

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_2017.csv")
df
```

Out[2]:

	date	BEN	CH4	CO	EBe	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCl
0	2017-06-01 01:00:00	NaN	NaN	0.3	NaN	NaN	4.0	38.0	NaN	NaN	NaN	NaN	5.0	NaN
1	2017-06-01 01:00:00	0.6	NaN	0.3	0.4	0.08	3.0	39.0	NaN	71.0	22.0	9.0	7.0	1.
2	2017-06-01 01:00:00	0.2	NaN	NaN	0.1	NaN	1.0	14.0	NaN	NaN	NaN	NaN	NaN	NaN
3	2017-06-01 01:00:00	NaN	NaN	0.2	NaN	NaN	1.0	9.0	NaN	91.0	NaN	NaN	NaN	NaN
4	2017-06-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	19.0	NaN	69.0	NaN	NaN	2.0	NaN
...
210115	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	27.0	NaN	65.0	NaN	NaN	NaN	NaN
210116	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	14.0	NaN	NaN	73.0	NaN	7.0	NaN
210117	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	4.0	NaN	83.0	NaN	NaN	NaN	NaN
210118	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	11.0	NaN	78.0	NaN	NaN	NaN	NaN
210119	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	14.0	NaN	77.0	60.0	NaN	NaN	NaN

210120 rows × 16 columns

Data Cleaning and Data Preprocessing

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

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

```
Out[4]: Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4127 entries, 87457 to 158286
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      4127 non-null   object 
 1   BEN        4127 non-null   float64
 2   CH4        4127 non-null   float64
 3   CO          4127 non-null   float64
 4   EBE        4127 non-null   float64
 5   NMHC       4127 non-null   float64
 6   NO          4127 non-null   float64
 7   NO_2        4127 non-null   float64
 8   NOx         4127 non-null   float64
 9   O_3          4127 non-null   float64
 10  PM10        4127 non-null   float64
 11  PM25        4127 non-null   float64
 12  SO_2        4127 non-null   float64
 13  TCH          4127 non-null   float64
 14  TOL          4127 non-null   float64
 15  station     4127 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 548.1+ KB
```

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

Out[6]:

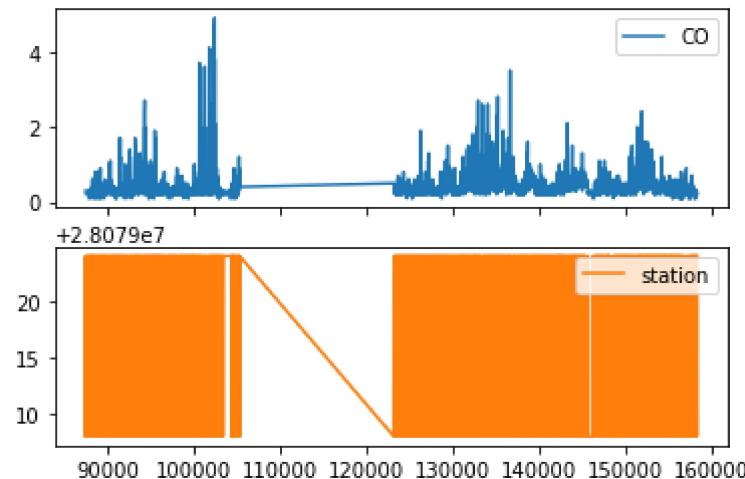
	CO	station
87457	0.3	28079008
87462	0.2	28079024
87481	0.2	28079008
87486	0.2	28079024
87505	0.2	28079008
...
158238	0.2	28079024
158257	0.3	28079008
158262	0.2	28079024
158281	0.2	28079008
158286	0.2	28079024

4127 rows × 2 columns

Line chart

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

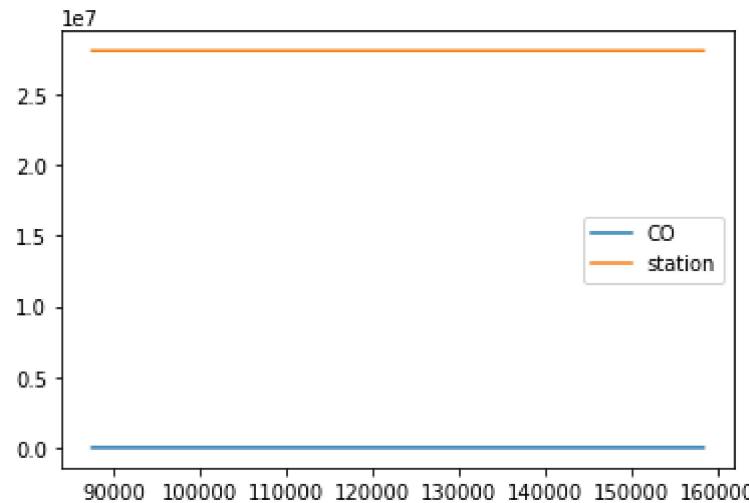
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

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

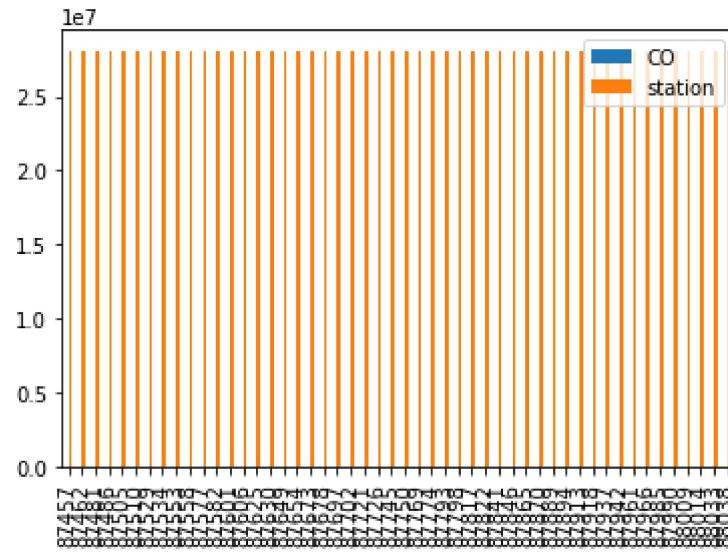


Bar chart

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

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

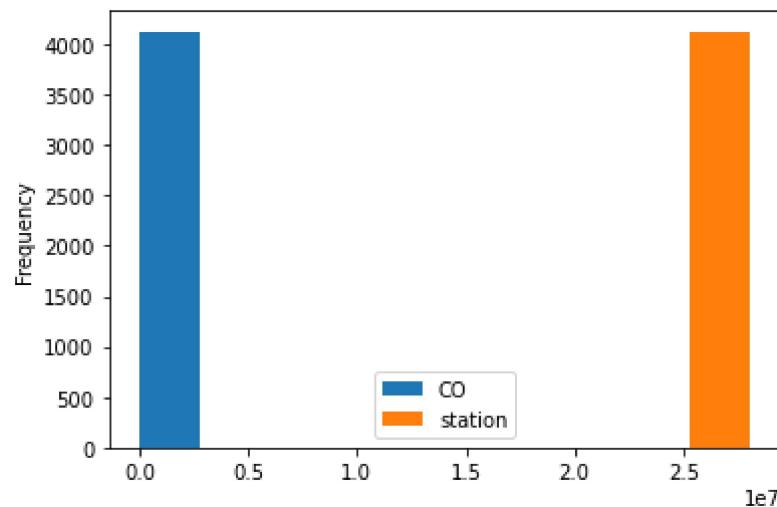
```
Out[10]: <AxesSubplot:>
```



Histogram

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

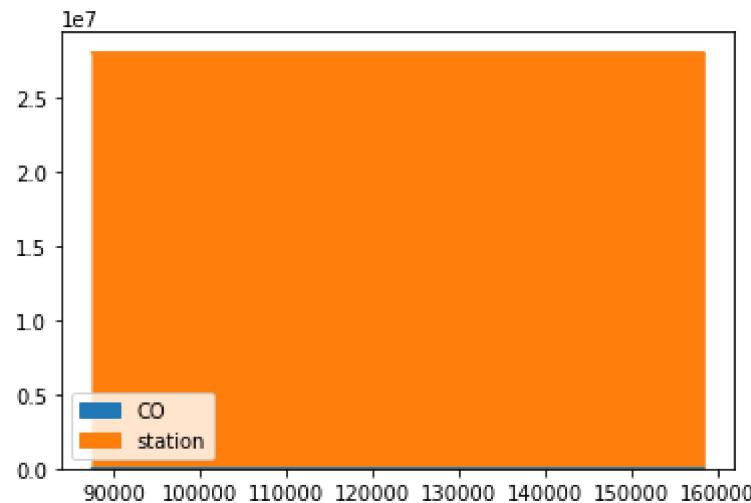
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

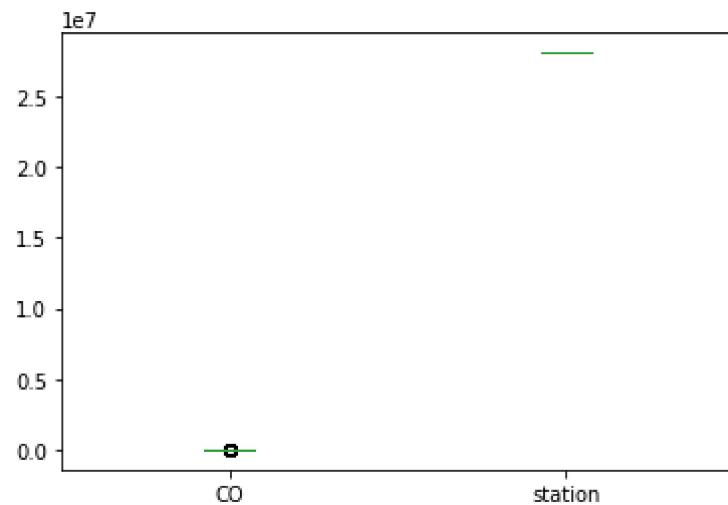
```
Out[12]: <AxesSubplot:>
```



Box chart

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

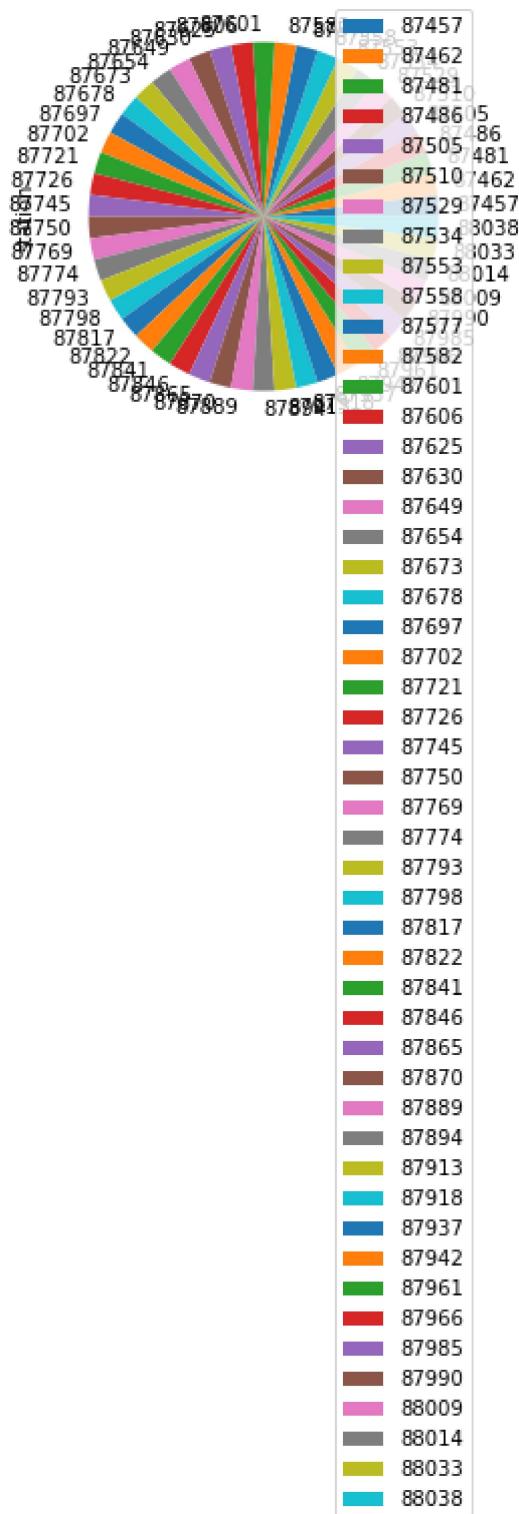
```
Out[13]: <AxesSubplot:>
```



Pie chart

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

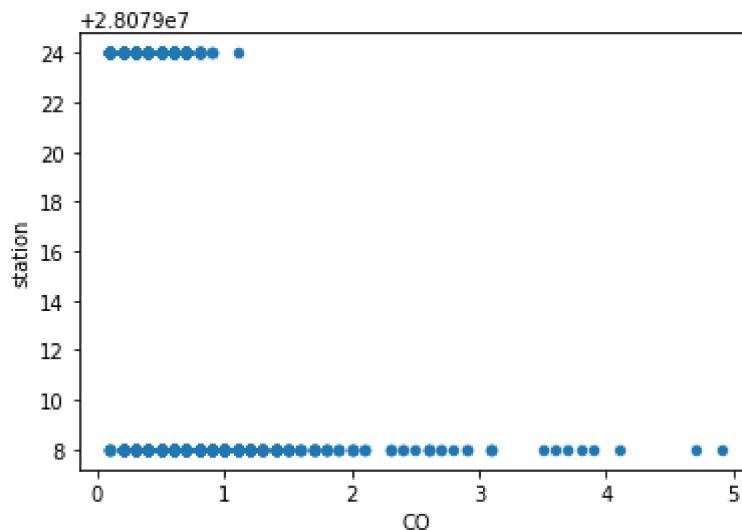
```
Out[14]: <AxesSubplot:ylabel='station'>
```



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: 4127 entries, 87457 to 158286
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      4127 non-null   object 
 1   BEN       4127 non-null   float64
 2   CH4       4127 non-null   float64
 3   CO        4127 non-null   float64
 4   EBE       4127 non-null   float64
 5   NMHC      4127 non-null   float64
 6   NO        4127 non-null   float64
 7   NO_2      4127 non-null   float64
 8   NOx       4127 non-null   float64
 9   O_3        4127 non-null   float64
 10  PM10      4127 non-null   float64
 11  PM25      4127 non-null   float64
 12  SO_2       4127 non-null   float64
 13  TCH       4127 non-null   float64
 14  TOI       4127 non-null   float64
```

In [17]: df.describe()

Out[17]:

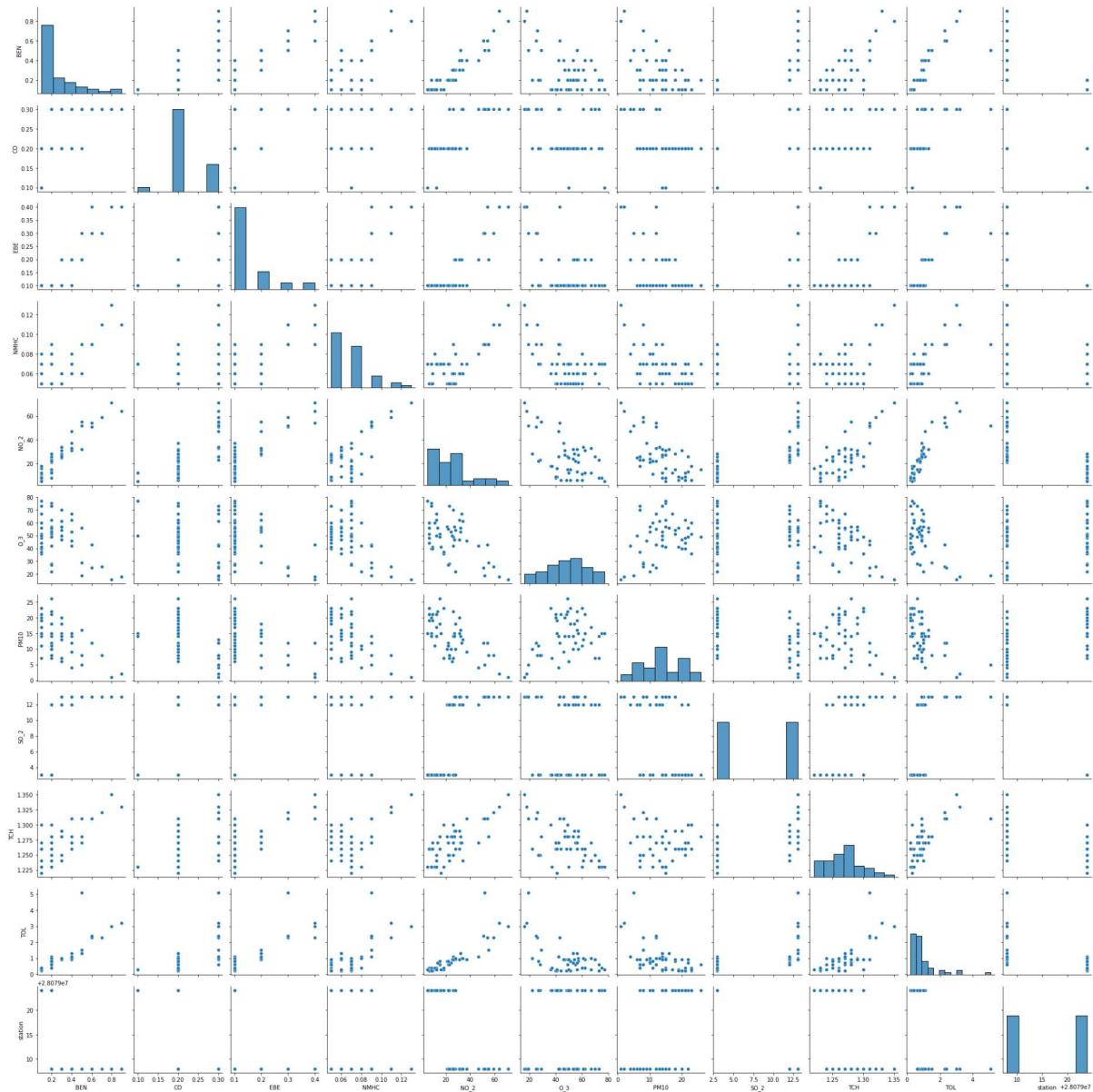
	BEN	CH4	CO	EBC	NMHC	NO	NO_
count	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000
mean	0.919918	1.323732	0.417858	0.578168	0.097269	41.785316	58.06905
std	1.123078	0.215742	0.342871	0.962000	0.094035	71.118499	38.97411
min	0.100000	1.100000	0.100000	0.100000	0.000000	1.000000	1.00000
25%	0.300000	1.180000	0.200000	0.100000	0.050000	3.000000	30.00000
50%	0.600000	1.270000	0.300000	0.300000	0.080000	16.000000	54.00000
75%	1.100000	1.400000	0.500000	0.700000	0.110000	50.000000	78.00000
max	19.600000	3.630000	4.900000	16.700001	1.420000	879.000000	349.00000

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

EDA AND VISUALIZATION

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

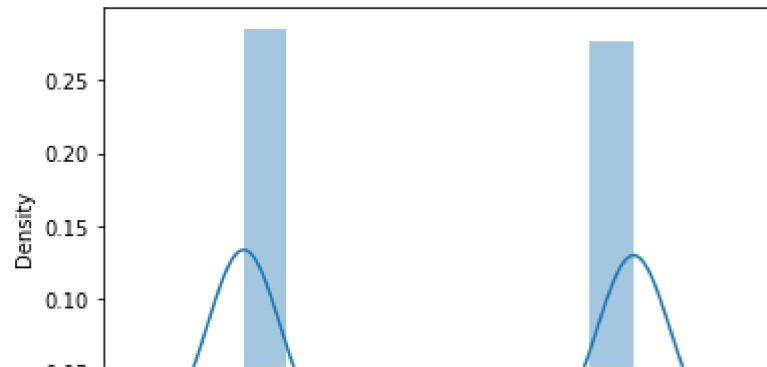
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x1ace2a29550>
```



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

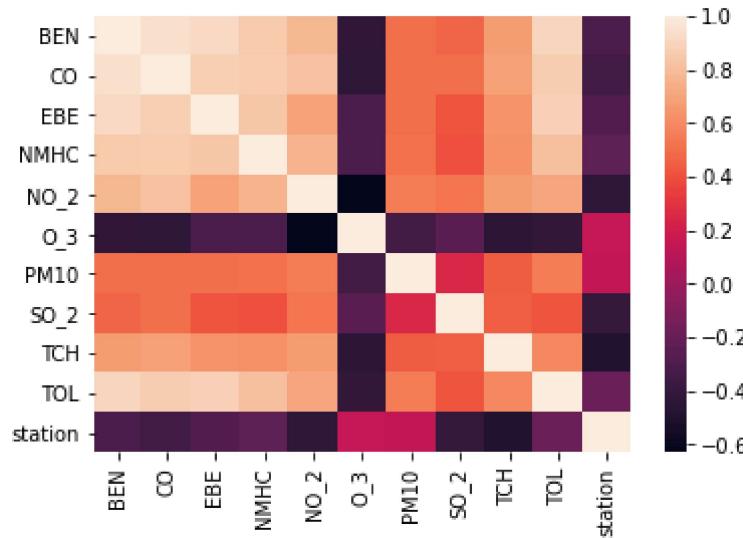
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

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]: 28079039.50936169
```

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

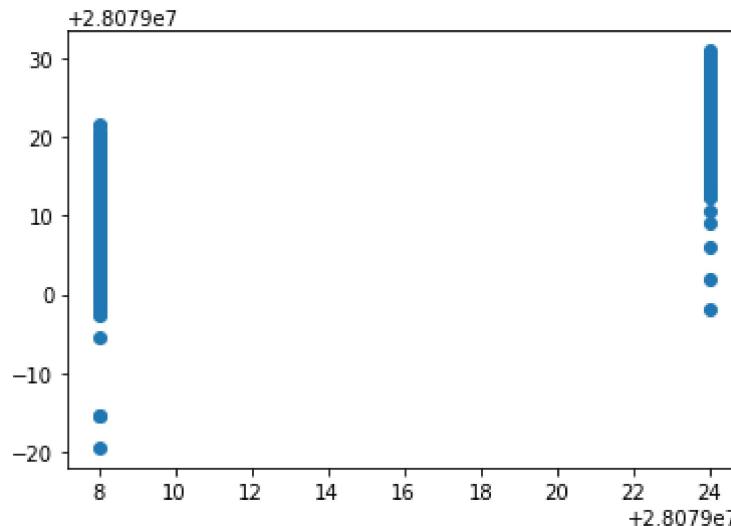
```
Out[26]:
```

Co-efficient

	Co-efficient
BEN	1.578112
CO	-1.825124
EBE	-3.396231
NMHC	38.870969
NO_2	-0.165395
O_3	-0.080338
PM10	0.340068
SO_2	-0.262697
TCH	-13.766528
TOL	0.217435

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

```
Out[27]: <matplotlib.collections.PathCollection at 0x1aceb0408b0>
```



ACCURACY

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

```
Out[28]: 0.5746958627679744
```

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

```
Out[29]: 0.6161819764222258
```

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]: 0.566821239323581
```

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

```
Out[33]: 0.5970683775928844
```

Accuracy(Lasso)

```
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]: 0.4015846058586593
```

ElasticNet

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

```
Out[36]: 0.38881058915846856
```

```
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.          , -0.          , -0.          ,  0.          , -0.16218276,  
   -0.05488369,  0.31659714, -0.37497712, -0.          ,  0.12036379])
```

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

```
Out[39]: 28079023.012416217
```

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

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

```
Out[41]: 0.442346717155201
```

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)))
```

```
5.1484779330922095  
35.677511456198836  
5.973065499071548
```

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]: (4127, 10)
```

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

```
Out[46]: (4127,)
```

```
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]: 0.9437848315968016
```

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

```
Out[54]: 0.9999999999725541
```

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

```
Out[55]: array([[1.0000000e+00, 2.74458959e-11]])
```

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="acc  
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]: 0.9702216066481995
```

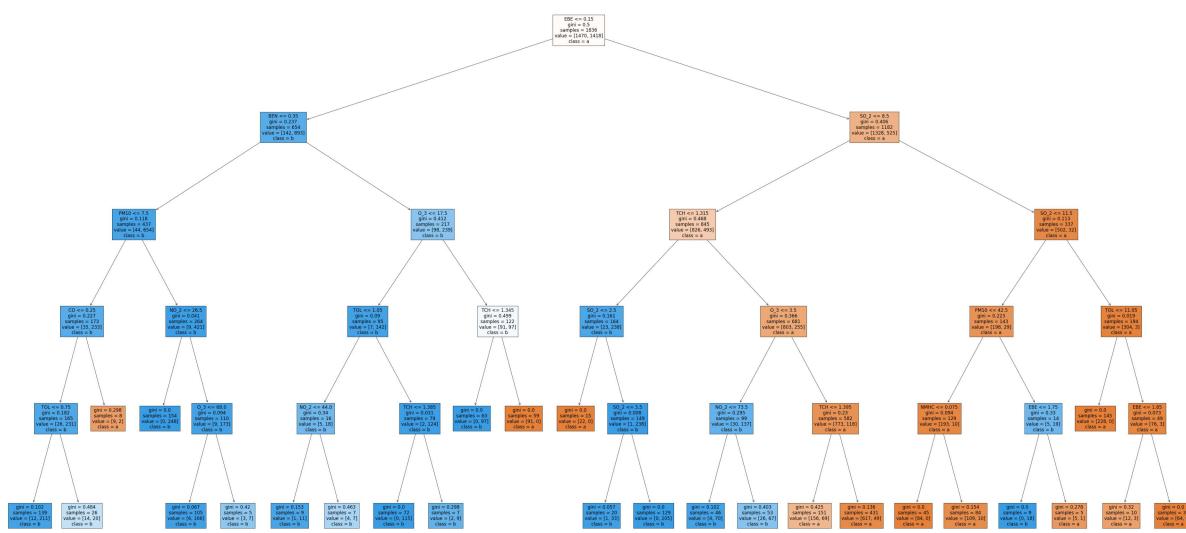
```
In [61]: rfc_best=grid_search.best_estimator_
```

```
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(2147.086956521739, 1993.2, 'EBE <= 0.15\nngini = 0.5\nsamples = 1836\nvalue = [1470, 1418]\nnclass = a'),  
Text(1043.2173913043478, 1630.8000000000002, 'BEN <= 0.35\nngini = 0.237\nsamples = 654\nvalue = [142, 893]\nnclass = b'),  
Text(485.2173913043478, 1268.4, 'PM10 <= 7.5\nngini = 0.118\nsamples = 437\nvalue = [44, 654]\nnclass = b'),  
Text(291.1304347826087, 906.0, 'CO <= 0.25\nngini = 0.227\nsamples = 173\nvalue = [35, 233]\nnclass = b'),  
Text(194.08695652173913, 543.5999999999999, 'TOL <= 0.75\nngini = 0.182\nsamples = 165\nvalue = [26, 231]\nnclass = b'),  
Text(97.04347826086956, 181.1999999999982, 'gini = 0.102\nsamples = 139\nvalue = [12, 211]\nnclass = b'),  
Text(291.1304347826087, 181.1999999999982, 'gini = 0.484\nsamples = 26\nvalue = [14, 20]\nnclass = b'),  
Text(388.17391304347825, 543.5999999999999, 'gini = 0.298\nsamples = 8\nvalue = [9, 2]\nnclass = a'),  
Text(679.304347826087, 906.0, 'NO_2 <= 26.5\nngini = 0.041\nsamples = 264\nvalue = [9, 421]\nnclass = b'),  
Text(582.2608695652174, 543.5999999999999, 'gini = 0.0\nsamples = 154\nvalue = [0, 248]\nnclass = b'),  
Text(776.3478260869565, 543.5999999999999, 'O_3 <= 68.0\nngini = 0.094\nsamples = 110\nvalue = [9, 173]\nnclass = b'),  
Text(679.304347826087, 181.1999999999982, 'gini = 0.067\nsamples = 105\nvalue = [6, 166]\nnclass = b'),  
Text(873.391304347826, 181.1999999999982, 'gini = 0.42\nsamples = 5\nvalue = [3, 7]\nnclass = b'),  
Text(1601.2173913043478, 1268.4, 'O_3 <= 17.5\nngini = 0.412\nsamples = 217\nvalue = [98, 239]\nnclass = b'),  
Text(1358.608695652174, 906.0, 'TOL <= 1.05\nngini = 0.09\nsamples = 95\nvalue = [7, 142]\nnclass = b'),  
Text(1164.5217391304348, 543.5999999999999, 'NO_2 <= 44.0\nngini = 0.34\nsamples = 16\nvalue = [5, 18]\nnclass = b'),  
Text(1067.4782608695652, 181.1999999999982, 'gini = 0.153\nsamples = 9\nvalue = [1, 11]\nnclass = b'),  
Text(1261.5652173913043, 181.1999999999982, 'gini = 0.463\nsamples = 7\nvalue = [4, 7]\nnclass = b'),  
Text(1552.695652173913, 543.5999999999999, 'TCH <= 1.385\nngini = 0.031\nsamples = 79\nvalue = [2, 124]\nnclass = b'),  
Text(1455.6521739130435, 181.1999999999982, 'gini = 0.0\nsamples = 72\nvalue = [0, 115]\nnclass = b'),  
Text(1649.7391304347825, 181.1999999999982, 'gini = 0.298\nsamples = 7\nvalue = [2, 9]\nnclass = b'),  
Text(1843.8260869565217, 906.0, 'TCH <= 1.345\nngini = 0.499\nsamples = 122\nvalue = [91, 97]\nnclass = b'),  
Text(1746.782608695652, 543.5999999999999, 'gini = 0.0\nsamples = 63\nvalue = [0, 97]\nnclass = b'),  
Text(1940.8695652173913, 543.5999999999999, 'gini = 0.0\nsamples = 59\nvalue = [91, 0]\nnclass = a'),  
Text(3250.9565217391305, 1630.8000000000002, 'SO_2 <= 8.5\nngini = 0.406\nsamples = 1182\nvalue = [1328, 525]\nnclass = a'),  
Text(2571.6521739130435, 1268.4, 'TCH <= 1.315\nngini = 0.468\nsamples = 845\nvalue = [826, 493]\nnclass = a'),  
Text(2232.0, 906.0, 'SO_2 <= 2.5\nngini = 0.161\nsamples = 164\nvalue = [23, 238]\nnclass = b'),  
Text(2134.9565217391305, 543.5999999999999, 'gini = 0.0\nsamples = 15\nvalue = [22, 0]\nnclass = a'),  
Text(2329.0434782608695, 543.5999999999999, 'SO_2 <= 3.5\nngini = 0.008\nsamples = 118\nvalue = [1470, 1418]\nnclass = a')
```

```
les = 149\nvalue = [1, 238]\nclass = b'),  
    Text(2232.0, 181.1999999999982, 'gini = 0.057\nsamples = 20\nvalue = [1, 3  
3]\nclass = b'),  
    Text(2426.086956521739, 181.1999999999982, 'gini = 0.0\nsamples = 129\nvalu  
e = [0, 205]\nclass = b'),  
    Text(2911.304347826087, 906.0, 'O_3 <= 3.5\ngini = 0.366\nsamples = 681\nval  
ue = [803, 255]\nclass = a'),  
    Text(2717.217391304348, 543.5999999999999, 'NO_2 <= 73.5\ngini = 0.295\nnsamp  
les = 99\nvalue = [30, 137]\nclass = b'),  
    Text(2620.173913043478, 181.1999999999982, 'gini = 0.102\nsamples = 46\nval  
ue = [4, 70]\nclass = b'),  
    Text(2814.2608695652175, 181.1999999999982, 'gini = 0.403\nsamples = 53\nva  
lue = [26, 67]\nclass = b'),  
    Text(3105.391304347826, 543.5999999999999, 'TCH <= 1.395\ngini = 0.23\nnsampl  
es = 582\nvalue = [773, 118]\nclass = a'),  
    Text(3008.3478260869565, 181.1999999999982, 'gini = 0.425\nsamples = 151\nva  
lue = [156, 69]\nclass = a'),  
    Text(3202.4347826086955, 181.1999999999982, 'gini = 0.136\nsamples = 431\nva  
lue = [617, 49]\nclass = a'),  
    Text(3930.2608695652175, 1268.4, 'SO_2 <= 11.5\ngini = 0.113\nsamples = 337  
\nvalue = [502, 32]\nclass = a'),  
    Text(3687.6521739130435, 906.0, 'PM10 <= 42.5\ngini = 0.223\nsamples = 143\nv  
alue = [198, 29]\nclass = a'),  
    Text(3493.565217391304, 543.5999999999999, 'NMHC <= 0.075\ngini = 0.094\nnsam  
ples = 129\nvalue = [193, 10]\nclass = a'),  
    Text(3396.5217391304345, 181.1999999999982, 'gini = 0.0\nsamples = 45\nval  
ue = [84, 0]\nclass = a'),  
    Text(3590.608695652174, 181.1999999999982, 'gini = 0.154\nsamples = 84\nval  
ue = [109, 10]\nclass = a'),  
    Text(3881.7391304347825, 543.5999999999999, 'EBE <= 1.75\ngini = 0.33\nnsampl  
es = 14\nvalue = [5, 19]\nclass = b'),  
    Text(3784.695652173913, 181.1999999999982, 'gini = 0.0\nsamples = 9\nvalue  
= [0, 18]\nclass = b'),  
    Text(3978.782608695652, 181.1999999999982, 'gini = 0.278\nsamples = 5\nvalu  
e = [5, 1]\nclass = a'),  
    Text(4172.869565217391, 906.0, 'TOL <= 11.05\ngini = 0.019\nsamples = 194\nv  
alue = [304, 3]\nclass = a'),  
    Text(4075.8260869565215, 543.5999999999999, 'gini = 0.0\nsamples = 145\nval  
ue = [228, 0]\nclass = a'),  
    Text(4269.913043478261, 543.5999999999999, 'EBE <= 1.85\ngini = 0.073\nnsampl  
es = 49\nvalue = [76, 3]\nclass = a'),  
    Text(4172.869565217391, 181.1999999999982, 'gini = 0.32\nsamples = 10\nvalu  
e = [12, 3]\nclass = a'),  
    Text(4366.95652173913, 181.1999999999982, 'gini = 0.0\nsamples = 39\nvalu  
e = [64, 0]\nclass = a')]
```



Conclusion

Accuracy

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

```
Out[63]: 0.6161819764222258
```

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

```
Out[64]: 0.5970683775928844
```

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

```
Out[65]: 0.4015846058586593
```

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

```
Out[66]: 0.442346717155201
```

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

```
Out[67]: 0.9437848315968016
```

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

```
Out[68]: 0.9702216066481995
```

Random Forest is suitable for this dataset

