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_2009")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH	TOL	s
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.260000	NaN	NaN	5.55	NaN	NaN	280
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.580000	NaN	NaN	8.84	NaN	NaN	280
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.190001	NaN	NaN	6.98	NaN	NaN	280
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.530001	6.82	1.30	8.88	1.38	4.62	280
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.760000	NaN	NaN	7.82	1.41	NaN	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.830000	7.15	0.74	6.25	1.25	0.69	280
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.920000	NaN	NaN	4.90	NaN	NaN	280
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.460000	6.77	NaN	8.40	1.34	0.13	280
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.030000	NaN	NaN	7.84	1.42	1.87	280
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.360000	11.61	0.83	6.93	1.34	2.02	280

215688 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: 24717 entries, 3 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     24717 non-null  object 
 1   BEN      24717 non-null  float64
 2   CO       24717 non-null  float64
 3   EBE      24717 non-null  float64
 4   MXY      24717 non-null  float64
 5   NMHC     24717 non-null  float64
 6   NO_2     24717 non-null  float64
 7   NOx      24717 non-null  float64
 8   OXY      24717 non-null  float64
 9   O_3      24717 non-null  float64
 10  PM10     24717 non-null  float64
 11  PM25     24717 non-null  float64
 12  PXY      24717 non-null  float64
 13  SO_2     24717 non-null  float64
 14  TCH      24717 non-null  float64
 15  TOL      24717 non-null  float64
 16  station  24717 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

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

Out[6]:

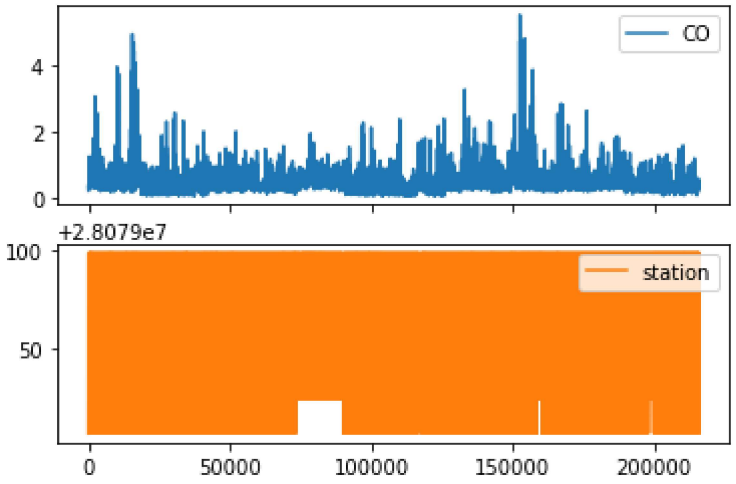
	CO	station
3	0.33	28079006
20	0.32	28079024
24	0.24	28079099
28	0.21	28079006
45	0.30	28079024
...	...	...
215659	0.27	28079024
215663	0.35	28079099
215667	0.29	28079006
215683	0.22	28079024
215687	0.32	28079099

24717 rows × 2 columns

Line chart

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

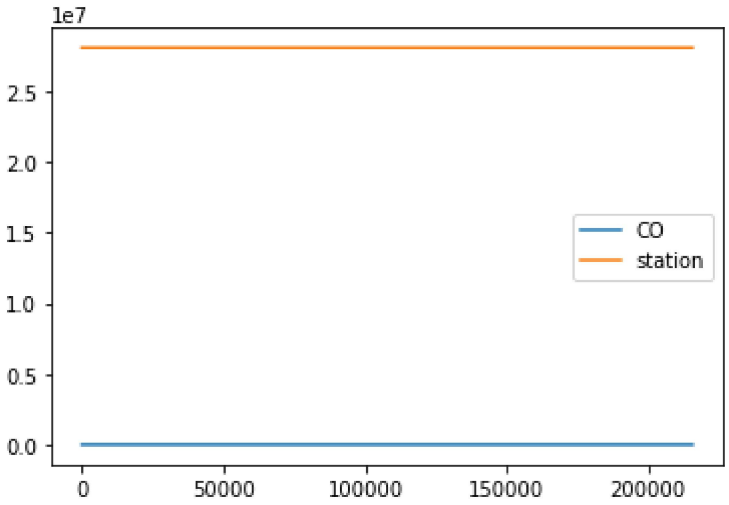
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

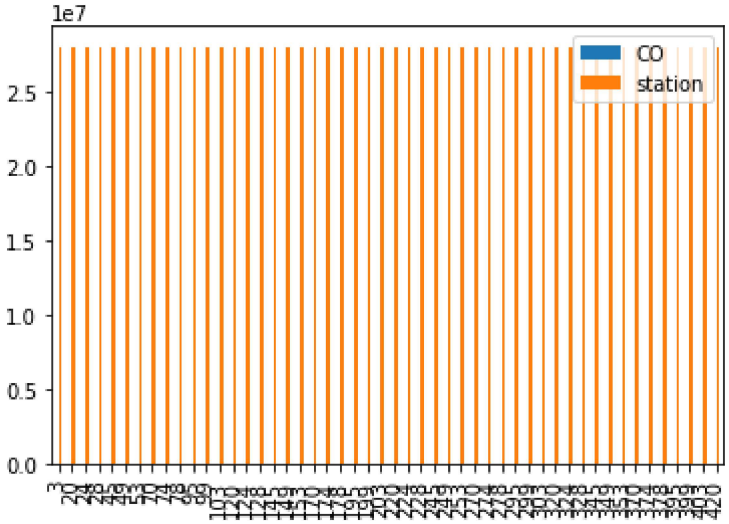


Bar chart

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

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

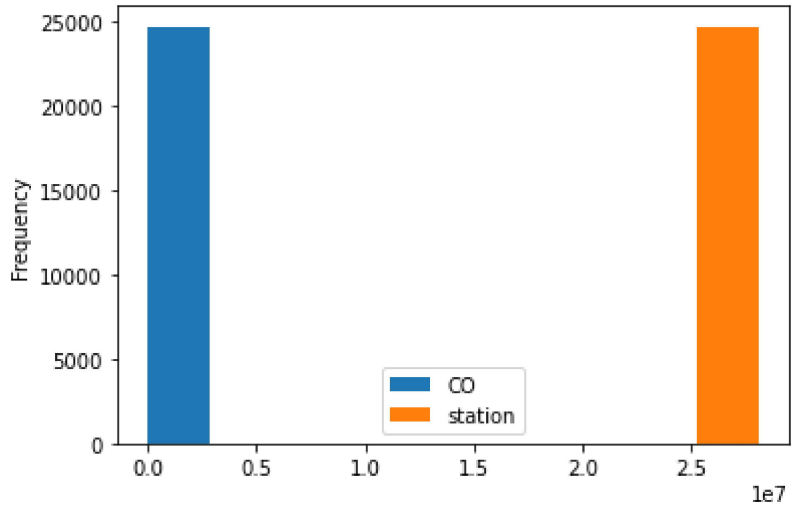
Out[10]: <AxesSubplot:>



Histogram

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

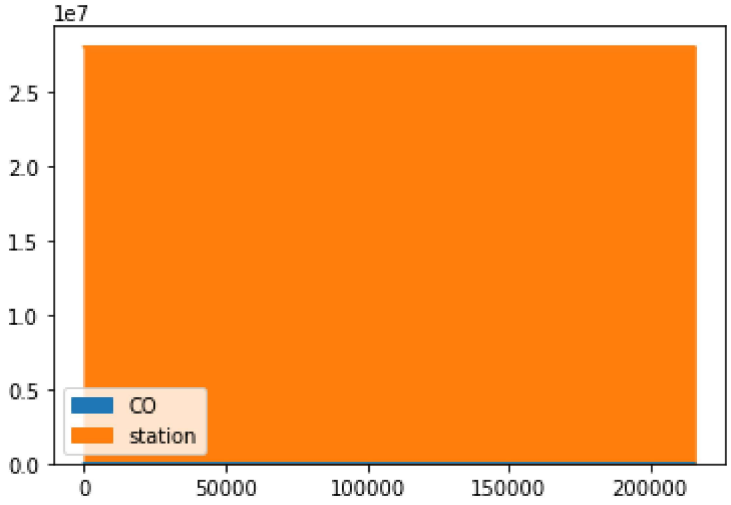
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

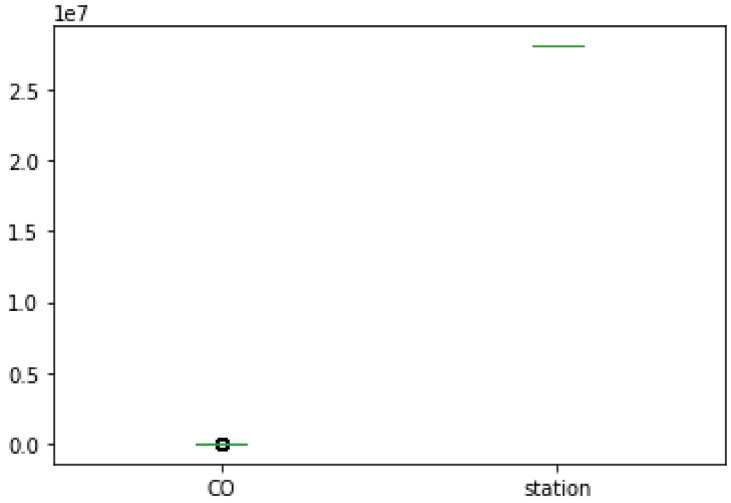
Out[12]: <AxesSubplot:>



Box chart

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

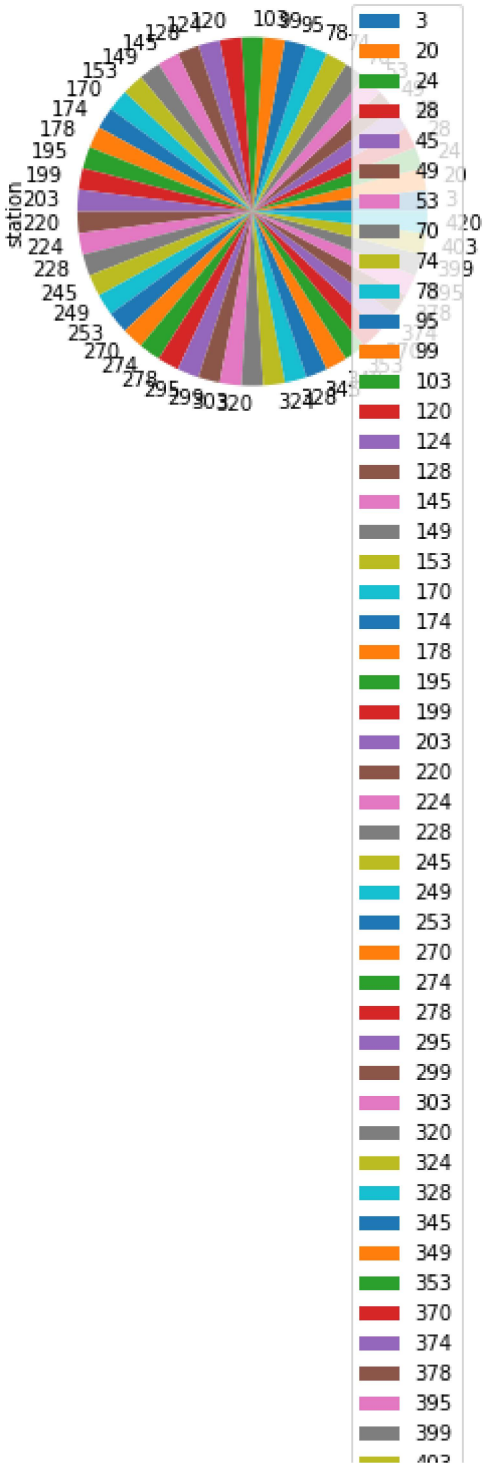
Out[13]: <AxesSubplot:>

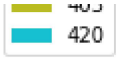


Pie chart

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

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

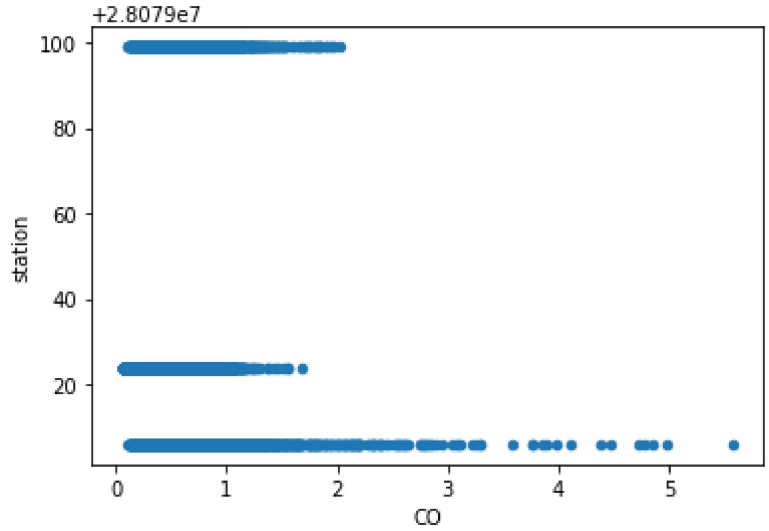

station
3
20
24
28
45
49
53
70
74
78
95
99
103
120
124
128
145
149
153
170
174
178
195
199
203
220
224
228
245
249
253
270
274
278
295
299
303
320
324
328
345
349
353
370
374
378
395
399



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: 24717 entries, 3 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     24717 non-null  object
 1   BEN      24717 non-null  float64
 2   CO       24717 non-null  float64
 3   EBE      24717 non-null  float64
 4   MXY      24717 non-null  float64
 5   NMHC     24717 non-null  float64
 6   NO_2     24717 non-null  float64
 7   NOx      24717 non-null  float64
 8   OXY      24717 non-null  float64
 9   O_3      24717 non-null  float64
 10  PM10     24717 non-null  float64
 11  PM25     24717 non-null  float64
 12  PXY      24717 non-null  float64
 13  SO_2     24717 non-null  float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000
mean	1.010583	0.448056	1.262430	2.244469	0.219582	55.563929	92.907188	1.356536	46.377159
std	1.007345	0.291706	1.074768	2.242214	0.141661	38.911677	91.985352	1.078515	31.205291
min	0.170000	0.060000	0.250000	0.240000	0.000000	0.600000	2.250000	0.150000	0.600000
25%	0.460000	0.270000	0.720000	0.990000	0.140000	26.510000	33.009998	0.870000	20.150000
50%	0.670000	0.370000	1.000000	1.490000	0.190000	47.930000	67.010002	1.000000	43.090000
75%	1.180000	0.570000	1.430000	2.820000	0.260000	76.269997	124.699997	1.550000	66.720001
max	22.379999	5.570000	47.669998	56.500000	2.580000	477.399994	1438.000000	45.349998	179.100006

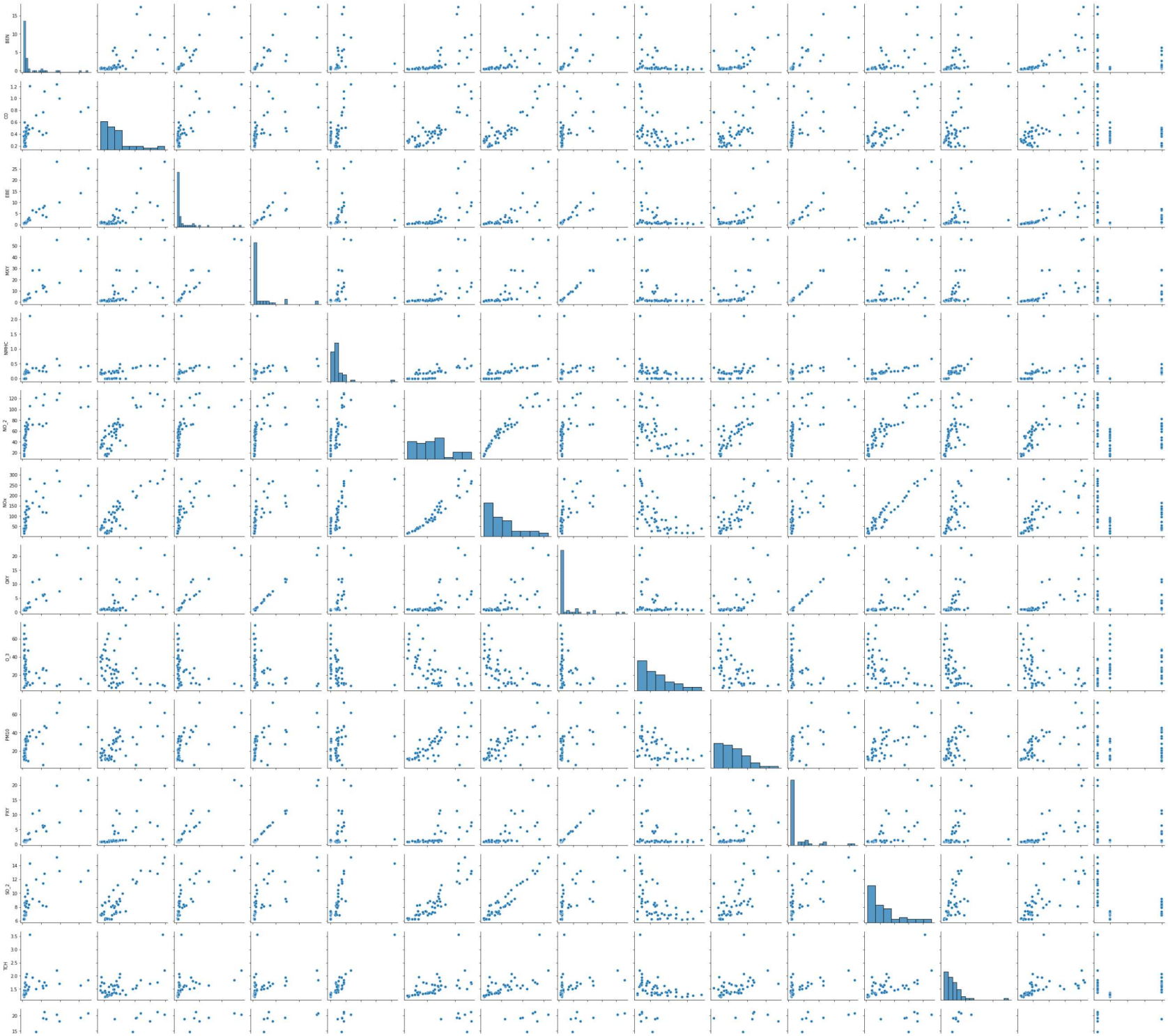
In [18]:

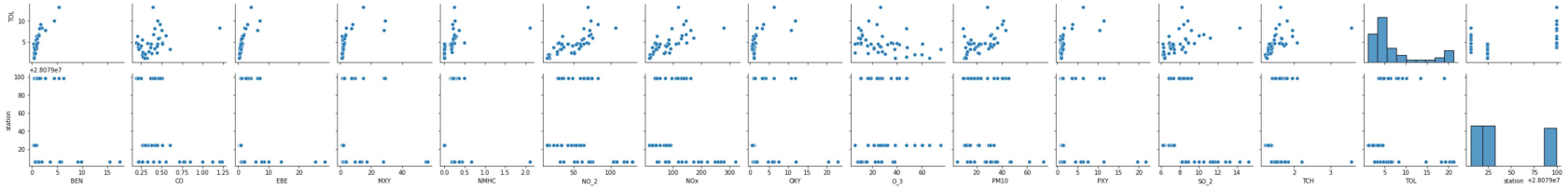
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1c4caa2f760>
```

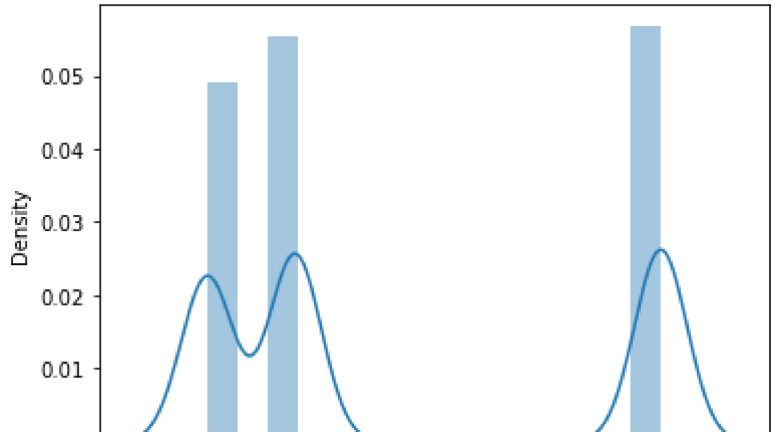

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



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

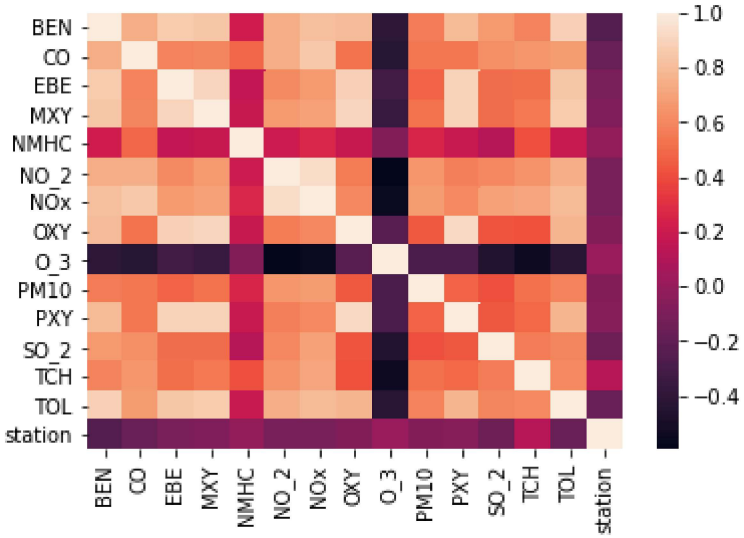
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28078903.997701336
```

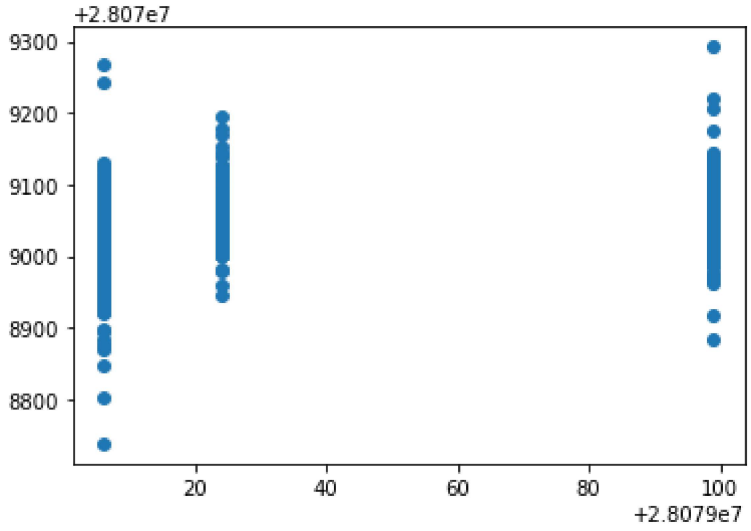
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-37.639974
CO	-29.330544
EBE	5.593453
MXY	-1.198743
NMHC	-13.264679
NO_2	-0.152884
NOx	0.199587
OXY	12.571548
O_3	0.017792
PM10	-0.064669
PXY	5.229602
SO_2	-0.307043
TCH	115.467407
TOL	-0.903509

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

Out[27]: <matplotlib.collections.PathCollection at 0x1c4d4614f10>



ACCURACY

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

Out[28]: 0.28033478363767506

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

Out[29]: 0.28926214656030325

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]: 0.27939171571102484

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

Out[33]: 0.2889861030207309

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]: 0.03703086015497614

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

Out[36]: 0.03503066696327528

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

Out[37]: ElasticNet()

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

Out[38]:
```
array([-7.00297263, -0.59669806,  0.08263944,  2.12600003,  0.        ,
       -0.22366682,  0.13035356,  1.30028136, -0.14512594,  0.06203972,
        2.16000957, -0.82415411,  1.47639194, -2.00821059])
```

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

Out[39]:
```
28079064.116266795
```

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

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

Out[41]:
```
0.1044671536609102
```

Evaluation Metrics

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

```
35.7669636926457
1467.2045652486788
38.304106375801
```

Logistic Regression

In [43]:
```
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]: (24717, 14)

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

Out[46]: (24717,)

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]: 0.8951733624630821

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

Out[54]: 5.447205522232353e-13

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

Out[55]: array([[5.44720552e-13, 8.28692830e-44, 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]:
```
0.8983877502884827
```

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

In [62]:

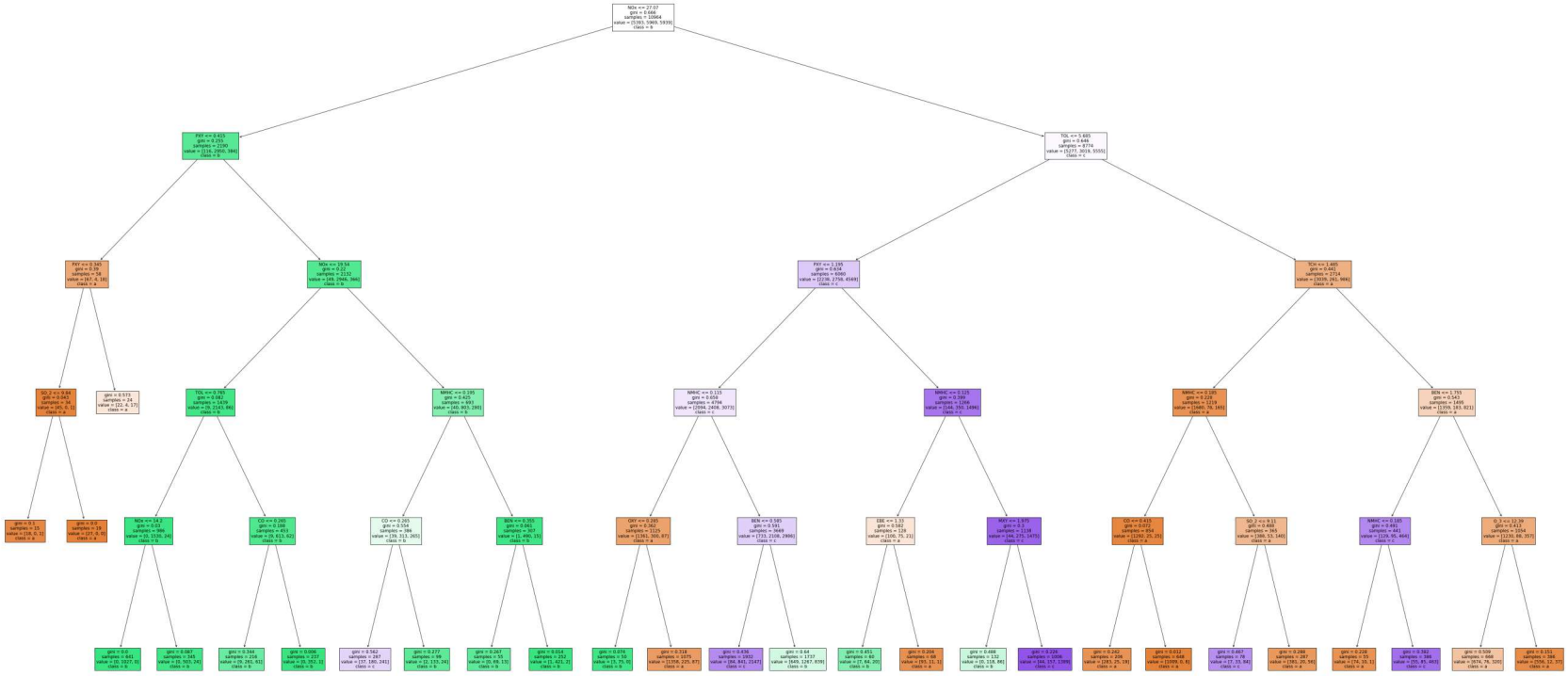
```
from sklearn.tree import plot_tree

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(1838.1176470588236, 1993.2, 'NOx <= 27.07\ngini = 0.666\nsamples = 10964\nvalue = [5393, 5969, 5939]\nclass = b'),
 Text(612.7058823529412, 1630.8000000000002, 'PXY <= 0.415\ngini = 0.255\nsamples = 2190\nvalue = [116, 2950, 384]\nclass = b'),
 Text(262.5882352941177, 1268.4, 'PXY <= 0.345\ngini = 0.39\nsamples = 58\nvalue = [67, 4, 18]\nclass = a'),
 Text(175.05882352941177, 906.0, 'SO_2 <= 9.84\ngini = 0.043\nsamples = 34\nvalue = [45, 0, 1]\nclass = a'),
 Text(87.52941176470588, 543.5999999999999, 'gini = 0.1\nsamples = 15\nvalue = [18, 0, 1]\nclass = a'),
 Text(262.5882352941177, 543.5999999999999, 'gini = 0.0\nsamples = 19\nvalue = [27, 0, 0]\nclass = a'),
 Text(350.11764705882354, 906.0, 'gini = 0.573\nsamples = 24\nvalue = [22, 4, 17]\nclass = a'),
 Text(962.8235294117648, 1268.4, 'NOx <= 19.54\ngini = 0.22\nsamples = 2132\nvalue = [49, 2946, 366]\nclass = b'),
 Text(612.7058823529412, 906.0, 'TOL <= 0.765\ngini = 0.082\nsamples = 1439\nvalue = [9, 2143, 86]\nclass = b'),
 Text(437.6470588235294, 543.5999999999999, 'NOx <= 14.2\ngini = 0.03\nsamples = 986\nvalue = [0, 1530, 24]\nclass = b'),
 Text(350.11764705882354, 181.19999999999982, 'gini = 0.0\nsamples = 641\nvalue = [0, 1027, 0]\nclass = b'),
 Text(525.1764705882354, 181.19999999999982, 'gini = 0.087\nsamples = 345\nvalue = [0, 503, 24]\nclass = b'),
 Text(787.7647058823529, 543.5999999999999, 'CO <= 0.265\ngini = 0.188\nsamples = 453\nvalue = [9, 613, 62]\nclass = b'),
 Text(700.2352941176471, 181.19999999999982, 'gini = 0.344\nsamples = 216\nvalue = [9, 261, 61]\nclass = b'),
 Text(875.2941176470588, 181.19999999999982, 'gini = 0.006\nsamples = 237\nvalue = [0, 352, 1]\nclass = b'),
 Text(1312.9411764705883, 906.0, 'NMHC <= 0.195\ngini = 0.425\nsamples = 693\nvalue = [40, 803, 280]\nclass = b'),
 Text(1137.8823529411766, 543.5999999999999, 'CO <= 0.265\ngini = 0.554\nsamples = 386\nvalue = [39, 313, 265]\nclass = b'),
 Text(1050.3529411764707, 181.19999999999982, 'gini = 0.562\nsamples = 287\nvalue = [37, 180, 241]\nclass = c'),
 Text(1225.4117647058824, 181.19999999999982, 'gini = 0.277\nsamples = 99\nvalue = [2, 133, 24]\nclass = b'),
 Text(1488.0, 543.5999999999999, 'BEN <= 0.355\ngini = 0.061\nsamples = 307\nvalue = [1, 490, 15]\nclass = b'),
 Text(1400.4705882352941, 181.19999999999982, 'gini = 0.267\nsamples = 55\nvalue = [0, 69, 13]\nclass = b'),
 Text(1575.5294117647059, 181.19999999999982, 'gini = 0.014\nsamples = 252\nvalue = [1, 421, 2]\nclass = b'),
 Text(3063.529411764706, 1630.8000000000002, 'TOL <= 5.685\ngini = 0.646\nsamples = 8774\nvalue = [5277, 3019, 5555]\nclass = c'),
 Text(2363.294117647059, 1268.4, 'PXY <= 1.195\ngini = 0.634\nsamples = 6060\nvalue = [2238, 2758, 4569]\nclass = c'),
 Text(2013.1764705882354, 906.0, 'NMHC <= 0.115\ngini = 0.658\nsamples = 4794\nvalue = [2094, 2408, 3073]\nclass = c'),
 Text(1838.1176470588236, 543.5999999999999, 'OXY <= 0.285\ngini = 0.362\nsamples = 1125\nvalue = [1361, 30

```
0, 87]\nclass = a'),
 Text(1750.5882352941176, 181.19999999999982, 'gini = 0.074\nsamples = 50\nvalue = [3, 75, 0]\nclass = b'),
 Text(1925.6470588235295, 181.19999999999982, 'gini = 0.318\nsamples = 1075\nvalue = [1358, 225, 87]\nclass
= a'),
 Text(2188.2352941176473, 543.5999999999999, 'BEN <= 0.585\ngini = 0.591\nsamples = 3669\nvalue = [733, 210
8, 2986]\nclass = c'),
 Text(2100.7058823529414, 181.19999999999982, 'gini = 0.436\nsamples = 1932\nvalue = [84, 841, 2147]\nclass
= c'),
 Text(2275.764705882353, 181.19999999999982, 'gini = 0.64\nsamples = 1737\nvalue = [649, 1267, 839]\nclass =
b'),
 Text(2713.4117647058824, 906.0, 'NMHC <= 0.125\ngini = 0.399\nsamples = 1266\nvalue = [144, 350, 1496]\ncla
ss = c'),
 Text(2538.3529411764707, 543.5999999999999, 'EBE <= 1.33\ngini = 0.582\nsamples = 128\nvalue = [100, 75, 2
1]\nclass = a'),
 Text(2450.823529411765, 181.19999999999982, 'gini = 0.451\nsamples = 60\nvalue = [7, 64, 20]\nclass = b'),
 Text(2625.8823529411766, 181.19999999999982, 'gini = 0.204\nsamples = 68\nvalue = [93, 11, 1]\nclass = a'),
 Text(2888.470588235294, 543.5999999999999, 'MXY <= 1.975\ngini = 0.3\nsamples = 1138\nvalue = [44, 275, 147
5]\nclass = c'),
 Text(2800.9411764705883, 181.19999999999982, 'gini = 0.488\nsamples = 132\nvalue = [0, 118, 86]\nclass =
b'),
 Text(2976.0, 181.19999999999982, 'gini = 0.226\nsamples = 1006\nvalue = [44, 157, 1389]\nclass = c'),
 Text(3763.764705882353, 1268.4, 'TCH <= 1.485\ngini = 0.441\nsamples = 2714\nvalue = [3039, 261, 986]\nclas
s = a'),
 Text(3413.6470588235293, 906.0, 'NMHC <= 0.185\ngini = 0.228\nsamples = 1219\nvalue = [1680, 78, 165]\nclas
s = a'),
 Text(3238.5882352941176, 543.5999999999999, 'CO <= 0.415\ngini = 0.072\nsamples = 854\nvalue = [1292, 25, 2
5]\nclass = a'),
 Text(3151.0588235294117, 181.19999999999982, 'gini = 0.242\nsamples = 206\nvalue = [283, 25, 19]\nclass =
a'),
 Text(3326.1176470588234, 181.19999999999982, 'gini = 0.012\nsamples = 648\nvalue = [1009, 0, 6]\nclass =
a'),
 Text(3588.7058823529414, 543.5999999999999, 'SO_2 <= 9.11\ngini = 0.488\nsamples = 365\nvalue = [388, 53, 1
40]\nclass = a'),
 Text(3501.176470588235, 181.19999999999982, 'gini = 0.467\nsamples = 78\nvalue = [7, 33, 84]\nclass = c'),
 Text(3676.2352941176473, 181.19999999999982, 'gini = 0.288\nsamples = 287\nvalue = [381, 20, 56]\nclass =
a'),
 Text(4113.882352941177, 906.0, 'BEN <= 1.755\ngini = 0.543\nsamples = 1495\nvalue = [1359, 183, 821]\nclass
= a'),
 Text(3938.823529411765, 543.5999999999999, 'NMHC <= 0.185\ngini = 0.491\nsamples = 441\nvalue = [129, 95, 4
64]\nclass = c'),
 Text(3851.294117647059, 181.19999999999982, 'gini = 0.228\nsamples = 55\nvalue = [74, 10, 1]\nclass = a'),
 Text(4026.3529411764707, 181.19999999999982, 'gini = 0.382\nsamples = 386\nvalue = [55, 85, 463]\nclass =
c'),
```

```
 Text(4288.941176470588, 543.5999999999999, 'O_3 <= 12.39\ngini = 0.413\nsamples = 1054\nvalue = [1230, 88,
357]\nclass = a'),
 Text(4201.411764705883, 181.19999999999982, 'gini = 0.509\nsamples = 668\nvalue = [674, 76, 320]\nclass =
a'),
 Text(4376.470588235295, 181.19999999999982, 'gini = 0.151\nsamples = 386\nvalue = [556, 12, 37]\nclass =
a')]
```



Conclusion

Accuracy

Linear Regression0.28926214656030325

Ridge Regression:0.2889861030207309

Lasso Regression:0.03703086015497614

ElasticNet Regression:0.1044671536609102

Logistic Regression:0.8951733624630821

Random Forest:0.8983877502884827

Random Forest is suitable for this dataset