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(r"C:\Users\user\Downloads\madrid_2013.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	01-11-2013 01:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
1	01-11-2013 01:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	28079008
2	01-11-2013 01:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	28079011
3	01-11-2013 01:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	28079016
4	01-11-2013 01:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	01-03-2013 00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	28079056
209876	01-03-2013 00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	28079057
209877	01-03-2013 00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	28079058

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
209878	01-03-2013 00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	28079059
209879	01-03-2013 00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	28079060

209880 rows × 14 columns

Data Cleaning and Data Preprocessing

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

In [4]:
```
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]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

Out[6]:

	CO	station
0	0.6	28079004

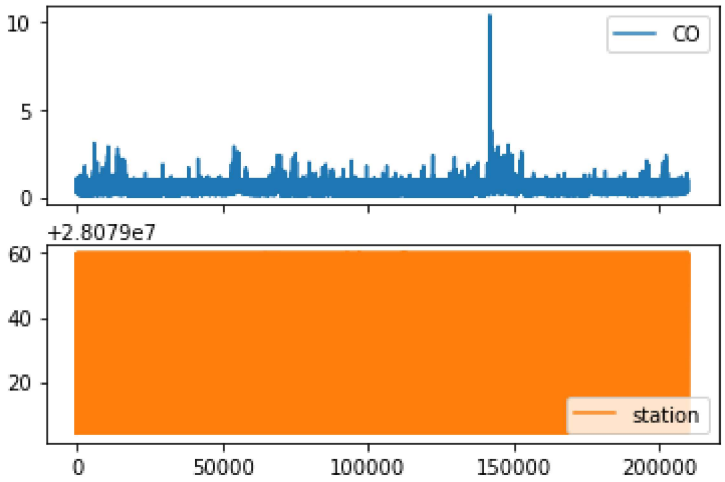
	CO	station
1	0.5	28079008
2	1.0	28079011
3	0.5	28079016
4	1.0	28079017
...	...	...
209875	0.4	28079056
209876	0.4	28079057
209877	1.0	28079058
209878	1.0	28079059
209879	1.0	28079060

209880 rows × 2 columns

Line chart

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

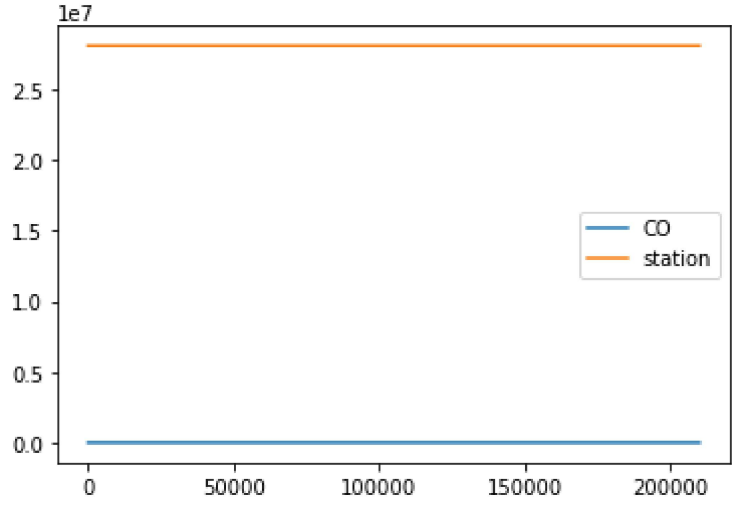
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

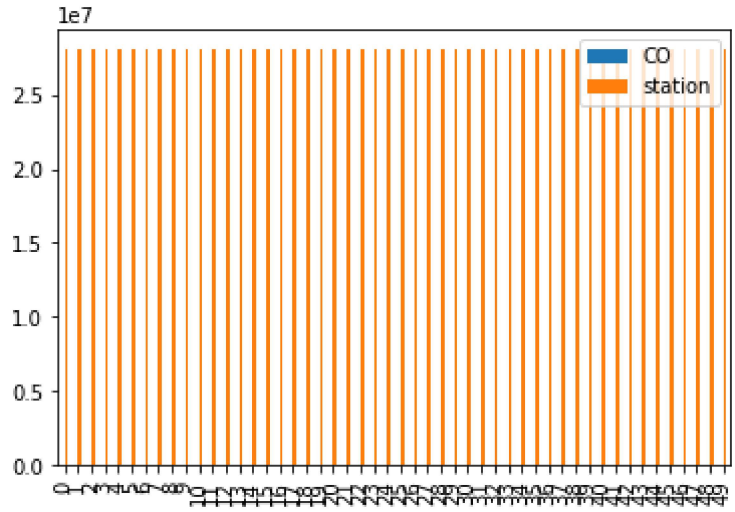


Bar chart

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

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

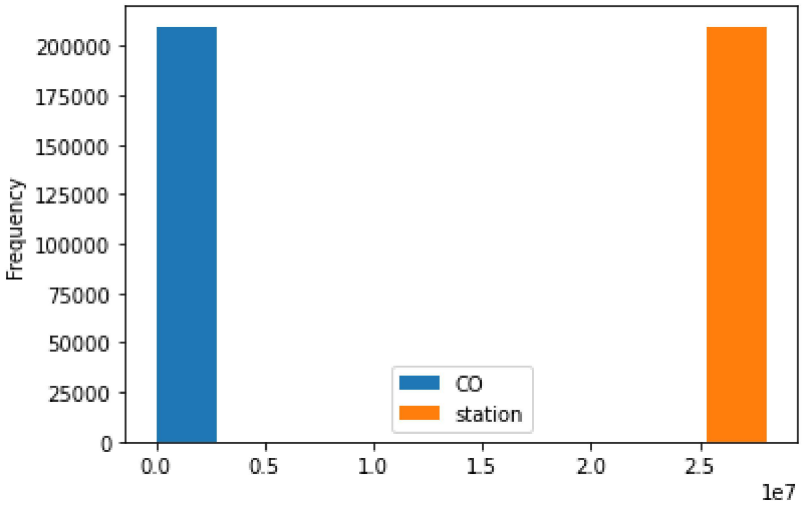
Out[10]: <AxesSubplot:>



Histogram

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

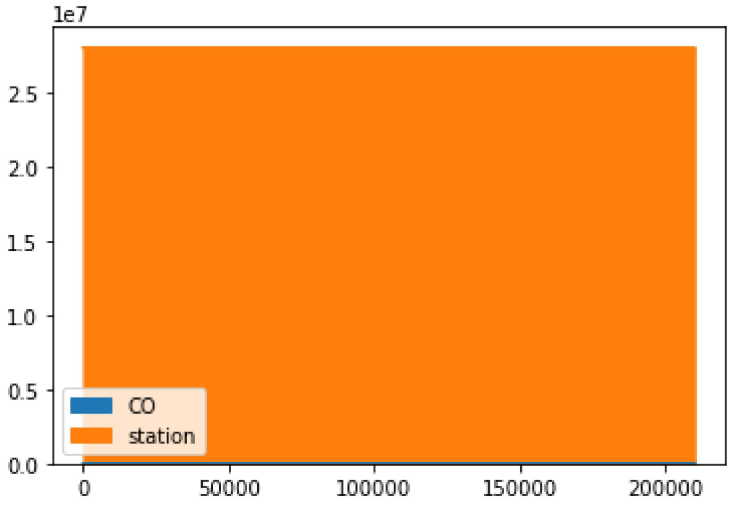
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

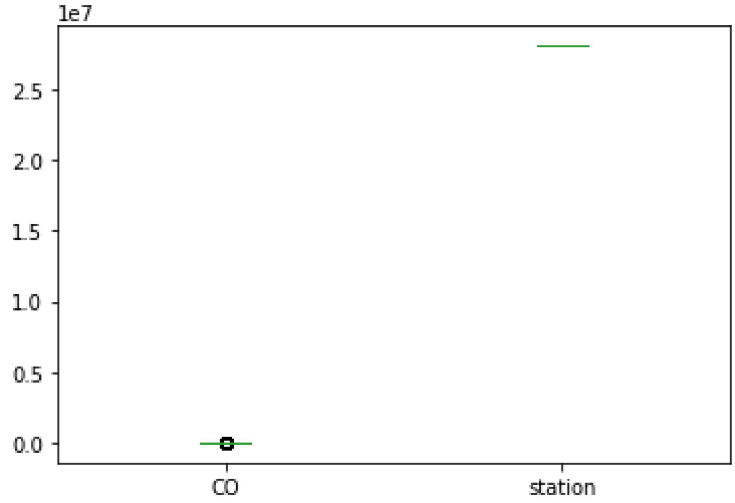
Out[12]: <AxesSubplot:>



Box chart

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

Out[13]: <AxesSubplot:>



Pie chart

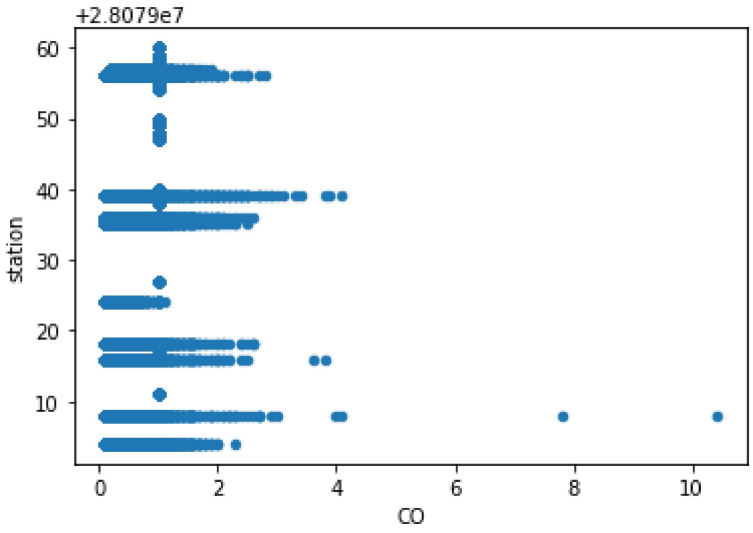
In [14]:
```
b.plot.pie(y='station' )
```

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

Scatter chart

In [15]:
```
data.plot.scatter(x='CO' ,y='station')
```

Out[15]:
```
<AxesSubplot:xlabel='CO', ylabel='station'>
```



In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [18]:
```
df.describe()
```

Out[18]:

	BEN	CO	EBE	NMHC	NO	NO_2	
count	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	20988
mean	0.931014	0.721695	0.954744	0.900223	20.101401	34.586402	2
std	0.430684	0.361528	0.301074	0.267139	44.319112	27.866588	3
min	0.100000	0.100000	0.100000	0.040000	1.000000	1.000000	

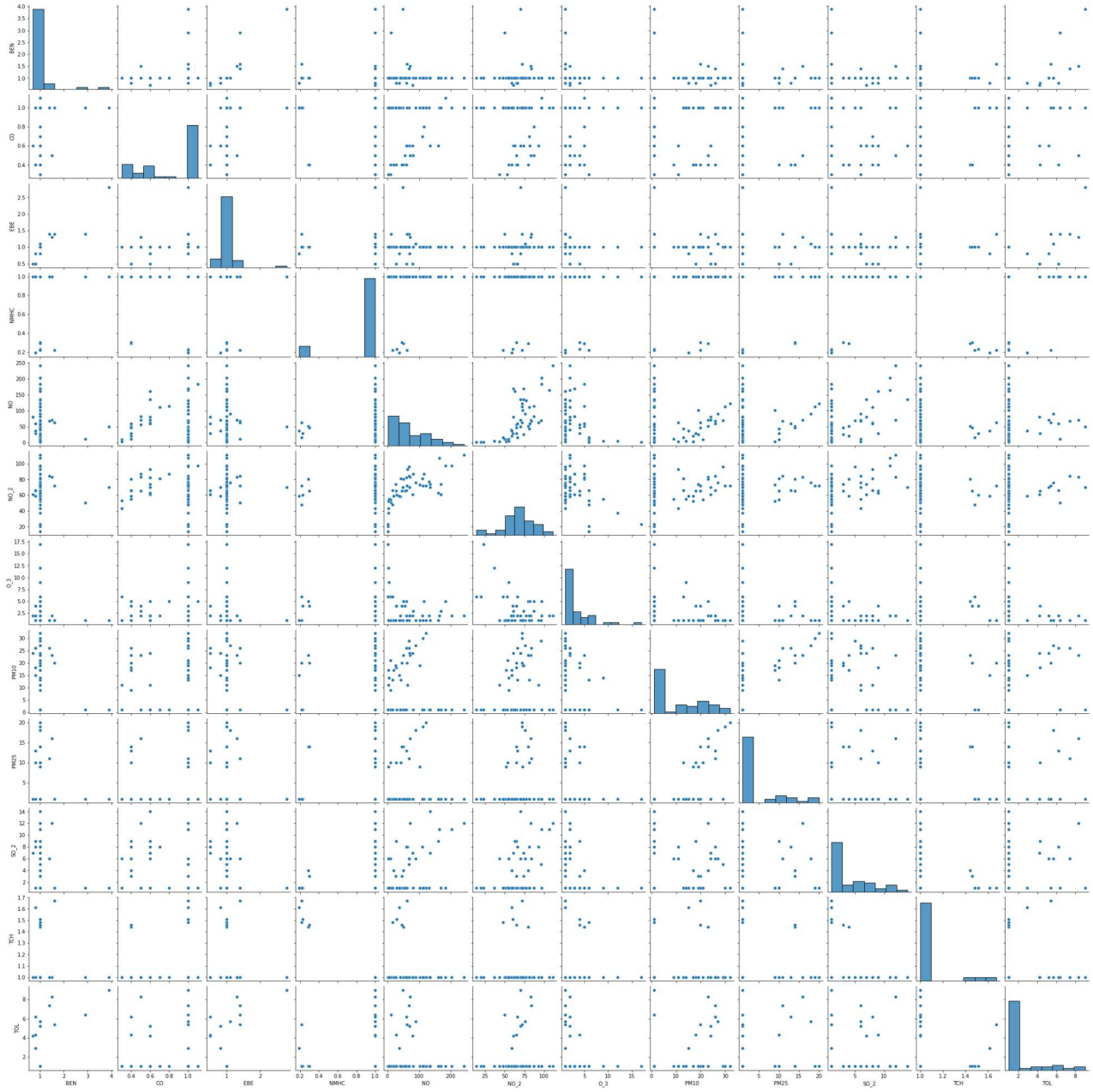
	BEN	CO	EBE	NMHC	NO	NO_2	
25%	1.000000	0.300000	1.000000	1.000000	2.000000	14.000000	
50%	1.000000	1.000000	1.000000	1.000000	5.000000	27.000000	
75%	1.000000	1.000000	1.000000	1.000000	17.000000	48.000000	5
max	12.100000	10.400000	11.800000	1.000000	1081.000000	388.000000	22

In [19]:
```
df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
```

EDA AND VISUALIZATION

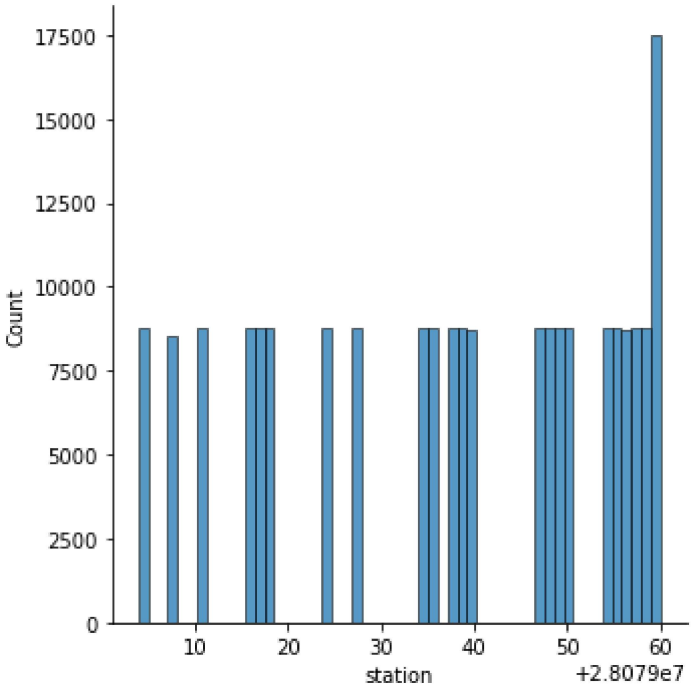
In [20]:
```
sns.pairplot(df1[0:50])
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x245151a37f0>



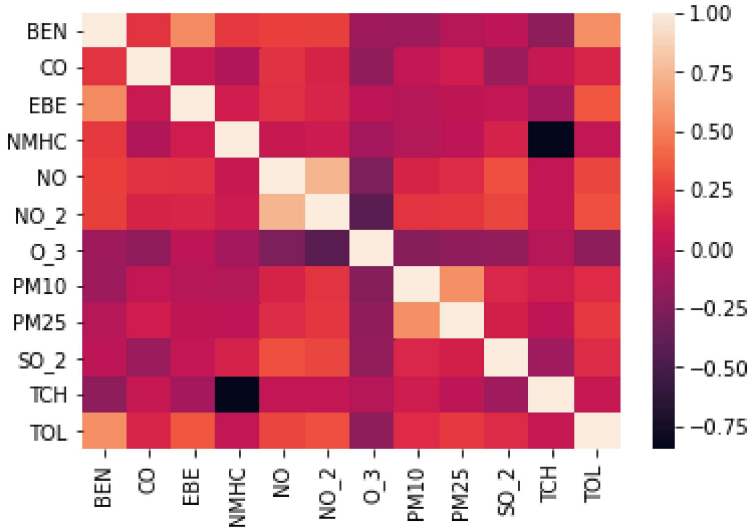
In [21]:
```
sns.displot(df['station'])
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x245161f6280>
```



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

Out[22]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [23]:
```
x=df[[ 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [24]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

Linear Regression

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

Out[25]:
```
LinearRegression()
```

In [26]:
```
lr.intercept_
```

Out[26]:
```
28078975.672733862
```

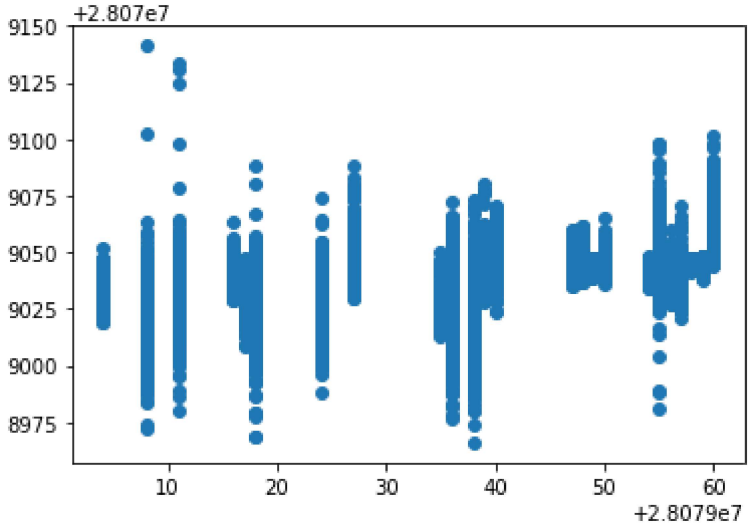
In [27]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[27]:

	Co-efficient
BEN	1.635542
CO	18.920436
EBE	10.456873
NMHC	18.327037
NO	-0.010087
NO_2	-0.039711
O_3	0.009499
PM10	0.276971
PM25	-0.374595
SO_2	-0.911238
TCH	25.866883
TOL	-3.424335

In [28]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x24521d921c0>
```



ACCURACY

In [29]:
```
lr.score(x_test,y_test)
```

Out[29]: 0.30506556835336696

In [30]:
```
lr.score(x_train,y_train)
```

Out[30]: 0.3081412318671143

Ridge and Lasso

In [31]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [32]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[32]: Ridge(alpha=10)

Accuracy(Ridge)

In [33]:
```
rr.score(x_test,y_test)
```

Out[33]: 0.30508743386847714

In [34]:
```
rr.score(x_train,y_train)
```

Out[34]: 0.30813824643119025

In [35]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[35]:
```
Lasso(alpha=10)
```

In [36]:
```
la.score(x_train,y_train)
```

Out[36]:
```
0.04463390964618652
```

Accuracy(Lasso)

In [37]:
```
la.score(x_test,y_test)
```

Out[37]:
```
0.04460958753352984
```

Elastic Net regression

In [38]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[38]:
```
ElasticNet()
```

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

Out[39]:
```
array([ 0.32903126,  2.64201627,  0.47938568,  0.        ,  0.0335253 ,
       -0.05511713, -0.01958163,  0.23580602, -0.35463623, -1.32196201,
       -0.        , -1.56274017])
```

In [40]:
```
en.intercept_
```

Out[40]:
```
28079041.14048494
```

In [41]:
```
prediction=en.predict(x_test)
```

In [42]:
```
en.score(x_test,y_test)
```

Out[42]:
```
0.16493993283490482
```

Evaluation Metrics

In [43]:
```
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)))
```

```
13.787013576913228
258.0365370666365
16.063515713150608
```

Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression
```

In [45]:
```
feature_matrix=df[[ 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

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

Out[46]:
```
(209880, 12)
```

In [47]:
```
target_vector.shape
```

Out[47]:
```
(209880,)
```

In [48]:
```
from sklearn.preprocessing import StandardScaler
```

In [49]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [50]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[50]:
```
LogisticRegression(max_iter=10000)
```

In [51]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12]]
```

In [52]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [53]:
```
logr.classes_
```

Out[53]:
```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060],
      dtype=int64)
```

In [54]:
```
logr.score(fs,target_vector)
```

Out[54]:
```
0.7180674671240709
```

In [55]:
```
logr.predict_proba(observation)[0][0]
```

Out[55]:
```
4.543759042440039e-255
```

In [56]:
```
logr.predict_proba(observation)
```

Out[56]:
```
array([[4.54375904e-255, 9.83679523e-001, 2.76618398e-209,
        3.04348606e-171, 8.45958554e-096, 7.64203284e-060,
        1.63204768e-002, 2.39850901e-120, 1.22679178e-105,
        3.54944067e-101, 5.42175008e-073, 6.53051675e-168,
        2.77411557e-136, 5.68074387e-122, 1.28228803e-127,
        2.81730461e-234, 3.10955481e-123, 5.21686422e-231,
        1.03167414e-109, 1.69201617e-172, 1.71839244e-097,
        1.30744919e-244, 5.96778314e-237, 4.25867005e-122]])
```

Random Forest

In [57]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [58]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[58]:
```
RandomForestClassifier()
```

In [59]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [60]:
```
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[60]:
```
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 [61]:
```
grid_search.best_score_
```

Out[61]:
```
0.7496460562498298
```

In [62]:
```
rfc_best=grid_search.best_estimator_
```

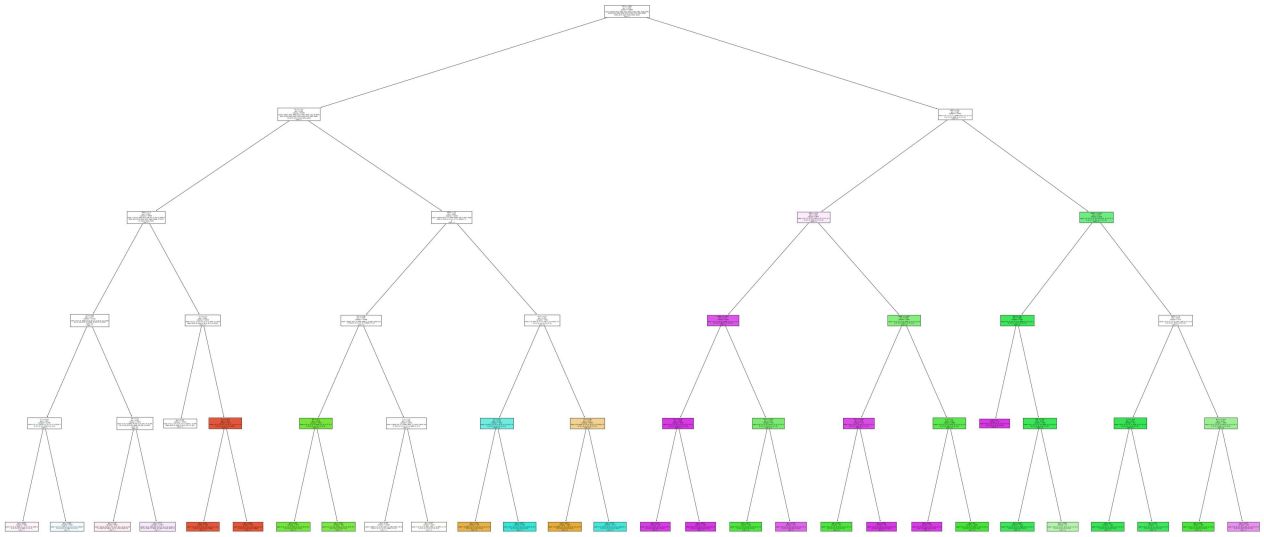
In [65]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d',
```

Out[65]:
```
[Text(2212.0714285714284, 1993.2, 'TCH <= 1.005\ngini = 0.958\nsamples = 92989\nvalue =
[6057, 6013, 6083, 6112, 6015, 6016, 6201, 6185, 6045\n6228, 6150, 6350, 6052, 6114, 612
9, 6175, 5997, 6068\n6163, 6173, 6127, 6153, 6055, 6255]\nclass = l'),
 Text(1056.2142857142856, 1630.8000000000002, 'SO_2 <= 1.5\ngini = 0.953\nsamples = 8148
4\nvalue = [6057, 6013, 6083, 6112, 6015, 6016, 133, 28, 6045\n6228, 6150, 6350, 6052, 6
114, 6129, 6175, 5997, 6068\n74, 6173, 6127, 6153, 6055, 6255]\nclass = l'),
 Text(518.1428571428571, 1268.4, 'PM10 <= 1.5\ngini = 0.924\nsamples = 49558\nvalue = [1
4, 42, 6083, 6112, 66, 52, 27, 28, 8, 3298, 8\n6350, 942, 6114, 6129, 6175, 5997, 6068,
74, 6173\n37, 6153, 6055, 6255]\nclass = l'),
 Text(318.85714285714283, 906.0, 'CO <= 0.95\ngini = 0.877\nsamples = 31320\nvalue = [1
4, 31, 6083, 6112, 66, 15, 6, 28, 8, 5, 6, 6350\n18, 31, 26, 6175, 67, 6068, 18, 6173, 1
2, 6153\n6055, 23]\nclass = l'),
 Text(159.42857142857142, 543.5999999999999, 'O_3 <= 53.5\ngini = 0.667\nsamples = 11349
\nvalue = [4, 0, 0, 6018, 0, 3, 0, 0, 4, 5, 0, 6125, 0\n0, 0, 0, 0, 0, 0, 5847, 0, 0, 0,
0]\nclass = l'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.666\nsamples = 6821\nvalue = [4,
0, 0, 3588, 0, 1, 0, 0, 4, 5, 0, 3306, 0\n0, 0, 0, 0, 0, 0, 3905, 0, 0, 0, 0]\nclass =
t'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.659\nsamples = 4528\nvalue = [0,
0, 0, 2430, 0, 2, 0, 0, 0, 0, 0, 2819, 0\n0, 0, 0, 0, 0, 0, 1942, 0, 0, 0, 0]\nclass =
l'),
 Text(478.2857142857142, 543.5999999999999, 'NO <= 2.5\ngini = 0.812\nsamples = 19971\nv
alue = [10, 31, 6083, 94, 66, 12, 6, 28, 4, 0, 6, 225\n18, 31, 26, 6175, 67, 6068, 18, 3
26, 12, 6153\n6055, 23]\nclass = p'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.742\nsamples = 8924\nvalue = [9,
31, 1673, 1, 47, 12, 6, 14, 1, 0, 6, 12, 18\n7, 4, 3870, 50, 189, 4, 42, 12, 4720, 3380,
21]\nclass = v'),
 Text(558.0, 181.19999999999982, 'gini = 0.773\nsamples = 11047\nvalue = [1, 0, 4410, 9
3, 19, 0, 0, 14, 3, 0, 0, 213, 0\n24, 22, 2305, 17, 5879, 14, 284, 0, 1433, 2675, 2]\ncl
ass = r'),
 Text(717.4285714285713, 906.0, 'O_3 <= 1.5\ngini = 0.806\nsamples = 18238\nvalue = [0,
11, 0, 0, 0, 37, 21, 0, 0, 3293, 2, 0, 924\n6083, 6103, 0, 5930, 0, 56, 0, 25, 0, 0, 623
2]\nclass = x'),
 Text(637.7142857142857, 543.5999999999999, 'gini = 0.761\nsamples = 14316\nvalue = [0,
1, 0, 0, 0, 10, 3, 0, 0, 3293, 2, 0, 924\n6083, 6103, 0, 5930, 0, 56, 0, 25, 0, 0, 65]\n
class = o'),
 Text(797.1428571428571, 543.5999999999999, 'O_3 <= 79.5\ngini = 0.018\nsamples = 3922\n
value = [0, 10, 0, 0, 0, 27, 18, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 6167]\n
class = x'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.022\nsamples = 3046\nvalue = [0,
10, 0, 0, 0, 27, 16, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 4799]\nclass = x'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.003\nsamples = 876\nvalue = [0,
0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 1368]\nclass = x'),
 Text(1594.2857142857142, 1268.4, 'PM25 <= 1.5\ngini = 0.886\nsamples = 31926\nvalue =
[6043, 5971, 0, 0, 5949, 5964, 106, 0, 6037, 2930\n6142, 0, 5110, 0, 0, 0, 0, 0, 0, 0, 6
090, 0, 0\n0]\nclass = k'),
 Text(1275.4285714285713, 906.0, 'TOL <= 0.95\ngini = 0.854\nsamples = 24298\nvalue = [6
043, 56, 0, 0, 5949, 5964, 1, 0, 6037, 2930, 211\n0, 5110, 0, 0, 0, 0, 0, 0, 0, 6090, 0,
0, 0]\nclass = u'),
 Text(1116.0, 543.5999999999999, 'NO <= 2.5\ngini = 0.053\nsamples = 2055\nvalue = [0, 1
2, 0, 0, 0, 3160, 0, 0, 0, 0, 77, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = f'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.031\nsamples = 1592\nvalue = [0,
1, 0, 0, 0, 2479, 0, 0, 0, 0, 38, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = f'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.129\nsamples = 463\nvalue = [0,
11, 0, 0, 0, 681, 0, 0, 0, 0, 39, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = f'),
 Text(1434.8571428571427, 543.5999999999999, 'O_3 <= 1.5\ngini = 0.848\nsamples = 22243
\nvalue = [6043, 44, 0, 0, 5949, 2804, 1, 0, 6037, 2930, 134\n0, 5110, 0, 0, 0, 0, 0, 0,
```

```
0, 6090, 0, 0, 0]\nclass = u'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.745\nsamples = 13009\nvalue = [60
43, 0, 0, 0, 29, 108, 0, 0, 186, 2930, 134, 0\n5110, 0, 0, 0, 0, 0, 0, 0, 6090, 0, 0, 0]
\nclass = u'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.637\nsamples = 9234\nvalue = [0,
44, 0, 0, 5920, 2696, 1, 0, 5851, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
e'),
 Text(1913.1428571428569, 906.0, 'SO_2 <= 3.5\ngini = 0.509\nsamples = 7628\nvalue = [0,
5915, 0, 0, 0, 0, 105, 0, 0, 0, 5931, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
k'),
 Text(1753.7142857142856, 543.5999999999999, 'CO <= 0.9\ngini = 0.349\nsamples = 3022\nv
alue = [0, 971, 0, 0, 0, 0, 75, 0, 0, 0, 3713, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nc
lass = k'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.134\nsamples = 676\nvalue = [0,
966, 0, 0, 0, 0, 75, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.003\nsamples = 2346\nvalue = [0,
5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3713, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = k'),
 Text(2072.5714285714284, 543.5999999999999, 'CO <= 0.95\ngini = 0.432\nsamples = 4606\n
value = [0, 4944, 0, 0, 0, 0, 30, 0, 0, 0, 2218, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
\nclass = b'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.012\nsamples = 3095\nvalue = [0,
4805, 0, 0, 0, 0, 28, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.112\nsamples = 1511\nvalue = [0,
139, 0, 0, 0, 0, 2, 0, 0, 0, 2218, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = k'),
 Text(3367.928571428571, 1630.8000000000002, 'BEN <= 0.95\ngini = 0.667\nsamples = 11505
\nvalue = [0, 0, 0, 0, 0, 0, 6068, 6157, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 6089, 0, 0, 0, 0,
0]\nclass = h'),
 Text(2869.7142857142853, 1268.4, 'BEN <= 0.25\ngini = 0.499\nsamples = 6447\nvalue =
[0, 0, 0, 0, 0, 0, 4875, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5438, 0, 0, 0, 0, 0]\nclass =
s'),
 Text(2550.8571428571427, 906.0, 'NMHC <= 0.205\ngini = 0.237\nsamples = 2857\nvalue =
[0, 0, 0, 0, 0, 0, 629, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3939, 0, 0, 0, 0, 0]\nclass =
s'),
 Text(2391.428571428571, 543.5999999999999, 'TOL <= 0.35\ngini = 0.009\nsamples = 2355\n
value = [0, 0, 0, 0, 0, 0, 17, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3766, 0, 0, 0, 0, 0]\ncl
ass = s'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.047\nsamples = 200\nvalue = [0,
0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 323, 0, 0, 0, 0, 0]\nclass = s'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.005\nsamples = 2155\nvalue = [0,
0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3443, 0, 0, 0, 0, 0]\nclass = s'),
 Text(2710.285714285714, 543.5999999999999, 'TCH <= 1.605\ngini = 0.344\nsamples = 502\n
value = [0, 0, 0, 0, 0, 0, 612, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 173, 0, 0, 0, 0, 0]\ncl
ass = g'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.077\nsamples = 390\nvalue = [0,
0, 0, 0, 0, 0, 578, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 24, 0, 0, 0, 0, 0]\nclass = g'),
 Text(2790.0, 181.19999999999982, 'gini = 0.303\nsamples = 112\nvalue = [0, 0, 0, 0, 0,
0, 34, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 149, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3188.5714285714284, 906.0, 'NMHC <= 0.195\ngini = 0.386\nsamples = 3590\nvalue =
[0, 0, 0, 0, 0, 0, 4246, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1499, 0, 0, 0, 0, 0]\nclass =
g'),
 Text(3029.142857142857, 543.5999999999999, 'TCH <= 1.355\ngini = 0.154\nsamples = 701\n
value = [0, 0, 0, 0, 0, 0, 95, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1036, 0, 0, 0, 0, 0]\ncl
ass = s'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.078\nsamples = 56\nvalue = [0, 0,
0, 0, 0, 0, 94, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 4, 0, 0, 0, 0, 0]\nclass = g'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.002\nsamples = 645\nvalue = [0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1032, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3347.9999999999995, 543.5999999999999, 'O_3 <= 1.5\ngini = 0.181\nsamples = 2889\n
value = [0, 0, 0, 0, 0, 0, 4151, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 463, 0, 0, 0, 0, 0]\nc
lass = g'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.025\nsamples = 301\nvalue = [0,
0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 463, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.0\nsamples = 2588\nvalue = [0,
0, 0, 0, 0, 0, 4145, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(3866.142857142857, 1268.4, 'NMHC <= 0.205\ngini = 0.379\nsamples = 5058\nvalue =
```

```
[0, 0, 0, 0, 0, 0, 1193, 6157, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 651, 0, 0, 0, 0, 0]\nclass
= h'),
 Text(3587.142857142857, 906.0, 'EBE <= 0.95\ngini = 0.066\nsamples = 3281\nvalue = [0,
0, 0, 0, 0, 0, 52, 4992, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 122, 0, 0, 0, 0, 0]\nclass = h'),
 Text(3507.428571428571, 543.5999999999999, 'gini = 0.0\nsamples = 37\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 57, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3666.8571428571427, 543.5999999999999, 'EBE <= 1.05\ngini = 0.045\nsamples = 3244
\nvalue = [0, 0, 0, 0, 0, 0, 52, 4992, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 65, 0, 0, 0, 0, 0]
\nclass = h'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.036\nsamples = 3228\nvalue = [0,
0, 0, 0, 0, 0, 37, 4992, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 56, 0, 0, 0, 0, 0]\nclass = h'),
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.469\nsamples = 16\nvalue = [0,
0, 0, 0, 0, 0, 15, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 9, 0, 0, 0, 0, 0]\nclass = g'),
 Text(4145.142857142857, 906.0, 'PM10 <= 1.5\ngini = 0.634\nsamples = 1777\nvalue = [0,
0, 0, 0, 0, 0, 1141, 1165, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 529, 0, 0, 0, 0, 0]\nclass =
h'),
 Text(3985.7142857142853, 543.5999999999999, 'O_3 <= 70.5\ngini = 0.002\nsamples = 737\n
value = [0, 0, 0, 0, 0, 0, 1, 1165, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\ncla
ss = h'),
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.0\nsamples = 720\nvalue = [0, 0,
0, 0, 0, 0, 0, 1140, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.074\nsamples = 17\nvalue = [0, 0,
0, 0, 0, 0, 1, 25, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(4304.571428571428, 543.5999999999999, 'NO_2 <= 52.5\ngini = 0.433\nsamples = 1040
\nvalue = [0, 0, 0, 0, 0, 0, 1140, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 529, 0, 0, 0, 0, 0]
\nclass = g'),
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.046\nsamples = 584\nvalue = [0,
0, 0, 0, 0, 0, 914, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 22, 0, 0, 0, 0, 0]\nclass = g'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.427\nsamples = 456\nvalue = [0,
0, 0, 0, 0, 0, 226, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 507, 0, 0, 0, 0, 0]\nclass = s')]
```



Conclusion

Accuracy

linear regression

In [66]:

```
lr.score(x_test,y_test)
```

Out[66]: 0.30506556835336696

Ridge regression

In [67]:
```
rr.score(x_test,y_test)
```

Out[67]: 0.30508743386847714

Lasso regression

In [68]:
```
la.score(x_test,y_test)
```

Out[68]: 0.04460958753352984

Elastic net regression

In [69]:
```
en.score(x_test,y_test)
```

Out[69]: 0.16493993283490482

Logistic regression

In [70]:
```
logr.score(fs,target_vector)
```

Out[70]: 0.7180674671240709

Random forest

In [71]:
```
grid_search.best_score_
```

Out[71]: 0.7496460562498298

Accuracy for random forest is higher so it is the best fit model

In [ ]:
```

```