Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [2]:
```
df=pd.read_csv("madrid_2001.csv")
df
```

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	105.0
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	100.5
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.0
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.7
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.1
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.8
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.8
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.7
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.8
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	35.3

217872 rows × 16 columns

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.dropna()
```

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

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]:
```
df.info()
```

```
<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
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

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

Out[6]:

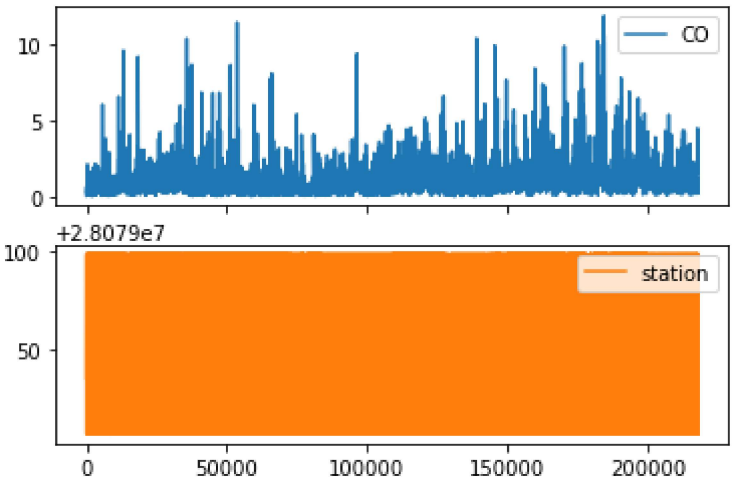
	CO	station
1	0.34	28079035
5	0.63	28079006
21	0.43	28079024
23	0.34	28079099
25	0.06	28079035
...	...	...
217829	4.48	28079006
217847	2.65	28079099
217849	1.22	28079035
217853	1.83	28079006
217871	1.62	28079099

29669 rows × 2 columns

Line chart

In [7]:
```
data.plot.line(subplots=True)
```

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

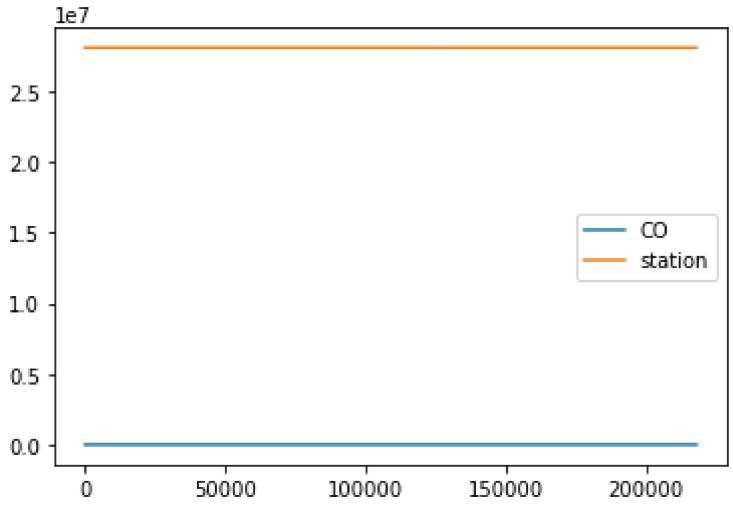


Line chart

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

In [8]:
```
data.plot.line()
```

Out[8]: <AxesSubplot:>

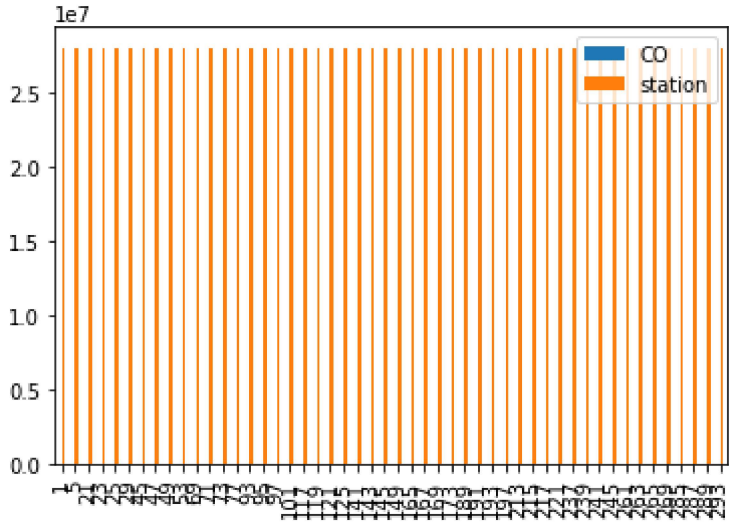


Bar chart

In [9]:
```
b=data[0:50]
```

In [10]:
```
b.plot.bar()
```

Out[10]: <AxesSubplot:>

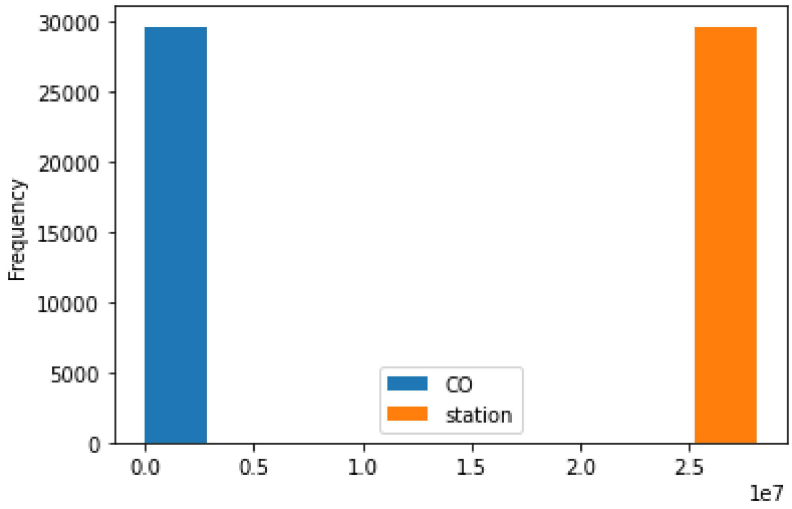


Histogram

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

In [11]:
```
data.plot.hist()
```

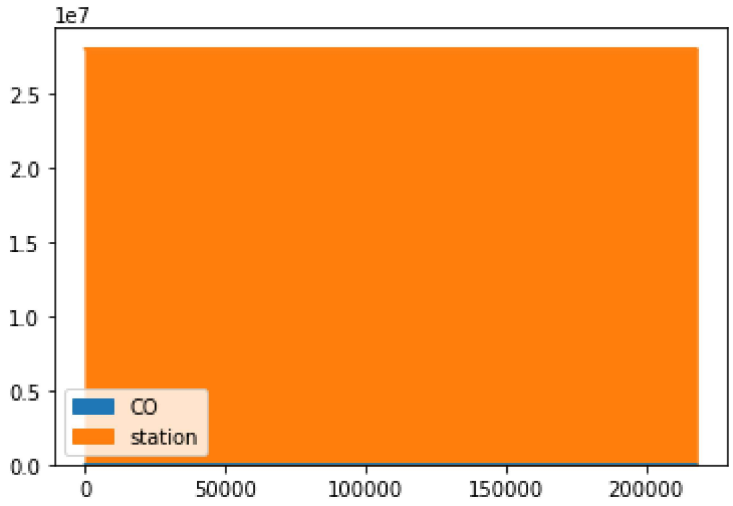
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:
```
data.plot.area()
```

Out[12]: <AxesSubplot:>

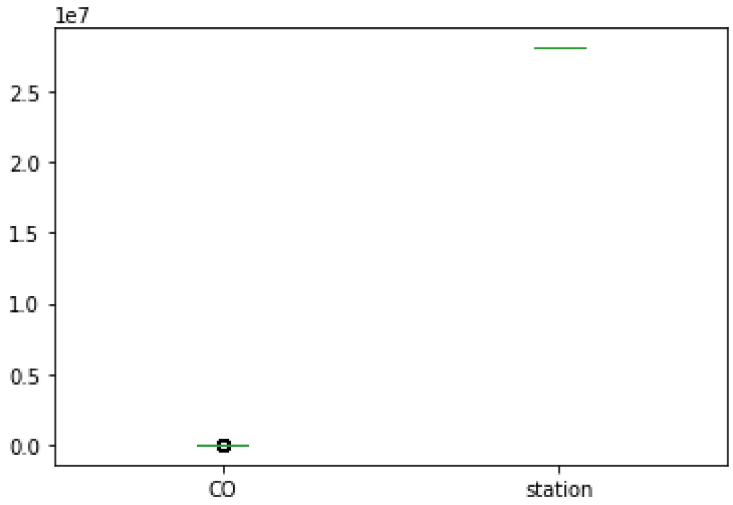


Box chart

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

In [13]:
```
data.plot.box()
```

Out[13]: <AxesSubplot:>



Pie chart

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

In [14]:
```
b.plot.pie(y='station' )
```

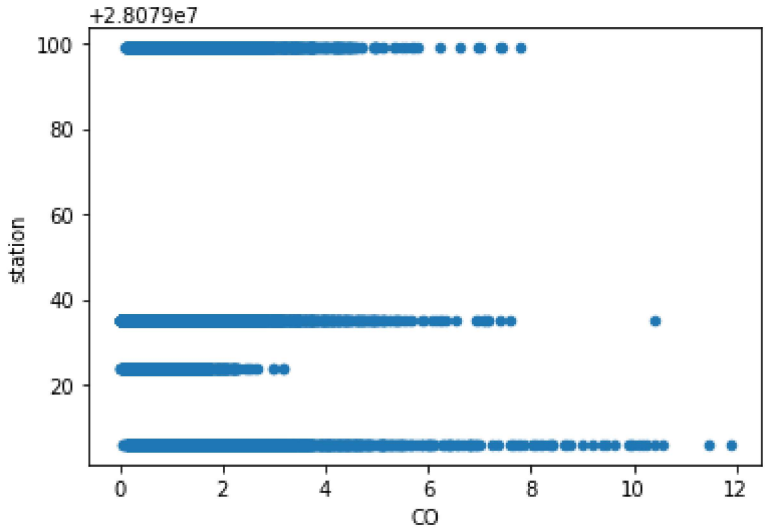
Out[14]: <AxesSubplot:ylabel='station'>

Scatter chart

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

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'>
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
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

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

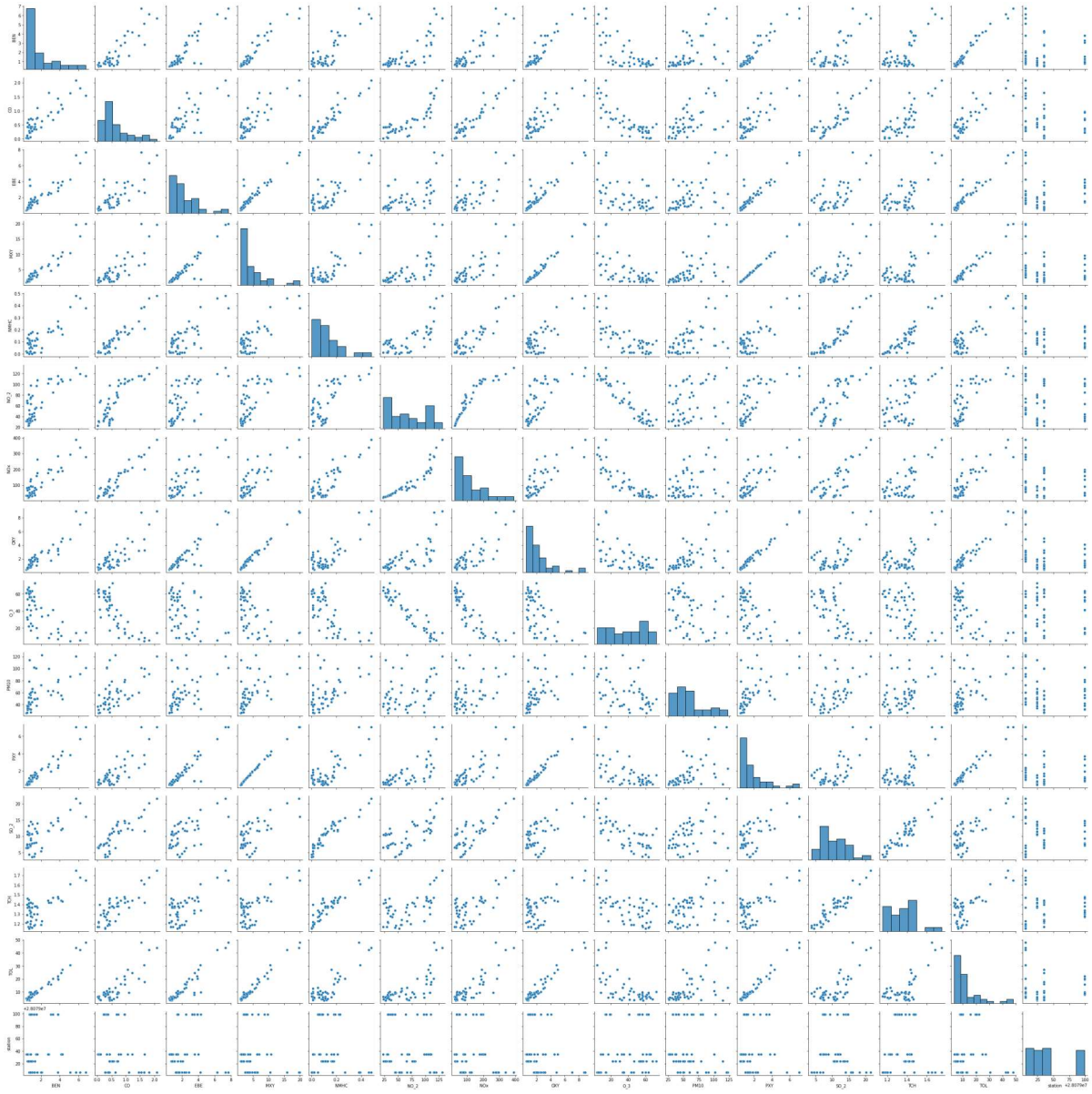
In [18]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x22ef2b014c0>
```

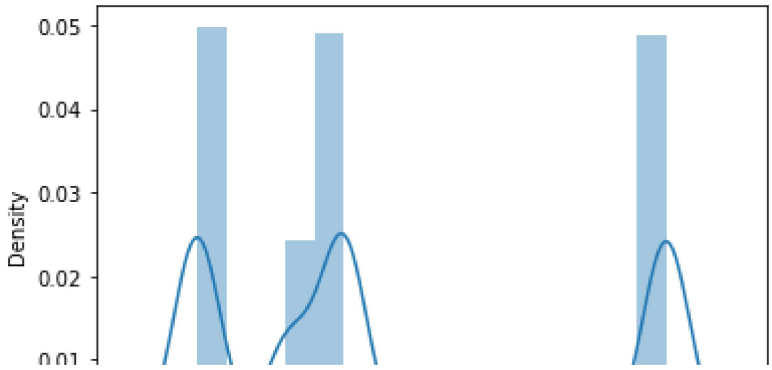


Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

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

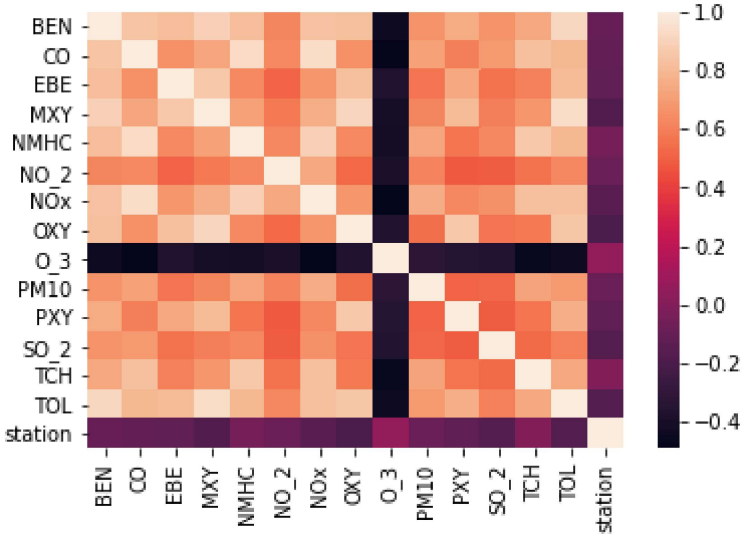
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n 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', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

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()
lr.fit(x_train,y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
lr.intercept_
```

Out[25]:
```
28079007.79248559
```

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

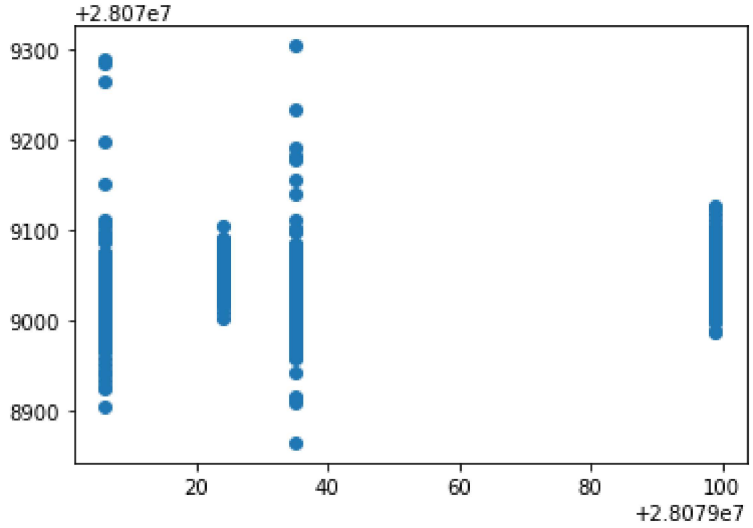
Out[26]:

	Co-efficient
BEN	7.903870
CO	-19.901441
EBE	0.747462
MXY	-0.097258
NMHC	92.293345
NO_2	0.102740
NOx	-0.073859
OXY	-3.440846
O_3	-0.035676
PM10	-0.080597
PXY	1.622929
SO_2	-0.304476
TCH	37.580609
TOL	-1.294369

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

In [27]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x22e838ab580>
```



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```

Out[28]:
```
0.13857439171729402
```

In [29]:
```
lr.score(x_train,y_train)
```

Out[29]:
```
0.17472674775670027
```

Ridge and Lasso

In [30]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [31]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

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

Accuracy(Ridge)

In [32]:
```
rr.score(x_test,y_test)
```

Out[32]:
```
0.14075080196859435
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

In [33]:
```
rr.score(x_train,y_train)
```

Out[33]:
```
0.17438229870216881
```

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

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

In [35]:
```
la.score(x_train,y_train)
```

Out[35]:
```
0.03984017509494553
```

Accuracy(Lasso)

In [36]:
```
la.score(x_test,y_test)
```

Out[36]:
```
0.037381392948549474
```

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

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

In [38]:
```
en.coef_
```

Out[38]:
```
array([ 5.11138649,  0.        ,  0.72430525, -0.18724666,  0.06098759,
        0.06123496, -0.03355092, -2.66878993, -0.03951734,  0.06784229,
        0.97496443, -0.32864235,  1.20938018, -0.72431939])
```

In [39]:
```
en.intercept_
```

Out[39]:
```
28079049.210427705
```

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

In [41]:
```
en.score(x_test,y_test)
```

Out[41]:
```
0.0971378267723827
```

Evaluation Metrics

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

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

```
30.537897257635084
1233.7255321876141
35.12442927917284
```

Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

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

In [45]:
```
feature_matrix.shape
```

Out[45]:
```
(29669, 14)
```

In [46]:
```
target_vector.shape
```

Out[46]:
```
(29669,)
```

In [47]:
```
from sklearn.preprocessing import StandardScaler
```

In [48]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [49]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[49]:
```
LogisticRegression(max_iter=10000)
```

In [50]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

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

```
[28079035]
```

In [52]:
```
logr.classes_
```

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

In [53]:
```
logr.score(fs,target_vector)
```

Out[53]:
```
0.8087229094340894
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

In [54]:
```
logr.predict_proba(observation)[0][0]
```

Out[54]:
```
1.724527777144498e-43
```

In [55]:
```
logr.predict_proba(observation)
```

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

Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestClassifier
```

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
grid_search.fit(x_train,y_train)
```

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]:
```
grid_search.best_score_
```

Out[60]:
```
0.7284283513097072
```

In [61]:
```
rfc_best=grid_search.best_estimator_
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

In [62]:
```
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'
```

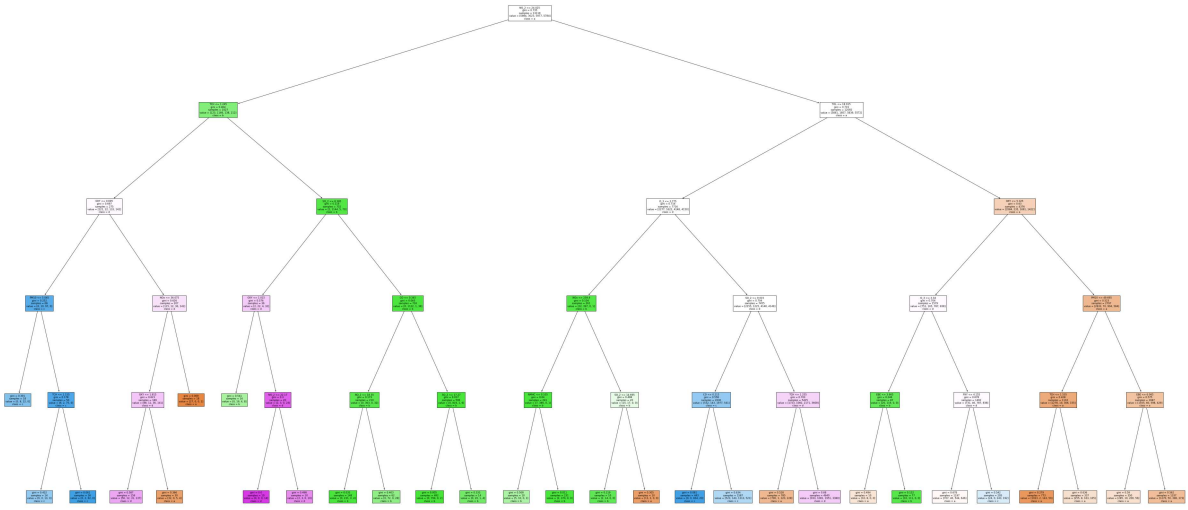
Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

Out[62]: [Text(1998.6545454545453, 1993.2, 'NO_2 <= 24.025\ngini = 0.735\nsamples = 13119\nvalue = [5984, 3023, 5977, 5784]\nclass = a'),
 Text(831.9272727272727, 1630.8000000000002, 'TCH <= 1.245\ngini = 0.464\nsamples = 1027\nvalue = [123, 1166, 138, 212]\nclass = b'),
 Text(405.81818181818176, 1268.4, 'OXY <= 0.695\ngini = 0.697\nsamples = 275\nvalue = [121, 22, 133, 142]\nclass = d'),
 Text(162.3272727272727, 906.0, 'PM10 <= 5.045\ngini = 0.252\nsamples = 68\nvalue = [6, 10, 97, 0]\nclass = c'),
 Text(81.16363636363636, 543.5999999999999, 'gini = 0.391\nsamples = 18\nvalue = [0, 8, 22, 0]\nclass = c'),
 Text(243.49090909090907, 543.5999999999999, 'TCH <= 1.155\ngini = 0.178\nsamples = 50\nvalue = [6, 2, 75, 0]\nclass = c'),
 Text(162.3272727272727, 181.19999999999982, 'gini = 0.432\nsamples = 14\nvalue = [6, 0, 13, 0]\nclass = c'),
 Text(324.6545454545454, 181.19999999999982, 'gini = 0.061\nsamples = 36\nvalue = [0, 2, 62, 0]\nclass = c'),
 Text(649.3090909090909, 906.0, 'NOx <= 36.075\ngini = 0.626\nsamples = 207\nvalue = [115, 12, 36, 142]\nclass = d'),
 Text(568.1454545454545, 543.5999999999999, 'OXY <= 1.815\ngini = 0.621\nsamples = 189\nvalue = [88, 12, 36, 141]\nclass = d'),
 Text(486.98181818181814, 181.19999999999982, 'gini = 0.587\nsamples = 159\nvalue = [56, 12, 31, 137]\nclass = d'),
 Text(649.3090909090909, 181.19999999999982, 'gini = 0.366\nsamples = 30\nvalue = [32, 0, 5, 4]\nclass = a'),
 Text(730.4727272727272, 543.5999999999999, 'gini = 0.069\nsamples = 18\nvalue = [27, 0, 0, 1]\nclass = a'),
 Text(1258.0363636363636, 1268.4, 'SO_2 <= 8.385\ngini = 0.119\nsamples = 752\nvalue = [2, 1144, 5, 70]\nclass = b'),
 Text(973.9636363636363, 906.0, 'OXY <= 1.015\ngini = 0.576\nsamples = 36\nvalue = [2, 22, 4, 32]\nclass = d'),
 Text(892.8, 543.5999999999999, 'gini = 0.541\nsamples = 16\nvalue = [0, 19, 4, 8]\nclass = b'),
 Text(1055.1272727272726, 543.5999999999999, 'NO_2 <= 22.17\ngini = 0.3\nsamples = 20\nvalue = [2, 3, 0, 24]\nclass = d'),
 Text(973.9636363636363, 181.19999999999982, 'gini = 0.0\nsamples = 10\nvalue = [0, 0, 0, 14]\nclass = d'),
 Text(1136.290909090909, 181.19999999999982, 'gini = 0.498\nsamples = 10\nvalue = [2, 3, 0, 10]\nclass = d'),
 Text(1542.1090909090908, 906.0, 'CO <= 0.345\ngini = 0.065\nsamples = 716\nvalue = [0, 1122, 1, 38]\nclass = b'),
 Text(1379.7818181818182, 543.5999999999999, 'NO_2 <= 18.35\ngini = 0.173\nsamples = 210\nvalue = [0, 303, 0, 32]\nclass = b'),
 Text(1298.6181818181817, 181.19999999999982, 'gini = 0.033\nsamples = 148\nvalue = [0, 231, 0, 4]\nclass = b'),
 Text(1460.9454545454544, 181.19999999999982, 'gini = 0.403\nsamples = 62\nvalue = [0, 72, 0, 28]\nclass = b'),
 Text(1704.4363636363635, 543.5999999999999, 'SO_2 <= 17.27\ngini = 0.017\nsamples = 506\nvalue = [0, 819, 1, 6]\nclass = b'),
 Text(1623.272727272727, 181.19999999999982, 'gini = 0.005\nsamples = 492\nvalue = [0, 799, 0, 2]\nclass = b'),
 Text(1785.6, 181.19999999999982, 'gini = 0.333\nsamples = 14\nvalue = [0, 20, 1, 4]\nclass = b'),
 Text(3165.381818181818, 1630.8000000000002, 'TOL <= 18.915\ngini = 0.719\nsamples = 12092\nvalue = [5861, 1857, 5839, 5572]\nclass = a'),
 Text(2516.072727272727, 1268.4, 'O_3 <= 3.775\ngini = 0.716\nsamples = 7716\nvalue = [2277, 1622, 4148, 4150]\nclass = d'),
 Text(2191.4181818181814, 906.0, 'NOx <= 259.4\ngini = 0.104\nsamples = 261\n

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js

```
value = [22, 397, 0, 1]\nclass = b'),
 Text(2029.090909090909, 543.5999999999999, 'NMHC <= 0.105\ngini = 0.04\nsamp
les = 241\nvalue = [7, 380, 0, 1]\nclass = b'),
 Text(1947.9272727272726, 181.19999999999982, 'gini = 0.508\nsamples = 10\nva
lue = [5, 10, 0, 1]\nclass = b'),
 Text(2110.254545454545, 181.19999999999982, 'gini = 0.011\nsamples = 231\nva
lue = [2, 370, 0, 0]\nclass = b'),
 Text(2353.7454545454543, 543.5999999999999, 'SO_2 <= 14.695\ngini = 0.498\ns
amples = 20\nvalue = [15, 17, 0, 0]\nclass = b'),
 Text(2272.581818181818, 181.19999999999982, 'gini = 0.219\nsamples = 10\nval
ue = [2, 14, 0, 0]\nclass = b'),
 Text(2434.9090909090905, 181.19999999999982, 'gini = 0.305\nsamples = 10\nva
lue = [13, 3, 0, 0]\nclass = a'),
 Text(2840.7272727272725, 906.0, 'SO_2 <= 9.015\ngini = 0.704\nsamples = 7455
\nvalue = [2255, 1225, 4148, 4149]\nclass = d'),
 Text(2678.3999999999996, 543.5999999999999, 'CO <= 0.215\ngini = 0.558\nsamp
les = 2030\nvalue = [532, 143, 1977, 541]\nclass = c'),
 Text(2597.2363636363634, 181.19999999999982, 'gini = 0.081\nsamples = 443\nv
alue = [9, 0, 664, 20]\nclass = c'),
 Text(2759.5636363636363, 181.19999999999982, 'gini = 0.634\nsamples = 1587\n
value = [523, 143, 1313, 521]\nclass = c'),
 Text(3003.0545454545454, 543.5999999999999, 'TCH <= 1.255\ngini = 0.703\nsam
ples = 5425\nvalue = [1723, 1082, 2171, 3608]\nclass = d'),
 Text(2921.8909090909087, 181.19999999999982, 'gini = 0.528\nsamples = 785\nv
alue = [790, 1, 220, 228]\nclass = a'),
 Text(3084.2181818181816, 181.19999999999982, 'gini = 0.68\nsamples = 4640\nv
alue = [933, 1081, 1951, 3380]\nclass = d'),
 Text(3814.690909090909, 1268.4, 'OXY <= 5.325\ngini = 0.63\nsamples = 4376\n
value = [3584, 235, 1691, 1422]\nclass = a'),
 Text(3490.036363636363, 906.0, 'O_3 <= 3.34\ngini = 0.704\nsamples = 1579\nv
alue = [751, 165, 787, 838]\nclass = d'),
 Text(3327.7090909090907, 543.5999999999999, 'EBE <= 3.665\ngini = 0.246\nsam
ples = 87\nvalue = [20, 119, 0, 0]\nclass = b'),
 Text(3246.545454545454, 181.19999999999982, 'gini = 0.494\nsamples = 10\nval
ue = [10, 8, 0, 0]\nclass = a'),
 Text(3408.872727272727, 181.19999999999982, 'gini = 0.152\nsamples = 77\nval
ue = [10, 111, 0, 0]\nclass = b'),
 Text(3652.363636363636, 543.5999999999999, 'PXY <= 4.155\ngini = 0.678\nsamp
les = 1492\nvalue = [731, 46, 787, 838]\nclass = d'),
 Text(3571.2, 181.19999999999982, 'gini = 0.678\nsamples = 1197\nvalue = [70
7, 46, 544, 646]\nclass = a'),
 Text(3733.5272727272722, 181.19999999999982, 'gini = 0.542\nsamples = 295\nv
alue = [24, 0, 243, 192]\nclass = c'),
 Text(4139.345454545454, 906.0, 'PM10 <= 49.695\ngini = 0.523\nsamples = 2797
\nvalue = [2833, 70, 904, 584]\nclass = a'),
 Text(3977.0181818181813, 543.5999999999999, 'TCH <= 1.535\ngini = 0.428\nsam
ples = 1110\nvalue = [1278, 10, 306, 155]\nclass = a'),
 Text(3895.854545454545, 181.19999999999982, 'gini = 0.279\nsamples = 773\nva
lue = [1023, 2, 143, 50]\nclass = a'),
 Text(4058.181818181818, 181.19999999999982, 'gini = 0.636\nsamples = 337\nva
lue = [255, 8, 163, 105]\nclass = a'),
 Text(4301.672727272727, 543.5999999999999, 'EBE <= 6.065\ngini = 0.575\nsamp
les = 1687\nvalue = [1555, 60, 598, 429]\nclass = a'),
 Text(4220.50909090909, 181.19999999999982, 'gini = 0.59\nsamples = 350\nvalu
e = [285, 10, 200, 56]\nclass = a'),
 Text(4382.836363636363, 181.19999999999982, 'gini = 0.562\nsamples = 1337\nv
alue = [1270, 50, 398, 373]\nclass = a')]
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js



Conclusion

Accuracy

Linear Regression:0.17472674775670027

Ridge Regression:0.17438229870216881

Lasso Regression:0.03984017509494553

ElasticNet Regression:0.0971378267723827

Logistic Regression:0.8087229094340894

Random Forest:0.7284283513097072

Logistic Regression is suitable for this dataset

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/Main/Italic/Main.js