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_2015.csv")
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
210091	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN
210092	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN
210093	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN
210094	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN
210095	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN

210096 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:

In [4]:

Out[4]:

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

In [5]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16026 entries, 1 to 210078
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16026 non-null  object 
 1   BEN      16026 non-null  float64
 2   CO       16026 non-null  float64
 3   EBE      16026 non-null  float64
 4   NMHC     16026 non-null  float64
 5   NO       16026 non-null  float64
 6   NO_2     16026 non-null  float64
 7   O_3      16026 non-null  float64
 8   PM10     16026 non-null  float64
 9   PM25     16026 non-null  float64
 10  SO_2     16026 non-null  float64
 11  TCH      16026 non-null  float64
 12  TOL      16026 non-null  float64
 13  station  16026 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.8+ MB
```

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

Out[6]:

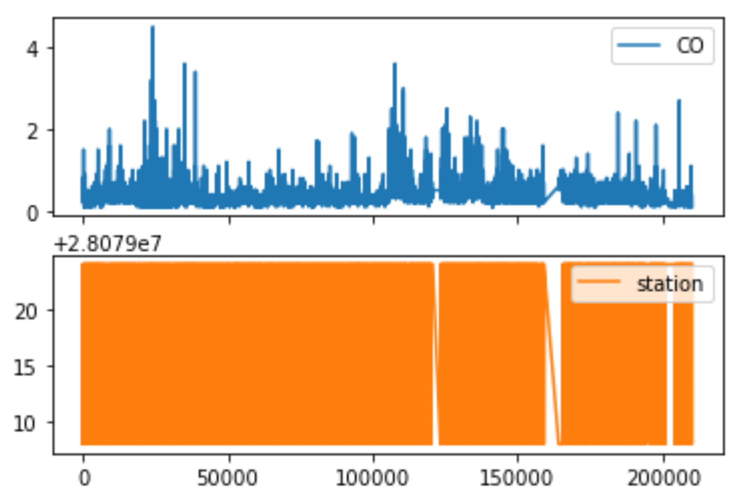
	CO	station
1	0.8	28079008
6	0.3	28079024
25	0.7	28079008
30	0.3	28079024
49	0.8	28079008
...	...	...
210030	0.1	28079024
210049	0.3	28079008
210054	0.1	28079024
210073	0.3	28079008
210078	0.1	28079024

16026 rows × 2 columns

Line chart

In [7]:

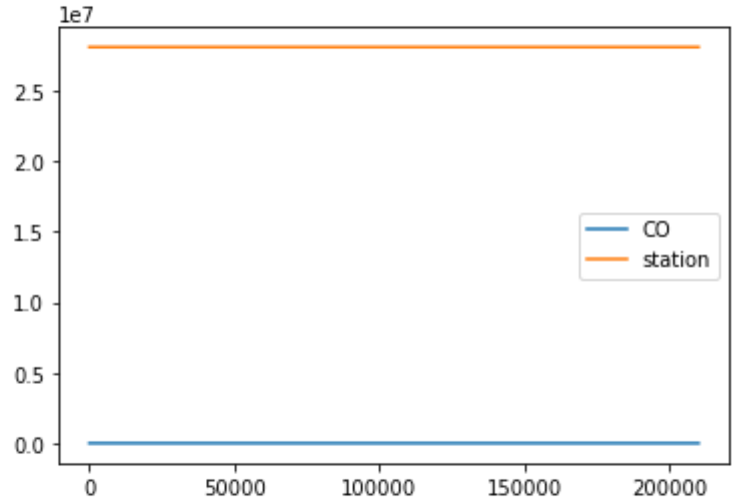
Out[7]: `array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)`



Line chart

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

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

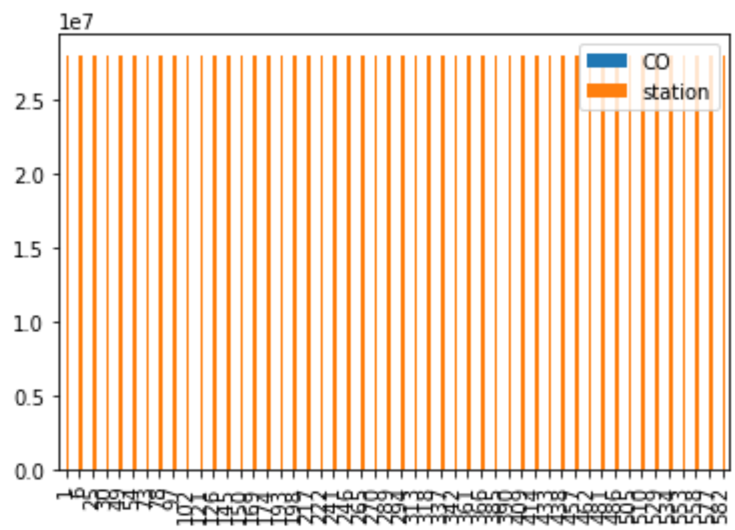


Bar chart

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

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

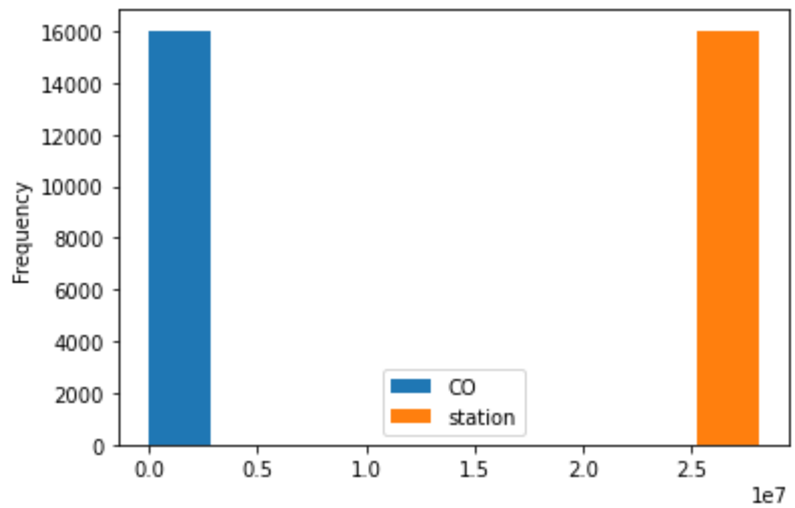
Out[10]:
```
<AxesSubplot:>
```



Histogram

In [11]:

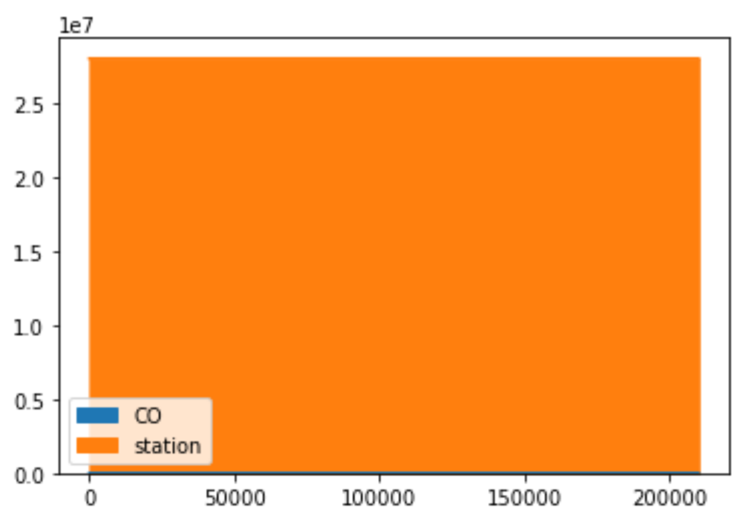
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:

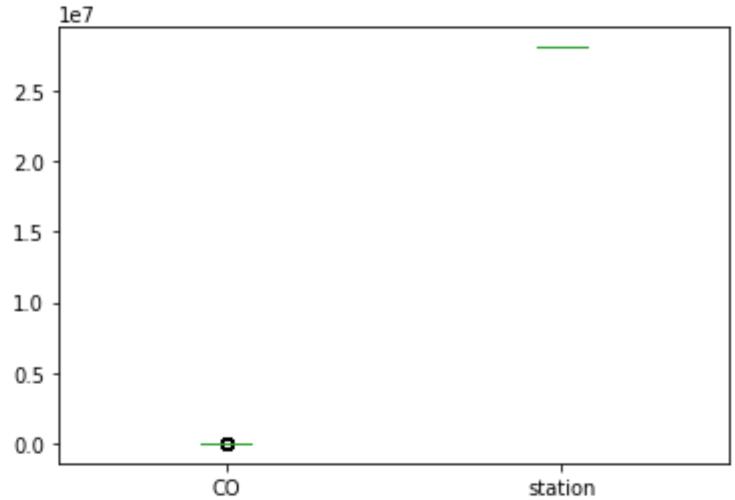
Out[12]: <AxesSubplot:>



Box chart

In [13]:

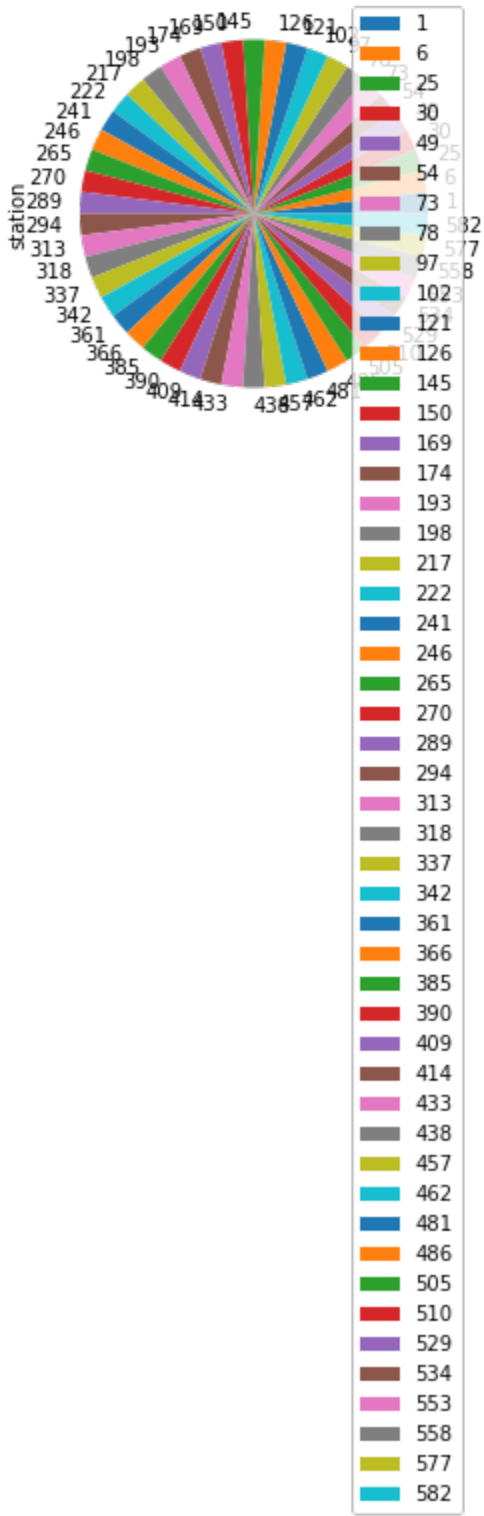
Out[13]: <AxesSubplot:>



Pie chart

In [14]:

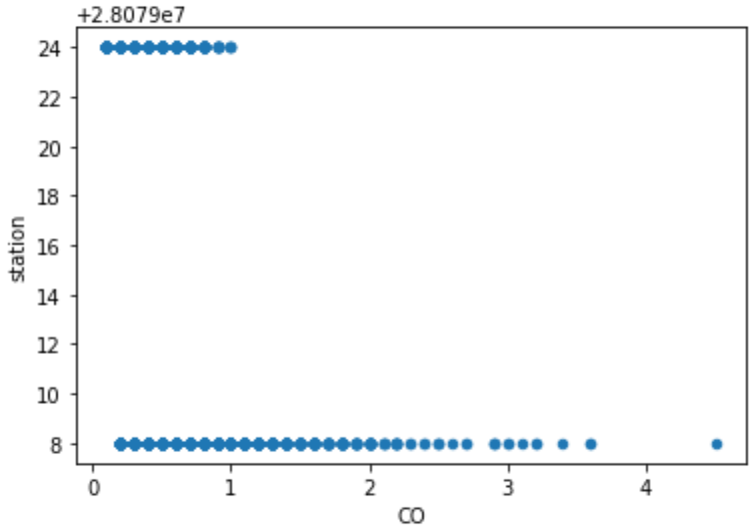
Out[14]: <AxesSubplot:ylabel='station'>



Scatter chart

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

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



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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16026 entries, 1 to 210078
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16026 non-null  object 
 1   BEN      16026 non-null  float64
 2   CO       16026 non-null  float64
 3   EBE      16026 non-null  float64
 4   NMHC     16026 non-null  float64
 5   NO       16026 non-null  float64
 6   NO_2     16026 non-null  float64
 7   O_3      16026 non-null  float64
 8   PM10     16026 non-null  float64
 9   PM25     16026 non-null  float64
 10  SO_2     16026 non-null  float64
 11  TCH      16026 non-null  float64
 12  TOL      16026 non-null  float64
 13  station  16026 non-null  int64  
dtypes: float64(12), int64(1), object(1)
```

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

Out[17]:

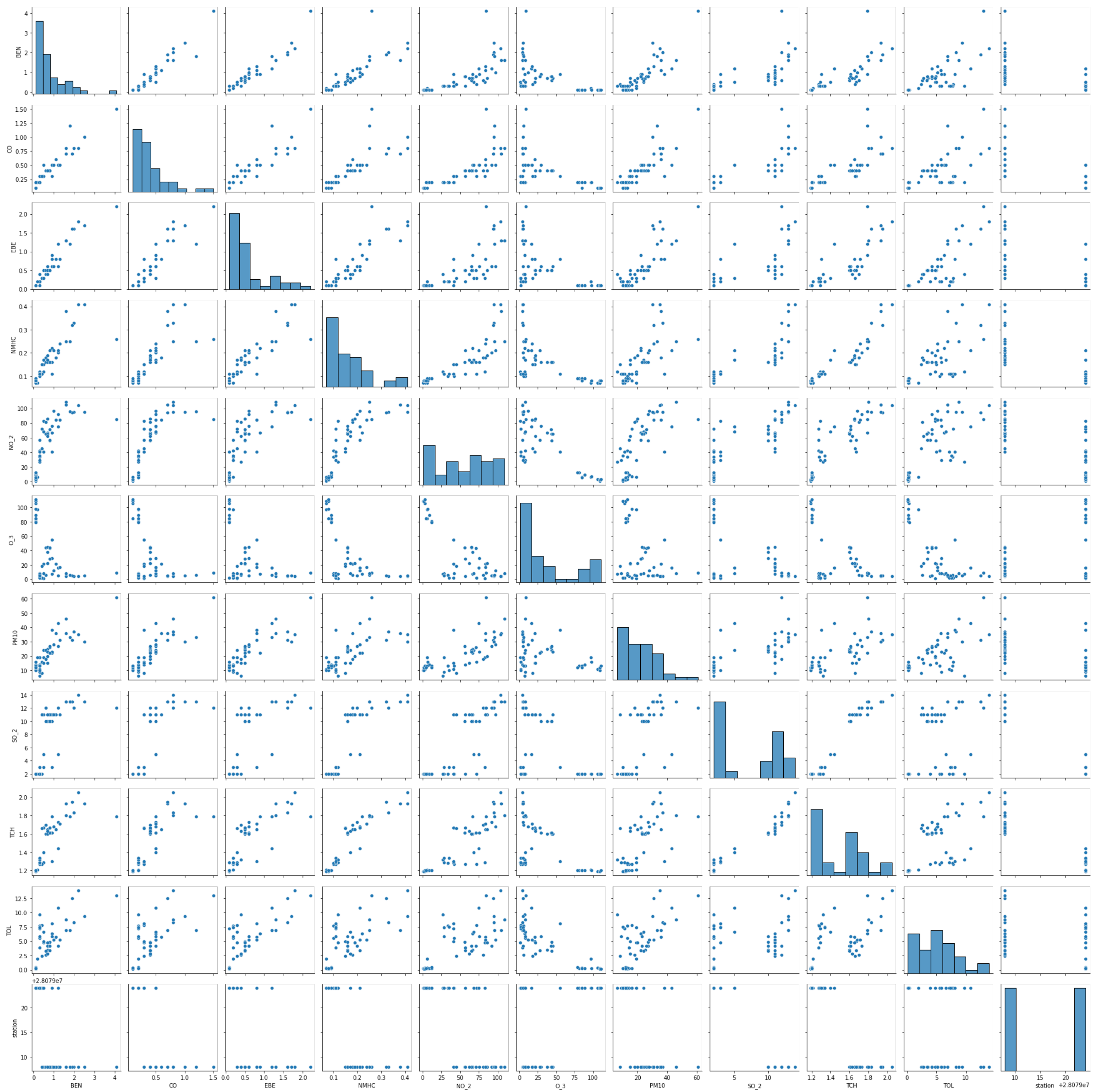
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16026.000000	16026.000000	16026.000000	16026.000000	16026.000000	16026.000000	160
mean	0.504823	0.380594	0.394247	0.123099	23.842256	40.948771	
std	0.716896	0.260805	0.678592	0.092368	51.255660	33.236098	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	
25%	0.100000	0.200000	0.100000	0.070000	1.000000	14.000000	
50%	0.200000	0.300000	0.100000	0.100000	6.000000	35.000000	
75%	0.700000	0.500000	0.400000	0.140000	24.000000	60.000000	
max	17.700001	4.500000	12.100000	1.090000	960.000000	369.000000	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])
```

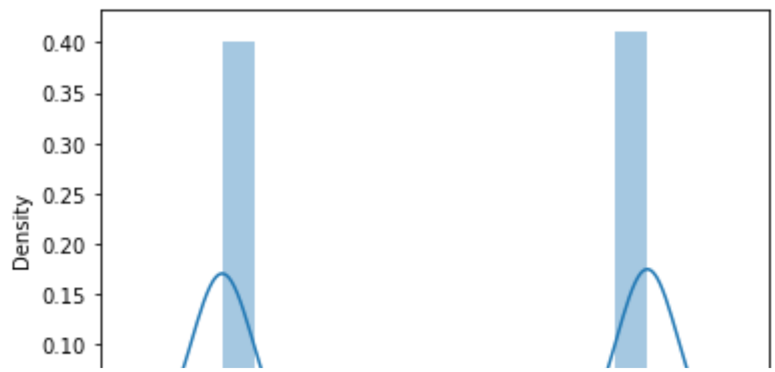
Out[19]: <seaborn.axisgrid.PairGrid at 0x15d6b6e96a0>



In [20]:

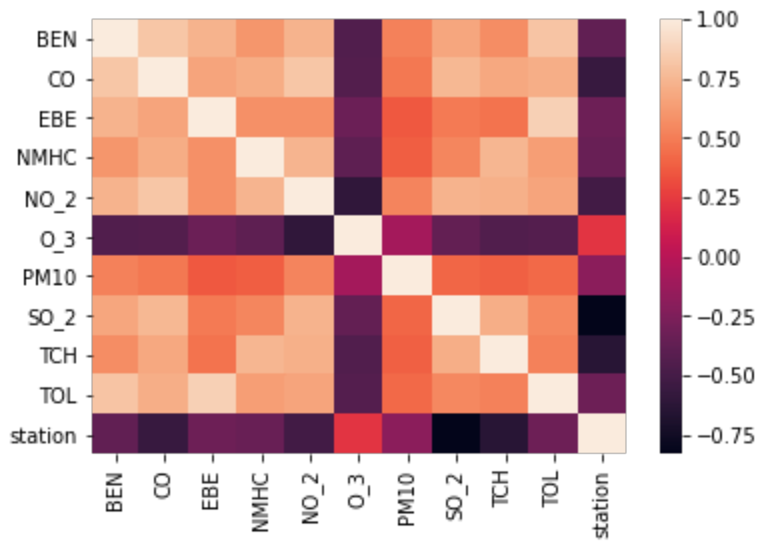
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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

Out[25]: 28079039.851153057

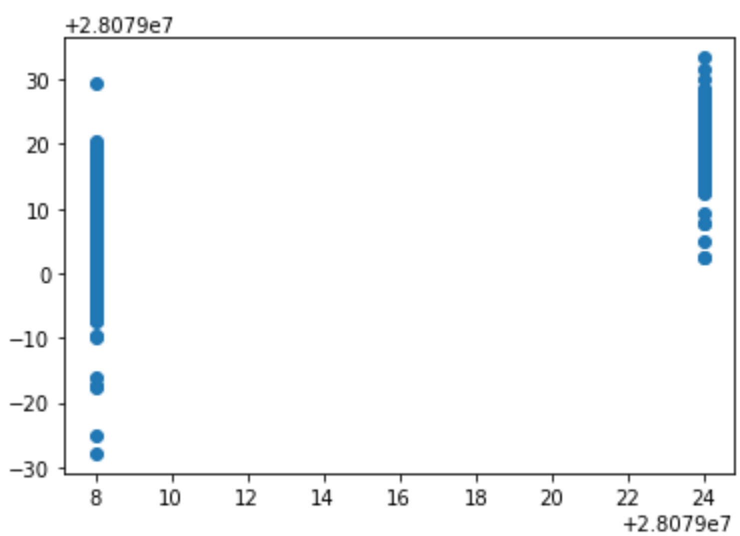
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[26]:

	Co-efficient
BEN	4.628165
CO	-7.125844
EBE	-0.859688
NMHC	25.784782
NO_2	-0.002893
O_3	-0.020427
PM10	0.064927
SO_2	-1.132136
TCH	-12.530883
TOL	-0.117378

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

Out[27]: <matplotlib.collections.PathCollection at 0x15d74cc4850>



ACCURACY

In [28]:

Out[28]: 0.8104176630101703

In [29]:

Out[29]: 0.8064141605763858

Ridge and Lasso

In [30]:

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

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:

Out[32]: 0.8124423265621671

In [33]:

Out[33]: 0.8040435501718737

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

Out[34]: Lasso(alpha=10)

In [35]:

Out[35]: 0.6267277652866841

Accuracy(Lasso)

In [36]:

Out[36]: 0.6451813981381826

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

Out[37]: ElasticNet()

In [38]:

Out[38]:
```
array([ 3.92657467e-02, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
       -1.03946313e-03, -1.42352104e-02,  7.42081514e-02, -1.24758557e+00,
       -0.00000000e+00,  1.81124335e-01])
```

In [39]:

Out[39]:
```
28079023.935699355
```

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

In [41]:

Out[41]:
```
0.7534161872553767
```

Evaluation Metrics

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

```
3.195467973791199
15.773693450734976
3.97161093899377
```

Logistic Regression

In [43]:

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

In [45]:

Out[45]:
```
(16026, 10)
```

In [46]:

Out[46]:
```
(16026,)
```

In [47]:

In [48]:

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

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]:

```
0.9947585174092101
```

In [54]:

Out[54]:

```
1.0
```

In [55]:

Out[55]:

```
array([[1.00000000e+00, 5.69793111e-39]])
```

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]:

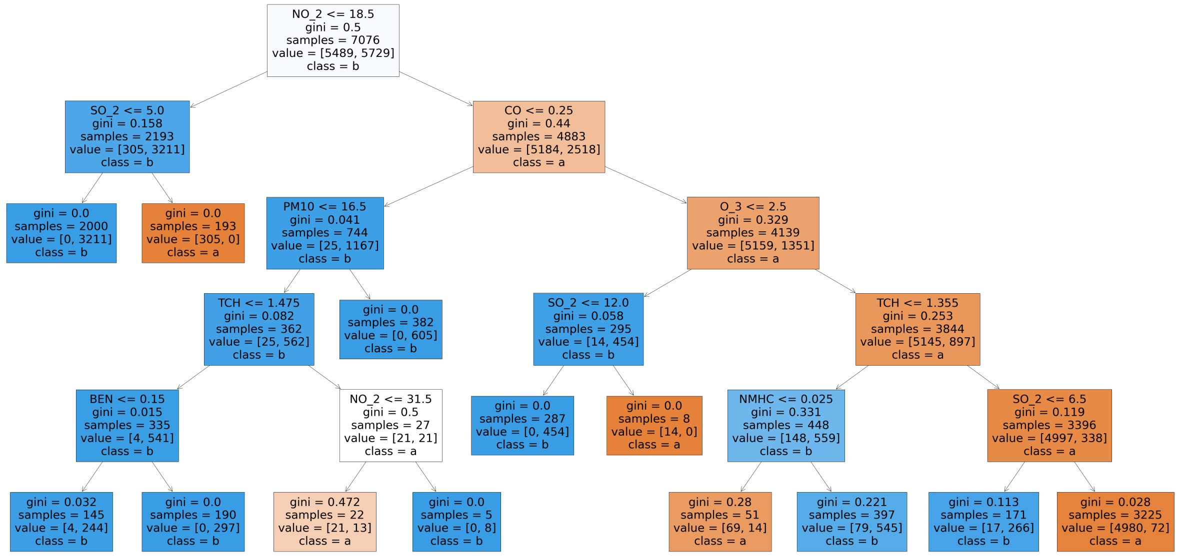
```
0.9950080228204671
```

In [61]:

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

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

Out[62]:
```
[Text(1271.0, 1993.2, 'NO_2 <= 18.5\ngini = 0.5\nsamples = 7076\nvalue = [548
9, 5729]\nclass = b'),
 Text(496.0, 1630.8000000000002, 'SO_2 <= 5.0\ngini = 0.158\nsamples = 2193\n
value = [305, 3211]\nclass = b'),
 Text(248.0, 1268.4, 'gini = 0.0\nsamples = 2000\nvalue = [0, 3211]\nclass =
b'),
 Text(744.0, 1268.4, 'gini = 0.0\nsamples = 193\nvalue = [305, 0]\nclass = a
'),
 Text(2046.0, 1630.8000000000002, 'CO <= 0.25\ngini = 0.44\nsamples = 4883\nv
alue = [5184, 2518]\nclass = a'),
 Text(1240.0, 1268.4, 'PM10 <= 16.5\ngini = 0.041\nsamples = 744\nvalue = [2
5, 1167]\nclass = b'),
 Text(992.0, 906.0, 'TCH <= 1.475\ngini = 0.082\nsamples = 362\nvalue = [25,
562]\nclass = b'),
 Text(496.0, 543.5999999999999, 'BEN <= 0.15\ngini = 0.015\nsamples = 335\nva
lue = [4, 541]\nclass = b'),
 Text(248.0, 181.19999999999982, 'gini = 0.032\nsamples = 145\nvalue = [4, 24
4]\nclass = b'),
 Text(744.0, 181.19999999999982, 'gini = 0.0\nsamples = 190\nvalue = [0, 29
7]\nclass = b'),
 Text(1488.0, 543.5999999999999, 'NO_2 <= 31.5\ngini = 0.5\nsamples = 27\nval
ue = [21, 21]\nclass = a'),
 Text(1240.0, 181.19999999999982, 'gini = 0.472\nsamples = 22\nvalue = [21, 1
3]\nclass = a'),
 Text(1736.0, 181.19999999999982, 'gini = 0.0\nsamples = 5\nvalue = [0, 8]\nc
lass = b'),
 Text(1488.0, 906.0, 'gini = 0.0\nsamples = 382\nvalue = [0, 605]\nclass = b
'),
 Text(2852.0, 1268.4, 'O_3 <= 2.5\ngini = 0.329\nsamples = 4139\nvalue = [515
9, 1351]\nclass = a'),
 Text(2232.0, 906.0, 'SO_2 <= 12.0\ngini = 0.058\nsamples = 295\nvalue = [14,
454]\nclass = b'),
 Text(1984.0, 543.5999999999999, 'gini = 0.0\nsamples = 287\nvalue = [0, 45
4]\nclass = b'),
 Text(2480.0, 543.5999999999999, 'gini = 0.0\nsamples = 8\nvalue = [14, 0]\nc
lass = a'),
 Text(3472.0, 906.0, 'TCH <= 1.355\ngini = 0.253\nsamples = 3844\nvalue = [51
45, 897]\nclass = a'),
 Text(2976.0, 543.5999999999999, 'NMHC <= 0.025\ngini = 0.331\nsamples = 448\
nvalue = [148, 559]\nclass = b'),
 Text(2728.0, 181.19999999999982, 'gini = 0.28\nsamples = 51\nvalue = [69, 1
4]\nclass = a'),
 Text(3224.0, 181.19999999999982, 'gini = 0.221\nsamples = 397\nvalue = [79,
545]\nclass = b'),
 Text(3968.0, 543.5999999999999, 'SO_2 <= 6.5\ngini = 0.119\nsamples = 3396\n
value = [4997, 338]\nclass = a'),
 Text(3720.0, 181.19999999999982, 'gini = 0.113\nsamples = 171\nvalue = [17,
266]\nclass = b'),
 Text(4216.0, 181.19999999999982, 'gini = 0.028\nsamples = 3225\nvalue = [498
0, 72]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression :0.8064141605763858

Ridge Regression :0.6267277652866841

Lasso Regression :0.6451813981381826

ElasticNet Regression : 0.7534161872553767

Logistic Regression : 0.9947585174092101

Random Forest :0.9950080228204671

Random Forest is suitable for this dataset