

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
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000
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
...
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
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

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]:

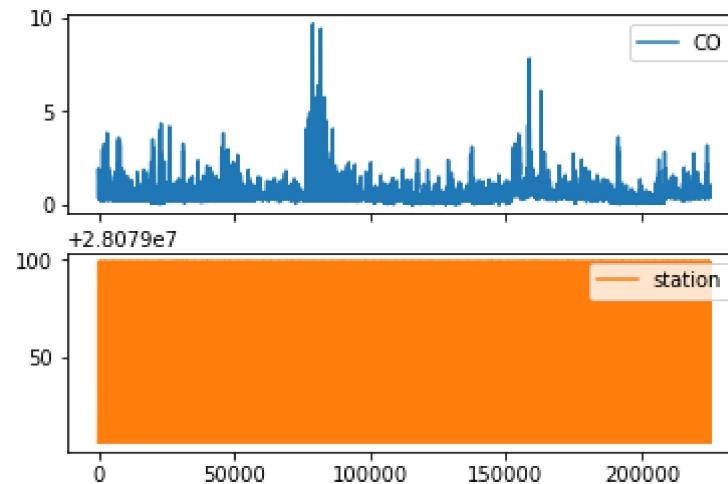
	CO	station
4	1.86	28079006
21	0.31	28079024
25	1.42	28079099
30	1.89	28079006
47	0.30	28079024
...
225073	0.47	28079006
225094	0.45	28079099
225098	0.41	28079006
225115	0.45	28079024
225119	0.40	28079099

25443 rows × 2 columns

Line chart

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

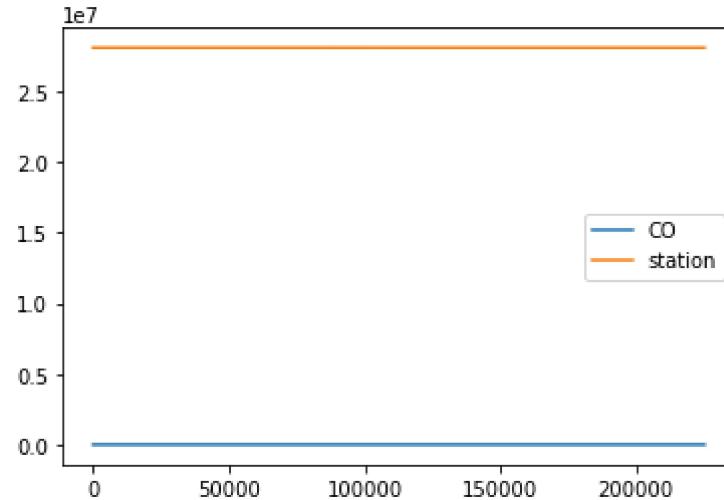
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

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

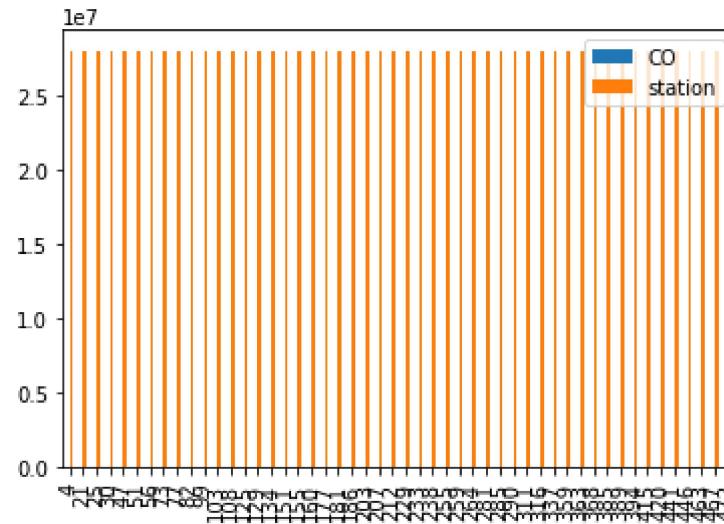


Bar chart

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

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

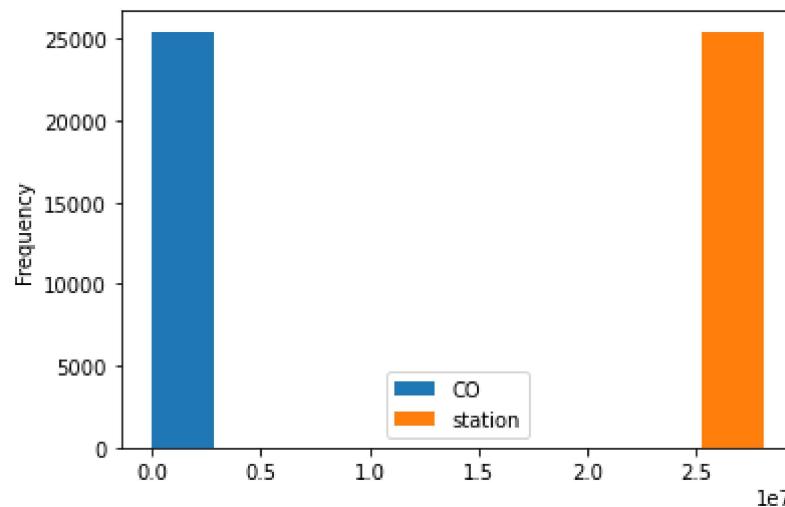
```
Out[10]: <AxesSubplot:>
```



Histogram

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

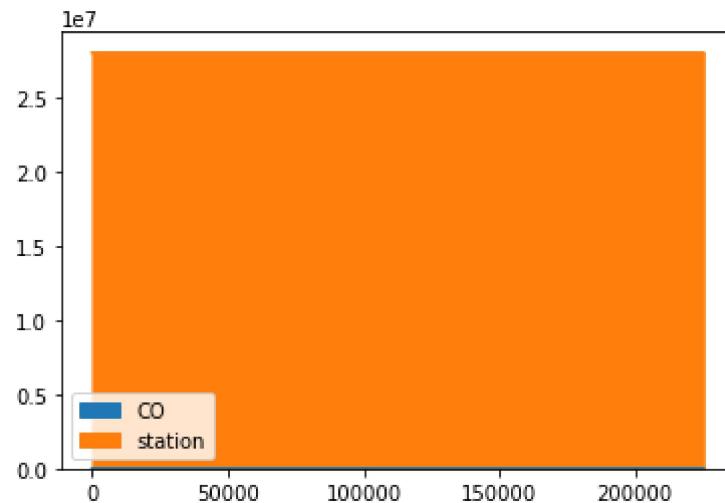
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

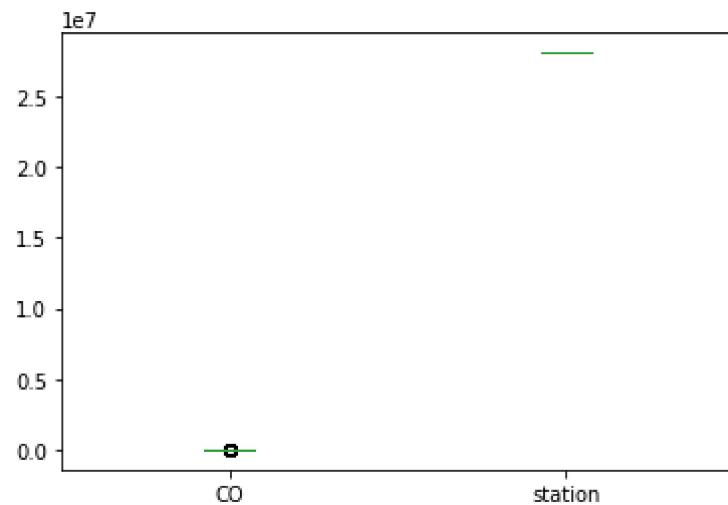
```
Out[12]: <AxesSubplot:>
```



Box chart

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

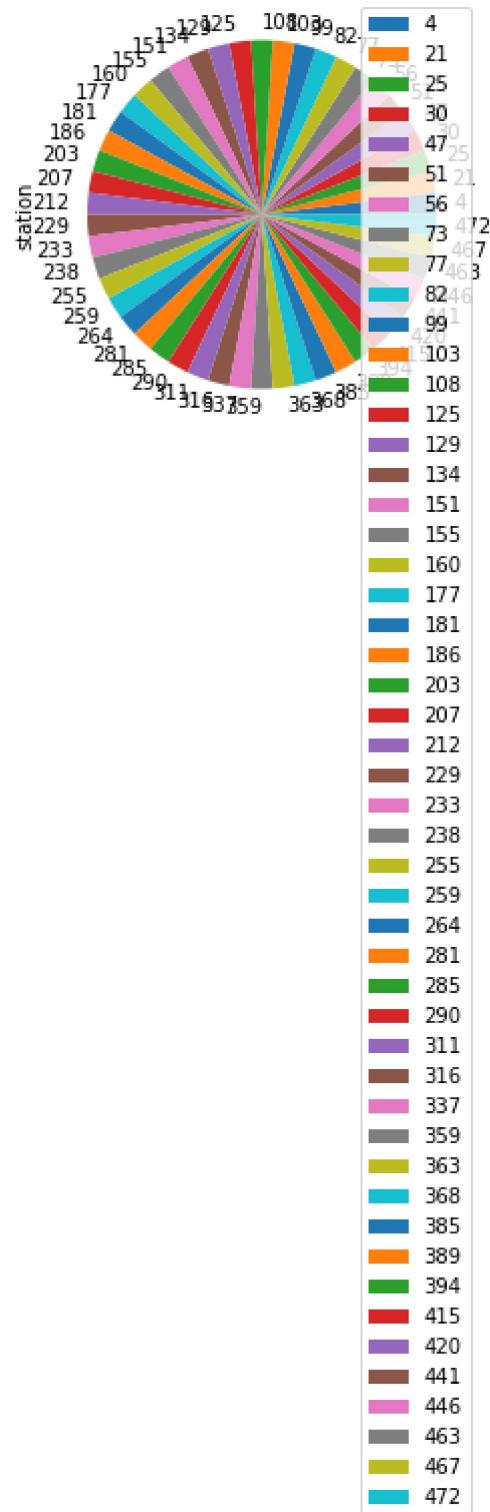
```
Out[13]: <AxesSubplot:>
```



Pie chart

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

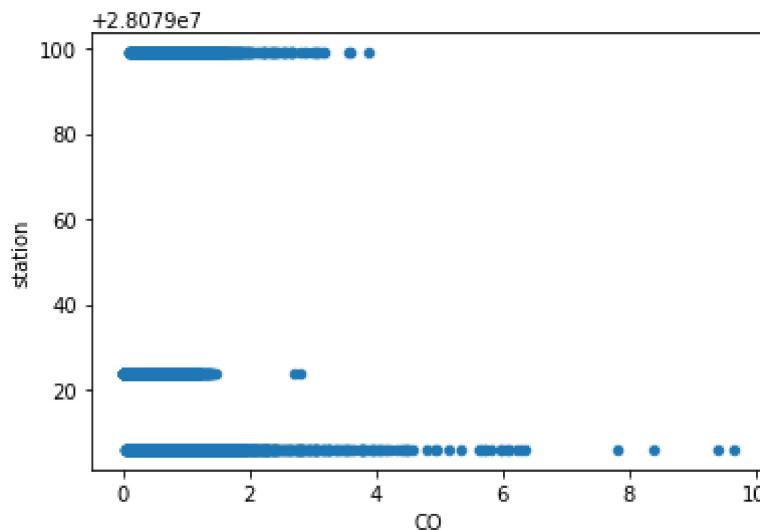
```
Out[14]: <AxesSubplot:ylabel='station'>
```



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  TCU       25443 non-null   float64
```

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

Out[17]:

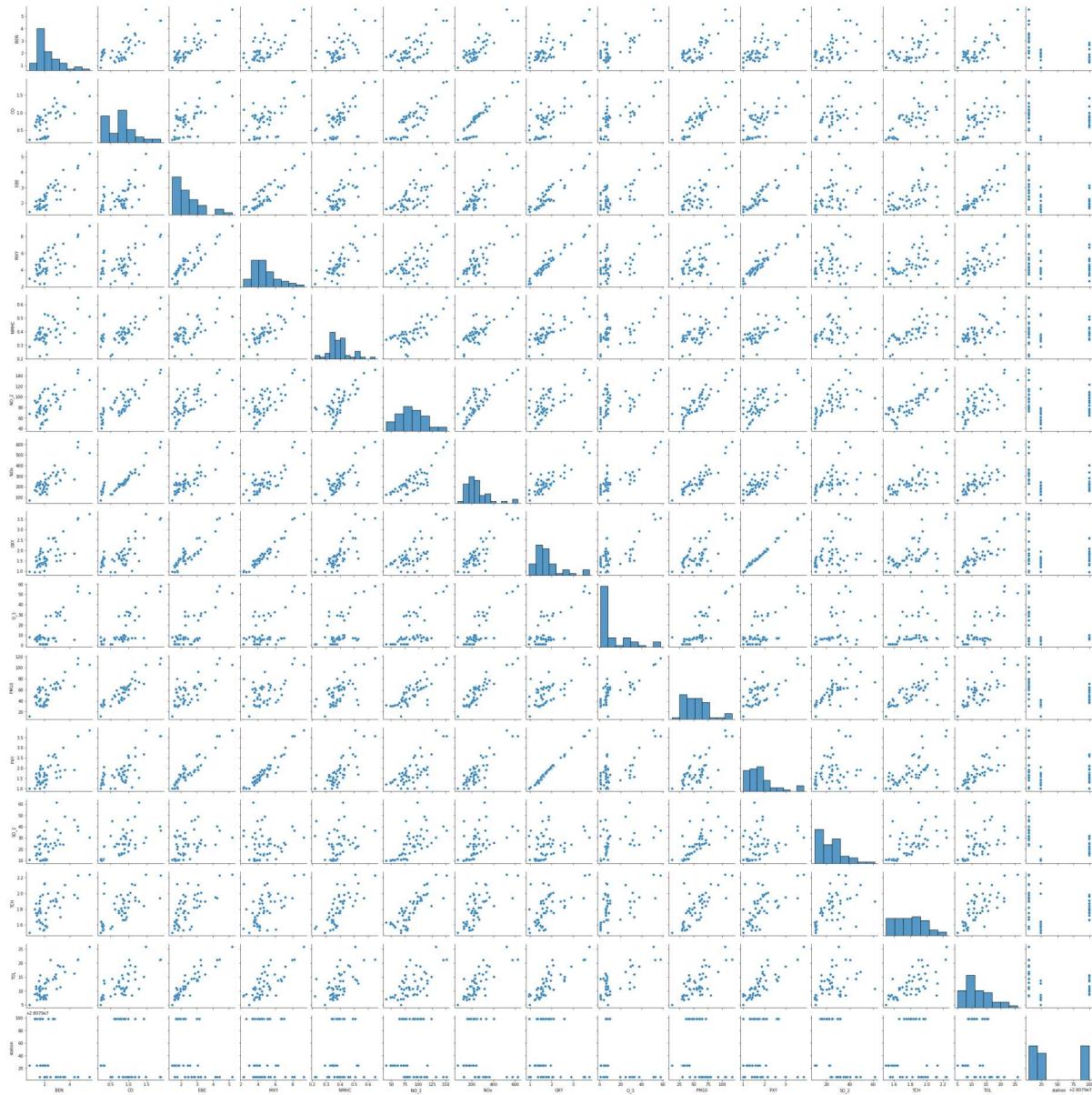
	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH	TOL	station
count	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000	25443.000000
mean	1.146744	0.505120	1.394071	2.392008	0.249967	58.532683	1.146744	0.505120	1.394071	2.392008	0.249967	58.532683	1.146744	0.505120	1.394071
std	1.278733	0.423231	1.268265	2.784302	0.142627	37.755029	1.278733	0.423231	1.268265	2.784302	0.142627	37.755029	1.278733	0.423231	1.268265
min	0.130000	0.000000	0.120000	0.150000	0.000000	1.690000	0.130000	0.000000	0.120000	0.150000	0.000000	1.690000	0.130000	0.000000	0.120000
25%	0.450000	0.260000	0.780000	0.960000	0.160000	31.285001	0.450000	0.260000	0.780000	0.960000	0.160000	31.285001	0.450000	0.260000	0.780000
50%	0.770000	0.400000	1.000000	1.500000	0.220000	54.080002	0.770000	0.400000	1.000000	1.500000	0.220000	54.080002	0.770000	0.400000	1.000000
75%	1.390000	0.640000	1.580000	2.855000	0.300000	79.230003	1.390000	0.640000	1.580000	2.855000	0.300000	79.230003	1.390000	0.640000	1.580000
max	30.139999	9.660000	31.680000	65.480003	2.570000	430.299988	30.139999	9.660000	31.680000	65.480003	2.570000	430.299988	30.139999	9.660000	31.680000

```
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])
```

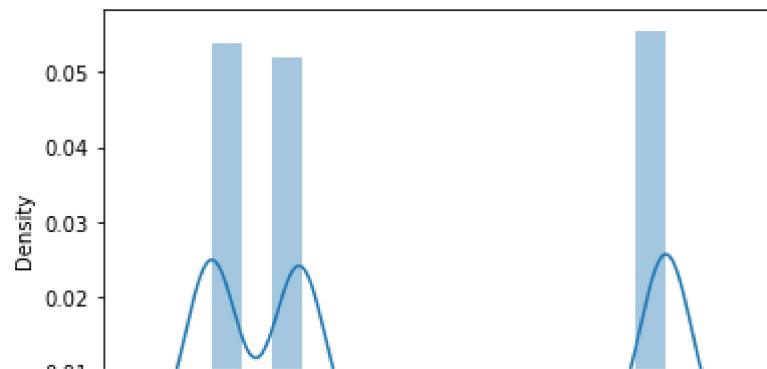
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x1ecd2511340>
```



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

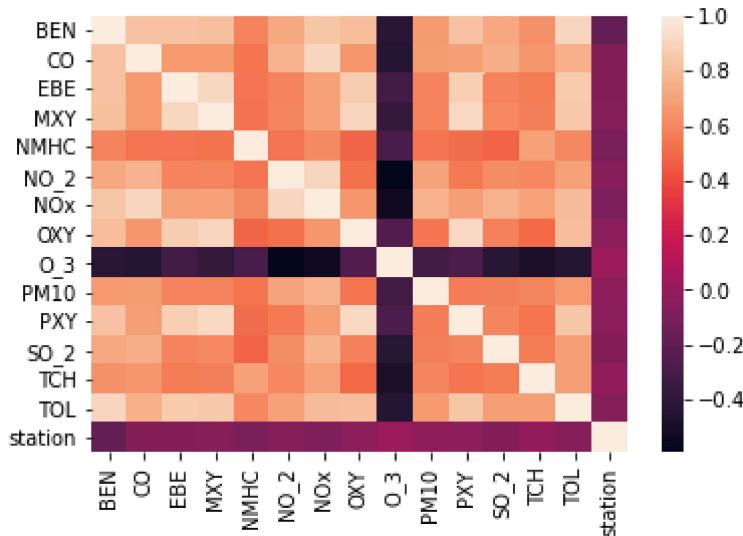
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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', '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]: 28079008.799052037
```

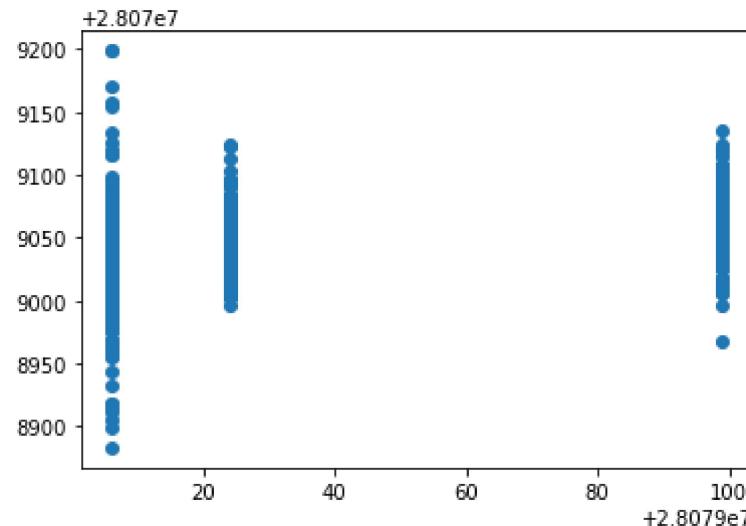
```
In [26]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[26]:
```

	Co-efficient
BEN	-31.984524
CO	18.872444
EBE	1.002761
MXY	-1.097711
NMHC	-40.641277
NO_2	0.115270
NOx	-0.054208
OXY	4.937038
O_3	-0.024302
PM10	0.161840
PXY	6.884508
SO_2	0.194289
TCH	25.821995
TOL	3.113284

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

```
Out[27]: <matplotlib.collections.PathCollection at 0x1ece0ea4bb0>
```



ACCURACY

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

```
Out[28]: 0.17013175288360394
```

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

```
Out[29]: 0.15414019483844354
```

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]: 0.17005971068495163
```

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

```
Out[33]: 0.15408873670733025
```

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]: 0.014721675432207926
```

ElasticNet

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

```
Out[36]: 0.010692798914223589
```

```
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([-7.77010227,  0.          ,  0.          ,  0.14958441, -0.          ,  
   0.05222803, -0.05752404,  0.62462209, -0.05418201,  0.17637898,  
   0.74892613,  0.          ,  0.          ,  0.95024653])
```

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

```
Out[39]: 28079045.782055568
```

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

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

```
Out[41]: 0.06718094988946799
```

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)))
```

36.70577386153973
1537.0618789020634
39.20538073915446

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_P10', '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="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]: 0.8215609208309939
```

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

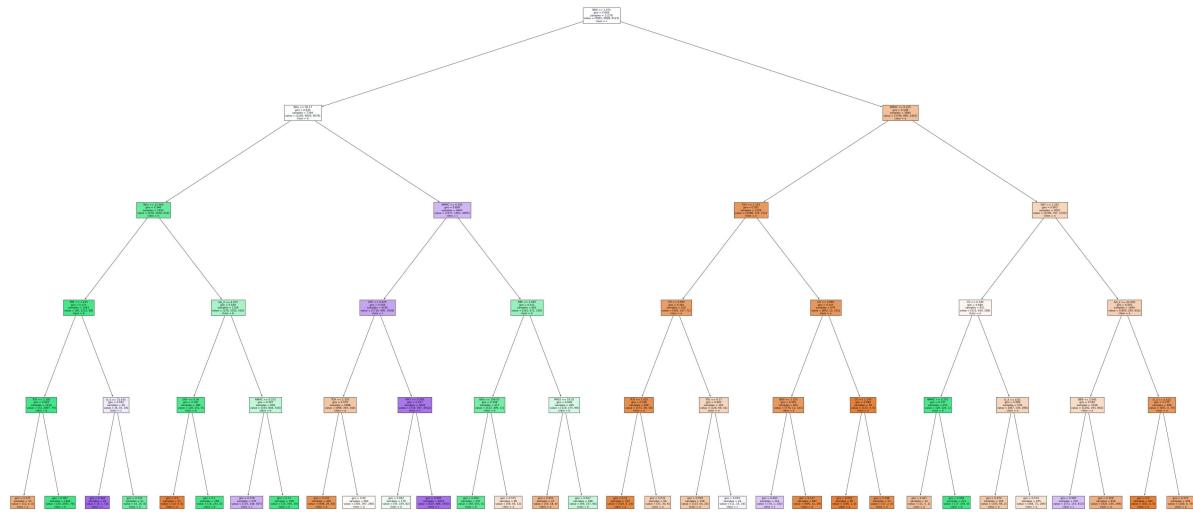
```
In [62]: from sklearn.tree import plot_tree
```

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

```
Out[62]: [Text(2232.0, 1993.2, 'BEN <= 1.075\ngini = 0.666\nsamples = 11278\nvalue = [5991, 5698, 6121]\nnclass = c'),
Text(1116.0, 1630.8000000000002, 'NOx <= 36.17\ngini = 0.635\nsamples = 7394\nvalue = [2195, 4829, 4679]\nnclass = b'),
Text(558.0, 1268.4, 'NOx <= 21.065\ngini = 0.346\nsamples = 2552\nvalue = [218, 3168, 614]\nnclass = b'),
Text(279.0, 906.0, 'EBE <= 1.015\ngini = 0.117\nsamples = 1443\nvalue = [40, 2113, 98]\nnclass = b'),
Text(139.5, 543.5999999999999, 'TCH <= 1.185\ngini = 0.097\nsamples = 1414\nvalue = [32, 2097, 79]\nnclass = b'),
Text(69.75, 181.1999999999982, 'gini = 0.375\nsamples = 10\nvalue = [12, 4, 0]\nnclass = a'),
Text(209.25, 181.1999999999982, 'gini = 0.087\nsamples = 1404\nvalue = [20, 2093, 79]\nnclass = b'),
Text(418.5, 543.5999999999999, 'O_3 <= 72.675\ngini = 0.632\nsamples = 29\nvalue = [8, 16, 19]\nnclass = c'),
Text(348.75, 181.1999999999982, 'gini = 0.344\nsamples = 18\nvalue = [4, 1, 19]\nnclass = c'),
Text(488.25, 181.1999999999982, 'gini = 0.332\nsamples = 11\nvalue = [4, 15, 0]\nnclass = b'),
Text(837.0, 906.0, 'SO_2 <= 4.975\ngini = 0.539\nsamples = 1109\nvalue = [178, 1055, 516]\nnclass = b'),
Text(697.5, 543.5999999999999, 'OXY <= 0.36\ngini = 0.181\nsamples = 180\nvalue = [28, 251, 0]\nnclass = b'),
Text(627.75, 181.1999999999982, 'gini = 0.0\nsamples = 11\nvalue = [14, 0, 0]\nnclass = a'),
Text(767.25, 181.1999999999982, 'gini = 0.1\nsamples = 169\nvalue = [14, 251, 0]\nnclass = b'),
Text(976.5, 543.5999999999999, 'NMHC <= 0.215\ngini = 0.567\nsamples = 929\nvalue = [150, 804, 516]\nnclass = b'),
Text(906.75, 181.1999999999982, 'gini = 0.578\nsamples = 570\nvalue = [135, 248, 497]\nnclass = c'),
Text(1046.25, 181.1999999999982, 'gini = 0.11\nsamples = 359\nvalue = [15, 556, 19]\nnclass = b'),
Text(1674.0, 1268.4, 'NMHC <= 0.295\ngini = 0.609\nsamples = 4842\nvalue = [1977, 1661, 4065]\nnclass = c'),
Text(1395.0, 906.0, 'OXY <= 0.625\ngini = 0.559\nsamples = 4180\nvalue = [1716, 990, 3956]\nnclass = c'),
Text(1255.5, 543.5999999999999, 'TCH <= 1.335\ngini = 0.579\nsamples = 1088\nvalue = [998, 393, 344]\nnclass = a'),
Text(1185.75, 181.1999999999982, 'gini = 0.232\nsamples = 426\nvalue = [606, 26, 64]\nnclass = a'),
Text(1325.25, 181.1999999999982, 'gini = 0.66\nsamples = 662\nvalue = [392, 367, 280]\nnclass = a'),
Text(1534.5, 543.5999999999999, 'MXY <= 0.595\ngini = 0.427\nsamples = 3092\nvalue = [718, 597, 3612]\nnclass = c'),
Text(1464.75, 181.1999999999982, 'gini = 0.662\nsamples = 179\nvalue = [93, 109, 82]\nnclass = b'),
Text(1604.25, 181.1999999999982, 'gini = 0.393\nsamples = 2913\nvalue = [625, 488, 3530]\nnclass = c'),
Text(1953.0, 906.0, 'EBE <= 1.045\ngini = 0.511\nsamples = 662\nvalue = [261, 671, 109]\nnclass = b'),
Text(1813.5, 543.5999999999999, 'NOx <= 108.05\ngini = 0.368\nsamples = 417\nvalue = [142, 496, 11]\nnclass = b'),
Text(1743.75, 181.1999999999982, 'gini = 0.264\nsamples = 337\nvalue = [84, 453, 0]\nnclass = b'),
Text(1883.25, 181.1999999999982, 'gini = 0.575\nsamples = 80\nvalue = [58,
```

```
43, 11]\nclass = a'),  
    Text(2092.5, 543.599999999999, 'PM10 <= 19.25\ngini = 0.646\nsamples = 245  
\nvalue = [119, 175, 98]\nclass = b'),  
    Text(2022.75, 181.1999999999982, 'gini = 0.452\nsamples = 47\nvalue = [50,  
18, 4]\nclass = a'),  
    Text(2162.25, 181.1999999999982, 'gini = 0.627\nsamples = 198\nvalue = [69,  
157, 94]\nclass = b'),  
    Text(3348.0, 1630.800000000002, 'NMHC <= 0.235\ngini = 0.538\nsamples = 388  
4\nvalue = [3796, 869, 1442]\nclass = a'),  
    Text(2790.0, 1268.4, 'PXY <= 1.115\ngini = 0.301\nsamples = 1229\nvalue = [1  
588, 122, 212]\nclass = a'),  
    Text(2511.0, 906.0, 'CO <= 0.655\ngini = 0.345\nsamples = 550\nvalue = [695,  
107, 71]\nclass = a'),  
    Text(2371.5, 543.599999999999, 'TCH <= 1.425\ngini = 0.196\nsamples = 408\n  
value = [571, 38, 30]\nclass = a'),  
    Text(2301.75, 181.1999999999982, 'gini = 0.11\nsamples = 352\nvalue = [520,  
8, 24]\nclass = a'),  
    Text(2441.25, 181.1999999999982, 'gini = 0.533\nsamples = 56\nvalue = [51,  
30, 6]\nclass = a'),  
    Text(2650.5, 543.599999999999, 'TOL <= 6.17\ngini = 0.602\nsamples = 142\nv  
alue = [124, 69, 41]\nclass = a'),  
    Text(2580.75, 181.1999999999982, 'gini = 0.558\nsamples = 118\nvalue = [11  
3, 54, 25]\nclass = a'),  
    Text(2720.25, 181.1999999999982, 'gini = 0.659\nsamples = 24\nvalue = [11,  
15, 16]\nclass = c'),  
    Text(3069.0, 906.0, 'CO <= 0.885\ngini = 0.257\nsamples = 679\nvalue = [893,  
15, 141]\nclass = a'),  
    Text(2929.5, 543.599999999999, 'BEN <= 1.325\ngini = 0.281\nsamples = 601\n  
value = [770, 12, 141]\nclass = a'),  
    Text(2859.75, 181.1999999999982, 'gini = 0.493\nsamples = 114\nvalue = [74,  
1, 102]\nclass = c'),  
    Text(2999.25, 181.1999999999982, 'gini = 0.127\nsamples = 487\nvalue = [69  
6, 11, 39]\nclass = a'),  
    Text(3208.5, 543.599999999999, 'CO <= 1.145\ngini = 0.046\nsamples = 78\nv  
alue = [123, 3, 0]\nclass = a'),  
    Text(3138.75, 181.1999999999982, 'gini = 0.019\nsamples = 65\nvalue = [106,  
1, 0]\nclass = a'),  
    Text(3278.25, 181.1999999999982, 'gini = 0.188\nsamples = 13\nvalue = [17,  
2, 0]\nclass = a'),  
    Text(3906.0, 1268.4, 'OXY <= 1.255\ngini = 0.603\nsamples = 2655\nvalue = [2  
208, 747, 1230]\nclass = a'),  
    Text(3627.0, 906.0, 'CO <= 0.345\ngini = 0.649\nsamples = 811\nvalue = [515,  
454, 289]\nclass = a'),  
    Text(3487.5, 543.599999999999, 'NMHC <= 0.255\ngini = 0.151\nsamples = 233  
\nvalue = [28, 324, 1]\nclass = b'),  
    Text(3417.75, 181.1999999999982, 'gini = 0.491\nsamples = 12\nvalue = [11,  
5, 1]\nclass = a'),  
    Text(3557.25, 181.1999999999982, 'gini = 0.096\nsamples = 221\nvalue = [17,  
319, 0]\nclass = b'),  
    Text(3766.5, 543.599999999999, 'O_3 <= 4.91\ngini = 0.589\nsamples = 578\nv  
alue = [487, 130, 288]\nclass = a'),  
    Text(3696.75, 181.1999999999982, 'gini = 0.474\nsamples = 103\nvalue = [10  
3, 58, 2]\nclass = a'),  
    Text(3836.25, 181.1999999999982, 'gini = 0.574\nsamples = 475\nvalue = [38  
4, 72, 286]\nclass = a'),  
    Text(4185.0, 906.0, 'SO_2 <= 32.005\ngini = 0.552\nsamples = 1844\nvalue =  
[1693, 293, 941]\nclass = a'),
```

```
Text(4045.5, 543.5999999999999, 'BEN <= 2.045\n gini = 0.583\n samples = 1538\n value = [1292, 293, 862]\n class = a'),  
Text(3975.75, 181.1999999999982, 'gini = 0.585\n samples = 722\n value = [372, 153, 613]\n class = c'),  
Text(4115.25, 181.1999999999982, 'gini = 0.458\n samples = 816\n value = [920, 140, 249]\n class = a'),  
Text(4324.5, 543.5999999999999, '0_3 <= 4.415\n gini = 0.275\n samples = 306\n value = [401, 0, 79]\n class = a'),  
Text(4254.75, 181.1999999999982, 'gini = 0.0\n samples = 102\n value = [161, 0, 0]\n class = a'),  
Text(4394.25, 181.1999999999982, 'gini = 0.373\n samples = 204\n value = [240, 0, 79]\n class = a')
```



Conclusion

Accuracy

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

Out[63]: 0.15414019483844354

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

Out[64]: 0.15408873670733025

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

Out[65]: 0.014721675432207926

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

Out[66]: 0.06718094988946799

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

```
Out[67]: 0.8146838030106512
```

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

```
Out[68]: 0.8215609208309939
```

Random is suitable for this dataset