In [45]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
```

In [89]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2018
a
```

Out[89]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	NaN	NaN
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	1.41	0.8
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	NaN	1.1
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	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	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	NaN	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	NaN	NaN
69093	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	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	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	NaN	NaN

69096 rows × 16 columns

In [90]:
```
a.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 [91]:
```
b=a.fillna(value=108)
b
```

Out[91]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH
0	2018-03-01 01:00:00	108.0	108.00	0.3	108.0	108.00	1.0	29.0	31.0	108.0	108.0	108.0	2.0	108.0(
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	1.4
2	2018-03-01 01:00:00	0.4	108.00	108.0	0.2	108.00	4.0	41.0	47.0	108.0	108.0	108.0	108.0	108.0(
3	2018-03-01 01:00:00	108.0	108.00	0.3	108.0	108.00	1.0	35.0	37.0	54.0	108.0	108.0	108.0	108.0(
4	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	27.0	29.0	49.0	108.0	108.0	3.0	108.0(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69091	2018-02-01 00:00:00	108.0	108.00	0.5	108.0	108.00	66.0	91.0	192.0	1.0	35.0	22.0	108.0	108.0(
69092	2018-02-01 00:00:00	108.0	108.00	0.7	108.0	108.00	87.0	107.0	241.0	108.0	29.0	108.0	15.0	108.0(
69093	2018-02-01 00:00:00	108.0	108.00	108.0	108.0	108.00	28.0	48.0	91.0	2.0	108.0	108.0	108.0	108.0(
69094	2018-02-01 00:00:00	108.0	108.00	108.0	108.0	108.00	141.0	103.0	320.0	2.0	108.0	108.0	108.0	108.0(
69095	2018-02-01 00:00:00	108.0	108.00	108.0	108.0	108.00	69.0	96.0	202.0	3.0	26.0	108.0	108.0	108.0(

69096 rows × 16 columns

In [92]:
```
b.columns
```

Out[92]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [93]:
```
c=b.head(30)
c
```

Out[93]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TC
0	2018-03-01 01:00:00	108.0	108.00	0.3	108.0	108.00	1.0	29.0	31.0	108.0	108.0	108.0	2.0	108.00	108
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	1.41	0
2	2018-03-01 01:00:00	0.4	108.00	108.0	0.2	108.00	4.0	41.0	47.0	108.0	108.0	108.0	108.0	108.00	1
3	2018-03-01 01:00:00	108.0	108.00	0.3	108.0	108.00	1.0	35.0	37.0	54.0	108.0	108.0	108.0	108.00	108
4	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	27.0	29.0	49.0	108.0	108.0	3.0	108.00	108
5	2018-03-01 01:00:00	0.3	108.00	0.3	0.2	108.00	1.0	27.0	29.0	57.0	8.0	108.0	6.0	108.00	1
6	2018-03-01 01:00:00	0.4	1.11	0.2	0.1	0.06	1.0	25.0	27.0	55.0	5.0	4.0	4.0	1.16	1
7	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	37.0	39.0	54.0	108.0	108.0	108.0	108.00	108
8	2018-03-01 01:00:00	108.0	108.00	0.5	108.0	108.00	3.0	43.0	47.0	29.0	108.0	108.0	5.0	108.00	108
9	2018-03-01 01:00:00	108.0	108.00	0.2	108.0	108.00	2.0	26.0	29.0	108.0	4.0	108.0	6.0	108.00	108
10	2018-03-01 01:00:00	0.4	108.00	108.0	0.3	108.00	2.0	30.0	34.0	108.0	2.0	2.0	3.0	108.00	0
11	2018-03-01 01:00:00	108.0	108.00	0.3	108.0	108.00	1.0	28.0	29.0	59.0	108.0	108.0	108.0	108.00	108
12	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	31.0	33.0	108.0	4.0	108.0	5.0	108.00	108
13	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	30.0	32.0	108.0	2.0	2.0	108.0	108.00	108
14	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	40.0	41.0	108.0	8.0	7.0	108.0	108.00	108
15	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	26.0	27.0	26.0	108.0	108.0	108.0	108.00	108
16	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	11.0	41.0	58.0	108.0	6.0	5.0	108.0	108.00	108
17	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	15.0	17.0	66.0	108.0	108.0	108.0	108.00	108

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TC
18	2018-03-01 01:00:00	0.3	1.37	108.0	0.3	0.03	1.0	49.0	51.0	108.0	5.0	108.0	108.0	1.41	1
19	2018-03-01 01:00:00	108.0	108.00	0.2	108.0	108.00	9.0	57.0	71.0	37.0	12.0	9.0	108.0	108.00	108
20	2018-03-01 01:00:00	108.0	108.00	0.3	108.0	108.00	1.0	29.0	31.0	108.0	3.0	108.0	11.0	108.00	108
21	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	23.0	25.0	41.0	108.0	108.0	108.0	108.00	108
22	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	29.0	31.0	54.0	108.0	108.0	108.0	108.00	108
23	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	23.0	25.0	67.0	1.0	108.0	108.0	108.00	108
24	2018-03-01 02:00:00	108.0	108.00	0.3	108.0	108.00	1.0	24.0	25.0	108.0	108.0	108.0	2.0	108.00	108
25	2018-03-01 02:00:00	0.4	1.42	0.2	0.1	0.01	4.0	26.0	32.0	64.0	4.0	4.0	3.0	1.44	0
26	2018-03-01 02:00:00	0.2	108.00	108.0	0.1	108.00	3.0	24.0	29.0	108.0	108.0	108.0	108.0	108.00	0
27	2018-03-01 02:00:00	108.0	108.00	0.2	108.0	108.00	1.0	17.0	19.0	67.0	108.0	108.0	108.0	108.00	108
28	2018-03-01 02:00:00	108.0	108.00	108.0	108.0	108.00	1.0	15.0	16.0	57.0	108.0	108.0	3.0	108.00	108
29	2018-03-01 02:00:00	0.2	108.00	0.2	0.1	108.00	4.0	21.0	26.0	62.0	6.0	108.0	6.0	108.00	0

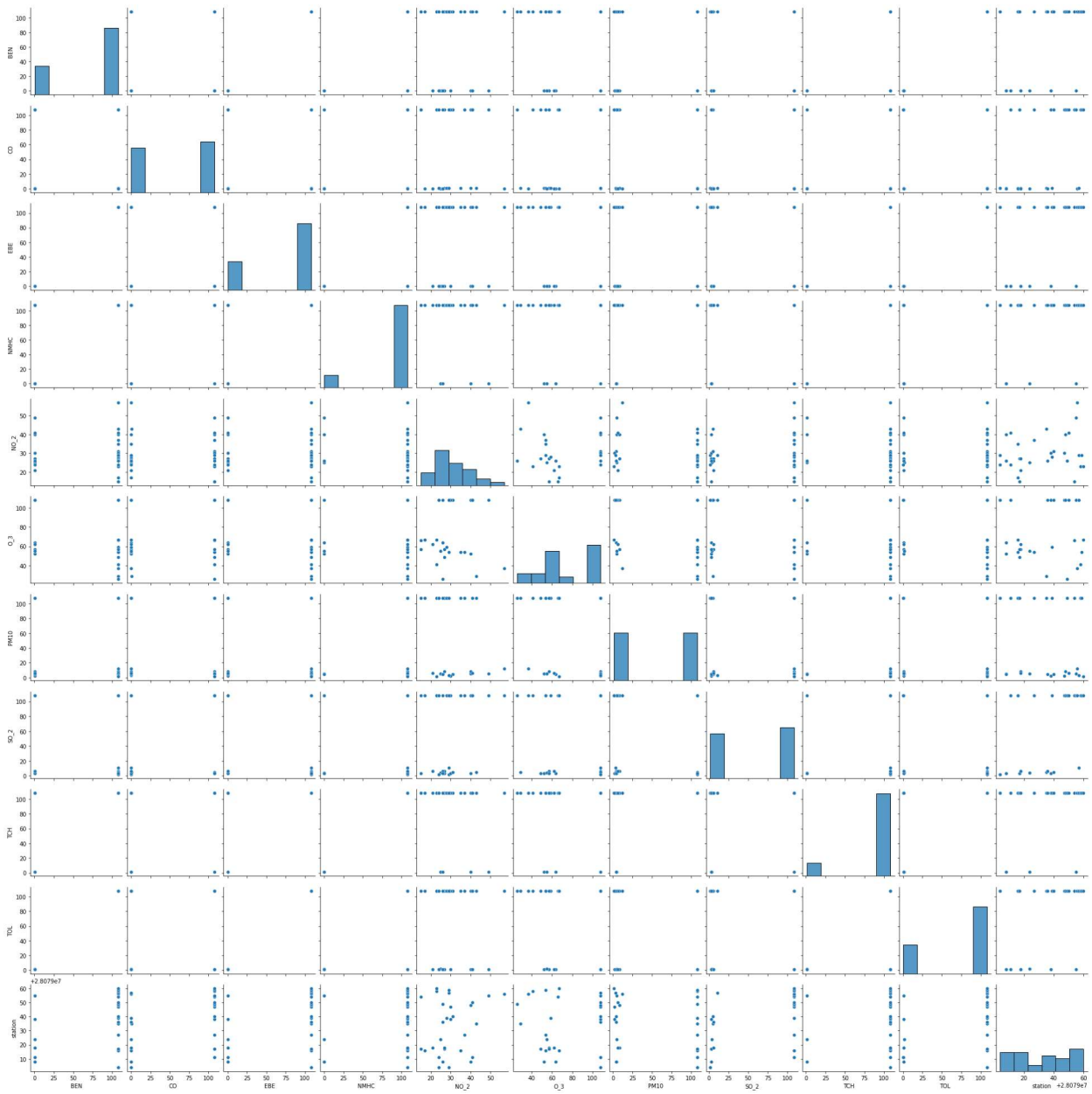
In [94]:
```
d=c[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[94]:

	BEN	CO	EBE	NMHC	NO_2	O_3	PM10	SO_2	TCH	TOL	station
0	108.0	0.3	108.0	108.00	29.0	108.0	108.0	2.0	108.00	108.0	28079004
1	0.5	0.3	0.2	0.02	40.0	52.0	5.0	3.0	1.41	0.8	28079008
2	0.4	108.0	0.2	108.00	41.0	108.0	108.0	108.0	108.00	1.1	28079011
3	108.0	0.3	108.0	108.00	35.0	54.0	108.0	108.0	108.00	108.0	28079016
4	108.0	108.0	108.0	108.00	27.0	49.0	108.0	3.0	108.00	108.0	28079017
5	0.3	0.3	0.2	108.00	27.0	57.0	8.0	6.0	108.00	1.0	28079018
6	0.4	0.2	0.1	0.06	25.0	55.0	5.0	4.0	1.16	1.4	28079024
7	108.0	108.0	108.0	108.00	37.0	54.0	108.0	108.0	108.00	108.0	28079027
8	108.0	0.5	108.0	108.00	43.0	29.0	108.0	5.0	108.00	108.0	28079035
9	108.0	0.2	108.0	108.00	26.0	108.0	4.0	6.0	108.00	108.0	28079036
10	0.4	108.0	0.3	108.00	30.0	108.0	2.0	3.0	108.00	0.9	28079038
11	108.0	0.3	108.0	108.00	28.0	59.0	108.0	108.0	108.00	108.0	28079039
12	108.0	108.0	108.0	108.00	31.0	108.0	4.0	5.0	108.00	108.0	28079040
13	108.0	108.0	108.0	108.00	30.0	108.0	2.0	108.0	108.00	108.0	28079047
14	108.0	108.0	108.0	108.00	40.0	108.0	8.0	108.0	108.00	108.0	28079048
15	108.0	108.0	108.0	108.00	26.0	26.0	108.0	108.0	108.00	108.0	28079049
16	108.0	108.0	108.0	108.00	41.0	108.0	6.0	108.0	108.00	108.0	28079050
17	108.0	108.0	108.0	108.00	15.0	66.0	108.0	108.0	108.00	108.0	28079054
18	0.3	108.0	0.3	0.03	49.0	108.0	5.0	108.0	1.41	1.1	28079055
19	108.0	0.2	108.0	108.00	57.0	37.0	12.0	108.0	108.00	108.0	28079056
20	108.0	0.3	108.0	108.00	29.0	108.0	3.0	11.0	108.00	108.0	28079057
21	108.0	108.0	108.0	108.00	23.0	41.0	108.0	108.0	108.00	108.0	28079058
22	108.0	108.0	108.0	108.00	29.0	54.0	108.0	108.0	108.00	108.0	28079059
23	108.0	108.0	108.0	108.00	23.0	67.0	1.0	108.0	108.00	108.0	28079060
24	108.0	0.3	108.0	108.00	24.0	108.0	108.0	2.0	108.00	108.0	28079004
25	0.4	0.2	0.1	0.01	26.0	64.0	4.0	3.0	1.44	0.7	28079008
26	0.2	108.0	0.1	108.00	24.0	108.0	108.0	108.0	108.00	0.5	28079011
27	108.0	0.2	108.0	108.00	17.0	67.0	108.0	108.0	108.00	108.0	28079016
28	108.0	108.0	108.0	108.00	15.0	57.0	108.0	3.0	108.00	108.0	28079017
29	0.2	0.2	0.1	108.00	21.0	62.0	6.0	6.0	108.00	0.8	28079018

In [95]:
```
sns.pairplot(d)
```

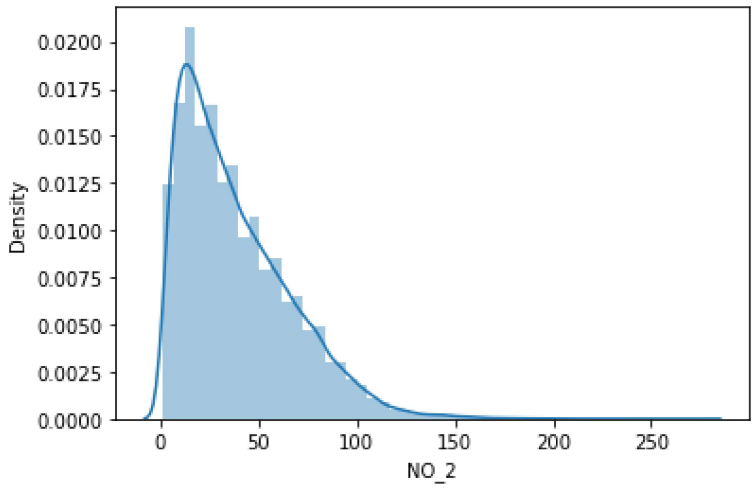
Out[95]:
```
<seaborn.axisgrid.PairGrid at 0x1d52eb02bb0>
```



In [96]:
```
sns.distplot(a['NO_2'])
```

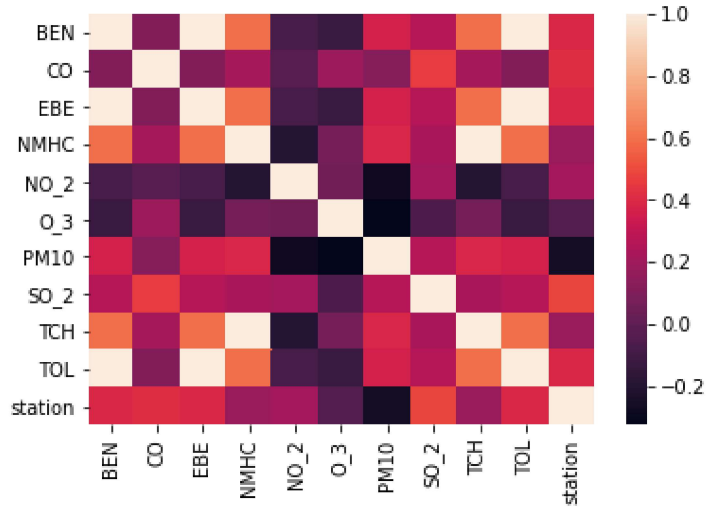
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[96]: <AxesSubplot:xlabel='NO_2', ylabel='Density'>



In [97]:
```
sns.heatmap(d.corr())
```

Out[97]: <AxesSubplot:>



In [98]:
```
x=d[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2']]
y=d['TCH']
```

In [99]:
```
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 [100]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[100]: LinearRegression()

In [101]:
```
print(lr.intercept_)
```

```
1.3113198883086454
```

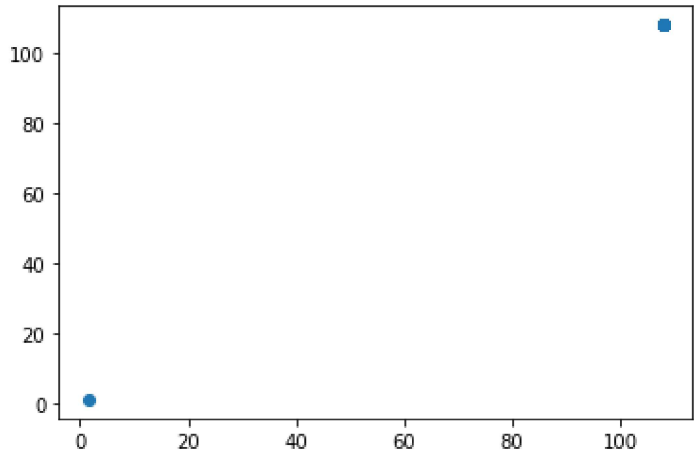
In [102]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[102]:

	Co-efficient
BEN	-0.343592
CO	0.000062
EBE	0.343201
NMHC	0.987878
NO_2	0.001203

In [103]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[103]:
```
<matplotlib.collections.PathCollection at 0x1d547eba550>
```



In [104]:
```
print(lr.score(x_test,y_test))
```

```
0.9999963827290002
```

In [105]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [106]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[106]:
```
Ridge(alpha=10)
```

In [107]:
```
rr.score(x_test,y_test)
```

Out[107]:
```
0.9999955176655577
```

In [108]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[108]:
```
Lasso(alpha=10)
```

In [109]:
```
la.score(x_test,y_test)
```

Out[109]:
```
0.9999650862226702
```

In [110]:
```
a1=b.head(7000)
a1
```

Out[110]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	
0	2018-03-01 01:00:00	108.0	108.00	0.3	108.0	108.00	1.0	29.0	31.0	108.0	108.0	108.0	2.0	108.00	1(
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	1.41	
2	2018-03-01 01:00:00	0.4	108.00	108.0	0.2	108.00	4.0	41.0	47.0	108.0	108.0	108.0	108.0	108.00	
3	2018-03-01 01:00:00	108.0	108.00	0.3	108.0	108.00	1.0	35.0	37.0	54.0	108.0	108.0	108.0	108.00	1(
4	2018-03-01 01:00:00	108.0	108.00	108.0	108.0	108.00	1.0	27.0	29.0	49.0	108.0	108.0	3.0	108.00	1(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
6995	2018-03-13 04:00:00	108.0	108.00	0.2	108.0	108.00	1.0	9.0	11.0	60.0	108.0	108.0	108.0	108.00	1(
6996	2018-03-13 04:00:00	108.0	108.00	108.0	108.0	108.00	1.0	38.0	39.0	108.0	15.0	108.0	3.0	108.00	1(
6997	2018-03-13 04:00:00	108.0	108.00	108.0	108.0	108.00	1.0	17.0	18.0	108.0	8.0	3.0	108.0	108.00	1(
6998	2018-03-13 04:00:00	108.0	108.00	108.0	108.0	108.00	1.0	14.0	16.0	108.0	7.0	5.0	108.0	108.00	1(
6999	2018-03-13 04:00:00	108.0	108.00	108.0	108.0	108.00	1.0	10.0	11.0	49.0	108.0	108.0	108.0	108.00	1(

7000 rows × 16 columns

In [111]:
```
e=a1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [112]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [113]:
```
h=StandardScaler().fit_transform(f)
```

In [114]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[114]:
```
LogisticRegression(max_iter=10000)
```

In [115]:
```
from sklearn.model_selection import train_test_split
h_train,h_test,g_train,g_test=train_test_split(h,g,test_size=0.3)
```

In [116]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58]]
```

In [117]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079050]
```

In [118]:
```
logr.classes_
```

Out[118]:
```
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 [119]:
```
logr.predict_proba(i)[0][0]
```

Out[119]:
```
0.0
```

In [120]:
```
logr.predict_proba(i)[0][1]
```

Out[120]:
```
0.0
```

In [121]:
```
logr.score(h_test,g_test)
```

Out[121]:
```
0.9514285714285714
```

In [122]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[122]:
```
ElasticNet()
```

In [123]:
```
print(en.coef_)
```

```
[1.07602847e-04 1.84741801e-04 4.51418614e-04 9.86808025e-01
 0.00000000e+00]
```

In [124]:
```
print(en.intercept_)
```

```
1.3480982529327292
```

In [125]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999978021904281
```

In [126]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[126]:
```
RandomForestClassifier()
```

In [127]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [128]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(h_train,g_train)
```

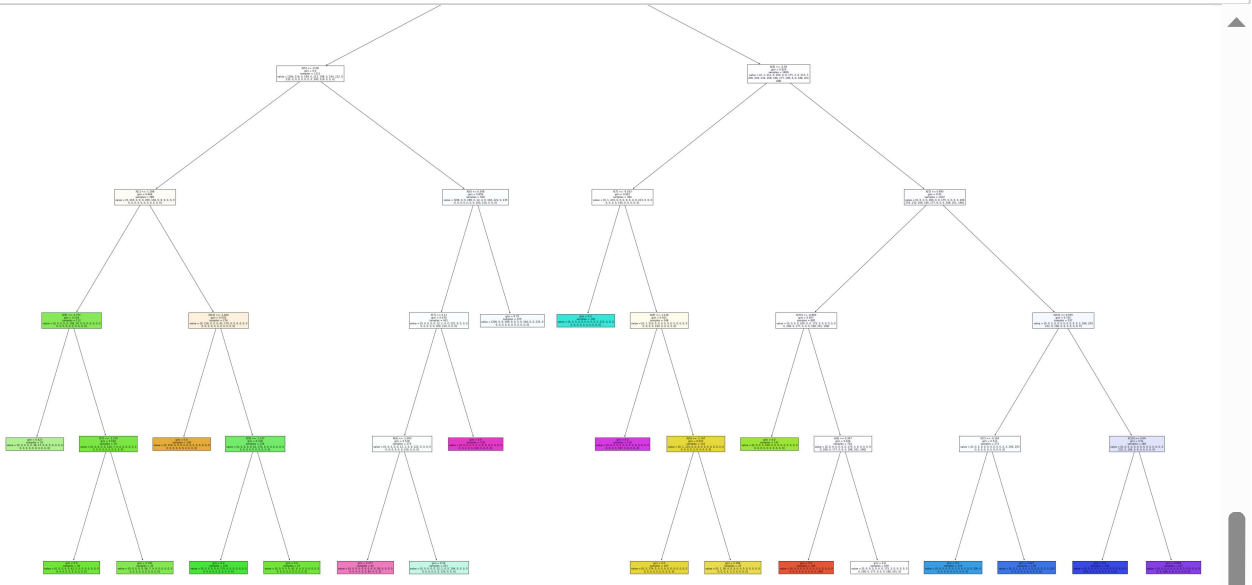
Out[128]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [129]:
```
grid_search.best_score_
```

Out[129]:
```
0.9959183673469387
```

In [130]:
```
rfc_best=grid_search.best_estimator_
```

In [131]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```



Conclusion: from this data set i observed that the ELASTIC NET has the highest accuracy of 0.9999978021904281

In [ ]: