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_2002.csv")
```

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

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', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32381 entries, 1 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     32381 non-null  object 
 1   BEN      32381 non-null  float64
 2   CO       32381 non-null  float64
 3   EBE      32381 non-null  float64
 4   MXY      32381 non-null  float64
 5   NMHC     32381 non-null  float64
 6   NO_2     32381 non-null  float64
 7   NOx      32381 non-null  float64
 8   OXY      32381 non-null  float64
 9   O_3      32381 non-null  float64
 10  PM10     32381 non-null  float64
 11  PXY      32381 non-null  float64
 12  SO_2     32381 non-null  float64
 13  TCH      32381 non-null  float64
 14  TOL      32381 non-null  float64
 15  station  32381 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.2+ MB
```

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

Out[6]:

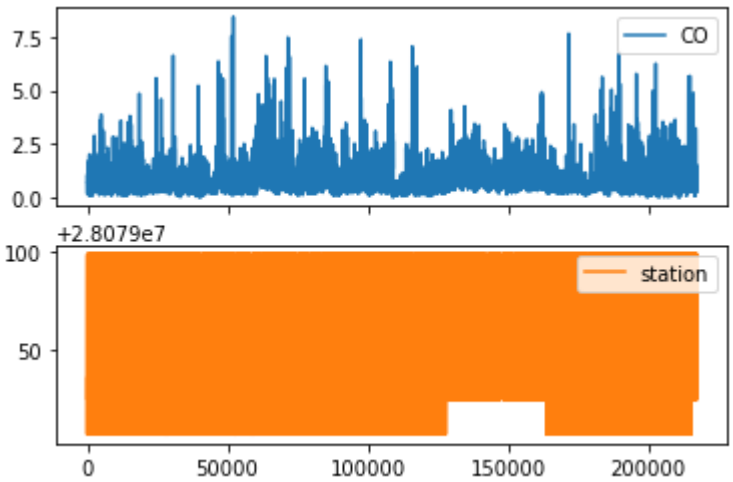
	CO	station
1	0.71	28079035
5	0.72	28079006
22	0.80	28079024
24	1.04	28079099
26	0.53	28079035
...	...	...
217269	0.28	28079024
217271	1.30	28079099
217273	0.97	28079035
217293	0.58	28079024
217295	1.17	28079099

32381 rows × 2 columns

Line chart

In [7]:

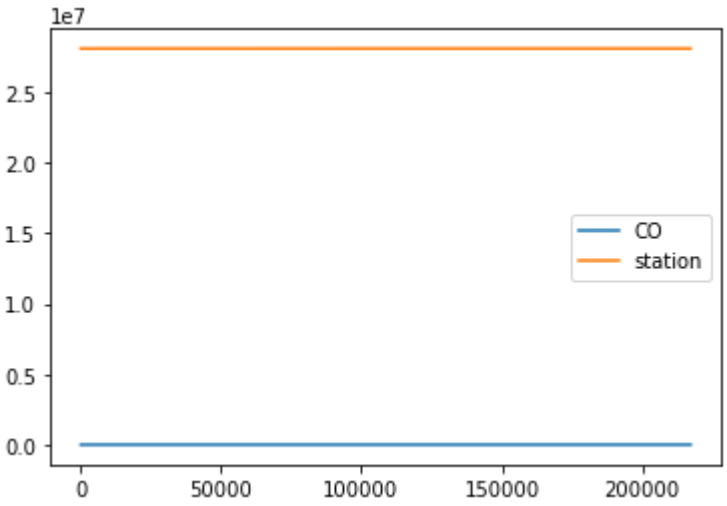
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

In [8]:

Out[8]: <AxesSubplot:>

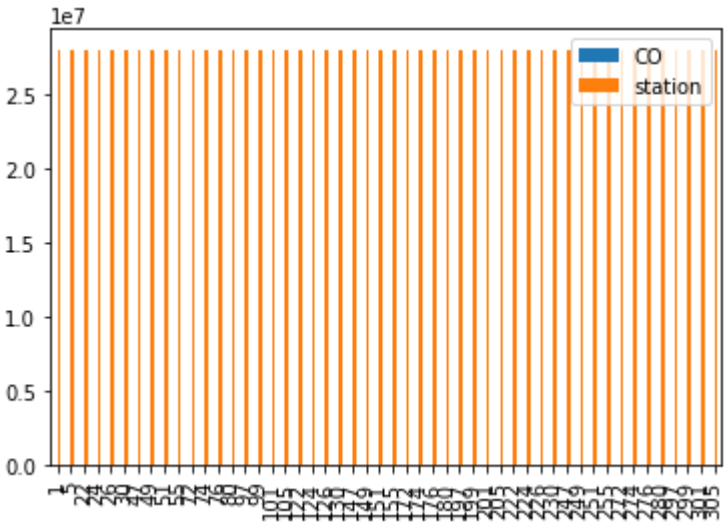


Bar chart

In [9]:

In [10]:

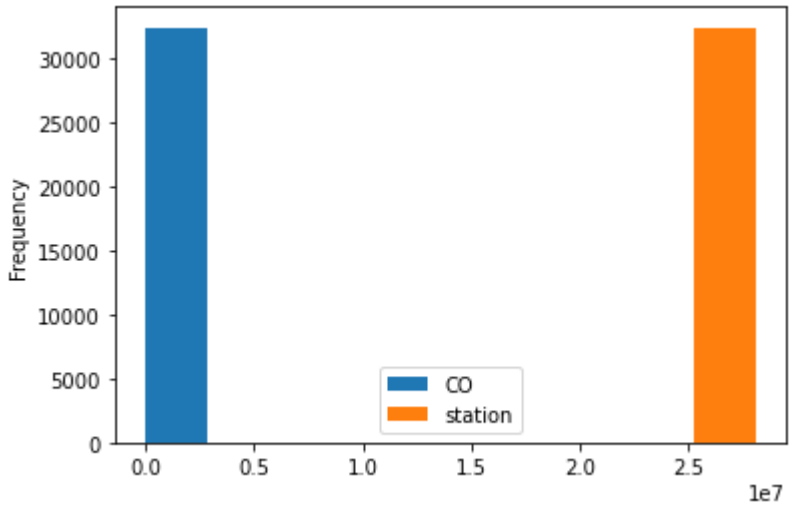
Out[10]: <AxesSubplot:>



Histogram

In [11]:

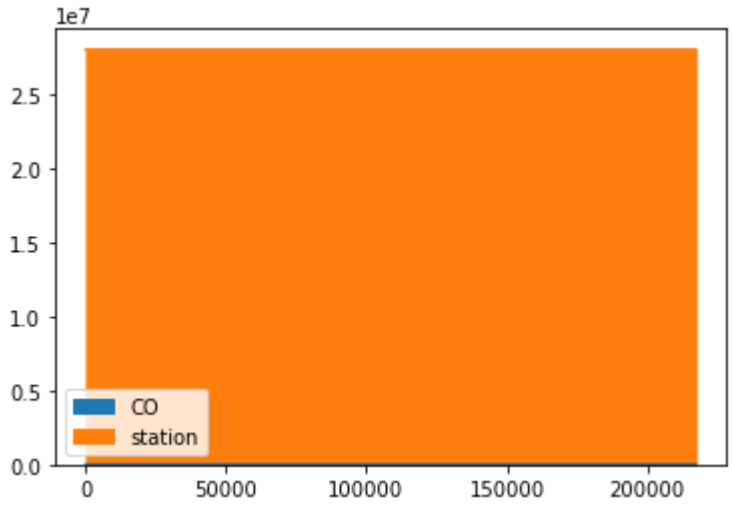
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:

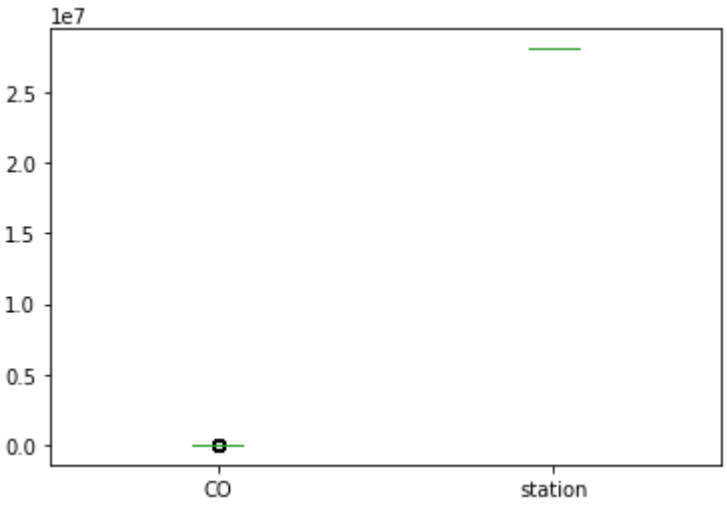
Out[12]: <AxesSubplot:>



Box chart

In [13]:

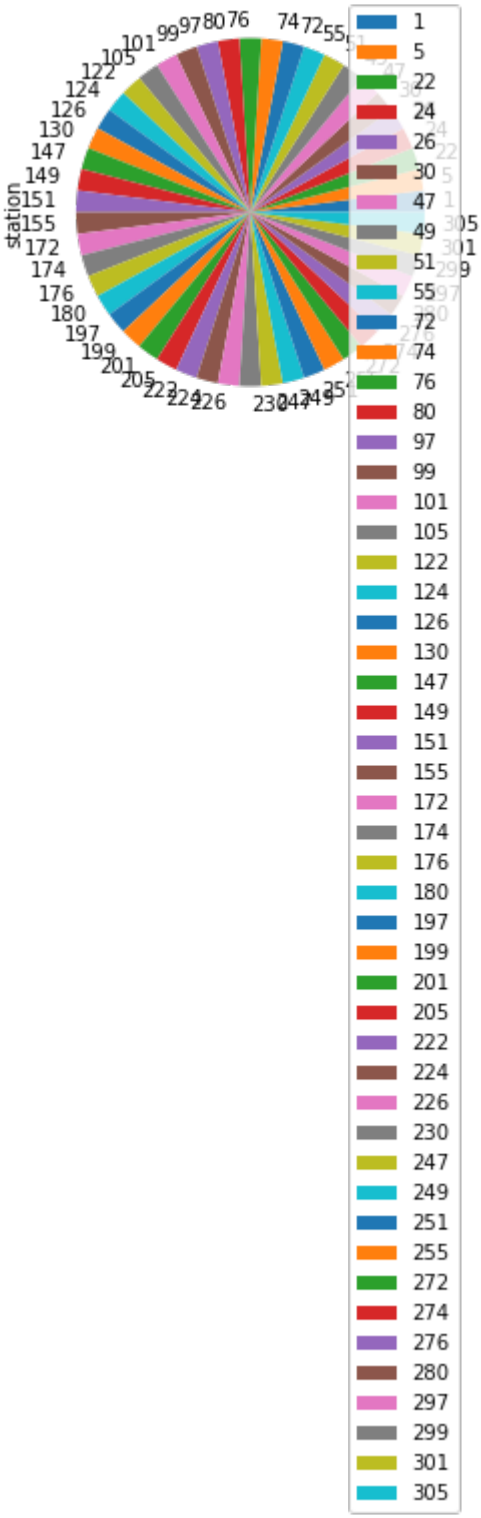
Out[13]: <AxesSubplot:>



Pie chart

In [14]:

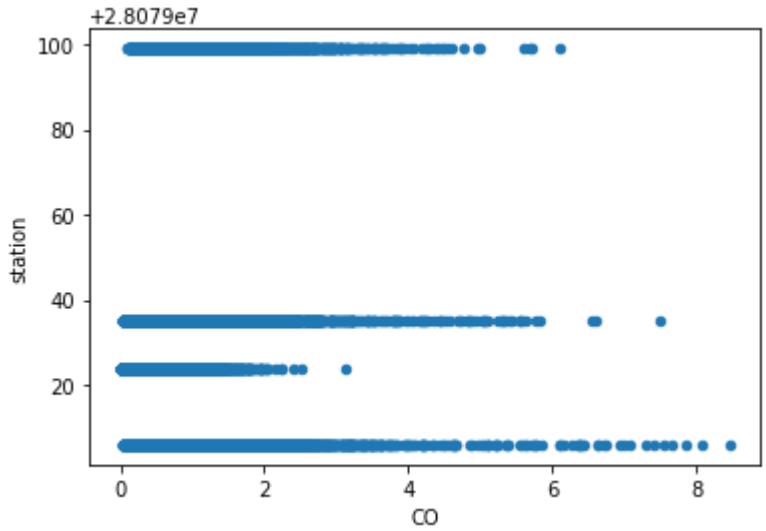
Out[14]: <AxesSubplot:ylabel='station'>



Scatter chart

In [15]:

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



In [16]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32381 entries, 1 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     32381 non-null  object 
 1   BEN      32381 non-null  float64
 2   CO       32381 non-null  float64
 3   EBE      32381 non-null  float64
 4   MXY      32381 non-null  float64
 5   NMHC     32381 non-null  float64
 6   NO_2     32381 non-null  float64
 7   NOx      32381 non-null  float64
 8   OXY      32381 non-null  float64
 9   O_3      32381 non-null  float64
 10  PM10     32381 non-null  float64
 11  PXY      32381 non-null  float64
 12  SO_2     32381 non-null  float64
 13  TCH      32381 non-null  float64
```

In [17]:

Out[17]:

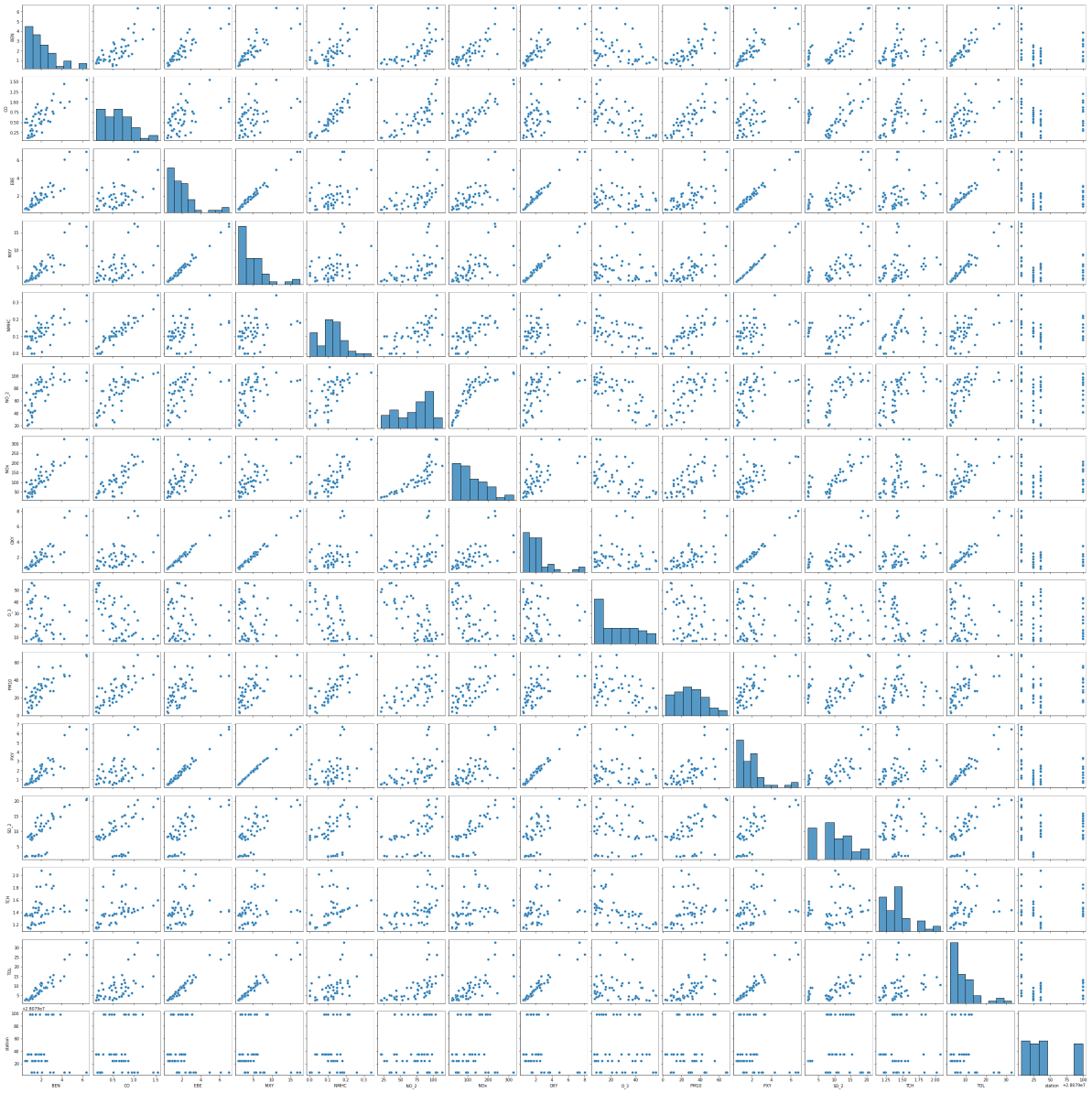
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	323
mean	2.479155	0.787323	2.914004	7.013636	0.155827	58.936796	1
std	2.280959	0.610810	2.667881	6.774365	0.135731	31.472733	1
min	0.180000	0.000000	0.180000	0.190000	0.000000	0.890000	
25%	0.970000	0.420000	1.140000	2.420000	0.080000	35.660000	
50%	1.840000	0.620000	2.130000	5.140000	0.130000	57.160000	
75%	3.250000	0.980000	3.830000	9.420000	0.200000	78.769997	1
max	32.660000	8.460000	41.740002	99.879997	2.700000	263.600006	13

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

EDA AND VISUALIZATION

In [19]:

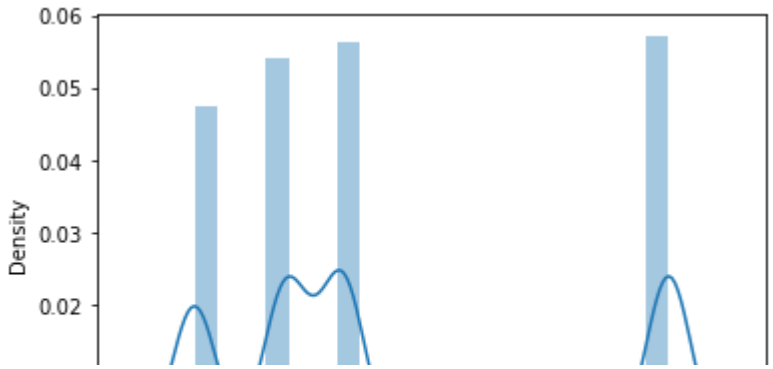
Out[19]: <seaborn.axisgrid.PairGrid at 0x17306ba3400>



In [20]:

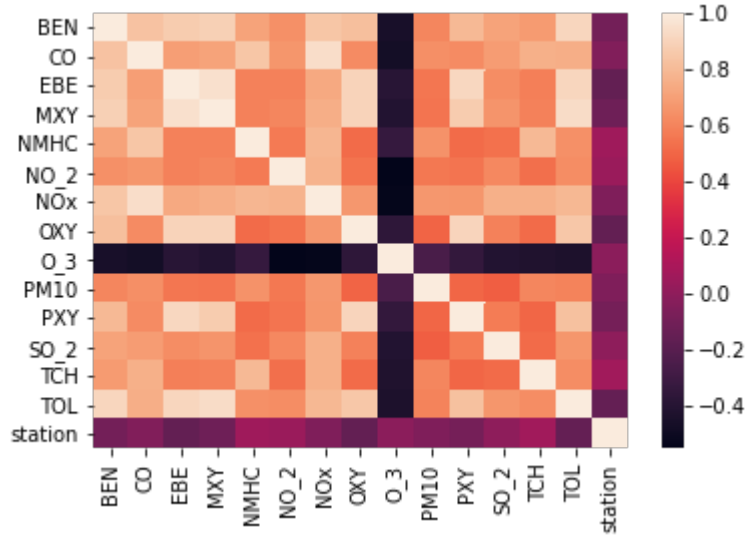
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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

In [23]:

```
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]:
```
28078992.450676896
```

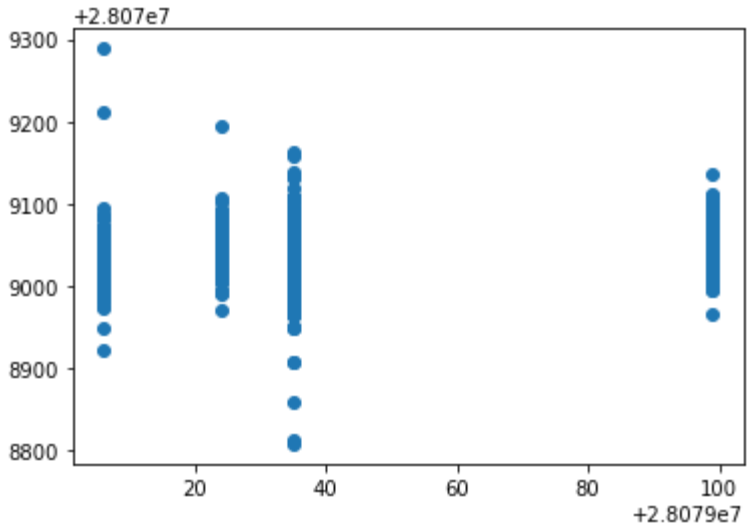
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[26]:

	Co-efficient
BEN	1.575128
CO	-14.345509
EBE	-11.362526
MXY	4.113362
NMHC	85.707203
NO_2	0.240931
NOx	-0.082249
OXY	-5.360800
O_3	-0.024806
PM10	-0.130848
PXY	7.964753
SO_2	0.535164
TCH	40.106519
TOL	-1.429867

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

Out[27]: <matplotlib.collections.PathCollection at 0x173163e7580>



ACCURACY

In [28]:

Out[28]: 0.20540165394087107

In [29]:

Out[29]: 0.1956676790822448

Ridge and Lasso

In [30]:

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

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:

Out[32]: 0.20505287518252757

In [33]:

Out[33]: 0.19544702567448624

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

Out[34]:
```
Lasso(alpha=10)
```

In [35]:

Out[35]:
```
0.05493421515458763
```

Accuracy(Lasso)

In [36]:

Out[36]:
```
0.05934587551963111
```

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

Out[37]:
```
ElasticNet()
```

In [38]:

Out[38]:
```
array([ 8.09427428e-01,  0.00000000e+00, -2.87739088e+00,  1.60918497e+00,
        2.05569488e-01,  2.18276131e-01, -1.96678093e-02, -2.48125952e+00,
       -1.95233114e-02,  1.48769312e-04,  2.28771347e+00,  3.37465269e-01,
        1.06777271e+00, -1.15488619e+00])
```

In [39]:

Out[39]:
```
28079038.695838805
```

In [40]:

In [41]:

Out[41]:
```
0.10366792051069895
```

Evaluation Metrics

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

```
28.6890120557367
1122.34559538234
33.501426766368326
```

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]: (32381, 14)

In [46]:

Out[46]: (32381,)

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)
```

[28079035]

In [52]:

Out[52]: array([28079006, 28079024, 28079035, 28079099], dtype=int64)

In [53]:

Out[53]: 0.8480899292795158

In [54]:

Out[54]: 2.5638972732451705e-10

In [55]:

Out[55]: array([[2.56389727e-10, 3.44199742e-71, 1.00000000e+00, 1.43898646e-13]])

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]:
```
0.7756992852730963
```

In [61]:

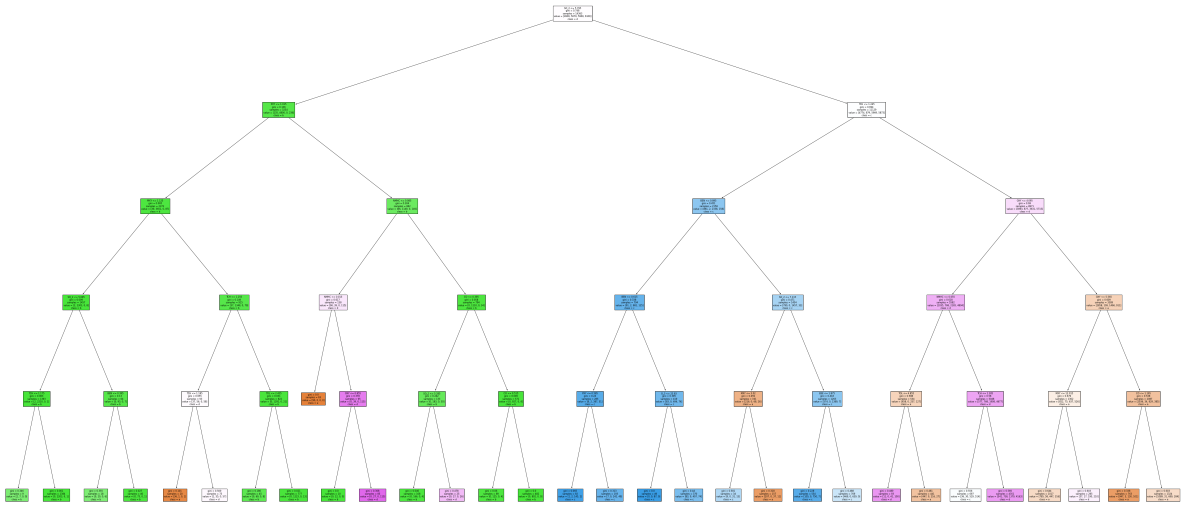
In [62]:
```
from sklearn.tree import plot_tree

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

Out[62]:
```
[Text(2166.9, 1993.2, 'SO_2 <= 5.795\ngini = 0.748\nsamples = 14363\nvalue =
[4909, 5679, 5969, 6109]\nclass = d'),
 Text(1060.2, 1630.8000000000002, 'PXY <= 1.015\ngini = 0.136\nsamples = 323
4\nvalue = [135, 4800, 0, 236]\nclass = b'),
 Text(595.2, 1268.4, 'MXY <= 1.125\ngini = 0.065\nsamples = 2372\nvalue = [3
9, 3651, 0, 87]\nclass = b'),
 Text(297.6, 906.0, 'SO_2 <= 5.005\ngini = 0.009\nsamples = 1457\nvalue = [2,
2303, 0, 8]\nclass = b'),
 Text(148.8, 543.5999999999999, 'TCH <= 1.175\ngini = 0.003\nsamples = 1393\n
value = [2, 2210, 0, 1]\nclass = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.346\nsamples = 9\nvalue = [2, 7, 0,
0]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.001\nsamples = 1384\n
value = [0, 2203, 0, 1]\nclass = b'),
 Text(446.40000000000003, 543.5999999999999, 'BEN <= 0.395\ngini = 0.13\nsamp
les = 64\nvalue = [0, 93, 0, 7]\nclass = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.355\nsamples = 18\nvalue = [0, 20,
0, 6]\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.027\nsamples = 46\nval
ue = [0, 73, 0, 1]\nclass = b'),
 Text(892.8000000000001, 906.0, 'TCH <= 1.255\ngini = 0.149\nsamples = 915\nv
alue = [37, 1348, 0, 79]\nclass = b'),
 Text(744.0, 543.5999999999999, 'TCH <= 1.185\ngini = 0.655\nsamples = 93\nva
lue = [37, 56, 0, 58]\nclass = d'),
 Text(669.6, 181.19999999999982, 'gini = 0.101\nsamples = 20\nvalue = [36, 1,
0, 1]\nclass = a'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.509\nsamples = 73\nval
ue = [1, 55, 0, 57]\nclass = d'),
 Text(1041.6000000000001, 543.5999999999999, 'TOL <= 2.425\ngini = 0.031\nsam
ples = 822\nvalue = [0, 1292, 0, 21]\nclass = b'),
 Text(967.2, 181.19999999999982, 'gini = 0.186\nsamples = 45\nvalue = [0, 69,
0, 8]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.021\nsamples = 777\nvalue = [0, 1
223, 0, 13]\nclass = b'),
 Text(1525.2, 1268.4, 'NMHC <= 0.085\ngini = 0.304\nsamples = 862\nvalue = [9
6, 1149, 0, 149]\nclass = b'),
 Text(1264.8000000000002, 906.0, 'NMHC <= 0.015\ngini = 0.617\nsamples = 158\
nvalue = [96, 39, 0, 115]\nclass = d'),
 Text(1190.4, 543.5999999999999, 'gini = 0.0\nsamples = 63\nvalue = [96, 0,
0, 0]\nclass = a'),
 Text(1339.2, 543.5999999999999, 'OXY <= 0.855\ngini = 0.378\nsamples = 95\nv
alue = [0, 39, 0, 115]\nclass = d'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.0\nsamples = 10\nvalu
e = [0, 12, 0, 0]\nclass = b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.308\nsamples = 85\nva
lue = [0, 27, 0, 115]\nclass = d'),
 Text(1785.6000000000001, 906.0, 'CO <= 0.395\ngini = 0.058\nsamples = 704\nv
alue = [0, 1110, 0, 34]\nclass = b'),
 Text(1636.8000000000002, 543.5999999999999, 'SO_2 <= 5.285\ngini = 0.242\nsa
mples = 133\nvalue = [0, 183, 0, 30]\nclass = b'),
 Text(1562.4, 181.19999999999982, 'gini = 0.046\nsamples = 108\nvalue = [0, 1
66, 0, 4]\nclass = b'),
```

```
 Text(1711.2, 181.19999999999982, 'gini = 0.478\nsamples = 25\nvalue = [0, 1
7, 0, 26]\nclass = d'),
 Text(1934.4, 543.5999999999999, 'CO <= 0.515\ngini = 0.009\nsamples = 571\nv
alue = [0, 927, 0, 4]\nclass = b'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.06\nsamples = 89\nval
ue = [0, 125, 0, 4]\nclass = b'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.0\nsamples = 482\nval
ue = [0, 802, 0, 0]\nclass = b'),
 Text(3273.6000000000004, 1630.8000000000002, 'TCH <= 1.245\ngini = 0.694\nsa
mples = 11129\nvalue = [4774, 879, 5969, 5873]\nclass = c'),
 Text(2678.4, 1268.4, 'BEN <= 0.895\ngini = 0.451\nsamples = 2158\nvalue = [8
81, 2, 2338, 158]\nclass = c'),
 Text(2380.8, 906.0, 'BEN <= 0.615\ngini = 0.338\nsamples = 704\nvalue = [91,
2, 881, 125]\nclass = c'),
 Text(2232.0, 543.5999999999999, 'PXY <= 0.585\ngini = 0.24\nsamples = 285\nv
alue = [8, 2, 387, 51]\nclass = c'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.065\nsamples = 92\nva
lue = [1, 2, 145, 2]\nclass = c'),
 Text(2306.4, 181.19999999999982, 'gini = 0.313\nsamples = 193\nvalue = [7,
0, 242, 49]\nclass = c'),
 Text(2529.6000000000004, 543.5999999999999, 'O_3 <= 23.83\ngini = 0.395\nsam
ples = 419\nvalue = [83, 0, 494, 74]\nclass = c'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.0\nsamples = 49\nvalu
e = [0, 0, 87, 0]\nclass = c'),
 Text(2604.0, 181.19999999999982, 'gini = 0.44\nsamples = 370\nvalue = [83,
0, 407, 74]\nclass = c'),
 Text(2976.0, 906.0, 'SO_2 <= 7.115\ngini = 0.471\nsamples = 1454\nvalue = [7
90, 0, 1457, 33]\nclass = c'),
 Text(2827.2000000000003, 543.5999999999999, 'MXY <= 3.02\ngini = 0.459\nsamp
les = 191\nvalue = [216, 0, 68, 26]\nclass = a'),
 Text(2752.8, 181.19999999999982, 'gini = 0.581\nsamples = 34\nvalue = [9, 0,
31, 15]\nclass = c'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.318\nsamples = 157\nv
alue = [207, 0, 37, 11]\nclass = a'),
 Text(3124.8, 543.5999999999999, 'EBE <= 1.875\ngini = 0.418\nsamples = 1263\
nvalue = [574, 0, 1389, 7]\nclass = c'),
 Text(3050.4, 181.19999999999982, 'gini = 0.228\nsamples = 554\nvalue = [105,
0, 750, 7]\nclass = c'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.488\nsamples = 709\nv
alue = [469, 0, 639, 0]\nclass = c'),
 Text(3868.8, 1268.4, 'OXY <= 4.095\ngini = 0.69\nsamples = 8971\nvalue = [38
93, 877, 3631, 5715]\nclass = d'),
 Text(3571.2000000000003, 906.0, 'NMHC <= 0.055\ngini = 0.618\nsamples = 558
2\nvalue = [1035, 768, 2165, 4804]\nclass = d'),
 Text(3422.4, 543.5999999999999, 'TOL <= 4.855\ngini = 0.588\nsamples = 534\n
value = [458, 0, 257, 127]\nclass = a'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.489\nsamples = 93\nva
lue = [11, 0, 41, 100]\nclass = d'),
 Text(3496.8, 181.19999999999982, 'gini = 0.481\nsamples = 441\nvalue = [447,
0, 216, 27]\nclass = a'),
 Text(3720.0000000000005, 543.5999999999999, 'TCH <= 1.285\ngini = 0.58\nsamp
les = 5048\nvalue = [577, 768, 1908, 4677]\nclass = d'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.556\nsamples = 697\nv
alue = [36, 30, 529, 514]\nclass = c'),
 Text(3794.4, 181.19999999999982, 'gini = 0.569\nsamples = 4351\nvalue = [54
1, 738, 1379, 4163]\nclass = d'),
```

```
 Text(4166.400000000001, 906.0, 'OXY <= 5.565\ngini = 0.609\nsamples = 3389\n
value = [2858, 109, 1466, 911]\nclass = a'),
 Text(4017.6000000000004, 543.5999999999999, 'MXY <= 12.015\ngini = 0.679\nsa
mples = 1302\nvalue = [822, 75, 637, 526]\nclass = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.644\nsamples = 1017\n
value = [785, 58, 447, 316]\nclass = a'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.603\nsamples = 285\nv
alue = [37, 17, 190, 210]\nclass = d'),
 Text(4315.200000000001, 543.5999999999999, 'CO <= 1.165\ngini = 0.538\nsampl
es = 2087\nvalue = [2036, 34, 829, 385]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.338\nsamples = 763\nvalue = [947,
3, 130, 101]\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.603\nsamples = 1324\nvalue = [108
9, 31, 699, 284]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression : 0.1956676790822448

Ridge Regression : 0.05493421515458763

Lasso Regression : 0.10366792051069895

ElasticNet Regression : 28079038.695838805

Logistic Regression : 0.8480899292795158

Random Forest : 0.7756992852730963

Logistic Regression is suitable for this dataset