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_2003")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH	TOL	stat
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209999	NaN	24.299999	NaN	NaN	28079(
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389999	NaN	14.230000	1.55	NaN	28079(
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240002	NaN	17.879999	NaN	NaN	28079(
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839996	NaN	24.900000	NaN	NaN	28079(
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779999	NaN	18.750000	NaN	NaN	28079(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000	1.20	4.870000	1.27	1.00	28079(
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400000	0.50	8.360000	NaN	0.88	28079(
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.830000	NaN	5.330000	1.55	NaN	28079(
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.570000	NaN	6.830000	1.27	NaN	28079(
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000	2.43	6.060000	1.32	5.67	28079(

243984 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: 33010 entries, 5 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     33010 non-null  object 
 1   BEN      33010 non-null  float64
 2   CO       33010 non-null  float64
 3   EBE      33010 non-null  float64
 4   MXY      33010 non-null  float64
 5   NMHC     33010 non-null  float64
 6   NO_2     33010 non-null  float64
 7   NOx      33010 non-null  float64
 8   OXY      33010 non-null  float64
 9   O_3      33010 non-null  float64
 10  PM10     33010 non-null  float64
 11  PXY      33010 non-null  float64
 12  SO_2     33010 non-null  float64
 13  TCH      33010 non-null  float64
 14  TOL      33010 non-null  float64
 15  station  33010 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.3+ MB
```

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

Out[6]:

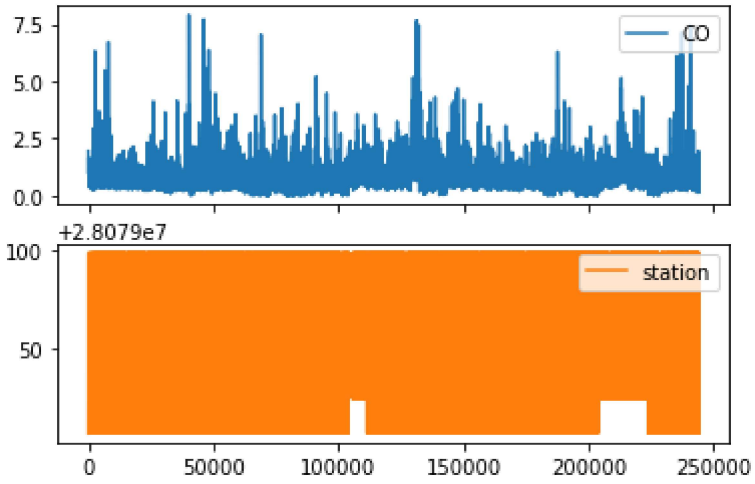
	CO	station
5	1.94	28079006
23	1.27	28079024
27	1.79	28079099
33	1.47	28079006
51	1.29	28079024
...	...	...
243955	0.41	28079099
243957	0.60	28079035
243961	0.82	28079006
243979	0.16	28079024
243983	0.29	28079099

33010 rows × 2 columns

Line chart

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

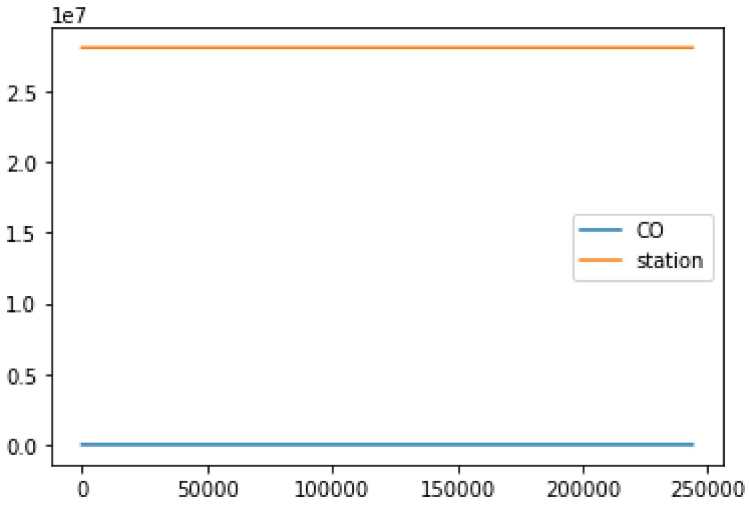
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

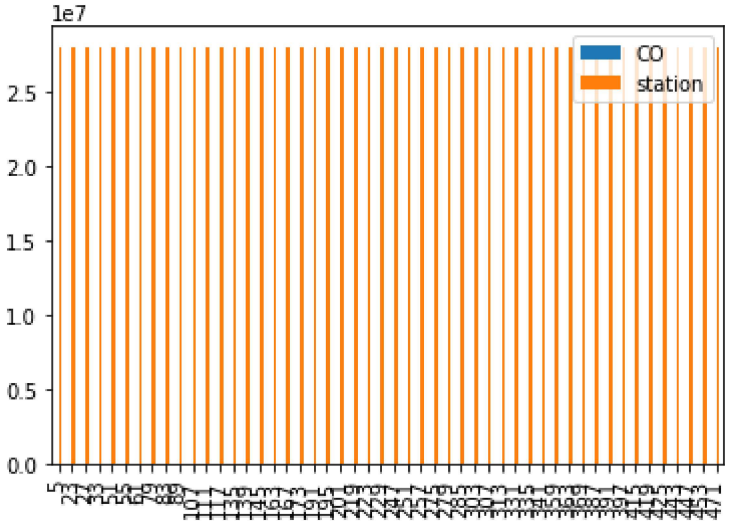


Bar chart

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

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

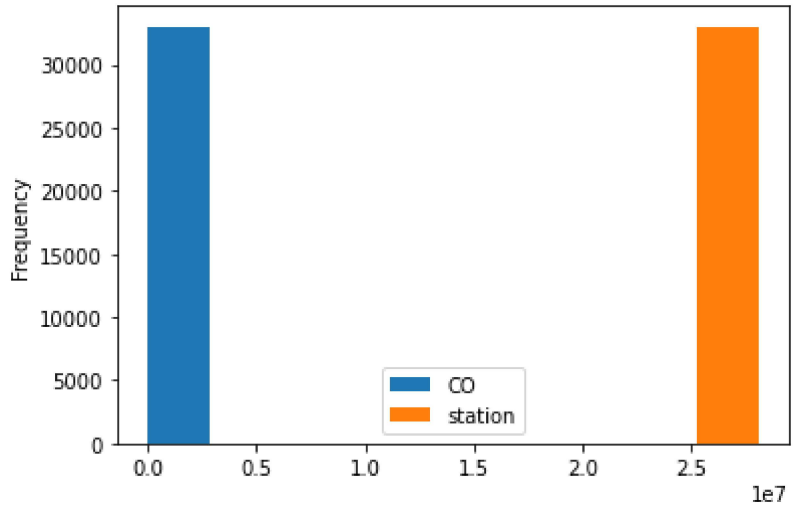
Out[10]: <AxesSubplot:>



Histogram

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

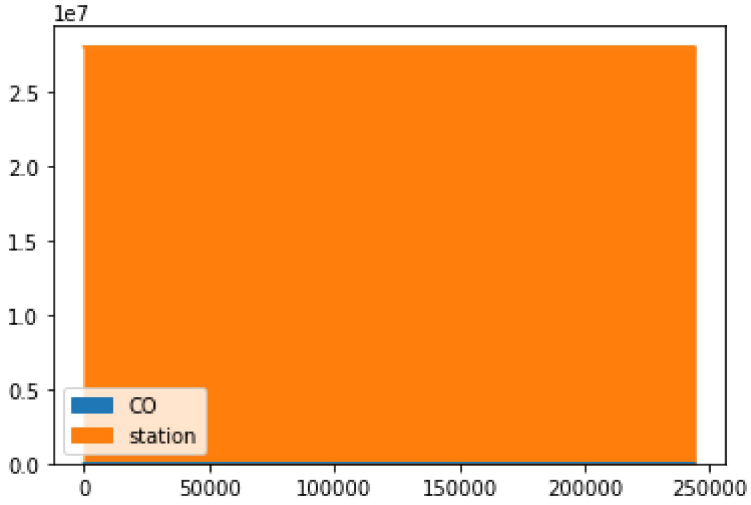
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

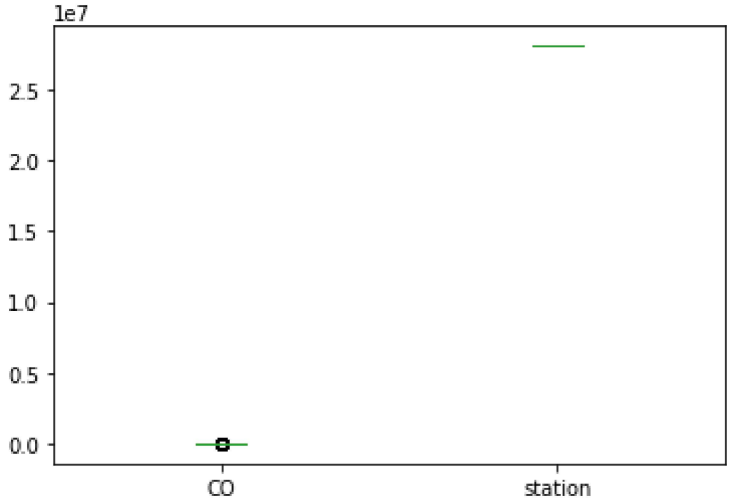
Out[12]: <AxesSubplot:>



Box chart

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

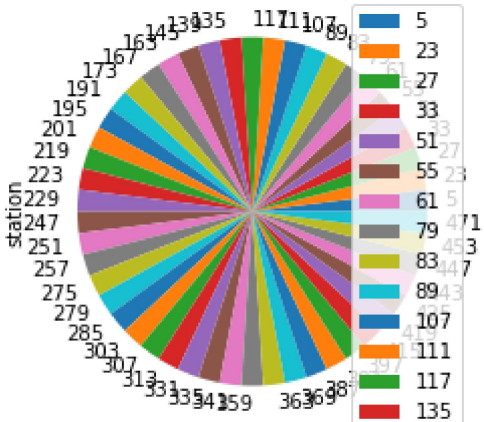
Out[13]: <AxesSubplot:>

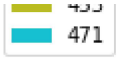


Pie chart

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

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

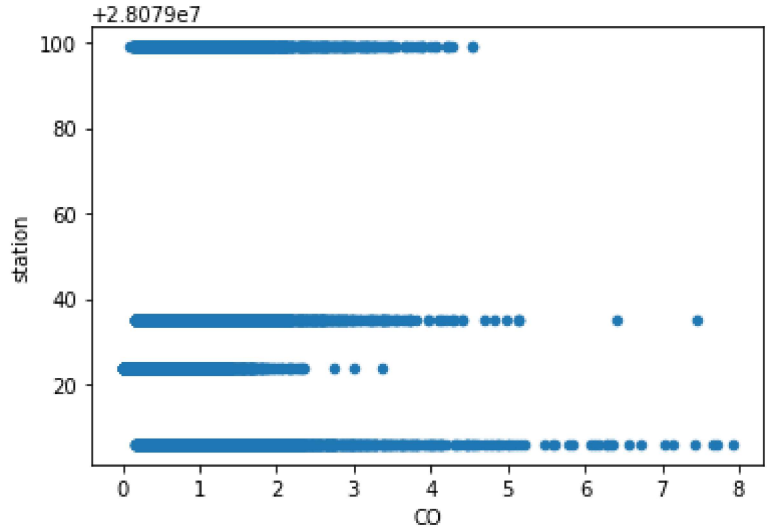

station
5
23
27
33
51
55
61
79
83
89
107
111
117
135
139
145
163
167
173
191
195
201
219
223
229
247
251
257
275
279
285
303
307
313
331
335
341
359
363
369
387
391
397
415
419
425
443
447
453



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: 33010 entries, 5 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     33010 non-null  object
 1   BEN      33010 non-null  float64
 2   CO       33010 non-null  float64
 3   EBE      33010 non-null  float64
 4   MXY      33010 non-null  float64
 5   NMHC     33010 non-null  float64
 6   NO_2     33010 non-null  float64
 7   NOx      33010 non-null  float64
 8   OXY      33010 non-null  float64
 9   O_3      33010 non-null  float64
 10  PM10     33010 non-null  float64
 11  PXY      33010 non-null  float64
 12  SO_2     33010 non-null  float64
 13  TCH      33010 non-null  float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
count	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000
mean	2.192633	0.759868	2.639726	5.838414	0.137177	57.328049	120.153676	2.684084	36.914485
std	2.064160	0.545999	2.825194	6.267296	0.127863	31.811082	104.521700	2.717832	28.988487
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.130000
25%	0.900000	0.430000	1.010000	1.880000	0.060000	34.529999	49.070000	1.000000	12.230000
50%	1.610000	0.620000	1.890000	4.070000	0.110000	55.105000	92.779999	1.790000	30.400000
75%	2.810000	0.930000	3.300000	7.530000	0.170000	76.160004	160.100006	3.340000	54.349998
max	66.389999	7.920000	92.589996	177.600006	2.180000	342.700012	1246.000000	88.180000	178.699997

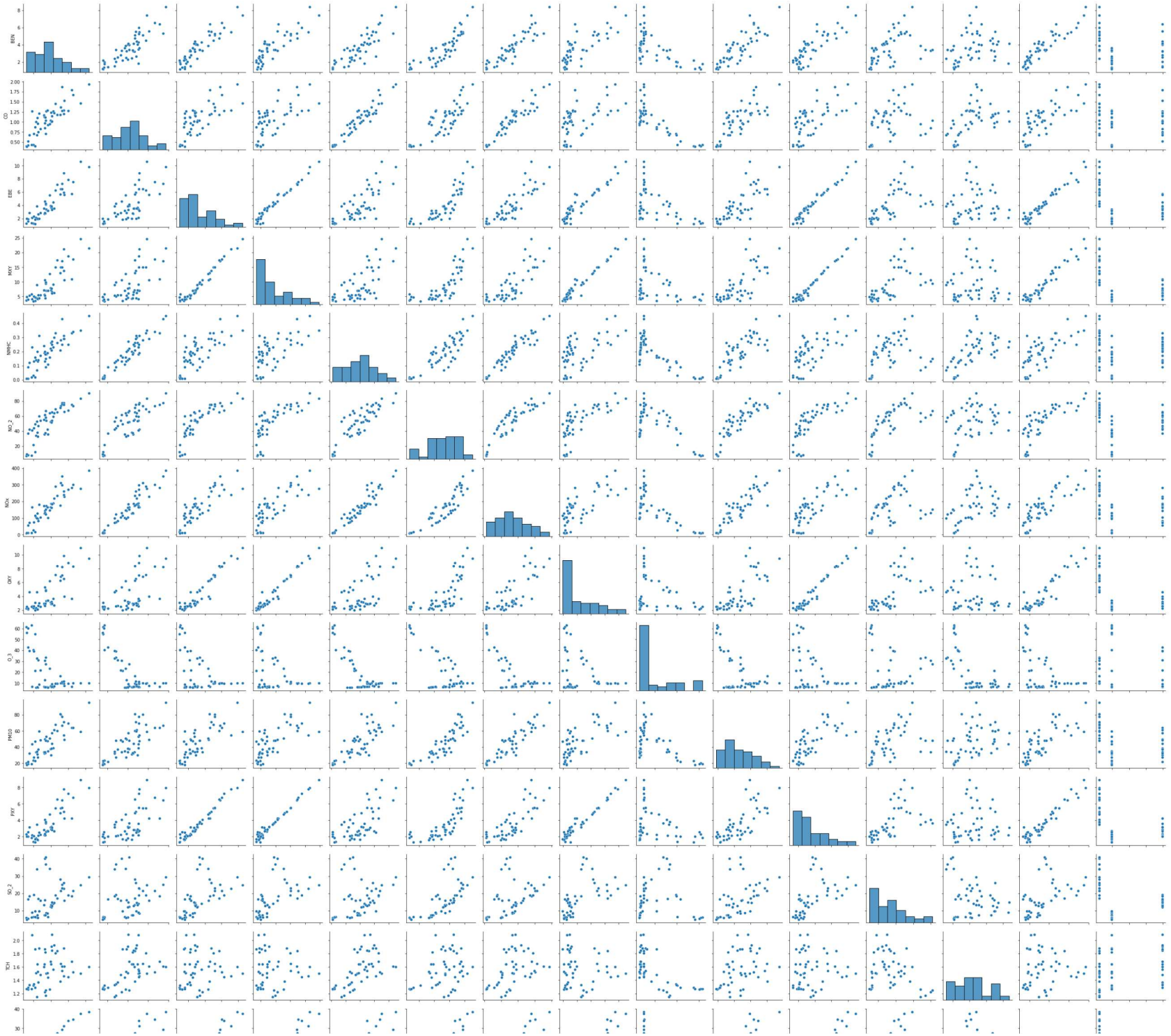
In [18]:

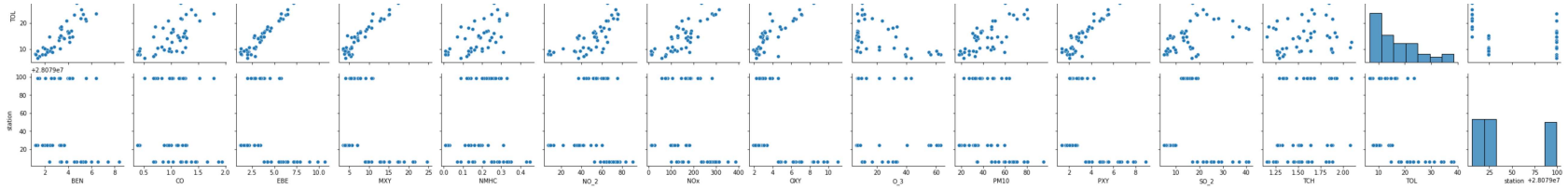
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x29a6651d670>
```

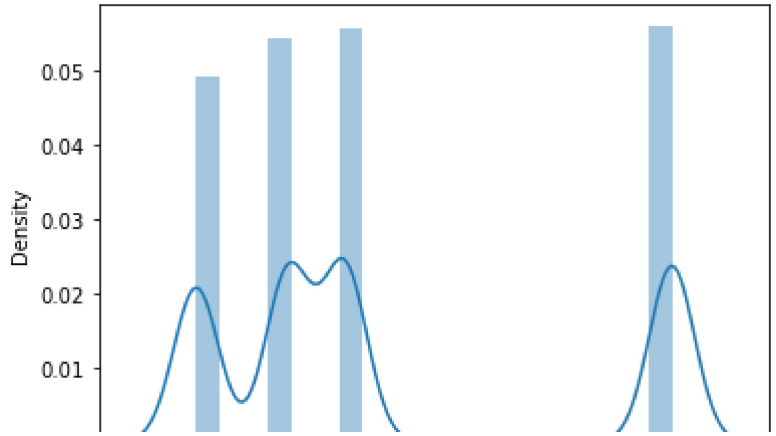

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



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

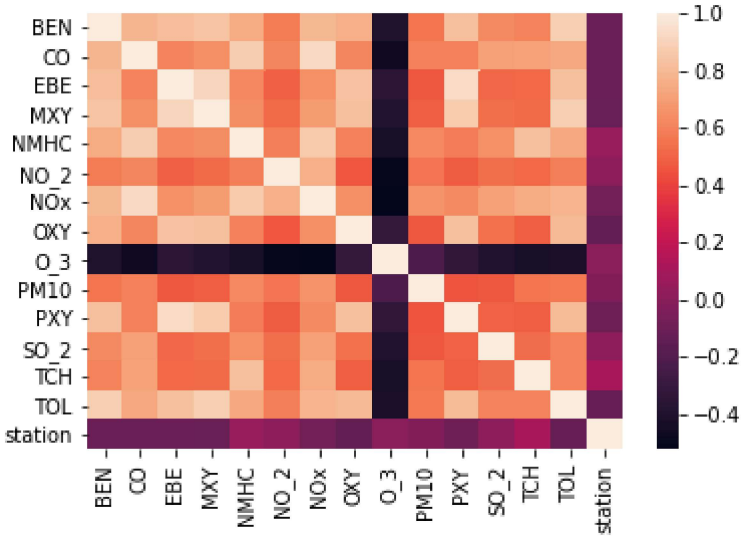
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28079002.289702866
```

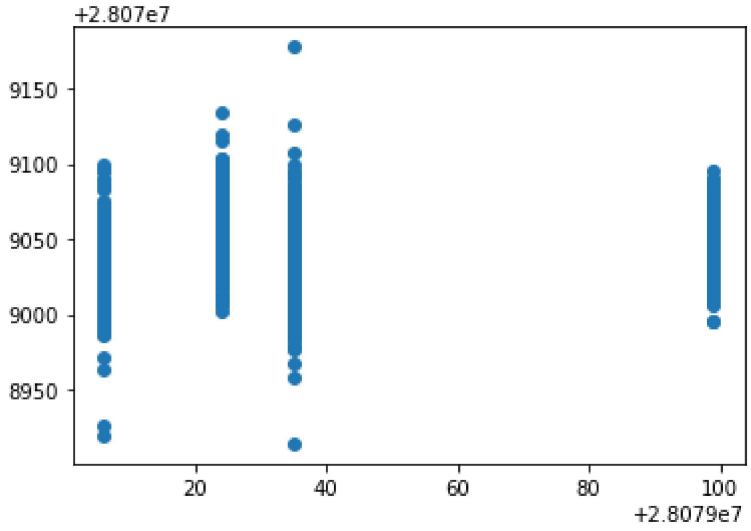
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	2.207835
CO	-37.683984
EBE	-1.881216
MXY	0.139755
NMHC	150.686970
NO_2	0.175650
NOx	-0.075895
OXY	-1.435456
O_3	-0.009738
PM10	-0.069049
PXY	1.811652
SO_2	0.809381
TCH	34.508579
TOL	-0.875362

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

Out[27]: <matplotlib.collections.PathCollection at 0x29a7620a670>



ACCURACY

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

Out[28]: 0.1809341834858923

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

Out[29]: 0.17328056958079485

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]: 0.17879287952357292

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

Out[33]: 0.17227501231797626

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]: 0.03578590238383095

Accuracy(Lasso)

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

Out[36]: 0.033703301186020385

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([ 0.        , -0.19683471,  0.        , -0.04925999,  0.14178781,
        0.16321412, -0.06889313, -1.27697124, -0.03629348,  0.05801668,
        0.27493181,  0.73121501,  1.57332176, -0.40068216])
```

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

Out[39]:
```
28079037.33122153
```

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

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

Out[41]:
```
0.04813959207478369
```

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)))
```

```
29.002865032872688
1173.984806997495
34.263461690224695
```

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]:
```
feature_matrix.shape
```

Out[45]: (33010, 14)

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

Out[46]: (33010,)

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]: 0.7584974250227204

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

Out[54]: 2.3306153265290618e-23

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

Out[55]: array([[2.33061533e-23, 1.44436075e-55, 1.00000000e+00, 6.68457491e-16]])

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]:
```
0.7276148693348015
```

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

In [62]:

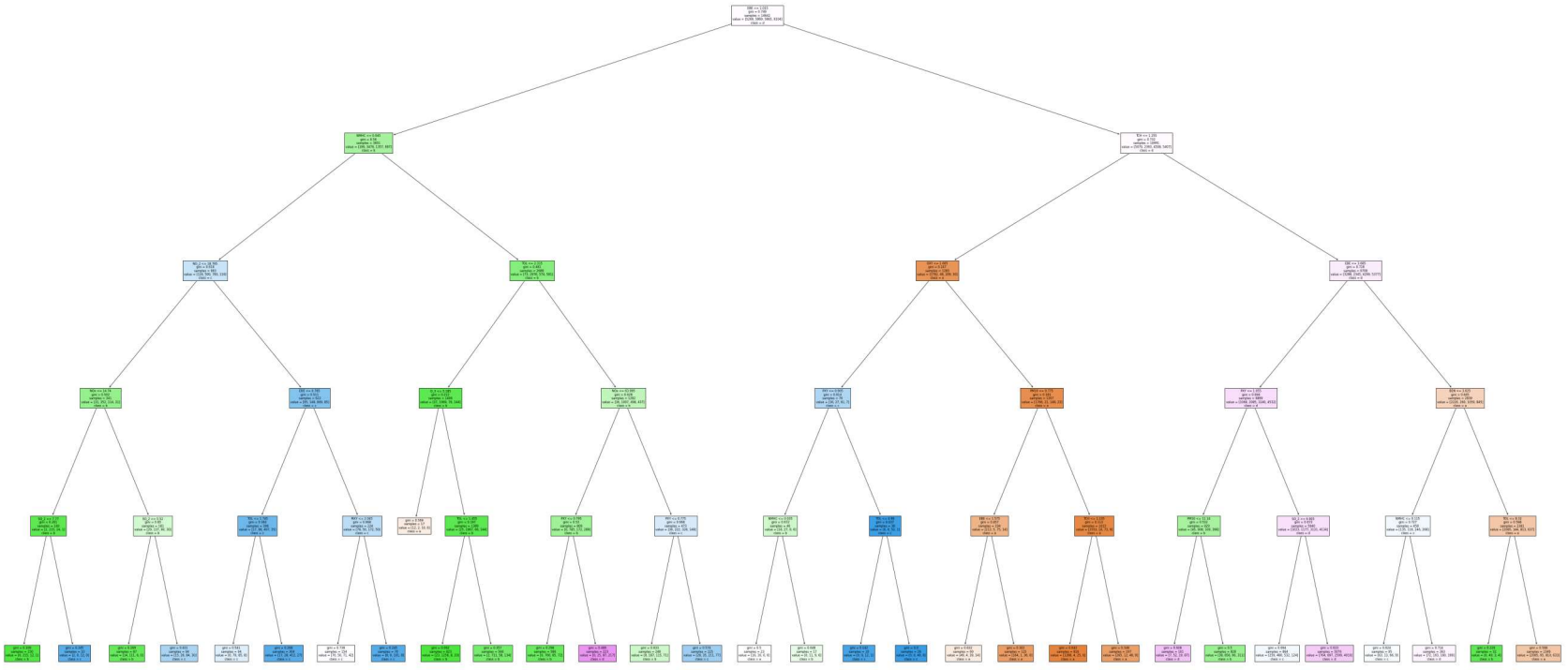
```
from sklearn.tree import plot_tree

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(2166.9, 1993.2, 'EBE <= 1.015\ngini = 0.749\nsamples = 14642\nvalue = [5269, 5869, 5865, 6104]\nclass = d'),
 Text(1060.2, 1630.8000000000002, 'NMHC <= 0.045\ngini = 0.56\nsamples = 3651\nvalue = [199, 3476, 1357, 697]\nclass = b'),
 Text(595.2, 1268.4, 'NO_2 <= 18.795\ngini = 0.616\nsamples = 963\nvalue = [126, 500, 783, 116]\nclass = c'),
 Text(297.6, 906.0, 'NOx <= 14.74\ngini = 0.502\nsamples = 341\nvalue = [31, 352, 114, 31]\nclass = b'),
 Text(148.8, 543.5999999999999, 'SO_2 <= 7.77\ngini = 0.201\nsamples = 160\nvalue = [2, 215, 24, 1]\nclass = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.108\nsamples = 150\nvalue = [0, 215, 12, 1]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.245\nsamples = 10\nvalue = [2, 0, 12, 0]\nclass = c'),
 Text(446.40000000000003, 543.5999999999999, 'SO_2 <= 5.52\ngini = 0.65\nsamples = 181\nvalue = [29, 137, 90, 30]\nclass = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.269\nsamples = 87\nvalue = [14, 111, 6, 0]\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.631\nsamples = 94\nvalue = [15, 26, 84, 30]\nclass = c'),
 Text(892.8000000000001, 906.0, 'EBE <= 0.785\ngini = 0.511\nsamples = 622\nvalue = [95, 148, 669, 85]\nclass = c'),
 Text(744.0, 543.5999999999999, 'TOL <= 1.745\ngini = 0.383\nsamples = 398\nvalue = [17, 98, 497, 35]\nclass = c'),
 Text(669.6, 181.19999999999982, 'gini = 0.541\nsamples = 94\nvalue = [0, 70, 85, 8]\nclass = c'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.268\nsamples = 304\nvalue = [17, 28, 412, 27]\nclass = c'),
 Text(1041.6000000000001, 543.5999999999999, 'MXY <= 2.045\ngini = 0.668\nsamples = 224\nvalue = [78, 50, 172, 50]\nclass = c'),
 Text(967.2, 181.19999999999982, 'gini = 0.738\nsamples = 154\nvalue = [70, 50, 71, 42]\nclass = c'),
 Text(1116.0, 181.19999999999982, 'gini = 0.245\nsamples = 70\nvalue = [8, 0, 101, 8]\nclass = c'),
 Text(1525.2, 1268.4, 'TOL <= 2.315\ngini = 0.461\nsamples = 2688\nvalue = [73, 2976, 574, 581]\nclass = b'),
 Text(1264.8000000000002, 906.0, 'O_3 <= 5.185\ngini = 0.212\nsamples = 1406\nvalue = [37, 1969, 76, 144]\nclass = b'),
 Text(1190.4, 543.5999999999999, 'gini = 0.569\nsamples = 17\nvalue = [12, 2, 10, 0]\nclass = a'),
 Text(1339.2, 543.5999999999999, 'TOL <= 1.455\ngini = 0.197\nsamples = 1389\nvalue = [25, 1967, 66, 144]\nclass = b'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.062\nsamples = 823\nvalue = [23, 1256, 8, 10]\nclass = b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.357\nsamples = 566\nvalue = [2, 711, 58, 134]\nclass = b'),
 Text(1785.6000000000001, 906.0, 'NOx <= 63.995\ngini = 0.628\nsamples = 1282\nvalue = [36, 1007, 498, 437]\nclass = b'),
 Text(1636.8000000000002, 543.5999999999999, 'PXY <= 0.795\ngini = 0.53\nsamples = 809\nvalue = [0, 785, 172, 289]\nclass = b'),

```
 Text(1562.4, 181.19999999999982, 'gini = 0.298\nsamples = 586\nvalue = [0, 760, 85, 72]\nclass = b'),
 Text(1711.2, 181.19999999999982, 'gini = 0.489\nsamples = 223\nvalue = [0, 25, 87, 217]\nclass = d'),
 Text(1934.4, 543.5999999999999, 'PXY <= 0.775\ngini = 0.666\nsamples = 473\nvalue = [36, 222, 326, 148]\ncl
ass = c'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.633\nsamples = 248\nvalue = [8, 187, 115, 71]\nclass
= b'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.574\nsamples = 225\nvalue = [28, 35, 211, 77]\nclass
= c'),
 Text(3273.6000000000004, 1630.8000000000002, 'TCH <= 1.255\ngini = 0.732\nsamples = 10991\nvalue = [5070, 2
393, 4508, 5407]\nclass = d'),
 Text(2678.4, 1268.4, 'OXY <= 1.005\ngini = 0.247\nsamples = 1283\nvalue = [1782, 48, 209, 30]\nclass = a'),
 Text(2380.8, 906.0, 'PXY <= 0.905\ngini = 0.614\nsamples = 76\nvalue = [16, 27, 61, 7]\nclass = c'),
 Text(2232.0, 543.5999999999999, 'NMHC <= 0.035\ngini = 0.672\nsamples = 40\nvalue = [16, 27, 9, 6]\nclass =
b'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.5\nsamples = 23\nvalue = [16, 16, 0, 0]\nclass =
a'),
 Text(2306.4, 181.19999999999982, 'gini = 0.648\nsamples = 17\nvalue = [0, 11, 9, 6]\nclass = b'),
 Text(2529.6000000000004, 543.5999999999999, 'TOL <= 4.96\ngini = 0.037\nsamples = 36\nvalue = [0, 0, 52, 1]
\nclass = c'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.142\nsamples = 10\nvalue = [0, 0, 12, 1]\nclass =
c'),
 Text(2604.0, 181.19999999999982, 'gini = 0.0\nsamples = 26\nvalue = [0, 0, 40, 0]\nclass = c'),
 Text(2976.0, 906.0, 'PM10 <= 9.775\ngini = 0.181\nsamples = 1207\nvalue = [1766, 21, 148, 23]\nclass = a'),
 Text(2827.2000000000003, 543.5999999999999, 'EBE <= 1.575\ngini = 0.457\nsamples = 194\nvalue = [213, 5, 7
5, 14]\nclass = a'),
 Text(2752.8, 181.19999999999982, 'gini = 0.632\nsamples = 69\nvalue = [49, 4, 39, 14]\nclass = a'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.302\nsamples = 125\nvalue = [164, 1, 36, 0]\nclass =
a'),
 Text(3124.8, 543.5999999999999, 'TCH <= 1.235\ngini = 0.113\nsamples = 1013\nvalue = [1553, 16, 73, 9]\ncla
ss = a'),
 Text(3050.4, 181.19999999999982, 'gini = 0.043\nsamples = 816\nvalue = [1288, 4, 25, 0]\nclass = a'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.348\nsamples = 197\nvalue = [265, 12, 48, 9]\nclass
= a'),
 Text(3868.8, 1268.4, 'EBE <= 3.685\ngini = 0.728\nsamples = 9708\nvalue = [3288, 2345, 4299, 5377]\nclass =
d'),
 Text(3571.2000000000003, 906.0, 'PXY <= 1.055\ngini = 0.694\nsamples = 6869\nvalue = [1068, 2085, 3240, 453
2]\nclass = d'),
 Text(3422.4, 543.5999999999999, 'PM10 <= 11.14\ngini = 0.532\nsamples = 929\nvalue = [45, 908, 109, 398]\nc
lass = b'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.608\nsamples = 101\nvalue = [7, 52, 19, 87]\nclass =
d'),
 Text(3496.8, 181.19999999999982, 'gini = 0.5\nsamples = 828\nvalue = [38, 856, 90, 311]\nclass = b'),
 Text(3720.0000000000005, 543.5999999999999, 'SO_2 <= 6.905\ngini = 0.673\nsamples = 5940\nvalue = [1023, 11
```

```
77, 3131, 4134]\nclass = d'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.694\nsamples = 864\nvalue = [259, 480, 532, 124]\ncl
ass = c'),
 Text(3794.4, 181.19999999999982, 'gini = 0.633\nsamples = 5076\nvalue = [764, 697, 2599, 4010]\nclass =
d'),
 Text(4166.400000000001, 906.0, 'BEN <= 3.025\ngini = 0.645\nsamples = 2839\nvalue = [2220, 260, 1059, 845]
\nclass = a'),
 Text(4017.6000000000004, 543.5999999999999, 'NMHC <= 0.115\ngini = 0.727\nsamples = 458\nvalue = [135, 116,
246, 208]\nclass = c'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.624\nsamples = 95\nvalue = [63, 13, 66, 9]\nclass =
c'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.714\nsamples = 363\nvalue = [72, 103, 180, 199]\ncla
ss = d'),
 Text(4315.200000000001, 543.5999999999999, 'TOL <= 8.32\ngini = 0.598\nsamples = 2381\nvalue = [2085, 144,
813, 637]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.226\nsamples = 32\nvalue = [0, 49, 3, 4]\nclass = b'),
 Text(4389.6, 181.19999999999982, 'gini = 0.588\nsamples = 2349\nvalue = [2085, 95, 810, 633]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression0.17328056958079485

Ridge Regression:0.17227501231797626

Lasso Regression:0.03578590238383095

ElasticNet Regression:0.04813959207478369

Logistic Regression:0.7584974250227204

Random Forest:0.7276148693348015

Logistic Regression is suitable for this dataset