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_2014.csv")
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3
2	2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1
3	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN
4	2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
210019	2014-09-01 00:00:00	NaN	0.5	NaN	NaN	20.0	84.0	29.0	NaN	NaN	NaN	NaN	NaN
210020	2014-09-01 00:00:00	NaN	0.3	NaN	NaN	1.0	22.0	NaN	15.0	NaN	6.0	NaN	NaN
210021	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	13.0	70.0	NaN	NaN	NaN	NaN	NaN
210022	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	38.0	42.0	NaN	NaN	NaN	NaN	NaN
210023	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	26.0	65.0	11.0	NaN	NaN	NaN	NaN

210024 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:

```
[illegible]
```

In [4]:

```
[illegible]
```

Out[4]:

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

In [5]:

```
[illegible]
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13946 entries, 1 to 210006
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     13946 non-null  object 
 1   BEN      13946 non-null  float64
 2   CO       13946 non-null  float64
 3   EBE      13946 non-null  float64
 4   NMHC     13946 non-null  float64
 5   NO       13946 non-null  float64
 6   NO_2     13946 non-null  float64
 7   O_3      13946 non-null  float64
 8   PM10     13946 non-null  float64
 9   PM25     13946 non-null  float64
 10  SO_2     13946 non-null  float64
 11  TCH      13946 non-null  float64
 12  TOL      13946 non-null  float64
 13  station  13946 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.6+ MB
```

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

Out[6]:

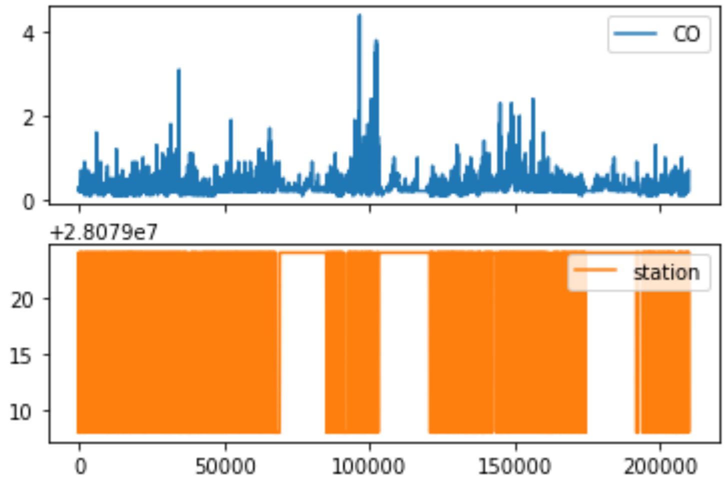
	CO	station
1	0.2	28079008
6	0.2	28079024
25	0.2	28079008
30	0.2	28079024
49	0.2	28079008
...	...	...
209958	0.2	28079024
209977	0.7	28079008
209982	0.2	28079024
210001	0.4	28079008
210006	0.2	28079024

13946 rows × 2 columns

Line chart

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

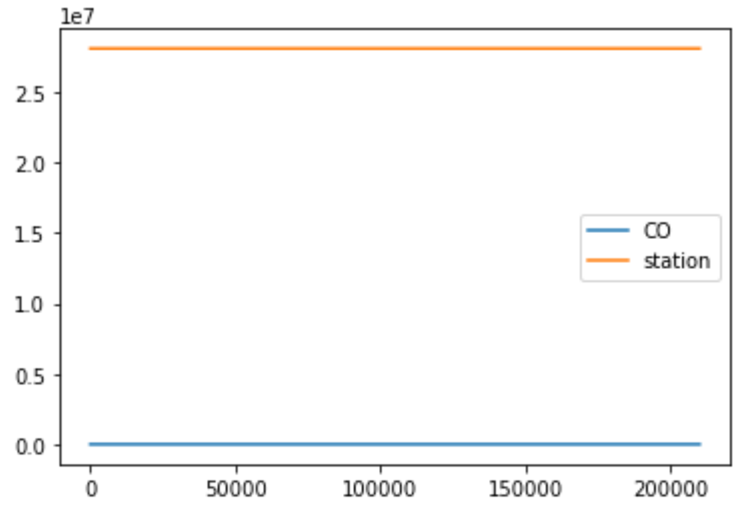
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:
```
[illegible]
```

Out[8]:
```
<AxesSubplot:>
```

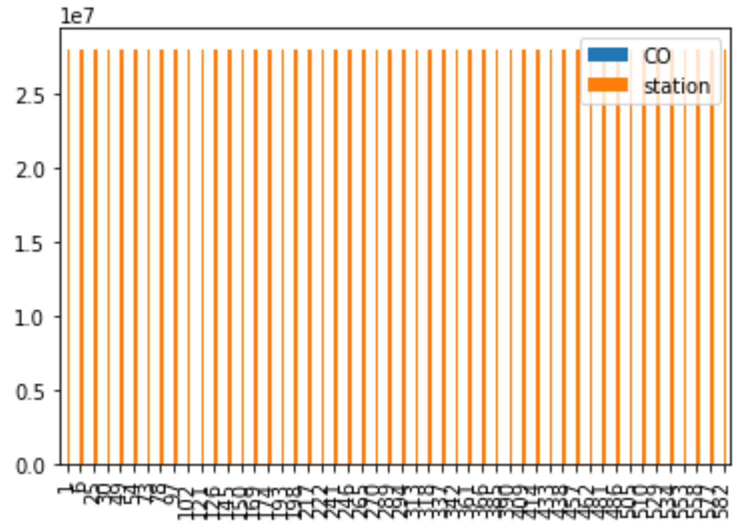


Bar chart

In [9]:
```
[illegible]
```

In [10]:
```
[illegible]
```

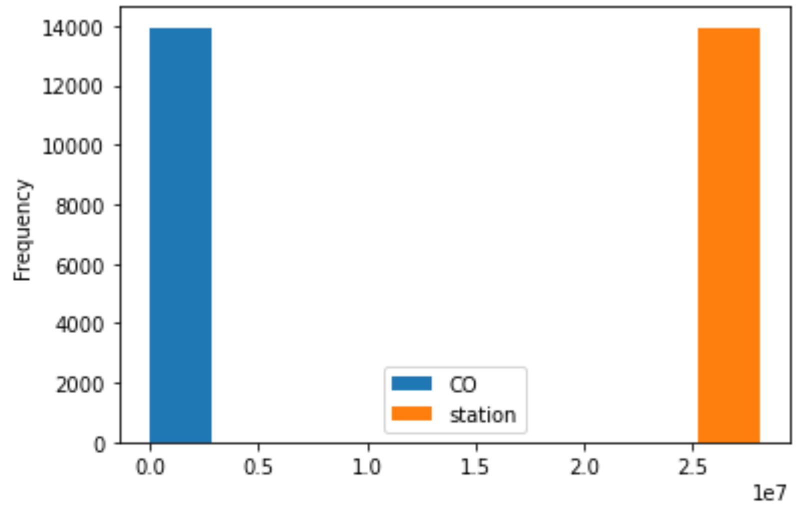
Out[10]:
```
<AxesSubplot:>
```



Histogram

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

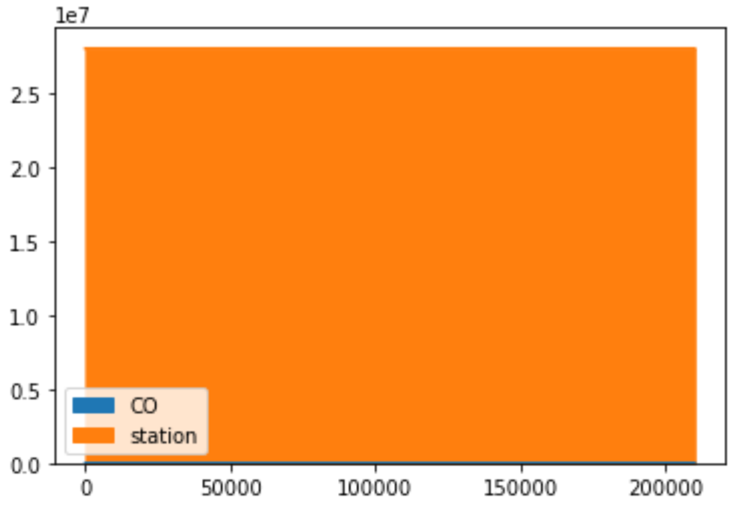
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

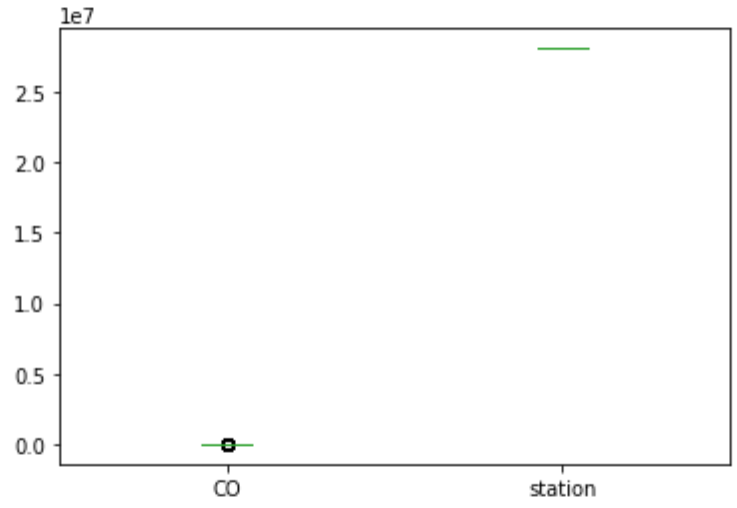
```
Out[12]: <AxesSubplot:>
```



Box chart

In [13]:
```
[illegible]
```

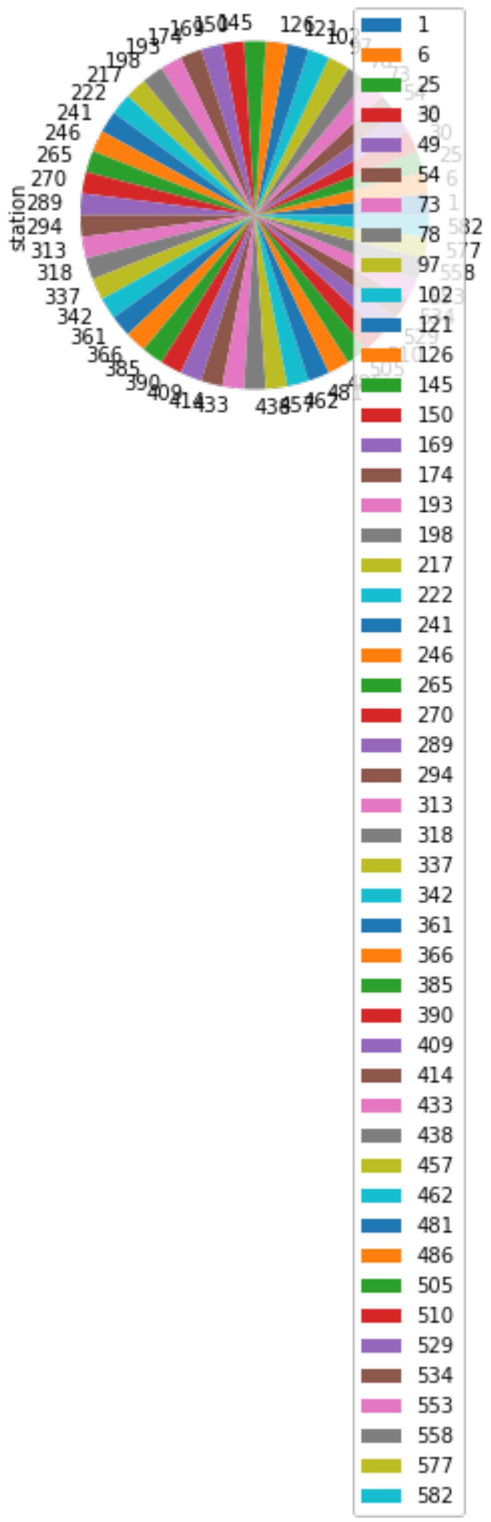
Out[13]:
```
<AxesSubplot:>
```



Pie chart

In [14]: [illegible]

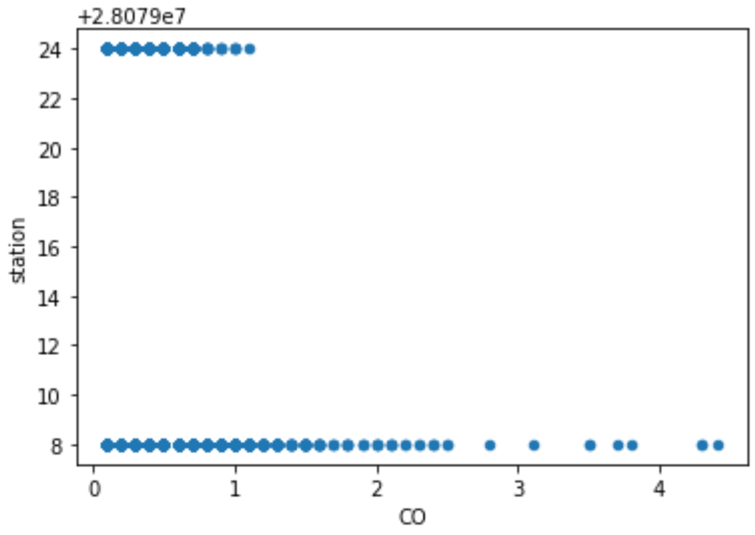
```
Out[14]: <AxesSubplot:ylabel='station'>
```



Scatter chart

In [15]: [illegible]

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



In [16]: [illegible]

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13946 entries, 1 to 210006
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     13946 non-null  object
 1   BEN      13946 non-null  float64
 2   CO       13946 non-null  float64
 3   EBE      13946 non-null  float64
 4   NMHC     13946 non-null  float64
 5   NO       13946 non-null  float64
 6   NO_2     13946 non-null  float64
 7   O_3      13946 non-null  float64
 8   PM10     13946 non-null  float64
 9   PM25     13946 non-null  float64
 10  SO_2     13946 non-null  float64
 11  TCH      13946 non-null  float64
 12  TOL      13946 non-null  float64
 13  station  13946 non-null  int64
[illegible]
```

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

Out[17]:

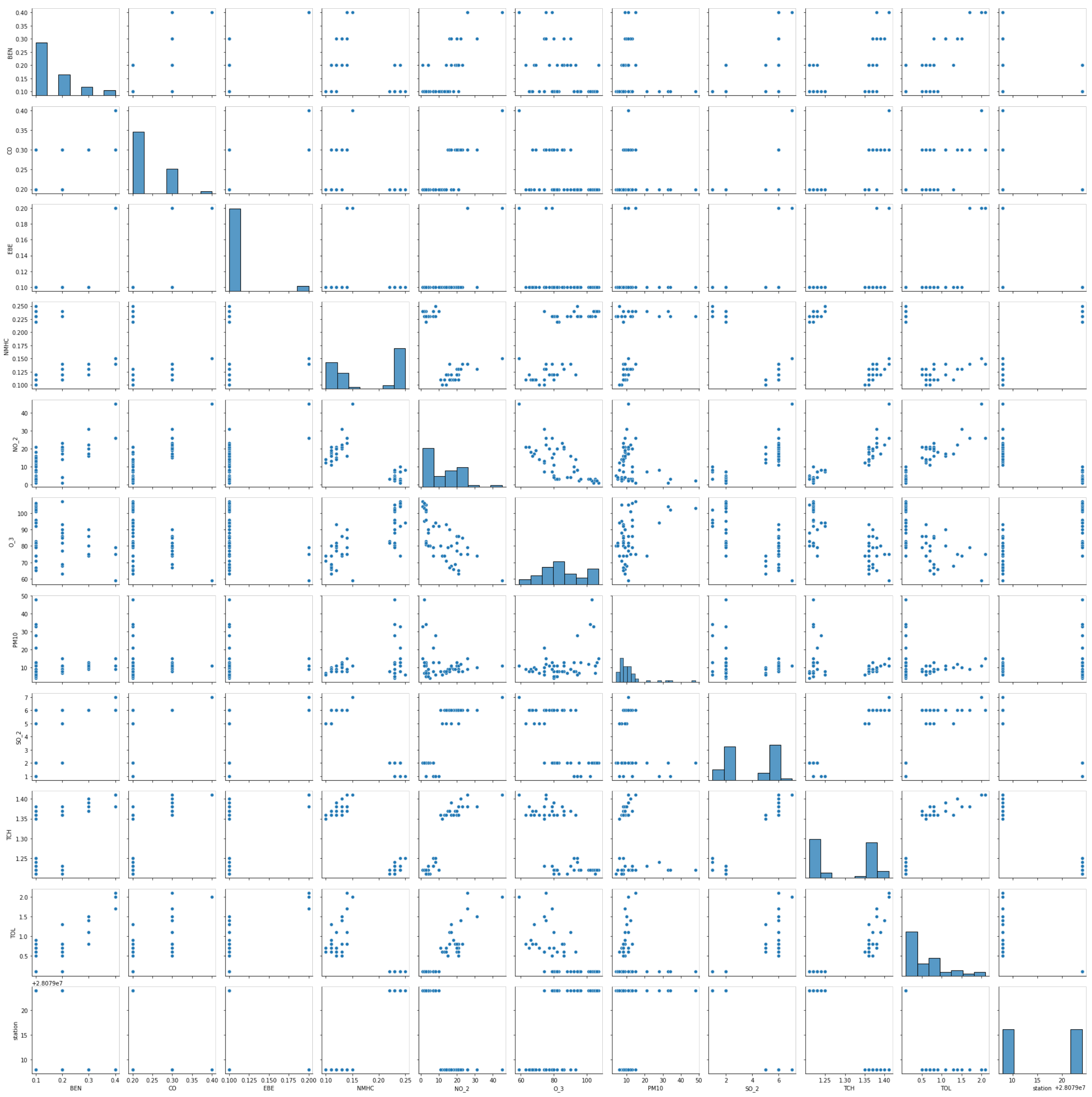
	BEN	CO	EBE	NMHC	NO	NO_2	
count	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000	139
mean	0.375921	0.314793	0.306016	0.222302	17.589129	34.240929	
std	0.555093	0.207375	0.635475	0.082403	39.432216	30.654229	
min	0.100000	0.100000	0.100000	0.060000	1.000000	1.000000	
25%	0.100000	0.200000	0.100000	0.160000	1.000000	10.000000	
50%	0.200000	0.300000	0.100000	0.230000	4.000000	27.000000	
75%	0.400000	0.400000	0.300000	0.260000	18.000000	51.000000	
max	9.400000	4.400000	16.200001	1.290000	725.000000	346.000000	2

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

EDA AND VISUALIZATION

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

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x2bee814c1f0>
```

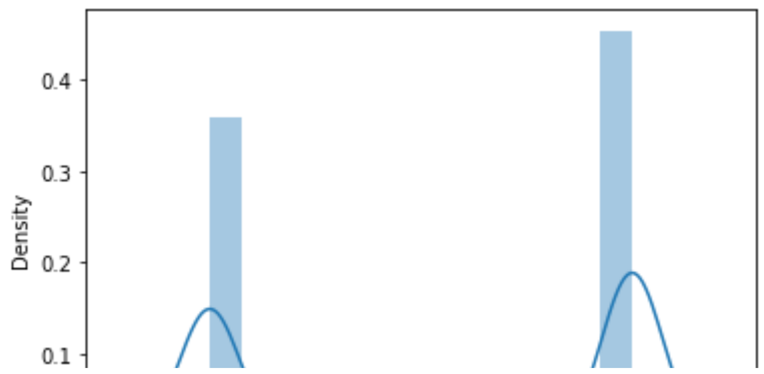


In [20]:

```
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]:

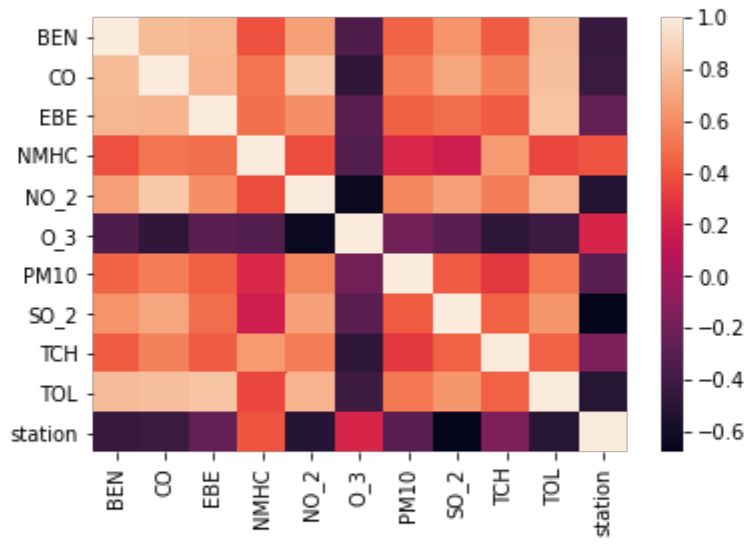
```
<AxesSubplot:xlabel='station', ylabel='Density'>
```



In [21]:

Out[21]:

```
<AxesSubplot:>
```



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:

```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', '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]:
```
lr.intercept_
```

Out[25]:
```
28079022.114812426
```

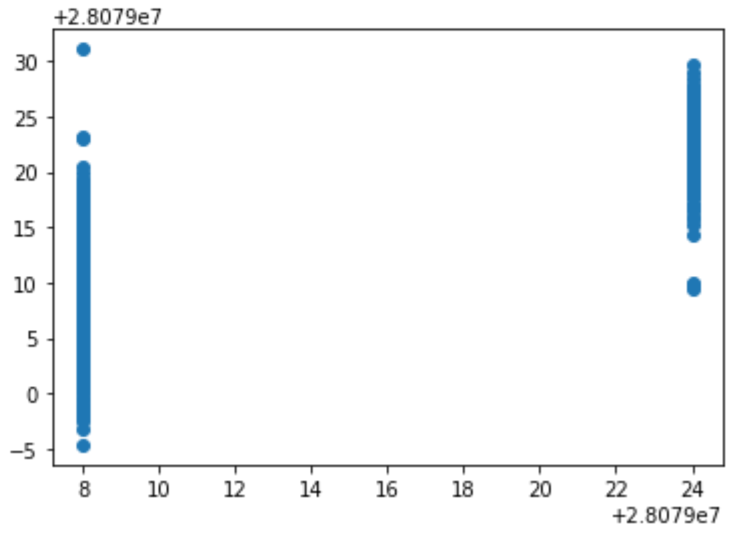
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[26]:

	Co-efficient
BEN	-1.774907
CO	-5.818234
EBE	0.641452
NMHC	83.360064
NO_2	-0.032044
O_3	0.002743
PM10	0.015477
SO_2	-0.851834
TCH	-11.580837
TOL	-0.429355

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x2bef1710460>
```



ACCURACY

In [28]:
```
[illegible]
```
Out[28]:
```
0.8890761358656307
```

In [29]:
```
[illegible]
```
Out[29]:
```
0.8818792538063775
```

Ridge and Lasso

In [30]:
```
[illegible]
```

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

Accuracy(Ridge)

In [32]:
```
[illegible]
```
Out[32]:
```
0.8673565597079445
```

In [33]:
```
[illegible]
```
Out[33]:
```
0.8589350374750626
```

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

In [35]:
```
[illegible]
```
Out[35]:
```
0.27689441343584587
```

Accuracy(Lasso)

In [36]:
```
[illegible]
```
Out[36]:
```
0.2675964292464892
```

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

In [38]:
```
[illegible]
```
Out[38]:
```
array([ 0.        ,  0.        ,  0.16588122,  0.        , -0.04765468,
       -0.01300503,  0.02201435, -1.21102474,  0.        , -0.19119636])
```

In [39]:
```
[illegible]
```
Out[39]:
```
28079024.778188206
```

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

In [41]:
```
[illegible]
```
Out[41]:
```
0.47692368939175334
```

Evaluation Metrics

In [42]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
[illegible]
```
```
5.022313464341247
33.01397887049697
5.745779222220166
```

Logistic Regression

In [43]:
```
[illegible]
```

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

In [45]:
```
[illegible]
```
Out[45]:
```
(13946, 10)
```

In [46]:
```
[illegible]
```
Out[46]:
```
(13946,)
```

In [47]:
```
[illegible]
```

In [48]:
```
[illegible]
```

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

In [50]:

In [51]:

```
prediction=logr.predict(observation)
```

```
[28079008]
```

In [52]:

Out[52]:

```
array([28079008, 28079024], dtype=int64)
```

In [53]:

Out[53]:

```
0.9926143697117453
```

In [54]:

Out[54]:

```
1.0
```

In [55]:

Out[55]:

```
array([[1.00000000e+00, 5.27113072e-18]])
```

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]:

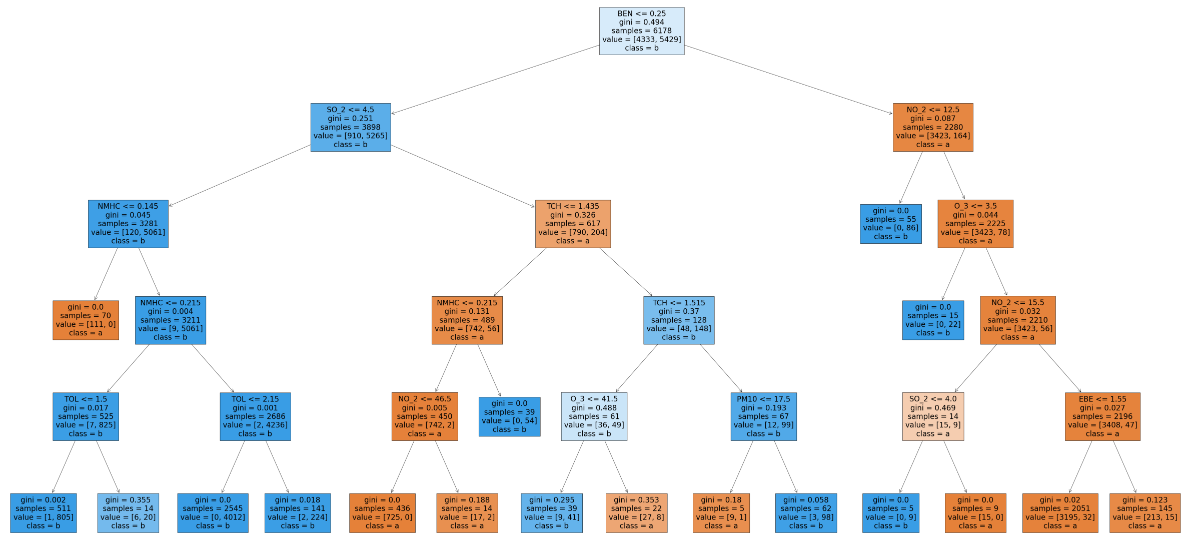
```
0.9960049170251998
```

In [61]:

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

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

Out[62]:
```
[Text(2411.3571428571427, 1993.2, 'BEN <= 0.25\ngini = 0.494\nsamples = 6178\
nvalue = [4333, 5429]\nclass = b'),
 Text(1315.2857142857142, 1630.8000000000002, 'SO_2 <= 4.5\ngini = 0.251\nsam
ples = 3898\nvalue = [910, 5265]\nclass = b'),
 Text(478.2857142857142, 1268.4, 'NMHC <= 0.145\ngini = 0.045\nsamples = 328
1\nvalue = [120, 5061]\nclass = b'),
 Text(318.85714285714283, 906.0, 'gini = 0.0\nsamples = 70\nvalue = [111, 0]\
nclass = a'),
 Text(637.7142857142857, 906.0, 'NMHC <= 0.215\ngini = 0.004\nsamples = 3211\
nvalue = [9, 5061]\nclass = b'),
 Text(318.85714285714283, 543.5999999999999, 'TOL <= 1.5\ngini = 0.017\nsampl
es = 525\nvalue = [7, 825]\nclass = b'),
 Text(159.42857142857142, 181.19999999999982, 'gini = 0.002\nsamples = 511\nv
alue = [1, 805]\nclass = b'),
 Text(478.2857142857142, 181.19999999999982, 'gini = 0.355\nsamples = 14\nval
ue = [6, 20]\nclass = b'),
 Text(956.5714285714284, 543.5999999999999, 'TOL <= 2.15\ngini = 0.001\nsampl
es = 2686\nvalue = [2, 4236]\nclass = b'),
 Text(797.1428571428571, 181.19999999999982, 'gini = 0.0\nsamples = 2545\nval
ue = [0, 4012]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.018\nsamples = 141\nvalue = [2, 2
24]\nclass = b'),
 Text(2152.285714285714, 1268.4, 'TCH <= 1.435\ngini = 0.326\nsamples = 617\n
value = [790, 204]\nclass = a'),
 Text(1753.7142857142856, 906.0, 'NMHC <= 0.215\ngini = 0.131\nsamples = 489\
nvalue = [742, 56]\nclass = a'),
 Text(1594.2857142857142, 543.5999999999999, 'NO_2 <= 46.5\ngini = 0.005\nsam
ples = 450\nvalue = [742, 2]\nclass = a'),
 Text(1434.8571428571427, 181.19999999999982, 'gini = 0.0\nsamples = 436\nval
ue = [725, 0]\nclass = a'),
 Text(1753.7142857142856, 181.19999999999982, 'gini = 0.188\nsamples = 14\nva
lue = [17, 2]\nclass = a'),
 Text(1913.1428571428569, 543.5999999999999, 'gini = 0.0\nsamples = 39\nvalue
= [0, 54]\nclass = b'),
 Text(2550.8571428571427, 906.0, 'TCH <= 1.515\ngini = 0.37\nsamples = 128\nv
alue = [48, 148]\nclass = b'),
 Text(2232.0, 543.5999999999999, 'O_3 <= 41.5\ngini = 0.488\nsamples = 61\nva
lue = [36, 49]\nclass = b'),
 Text(2072.5714285714284, 181.19999999999982, 'gini = 0.295\nsamples = 39\nva
lue = [9, 41]\nclass = b'),
 Text(2391.428571428571, 181.19999999999982, 'gini = 0.353\nsamples = 22\nval
ue = [27, 8]\nclass = a'),
 Text(2869.7142857142853, 543.5999999999999, 'PM10 <= 17.5\ngini = 0.193\nsam
ples = 67\nvalue = [12, 99]\nclass = b'),
 Text(2710.285714285714, 181.19999999999982, 'gini = 0.18\nsamples = 5\nvalue
= [9, 1]\nclass = a'),
 Text(3029.142857142857, 181.19999999999982, 'gini = 0.058\nsamples = 62\nval
ue = [3, 98]\nclass = b'),
 Text(3507.428571428571, 1630.8000000000002, 'NO_2 <= 12.5\ngini = 0.087\nsam
ples = 2280\nvalue = [3423, 164]\nclass = a'),
 Text(3347.9999999999995, 1268.4, 'gini = 0.0\nsamples = 55\nvalue = [0, 86]\
nclass = b'),
 Text(3666.8571428571427, 1268.4, 'O_3 <= 3.5\ngini = 0.044\nsamples = 2225\n
value = [3423, 78]\nclass = a'),
 Text(3507.428571428571, 906.0, 'gini = 0.0\nsamples = 15\nvalue = [0, 22]\nc
lass = b'),
 Text(3826.2857142857138, 906.0, 'NO_2 <= 15.5\ngini = 0.032\nsamples = 2210\
nvalue = [3423, 56]\nclass = a'),
 Text(3507.428571428571, 543.5999999999999, 'SO_2 <= 4.0\ngini = 0.469\nsampl
es = 14\nvalue = [15, 9]\nclass = a'),
 Text(3347.9999999999995, 181.19999999999982, 'gini = 0.0\nsamples = 5\nvalue
= [0, 9]\nclass = b'),
 Text(3666.8571428571427, 181.19999999999982, 'gini = 0.0\nsamples = 9\nvalue
= [15, 0]\nclass = a'),
 Text(4145.142857142857, 543.5999999999999, 'EBE <= 1.55\ngini = 0.027\nsampl
es = 2196\nvalue = [3408, 47]\nclass = a'),
 Text(3985.7142857142853, 181.19999999999982, 'gini = 0.02\nsamples = 2051\nv
alue = [3195, 32]\nclass = a'),
 Text(4304.571428571428, 181.19999999999982, 'gini = 0.123\nsamples = 145\nva
lue = [213, 15]\nclass = a')]
```


BEN <= 0.25 gini = 0.494 samples = 6178 value = [4333, 5429] class = b
SO_2 <= 4.5 gini = 0.251 samples = 3898 value = [910, 5265] class = b
NO_2 <= 12.5 gini = 0.087 samples = 2280 value = [3423, 164] class = a
NMHC <= 0.145 gini = 0.045 samples = 3281 value = [120, 5061] class = b
TCH <= 1.435 gini = 0.326 samples = 617 value = [790, 204] class = a
gini = 0.0 samples = 55 value = [0, 86] class = b
O_3 <= 3.5 gini = 0.044 samples = 2225 value = [3423, 78] class = a
gini = 0.0 samples = 70 value = [111, 0] class = a
NMHC <= 0.215 gini = 0.004 samples = 3211 value = [9, 5061] class = b
NMHC <= 0.215 gini = 0.131 samples = 489 value = [742, 56] class = a
TCH <= 1.515 gini = 0.37 samples = 128 value = [48, 148] class = b
gini = 0.0 samples = 15 value = [0, 22] class = b
NO_2 <= 15.5 gini = 0.032 samples = 2210 value = [3423, 56] class = a
TOL <= 1.5 gini = 0.017 samples = 525 value = [7, 825] class = b
TOL <= 2.15 gini = 0.001 samples = 2686 value = [2, 4236] class = b
NO_2 <= 46.5 gini = 0.005 samples = 450 value = [742, 2] class = a
gini = 0.0 samples = 39 value = [0, 54] class = b
O_3 <= 41.5 gini = 0.488 samples = 61 value = [36, 49] class = b
PM10 <= 17.5 gini = 0.193 samples = 67 value = [12, 99] class = b
SO_2 <= 4.0 gini = 0.469 samples = 14 value = [15, 9] class = a
EBE <= 1.55 gini = 0.027 samples = 2196 value = [3408, 47] class = a
gini = 0.002 samples = 511 value = [1, 805] class = b
gini = 0.355 samples = 14 value = [6, 20] class = b
gini = 0.0 samples = 2545 value = [0, 4012] class = b
gini = 0.018 samples = 141 value = [2, 224] class = b
gini = 0.0 samples = 436 value = [725, 0] class = a
gini = 0.188 samples = 14 value = [17, 2] class = a
gini = 0.295 samples = 39 value = [9, 41] class = b
gini = 0.353 samples = 22 value = [27, 8] class = a
gini = 0.18 samples = 5 value = [9, 1] class = a
gini = 0.058 samples = 62 value = [3, 98] class = b
gini = 0.0 samples = 5 value = [0, 9] class = b
gini = 0.0 samples = 9 value = [15, 0] class = a
gini = 0.02 samples = 2051 value = [3195, 32] class = a
gini = 0.123 samples = 145 value = [213, 15] class = a

Conclusion

Accuracy

Linear Regression :0.8818792538063775

Ridge Regression :0.8673565597079445

Lasso Regression :0.2675964292464892

ElasticNet Regression : 0.47692368939175334

Logistic Regression : 0.9926143697117453

Random Forest: 0.9960049170351008

Random Forest is suitable for this dataset