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_2007")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH	
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000	156.199997	97.43	NaN	64.519997	NaN	
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000	80.809998	NaN	NaN	35.419998	NaN	
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000	53.099998	NaN	NaN	19.080000	NaN	
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000	105.300003	NaN	NaN	17.670000	NaN	
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	106.500000	15.90	3.56	40.230000	1.94	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999	6.760000	5.14	1.00	7.420000	1.44	
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN	22.150000	5.700000	NaN	NaN	7.130000	NaN	
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN	18.540001	13.010000	6.95	NaN	8.740000	1.30	
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN	NaN	6.610000	NaN	NaN	9.890000	1.29	
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000	10.260000	7.08	1.23	9.890000	1.32	

225120 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: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25443 non-null  object 
 1   BEN      25443 non-null  float64
 2   CO       25443 non-null  float64
 3   EBE      25443 non-null  float64
 4   MXY      25443 non-null  float64
 5   NMHC     25443 non-null  float64
 6   NO_2     25443 non-null  float64
 7   NOx      25443 non-null  float64
 8   OXY      25443 non-null  float64
 9   O_3      25443 non-null  float64
 10  PM10     25443 non-null  float64
 11  PM25     25443 non-null  float64
 12  PXY      25443 non-null  float64
 13  SO_2     25443 non-null  float64
 14  TCH      25443 non-null  float64
 15  TOL      25443 non-null  float64
 16  station  25443 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]:

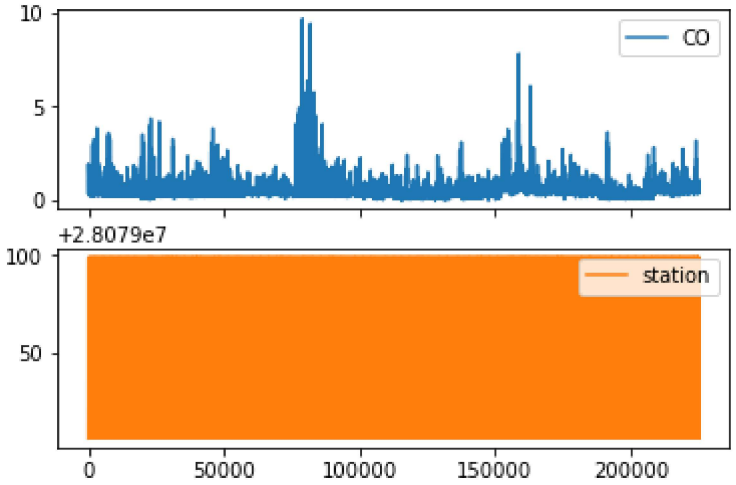
	CO	station
4	1.86	28079006
21	0.31	28079024
25	1.42	28079099
30	1.89	28079006
47	0.30	28079024
...	...	...
225073	0.47	28079006
225094	0.45	28079099
225098	0.41	28079006
225115	0.45	28079024
225119	0.40	28079099

25443 rows × 2 columns

Line chart

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

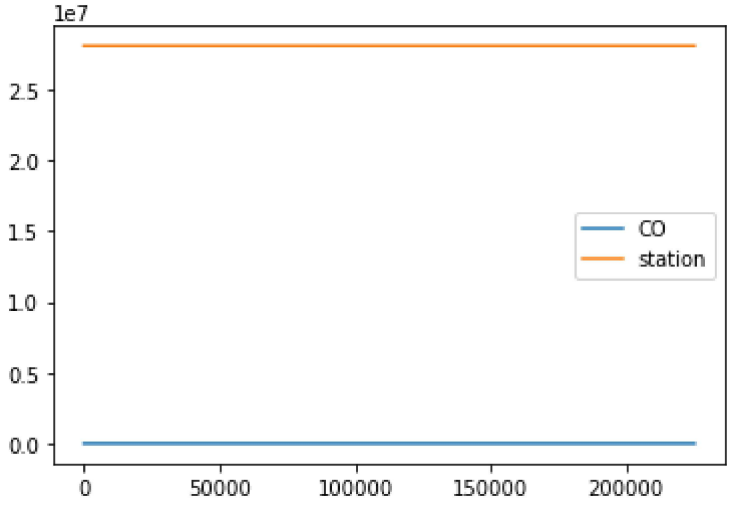
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

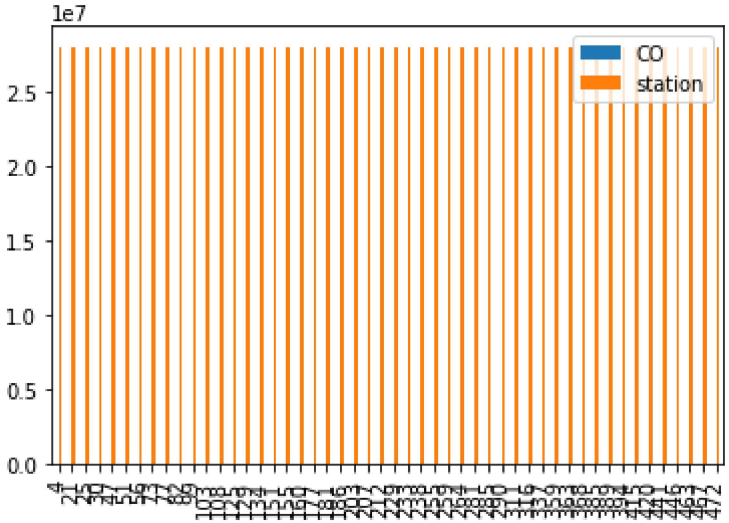


Bar chart

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

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

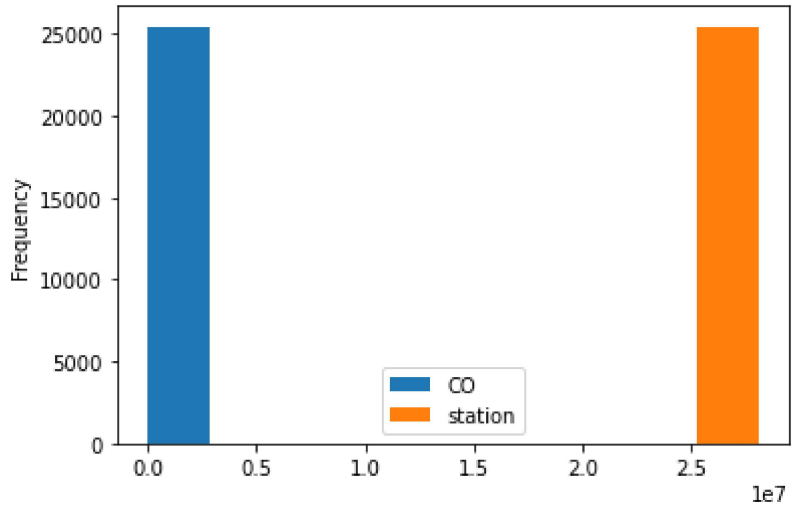
Out[10]: <AxesSubplot:>



Histogram

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

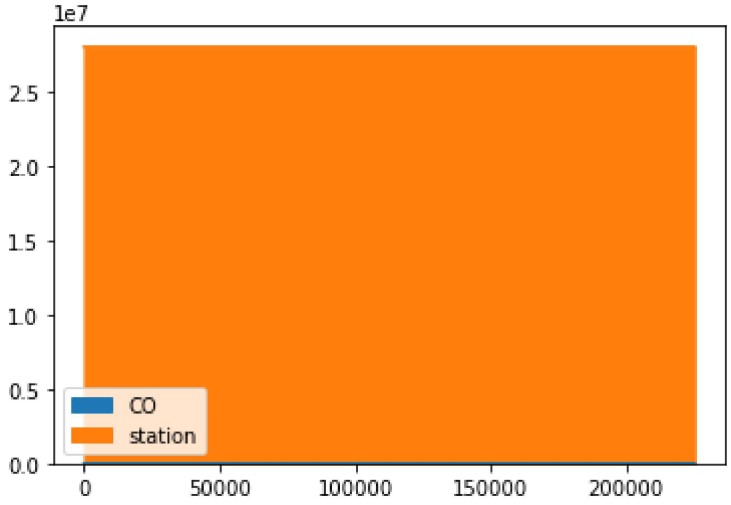
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

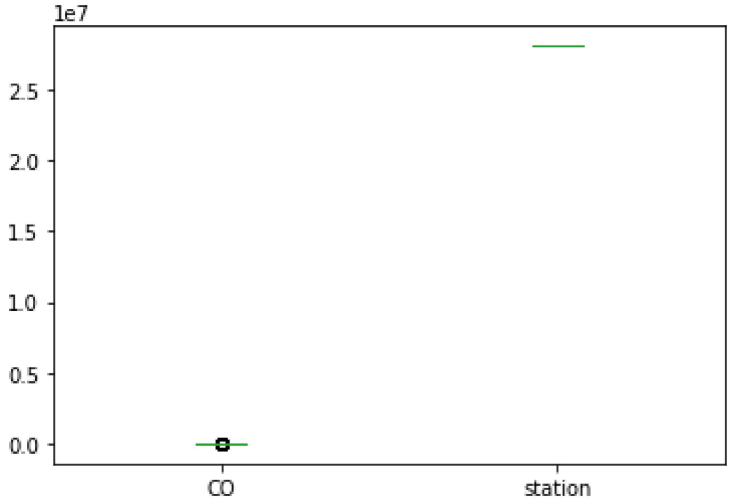
Out[12]: <AxesSubplot:>



Box chart

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

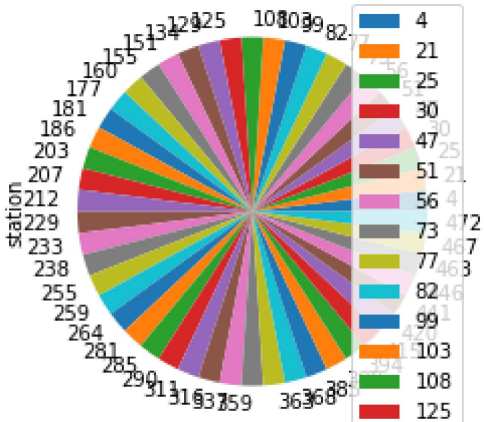
Out[13]: <AxesSubplot:>

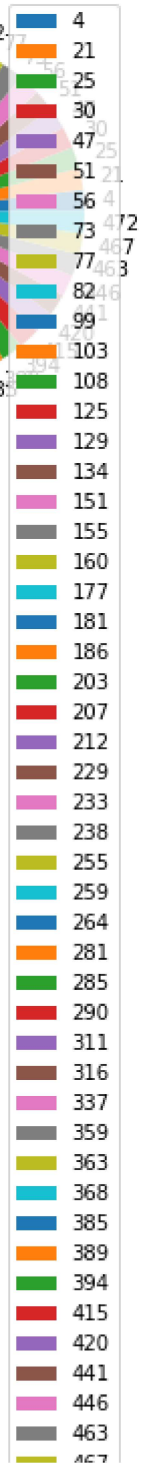


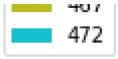
Pie chart

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

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


station

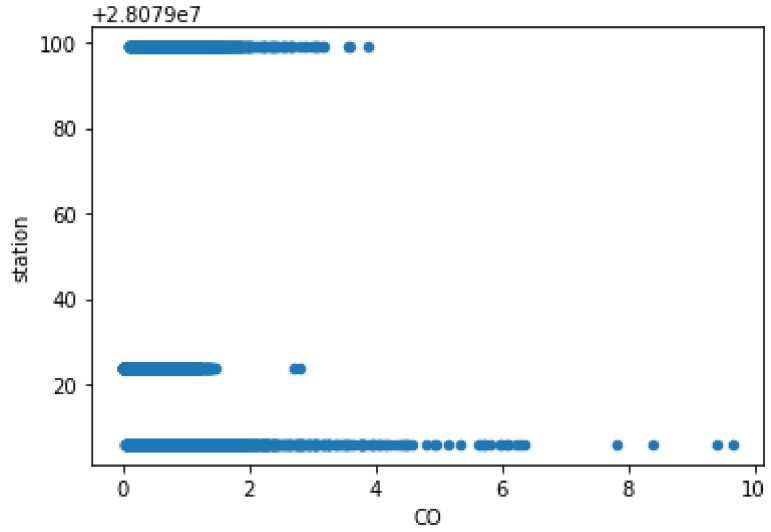

4
21
25
30
47
51
56
73
77
82
99
103
108
125
129
134
151
155
160
177
181
186
203
207
212
229
233
238
255
259
264
281
285
290
311
316
337
359
363
368
385
389
394
415
420
441
446
463



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: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     25443 non-null  object
 1   BEN      25443 non-null  float64
 2   CO       25443 non-null  float64
 3   EBE      25443 non-null  float64
 4   MXY      25443 non-null  float64
 5   NMHC     25443 non-null  float64
 6   NO_2     25443 non-null  float64
 7   NOx      25443 non-null  float64
 8   OXY      25443 non-null  float64
 9   O_3      25443 non-null  float64
 10  PM10     25443 non-null  float64
 11  PM25     25443 non-null  float64
 12  PXY      25443 non-null  float64
 13  SO_2     25443 non-null  float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000
mean	1.146744	0.505120	1.394071	2.392008	0.249967	58.532683	112.741861	1.270278	35.653534
std	1.278733	0.423231	1.268265	2.784302	0.142627	37.755029	115.527006	1.143188	26.911492
min	0.130000	0.000000	0.120000	0.150000	0.000000	1.690000	1.780000	0.110000	0.280000
25%	0.450000	0.260000	0.780000	0.960000	0.160000	31.285001	39.910000	0.740000	11.680000
50%	0.770000	0.400000	1.000000	1.500000	0.220000	54.080002	82.809998	1.000000	30.980000
75%	1.390000	0.640000	1.580000	2.855000	0.300000	79.230003	149.300003	1.360000	53.310001
max	30.139999	9.660000	31.680000	65.480003	2.570000	430.299988	1893.000000	32.540001	149.100006

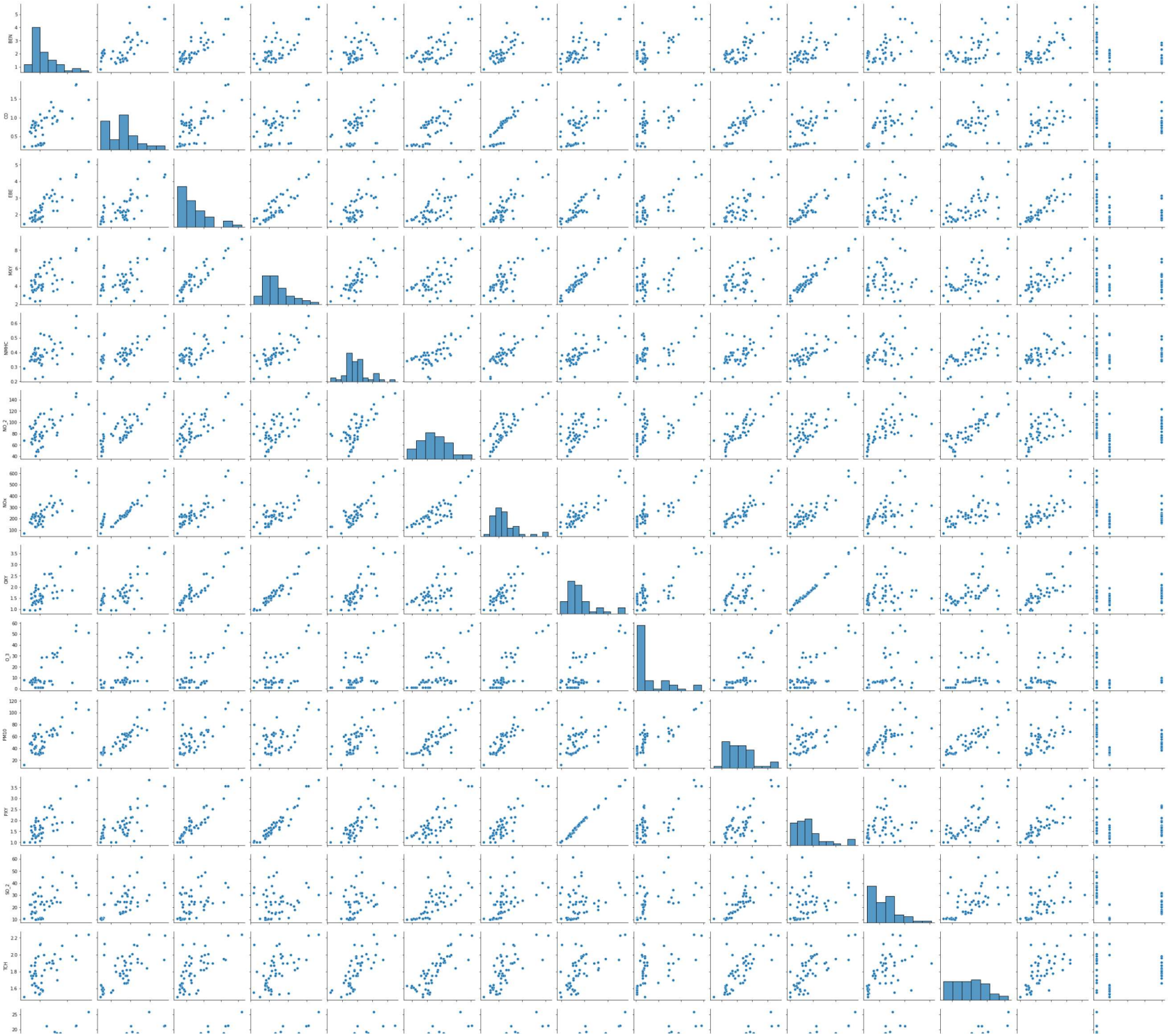
In [18]:

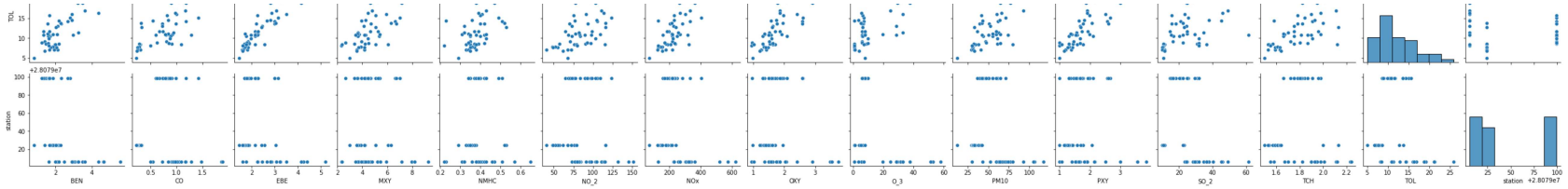
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x245ef4504f0>
```

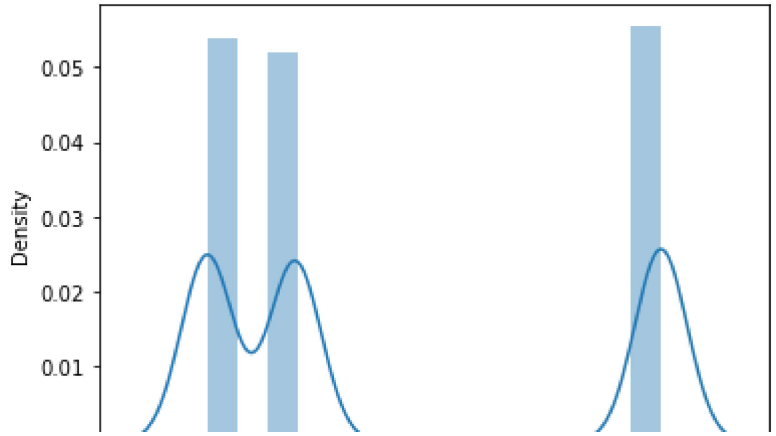

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



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

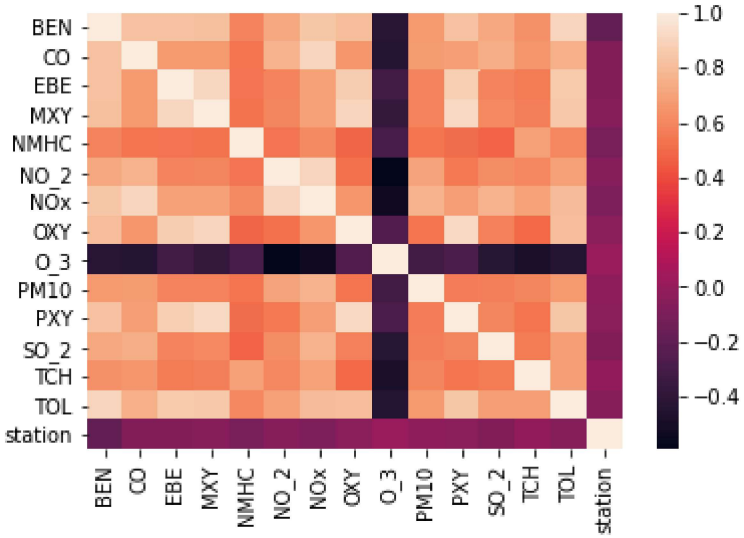
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28079009.799331877
```

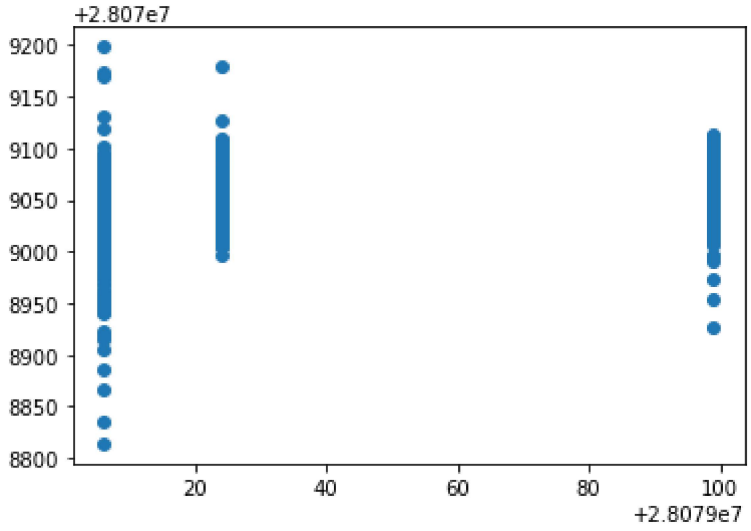
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-32.676195
CO	17.012253
EBE	0.846044
MXY	-1.163146
NMHC	-41.008556
NO_2	0.100799
NOx	-0.035712
OXY	5.301298
O_3	-0.037435
PM10	0.137878
PXY	6.509756
SO_2	0.233654
TCH	26.243788
TOL	3.097605

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

Out[27]: <matplotlib.collections.PathCollection at 0x245fddb6f70>



ACCURACY

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

Out[28]: 0.15888618593861437

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

Out[29]: 0.15930303157272896

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]:
```
0.15883525144058097
```

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

Out[33]:
```
0.15925082658615208
```

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]:
```
0.012581248316343108
```

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

Out[36]:
```
0.014373582045566824
```

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([-8.15496477e+00,  0.00000000e+00,  0.00000000e+00,  1.52019723e-01,
       -0.00000000e+00,  3.80128154e-02, -4.42837414e-02,  5.71943162e-01,
       -6.15289655e-02,  1.57697948e-01,  5.63096780e-01,  5.54937274e-04,
        0.00000000e+00,  9.18614145e-01])
```

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

Out[39]:
```
28079046.59911058
```

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

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

Out[41]:
```
0.06773347039258326
```

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)))
```

```
36.616311701215956
1537.9390731126289
39.21656630956653
```

Logistic Regression

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

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

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

Out[45]: (25443, 14)

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

Out[46]: (25443,)

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)
```

```
[28079099]
```

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

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

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

Out[53]: 0.8146838030106512

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

Out[54]:
```
1.082753977181323e-19
```

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

Out[55]:
```
array([[1.08275398e-19, 1.80383815e-19, 1.00000000e+00]])
```

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]:
```
0.8211117349803481
```

In [61]:

```
1 rfc_best=grid_search.best_estimator_
```

In [62]:

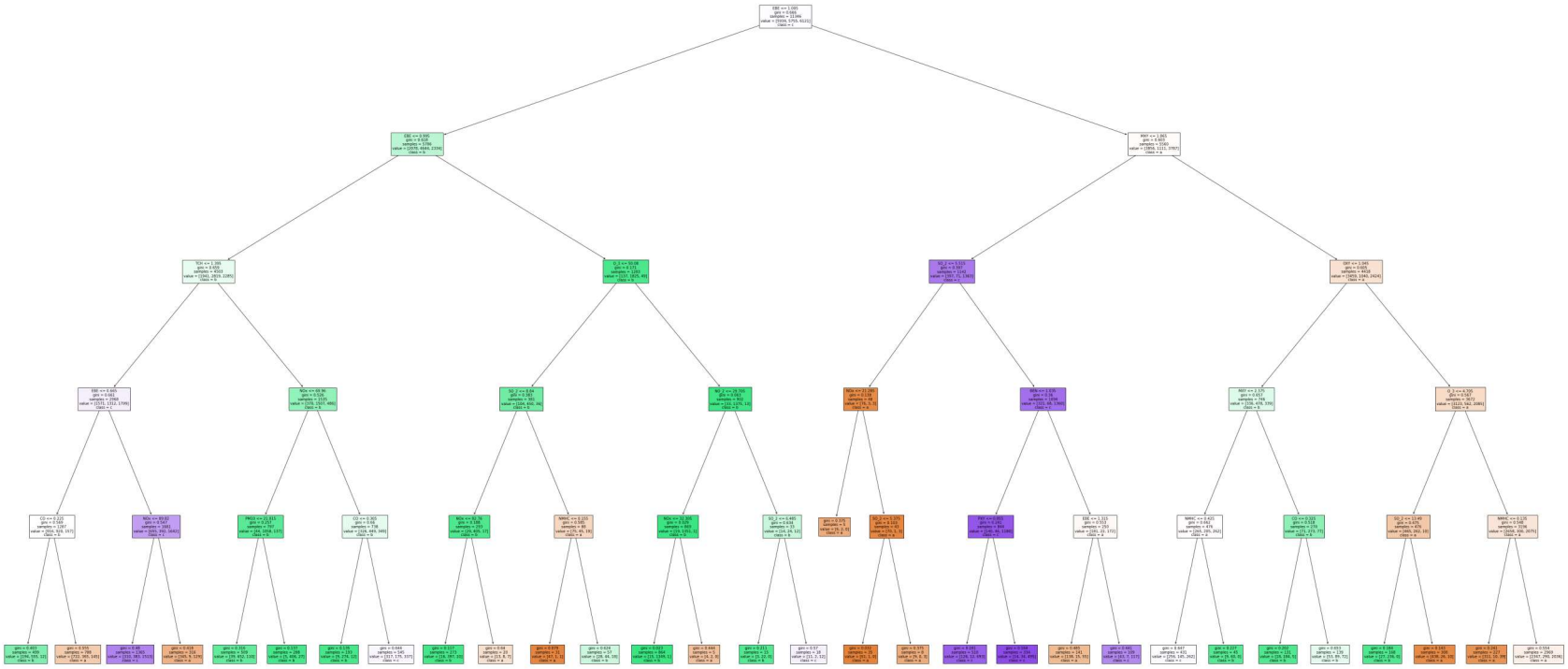
```
from sklearn.tree import plot_tree

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, 'EBE <= 1.005\ngini = 0.666\nsamples = 11346\nvalue = [5934, 5755, 6121]\nclass = c'),
 Text(1190.4, 1630.8000000000002, 'EBE <= 0.995\ngini = 0.618\nsamples = 5786\nvalue = [2078, 4644, 2334]\nclass = b'),
 Text(595.2, 1268.4, 'TCH <= 1.395\ngini = 0.659\nsamples = 4503\nvalue = [1941, 2819, 2285]\nclass = b'),
 Text(297.6, 906.0, 'EBE <= 0.665\ngini = 0.661\nsamples = 2968\nvalue = [1571, 1312, 1799]\nclass = c'),
 Text(148.8, 543.5999999999999, 'CO <= 0.225\ngini = 0.569\nsamples = 1287\nvalue = [916, 920, 157]\nclass = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.403\nsamples = 499\nvalue = [194, 555, 12]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.555\nsamples = 788\nvalue = [722, 365, 145]\nclass = a'),
 Text(446.40000000000003, 543.5999999999999, 'NOx <= 89.82\ngini = 0.547\nsamples = 1681\nvalue = [655, 392, 1642]\nclass = c'),
 Text(372.0, 181.19999999999982, 'gini = 0.48\nsamples = 1365\nvalue = [310, 383, 1513]\nclass = c'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.418\nsamples = 316\nvalue = [345, 9, 129]\nclass = a'),
 Text(892.8000000000001, 906.0, 'NOx <= 69.96\ngini = 0.526\nsamples = 1535\nvalue = [370, 1507, 486]\nclass = b'),
 Text(744.0, 543.5999999999999, 'PM10 <= 21.015\ngini = 0.257\nsamples = 797\nvalue = [44, 1058, 137]\nclass = b'),
 Text(669.6, 181.19999999999982, 'gini = 0.316\nsamples = 509\nvalue = [39, 652, 110]\nclass = b'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.137\nsamples = 288\nvalue = [5, 406, 27]\nclass = b'),
 Text(1041.6000000000001, 543.5999999999999, 'CO <= 0.305\ngini = 0.66\nsamples = 738\nvalue = [326, 449, 349]\nclass = b'),
 Text(967.2, 181.19999999999982, 'gini = 0.135\nsamples = 193\nvalue = [9, 274, 12]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.644\nsamples = 545\nvalue = [317, 175, 337]\nclass = c'),
 Text(1785.6000000000001, 1268.4, 'O_3 <= 50.08\ngini = 0.171\nsamples = 1283\nvalue = [137, 1825, 49]\nclass = b'),
 Text(1488.0, 906.0, 'SO_2 <= 8.84\ngini = 0.383\nsamples = 381\nvalue = [104, 450, 36]\nclass = b'),
 Text(1339.2, 543.5999999999999, 'NOx <= 82.76\ngini = 0.188\nsamples = 293\nvalue = [29, 405, 17]\nclass = b'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.117\nsamples = 273\nvalue = [16, 397, 10]\nclass = b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.64\nsamples = 20\nvalue = [13, 8, 7]\nclass = a'),
 Text(1636.8000000000002, 543.5999999999999, 'NMHC <= 0.155\ngini = 0.585\nsamples = 88\nvalue = [75, 45, 19]\nclass = a'),
 Text(1562.4, 181.19999999999982, 'gini = 0.079\nsamples = 31\nvalue = [47, 1, 1]\nclass = a'),
 Text(1711.2, 181.19999999999982, 'gini = 0.624\nsamples = 57\nvalue = [28, 44, 18]\nclass = b'),
 Text(2083.2000000000003, 906.0, 'NO_2 <= 29.705\ngini = 0.063\nsamples = 902\nvalue = [33, 1375, 13]\nclass = b'),
 Text(1934.4, 543.5999999999999, 'NOx <= 32.305\ngini = 0.029\nsamples = 869\nvalue = [19, 1351, 1]\nclass = b'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.023\nsamples = 864\nvalue = [15, 1349, 1]\nclass =
```

```
b'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.444\nsamples = 5\nvalue = [4, 2, 0]\nclass = a'),
 Text(2232.0, 543.5999999999999, 'SO_2 <= 6.485\ngini = 0.634\nsamples = 33\nvalue = [14, 24, 12]\nclass =
b'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.211\nsamples = 15\nvalue = [3, 22, 0]\nclass = b'),
 Text(2306.4, 181.19999999999982, 'gini = 0.57\nsamples = 18\nvalue = [11, 2, 12]\nclass = c'),
 Text(3292.2000000000003, 1630.8000000000002, 'MXY <= 1.865\ngini = 0.603\nsamples = 5560\nvalue = [3856, 11
11, 3787]\nclass = a'),
 Text(2715.6000000000004, 1268.4, 'SO_2 <= 5.515\ngini = 0.397\nsamples = 1142\nvalue = [397, 71, 1363]\ncla
ss = c'),
 Text(2455.2000000000003, 906.0, 'NOx <= 21.285\ngini = 0.138\nsamples = 48\nvalue = [76, 3, 3]\nclass =
a'),
 Text(2380.8, 543.5999999999999, 'gini = 0.375\nsamples = 5\nvalue = [6, 2, 0]\nclass = a'),
 Text(2529.6000000000004, 543.5999999999999, 'SO_2 <= 5.375\ngini = 0.103\nsamples = 43\nvalue = [70, 1, 3]
\nclass = a'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.032\nsamples = 35\nvalue = [61, 1, 0]\nclass = a'),
 Text(2604.0, 181.19999999999982, 'gini = 0.375\nsamples = 8\nvalue = [9, 0, 3]\nclass = a'),
 Text(2976.0, 906.0, 'BEN <= 1.035\ngini = 0.36\nsamples = 1094\nvalue = [321, 68, 1360]\nclass = c'),
 Text(2827.2000000000003, 543.5999999999999, 'PXY <= 0.855\ngini = 0.241\nsamples = 844\nvalue = [140, 46, 1
188]\nclass = c'),
 Text(2752.8, 181.19999999999982, 'gini = 0.281\nsamples = 510\nvalue = [126, 12, 693]\nclass = c'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.164\nsamples = 334\nvalue = [14, 34, 495]\nclass =
c'),
 Text(3124.8, 543.5999999999999, 'EBE <= 1.315\ngini = 0.553\nsamples = 250\nvalue = [181, 22, 172]\nclass =
a'),
 Text(3050.4, 181.19999999999982, 'gini = 0.485\nsamples = 141\nvalue = [138, 15, 55]\nclass = a'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.441\nsamples = 109\nvalue = [43, 7, 117]\nclass =
c'),
 Text(3868.8, 1268.4, 'OXY <= 1.045\ngini = 0.605\nsamples = 4418\nvalue = [3459, 1040, 2424]\nclass = a'),
 Text(3571.2000000000003, 906.0, 'MXY <= 2.375\ngini = 0.657\nsamples = 746\nvalue = [336, 478, 339]\nclass
= b'),
 Text(3422.4, 543.5999999999999, 'NMHC <= 0.425\ngini = 0.662\nsamples = 476\nvalue = [265, 205, 262]\nclass
= a'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.647\nsamples = 431\nvalue = [256, 145, 262]\nclass =
c'),
 Text(3496.8, 181.19999999999982, 'gini = 0.227\nsamples = 45\nvalue = [9, 60, 0]\nclass = b'),
 Text(3720.0000000000005, 543.5999999999999, 'CO <= 0.325\ngini = 0.518\nsamples = 270\nvalue = [71, 273, 7
7]\nclass = b'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.202\nsamples = 131\nvalue = [18, 184, 5]\nclass =
b'),
 Text(3794.4, 181.19999999999982, 'gini = 0.653\nsamples = 139\nvalue = [53, 89, 72]\nclass = b'),
 Text(4166.400000000001, 906.0, 'O_3 <= 4.705\ngini = 0.567\nsamples = 3672\nvalue = [3123, 562, 2085]\nclas
s = a'),
```

```
 Text(4017.6000000000004, 543.5999999999999, 'SO_2 <= 13.49\ngini = 0.475\nsamples = 476\nvalue = [465, 262,
10]\nclass = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.184\nsamples = 168\nvalue = [27, 236, 0]\nclass =
b'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.143\nsamples = 308\nvalue = [438, 26, 10]\nclass =
a'),
 Text(4315.200000000001, 543.5999999999999, 'NMHC <= 0.135\ngini = 0.548\nsamples = 3196\nvalue = [2658, 30
0, 2075]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.241\nsamples = 227\nvalue = [311, 10, 39]\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.554\nsamples = 2969\nvalue = [2347, 290, 2036]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression0.15930303157272896

Ridge Regression:0.15925082658615208

Lasso Regression:0.012581248316343108

ElasticNet Regression:0.06773347039258326

Logistic Regression:0.8146838030106512

Random Forest:0.8211117349803481

Random Forest is suitable for this dataset