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_2013.csv")
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN

209880 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.fillna(1)
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25
',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

Out[6]:

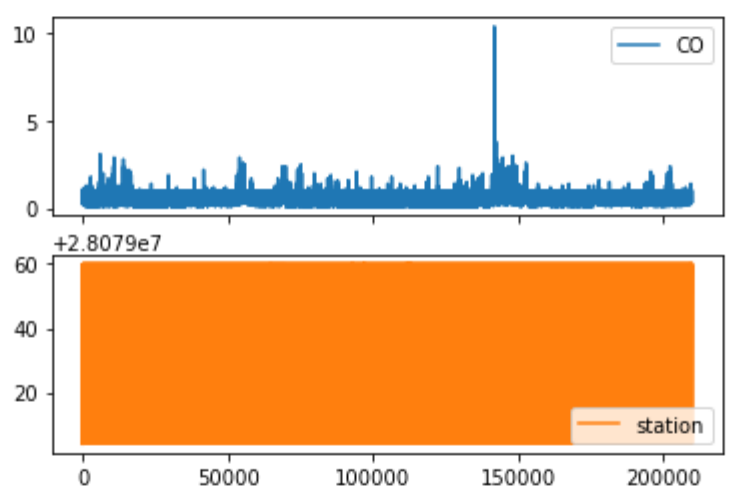
	CO	station
0	0.6	28079004
1	0.5	28079008
2	1.0	28079011
3	0.5	28079016
4	1.0	28079017
...	...	...
209875	0.4	28079056
209876	0.4	28079057
209877	1.0	28079058
209878	1.0	28079059
209879	1.0	28079060

209880 rows × 2 columns

Line chart

In [7]:

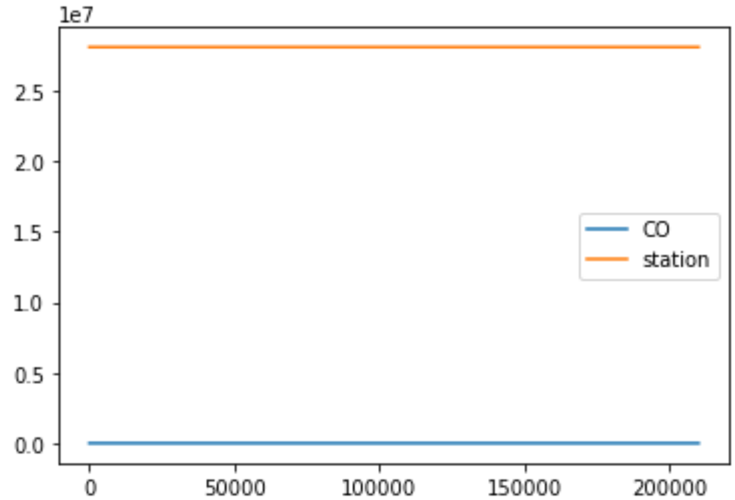
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:

Out[8]: <AxesSubplot:>

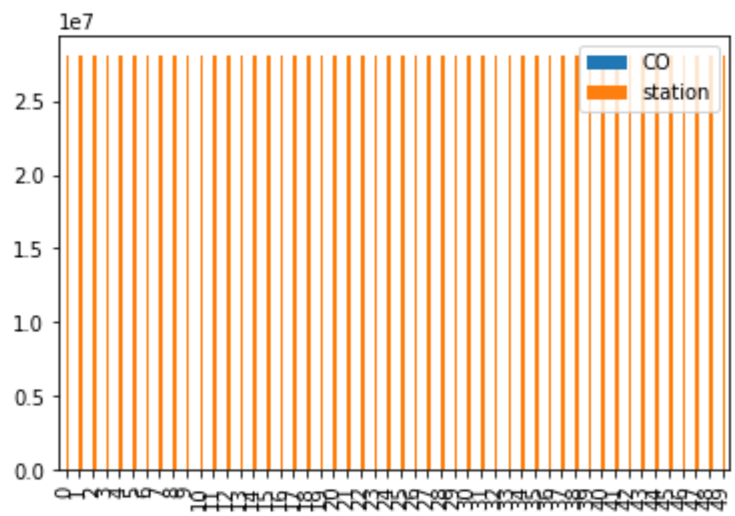


Bar chart

In [9]:

In [10]:

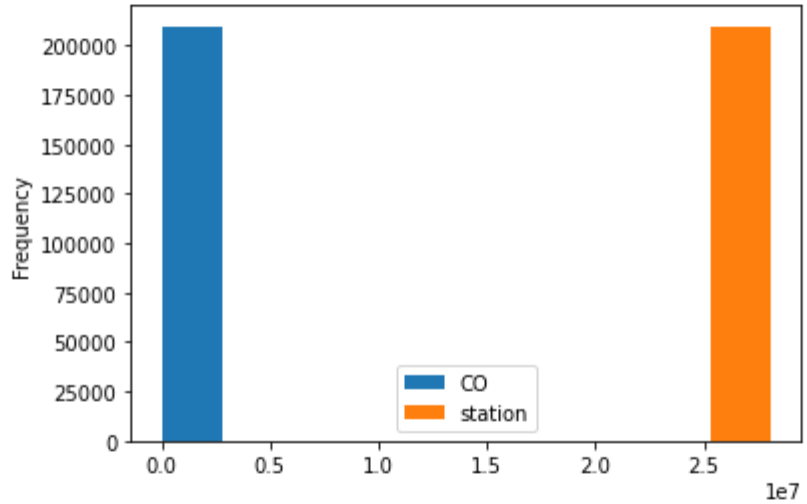
Out[10]: <AxesSubplot:>



Histogram

In [11]:

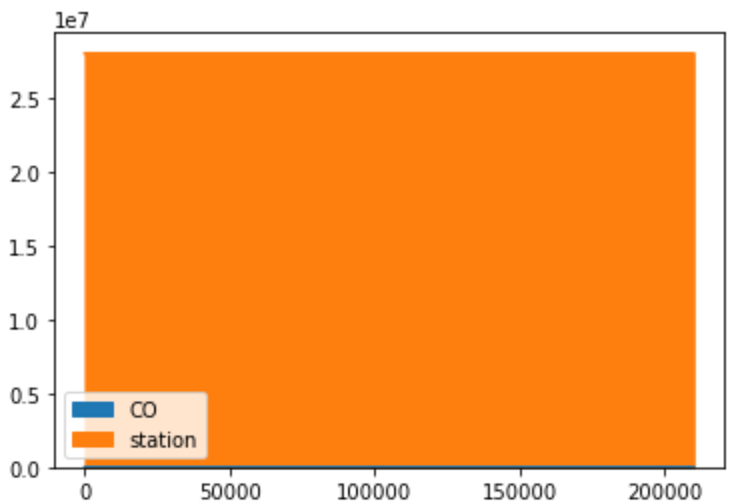
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:

Out[12]: <AxesSubplot:>

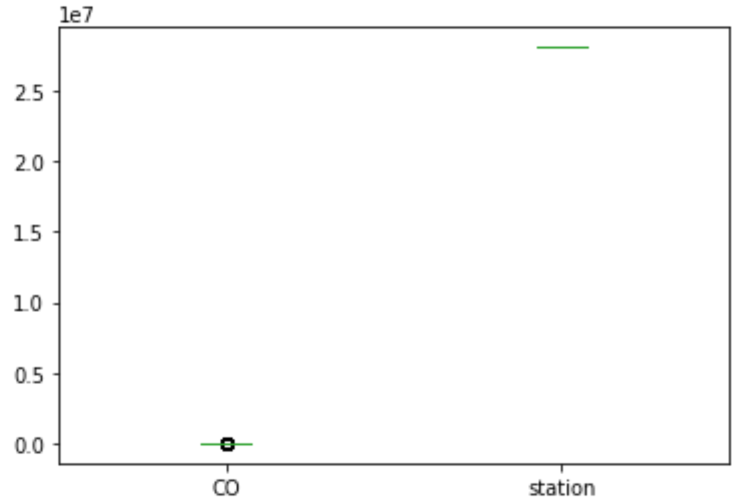


Box chart

In [13]:

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

Out[13]: <AxesSubplot:>



Pie chart

In [14]:

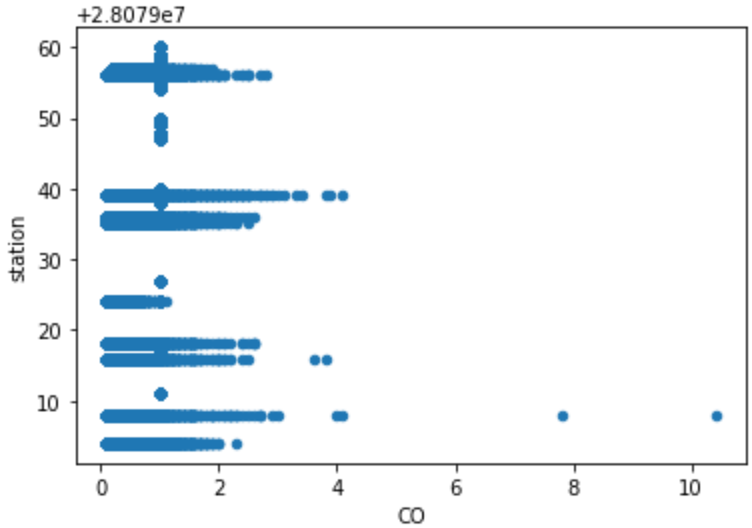
```
[illegible]
```

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

Scatter chart

In [15]:
```
data.plot.scatter(x='CO',y='station')
```

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



In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
```

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

Out[17]:

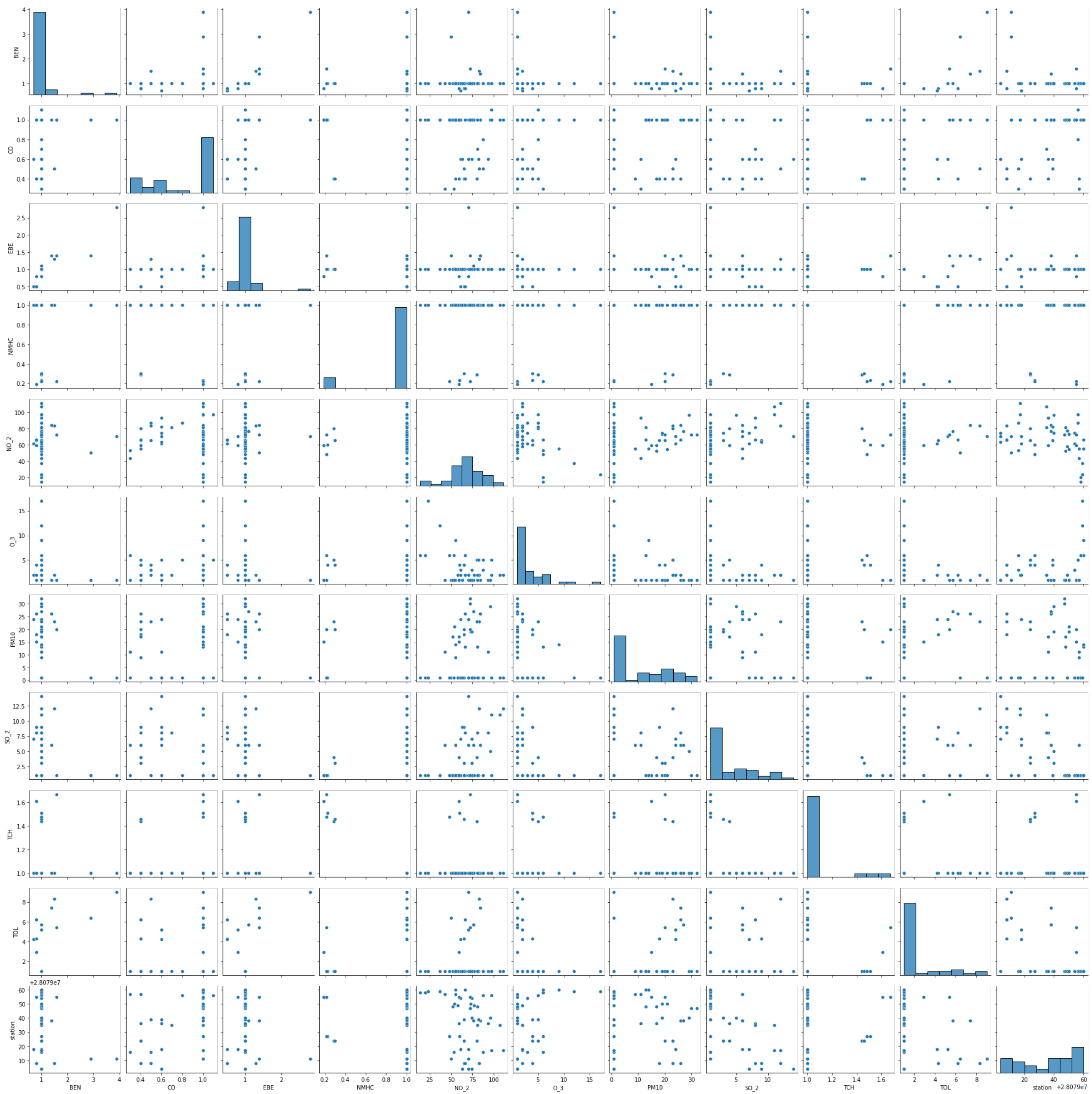
	BEN	CO	EBE	NMHC	NO	NO_
count	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.00000
mean	0.931014	0.721695	0.954744	0.900223	20.101401	34.5864
std	0.430684	0.361528	0.301074	0.267139	44.319112	27.8665
min	0.100000	0.100000	0.100000	0.040000	1.000000	1.00000
25%	1.000000	0.300000	1.000000	1.000000	2.000000	14.00000
50%	1.000000	1.000000	1.000000	1.000000	5.000000	27.00000
75%	1.000000	1.000000	1.000000	1.000000	17.000000	48.00000
max	12.100000	10.400000	11.800000	1.000000	1081.000000	388.00000

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

EDA AND VISUALIZATION

In [19]:

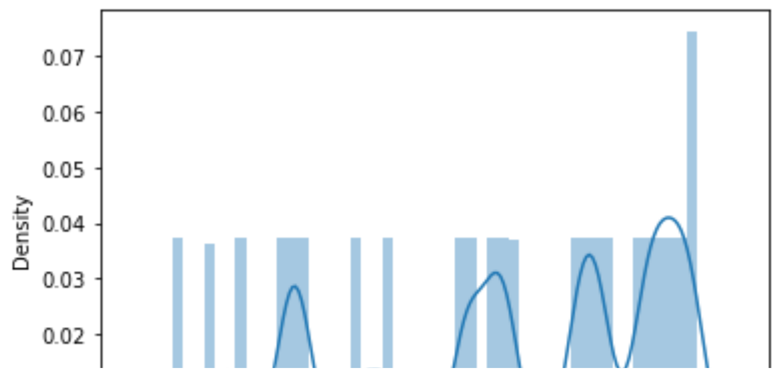
Out[19]: <seaborn.axisgrid.PairGrid at 0x1b7069c4880>



In [20]:
```
sns.distplot(df1['station'])
```

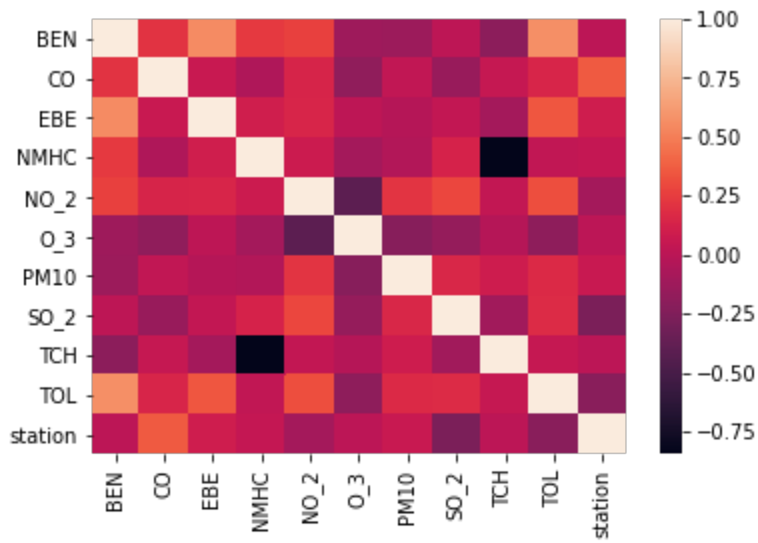
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

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



In [21]:
```
sns.heatmap(df1.corr())
```

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:
```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

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]:
```
lr.intercept_
```

Out[25]: 28078974.774991233

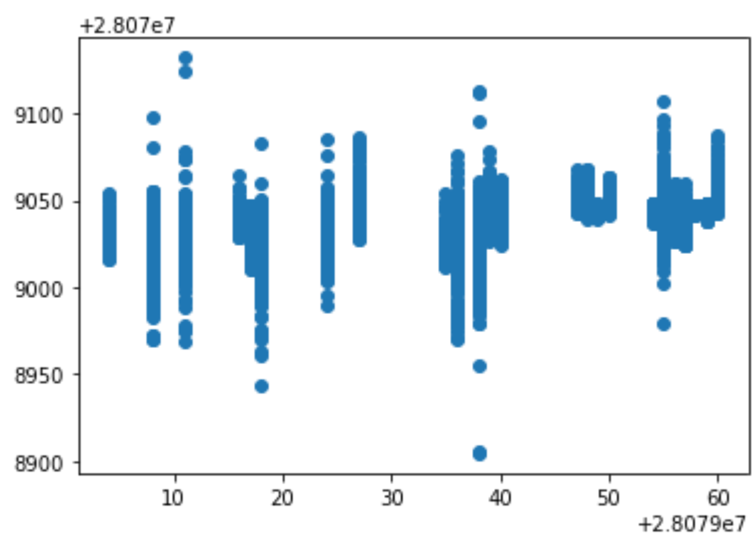
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[26]:

	Co-efficient
BEN	2.279250
CO	18.361725
EBE	9.781230
NMHC	18.723701
NO_2	-0.056757
O_3	0.009418
PM10	0.207129
SO_2	-0.935167
TCH	27.084232
TOL	-3.705202

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

Out[27]: <matplotlib.collections.PathCollection at 0x1b712d01970>



ACCURACY

In [28]:

Out[28]: 0.2969731290074158

In [29]:

Out[29]: 0.3009959817638057

Ridge and Lasso

In [30]:

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

Accuracy(Ridge)

In [32]:

Out[32]: 0.2969761035146333

In [33]:

Out[33]: 0.3009928209761411

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

In [35]:

Out[35]: 0.04573400320033738

Accuracy(Lasso)

In [36]:

Out[36]: 0.044374626416004426

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

In [38]:

Out[38]:
```
array([ 0.41333256,  2.69455508,  0.50163331,  0.        , -0.0207769 ,
       -0.01652729,  0.16267641, -1.27767382, -0.        , -1.68426817])
```

In [39]:

Out[39]:
```
28079040.024867676
```

In [40]:
```
prediction=en.predict(x_test)
```

In [41]:

Out[41]:
```
0.15081708322622445
```

Evaluation Metrics

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

```
13.747897675434553
262.88770639903544
16.2138122105517
```

Logistic Regression

In [43]:

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2',  'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
```

In [45]:

Out[45]:
```
(209880, 10)
```

In [46]:

Out[46]:
```
(209880,)
```

In [47]:

In [48]:

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

In [50]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]
```

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

```
[28079008]
```

In [52]:

Out[52]:
```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060],
      dtype=int64)
```

In [53]:

Out[53]:
```
0.6612921669525443
```

In [54]:

Out[54]:
```
9.49253547859177e-217
```

In [55]:

Out[55]:
```
array([[9.49253548e-217, 6.03969072e-001, 1.69773000e-169,
        1.44179094e-134, 1.71060740e-074, 3.96021369e-001,
        9.55808997e-006, 5.22717178e-089, 5.48319507e-081,
        1.32436170e-079, 1.07294134e-076, 3.50636612e-129,
        1.69529056e-079, 3.82520459e-158, 4.22872970e-161,
        3.57928159e-187, 2.10845766e-164, 8.33937392e-188,
        1.12752042e-082, 7.42692411e-129, 7.66872499e-080,
        6.30044443e-191, 4.32093567e-191, 3.26054498e-071]])
```

Random Forest

In [56]:

In [57]:
```
rfc=RandomForestClassifier()
```

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]:
```
0.6941041139154347
```

In [61]:

In [63]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b'
```

Out[63]:
```
[Text(2243.16, 1993.2, 'O_3 <= 1.5\ngini = 0.958\nsamples = 93062\nvalue = [6
290, 5960, 5974, 5955, 6065, 6155, 6034, 6250, 6087\n6196, 6111, 6085, 6076,
6255, 6177, 6181, 6149, 6150\n6178, 6064, 6116, 6098, 6098, 6212]\nclass = a
'),
 Text(1227.6, 1630.8000000000002, 'SO_2 <= 1.5\ngini = 0.904\nsamples = 3966
9\nvalue = [6290, 46, 5974, 10, 44, 150, 30, 7, 181, 6196\n6111, 36, 6076, 62
55, 6177, 361, 6149, 31, 6178, 48\n6116, 111, 53, 86]\nclass = a'),
 Text(714.24, 1268.4, 'NMHC <= 0.8\ngini = 0.844\nsamples = 22695\nvalue = [2
2, 40, 5974, 10, 18, 13, 22, 7, 2, 3340, 10, 36\n882, 6255, 6177, 361, 6149,
31, 6178, 48, 32, 111\n53, 86]\nclass = n'),
 Text(357.12, 906.0, 'TCH <= 1.335\ngini = 0.004\nsamples = 3870\nvalue = [0,
0, 0, 0, 0, 0, 9, 4, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 6089, 0, 0, 0, 0, 0]\nclas
s = s'),
 Text(178.56, 543.5999999999999, 'EBE <= 0.85\ngini = 0.349\nsamples = 35\nva
lue = [0, 0, 0, 0, 0, 0, 9, 1, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 36, 0, 0, 0, 0,
0]\nclass = s'),
 Text(89.28, 181.19999999999982, 'gini = 0.484\nsamples = 12\nvalue = [0, 0,
0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 10, 0, 0, 0, 0, 0]\nclass = s
'),
```

Conclusion

Accuracy

Linear Regression :0.3009959817638057

Ridge Regression :0.04573400320033738

Lasso Regression :0.044374626416004426

ElasticNet Regression : 0.15081708322622445

Logistic Regression : 0.6612921669525443

Random Forest is suitable for this dataset