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	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.2
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.3
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.2
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.8
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.7
...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.3
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.4
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.8
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.5
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.3

243984 rows × 16 columns

In [3]:

```
.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243984 entries, 0 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     243984 non-null  object 
 1   BEN      69745 non-null   float64
 2   CO       225340 non-null  float64
 3   EBE      61244 non-null   float64
 4   MXY      42045 non-null   float64
 5   NMHC     111951 non-null  float64
 6   NO_2     242625 non-null  float64
 7   NOx      242629 non-null  float64
 8   OXY      42072 non-null   float64
 9   O_3      234131 non-null  float64
 10  PM10     240896 non-null  float64
 11  PXY      42063 non-null   float64
 12  SO_2     242729 non-null  float64
 13  TCH      111991 non-null  float64
 14  TOL      69439 non-null   float64
 15  station  243984 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 29.8+ MB
```

In [4]:
```
b=a.fillna(value=126)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
0	2003-03-01 01:00:00	126.00	1.72	126.00	126.00	126.00	73.900002	316.299988	126.00	10.5
1	2003-03-01 01:00:00	126.00	1.45	126.00	126.00	0.26	72.110001	250.000000	0.73	6.7
2	2003-03-01 01:00:00	126.00	1.57	126.00	126.00	126.00	80.559998	224.199997	126.00	21.0
3	2003-03-01 01:00:00	126.00	2.45	126.00	126.00	126.00	78.370003	450.399994	126.00	4.2
4	2003-03-01 01:00:00	126.00	3.26	126.00	126.00	126.00	96.250000	479.100006	126.00	8.4
...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.0
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	126.00	10.450000	14.760000	1.00	34.6
243981	2003-10-01 00:00:00	126.00	126.00	126.00	126.00	0.07	34.639999	50.810001	126.00	32.1
243982	2003-10-01 00:00:00	126.00	126.00	126.00	126.00	0.07	32.580002	41.020000	126.00	126.0
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.4

243984 rows × 16 columns

In [5]:

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3
',
       'PM10', '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_
0	2003-03-01 01:00:00	126.00	1.72	126.00	126.00	126.00	73.900002	316.299988	126.00	10.55000
1	2003-03-01 01:00:00	126.00	1.45	126.00	126.00	0.26	72.110001	250.000000	0.73	6.72000
2	2003-03-01 01:00:00	126.00	1.57	126.00	126.00	126.00	80.559998	224.199997	126.00	21.04999
3	2003-03-01 01:00:00	126.00	2.45	126.00	126.00	126.00	78.370003	450.399994	126.00	4.22000
4	2003-03-01 01:00:00	126.00	3.26	126.00	126.00	126.00	96.250000	479.100006	126.00	8.46000
5	2003-03-01 01:00:00	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.95000
6	2003-03-01 01:00:00	126.00	1.38	126.00	126.00	0.29	89.580002	230.000000	126.00	7.20000
7	2003-03-01 01:00:00	126.00	1.58	126.00	126.00	0.30	93.639999	334.600006	126.00	4.19000
8	2003-03-01 01:00:00	126.00	126.00	126.00	126.00	126.00	126.000000	126.000000	126.00	126.00000
9	2003-03-01 01:00:00	126.00	1.92	126.00	126.00	126.00	71.839996	181.399994	126.00	5.33000
10	2003-03-01 01:00:00	126.00	1.33	126.00	126.00	0.31	87.919998	273.399994	126.00	4.25000

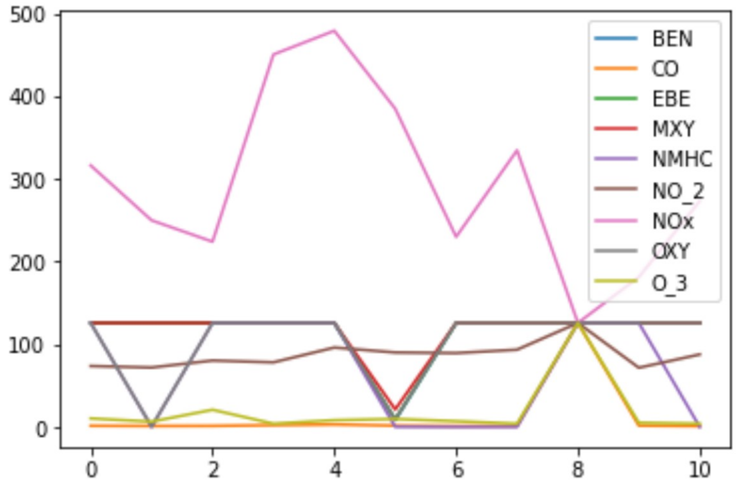
In [7]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3']]
```

Out[7]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	126.00	1.72	126.00	126.00	126.00	73.900002	316.299988	126.00	10.550000
1	126.00	1.45	126.00	126.00	0.26	72.110001	250.000000	0.73	6.720000
2	126.00	1.57	126.00	126.00	126.00	80.559998	224.199997	126.00	21.049999
3	126.00	2.45	126.00	126.00	126.00	78.370003	450.399994	126.00	4.220000
4	126.00	3.26	126.00	126.00	126.00	96.250000	479.100006	126.00	8.460000
5	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.950000
6	126.00	1.38	126.00	126.00	0.29	89.580002	230.000000	126.00	7.200000
7	126.00	1.58	126.00	126.00	0.30	93.639999	334.600006	126.00	4.190000
8	126.00	126.00	126.00	126.00	126.00	126.000000	126.000000	126.00	126.000000
9	126.00	1.92	126.00	126.00	126.00	71.839996	181.399994	126.00	5.330000
10	126.00	1.33	126.00	126.00	0.31	87.919998	273.399994	126.00	4.250000

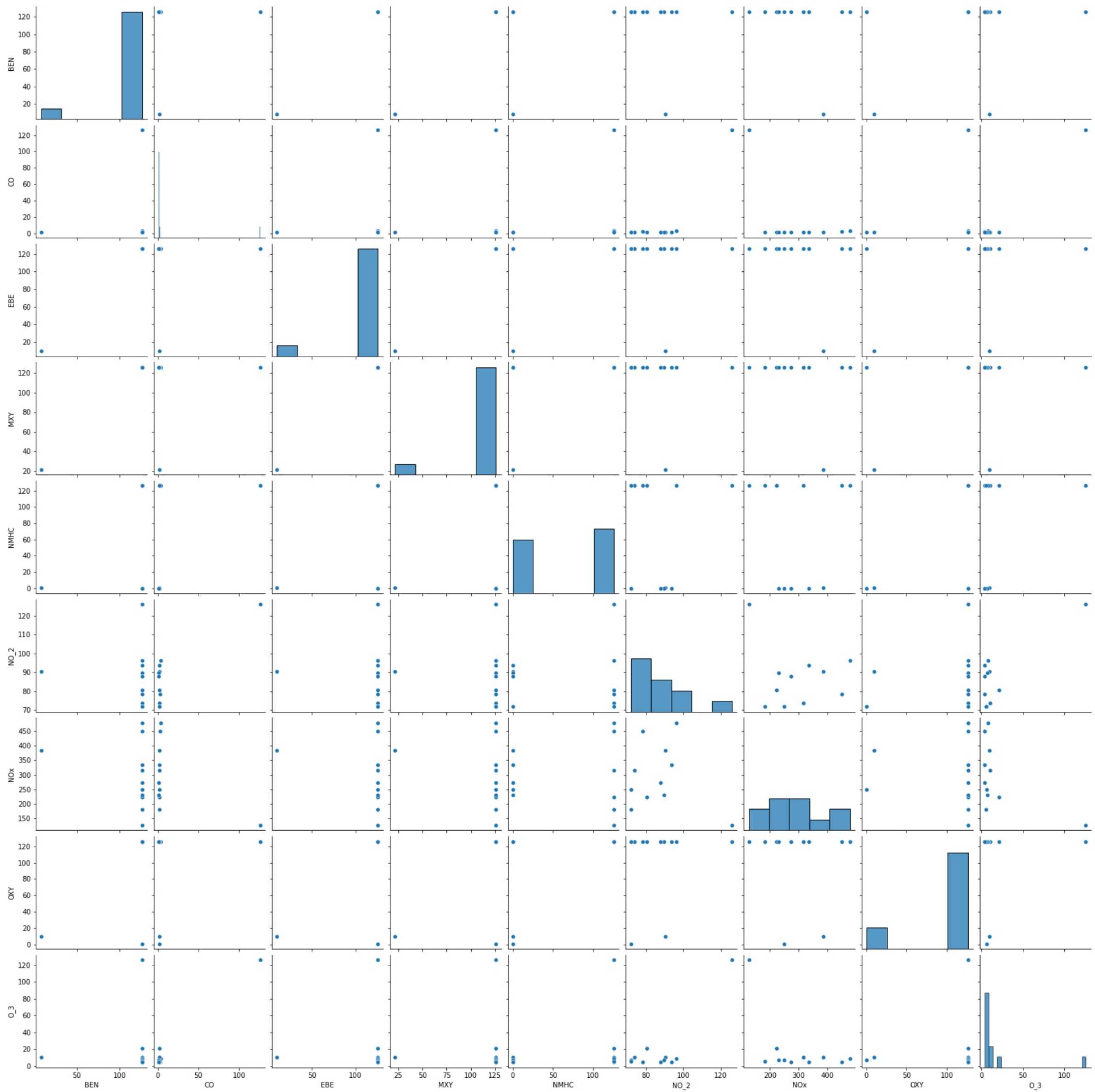
In [8]:
```
d.plot.line()
```

Out[8]: <AxesSubplot:>



In [9]:

Out[9]: <seaborn.axisgrid.PairGrid at 0x1fdea615250>



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]:

```
1.0231815394945443e-12
```

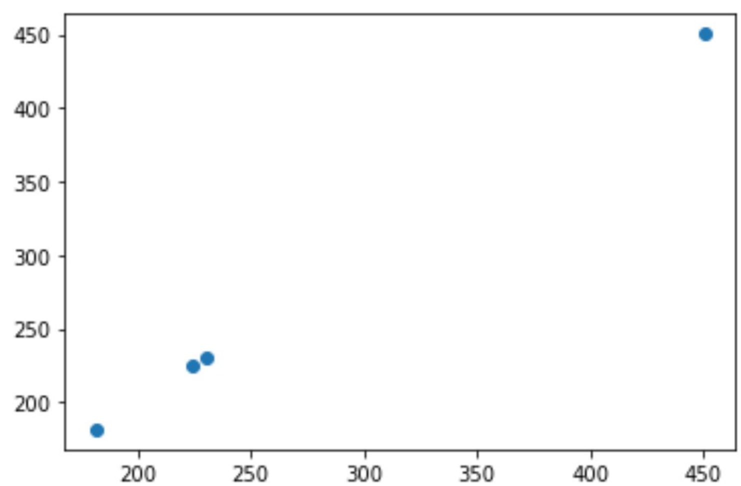
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[14]:

	Co-efficient
BEN	-1.695639e-16
CO	1.653345e-14
EBE	-3.975580e-16
MXY	4.284913e-16
NMHC	-4.011195e-15
NO_2	-2.801221e-14
NOx	1.000000e+00
OXY	2.871689e-15

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

Out[15]: <matplotlib.collections.PathCollection at 0x1fdef63b220>



In [16]:

```
1.0
```

In [17]:

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

Out[18]: Ridge(alpha=10)

In [19]:

Out[19]: 0.9998869821152107

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

Out[20]: Lasso(alpha=10)

In [21]:

Out[21]: 0.9999989799894025

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

Out[22]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2003-03-01 01:00:00	126.00	1.72	126.00	126.0	126.00	73.900002	316.299988	126.00	10.550000
1	2003-03-01 01:00:00	126.00	1.45	126.00	126.0	0.26	72.110001	250.000000	0.73	6.720000
2	2003-03-01 01:00:00	126.00	1.57	126.00	126.0	126.00	80.559998	224.199997	126.00	21.049999
3	2003-03-01 01:00:00	126.00	2.45	126.00	126.0	126.00	78.370003	450.399994	126.00	4.220000
4	2003-03-01 01:00:00	126.00	3.26	126.00	126.0	126.00	96.250000	479.100006	126.00	8.460000
...	...	...	...	...	...	...	...	...	...	...
4995	2003-03-08 11:00:00	126.00	0.46	126.00	126.0	126.00	27.469999	36.110001	126.00	43.799999
4996	2003-03-08 11:00:00	126.00	0.69	126.00	126.0	0.07	39.020000	56.230000	126.00	39.720001
4997	2003-03-08 11:00:00	126.00	1.74	126.00	126.0	126.00	65.790001	148.500000	126.00	23.760000
4998	2003-03-08 11:00:00	1.85	0.56	1.62	126.0	0.13	52.610001	96.300003	126.00	28.770000
4999	2003-03-08 11:00:00	126.00	0.84	126.00	126.0	126.00	33.310001	41.720001	126.00	38.459999

5000 rows × 16 columns

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

```
[28079003]
```

In [30]:

Out[30]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079025, 28079026, 28079027, 28079035, 28079036,
       28079038, 28079039, 28079040, 28079099], dtype=int64)
```

In [31]:

Out[31]:
```
9.896956293105757e-13
```

In [32]:

Out[32]:
```
0.9999999999683131
```

In [33]:

Out[33]:
```
0.594
```

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

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

In [35]:

```
[-0.         -0.         -0.         -0.         -0.         -0.
  0.99990495 -0.        ]
```

In [36]:

```
0.029388644480661696
```

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

```
0.9999999898008636
```

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]:
```
0.5691428571428572
```

In [42]:

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

Out[43]:
```
[Text(1900.2162162162163, 2491.5, 'X[0] <= -0.355\ngini = 0.964\nsamples = 22
24\nvalue = [126, 136, 138, 119, 113, 99, 117, 120, 124, 137\n106, 131, 127,
135, 114, 127, 146, 129, 108, 109\n127, 103, 129, 142, 147, 131, 123, 137]'),
 Text(573.081081081081, 2038.5, 'X[11] <= -0.864\ngini = 0.872\nsamples = 59
1\nvalue = [0, 0, 0, 119, 0, 0, 0, 0, 0, 0, 106, 0, 0, 0\n0, 0, 137, 129, 10
8, 109, 0, 0, 75, 0, 0, 0, 0\n136]'),
 Text(241.2972972972973, 1585.5, 'X[10] <= -2.287\ngini = 0.157\nsamples = 4
2\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 6, 64, 0, 0,
0, 0, 0, 0, 0, 0, 0]'),
 Text(120.64864864864865, 1132.5, 'gini = 0.0\nsamples = 37\nvalue = [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 62, 0, 0, 0, 0, 0, 0, 0, 0,
0]'),
 Text(361.94594594594594, 1132.5, 'gini = 0.375\nsamples = 5\nvalue = [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 6, 2, 0, 0, 0, 0, 0, 0, 0, 0,
0]'),
 Text(904.8648648648648, 1585.5, 'X[7] <= -0.661\ngini = 0.865\nsamples = 54
9\nvalue = [0, 0, 0, 119, 0, 0, 0, 0, 0, 0, 106, 0, 0, 0\n0, 0, 137, 123, 44,
109, 0, 0, 75, 0, 0, 0, 0\n136]'),
 Text(603.2432432432432, 1132.5, 'X[4] <= -0.069\ngini = 0.777\nsamples = 31
```

From this observation I had observe that the ELASTICNET highest accuracy of 0.9999999898008636

In [ ]: