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_2011.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	2
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	2
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	2
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	2
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	2
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	2
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	2
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	2
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	2

209928 rows × 14 columns

Data Cleaning and Data Preprocessing

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

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

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16460 entries, 1 to 209910
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16460 non-null  object 
 1   BEN      16460 non-null  float64
 2   CO       16460 non-null  float64
 3   EBE      16460 non-null  float64
 4   NMHC     16460 non-null  float64
 5   NO       16460 non-null  float64
 6   NO_2     16460 non-null  float64
 7   O_3      16460 non-null  float64
 8   PM10     16460 non-null  float64
 9   PM25     16460 non-null  float64
 10  SO_2     16460 non-null  float64
 11  TCH      16460 non-null  float64
 12  TOL      16460 non-null  float64
 13  station  16460 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.9+ MB
```

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

Out[6]:

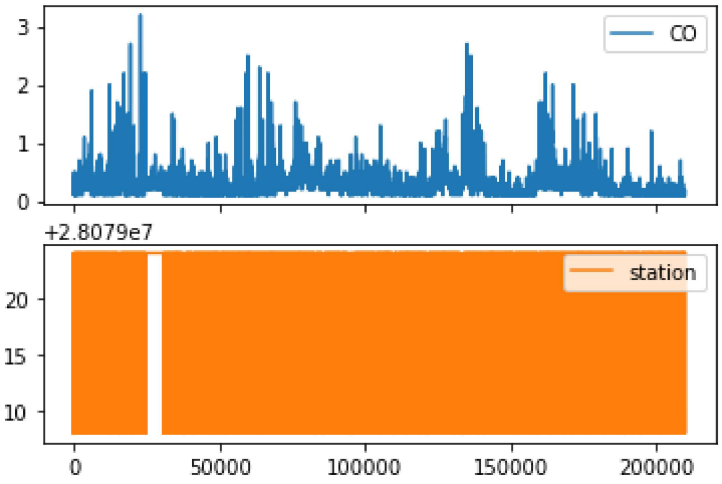
	CO	station
1	0.4	28079008
6	0.3	28079024
25	0.3	28079008
30	0.4	28079024
49	0.2	28079008
...	...	...
209862	0.1	28079024
209881	0.1	28079008
209886	0.1	28079024
209905	0.1	28079008
209910	0.1	28079024

16460 rows × 2 columns

Line chart

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

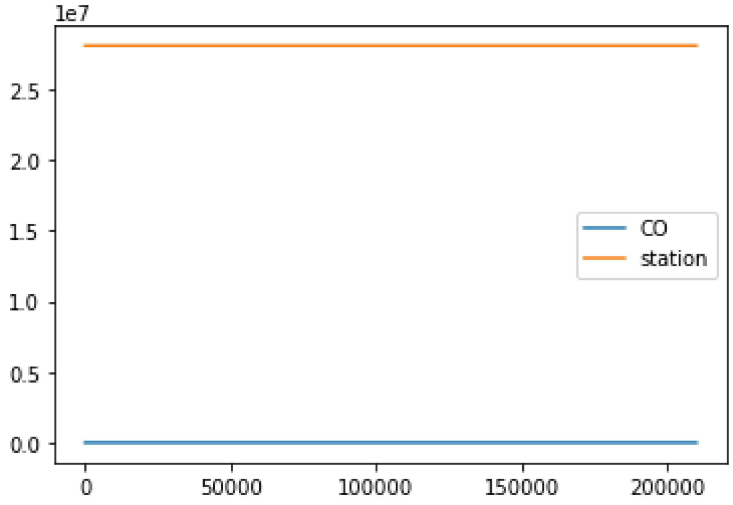
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

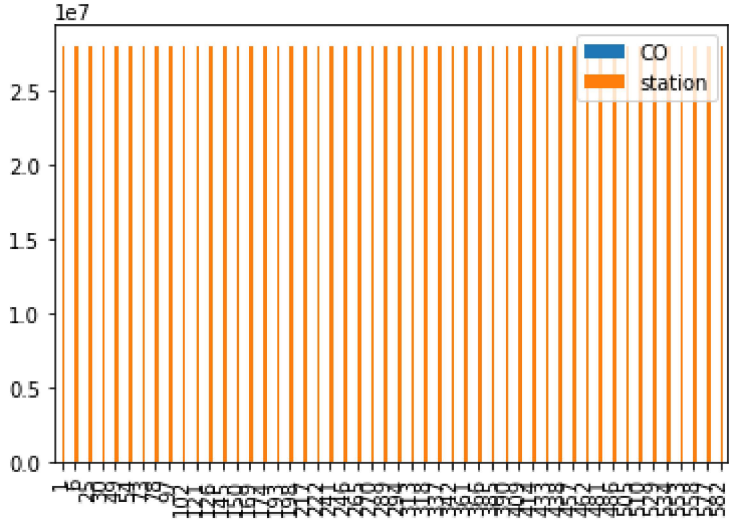


Bar chart

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

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

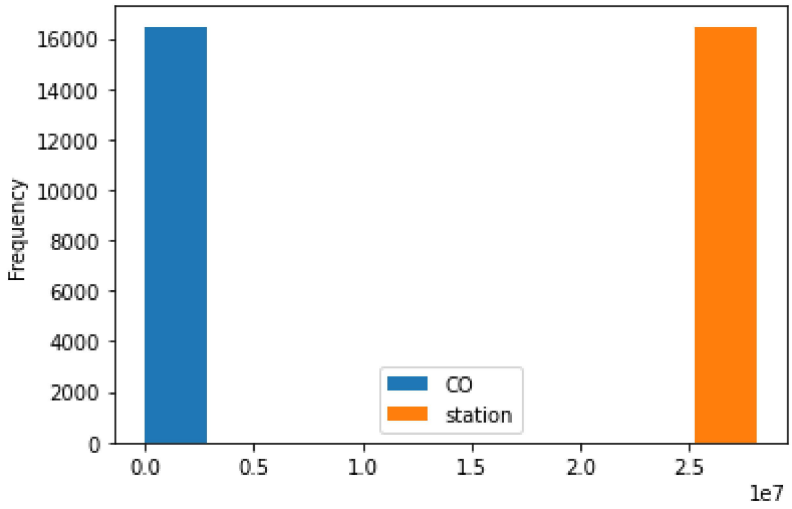
Out[10]: <AxesSubplot:>



Histogram

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

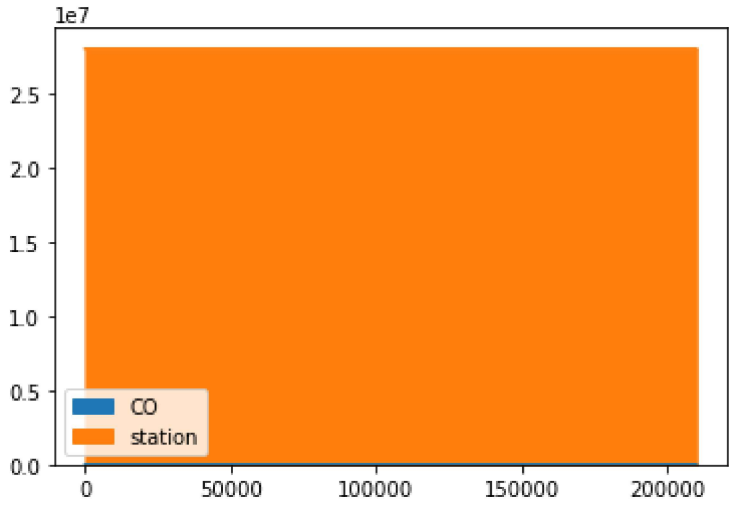
Out[11]:
```
<AxesSubplot:ylabel='Frequency'>
```



Area chart

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

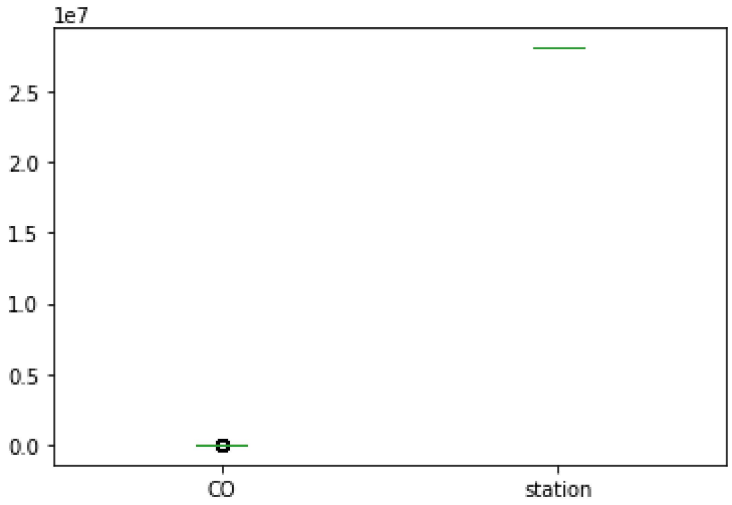
Out[12]:
```
<AxesSubplot:>
```



Box chart

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

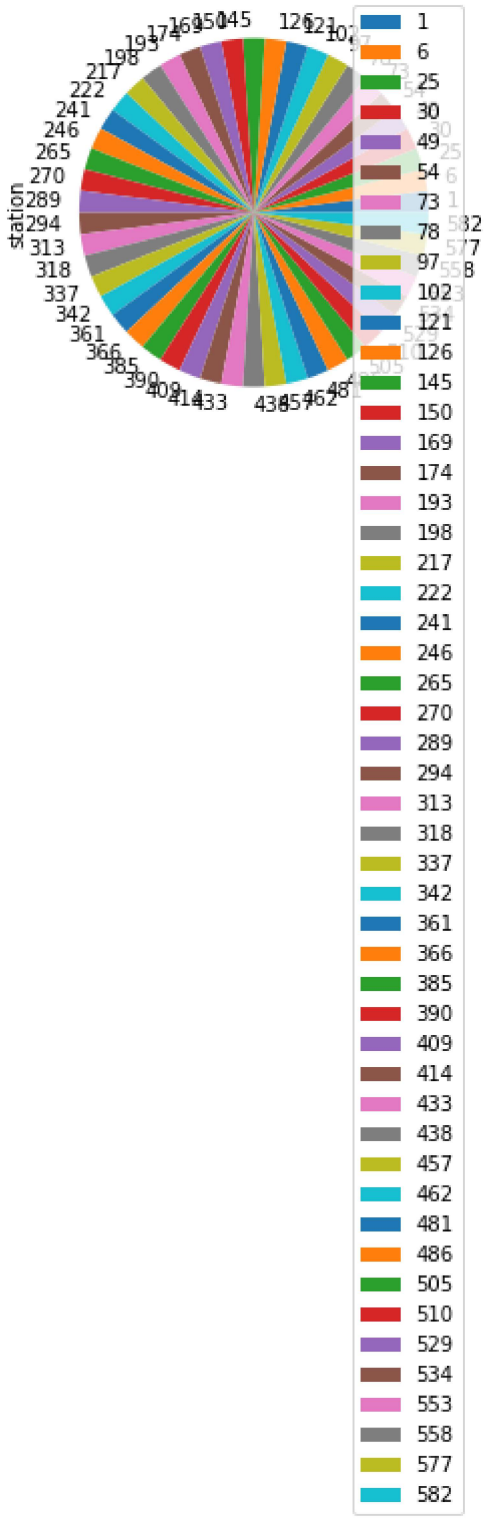
Out[13]: <AxesSubplot:>



Pie chart

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

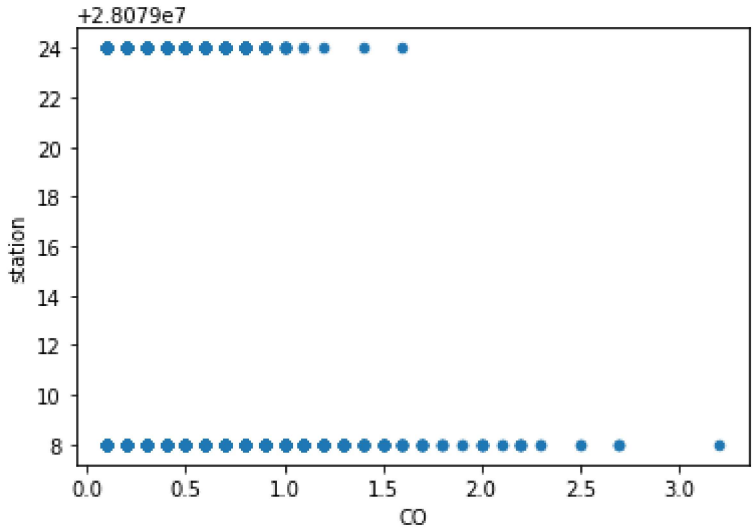
Out[14]: <AxesSubplot:ylabel='station'>



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: 16460 entries, 1 to 209910
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16460 non-null  object 
 1   BEN      16460 non-null  float64
 2   CO       16460 non-null  float64
 3   EBE      16460 non-null  float64
 4   NMHC     16460 non-null  float64
 5   NO       16460 non-null  float64
 6   NO_2     16460 non-null  float64
 7   O_3      16460 non-null  float64
 8   PM10     16460 non-null  float64
 9   PM25     16460 non-null  float64
 10  SO_2     16460 non-null  float64
 11  TCH      16460 non-null  float64
 12  TOL      16460 non-null  float64
 13  station  16460 non-null  int64  
dtypes: float64(12), int64(1), object(1)
```

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

Out[17]:

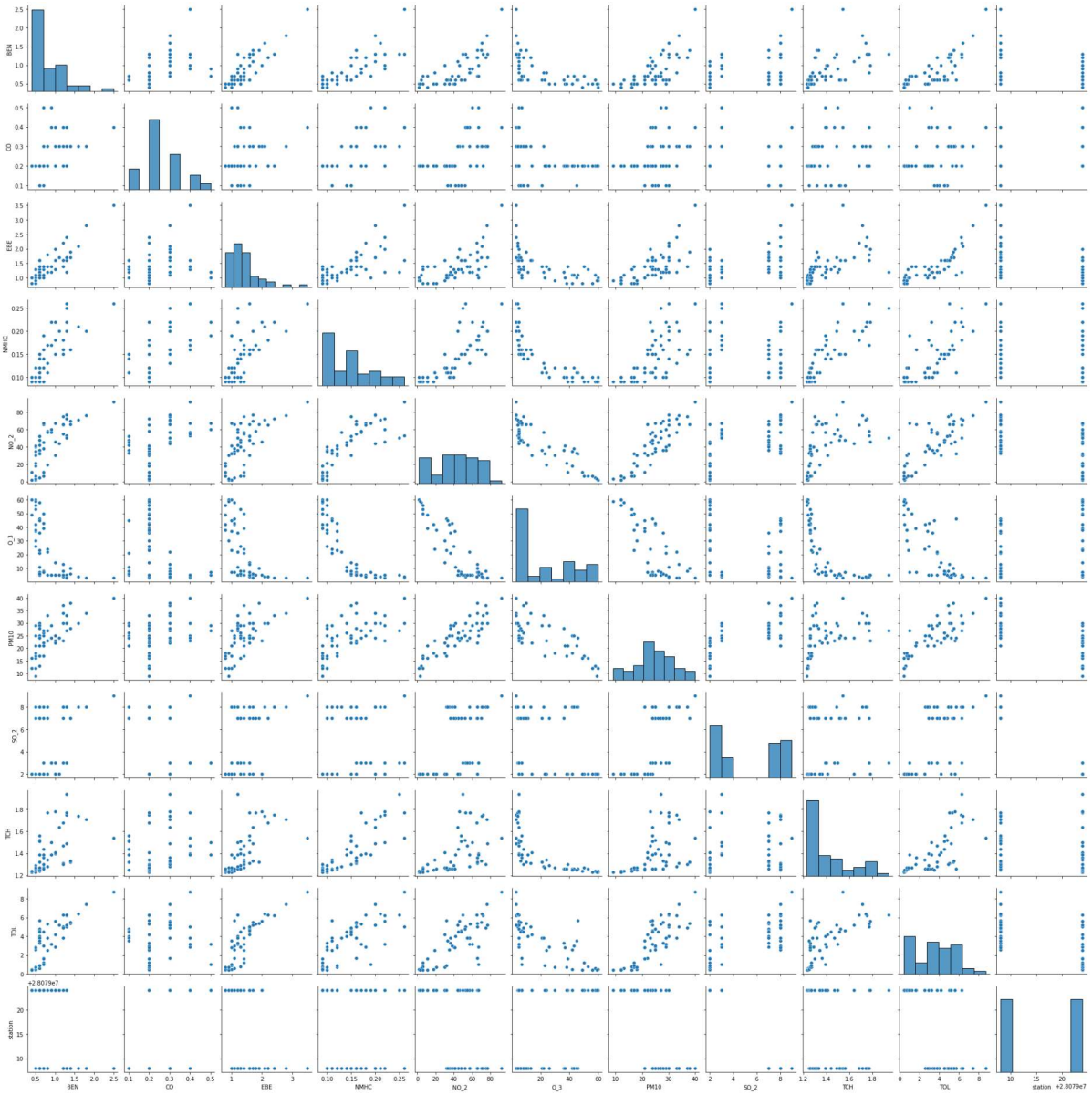
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	164
mean	0.900680	0.277758	1.471871	0.167043	23.671810	44.583961	
std	0.768892	0.206143	1.051004	0.075068	44.362859	31.569185	
min	0.100000	0.100000	0.200000	0.010000	1.000000	1.000000	
25%	0.500000	0.200000	0.800000	0.120000	2.000000	19.000000	
50%	0.700000	0.200000	1.200000	0.160000	7.000000	40.000000	
75%	1.100000	0.300000	1.700000	0.200000	25.000000	63.000000	
max	9.500000	3.200000	12.800000	0.840000	615.000000	289.000000	1

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])
```

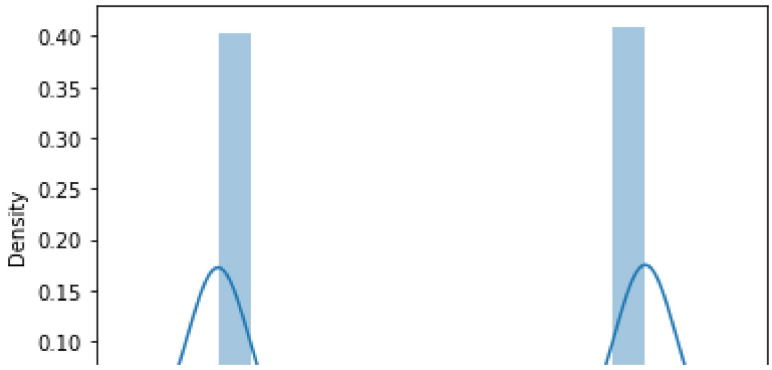
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x23044fc0ac0>
```



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

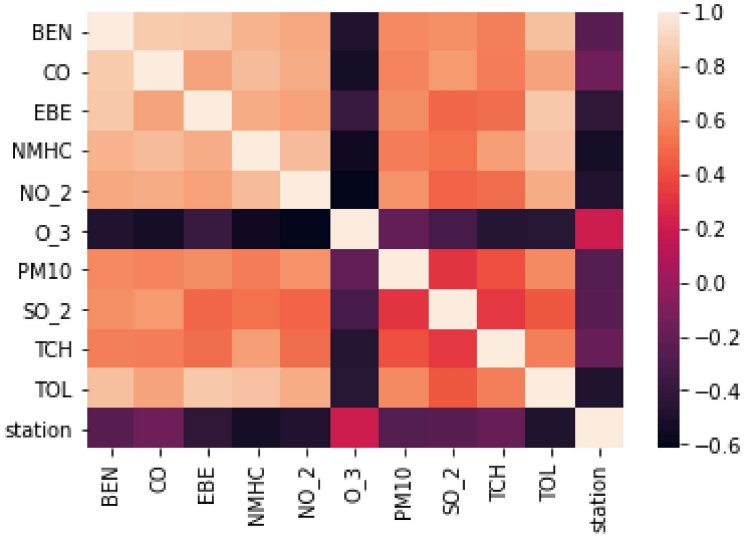
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
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', 'NMHC', 'NO_2', 'O_3',
       'PM10', '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]:
```
28079018.402561016
```

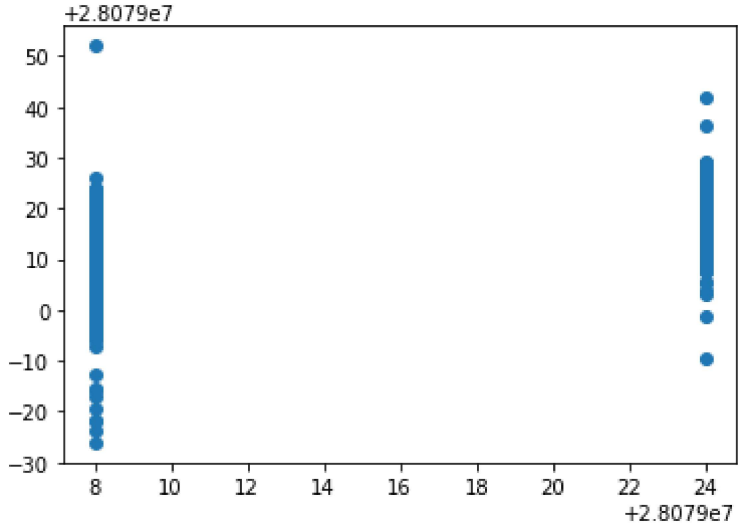
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	3.749850
CO	32.831633
EBE	-1.877801
NMHC	-97.012374
NO_2	-0.084491
O_3	-0.016401
PM10	0.003010
SO_2	-0.494736
TCH	9.653159
TOL	-0.496785

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

Out[27]: <matplotlib.collections.PathCollection at 0x2304e553c10>



ACCURACY

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

Out[28]: 0.6056445626804171

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

Out[29]: 0.628307978226927

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]: 0.5767843500887779

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

Out[33]:
```
0.5933418236531538
```

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]:
```
0.23654073035008205
```

Accuracy(Lasso)

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

Out[36]:
```
0.2283708081104605
```

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([ 0.59974212,  0.        , -0.        , -0.        , -0.13160579,
       -0.05333432,  0.07826025, -0.        ,  0.        , -0.75091039])
```

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

Out[39]:
```
28079024.311707657
```

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

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

Out[41]:
```
0.3115020797281288
```

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)))
```

```
5.86439916833182
44.06299226452486
6.637996103081476
```

Logistic Regression

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

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

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

Out[45]:
```
(16460, 10)
```

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

Out[46]:
```
(16460,)
```

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]]
```

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

```
[28079008]
```

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

Out[52]:
```
array([28079008, 28079024], dtype=int64)
```

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

Out[53]:
```
0.9237545565006076
```

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

Out[54]:
```
0.9999999999999966
```

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

Out[55]:
```
array([[1.00000000e+00, 3.47334507e-15]])
```

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]:
```
0.9337788578371811
```

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

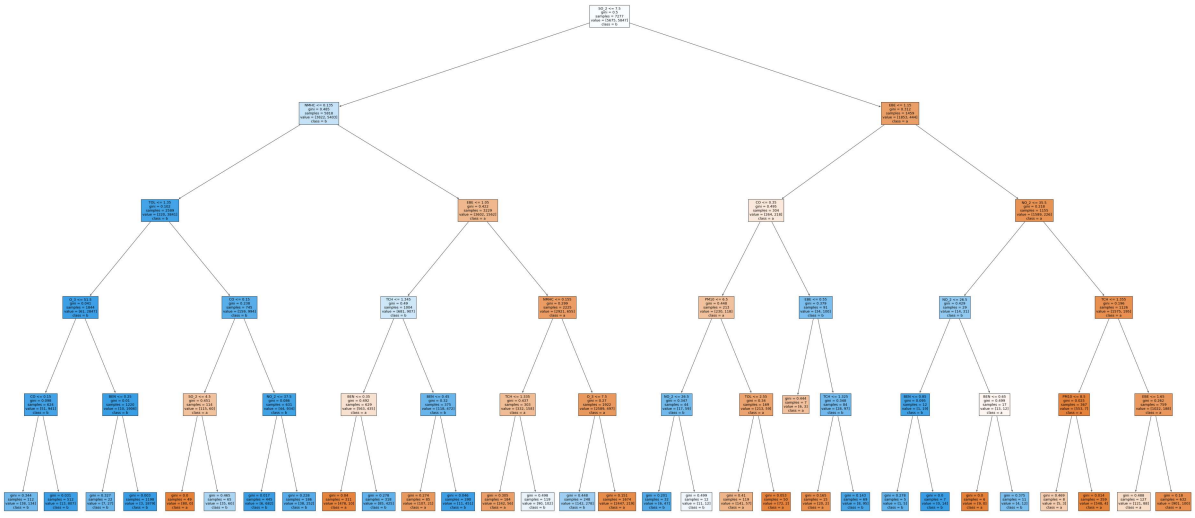
In [62]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b'
```

Out[62]:
```
[Text(2278.5, 1993.2, 'SO_2 <= 7.5\ngini = 0.5\nsamples = 7277\nvalue = [567
5, 5847]\nclass = b'),
 Text(1190.4, 1630.8000000000002, 'NMHC <= 0.135\ngini = 0.485\nsamples = 581
8\nvalue = [3822, 5403]\nclass = b'),
 Text(595.2, 1268.4, 'TOL <= 1.35\ngini = 0.102\nsamples = 2589\nvalue = [22
0, 3841]\nclass = b'),
 Text(297.6, 906.0, 'O_3 <= 51.5\ngini = 0.041\nsamples = 1844\nvalue = [61,
2847]\nclass = b'),
 Text(148.8, 543.5999999999999, 'CO <= 0.15\ngini = 0.098\nsamples = 624\nval
ue = [51, 941]\nclass = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.344\nsamples = 112\nvalue = [38, 13
4]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.031\nsamples = 512\nv
alue = [13, 807]\nclass = b'),
 Text(446.40000000000003, 543.5999999999999, 'BEN <= 0.25\ngini = 0.01\nsampl
es = 1220\nvalue = [10, 1906]\nclass = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.327\nsamples = 22\nvalue = [7, 27]
\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.003\nsamples = 1198\nv
alue = [3, 1879]\nclass = b'),
 Text(892.8000000000001, 906.0, 'CO <= 0.15\ngini = 0.238\nsamples = 745\nval
ue = [159, 994]\nclass = b'),
 Text(744.0, 543.5999999999999, 'SO_2 <= 4.5\ngini = 0.451\nsamples = 114\nva
lue = [115, 60]\nclass = a'),
 Text(669.6, 181.19999999999982, 'gini = 0.0\nsamples = 49\nvalue = [80, 0]\n
class = a'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.465\nsamples = 65\nval
ue = [35, 60]\nclass = b'),
 Text(1041.6000000000001, 543.5999999999999, 'NO_2 <= 37.5\ngini = 0.086\nsam
ples = 631\nvalue = [44, 934]\nclass = b'),
 Text(967.2, 181.19999999999982, 'gini = 0.017\nsamples = 445\nvalue = [6, 68
2]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.228\nsamples = 186\nvalue = [38,
252]\nclass = b'),
 Text(1785.6000000000001, 1268.4, 'EBE <= 1.05\ngini = 0.422\nsamples = 3229
\nvalue = [3602, 1562]\nclass = a'),
 Text(1488.0, 906.0, 'TCH <= 1.345\ngini = 0.49\nsamples = 1004\nvalue = [68
1, 907]\nclass = b'),
 Text(1339.2, 543.5999999999999, 'BEN <= 0.35\ngini = 0.492\nsamples = 629\nv
alue = [563, 435]\nclass = a'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.04\nsamples = 311\nva
lue = [478, 10]\nclass = a'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.278\nsamples = 318\nv
alue = [85, 425]\nclass = b'),
 Text(1636.8000000000002, 543.5999999999999, 'BEN <= 0.45\ngini = 0.32\nsampl
es = 375\nvalue = [118, 472]\nclass = b'),
 Text(1562.4, 181.19999999999982, 'gini = 0.274\nsamples = 85\nvalue = [107,
21]\nclass = a'),
 Text(1711.2, 181.19999999999982, 'gini = 0.046\nsamples = 290\nvalue = [11,
451]\nclass = b'),
 Text(2083.2000000000003, 906.0, 'NMHC <= 0.155\ngini = 0.299\nsamples = 2225
\nvalue = [2921, 655]\nclass = a'),
 Text(1934.4, 543.5999999999999, 'TCH <= 1.335\ngini = 0.437\nsamples = 303\n
value = [332, 158]\nclass = a'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.305\nsamples = 184\nv
alue = [242, 56]\nclass = a'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.498\nsamples = 119\nv
```

```
alue = [90, 102]\nclass = b'),
 Text(2232.0, 543.5999999999999, 'O_3 <= 7.5\ngini = 0.27\nsamples = 1922\nva
lue = [2589, 497]\nclass = a'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.448\nsamples = 248\nv
alue = [142, 278]\nclass = b'),
 Text(2306.4, 181.19999999999982, 'gini = 0.151\nsamples = 1674\nvalue = [244
7, 219]\nclass = a'),
 Text(3366.6000000000004, 1630.8000000000002, 'EBE <= 1.15\ngini = 0.312\nsam
ples = 1459\nvalue = [1853, 444]\nclass = a'),
 Text(2864.4, 1268.4, 'CO <= 0.35\ngini = 0.495\nsamples = 304\nvalue = [264,
218]\nclass = a'),
 Text(2678.4, 906.0, 'PM10 <= 6.5\ngini = 0.448\nsamples = 213\nvalue = [230,
118]\nclass = a'),
 Text(2529.6000000000004, 543.5999999999999, 'NO_2 <= 26.5\ngini = 0.347\nsam
ples = 44\nvalue = [17, 59]\nclass = b'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.201\nsamples = 32\nva
lue = [6, 47]\nclass = b'),
 Text(2604.0, 181.19999999999982, 'gini = 0.499\nsamples = 12\nvalue = [11, 1
2]\nclass = b'),
 Text(2827.2000000000003, 543.5999999999999, 'TOL <= 2.55\ngini = 0.34\nsampl
es = 169\nvalue = [213, 59]\nclass = a'),
 Text(2752.8, 181.19999999999982, 'gini = 0.41\nsamples = 119\nvalue = [141,
57]\nclass = a'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.053\nsamples = 50\nva
lue = [72, 2]\nclass = a'),
 Text(3050.4, 906.0, 'EBE <= 0.55\ngini = 0.379\nsamples = 91\nvalue = [34, 1
00]\nclass = b'),
 Text(2976.0, 543.5999999999999, 'gini = 0.444\nsamples = 7\nvalue = [6, 3]\n
class = a'),
 Text(3124.8, 543.5999999999999, 'TCH <= 1.325\ngini = 0.348\nsamples = 84\nv
alue = [28, 97]\nclass = b'),
 Text(3050.4, 181.19999999999982, 'gini = 0.165\nsamples = 15\nvalue = [20,
2]\nclass = a'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.143\nsamples = 69\nva
lue = [8, 95]\nclass = b'),
 Text(3868.8, 1268.4, 'NO_2 <= 35.5\ngini = 0.218\nsamples = 1155\nvalue = [1
589, 226]\nclass = a'),
 Text(3571.2000000000003, 906.0, 'NO_2 <= 26.5\ngini = 0.429\nsamples = 29\nv
alue = [14, 31]\nclass = b'),
 Text(3422.4, 543.5999999999999, 'BEN <= 0.85\ngini = 0.095\nsamples = 12\nva
lue = [1, 19]\nclass = b'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.278\nsamples = 5\nval
ue = [1, 5]\nclass = b'),
 Text(3496.8, 181.19999999999982, 'gini = 0.0\nsamples = 7\nvalue = [0, 14]\n
class = b'),
 Text(3720.0000000000005, 543.5999999999999, 'BEN <= 0.65\ngini = 0.499\nsamp
les = 17\nvalue = [13, 12]\nclass = a'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.0\nsamples = 6\nvalue
= [9, 0]\nclass = a'),
 Text(3794.4, 181.19999999999982, 'gini = 0.375\nsamples = 11\nvalue = [4, 1
2]\nclass = b'),
 Text(4166.400000000001, 906.0, 'TCH <= 1.355\ngini = 0.196\nsamples = 1126\n
value = [1575, 195]\nclass = a'),
 Text(4017.6000000000004, 543.5999999999999, 'PM10 <= 8.5\ngini = 0.025\nsamp
les = 367\nvalue = [553, 7]\nclass = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.469\nsamples = 8\nval
ue = [5, 3]\nclass = a'),
```

```
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.014\nsamples = 359\nv
alue = [548, 4]\nclass = a'),
 Text(4315.200000000001, 543.5999999999999, 'EBE <= 1.65\ngini = 0.262\nsampl
es = 759\nvalue = [1022, 188]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.488\nsamples = 127\nvalue = [121,
88]\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.18\nsamples = 632\nvalue = [901,
100]\nclass = a')]
```



Conclusion

Accuracy

In [63]:
```
lr.score(x_train,y_train)
```

Out[63]: 0.628307978226927

In [64]:
```
rr.score(x_train,y_train)
```

Out[64]: 0.5933418236531538

In [65]:
```
la.score(x_train,y_train)
```

Out[65]: 0.23654073035008205

In [66]:
```
en.score(x_test,y_test)
```

Out[66]: 0.3115020797281288

In [67]:
```
logr.score(fs,target_vector)
```

Out[67]: 0.9237545565006076

In [68]:
```
grid_search.best_score_
```

Out[68]:
```
0.9337788578371811
```

Random Forest is suitable for this dataset

In [ ]:
```

```