

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(r"E:\154\C10_air\csvs_per_year\csvs_per_year\madrid_2017.csv")
df
```

Out[2]:

	date	BEN	CH4	CO	EBC	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TC
0	2017-06-01 01:00:00	NaN	NaN	0.3	NaN	NaN	4.0	38.0	NaN	NaN	NaN	NaN	5.0	Na
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	Na
3	2017-06-01 01:00:00	NaN	NaN	0.2	NaN	NaN	1.0	9.0	NaN	91.0	NaN	NaN	NaN	Na
4	2017-06-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	19.0	NaN	69.0	NaN	NaN	2.0	Na
...
210115	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	27.0	NaN	65.0	NaN	NaN	NaN	Na
210116	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	14.0	NaN	NaN	73.0	NaN	7.0	Na
210117	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	4.0	NaN	83.0	NaN	NaN	NaN	Na
210118	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	11.0	NaN	78.0	NaN	NaN	NaN	Na
210119	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	14.0	NaN	77.0	60.0	NaN	NaN	Na

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]:

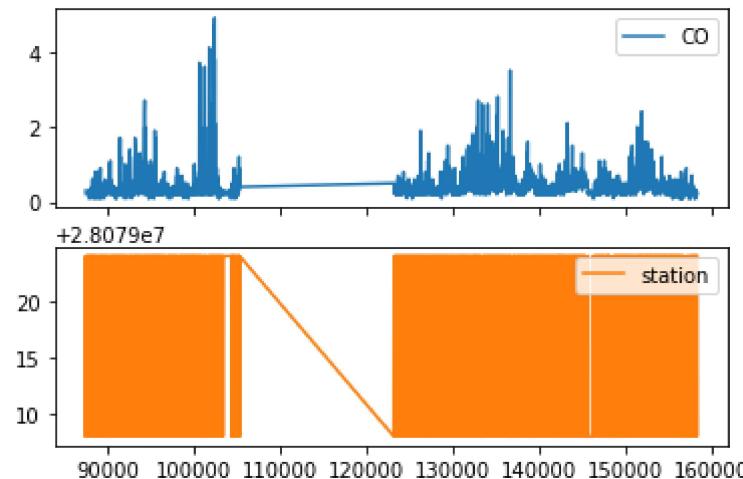
	CO	station
87457	0.3	28079008
87462	0.2	28079024
87481	0.2	28079008
87486	0.2	28079024
87505	0.2	28079008
...
158238	0.2	28079024
158257	0.3	28079008
158262	0.2	28079024
158281	0.2	28079008
158286	0.2	28079024

4127 rows × 2 columns

Line chart

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

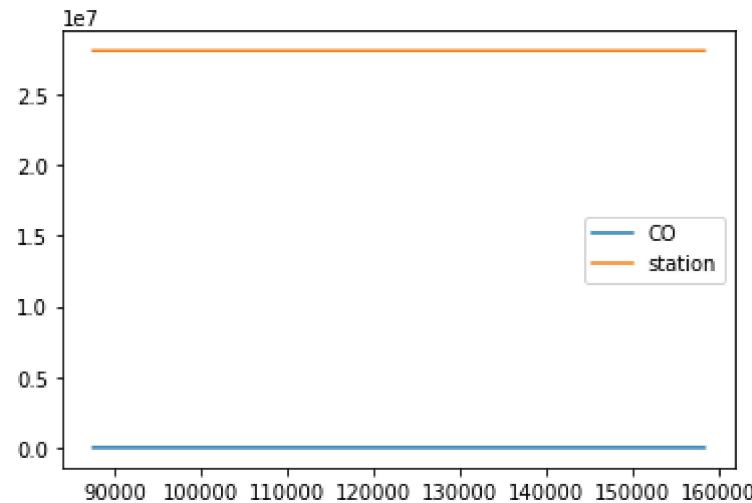
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

Out[8]: <AxesSubplot:>

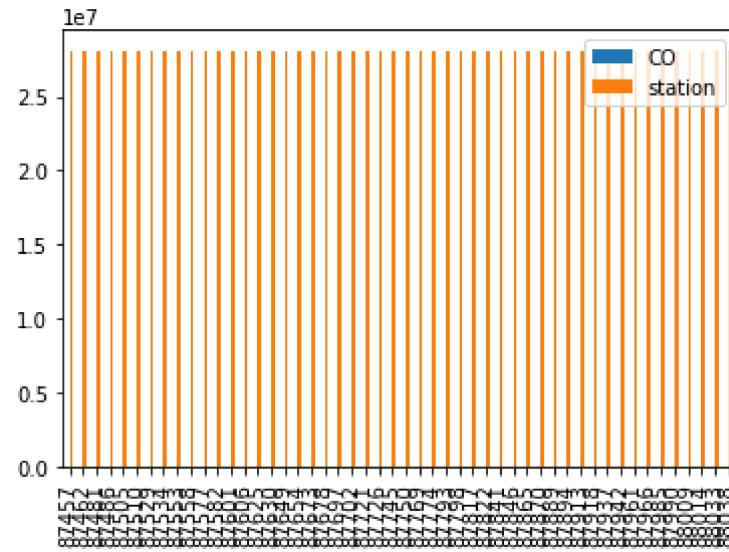


Bar chart

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

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

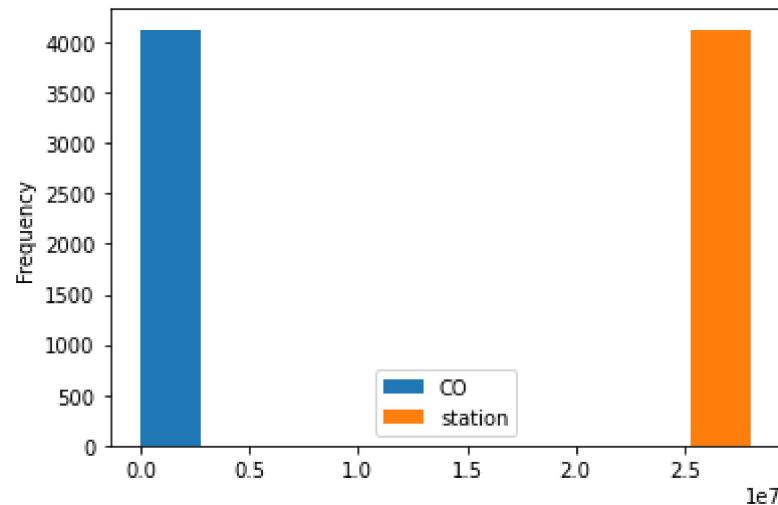
Out[10]: <AxesSubplot:>



Histogram

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

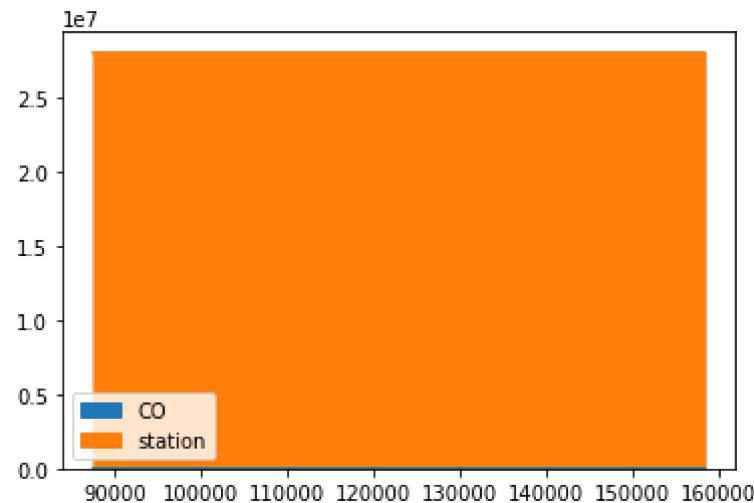
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

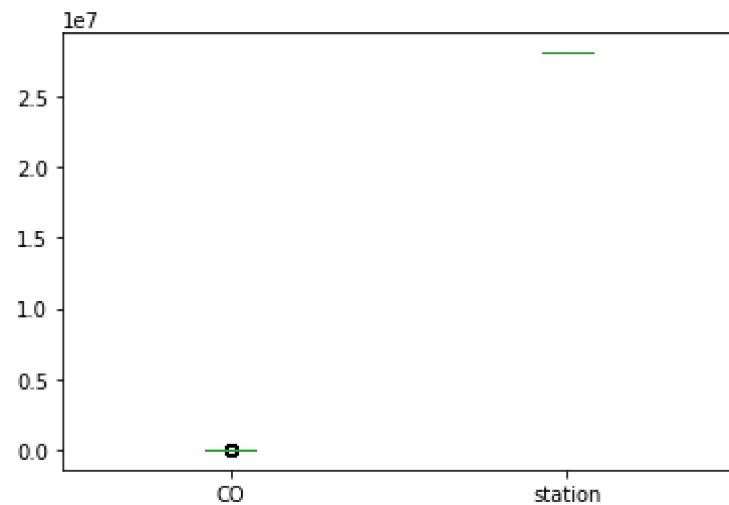
```
Out[12]: <AxesSubplot:>
```



Box chart

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

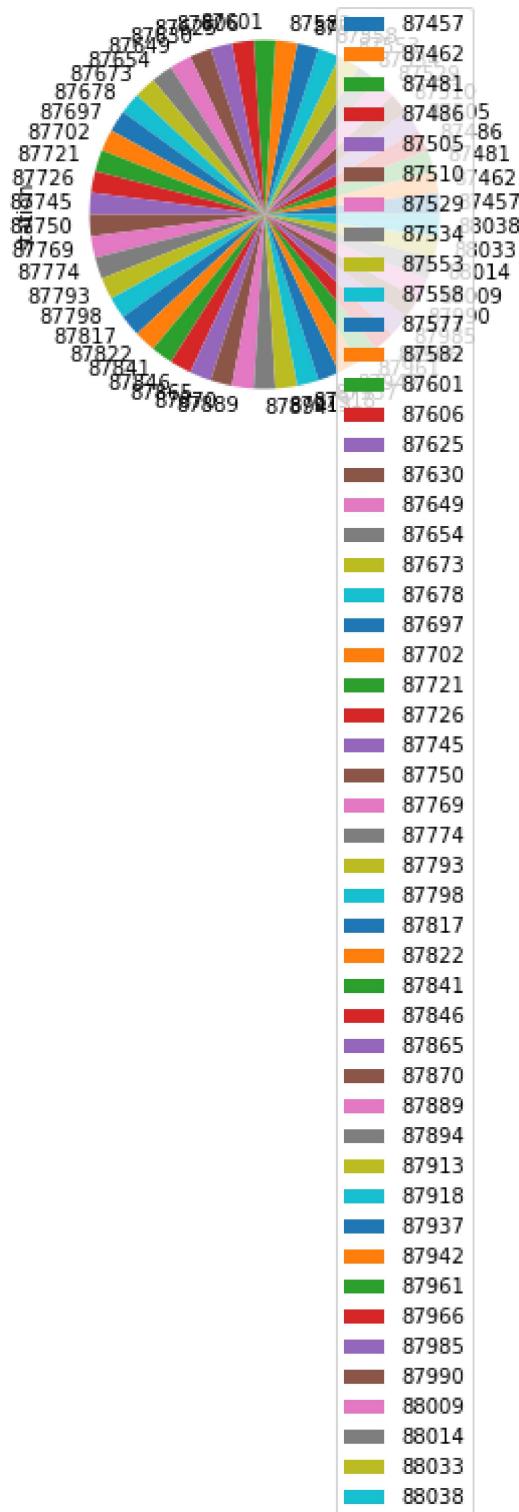
Out[13]: <AxesSubplot:>



Pie chart

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

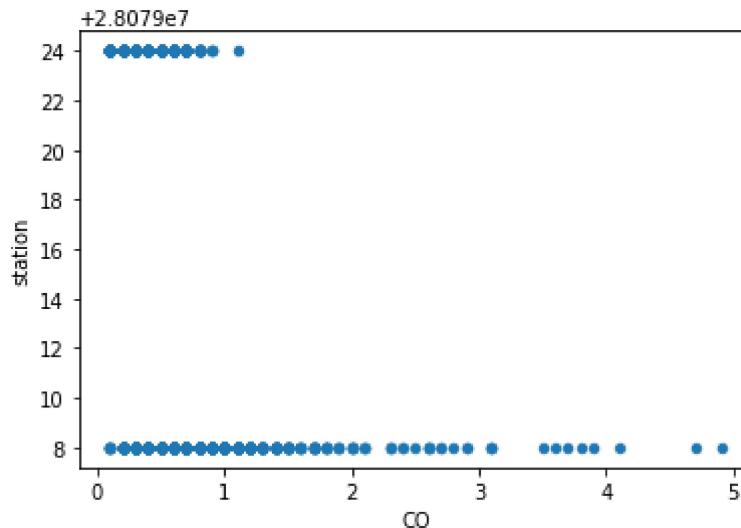
```
Out[14]: <AxesSubplot:ylabel='station'>
```



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]:

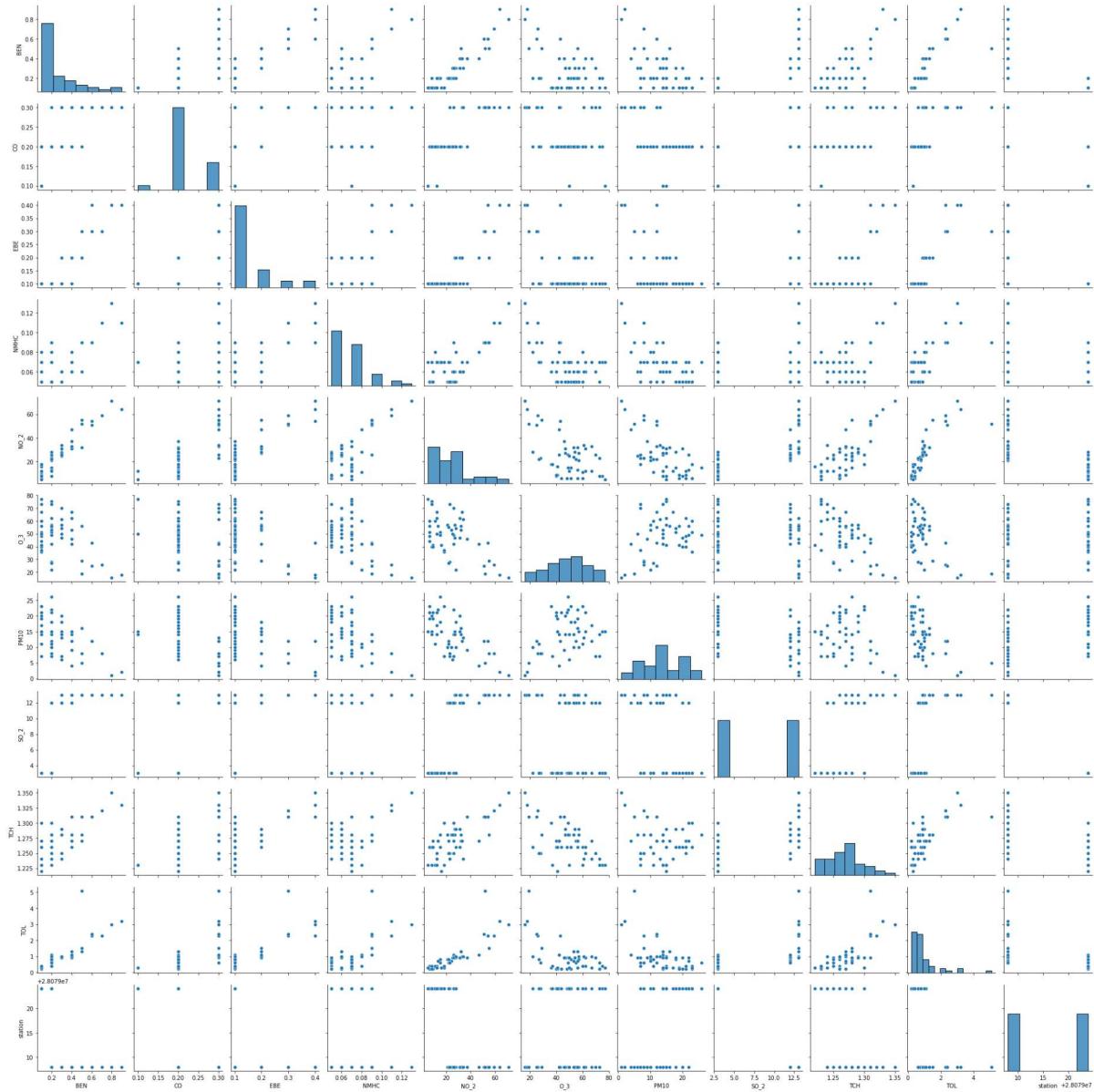
	BEN	CH4	CO	EBE	NMHC	NO	NO_2
count	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000
mean	0.919918	1.323732	0.417858	0.578168	0.097269	41.785316	58.069050
std	1.123078	0.215742	0.342871	0.962000	0.094035	71.118499	38.974150
min	0.100000	1.100000	0.100000	0.100000	0.000000	1.000000	1.000000
25%	0.300000	1.180000	0.200000	0.100000	0.050000	3.000000	30.000000
50%	0.600000	1.270000	0.300000	0.300000	0.080000	16.000000	54.000000
75%	1.100000	1.400000	0.500000	0.700000	0.110000	50.000000	78.000000
max	19.600000	3.630000	4.900000	16.700001	1.420000	879.000000	349.000000

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])
```

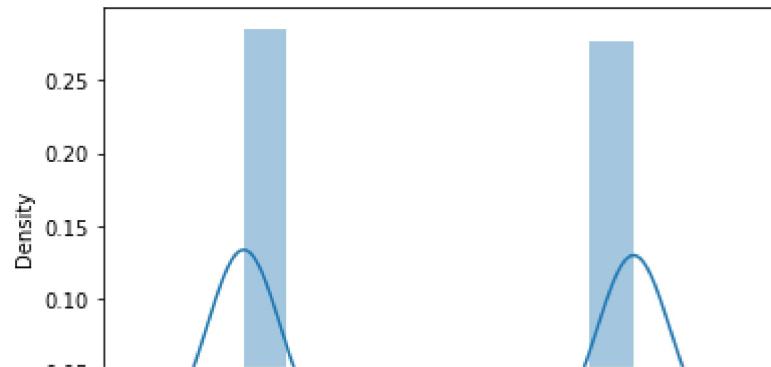
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x184521fe580>
```



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

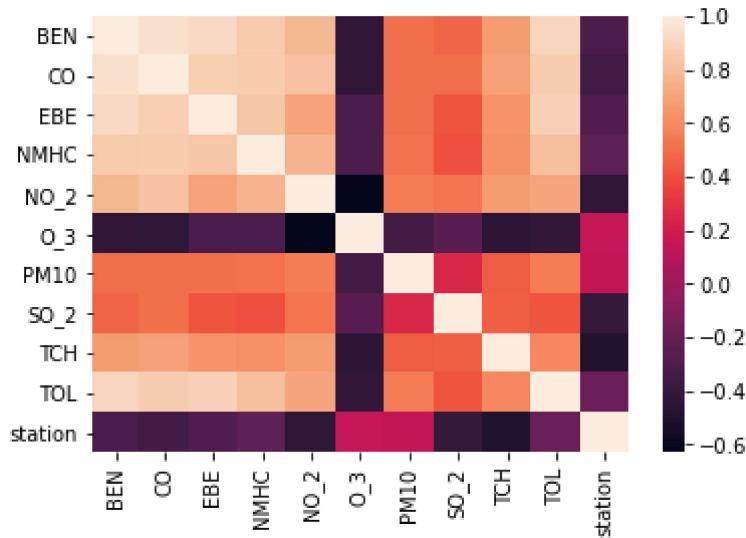
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]: 28079039.775936596
```

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

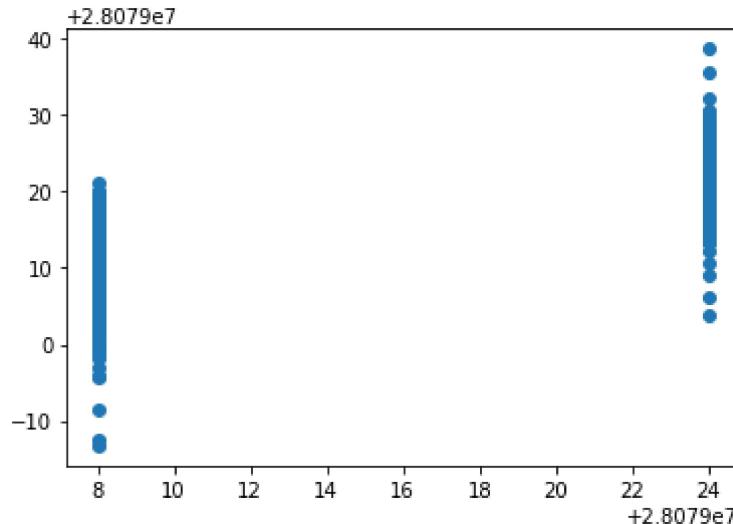
```
Out[26]:
```

Co-efficient

BEN	1.389736
CO	-2.774439
EBE	-2.643883
NMHC	35.811462
NO_2	-0.159580
O_3	-0.080293
PM10	0.345301
SO_2	-0.246303
TCH	-13.976820
TOL	0.198283

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

```
Out[27]: <matplotlib.collections.PathCollection at 0x1845a806910>
```



ACCURACY

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

```
Out[28]: 0.6144352073523935
```

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

```
Out[29]: 0.6003841386200994
```

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]: 0.5935258508848913
```

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

```
Out[33]: 0.584004470302783
```

```
In [34]: la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
Out[34]: Lasso(alpha=10)
```

Accuracy(Lasso)

```
In [35]: la.score(x_train,y_train)
```

```
Out[35]: 0.4028095622379414
```

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

```
Out[36]: 0.38886891479576713
```

ElasticNet

```
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.          ,
 -0.05406819,   0.32032273, -0.35514433, -0.          ,
  0.10605332])
```

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

```
Out[39]: 28079022.78698512
```

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

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

```
Out[41]: 0.4473007719821226
```

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)))
```

```
5.17635055562546  
35.29788609277114  
5.941202411361789
```

Logistic Regression

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

```
In [70]: feature_matrix=df[['BEN', 'CO', 'EBE','NMHC', 'NO_2','O_3',  
'PM10', 'SO_2', 'TCH', 'TOL']]  
target_vector=df[ 'station']
```

```
In [71]: feature_matrix.shape
```

```
Out[71]: (4127, 10)
```

```
In [72]: target_vector.shape
```

```
Out[72]: (4127,)
```

```
In [73]: from sklearn.preprocessing import StandardScaler
```

```
In [74]: fs=StandardScaler().fit_transform(feature_matrix)
```

```
In [75]: logr=LogisticRegression(max_iter=10000)  
logr.fit(fs,target_vector)
```

```
Out[75]: LogisticRegression(max_iter=10000)
```

```
In [79]: observation=[[1,2,3,4,5,6,7,8,9,10]]
```

```
In [80]: prediction=logr.predict(observation)  
print(prediction)
```

```
[28079008]
```

```
In [81]: logr.classes_
```

```
Out[81]: array([28079008, 28079024], dtype=int64)
```

```
In [82]: logr.score(fs,target_vector)
```

```
Out[82]: 0.9437848315968016
```

```
In [83]: logr.predict_proba(observation)[0][0]
```

```
Out[83]: 0.9999999999725541
```

```
In [84]: logr.predict_proba(observation)
```

```
Out[84]: 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="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]: 0.9709141274238227
```

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

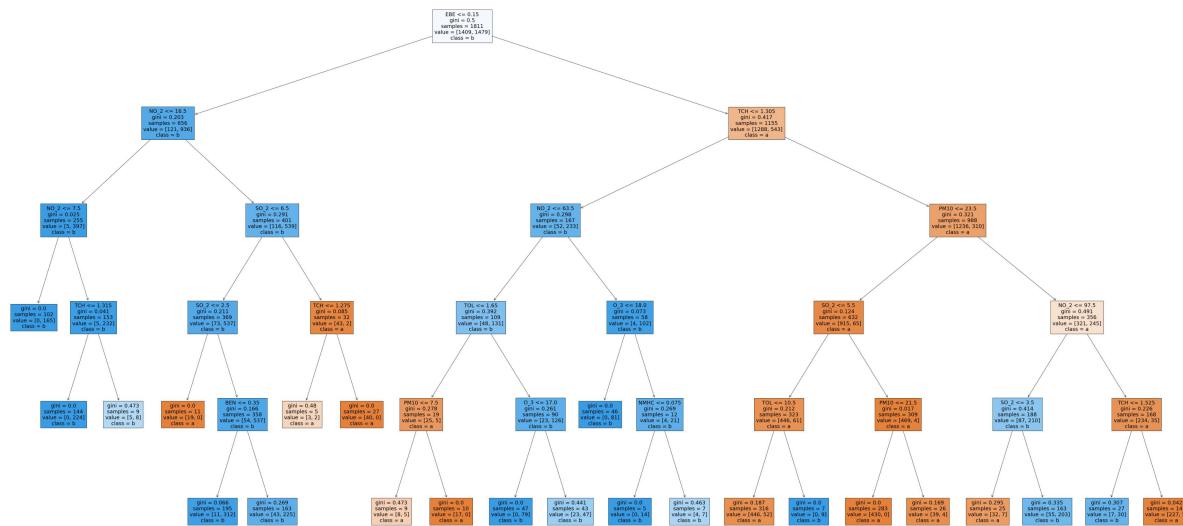
```
In [62]: from sklearn.tree import plot_tree  
  
plt.figure(figsize=(80,40))  
plot_tree(rfc_best.estimators_[5], feature_names=x.columns, class_names=['a', 'b'])
```

```
Out[62]: [Text(1715.85, 1993.2, 'EBE <= 0.15\ngini = 0.5\nsamples = 1811\nvalue = [140  
9, 1479]\nclass = b'),  
Text(613.8, 1630.8000000000002, 'NO_2 <= 16.5\ngini = 0.203\nsamples = 656\nvalue = [121, 936]\nclass = b'),  
Text(223.2, 1268.4, 'NO_2 <= 7.5\ngini = 0.025\nsamples = 255\nvalue = [5, 3  
97]\nclass = b'),  
Text(111.6, 906.0, 'gini = 0.0\nsamples = 102\nvalue = [0, 165]\nclass =  
b'),  
Text(334.7999999999995, 906.0, 'TCH <= 1.315\ngini = 0.041\nsamples = 153\nvalue = [5, 232]\nclass = b'),  
Text(223.2, 543.599999999999, 'gini = 0.0\nsamples = 144\nvalue = [0, 224]  
\nclass = b'),  
Text(446.4, 543.599999999999, 'gini = 0.473\nsamples = 9\nvalue = [5, 8]\nclass = b'),  
Text(1004.4, 1268.4, 'SO_2 <= 6.5\ngini = 0.291\nsamples = 401\nvalue = [11  
6, 539]\nclass = b'),  
Text(781.199999999999, 906.0, 'SO_2 <= 2.5\ngini = 0.211\nsamples = 369\nvalue = [73, 537]\nclass = b'),  
Text(669.599999999999, 543.599999999999, 'gini = 0.0\nsamples = 11\nvalue = [19, 0]\nclass = a'),  
Text(892.8, 543.599999999999, 'BEN <= 0.35\ngini = 0.166\nsamples = 358\nvalue = [54, 537]\nclass = b'),  
Text(781.199999999999, 181.1999999999982, 'gini = 0.066\nsamples = 195\nvalue = [11, 312]\nclass = b'),  
Text(1004.4, 181.1999999999982, 'gini = 0.269\nsamples = 163\nvalue = [43,  
225]\nclass = b'),  
Text(1227.6, 906.0, 'TCH <= 1.275\ngini = 0.085\nsamples = 32\nvalue = [43,  
2]\nclass = a'),  
Text(1116.0, 543.599999999999, 'gini = 0.48\nsamples = 5\nvalue = [3, 2]\nclass = a'),  
Text(1339.199999999998, 543.599999999999, 'gini = 0.0\nsamples = 27\nvalue = [40, 0]\nclass = a'),  
Text(2817.899999999996, 1630.800000000002, 'TCH <= 1.305\ngini = 0.417\nsamples = 1155\nvalue = [1288, 543]\nclass = a'),  
Text(2064.6, 1268.4, 'NO_2 <= 63.5\ngini = 0.298\nsamples = 167\nvalue = [5  
2, 233]\nclass = b'),  
Text(1785.6, 906.0, 'TOL <= 1.65\ngini = 0.392\nsamples = 109\nvalue = [48,  
131]\nclass = b'),  
Text(1562.399999999999, 543.599999999999, 'PM10 <= 7.5\ngini = 0.278\nsamples = 19\nvalue = [25, 5]\nclass = a'),  
Text(1450.8, 181.1999999999982, 'gini = 0.473\nsamples = 9\nvalue = [8, 5]  
\nclass = a'),  
Text(1674.0, 181.1999999999982, 'gini = 0.0\nsamples = 10\nvalue = [17, 0]  
\nclass = a'),  
Text(2008.8, 543.599999999999, 'O_3 <= 17.0\ngini = 0.261\nsamples = 90\nvalue = [23, 126]\nclass = b'),  
Text(1897.199999999998, 181.1999999999982, 'gini = 0.0\nsamples = 47\nvalue = [0, 79]\nclass = b'),  
Text(2120.4, 181.1999999999982, 'gini = 0.441\nsamples = 43\nvalue = [23, 4  
7]\nclass = b'),  
Text(2343.6, 906.0, 'O_3 <= 18.0\ngini = 0.073\nsamples = 58\nvalue = [4, 10  
2]\nclass = b'),  
Text(2232.0, 543.599999999999, 'gini = 0.0\nsamples = 46\nvalue = [0, 81]\nclass = b'),  
Text(2455.2, 543.599999999999, 'NMHC <= 0.075\ngini = 0.269\nsamples = 12\nvalue = [4, 21]\nclass = b'),  
Text(2343.6, 181.1999999999982, 'gini = 0.0\nsamples = 5\nvalue = [0, 14]\nclass = b')]
```

```

class = b'),
Text(2566.7999999999997, 181.1999999999982, 'gini = 0.463\nsamples = 7\nvalue = [4, 7]\nclass = b'),
Text(3571.2, 1268.4, 'PM10 <= 23.5\ngini = 0.321\nsamples = 988\nvalue = [1236, 310]\nclass = a'),
Text(3124.7999999999997, 906.0, 'SO_2 <= 5.5\ngini = 0.124\nsamples = 632\nvalue = [915, 65]\nclass = a'),
Text(2901.6, 543.5999999999999, 'TOL <= 10.5\ngini = 0.212\nsamples = 323\nvalue = [446, 61]\nclass = a'),
Text(2790.0, 181.1999999999982, 'gini = 0.187\nsamples = 316\nvalue = [446, 52]\nclass = a'),
Text(3013.2, 181.1999999999982, 'gini = 0.0\nsamples = 7\nvalue = [0, 9]\nclass = b'),
Text(3348.0, 543.5999999999999, 'PM10 <= 21.5\ngini = 0.017\nsamples = 309\nvalue = [469, 4]\nclass = a'),
Text(3236.3999999999996, 181.1999999999982, 'gini = 0.0\nsamples = 283\nvalue = [430, 0]\nclass = a'),
Text(3459.6, 181.1999999999982, 'gini = 0.169\nsamples = 26\nvalue = [39, 4]\nclass = a'),
Text(4017.6, 906.0, 'NO_2 <= 97.5\ngini = 0.491\nsamples = 356\nvalue = [321, 245]\nclass = a'),
Text(3794.3999999999996, 543.5999999999999, 'SO_2 <= 3.5\ngini = 0.414\nsamples = 188\nvalue = [87, 210]\nclass = b'),
Text(3682.7999999999997, 181.1999999999982, 'gini = 0.295\nsamples = 25\nvalue = [32, 7]\nclass = a'),
Text(3906.0, 181.1999999999982, 'gini = 0.335\nsamples = 163\nvalue = [55, 203]\nclass = b'),
Text(4240.8, 543.5999999999999, 'TCH <= 1.525\ngini = 0.226\nsamples = 168\nvalue = [234, 35]\nclass = a'),
Text(4129.2, 181.1999999999982, 'gini = 0.307\nsamples = 27\nvalue = [7, 30]\nclass = b'),
Text(4352.4, 181.1999999999982, 'gini = 0.042\nsamples = 141\nvalue = [227, 5]\nclass = a')]

```



Accuracy

Linear Regression

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

```
Out[63]: 0.6003841386200994
```

Ridge Regression

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

```
Out[64]: 0.584004470302783
```

Lasso Regression

```
In [65]: la.score(x_test,y_test)
```

```
Out[65]: 0.38886891479576713
```

ElasticNet Regression

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

```
Out[66]: 0.4473007719821226
```

Logistic Regression

```
In [85]: logr.score(fs,target_vector)
```

```
Out[85]: 0.9437848315968016
```

Random Forest

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

```
Out[68]: 0.9709141274238227
```

Conclusion

RandomForest for this dataset

