In [1]:
```
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
```

In [2]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\ma
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN

209496 rows × 14 columns

In [3]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209496 entries, 0 to 209495
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209496 non-null  object 
 1   BEN      50755 non-null   float64
 2   CO       85999 non-null   float64
 3   EBE      50335 non-null   float64
 4   NMHC     25970 non-null   float64
 5   NO       208614 non-null  float64
 6   NO_2     208614 non-null  float64
 7   O_3      121197 non-null  float64
 8   PM10     102892 non-null  float64
 9   PM25     52165 non-null   float64
 10  SO_2     86023 non-null   float64
 11  TCH      25970 non-null   float64
 12  TOL      50662 non-null   float64
 13  station  209496 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [4]:
```
b=a.fillna(value=192)
```

Out[4]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH
0	2016-11-01 01:00:00	192.0	0.7	192.0	192.00	153.0	77.0	192.0	192.0	192.0	7.0	192.00
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44
2	2016-11-01 01:00:00	5.9	192.0	7.5	192.00	297.0	139.0	192.0	192.0	192.0	192.0	192.00
3	2016-11-01 01:00:00	192.0	1.0	192.0	192.00	154.0	113.0	2.0	192.0	192.0	192.0	192.00
4	2016-11-01 01:00:00	192.0	192.0	192.0	192.00	275.0	127.0	2.0	192.0	192.0	18.0	192.00
...	...	...	...	...	...	...	...	...	...	...	...	...
209491	2016-07-01 00:00:00	192.0	0.2	192.0	192.00	2.0	29.0	73.0	192.0	192.0	192.0	192.00
209492	2016-07-01 00:00:00	192.0	0.3	192.0	192.00	1.0	29.0	192.0	36.0	192.0	5.0	192.00
209493	2016-07-01 00:00:00	192.0	192.0	192.0	192.00	1.0	19.0	71.0	192.0	192.0	192.0	192.00
209494	2016-07-01 00:00:00	192.0	192.0	192.0	192.00	6.0	17.0	85.0	192.0	192.0	192.0	192.00
209495	2016-07-01 00:00:00	192.0	192.0	192.0	192.00	2.0	46.0	61.0	34.0	192.0	192.0	192.00

209496 rows × 14 columns

In [5]:

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25
',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:
```
c=b.head(11)
```

Out[6]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	
0	2016-11-01 01:00:00	192.0	0.7	192.0	192.00	153.0	77.0	192.0	192.0	192.0	7.0	192.00	19
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14
2	2016-11-01 01:00:00	5.9	192.0	7.5	192.00	297.0	139.0	192.0	192.0	192.0	192.0	192.00	26
3	2016-11-01 01:00:00	192.0	1.0	192.0	192.00	154.0	113.0	2.0	192.0	192.0	192.0	192.00	19
4	2016-11-01 01:00:00	192.0	192.0	192.0	192.00	275.0	127.0	2.0	192.0	192.0	18.0	192.00	19
5	2016-11-01 01:00:00	0.9	0.5	0.5	192.00	66.0	82.0	1.0	27.0	192.0	8.0	192.00	6
6	2016-11-01 01:00:00	0.7	0.8	0.4	0.13	57.0	66.0	3.0	23.0	15.0	4.0	1.35	5
7	2016-11-01 01:00:00	192.0	192.0	192.0	192.00	52.0	78.0	1.0	192.0	192.0	192.0	192.00	19
8	2016-11-01 01:00:00	192.0	1.2	192.0	192.00	205.0	85.0	6.0	192.0	192.0	192.0	192.00	19
9	2016-11-01 01:00:00	192.0	0.7	192.0	192.00	114.0	91.0	192.0	37.0	192.0	6.0	192.00	19
10	2016-11-01 01:00:00	2.5	192.0	3.3	192.00	166.0	114.0	192.0	45.0	27.0	8.0	192.00	16

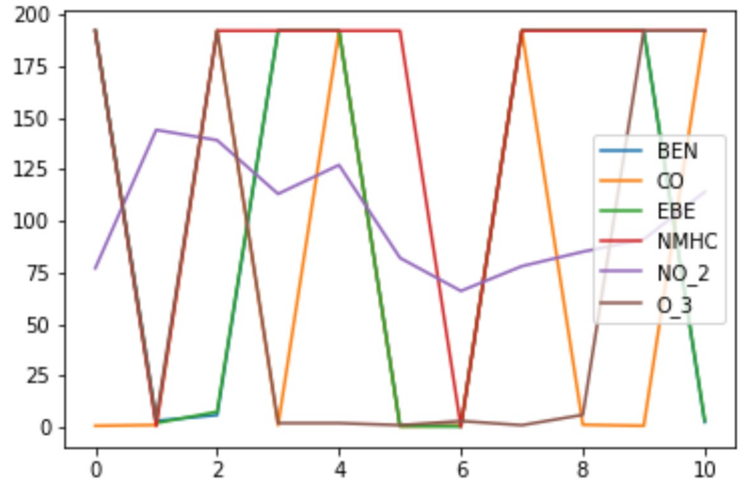
In [7]:
```
d=c[['BEN','CO','EBE','NMHC','NO_2','O_3']]
```

Out[7]:

	BEN	CO	EBE	NMHC	NO_2	O_3
0	192.0	0.7	192.0	192.00	77.0	192.0
1	3.1	1.1	2.0	0.53	144.0	4.0
2	5.9	192.0	7.5	192.00	139.0	192.0
3	192.0	1.0	192.0	192.00	113.0	2.0
4	192.0	192.0	192.0	192.00	127.0	2.0
5	0.9	0.5	0.5	192.00	82.0	1.0
6	0.7	0.8	0.4	0.13	66.0	3.0
7	192.0	192.0	192.0	192.00	78.0	1.0
8	192.0	1.2	192.0	192.00	85.0	6.0
9	192.0	0.7	192.0	192.00	91.0	192.0
10	2.5	192.0	3.3	192.00	114.0	192.0

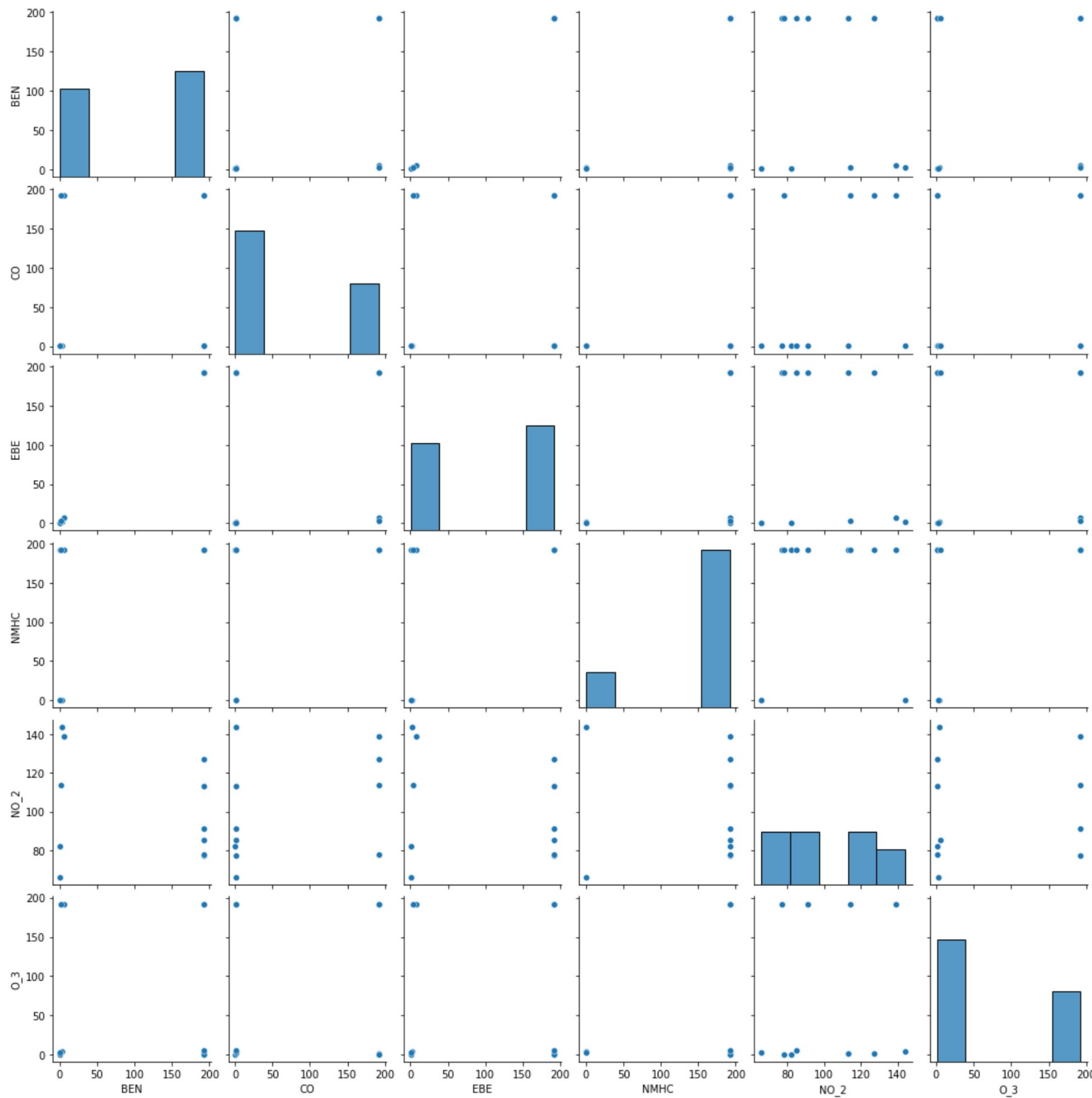
In [8]:
```
d.plot.line()
```

Out[8]: <AxesSubplot:>



In [9]:

Out[9]: <seaborn.axisgrid.PairGrid at 0x2241e53a670>



In [10]:
```
x=d[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2']]
```

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

Out[12]: LinearRegression()

In [13]:

-2.2737367544323206e-13

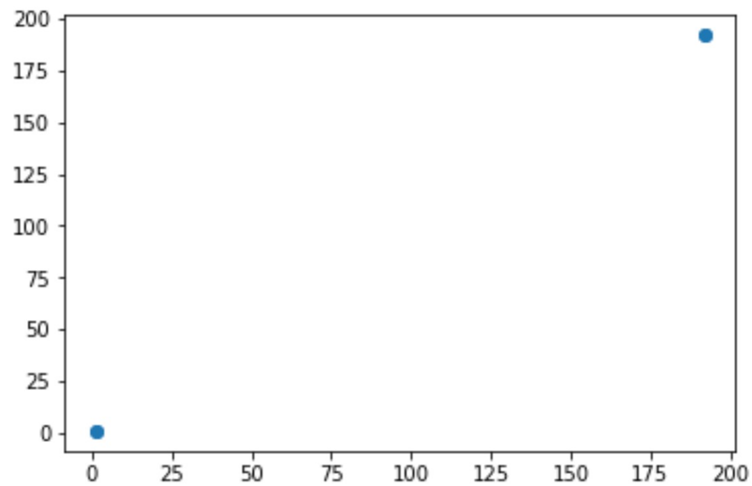
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[14]:

	Co-efficient
BEN	5.344282e-14
CO	1.000000e+00
EBE	-5.238483e-14
NMHC	4.579437e-17
NO_2	8.577800e-16

In [15]:
```
prediction=lr.predict(x_test)
```

Out[15]: <matplotlib.collections.PathCollection at 0x2241f710370>



In [16]:

1.0

In [17]:

In [18]:
```
rr=Ridge(alpha=10)
```

Out[18]: Ridge(alpha=10)

In [19]:

Out[19]: 0.9999905418232936

In [20]:
```
la=Lasso(alpha=10)
```

Out[20]: Lasso(alpha=10)

In [21]:

Out[21]: 0.9999978712598887

In [22]:
```
a1=b.head(6000)
```

Out[22]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH
0	2016-11-01 01:00:00	192.0	0.7	192.0	192.00	153.0	77.0	192.0	192.0	192.0	7.0	192.00
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44
2	2016-11-01 01:00:00	5.9	192.0	7.5	192.00	297.0	139.0	192.0	192.0	192.0	192.0	192.00
3	2016-11-01 01:00:00	192.0	1.0	192.0	192.00	154.0	113.0	2.0	192.0	192.0	192.0	192.00
4	2016-11-01 01:00:00	192.0	192.0	192.0	192.00	275.0	127.0	2.0	192.0	192.0	18.0	192.00
...	...	...	...	...	...	...	...	...	...	...	...	...
5995	2016-11-11 10:00:00	192.0	1.1	192.0	192.00	203.0	111.0	8.0	192.0	192.0	192.0	192.00
5996	2016-11-11 10:00:00	192.0	0.5	192.0	192.00	192.0	192.0	192.0	18.0	192.0	192.0	192.00
5997	2016-11-11 10:00:00	192.0	192.0	192.0	192.00	18.0	15.0	13.0	192.0	192.0	192.0	192.00
5998	2016-11-11 10:00:00	192.0	192.0	192.0	192.00	62.0	40.0	12.0	192.0	192.0	192.0	192.00
5999	2016-11-11 10:00:00	192.0	192.0	192.0	192.00	132.0	86.0	7.0	36.0	192.0	192.0	192.00

6000 rows × 14 columns

In [23]:
```
e=a1[['BEN', 'CO', 'EBE','NMHC', 'NO_2','O_3',
```

In [24]:
```
f=e.iloc[:,0:14]
```

In [25]:
```
```

In [26]:
```
logr=LogisticRegression(max_iter=10000)
```

Out[26]: LogisticRegression(max_iter=10000)

In [27]:
```
from sklearn.model_selection import train_test_split
```

In [28]:
```
```

In [29]:
```
prediction=logr.predict(i)
```

[28079059]

In [30]:

Out[30]:
```
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 [31]:

Out[31]:
```
2.42973207230404e-310
```

In [32]:

Out[32]:
```
0.0
```

In [33]:

Out[33]:
```
0.9455555555555556
```

In [34]:
```python
from sklearn.linear_model import ElasticNet
en=ElasticNet()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_d
escent.py:530: ConvergenceWarning: Objective did not converge. You might want
to increase the number of iterations. Duality gap: 7.271092215353746, toleran
ce: 5.223610856756517
  model = cd_fast.enet_coordinate_descent(
```

Out[34]:
```
ElasticNet()
```

In [35]:

```
[-3.32031653e-01  9.97103512e-01  3.31185864e-01  1.84669016e-04
 -0.00000000e+00]
```

In [36]:

```
0.12048587498125585
```

In [37]:
```python
prediction=en.predict(x_test)
```

```
0.9999809298885648
```

In [38]:
```python
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

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

In [39]:
```python
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
 }
```

In [40]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
```

Out[40]:
```
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 [41]:

Out[41]:
```
0.9776190476190476
```

In [42]:

In [43]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
```

Out[43]:
```
[Text(1639.125, 2491.5, 'X[3] <= -1.132\ngini = 0.958\nsamples = 2622\nvalue
= [166, 191, 146, 161, 195, 210, 195, 182, 162, 181\n155, 139, 178, 175, 194,
186, 175, 152, 175, 161\n169, 188, 186, 178]'),
 Text(558.0, 2038.5, 'X[8] <= -2.649\ngini = 0.666\nsamples = 340\nvalue =
[0, 191, 0, 0, 0, 0, 194, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 175, 0, 0, 0, 0,
0]'),
 Text(279.0, 1585.5, 'X[10] <= 0.099\ngini = 0.493\nsamples = 201\nvalue =
[0, 0, 0, 0, 0, 0, 189, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 148, 0, 0, 0, 0,
0]'),
 Text(139.5, 1132.5, 'gini = 0.0\nsamples = 114\nvalue = [0, 0, 0, 0, 0, 0, 1
89, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(418.5, 1132.5, 'gini = 0.0\nsamples = 87\nvalue = [0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 148, 0, 0, 0, 0, 0]'),
 Text(837.0, 1585.5, 'X[10] <= -1.235\ngini = 0.251\nsamples = 139\nvalue =
[0, 191, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 27, 0, 0, 0, 0,
0]'),
 Text(697.5, 1132.5, 'gini = 0.0\nsamples = 120\nvalue = [0, 191, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(976.5, 1132.5, 'gini = 0.264\nsamples = 19\nvalue = [0, 0, 0, 0, 0, 0,
```

From this observation I had observe that the LASSO is a highest accuracy of 0.9999978712598887

In [ ]: