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	PM1
0	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	14.720000	14.9
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.9
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	30.469999	14.6
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	21.379999	15.1
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	33.410000	5.0
...	...	...	...	...	...	...	...	...	...	...	
236995	2006-01-01 00:00:00	1.08	0.36	1.01	NaN	0.11	21.990000	23.610001	NaN	43.349998	5.0
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.9
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	43.840000	4.3
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	NaN	5.0
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.6

237000 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      70370 non-null   float64
 2   CO       217656 non-null  float64
 3   EBE      68955 non-null   float64
 4   MXY      32549 non-null   float64
 5   NMHC     92854 non-null   float64
 6   NO_2     235022 non-null  float64
 7   NOx      235049 non-null  float64
 8   OXY      32555 non-null   float64
 9   O_3      223162 non-null  float64
 10  PM10     232142 non-null  float64
 11  PM25     69407 non-null   float64
 12  PXY      32549 non-null   float64
 13  SO_2     235277 non-null  float64
 14  TCH      93076 non-null   float64
 15  TOL      70255 non-null   float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ MB
```

In [4]:
```
b=a.fillna(value=96)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2005-11-01 01:00:00	96.00	0.77	96.00	96.00	96.00	57.130001	128.699997	96.00	14.720000
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000
2	2005-11-01 01:00:00	96.00	0.40	96.00	96.00	96.00	46.119999	53.000000	96.00	30.469999
3	2005-11-01 01:00:00	96.00	0.42	96.00	96.00	96.00	37.220001	52.009998	96.00	21.379999
4	2005-11-01 01:00:00	96.00	0.57	96.00	96.00	96.00	32.160000	36.680000	96.00	33.410000
...	...	...	...	...	...	...	...	...	...	...
236995	2006-01-01 00:00:00	1.08	0.36	1.01	96.00	0.11	21.990000	23.610001	96.00	43.349998
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999
236997	2006-01-01 00:00:00	0.19	96.00	0.26	96.00	0.08	26.730000	30.809999	96.00	43.840000
236998	2006-01-01 00:00:00	0.14	96.00	1.00	96.00	0.06	13.770000	17.770000	96.00	96.000000
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998

237000 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	P
0	2005-11-01 01:00:00	96.00	0.77	96.00	96.00	96.00	57.130001	128.699997	96.00	14.720000	14.91(
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.93(
2	2005-11-01 01:00:00	96.00	0.40	96.00	96.00	96.00	46.119999	53.000000	96.00	30.469999	14.60(
3	2005-11-01 01:00:00	96.00	0.42	96.00	96.00	96.00	37.220001	52.009998	96.00	21.379999	15.16(
4	2005-11-01 01:00:00	96.00	0.57	96.00	96.00	96.00	32.160000	36.680000	96.00	33.410000	5.00(
5	2005-11-01 01:00:00	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000	18.07(
6	2005-11-01 01:00:00	96.00	0.55	96.00	96.00	0.27	50.279999	77.209999	96.00	19.120001	18.20(
7	2005-11-01 01:00:00	0.20	0.38	1.00	96.00	0.27	51.759998	72.989998	96.00	14.810000	16.43(
8	2005-11-01 01:00:00	96.00	0.70	96.00	96.00	96.00	39.040001	43.860001	96.00	25.379999	16.13(
9	2005-11-01 01:00:00	96.00	0.56	96.00	96.00	96.00	41.820000	51.869999	96.00	24.290001	7.13(
10	2005-11-01 01:00:00	96.00	0.65	96.00	96.00	0.18	46.040001	76.610001	96.00	34.750000	10.53(

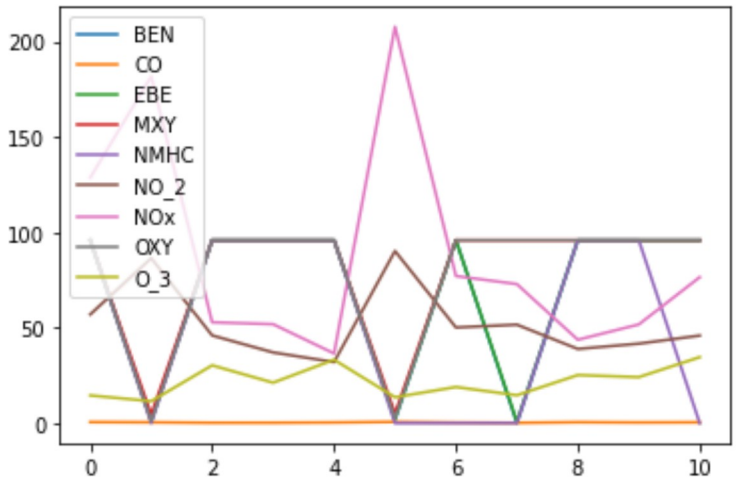
In [7]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3']]
```

Out[7]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	96.00	0.77	96.00	96.00	96.00	57.130001	128.699997	96.00	14.720000
1	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000
2	96.00	0.40	96.00	96.00	96.00	46.119999	53.000000	96.00	30.469999
3	96.00	0.42	96.00	96.00	96.00	37.220001	52.009998	96.00	21.379999
4	96.00	0.57	96.00	96.00	96.00	32.160000	36.680000	96.00	33.410000
5	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000
6	96.00	0.55	96.00	96.00	0.27	50.279999	77.209999	96.00	19.120001
7	0.20	0.38	1.00	96.00	0.27	51.759998	72.989998	96.00	14.810000
8	96.00	0.70	96.00	96.00	96.00	39.040001	43.860001	96.00	25.379999
9	96.00	0.56	96.00	96.00	96.00	41.820000	51.869999	96.00	24.290001
10	96.00	0.65	96.00	96.00	0.18	46.040001	76.610001	96.00	34.750000

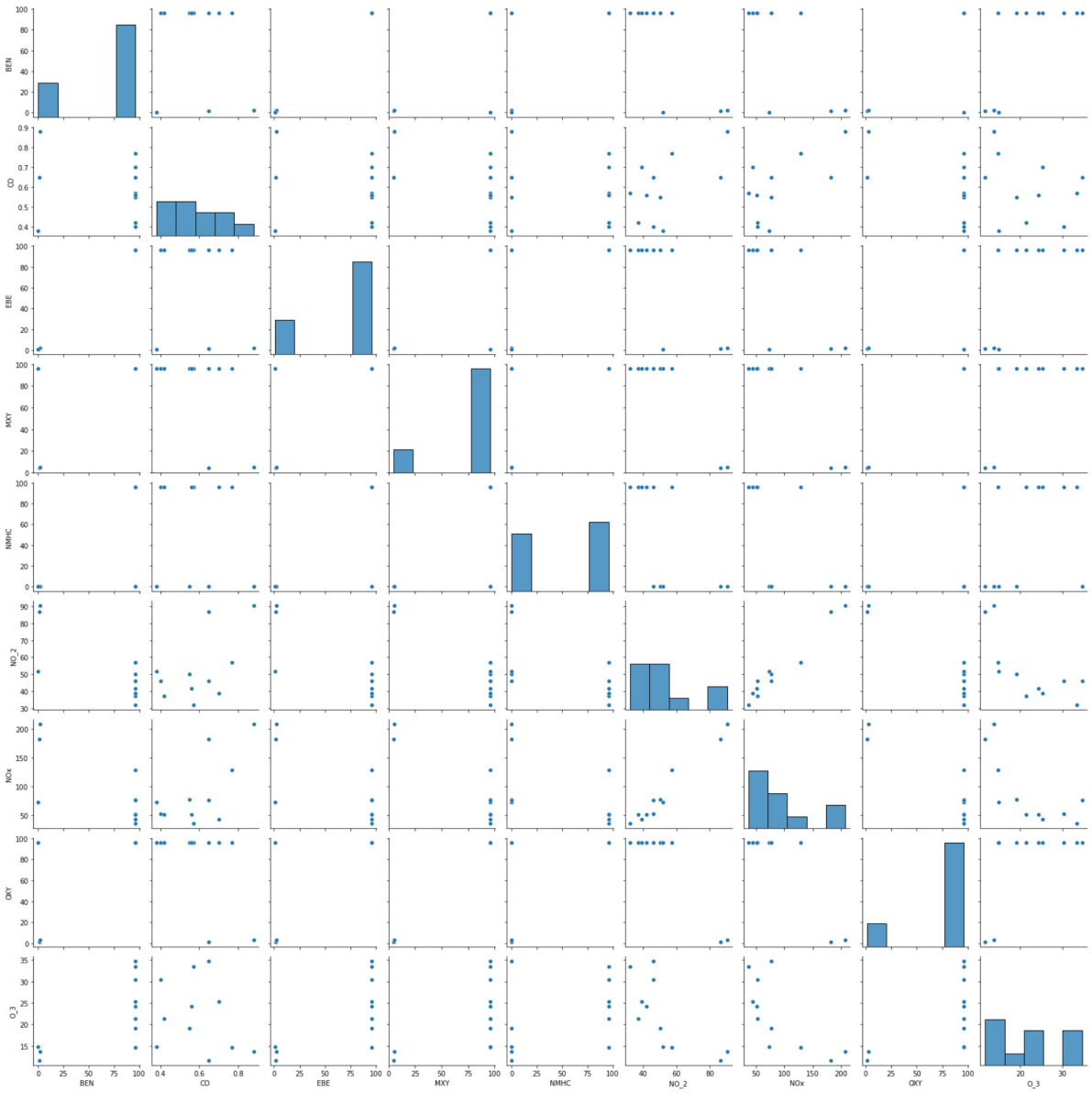
In [8]:

Out[8]: <AxesSubplot:>



In [9]:

Out[9]: <seaborn.axisgrid.PairGrid at 0x20b88d16f40>



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]:

```
2.8421709430404007e-13
```

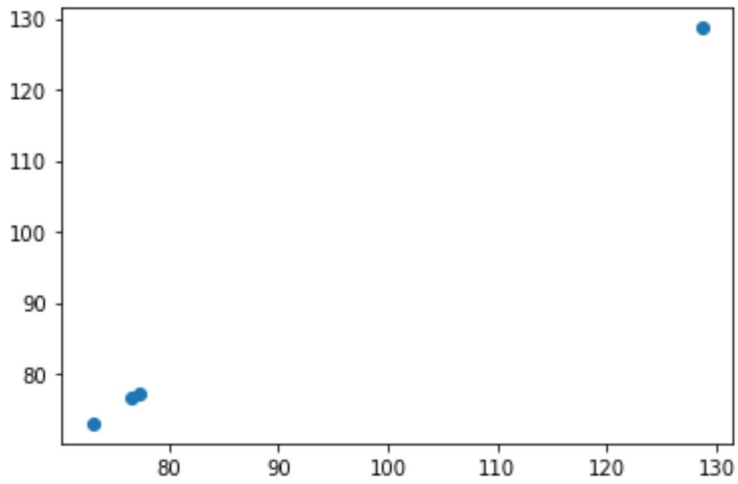
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[14]:

	Co-efficient
BEN	-4.788418e-15
CO	-1.443288e-13
EBE	4.406660e-15
MXY	-2.452388e-15
NMHC	-1.417675e-14
NO_2	-1.494051e-15
NOx	1.000000e+00
OXY	1.516668e-14

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x20b911b5250>
```



In [16]:

```
1.0
```

In [17]:

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

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

In [19]:

Out[19]:
```
0.9888688906654722
```

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

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

In [21]:

Out[21]: 0.9999950571988433

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

Out[22]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2005-11-01 01:00:00	96.00	0.77	96.00	96.00	96.00	57.130001	128.699997	96.00	14.720000	14
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30
2	2005-11-01 01:00:00	96.00	0.40	96.00	96.00	96.00	46.119999	53.000000	96.00	30.469999	14
3	2005-11-01 01:00:00	96.00	0.42	96.00	96.00	96.00	37.220001	52.009998	96.00	21.379999	15
4	2005-11-01 01:00:00	96.00	0.57	96.00	96.00	96.00	32.160000	36.680000	96.00	33.410000	5
...	...	...	...	...	...	...	...	...	...	...	
5995	2005-11-10 08:00:00	0.55	0.50	0.63	96.00	0.09	79.129997	153.500000	96.00	19.500000	10
5996	2005-11-10 08:00:00	0.32	0.40	1.00	1.00	0.13	23.120001	23.660000	1.00	53.139999	2
5997	2005-11-10 08:00:00	0.15	96.00	0.18	96.00	0.10	84.070000	132.600006	96.00	19.330000	10
5998	2005-11-10 08:00:00	0.21	96.00	1.00	96.00	0.08	72.959999	103.400002	96.00	96.000000	7
5999	2005-11-10 08:00:00	0.45	0.62	0.81	1.07	0.19	66.699997	107.800003	0.94	26.920000	10

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)
```

```
[28079039]
```

In [30]:

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

In [31]:

Out[31]:
```
7.7059476284601e-241
```

In [32]:

Out[32]:
```
7.683663948336802e-174
```

In [33]:

Out[33]:
```
0.5172222222222222
```

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: 15.006007753807612, tolera
nce: 3.149879320626048
  model = cd_fast.enet_coordinate_descent(
```

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

In [35]:

```
[-0.49777582 -0.         -0.21385573 -0.04162864 -0.05172053  0.0099412
  0.97544523  0.7726987 ]
```

In [36]:

```
3.8611902301954046
```

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

```
-1.4885415291779673
```

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]:
```
0.5452380952380953
```

In [42]:

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

Out[43]:
```
[Text(2334.3, 2491.5, 'X[4] <= -0.188\ngini = 0.963\nsamples = 2650\nvalue =
[185, 136, 144, 154, 158, 150, 143, 155, 162, 169\n162, 174, 157, 159, 177, 1
75, 143, 164, 154, 179\n133, 147, 105, 157, 146, 153, 159]'),
 Text(1190.4, 2038.5, 'X[2] <= -0.098\ngini = 0.908\nsamples = 1062\nvalue =
[0, 0, 0, 154, 158, 150, 0, 141, 0, 0, 133, 0\n0, 0, 0, 0, 0, 163, 154, 179,
132, 147, 0, 0\n0, 0, 159]'),
 Text(595.2, 1585.5, 'X[10] <= -0.619\ngini = 0.888\nsamples = 852\nvalue =
[0, 0, 0, 154, 0, 150, 0, 0, 0, 0, 133, 0, 0\n0, 0, 0, 0, 163, 143, 179, 132,
147, 0, 0, 0\n0, 159]'),
 Text(297.6, 1132.5, 'X[7] <= -2.52\ngini = 0.747\nsamples = 359\nvalue = [0,
0, 0, 154, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 116, 0, 0, 147, 0, 0, 0,
0, 159]'),
 Text(148.8, 679.5, 'X[4] <= -1.228\ngini = 0.688\nsamples = 184\nvalue = [0,
0, 0, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 101, 0, 0, 98, 0, 0, 0,
0, 76]'),
 Text(74.4, 226.5, 'gini = 0.587\nsamples = 60\nvalue = [0, 0, 0, 12, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 49, 0, 0, 0, 0, 0, 0, 0, 32]'),
 Text(223.20000000000002, 226.5, 'gini = 0.622\nsamples = 124\nvalue = [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 52, 0, 0, 98, 0, 0, 0, 0, 4
```

From this observation I had observe that the LASSO is a highest accuracy of 0.999950571988433

In [ ]: