In [45]:
```python
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 [46]:
```python
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2017
a
```

Out[46]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2017-06-01 01:00:00	NaN	NaN	0.3	NaN	NaN	4.0	38.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	2
1	2017-06-01 01:00:00	0.6	NaN	0.3	0.4	0.08	3.0	39.0	NaN	71.0	22.0	9.0	7.0	1.4	2.9	2
2	2017-06-01 01:00:00	0.2	NaN	NaN	0.1	NaN	1.0	14.0	NaN	NaN	NaN	NaN	NaN	NaN	0.9	2
3	2017-06-01 01:00:00	NaN	NaN	0.2	NaN	NaN	1.0	9.0	NaN	91.0	NaN	NaN	NaN	NaN	NaN	2
4	2017-06-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	19.0	NaN	69.0	NaN	NaN	2.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210115	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	27.0	NaN	65.0	NaN	NaN	NaN	NaN	NaN	2
210116	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	14.0	NaN	NaN	73.0	NaN	7.0	NaN	NaN	2
210117	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	4.0	NaN	83.0	NaN	NaN	NaN	NaN	NaN	2
210118	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	11.0	NaN	78.0	NaN	NaN	NaN	NaN	NaN	2
210119	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	14.0	NaN	77.0	60.0	NaN	NaN	NaN	NaN	2

210120 rows × 16 columns

In [47]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210120 entries, 0 to 210119
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210120 non-null  object 
 1   BEN      50201 non-null   float64
 2   CH4      6410 non-null    float64
 3   CO       87001 non-null   float64
 4   EBE      49973 non-null   float64
 5   NMHC     25472 non-null   float64
 6   NO       209065 non-null  float64
 7   NO_2     209065 non-null  float64
 8   NOx      52818 non-null   float64
 9   O_3      121398 non-null  float64
 10  PM10     104141 non-null  float64
 11  PM25     52023 non-null   float64
 12  SO_2     86803 non-null   float64
 13  TCH      25472 non-null   float64
 14  TOL      50117 non-null   float64
 15  station  210120 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 25.6+ MB
```

In [48]:
```
b=a.fillna(value=108)
b
```

Out[48]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH
0	2017-06-01 01:00:00	108.0	108.0	0.3	108.0	108.00	4.0	38.0	108.0	108.0	108.0	108.0	5.0	108.0
1	2017-06-01 01:00:00	0.6	108.0	0.3	0.4	0.08	3.0	39.0	108.0	71.0	22.0	9.0	7.0	1.4
2	2017-06-01 01:00:00	0.2	108.0	108.0	0.1	108.00	1.0	14.0	108.0	108.0	108.0	108.0	108.0	108.0
3	2017-06-01 01:00:00	108.0	108.0	0.2	108.0	108.00	1.0	9.0	108.0	91.0	108.0	108.0	108.0	108.0
4	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	19.0	108.0	69.0	108.0	108.0	2.0	108.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210115	2017-08-01 00:00:00	108.0	108.0	0.2	108.0	108.00	1.0	27.0	108.0	65.0	108.0	108.0	108.0	108.0
210116	2017-08-01 00:00:00	108.0	108.0	0.2	108.0	108.00	1.0	14.0	108.0	108.0	73.0	108.0	7.0	108.0
210117	2017-08-01 00:00:00	108.0	108.0	108.0	108.0	108.00	1.0	4.0	108.0	83.0	108.0	108.0	108.0	108.0
210118	2017-08-01 00:00:00	108.0	108.0	108.0	108.0	108.00	1.0	11.0	108.0	78.0	108.0	108.0	108.0	108.0
210119	2017-08-01 00:00:00	108.0	108.0	108.0	108.0	108.00	1.0	14.0	108.0	77.0	60.0	108.0	108.0	108.0

210120 rows × 16 columns

In [49]:
```
b.columns
```

Out[49]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [50]:
```
c=b.head(30)
c
```

Out[50]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TO
0	2017-06-01 01:00:00	108.0	108.0	0.3	108.0	108.00	4.0	38.0	108.0	108.0	108.0	108.0	5.0	108.00	108.
1	2017-06-01 01:00:00	0.6	108.0	0.3	0.4	0.08	3.0	39.0	108.0	71.0	22.0	9.0	7.0	1.40	2.
2	2017-06-01 01:00:00	0.2	108.0	108.0	0.1	108.00	1.0	14.0	108.0	108.0	108.0	108.0	108.0	108.00	0.
3	2017-06-01 01:00:00	108.0	108.0	0.2	108.0	108.00	1.0	9.0	108.0	91.0	108.0	108.0	108.0	108.00	108.
4	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	19.0	108.0	69.0	108.0	108.0	2.0	108.00	108.
5	2017-06-01 01:00:00	0.1	108.0	0.3	0.2	108.00	1.0	26.0	108.0	70.0	26.0	108.0	1.0	108.00	0.
6	2017-06-01 01:00:00	0.3	108.0	0.2	0.1	0.17	1.0	19.0	108.0	79.0	23.0	9.0	3.0	0.86	1.
7	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	9.0	108.0	87.0	108.0	108.0	108.0	108.00	108.
8	2017-06-01 01:00:00	108.0	108.0	0.3	108.0	108.00	3.0	30.0	108.0	70.0	108.0	108.0	108.0	108.00	108.
9	2017-06-01 01:00:00	108.0	108.0	0.1	108.0	108.00	1.0	15.0	108.0	108.0	22.0	108.0	10.0	108.00	108.
10	2017-06-01 01:00:00	0.7	108.0	108.0	1.0	108.00	1.0	25.0	108.0	108.0	21.0	10.0	2.0	108.00	3.
11	2017-06-01 01:00:00	108.0	108.0	0.2	108.0	108.00	1.0	21.0	108.0	75.0	108.0	108.0	108.0	108.00	108.
12	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	2.0	17.0	108.0	108.0	24.0	108.0	9.0	108.00	108.
13	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	22.0	108.0	108.0	23.0	15.0	108.0	108.00	108.
14	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	2.0	30.0	108.0	108.0	17.0	9.0	108.0	108.00	108.
15	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	12.0	108.0	74.0	108.0	108.0	108.0	108.00	108.
16	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	2.0	15.0	108.0	108.0	16.0	12.0	108.0	108.00	108.
17	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	3.0	12.0	108.0	84.0	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	TO
18	2017-06-01 01:00:00	0.2	108.0	108.0	0.6	0.08	1.0	12.0	108.0	108.0	15.0	108.0	108.0	1.16	1.
19	2017-06-01 01:00:00	108.0	108.0	0.1	108.0	108.00	9.0	47.0	108.0	59.0	108.0	108.0	108.0	108.00	108.
20	2017-06-01 01:00:00	108.0	108.0	0.3	108.0	108.00	1.0	17.0	108.0	108.0	48.0	108.0	3.0	108.00	108.
21	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	10.0	108.0	66.0	108.0	108.0	108.0	108.00	108.
22	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	6.0	108.0	91.0	108.0	108.0	108.0	108.00	108.
23	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	26.0	108.0	79.0	86.0	108.0	108.0	108.00	108.
24	2017-06-01 02:00:00	108.0	108.0	0.3	108.0	108.00	4.0	27.0	108.0	108.0	108.0	108.0	5.0	108.00	108.
25	2017-06-01 02:00:00	0.3	108.0	0.3	0.2	0.07	2.0	27.0	108.0	72.0	16.0	7.0	7.0	1.40	2.
26	2017-06-01 02:00:00	0.1	108.0	108.0	0.1	108.00	1.0	13.0	108.0	108.0	108.0	108.0	108.0	108.00	1.
27	2017-06-01 02:00:00	108.0	108.0	0.2	108.0	108.00	1.0	8.0	108.0	90.0	108.0	108.0	108.0	108.00	108.
28	2017-06-01 02:00:00	108.0	108.0	108.0	108.0	108.00	1.0	10.0	108.0	77.0	108.0	108.0	2.0	108.00	108.
29	2017-06-01 02:00:00	0.1	108.0	0.3	0.5	108.00	2.0	12.0	108.0	84.0	23.0	108.0	1.0	108.00	0.

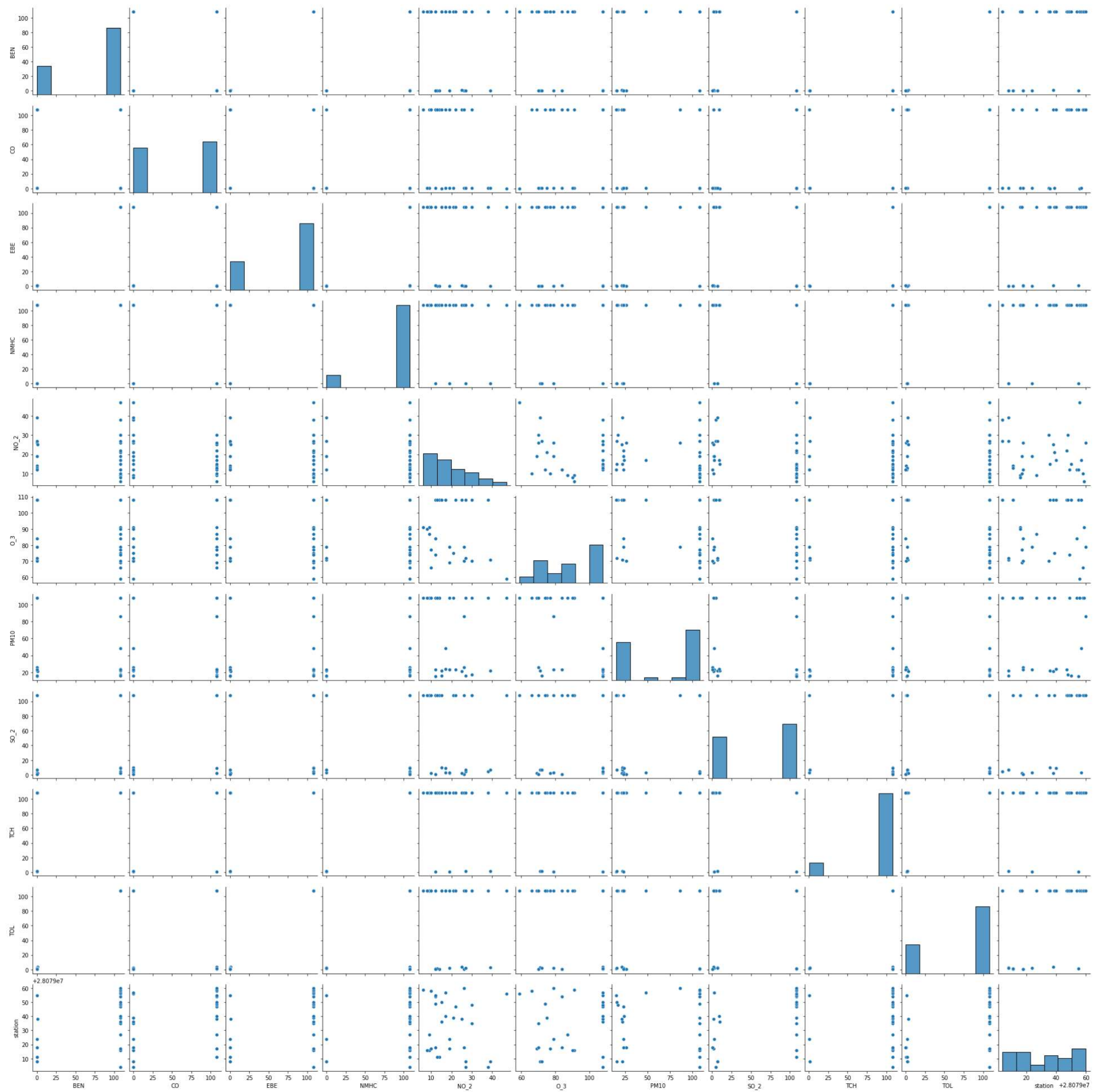
In [51]:
```
d=c[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[51]:

	BEN	CO	EBE	NMHC	NO_2	O_3	PM10	SO_2	TCH	TOL	station
0	108.0	0.3	108.0	108.00	38.0	108.0	108.0	5.0	108.00	108.0	28079004
1	0.6	0.3	0.4	0.08	39.0	71.0	22.0	7.0	1.40	2.9	28079008
2	0.2	108.0	0.1	108.00	14.0	108.0	108.0	108.0	108.00	0.9	28079011
3	108.0	0.2	108.0	108.00	9.0	91.0	108.0	108.0	108.00	108.0	28079016
4	108.0	108.0	108.0	108.00	19.0	69.0	108.0	2.0	108.00	108.0	28079017
5	0.1	0.3	0.2	108.00	26.0	70.0	26.0	1.0	108.00	0.3	28079018
6	0.3	0.2	0.1	0.17	19.0	79.0	23.0	3.0	0.86	1.8	28079024
7	108.0	108.0	108.0	108.00	9.0	87.0	108.0	108.0	108.00	108.0	28079027
8	108.0	0.3	108.0	108.00	30.0	70.0	108.0	108.0	108.00	108.0	28079035
9	108.0	0.1	108.0	108.00	15.0	108.0	22.0	10.0	108.00	108.0	28079036
10	0.7	108.0	1.0	108.00	25.0	108.0	21.0	2.0	108.00	3.5	28079038
11	108.0	0.2	108.0	108.00	21.0	75.0	108.0	108.0	108.00	108.0	28079039
12	108.0	108.0	108.0	108.00	17.0	108.0	24.0	9.0	108.00	108.0	28079040
13	108.0	108.0	108.0	108.00	22.0	108.0	23.0	108.0	108.00	108.0	28079047
14	108.0	108.0	108.0	108.00	30.0	108.0	17.0	108.0	108.00	108.0	28079048
15	108.0	108.0	108.0	108.00	12.0	74.0	108.0	108.0	108.00	108.0	28079049
16	108.0	108.0	108.0	108.00	15.0	108.0	16.0	108.0	108.00	108.0	28079050
17	108.0	108.0	108.0	108.00	12.0	84.0	108.0	108.0	108.00	108.0	28079054
18	0.2	108.0	0.6	0.08	12.0	108.0	15.0	108.0	1.16	1.5	28079055
19	108.0	0.1	108.0	108.00	47.0	59.0	108.0	108.0	108.00	108.0	28079056
20	108.0	0.3	108.0	108.00	17.0	108.0	48.0	3.0	108.00	108.0	28079057
21	108.0	108.0	108.0	108.00	10.0	66.0	108.0	108.0	108.00	108.0	28079058
22	108.0	108.0	108.0	108.00	6.0	91.0	108.0	108.0	108.00	108.0	28079059
23	108.0	108.0	108.0	108.00	26.0	79.0	86.0	108.0	108.00	108.0	28079060
24	108.0	0.3	108.0	108.00	27.0	108.0	108.0	5.0	108.00	108.0	28079004
25	0.3	0.3	0.2	0.07	27.0	72.0	16.0	7.0	1.40	2.3	28079008
26	0.1	108.0	0.1	108.00	13.0	108.0	108.0	108.0	108.00	1.7	28079011
27	108.0	0.2	108.0	108.00	8.0	90.0	108.0	108.0	108.00	108.0	28079016
28	108.0	108.0	108.0	108.00	10.0	77.0	108.0	2.0	108.00	108.0	28079017
29	0.1	0.3	0.5	108.00	12.0	84.0	23.0	1.0	108.00	0.2	28079018

In [52]:
```
sns.pairplot(d)
```

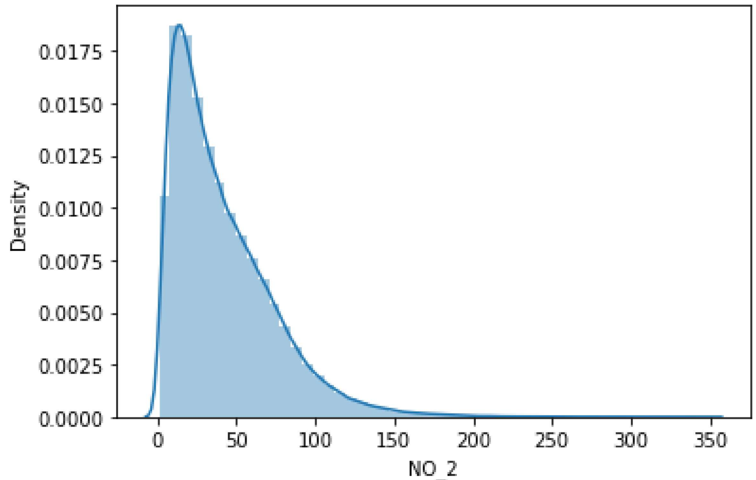
Out[52]: <seaborn.axisgrid.PairGrid at 0x1d51f696b80>



In [53]:
```
sns.distplot(a['NO_2'])
```

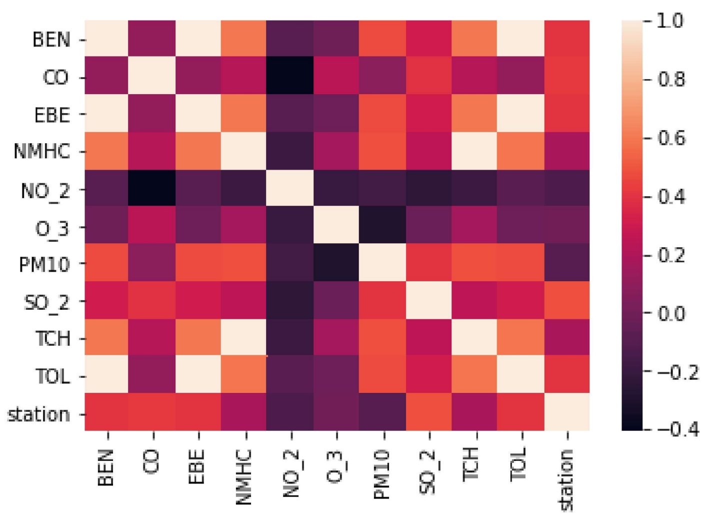
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[53]: <AxesSubplot:xlabel='NO_2', ylabel='Density'>



In [54]:
```
sns.heatmap(d.corr())
```

Out[54]: <AxesSubplot:>



In [55]:
```
x=d[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2']]
y=d['TCH']
```

In [56]:
```
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 [57]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[57]: LinearRegression()

In [58]:
```
print(lr.intercept_)
```

```
1.1687473095814198
```

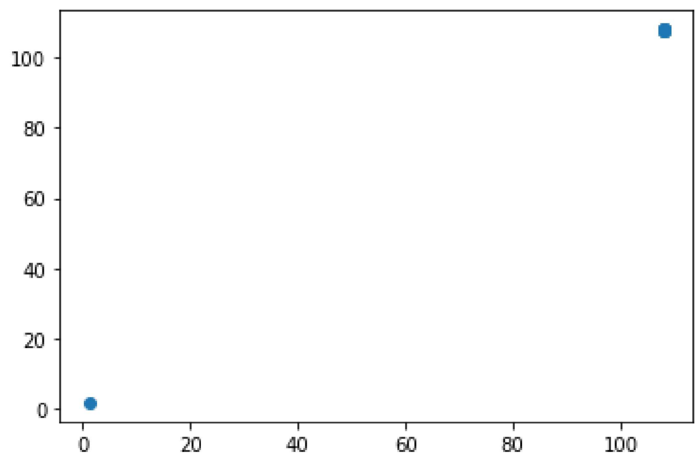
In [59]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[59]:

	Co-efficient
BEN	-0.832336
CO	0.000011
EBE	0.834073
NMHC	0.986916
NO_2	0.002979

In [60]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[60]: <matplotlib.collections.PathCollection at 0x1d53c9688e0>



In [61]:
```
print(lr.score(x_test,y_test))
```

```
0.9999652404027495
```

In [62]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [63]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[63]: Ridge(alpha=10)

In [64]:
```
rr.score(x_test,y_test)
```

Out[64]: 0.9999973930952113

In [65]:
```python
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[65]:
```
Lasso(alpha=10)
```

In [66]:
```python
la.score(x_test,y_test)
```

Out[66]:
```
0.9999448413207095
```

In [67]:
```python
a1=b.head(7000)
a1
```

Out[67]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	T
0	2017-06-01 01:00:00	108.0	108.0	0.3	108.0	108.00	4.0	38.0	108.0	108.0	108.0	108.0	5.0	108.0	10
1	2017-06-01 01:00:00	0.6	108.0	0.3	0.4	0.08	3.0	39.0	108.0	71.0	22.0	9.0	7.0	1.4	2
2	2017-06-01 01:00:00	0.2	108.0	108.0	0.1	108.00	1.0	14.0	108.0	108.0	108.0	108.0	108.0	108.0	(
3	2017-06-01 01:00:00	108.0	108.0	0.2	108.0	108.00	1.0	9.0	108.0	91.0	108.0	108.0	108.0	108.0	10
4	2017-06-01 01:00:00	108.0	108.0	108.0	108.0	108.00	1.0	19.0	108.0	69.0	108.0	108.0	2.0	108.0	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
6995	2017-06-13 06:00:00	108.0	108.0	0.2	108.0	108.00	1.0	9.0	108.0	84.0	108.0	108.0	108.0	108.0	10
6996	2017-06-13 06:00:00	108.0	108.0	108.0	108.0	108.00	1.0	13.0	108.0	108.0	7.0	108.0	9.0	108.0	10
6997	2017-06-13 06:00:00	108.0	108.0	108.0	108.0	108.00	1.0	11.0	108.0	108.0	20.0	17.0	108.0	108.0	10
6998	2017-06-13 06:00:00	108.0	108.0	108.0	108.0	108.00	1.0	2.0	108.0	108.0	8.0	4.0	108.0	108.0	10
6999	2017-06-13 06:00:00	108.0	108.0	108.0	108.0	108.00	1.0	3.0	108.0	76.0	108.0	108.0	108.0	108.0	10

7000 rows × 16 columns

In [68]:
```python
e=a1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [69]:
```python
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [70]:
```
h=StandardScaler().fit_transform(f)
```

In [71]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[71]:
```
LogisticRegression(max_iter=10000)
```

In [72]:
```
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 [73]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58]]
```

In [74]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079059]
```

In [75]:
```
logr.classes_
```

Out[75]:
```
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 [76]:
```
logr.predict_proba(i)[0][0]
```

Out[76]:
```
0.0
```

In [77]:
```
logr.predict_proba(i)[0][1]
```

Out[77]:
```
0.0
```

In [78]:
```
logr.score(h_test,g_test)
```

Out[78]:
```
0.9376190476190476
```

In [79]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[79]:
```
ElasticNet()
```

In [80]:
```
print(en.coef_)
```

```
[0.00000000e+00 9.79297475e-06 4.03498376e-04 9.88618444e-01
 0.00000000e+00]
```

In [81]:
```
print(en.intercept_)
```

```
1.179084449674619
```

In [82]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999982880826986
```

In [83]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[83]:
```
RandomForestClassifier()
```

In [84]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [85]:
```
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[85]:
```
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 [86]:
```
grid_search.best_score_
```

Out[86]:
```
0.9824489795918367
```

In [87]:
```
rfc_best=grid_search.best_estimator_
```

In [88]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```

```
0, 0, 0]'),
 Text(2232.0, 679.5, 'X[9] <= -1.819\ngini = 0.122\nsamples = 111\nvalue = [0, 0,
0, 0, 0, 157, 11, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2008.8, 226.5, 'gini = 0.349\nsamples = 27\nvalue = [0, 0, 0, 0, 0, 31, 9, 0,
0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(2455.2, 226.5, 'gini = 0.031\nsamples = 84\nvalue = [0, 0, 0, 0, 0, 126, 2,
0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(3124.7999999999997, 679.5, 'X[6] <= -0.203\ngini = 0.835\nsamples = 763\nvalu
e = [0, 0, 0, 224, 0, 14, 0, 0, 180, 220, 0, 212, 0\n0, 0, 0, 0, 0, 0, 186, 177, 0,
0, 0]'),
 Text(2901.6, 226.5, 'gini = 0.521\nsamples = 264\nvalue = [0, 0, 0, 0, 0, 11, 0,
0, 0, 219, 0, 0, 0, 0\n0, 0, 0, 0, 0, 177, 0, 0, 0]'),
 Text(3348.0, 226.5, 'gini = 0.75\nsamples = 499\nvalue = [0, 0, 0, 224, 0, 3, 0,
0, 180, 1, 0, 212, 0\n0, 0, 0, 0, 0, 186, 0, 0, 0]'),
 Text(3794.3999999999996, 1132.5, 'X[4] <= -0.923\ngini = 0.925\nsamples = 1691\nva
lue = [0, 0, 203, 0, 208, 0, 19, 211, 1, 0, 219, 0\n175, 185, 191, 234, 173, 227,
2, 23, 0, 216, 207\n214]'),
 Text(3571.2, 679.5, 'gini = 0.814\nsamples = 175\nvalue = [0, 0, 11, 0, 0, 0, 0, 3
0, 0, 0, 3, 0, 1, 8\n20, 43, 3, 7, 0, 0, 0, 74, 80, 7]'),
```

Conclusion: from this data set i observed that the ELASTIC NET has the highest accuracy of 0.9999982880826986

In [ ]: