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	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.
...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.

230568 rows × 17 columns

In [3]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230568 entries, 0 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     230568 non-null  object 
 1   BEN      73979 non-null   float64
 2   CO       211665 non-null  float64
 3   EBE      73948 non-null   float64
 4   MXY      33422 non-null   float64
 5   NMHC     90829 non-null   float64
 6   NO_2     228855 non-null  float64
 7   NOx      228855 non-null  float64
 8   OXY      33472 non-null   float64
 9   O_3      216511 non-null  float64
 10  PM10     227469 non-null  float64
 11  PM25     61758 non-null   float64
 12  PXY      33447 non-null   float64
 13  SO_2     229125 non-null  float64
 14  TCH      90887 non-null   float64
 15  TOL      73840 non-null   float64
 16  station  230568 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.9+ MB
```

In [4]:
```python
b=a.fillna(value=124)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
0	2006-02-01 01:00:00	124.00	1.84	124.00	124.00	124.00	155.100006	490.100006	124.00	4.8
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.1
2	2006-02-01 01:00:00	124.00	1.25	124.00	124.00	124.00	66.800003	192.000000	124.00	4.4
3	2006-02-01 01:00:00	124.00	1.68	124.00	124.00	124.00	103.000000	407.799988	124.00	4.8
4	2006-02-01 01:00:00	124.00	1.31	124.00	124.00	124.00	105.400002	269.200012	124.00	6.9
...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	124.00	0.20	112.500000	218.000000	124.00	24.3
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.4
230565	2006-05-01 00:00:00	0.96	124.00	0.69	124.00	0.19	135.100006	179.199997	124.00	11.4
230566	2006-05-01 00:00:00	0.50	124.00	0.67	124.00	0.10	82.599998	105.599998	124.00	124.0
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.7

230568 rows × 17 columns

In [5]:

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3
',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

Out[6]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2006-02-01 01:00:00	124.00	1.84	124.00	124.000000	124.00	155.100006	490.100006	124.00	4.88	
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.360000	0.32	94.339996	229.699997	3.04	7.10	
2	2006-02-01 01:00:00	124.00	1.25	124.00	124.000000	124.00	66.800003	192.000000	124.00	4.43	
3	2006-02-01 01:00:00	124.00	1.68	124.00	124.000000	124.00	103.000000	407.799988	124.00	4.83	
4	2006-02-01 01:00:00	124.00	1.31	124.00	124.000000	124.00	105.400002	269.200012	124.00	6.99	
5	2006-02-01 01:00:00	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.99	
6	2006-02-01 01:00:00	124.00	1.28	124.00	124.000000	0.57	94.320000	294.000000	124.00	6.77	
7	2006-02-01 01:00:00	0.27	1.51	0.28	124.000000	0.46	144.699997	385.299988	124.00	5.30	
8	2006-02-01 01:00:00	124.00	2.65	124.00	124.000000	124.00	197.100006	673.099976	124.00	2.64	1
9	2006-02-01 01:00:00	124.00	1.30	124.00	124.000000	124.00	130.899994	282.000000	124.00	5.14	
10	2006-02-01 01:00:00	124.00	1.48	124.00	124.000000	0.50	75.260002	248.899994	124.00	2.20	

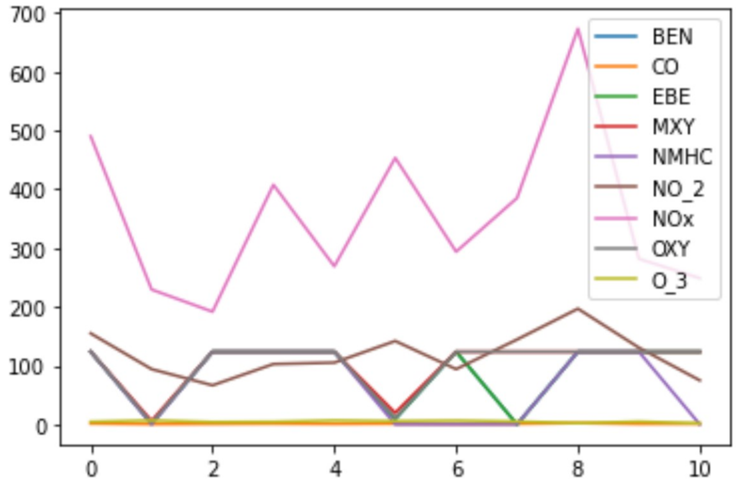
In [7]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3']]
```

Out[7]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	124.00	1.84	124.00	124.000000	124.00	155.100006	490.100006	124.00	4.88
1	1.68	1.01	2.38	6.360000	0.32	94.339996	229.699997	3.04	7.10
2	124.00	1.25	124.00	124.000000	124.00	66.800003	192.000000	124.00	4.43
3	124.00	1.68	124.00	124.000000	124.00	103.000000	407.799988	124.00	4.83
4	124.00	1.31	124.00	124.000000	124.00	105.400002	269.200012	124.00	6.99
5	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.99
6	124.00	1.28	124.00	124.000000	0.57	94.320000	294.000000	124.00	6.77
7	0.27	1.51	0.28	124.000000	0.46	144.699997	385.299988	124.00	5.30
8	124.00	2.65	124.00	124.000000	124.00	197.100006	673.099976	124.00	2.64
9	124.00	1.30	124.00	124.000000	124.00	130.899994	282.000000	124.00	5.14
10	124.00	1.48	124.00	124.000000	0.50	75.260002	248.899994	124.00	2.20

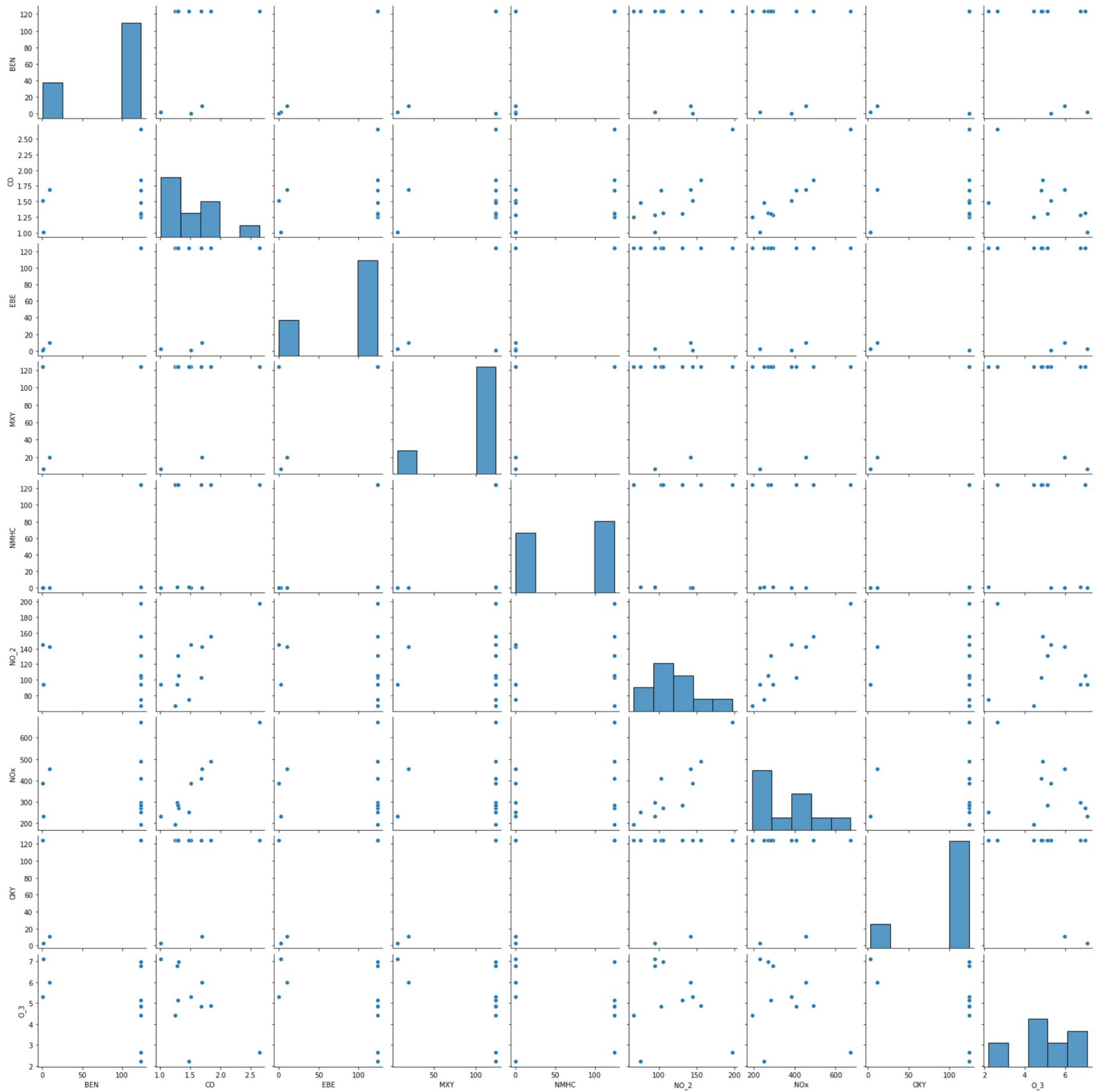
In [8]:
```
df.plot.line()
```

Out[8]: <AxesSubplot:>



In [9]:

Out[9]: <seaborn.axisgrid.PairGrid at 0x182812a9280>



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

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]:

-1.7053025658242404e-13

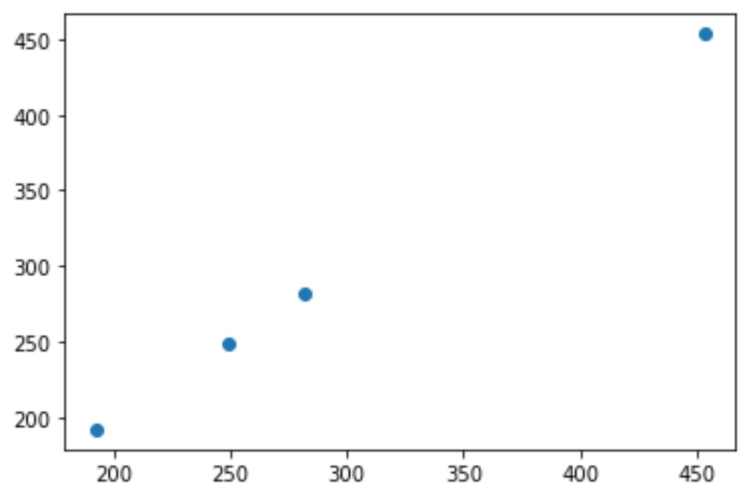
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[14]:

	Co-efficient
BEN	-5.592458e-17
CO	2.145161e-13
EBE	-1.263043e-16
MXY	2.236237e-16
NMHC	2.732046e-16
NO_2	-1.270519e-15
NOx	1.000000e+00
OXY	4.694919e-16

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x18288a9c790>
```



In [16]:

```
1.0
```

In [17]:

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

Out[18]:
```
Ridge(alpha=10)
```

In [19]:

Out[19]:
```
0.9999995223798749
```

In [20]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[20]:
```
Lasso(alpha=10)
```

In [21]:

Out[21]: 0.9999994937723611

In [22]:
```python
a1=b.head(5000)
```

Out[22]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2006-02-01 01:00:00	124.00	1.84	124.00	124.00	124.00	155.100006	490.100006	124.00	4.88	97.
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.10	25.
2	2006-02-01 01:00:00	124.00	1.25	124.00	124.00	124.00	66.800003	192.000000	124.00	4.43	34.
3	2006-02-01 01:00:00	124.00	1.68	124.00	124.00	124.00	103.000000	407.799988	124.00	4.83	28.
4	2006-02-01 01:00:00	124.00	1.31	124.00	124.00	124.00	105.400002	269.200012	124.00	6.99	54.
...	...	...	...	...	...	...	...	...	...	...	
4995	2006-02-09 01:00:00	124.00	1.52	124.00	124.00	124.00	92.889999	317.600006	124.00	4.27	40.
4996	2006-02-09 01:00:00	124.00	1.67	124.00	124.00	124.00	133.399994	392.000000	124.00	7.41	97.
4997	2006-02-09 01:00:00	6.72	1.96	7.57	15.48	0.55	171.300003	544.700012	7.96	5.94	92.
4998	2006-02-09 01:00:00	124.00	1.43	124.00	124.00	0.67	108.199997	353.799988	124.00	7.08	73.
4999	2006-02-09 01:00:00	0.35	1.60	0.46	124.00	0.52	160.600006	438.299988	124.00	5.30	90.

5000 rows × 17 columns

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

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

In [25]:

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

Out[26]: LogisticRegression(max_iter=10000)

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

In [28]:

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

```
[28079018]
```

In [30]:

Out[30]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079026, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

In [31]:

Out[31]:
```
1.5038383254433008e-81
```

In [32]:

Out[32]:
```
3.4251164754248804e-124
```

In [33]:

Out[33]:
```
0.5446666666666666
```

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

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

In [35]:

```
[-1.06766919e-04 -0.00000000e+00 -0.00000000e+00  4.70974802e-01
  5.23905245e-05  0.00000000e+00  1.00000687e+00 -4.57597899e-01]
```

In [36]:

```
-1.6478928724587263
```

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

```
0.9998250049427502
```

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

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

In [39]:
```
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]:
```
0.5660000000000001
```

In [42]:

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

Out[43]:
```
[Text(2371.5, 2491.5, 'X[7] <= -0.802\ngini = 0.961\nsamples = 2212\nvalue =
[157, 121, 140, 146, 136, 141, 118, 122, 130, 110\n124, 158, 149, 148, 153, 1
31, 119, 131, 166, 136\n129, 136, 141, 120, 104, 134]'),
 Text(1275.4285714285713, 2038.5, 'X[5] <= -0.21\ngini = 0.749\nsamples = 33
7\nvalue = [0, 0, 0, 146, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 131, 0, 0, 1
29, 0, 0, 0, 0, 134]'),
 Text(637.7142857142857, 1585.5, 'X[10] <= -2.382\ngini = 0.644\nsamples = 14
4\nvalue = [0, 0, 0, 38, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 125, 0, 0, 2
6, 0, 0, 0, 0, 49]'),
 Text(318.85714285714283, 1132.5, 'X[12] <= -1.175\ngini = 0.498\nsamples = 9
0\nvalue = [0, 0, 0, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 109, 0, 0, 1
5, 0, 0, 0, 0, 25]'),
 Text(159.42857142857142, 679.5, 'X[8] <= 0.493\ngini = 0.117\nsamples = 12\n
value = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 15, 0,
0, 0, 0, 0]'),
 Text(79.71428571428571, 226.5, 'gini = 0.278\nsamples = 5\nvalue = [0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0]'),
 Text(239.1428571428571, 226.5, 'gini = 0.0\nsamples = 7\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0]'),
```

From this observation I had observe that the LASSO is a highest accuracy of 0.9999994937723611

In [ ]: