Importing Libraries

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

Importing Datasets

In [2]:

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

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH	TOL	
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.889999	10.40	NaN	8.98	NaN	NaN	28
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.040001	NaN	NaN	5.85	NaN	NaN	28
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.270000	NaN	NaN	6.95	NaN	NaN	28
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.850000	NaN	NaN	5.96	NaN	NaN	28
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	37.160000	21.90	1.43	10.92	1.53	6.67	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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	5.450000	5.15	1.86	9.68	1.23	2.05	28
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.020000	NaN	NaN	8.90	NaN	NaN	28
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.540001	13.91	NaN	7.00	1.56	0.60	28
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.910000	NaN	NaN	8.02	1.57	2.97	28
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	12.690000	11.42	1.98	8.74	1.43	4.15	28

226392 rows × 17 columns

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.dropna()
```

In [4]:
```
df.columns
```

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

```
<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]:
```
1 data=df[['CO' ,'station']]
2 data
```

Out[6]:

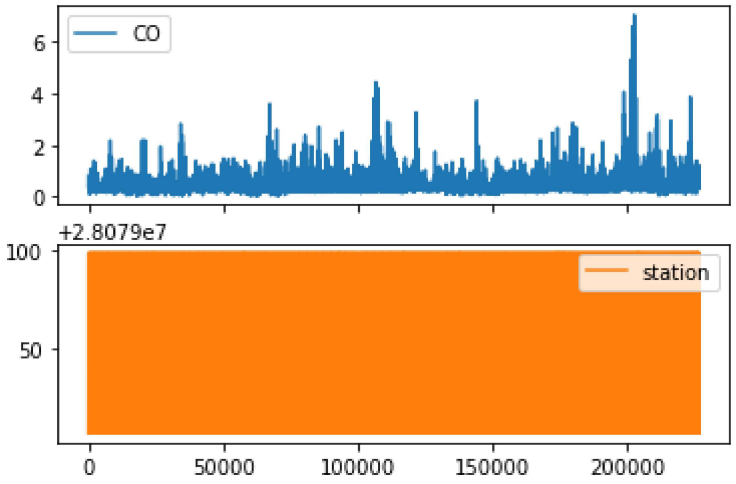
	CO	station
4	0.80	28079006
21	0.37	28079024
25	0.39	28079099
30	0.51	28079006
47	0.39	28079024
...	...	...
226362	0.35	28079024
226366	0.46	28079099
226371	0.53	28079006
226387	0.30	28079024
226391	0.36	28079099

25631 rows × 2 columns

Line chart

In [7]:
```
1 data.plot.line(subplots=True)
```

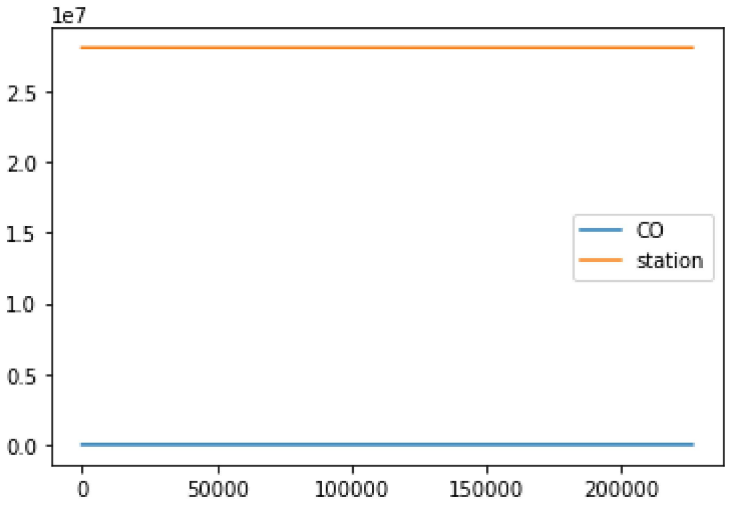
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:
```
1 data.plot.line()
```

Out[8]:
```
<AxesSubplot:>
```

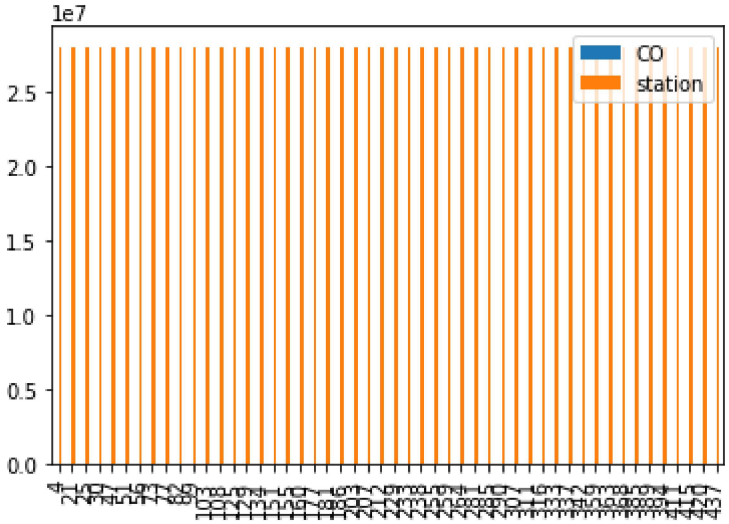


Bar chart

In [9]:
```
1 b=data[0:50]
```

In [10]:
```
1 b.plot.bar()
```

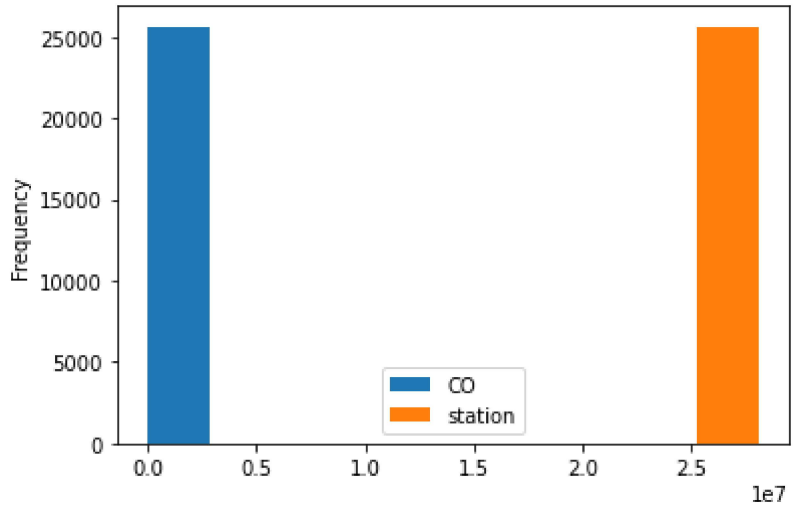
Out[10]: <AxesSubplot:>



Histogram

In [11]:
```
1 data.plot.hist()
```

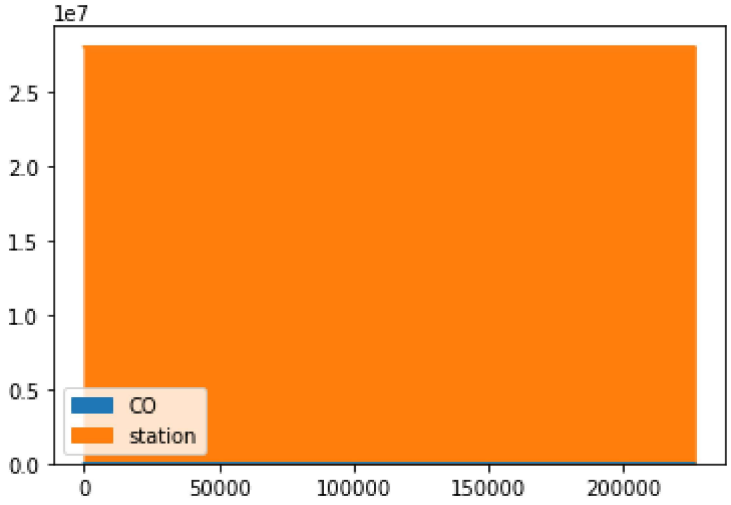
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:
```
1 data.plot.area()
```

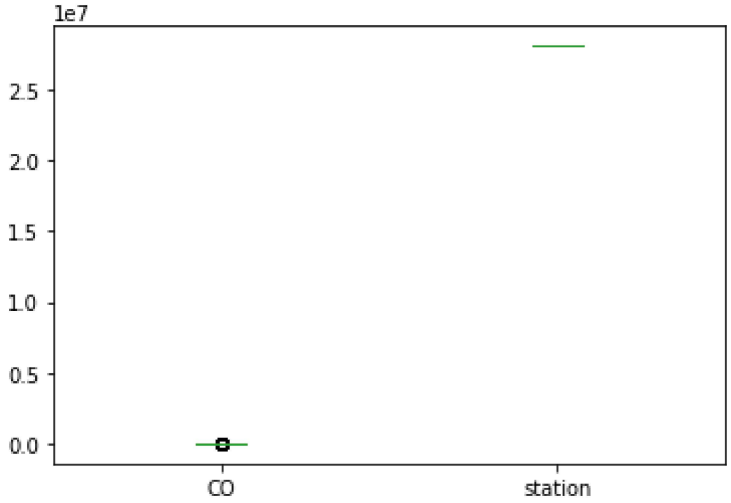
Out[12]: <AxesSubplot:>



Box chart

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

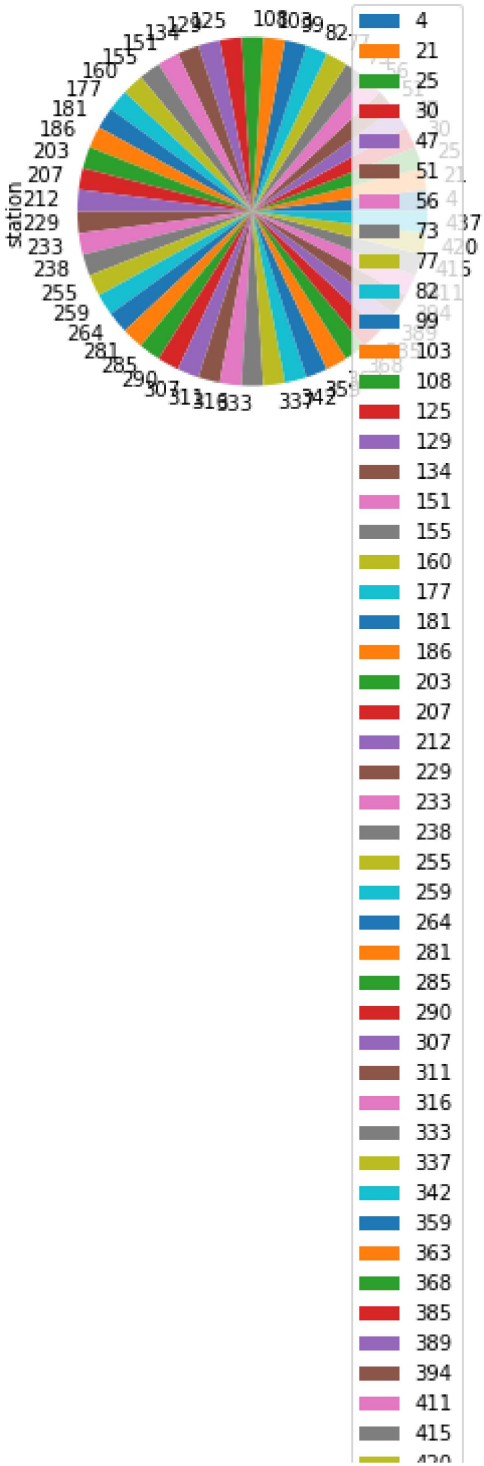
Out[13]: <AxesSubplot:>



Pie chart

In [14]:
```
1 b.plot.pie(y='station' )
```

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

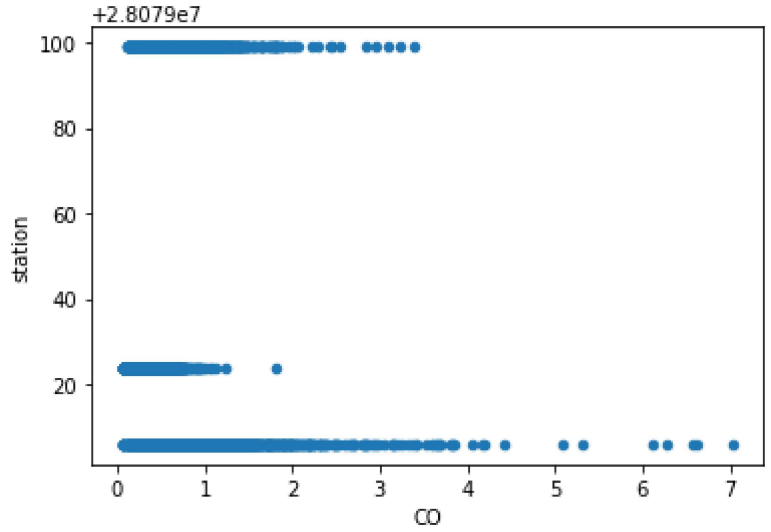

station
4
21
25
30
47
51
56
73
77
82
99
103
108
125
129
134
151
155
160
177
181
186
203
207
212
229
233
238
255
259
264
281
285
290
307
311
316
333
337
342
359
363
368
385
389
394
411
415
420

420
437

Scatter chart

In [15]:
```
1 data.plot.scatter(x='CO' ,y='station')
```

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



In [16]:
```
1 df.info()
```

```
<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]:
```
1 df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000
mean	1.090541	0.440632	1.352355	2.446045	0.213323	54.225261	98.007732	1.479964	41.602705
std	1.146461	0.317853	1.118191	2.390023	0.123409	38.164647	101.448238	1.258928	29.075860
min	0.100000	0.060000	0.170000	0.240000	0.000000	0.240000	2.110000	0.140000	1.530000
25%	0.430000	0.260000	0.740000	1.000000	0.130000	25.719999	32.635000	0.870000	16.905000
50%	0.750000	0.350000	1.000000	1.620000	0.190000	48.000000	71.110001	1.000000	37.480000
75%	1.320000	0.510000	1.580000	3.105000	0.270000	74.924999	131.550003	1.760000	60.240002
max	27.230000	7.030000	26.740000	55.889999	1.760000	554.900024	2004.000000	28.020000	201.000000

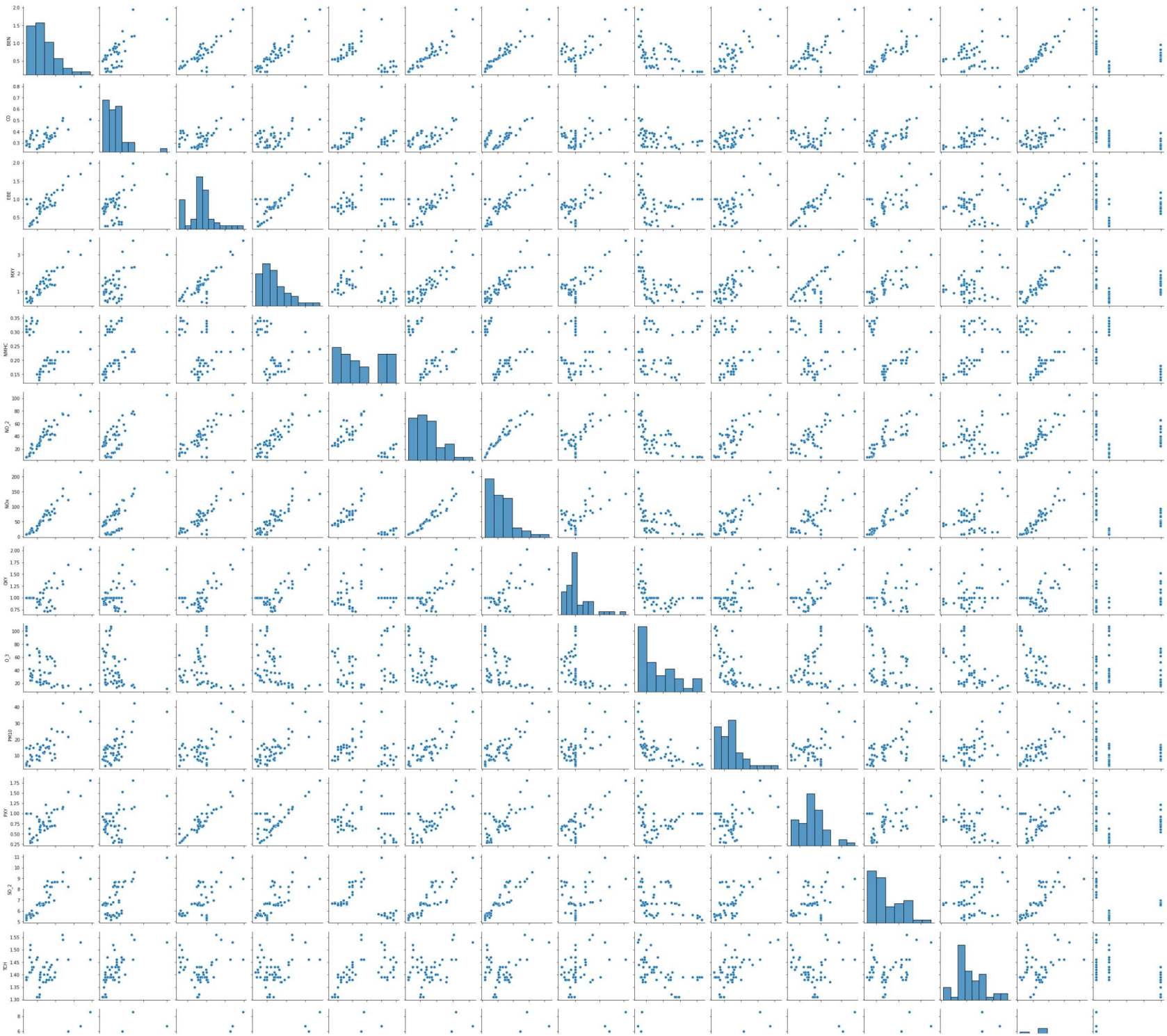
In [18]:

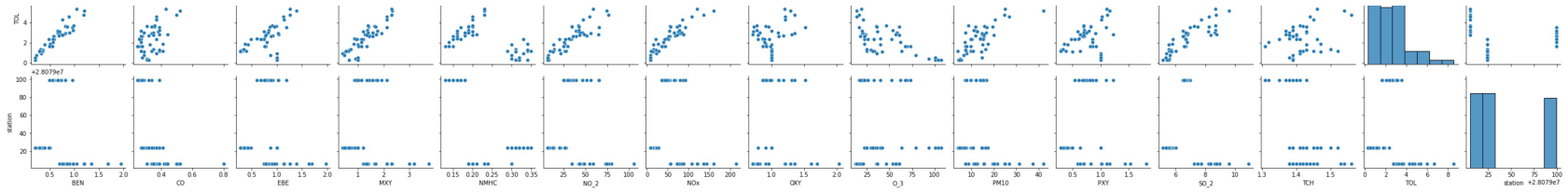
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [19]:
```
1 sns.pairplot(df1[0:50])
```

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x223432809d0>
```

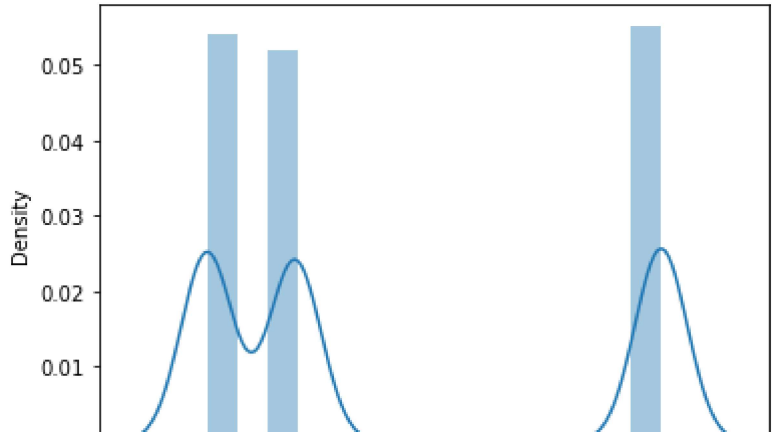

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



In [20]:
```
1 sns.distplot(df1['station'])
```

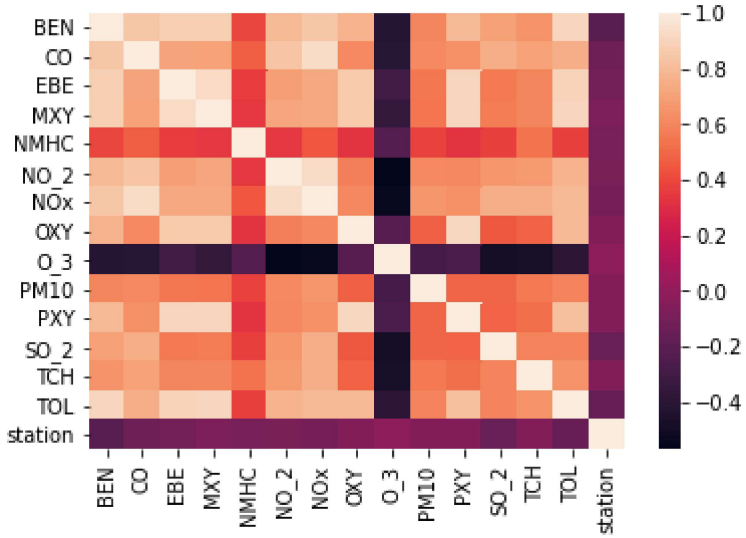
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function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

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



In [21]:
```
1 sns.heatmap(df1.corr())
```

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:
```
1 x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
2        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
3 y=df['station']
```

In [23]:
```
1 from sklearn.model_selection import train_test_split
2 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()
lr.fit(x_train,y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
lr.intercept_
```

Out[25]:
```
28079033.40354286
```

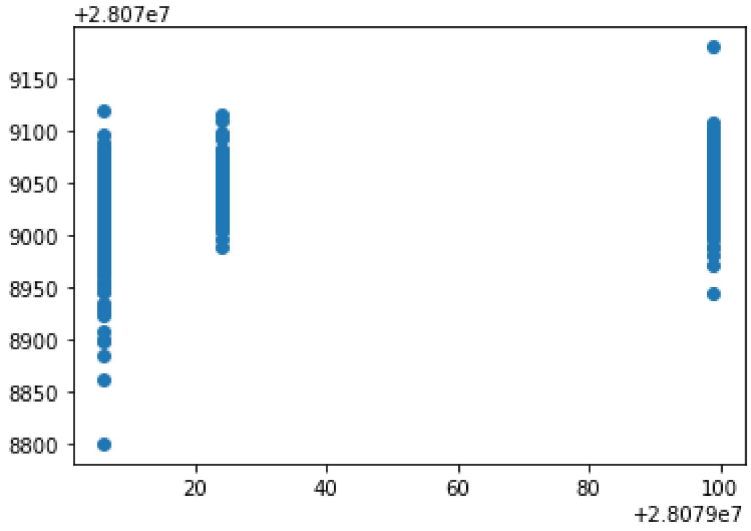
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-24.711386
CO	-0.770521
EBE	-0.352426
MXY	7.317558
NMHC	-28.782289
NO_2	-0.028778
NOx	0.124037
OXY	4.077702
O_3	-0.129650
PM10	0.139485
PXY	2.243872
SO_2	-0.638120
TCH	18.449325
TOL	-2.162716

In [27]:
```
1 prediction =lr.predict(x_test)
2 plt.scatter(y_test,prediction)
```

Out[27]: <matplotlib.collections.PathCollection at 0x22351c7c700>



ACCURACY

In [28]:
```
1 lr.score(x_test,y_test)
```

Out[28]: 0.15060765473352178

In [29]:
```
1 lr.score(x_train,y_train)
```

Out[29]: 0.14037527802328775

Ridge and Lasso

In [30]:
```
1 from sklearn.linear_model import Ridge,Lasso
```

In [31]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:
```
rr.score(x_test,y_test)
```

Out[32]: 0.15064143062649826

In [33]:
```
rr.score(x_train,y_train)
```

Out[33]: 0.1403481649670576

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[34]: Lasso(alpha=10)

Accuracy(Lasso)

In [35]:
```
la.score(x_train,y_train)
```

Out[35]: 0.0401780815850018

In [36]:
```
la.score(x_test,y_test)
```

Out[36]: 0.04437895589711793

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

Out[37]: ElasticNet()

In [38]:
```
1 en.coef_
```

Out[38]:
```
array([-4.2719613 , -0.        ,  0.        ,  3.44178437, -0.        ,
        0.06227284,  0.02895985,  1.65136023, -0.1442674 ,  0.13481328,
        1.71488323, -0.96948607,  0.        , -2.794083  ])
```

In [39]:
```
1 en.intercept_
```

Out[39]:
```
28079056.6075464
```

In [40]:
```
1 prediction=en.predict(x_test)
```

In [41]:
```
1 en.score(x_test,y_test)
```

Out[41]:
```
0.0944718215443624
```

Evaluation Metrics

In [42]:
```
1 from sklearn import metrics
2 print(metrics.mean_absolute_error(y_test,prediction))
3 print(metrics.mean_squared_error(y_test,prediction))
4 print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
35.783832700928414
1494.5733081484284
38.659711692515614
```

Logistic Regression

In [43]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [44]:
```
1 feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
2        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
3 target_vector=df[ 'station']
```

In [45]:
```
1 feature_matrix.shape
```

Out[45]: (25631, 14)

In [46]:
```
1 target_vector.shape
```

Out[46]: (25631,)

In [47]:
```
1 from sklearn.preprocessing import StandardScaler
```

In [48]:
```
1 fs=StandardScaler().fit_transform(feature_matrix)
```

In [49]:
```
1 logr=LogisticRegression(max_iter=10000)
2 logr.fit(fs,target_vector)
```

Out[49]: LogisticRegression(max_iter=10000)

In [50]:
```
1 observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

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

```
[28079099]
```

In [52]:
```
1 logr.classes_
```

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

In [53]:
```
1 logr.score(fs,target_vector)
```

Out[53]: 0.794194530061254

In [54]:
```
1 logr.predict_proba(observation)[0][0]
```

Out[54]: 8.321803242555043e-09

In [55]:
```
1 logr.predict_proba(observation)
```

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

Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestClassifier
```

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="accuracy")
grid_search.fit(x_train,y_train)
```

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]:
```
grid_search.best_score_
```

Out[60]:
```
0.854244619756438
```

In [61]:
```
rfc_best=grid_search.best_estimator_
```

In [62]:

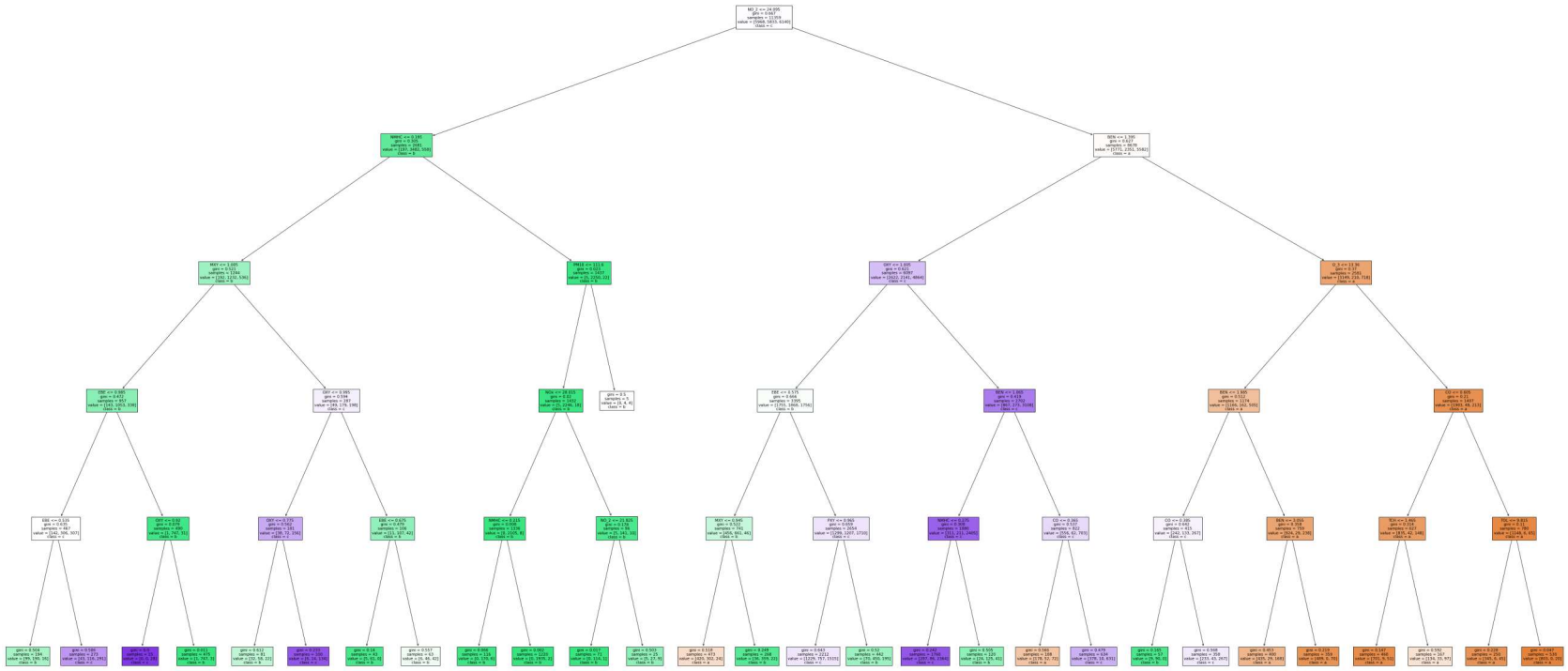
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d'],filled=True)
```

Out[62]: [Text(2172.2142857142853, 1993.2, 'NO_2 <= 24.095\ngini = 0.667\nsamples = 11359\nvalue = [5968, 5833, 6140]\nclass = c'),
 Text(1155.8571428571427, 1630.8000000000002, 'NMHC <= 0.195\ngini = 0.305\nsamples = 2681\nvalue = [197, 3482, 558]\nclass = b'),
 Text(637.7142857142857, 1268.4, 'MXY <= 1.005\ngini = 0.521\nsamples = 1244\nvalue = [192, 1232, 536]\nclass = b'),
 Text(318.85714285714283, 906.0, 'EBE <= 0.985\ngini = 0.472\nsamples = 957\nvalue = [143, 1053, 338]\nclass = b'),
 Text(159.42857142857142, 543.5999999999999, 'EBE <= 0.535\ngini = 0.635\nsamples = 467\nvalue = [142, 306, 307]\nclass = c'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.504\nsamples = 194\nvalue = [99, 190, 16]\nclass = b'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.506\nsamples = 273\nvalue = [43, 116, 291]\nclass = c'),
 Text(478.2857142857142, 543.5999999999999, 'OXY <= 0.92\ngini = 0.079\nsamples = 490\nvalue = [1, 747, 31]\nclass = b'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.0\nsamples = 15\nvalue = [0, 0, 28]\nclass = c'),
 Text(558.0, 181.19999999999982, 'gini = 0.011\nsamples = 475\nvalue = [1, 747, 3]\nclass = b'),
 Text(956.5714285714284, 906.0, 'OXY <= 0.995\ngini = 0.594\nsamples = 287\nvalue = [49, 179, 198]\nclass = c'),
 Text(797.1428571428571, 543.5999999999999, 'OXY <= 0.775\ngini = 0.562\nsamples = 181\nvalue = [38, 72, 156]\nclass = c'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.612\nsamples = 81\nvalue = [32, 58, 22]\nclass = b'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.233\nsamples = 100\nvalue = [6, 14, 134]\nclass = c'),
 Text(1116.0, 543.5999999999999, 'EBE <= 0.675\ngini = 0.479\nsamples = 106\nvalue = [11, 107, 42]\nclass = b'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.14\nsamples = 43\nvalue = [5, 61, 0]\nclass = b'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.557\nsamples = 63\nvalue = [6, 46, 42]\nclass = b'),
 Text(1673.9999999999998, 1268.4, 'PM10 <= 111.6\ngini = 0.023\nsamples = 1437\nvalue = [5, 2250, 22]\nclass = b'),
 Text(1594.2857142857142, 906.0, 'NOx <= 28.015\ngini = 0.02\nsamples = 1432\nvalue = [5, 2246, 18]\nclass = b'),
 Text(1434.8571428571427, 543.5999999999999, 'NMHC <= 0.215\ngini = 0.008\nsamples = 1336\nvalue = [0, 2105, 8]\nclass = b'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.066\nsamples = 116\nvalue = [0, 170, 6]\nclass = b'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.002\nsamples = 1220\nvalue = [0, 1935, 2]\nclass = b'),
 Text(1753.7142857142856, 543.5999999999999, 'NO_2 <= 21.825\ngini = 0.178\nsamples = 96\nvalue = [5, 141, 10]\nclass = b'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.017\nsamples = 71\nvalue = [0, 114, 1]\nclass = b'),
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.503\nsamples = 25\nvalue = [5, 27, 9]\nclass = b'),
 Text(1753.7142857142856, 906.0, 'gini = 0.5\nsamples = 5\nvalue = [0, 4, 4]\nclass = b'),

```
 Text(3188.5714285714284, 1630.8000000000002, 'BEN <= 1.395\ngini = 0.627\nsamples = 8678\nvalue = [5771, 23
51, 5582]\nclass = a'),
 Text(2550.8571428571427, 1268.4, 'OXY <= 1.005\ngini = 0.621\nsamples = 6097\nvalue = [2622, 2141, 4864]\nc
lass = c'),
 Text(2232.0, 906.0, 'EBE <= 0.575\ngini = 0.666\nsamples = 3395\nvalue = [1755, 1868, 1756]\nclass = b'),
 Text(2072.5714285714284, 543.5999999999999, 'MXY <= 0.945\ngini = 0.522\nsamples = 741\nvalue = [456, 661,
46]\nclass = b'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.518\nsamples = 473\nvalue = [420, 302, 24]\nclass =
a'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.249\nsamples = 268\nvalue = [36, 359, 22]\nclass =
b'),
 Text(2391.428571428571, 543.5999999999999, 'PXY <= 0.965\ngini = 0.659\nsamples = 2654\nvalue = [1299, 120
7, 1710]\nclass = c'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.643\nsamples = 2212\nvalue = [1229, 757, 1515]\nclas
s = c'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.52\nsamples = 442\nvalue = [70, 450, 195]\nclass =
b'),
 Text(2869.7142857142853, 906.0, 'BEN <= 1.065\ngini = 0.419\nsamples = 2702\nvalue = [867, 273, 3108]\nclas
s = c'),
 Text(2710.285714285714, 543.5999999999999, 'NMHC <= 0.275\ngini = 0.308\nsamples = 1880\nvalue = [311, 211,
2405]\nclass = c'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.242\nsamples = 1760\nvalue = [287, 86, 2364]\nclass
= c'),
 Text(2790.0, 181.19999999999982, 'gini = 0.505\nsamples = 120\nvalue = [24, 125, 41]\nclass = b'),
 Text(3029.142857142857, 543.5999999999999, 'CO <= 0.365\ngini = 0.537\nsamples = 822\nvalue = [556, 62, 70
3]\nclass = c'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.566\nsamples = 188\nvalue = [178, 52, 72]\nclass =
a'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.479\nsamples = 634\nvalue = [378, 10, 631]\nclass =
c'),
 Text(3826.2857142857138, 1268.4, 'O_3 <= 13.36\ngini = 0.37\nsamples = 2581\nvalue = [3149, 210, 718]\nclas
s = a'),
 Text(3507.428571428571, 906.0, 'BEN <= 1.985\ngini = 0.512\nsamples = 1174\nvalue = [1166, 162, 505]\nclass
= a'),
 Text(3347.9999999999995, 543.5999999999999, 'CO <= 0.385\ngini = 0.642\nsamples = 415\nvalue = [242, 133, 2
67]\nclass = c'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.165\nsamples = 57\nvalue = [9, 90, 0]\nclass = b'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.568\nsamples = 358\nvalue = [233, 43, 267]\nclass =
c'),
 Text(3666.8571428571427, 543.5999999999999, 'BEN <= 3.055\ngini = 0.358\nsamples = 759\nvalue = [924, 29, 2
38]\nclass = a'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.453\nsamples = 400\nvalue = [435, 29, 168]\nclass =
a'),
```

```
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.219\nsamples = 359\nvalue = [489, 0, 70]\nclass =
a'),
 Text(4145.142857142857, 906.0, 'CO <= 0.605\ngini = 0.21\nsamples = 1407\nvalue = [1983, 48, 213]\nclass =
a'),
 Text(3985.7142857142853, 543.5999999999999, 'TCH <= 1.465\ngini = 0.314\nsamples = 627\nvalue = [835, 42, 1
48]\nclass = a'),
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.147\nsamples = 460\nvalue = [701, 9, 51]\nclass =
a'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.592\nsamples = 167\nvalue = [134, 33, 97]\nclass =
a'),
 Text(4304.571428571428, 543.5999999999999, 'TOL <= 9.815\ngini = 0.11\nsamples = 780\nvalue = [1148, 6, 65]
\nclass = a'),
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.228\nsamples = 250\nvalue = [345, 6, 45]\nclass =
a'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.047\nsamples = 530\nvalue = [803, 0, 20]\nclass =
a')]
```



Conclusion

Accuracy

Linear Regression0.14037527802328775

Ridge Regression:0.1403481649670576

Lasso Regression:0.0401780815850018

ElasticNet Regression:0.0944718215443624

Logistic Regression:0.794194530061254

Random Forest:0.854244619756438

Random Forest is suitable for this dataset