Importing Libraries

In [1]:
```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
```

Importing Datasets

In [2]:
```
1 df=pd.read_csv("madrid_2011")
2 df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	28079004
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	28079011
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	28079056
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	28079057
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	28079058
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	28079059
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	28079060

209928 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: 16460 entries, 1 to 209910
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16460 non-null  object 
 1   BEN      16460 non-null  float64
 2   CO       16460 non-null  float64
 3   EBE      16460 non-null  float64
 4   NMHC     16460 non-null  float64
 5   NO       16460 non-null  float64
 6   NO_2     16460 non-null  float64
 7   O_3      16460 non-null  float64
 8   PM10     16460 non-null  float64
 9   PM25     16460 non-null  float64
 10  SO_2     16460 non-null  float64
 11  TCH      16460 non-null  float64
 12  TOL      16460 non-null  float64
 13  station  16460 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.9+ MB
```

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

Out[6]:

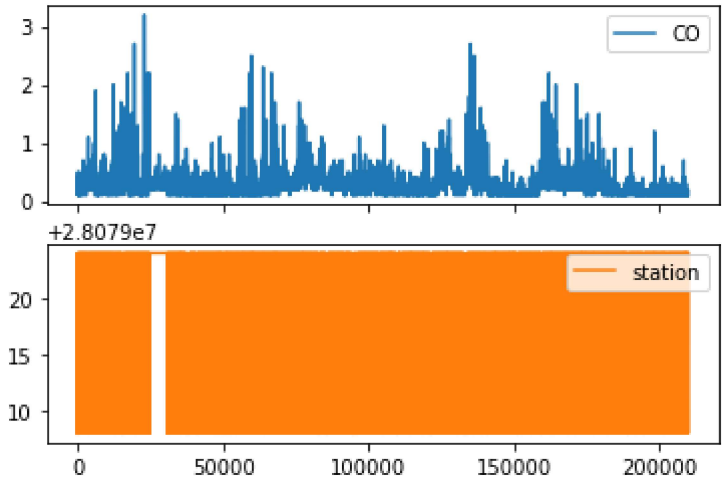
	CO	station
1	0.4	28079008
6	0.3	28079024
25	0.3	28079008
30	0.4	28079024
49	0.2	28079008
...	...	...
209862	0.1	28079024
209881	0.1	28079008
209886	0.1	28079024
209905	0.1	28079008
209910	0.1	28079024

16460 rows × 2 columns

Line chart

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

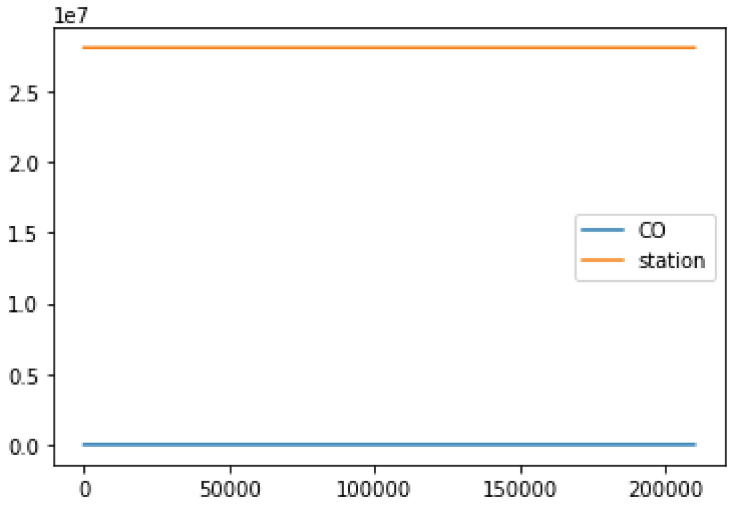
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

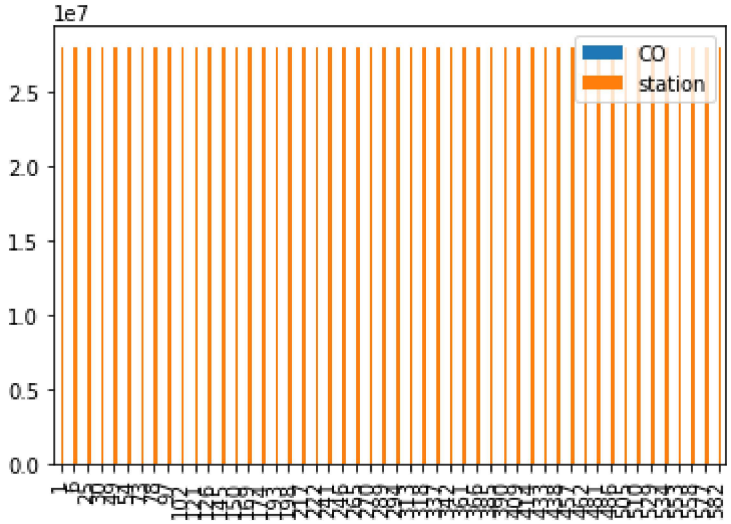


Bar chart

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

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

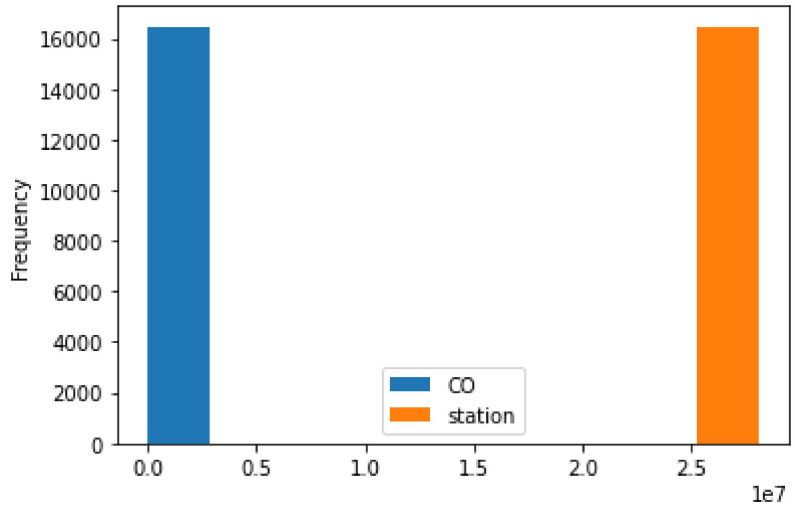
Out[10]: <AxesSubplot:>



Histogram

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

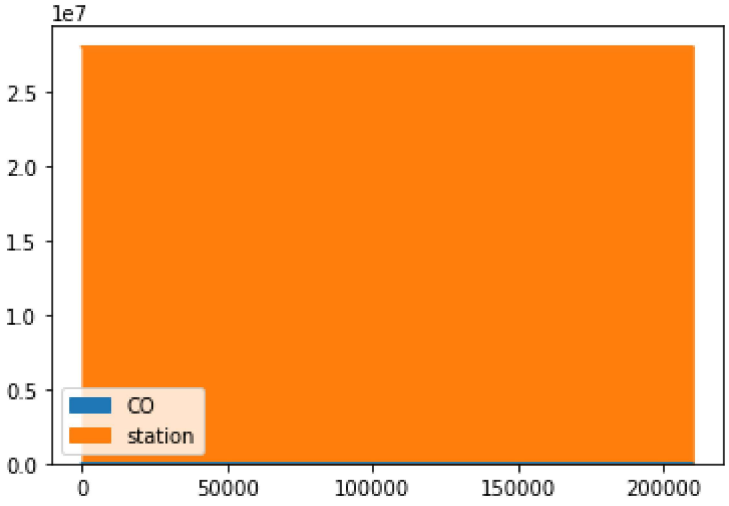
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

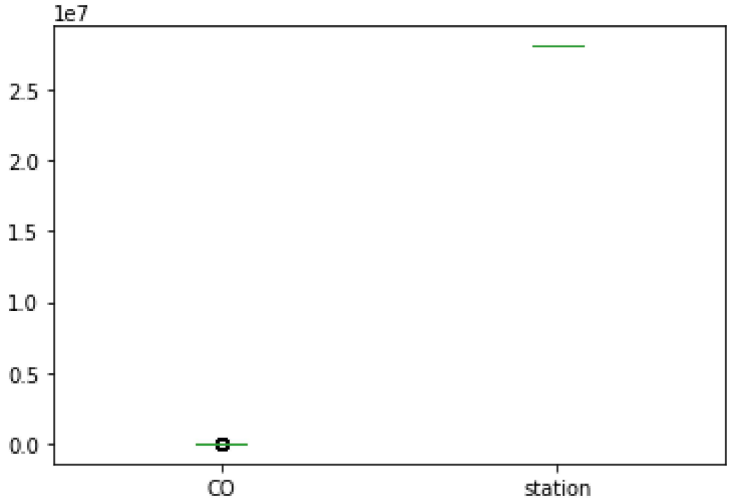
Out[12]: <AxesSubplot:>



Box chart

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

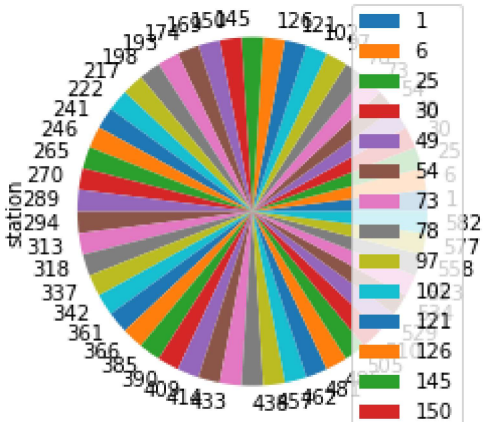
Out[13]: <AxesSubplot:>



Pie chart

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

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

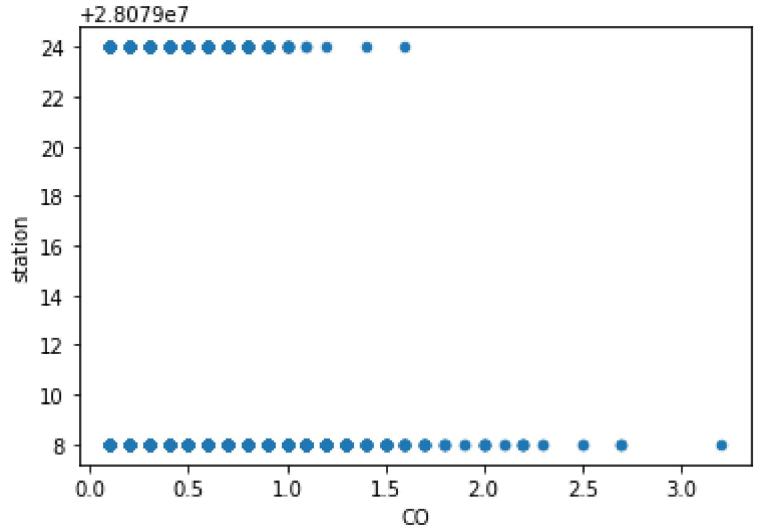

station
145
126
121
102
97
78
73
54
49
30
25
6
1
150
169
174
193
198
217
222
241
246
265
270
289
294
313
318
337
342
361
366
385
390
409
414
433
438
457
462
481
486
505
510
529
534
553
558
577

582

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: 16460 entries, 1 to 209910
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     16460 non-null  object
 1   BEN      16460 non-null  float64
 2   CO       16460 non-null  float64
 3   EBE      16460 non-null  float64
 4   NMHC     16460 non-null  float64
 5   NO       16460 non-null  float64
 6   NO_2     16460 non-null  float64
 7   O_3      16460 non-null  float64
 8   PM10     16460 non-null  float64
 9   PM25     16460 non-null  float64
 10  SO_2     16460 non-null  float64
 11  TCH      16460 non-null  float64
 12  TOL      16460 non-null  float64
 13  station  16460 non-null  int64
```

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

Out[17]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25
count	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000
mean	0.900680	0.277758	1.471871	0.167043	23.671810	44.583961	41.580377	24.670109	12.418226
std	0.768892	0.206143	1.051004	0.075068	44.362859	31.569185	28.113385	18.758383	7.783435
min	0.100000	0.100000	0.200000	0.010000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	0.500000	0.200000	0.800000	0.120000	2.000000	19.000000	17.000000	13.000000	7.000000
50%	0.700000	0.200000	1.200000	0.160000	7.000000	40.000000	39.000000	20.000000	11.000000
75%	1.100000	0.300000	1.700000	0.200000	25.000000	63.000000	61.000000	31.000000	16.000000
max	9.500000	3.200000	12.800000	0.840000	615.000000	289.000000	154.000000	281.000000	88.000000

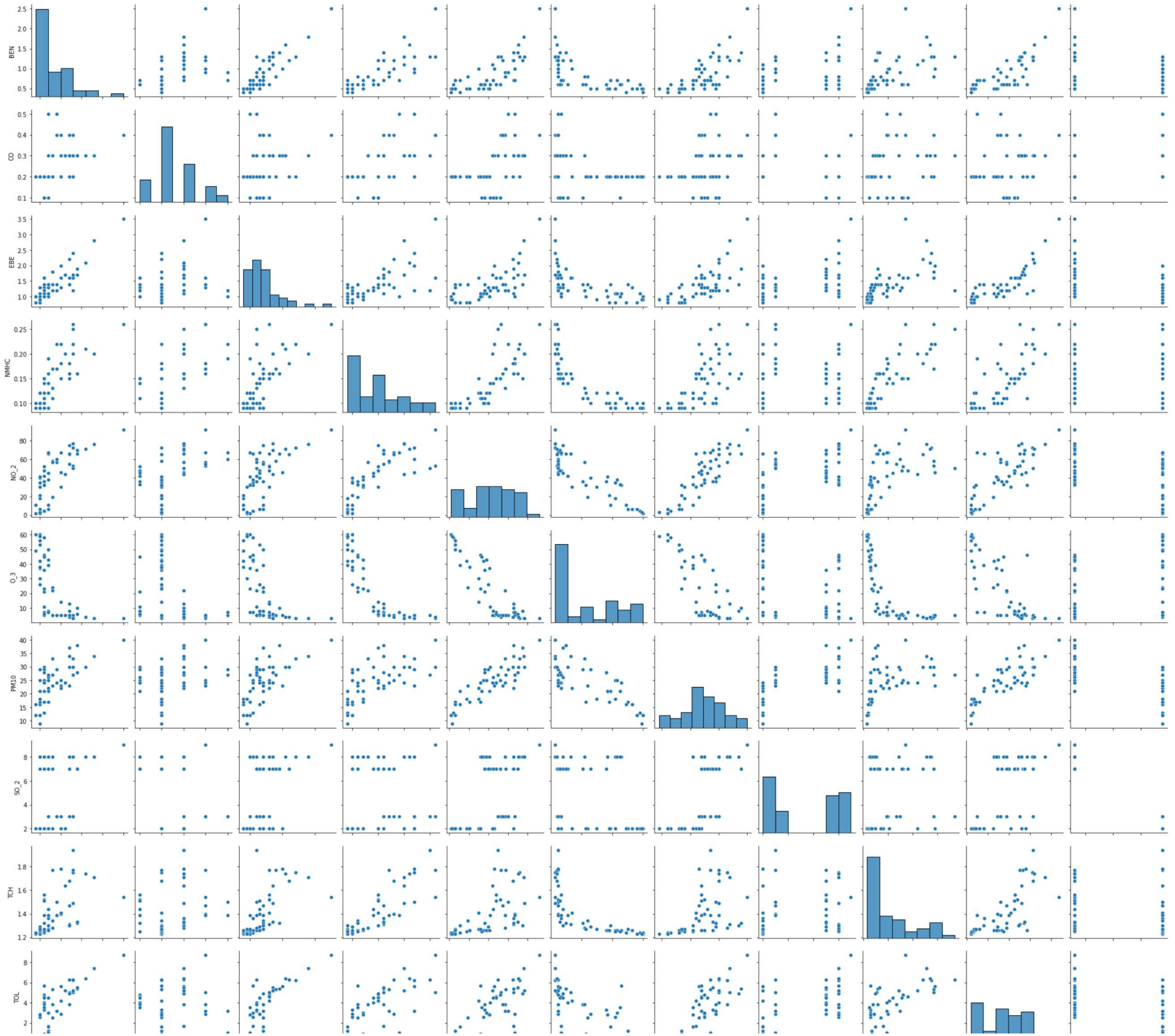
In [19]:

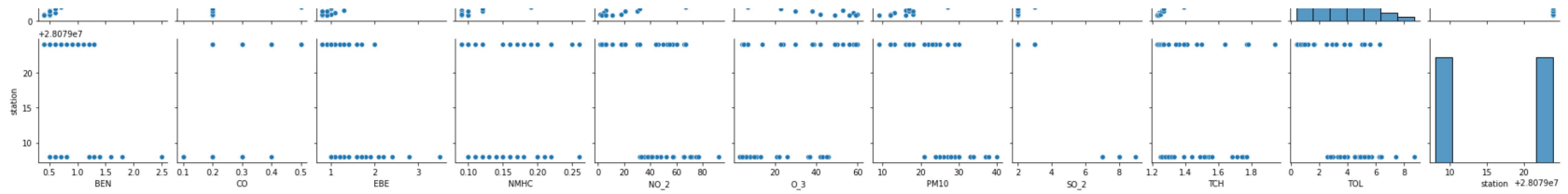
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [20]:
```
1 sns.pairplot(df1[0:50])
```

Out[20]:
```
<seaborn.axisgrid.PairGrid at 0x20c826a6c40>
```

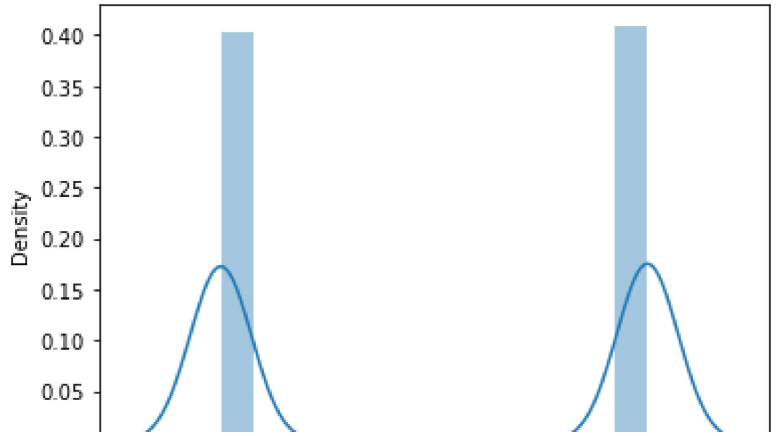

BEN
CO
EBE
NMHC
NO_2
O_3
PM10
SO_2
TCH
TOL



In [21]:
```
1 sns.distplot(df1['station'])
```

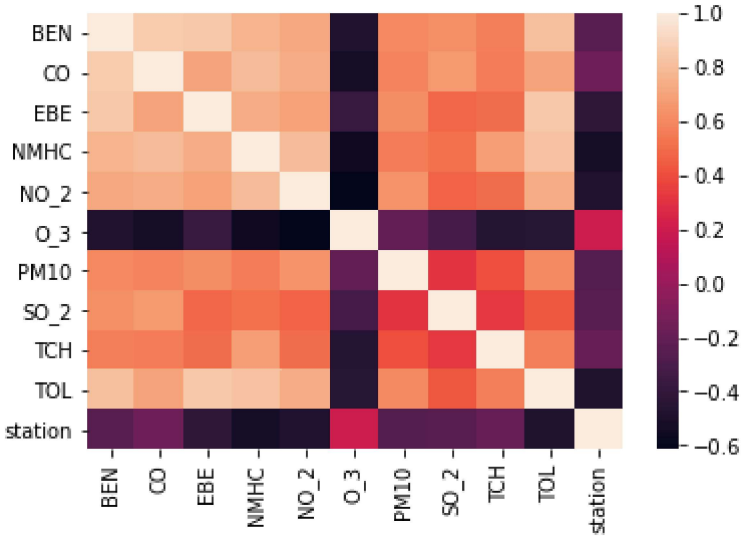
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[21]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [22]:
```
1 sns.heatmap(df1.corr())
```

Out[22]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [24]:
```
1 x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
2        'PM10', 'SO_2', 'TCH', 'TOL']]
3 y=df['station']
```

In [25]:
```
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 [26]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[26]:
```
LinearRegression()
```

In [27]:
```
lr.intercept_
```

Out[27]:
```
28079017.676045153
```

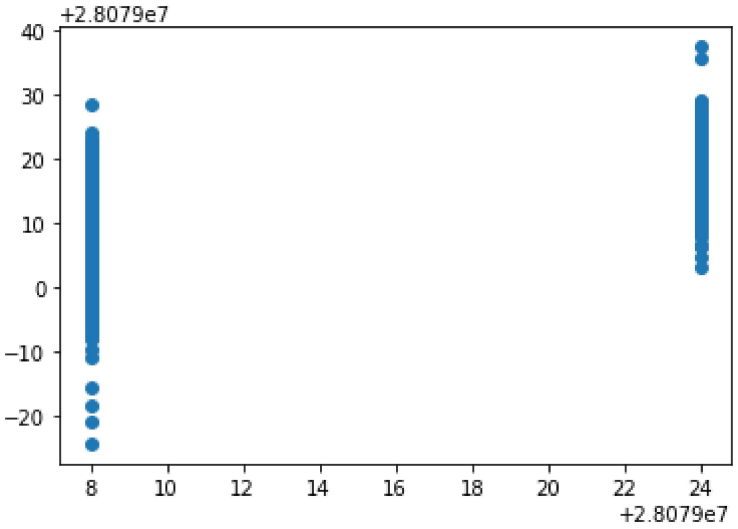
In [28]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[28]:

	Co-efficient
BEN	4.023567
CO	31.915215
EBE	-2.102587
NMHC	-94.345524
NO_2	-0.083829
O_3	-0.014059
PM10	-0.000955
SO_2	-0.513715
TCH	10.053636
TOL	-0.449215

In [29]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[29]: <matplotlib.collections.PathCollection at 0x20c88d7d9d0>



ACCURACY

In [30]:
```
lr.score(x_test,y_test)
```

Out[30]: 0.6231575999363752

In [31]:
```
lr.score(x_train,y_train)
```

Out[31]: 0.6211760778574713

Ridge and Lasso

In [32]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [33]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[33]: Ridge(alpha=10)

Accuracy(Ridge)

In [34]:
```
rr.score(x_test,y_test)
```

Out[34]: 0.5880696376912109

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

Out[35]: 0.5881170255967527

In [36]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[36]: Lasso(alpha=10)

Accuracy(Lasso)

In [37]:
```
la.score(x_train,y_train)
```

Out[37]: 0.23374165343580144

In [38]:
```
la.score(x_test,y_test)
```

Out[38]: 0.23366892325619637

In [39]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[39]: ElasticNet()

In [40]:
```
1 en.coef_
```

Out[40]:
```
array([ 0.63276703,  0.        , -0.        , -0.        , -0.1324441 ,
       -0.05196152,  0.07314671, -0.        ,  0.        , -0.68063259])
```

In [41]:
```
1 en.intercept_
```

Out[41]:
```
28079024.165214125
```

In [42]:
```
1 prediction=en.predict(x_test)
```

In [43]:
```
1 en.score(x_test,y_test)
```

Out[43]:
```
0.329971692163077
```

Evaluation Metrics

In [44]:
```
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)))
```

```
5.861446828255864
42.87977874177515
6.5482653231046735
```

Logistic Regression

In [45]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [46]:
```
1 feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
2        'PM10', 'SO_2', 'TCH', 'TOL']]
3 target_vector=df[ 'station']
```

In [47]:
```
1 feature_matrix.shape
```

Out[47]: (16460, 10)

In [48]:
```
1 target_vector.shape
```

Out[48]: (16460,)

In [49]:
```
1 from sklearn.preprocessing import StandardScaler
```

In [50]:
```
1 fs=StandardScaler().fit_transform(feature_matrix)
```

In [51]:
```
1 logr=LogisticRegression(max_iter=10000)
2 logr.fit(fs,target_vector)
```

Out[51]: LogisticRegression(max_iter=10000)

In [56]:
```
1 observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [57]:
```
1 prediction=logr.predict(observation)
2 print(prediction)
```

```
[28079008]
```

In [58]:
```
1 logr.classes_
```

Out[58]: array([28079008, 28079024], dtype=int64)

In [59]:
```
1 logr.score(fs,target_vector)
```

Out[59]: 0.9237545565006076

In [60]:
```
1 logr.predict_proba(observation)[0][0]
```

Out[60]: 0.9999999999999966

In [61]:
```
1 logr.predict_proba(observation)
```

Out[61]: array([[1.00000000e+00, 3.47334507e-15]])

Random Forest

In [62]:
```
1 from sklearn.ensemble import RandomForestClassifier
```

In [63]:
```
1 rfc=RandomForestClassifier()
2 rfc.fit(x_train,y_train)
```

Out[63]:
```
RandomForestClassifier()
```

In [64]:
```
1 parameters={'max_depth':[1,2,3,4,5],
2             'min_samples_leaf':[5,10,15,20,25],
3             'n_estimators':[10,20,30,40,50]
4 }
```

In [65]:
```
1 from sklearn.model_selection import GridSearchCV
2 grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
3 grid_search.fit(x_train,y_train)
```

Out[65]:
```
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 [66]:
```
1 grid_search.best_score_
```

Out[66]:
```
0.9327373719840306
```

In [67]:
```
1 rfc_best=grid_search.best_estimator_
```

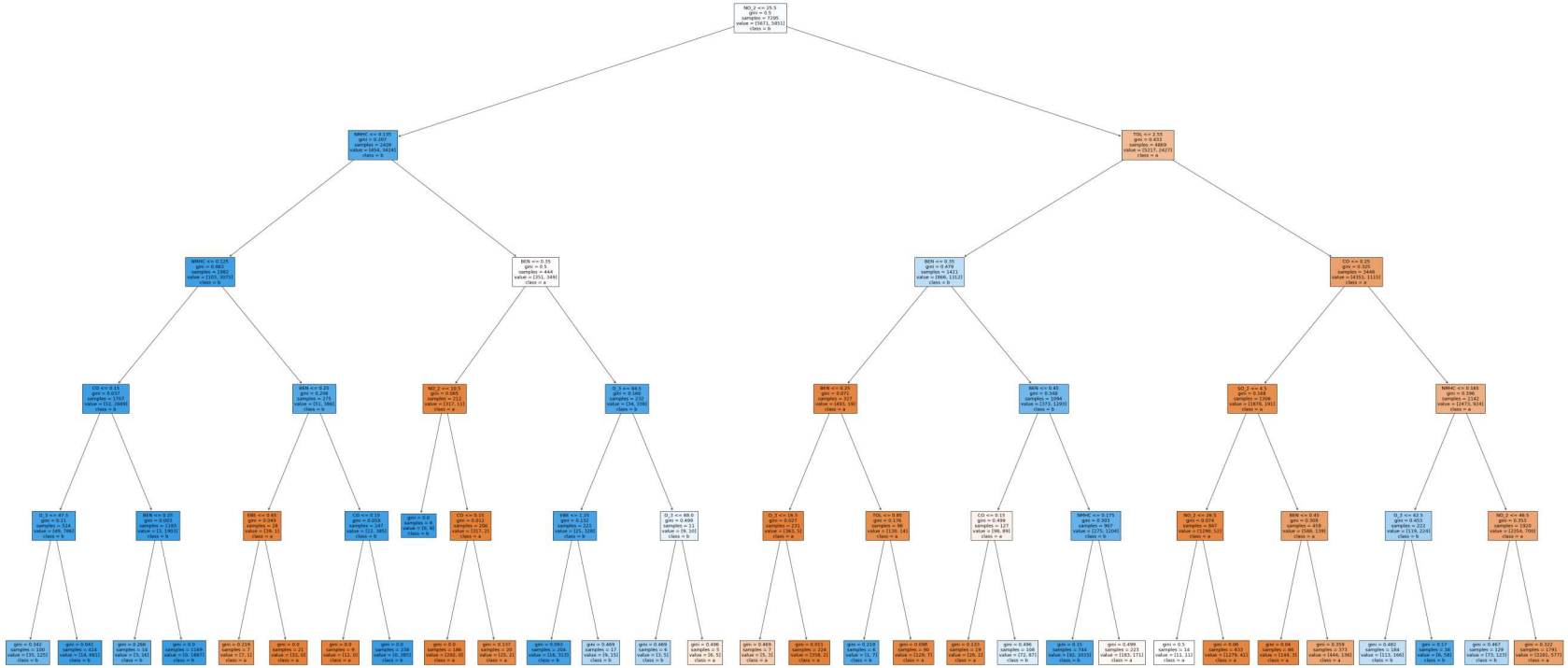
In [68]:

```
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[68]:
```
[Text(2166.9, 1993.2, 'NO_2 <= 25.5\ngini = 0.5\nsamples = 7295\nvalue = [5671, 5851]\nclass = b'),
 Text(1060.2, 1630.8000000000002, 'NMHC <= 0.135\ngini = 0.207\nsamples = 2426\nvalue = [454, 3424]\nclass =
b'),
 Text(595.2, 1268.4, 'NMHC <= 0.125\ngini = 0.063\nsamples = 1982\nvalue = [103, 3075]\nclass = b'),
 Text(297.6, 906.0, 'CO <= 0.15\ngini = 0.037\nsamples = 1707\nvalue = [52, 2689]\nclass = b'),
 Text(148.8, 543.5999999999999, 'O_3 <= 47.5\ngini = 0.11\nsamples = 524\nvalue = [49, 786]\nclass = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.342\nsamples = 100\nvalue = [35, 125]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.041\nsamples = 424\nvalue = [14, 661]\nclass = b'),
 Text(446.40000000000003, 543.5999999999999, 'BEN <= 0.25\ngini = 0.003\nsamples = 1183\nvalue = [3, 1903]\n
class = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.266\nsamples = 14\nvalue = [3, 16]\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.0\nsamples = 1169\nvalue = [0, 1887]\nclass = b'),
 Text(892.8000000000001, 906.0, 'BEN <= 0.25\ngini = 0.206\nsamples = 275\nvalue = [51, 386]\nclass = b'),
 Text(744.0, 543.5999999999999, 'EBE <= 0.65\ngini = 0.049\nsamples = 28\nvalue = [39, 1]\nclass = a'),
 Text(669.6, 181.19999999999982, 'gini = 0.219\nsamples = 7\nvalue = [7, 1]\nclass = a'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.0\nsamples = 21\nvalue = [32, 0]\nclass = a'),
 Text(1041.6000000000001, 543.5999999999999, 'CO <= 0.15\ngini = 0.059\nsamples = 247\nvalue = [12, 385]\ncl
ass = b'),
 Text(967.2, 181.19999999999982, 'gini = 0.0\nsamples = 9\nvalue = [12, 0]\nclass = a'),
 Text(1116.0, 181.19999999999982, 'gini = 0.0\nsamples = 238\nvalue = [0, 385]\nclass = b'),
 Text(1525.2, 1268.4, 'BEN <= 0.35\ngini = 0.5\nsamples = 444\nvalue = [351, 349]\nclass = a'),
 Text(1264.8000000000002, 906.0, 'NO_2 <= 10.5\ngini = 0.065\nsamples = 212\nvalue = [317, 11]\nclass = a'),
 Text(1190.4, 543.5999999999999, 'gini = 0.0\nsamples = 6\nvalue = [0, 9]\nclass = b'),
 Text(1339.2, 543.5999999999999, 'CO <= 0.15\ngini = 0.012\nsamples = 206\nvalue = [317, 2]\nclass = a'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.0\nsamples = 186\nvalue = [292, 0]\nclass = a'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.137\nsamples = 20\nvalue = [25, 2]\nclass = a'),
 Text(1785.6000000000001, 906.0, 'O_3 <= 84.5\ngini = 0.166\nsamples = 232\nvalue = [34, 338]\nclass = b'),
 Text(1636.8000000000002, 543.5999999999999, 'EBE <= 1.25\ngini = 0.132\nsamples = 221\nvalue = [25, 328]\nc
lass = b'),
 Text(1562.4, 181.19999999999982, 'gini = 0.093\nsamples = 204\nvalue = [16, 313]\nclass = b'),
 Text(1711.2, 181.19999999999982, 'gini = 0.469\nsamples = 17\nvalue = [9, 15]\nclass = b'),
 Text(1934.4, 543.5999999999999, 'O_3 <= 88.0\ngini = 0.499\nsamples = 11\nvalue = [9, 10]\nclass = b'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.469\nsamples = 6\nvalue = [3, 5]\nclass = b'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.496\nsamples = 5\nvalue = [6, 5]\nclass = a'),
 Text(3273.6000000000004, 1630.8000000000002, 'TOL <= 2.55\ngini = 0.433\nsamples = 4869\nvalue = [5217, 242
7]\nclass = a'),
 Text(2678.4, 1268.4, 'BEN <= 0.35\ngini = 0.479\nsamples = 1421\nvalue = [866, 1312]\nclass = b'),
 Text(2380.8, 906.0, 'BEN <= 0.25\ngini = 0.071\nsamples = 327\nvalue = [493, 19]\nclass = a'),
 Text(2232.0, 543.5999999999999, 'O_3 <= 16.5\ngini = 0.027\nsamples = 231\nvalue = [363, 5]\nclass = a'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.469\nsamples = 7\nvalue = [5, 3]\nclass = a'),
 Text(2306.4, 181.19999999999982, 'gini = 0.011\nsamples = 224\nvalue = [358, 2]\nclass = a'),
 Text(2529.6000000000004, 543.5999999999999, 'TOL <= 0.85\ngini = 0.176\nsamples = 96\nvalue = [130, 14]\ncl
ass = a'),
```

```
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.219\nsamples = 6\nvalue = [1, 7]\nclass = b'),
 Text(2604.0, 181.19999999999982, 'gini = 0.098\nsamples = 90\nvalue = [129, 7]\nclass = a'),
 Text(2976.0, 906.0, 'BEN <= 0.45\ngini = 0.348\nsamples = 1094\nvalue = [373, 1293]\nclass = b'),
 Text(2827.2000000000003, 543.5999999999999, 'CO <= 0.15\ngini = 0.499\nsamples = 127\nvalue = [98, 89]\ncla
ss = a'),
 Text(2752.8, 181.19999999999982, 'gini = 0.133\nsamples = 19\nvalue = [26, 2]\nclass = a'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.496\nsamples = 108\nvalue = [72, 87]\nclass = b'),
 Text(3124.8, 543.5999999999999, 'NMHC <= 0.175\ngini = 0.303\nsamples = 967\nvalue = [275, 1204]\nclass =
b'),
 Text(3050.4, 181.19999999999982, 'gini = 0.15\nsamples = 744\nvalue = [92, 1033]\nclass = b'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.499\nsamples = 223\nvalue = [183, 171]\nclass = a'),
 Text(3868.8, 1268.4, 'CO <= 0.25\ngini = 0.325\nsamples = 3448\nvalue = [4351, 1115]\nclass = a'),
 Text(3571.2000000000003, 906.0, 'SO_2 <= 4.5\ngini = 0.168\nsamples = 1306\nvalue = [1878, 191]\nclass =
a'),
 Text(3422.4, 543.5999999999999, 'NO_2 <= 26.5\ngini = 0.074\nsamples = 847\nvalue = [1290, 52]\nclass =
a'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.5\nsamples = 14\nvalue = [11, 11]\nclass = a'),
 Text(3496.8, 181.19999999999982, 'gini = 0.06\nsamples = 833\nvalue = [1279, 41]\nclass = a'),
 Text(3720.0000000000005, 543.5999999999999, 'BEN <= 0.45\ngini = 0.309\nsamples = 459\nvalue = [588, 139]\n
class = a'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.04\nsamples = 86\nvalue = [144, 3]\nclass = a'),
 Text(3794.4, 181.19999999999982, 'gini = 0.359\nsamples = 373\nvalue = [444, 136]\nclass = a'),
 Text(4166.400000000001, 906.0, 'NMHC <= 0.165\ngini = 0.396\nsamples = 2142\nvalue = [2473, 924]\nclass =
a'),
 Text(4017.6000000000004, 543.5999999999999, 'O_3 <= 42.5\ngini = 0.453\nsamples = 222\nvalue = [119, 224]\n
class = b'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.482\nsamples = 184\nvalue = [113, 166]\nclass = b'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.17\nsamples = 38\nvalue = [6, 58]\nclass = b'),
 Text(4315.200000000001, 543.5999999999999, 'NO_2 <= 46.5\ngini = 0.353\nsamples = 1920\nvalue = [2354, 700]
\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.467\nsamples = 129\nvalue = [73, 123]\nclass = b'),
 Text(4389.6, 181.19999999999982, 'gini = 0.322\nsamples = 1791\nvalue = [2281, 577]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression0.6211760778574713

Ridge Regression:0.5881170255967527

Lasso Regression:0.23374165343580144

ElasticNet Regression:0.329971692163077

Logistic Regression:0.9237545565006076

Random Forest:0.9327373719840306

Random Forest is suitable for this dataset