In [52]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [53]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\madrid_2018.csv")
data
```

Out[53]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69091	2018-02-01 00:00:00	NaN	NaN	0.5	NaN	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN
69092	2018-02-01 00:00:00	NaN	NaN	0.7	NaN	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0
69093	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN
69094	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN
69095	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN

69096 rows × 16 columns

In [54]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69096 entries, 0 to 69095
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     69096 non-null  object 
 1   BEN      16950 non-null  float64
 2   CH4      8440 non-null   float64
 3   CO       28598 non-null  float64
 4   EBE      16949 non-null  float64
 5   NMHC     8440 non-null   float64
 6   NO       68826 non-null  float64
 7   NO_2     68826 non-null  float64
 8   NOx      68826 non-null  float64
 9   O_3      40049 non-null  float64
 10  PM10     36911 non-null  float64
 11  PM25     18912 non-null  float64
 12  SO_2     28586 non-null  float64
 13  TCH      8440 non-null   float64
 14  TOL      16950 non-null  float64
 15  station  69096 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 8.4+ MB
```

In [55]:
```
df=data.fillna(value=0)
df
```

Out[55]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	T
0	2018-03-01 01:00:00	0.0	0.00	0.3	0.0	0.00	1.0	29.0	31.0	0.0	0.0	0.0	2.0	0
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1
2	2018-03-01 01:00:00	0.4	0.00	0.0	0.2	0.00	4.0	41.0	47.0	0.0	0.0	0.0	0.0	0
3	2018-03-01 01:00:00	0.0	0.00	0.3	0.0	0.00	1.0	35.0	37.0	54.0	0.0	0.0	0.0	0
4	2018-03-01 01:00:00	0.0	0.00	0.0	0.0	0.00	1.0	27.0	29.0	49.0	0.0	0.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69091	2018-02-01 00:00:00	0.0	0.00	0.5	0.0	0.00	66.0	91.0	192.0	1.0	35.0	22.0	0.0	0
69092	2018-02-01 00:00:00	0.0	0.00	0.7	0.0	0.00	87.0	107.0	241.0	0.0	29.0	0.0	15.0	0
69093	2018-02-01 00:00:00	0.0	0.00	0.0	0.0	0.00	28.0	48.0	91.0	2.0	0.0	0.0	0.0	0
69094	2018-02-01 00:00:00	0.0	0.00	0.0	0.0	0.00	141.0	103.0	320.0	2.0	0.0	0.0	0.0	0
69095	2018-02-01 00:00:00	0.0	0.00	0.0	0.0	0.00	69.0	96.0	202.0	3.0	26.0	0.0	0.0	0

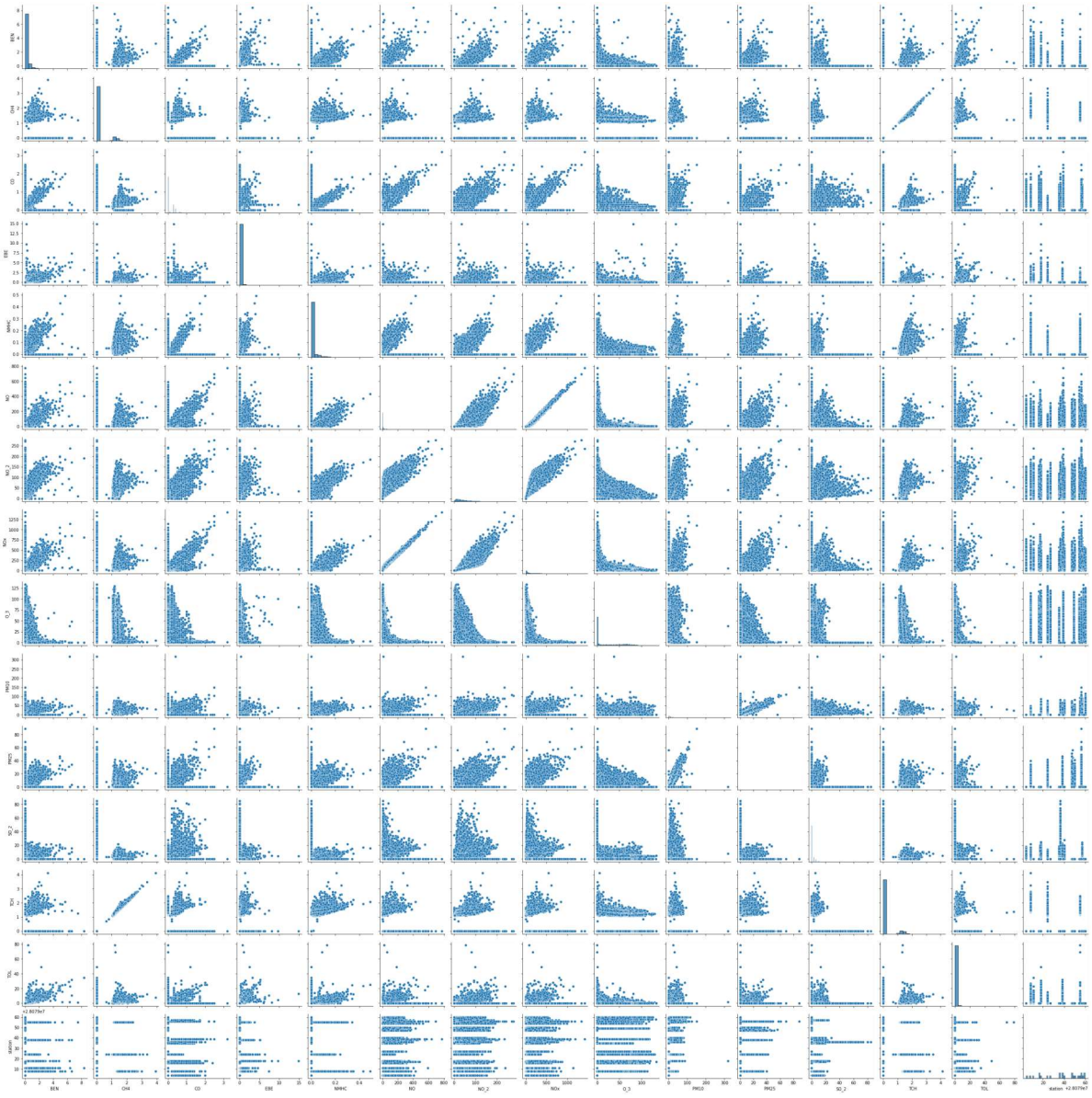
69096 rows × 16 columns

In [56]:
```
df.columns
```

Out[56]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [57]:
```
sns.pairplot(df)
```

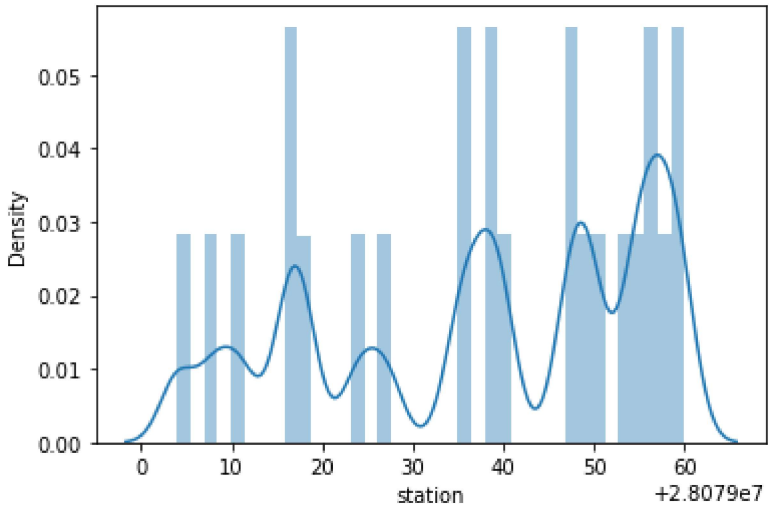
Out[57]: <seaborn.axisgrid.PairGrid at 0x18238c59a30>



In [58]:
```
sns.distplot(data["station"])
```

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

Out[58]: <AxesSubplot:xlabel='station', ylabel='Density'>



MODEL BUILDING

Linear Regression

In [59]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [60]:
```
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df1[['station']]
```

In [61]:
```
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)
```

In [62]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

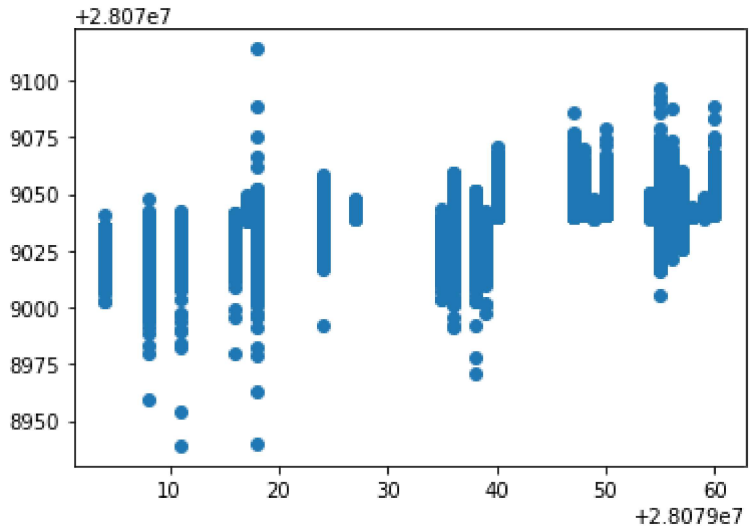
Out[62]: LinearRegression()

In [63]:
```
print(lr.intercept_)
```

```
[28079041.03513529]
```

In [64]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[64]: <matplotlib.collections.PathCollection at 0x182a1d44a30>



In [65]:
```
print(lr.score(x_test,y_test))
```

```
0.30613307142959867
```

Ridge Regression

In [66]:
```
from sklearn.linear_model import Ridge
```

In [67]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[67]: Ridge(alpha=10)

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

Out[68]: 0.2965713310473127

Lasso Regression

In [69]:
```
from sklearn.linear_model import Lasso
```

In [70]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[70]:
```
Lasso(alpha=10)
```

In [71]:
```
la.score(x_test,y_test)
```

Out[71]:
```
0.07141617353063268
```

Elastic Regression

In [72]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[72]:
```
ElasticNet()
```

In [73]:
```
print(en.coef_)
```

```
[-0.92072712 -0.76756663 -0.         -0.          0.02173633 -0.09149511
 -0.01362873  0.44150687 -0.01188086 -0.89412392 -1.22184198 -1.7236831 ]
```

In [74]:
```
print(en.predict(x_test))
```

```
[28079039.15137294 28079041.5121703  28079039.84982342 ...
 28079039.98346908 28079038.02854953 28079039.78244692]
```

In [75]:
```
print(en.score(x_test,y_test))
```

```
0.15077233398893786
```

Logistic Regression

In [76]:
```
from sklearn.linear_model import LogisticRegression
```

In [77]:
```
feature_matrix=df1.iloc[:,0:15]
target_vector=df1.iloc[:,-1]
```

In [78]:
```
feature_matrix.shape
```

Out[78]:
```
(69096, 13)
```

In [79]:
```
target_vector.shape
```

Out[79]:
```
(69096,)
```

In [80]:
```
from sklearn.preprocessing import StandardScaler
```

In [81]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [82]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[82]:
```
LogisticRegression()
```

In [83]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13]]
```

In [84]:
```
prediction=logr.predict(observation)
print(observation)
```

```
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]]
```

In [85]:
```
logr.classes_
```

Out[85]:
```
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 [86]:
```
logr.score(fs,target_vector)
```

Out[86]:
```
0.9570163251128864
```

Random Forest

In [87]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [88]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
x=df1[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
y=df1['station']
```

In [89]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.70)
```

In [90]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[90]:
```
RandomForestClassifier()
```

In [91]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [92]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring='acc
grid_search.fit(x_train,y_train)
```

Out[92]:
```
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 [93]:
```
grid_search.best_score_
```

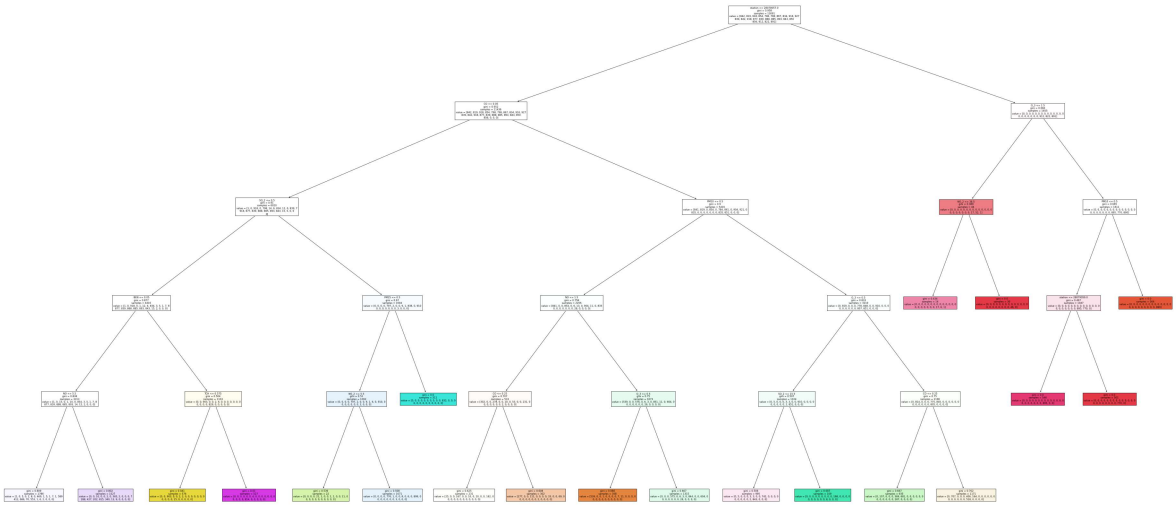
Out[93]:
```
0.9585584716325743
```

In [94]:
```
rfc_best=grid_search.best_estimator_
```

In [95]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

Out[95]:
```
[Text(2891.4545454545455, 1993.2, 'station <= 28079057.0\ngini = 0.958\nsampl
es = 13091\nvalue = [842, 819, 918, 854, 798, 798, 867, 834, 916, 927\n839, 8
42, 918, 877, 839, 888, 885, 893, 843, 850\n856, 912, 822, 891]'),
 Text(1860.0, 1630.8000000000002, 'CO <= 0.05\ngini = 0.952\nsamples = 11436
\nvalue = [842, 819, 918, 854, 798, 798, 867, 834, 916, 927\n839, 842, 918, 8
77, 839, 888, 885, 893, 843, 850\n856, 0, 0, 0]'),
 Text(1014.5454545454546, 1268.4, 'SO_2 <= 0.5\ngini = 0.91\nsamples = 6020\n
value = [1, 0, 918, 0, 798, 14, 6, 834, 12, 6, 839, 7\n918, 877, 839, 888, 88
5, 893, 843, 15, 5, 0, 0\n0]'),
 Text(541.0909090909091, 906.0, 'BEN <= 0.05\ngini = 0.877\nsamples = 4416\nv
alue = [1, 0, 918, 0, 1, 12, 6, 834, 3, 5, 1, 7, 8\n877, 839, 888, 885, 893,
843, 15, 2, 0, 0, 0]'),
 Text(270.54545454545456, 543.5999999999999, 'NO <= 5.5\ngini = 0.838\nsample
s = 3313\nvalue = [1, 0, 15, 0, 1, 10, 0, 834, 3, 5, 1, 7, 8\n877, 839, 888,
885, 893, 14, 15, 2, 0, 0, 0]'),
 Text(135.27272727272728, 181.19999999999982, 'gini = 0.809\nsamples = 1786\n
value = [1, 0, 5, 0, 1, 8, 0, 469, 1, 5, 1, 7, 1, 589\n412, 686, 70, 553, 1,
6, 2, 0, 0, 0]'),
 Text(405.81818181818187, 181.19999999999982, 'gini = 0.802\nsamples = 1527\n
value = [0, 0, 10, 0, 0, 2, 0, 365, 2, 0, 0, 0, 7\n288, 427, 202, 815, 340, 1
3, 9, 0, 0, 0, 0]'),
 Text(811.6363636363637, 543.5999999999999, 'TCH <= 0.575\ngini = 0.504\nsamp
les = 1103\nvalue = [0, 0, 903, 0, 0, 2, 6, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0,
829, 0, 0, 0, 0, 0]'),
 Text(676.3636363636364, 181.19999999999982, 'gini = 0.041\nsamples = 576\nva
lue = [0, 0, 903, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 15, 0, 0, 0,
0, 0]'),
 Text(946.909090909091, 181.19999999999982, 'gini = 0.01\nsamples = 527\nvalu
e = [0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 814, 0, 0, 0, 0,
0]'),
 Text(1488.0, 906.0, 'PM25 <= 0.5\ngini = 0.67\nsamples = 1604\nvalue = [0,
0, 0, 0, 797, 2, 0, 0, 9, 1, 838, 0, 910\n0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0]'),
 Text(1352.7272727272727, 543.5999999999999, 'NO_2 <= 5.5\ngini = 0.51\nsampl
es = 1093\nvalue = [0, 0, 0, 0, 797, 2, 0, 0, 9, 1, 6, 0, 910, 0\n0, 0, 0, 0,
0, 0, 3, 0, 0, 0]'),
 Text(1217.4545454545455, 181.19999999999982, 'gini = 0.539\nsamples = 22\nva
lue = [0, 0, 0, 0, 21, 1, 0, 0, 1, 1, 0, 0, 11, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0,
0]'),
 Text(1488.0, 181.19999999999982, 'gini = 0.508\nsamples = 1071\nvalue = [0,
0, 0, 0, 776, 1, 0, 0, 8, 0, 6, 0, 899, 0\n0, 0, 0, 0, 0, 0, 3, 0, 0, 0]'),
 Text(1623.2727272727275, 543.5999999999999, 'gini = 0.0\nsamples = 511\nvalu
e = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 832, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0,
0]'),
 Text(2705.4545454545455, 1268.4, 'PM10 <= 0.5\ngini = 0.9\nsamples = 5416\nv
alue = [841, 819, 0, 854, 0, 784, 861, 0, 904, 921, 0\n835, 0, 0, 0, 0, 0, 0,
0, 835, 851, 0, 0, 0]'),
 Text(2164.3636363636365, 906.0, 'NO <= 1.5\ngini = 0.758\nsamples = 2206\nva
lue = [841, 0, 0, 854, 0, 6, 13, 0, 904, 11, 0, 835\n0, 0, 0, 0, 0, 0, 0, 28,
0, 0, 0, 0]'),
 Text(1893.818181818182, 543.5999999999999, 'CO <= 0.25\ngini = 0.707\nsample
s = 533\nvalue = [302, 0, 0, 278, 0, 0, 10, 0, 53, 0, 0, 231, 0\n0, 0, 0, 0,
0, 0, 2, 0, 0, 0, 0]'),
 Text(1758.5454545454547, 181.19999999999982, 'gini = 0.625\nsamples = 231\nv
alue = [25, 0, 0, 167, 0, 0, 10, 0, 20, 0, 0, 162, 0\n0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]'),
 Text(2029.0909090909092, 181.19999999999982, 'gini = 0.608\nsamples = 302\nv
alue = [277, 0, 0, 111, 0, 0, 0, 0, 33, 0, 0, 69, 0\n0, 0, 0, 0, 0, 0, 2, 0,
```

```
0, 0, 0]'),
 Text(2434.909090909091, 543.5999999999999, 'O_3 <= 0.5\ngini = 0.75\nsamples
= 1673\nvalue = [539, 0, 0, 576, 0, 6, 3, 0, 851, 11, 0, 604, 0\n0, 0, 0, 0,
0, 0, 26, 0, 0, 0, 0]'),
 Text(2299.636363636364, 181.19999999999982, 'gini = 0.066\nsamples = 358\nva
lue = [539, 0, 0, 1, 0, 0, 0, 0, 7, 11, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0,
0, 0]'),
 Text(2570.1818181818185, 181.19999999999982, 'gini = 0.667\nsamples = 1315\n
value = [0, 0, 0, 575, 0, 6, 3, 0, 844, 0, 0, 604, 0\n0, 0, 0, 0, 0, 0, 26,
0, 0, 0, 0]'),
 Text(3246.545454545455, 906.0, 'O_3 <= 0.5\ngini = 0.833\nsamples = 3210\nva
lue = [0, 819, 0, 0, 0, 778, 848, 0, 0, 910, 0, 0, 0\n0, 0, 0, 0, 0, 0, 807,
851, 0, 0, 0]'),
 Text(2976.0, 543.5999999999999, 'SO_2 <= 15.5\ngini = 0.507\nsamples = 1104
\nvalue = [0, 5, 0, 0, 0, 3, 3, 0, 0, 910, 0, 0, 0, 0\n0, 0, 0, 0, 0, 2, 851,
0, 0, 0]'),
 Text(2840.727272727273, 181.19999999999982, 'gini = 0.506\nsamples = 995\nva
lue = [0, 5, 0, 0, 0, 3, 3, 0, 0, 742, 0, 0, 0, 0\n0, 0, 0, 0, 0, 2, 843, 0,
0, 0]'),
 Text(3111.2727272727275, 181.19999999999982, 'gini = 0.087\nsamples = 109\nv
alue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 168, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 8, 0,
0, 0]'),
 Text(3517.0909090909095, 543.5999999999999, 'CO <= 0.25\ngini = 0.75\nsample
s = 2106\nvalue = [0, 814, 0, 0, 0, 775, 845, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0,
0, 0, 805, 0, 0, 0, 0]'),
 Text(3381.818181818182, 181.19999999999982, 'gini = 0.667\nsamples = 935\nva
lue = [0, 107, 0, 0, 0, 369, 681, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 287, 0,
0, 0, 0]'),
 Text(3652.3636363636365, 181.19999999999982, 'gini = 0.702\nsamples = 1171\n
value = [0, 707, 0, 0, 0, 406, 164, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 518,
0, 0, 0, 0]'),
 Text(3922.909090909091, 1630.8000000000002, 'O_3 <= 1.5\ngini = 0.666\nsampl
es = 1655\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0,
0, 0, 912, 822, 891]'),
 Text(3652.3636363636365, 1268.4, 'NO_2 <= 38.5\ngini = 0.389\nsamples = 44\n
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 17, 5
2, 1]'),
 Text(3517.0909090909095, 906.0, 'gini = 0.434\nsamples = 14\nvalue = [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 17, 6, 1]'),
 Text(3787.636363636364, 906.0, 'gini = 0.0\nsamples = 30\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 46, 0]'),
 Text(4193.454545454546, 1268.4, 'PM10 <= 0.5\ngini = 0.665\nsamples = 1611\n
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 895,
770, 890]'),
 Text(4058.1818181818185, 906.0, 'station <= 28079059.0\ngini = 0.497\nsample
s = 1047\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0,
0, 0, 895, 770, 0]'),
 Text(3922.909090909091, 543.5999999999999, 'gini = 0.0\nsamples = 546\nvalue
= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 895, 0,
0]'),
 Text(4193.454545454546, 543.5999999999999, 'gini = 0.0\nsamples = 501\nvalue
= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 770,
0]'),
 Text(4328.727272727273, 906.0, 'gini = 0.0\nsamples = 564\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 890]')]
```



Results

The best model is Random Forest 0.9585584716325743

In [ ]: