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_2004")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH	TO
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002	25.860001	NaN	12.20	NaN	Na
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	NaN	3.38	6.12	1.57	11.3
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000	NaN	NaN	8.99	NaN	Na
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000	NaN	NaN	8.82	NaN	Na
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002	NaN	NaN	8.71	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900000	14.860000	0.52	6.62	1.28	2.5
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689999	NaN	2.35	6.92	NaN	11.9
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840000	12.040000	NaN	7.82	1.52	Na
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630001	NaN	NaN	5.53	1.33	Na
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389999	17.959999	2.29	8.68	1.47	12.9

245496 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: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     19397 non-null  object 
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
 14  TCH      19397 non-null  float64
 15  TOL      19397 non-null  float64
 16  station  19397 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

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

Out[6]:

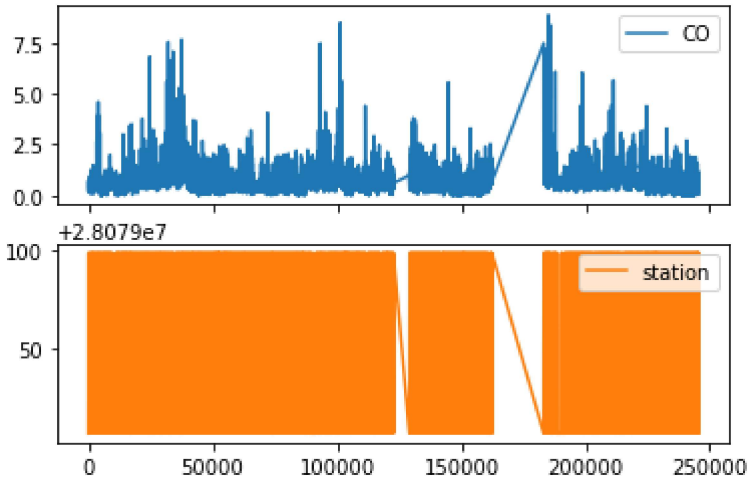
	CO	station
5	0.63	28079006
22	0.36	28079024
26	0.46	28079099
32	0.67	28079006
49	0.30	28079024
...	...	...
245463	0.08	28079024
245467	0.67	28079099
245473	1.12	28079006
245491	0.21	28079024
245495	0.67	28079099

19397 rows × 2 columns

Line chart

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

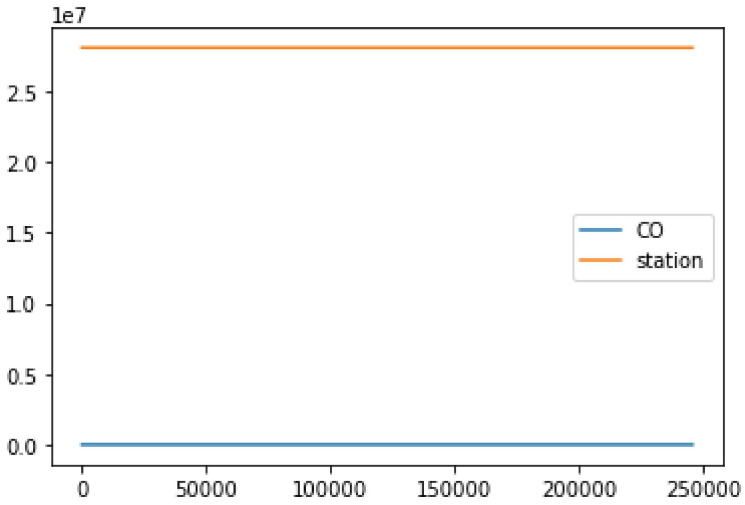
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

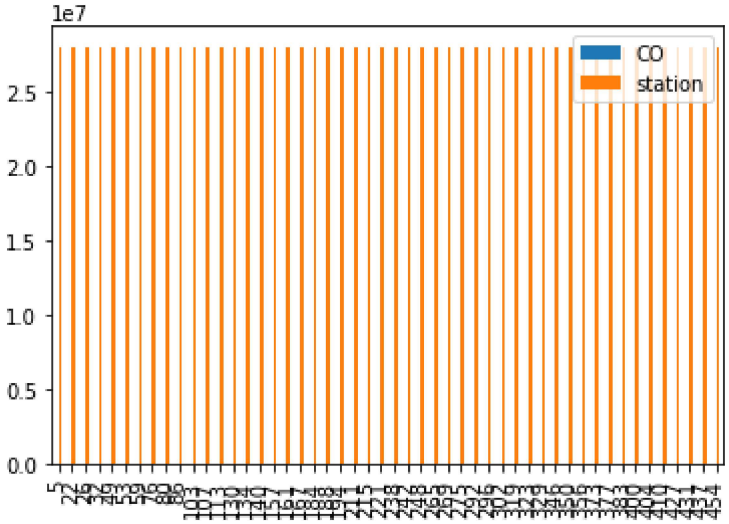


Bar chart

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

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

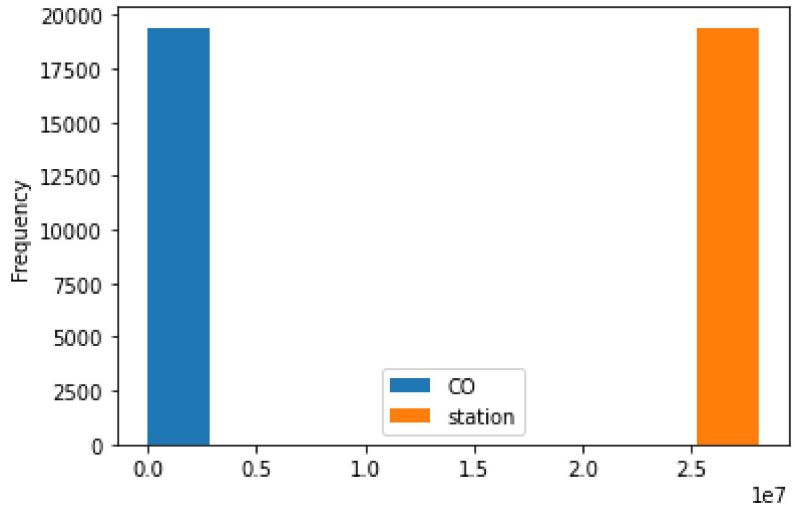
Out[10]: <AxesSubplot:>



Histogram

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

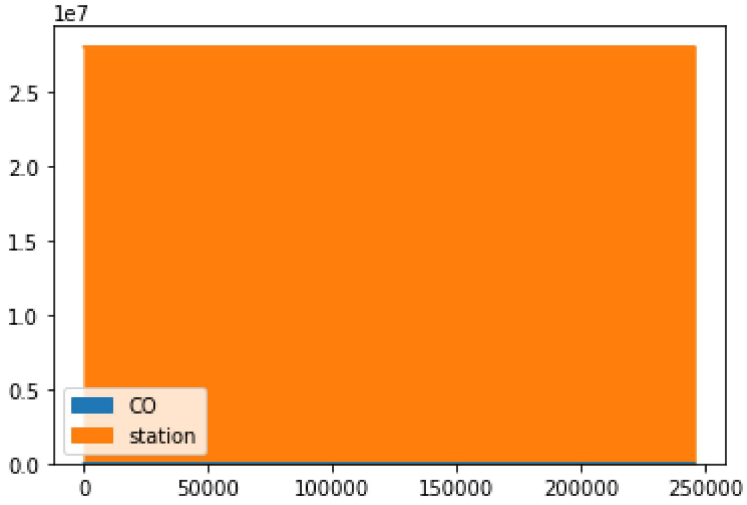
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

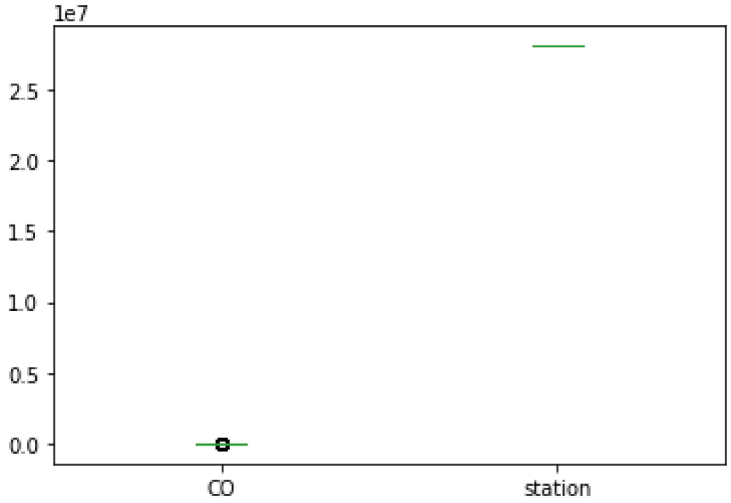
Out[12]: <AxesSubplot:>



Box chart

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

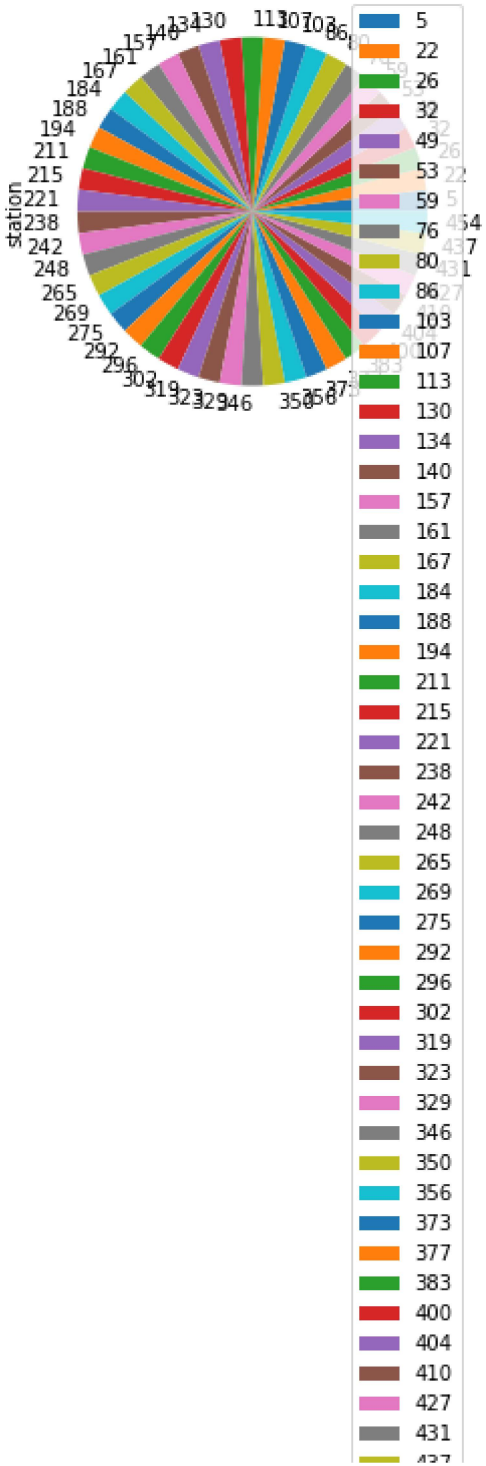
Out[13]: <AxesSubplot:>



Pie chart

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

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

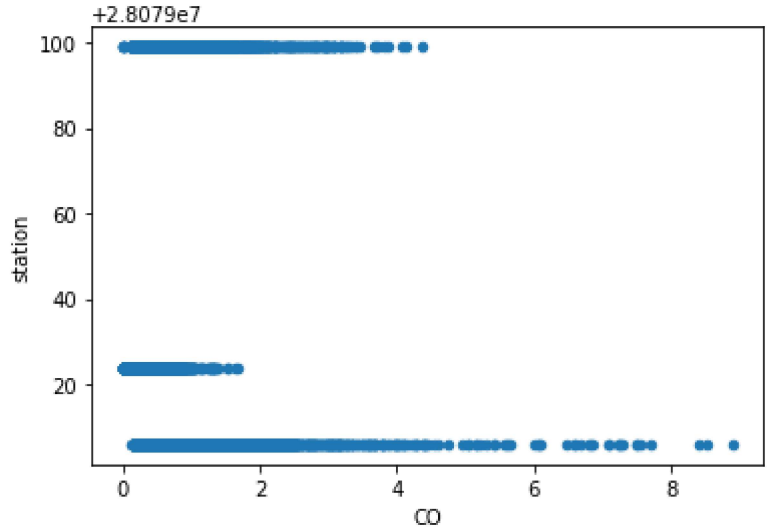

station
5
22
26
32
49
53
59
76
80
86
103
107
113
130
134
140
157
161
167
184
188
194
211
215
221
238
242
248
265
269
275
292
296
302
319
323
329
346
350
356
373
377
383
400
404
410
427
431

454

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: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     19397 non-null  object 
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.00000	19397.000000
mean	2.250781	0.675347	2.775913	5.424809	0.151024	62.887023	128.554023	2.71421	38.292501
std	2.184724	0.591026	2.729622	5.554358	0.158603	37.952255	127.912411	2.54485	29.808431
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.090000	2.410000	0.00000	0.770000
25%	0.870000	0.320000	1.020000	1.780000	0.060000	35.150002	45.209999	1.00000	12.710000
50%	1.620000	0.520000	1.970000	3.800000	0.110000	58.310001	93.220001	1.93000	31.830000
75%	2.910000	0.860000	3.580000	7.260000	0.200000	85.730003	174.300003	3.55000	56.139999
max	34.180000	8.900000	41.880001	91.599998	4.810000	355.100006	1700.000000	45.34000	192.500000

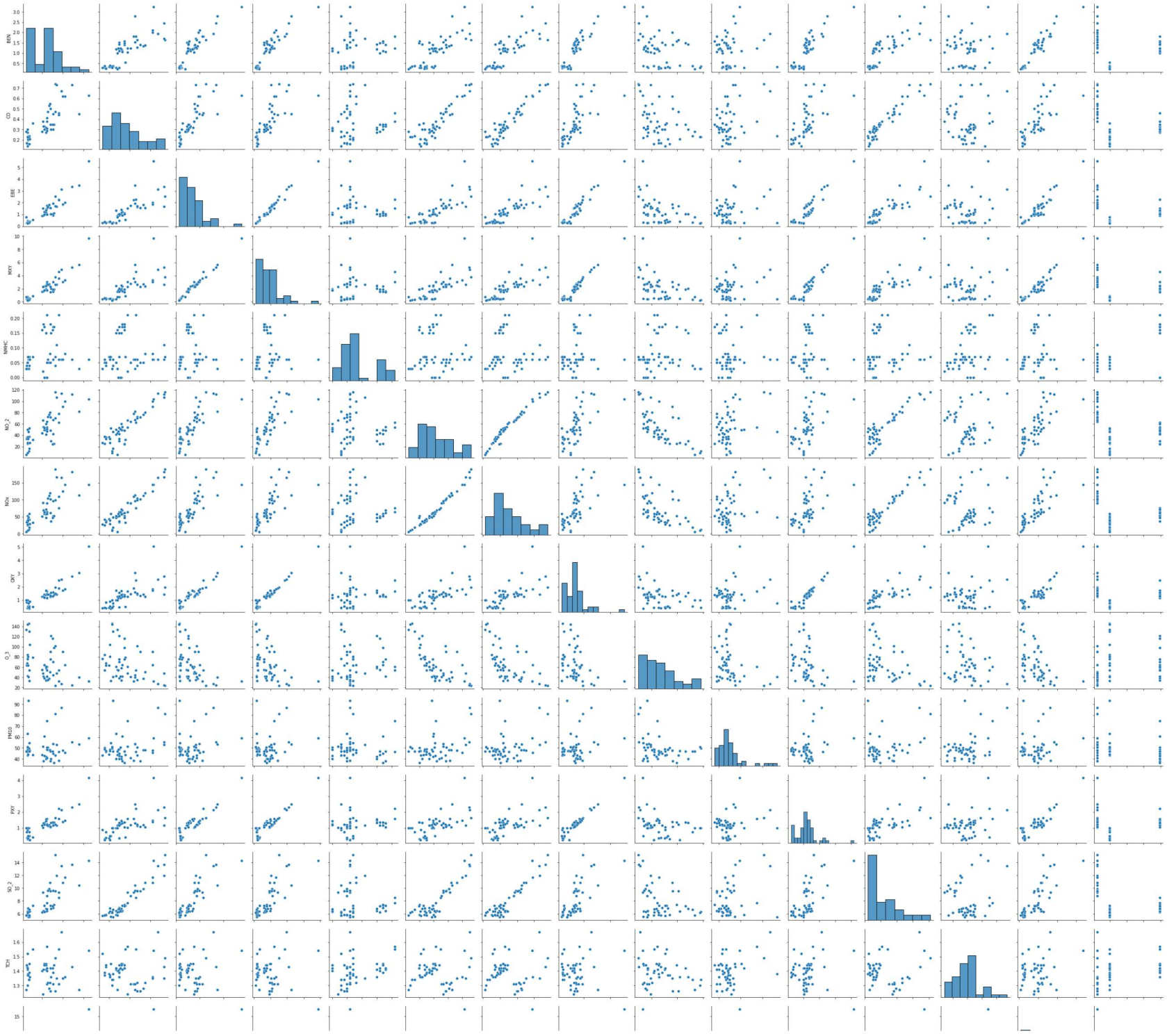
In [18]:

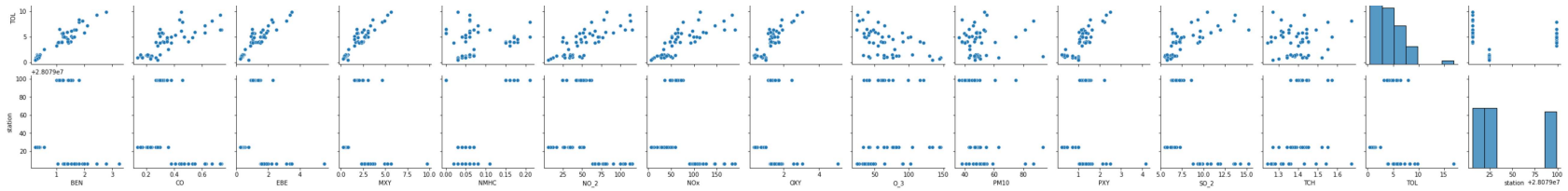
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1e04b4fd8b0>
```

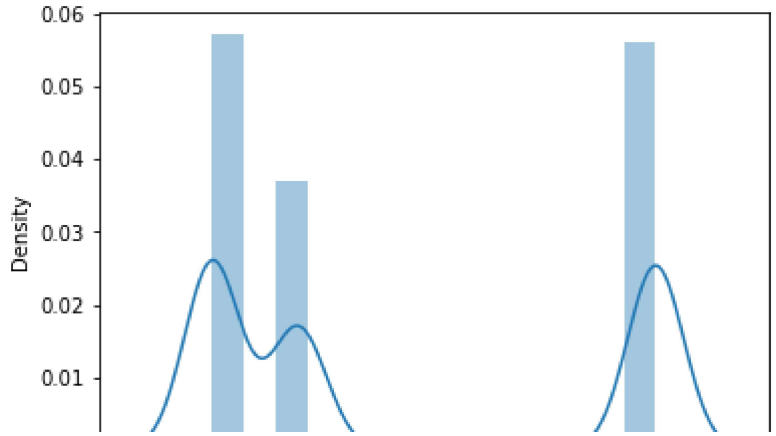

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



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

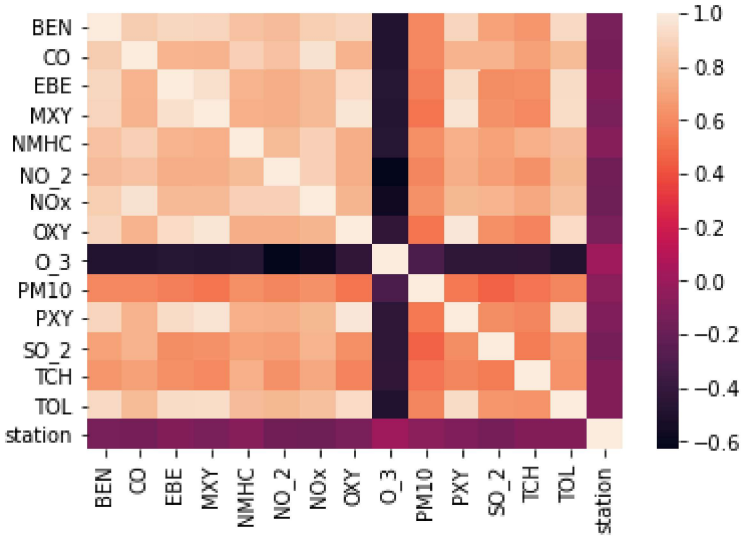
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28079076.59600632
```

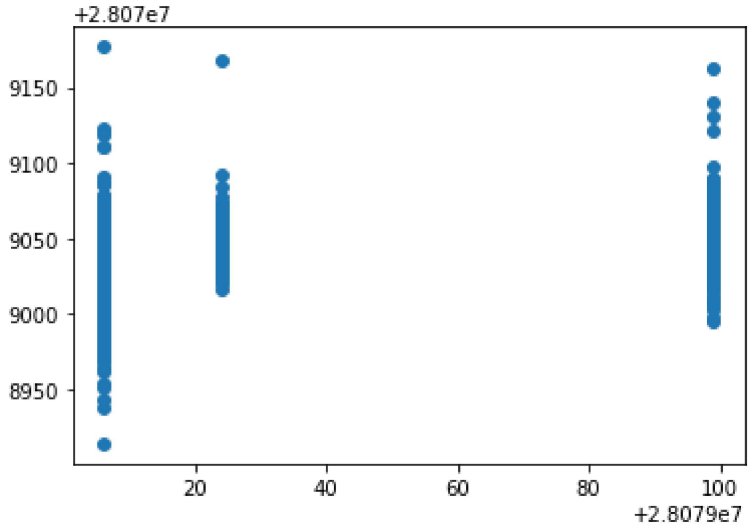
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-4.773847
CO	27.162786
EBE	4.108199
MXY	-3.078540
NMHC	78.272374
NO_2	-0.168998
NOx	-0.246871
OXY	-2.892686
O_3	-0.289682
PM10	0.068920
PXY	5.847097
SO_2	-0.132592
TCH	-7.286821
TOL	1.197092

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

Out[27]: <matplotlib.collections.PathCollection at 0x1e059fc1190>



ACCURACY

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

Out[28]: 0.1112746228512006

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

Out[29]: 0.10390101399679486

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]: 0.10899529791144424

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

Out[33]: 0.10358897784064869

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]: 0.054967136004089534

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

Out[36]: 0.05002837680426531

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([-0.28915406,  0.36158915,  1.39965368, -1.68984222,  0.        ,
       -0.18932432, -0.08633701, -0.06288545, -0.22891864,  0.09380176,
        0.3580717 , -0.05759837,  0.        ,  1.1580202 ])
```

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

Out[39]:
```
28079067.816384424
```

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

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

Out[41]:
```
0.06584479039848501
```

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)))
```

```
38.56286219795878
1660.9196913918136
40.75438248080584
```

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]:
```
feature_matrix.shape
```

Out[45]: (19397, 14)

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

Out[46]: (19397,)

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

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

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

Out[49]: LogisticRegression(max_iter=10000)

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

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

```
[28079006]
```

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

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

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

Out[53]: 0.7360416559261741

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

Out[54]: 0.9999978255573396

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

Out[55]: array([[9.99997826e-01, 7.75018107e-20, 2.17444266e-06]])

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]:
```
0.7751348675493556
```

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

In [62]:

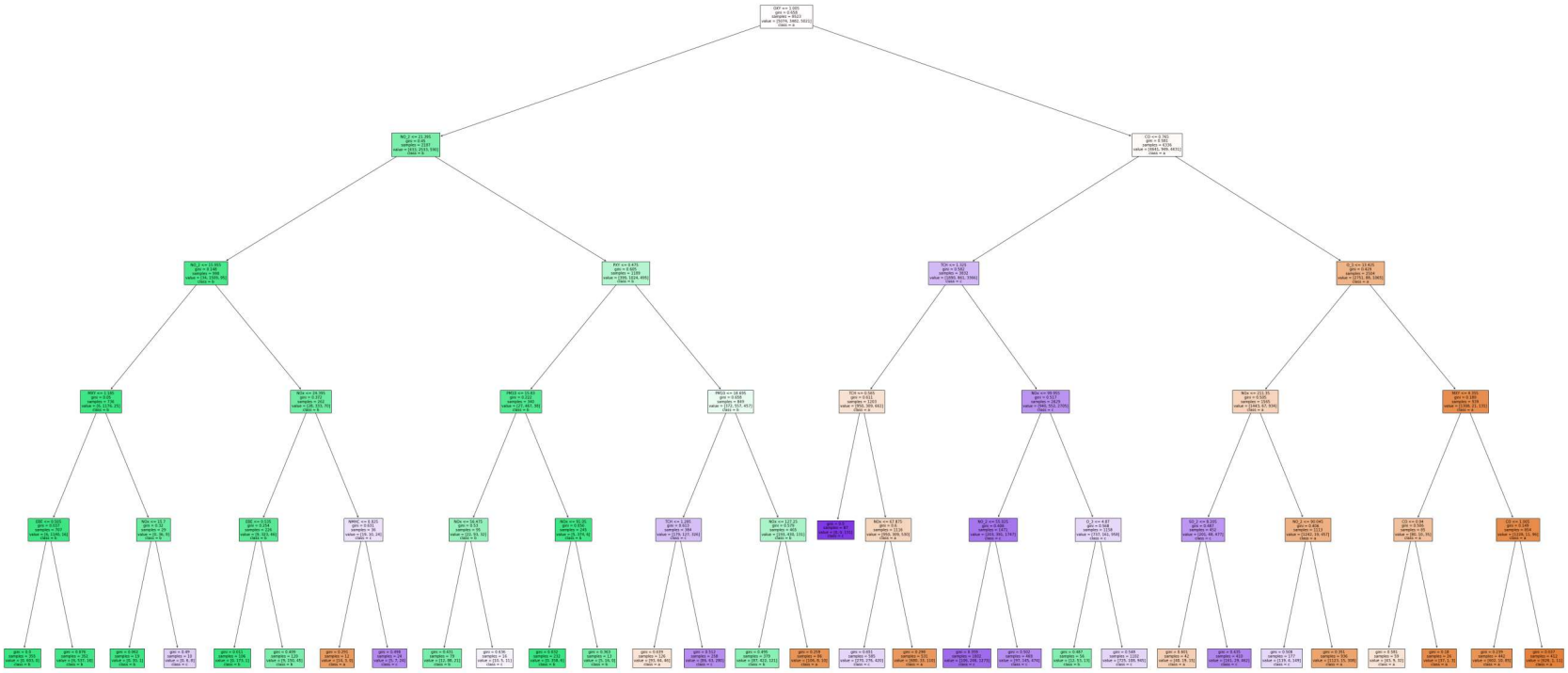
```
from sklearn.tree import plot_tree

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(2241.3, 1993.2, 'OXY <= 1.005\ngini = 0.658\nsamples = 8523\nvalue = [5074, 3482, 5021]\nclass = a'),
 Text(1190.4, 1630.8000000000002, 'NO_2 <= 21.395\ngini = 0.45\nsamples = 2187\nvalue = [433, 2533, 590]\ncl
ass = b'),
 Text(595.2, 1268.4, 'NO_2 <= 15.955\ngini = 0.148\nsamples = 998\nvalue = [34, 1509, 95]\nclass = b'),
 Text(297.6, 906.0, 'MXY <= 1.185\ngini = 0.05\nsamples = 736\nvalue = [6, 1176, 25]\nclass = b'),
 Text(148.8, 543.5999999999999, 'EBE <= 0.505\ngini = 0.037\nsamples = 707\nvalue = [6, 1140, 16]\nclass =
b'),
 Text(74.4, 181.19999999999982, 'gini = 0.0\nsamples = 355\nvalue = [0, 603, 0]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.076\nsamples = 352\nvalue = [6, 537, 16]\nclass =
b'),
 Text(446.40000000000003, 543.5999999999999, 'NOx <= 15.7\ngini = 0.32\nsamples = 29\nvalue = [0, 36, 9]\ncl
ass = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.062\nsamples = 19\nvalue = [0, 30, 1]\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.49\nsamples = 10\nvalue = [0, 6, 8]\nclass = c'),
 Text(892.8000000000001, 906.0, 'NOx <= 24.395\ngini = 0.372\nsamples = 262\nvalue = [28, 333, 70]\nclass =
b'),
 Text(744.0, 543.5999999999999, 'EBE <= 0.535\ngini = 0.254\nsamples = 226\nvalue = [9, 323, 46]\nclass =
b'),
 Text(669.6, 181.19999999999982, 'gini = 0.011\nsamples = 106\nvalue = [0, 173, 1]\nclass = b'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.409\nsamples = 120\nvalue = [9, 150, 45]\nclass =
b'),
 Text(1041.6000000000001, 543.5999999999999, 'NMHC <= 0.025\ngini = 0.631\nsamples = 36\nvalue = [19, 10, 2
4]\nclass = c'),
 Text(967.2, 181.19999999999982, 'gini = 0.291\nsamples = 12\nvalue = [14, 3, 0]\nclass = a'),
 Text(1116.0, 181.19999999999982, 'gini = 0.498\nsamples = 24\nvalue = [5, 7, 24]\nclass = c'),
 Text(1785.6000000000001, 1268.4, 'PXY <= 0.475\ngini = 0.605\nsamples = 1189\nvalue = [399, 1024, 495]\ncla
ss = b'),
 Text(1488.0, 906.0, 'PM10 <= 15.83\ngini = 0.222\nsamples = 340\nvalue = [27, 467, 38]\nclass = b'),
 Text(1339.2, 543.5999999999999, 'NOx <= 56.475\ngini = 0.53\nsamples = 95\nvalue = [22, 93, 32]\nclass =
b'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.431\nsamples = 79\nvalue = [12, 88, 21]\nclass =
b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.636\nsamples = 16\nvalue = [10, 5, 11]\nclass = c'),
 Text(1636.8000000000002, 543.5999999999999, 'NOx <= 91.05\ngini = 0.056\nsamples = 245\nvalue = [5, 374, 6]
\nclass = b'),
 Text(1562.4, 181.19999999999982, 'gini = 0.032\nsamples = 232\nvalue = [0, 358, 6]\nclass = b'),
 Text(1711.2, 181.19999999999982, 'gini = 0.363\nsamples = 13\nvalue = [5, 16, 0]\nclass = b'),
 Text(2083.2000000000003, 906.0, 'PM10 <= 18.695\ngini = 0.658\nsamples = 849\nvalue = [372, 557, 457]\nclas
s = b'),
 Text(1934.4, 543.5999999999999, 'TCH <= 1.285\ngini = 0.613\nsamples = 384\nvalue = [179, 127, 326]\nclass
= c'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.639\nsamples = 126\nvalue = [93, 64, 46]\nclass =
a'),
```

```
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.512\nsamples = 258\nvalue = [86, 63, 280]\nclass =
c'),
 Text(2232.0, 543.5999999999999, 'NOx <= 127.25\ngini = 0.579\nsamples = 465\nvalue = [193, 430, 131]\nclass
= b'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.495\nsamples = 379\nvalue = [87, 422, 121]\nclass =
b'),
 Text(2306.4, 181.19999999999982, 'gini = 0.259\nsamples = 86\nvalue = [106, 8, 10]\nclass = a'),
 Text(3292.2000000000003, 1630.8000000000002, 'CO <= 0.765\ngini = 0.581\nsamples = 6336\nvalue = [4641, 94
9, 4431]\nclass = a'),
 Text(2715.6000000000004, 1268.4, 'TCH <= 1.325\ngini = 0.582\nsamples = 3832\nvalue = [1890, 861, 3366]\ncl
ass = c'),
 Text(2455.2000000000003, 906.0, 'TCH <= 0.565\ngini = 0.611\nsamples = 1203\nvalue = [950, 309, 661]\nclass
= a'),
 Text(2380.8, 543.5999999999999, 'gini = 0.0\nsamples = 87\nvalue = [0, 0, 131]\nclass = c'),
 Text(2529.6000000000004, 543.5999999999999, 'NOx <= 67.875\ngini = 0.6\nsamples = 1116\nvalue = [950, 309,
530]\nclass = a'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.651\nsamples = 585\nvalue = [270, 276, 420]\nclass =
c'),
 Text(2604.0, 181.19999999999982, 'gini = 0.298\nsamples = 531\nvalue = [680, 33, 110]\nclass = a'),
 Text(2976.0, 906.0, 'NOx <= 99.955\ngini = 0.517\nsamples = 2629\nvalue = [940, 552, 2705]\nclass = c'),
 Text(2827.2000000000003, 543.5999999999999, 'NO_2 <= 55.025\ngini = 0.408\nsamples = 1471\nvalue = [203, 39
1, 1747]\nclass = c'),
 Text(2752.8, 181.19999999999982, 'gini = 0.359\nsamples = 1002\nvalue = [106, 246, 1273]\nclass = c'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.502\nsamples = 469\nvalue = [97, 145, 474]\nclass =
c'),
 Text(3124.8, 543.5999999999999, 'O_3 <= 4.87\ngini = 0.568\nsamples = 1158\nvalue = [737, 161, 958]\nclass
= c'),
 Text(3050.4, 181.19999999999982, 'gini = 0.487\nsamples = 56\nvalue = [12, 53, 13]\nclass = b'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.548\nsamples = 1102\nvalue = [725, 108, 945]\nclass
= c'),
 Text(3868.8, 1268.4, 'O_3 <= 13.425\ngini = 0.429\nsamples = 2504\nvalue = [2751, 88, 1065]\nclass = a'),
 Text(3571.2000000000003, 906.0, 'NOx <= 211.35\ngini = 0.505\nsamples = 1565\nvalue = [1443, 67, 934]\nclas
s = a'),
 Text(3422.4, 543.5999999999999, 'SO_2 <= 8.205\ngini = 0.487\nsamples = 452\nvalue = [201, 48, 477]\nclass
= c'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.601\nsamples = 42\nvalue = [40, 19, 15]\nclass =
a'),
 Text(3496.8, 181.19999999999982, 'gini = 0.435\nsamples = 410\nvalue = [161, 29, 462]\nclass = c'),
 Text(3720.0000000000005, 543.5999999999999, 'NO_2 <= 90.045\ngini = 0.406\nsamples = 1113\nvalue = [1242, 1
9, 457]\nclass = a'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.508\nsamples = 177\nvalue = [119, 4, 149]\nclass =
c'),
 Text(3794.4, 181.19999999999982, 'gini = 0.351\nsamples = 936\nvalue = [1123, 15, 308]\nclass = a'),
```

```
 Text(4166.400000000001, 906.0, 'MXY <= 4.355\ngini = 0.189\nsamples = 939\nvalue = [1308, 21, 131]\nclass =
a'),
 Text(4017.6000000000004, 543.5999999999999, 'CO <= 0.94\ngini = 0.506\nsamples = 85\nvalue = [80, 10, 35]\n
class = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.581\nsamples = 59\nvalue = [43, 9, 32]\nclass = a'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.18\nsamples = 26\nvalue = [37, 1, 3]\nclass = a'),
 Text(4315.200000000001, 543.5999999999999, 'CO <= 1.005\ngini = 0.149\nsamples = 854\nvalue = [1228, 11, 9
6]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.239\nsamples = 442\nvalue = [602, 10, 85]\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.037\nsamples = 412\nvalue = [626, 1, 11]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression0.10390101399679486

Ridge Regression:0.10358897784064869

Lasso Regression:0.054967136004089534

ElasticNet Regression:0.06584479039848501

Logistic Regression:0.7360416559261741

Random Forest:0.7751348675493556

Random Forest is suitable for this dataset

In [ ]:
```
1
```