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	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.
...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.

245496 rows × 17 columns

In [3]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245496 entries, 0 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     245496 non-null  object 
 1   BEN      65158 non-null   float64
 2   CO       226043 non-null  float64
 3   EBE      56781 non-null   float64
 4   MXY      39867 non-null   float64
 5   NMHC     107630 non-null  float64
 6   NO_2     243280 non-null  float64
 7   NOx      243283 non-null  float64
 8   OXY      39882 non-null   float64
 9   O_3      233811 non-null  float64
 10  PM10     234655 non-null  float64
 11  PM25     58145 non-null   float64
 12  PXY      39891 non-null   float64
 13  SO_2     243402 non-null  float64
 14  TCH      107650 non-null  float64
 15  TOL      64914 non-null   float64
 16  station  245496 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 31.8+ MB
```

In [4]:
```
b=a.fillna(value=106)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
0	2004-08-01 01:00:00	106.00	0.66	106.00	106.00	106.00	89.550003	118.900002	106.00	40.(
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.(
2	2004-08-01 01:00:00	106.00	1.02	106.00	106.00	106.00	93.389999	138.600006	106.00	20.8
3	2004-08-01 01:00:00	106.00	0.53	106.00	106.00	106.00	87.290001	105.000000	106.00	36.7
4	2004-08-01 01:00:00	106.00	0.17	106.00	106.00	106.00	34.910000	35.349998	106.00	86.2
...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.(
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	106.00	97.139999	146.899994	2.34	7.7
245493	2004-06-01 00:00:00	106.00	106.00	106.00	106.00	0.13	102.699997	132.600006	106.00	17.8
245494	2004-06-01 00:00:00	106.00	106.00	106.00	106.00	0.09	82.599998	102.599998	106.00	106.(
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.4

245496 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	2004-08-01 01:00:00	106.00	0.66	106.00	106.00	106.00	89.550003	118.900002	106.00	40.020000
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999
2	2004-08-01 01:00:00	106.00	1.02	106.00	106.00	106.00	93.389999	138.600006	106.00	20.860001
3	2004-08-01 01:00:00	106.00	0.53	106.00	106.00	106.00	87.290001	105.000000	106.00	36.730000
4	2004-08-01 01:00:00	106.00	0.17	106.00	106.00	106.00	34.910000	35.349998	106.00	86.269997
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000
6	2004-08-01 01:00:00	106.00	0.43	106.00	106.00	0.17	54.270000	64.279999	106.00	66.589996
7	2004-08-01 01:00:00	1.41	0.47	2.35	106.00	0.02	71.730003	87.519997	106.00	53.270000
8	2004-08-01 01:00:00	106.00	1.28	106.00	106.00	106.00	147.699997	202.500000	106.00	10.280000
9	2004-08-01 01:00:00	106.00	0.43	106.00	106.00	0.27	54.290001	68.099998	106.00	66.709999
10	2004-08-01 01:00:00	106.00	0.60	106.00	106.00	106.00	73.410004	87.059998	106.00	40.480000

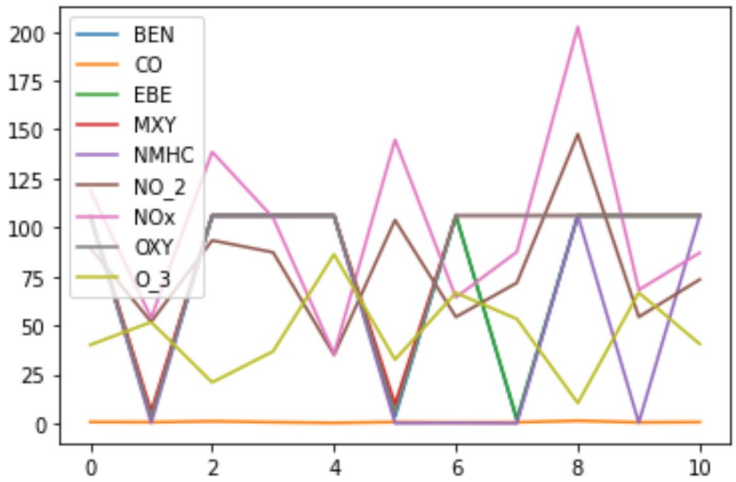
In [7]:
```
d=c[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3']]
```

Out[7]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	106.00	0.66	106.00	106.00	106.00	89.550003	118.900002	106.00	40.020000
1	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999
2	106.00	1.02	106.00	106.00	106.00	93.389999	138.600006	106.00	20.860001
3	106.00	0.53	106.00	106.00	106.00	87.290001	105.000000	106.00	36.730000
4	106.00	0.17	106.00	106.00	106.00	34.910000	35.349998	106.00	86.269997
5	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000
6	106.00	0.43	106.00	106.00	0.17	54.270000	64.279999	106.00	66.589996
7	1.41	0.47	2.35	106.00	0.02	71.730003	87.519997	106.00	53.270000
8	106.00	1.28	106.00	106.00	106.00	147.699997	202.500000	106.00	10.280000
9	106.00	0.43	106.00	106.00	0.27	54.290001	68.099998	106.00	66.709999
10	106.00	0.60	106.00	106.00	106.00	73.410004	87.059998	106.00	40.480000

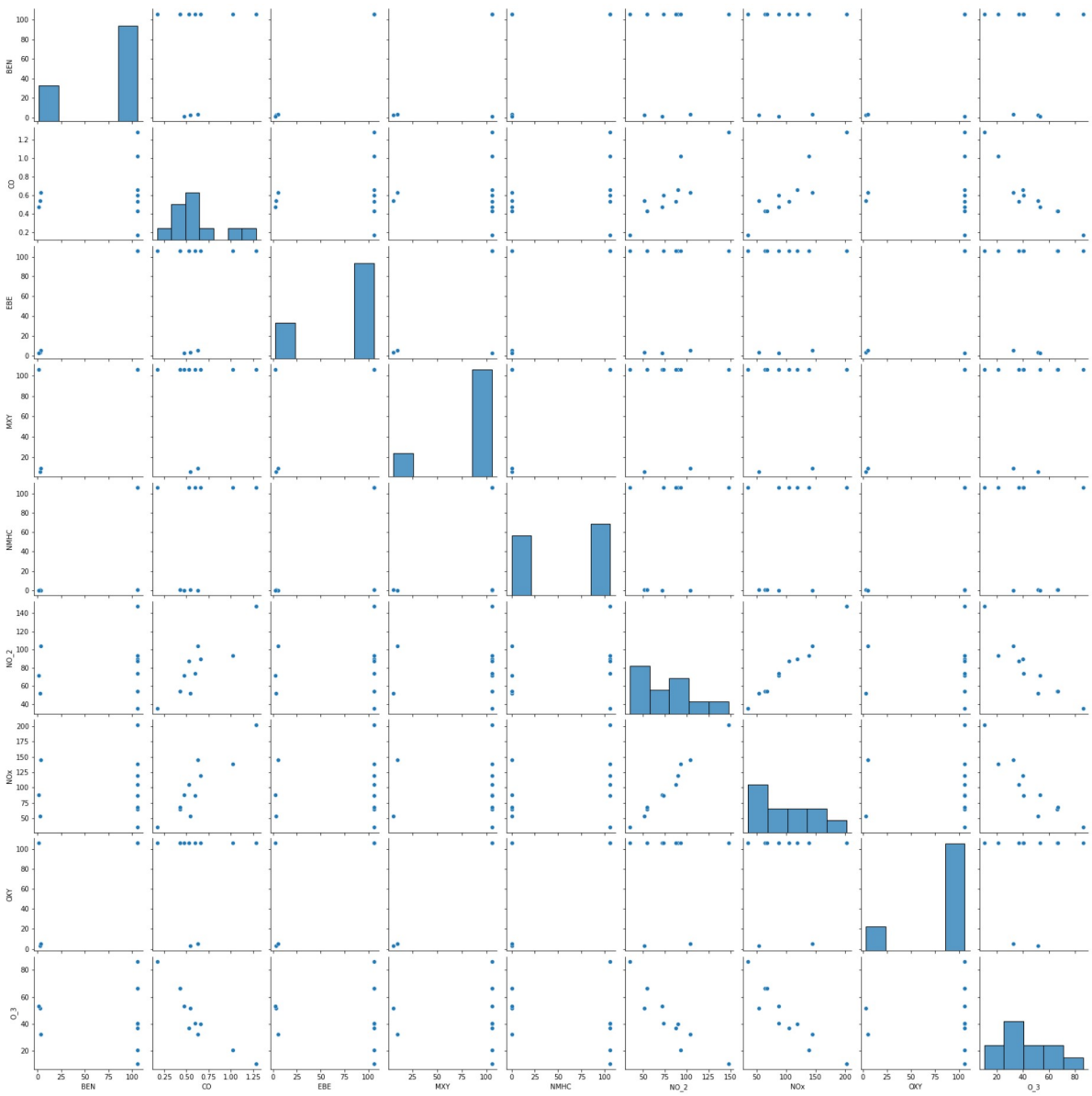
In [8]:

Out[8]: <AxesSubplot:>



In [9]:

Out[9]: <seaborn.axisgrid.PairGrid at 0x2192516a160>



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]:

```
-0.0008645494699663914
```

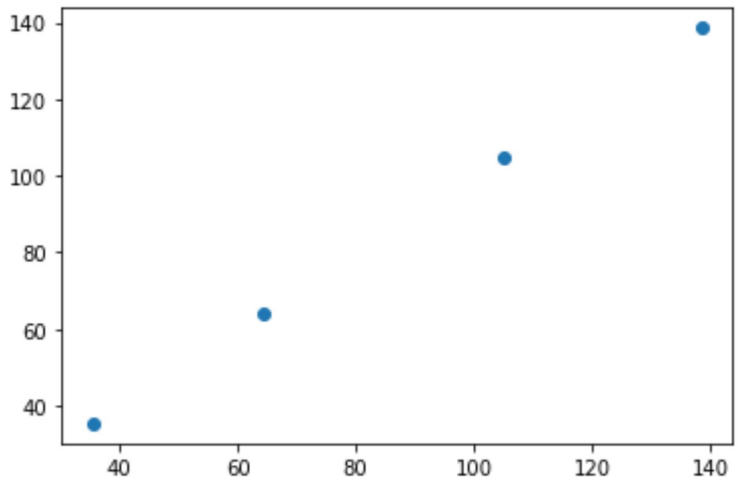
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[14]:

	Co-efficient
BEN	0.000833
CO	-0.012756
EBE	-0.000813
MXY	-0.000835
NMHC	-0.000017
NO_2	0.000374
NOx	0.999834
OXY	0.000790

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x2192dbc92b0>
```



In [16]:

```
0.9999999916628053
```

In [17]:

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

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

In [19]:

Out[19]:
```
0.9993279789663293
```

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

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

In [21]:

Out[21]: 0.9999734689666656

In [22]:
```
a1=b.head(4000)
```

Out[22]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2004-08-01 01:00:00	106.00	0.66	106.00	106.00	106.00	89.550003	118.900002	106.00	40.020000
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999
2	2004-08-01 01:00:00	106.00	1.02	106.00	106.00	106.00	93.389999	138.600006	106.00	20.860001
3	2004-08-01 01:00:00	106.00	0.53	106.00	106.00	106.00	87.290001	105.000000	106.00	36.730000
4	2004-08-01 01:00:00	106.00	0.17	106.00	106.00	106.00	34.910000	35.349998	106.00	86.269997
...	...	...	...	...	...	...	...	...	...	...
3995	2004-08-07 00:00:00	106.00	0.75	106.00	106.00	106.00	114.900002	189.300003	106.00	12.330000
3996	2004-08-07 00:00:00	106.00	1.68	106.00	106.00	106.00	143.199997	321.000000	106.00	8.710000
3997	2004-08-07 00:00:00	106.00	0.65	106.00	106.00	0.65	130.199997	156.500000	106.00	7.600000
3998	2004-08-07 00:00:00	106.00	0.82	106.00	106.00	106.00	125.400002	207.199997	106.00	9.000000
3999	2004-08-07 00:00:00	106.00	0.74	106.00	106.00	106.00	104.800003	121.599998	106.00	18.430000

4000 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)
```

```
[28079004]
```

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]:
```
2.793068192100045e-39
```

In [32]:

Out[32]:
```
5.075635610824596e-186
```

In [33]:

Out[33]:
```
0.6116666666666667
```

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

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

In [35]:

```
[ 0.00000000e+00 -0.00000000e+00  4.45095914e-04 -0.00000000e+00
 -7.95362564e-04  3.16610023e-02  9.79393568e-01 -0.00000000e+00]
```

In [36]:

```
-0.4250071227973251
```

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

```
0.9999745139676599
```

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]:
```
0.6492857142857142
```

In [42]:

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

Out[43]:
```
[Text(2028.1153846153845, 2491.5, 'X[0] <= -0.435\ngini = 0.964\nsamples = 17
64\nvalue = [117, 102, 127, 102, 115, 128, 102, 104, 77, 90, 87\n84, 80, 87,
95, 88, 107, 92, 94, 82, 115, 96\n101, 109, 109, 100, 101, 109]'),
 Text(965.7692307692307, 2038.5, 'X[3] <= -0.699\ngini = 0.873\nsamples = 49
9\nvalue = [0, 0, 0, 101, 0, 128, 0, 0, 0, 0, 87, 0, 0\n0, 0, 0, 102, 0, 94,
80, 0, 0, 101, 0, 0, 0\n0, 109]'),
 Text(472.15384615384613, 1585.5, 'X[3] <= -2.184\ngini = 0.798\nsamples = 30
9\nvalue = [0, 0, 0, 101, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 94, 80,
0, 0, 101, 0, 0, 0, 0\n109]'),
 Text(257.53846153846155, 1132.5, 'X[12] <= -0.07\ngini = 0.613\nsamples = 10
7\nvalue = [0, 0, 0, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 80, 58, 0,
0, 0, 0, 0, 0, 0, 11]'),
 Text(171.69230769230768, 679.5, 'X[0] <= -1.597\ngini = 0.372\nsamples = 71\
nvalue = [0, 0, 0, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 80, 0, 0, 0,
0, 0, 0, 0, 0, 11]'),
 Text(85.84615384615384, 226.5, 'gini = 0.185\nsamples = 61\nvalue = [0, 0,
0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 80, 0, 0, 0, 0, 0, 0, 0, 0,
2]'),
 Text(257.53846153846155, 226.5, 'gini = 0.459\nsamples = 10\nvalue = [0, 0,
```

From this observation I had observe that the LINEAR REFRESSION highest accuracy of 0.9999999916628053

In [ ]: