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	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.

226392 rows × 17 columns

```
In [3]: info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 226392 entries, 0 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     226392 non-null  object 
 1   BEN      67047 non-null   float64
 2   CO       208109 non-null  float64
 3   EBE      67044 non-null   float64
 4   MXY      25867 non-null   float64
 5   NMHC     85079 non-null   float64
 6   NO_2     225315 non-null  float64
 7   NOx      225311 non-null  float64
 8   OXY      25878 non-null   float64
 9   O_3      215716 non-null  float64
 10  PM10     220179 non-null  float64
 11  PM25     67833 non-null   float64
 12  PXY      25877 non-null   float64
 13  SO_2     225405 non-null  float64
 14  TCH      85107 non-null   float64
 15  TOL      66940 non-null   float64
 16  station  226392 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.4+ MB
```

In [4]:

```
b=a.fillna(value=119)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
0	2008-06-01 01:00:00	119.00	0.47	119.00	119.00	119.00	83.089996	120.699997	119.00	16.9
1	2008-06-01 01:00:00	119.00	0.59	119.00	119.00	119.00	94.820000	130.399994	119.00	17.4
2	2008-06-01 01:00:00	119.00	0.55	119.00	119.00	119.00	75.919998	104.599998	119.00	13.4
3	2008-06-01 01:00:00	119.00	0.36	119.00	119.00	119.00	61.029999	66.559998	119.00	23.1
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.1
...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.4
226388	2008-11-01 00:00:00	119.00	0.30	119.00	119.00	119.00	41.880001	48.500000	119.00	35.8
226389	2008-11-01 00:00:00	0.25	119.00	0.56	119.00	0.11	83.610001	102.199997	119.00	14.1
226390	2008-11-01 00:00:00	0.54	119.00	2.70	119.00	0.18	70.639999	81.860001	119.00	119.0
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.9

226392 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	2008-06-01 01:00:00	119.00	0.47	119.00	119.00	119.00	83.089996	120.699997	119.00	16.990000
1	2008-06-01 01:00:00	119.00	0.59	119.00	119.00	119.00	94.820000	130.399994	119.00	17.469999
2	2008-06-01 01:00:00	119.00	0.55	119.00	119.00	119.00	75.919998	104.599998	119.00	13.470000
3	2008-06-01 01:00:00	119.00	0.36	119.00	119.00	119.00	61.029999	66.559998	119.00	23.110001
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000
5	2008-06-01 01:00:00	119.00	0.47	119.00	119.00	0.22	67.820000	101.099998	119.00	20.610001
6	2008-06-01 01:00:00	0.17	0.40	0.44	119.00	0.15	72.639999	91.220001	119.00	17.040001
7	2008-06-01 01:00:00	119.00	0.51	119.00	119.00	119.00	80.440002	141.500000	119.00	10.310000
8	2008-06-01 01:00:00	119.00	0.36	119.00	119.00	119.00	68.150002	85.639999	119.00	23.580000
9	2008-06-01 01:00:00	119.00	0.18	119.00	119.00	0.16	58.330002	64.769997	119.00	35.060001
10	2008-06-01 01:00:00	119.00	0.45	119.00	119.00	119.00	53.700001	66.610001	119.00	27.180000

In [7]:

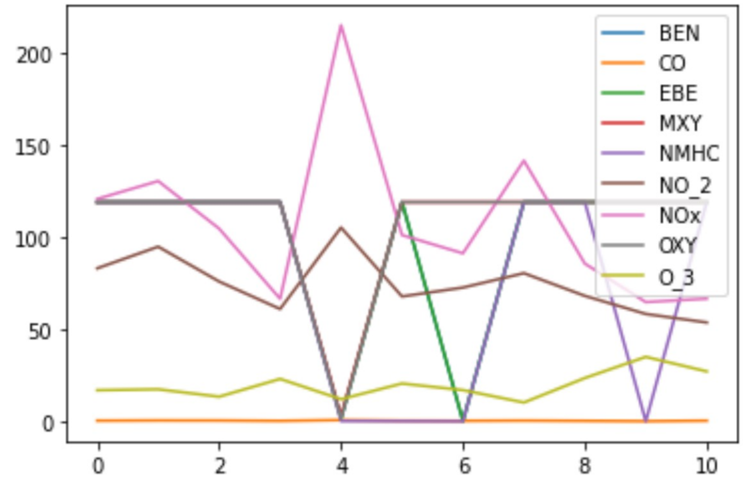
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3']]
```

Out[7]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	119.00	0.47	119.00	119.00	119.00	83.089996	120.699997	119.00	16.990000
1	119.00	0.59	119.00	119.00	119.00	94.820000	130.399994	119.00	17.469999
2	119.00	0.55	119.00	119.00	119.00	75.919998	104.599998	119.00	13.470000
3	119.00	0.36	119.00	119.00	119.00	61.029999	66.559998	119.00	23.110001
4	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000
5	119.00	0.47	119.00	119.00	0.22	67.820000	101.099998	119.00	20.610001
6	0.17	0.40	0.44	119.00	0.15	72.639999	91.220001	119.00	17.040001
7	119.00	0.51	119.00	119.00	119.00	80.440002	141.500000	119.00	10.310000
8	119.00	0.36	119.00	119.00	119.00	68.150002	85.639999	119.00	23.580000
9	119.00	0.18	119.00	119.00	0.16	58.330002	64.769997	119.00	35.060001
10	119.00	0.45	119.00	119.00	119.00	53.700001	66.610001	119.00	27.180000

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

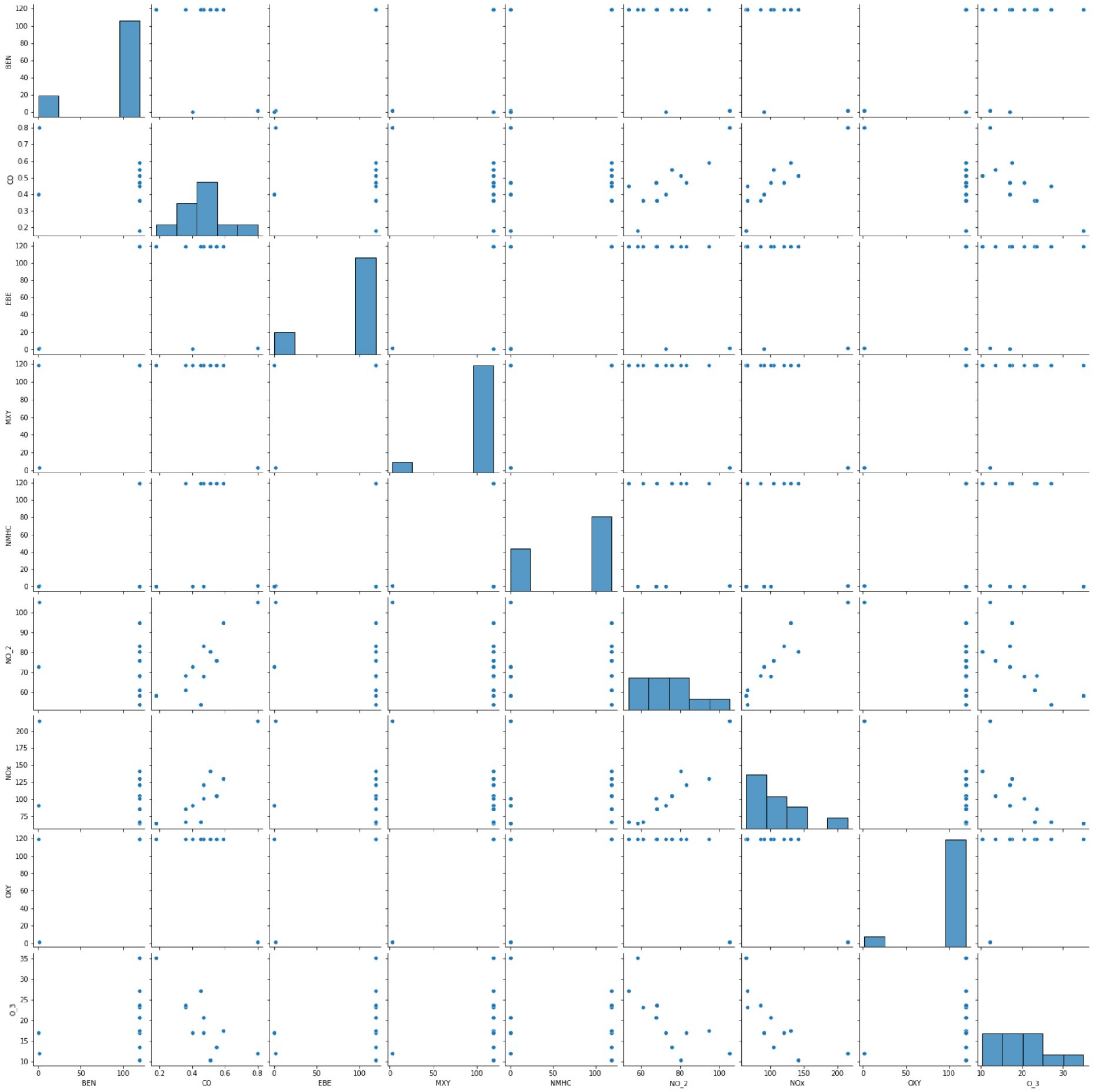
Out[8]:
```
<AxesSubplot:>
```



In [9]:

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x11e6de54280>
```



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]:

```
-7.105427357601002e-14
```

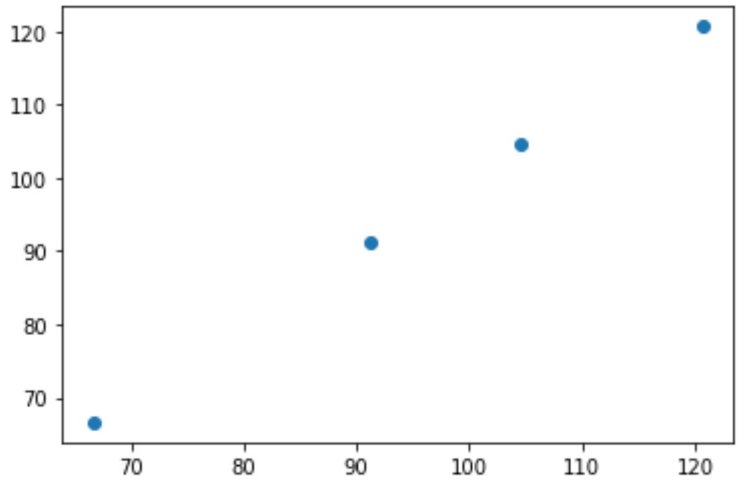
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[14]:

	Co-efficient
BEN	2.989965e-16
CO	9.585229e-14
EBE	-6.116106e-16
MXY	-5.266844e-16
NMHC	2.110477e-15
NO_2	7.048595e-16
NOx	1.000000e+00
OXY	-5.520637e-16

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x11e723ed4c0>
```



In [16]:

```
1.0
```

In [17]:

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

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

In [19]:

Out[19]:
```
0.99994569566436
```

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

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

In [21]:

Out[21]:
```
0.9999665160868221
```

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

Out[22]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_
0	2008-06-01 01:00:00	119.00	0.47	119.0	119.00	119.0	83.089996	120.699997	119.00	16.9900
1	2008-06-01 01:00:00	119.00	0.59	119.0	119.00	119.0	94.820000	130.399994	119.00	17.4699
2	2008-06-01 01:00:00	119.00	0.55	119.0	119.00	119.0	75.919998	104.599998	119.00	13.4700
3	2008-06-01 01:00:00	119.00	0.36	119.0	119.00	119.0	61.029999	66.559998	119.00	23.1100
4	2008-06-01 01:00:00	1.68	0.80	1.7	3.01	0.3	105.199997	214.899994	1.61	12.1200
...	...	...	...	...	...	...	...	...	...	
5995	2008-06-10 15:00:00	119.00	0.27	119.0	119.00	119.0	32.000000	35.639999	119.00	67.9800
5996	2008-06-10 15:00:00	119.00	0.32	119.0	119.00	119.0	45.299999	57.360001	119.00	59.1100
5997	2008-06-10 15:00:00	119.00	0.24	119.0	119.00	119.0	26.160000	39.930000	119.00	52.2000
5998	2008-06-10 15:00:00	119.00	119.00	119.0	119.00	119.0	119.000000	119.000000	119.00	119.0000
5999	2008-06-10 15:00:00	119.00	0.31	119.0	119.00	119.0	68.599998	94.559998	119.00	42.7200

6000 rows × 17 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,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079026, 28079027, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

In [31]:

Out[31]:
```
4.63437247861083e-27
```

In [32]:

Out[32]:
```
0.7955388500721003
```

In [33]:

Out[33]:
```
0.575
```

In [34]:
```
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: 8.3059915445693, toleranc
e: 1.6844122929515772
  model = cd_fast.enet_coordinate_descent(
```

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

In [35]:

```
[-0.0802347   0.          -0.12234881 -0.02045458 -0.           0.
  0.99934409  0.22163918]
```

In [36]:

```
0.22489545373029785
```

In [37]:
```
prediction=en.predict(x_test)
```
```
0.6320535313373075
```

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]:

```
0.6535714285714286
```

In [42]:

In [43]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,50))
```

Out[43]:

```
[Text(2178.346153846154, 2491.5, 'X[13] <= -0.225\ngini = 0.961\nsamples = 26
23\nvalue = [160, 159, 143, 166, 159, 171, 177, 179, 185, 125\n158, 154, 152,
166, 148, 185, 153, 188, 160, 151\n179, 143, 153, 163, 163, 160]'),
 Text(1244.7692307692307, 2038.5, 'X[11] <= -0.06\ngini = 0.874\nsamples = 81
9\nvalue = [0, 0, 0, 164, 0, 169, 0, 0, 0, 0, 157, 0, 0\n0, 0, 0, 153, 188,
0, 151, 179, 0, 0, 0, 0\n160]'),
 Text(686.7692307692307, 1585.5, 'X[5] <= -0.845\ngini = 0.763\nsamples = 37
4\nvalue = [0, 0, 0, 27, 0, 143, 0, 0, 0, 0, 1, 0, 0, 0\n0, 0, 143, 176, 0,
0, 0, 0, 0, 0, 101]'),
 Text(343.38461538461536, 1132.5, 'X[2] <= -1.505\ngini = 0.44\nsamples = 13
0\nvalue = [0, 0, 0, 15, 0, 5, 0, 0, 0, 0, 1, 0, 0, 0\n0, 0, 10, 149, 0, 0,
0, 0, 0, 0, 0, 23]'),
 Text(171.69230769230768, 679.5, 'X[8] <= 0.585\ngini = 0.727\nsamples = 52\n
value = [0, 0, 0, 12, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 8, 30, 0, 0, 0, 0,
0, 0, 0, 23]'),
 Text(85.84615384615384, 226.5, 'gini = 0.728\nsamples = 35\nvalue = [0, 0,
0, 12, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 5, 10, 0, 0, 0, 0, 0, 0, 0, 23]'),
 Text(257.53846153846155, 226.5, 'gini = 0.227\nsamples = 17\nvalue = [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 3, 20, 0, 0, 0, 0, 0, 0, 0, 0]'),
```

From this observation I had observe that the LASSO is a highest accuracy of 0.9999665160868221

In [ ]: