20104016

DEENA

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_2007.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000	15
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000	8
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000	5
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000	10
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	10
...	...	...	...	...	...	...	...	...	...	...	
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999	
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN	22.150000	
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN	18.540001	1
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN	NaN	
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000	1

225120 rows × 17 columns

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', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25443 non-null  object 
 1   BEN      25443 non-null  float64
 2   CO       25443 non-null  float64
 3   EBE      25443 non-null  float64
 4   MXY      25443 non-null  float64
 5   NMHC     25443 non-null  float64
 6   NO_2     25443 non-null  float64
 7   NOx      25443 non-null  float64
 8   OXY      25443 non-null  float64
 9   O_3      25443 non-null  float64
 10  PM10     25443 non-null  float64
 11  PM25     25443 non-null  float64
 12  PXY      25443 non-null  float64
 13  SO_2     25443 non-null  float64
 14  TCH      25443 non-null  float64
 15  TOL      25443 non-null  float64
 16  station  25443 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

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

Out[6]:

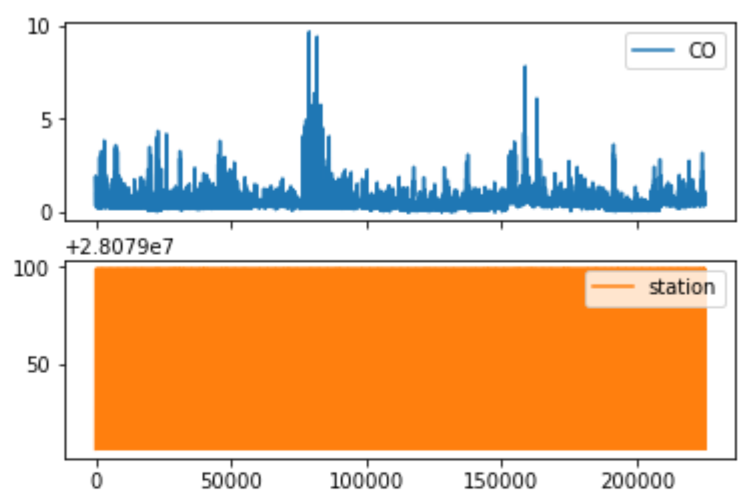
	CO	station
4	1.86	28079006
21	0.31	28079024
25	1.42	28079099
30	1.89	28079006
47	0.30	28079024
...	...	...
225073	0.47	28079006
225094	0.45	28079099
225098	0.41	28079006
225115	0.45	28079024
225119	0.40	28079099

25443 rows × 2 columns

Line chart

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

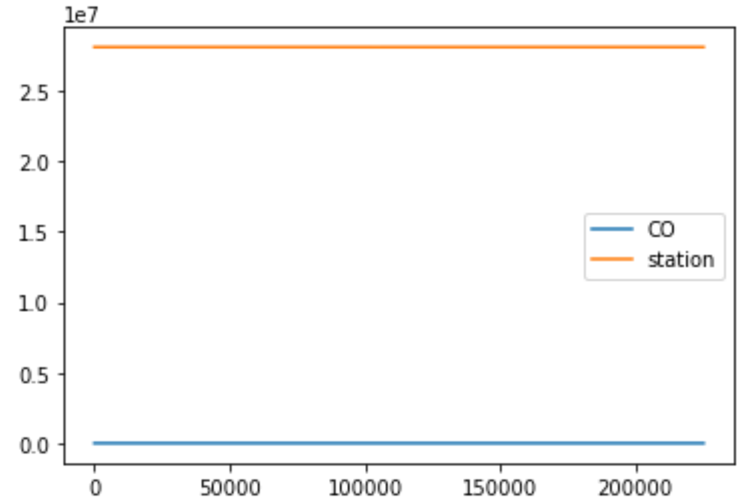
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

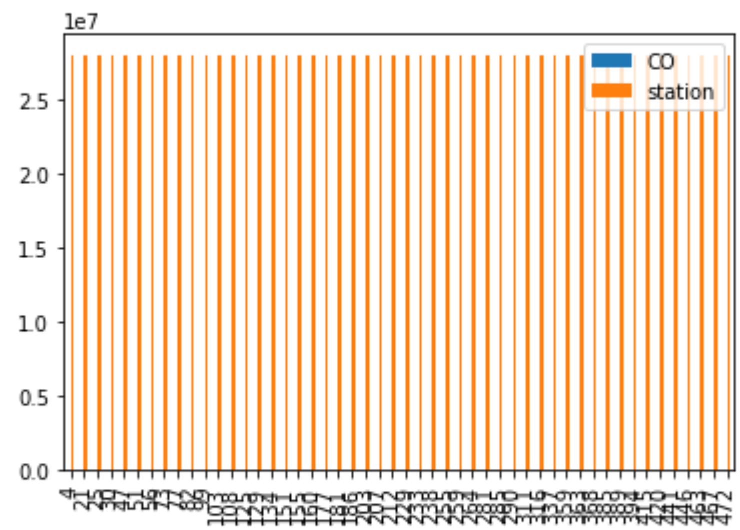


Bar chart

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

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

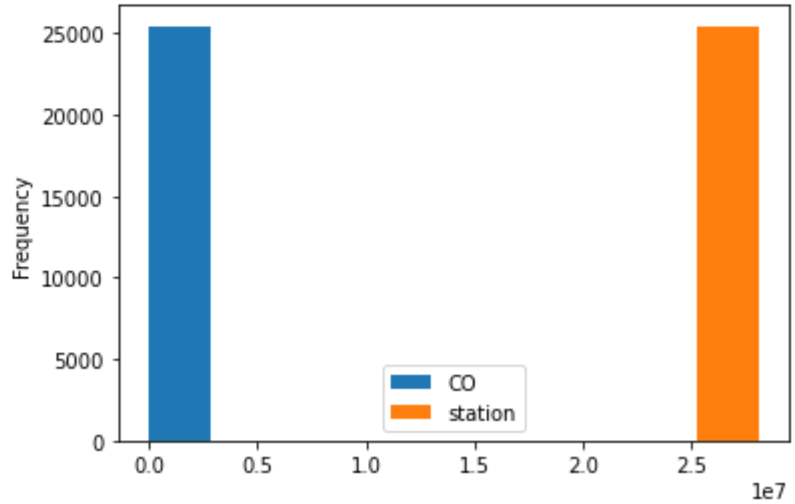
Out[10]: <AxesSubplot:>



Histogram

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

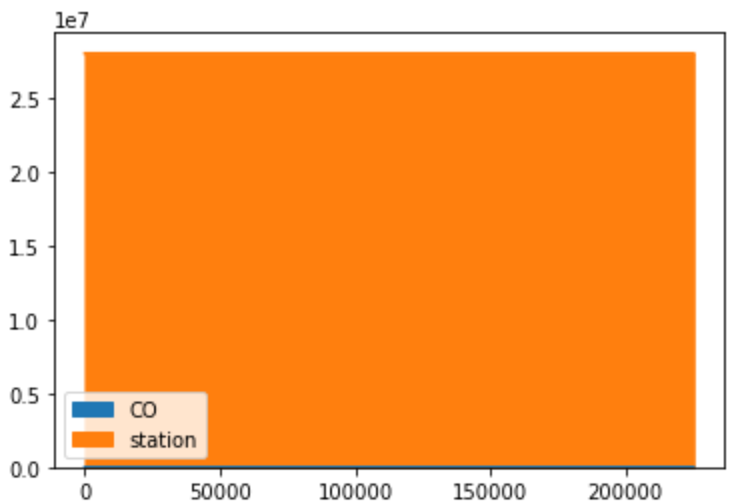
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

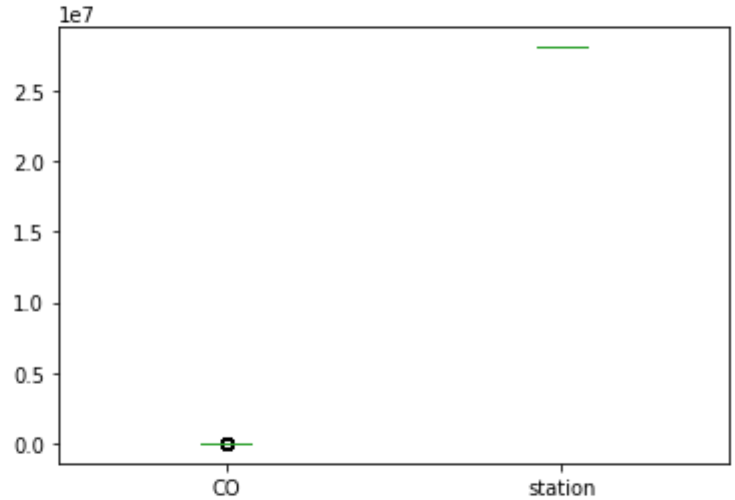
Out[12]: <AxesSubplot:>



Box chart

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

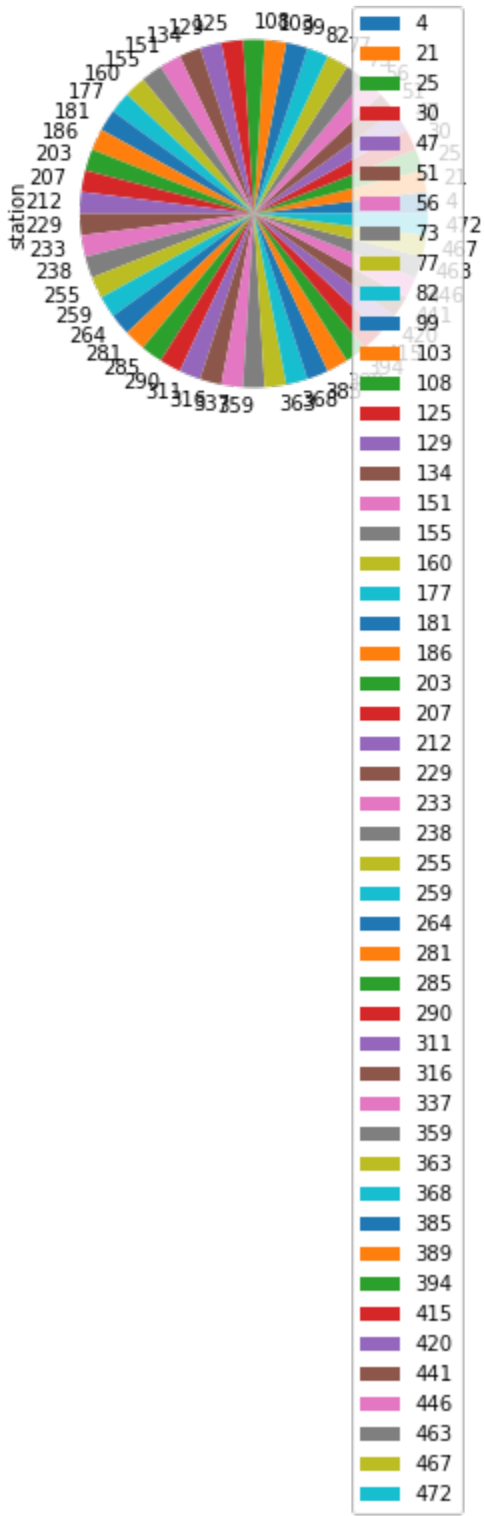
Out[13]: <AxesSubplot:>



Pie chart

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

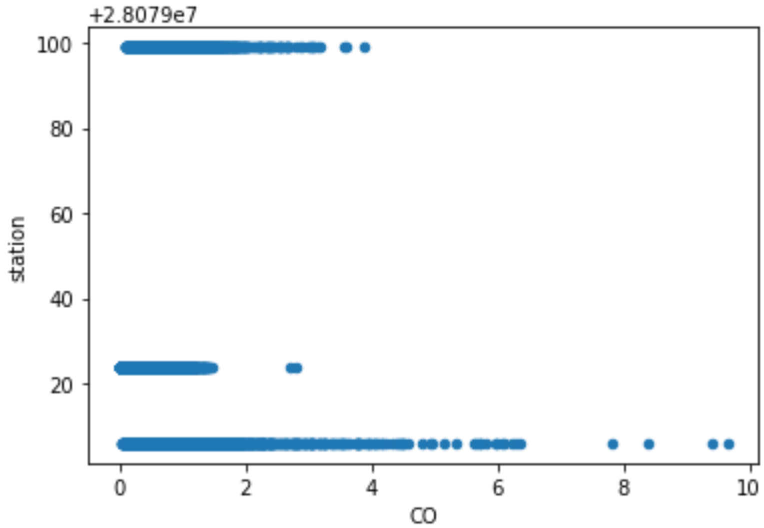
Out[14]: <AxesSubplot:ylabel='station'>



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: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25443 non-null  object 
 1   BEN      25443 non-null  float64
 2   CO       25443 non-null  float64
 3   EBE      25443 non-null  float64
 4   MXY      25443 non-null  float64
 5   NMHC     25443 non-null  float64
 6   NO_2     25443 non-null  float64
 7   NOx      25443 non-null  float64
 8   OXY      25443 non-null  float64
 9   O_3      25443 non-null  float64
 10  PM10     25443 non-null  float64
 11  PM25     25443 non-null  float64
 12  PXY      25443 non-null  float64
 13  SO_2     25443 non-null  float64
 14  TCH      25443 non-null  float64
```

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

Out[17]:

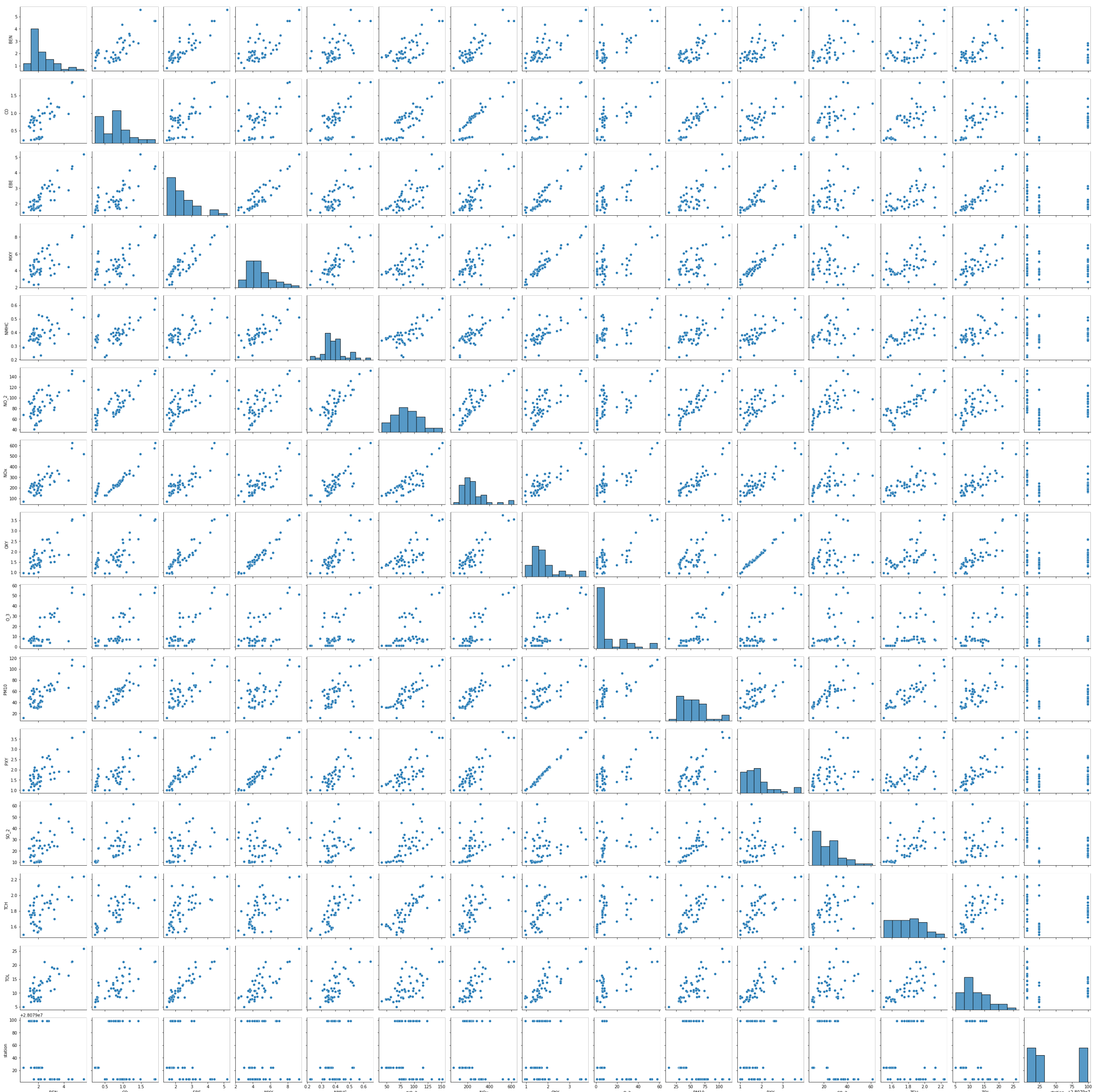
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	254
mean	1.146744	0.505120	1.394071	2.392008	0.249967	58.532683	1
std	1.278733	0.423231	1.268265	2.784302	0.142627	37.755029	1
min	0.130000	0.000000	0.120000	0.150000	0.000000	1.690000	
25%	0.450000	0.260000	0.780000	0.960000	0.160000	31.285001	
50%	0.770000	0.400000	1.000000	1.500000	0.220000	54.080002	
75%	1.390000	0.640000	1.580000	2.855000	0.300000	79.230003	1
max	30.139999	9.660000	31.680000	65.480003	2.570000	430.299988	18

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

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

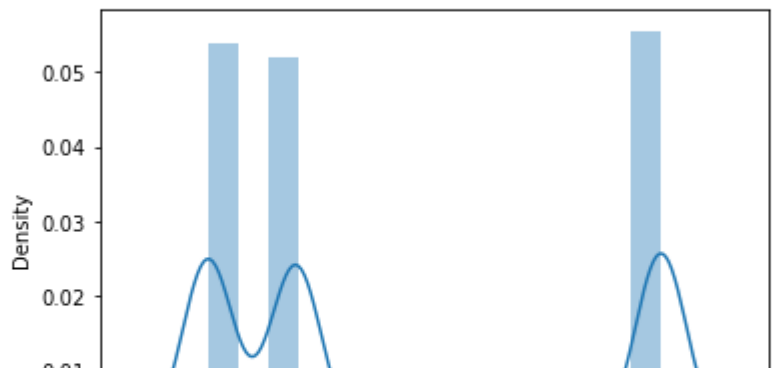
Out[19]: <seaborn.axisgrid.PairGrid at 0x1d69d281100>



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

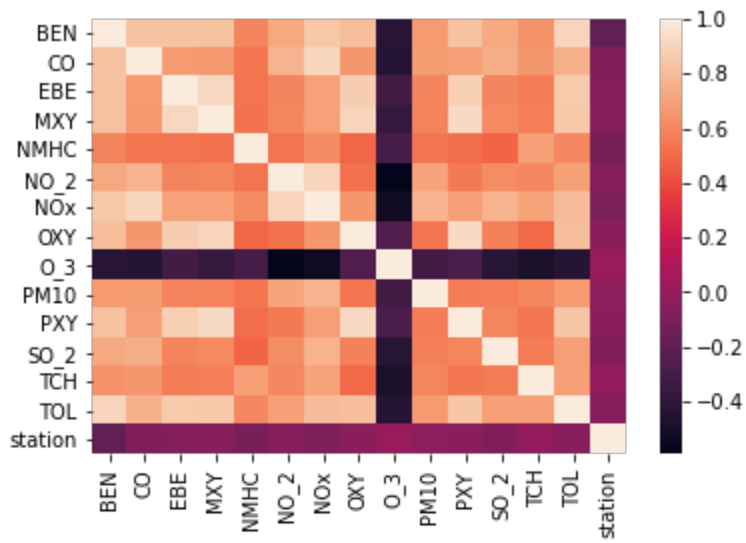
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
        'PM10', 'PXY', '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()
lr.fit(x_train,y_train)
```

Out[24]: LinearRegression()

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

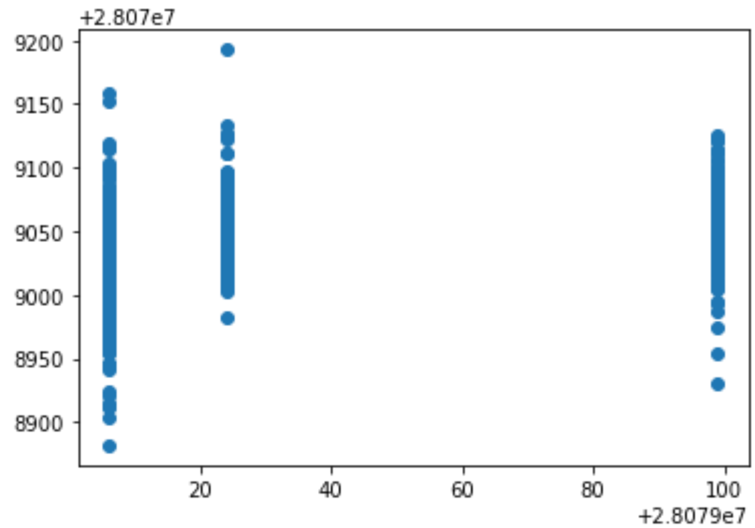
Out[25]: 28079009.192362268

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

Out[26]:

	Co-efficient
BEN	-32.438372
CO	18.343146
EBE	-0.101886
MXY	-1.087555
NMHC	-40.099581
NO_2	0.110310
NOx	-0.040358
OXY	5.547378
O_3	-0.016867
PM10	0.127035
PXY	7.050148
SO_2	0.136247
TCH	26.020745
TOL	3.252164

In [27]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```
Out[27]: <matplotlib.collections.PathCollection at 0x1d6abc1cf70>



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```
Out[28]: 0.16379162671518677

In [29]:
```
lr.score(x_train,y_train)
```
Out[29]: 0.15706639562597824

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]: 0.16369555416612636

In [33]:
```
rr.score(x_train,y_train)
```
Out[33]: 0.1570159207858478

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]:
```
0.01247796461964934
```

Accuracy(Lasso)

In [36]:
```
la.score(x_test,y_test)
```
Out[36]:
```
0.014360120930396403
```

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([-8.03719675,  0.        , -0.        ,  0.14618211, -0.        ,
        0.04709276, -0.04847005,  0.64915365, -0.0485963 ,  0.14996697,
        0.75777493, -0.01733057,  0.        ,  1.01190089])
```

In [39]:
```
en.intercept_
```
Out[39]:
```
28079045.901221745
```

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

In [41]:
```
en.score(x_test,y_test)
```
Out[41]:
```
0.06928545501091632
```

Evaluation Metrics

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)))
```
```
36.455485646200415
1521.148584724685
39.00190488584737
```

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]:
```
(25443, 14)
```

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

Out[46]:
```
(25443,)
```

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)
```

```
[28079099]
```

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

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

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

Out[53]:
```
0.8146838030106512
```

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

Out[54]:
```
1.082753977181323e-19
```

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

Out[55]:
```
array([[1.08275398e-19, 1.80383815e-19, 1.00000000e+00]])
```

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]:
```
0.8235261089275687
```

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

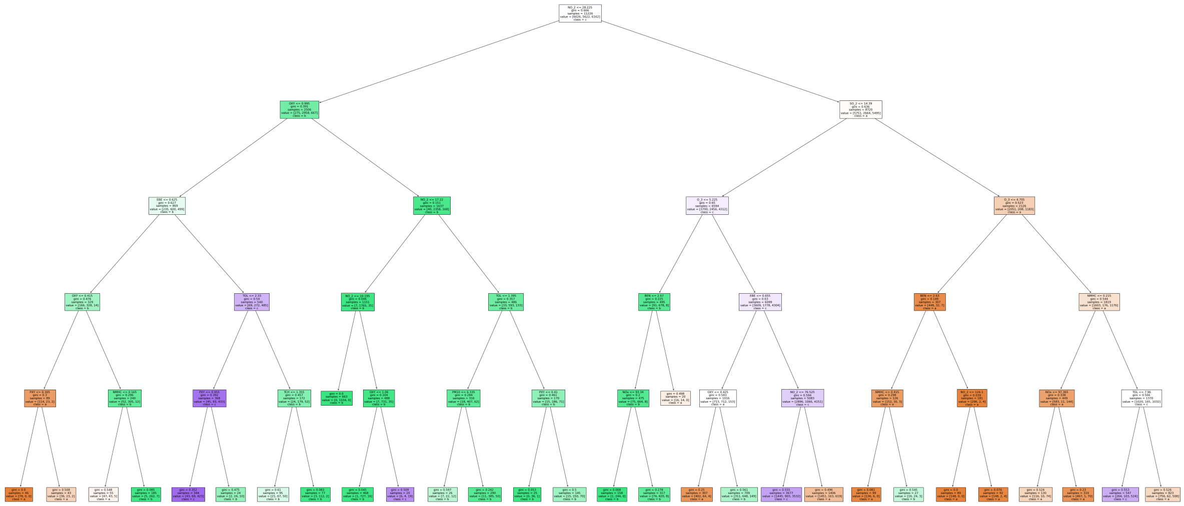
In [62]:
```
from sklearn.tree import plot_tree

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

Out[62]:
```
[Text(2192.142857142857, 1993.2, 'NO_2 <= 28.225\ngini = 0.666\nsamples = 112
26\nvalue = [6026, 5622, 6162]\nclass = c'),
 Text(1135.9285714285713, 1630.8000000000002, 'OXY <= 0.995\ngini = 0.391\nsa
mples = 2506\nvalue = [275, 2958, 667]\nclass = b'),
 Text(637.7142857142857, 1268.4, 'EBE <= 0.625\ngini = 0.627\nsamples = 869\n
value = [235, 600, 499]\nclass = b'),
 Text(318.85714285714283, 906.0, 'OXY <= 0.415\ngini = 0.476\nsamples = 329\n
value = [166, 328, 14]\nclass = b'),
 Text(159.42857142857142, 543.5999999999999, 'PXY <= 0.305\ngini = 0.3\nsampl
es = 89\nvalue = [114, 23, 2]\nclass = a'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.0\nsamples = 46\nvalue
= [78, 0, 0]\nclass = a'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.508\nsamples = 43\nval
ue = [36, 23, 2]\nclass = a'),
 Text(478.2857142857142, 543.5999999999999, 'NMHC <= 0.165\ngini = 0.296\nsam
ples = 240\nvalue = [52, 305, 12]\nclass = b'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.548\nsamples = 55\nva
lue = [47, 43, 5]\nclass = a'),
 Text(558.0, 181.19999999999982, 'gini = 0.085\nsamples = 185\nvalue = [5, 26
2, 7]\nclass = b'),
 Text(956.5714285714284, 906.0, 'TOL <= 2.33\ngini = 0.54\nsamples = 540\nval
ue = [69, 272, 485]\nclass = c'),
 Text(797.1428571428571, 543.5999999999999, 'PXY <= 0.955\ngini = 0.392\nsamp
les = 368\nvalue = [45, 93, 433]\nclass = c'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.352\nsamples = 344\nva
lue = [43, 69, 423]\nclass = c'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.475\nsamples = 24\nval
ue = [2, 24, 10]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'TCH <= 1.355\ngini = 0.457\nsamples = 172\n
value = [24, 179, 52]\nclass = b'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.61\nsamples = 95\nval
ue = [21, 67, 50]\nclass = b'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.083\nsamples = 77\nva
lue = [3, 112, 2]\nclass = b'),
 Text(1634.142857142857, 1268.4, 'NO_2 <= 17.22\ngini = 0.151\nsamples = 163
7\nvalue = [40, 2358, 168]\nclass = b'),
 Text(1355.142857142857, 906.0, 'NO_2 <= 10.195\ngini = 0.046\nsamples = 115
1\nvalue = [7, 1765, 35]\nclass = b'),
 Text(1275.4285714285713, 543.5999999999999, 'gini = 0.0\nsamples = 663\nvalu
e = [0, 1034, 0]\nclass = b'),
 Text(1434.8571428571427, 543.5999999999999, 'OXY <= 1.06\ngini = 0.104\nsamp
les = 488\nvalue = [7, 731, 35]\nclass = b'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.045\nsamples = 468\nva
lue = [1, 727, 16]\nclass = b'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.509\nsamples = 20\nva
lue = [6, 4, 19]\nclass = c'),
 Text(1913.1428571428569, 906.0, 'TOL <= 1.395\ngini = 0.357\nsamples = 486\n
value = [33, 593, 133]\nclass = b'),
 Text(1753.7142857142856, 543.5999999999999, 'PM10 <= 6.335\ngini = 0.284\nsa
mples = 316\nvalue = [18, 407, 62]\nclass = b'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.597\nsamples = 26\nva
```

```
lue = [7, 22, 12]\nclass = b'),
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.242\nsamples = 290\nv
alue = [11, 385, 50]\nclass = b'),
 Text(2072.5714285714284, 543.5999999999999, 'PXY <= 0.41\ngini = 0.461\nsamp
les = 170\nvalue = [15, 186, 71]\nclass = b'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.053\nsamples = 25\nva
lue = [0, 36, 1]\nclass = b'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.5\nsamples = 145\nvalu
e = [15, 150, 70]\nclass = b'),
 Text(3248.3571428571427, 1630.8000000000002, 'SO_2 <= 14.39\ngini = 0.636\ns
amples = 8720\nvalue = [5751, 2664, 5495]\nclass = a'),
 Text(2670.428571428571, 1268.4, 'O_3 <= 5.225\ngini = 0.65\nsamples = 6594\n
value = [3700, 2456, 4312]\nclass = c'),
 Text(2471.142857142857, 906.0, 'BEN <= 2.67\ngini = 0.225\nsamples = 495\nva
lue = [91, 678, 8]\nclass = b'),
 Text(2391.428571428571, 543.5999999999999, 'NOx <= 93.34\ngini = 0.2\nsample
s = 475\nvalue = [75, 664, 8]\nclass = b'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.008\nsamples = 158\nv
alue = [1, 244, 0]\nclass = b'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.278\nsamples = 317\nva
lue = [74, 420, 8]\nclass = b'),
 Text(2550.8571428571427, 543.5999999999999, 'gini = 0.498\nsamples = 20\nval
ue = [16, 14, 0]\nclass = a'),
 Text(2869.7142857142853, 906.0, 'EBE <= 0.655\ngini = 0.63\nsamples = 6099\n
value = [3609, 1778, 4304]\nclass = c'),
 Text(2710.285714285714, 543.5999999999999, 'OXY <= 0.425\ngini = 0.583\nsamp
les = 1016\nvalue = [713, 712, 153]\nclass = a'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.25\nsamples = 307\nva
lue = [402, 64, 4]\nclass = a'),
 Text(2790.0, 181.19999999999982, 'gini = 0.561\nsamples = 709\nvalue = [311,
648, 149]\nclass = b'),
 Text(3029.142857142857, 543.5999999999999, 'NO_2 <= 76.525\ngini = 0.594\nsa
mples = 5083\nvalue = [2896, 1066, 4151]\nclass = c'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.555\nsamples = 3677\nv
alue = [1445, 903, 3532]\nclass = c'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.496\nsamples = 1406\n
value = [1451, 163, 619]\nclass = a'),
 Text(3826.2857142857138, 1268.4, 'O_3 <= 4.705\ngini = 0.523\nsamples = 212
6\nvalue = [2051, 208, 1183]\nclass = a'),
 Text(3507.428571428571, 906.0, 'BEN <= 2.63\ngini = 0.149\nsamples = 307\nva
lue = [448, 32, 7]\nclass = a'),
 Text(3347.9999999999995, 543.5999999999999, 'NMHC <= 0.415\ngini = 0.298\nsa
mples = 126\nvalue = [152, 30, 3]\nclass = a'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.081\nsamples = 99\nval
ue = [136, 6, 0]\nclass = a'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.545\nsamples = 27\nva
lue = [16, 24, 3]\nclass = b'),
 Text(3666.8571428571427, 543.5999999999999, 'NO_2 <= 126.1\ngini = 0.039\nsa
mples = 181\nvalue = [296, 2, 4]\nclass = a'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.0\nsamples = 89\nvalue
= [148, 0, 0]\nclass = a'),
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.076\nsamples = 92\nva
lue = [148, 2, 4]\nclass = a'),
 Text(4145.142857142857, 906.0, 'NMHC <= 0.215\ngini = 0.544\nsamples = 1819\
nvalue = [1603, 176, 1176]\nclass = a'),
 Text(3985.7142857142853, 543.5999999999999, 'NOx <= 97.365\ngini = 0.338\nsa
```

```
mples = 449\nvalue = [583, 11, 144]\nclass = a'),
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.524\nsamples = 130\nv
alue = [116, 10, 74]\nclass = a'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.23\nsamples = 319\nval
ue = [467, 1, 70]\nclass = a'),
 Text(4304.571428571428, 543.5999999999999, 'TOL <= 7.96\ngini = 0.566\nsampl
es = 1370\nvalue = [1020, 165, 1032]\nclass = c'),
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.553\nsamples = 547\nva
lue = [264, 103, 524]\nclass = c'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.526\nsamples = 823\nva
```



Conclusion

Accuracy

Linear Regression :0.15706639562597824

Ridge Regression : 0.012477964619649934

Lasso Regression : 0.014360120930396403

ElasticNet Regression : 0.06928545501091632

Logistic Regression : 0.8146838030106512

Random Forest :0.8235261089275687

Random Forest is suitable for this dataset