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_2001")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH	TOL	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.000000	NaN	6.340000	NaN	NaN	28(
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.110000	1.24	10.82	28(
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.099998	NaN	7.850000	NaN	NaN	28(
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.779999	NaN	6.460000	NaN	NaN	28(
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.180000	NaN	8.800000	NaN	NaN	28(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.880001	NaN	39.910000	NaN	28.35	28(
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.809999	NaN	13.450000	1.32	16.08	28(
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.770000	0.61	14.700000	1.40	1.55	28(
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.889999	4.31	39.919998	NaN	20.75	28(
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.369999	4.95	27.340000	1.41	22.27	28(

217872 rows × 16 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', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29669 entries, 1 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     29669 non-null  object 
 1   BEN      29669 non-null  float64
 2   CO       29669 non-null  float64
 3   EBE      29669 non-null  float64
 4   MXY      29669 non-null  float64
 5   NMHC     29669 non-null  float64
 6   NO_2     29669 non-null  float64
 7   NOx      29669 non-null  float64
 8   OXY      29669 non-null  float64
 9   O_3      29669 non-null  float64
 10  PM10     29669 non-null  float64
 11  PXY      29669 non-null  float64
 12  SO_2     29669 non-null  float64
 13  TCH      29669 non-null  float64
 14  TOL      29669 non-null  float64
 15  station  29669 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 3.8+ MB
```

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

Out[6]:

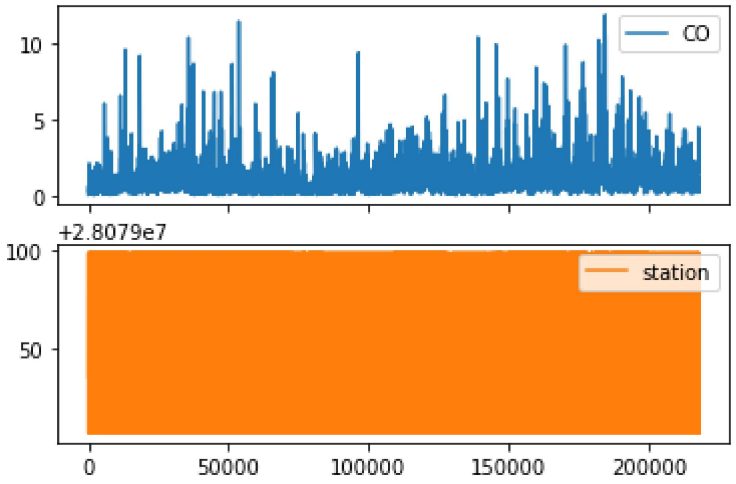
	CO	station
1	0.34	28079035
5	0.63	28079006
21	0.43	28079024
23	0.34	28079099
25	0.06	28079035
...	...	...
217829	4.48	28079006
217847	2.65	28079099
217849	1.22	28079035
217853	1.83	28079006
217871	1.62	28079099

29669 rows × 2 columns

Line chart

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

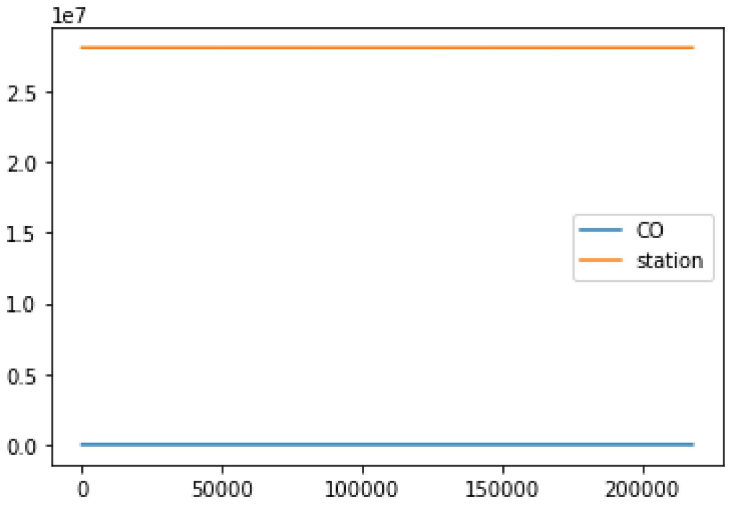
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

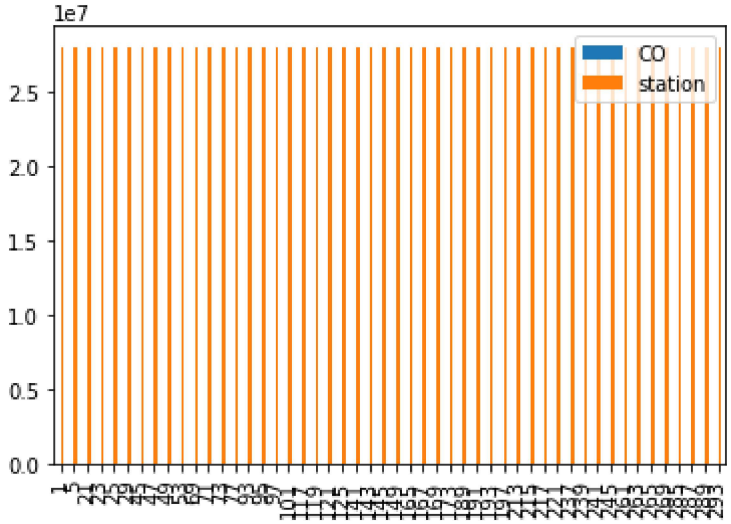


Bar chart

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

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

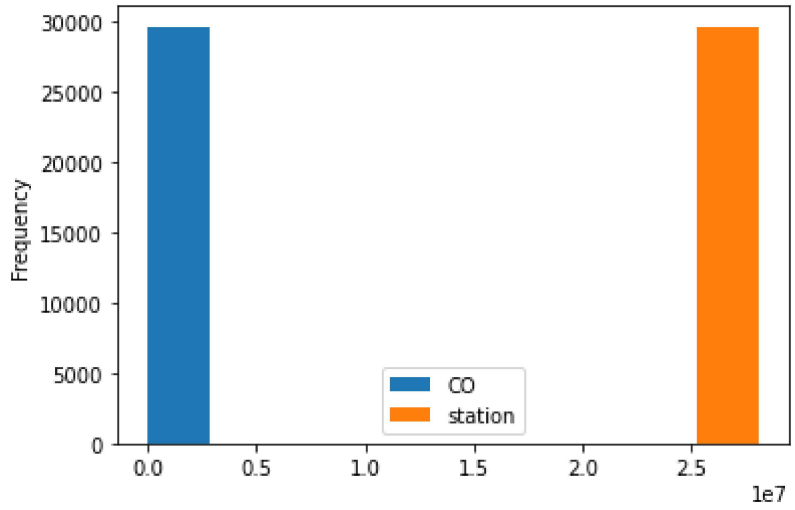
Out[10]: <AxesSubplot:>



Histogram

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

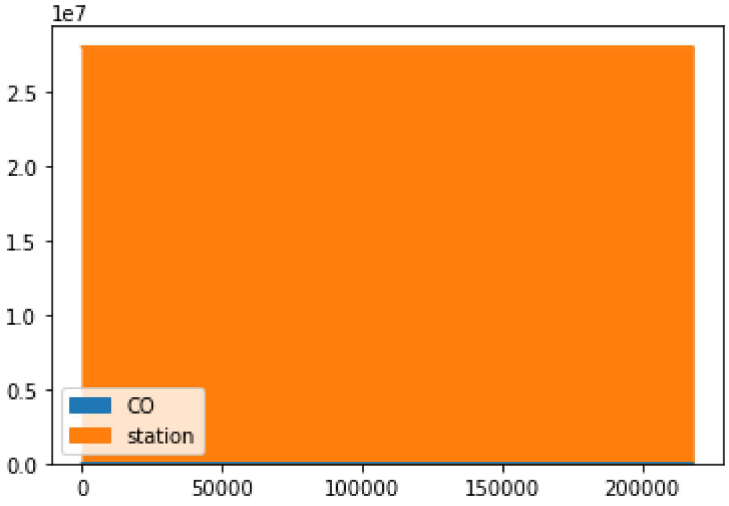
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

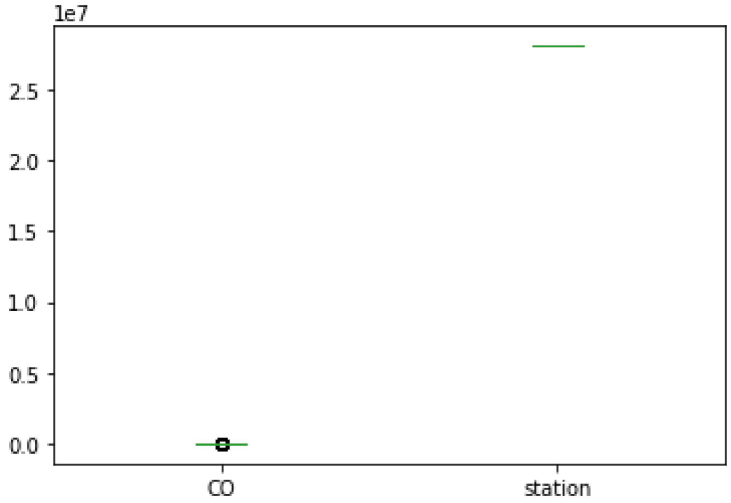
Out[12]: <AxesSubplot:>



Box chart

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

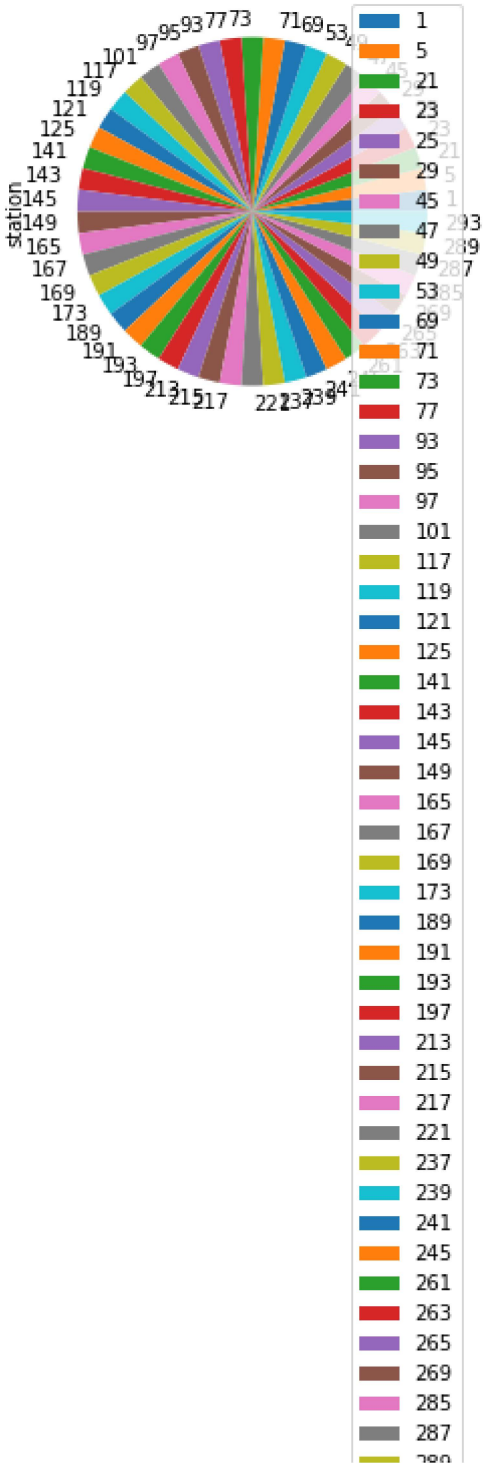
Out[13]: <AxesSubplot:>



Pie chart

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

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

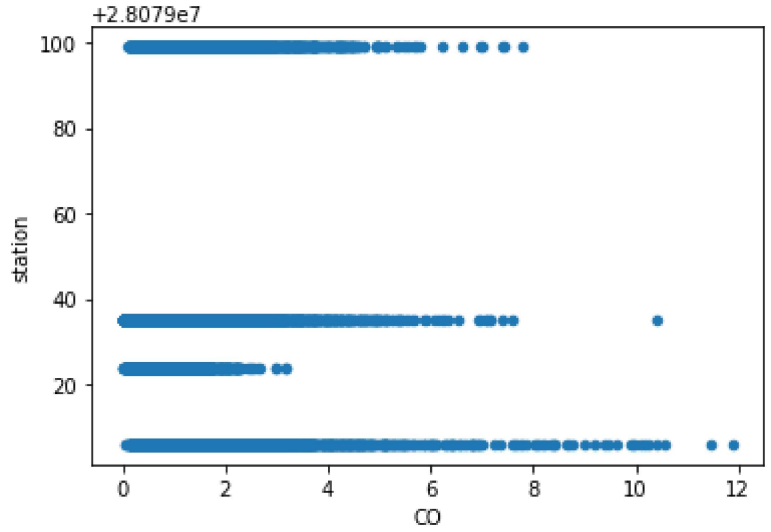

station
1
5
21
23
25
29
45
47
49
53
69
71
73
77
93
95
97
101
117
119
121
125
141
143
145
149
165
167
169
173
189
191
193
197
213
215
217
221
237
239
241
245
261
263
265
269
285
287
289

289
293

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: 29669 entries, 1 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     29669 non-null  object
 1   BEN      29669 non-null  float64
 2   CO       29669 non-null  float64
 3   EBE      29669 non-null  float64
 4   MXY      29669 non-null  float64
 5   NMHC     29669 non-null  float64
 6   NO_2     29669 non-null  float64
 7   NOx      29669 non-null  float64
 8   OXY      29669 non-null  float64
 9   O_3      29669 non-null  float64
 10  PM10     29669 non-null  float64
 11  PXY      29669 non-null  float64
 12  SO_2     29669 non-null  float64
 13  TCH      29669 non-null  float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000
mean	3.361895	1.005413	3.580229	8.113086	0.195222	67.652292	163.004858	3.736694	29.801946
std	3.176669	0.863135	3.744496	7.909701	0.192585	34.003120	147.491777	3.702559	24.099665
min	0.100000	0.000000	0.140000	0.210000	0.000000	1.180000	1.280000	0.190000	0.290000
25%	1.280000	0.470000	1.390000	3.040000	0.080000	44.299999	68.089996	1.480000	9.310000
50%	2.510000	0.760000	2.600000	5.830000	0.140000	64.449997	123.699997	2.730000	23.830000
75%	4.420000	1.270000	4.580000	10.640000	0.250000	86.540001	213.199997	4.830000	43.980000
max	54.560001	11.890000	77.260002	150.600006	2.880000	292.700012	1940.000000	89.510002	173.600006

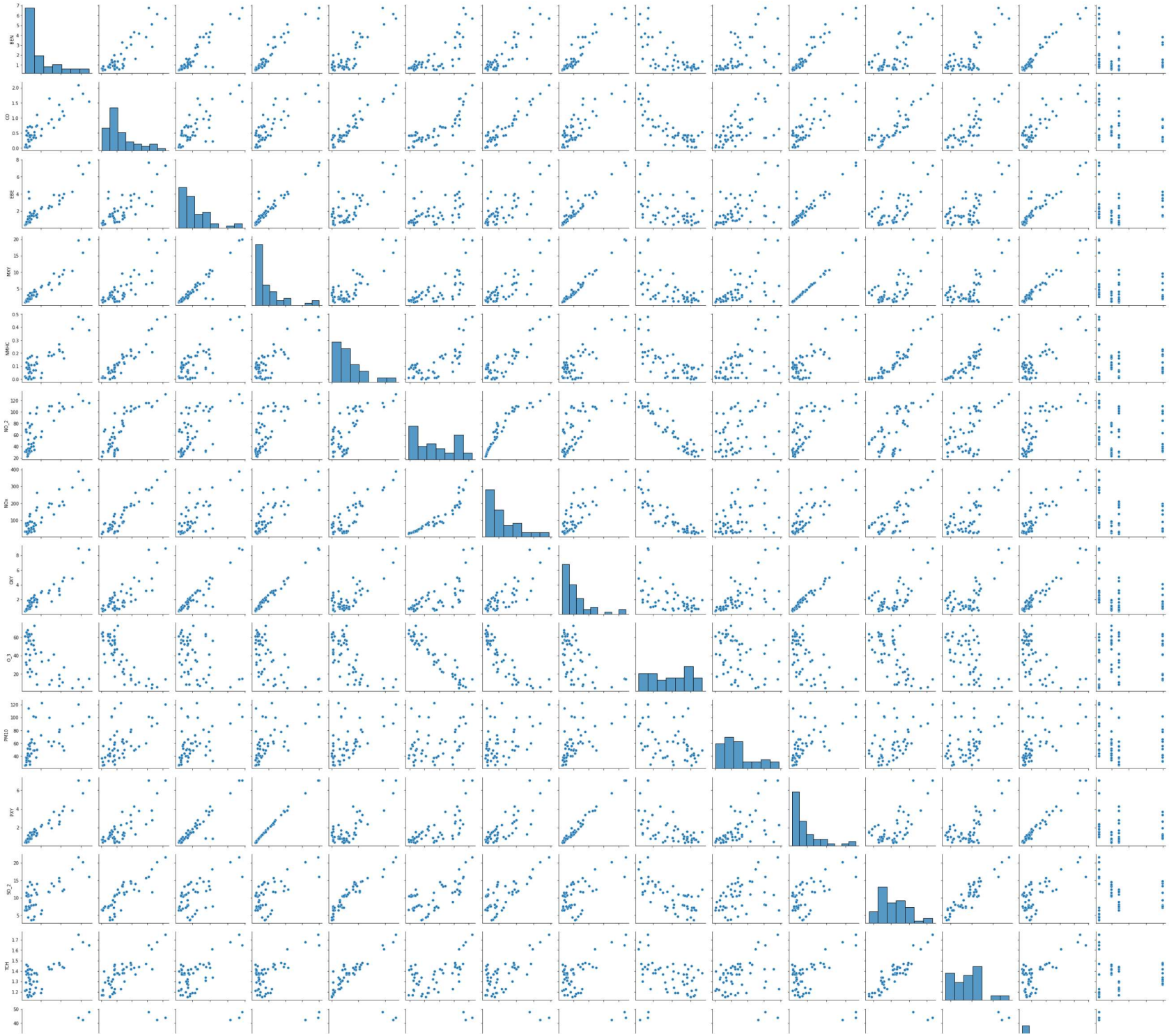
In [18]:

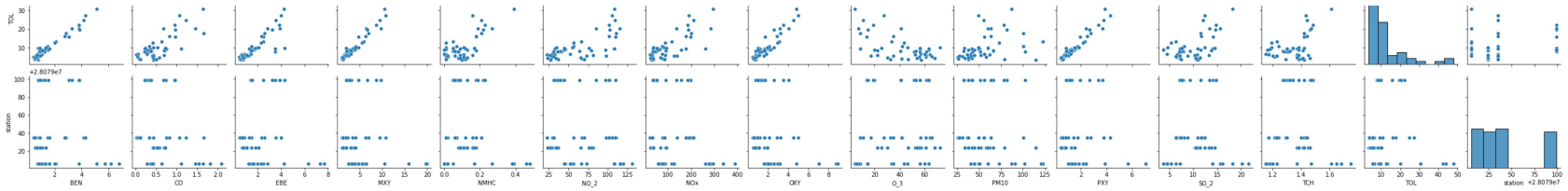
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1c7cf7600a0>
```

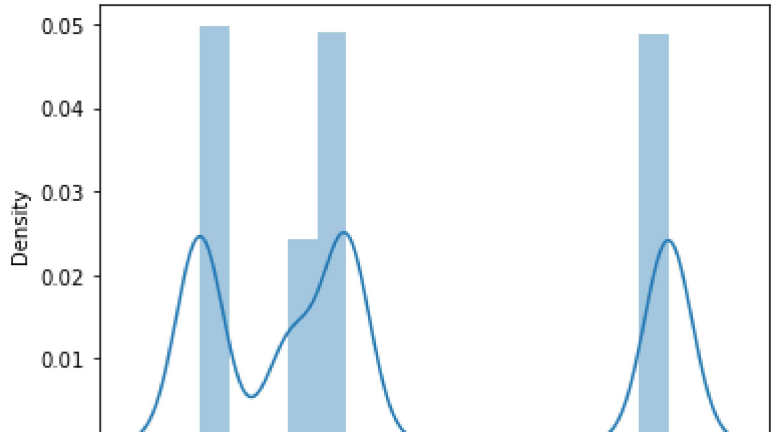

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



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

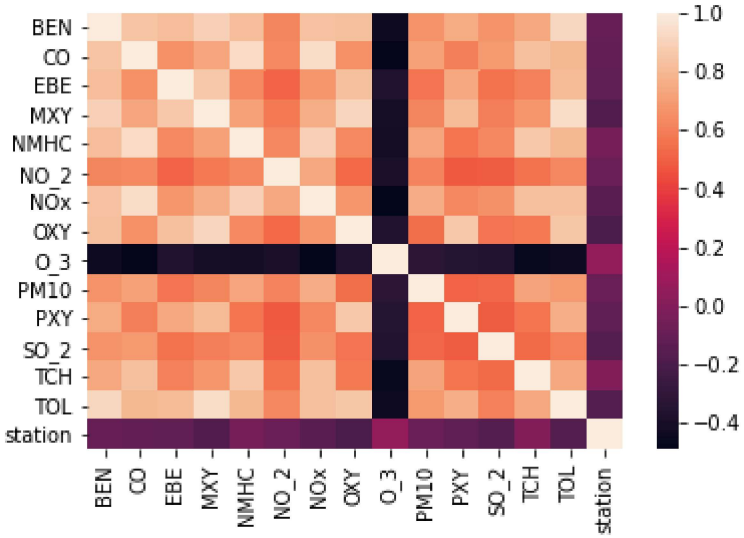
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28079006.920903727
```

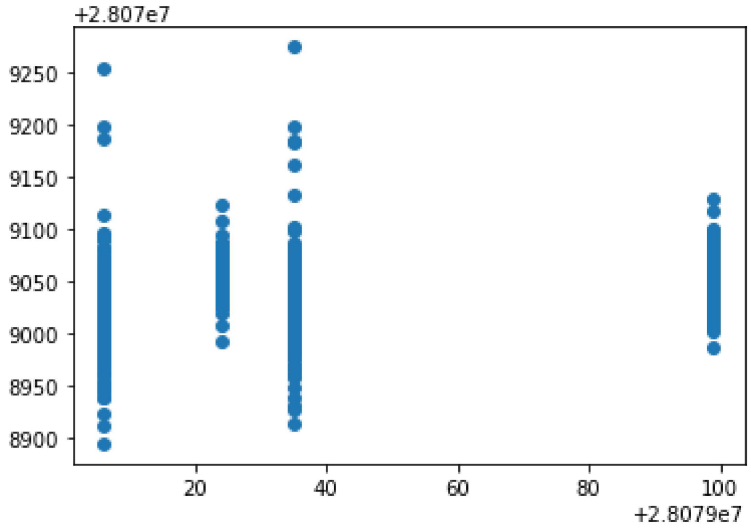
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	7.167385
CO	-17.219286
EBE	0.695649
MXY	-0.312471
NMHC	84.856456
NO_2	0.121145
NOx	-0.081024
OXY	-3.089740
O_3	-0.029332
PM10	-0.053788
PXY	1.724651
SO_2	-0.280828
TCH	37.211048
TOL	-1.210974

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

Out[27]: <matplotlib.collections.PathCollection at 0x1c7deef35b0>



ACCURACY

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

Out[28]: 0.1625629724433405

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

Out[29]: 0.1653930183128306

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]: 0.16291711658435537

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

Out[33]: 0.16511999698249946

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

Out[34]: Lasso(alpha=10)

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

Out[35]: 0.03882024770011494

Accuracy(Lasso)

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

Out[36]: 0.0405390347370842

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([ 4.86193102,  0.        ,  0.68085738, -0.45047435,  0.07055131,
        0.07159909, -0.03470946, -2.43261621, -0.03199313,  0.07877159,
        1.06512407, -0.31461248,  1.20230068, -0.64051274])
```

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

Out[39]:
```
28079048.57322989
```

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

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

Out[41]:
```
0.1044284862453626
```

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)))
```

```
30.382443453690524
1213.0170622202952
34.828394482380254
```

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]: (29669, 14)

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

Out[46]: (29669,)

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)
```

```
[28079035]
```

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

Out[52]: array([28079006, 28079024, 28079035, 28079099], dtype=int64)

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

Out[53]: 0.8087229094340894

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

Out[54]: 1.724527777144498e-43

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

Out[55]: array([[1.72452778e-43, 2.43756289e-56, 9.99998565e-01, 1.43537418e-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]:
```
0.7245762711864407
```

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

In [62]:

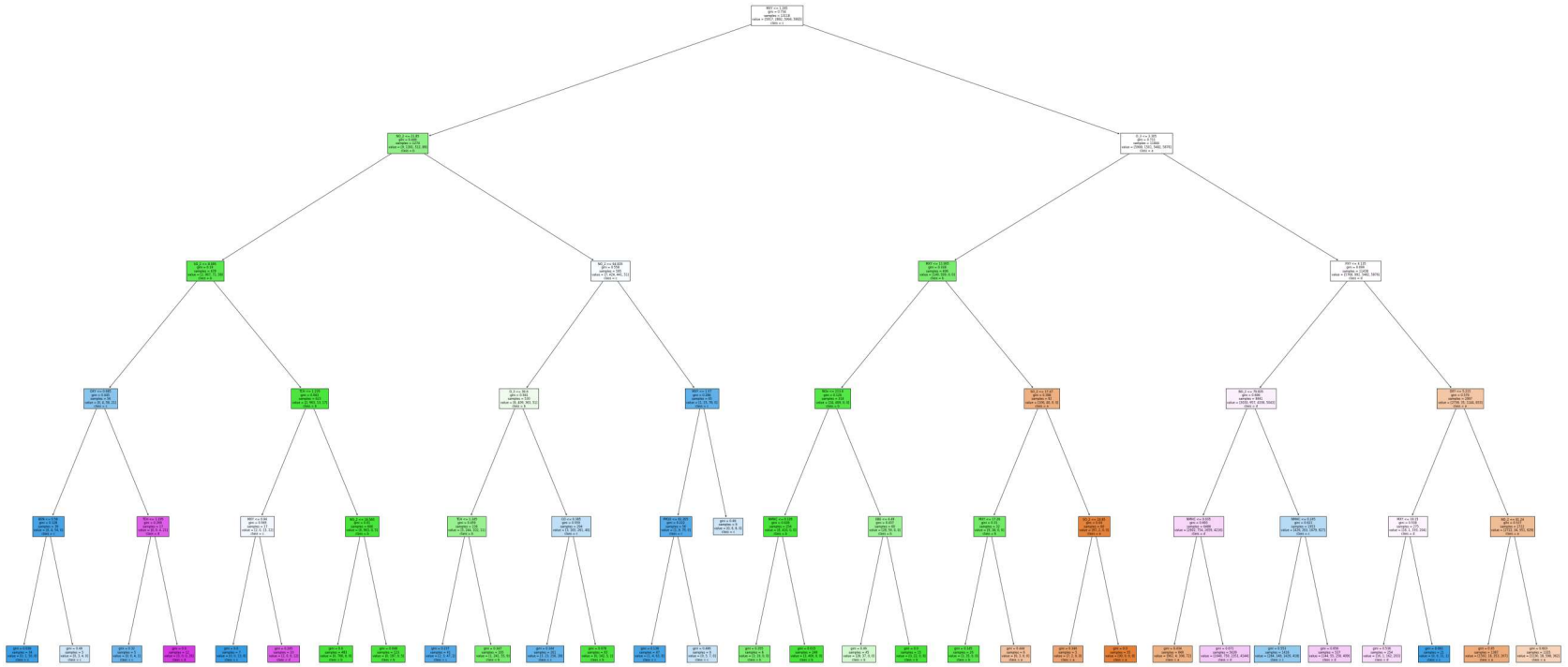
```
from sklearn.tree import plot_tree

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(2222.7000000000003, 1993.2, 'MXY <= 1.265\ngini = 0.734\nsamples = 13118\nvalue = [5917, 2892, 5994, 5
965]\nclass = c'),
 Text(1171.8000000000002, 1630.8000000000002, 'NO_2 <= 21.85\ngini = 0.449\nsamples = 1274\nvalue = [9, 139
1, 512, 89]\nclass = b'),
 Text(595.2, 1268.4, 'SO_2 <= 8.095\ngini = 0.19\nsamples = 679\nvalue = [2, 967, 71, 38]\nclass = b'),
 Text(297.6, 906.0, 'OXY <= 0.685\ngini = 0.445\nsamples = 56\nvalue = [0, 4, 58, 21]\nclass = c'),
 Text(148.8, 543.5999999999999, 'BEN <= 0.58\ngini = 0.128\nsamples = 39\nvalue = [0, 4, 54, 0]\nclass =
c'),
 Text(74.4, 181.19999999999982, 'gini = 0.038\nsamples = 34\nvalue = [0, 1, 50, 0]\nclass = c'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.49\nsamples = 5\nvalue = [0, 3, 4, 0]\nclass = c'),
 Text(446.40000000000003, 543.5999999999999, 'TCH <= 1.195\ngini = 0.269\nsamples = 17\nvalue = [0, 0, 4, 2
1]\nclass = d'),
 Text(372.0, 181.19999999999982, 'gini = 0.32\nsamples = 5\nvalue = [0, 0, 4, 1]\nclass = c'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.0\nsamples = 12\nvalue = [0, 0, 0, 20]\nclass = d'),
 Text(892.8000000000001, 906.0, 'TCH <= 1.235\ngini = 0.063\nsamples = 623\nvalue = [2, 963, 13, 17]\nclass
= b'),
 Text(744.0, 543.5999999999999, 'MXY <= 0.94\ngini = 0.565\nsamples = 17\nvalue = [2, 0, 13, 12]\nclass =
c'),
 Text(669.6, 181.19999999999982, 'gini = 0.0\nsamples = 7\nvalue = [0, 0, 13, 0]\nclass = c'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.245\nsamples = 10\nvalue = [2, 0, 0, 12]\nclass =
d'),
 Text(1041.6000000000001, 543.5999999999999, 'NO_2 <= 16.565\ngini = 0.01\nsamples = 606\nvalue = [0, 963,
0, 5]\nclass = b'),
 Text(967.2, 181.19999999999982, 'gini = 0.0\nsamples = 483\nvalue = [0, 766, 0, 0]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.048\nsamples = 123\nvalue = [0, 197, 0, 5]\nclass = b'),
 Text(1748.4, 1268.4, 'NO_2 <= 64.835\ngini = 0.558\nsamples = 595\nvalue = [7, 424, 441, 51]\nclass = c'),
 Text(1488.0, 906.0, 'O_3 <= 36.6\ngini = 0.561\nsamples = 530\nvalue = [6, 409, 363, 51]\nclass = b'),
 Text(1339.2, 543.5999999999999, 'TCH <= 1.245\ngini = 0.459\nsamples = 236\nvalue = [3, 244, 102, 11]\nclas
s = b'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.237\nsamples = 41\nvalue = [2, 3, 47, 2]\nclass =
c'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.347\nsamples = 195\nvalue = [1, 241, 55, 9]\nclass =
b'),
 Text(1636.8000000000002, 543.5999999999999, 'CO <= 0.385\ngini = 0.559\nsamples = 294\nvalue = [3, 165, 26
1, 40]\nclass = c'),
 Text(1562.4, 181.19999999999982, 'gini = 0.344\nsamples = 201\nvalue = [3, 23, 256, 39]\nclass = c'),
 Text(1711.2, 181.19999999999982, 'gini = 0.078\nsamples = 93\nvalue = [0, 142, 5, 1]\nclass = b'),
 Text(2008.8000000000002, 906.0, 'MXY <= 1.07\ngini = 0.286\nsamples = 65\nvalue = [1, 15, 78, 0]\nclass =
c'),
 Text(1934.4, 543.5999999999999, 'PM10 <= 61.205\ngini = 0.222\nsamples = 56\nvalue = [1, 9, 70, 0]\nclass =
c'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.138\nsamples = 47\nvalue = [1, 4, 63, 0]\nclass =
c'),
```

```
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.486\nsamples = 9\nvalue = [0, 5, 7, 0]\nclass = c'),
 Text(2083.2000000000003, 543.5999999999999, 'gini = 0.49\nsamples = 9\nvalue = [0, 6, 8, 0]\nclass = c'),
 Text(3273.6000000000004, 1630.8000000000002, 'O_3 <= 3.305\ngini = 0.711\nsamples = 11844\nvalue = [5908, 1
501, 5482, 5876]\nclass = a'),
 Text(2678.4, 1268.4, 'MXY <= 11.995\ngini = 0.338\nsamples = 406\nvalue = [140, 509, 0, 0]\nclass = b'),
 Text(2380.8, 906.0, 'NOx <= 213.6\ngini = 0.126\nsamples = 314\nvalue = [34, 469, 0, 0]\nclass = b'),
 Text(2232.0, 543.5999999999999, 'NMHC <= 0.105\ngini = 0.028\nsamples = 254\nvalue = [6, 410, 0, 0]\nclass
= b'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.355\nsamples = 6\nvalue = [3, 10, 0, 0]\nclass =
b'),
 Text(2306.4, 181.19999999999982, 'gini = 0.015\nsamples = 248\nvalue = [3, 400, 0, 0]\nclass = b'),
 Text(2529.6000000000004, 543.5999999999999, 'EBE <= 4.49\ngini = 0.437\nsamples = 60\nvalue = [28, 59, 0,
0]\nclass = b'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.49\nsamples = 45\nvalue = [28, 37, 0, 0]\nclass =
b'),
 Text(2604.0, 181.19999999999982, 'gini = 0.0\nsamples = 15\nvalue = [0, 22, 0, 0]\nclass = b'),
 Text(2976.0, 906.0, 'SO_2 <= 17.47\ngini = 0.398\nsamples = 92\nvalue = [106, 40, 0, 0]\nclass = a'),
 Text(2827.2000000000003, 543.5999999999999, 'MXY <= 17.26\ngini = 0.31\nsamples = 32\nvalue = [9, 38, 0, 0]
\nclass = b'),
 Text(2752.8, 181.19999999999982, 'gini = 0.145\nsamples = 26\nvalue = [3, 35, 0, 0]\nclass = b'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.444\nsamples = 6\nvalue = [6, 3, 0, 0]\nclass = a'),
 Text(3124.8, 543.5999999999999, 'SO_2 <= 19.85\ngini = 0.04\nsamples = 60\nvalue = [97, 2, 0, 0]\nclass =
a'),
 Text(3050.4, 181.19999999999982, 'gini = 0.346\nsamples = 5\nvalue = [7, 2, 0, 0]\nclass = a'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.0\nsamples = 55\nvalue = [90, 0, 0, 0]\nclass = a'),
 Text(3868.8, 1268.4, 'PXY <= 4.135\ngini = 0.699\nsamples = 11438\nvalue = [5768, 992, 5482, 5876]\nclass =
d'),
 Text(3571.2000000000003, 906.0, 'NO_2 <= 79.835\ngini = 0.696\nsamples = 8441\nvalue = [3030, 957, 4338, 50
43]\nclass = d'),
 Text(3422.4, 543.5999999999999, 'NMHC <= 0.035\ngini = 0.693\nsamples = 6488\nvalue = [2602, 754, 2659, 421
6]\nclass = d'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.434\nsamples = 849\nvalue = [962, 4, 308, 72]\nclass
= a'),
 Text(3496.8, 181.19999999999982, 'gini = 0.671\nsamples = 5639\nvalue = [1640, 750, 2351, 4144]\nclass =
d'),
 Text(3720.0000000000005, 543.5999999999999, 'NMHC <= 0.245\ngini = 0.621\nsamples = 1953\nvalue = [428, 20
3, 1679, 827]\nclass = c'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.553\nsamples = 1416\nvalue = [284, 148, 1429, 418]\n
class = c'),
 Text(3794.4, 181.19999999999982, 'gini = 0.656\nsamples = 537\nvalue = [144, 55, 250, 409]\nclass = d'),
 Text(4166.400000000001, 906.0, 'OXY <= 5.215\ngini = 0.579\nsamples = 2997\nvalue = [2738, 35, 1144, 833]\n
class = a'),
 Text(4017.6000000000004, 543.5999999999999, 'MXY <= 18.15\ngini = 0.538\nsamples = 275\nvalue = [16, 1, 19
```

```
3, 204]\nclass = d'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.536\nsamples = 254\nvalue = [16, 1, 162, 203]\nclass
= d'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.061\nsamples = 21\nvalue = [0, 0, 31, 1]\nclass =
c'),
 Text(4315.200000000001, 543.5999999999999, 'NO_2 <= 91.24\ngini = 0.537\nsamples = 2722\nvalue = [2722, 34,
951, 629]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.45\nsamples = 1387\nvalue = [1592, 16, 353, 267]\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.603\nsamples = 1335\nvalue = [1130, 18, 598, 362]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression0.1653930183128306

Ridge Regression:0.16511999698249946

Lasso Regression:0.03882024770011494

ElasticNet Regression:0.1044284862453626

Logistic Regression:0.8087229094340894

Random Forest:0.7245762711864407

Logistic Regression is suitable for this dataset