20104016

DEENA

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
```

Importing Datasets

In [2]:

```
df=pd.read_csv("madrid_2004.csv")
```

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

Data Cleaning and Data Preprocessing

In [3]:

In [4]:

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3
',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     19397 non-null  object 
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
 14  TCH      19397 non-null  float64
 15  TOL      19397 non-null  float64
 16  station  19397 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

In [6]:
```
data=df[['CO' ,'station']]
```

Out[6]:

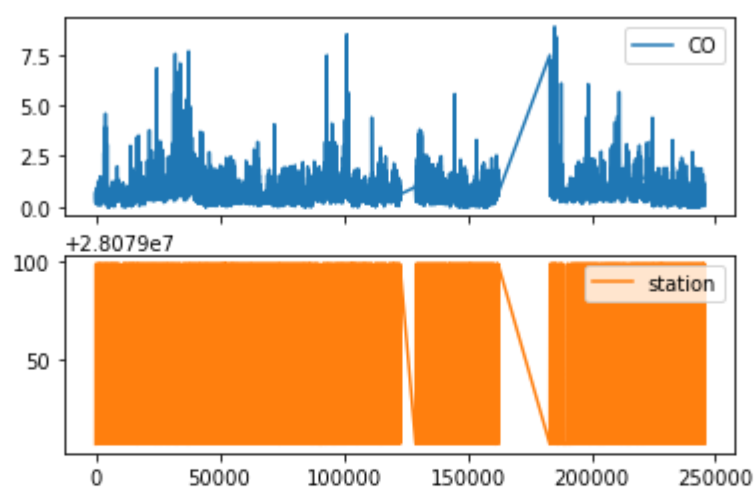
	CO	station
5	0.63	28079006
22	0.36	28079024
26	0.46	28079099
32	0.67	28079006
49	0.30	28079024
...	...	...
245463	0.08	28079024
245467	0.67	28079099
245473	1.12	28079006
245491	0.21	28079024
245495	0.67	28079099

19397 rows × 2 columns

Line chart

In [7]:

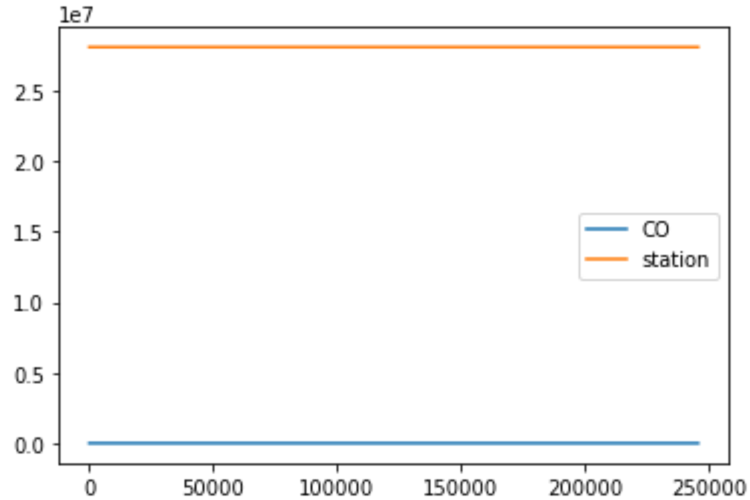
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:

Out[8]: <AxesSubplot:>

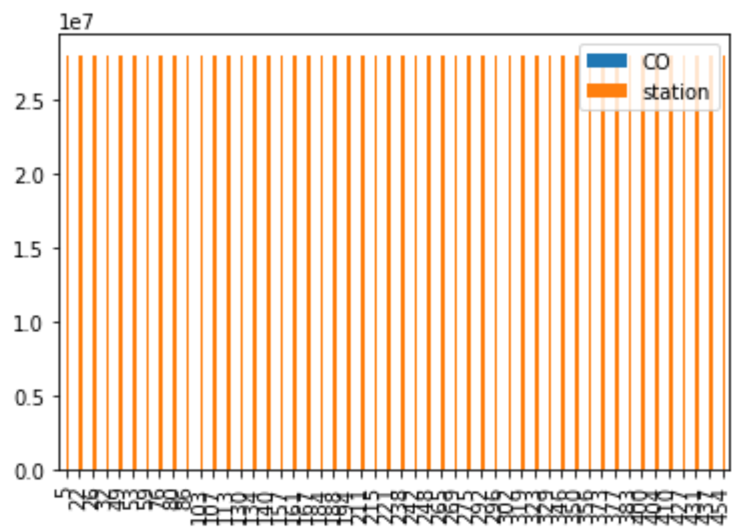


Bar chart

In [9]:

In [10]:

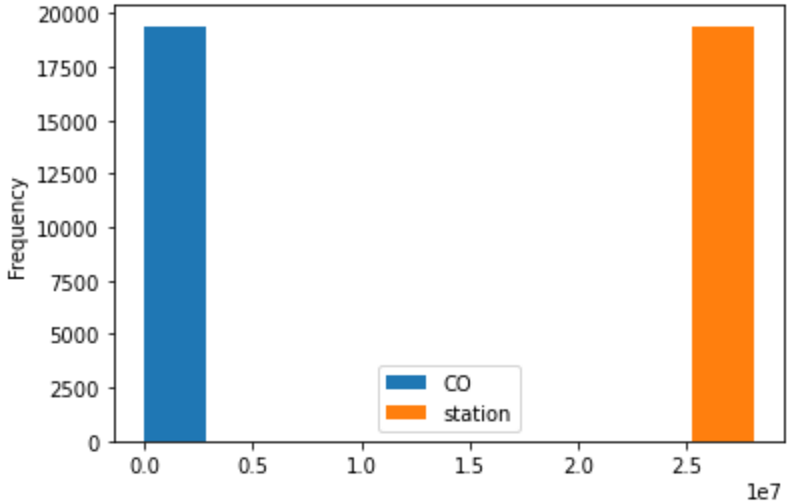
Out[10]: <AxesSubplot:>



Histogram

In [11]:

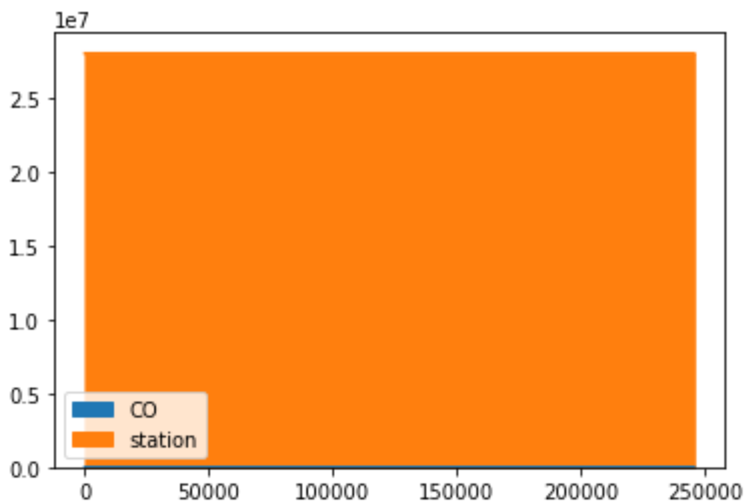
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:

Out[12]: <AxesSubplot:>

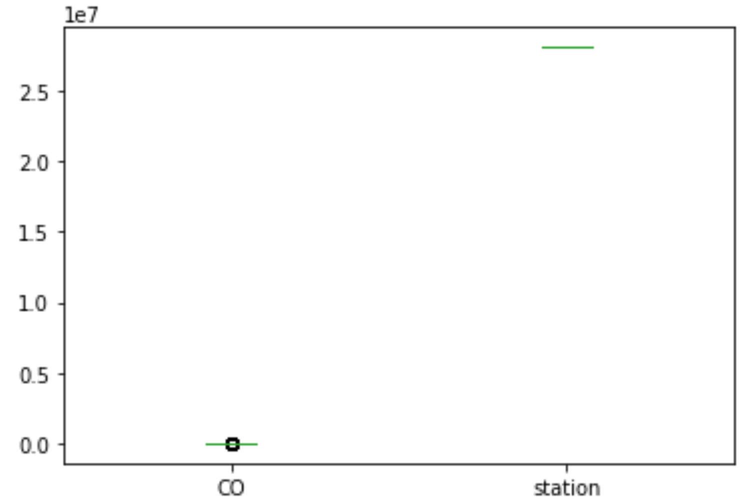


Box chart

In [13]:

```
data.plot.box()
```

Out[13]: <AxesSubplot:>

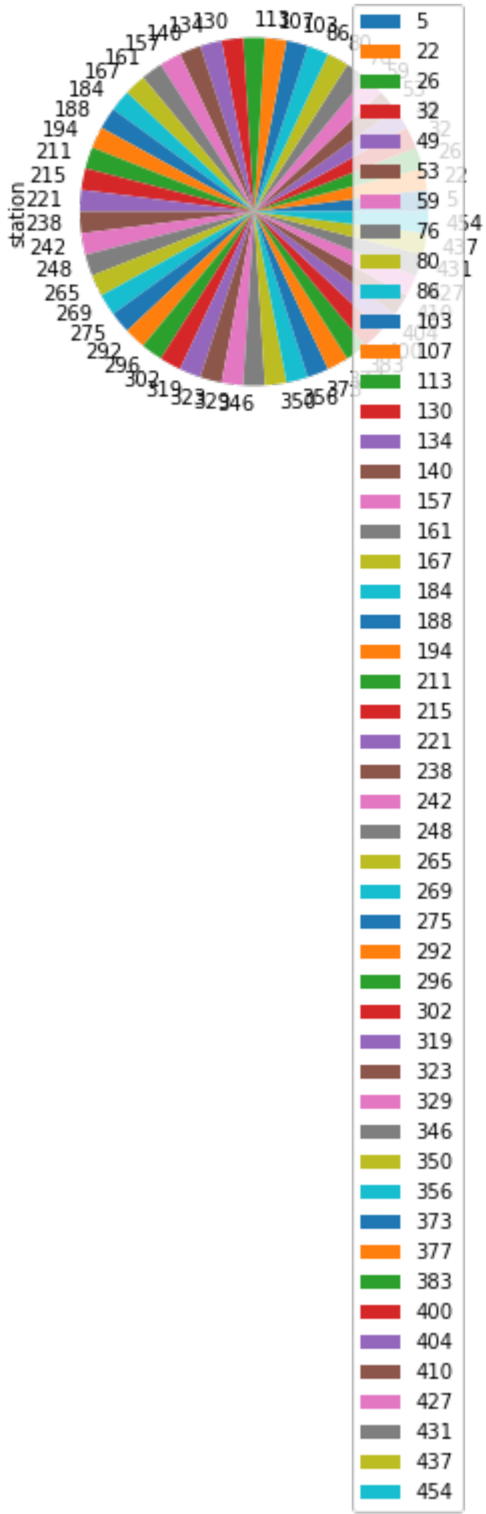


Pie chart

In [14]:

```
[illegible]
```

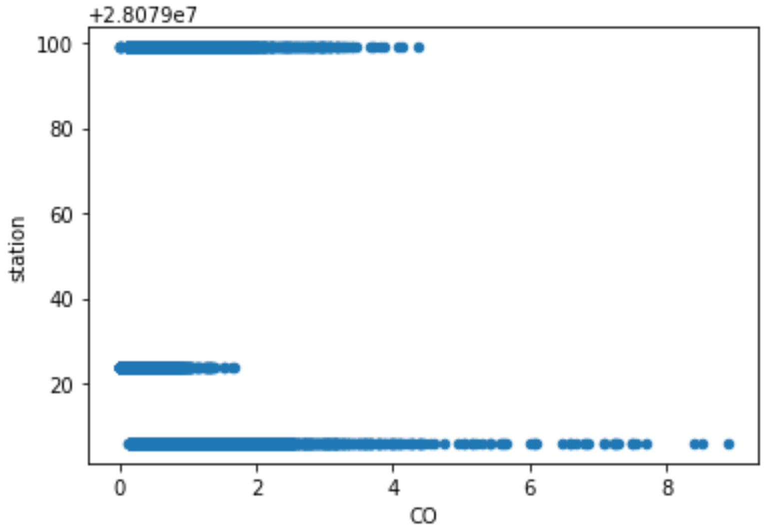
Out[14]: <AxesSubplot:ylabel='station'>



Scatter chart

In [15]:

Out[15]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [16]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     19397 non-null  object
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
```

In [17]:
```
df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	193
mean	2.250781	0.675347	2.775913	5.424809	0.151024	62.887023	1
std	2.184724	0.591026	2.729622	5.554358	0.158603	37.952255	1
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.090000	
25%	0.870000	0.320000	1.020000	1.780000	0.060000	35.150002	
50%	1.620000	0.520000	1.970000	3.800000	0.110000	58.310001	
75%	2.910000	0.860000	3.580000	7.260000	0.200000	85.730003	1
max	34.180000	8.900000	41.880001	91.599998	4.810000	355.100006	17

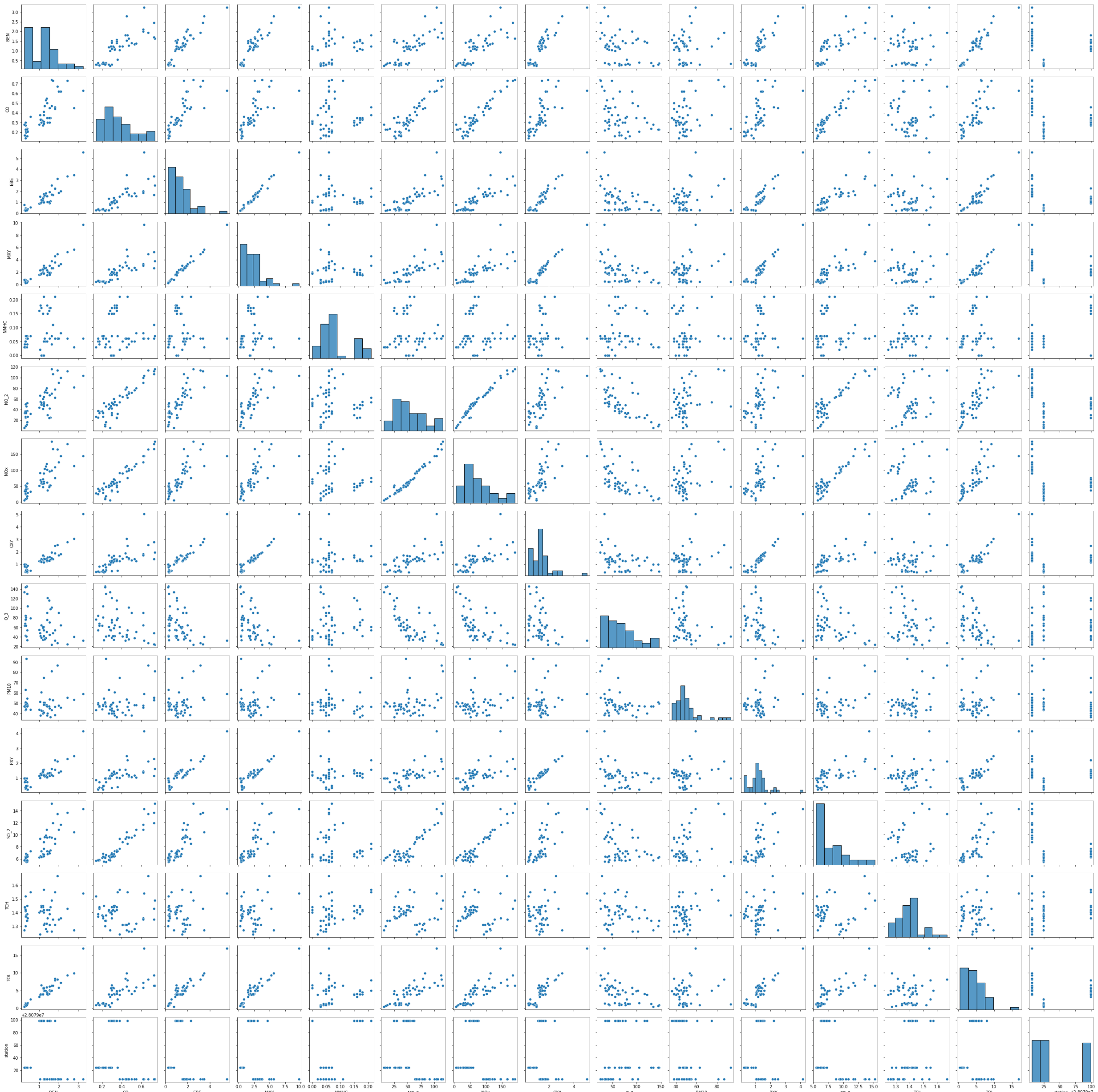
In [18]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
```

EDA AND VISUALIZATION

In [19]:

```
sns.pairplot(df1[0:50])
```

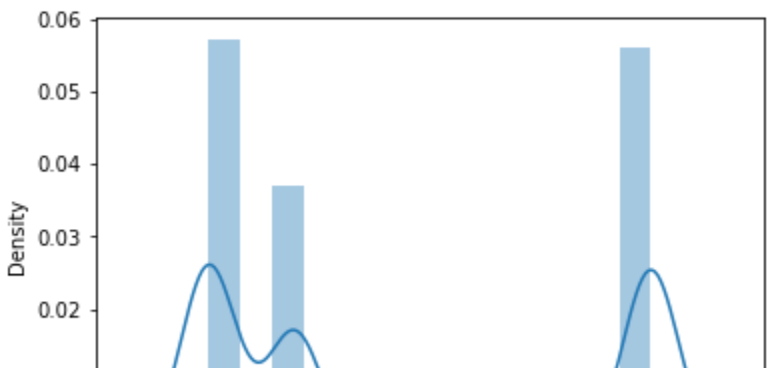
Out[19]: <seaborn.axisgrid.PairGrid at 0x281e42d27c0>



In [20]:

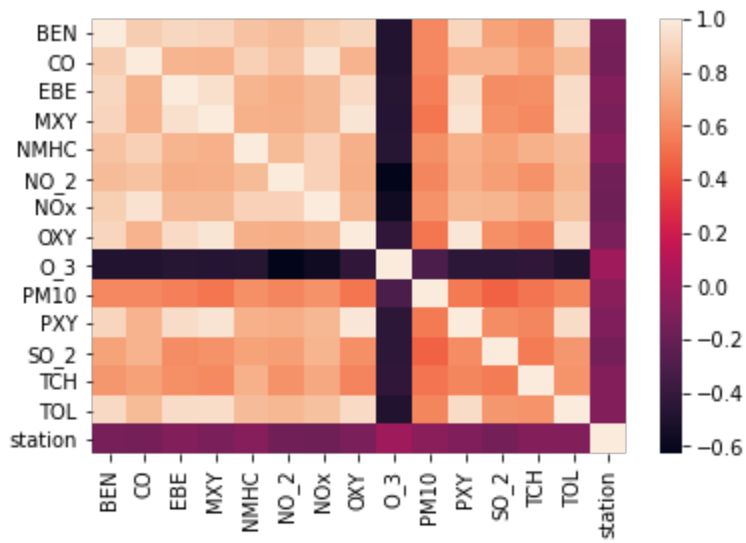
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[20]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [21]:

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:

```python
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
```

In [23]:

```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

Linear Regression

In [24]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

Out[24]: LinearRegression()

In [25]:

Out[25]: 28079074.797222655

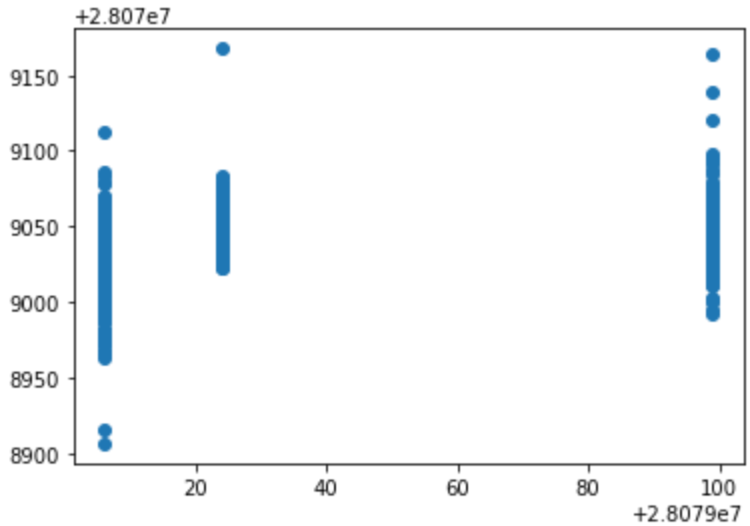
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[26]:

	Co-efficient
BEN	-3.979833
CO	26.686489
EBE	4.031195
MXY	-3.630869
NMHC	78.488600
NO_2	-0.146845
NOx	-0.247987
OXY	-1.116248
O_3	-0.282896
PM10	0.077967
PXY	5.333061
SO_2	-0.193799
TCH	-6.894389
TOL	0.996277

In [27]:
```
prediction =lr.predict(x_test)
```

Out[27]: <matplotlib.collections.PathCollection at 0x281f2d9a190>



ACCURACY

In [28]:

Out[28]: 0.11777881564565851

In [29]:

Out[29]: 0.10128873662984172

Ridge and Lasso

In [30]:

In [31]:
```
rr=Ridge(alpha=10)
```

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:

Out[32]: 0.11551473614343155

In [33]:

Out[33]: 0.10097945478412673

In [34]:
```
la=Lasso(alpha=10)
```

Out[34]: Lasso(alpha=10)

In [35]:

Out[35]: 0.0520104650845411

Accuracy(Lasso)

In [36]:

Out[36]: 0.054195477926028524

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
```

Out[37]: ElasticNet()

In [38]:

Out[38]:
```
array([-0.        ,  0.3550942 ,  1.49111275, -1.90128249,  0.        ,
       -0.15961622, -0.08669339, -0.        , -0.2136377 ,  0.10080337,
        0.57981765, -0.11981261,  0.        ,  1.08087946])
```

In [39]:

Out[39]: 28079066.10971685

In [40]:

In [41]:

Out[41]: 0.0700726667853745

Evaluation Metrics

In [42]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
```

```
38.41876024388903
1648.1562837904671
40.59749110216624
```

Logistic Regression

In [43]:

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
```

In [45]:

Out[45]: (19397, 14)

In [46]:

Out[46]: (19397,)

In [47]:

In [48]:

In [49]:
```
logr=LogisticRegression(max_iter=10000)
```

Out[49]: LogisticRegression(max_iter=10000)

In [50]:

In [51]:
```
prediction=logr.predict(observation)
```

[28079006]

In [52]:

Out[52]: array([28079006, 28079024, 28079099], dtype=int64)

In [53]:

Out[53]: 0.7360416559261741

In [54]:

Out[54]: 0.9999978255573396

In [55]:

Out[55]: array([[9.99997826e-01, 7.75018107e-20, 2.17444266e-06]])

Random Forest

In [56]:

In [57]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[57]: RandomForestClassifier()

In [58]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
```

In [59]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="ac
```

Out[59]:
```
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 [60]:

Out[60]:
```
0.7745449956179764
```

In [61]:

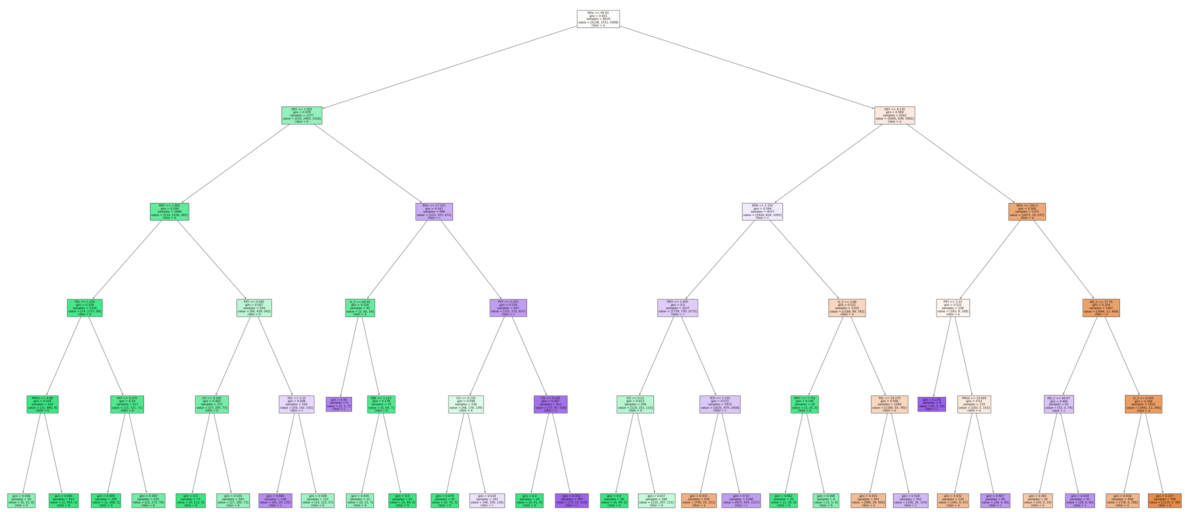
In [62]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
```

Out[62]:
```
[Text(2251.928571428571, 1993.2, 'NOx <= 49.43\ngini = 0.655\nsamples = 8639\
nvalue = [5238, 3331, 5008]\nclass = a'),
 Text(1135.9285714285713, 1630.8000000000002, 'OXY <= 1.005\ngini = 0.479\nsa
mples = 2377\nvalue = [233, 2493, 1016]\nclass = b'),
 Text(637.7142857142857, 1268.4, 'MXY <= 1.005\ngini = 0.299\nsamples = 1688\
nvalue = [110, 2156, 345]\nclass = b'),
 Text(318.85714285714283, 906.0, 'TOL <= 1.205\ngini = 0.109\nsamples = 1159\
nvalue = [24, 1717, 80]\nclass = b'),
 Text(159.42857142857142, 543.5999999999999, 'PM10 <= 4.68\ngini = 0.039\nsam
ples = 642\nvalue = [11, 996, 9]\nclass = b'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.506\nsamples = 30\nval
ue = [9, 33, 8]\nclass = b'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.006\nsamples = 612\nva
lue = [2, 963, 1]\nclass = b'),
 Text(478.2857142857142, 543.5999999999999, 'PXY <= 0.375\ngini = 0.19\nsampl
es = 517\nvalue = [13, 721, 71]\nclass = b'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.009\nsamples = 285\nv
alue = [1, 448, 1]\nclass = b'),
 Text(558.0, 181.19999999999982, 'gini = 0.369\nsamples = 232\nvalue = [12, 2
73, 70]\nclass = b'),
 Text(956.5714285714284, 906.0, 'PXY <= 0.585\ngini = 0.567\nsamples = 529\nv
alue = [86, 439, 265]\nclass = b'),
 Text(797.1428571428571, 543.5999999999999, 'CO <= 0.165\ngini = 0.402\nsampl
es = 271\nvalue = [27, 297, 73]\nclass = b'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.0\nsamples = 71\nvalue
= [0, 112, 0]\nclass = b'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.504\nsamples = 200\nva
lue = [27, 185, 73]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'TOL <= 3.15\ngini = 0.608\nsamples = 258\nv
alue = [59, 142, 192]\nclass = c'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.484\nsamples = 136\nv
alue = [45, 20, 135]\nclass = c'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.508\nsamples = 122\nv
alue = [14, 122, 57]\nclass = b'),
 Text(1634.142857142857, 1268.4, 'NOx <= 17.525\ngini = 0.547\nsamples = 689\
nvalue = [123, 337, 671]\nclass = c'),
 Text(1355.142857142857, 906.0, 'O_3 <= 60.65\ngini = 0.326\nsamples = 42\nva
lue = [2, 65, 14]\nclass = b'),
 Text(1275.4285714285713, 543.5999999999999, 'gini = 0.46\nsamples = 5\nvalue
= [2, 1, 7]\nclass = c'),
 Text(1434.8571428571427, 543.5999999999999, 'EBE <= 1.115\ngini = 0.178\nsam
ples = 37\nvalue = [0, 64, 7]\nclass = b'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.434\nsamples = 12\nval
ue = [0, 15, 7]\nclass = b'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.0\nsamples = 25\nvalu
e = [0, 49, 0]\nclass = b'),
 Text(1913.1428571428569, 906.0, 'PXY <= 1.015\ngini = 0.528\nsamples = 647\n
value = [121, 272, 657]\nclass = c'),
 Text(1753.7142857142856, 543.5999999999999, 'CO <= 0.135\ngini = 0.599\nsamp
les = 236\nvalue = [48, 179, 139]\nclass = b'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.079\nsamples = 45\nva
lue = [0, 70, 3]\nclass = b'),
```

```
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.619\nsamples = 191\nv
alue = [48, 109, 136]\nclass = c'),
 Text(2072.5714285714284, 543.5999999999999, 'CO <= 0.125\ngini = 0.397\nsamp
les = 411\nvalue = [73, 93, 518]\nclass = c'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.0\nsamples = 24\nvalu
e = [0, 41, 0]\nclass = b'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.332\nsamples = 387\nva
lue = [73, 52, 518]\nclass = c'),
 Text(3367.928571428571, 1630.8000000000002, 'OXY <= 4.135\ngini = 0.569\nsam
ples = 6262\nvalue = [5005, 838, 3992]\nclass = a'),
 Text(2869.7142857142853, 1268.4, 'BEN <= 2.335\ngini = 0.594\nsamples = 453
7\nvalue = [2928, 819, 3355]\nclass = c'),
 Text(2550.8571428571427, 906.0, 'MXY <= 1.605\ngini = 0.6\nsamples = 3227\nv
alue = [1739, 730, 2573]\nclass = c'),
 Text(2391.428571428571, 543.5999999999999, 'CO <= 0.21\ngini = 0.613\nsample
s = 294\nvalue = [114, 251, 115]\nclass = b'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.0\nsamples = 28\nvalu
e = [0, 44, 0]\nclass = b'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.637\nsamples = 266\nva
lue = [114, 207, 115]\nclass = b'),
 Text(2710.285714285714, 543.5999999999999, 'TCH <= 1.315\ngini = 0.572\nsamp
les = 2933\nvalue = [1625, 479, 2458]\nclass = c'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.431\nsamples = 635\nv
alue = [700, 55, 221]\nclass = a'),
 Text(2790.0, 181.19999999999982, 'gini = 0.53\nsamples = 2298\nvalue = [925,
424, 2237]\nclass = c'),
 Text(3188.5714285714284, 906.0, 'O_3 <= 3.88\ngini = 0.521\nsamples = 1310\n
value = [1189, 89, 782]\nclass = a'),
 Text(3029.142857142857, 543.5999999999999, 'MXY <= 7.715\ngini = 0.145\nsamp
les = 26\nvalue = [3, 35, 0]\nclass = b'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.062\nsamples = 20\nval
ue = [1, 30, 0]\nclass = b'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.408\nsamples = 6\nval
ue = [2, 5, 0]\nclass = b'),
 Text(3347.9999999999995, 543.5999999999999, 'TOL <= 14.175\ngini = 0.506\nsa
mples = 1284\nvalue = [1186, 54, 782]\nclass = a'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.454\nsamples = 941\nva
lue = [988, 28, 458]\nclass = a'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.518\nsamples = 343\nv
alue = [198, 26, 324]\nclass = c'),
 Text(3866.142857142857, 1268.4, 'NOx <= 156.3\ngini = 0.368\nsamples = 1725\
nvalue = [2077, 19, 637]\nclass = a'),
 Text(3587.142857142857, 906.0, 'PXY <= 3.22\ngini = 0.521\nsamples = 228\nva
lue = [183, 8, 168]\nclass = a'),
 Text(3507.428571428571, 543.5999999999999, 'gini = 0.278\nsamples = 9\nvalue
= [0, 3, 15]\nclass = c'),
 Text(3666.8571428571427, 543.5999999999999, 'PM10 <= 33.605\ngini = 0.51\nsa
mples = 219\nvalue = [183, 5, 153]\nclass = a'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.432\nsamples = 134\nva
lue = [145, 0, 67]\nclass = a'),
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.467\nsamples = 85\nva
lue = [38, 5, 86]\nclass = c'),
 Text(4145.142857142857, 906.0, 'NO_2 <= 77.76\ngini = 0.324\nsamples = 1497\
nvalue = [1894, 11, 469]\nclass = a'),
 Text(3985.7142857142853, 543.5999999999999, 'NO_2 <= 69.67\ngini = 0.485\nsa
mples = 81\nvalue = [52, 0, 74]\nclass = c'),
```

```
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.465\nsamples = 26\nva
lue = [24, 0, 14]\nclass = a'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.434\nsamples = 55\nval
ue = [28, 0, 60]\nclass = c'),
 Text(4304.571428571428, 543.5999999999999, 'O_3 <= 8.565\ngini = 0.298\nsamp
les = 1416\nvalue = [1842, 11, 395]\nclass = a'),
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.418\nsamples = 658\nva
lue = [728, 6, 296]\nclass = a'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.157\nsamples = 758\nva
lue = [1114, 5, 99]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression :0.10128873662984172

Ridge Regression : 0.0520104650845411

Lasso Regression : 0.054195477926028524

ElasticNet Regression : 0.0700726667853745

Logistic Regression : 0.9999978255573396

Random Forest : 0.7745449956179764

Logistic Regression is suitable for this dataset