

Importing Libraries

In [72]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [73]:

```
df=pd.read_csv("madrid_2004.csv")
df
```

Out[73]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002
...
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900000
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689999
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840000
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630001
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389999

245496 rows × 17 columns

Data Cleaning and Data Preprocessing

In [74]:
df=df.dropna()In [75]:
df.columnsOut[75]: Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3', 'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
dtype='object')In [76]:
df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   date      19397 non-null   object 
 1   BEN        19397 non-null   float64
 2   CO         19397 non-null   float64
 3   EBE        19397 non-null   float64
 4   MXY        19397 non-null   float64
 5   NMHC       19397 non-null   float64
 6   NO_2       19397 non-null   float64
 7   NOx        19397 non-null   float64
 8   OXY        19397 non-null   float64
 9   O_3         19397 non-null   float64
 10  PM10       19397 non-null   float64
 11  PM25       19397 non-null   float64
 12  PXY        19397 non-null   float64
 13  SO_2       19397 non-null   float64
 14  TCH        19397 non-null   float64
 15  TOL        19397 non-null   float64
 16  station    19397 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

In [77]:
data=df[['CO', 'station']]
dataOut[77]:

	CO	station
5	0.63	28079006
22	0.36	28079024
26	0.46	28079099
32	0.67	28079006
49	0.30	28079024
...

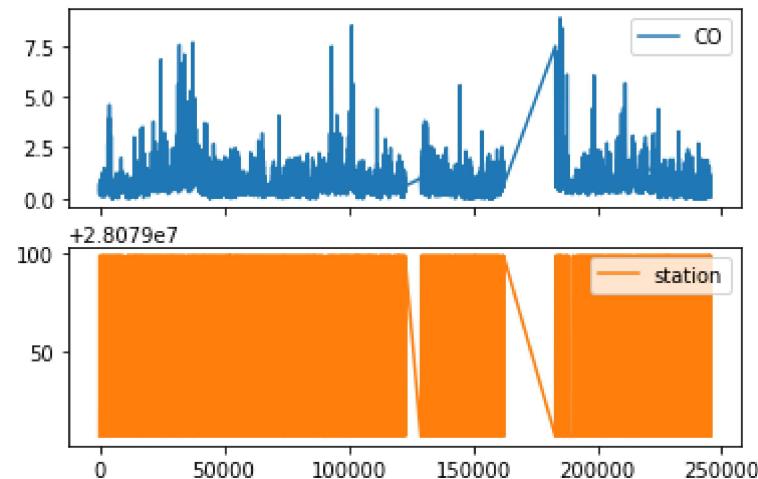
	CO	station
245463	0.08	28079024
245467	0.67	28079099
245473	1.12	28079006
245491	0.21	28079024
245495	0.67	28079099

19397 rows × 2 columns

Line chart

```
In [78]: data.plot.line(subplots=True)
```

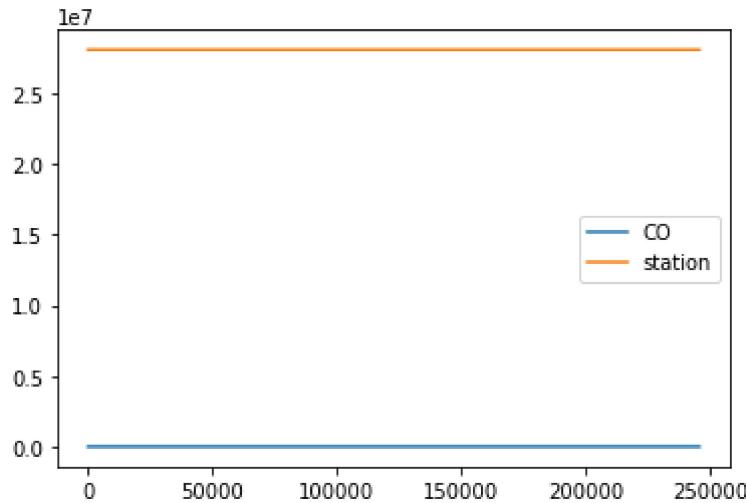
```
Out[78]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

```
In [79]: data.plot.line()
```

```
Out[79]: <AxesSubplot:>
```

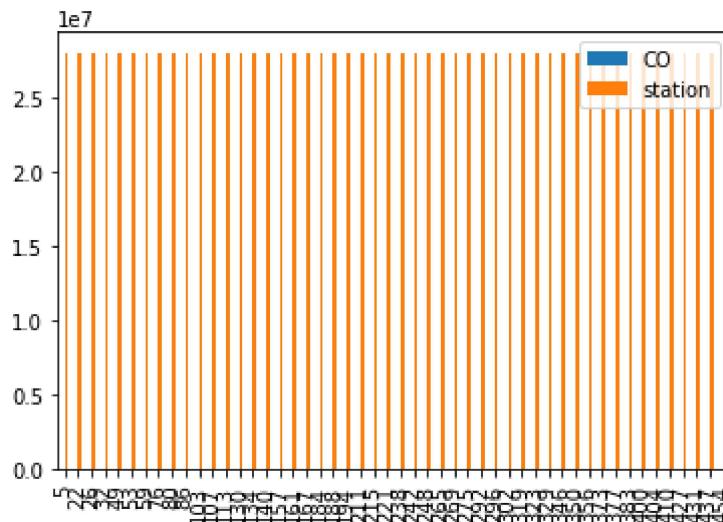


Bar chart

```
In [80]: b=data[0:50]
```

```
In [81]: b.plot.bar()
```

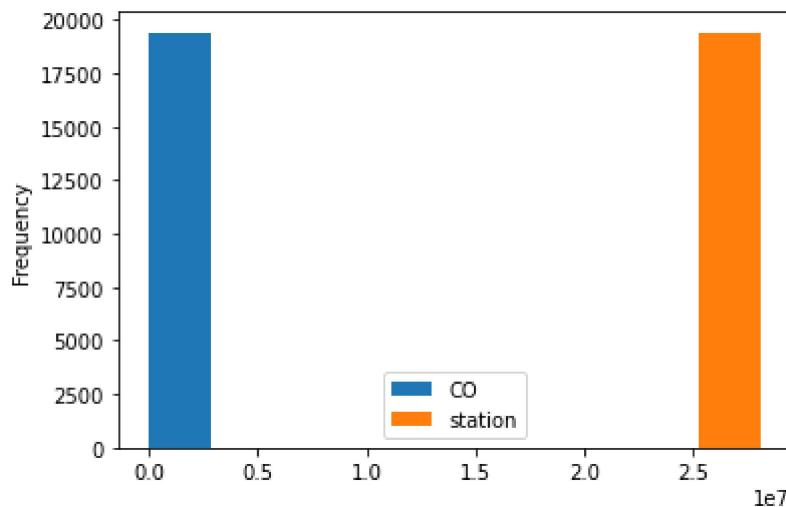
```
Out[81]: <AxesSubplot:>
```



Histogram

```
In [82]: data.plot.hist()
```

```
Out[82]: <AxesSubplot:ylabel='Frequency'>
```

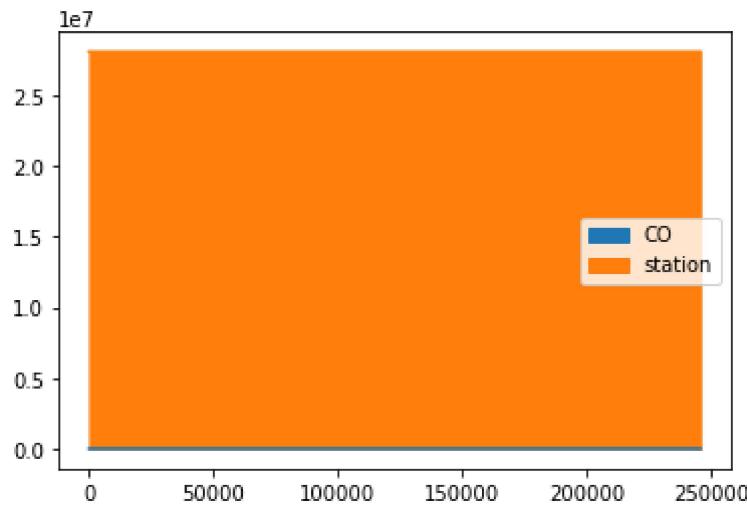


Area chart

In [83]:

```
data.plot.area()
```

Out[83]: <AxesSubplot:>

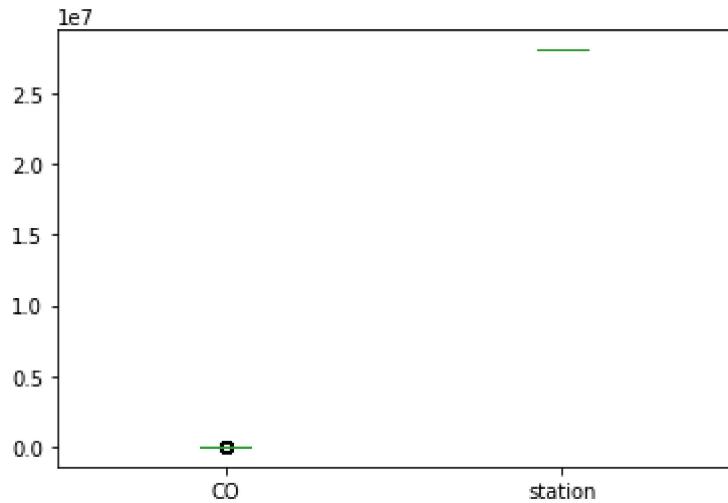


Box chart

In [84]:

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

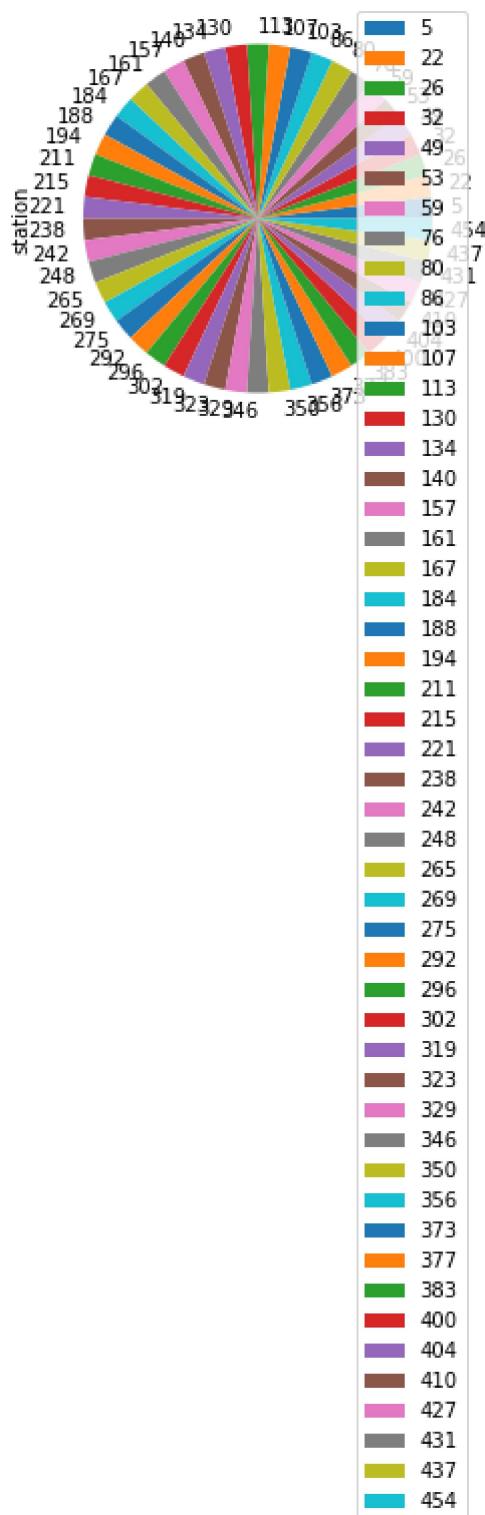
Out[84]: <AxesSubplot:>



Pie chart

```
In [85]: b.plot.pie(y='station' )
```

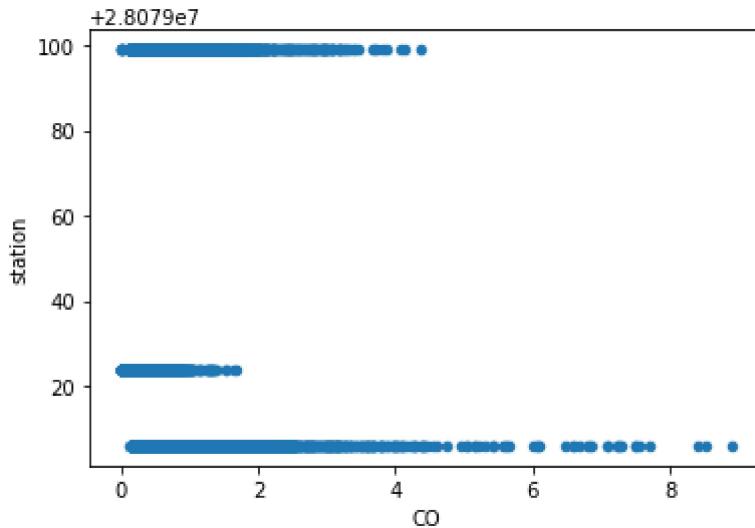
```
Out[85]: <AxesSubplot:ylabel='station'>
```



Scatter chart

```
In [86]: data.plot.scatter(x='CO' ,y='station')
```

```
Out[86]: <AxesSubplot:xlabel='CO', ylabel='station'>
```



In [87]:

`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   date        19397 non-null   object 
 1   BEN          19397 non-null   float64
 2   CO           19397 non-null   float64
 3   EBE          19397 non-null   float64
 4   MXY          19397 non-null   float64
 5   NMHC         19397 non-null   float64
 6   NO_2          19397 non-null   float64
 7   NOx          19397 non-null   float64
 8   OXY          19397 non-null   float64
 9   O_3           19397 non-null   float64
 10  PM10         19397 non-null   float64
 11  PM25         19397 non-null   float64
 12  PXY          19397 non-null   float64
 13  SO_2          19397 non-null   float64
 14  TCH          19397 non-null   float64
 15  TOL          19397 non-null   float64
 16  station       19397 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

In [88]:

`df.describe()`

Out[88]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NO
count	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000
mean	2.250781	0.675347	2.775913	5.424809	0.151024	62.887023	128.554023
std	2.184724	0.591026	2.729622	5.554358	0.158603	37.952255	127.91241
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.090000	2.410000
25%	0.870000	0.320000	1.020000	1.780000	0.060000	35.150002	45.209991
50%	1.620000	0.520000	1.970000	3.800000	0.110000	58.310001	93.22000

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
75%	2.910000	0.860000	3.580000	7.260000	0.200000	85.730003	174.300000
max	34.180000	8.900000	41.880001	91.599998	4.810000	355.100006	1700.000000

In [89]:

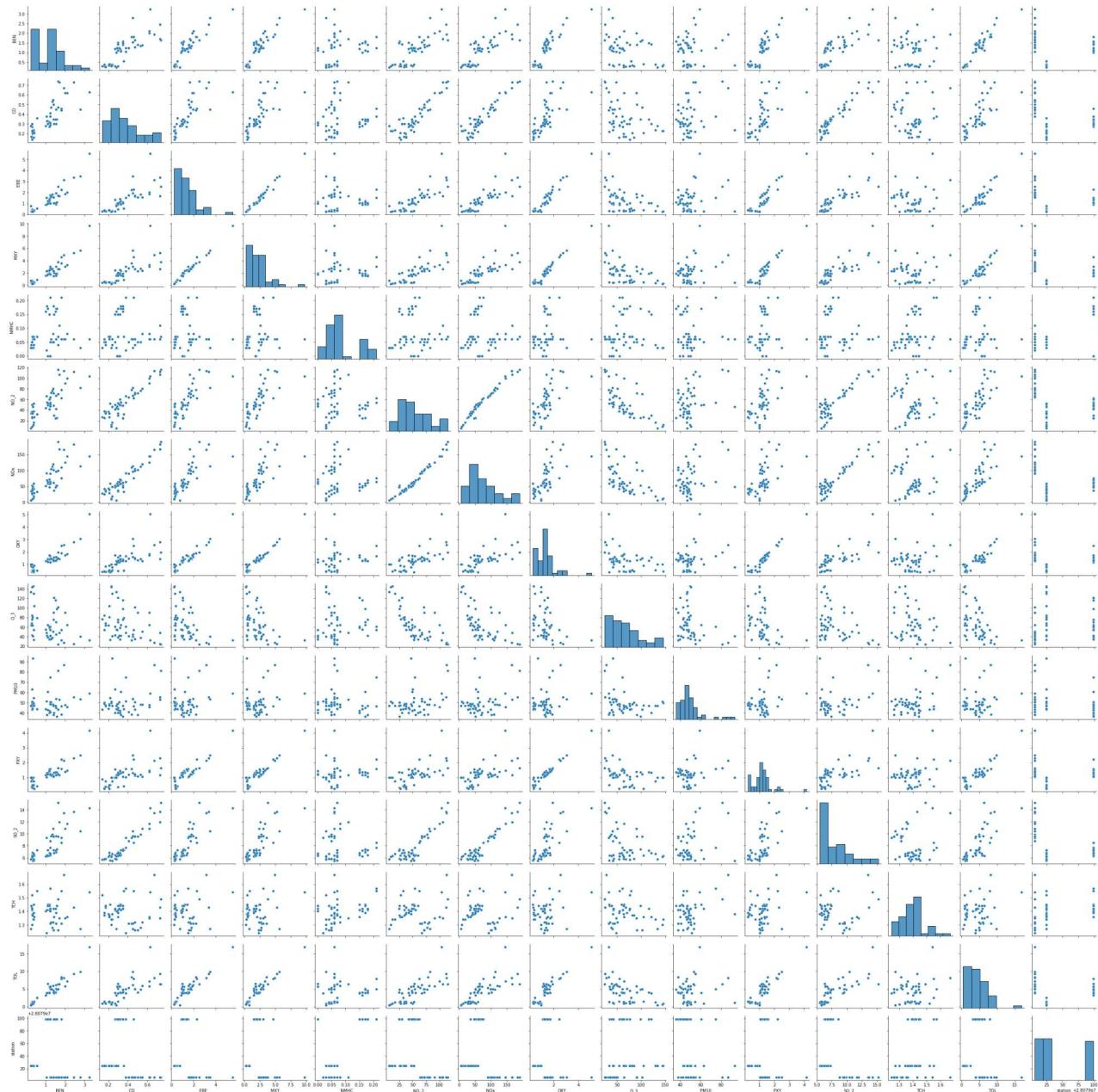
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [90]:

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

Out[90]: <seaborn.axisgrid.PairGrid at 0x1ab4f354880>

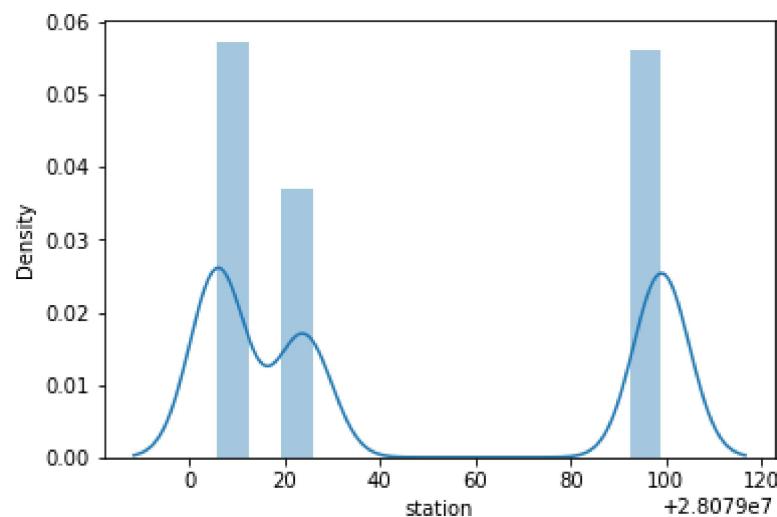


In [91]:

```
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)

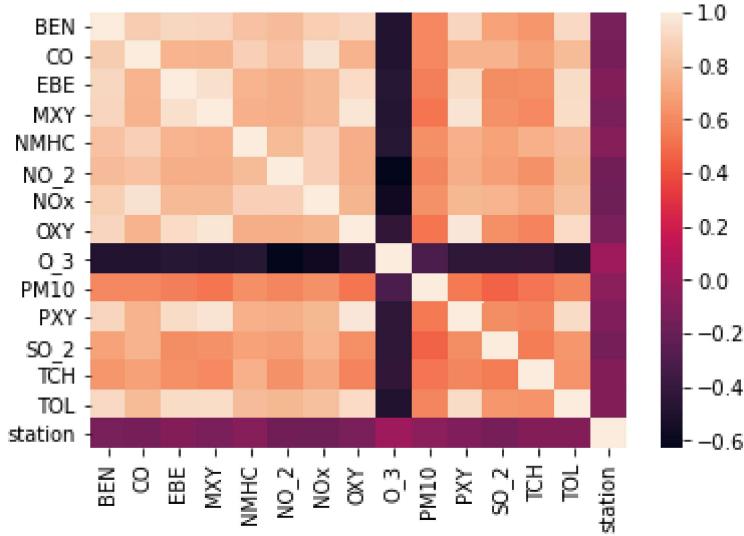
Out[91]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [92]:

```
sns.heatmap(df1.corr())
```

Out[92]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BUILDING

In [93]:

```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

```
In [94]: 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 [95]: from sklearn.linear_model import LinearRegression  
lr=LinearRegression()  
lr.fit(x_train,y_train)
```

```
Out[95]: LinearRegression()
```

```
In [96]: lr.intercept_
```

```
Out[96]: 28079075.61417405
```

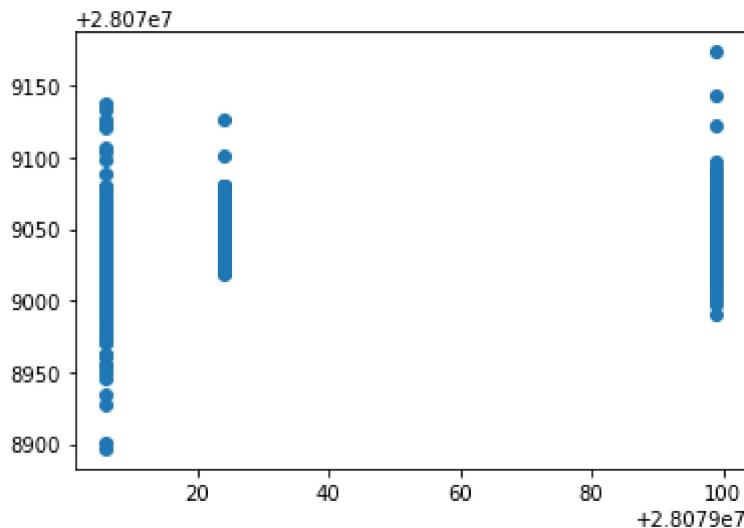
```
In [97]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[97]: Co-efficient
```

	Co-efficient
BEN	-3.517427
CO	27.461094
EBE	3.898673
MXY	-3.478677
NMHC	82.156199
NO_2	-0.143265
NOx	-0.262874
OXY	-2.077880
O_3	-0.296504
PM10	0.063270
PXY	5.877506
SO_2	-0.212709
TCH	-7.104642
TOL	1.089244

```
In [98]: prediction =lr.predict(x_test)  
plt.scatter(y_test,prediction)
```

```
Out[98]: <matplotlib.collections.PathCollection at 0x1ab49a01130>
```



ACCURACY

```
In [99]: lr.score(x_test,y_test)
```

```
Out[99]: 0.111137411829107835
```

```
In [100... lr.score(x_train,y_train)
```

```
Out[100... 0.10400635277362469
```

Ridge and Lasso

```
In [101... from sklearn.linear_model import Ridge,Lasso
```

```
In [102... rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
Out[102... Ridge(alpha=10)
```

Accuracy(Ridge)

```
In [103... rr.score(x_test,y_test)
```

```
Out[103... 0.10962875246299952
```

```
In [104... rr.score(x_train,y_train)
```

```
Out[104... 0.1036576344603446
```

Accuracy(Lasso)

```
In [105... la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
Out[105... Lasso(alpha=10)
```

```
In [106... la.score(x_train,y_train)
```

```
Out[106... 0.052261661780697244
```

ElasticNet

```
In [107... la.score(x_test,y_test)
```

```
Out[107... 0.054773948193416966
```

```
In [108... from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

```
Out[108... ElasticNet()
```

```
In [109... en.coef_
```

```
Out[109... array([-0.          ,  0.45516037,  1.54426098, -1.84529085,  0.
   -0.16525618, -0.08808655, -0.          , -0.22703948,  0.08743881,
   0.45626362, -0.13511099,  0.          ,  1.13316957])
```

```
In [110... en.intercept_
```

```
Out[110... 28079066.72727987
```

```
In [111... prediction=en.predict(x_test)
```

```
In [112... en.score(x_test,y_test)
```

```
Out[112... 0.06880923607719236
```

Evaluation Metrics

```
In [113... 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)))
```

38.55372332959433
1670.234151445275
40.86849827734407

Logistic Regression

In [114...]

```
from sklearn.linear_model import LogisticRegression
```

In [115...]

```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [116...]

```
feature_matrix.shape
```

Out[116...]

```
(19397, 14)
```

In [117...]

```
target_vector.shape
```

Out[117...]

```
(19397, )
```

In [118...]

```
from sklearn.preprocessing import StandardScaler
```

In [119...]

```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [120...]

```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[120...]

```
LogisticRegression(max_iter=10000)
```

In [121...]

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

In [122...]

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

```
[28079006]
```

In [123...]

```
logr.classes_
```

Out[123...]

```
array([28079006, 28079024, 28079099], dtype=int64)
```

In [124...]

```
logr.score(fs,target_vector)
```

```
Out[124... 0.7360416559261741
```

```
In [125... logr.predict_proba(observation)[0][0]
```

```
Out[125... 0.9999978255573396
```

```
In [126... logr.predict_proba(observation)
```

```
Out[126... array([[9.99997826e-01, 7.75018107e-20, 2.17444266e-06]])
```

Random Forest

```
In [127... from sklearn.ensemble import RandomForestClassifier
```

```
In [128... rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
Out[128... RandomForestClassifier()
```

```
In [129... parameters={'max_depth':[1,2,3,4,5],
                 'min_samples_leaf':[5,10,15,20,25],
                 'n_estimators':[10,20,30,40,50]
                }
```

```
In [130... from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

```
Out[130... 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 [131... grid_search.best_score_
```

```
Out[131... 0.7749137266053019
```

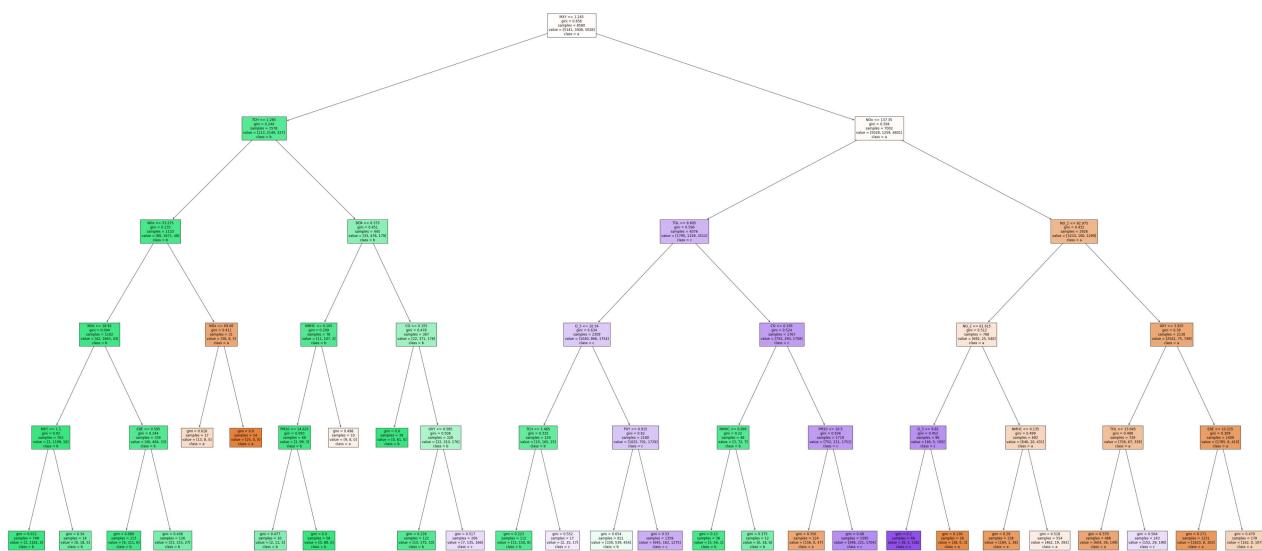
```
In [132... rfc_best=grid_search.best_estimator_
```

```
In [133... 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','c','d'])
```

```
Out[133...]
[Text(2006.653846153846, 1993.2, 'MXY <= 1.245\ngini = 0.656\nsamples = 8580\nvalue = [5
141, 3408, 5028]\nclass = a'),
 Text(922.8461538461538, 1630.8000000000002, 'TCH <= 1.285\ngini = 0.244\nsamples = 1578
\nvalue = [113, 2149, 227]\nclass = b'),
 Text(558.0, 1268.4, 'NOx <= 53.275\ngini = 0.135\nsamples = 1133\nvalue = [80, 1671, 4
8]\nclass = b'),
 Text(343.38461538461536, 906.0, 'NOx <= 18.92\ngini = 0.094\nsamples = 1102\nvalue = [4
2, 1663, 43]\nclass = b'),
 Text(171.69230769230768, 543.5999999999999, 'MXY <= 1.1\ngini = 0.02\nsamples = 763\nva
lue = [2, 1199, 10]\nclass = b'),
 Text(85.84615384615384, 181.1999999999982, 'gini = 0.012\nsamples = 749\nvalue = [2, 1
181, 5]\nclass = b'),
 Text(257.53846153846155, 181.1999999999982, 'gini = 0.34\nsamples = 14\nvalue = [0, 1
8, 5]\nclass = b'),
 Text(515.0769230769231, 543.5999999999999, 'EBE <= 0.595\ngini = 0.244\nsamples = 339\nn
value = [40, 464, 33]\nclass = b'),
 Text(429.23076923076917, 181.1999999999982, 'gini = 0.089\nsamples = 213\nvalue = [9,
311, 6]\nclass = b'),
 Text(600.9230769230769, 181.1999999999982, 'gini = 0.436\nsamples = 126\nvalue = [31,
153, 27]\nclass = b'),
 Text(772.6153846153845, 906.0, 'NOx <= 69.08\ngini = 0.411\nsamples = 31\nvalue = [38,
8, 5]\nclass = a'),
 Text(686.7692307692307, 543.5999999999999, 'gini = 0.618\nsamples = 17\nvalue = [13, 8,
5]\nclass = a'),
 Text(858.4615384615383, 543.5999999999999, 'gini = 0.0\nsamples = 14\nvalue = [25, 0,
0]\nclass = a'),
 Text(1287.6923076923076, 1268.4, 'BEN <= 0.375\ngini = 0.451\nsamples = 445\nvalue = [3
3, 478, 179]\nclass = b'),
 Text(1116.0, 906.0, 'NMHC <= 0.105\ngini = 0.209\nsamples = 78\nvalue = [11, 107, 3]\nc
lass = b'),
 Text(1030.1538461538462, 543.5999999999999, 'PM10 <= 14.625\ngini = 0.093\nsamples = 68
\nvalue = [2, 99, 3]\nclass = b'),
 Text(944.3076923076923, 181.1999999999982, 'gini = 0.477\nsamples = 10\nvalue = [2, 1
1, 3]\nclass = b'),
 Text(1116.0, 181.1999999999982, 'gini = 0.0\nsamples = 58\nvalue = [0, 88, 0]\nclass = b'),
 Text(1201.8461538461538, 543.5999999999999, 'gini = 0.498\nsamples = 10\nvalue = [9, 8,
0]\nclass = a'),
 Text(1459.3846153846152, 906.0, 'CO <= 0.155\ngini = 0.478\nsamples = 367\nvalue = [22,
371, 176]\nclass = b'),
 Text(1373.5384615384614, 543.5999999999999, 'gini = 0.0\nsamples = 39\nvalue = [0, 61,
0]\nclass = b'),
 Text(1545.230769230769, 543.5999999999999, 'OXY <= 0.585\ngini = 0.506\nsamples = 328\nn
value = [22, 310, 176]\nclass = b'),
 Text(1459.3846153846152, 181.1999999999982, 'gini = 0.226\nsamples = 122\nvalue = [15,
175, 10]\nclass = b'),
 Text(1631.0769230769229, 181.1999999999982, 'gini = 0.517\nsamples = 206\nvalue = [7,
135, 166]\nclass = c'),
 Text(3090.461538461538, 1630.8000000000002, 'NOx <= 137.35\ngini = 0.594\nsamples = 700
2\nvalue = [5028, 1259, 4801]\nclass = a'),
 Text(2403.6923076923076, 1268.4, 'TOL <= 6.605\ngini = 0.596\nsamples = 4076\nvalue =
[1795, 1159, 3512]\nclass = c'),
 Text(2060.3076923076924, 906.0, 'O_3 <= 10.34\ngini = 0.634\nsamples = 2309\nvalue = [1
040, 866, 1754]\nclass = c'),
 Text(1888.6153846153845, 543.5999999999999, 'TCH <= 1.465\ngini = 0.332\nsamples = 129
\nvalue = [15, 165, 25]\nclass = b'),
 Text(1802.7692307692307, 181.1999999999982, 'gini = 0.223\nsamples = 112\nvalue = [13,
150, 8]\nclass = b'),
 Text(1974.4615384615383, 181.1999999999982, 'gini = 0.552\nsamples = 17\nvalue = [2, 1
5, 17]\nclass = c'),
 Text(2232.0, 543.5999999999999, 'PXY <= 0.915\ngini = 0.62\nsamples = 2180\nvalue = [10
25, 701, 1729]\nclass = c'),
 Text(2146.153846153846, 181.1999999999982, 'gini = 0.654\nsamples = 821\nvalue = [330,
539, 454]\nclass = b'),
 Text(2317.846153846154, 181.1999999999982, 'gini = 0.53\nsamples = 1359\nvalue = [695,
```

```
162, 1275]\nclass = c'),
Text(2747.076923076923, 906.0, 'CO <= 0.195\ngini = 0.524\nsamples = 1767\nvalue = [75
5, 293, 1758]\nclass = c'),
Text(2575.3846153846152, 543.5999999999999, 'NMHC <= 0.095\ngini = 0.22\nsamples = 48\n
value = [3, 72, 7]\nclass = b'),
Text(2489.5384615384614, 181.1999999999982, 'gini = 0.13\nsamples = 36\nvalue = [3, 5
4, 1]\nclass = b'),
Text(2661.230769230769, 181.1999999999982, 'gini = 0.375\nsamples = 12\nvalue = [0, 1
8, 6]\nclass = b'),
Text(2918.7692307692305, 543.5999999999999, 'PM10 <= 10.5\ngini = 0.504\nsamples = 1719
\nvalue = [752, 221, 1751]\nclass = c'),
Text(2832.9230769230767, 181.1999999999982, 'gini = 0.356\nsamples = 124\nvalue = [15
6, 0, 47]\nclass = a'),
Text(3004.6153846153843, 181.1999999999982, 'gini = 0.48\nsamples = 1595\nvalue = [59
6, 221, 1704]\nclass = c'),
Text(3777.230769230769, 1268.4, 'NO_2 <= 82.975\ngini = 0.432\nsamples = 2926\nvalue =
[3233, 100, 1289]\nclass = a'),
Text(3433.8461538461534, 906.0, 'NO_2 <= 61.915\ngini = 0.512\nsamples = 788\nvalue =
[692, 25, 540]\nclass = a'),
Text(3262.1538461538457, 543.5999999999999, 'O_3 <= 9.81\ngini = 0.452\nsamples = 96\nv
alue = [46, 5, 109]\nclass = c'),
Text(3176.307692307692, 181.1999999999982, 'gini = 0.2\nsamples = 68\nvalue = [8, 5, 1
06]\nclass = c'),
Text(3347.999999999995, 181.1999999999982, 'gini = 0.136\nsamples = 28\nvalue = [38,
0, 3]\nclass = a'),
Text(3605.5384615384614, 543.5999999999999, 'NMHC <= 0.135\ngini = 0.499\nsamples = 692
\nvalue = [646, 20, 431]\nclass = a'),
Text(3519.6923076923076, 181.1999999999982, 'gini = 0.29\nsamples = 138\nvalue = [184,
1, 38]\nclass = a'),
Text(3691.3846153846152, 181.1999999999982, 'gini = 0.518\nsamples = 554\nvalue = [46
2, 19, 393]\nclass = a'),
Text(4120.615384615385, 906.0, 'OXY <= 3.925\ngini = 0.38\nsamples = 2138\nvalue = [254
1, 75, 749]\nclass = a'),
Text(3948.9230769230767, 543.5999999999999, 'TOL <= 13.045\ngini = 0.488\nsamples = 729
\nvalue = [756, 67, 339]\nclass = a'),
Text(3863.076923076923, 181.1999999999982, 'gini = 0.379\nsamples = 486\nvalue = [604,
38, 149]\nclass = a'),
Text(4034.7692307692305, 181.1999999999982, 'gini = 0.564\nsamples = 243\nvalue = [15
2, 29, 190]\nclass = c'),
Text(4292.307692307692, 543.5999999999999, 'EBE <= 10.215\ngini = 0.309\nsamples = 1409
\nvalue = [1785, 8, 410]\nclass = a'),
Text(4206.461538461538, 181.1999999999982, 'gini = 0.271\nsamples = 1231\nvalue = [162
3, 8, 303]\nclass = a'),
Text(4378.153846153846, 181.1999999999982, 'gini = 0.479\nsamples = 178\nvalue = [162,
0, 107]\nclass = a')
```



Conclusion

Accuracy

In [134...]
`lr.score(x_train,y_train)`

Out[134...]
`0.10400635277362469`

In [135...]
`rr.score(x_train,y_train)`

Out[135...]
`0.1036576344603446`

In [136...]
`la.score(x_train,y_train)`

Out[136...]
`0.052261661780697244`

In [137...]
`en.score(x_test,y_test)`

Out[137...]
`0.06880923607719236`

In [138...]
`logr.score(fs,target_vector)`

Out[138...]
`0.7360416559261741`

In [139...]
`grid_search.best_score_`

Out[139...]
`0.7749137266053019`

Random Forest is suitable for this dataset