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_2001.csv")
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	10
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	10
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	10
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	6
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	7
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	4
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	2
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	4
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	3
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	3

217872 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: 29669 entries, 1 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     29669 non-null  object 
 1   BEN      29669 non-null  float64
 2   CO       29669 non-null  float64
 3   EBE      29669 non-null  float64
 4   MXY      29669 non-null  float64
 5   NMHC     29669 non-null  float64
 6   NO_2     29669 non-null  float64
 7   NOx      29669 non-null  float64
 8   OXY      29669 non-null  float64
 9   O_3      29669 non-null  float64
 10  PM10     29669 non-null  float64
 11  PXY      29669 non-null  float64
 12  SO_2     29669 non-null  float64
 13  TCH      29669 non-null  float64
 14  TOL      29669 non-null  float64
 15  station  29669 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 3.8+ MB
```

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

Out[6]:

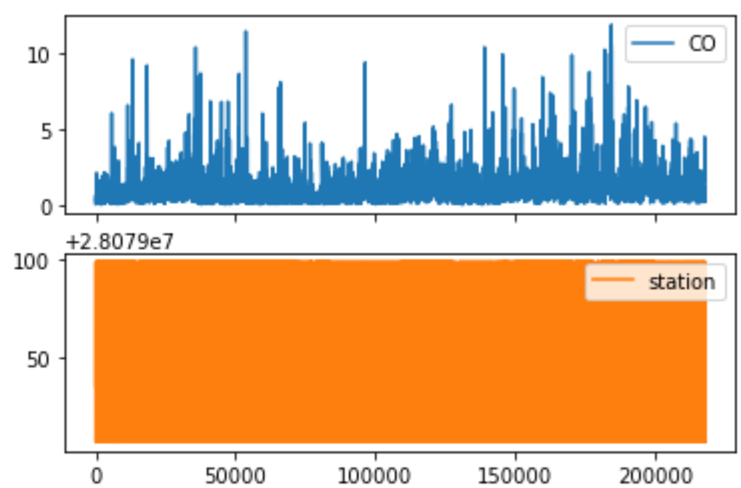
	CO	station
1	0.34	28079035
5	0.63	28079006
21	0.43	28079024
23	0.34	28079099
25	0.06	28079035
...	...	...
217829	4.48	28079006
217847	2.65	28079099
217849	1.22	28079035
217853	1.83	28079006
217871	1.62	28079099

29669 rows × 2 columns

Line chart

In [7]:

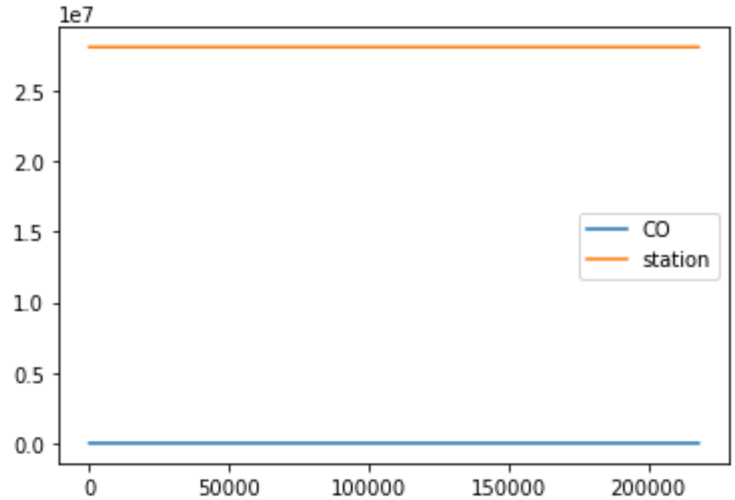
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

In [8]:

Out[8]: <AxesSubplot:>

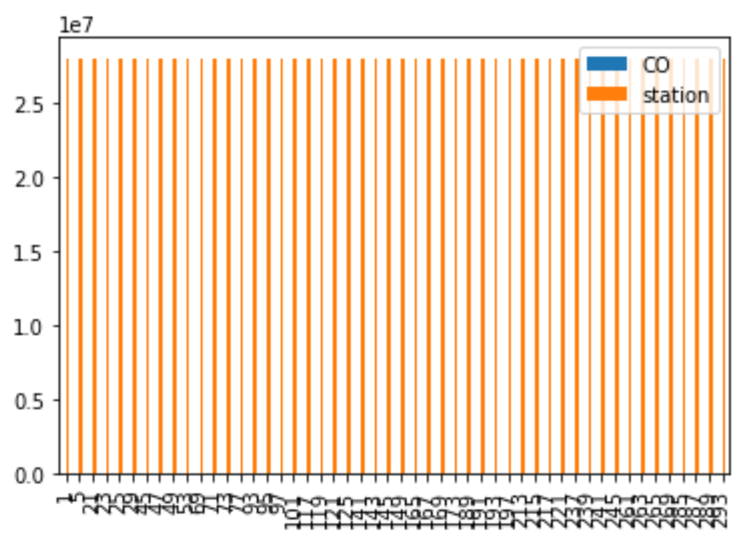


Bar chart

In [9]:

In [10]:

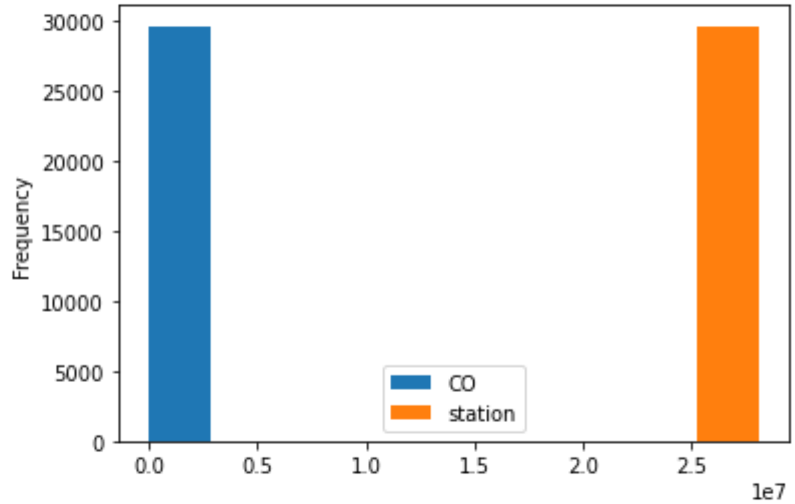
Out[10]: <AxesSubplot:>



Histogram

In [11]:

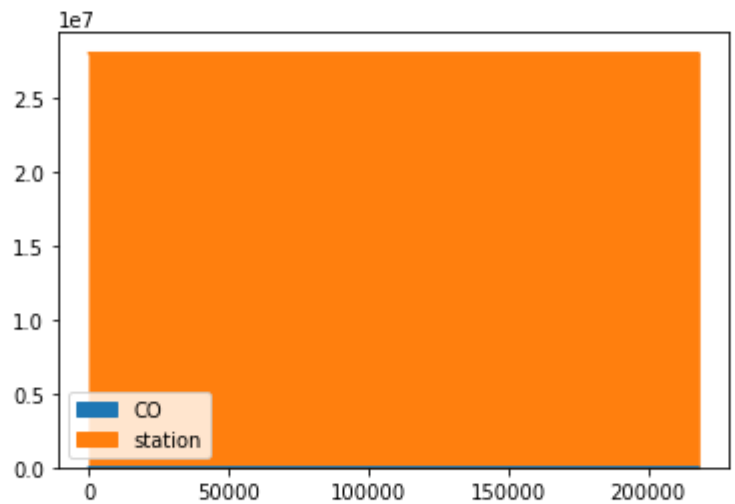
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:

Out[12]: <AxesSubplot:>

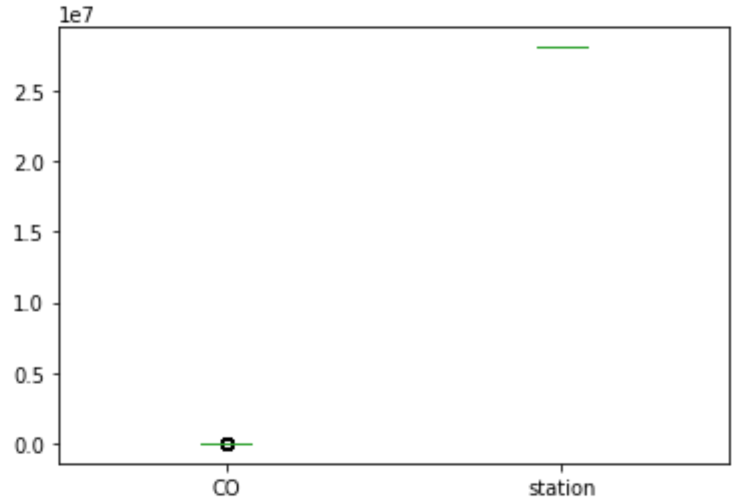


Box chart

In [13]:

```
[illegible]
```

Out[13]: <AxesSubplot:>



Pie chart

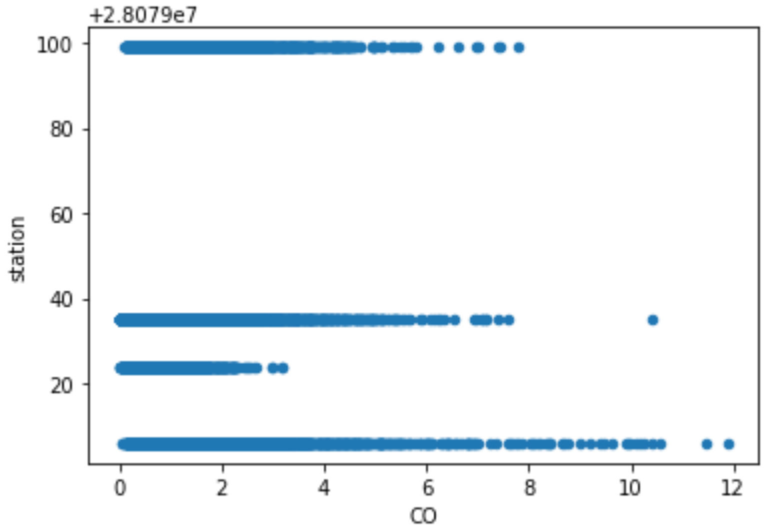
In [14]:

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

Scatter chart

In [15]:

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



In [16]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29669 entries, 1 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     29669 non-null  object 
 1   BEN      29669 non-null  float64
 2   CO       29669 non-null  float64
 3   EBE      29669 non-null  float64
 4   MXY      29669 non-null  float64
 5   NMHC     29669 non-null  float64
 6   NO_2     29669 non-null  float64
 7   NOx      29669 non-null  float64
 8   OXY      29669 non-null  float64
 9   O_3      29669 non-null  float64
 10  PM10     29669 non-null  float64
 11  PXY      29669 non-null  float64
 12  SO_2     29669 non-null  float64
 13  TCH      29669 non-null  float64
```

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

Out[17]:

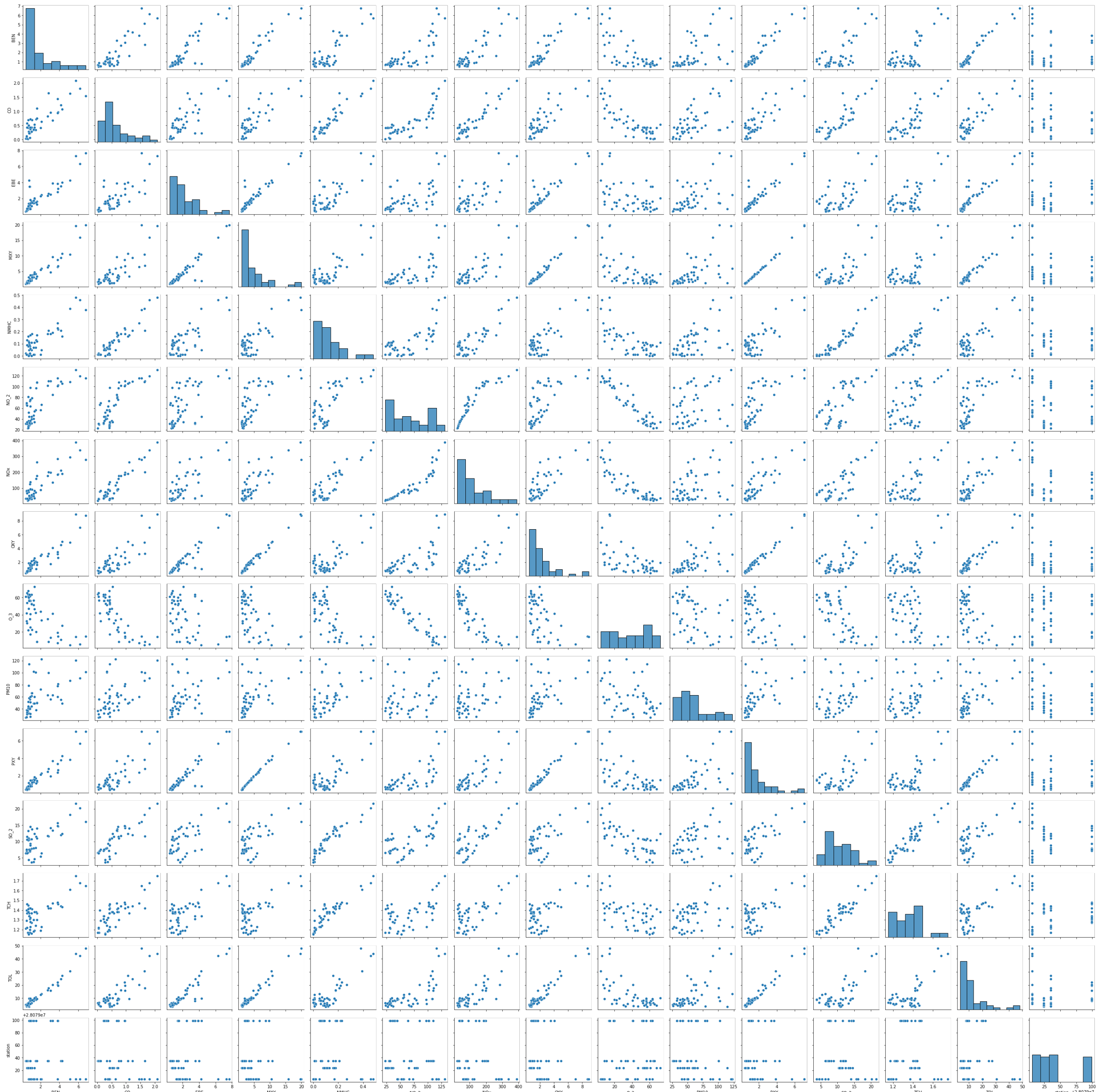
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	296
mean	3.361895	1.005413	3.580229	8.113086	0.195222	67.652292	1
std	3.176669	0.863135	3.744496	7.909701	0.192585	34.003120	1
min	0.100000	0.000000	0.140000	0.210000	0.000000	1.180000	
25%	1.280000	0.470000	1.390000	3.040000	0.080000	44.299999	
50%	2.510000	0.760000	2.600000	5.830000	0.140000	64.449997	1
75%	4.420000	1.270000	4.580000	10.640000	0.250000	86.540001	2
max	54.560001	11.890000	77.260002	150.600006	2.880000	292.700012	19

In [18]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [19]:
```
sns.pairplot(df1[0:50])
```

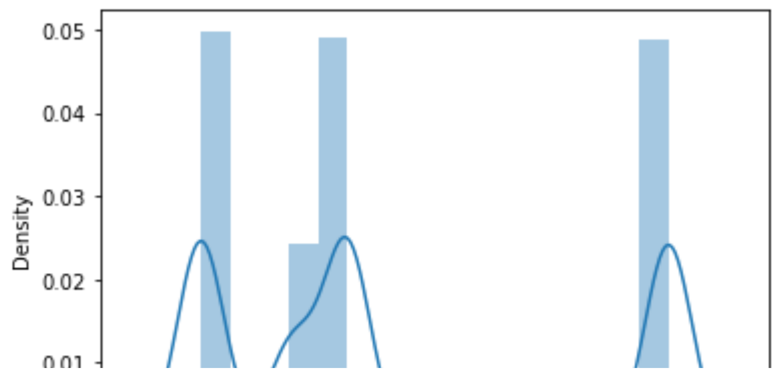
Out[19]: <seaborn.axisgrid.PairGrid at 0x15de4ed45e0>



In [20]:

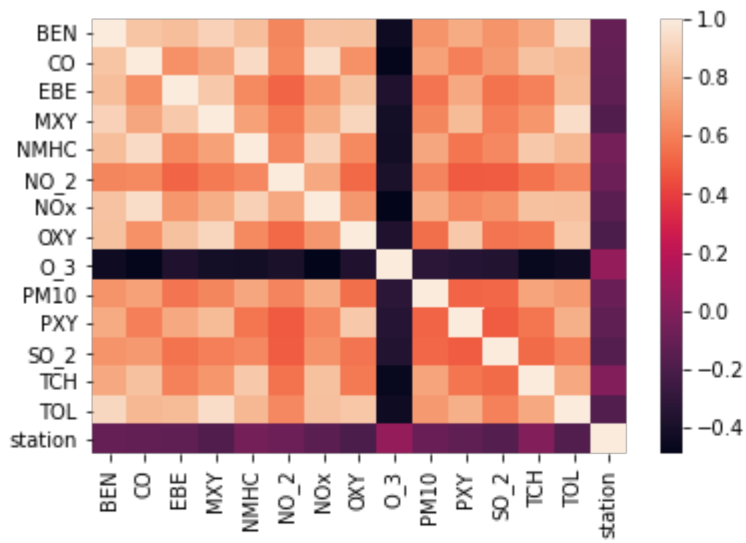
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  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]:
```
28079007.830716707
```

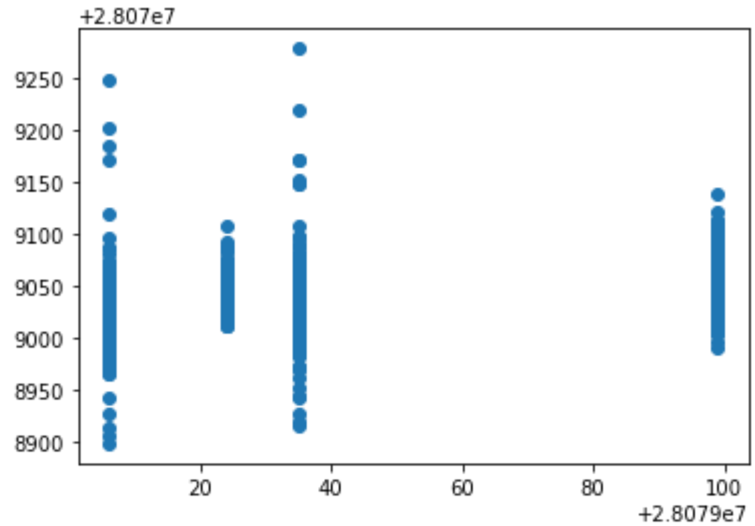
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[26]:

	Co-efficient
BEN	7.510963
CO	-15.360601
EBE	0.584675
MXY	-0.062319
NMHC	83.529834
NO_2	0.105531
NOx	-0.083335
OXY	-3.222834
O_3	-0.029922
PM10	-0.068388
PXY	1.535639
SO_2	-0.297259
TCH	36.864419
TOL	-1.295623

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

Out[27]: <matplotlib.collections.PathCollection at 0x15df45f3c10>



ACCURACY

In [28]:

Out[28]: 0.15333059191475773

In [29]:

Out[29]: 0.16966115058624776

Ridge and Lasso

In [30]:

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

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:

Out[32]: 0.15336555741871216

In [33]:

Out[33]: 0.16939908379553004

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

In [35]:

Out[35]: 0.03909802571844945

Accuracy(Lasso)

In [36]:

Out[36]: 0.03896350073644961

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

In [38]:

Out[38]:
```
array([ 5.1196482 ,  0.00858417,  0.58524365, -0.22278153,  0.09260819,
        0.05634334, -0.03149443, -2.55830102, -0.03761492,  0.06973542,
        0.96828664, -0.3129868 ,  1.20899921, -0.7295048 ])
```

In [39]:

Out[39]: 28079049.39181029

In [40]:

In [41]:

Out[41]: 0.09871426228846358

Evaluation Metrics

In [42]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
```
```
30.47036132577808
1226.7311838312069
35.024722466155346
```

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]: (29669, 14)

In [46]:

Out[46]: (29669,)

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]: 0.8087229094340894

In [54]:

Out[54]: 1.724527777144498e-43

In [55]:

Out[55]: array([[1.72452778e-43, 2.43756289e-56, 9.99998565e-01, 1.43537418e-06]])

Random Forest

In [56]:

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

Out[57]: RandomForestClassifier()

In [67]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
```

In [68]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="ac
```

Out[68]:
```
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 [69]:

Out[69]:
```
0.7331953004622496
```

In [61]:

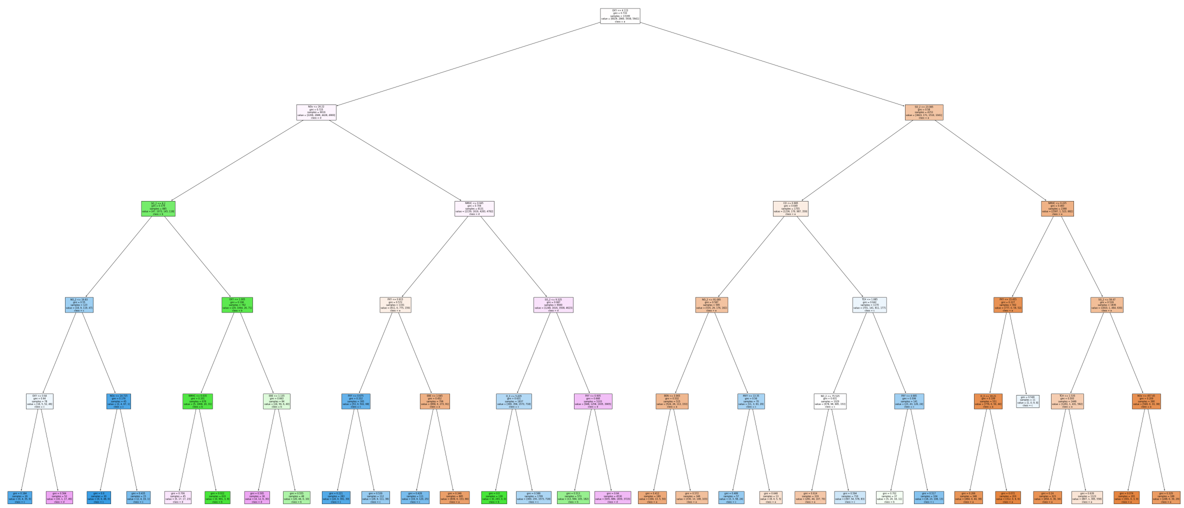
In [70]:
```
from sklearn.tree import plot_tree

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

Out[70]:
```
[Text(2334.3, 1993.2, 'OXY <= 4.115\ngini = 0.733\nsamples = 13169\nvalue =
[6029, 2860, 5938, 5941]\nclass = a'),
 Text(1190.4, 1630.8000000000002, 'NOx <= 26.22\ngini = 0.725\nsamples = 901
6\nvalue = [2206, 2689, 4428, 4900]\nclass = d'),
 Text(595.2, 1268.4, 'SO_2 <= 8.2\ngini = 0.379\nsamples = 885\nvalue = [47,
1073, 145, 118]\nclass = b'),
 Text(297.6, 906.0, 'NO_2 <= 19.83\ngini = 0.55\nsamples = 123\nvalue = [18,
9, 119, 47]\nclass = c'),
 Text(148.8, 543.5999999999999, 'OXY <= 0.64\ngini = 0.64\nsamples = 78\nvalu
e = [16, 5, 52, 46]\nclass = c'),
 Text(74.4, 181.19999999999982, 'gini = 0.184\nsamples = 26\nvalue = [0, 4, 3
5, 0]\nclass = c'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.584\nsamples = 52\nva
lue = [16, 1, 17, 46]\nclass = d'),
 Text(446.40000000000003, 543.5999999999999, 'NOx <= 24.725\ngini = 0.176\nsa
mples = 45\nvalue = [2, 4, 67, 1]\nclass = c'),
 Text(372.0, 181.19999999999982, 'gini = 0.0\nsamples = 30\nvalue = [0, 0, 4
8, 0]\nclass = c'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.435\nsamples = 15\nval
ue = [2, 4, 19, 1]\nclass = c'),
 Text(892.8000000000001, 906.0, 'OXY <= 1.005\ngini = 0.196\nsamples = 762\nv
alue = [29, 1064, 26, 71]\nclass = b'),
 Text(744.0, 543.5999999999999, 'NMHC <= 0.035\ngini = 0.101\nsamples = 678\n
value = [5, 1008, 20, 31]\nclass = b'),
 Text(669.6, 181.19999999999982, 'gini = 0.706\nsamples = 45\nvalue = [5, 17,
17, 23]\nclass = d'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.022\nsamples = 633\nva
lue = [0, 991, 3, 8]\nclass = b'),
 Text(1041.6000000000001, 543.5999999999999, 'EBE <= 1.135\ngini = 0.663\nsam
ples = 84\nvalue = [24, 56, 6, 40]\nclass = b'),
 Text(967.2, 181.19999999999982, 'gini = 0.595\nsamples = 36\nvalue = [4, 12,
6, 30]\nclass = d'),
 Text(1116.0, 181.19999999999982, 'gini = 0.555\nsamples = 48\nvalue = [20, 4
4, 0, 10]\nclass = b'),
 Text(1785.6000000000001, 1268.4, 'NMHC <= 0.045\ngini = 0.706\nsamples = 813
1\nvalue = [2159, 1616, 4283, 4782]\nclass = d'),
 Text(1488.0, 906.0, 'PXY <= 0.815\ngini = 0.572\nsamples = 1191\nvalue = [91
1, 0, 775, 159]\nclass = a'),
 Text(1339.2, 543.5999999999999, 'PXY <= 0.675\ngini = 0.332\nsamples = 395\n
value = [53, 0, 502, 68]\nclass = c'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.221\nsamples = 283\nv
alue = [24, 0, 391, 30]\nclass = c'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.539\nsamples = 112\nv
alue = [29, 0, 111, 38]\nclass = c'),
 Text(1636.8000000000002, 543.5999999999999, 'EBE <= 1.045\ngini = 0.452\nsam
ples = 796\nvalue = [858, 0, 273, 91]\nclass = a'),
 Text(1562.4, 181.19999999999982, 'gini = 0.428\nsamples = 111\nvalue = [19,
0, 120, 25]\nclass = c'),
 Text(1711.2, 181.19999999999982, 'gini = 0.346\nsamples = 685\nvalue = [839,
0, 153, 66]\nclass = a'),
 Text(2083.2000000000003, 906.0, 'SO_2 <= 9.325\ngini = 0.687\nsamples = 694
0\nvalue = [1248, 1616, 3508, 4623]\nclass = d'),
```

```
 Text(1934.4, 543.5999999999999, 'O_3 <= 5.435\ngini = 0.631\nsamples = 1837\
nvalue = [300, 358, 1573, 718]\nclass = c'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.0\nsamples = 128\nval
ue = [0, 203, 0, 0]\nclass = b'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.588\nsamples = 1709\n
value = [300, 155, 1573, 718]\nclass = c'),
 Text(2232.0, 543.5999999999999, 'PXY <= 0.905\ngini = 0.668\nsamples = 5103\
nvalue = [948, 1258, 1935, 3905]\nclass = d'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.513\nsamples = 573\nv
alue = [13, 569, 105, 182]\nclass = b'),
 Text(2306.4, 181.19999999999982, 'gini = 0.64\nsamples = 4530\nvalue = [935,
689, 1830, 3723]\nclass = d'),
 Text(3478.2000000000003, 1630.8000000000002, 'SO_2 <= 23.085\ngini = 0.58\ns
amples = 4153\nvalue = [3823, 171, 1510, 1041]\nclass = a'),
 Text(2976.0, 1268.4, 'CO <= 0.985\ngini = 0.649\nsamples = 1755\nvalue = [12
36, 170, 987, 359]\nclass = a'),
 Text(2678.4, 906.0, 'NO_2 <= 91.085\ngini = 0.587\nsamples = 585\nvalue = [5
35, 29, 176, 182]\nclass = a'),
 Text(2529.6000000000004, 543.5999999999999, 'BEN <= 3.065\ngini = 0.532\nsam
ples = 515\nvalue = [524, 26, 113, 153]\nclass = a'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.412\nsamples = 169\nv
alue = [194, 13, 5, 50]\nclass = a'),
 Text(2604.0, 181.19999999999982, 'gini = 0.572\nsamples = 346\nvalue = [330,
13, 108, 103]\nclass = a'),
 Text(2827.2000000000003, 543.5999999999999, 'MXY <= 13.35\ngini = 0.56\nsamp
les = 70\nvalue = [11, 3, 63, 29]\nclass = c'),
 Text(2752.8, 181.19999999999982, 'gini = 0.489\nsamples = 57\nvalue = [3, 3,
58, 24]\nclass = c'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.648\nsamples = 13\nva
lue = [8, 0, 5, 5]\nclass = a'),
 Text(3273.6000000000004, 906.0, 'TCH <= 1.885\ngini = 0.642\nsamples = 1170\
nvalue = [701, 141, 811, 177]\nclass = c'),
 Text(3124.8, 543.5999999999999, 'NO_2 <= 75.525\ngini = 0.631\nsamples = 102
9\nvalue = [678, 98, 685, 153]\nclass = c'),
 Text(3050.4, 181.19999999999982, 'gini = 0.614\nsamples = 319\nvalue = [281,
44, 107, 70]\nclass = a'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.594\nsamples = 710\nv
alue = [397, 54, 578, 83]\nclass = c'),
 Text(3422.4, 543.5999999999999, 'PXY <= 4.685\ngini = 0.596\nsamples = 141\n
value = [23, 43, 126, 24]\nclass = c'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.702\nsamples = 37\nva
lue = [5, 20, 18, 11]\nclass = b'),
 Text(3496.8, 181.19999999999982, 'gini = 0.517\nsamples = 104\nvalue = [18,
23, 108, 13]\nclass = c'),
 Text(3980.4, 1268.4, 'NMHC <= 0.225\ngini = 0.483\nsamples = 2398\nvalue =
[2587, 1, 523, 682]\nclass = a'),
 Text(3794.4, 906.0, 'PXY <= 15.455\ngini = 0.227\nsamples = 562\nvalue = [77
7, 0, 59, 52]\nclass = a'),
 Text(3720.0000000000005, 543.5999999999999, 'O_3 <= 19.21\ngini = 0.199\nsam
ples = 551\nvalue = [776, 0, 50, 44]\nclass = a'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.266\nsamples = 348\nv
alue = [464, 0, 44, 38]\nclass = a'),
 Text(3794.4, 181.19999999999982, 'gini = 0.072\nsamples = 203\nvalue = [312,
0, 6, 6]\nclass = a'),
 Text(3868.8, 543.5999999999999, 'gini = 0.549\nsamples = 11\nvalue = [1, 0,
9, 8]\nclass = c'),
```

```
 Text(4166.400000000001, 906.0, 'SO_2 <= 59.47\ngini = 0.539\nsamples = 1836\
nvalue = [1810, 1, 464, 630]\nclass = a'),
 Text(4017.6000000000004, 543.5999999999999, 'TCH <= 1.535\ngini = 0.593\nsam
ples = 1446\nvalue = [1261, 1, 431, 592]\nclass = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.24\nsamples = 323\nva
lue = [454, 0, 36, 34]\nclass = a'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.639\nsamples = 1123\n
value = [807, 1, 395, 558]\nclass = a'),
 Text(4315.200000000001, 543.5999999999999, 'NOx <= 457.45\ngini = 0.209\nsam
ples = 390\nvalue = [549, 0, 33, 38]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.074\nsamples = 201\nvalue = [301,
0, 3, 9]\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.329\nsamples = 189\nvalue = [248,
0, 30, 29]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression:0.15333059191475773

Ridge Regression:0.15336555741871216

Lasso Regression:0.03896350073644961

ElasticNet Regression:0.09871426228846358

Logistic Regression:0.8087229094340894

Random Forest:0.7331953004622496

Logistic Regression is suitable for this dataset