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_2006")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570000	40.259998	NaN	33.779999	NaN
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820000	NaN	2.48	11.890000	1.59
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.419998	NaN	NaN	19.719999	NaN
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260000	NaN	NaN	21.129999	NaN
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180000	NaN	NaN	11.050000	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120003	NaN	NaN	7.400000	1.50
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469999	15.640000	0.50	8.840000	1.32
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680000	35.000000	NaN	12.110000	1.51
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360001	NaN	NaN	4.890000	1.47
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490002	27.920000	1.70	9.170000	1.50

230568 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: 24758 entries, 5 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     24758 non-null  object 
 1   BEN      24758 non-null  float64
 2   CO       24758 non-null  float64
 3   EBE      24758 non-null  float64
 4   MXY      24758 non-null  float64
 5   NMHC     24758 non-null  float64
 6   NO_2     24758 non-null  float64
 7   NOx      24758 non-null  float64
 8   OXY      24758 non-null  float64
 9   O_3      24758 non-null  float64
 10  PM10     24758 non-null  float64
 11  PM25     24758 non-null  float64
 12  PXY      24758 non-null  float64
 13  SO_2     24758 non-null  float64
 14  TCH      24758 non-null  float64
 15  TOL      24758 non-null  float64
 16  station  24758 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

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

Out[6]:

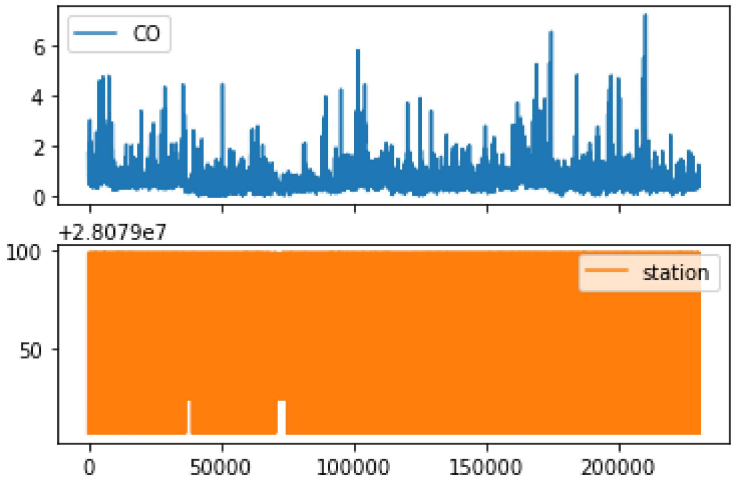
	CO	station
5	1.69	28079006
22	0.79	28079024
25	1.47	28079099
31	0.85	28079006
48	0.79	28079024
...	...	...
230538	0.40	28079024
230541	0.94	28079099
230547	1.06	28079006
230564	0.32	28079024
230567	0.74	28079099

24758 rows × 2 columns

Line chart

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

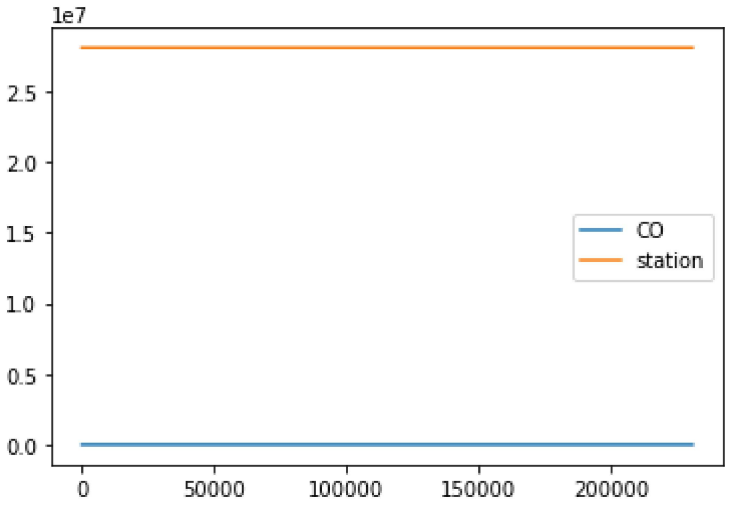
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

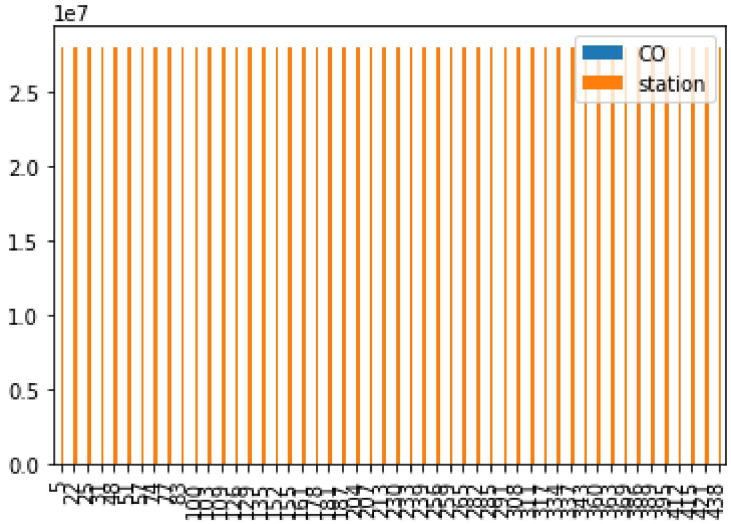


Bar chart

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

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

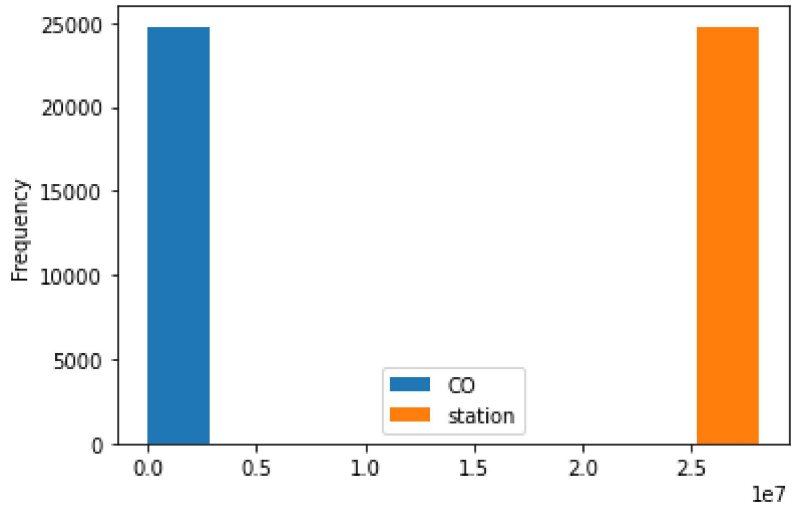
Out[10]: <AxesSubplot:>



Histogram

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

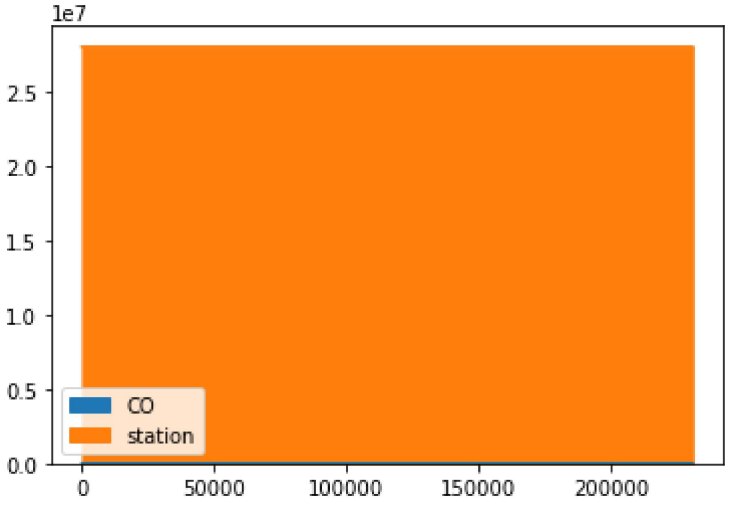
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

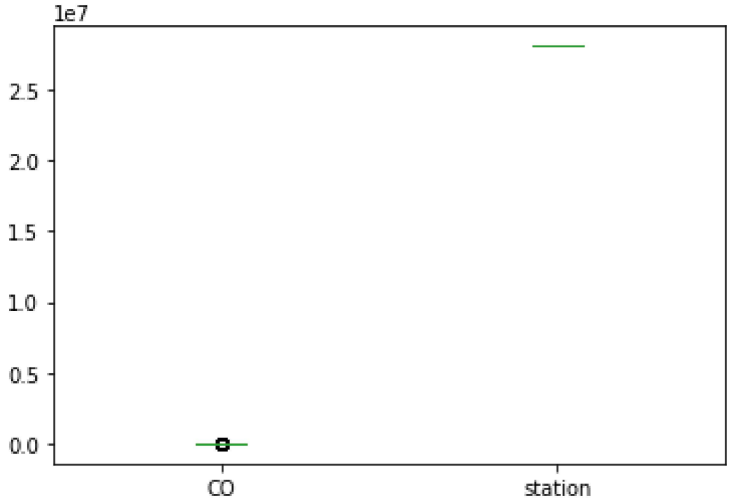
Out[12]: <AxesSubplot:>



Box chart

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

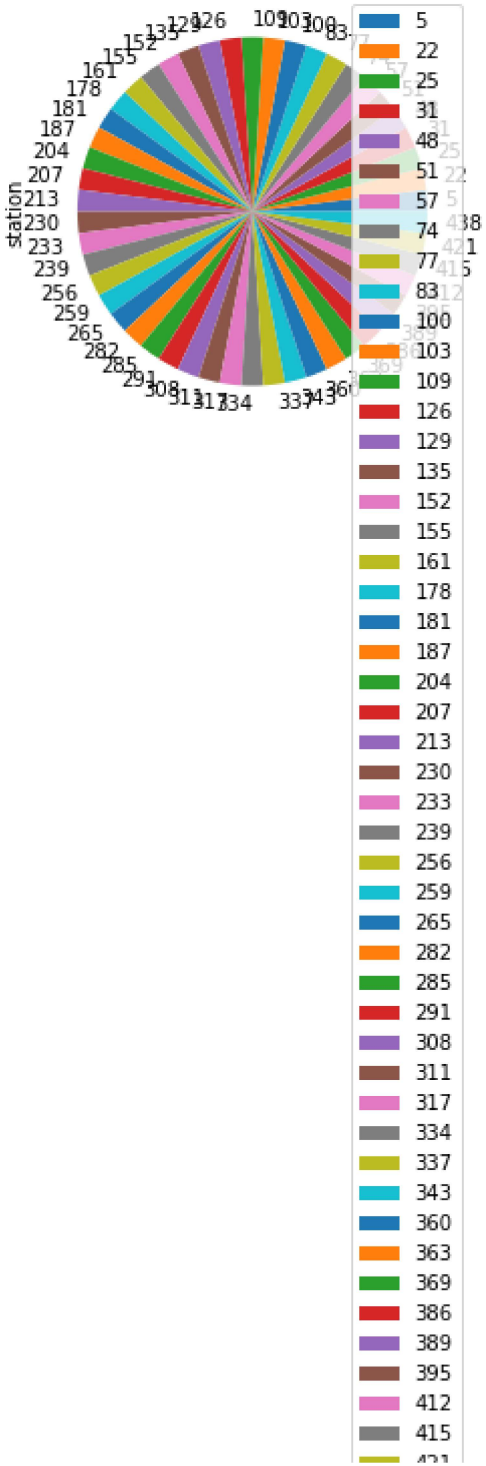
Out[13]: <AxesSubplot:>



Pie chart

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

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

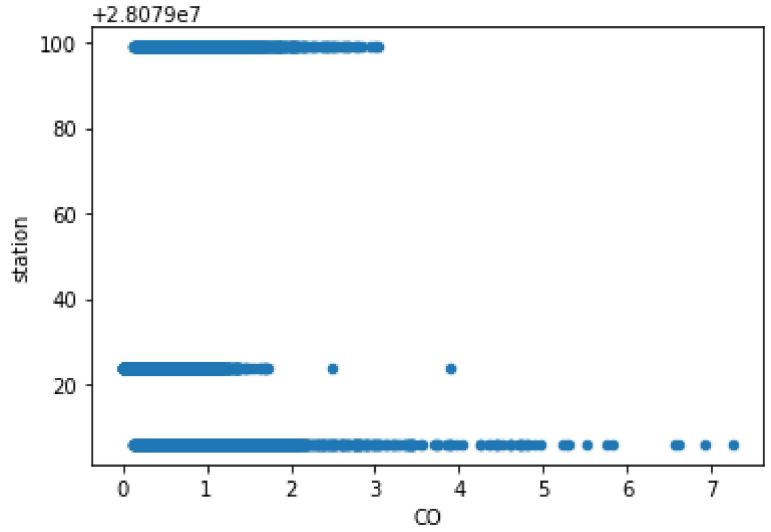

station
5
22
25
31
48
51
57
74
77
83
100
103
109
126
129
135
152
155
161
178
181
187
204
207
213
230
233
239
256
259
265
282
285
291
308
311
317
334
337
343
360
363
369
386
389
395
412
415

421
438

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: 24758 entries, 5 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     24758 non-null  object
 1   BEN      24758 non-null  float64
 2   CO       24758 non-null  float64
 3   EBE      24758 non-null  float64
 4   MXY      24758 non-null  float64
 5   NMHC     24758 non-null  float64
 6   NO_2     24758 non-null  float64
 7   NOx      24758 non-null  float64
 8   OXY      24758 non-null  float64
 9   O_3      24758 non-null  float64
 10  PM10     24758 non-null  float64
 11  PM25     24758 non-null  float64
 12  PXY      24758 non-null  float64
 13  SO_2     24758 non-null  float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000
mean	1.350624	0.600713	1.824534	3.835034	0.176546	58.333481	116.419090	1.990347	37.155685
std	1.541636	0.419048	1.868939	4.069036	0.126683	40.529382	117.557064	1.931620	31.127681
min	0.110000	0.000000	0.170000	0.150000	0.000000	1.680000	2.020000	0.190000	0.310000
25%	0.450000	0.360000	0.810000	1.060000	0.100000	28.450001	36.882501	0.960000	12.110000
50%	0.850000	0.500000	1.130000	2.500000	0.150000	52.959999	85.180000	1.260000	28.675000
75%	1.680000	0.720000	2.160000	5.090000	0.220000	79.347498	158.300003	2.470000	53.029999
max	45.430000	7.250000	57.799999	66.900002	2.020000	461.299988	1680.000000	63.000000	178.899994

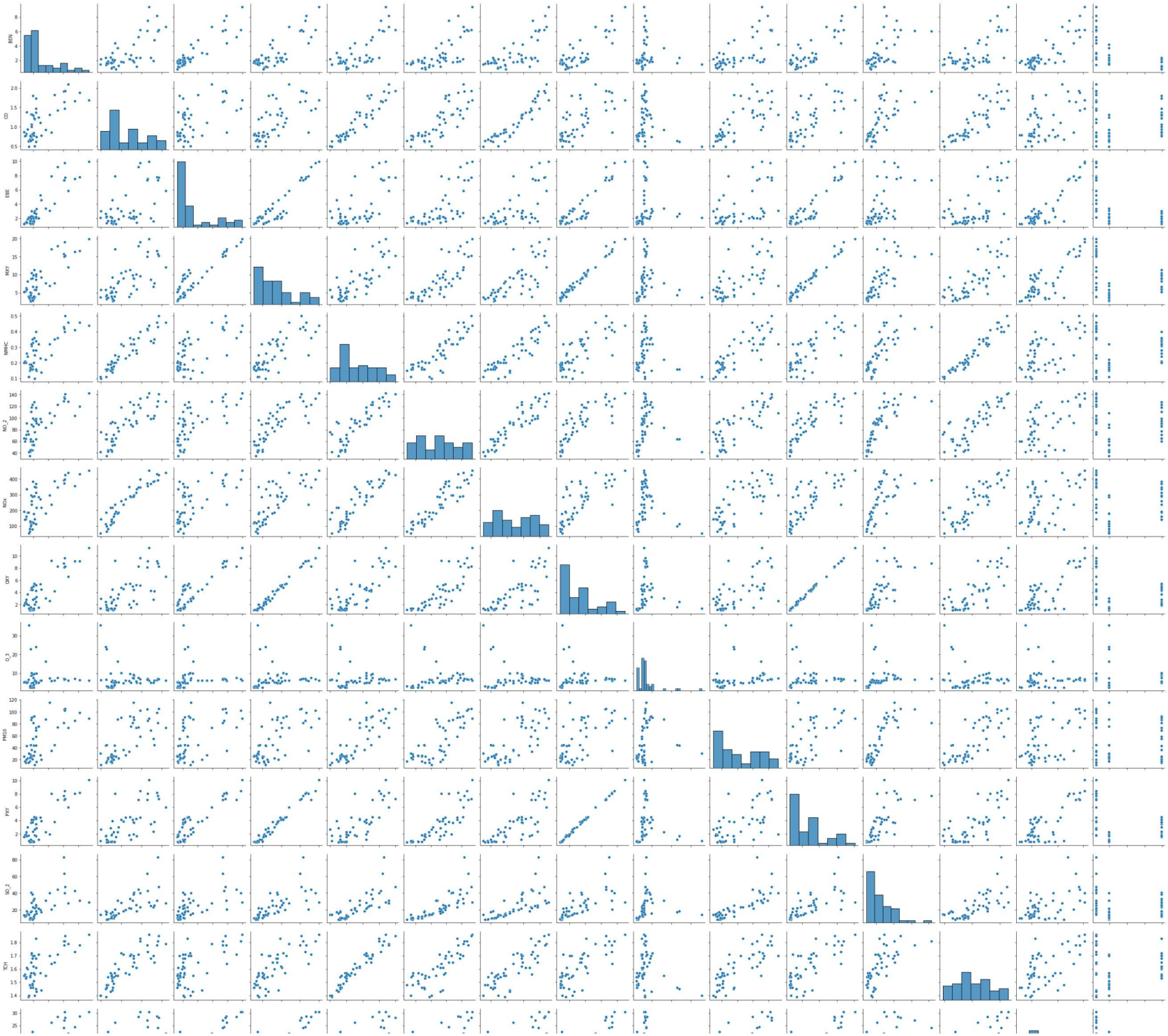
In [18]:

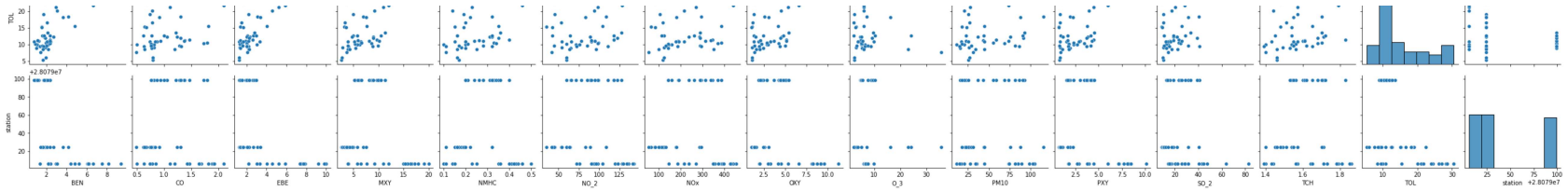
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x11e9d425880>
```

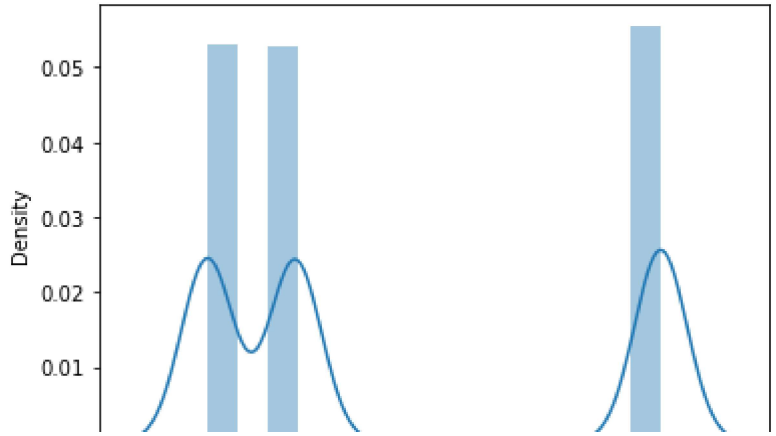

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



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

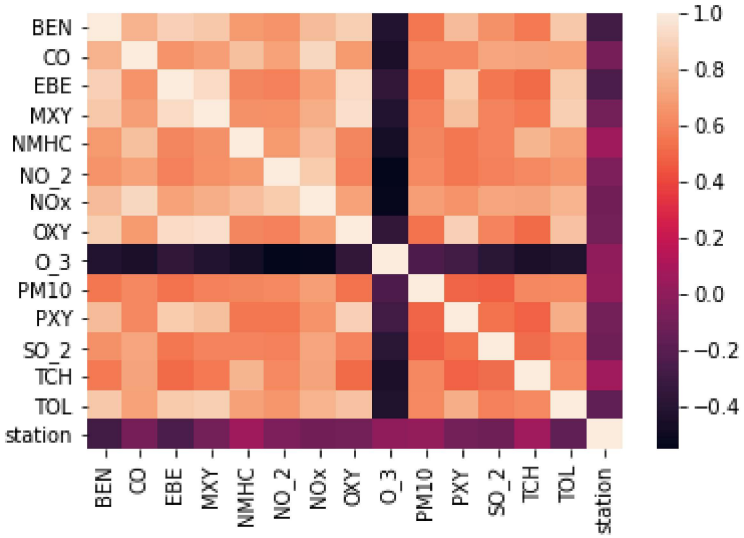
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28079017.64075772
```

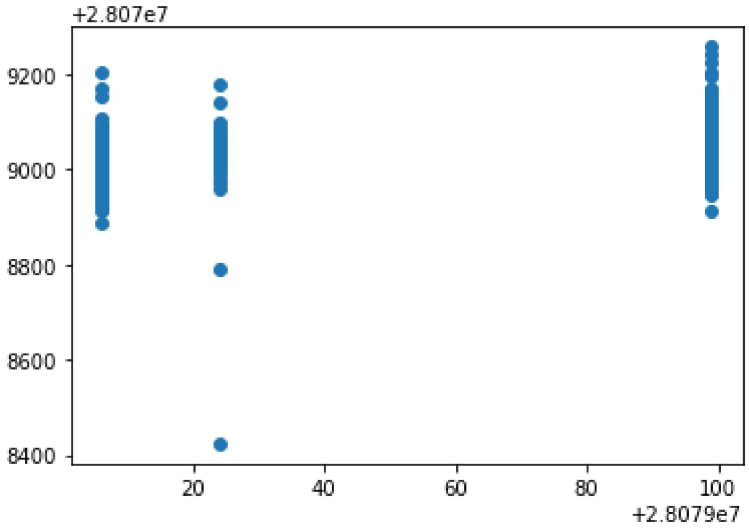
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-18.174056
CO	-12.057900
EBE	-23.022451
MXY	4.652349
NMHC	131.095210
NO_2	-0.034385
NOx	0.000609
OXY	14.122589
O_3	-0.055439
PM10	0.144916
PXY	6.404906
SO_2	-0.630762
TCH	20.843452
TOL	-0.566375

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

Out[27]: <matplotlib.collections.PathCollection at 0x11eabde0b80>



ACCURACY

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

Out[28]: 0.3788685188488047

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

Out[29]: 0.3989029834268095

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]:
```
0.3786409396350162
```

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

Out[33]:
```
0.3982129859207212
```

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]:
```
0.06309829772793729
```

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

Out[36]:
```
0.05729762636752733
```

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([-8.54250558,  0.        , -9.02976539,  3.25611115,  0.42688624,
       -0.01771263,  0.01021728,  3.36745815, -0.12640861,  0.31114115,
        2.74860128, -0.4380658 ,  0.56883655, -1.01138549])
```

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

Out[39]:
```
28079052.05014169
```

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

In [65]:
```
1 en.score(x_test,y_test)
```

Out[65]:
```
0.2319550412034177
```

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)))
```

```
32.39661552047374
1274.3280704097153
35.697732006525506
```

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]: (24758, 14)

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

Out[46]: (24758,)

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]: 0.8741416915744405

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

Out[54]: 3.5557727473608076e-15

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

Out[55]: array([[3.55577275e-15, 7.80743173e-29, 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]:
```
0.8706866705135603
```

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

In [62]:

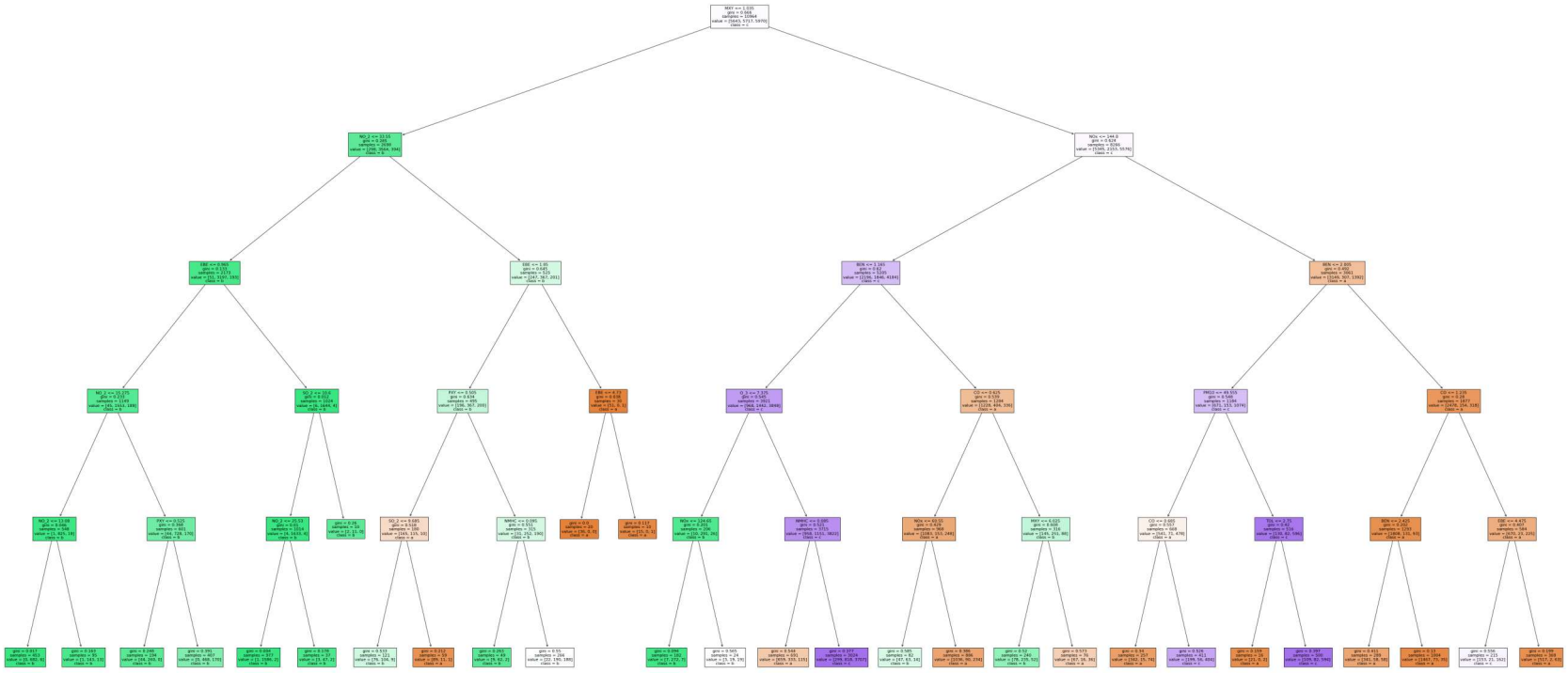
```
from sklearn.tree import plot_tree

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(2108.0, 1993.2, 'MXY <= 1.035\ngini = 0.666\nsamples = 10964\nvalue = [5643, 5717, 5970]\nclass = c'),
 Text(1074.6666666666667, 1630.8000000000002, 'NO_2 <= 33.55\ngini = 0.285\nsamples = 2698\nvalue = [298, 3564, 394]\nclass = b'),
 Text(620.0, 1268.4, 'EBE <= 0.965\ngini = 0.133\nsamples = 2173\nvalue = [51, 3197, 193]\nclass = b'),
 Text(330.6666666666667, 906.0, 'NO_2 <= 15.275\ngini = 0.233\nsamples = 1149\nvalue = [45, 1553, 189]\nclass = b'),
 Text(165.33333333333334, 543.5999999999999, 'NO_2 <= 13.08\ngini = 0.046\nsamples = 548\nvalue = [1, 825, 19]\nclass = b'),
 Text(82.66666666666667, 181.19999999999982, 'gini = 0.017\nsamples = 453\nvalue = [0, 682, 6]\nclass = b'),
 Text(248.0, 181.19999999999982, 'gini = 0.163\nsamples = 95\nvalue = [1, 143, 13]\nclass = b'),
 Text(496.0, 543.5999999999999, 'PXY <= 0.525\ngini = 0.368\nsamples = 601\nvalue = [44, 728, 170]\nclass = b'),
 Text(413.33333333333337, 181.19999999999982, 'gini = 0.248\nsamples = 194\nvalue = [44, 260, 0]\nclass = b'),
 Text(578.6666666666667, 181.19999999999982, 'gini = 0.391\nsamples = 407\nvalue = [0, 468, 170]\nclass = b'),
 Text(909.3333333333334, 906.0, 'SO_2 <= 10.6\ngini = 0.012\nsamples = 1024\nvalue = [6, 1644, 4]\nclass = b'),
 Text(826.6666666666667, 543.5999999999999, 'NO_2 <= 25.53\ngini = 0.01\nsamples = 1014\nvalue = [4, 1633, 4]\nclass = b'),
 Text(744.0, 181.19999999999982, 'gini = 0.004\nsamples = 977\nvalue = [1, 1586, 2]\nclass = b'),
 Text(909.3333333333334, 181.19999999999982, 'gini = 0.178\nsamples = 37\nvalue = [3, 47, 2]\nclass = b'),
 Text(992.0, 543.5999999999999, 'gini = 0.26\nsamples = 10\nvalue = [2, 11, 0]\nclass = b'),
 Text(1529.3333333333335, 1268.4, 'EBE <= 1.85\ngini = 0.645\nsamples = 525\nvalue = [247, 367, 201]\nclass = b'),
 Text(1322.6666666666667, 906.0, 'PXY <= 0.505\ngini = 0.634\nsamples = 495\nvalue = [196, 367, 200]\nclass = b'),
 Text(1157.3333333333335, 543.5999999999999, 'SO_2 <= 9.685\ngini = 0.518\nsamples = 180\nvalue = [165, 115, 10]\nclass = a'),
 Text(1074.6666666666667, 181.19999999999982, 'gini = 0.533\nsamples = 121\nvalue = [76, 104, 9]\nclass = b'),
 Text(1240.0, 181.19999999999982, 'gini = 0.212\nsamples = 59\nvalue = [89, 11, 1]\nclass = a'),
 Text(1488.0, 543.5999999999999, 'NMHC <= 0.095\ngini = 0.551\nsamples = 315\nvalue = [31, 252, 190]\nclass = b'),
 Text(1405.3333333333335, 181.19999999999982, 'gini = 0.263\nsamples = 49\nvalue = [9, 62, 2]\nclass = b'),
 Text(1570.6666666666667, 181.19999999999982, 'gini = 0.55\nsamples = 266\nvalue = [22, 190, 188]\nclass = b'),
 Text(1736.0, 906.0, 'EBE <= 4.73\ngini = 0.038\nsamples = 30\nvalue = [51, 0, 1]\nclass = a'),
 Text(1653.3333333333335, 543.5999999999999, 'gini = 0.0\nsamples = 20\nvalue = [36, 0, 0]\nclass = a'),
 Text(1818.6666666666667, 543.5999999999999, 'gini = 0.117\nsamples = 10\nvalue = [15, 0, 1]\nclass = a'),
 Text(3141.3333333333335, 1630.8000000000002, 'NOx <= 144.8\ngini = 0.624\nsamples = 8266\nvalue = [5345, 2153, 5576]\nclass = c'),
 Text(2480.0, 1268.4, 'BEN <= 1.165\ngini = 0.62\nsamples = 5205\nvalue = [2196, 1846, 4184]\nclass = c'),
```

```
 Text(2149.3333333333335, 906.0, 'O_3 <= 7.375\ngini = 0.545\nsamples = 3921\nvalue = [968, 1442, 3848]\ncla
ss = c'),
 Text(1984.0, 543.5999999999999, 'NOx <= 124.65\ngini = 0.201\nsamples = 206\nvalue = [10, 291, 26]\nclass =
b'),
 Text(1901.3333333333335, 181.19999999999982, 'gini = 0.094\nsamples = 182\nvalue = [7, 272, 7]\nclass =
b'),
 Text(2066.666666666667, 181.19999999999982, 'gini = 0.565\nsamples = 24\nvalue = [3, 19, 19]\nclass = b'),
 Text(2314.666666666667, 543.5999999999999, 'NMHC <= 0.085\ngini = 0.521\nsamples = 3715\nvalue = [958, 115
1, 3822]\nclass = c'),
 Text(2232.0, 181.19999999999982, 'gini = 0.544\nsamples = 691\nvalue = [659, 333, 115]\nclass = a'),
 Text(2397.3333333333335, 181.19999999999982, 'gini = 0.377\nsamples = 3024\nvalue = [299, 818, 3707]\nclass
= c'),
 Text(2810.666666666667, 906.0, 'CO <= 0.625\ngini = 0.539\nsamples = 1284\nvalue = [1228, 404, 336]\nclass
= a'),
 Text(2645.3333333333335, 543.5999999999999, 'NOx <= 60.55\ngini = 0.429\nsamples = 968\nvalue = [1083, 153,
248]\nclass = a'),
 Text(2562.666666666667, 181.19999999999982, 'gini = 0.585\nsamples = 82\nvalue = [47, 63, 14]\nclass = b'),
 Text(2728.0, 181.19999999999982, 'gini = 0.386\nsamples = 886\nvalue = [1036, 90, 234]\nclass = a'),
 Text(2976.0, 543.5999999999999, 'MXY <= 6.025\ngini = 0.608\nsamples = 316\nvalue = [145, 251, 88]\nclass =
b'),
 Text(2893.3333333333335, 181.19999999999982, 'gini = 0.52\nsamples = 240\nvalue = [78, 235, 52]\nclass =
b'),
 Text(3058.666666666667, 181.19999999999982, 'gini = 0.573\nsamples = 76\nvalue = [67, 16, 36]\nclass = a'),
 Text(3802.666666666667, 1268.4, 'BEN <= 2.005\ngini = 0.492\nsamples = 3061\nvalue = [3149, 307, 1392]\ncla
ss = a'),
 Text(3472.0, 906.0, 'PM10 <= 49.555\ngini = 0.548\nsamples = 1184\nvalue = [671, 153, 1074]\nclass = c'),
 Text(3306.666666666667, 543.5999999999999, 'CO <= 0.685\ngini = 0.557\nsamples = 668\nvalue = [541, 71, 47
8]\nclass = a'),
 Text(3224.0, 181.19999999999982, 'gini = 0.34\nsamples = 257\nvalue = [342, 15, 74]\nclass = a'),
 Text(3389.3333333333335, 181.19999999999982, 'gini = 0.526\nsamples = 411\nvalue = [199, 56, 404]\nclass =
c'),
 Text(3637.3333333333335, 543.5999999999999, 'TOL <= 2.75\ngini = 0.42\nsamples = 516\nvalue = [130, 82, 59
6]\nclass = c'),
 Text(3554.666666666667, 181.19999999999982, 'gini = 0.159\nsamples = 16\nvalue = [21, 0, 2]\nclass = a'),
 Text(3720.0, 181.19999999999982, 'gini = 0.397\nsamples = 500\nvalue = [109, 82, 594]\nclass = c'),
 Text(4133.333333333334, 906.0, 'CO <= 1.235\ngini = 0.28\nsamples = 1877\nvalue = [2478, 154, 318]\nclass =
a'),
 Text(3968.0, 543.5999999999999, 'BEN <= 2.425\ngini = 0.202\nsamples = 1293\nvalue = [1808, 131, 93]\nclass
= a'),
 Text(3885.3333333333335, 181.19999999999982, 'gini = 0.411\nsamples = 289\nvalue = [341, 58, 58]\nclass =
a'),
 Text(4050.666666666667, 181.19999999999982, 'gini = 0.13\nsamples = 1004\nvalue = [1467, 73, 35]\nclass =
a'),
```

```
 Text(4298.666666666667, 543.5999999999999, 'EBE <= 4.475\ngini = 0.407\nsamples = 584\nvalue = [670, 23, 225]\nclass = a'),
 Text(4216.0, 181.19999999999982, 'gini = 0.556\nsamples = 215\nvalue = [153, 21, 162]\nclass = c'),
 Text(4381.333333333334, 181.19999999999982, 'gini = 0.199\nsamples = 369\nvalue = [517, 2, 63]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression0.3989029834268095

Ridge Regression:0.3982129859207212

Lasso Regression:0.06309829772793729

ElasticNet Regression:0.2319550412034177

Logistic Regression:0.8741416915744405

Random Forest:0.8706866705135603

Logistic Regression is suitable for this dataset