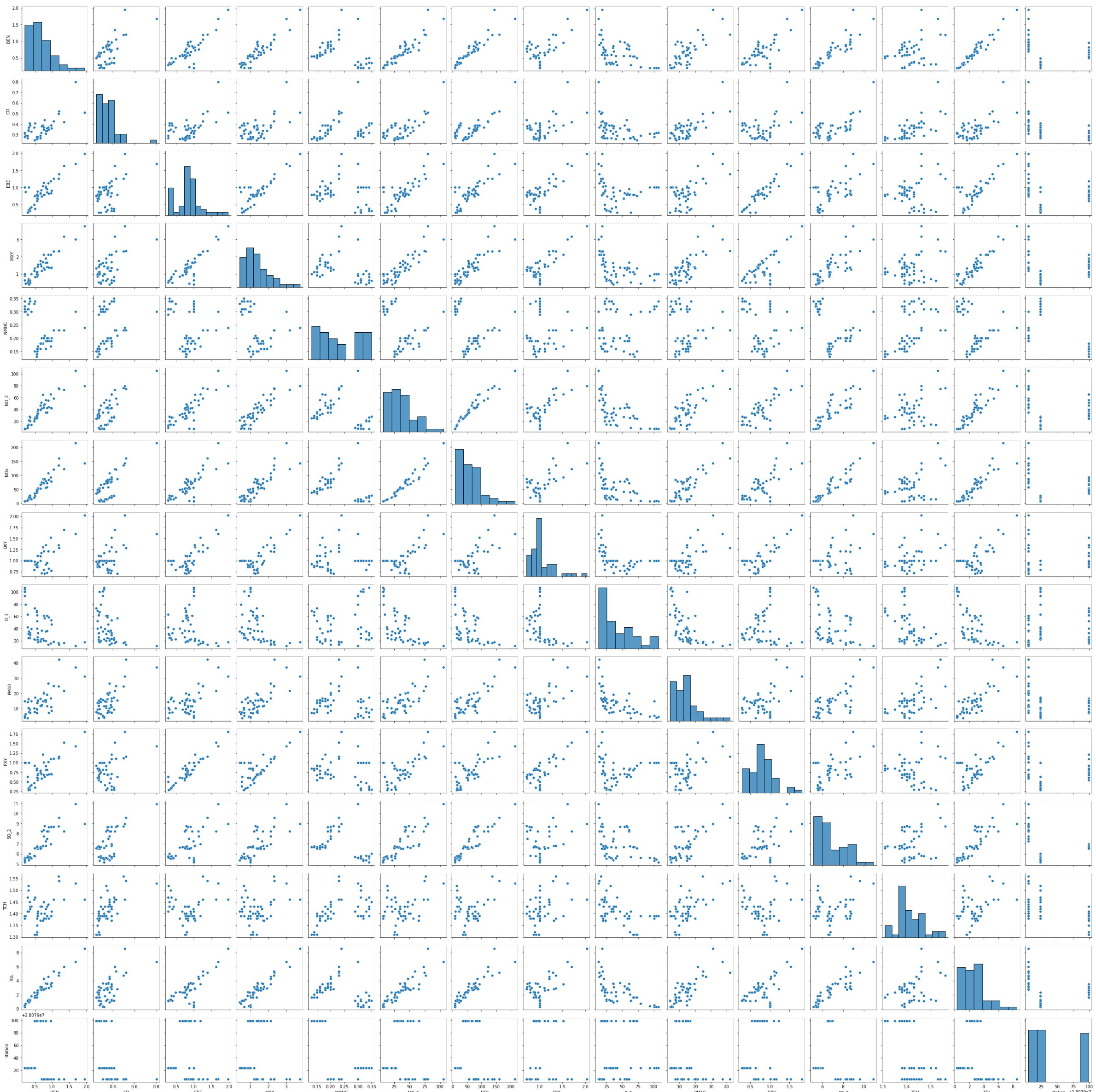
In [18]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [19]:

```
sns.pairplot(df1[0:50])
```

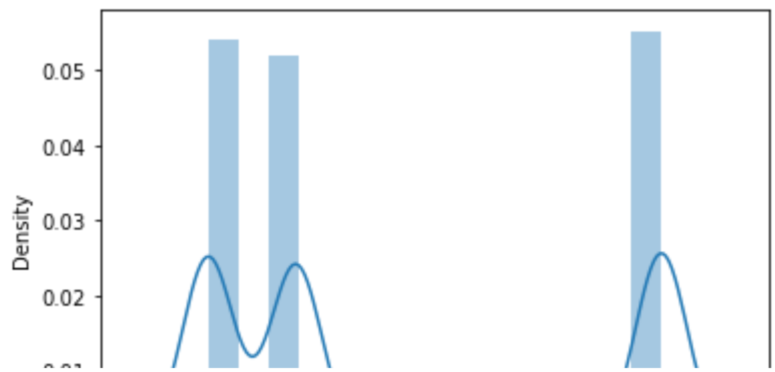
Out[19]: <seaborn.axisgrid.PairGrid at 0x29e417b2be0>



In [20]:

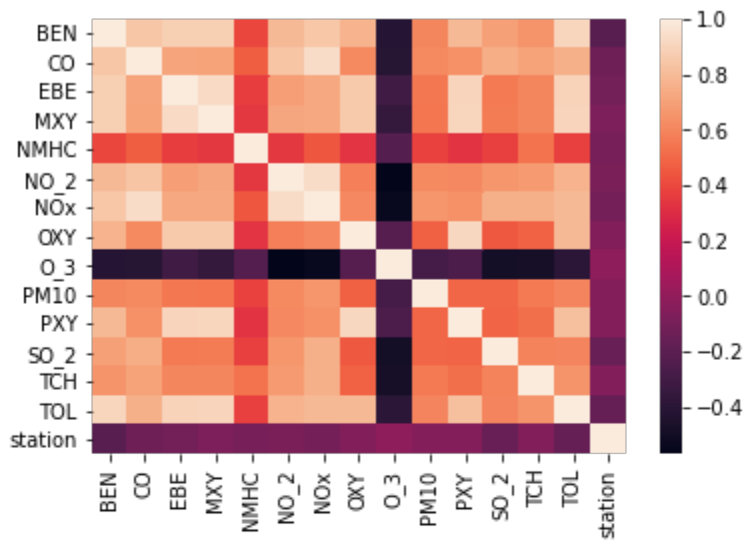
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[20]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [21]:

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:

```python
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
```

In [23]:

```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

Linear Regression

In [24]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

Out[24]: LinearRegression()

In [25]:

Out[25]: 28079031.439269386

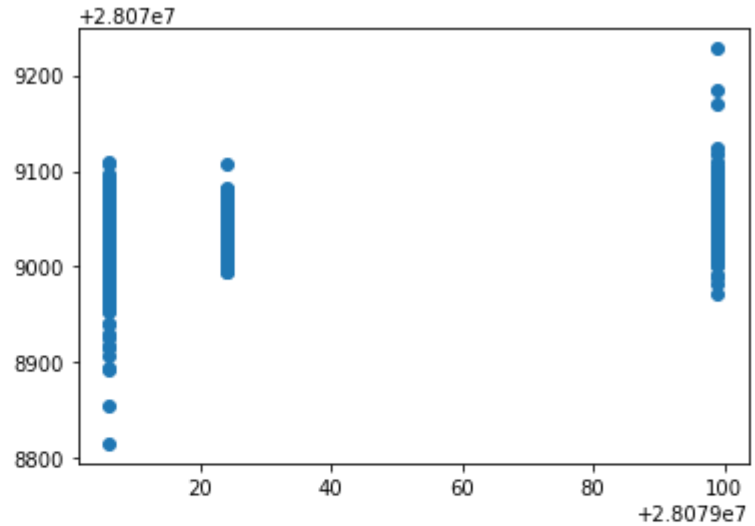
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[26]:

	Co-efficient
BEN	-25.935634
CO	-2.878921
EBE	-0.601321
MXY	7.361963
NMHC	-24.981870
NO_2	-0.007740
NOx	0.118970
OXY	3.609772
O_3	-0.122802
PM10	0.141612
PXY	2.563398
SO_2	-0.555249
TCH	18.891868
TOL	-1.891016

In [27]:
```
prediction =lr.predict(x_test)
```

Out[27]: <matplotlib.collections.PathCollection at 0x29e50178700>



ACCURACY

In [28]:

Out[28]: 0.14589220039799633

In [29]:

Out[29]: 0.14249828065737014

Ridge and Lasso

In [30]:

In [31]:
```
rr=Ridge(alpha=10)
```

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:

Out[32]: 0.14580956282768287

In [33]:

Out[33]: 0.14247468266116436

In [34]:
```
la=Lasso(alpha=10)
```

Out[34]: Lasso(alpha=10)

In [35]:

Out[35]: 0.04072393966873966

Accuracy(Lasso)

In [36]:

Out[36]: 0.04357378194132566

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
```

Out[37]: ElasticNet()

In [38]:

Out[38]:
```
array([-4.64736325, -0.        ,  0.        ,  3.28696488, -0.        ,
        0.077272  ,  0.01721591,  1.48519662, -0.14541509,  0.14244237,
        1.67756072, -0.91206623,  0.        , -2.5707591 ])
```

In [39]:

Out[39]: 28079056.107349645

In [40]:

In [41]:

Out[41]: 0.096440753633108

Evaluation Metrics

In [42]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
```

```
35.69170876745575
1483.646482465846
38.51813186625029
```

Logistic Regression

In [43]:

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
```

In [45]:

Out[45]: (25631, 14)

In [46]:

Out[46]: (25631,)

In [47]:

In [48]:

In [49]:
```
logr=LogisticRegression(max_iter=10000)
```

Out[49]: LogisticRegression(max_iter=10000)

In [50]:

In [51]:
```
prediction=logr.predict(observation)
```

[28079099]

In [52]:

Out[52]: array([28079006, 28079024, 28079099], dtype=int64)

In [53]:

Out[53]: 0.794194530061254

In [54]:

Out[54]: 8.321803242555043e-09

In [55]:

Out[55]: array([[8.32180324e-09, 1.19114634e-13, 9.99999992e-01]])

Random Forest

In [56]:

In [57]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[57]: RandomForestClassifier()

In [58]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
```

In [59]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="ac
```

Out[59]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [60]:

Out[60]:
```
0.8505095074715543
```

In [61]:

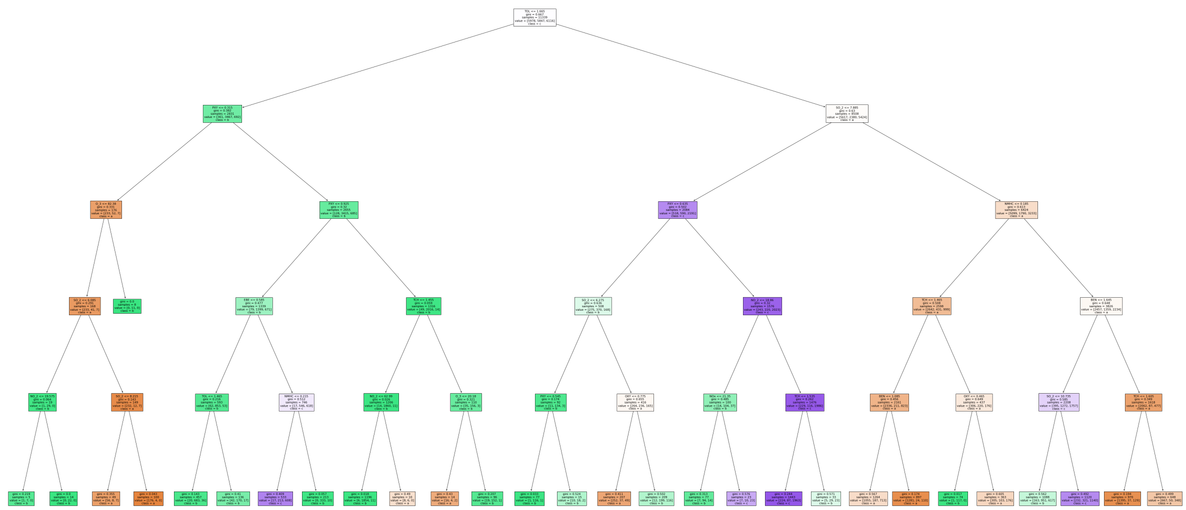
In [62]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
```

Out[62]:
```
[Text(2012.7857142857142, 1993.2, 'TOL <= 1.665\ngini = 0.667\nsamples = 1133
9\nvalue = [5978, 5847, 6116]\nclass = c'),
 Text(836.9999999999999, 1630.8000000000002, 'PXY <= 0.315\ngini = 0.382\nsam
ples = 2831\nvalue = [361, 3467, 692]\nclass = b'),
 Text(398.57142857142856, 1268.4, 'O_3 <= 82.38\ngini = 0.331\nsamples = 176\
nvalue = [233, 52, 7]\nclass = a'),
 Text(318.85714285714283, 906.0, 'SO_2 <= 6.085\ngini = 0.291\nsamples = 168\
nvalue = [233, 41, 7]\nclass = a'),
 Text(159.42857142857142, 543.5999999999999, 'NO_2 <= 19.575\ngini = 0.064\ns
amples = 19\nvalue = [1, 29, 0]\nclass = b'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.219\nsamples = 5\nvalu
e = [1, 7, 0]\nclass = b'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.0\nsamples = 14\nvalue
= [0, 22, 0]\nclass = b'),
 Text(478.2857142857142, 543.5999999999999, 'SO_2 <= 8.215\ngini = 0.143\nsam
ples = 149\nvalue = [232, 12, 7]\nclass = a'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.355\nsamples = 49\nva
lue = [56, 8, 7]\nclass = a'),
 Text(558.0, 181.19999999999982, 'gini = 0.043\nsamples = 100\nvalue = [176,
4, 0]\nclass = a'),
 Text(478.2857142857142, 906.0, 'gini = 0.0\nsamples = 8\nvalue = [0, 11, 0]\
nclass = b'),
 Text(1275.4285714285713, 1268.4, 'PXY <= 0.925\ngini = 0.32\nsamples = 2655\
nvalue = [128, 3415, 685]\nclass = b'),
 Text(956.5714285714284, 906.0, 'EBE <= 0.585\ngini = 0.477\nsamples = 1339\n
value = [79, 1399, 671]\nclass = b'),
 Text(797.1428571428571, 543.5999999999999, 'TOL <= 1.465\ngini = 0.216\nsamp
les = 593\nvalue = [62, 853, 53]\nclass = b'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.143\nsamples = 457\nva
lue = [20, 683, 36]\nclass = b'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.41\nsamples = 136\nval
ue = [42, 170, 17]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'NMHC <= 0.215\ngini = 0.512\nsamples = 746\
nvalue = [17, 546, 618]\nclass = c'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.409\nsamples = 533\nv
alue = [17, 213, 608]\nclass = c'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.057\nsamples = 213\nv
alue = [0, 333, 10]\nclass = b'),
 Text(1594.2857142857142, 906.0, 'TCH <= 1.455\ngini = 0.059\nsamples = 1316\
nvalue = [49, 2016, 14]\nclass = b'),
 Text(1434.8571428571427, 543.5999999999999, 'NO_2 <= 62.99\ngini = 0.026\nsa
mples = 1206\nvalue = [14, 1860, 11]\nclass = b'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.018\nsamples = 1196\nv
alue = [6, 1854, 11]\nclass = b'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.49\nsamples = 10\nval
ue = [8, 6, 0]\nclass = a'),
 Text(1753.7142857142856, 543.5999999999999, 'O_3 <= 20.18\ngini = 0.321\nsam
ples = 110\nvalue = [35, 156, 3]\nclass = b'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.43\nsamples = 14\nval
ue = [16, 4, 2]\nclass = a'),
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.207\nsamples = 96\nva
lue = [19, 152, 1]\nclass = b'),
```

```
 Text(3188.5714285714284, 1630.8000000000002, 'SO_2 <= 7.885\ngini = 0.63\nsa
mples = 8508\nvalue = [5617, 2380, 5424]\nclass = a'),
 Text(2550.8571428571427, 1268.4, 'PXY <= 0.635\ngini = 0.502\nsamples = 208
4\nvalue = [518, 590, 2191]\nclass = c'),
 Text(2232.0, 906.0, 'SO_2 <= 6.275\ngini = 0.636\nsamples = 508\nvalue = [27
5, 370, 168]\nclass = b'),
 Text(2072.5714285714284, 543.5999999999999, 'PXY <= 0.545\ngini = 0.174\nsam
ples = 92\nvalue = [11, 134, 3]\nclass = b'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.033\nsamples = 77\nva
lue = [1, 116, 1]\nclass = b'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.524\nsamples = 15\nval
ue = [10, 18, 2]\nclass = b'),
 Text(2391.428571428571, 543.5999999999999, 'OXY <= 0.775\ngini = 0.655\nsamp
les = 416\nvalue = [264, 236, 165]\nclass = a'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.411\nsamples = 207\nv
alue = [252, 37, 49]\nclass = a'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.502\nsamples = 209\nva
lue = [12, 199, 116]\nclass = b'),
 Text(2869.7142857142853, 906.0, 'NO_2 <= 18.86\ngini = 0.32\nsamples = 1576\
nvalue = [243, 220, 2023]\nclass = c'),
 Text(2710.285714285714, 543.5999999999999, 'NOx <= 21.35\ngini = 0.485\nsamp
les = 100\nvalue = [14, 104, 37]\nclass = b'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.313\nsamples = 77\nva
lue = [7, 94, 14]\nclass = b'),
 Text(2790.0, 181.19999999999982, 'gini = 0.576\nsamples = 23\nvalue = [7, 1
0, 23]\nclass = c'),
 Text(3029.142857142857, 543.5999999999999, 'TCH <= 1.535\ngini = 0.262\nsamp
les = 1476\nvalue = [229, 116, 1986]\nclass = c'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.244\nsamples = 1443\nv
alue = [224, 87, 1963]\nclass = c'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.571\nsamples = 33\nva
lue = [5, 29, 23]\nclass = b'),
 Text(3826.2857142857138, 1268.4, 'NMHC <= 0.185\ngini = 0.613\nsamples = 642
4\nvalue = [5099, 1790, 3233]\nclass = a'),
 Text(3507.428571428571, 906.0, 'TCH <= 1.465\ngini = 0.508\nsamples = 2598\n
value = [2642, 431, 999]\nclass = a'),
 Text(3347.9999999999995, 543.5999999999999, 'BEN <= 1.085\ngini = 0.456\nsam
ples = 2161\nvalue = [2336, 211, 823]\nclass = a'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.567\nsamples = 1264\nv
alue = [1055, 187, 713]\nclass = a'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.174\nsamples = 897\nv
alue = [1281, 24, 110]\nclass = a'),
 Text(3666.8571428571427, 543.5999999999999, 'OXY <= 0.465\ngini = 0.649\nsam
ples = 437\nvalue = [306, 220, 176]\nclass = a'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.017\nsamples = 74\nval
ue = [1, 117, 0]\nclass = b'),
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.605\nsamples = 363\nv
alue = [305, 103, 176]\nclass = a'),
 Text(4145.142857142857, 906.0, 'BEN <= 1.645\ngini = 0.648\nsamples = 3826\n
value = [2457, 1359, 2234]\nclass = a'),
 Text(3985.7142857142853, 543.5999999999999, 'SO_2 <= 10.735\ngini = 0.585\ns
amples = 2208\nvalue = [395, 1272, 1757]\nclass = c'),
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.562\nsamples = 1088\n
value = [163, 951, 617]\nclass = b'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.492\nsamples = 1120\nv
alue = [232, 321, 1140]\nclass = c'),
```

```
 Text(4304.571428571428, 543.5999999999999, 'TCH <= 1.605\ngini = 0.349\nsamp
les = 1618\nvalue = [2062, 87, 477]\nclass = a'),
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.194\nsamples = 970\nva
lue = [1395, 37, 129]\nclass = a'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.499\nsamples = 648\nva
lue = [667, 50, 348]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression :0.14249828065737014

Ridge Regression : 0.04072393966873966

Lasso Regression : 0.04357378194132566

ElasticNet Regression : 0.096440753633108

Logistic Regression : 0.794194530061254

Random Forest :0.8505095074715543

Random Forest is suitable for this dataset