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_2002")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH	TOL	s
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002	NaN	21.320000	NaN	NaN	280
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.660000	1.82	10.980000	280
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001	NaN	13.670000	NaN	NaN	280
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000	NaN	16.990000	NaN	NaN	280
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002	NaN	15.260000	NaN	NaN	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000	NaN	13.210000	NaN	NaN	280
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999	NaN	15.640000	1.78	15.690000	280
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.620000	1.43	4.330000	280
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN	5.52	24.219999	NaN	22.129999	280
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.910000	1.54	15.510000	280

217296 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: 32381 entries, 1 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     32381 non-null  object 
 1   BEN      32381 non-null  float64
 2   CO       32381 non-null  float64
 3   EBE      32381 non-null  float64
 4   MXY      32381 non-null  float64
 5   NMHC     32381 non-null  float64
 6   NO_2     32381 non-null  float64
 7   NOx      32381 non-null  float64
 8   OXY      32381 non-null  float64
 9   O_3      32381 non-null  float64
 10  PM10     32381 non-null  float64
 11  PXY      32381 non-null  float64
 12  SO_2     32381 non-null  float64
 13  TCH      32381 non-null  float64
 14  TOL      32381 non-null  float64
 15  station  32381 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.2+ MB
```

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

Out[6]:

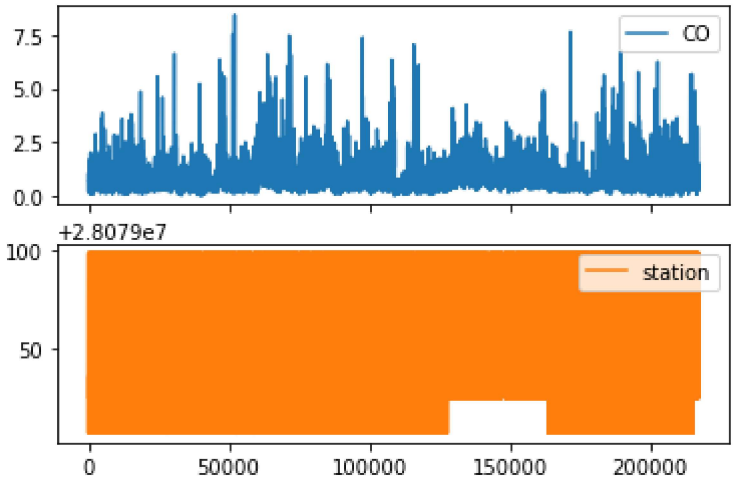
	CO	station
1	0.71	28079035
5	0.72	28079006
22	0.80	28079024
24	1.04	28079099
26	0.53	28079035
...	...	...
217269	0.28	28079024
217271	1.30	28079099
217273	0.97	28079035
217293	0.58	28079024
217295	1.17	28079099

32381 rows × 2 columns

Line chart

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

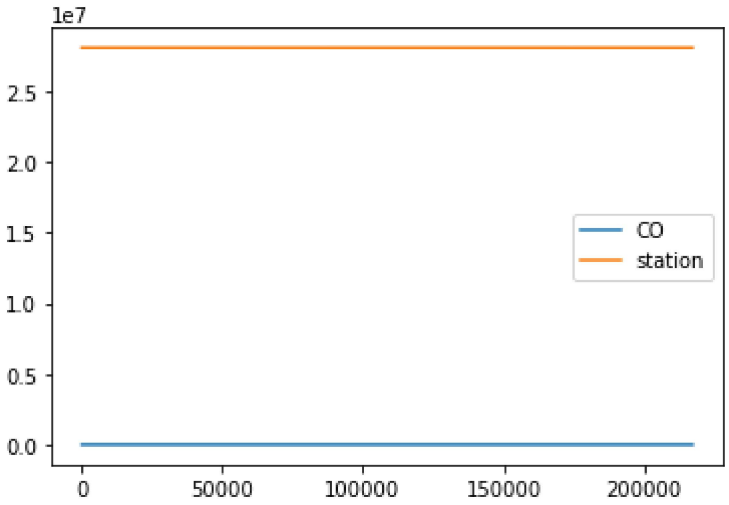
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

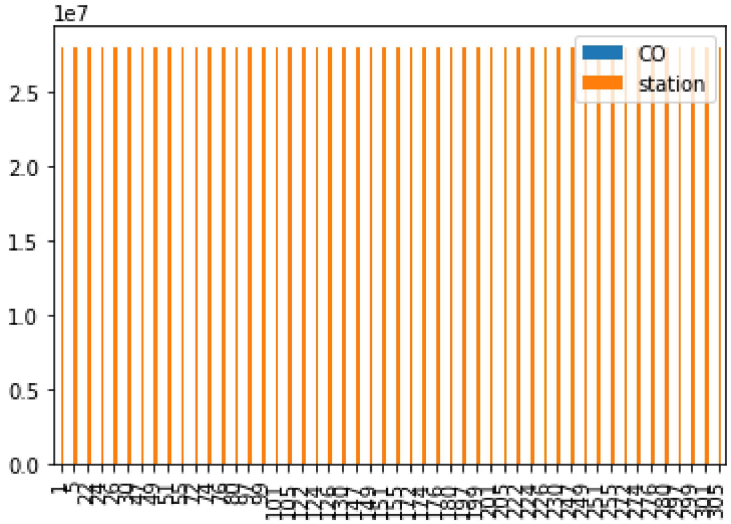


Bar chart

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

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

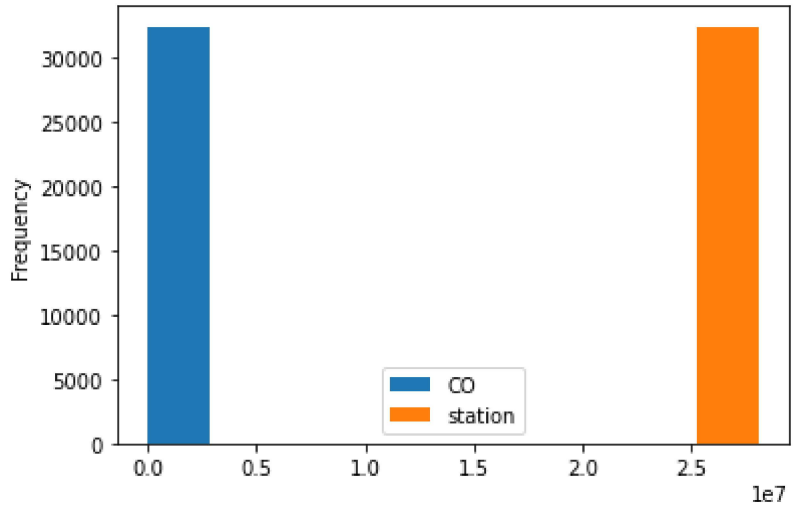
Out[10]: <AxesSubplot:>



Histogram

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

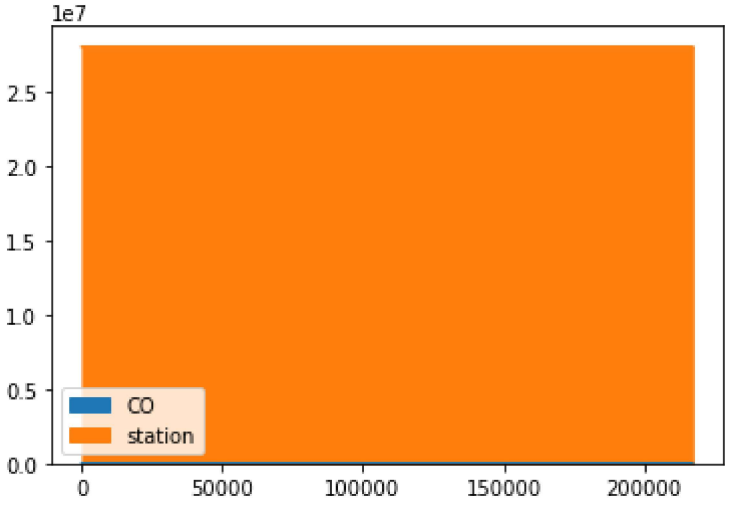
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

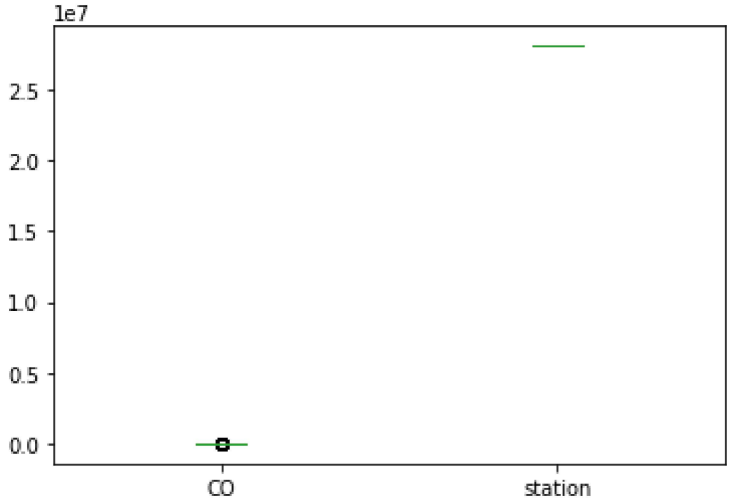
Out[12]: <AxesSubplot:>



Box chart

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

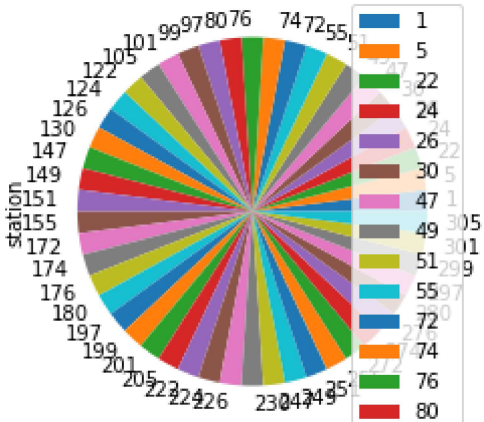
Out[13]: <AxesSubplot:>



Pie chart

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

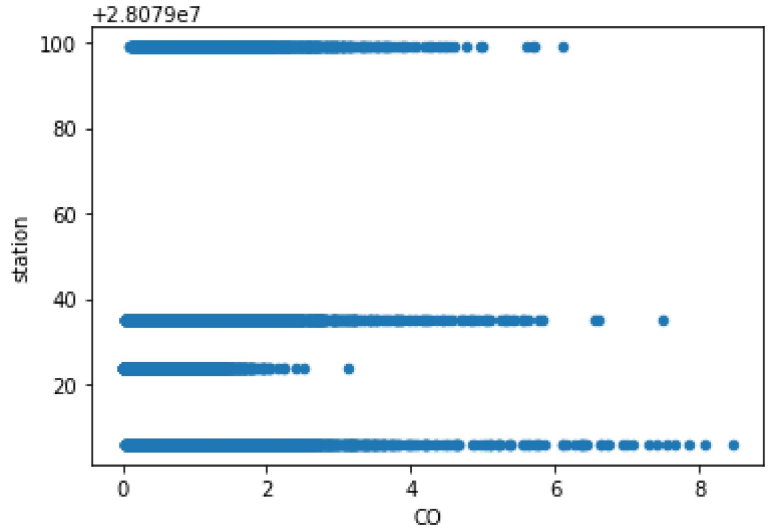
Out[14]:
```
<AxesSubplot:ylabel='station'>
```


station
76
74
72
55
101
99
97
80
105
122
124
126
130
147
149
151
155
172
174
176
180
197
199
201
205
222
224
226
230
247
249
1
5
22
24
26
30
47
49
51
55
72
74
76
80
97
99
101
105
122
124
126
130
147
149
151
155
172
174
176
180
197
199
201
205
222
224
226
230
247
249
251
255
272
274
276
280
297
299

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: 32381 entries, 1 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     32381 non-null  object
 1   BEN      32381 non-null  float64
 2   CO       32381 non-null  float64
 3   EBE      32381 non-null  float64
 4   MXY      32381 non-null  float64
 5   NMHC     32381 non-null  float64
 6   NO_2     32381 non-null  float64
 7   NOx      32381 non-null  float64
 8   OXY      32381 non-null  float64
 9   O_3      32381 non-null  float64
 10  PM10     32381 non-null  float64
 11  PXY      32381 non-null  float64
 12  SO_2     32381 non-null  float64
 13  TCH      32381 non-null  float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000
mean	2.479155	0.787323	2.914004	7.013636	0.155827	58.936796	126.009340	3.169093	33.128518
std	2.280959	0.610810	2.667881	6.774365	0.135731	31.472733	114.035078	2.950771	25.679650
min	0.180000	0.000000	0.180000	0.190000	0.000000	0.890000	1.710000	0.180000	1.880000
25%	0.970000	0.420000	1.140000	2.420000	0.080000	35.660000	48.720001	1.190000	10.290000
50%	1.840000	0.620000	2.130000	5.140000	0.130000	57.160000	96.830002	2.340000	27.290001
75%	3.250000	0.980000	3.830000	9.420000	0.200000	78.769997	167.500000	4.130000	50.160000
max	32.660000	8.460000	41.740002	99.879997	2.700000	263.600006	1336.000000	76.339996	151.399994

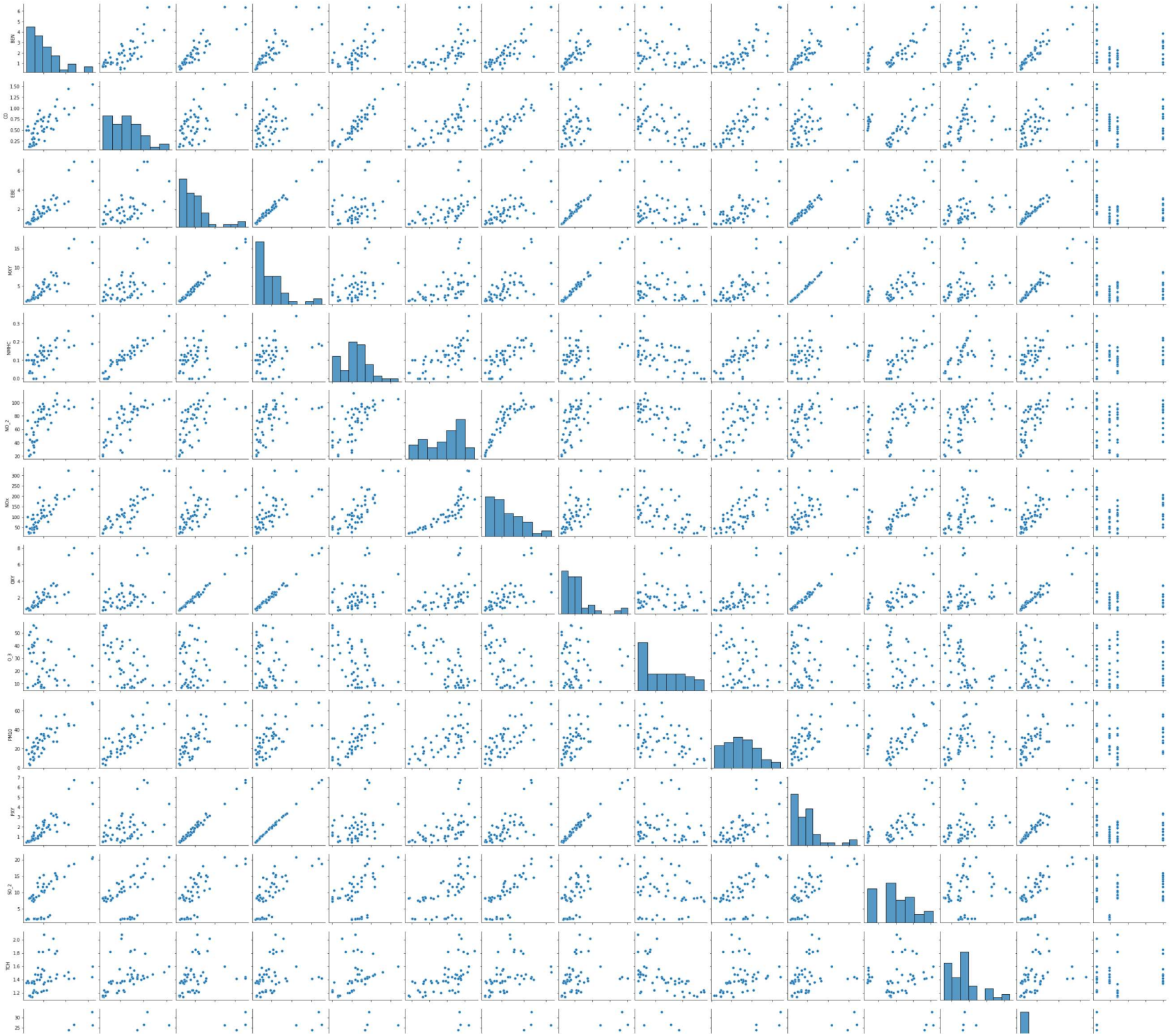
In [18]:

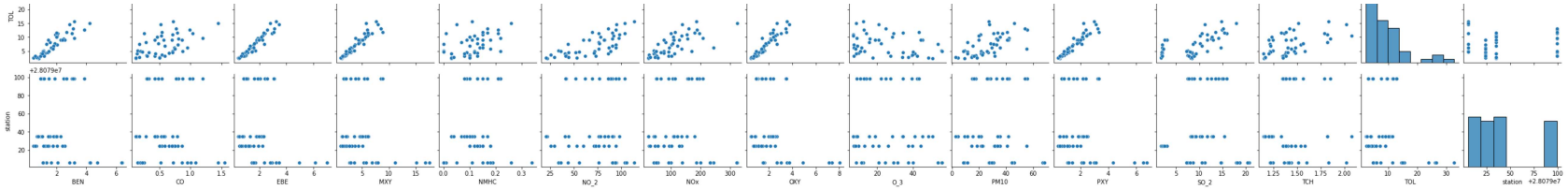
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x195dde002e0>
```

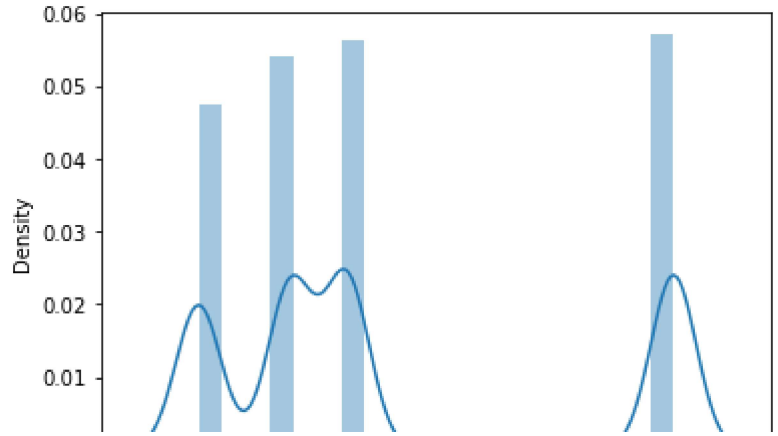

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



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

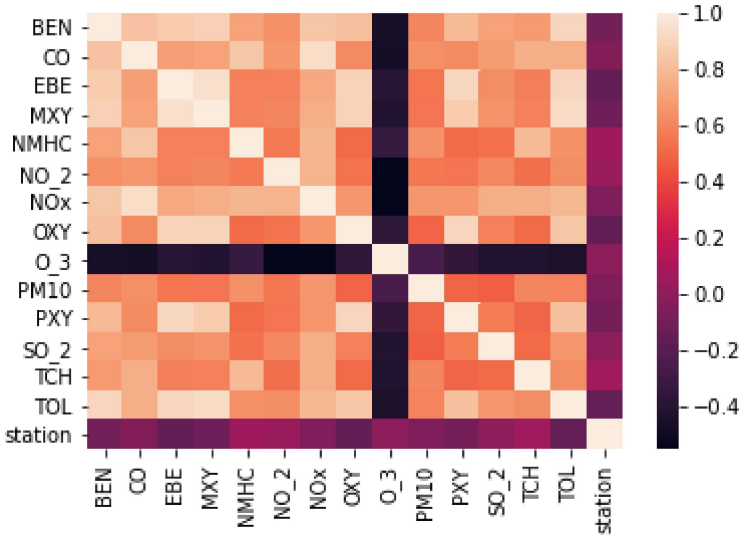
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28078988.04789076
```

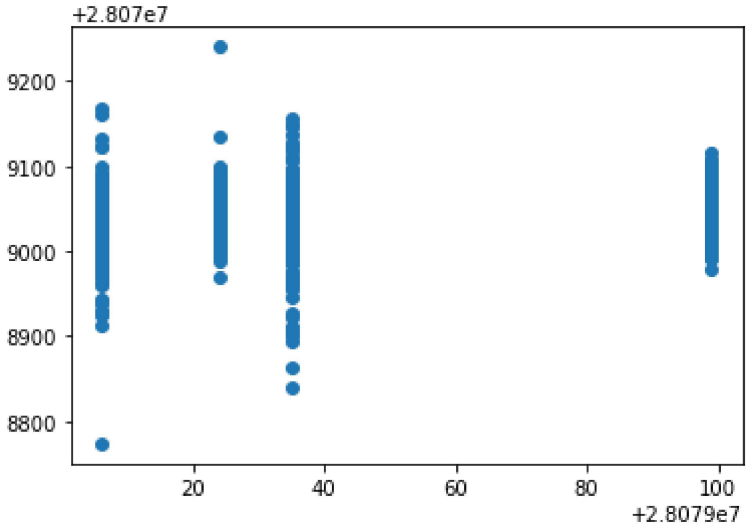
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	2.127880
CO	-13.377859
EBE	-11.898731
MXY	4.213690
NMHC	85.350840
NO_2	0.249613
NOx	-0.102582
OXY	-5.063797
O_3	-0.023072
PM10	-0.126225
PXY	7.819506
SO_2	0.631406
TCH	43.085702
TOL	-1.502361

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

Out[27]: <matplotlib.collections.PathCollection at 0x195ed610580>



ACCURACY

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

Out[28]: 0.19084160353876733

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

Out[29]: 0.20226473515319232

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]: 0.19070124926960652

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

Out[33]: 0.20204408096897053

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]: 0.05783026530167035

Accuracy(Lasso)

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

Out[36]: 0.05896723554942673

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([ 1.05363840e+00,  0.00000000e+00, -2.94374581e+00,  1.54056083e+00,
        2.08377419e-01,  2.26935966e-01, -2.78379496e-02, -2.34722815e+00,
       -1.84086853e-02,  5.92739824e-04,  2.21824675e+00,  4.20780309e-01,
        1.12599389e+00, -1.17689144e+00])
```

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

Out[39]:
```
28079038.032892514
```

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

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

Out[41]:
```
0.09906557806958294
```

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)))
```

```
28.665579995440105
1128.1578624152505
33.58806130778093
```

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]: (32381, 14)

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

Out[46]: (32381,)

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]: 0.8480899292795158

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

Out[54]:
```
2.5638972732451705e-10
```

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

Out[55]:
```
array([[2.56389727e-10, 3.44199742e-71, 1.00000000e+00, 1.43898646e-13]])
```

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]:
```
0.7758757610517957
```

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

In [62]:

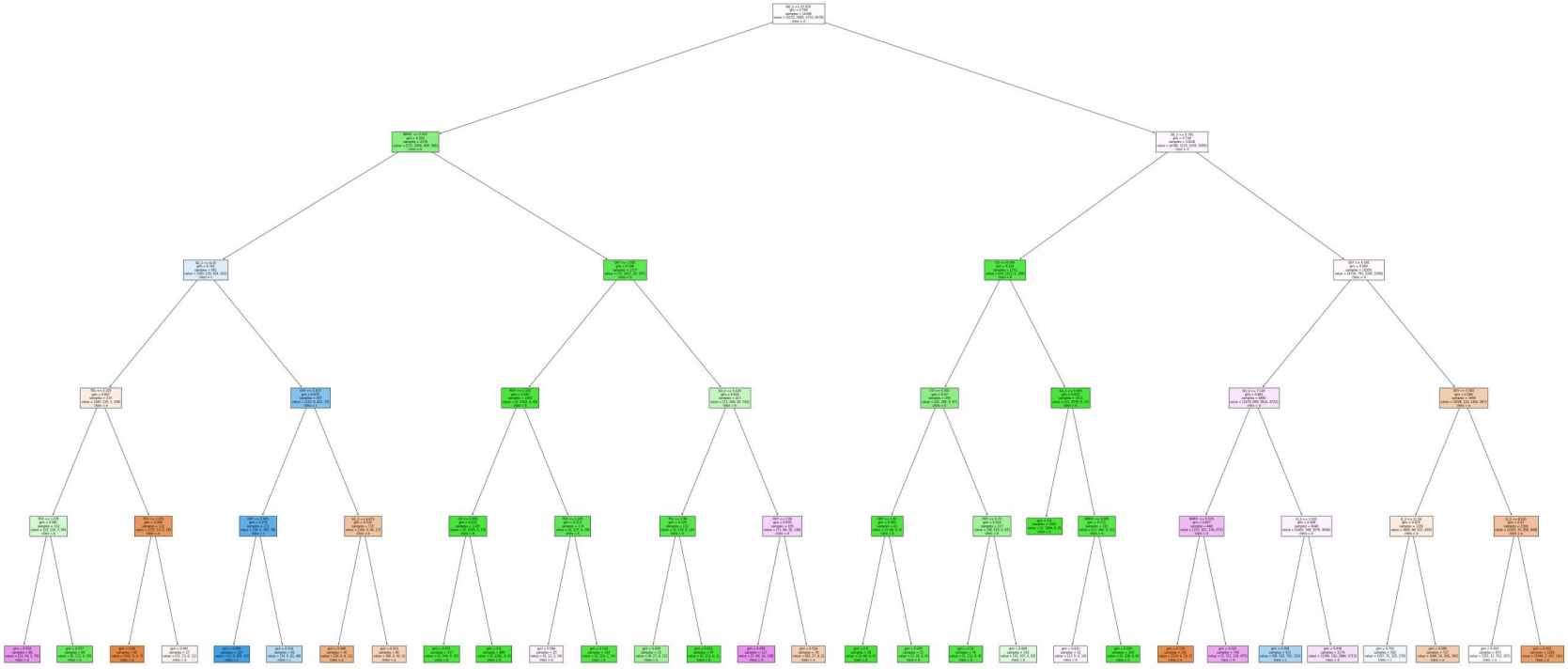
```
from sklearn.tree import plot_tree

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(2278.5, 1993.2, 'NO_2 <= 27.015\ngini = 0.749\nsamples = 14306\nvalue = [5152, 5682, 5753, 6079]\nclass = d'),
 Text(1190.4, 1630.8000000000002, 'NMHC <= 0.055\ngini = 0.504\nsamples = 2378\nvalue = [372, 2566, 458, 380]\nclass = b'),
 Text(595.2, 1268.4, 'SO_2 <= 6.33\ngini = 0.703\nsamples = 661\nvalue = [300, 139, 434, 183]\nclass = c'),
 Text(297.6, 906.0, 'TOL <= 3.105\ngini = 0.657\nsamples = 274\nvalue = [180, 139, 3, 108]\nclass = a'),
 Text(148.8, 543.5999999999999, 'TCH <= 1.235\ngini = 0.581\nsamples = 152\nvalue = [23, 126, 3, 90]\nclass = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.534\nsamples = 68\nvalue = [23, 14, 3, 70]\nclass = d'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.257\nsamples = 84\nvalue = [0, 112, 0, 20]\nclass = b'),
 Text(446.40000000000003, 543.5999999999999, 'TCH <= 1.225\ngini = 0.289\nsamples = 122\nvalue = [157, 13, 0, 18]\nclass = a'),
 Text(372.0, 181.19999999999982, 'gini = 0.09\nsamples = 95\nvalue = [142, 0, 0, 7]\nclass = a'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.661\nsamples = 27\nvalue = [15, 13, 0, 11]\nclass = a'),
 Text(892.8000000000001, 906.0, 'OXY <= 1.415\ngini = 0.475\nsamples = 387\nvalue = [120, 0, 431, 75]\nclass = c'),
 Text(744.0, 543.5999999999999, 'OXY <= 0.905\ngini = 0.279\nsamples = 272\nvalue = [16, 0, 385, 58]\nclass = c'),
 Text(669.6, 181.19999999999982, 'gini = 0.096\nsamples = 187\nvalue = [6, 0, 303, 10]\nclass = c'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.534\nsamples = 85\nvalue = [10, 0, 82, 48]\nclass = c'),
 Text(1041.6000000000001, 543.5999999999999, 'SO_2 <= 6.875\ngini = 0.526\nsamples = 115\nvalue = [104, 0, 46, 17]\nclass = a'),
 Text(967.2, 181.19999999999982, 'gini = 0.449\nsamples = 30\nvalue = [36, 0, 4, 11]\nclass = a'),
 Text(1116.0, 181.19999999999982, 'gini = 0.523\nsamples = 85\nvalue = [68, 0, 42, 6]\nclass = a'),
 Text(1785.6000000000001, 1268.4, 'OXY <= 1.005\ngini = 0.198\nsamples = 1717\nvalue = [72, 2427, 24, 197]\nclass = b'),
 Text(1488.0, 906.0, 'MXY <= 1.245\ngini = 0.042\nsamples = 1400\nvalue = [0, 2163, 4, 43]\nclass = b'),
 Text(1339.2, 543.5999999999999, 'CO <= 0.365\ngini = 0.015\nsamples = 1226\nvalue = [0, 1926, 0, 15]\nclass = b'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.052\nsamples = 337\nvalue = [0, 544, 0, 15]\nclass = b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.0\nsamples = 889\nvalue = [0, 1382, 0, 0]\nclass = b'),
 Text(1636.8000000000002, 543.5999999999999, 'TCH <= 1.245\ngini = 0.213\nsamples = 174\nvalue = [0, 237, 4, 28]\nclass = b'),
 Text(1562.4, 181.19999999999982, 'gini = 0.584\nsamples = 20\nvalue = [0, 13, 3, 14]\nclass = d'),
 Text(1711.2, 181.19999999999982, 'gini = 0.118\nsamples = 154\nvalue = [0, 224, 1, 14]\nclass = b'),
 Text(2083.2000000000003, 906.0, 'SO_2 <= 5.125\ngini = 0.619\nsamples = 317\nvalue = [72, 264, 20, 154]\nclass = b'),
 Text(1934.4, 543.5999999999999, 'TOL <= 2.98\ngini = 0.135\nsamples = 122\nvalue = [0, 178, 0, 14]\nclass =

```
b'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.439\nsamples = 25\nvalue = [0, 27, 0, 13]\nclass =
b'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.013\nsamples = 97\nvalue = [0, 151, 0, 1]\nclass =
b'),
 Text(2232.0, 543.5999999999999, 'OXY <= 2.06\ngini = 0.678\nsamples = 195\nvalue = [72, 86, 20, 140]\nclass
= d'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.499\nsamples = 117\nvalue = [7, 49, 14, 138]\nclass
= d'),
 Text(2306.4, 181.19999999999982, 'gini = 0.534\nsamples = 78\nvalue = [65, 37, 6, 2]\nclass = a'),
 Text(3366.6000000000004, 1630.8000000000002, 'SO_2 <= 5.795\ngini = 0.739\nsamples = 11928\nvalue = [4780,
3116, 5295, 5699]\nclass = d'),
 Text(2864.4, 1268.4, 'CO <= 0.345\ngini = 0.124\nsamples = 1572\nvalue = [54, 2323, 0, 109]\nclass = b'),
 Text(2678.4, 906.0, 'CO <= 0.165\ngini = 0.47\nsamples = 260\nvalue = [41, 285, 0, 87]\nclass = b'),
 Text(2529.6000000000004, 543.5999999999999, 'OXY <= 1.67\ngini = 0.083\nsamples = 43\nvalue = [3, 66, 0, 0]
\nclass = b'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.0\nsamples = 28\nvalue = [0, 48, 0, 0]\nclass = b'),
 Text(2604.0, 181.19999999999982, 'gini = 0.245\nsamples = 15\nvalue = [3, 18, 0, 0]\nclass = b'),
 Text(2827.2000000000003, 543.5999999999999, 'PXY <= 0.73\ngini = 0.519\nsamples = 217\nvalue = [38, 219, 0,
87]\nclass = b'),
 Text(2752.8, 181.19999999999982, 'gini = 0.14\nsamples = 74\nvalue = [5, 112, 0, 4]\nclass = b'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.609\nsamples = 143\nvalue = [33, 107, 0, 83]\nclass
= b'),
 Text(3050.4, 906.0, 'SO_2 <= 5.005\ngini = 0.033\nsamples = 1312\nvalue = [13, 2038, 0, 22]\nclass = b'),
 Text(2976.0, 543.5999999999999, 'gini = 0.0\nsamples = 1081\nvalue = [0, 1694, 0, 0]\nclass = b'),
 Text(3124.8, 543.5999999999999, 'NMHC <= 0.095\ngini = 0.172\nsamples = 231\nvalue = [13, 344, 0, 22]\nclas
s = b'),
 Text(3050.4, 181.19999999999982, 'gini = 0.632\nsamples = 22\nvalue = [13, 6, 0, 14]\nclass = d'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.045\nsamples = 209\nvalue = [0, 338, 0, 8]\nclass =
b'),
 Text(3868.8, 1268.4, 'OXY <= 4.185\ngini = 0.694\nsamples = 10356\nvalue = [4726, 793, 5295, 5590]\nclass =
d'),
 Text(3571.2000000000003, 906.0, 'SO_2 <= 7.145\ngini = 0.661\nsamples = 6896\nvalue = [1678, 669, 3814, 472
3]\nclass = d'),
 Text(3422.4, 543.5999999999999, 'NMHC <= 0.035\ngini = 0.657\nsamples = 848\nvalue = [215, 321, 138, 677]\n
class = d'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.158\nsamples = 142\nvalue = [214, 0, 18, 2]\nclass =
a'),
 Text(3496.8, 181.19999999999982, 'gini = 0.541\nsamples = 706\nvalue = [1, 321, 120, 675]\nclass = d'),
 Text(3720.0000000000005, 543.5999999999999, 'O_3 <= 7.035\ngini = 0.646\nsamples = 6048\nvalue = [1463, 34
8, 3676, 4046]\nclass = d'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.598\nsamples = 872\nvalue = [99, 165, 792, 333]\ncla
ss = c'),
```

```
 Text(3794.4, 181.19999999999982, 'gini = 0.638\nsamples = 5176\nvalue = [1364, 183, 2884, 3713]\nclass =
d'),
 Text(4166.400000000001, 906.0, 'OXY <= 5.565\ngini = 0.598\nsamples = 3460\nvalue = [3048, 124, 1481, 867]
\nclass = a'),
 Text(4017.6000000000004, 543.5999999999999, 'O_3 <= 11.56\ngini = 0.675\nsamples = 1292\nvalue = [845, 89,
631, 459]\nclass = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.703\nsamples = 565\nvalue = [197, 75, 325, 279]\ncla
ss = c'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.586\nsamples = 727\nvalue = [648, 14, 306, 180]\ncla
ss = a'),
 Text(4315.200000000001, 543.5999999999999, 'O_3 <= 8.505\ngini = 0.53\nsamples = 2168\nvalue = [2203, 35, 8
50, 408]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.653\nsamples = 870\nvalue = [555, 32, 553, 267]\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.353\nsamples = 1298\nvalue = [1648, 3, 297, 141]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression0.20226473515319232

Ridge Regression:0.20204408096897053

Lasso Regression:0.05896723554942673

ElasticNet Regression:0.09906557806958294

Logistic Regression:0.8480899292795158

Random Forest:0.7758757610517957

Logistic Regression is suitable for this dataset