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_2010")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH	TOL
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	NaN	NaN	NaN	10.15	NaN	NaN
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	NaN	NaN	NaN	12.24	NaN	NaN
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	NaN	NaN	NaN	NaN	NaN	NaN
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.410000	7.870000	NaN	10.06	1.52	1.49
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.670000	22.030001	NaN	10.68	NaN	2.88
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	NaN	NaN	NaN	NaN	NaN	NaN
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259998	NaN	NaN	7.26	NaN	NaN
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	NaN	NaN	NaN	NaN	1.47	NaN
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	NaN	NaN	NaN	NaN	NaN	NaN
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.150002	26.860001	NaN	7.03	1.44	2.87

209448 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: 6666 entries, 11 to 191927
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     6666 non-null   object 
 1   BEN      6666 non-null   float64
 2   CO       6666 non-null   float64
 3   EBE      6666 non-null   float64
 4   MXY      6666 non-null   float64
 5   NMHC     6666 non-null   float64
 6   NO_2     6666 non-null   float64
 7   NOx      6666 non-null   float64
 8   OXY      6666 non-null   float64
 9   O_3      6666 non-null   float64
 10  PM10     6666 non-null   float64
 11  PM25     6666 non-null   float64
 12  PXY      6666 non-null   float64
 13  SO_2     6666 non-null   float64
 14  TCH      6666 non-null   float64
 15  TOL      6666 non-null   float64
 16  station  6666 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 937.4+ KB
```

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

Out[6]:

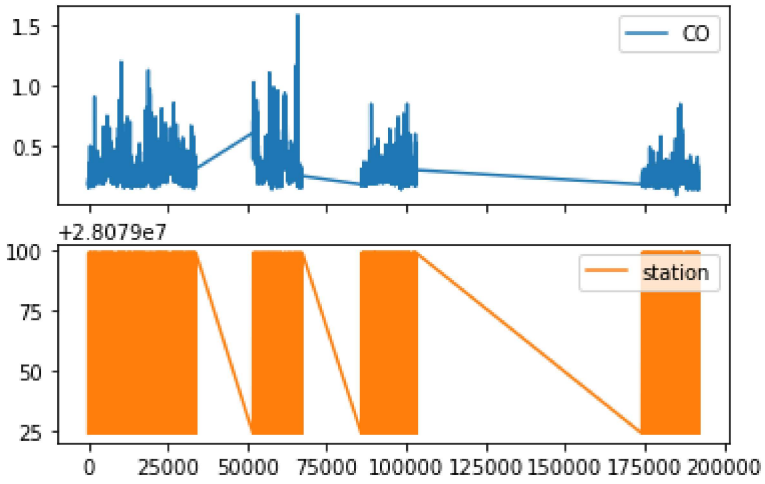
	CO	station
11	0.18	28079024
23	0.23	28079099
35	0.17	28079024
47	0.21	28079099
59	0.16	28079024
...	...	...
191879	0.26	28079099
191891	0.16	28079024
191903	0.28	28079099
191915	0.16	28079024
191927	0.25	28079099

6666 rows × 2 columns

Line chart

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

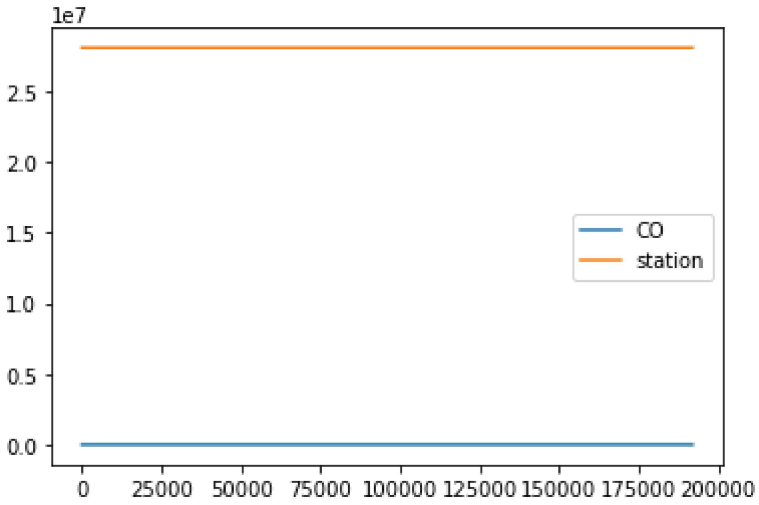
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

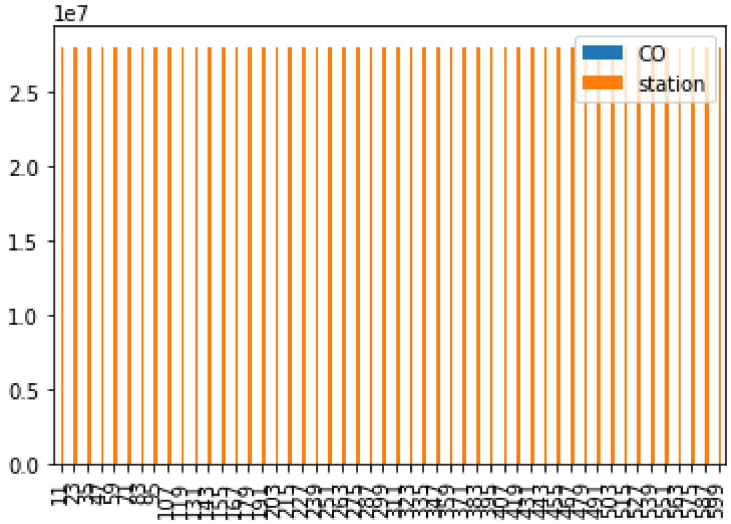


Bar chart

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

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

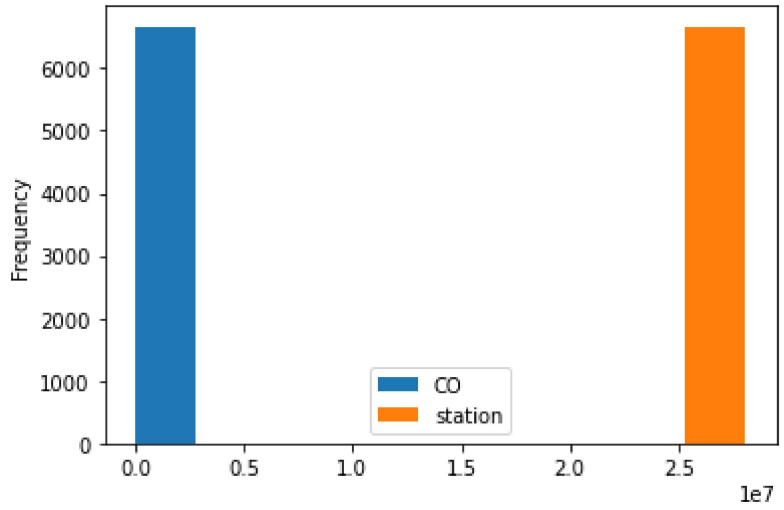
Out[10]: <AxesSubplot:>



Histogram

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

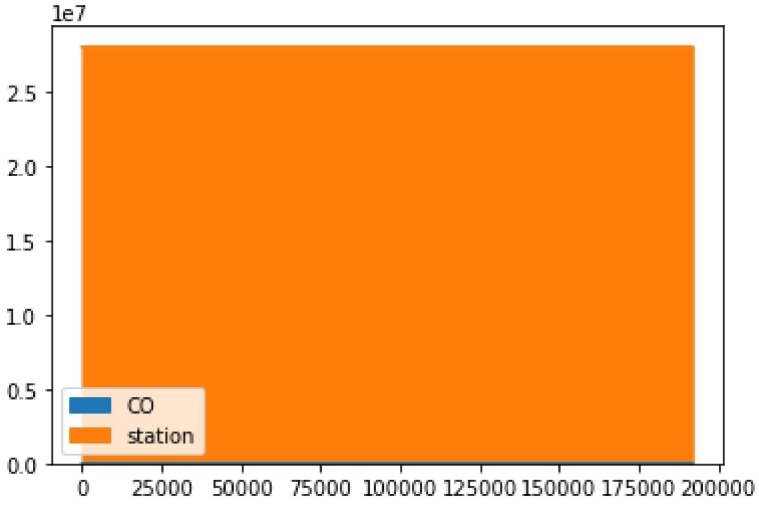
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

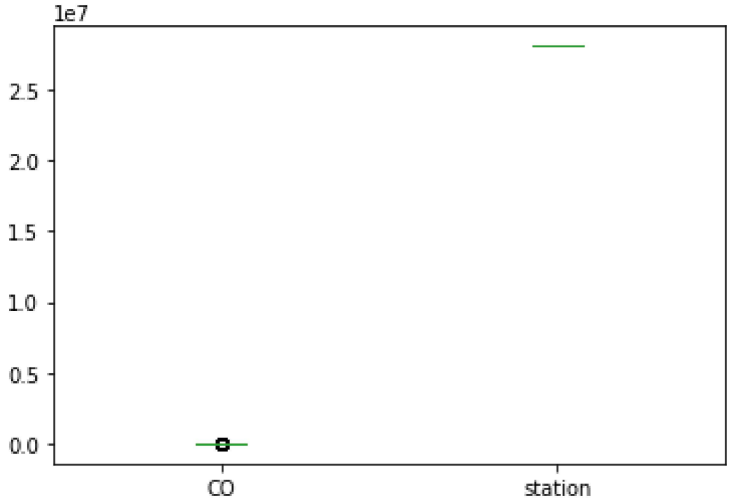
Out[12]: <AxesSubplot:>



Box chart

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

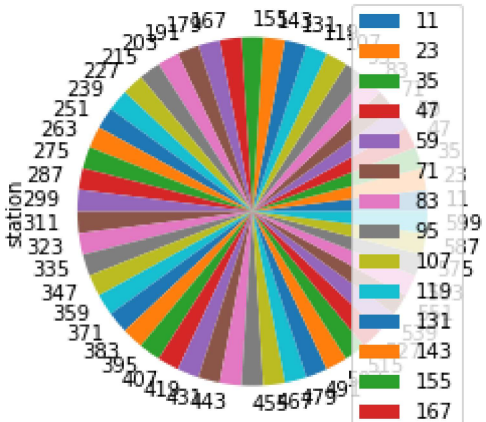
Out[13]: <AxesSubplot:>



Pie chart

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

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

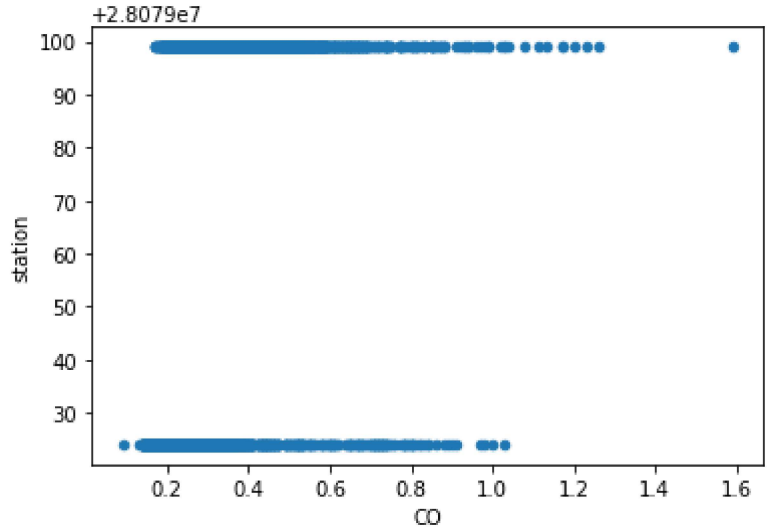

station
11
23
35
47
59
71
83
95
107
119
131
143
155
167
179
191
203
215
227
239
251
263
275
287
299
311
323
335
347
359
371
383
395
407
419
431
443
455
467
479
491
503
515
527
539
551
563
575

599

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: 6666 entries, 11 to 191927
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     6666 non-null   object
 1   BEN      6666 non-null   float64
 2   CO       6666 non-null   float64
 3   EBE      6666 non-null   float64
 4   MXY      6666 non-null   float64
 5   NMHC     6666 non-null   float64
 6   NO_2     6666 non-null   float64
 7   NOx      6666 non-null   float64
 8   OXY      6666 non-null   float64
 9   O_3      6666 non-null   float64
 10  PM10     6666 non-null   float64
 11  PM25     6666 non-null   float64
 12  PXY      6666 non-null   float64
 13  SO_2     6666 non-null   float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
count	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.0000
mean	0.648425	0.296280	0.840585	0.839959	0.243378	33.888744	47.540617	0.916668	56.246101	17.1473
std	0.395346	0.133296	0.508031	0.382263	0.115730	23.465169	41.230578	0.192521	30.380535	13.6790
min	0.170000	0.090000	0.140000	0.110000	0.000000	1.290000	2.760000	0.200000	0.600000	1.3200
25%	0.380000	0.200000	0.470000	0.590000	0.180000	15.752500	19.442501	1.000000	31.740000	8.9700
50%	0.540000	0.260000	0.755000	1.000000	0.220000	29.320000	36.770000	1.000000	57.750000	14.0000
75%	0.810000	0.340000	1.000000	1.000000	0.280000	47.657500	62.102501	1.000000	79.489998	21.1500
max	5.110000	1.590000	5.190000	6.810000	0.930000	133.399994	409.299988	2.760000	161.100006	174.6999

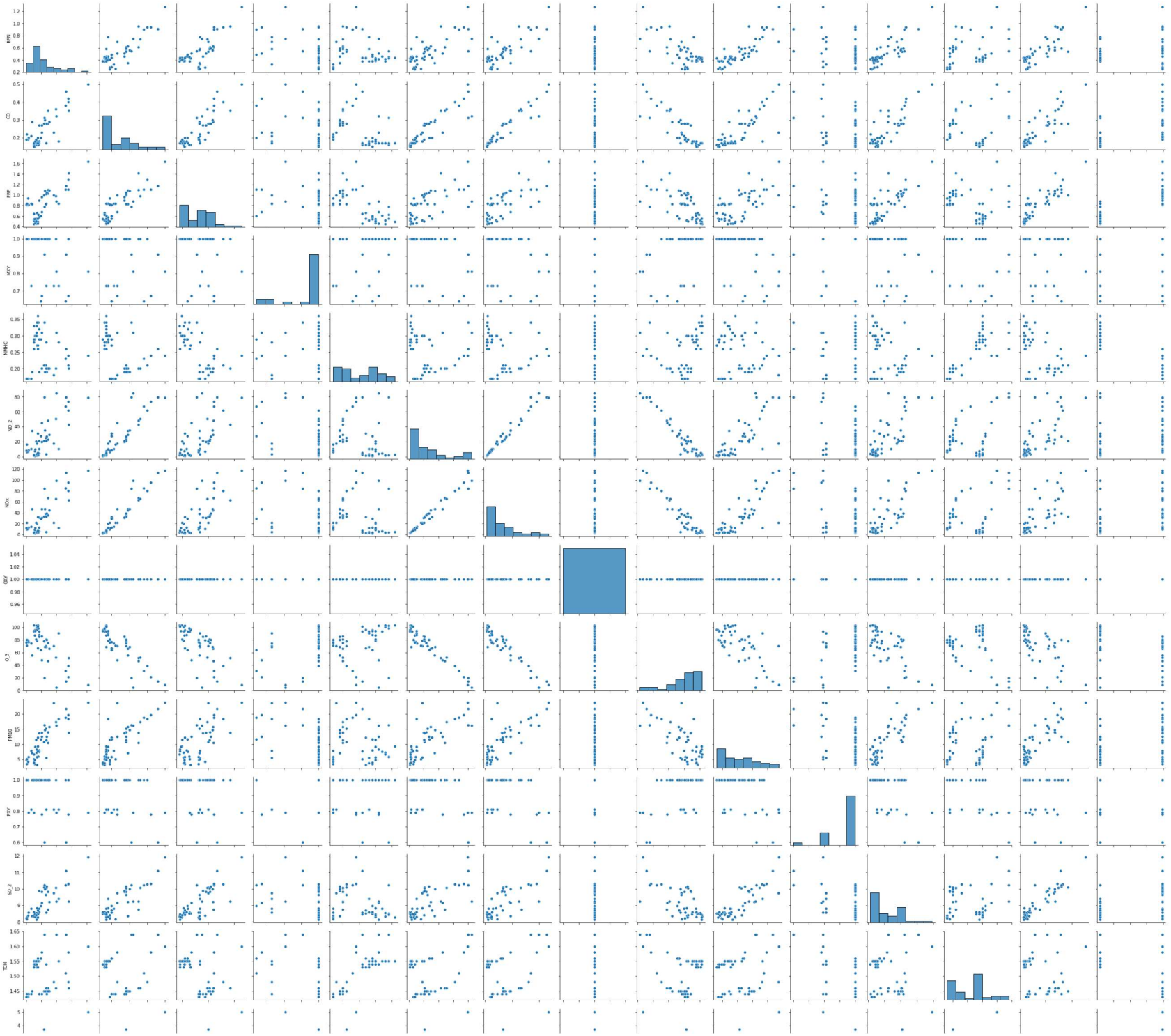
In [18]:

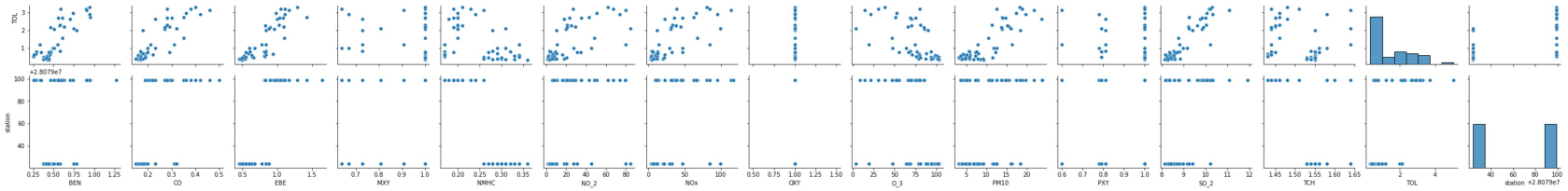
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x2868f6c7220>
```

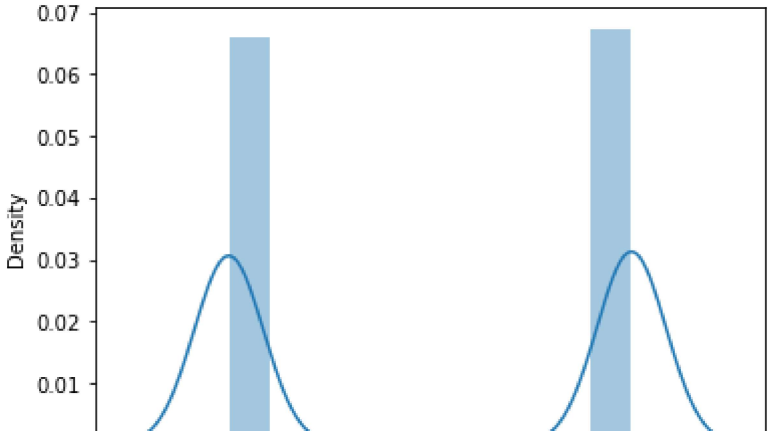

BEN
CO
EBE
MXY
NMHC
NO_2
NOx
OXY
O_3
PM10
PXY
SO_2
TCH



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

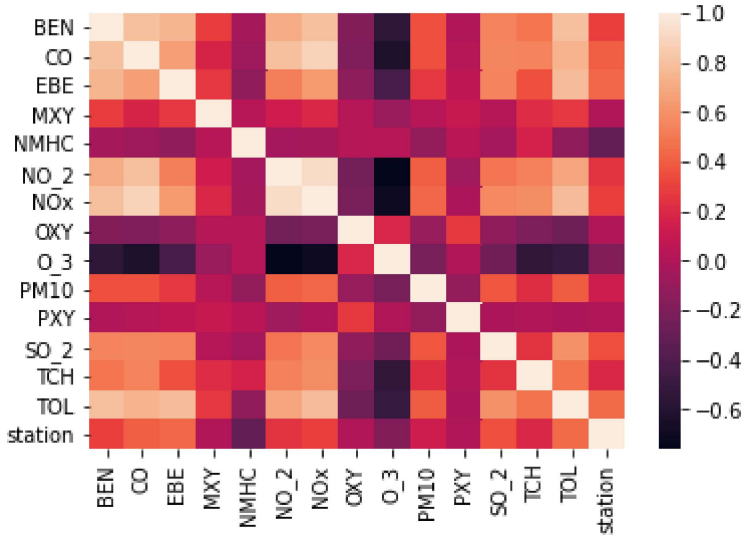
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28078931.02099493
```

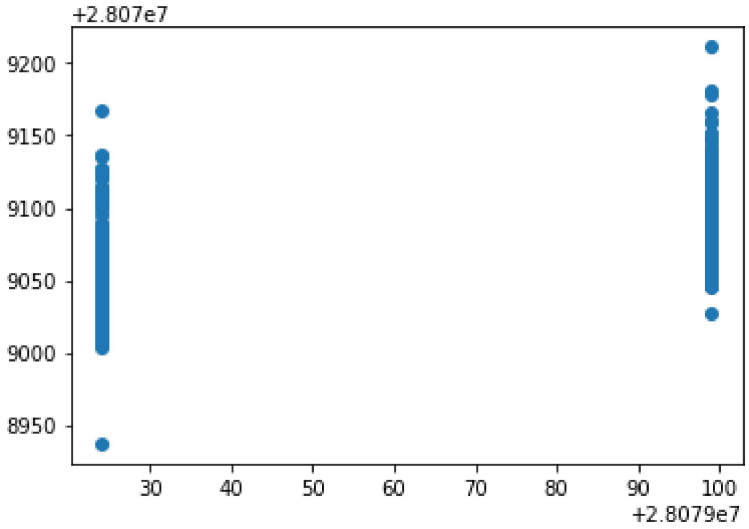
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-35.980070
CO	174.138112
EBE	16.081525
MXY	-9.700544
NMHC	-70.949113
NO_2	0.278852
NOx	-0.637720
OXY	33.041587
O_3	0.067798
PM10	-0.165901
PXY	-5.456597
SO_2	1.700364
TCH	48.133074
TOL	10.323340

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

Out[27]: <matplotlib.collections.PathCollection at 0x2869d6ee880>



ACCURACY

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

Out[28]: 0.41834875044261377

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

Out[29]: 0.4313187068979043

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]:
```
0.41071757471844017
```

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

Out[33]:
```
0.4183715947196103
```

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]:
```
0.1793088808576867
```

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

Out[36]:
```
0.18729059757800426
```

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([-0.        ,  0.20397676,  2.73065235, -1.09460886, -1.23331305,
        0.        , -0.097484  ,  0.65885519, -0.03918052, -0.11680219,
       -0.        ,  2.44426388,  0.        ,  7.22599835])
```

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

Out[39]:
```
28079028.297450814
```

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

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

Out[41]:
```
0.2426219931577691
```

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)))
```

```
30.542128097679466
1064.8540698087513
32.63210182946773
```

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]: (6666, 14)

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

Out[46]: (6666,)

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([28079024, 28079099], dtype=int64)

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

Out[53]: 0.8660366036603661

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

Out[54]: 0.0

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

Out[55]: array([[0., 1.]])

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]:
```
0.9258465495070725
```

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

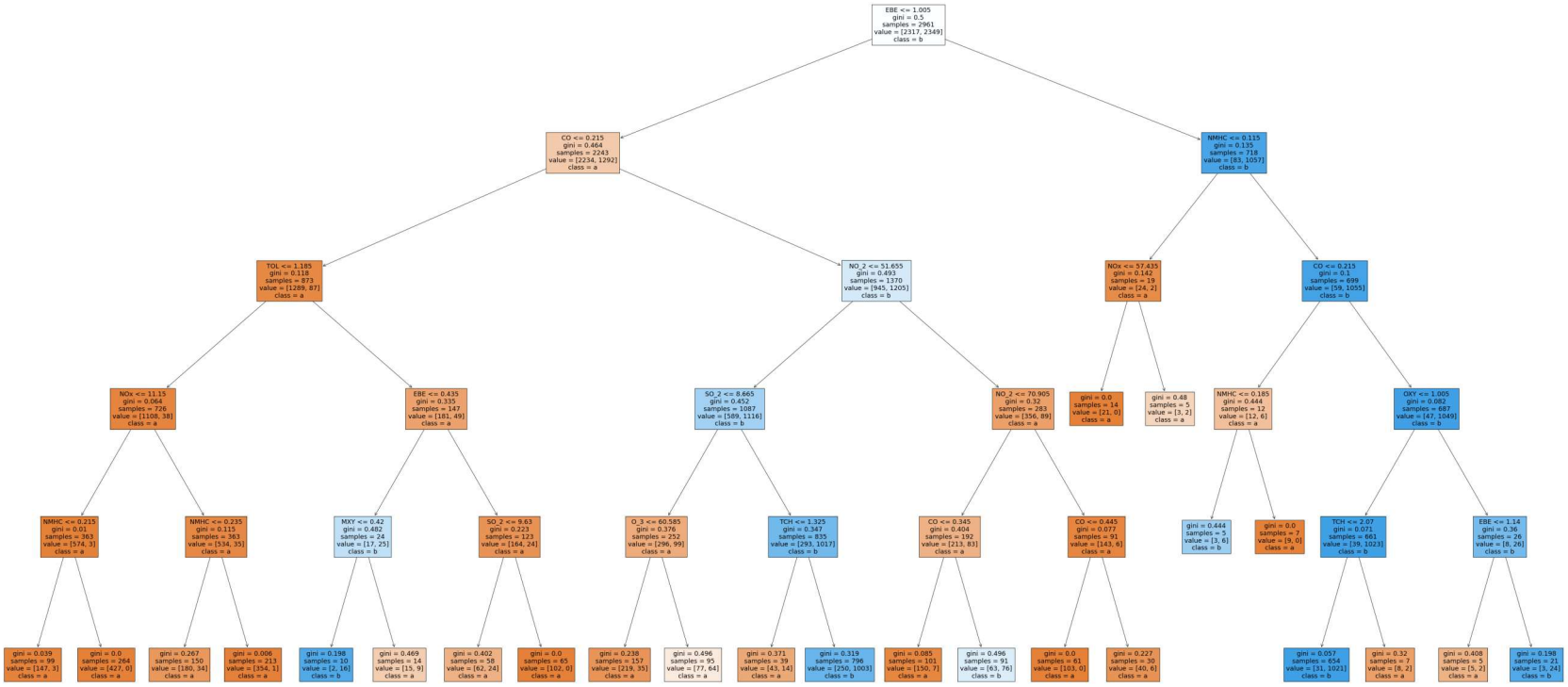
In [62]:

```
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(2582.3720930232557, 1993.2, 'EBE <= 1.005\ngini = 0.5\nsamples = 2961\nvalue = [2317, 2349]\nclass = b'),
 Text(1661.0232558139535, 1630.8000000000002, 'CO <= 0.215\ngini = 0.464\nsamples = 2243\nvalue = [2234, 1292]\nclass = a'),
 Text(830.5116279069767, 1268.4, 'TOL <= 1.185\ngini = 0.118\nsamples = 873\nvalue = [1289, 87]\nclass = a'),
 Text(415.25581395348837, 906.0, 'NOx <= 11.15\ngini = 0.064\nsamples = 726\nvalue = [1108, 38]\nclass = a'),
 Text(207.62790697674419, 543.5999999999999, 'NMHC <= 0.215\ngini = 0.01\nsamples = 363\nvalue = [574, 3]\nclass = a'),
 Text(103.81395348837209, 181.19999999999982, 'gini = 0.039\nsamples = 99\nvalue = [147, 3]\nclass = a'),
 Text(311.4418604651163, 181.19999999999982, 'gini = 0.0\nsamples = 264\nvalue = [427, 0]\nclass = a'),
 Text(622.8837209302326, 543.5999999999999, 'NMHC <= 0.235\ngini = 0.115\nsamples = 363\nvalue = [534, 35]\nclass = a'),
 Text(519.0697674418604, 181.19999999999982, 'gini = 0.267\nsamples = 150\nvalue = [180, 34]\nclass = a'),
 Text(726.6976744186046, 181.19999999999982, 'gini = 0.006\nsamples = 213\nvalue = [354, 1]\nclass = a'),
 Text(1245.7674418604652, 906.0, 'EBE <= 0.435\ngini = 0.335\nsamples = 147\nvalue = [181, 49]\nclass = a'),
 Text(1038.139534883721, 543.5999999999999, 'MXY <= 0.42\ngini = 0.482\nsamples = 24\nvalue = [17, 25]\nclass = b'),
 Text(934.3255813953489, 181.19999999999982, 'gini = 0.198\nsamples = 10\nvalue = [2, 16]\nclass = b'),
 Text(1141.953488372093, 181.19999999999982, 'gini = 0.469\nsamples = 14\nvalue = [15, 9]\nclass = a'),
 Text(1453.3953488372092, 543.5999999999999, 'SO_2 <= 9.63\ngini = 0.223\nsamples = 123\nvalue = [164, 24]\nclass = a'),
 Text(1349.5813953488373, 181.19999999999982, 'gini = 0.402\nsamples = 58\nvalue = [62, 24]\nclass = a'),
 Text(1557.2093023255813, 181.19999999999982, 'gini = 0.0\nsamples = 65\nvalue = [102, 0]\nclass = a'),
 Text(2491.5348837209303, 1268.4, 'NO_2 <= 51.655\ngini = 0.493\nsamples = 1370\nvalue = [945, 1205]\nclass = b'),
 Text(2076.279069767442, 906.0, 'SO_2 <= 8.665\ngini = 0.452\nsamples = 1087\nvalue = [589, 1116]\nclass = b'),
 Text(1868.6511627906978, 543.5999999999999, 'O_3 <= 60.585\ngini = 0.376\nsamples = 252\nvalue = [296, 99]\nclass = a'),
 Text(1764.8372093023256, 181.19999999999982, 'gini = 0.238\nsamples = 157\nvalue = [219, 35]\nclass = a'),
 Text(1972.4651162790697, 181.19999999999982, 'gini = 0.496\nsamples = 95\nvalue = [77, 64]\nclass = a'),
 Text(2283.906976744186, 543.5999999999999, 'TCH <= 1.325\ngini = 0.347\nsamples = 835\nvalue = [293, 1017]\nclass = b'),
 Text(2180.093023255814, 181.19999999999982, 'gini = 0.371\nsamples = 39\nvalue = [43, 14]\nclass = a'),
 Text(2387.720930232558, 181.19999999999982, 'gini = 0.319\nsamples = 796\nvalue = [250, 1003]\nclass = b'),
 Text(2906.7906976744184, 906.0, 'NO_2 <= 70.905\ngini = 0.32\nsamples = 283\nvalue = [356, 89]\nclass = a'),
 Text(2699.1627906976746, 543.5999999999999, 'CO <= 0.345\ngini = 0.404\nsamples = 192\nvalue = [213, 83]\nclass = a'),
 Text(2595.3488372093025, 181.19999999999982, 'gini = 0.085\nsamples = 101\nvalue = [150, 7]\nclass = a'),
 Text(2802.9767441860463, 181.19999999999982, 'gini = 0.496\nsamples = 91\nvalue = [63, 76]\nclass = b'),

```
 Text(3114.4186046511627, 543.5999999999999, 'CO <= 0.445\ngini = 0.077\nsamples = 91\nvalue = [143, 6]\ncla
ss = a'),
 Text(3010.6046511627906, 181.19999999999982, 'gini = 0.0\nsamples = 61\nvalue = [103, 0]\nclass = a'),
 Text(3218.232558139535, 181.19999999999982, 'gini = 0.227\nsamples = 30\nvalue = [40, 6]\nclass = a'),
 Text(3503.720930232558, 1630.8000000000002, 'NMHC <= 0.115\ngini = 0.135\nsamples = 718\nvalue = [83, 1057]
\nclass = b'),
 Text(3218.232558139535, 1268.4, 'NOx <= 57.435\ngini = 0.142\nsamples = 19\nvalue = [24, 2]\nclass = a'),
 Text(3114.4186046511627, 906.0, 'gini = 0.0\nsamples = 14\nvalue = [21, 0]\nclass = a'),
 Text(3322.046511627907, 906.0, 'gini = 0.48\nsamples = 5\nvalue = [3, 2]\nclass = a'),
 Text(3789.2093023255816, 1268.4, 'CO <= 0.215\ngini = 0.1\nsamples = 699\nvalue = [59, 1055]\nclass = b'),
 Text(3529.6744186046512, 906.0, 'NMHC <= 0.185\ngini = 0.444\nsamples = 12\nvalue = [12, 6]\nclass = a'),
 Text(3425.860465116279, 543.5999999999999, 'gini = 0.444\nsamples = 5\nvalue = [3, 6]\nclass = b'),
 Text(3633.4883720930234, 543.5999999999999, 'gini = 0.0\nsamples = 7\nvalue = [9, 0]\nclass = a'),
 Text(4048.7441860465115, 906.0, 'OXY <= 1.005\ngini = 0.082\nsamples = 687\nvalue = [47, 1049]\nclass =
b'),
 Text(3841.1162790697676, 543.5999999999999, 'TCH <= 2.07\ngini = 0.071\nsamples = 661\nvalue = [39, 1023]\n
class = b'),
 Text(3737.3023255813955, 181.19999999999982, 'gini = 0.057\nsamples = 654\nvalue = [31, 1021]\nclass = b'),
 Text(3944.9302325581393, 181.19999999999982, 'gini = 0.32\nsamples = 7\nvalue = [8, 2]\nclass = a'),
 Text(4256.372093023256, 543.5999999999999, 'EBE <= 1.14\ngini = 0.36\nsamples = 26\nvalue = [8, 26]\nclass
= b'),
 Text(4152.558139534884, 181.19999999999982, 'gini = 0.408\nsamples = 5\nvalue = [5, 2]\nclass = a'),
 Text(4360.186046511628, 181.19999999999982, 'gini = 0.198\nsamples = 21\nvalue = [3, 24]\nclass = b')]
```



Conclusion

Accuracy

Linear Regression0.4313187068979043

Ridge Regression:0.4183715947196103

Lasso Regression:0.1793088808576867

ElasticNet Regression:0.2426219931577691

Logistic Regression:0.8660366036603661

Random Forest:0.9258465495070725

Random Forest is suitable for this dataset