

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_2006.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570000
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820000
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.410000
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260000
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180000
...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120000
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.460000
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680000
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360000
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490000

230568 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: 24758 entries, 5 to 230567
Data columns (total 17 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      24758 non-null   object 
 1   BEN        24758 non-null   float64
 2   CO         24758 non-null   float64
 3   EBE        24758 non-null   float64
 4   MXY        24758 non-null   float64
 5   NMHC       24758 non-null   float64
 6   NO_2       24758 non-null   float64
 7   NOx        24758 non-null   float64
 8   OXY        24758 non-null   float64
 9   O_3         24758 non-null   float64
 10  PM10       24758 non-null   float64
 11  PM25       24758 non-null   float64
 12  PXY        24758 non-null   float64
 13  SO_2       24758 non-null   float64
 14  TCH         24758 non-null   float64
 15  TOL         24758 non-null   float64
 16  station    24758 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

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

Out[6]:

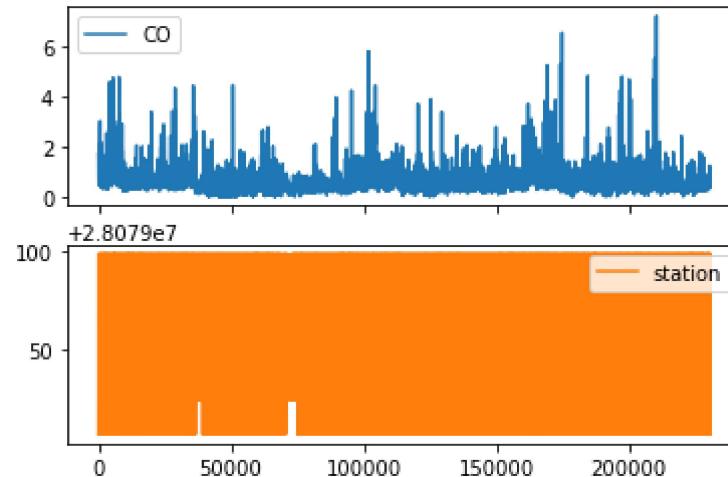
	CO	station
5	1.69	28079006
22	0.79	28079024
25	1.47	28079099
31	0.85	28079006
48	0.79	28079024
...
230538	0.40	28079024
230541	0.94	28079099
230547	1.06	28079006
230564	0.32	28079024
230567	0.74	28079099

24758 rows × 2 columns

Line chart

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

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

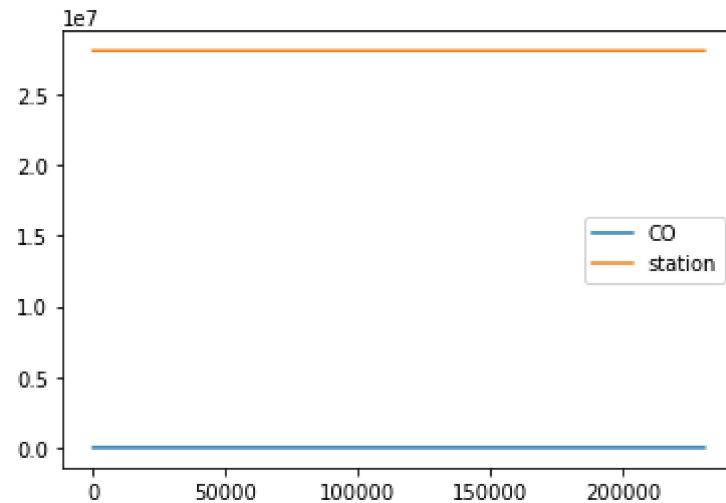


Line chart

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

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

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

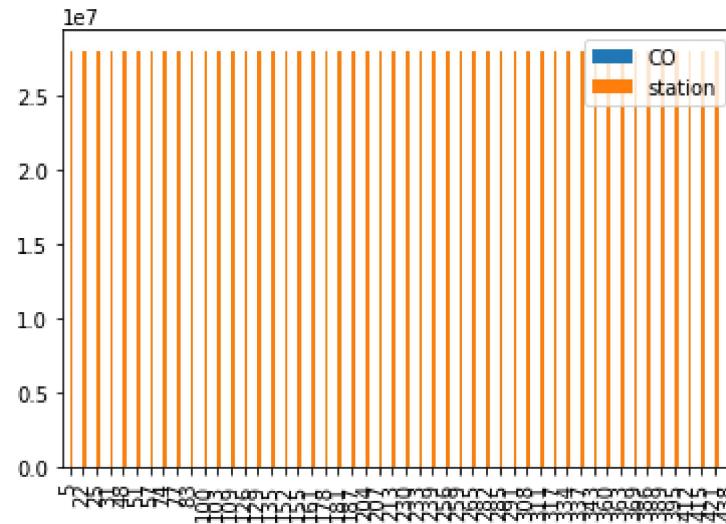


Bar chart

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

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

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

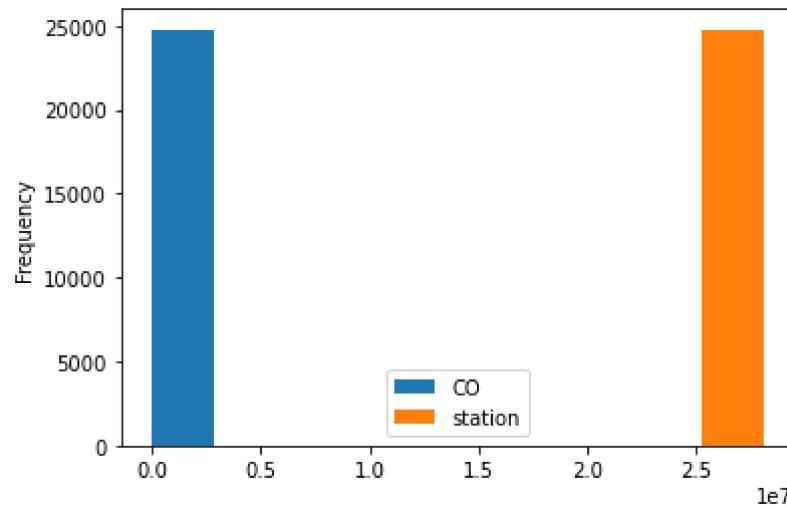


Histogram

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

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

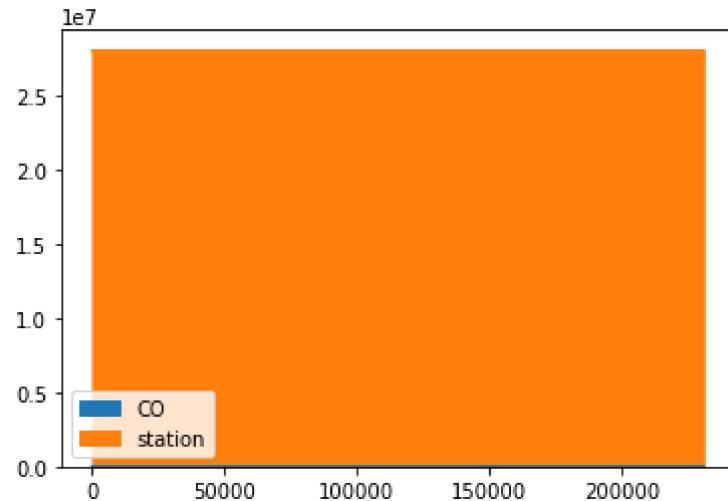
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

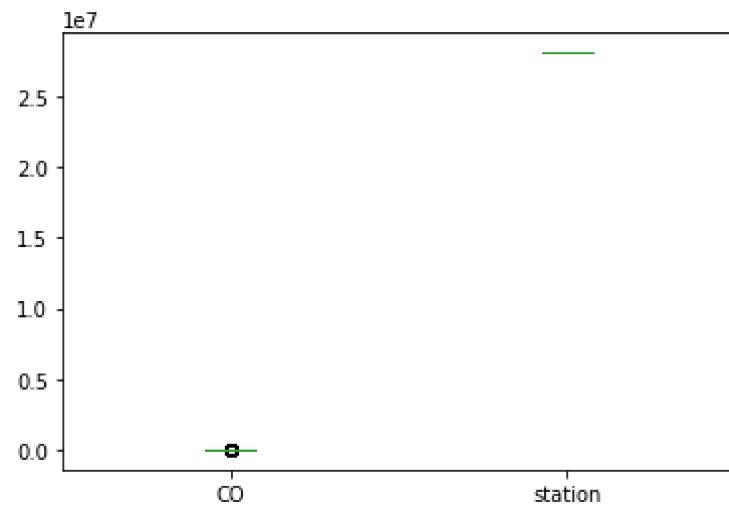
```
Out[12]: <AxesSubplot:>
```



Box chart

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

Out[13]: <AxesSubplot:>

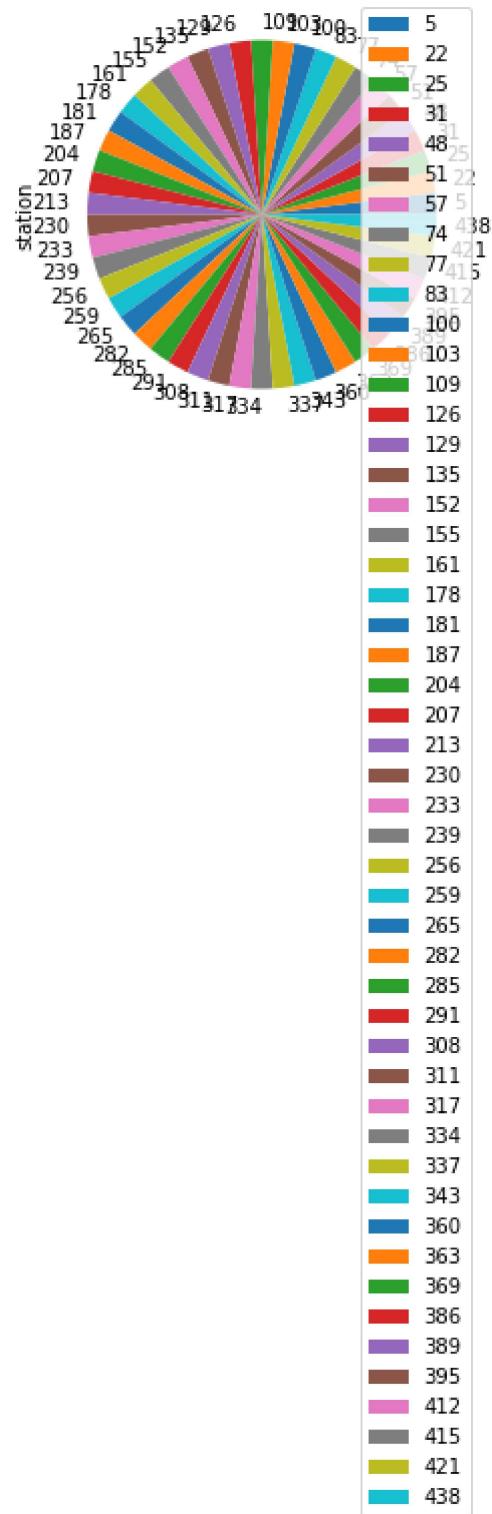


Pie chart

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

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

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

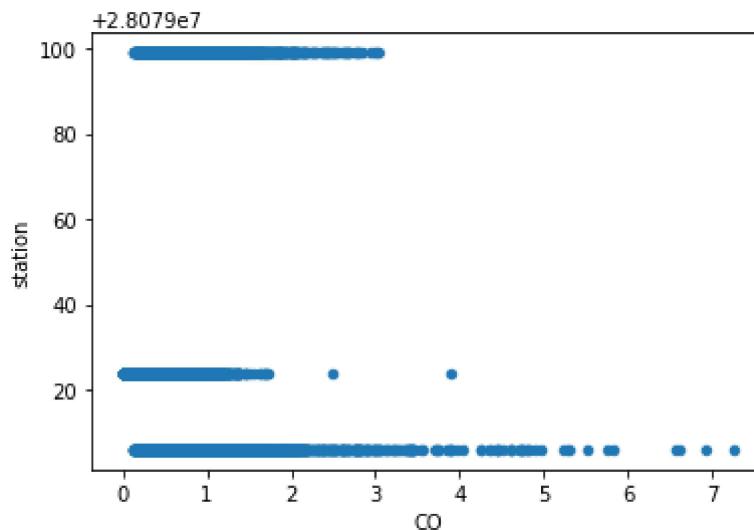


Scatter chart

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

```
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: 24758 entries, 5 to 230567
Data columns (total 17 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      24758 non-null   object 
 1   BEN       24758 non-null   float64
 2   CO        24758 non-null   float64
 3   EBE       24758 non-null   float64
 4   MXY       24758 non-null   float64
 5   NMHC      24758 non-null   float64
 6   NO_2      24758 non-null   float64
 7   NOx       24758 non-null   float64
 8   OXY       24758 non-null   float64
 9   O_3        24758 non-null   float64
 10  PM10      24758 non-null   float64
 11  PM25      24758 non-null   float64
 12  PXY       24758 non-null   float64
 13  SO_2      24758 non-null   float64
 14  TCU       24758 non-null   float64
```

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

Out[17]:

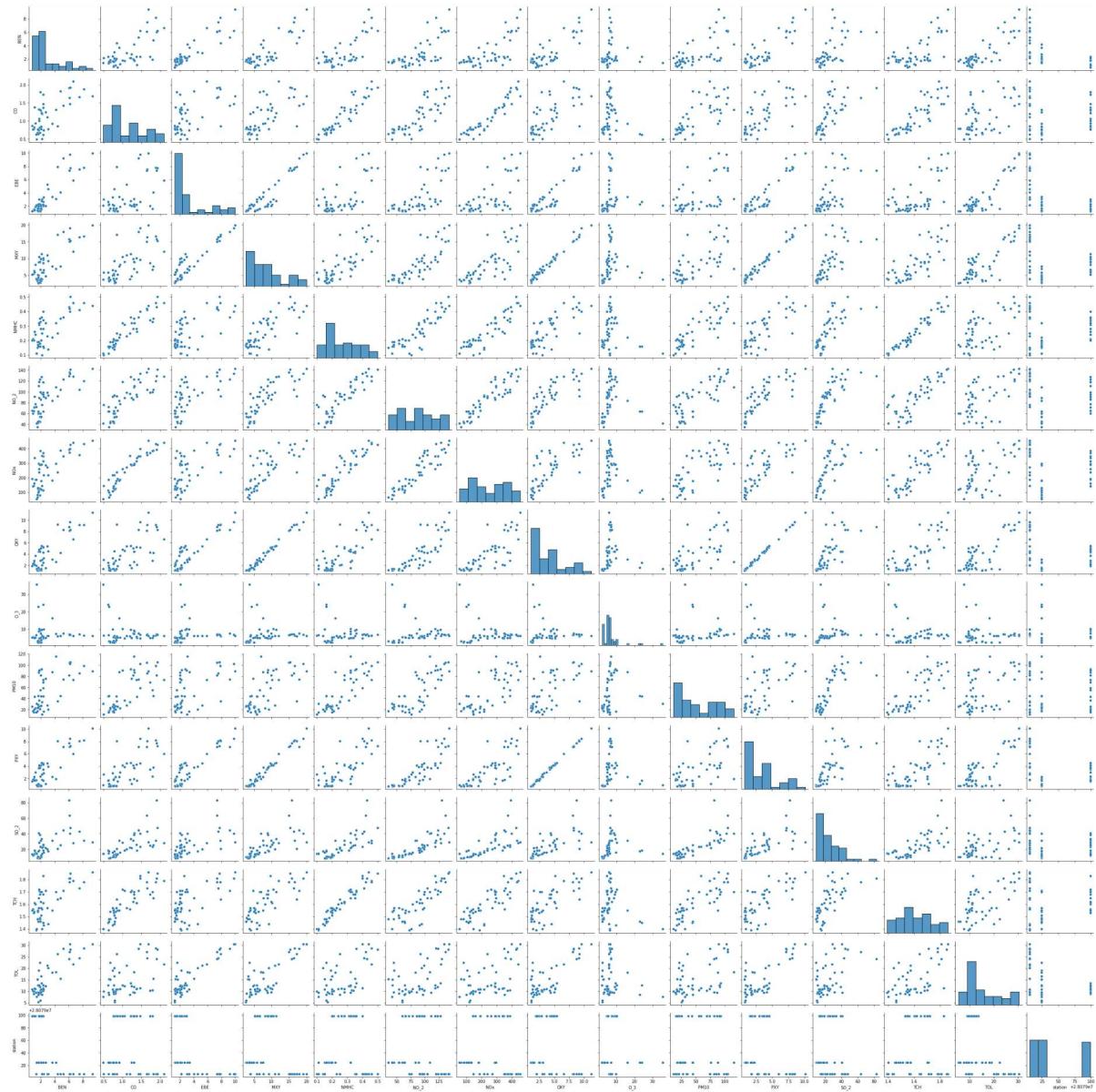
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	247
mean	1.350624	0.600713	1.824534	3.835034	0.176546	58.333481	1
std	1.541636	0.419048	1.868939	4.069036	0.126683	40.529382	1
min	0.110000	0.000000	0.170000	0.150000	0.000000	1.680000	
25%	0.450000	0.360000	0.810000	1.060000	0.100000	28.450001	
50%	0.850000	0.500000	1.130000	2.500000	0.150000	52.959999	
75%	1.680000	0.720000	2.160000	5.090000	0.220000	79.347498	1
max	45.430000	7.250000	57.799999	66.900002	2.020000	461.299988	16

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])
```

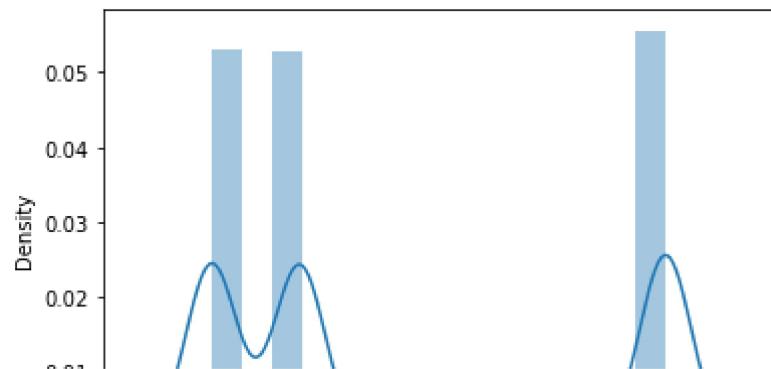
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x20746218310>
```



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

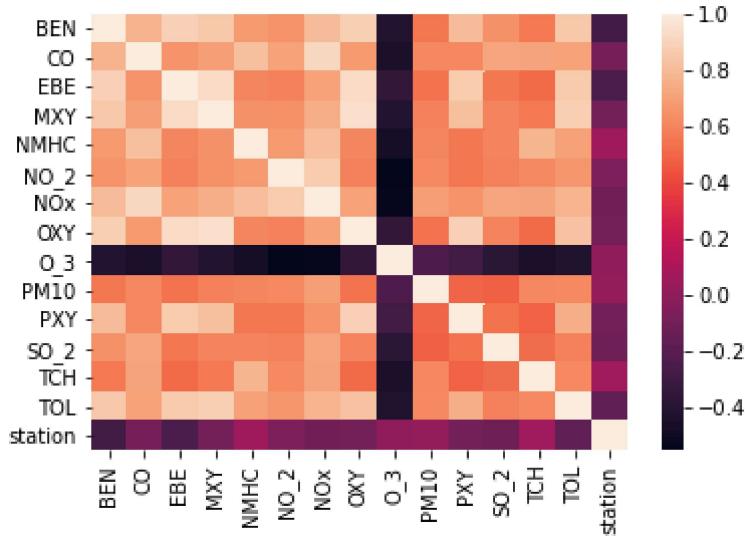
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]: 28079015.588567782
```

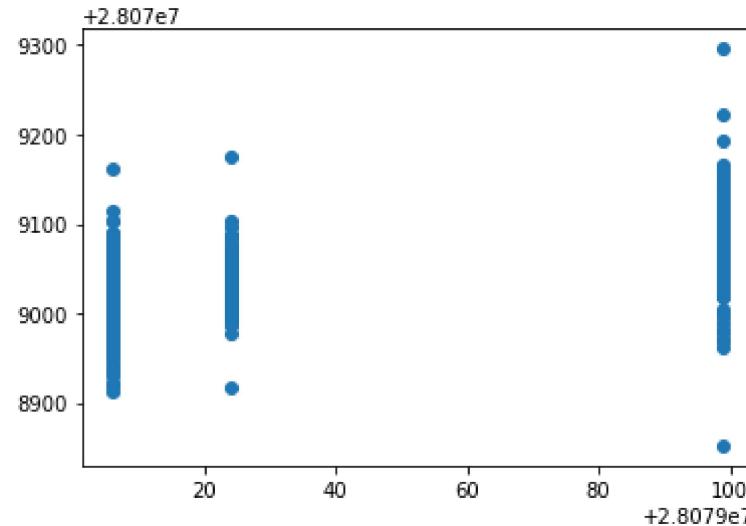
```
In [26]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[26]:
```

	Co-efficient
BEN	-17.939387
CO	-12.484279
EBE	-20.896338
MXY	3.112156
NMHC	129.712387
NO_2	-0.012739
NOx	-0.007219
OXY	17.009351
O_3	-0.062172
PM10	0.139655
PXY	5.690253
SO_2	-0.650941
TCH	21.845341
TOL	-0.746093

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

```
Out[27]: <matplotlib.collections.PathCollection at 0x20754b78b80>
```



ACCURACY

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

```
Out[28]: 0.4240734625709568
```

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

```
Out[29]: 0.3800786591927453
```

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]: 0.4235020625145921
```

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

```
Out[33]: 0.37940784076369816
```

```
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]: 0.06075911859202254
```

Accuracy(Lasso)

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

```
Out[36]: 0.061448685032805095
```

```
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([-8.20834801e+00,  0.00000000e+00, -8.34211906e+00,  2.96605173e+00,
        4.20172677e-01, -4.79797369e-03,  4.42047172e-03,  3.93194444e+00,
       -1.22700649e-01,  2.97521903e-01,  2.72500337e+00, -4.50154020e-01,
       5.87061868e-01, -1.14050950e+00])
```

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

```
Out[39]: 28079051.76643599
```

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

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

```
Out[41]: 0.2458875645592553
```

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)))
```

32.336060271781456
1244.7245812687047
35.28065449036773

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]: (24758, 14)
```

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

```
Out[46]: (24758,)
```

```
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]: 0.8741416915744405
```

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

```
Out[54]: 3.5557727473608076e-15
```

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

```
Out[55]: array([[3.55577275e-15, 7.80743173e-29, 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="accuracy")  
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]: 0.8728216964800923
```

```
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'])
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

```
Out[62]: [Text(2152.285714285714, 1993.2, 'BEN <= 1.365\n gini = 0.666\n samples = 10930\n nvalue = [5600, 5686, 6044]\n nclass = c'),\n    Text(1096.0714285714284, 1630.8000000000002, 'OXY <= 1.005\n gini = 0.61\n samples = 7446\n nvalue = [1647, 5051, 5145]\n nclass = c'),\n    Text(597.8571428571428, 1268.4, 'PXY <= 0.995\n gini = 0.477\n samples = 4315\n nvalue = [791, 4703, 1369]\n nclass = b'),\n    Text(318.85714285714283, 906.0, 'TCH <= 1.265\n gini = 0.595\n samples = 2772\n nvalue = [745, 2390, 1293]\n nclass = b'),\n    Text(159.42857142857142, 543.5999999999999, 'TCH <= 1.155\n gini = 0.581\n samples = 460\n nvalue = [342, 305, 69]\n nclass = a'),\n    Text(79.71428571428571, 181.1999999999982, 'gini = 0.235\n samples = 123\n nvalue = [26, 165, 0]\n nclass = b'),\n    Text(239.1428571428571, 181.1999999999982, 'gini = 0.549\n samples = 337\n nvalue = [316, 140, 69]\n nclass = a'),\n    Text(478.2857142857142, 543.5999999999999, 'EBE <= 0.545\n gini = 0.564\n samples = 2312\n nvalue = [403, 2085, 1224]\n nclass = b'),\n    Text(398.57142857142856, 181.1999999999982, 'gini = 0.367\n samples = 557\n nvalue = [95, 695, 101]\n nclass = b'),\n    Text(558.0, 181.1999999999982, 'gini = 0.587\n samples = 1755\n nvalue = [308, 1390, 1123]\n nclass = b'),\n    Text(876.8571428571428, 906.0, 'EBE <= 1.08\n gini = 0.096\n samples = 1543\n nvalue = [46, 2313, 76]\n nclass = b'),\n    Text(797.1428571428571, 543.5999999999999, 'NOx <= 47.045\n gini = 0.078\n samples = 1517\n nvalue = [39, 2291, 57]\n nclass = b'),\n    Text(717.4285714285713, 181.1999999999982, 'gini = 0.015\n samples = 1412\n nvalue = [2, 2217, 15]\n nclass = b'),\n    Text(876.8571428571428, 181.1999999999982, 'gini = 0.632\n samples = 105\n nvalue = [37, 74, 42]\n nclass = b'),\n    Text(956.5714285714284, 543.5999999999999, 'gini = 0.612\n samples = 26\n nvalue = [7, 22, 19]\n nclass = b'),\n    Text(1594.2857142857142, 1268.4, 'EBE <= 1.235\n gini = 0.391\n samples = 3131\n nvalue = [856, 348, 3776]\n nclass = c'),\n    Text(1275.4285714285713, 906.0, 'NMHC <= 0.075\n gini = 0.167\n samples = 1491\n nvalue = [137, 75, 2147]\n nclass = c'),\n    Text(1116.0, 543.5999999999999, 'NMHC <= 0.045\n gini = 0.342\n samples = 63\n nvalue = [79, 8, 12]\n nclass = a'),\n    Text(1036.2857142857142, 181.1999999999982, 'gini = 0.165\n samples = 34\n nvalue = [50, 5, 0]\n nclass = a'),\n    Text(1195.7142857142856, 181.1999999999982, 'gini = 0.487\n samples = 29\n nvalue = [29, 3, 12]\n nclass = a'),\n    Text(1434.8571428571427, 543.5999999999999, 'BEN <= 0.915\n gini = 0.106\n samples = 1428\n nvalue = [58, 67, 2135]\n nclass = c'),\n    Text(1355.142857142857, 181.1999999999982, 'gini = 0.047\n samples = 1233\n nvalue = [15, 32, 1902]\n nclass = c'),\n    Text(1514.5714285714284, 181.1999999999982, 'gini = 0.407\n samples = 195\n nvalue = [43, 35, 233]\n nclass = c'),\n    Text(1913.1428571428569, 906.0, 'CO <= 0.435\n gini = 0.528\n samples = 1640\n nvalue = [719, 273, 1629]\n nclass = c'),\n    Text(1753.7142857142856, 543.5999999999999, 'TCH <= 1.305\n gini = 0.588\n samples = 633\n nvalue = [516, 119, 384]\n nclass = a'),\n    Text(1673.999999999998, 181.1999999999982, 'gini = 0.216\n samples = 265\n nvalue = [381, 11, 41]\n nclass = a'),\n    Text(1833.4285714285713, 181.1999999999982, 'gini = 0.57\n samples = 368\n nvalue = [135, 108, 343]\n nclass = c'),\n    Text(2072.5714285714284, 543.5999999999999, 'TCH <= 1.325\n gini = 0.371\n samples = 107\n nvalue = [203, 154, 1245]\n nclass = c'),\n    Text(1992.8571428571427, 181.1999999999982, 'gini = 0.423\n samples = 99\n nvalue = [100, 100, 99]\n nclass = a')]
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/HTML-CSS/WebFontdata.js

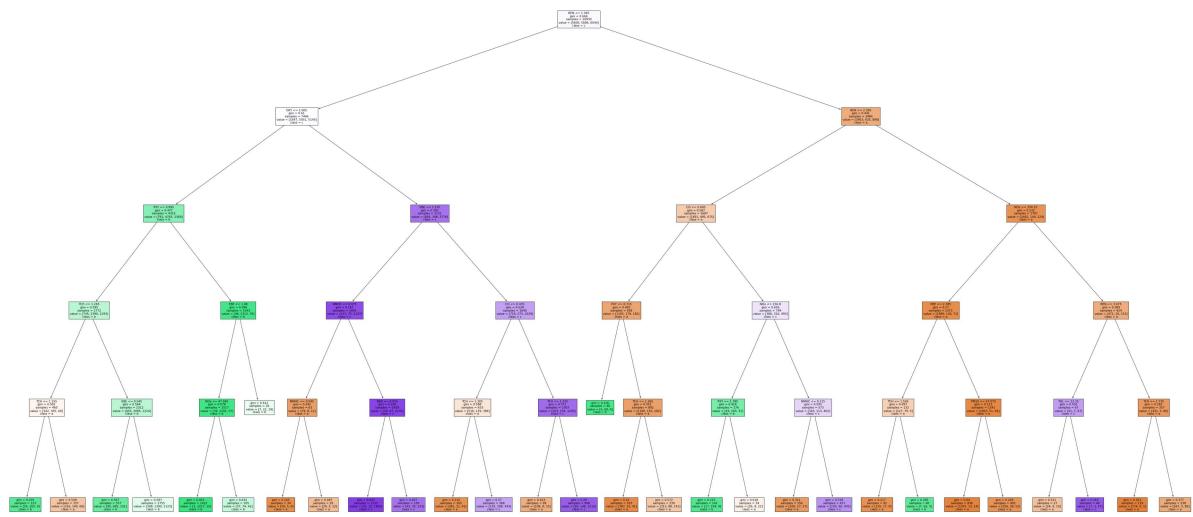
```

lue = [108, 8, 33]\nclass = a'),
Text(2152.285714285714, 181.19999999999982, 'gini = 0.29\nsamples = 908\nval
ue = [95, 146, 1212]\nclass = c'),
Text(3208.499999999995, 1630.8000000000002, 'BEN <= 2.305\ngini = 0.441\nsa
mples = 3484\nvalue = [3953, 635, 899]\nclass = a'),
Text(2590.7142857142853, 1268.4, 'CO <= 0.685\ngini = 0.587\nsamples = 1687
\nvalue = [1491, 495, 675]\nclass = a'),
Text(2311.7142857142853, 906.0, 'PXY <= 0.715\ngini = 0.402\nsamples = 938\n
value = [1105, 179, 182]\nclass = a'),
Text(2232.0, 543.599999999999, 'gini = 0.126\nsamples = 46\nvalue = [5, 69,
0]\nclass = b'),
Text(2391.428571428571, 543.599999999999, 'TCH <= 1.365\ngini = 0.352\nsa
mple = 892\nvalue = [1100, 110, 182]\nclass = a'),
Text(2311.7142857142853, 181.1999999999982, 'gini = 0.14\nsamples = 553\nva
lue = [787, 22, 41]\nclass = a'),
Text(2471.142857142857, 181.1999999999982, 'gini = 0.572\nsamples = 339\nva
lue = [313, 88, 141]\nclass = a'),
Text(2869.7142857142853, 906.0, 'NOx <= 156.8\ngini = 0.656\nsamples = 749\n
value = [386, 316, 493]\nclass = c'),
Text(2710.285714285714, 543.599999999999, 'PXY <= 1.785\ngini = 0.426\nsa
mple = 176\nvalue = [43, 203, 31]\nclass = b'),
Text(2630.5714285714284, 181.1999999999982, 'gini = 0.215\nsamples = 142\nv
alue = [17, 194, 9]\nclass = b'),
Text(2790.0, 181.1999999999982, 'gini = 0.618\nsamples = 34\nvalue = [26,
9, 22]\nclass = a'),
Text(3029.142857142857, 543.599999999999, 'NMHC <= 0.215\ngini = 0.592\nsa
mple = 573\nvalue = [343, 113, 462]\nclass = c'),
Text(2949.428571428571, 181.1999999999982, 'gini = 0.311\nsamples = 156\nva
lue = [200, 17, 27]\nclass = a'),
Text(3108.8571428571427, 181.1999999999982, 'gini = 0.518\nsamples = 417\nv
alue = [143, 96, 435]\nclass = c'),
Text(3826.2857142857138, 1268.4, 'NOx <= 334.15\ngini = 0.232\nsamples = 179
7\nvalue = [2462, 140, 224]\nclass = a'),
Text(3507.428571428571, 906.0, 'EBE <= 2.385\ngini = 0.17\nsamples = 1373\nv
alue = [1990, 130, 71]\nclass = a'),
Text(3347.999999999995, 543.599999999999, 'TCH <= 1.505\ngini = 0.497\nsa
mple = 132\nvalue = [127, 79, 5]\nclass = a'),
Text(3268.285714285714, 181.1999999999982, 'gini = 0.217\nsamples = 92\nval
ue = [120, 17, 0]\nclass = a'),
Text(3427.7142857142853, 181.1999999999982, 'gini = 0.285\nsamples = 40\nva
lue = [7, 62, 5]\nclass = b'),
Text(3666.8571428571427, 543.599999999999, 'PM10 <= 57.075\ngini = 0.113\ns
amples = 1241\nvalue = [1863, 51, 66]\nclass = a'),
Text(3587.142857142857, 181.1999999999982, 'gini = 0.04\nsamples = 836\nval
ue = [1307, 13, 14]\nclass = a'),
Text(3746.5714285714284, 181.1999999999982, 'gini = 0.249\nsamples = 405\nv
alue = [556, 38, 52]\nclass = a'),
Text(4145.142857142857, 906.0, 'BEN <= 3.075\ngini = 0.389\nsamples = 424\nv
alue = [472, 10, 153]\nclass = a'),
Text(3985.7142857142853, 543.599999999999, 'TOL <= 13.35\ngini = 0.501\nsa
mple = 67\nvalue = [31, 7, 67]\nclass = c'),
Text(3905.999999999995, 181.1999999999982, 'gini = 0.521\nsamples = 27\nva
lue = [24, 4, 10]\nclass = a'),
Text(4065.428571428571, 181.1999999999982, 'gini = 0.263\nsamples = 40\nval
ue = [7, 3, 57]\nclass = c'),
Text(4304.571428571428, 543.599999999999, 'TCH <= 1.735\ngini = 0.281\nsa
mple = 357\nvalue = [441, 3, 86]\nclass = a'),

```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

```
Text(4224.857142857142, 181.19999999999982, 'gini = 0.011\nsamples = 119\nvalue = [174, 0, 1]\nclass = a'),
Text(4384.285714285714, 181.19999999999982, 'gini = 0.377\nsamples = 238\nvalue = [267, 3, 85]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression:0.3800786591927453

Ridge Regression:0.37940784076369816

Lasso Regression:0.06075911859202254

ElasticNet Regression:0.2458875645592553

Logistic Regression:0.8741416915744405

Random Forest:0.8728216964800923

Logistic Regression is suitable for this dataset