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_2012")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	28079004
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4	28079008
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	28079011
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	28079056
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	28079057
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	28079058
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	28079059
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	28079060

210720 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:
```
1 df=df.dropna()
```

In [4]:
```
1 df.columns
```

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10916 entries, 6 to 210702
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     10916 non-null  object 
 1   BEN      10916 non-null  float64
 2   CO       10916 non-null  float64
 3   EBE      10916 non-null  float64
 4   NMHC     10916 non-null  float64
 5   NO       10916 non-null  float64
 6   NO_2     10916 non-null  float64
 7   O_3      10916 non-null  float64
 8   PM10     10916 non-null  float64
 9   PM25     10916 non-null  float64
 10  SO_2     10916 non-null  float64
 11  TCH      10916 non-null  float64
 12  TOL      10916 non-null  float64
 13  station  10916 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.2+ MB
```

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

Out[6]:

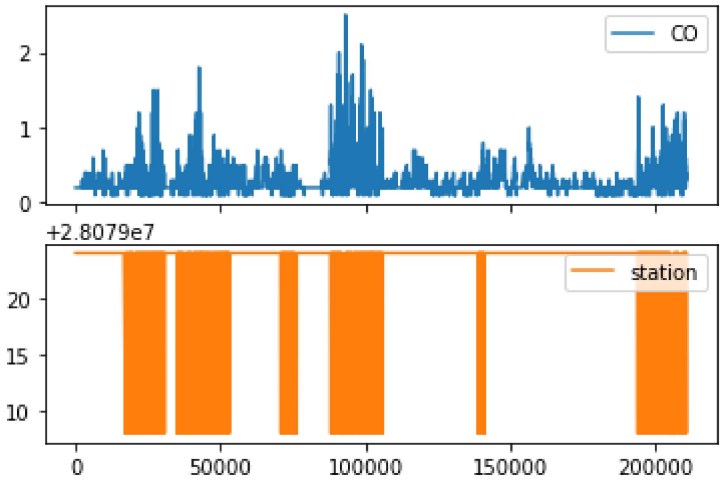
	CO	station
6	0.2	28079024
30	0.2	28079024
54	0.2	28079024
78	0.2	28079024
102	0.2	28079024
...	...	...
210654	0.3	28079024
210673	0.4	28079008
210678	0.3	28079024
210697	0.4	28079008
210702	0.3	28079024

10916 rows × 2 columns

Line chart

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

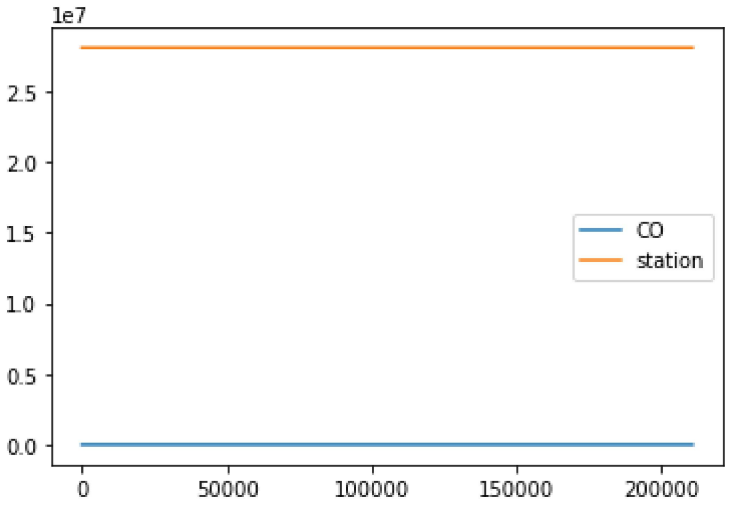
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

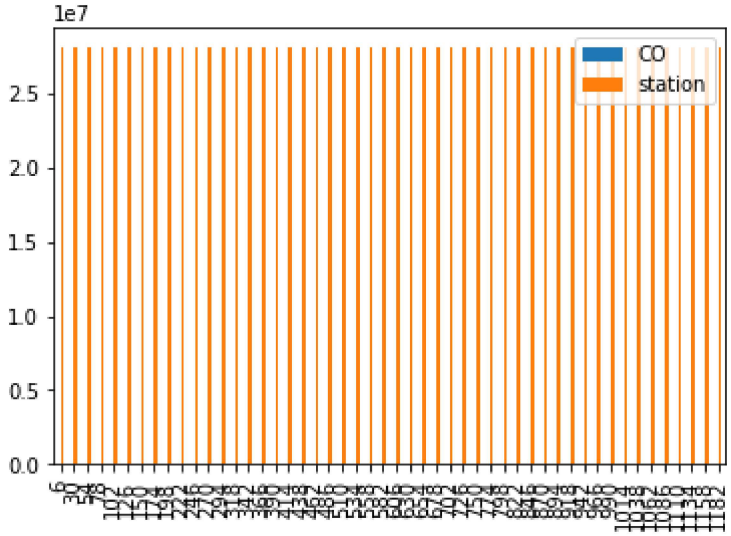


Bar chart

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

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

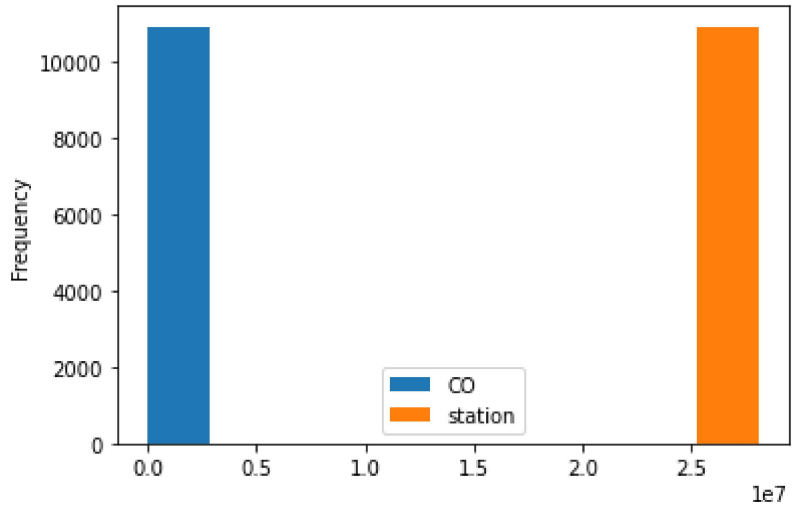
Out[10]: <AxesSubplot:>



Histogram

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

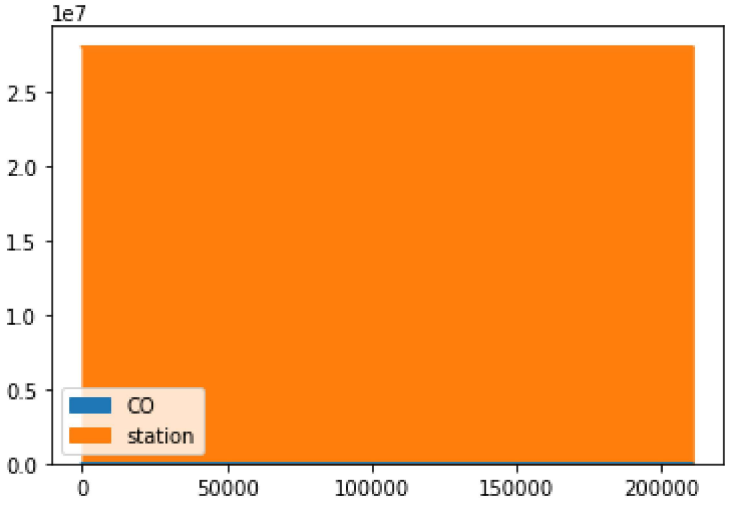
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

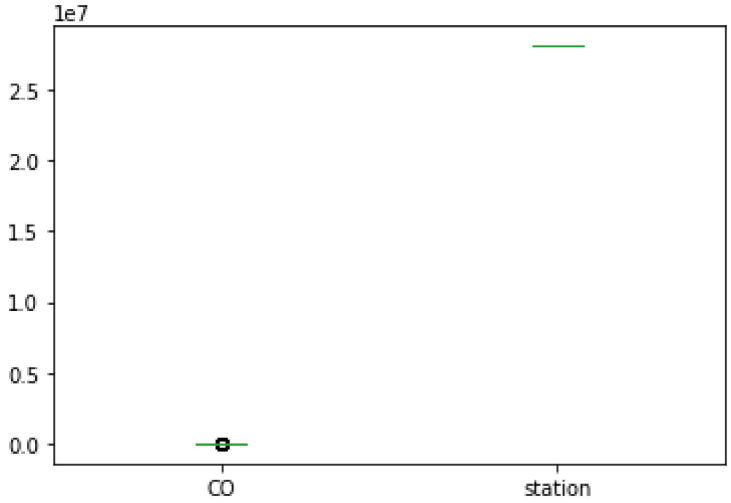
Out[12]: <AxesSubplot:>



Box chart

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

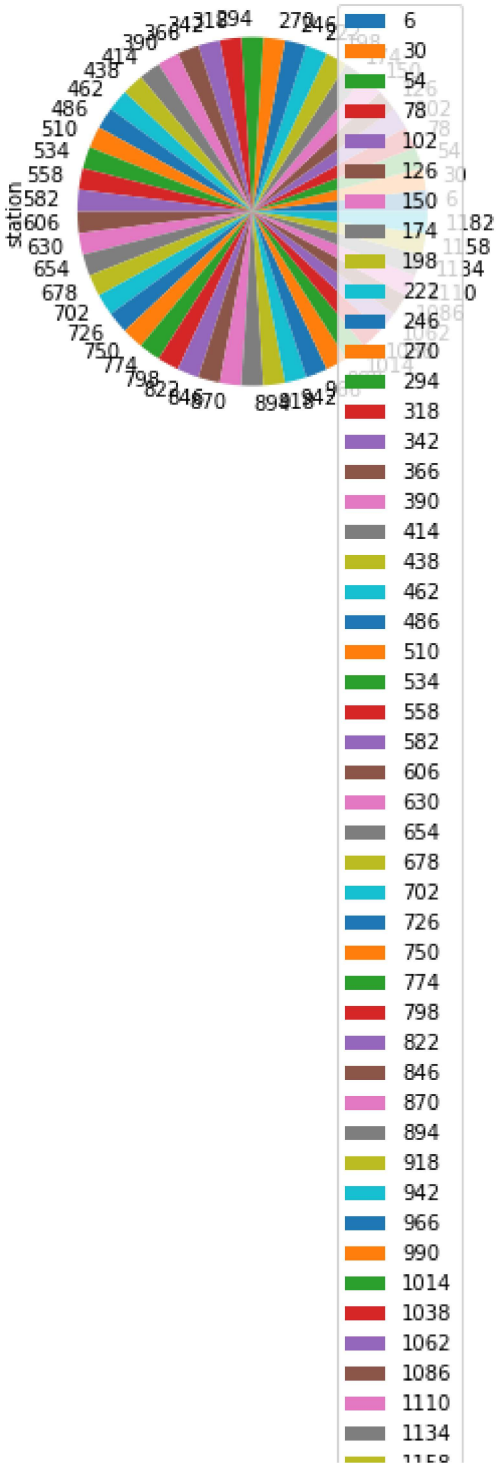
Out[13]: <AxesSubplot:>



Pie chart

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

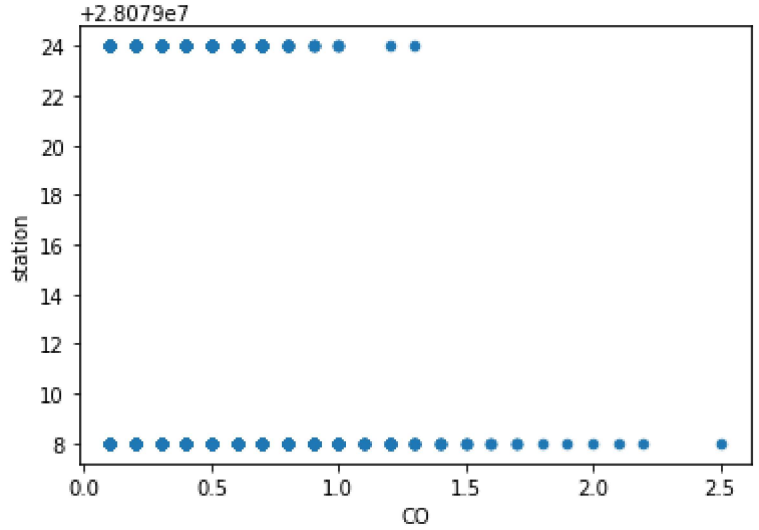
Out[14]:
```
<AxesSubplot:ylabel='station'>
```


station
6
30
54
78
102
126
150
174
198
222
246
270
294
318
342
366
390
414
438
462
486
510
534
558
582
606
630
654
678
702
726
750
774
798
822
846
870
894
918
942
966
990
1014
1038
1062
1086
1110
1134
1158

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: 10916 entries, 6 to 210702
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     10916 non-null  object
 1   BEN      10916 non-null  float64
 2   CO       10916 non-null  float64
 3   EBE      10916 non-null  float64
 4   NMHC     10916 non-null  float64
 5   NO       10916 non-null  float64
 6   NO_2     10916 non-null  float64
 7   O_3      10916 non-null  float64
 8   PM10     10916 non-null  float64
 9   PM25     10916 non-null  float64
 10  SO_2     10916 non-null  float64
 11  TCH      10916 non-null  float64
 12  TOL      10916 non-null  float64
 13  station  10916 non-null  int64
```

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

Out[17]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25
count	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000
mean	0.784014	0.279333	0.992213	0.215755	18.795529	31.262642	44.239557	22.875687	11.467662
std	0.632755	0.167922	0.804554	0.075169	40.038872	27.234732	29.535560	22.266862	8.303935
min	0.100000	0.100000	0.100000	0.050000	0.000000	1.000000	1.000000	1.000000	0.000000
25%	0.400000	0.200000	0.500000	0.160000	1.000000	9.000000	18.000000	10.000000	6.000000
50%	0.600000	0.200000	0.800000	0.220000	3.000000	24.000000	44.000000	17.000000	9.000000
75%	0.900000	0.300000	1.200000	0.250000	18.000000	47.000000	65.000000	28.000000	14.000000
max	7.000000	2.500000	9.700000	0.670000	525.000000	225.000000	157.000000	267.000000	96.000000

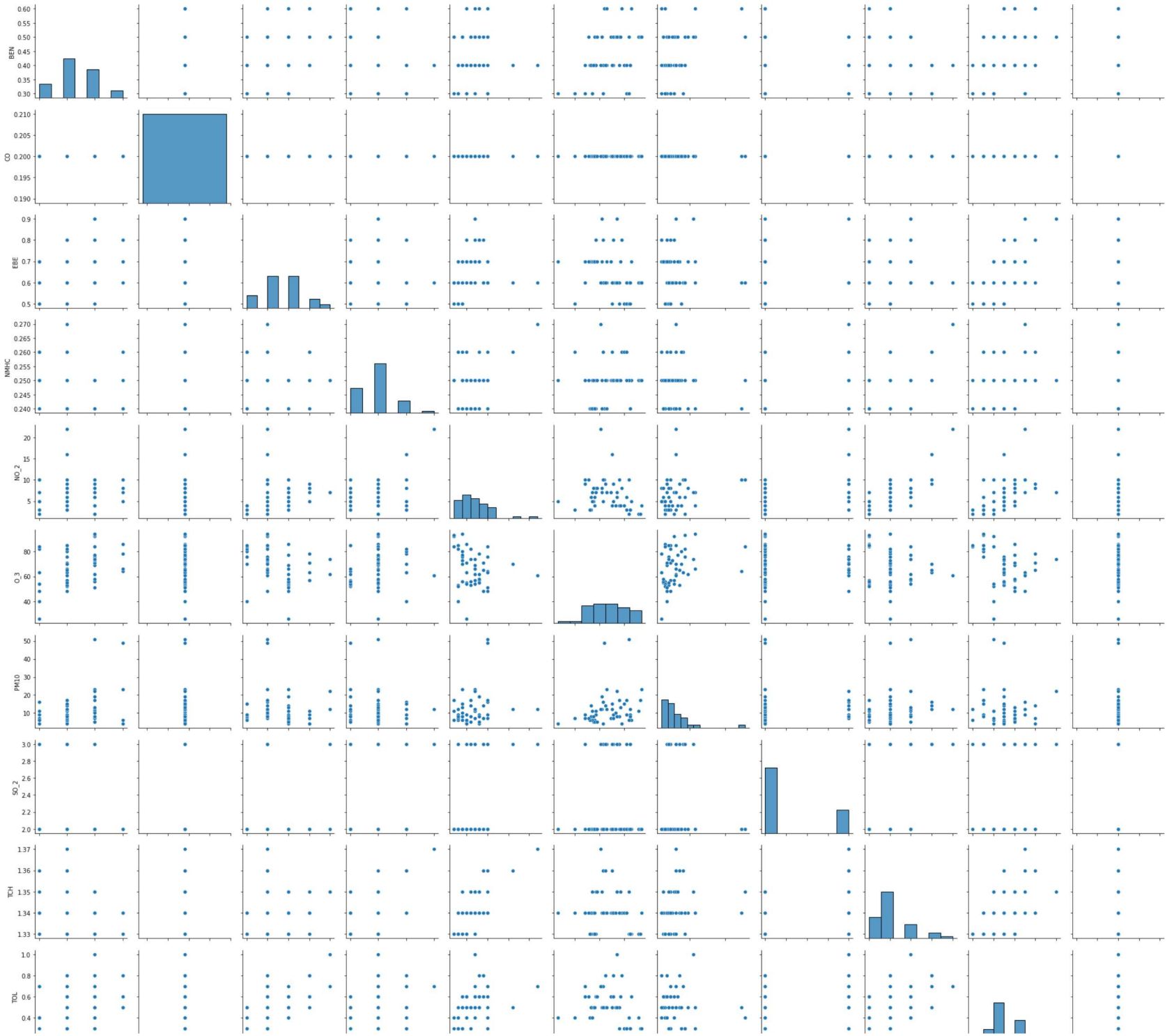
In [18]:

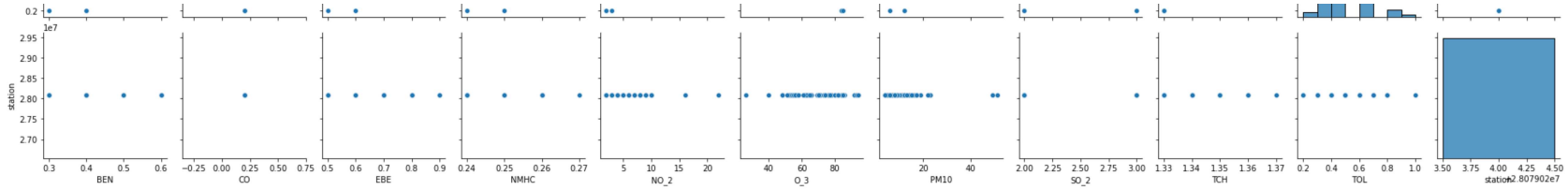
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1da679374f0>
```

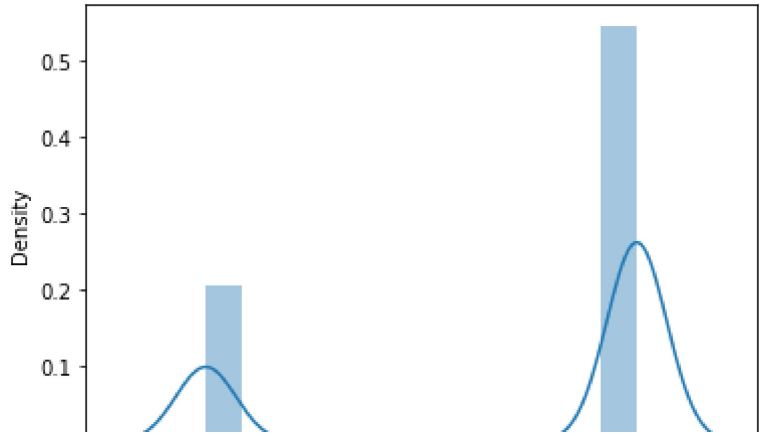

BEN
CO
EBE
NMHC
NO_2
O_3
PM10
SO_2
TCH
TOL



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

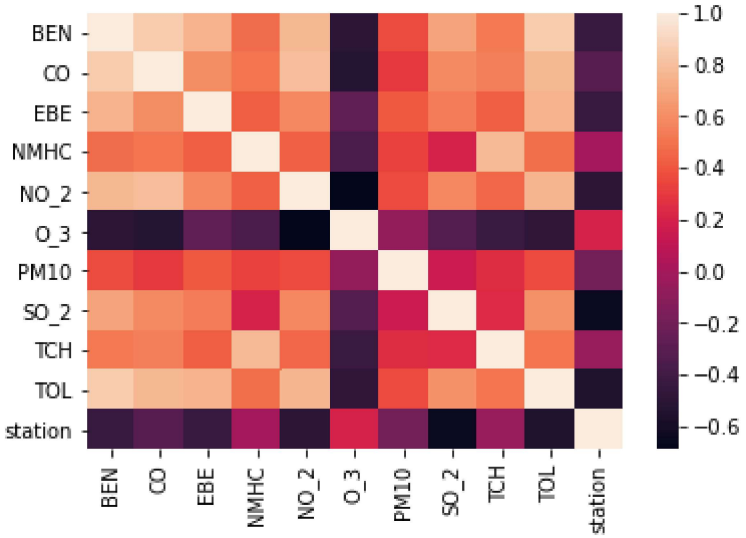
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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', 'NMHC', 'NO_2', 'O_3',
2        'PM10', '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]:
```
28079019.111901358
```

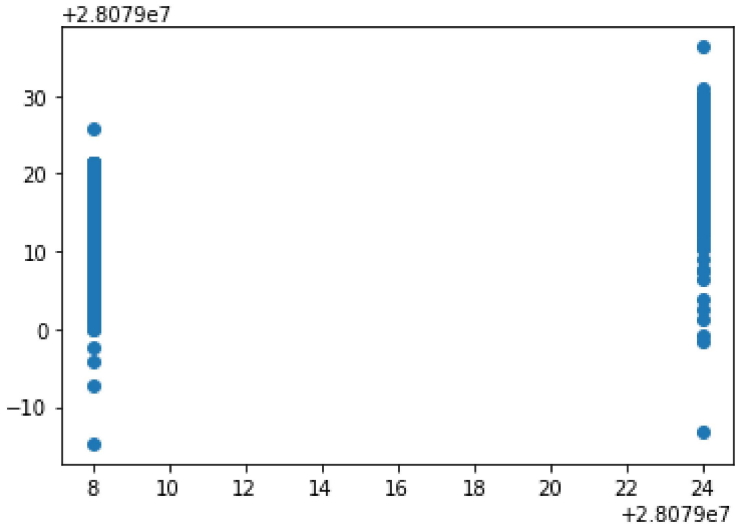
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	3.819738
CO	17.058184
EBE	-0.503572
NMHC	16.374528
NO_2	-0.113416
O_3	-0.028529
PM10	-0.009260
SO_2	-0.701373
TCH	1.196935
TOL	-1.471294

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

Out[27]: <matplotlib.collections.PathCollection at 0x1da70cb8880>



ACCURACY

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

Out[28]: 0.6123911138521282

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

Out[29]: 0.6254728538463179

Ridge and Lasso

In [30]:
```
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]: 0.6077859646862067

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

Out[33]: 0.623137940101393

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]: 0.3732245099621311

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

Out[36]: 0.3592306264705415

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.        , -0.        ,  0.        , -0.05919507,
       -0.03951284,  0.00295776, -0.58140864,  0.        , -0.35492406])
```

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

Out[39]:
```
28079026.74552566
```

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

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

Out[41]:
```
0.45947325040594844
```

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)))
```

```
3.821457859094589
27.461955688312848
5.240415602632376
```

Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [45]:
```
feature_matrix.shape
```

Out[45]: (10916, 10)

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

Out[46]: (10916,)

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]]
```

In [51]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

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

Out[52]: array([28079008, 28079024], dtype=int64)

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

Out[53]: 0.9293697325027482

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

Out[54]: 1.0

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

Out[55]: array([[1.00000000e+00, 3.50349553e-26]])

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]:
```
0.9611309982995597
```

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

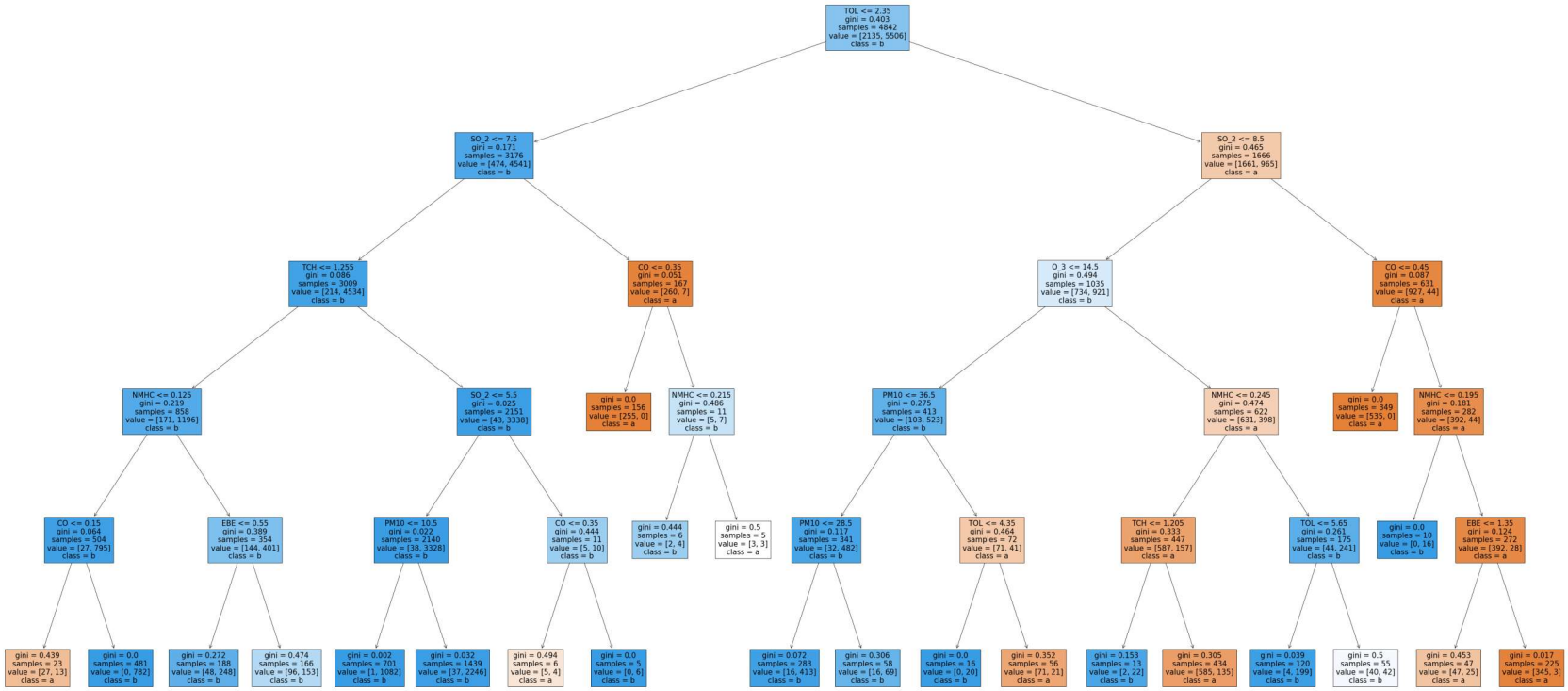
In [62]:

```
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(2466.9473684210525, 1993.2, 'TOL <= 2.35\ngini = 0.403\nsamples = 4842\nvalue = [2135, 5506]\nclass = b'),
 Text(1409.6842105263158, 1630.8000000000002, 'SO_2 <= 7.5\ngini = 0.171\nsamples = 3176\nvalue = [474, 4541]\nclass = b'),
 Text(939.7894736842105, 1268.4, 'TCH <= 1.255\ngini = 0.086\nsamples = 3009\nvalue = [214, 4534]\nclass = b'),
 Text(469.89473684210526, 906.0, 'NMHC <= 0.125\ngini = 0.219\nsamples = 858\nvalue = [171, 1196]\nclass = b'),
 Text(234.94736842105263, 543.5999999999999, 'CO <= 0.15\ngini = 0.064\nsamples = 504\nvalue = [27, 795]\nclass = b'),
 Text(117.47368421052632, 181.19999999999982, 'gini = 0.439\nsamples = 23\nvalue = [27, 13]\nclass = a'),
 Text(352.42105263157896, 181.19999999999982, 'gini = 0.0\nsamples = 481\nvalue = [0, 782]\nclass = b'),
 Text(704.8421052631579, 543.5999999999999, 'EBE <= 0.55\ngini = 0.389\nsamples = 354\nvalue = [144, 401]\nclass = b'),
 Text(587.3684210526316, 181.19999999999982, 'gini = 0.272\nsamples = 188\nvalue = [48, 248]\nclass = b'),
 Text(822.3157894736842, 181.19999999999982, 'gini = 0.474\nsamples = 166\nvalue = [96, 153]\nclass = b'),
 Text(1409.6842105263158, 906.0, 'SO_2 <= 5.5\ngini = 0.025\nsamples = 2151\nvalue = [43, 3338]\nclass = b'),
 Text(1174.7368421052631, 543.5999999999999, 'PM10 <= 10.5\ngini = 0.022\nsamples = 2140\nvalue = [38, 3328]\nclass = b'),
 Text(1057.2631578947369, 181.19999999999982, 'gini = 0.002\nsamples = 701\nvalue = [1, 1082]\nclass = b'),
 Text(1292.2105263157894, 181.19999999999982, 'gini = 0.032\nsamples = 1439\nvalue = [37, 2246]\nclass = b'),
 Text(1644.6315789473683, 543.5999999999999, 'CO <= 0.35\ngini = 0.444\nsamples = 11\nvalue = [5, 10]\nclass = b'),
 Text(1527.157894736842, 181.19999999999982, 'gini = 0.494\nsamples = 6\nvalue = [5, 4]\nclass = a'),
 Text(1762.1052631578948, 181.19999999999982, 'gini = 0.0\nsamples = 5\nvalue = [0, 6]\nclass = b'),
 Text(1879.578947368421, 1268.4, 'CO <= 0.35\ngini = 0.051\nsamples = 167\nvalue = [260, 7]\nclass = a'),
 Text(1762.1052631578948, 906.0, 'gini = 0.0\nsamples = 156\nvalue = [255, 0]\nclass = a'),
 Text(1997.0526315789473, 906.0, 'NMHC <= 0.215\ngini = 0.486\nsamples = 11\nvalue = [5, 7]\nclass = b'),
 Text(1879.578947368421, 543.5999999999999, 'gini = 0.444\nsamples = 6\nvalue = [2, 4]\nclass = b'),
 Text(2114.5263157894738, 543.5999999999999, 'gini = 0.5\nsamples = 5\nvalue = [3, 3]\nclass = a'),
 Text(3524.2105263157896, 1630.8000000000002, 'SO_2 <= 8.5\ngini = 0.465\nsamples = 1666\nvalue = [1661, 965]\nclass = a'),
 Text(3054.315789473684, 1268.4, 'O_3 <= 14.5\ngini = 0.494\nsamples = 1035\nvalue = [734, 921]\nclass = b'),
 Text(2584.4210526315787, 906.0, 'PM10 <= 36.5\ngini = 0.275\nsamples = 413\nvalue = [103, 523]\nclass = b'),
 Text(2349.4736842105262, 543.5999999999999, 'PM10 <= 28.5\ngini = 0.117\nsamples = 341\nvalue = [32, 482]\nclass = b'),
 Text(2232.0, 181.19999999999982, 'gini = 0.072\nsamples = 283\nvalue = [16, 413]\nclass = b'),
 Text(2466.9473684210525, 181.19999999999982, 'gini = 0.306\nsamples = 58\nvalue = [16, 69]\nclass = b'),
 Text(2819.3684210526317, 543.5999999999999, 'TOL <= 4.35\ngini = 0.464\nsamples = 72\nvalue = [71, 41]\ncla

```
ss = a'),
 Text(2701.8947368421054, 181.19999999999982, 'gini = 0.0\nsamples = 16\nvalue = [0, 20]\nclass = b'),
 Text(2936.842105263158, 181.19999999999982, 'gini = 0.352\nsamples = 56\nvalue = [71, 21]\nclass = a'),
 Text(3524.2105263157896, 906.0, 'NMHC <= 0.245\ngini = 0.474\nsamples = 622\nvalue = [631, 398]\nclass =
a'),
 Text(3289.2631578947367, 543.5999999999999, 'TCH <= 1.205\ngini = 0.333\nsamples = 447\nvalue = [587, 157]
\nclass = a'),
 Text(3171.7894736842104, 181.19999999999982, 'gini = 0.153\nsamples = 13\nvalue = [2, 22]\nclass = b'),
 Text(3406.7368421052633, 181.19999999999982, 'gini = 0.305\nsamples = 434\nvalue = [585, 135]\nclass = a'),
 Text(3759.157894736842, 543.5999999999999, 'TOL <= 5.65\ngini = 0.261\nsamples = 175\nvalue = [44, 241]\ncl
ass = b'),
 Text(3641.684210526316, 181.19999999999982, 'gini = 0.039\nsamples = 120\nvalue = [4, 199]\nclass = b'),
 Text(3876.6315789473683, 181.19999999999982, 'gini = 0.5\nsamples = 55\nvalue = [40, 42]\nclass = b'),
 Text(3994.1052631578946, 1268.4, 'CO <= 0.45\ngini = 0.087\nsamples = 631\nvalue = [927, 44]\nclass = a'),
 Text(3876.6315789473683, 906.0, 'gini = 0.0\nsamples = 349\nvalue = [535, 0]\nclass = a'),
 Text(4111.578947368421, 906.0, 'NMHC <= 0.195\ngini = 0.181\nsamples = 282\nvalue = [392, 44]\nclass = a'),
 Text(3994.1052631578946, 543.5999999999999, 'gini = 0.0\nsamples = 10\nvalue = [0, 16]\nclass = b'),
 Text(4229.0526315789475, 543.5999999999999, 'EBE <= 1.35\ngini = 0.124\nsamples = 272\nvalue = [392, 28]\nc
lass = a'),
 Text(4111.578947368421, 181.19999999999982, 'gini = 0.453\nsamples = 47\nvalue = [47, 25]\nclass = a'),
 Text(4346.526315789473, 181.19999999999982, 'gini = 0.017\nsamples = 225\nvalue = [345, 3]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression0.6254728538463179

Ridge Regression0.623137940101393

Lasso Regression:0.3732245099621311

ElasticNet Regression:0.45947325040594844

Logistic Regression:0.9293697325027482

Random Forest:0.9611309982995597

Random Forest is suitable for this dataset