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	PM
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990(
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980(
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440(
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180(
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330(
...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750(
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389(
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640(
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	N
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240(

217296 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217296 entries, 0 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217296 non-null  object 
 1   BEN      66747 non-null   float64
 2   CO       216637 non-null  float64
 3   EBE      58547 non-null   float64
 4   MXY      41255 non-null   float64
 5   NMHC     87045 non-null   float64
 6   NO_2     216439 non-null  float64
 7   NOx      216439 non-null  float64
 8   OXY      41314 non-null   float64
 9   O_3      216726 non-null  float64
 10  PM10     209113 non-null  float64
 11  PXY      41256 non-null   float64
 12  SO_2     216507 non-null  float64
 13  TCH      87115 non-null   float64
 14  TOL      66619 non-null   float64
 15  station  217296 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.5+ MB
```

In [4]:
```
b=a.fillna(value=135)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2002-04-01 01:00:00	135.00	1.39	135.00	135.00	135.00	145.100006	352.100006	135.00	6.54
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85
2	2002-04-01 01:00:00	135.00	0.80	135.00	135.00	135.00	103.699997	134.000000	135.00	13.01
3	2002-04-01 01:00:00	135.00	1.61	135.00	135.00	135.00	97.599998	268.000000	135.00	5.12
4	2002-04-01 01:00:00	135.00	1.90	135.00	135.00	135.00	92.089996	237.199997	135.00	7.28
...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	135.00	135.00	135.00	81.080002	265.700012	135.00	7.21
217292	2002-11-01 00:00:00	3.67	1.73	2.89	135.00	0.38	113.900002	373.100006	135.00	5.66
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	135.00	149.800003	202.199997	1.00	5.75
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38

217296 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_3
0	2002-04-01 01:00:00	135.00	1.39	135.00	135.00	135.00	145.100006	352.100006	135.00	6.540000
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.850000
2	2002-04-01 01:00:00	135.00	0.80	135.00	135.00	135.00	103.699997	134.000000	135.00	13.010000
3	2002-04-01 01:00:00	135.00	1.61	135.00	135.00	135.00	97.599998	268.000000	135.00	5.120000
4	2002-04-01 01:00:00	135.00	1.90	135.00	135.00	135.00	92.089996	237.199997	135.00	7.280000
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.370000
6	2002-04-01 01:00:00	135.00	0.78	135.00	135.00	0.09	101.000000	119.300003	135.00	20.549999
7	2002-04-01 01:00:00	135.00	1.06	135.00	135.00	135.00	127.300003	204.100006	135.00	3.150000
8	2002-04-01 01:00:00	135.00	1.21	135.00	135.00	135.00	106.300003	126.599998	135.00	22.389999
9	2002-04-01 01:00:00	135.00	0.61	135.00	135.00	0.14	95.540001	110.699997	135.00	27.770000
10	2002-04-01 01:00:00	135.00	0.70	135.00	135.00	135.00	110.099998	138.600006	135.00	11.580000

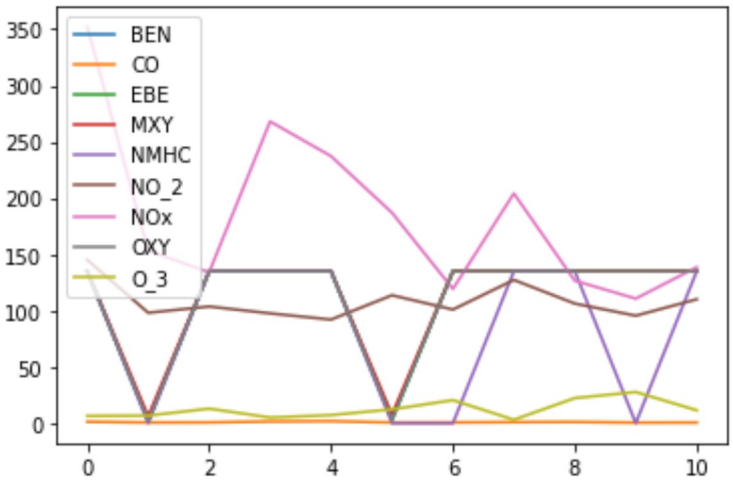
In [7]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3']]
```

Out[7]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	135.00	1.39	135.00	135.00	135.00	145.100006	352.100006	135.00	6.540000
1	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.850000
2	135.00	0.80	135.00	135.00	135.00	103.699997	134.000000	135.00	13.010000
3	135.00	1.61	135.00	135.00	135.00	97.599998	268.000000	135.00	5.120000
4	135.00	1.90	135.00	135.00	135.00	92.089996	237.199997	135.00	7.280000
5	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.370000
6	135.00	0.78	135.00	135.00	0.09	101.000000	119.300003	135.00	20.549999
7	135.00	1.06	135.00	135.00	135.00	127.300003	204.100006	135.00	3.150000
8	135.00	1.21	135.00	135.00	135.00	106.300003	126.599998	135.00	22.389999
9	135.00	0.61	135.00	135.00	0.14	95.540001	110.699997	135.00	27.770000
10	135.00	0.70	135.00	135.00	135.00	110.099998	138.600006	135.00	11.580000

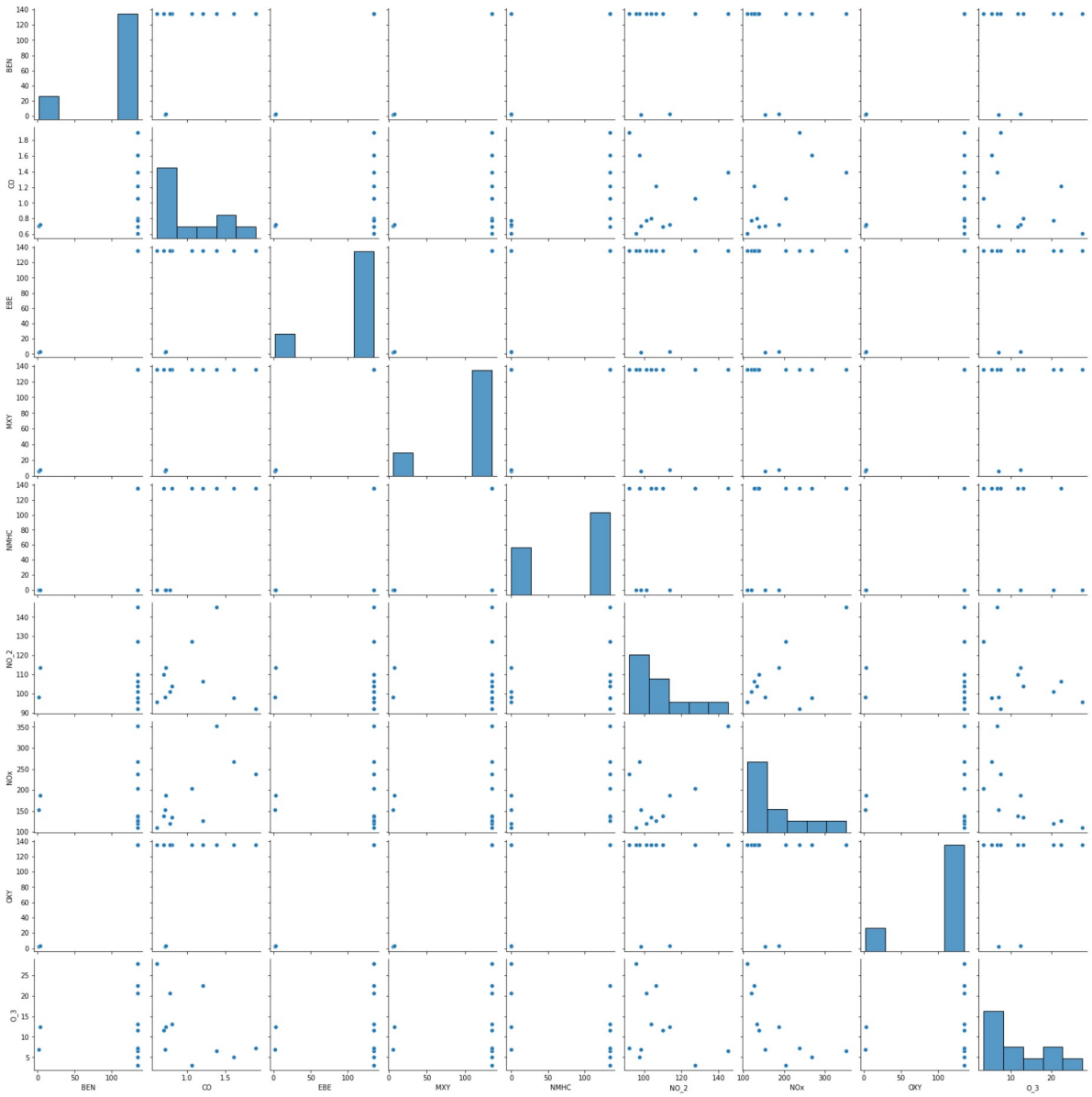
In [8]:

Out[8]: <AxesSubplot:>



In [9]:

Out[9]: <seaborn.axisgrid.PairGrid at 0x21852285490>



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]:

```
3.410605131648481e-13
```

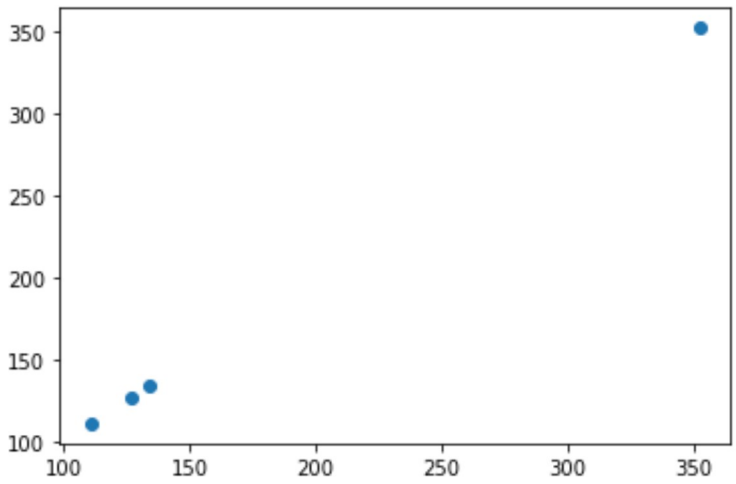
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[14]:

	Co-efficient
BEN	1.513985e-15
CO	2.304437e-15
EBE	-1.502077e-15
MXY	3.275995e-15
NMHC	-1.602261e-15
NO_2	-2.340866e-17
NOx	1.000000e+00
OXY	-1.373097e-15

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x21856098fd0>
```



In [16]:

```
1.0
```

In [17]:

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

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

In [19]:

Out[19]:
```
0.9999982814360505
```

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

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

In [21]:

Out[21]: 0.9999838094834425

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

Out[22]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2002-04-01 01:00:00	135.00	1.39	135.00	135.00	135.00	145.100006	352.100006	135.00	6.54	4
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	2
2	2002-04-01 01:00:00	135.00	0.80	135.00	135.00	135.00	103.699997	134.000000	135.00	13.01	2
3	2002-04-01 01:00:00	135.00	1.61	135.00	135.00	135.00	97.599998	268.000000	135.00	5.12	4
4	2002-04-01 01:00:00	135.00	1.90	135.00	135.00	135.00	92.089996	237.199997	135.00	7.28	7
...	...	...	...	...	...	...	...	...	...	...	
4995	2002-04-09 08:00:00	135.00	1.16	135.00	135.00	135.00	51.880001	268.399994	135.00	6.18	3
4996	2002-04-09 08:00:00	1.63	0.97	1.60	135.00	0.18	58.910000	184.399994	135.00	7.95	2
4997	2002-04-09 08:00:00	1.09	0.60	0.95	1.84	0.13	34.119999	39.279999	0.89	4.47	1
4998	2002-04-09 08:00:00	2.03	1.14	1.85	3.71	135.00	79.870003	362.000000	0.80	7.37	13
4999	2002-04-09 08:00:00	2.12	1.28	2.14	5.70	0.22	70.230003	214.699997	2.30	6.17	3

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, 28079009,
       28079011, 28079012, 28079014, 28079015, 28079016, 28079017,
       28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
       28079025, 28079035, 28079036, 28079038, 28079039, 28079040,
       28079099], dtype=int64)
```

In [31]:

Out[31]:
```
5.6002597380155035e-62
```

In [32]:

Out[32]:
```
0.9999998612380067
```

In [33]:

Out[33]:
```
0.5473333333333333
```

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

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

In [35]:

```
[-0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  1.18989985e-04 -0.00000000e+00  9.99505705e-01 -0.00000000e+00]
```

In [36]:

```
0.08314912939138708
```

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

```
0.9999997755985716
```

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]:
```
0.5697142857142857
```

In [42]:

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

Out[43]:
```
[Text(2182.7647058823527, 2491.5, 'X[2] <= -0.403\ngini = 0.959\nsamples = 22
14\nvalue = [129, 107, 107, 153, 150, 129, 146, 131, 136, 149\n141, 121, 151,
138, 181, 151, 151, 130, 128, 131\n137, 154, 151, 154, 144]'),
 Text(1083.1764705882351, 2038.5, 'X[10] <= -0.664\ngini = 0.856\nsamples = 6
09\nvalue = [0, 0, 0, 151, 0, 0, 0, 0, 0, 149, 0, 0, 0, 0\n0, 0, 151, 122, 12
8, 131, 0, 0, 0, 0, 144]'),
 Text(656.470588235294, 1585.5, 'X[4] <= -0.208\ngini = 0.799\nsamples = 430\
nvalue = [0, 0, 0, 151, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 122, 128, 131,
0, 0, 0, 0, 144]'),
 Text(525.1764705882352, 1132.5, 'X[0] <= -1.47\ngini = 0.748\nsamples = 341\
nvalue = [0, 0, 0, 151, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 122, 0, 131,
0, 0, 0, 0, 144]'),
 Text(262.5882352941176, 679.5, 'X[6] <= -0.501\ngini = 0.706\nsamples = 209\
nvalue = [0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 118, 0, 72, 0,
0, 0, 0, 117]'),
 Text(131.2941176470588, 226.5, 'gini = 0.55\nsamples = 91\nvalue = [0, 0, 0,
11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 93, 0, 14, 0, 0, 0, 0, 30]'),
 Text(393.88235294117646, 226.5, 'gini = 0.671\nsamples = 118\nvalue = [0, 0,
0, 21, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 25, 0, 58, 0, 0, 0, 0, 87]'),
```

From this observation I had observe that the RIDGE has the highest accuracy of 0.9999982814360505

In [ ]: