In [652]:
```
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 [696]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\mad
a
```

Out[696]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210091	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN	28
210092	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN	28
210093	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN	28
210094	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN	28
210095	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN	28

210096 rows × 14 columns

In [697]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210096 entries, 0 to 210095
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210096 non-null  object 
 1   BEN      51039 non-null   float64
 2   CO       86827 non-null   float64
 3   EBE      50962 non-null   float64
 4   NMHC     25756 non-null   float64
 5   NO       208805 non-null  float64
 6   NO_2     208805 non-null  float64
 7   O_3      121574 non-null  float64
 8   PM10     102745 non-null  float64
 9   PM25     48798 non-null   float64
 10  SO_2     86898 non-null   float64
 11  TCH      25756 non-null   float64
 12  TOL      50626 non-null   float64
 13  station  210096 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [698]:
```
b=a.fillna(value=86)
b
```

Out[698]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2015-10-01 01:00:00	86.0	0.8	86.0	86.00	90.0	82.0	86.0	86.0	86.0	10.0	86.00	86.0	28
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28
2	2015-10-01 01:00:00	3.1	86.0	1.8	86.00	29.0	97.0	86.0	86.0	86.0	86.0	86.00	7.1	28
3	2015-10-01 01:00:00	86.0	0.6	86.0	86.00	30.0	103.0	2.0	86.0	86.0	86.0	86.00	86.0	28
4	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	95.0	96.0	2.0	86.0	86.0	9.0	86.00	86.0	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210091	2015-08-01 00:00:00	86.0	0.2	86.0	86.00	11.0	33.0	53.0	86.0	86.0	86.0	86.00	86.0	28
210092	2015-08-01 00:00:00	86.0	0.2	86.0	86.00	1.0	5.0	86.0	26.0	86.0	10.0	86.00	86.0	28
210093	2015-08-01 00:00:00	86.0	86.0	86.0	86.00	1.0	7.0	74.0	86.0	86.0	86.0	86.00	86.0	28
210094	2015-08-01 00:00:00	86.0	86.0	86.0	86.00	3.0	7.0	65.0	86.0	86.0	86.0	86.00	86.0	28
210095	2015-08-01 00:00:00	86.0	86.0	86.0	86.00	1.0	9.0	54.0	29.0	86.0	86.0	86.00	86.0	28

210096 rows × 14 columns

In [699]:
```
b.columns
```

Out[699]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [700]:
```
c=b.head(30)
c
```

Out[700]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	sta
0	2015-10-01 01:00:00	86.0	0.8	86.0	86.00	90.0	82.0	86.0	86.0	86.0	10.0	86.00	86.0	2807
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	2807
2	2015-10-01 01:00:00	3.1	86.0	1.8	86.00	29.0	97.0	86.0	86.0	86.0	86.0	86.00	7.1	2807
3	2015-10-01 01:00:00	86.0	0.6	86.0	86.00	30.0	103.0	2.0	86.0	86.0	86.0	86.00	86.0	2807
4	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	95.0	96.0	2.0	86.0	86.0	9.0	86.00	86.0	2807
5	2015-10-01 01:00:00	0.7	0.4	0.3	86.00	35.0	104.0	1.0	26.0	86.0	3.0	86.00	3.3	2807
6	2015-10-01 01:00:00	0.5	0.3	0.3	0.12	6.0	83.0	1.0	19.0	12.0	3.0	1.29	4.8	2807
7	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	54.0	94.0	1.0	86.0	86.0	86.0	86.00	86.0	2807
8	2015-10-01 01:00:00	86.0	0.5	86.0	86.00	38.0	114.0	16.0	86.0	86.0	86.0	86.00	86.0	2807
9	2015-10-01 01:00:00	86.0	0.7	86.0	86.00	64.0	97.0	86.0	34.0	86.0	6.0	86.00	86.0	2807
10	2015-10-01 01:00:00	0.3	86.0	0.4	86.00	16.0	69.0	86.0	18.0	12.0	3.0	86.00	3.1	2807
11	2015-10-01 01:00:00	86.0	0.6	86.0	86.00	59.0	99.0	7.0	86.0	86.0	86.0	86.00	86.0	2807
12	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	74.0	106.0	86.0	31.0	86.0	6.0	86.00	86.0	2807
13	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	77.0	97.0	86.0	23.0	15.0	86.0	86.00	86.0	2807
14	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	2.0	63.0	86.0	18.0	13.0	86.0	86.00	86.0	2807
15	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	2.0	70.0	15.0	86.0	86.0	86.0	86.00	86.0	2807
16	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	13.0	76.0	86.0	27.0	16.0	86.0	86.00	86.0	2807

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	sta
17	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	59.0	111.0	5.0	86.0	86.0	86.0	86.00	86.0	2807
18	2015-10-01 01:00:00	0.6	86.0	1.9	0.42	94.0	106.0	86.0	31.0	86.0	86.0	1.89	5.8	2807
19	2015-10-01 01:00:00	86.0	0.7	86.0	86.00	85.0	87.0	6.0	86.0	86.0	86.0	86.00	86.0	2807
20	2015-10-01 01:00:00	86.0	0.5	86.0	86.00	20.0	79.0	86.0	21.0	86.0	11.0	86.00	86.0	2807
21	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	1.0	32.0	23.0	86.0	86.0	86.0	86.00	86.0	2807
22	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	64.0	71.0	8.0	86.0	86.0	86.0	86.00	86.0	2807
23	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	2.0	79.0	22.0	9.0	86.0	86.0	86.00	86.0	2807
24	2015-10-01 02:00:00	86.0	0.8	86.0	86.00	102.0	76.0	86.0	86.0	86.0	10.0	86.00	86.0	2807
25	2015-10-01 02:00:00	1.6	0.7	1.3	0.38	81.0	105.0	4.0	36.0	19.0	13.0	1.93	6.9	2807
26	2015-10-01 02:00:00	3.6	86.0	4.2	86.00	75.0	111.0	86.0	86.0	86.0	86.0	86.00	86.0	2807
27	2015-10-01 02:00:00	86.0	0.5	86.0	86.00	42.0	102.0	2.0	86.0	86.0	86.0	86.00	86.0	2807
28	2015-10-01 02:00:00	86.0	86.0	86.0	86.00	87.0	89.0	2.0	86.0	86.0	9.0	86.00	86.0	2807
29	2015-10-01 02:00:00	0.8	0.5	0.4	86.00	70.0	97.0	1.0	23.0	86.0	4.0	86.00	5.5	2807

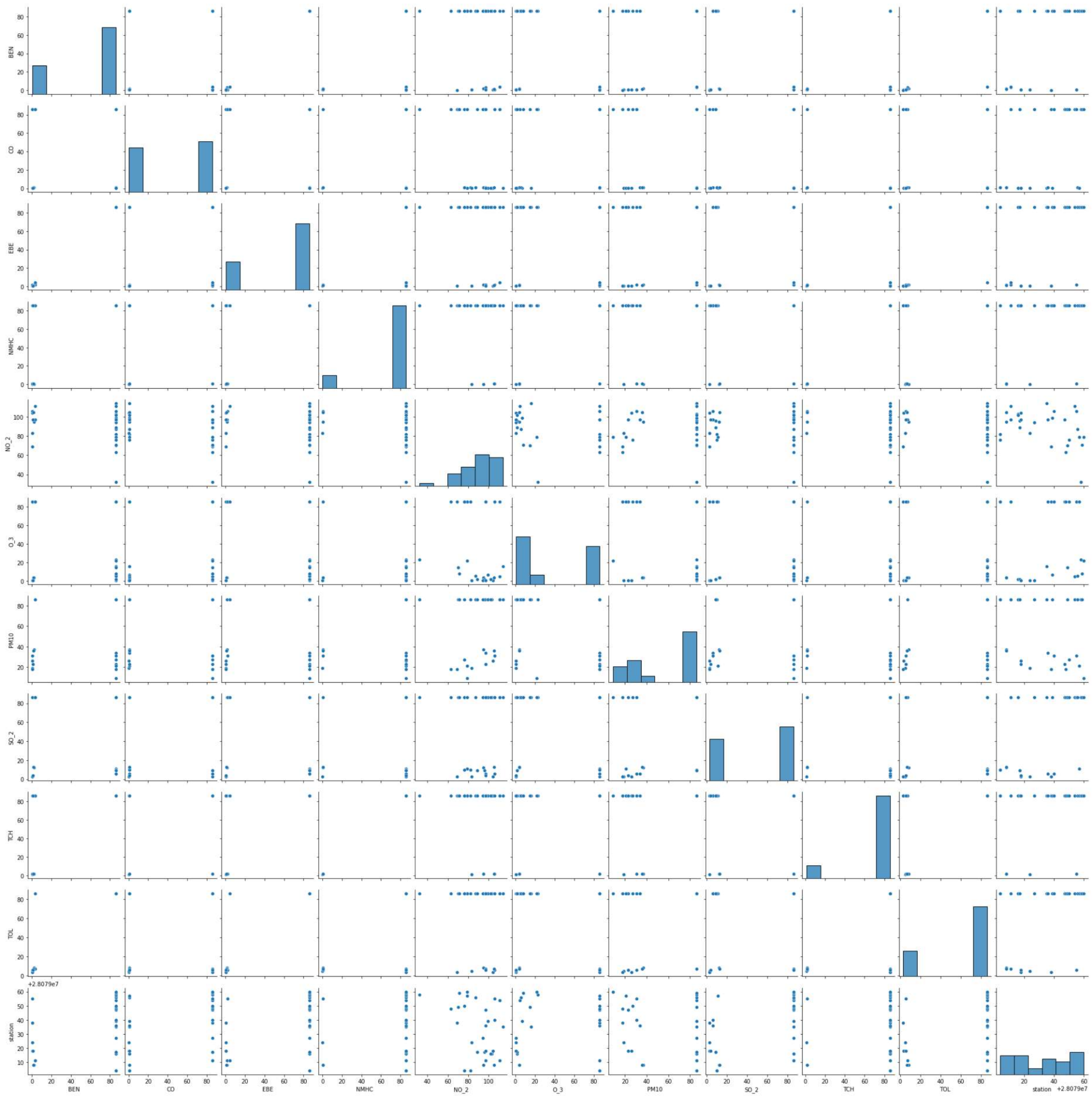
In [701]:
```
d=c[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
d
```

Out[701]:

	BEN	CO	EBE	NMHC	NO_2	O_3	PM10	SO_2	TCH	TOL	station
0	86.0	0.8	86.0	86.00	82.0	86.0	86.0	10.0	86.00	86.0	28079004
1	2.0	0.8	1.6	0.33	95.0	4.0	37.0	12.0	1.83	8.3	28079008
2	3.1	86.0	1.8	86.00	97.0	86.0	86.0	86.0	86.00	7.1	28079011
3	86.0	0.6	86.0	86.00	103.0	2.0	86.0	86.0	86.00	86.0	28079016
4	86.0	86.0	86.0	86.00	96.0	2.0	86.0	9.0	86.00	86.0	28079017
5	0.7	0.4	0.3	86.00	104.0	1.0	26.0	3.0	86.00	3.3	28079018
6	0.5	0.3	0.3	0.12	83.0	1.0	19.0	3.0	1.29	4.8	28079024
7	86.0	86.0	86.0	86.00	94.0	1.0	86.0	86.0	86.00	86.0	28079027
8	86.0	0.5	86.0	86.00	114.0	16.0	86.0	86.0	86.00	86.0	28079035
9	86.0	0.7	86.0	86.00	97.0	86.0	34.0	6.0	86.00	86.0	28079036
10	0.3	86.0	0.4	86.00	69.0	86.0	18.0	3.0	86.00	3.1	28079038
11	86.0	0.6	86.0	86.00	99.0	7.0	86.0	86.0	86.00	86.0	28079039
12	86.0	86.0	86.0	86.00	106.0	86.0	31.0	6.0	86.00	86.0	28079040
13	86.0	86.0	86.0	86.00	97.0	86.0	23.0	86.0	86.00	86.0	28079047
14	86.0	86.0	86.0	86.00	63.0	86.0	18.0	86.0	86.00	86.0	28079048
15	86.0	86.0	86.0	86.00	70.0	15.0	86.0	86.0	86.00	86.0	28079049
16	86.0	86.0	86.0	86.00	76.0	86.0	27.0	86.0	86.00	86.0	28079050
17	86.0	86.0	86.0	86.00	111.0	5.0	86.0	86.0	86.00	86.0	28079054
18	0.6	86.0	1.9	0.42	106.0	86.0	31.0	86.0	1.89	5.8	28079055
19	86.0	0.7	86.0	86.00	87.0	6.0	86.0	86.0	86.00	86.0	28079056
20	86.0	0.5	86.0	86.00	79.0	86.0	21.0	11.0	86.00	86.0	28079057
21	86.0	86.0	86.0	86.00	32.0	23.0	86.0	86.0	86.00	86.0	28079058
22	86.0	86.0	86.0	86.00	71.0	8.0	86.0	86.0	86.00	86.0	28079059
23	86.0	86.0	86.0	86.00	79.0	22.0	9.0	86.0	86.00	86.0	28079060
24	86.0	0.8	86.0	86.00	76.0	86.0	86.0	10.0	86.00	86.0	28079004
25	1.6	0.7	1.3	0.38	105.0	4.0	36.0	13.0	1.93	6.9	28079008
26	3.6	86.0	4.2	86.00	111.0	86.0	86.0	86.0	86.00	86.0	28079011
27	86.0	0.5	86.0	86.00	102.0	2.0	86.0	86.0	86.00	86.0	28079016
28	86.0	86.0	86.0	86.00	89.0	2.0	86.0	9.0	86.00	86.0	28079017
29	0.8	0.5	0.4	86.00	97.0	1.0	23.0	4.0	86.00	5.5	28079018

In [702]:
```
sns.pairplot(d)
```

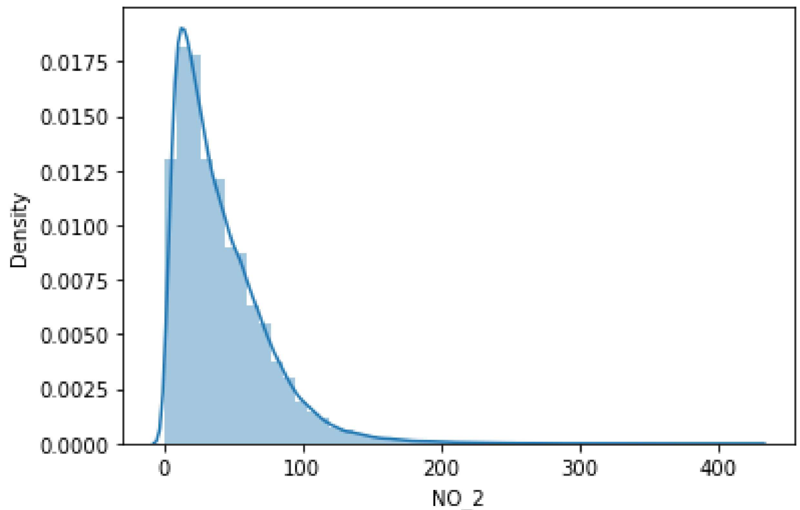
Out[702]: <seaborn.axisgrid.PairGrid at 0x1b72b021a90>



In [703]:
```
sns.distplot(a['NO_2'])
```

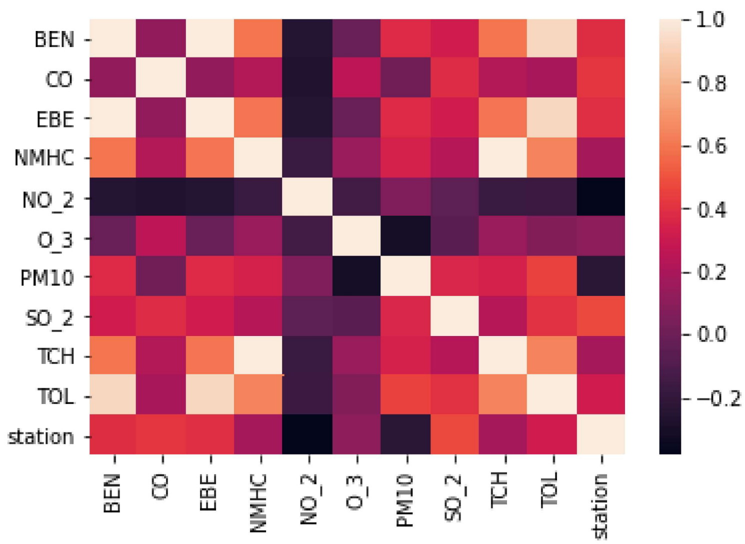
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[703]: <AxesSubplot:xlabel='NO_2', ylabel='Density'>



In [704]:
```
sns.heatmap(d.corr())
```

Out[704]: <AxesSubplot:>



In [705]:
```
x=d[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2']]
y=d['TCH']
```

In [706]:
```
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 [707]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[707]: LinearRegression()

In [708]:
```
print(lr.intercept_)
```

1.495083223867809

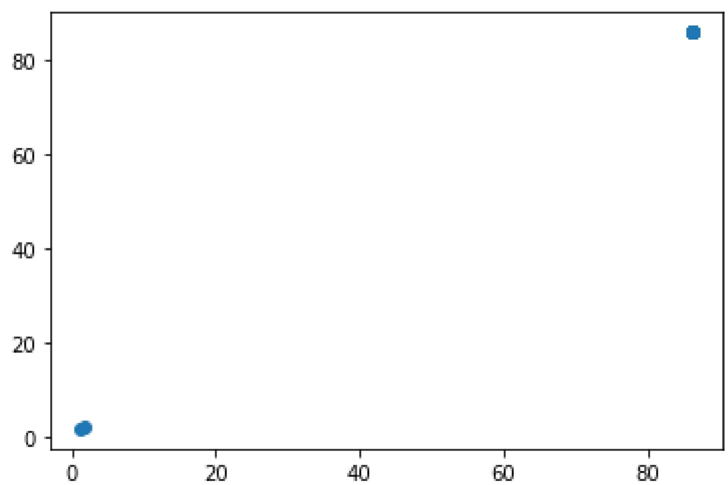
In [709]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[709]:

	Co-efficient
BEN	0.012264
CO	0.000011
EBE	-0.012253
NMHC	0.982583
NO_2	0.000015

In [710]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[710]: <matplotlib.collections.PathCollection at 0x1b73e5bc580>



In [711]:
```
print(lr.score(x_test,y_test))
```

0.9999900592725229

In [712]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [713]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[713]: Ridge(alpha=10)

In [714]:
```
rr.score(x_test,y_test)
```
Out[714]: 0.9999865717116763

In [715]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[715]: Lasso(alpha=10)

In [716]:
```
la.score(x_test,y_test)
```
Out[716]: 0.9996509829127287

In [717]:
```
a1=b.head(7000)
a1
```
Out[717]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	st
0	2015-10-01 01:00:00	86.0	0.8	86.0	86.00	90.0	82.0	86.0	86.0	86.0	10.0	86.00	86.0	2807
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	2807
2	2015-10-01 01:00:00	3.1	86.0	1.8	86.00	29.0	97.0	86.0	86.0	86.0	86.0	86.00	7.1	2807
3	2015-10-01 01:00:00	86.0	0.6	86.0	86.00	30.0	103.0	2.0	86.0	86.0	86.0	86.00	86.0	2807
4	2015-10-01 01:00:00	86.0	86.0	86.0	86.00	95.0	96.0	2.0	86.0	86.0	9.0	86.00	86.0	2807
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
6995	2015-10-13 04:00:00	86.0	0.3	86.0	86.00	32.0	44.0	7.0	86.0	86.0	86.0	86.00	86.0	2807
6996	2015-10-13 04:00:00	86.0	86.0	86.0	86.00	9.0	44.0	86.0	11.0	86.0	4.0	86.00	86.0	2807
6997	2015-10-13 04:00:00	86.0	86.0	86.0	86.00	30.0	27.0	86.0	28.0	23.0	86.0	86.00	86.0	2807
6998	2015-10-13 04:00:00	86.0	86.0	86.0	86.00	32.0	46.0	86.0	14.0	11.0	86.0	86.00	86.0	2807
6999	2015-10-13 04:00:00	86.0	86.0	86.0	86.00	44.0	37.0	2.0	86.0	86.0	86.0	86.00	86.0	2807

7000 rows × 14 columns

In [718]:
```
e=a1[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [719]:
```
f=e.iloc[:,0:14]
g=e.iloc[:,-1]
```

In [720]:
```
h=StandardScaler().fit_transform(f)
```

In [721]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(h,g)
```

Out[721]: LogisticRegression(max_iter=10000)

In [722]:
```
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 [723]:
```
i=[[10,20,30,40,50,60,15,26,37,47,58]]
```

In [724]:
```
prediction=logr.predict(i)
print(prediction)
```

```
[28079059]
```

In [725]:
```
logr.classes_
```

Out[725]:
```
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 [726]:
```
logr.predict_proba(i)[0][0]
```

Out[726]: 0.0

In [727]:
```
logr.predict_proba(i)[0][1]
```

Out[727]: 0.0

In [728]:
```
logr.score(h_test,g_test)
```

Out[728]: 0.9238095238095239

In [729]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[729]: ElasticNet()

In [730]:
```
print(en.coef_)
```

```
[ 4.15512330e-04 -0.00000000e+00  0.00000000e+00  9.80742749e-01
 -0.00000000e+00]
```

In [731]:
```
print(en.intercept_)
```

```
1.6131355931421183
```

In [732]:
```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999818078484011
```

In [733]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(h_train,g_train)
```

Out[733]:
```
RandomForestClassifier()
```

In [734]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [735]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accu
grid_search.fit(h_train,g_train)
```

Out[735]:
```
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 [736]:
```
grid_search.best_score_
```

Out[736]:
```
0.9904081632653061
```

In [737]:
```
rfc_best=grid_search.best_estimator_
```

In [738]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[2],filled=True)
```

Out[738]:
```
[Text(2612.4545454545455, 2491.5, 'X[10] <= 1.179\ngini = 0.958\nsamples =
3099\nvalue = [196, 199, 213, 198, 195, 216, 228, 213, 203, 198\n212, 199,
235, 208, 218, 178, 197, 214, 188, 173\n204, 182, 225, 208]'),
 Text(1369.6363636363635, 2038.5, 'X[3] <= -1.138\ngini = 0.954\nsamples =
2822\nvalue = [196, 199, 213, 198, 195, 216, 228, 213, 203, 198\n212, 199,
235, 208, 218, 178, 197, 214, 188, 173\n204, 182, 0, 0]'),
 Text(405.8181818181818, 1585.5, 'X[10] <= -1.234\ngini = 0.664\nsamples =
388\nvalue = [0, 199, 0, 0, 0, 0, 228, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 18
4, 0, 0, 0, 0, 0]'),
 Text(202.9090909090909, 1132.5, 'gini = 0.0\nsamples = 132\nvalue = [0, 19
9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(608.7272727272727, 1132.5, 'X[5] <= 0.893\ngini = 0.494\nsamples = 25
6\nvalue = [0, 0, 0, 0, 0, 0, 228, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 184, 0,
0, 0, 0, 0]'),
 Text(405.8181818181818, 679.5, 'gini = 0.0\nsamples = 125\nvalue = [0, 0,
0, 0, 0, 0, 206, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(811.6363636363636, 679.5, 'X[7] <= -0.217\ngini = 0.191\nsamples = 13
1\nvalue = [0, 0, 0, 0, 0, 0, 22, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 184, 0,
0, 0, 0, 0]'),
```

Conclusion: from this data set i observed that the elasticnet has the highest accuracy of 0.9999818078484011

In [ ]: