

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_2016.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN 2
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4 2
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0 2
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN 2
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN 2
...
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN 2
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN 2
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN 2
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN 2
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN 2

209496 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', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16932 entries, 1 to 209478
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      16932 non-null   object 
 1   BEN        16932 non-null   float64
 2   CO         16932 non-null   float64
 3   EBE        16932 non-null   float64
 4   NMHC       16932 non-null   float64
 5   NO         16932 non-null   float64
 6   NO_2       16932 non-null   float64
 7   O_3        16932 non-null   float64
 8   PM10       16932 non-null   float64
 9   PM25       16932 non-null   float64
 10  SO_2       16932 non-null   float64
 11  TCH        16932 non-null   float64
 12  TOL        16932 non-null   float64
 13  station    16932 non-null   int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.9+ MB
```

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

Out[6]:

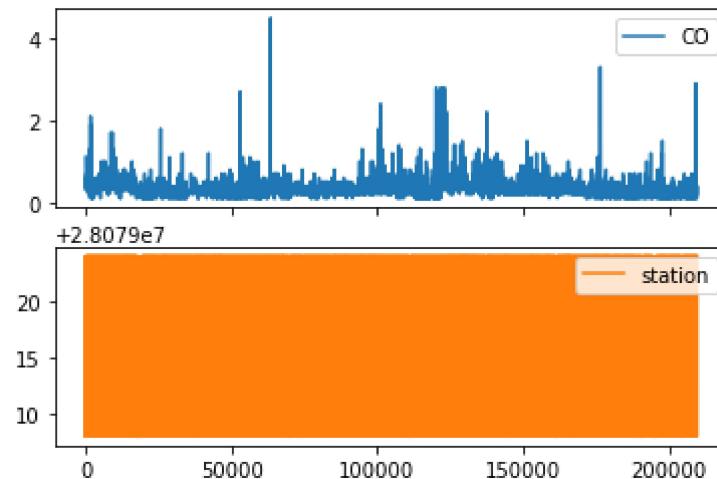
	CO	station
1	1.1	28079008
6	0.8	28079024
25	1.0	28079008
30	0.7	28079024
49	0.8	28079008
...
209430	0.2	28079024
209449	0.4	28079008
209454	0.2	28079024
209473	0.4	28079008
209478	0.2	28079024

16932 rows × 2 columns

Line chart

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

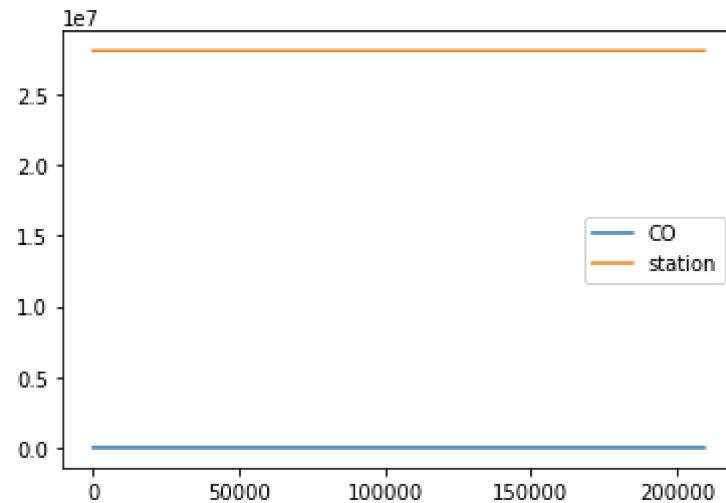
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

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

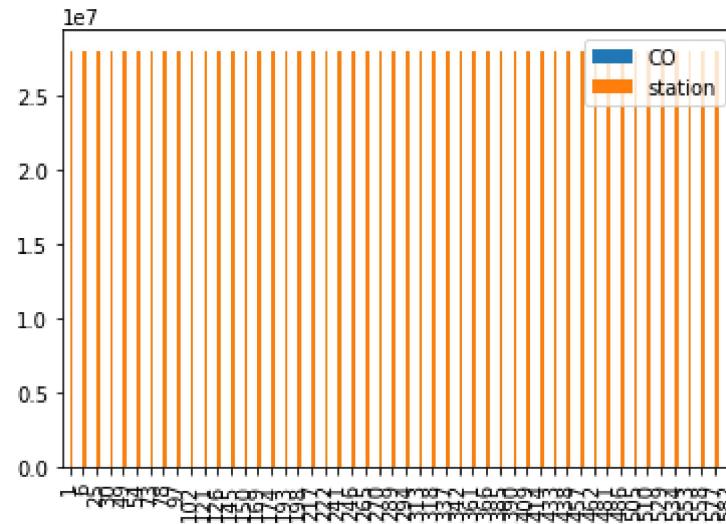


Bar chart

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

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

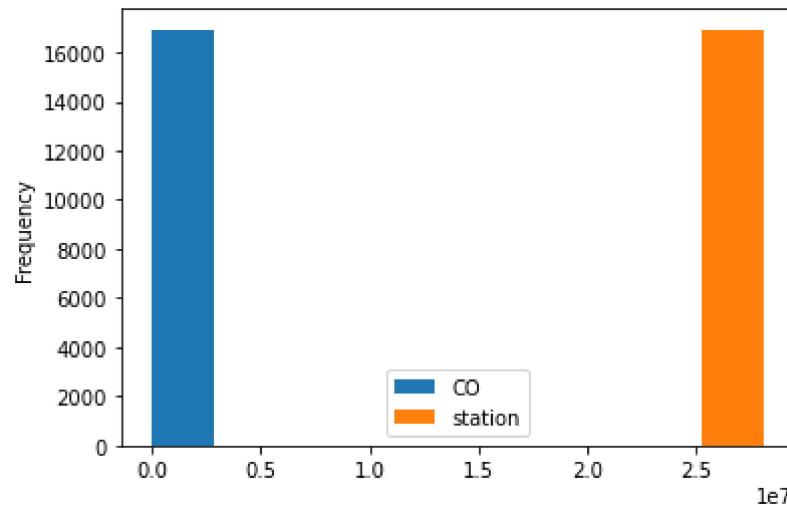
```
Out[10]: <AxesSubplot:>
```



Histogram

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

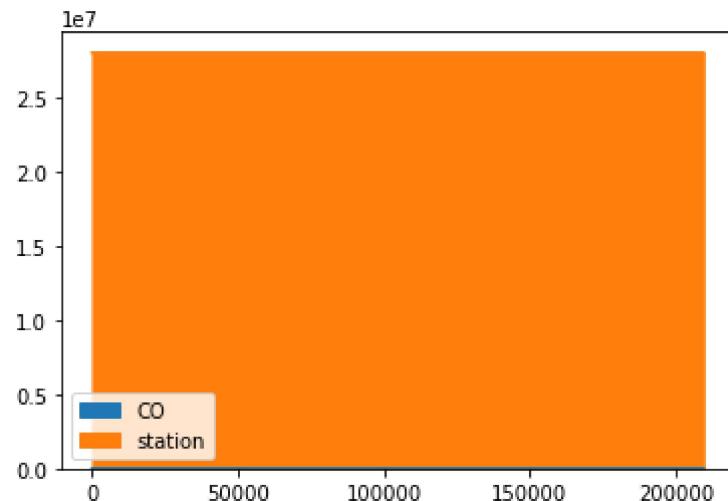
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

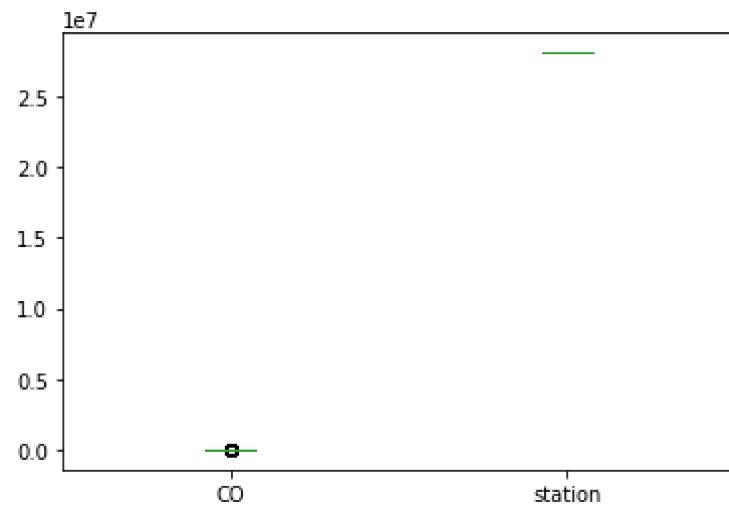
```
Out[12]: <AxesSubplot:>
```



Box chart

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

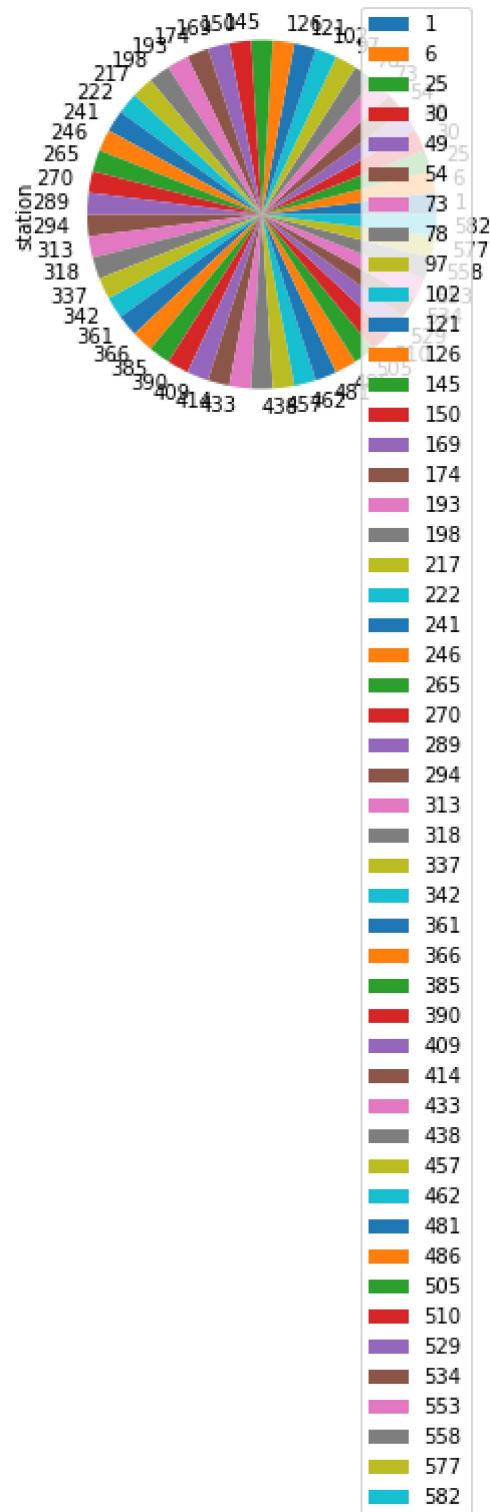
Out[13]: <AxesSubplot:>



Pie chart

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

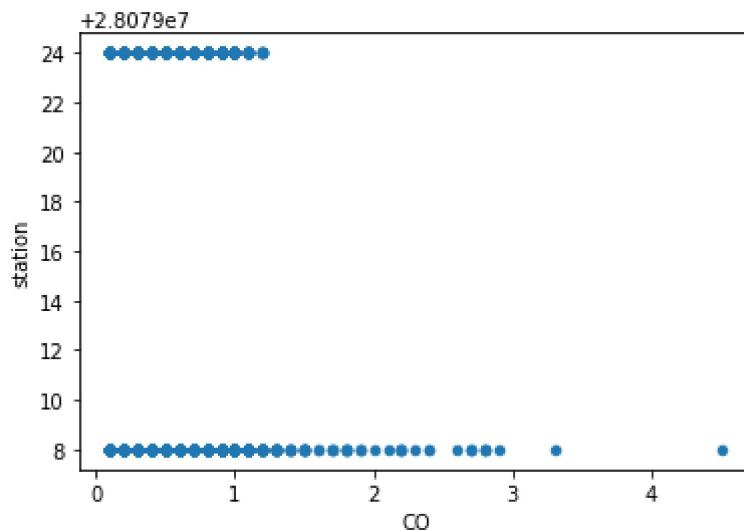
```
Out[14]: <AxesSubplot:ylabel='station'>
```



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: 16932 entries, 1 to 209478
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      16932 non-null   object 
 1   BEN       16932 non-null   float64
 2   CO        16932 non-null   float64
 3   EBE       16932 non-null   float64
 4   NMHC      16932 non-null   float64
 5   NO        16932 non-null   float64
 6   NO_2      16932 non-null   float64
 7   O_3       16932 non-null   float64
 8   PM10      16932 non-null   float64
 9   PM25      16932 non-null   float64
 10  SO_2      16932 non-null   float64
 11  TCH       16932 non-null   float64
 12  TOL       16932 non-null   float64
 13  station   16932 non-null   int64
```

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

Out[17]:

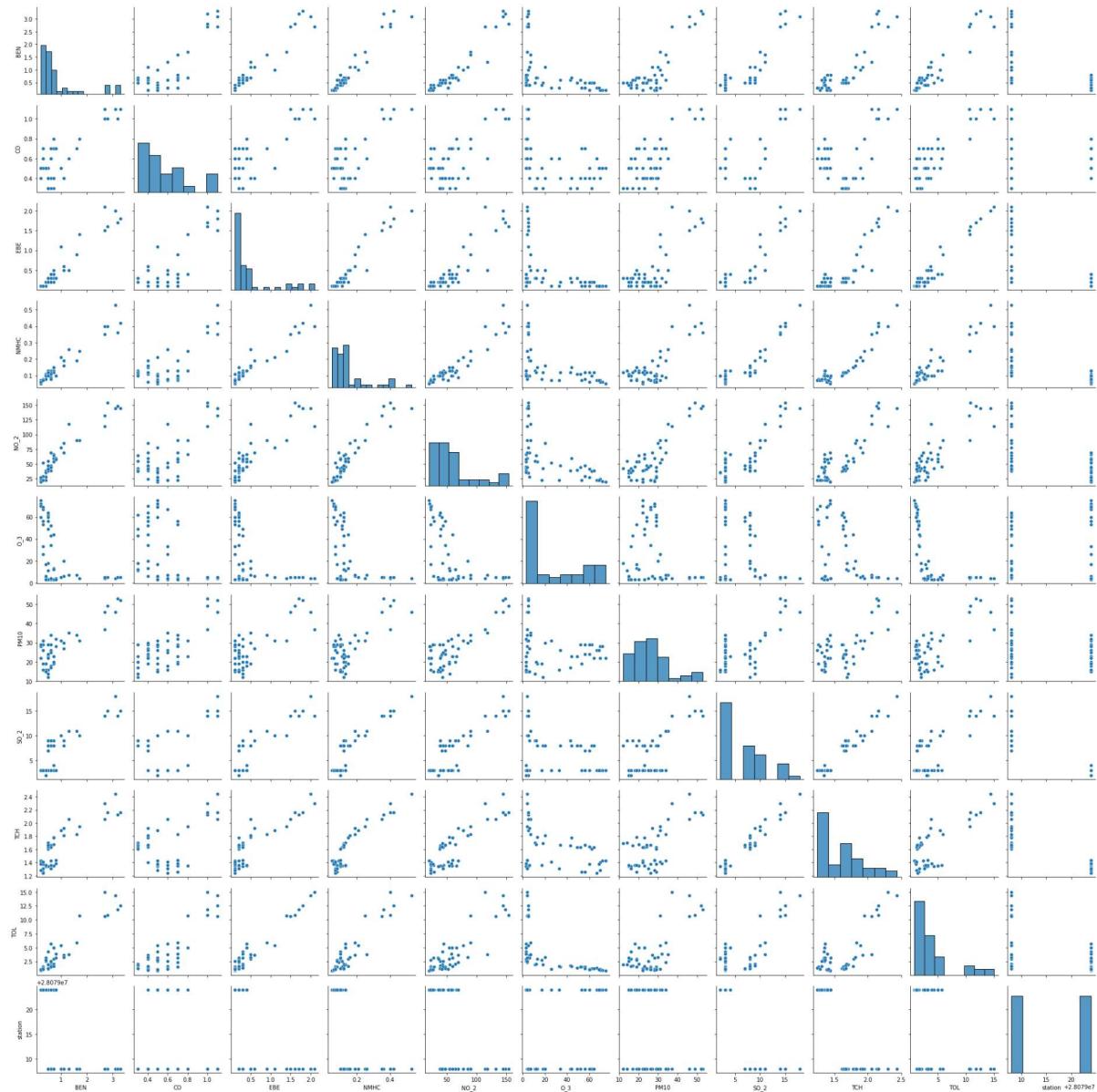
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	169
mean	0.537970	0.349941	0.298955	0.099913	20.815734	39.373376	
std	0.599479	0.203807	0.450204	0.079850	40.986063	31.170307	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	
25%	0.200000	0.200000	0.100000	0.050000	1.000000	14.000000	
50%	0.400000	0.300000	0.200000	0.090000	7.000000	34.000000	
75%	0.700000	0.400000	0.300000	0.120000	23.000000	58.000000	
max	12.300000	4.500000	13.500000	2.210000	829.000000	319.000000	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])
```

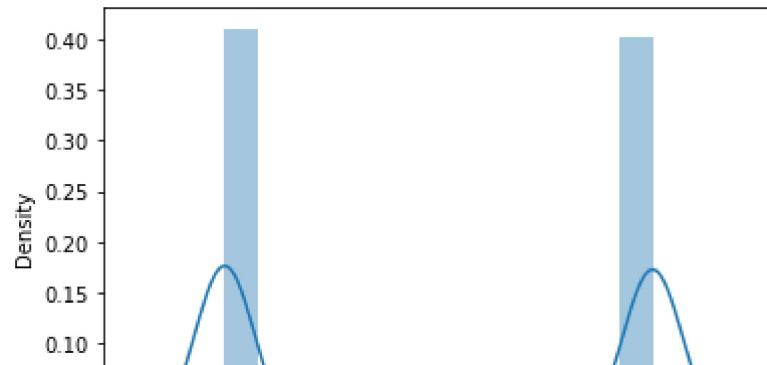
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x2077032a490>
```



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

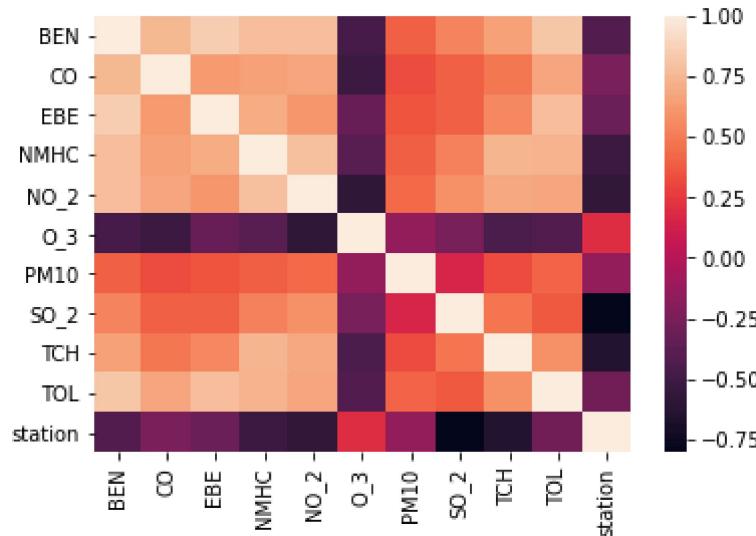
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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', 'NO2', 'O3', 'PM10', 'SO2', '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]: 28079039.943278283
```

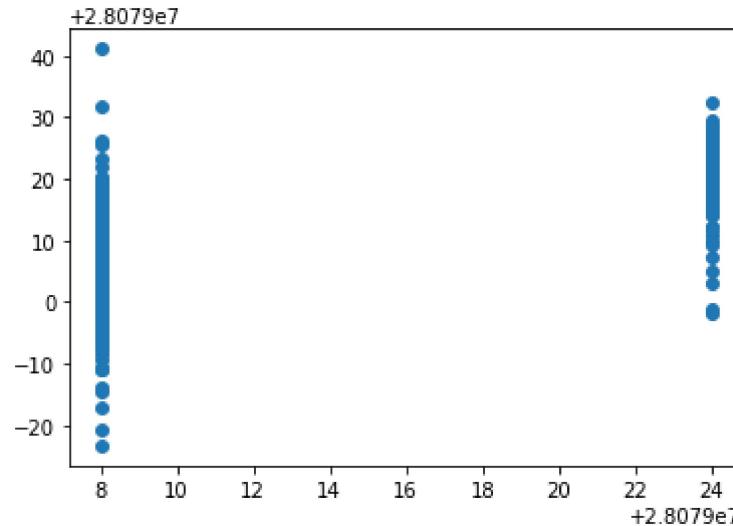
```
In [26]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[26]:
```

	Co-efficient
BEN	1.522051
CO	6.628144
EBE	0.064413
NMHC	6.997822
NO_2	-0.072747
O_3	-0.027678
PM10	0.027025
SO_2	-0.843950
TCH	-13.034847
TOL	0.246269

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

```
Out[27]: <matplotlib.collections.PathCollection at 0x20779849bb0>
```



ACCURACY

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

```
Out[28]: 0.78758209218309
```

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

```
Out[29]: 0.797504987278778
```

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]: 0.7881826615847998
```

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

```
Out[33]: 0.7972435890975537
```

```
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]: 0.6211989444968682
```

Accuracy(Lasso)

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

```
Out[36]: 0.6109466015426752
```

```
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.          ,
 -0.02629724,  0.02695759, -0.84653295, -0.          ,
 0.25813861])
```

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

```
Out[39]: 28079025.90684084
```

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

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

```
Out[41]: 0.6733182949330822
```

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)))
```

```
3.544505280449136
```

```
20.901066729066564
```

```
4.571768446571475
```

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]: (16932, 10)
```

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

```
Out[46]: (16932,)
```

```
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]: 0.9923812898653437
```

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

```
Out[54]: 1.0
```

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

```
Out[55]: array([[1.0000000e+00, 1.6336121e-46]])
```

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_
```

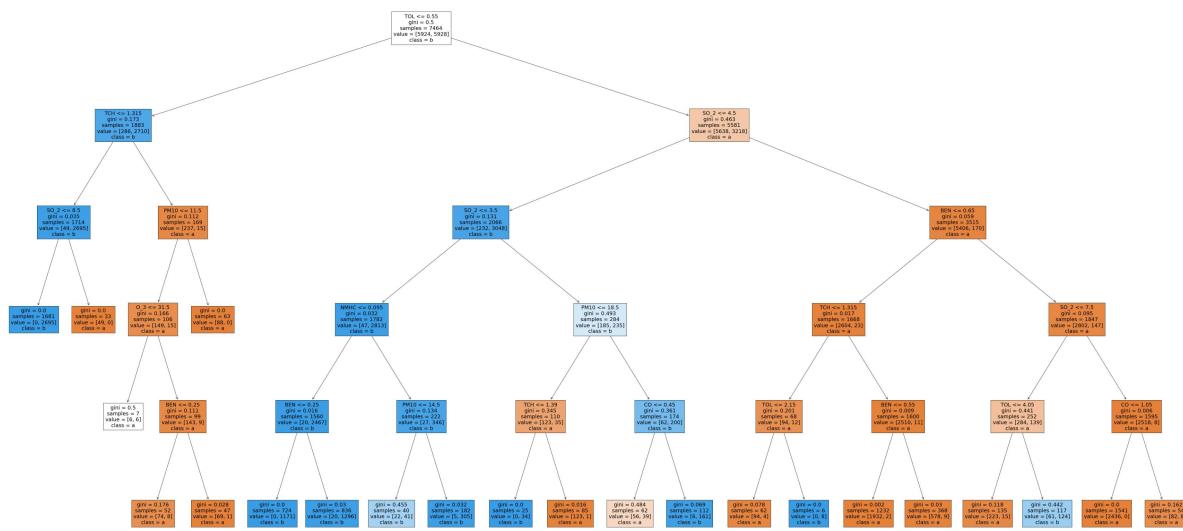
```
Out[60]: 0.9956125548430645
```

```
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(1562.399999999999, 1993.2, 'TOL <= 0.55\nngini = 0.5\nsamples = 7464\nvalue = [5924, 5928]\nnclass = b'),  
Text(446.4, 1630.800000000002, 'TCH <= 1.315\nngini = 0.173\nsamples = 1883\nvalue = [286, 2710]\nnclass = b'),  
Text(223.2, 1268.4, 'SO_2 <= 8.5\nngini = 0.035\nsamples = 1714\nvalue = [49, 2695]\nnclass = b'),  
Text(111.6, 906.0, 'gini = 0.0\nsamples = 1681\nvalue = [0, 2695]\nnclass = b'),  
Text(334.7999999999995, 906.0, 'gini = 0.0\nsamples = 33\nvalue = [49, 0]\nnclass = a'),  
Text(669.5999999999999, 1268.4, 'PM10 <= 11.5\nngini = 0.112\nsamples = 169\nvalue = [237, 15]\nnclass = a'),  
Text(558.0, 906.0, 'O_3 <= 31.5\nngini = 0.166\nsamples = 106\nvalue = [149, 15]\nnclass = a'),  
Text(446.4, 543.599999999999, 'gini = 0.5\nsamples = 7\nvalue = [6, 6]\nnclass = a'),  
Text(669.5999999999999, 543.5999999999999, 'BEN <= 0.25\nngini = 0.111\nsamples = 99\nvalue = [143, 9]\nnclass = a'),  
Text(558.0, 181.1999999999982, 'gini = 0.176\nsamples = 52\nvalue = [74, 8]\nnclass = a'),  
Text(781.1999999999999, 181.1999999999982, 'gini = 0.028\nsamples = 47\nvalue = [69, 1]\nnclass = a'),  
Text(781.1999999999999, 906.0, 'gini = 0.0\nsamples = 63\nvalue = [88, 0]\nnclass = a'),  
Text(2678.399999999996, 1630.800000000002, 'SO_2 <= 4.5\nngini = 0.463\nsamples = 5581\nvalue = [5638, 3218]\nnclass = a'),  
Text(1785.6, 1268.4, 'SO_2 <= 3.5\nngini = 0.131\nsamples = 2066\nvalue = [232, 3048]\nnclass = b'),  
Text(1339.199999999998, 906.0, 'NMHC <= 0.095\nngini = 0.032\nsamples = 1782\nvalue = [47, 2813]\nnclass = b'),  
Text(1116.0, 543.599999999999, 'BEN <= 0.25\nngini = 0.016\nsamples = 1560\nvalue = [20, 2467]\nnclass = b'),  
Text(1004.4, 181.1999999999982, 'gini = 0.0\nsamples = 724\nvalue = [0, 1171]\nnclass = b'),  
Text(1227.6, 181.1999999999982, 'gini = 0.03\nsamples = 836\nvalue = [20, 1296]\nnclass = b'),  
Text(1562.399999999999, 543.599999999999, 'PM10 <= 14.5\nngini = 0.134\nsamples = 222\nvalue = [27, 346]\nnclass = b'),  
Text(1450.8, 181.1999999999982, 'gini = 0.455\nsamples = 40\nvalue = [22, 41]\nnclass = b'),  
Text(1674.0, 181.1999999999982, 'gini = 0.032\nsamples = 182\nvalue = [5, 305]\nnclass = b'),  
Text(2232.0, 906.0, 'PM10 <= 18.5\nngini = 0.493\nsamples = 284\nvalue = [185, 235]\nnclass = b'),  
Text(2008.8, 543.599999999999, 'TCH <= 1.39\nngini = 0.345\nsamples = 110\nvalue = [123, 35]\nnclass = a'),  
Text(1897.199999999998, 181.1999999999982, 'gini = 0.0\nsamples = 25\nvalue = [0, 34]\nnclass = b'),  
Text(2120.4, 181.1999999999982, 'gini = 0.016\nsamples = 85\nvalue = [123, 1]\nnclass = a'),  
Text(2455.2, 543.599999999999, 'CO <= 0.45\nngini = 0.361\nsamples = 174\nvalue = [62, 200]\nnclass = b'),  
Text(2343.6, 181.1999999999982, 'gini = 0.484\nsamples = 62\nvalue = [56, 39]\nnclass = a'),  
Text(2566.799999999997, 181.1999999999982, 'gini = 0.069\nsamples = 112\nvalue = [6, 161]\nnclass = b'),  
Text(3571.2, 1268.4, 'BEN <= 0.65\nngini = 0.059\nsamples = 3515\nvalue = [54
```

```
06, 170]\nclass = a'),  
    Text(3124.7999999999997, 906.0, 'TCH <= 1.315\ngini = 0.017\nsamples = 1668  
\nvalue = [2604, 23]\nclass = a'),  
    Text(2901.6, 543.5999999999999, 'TOL <= 2.15\ngini = 0.201\nsamples = 68\nva  
lue = [94, 12]\nclass = a'),  
    Text(2790.0, 181.1999999999982, 'gini = 0.078\nsamples = 62\nvalue = [94,  
4]\nclass = a'),  
    Text(3013.2, 181.1999999999982, 'gini = 0.0\nsamples = 6\nvalue = [0, 8]\nnc  
lass = b'),  
    Text(3348.0, 543.5999999999999, 'BEN <= 0.55\ngini = 0.009\nsamples = 1600\nn  
value = [2510, 11]\nclass = a'),  
    Text(3236.399999999996, 181.1999999999982, 'gini = 0.002\nsamples = 1232\nn  
value = [1932, 2]\nclass = a'),  
    Text(3459.6, 181.1999999999982, 'gini = 0.03\nsamples = 368\nvalue = [578,  
9]\nclass = a'),  
    Text(4017.6, 906.0, 'SO_2 <= 7.5\ngini = 0.095\nsamples = 1847\nvalue = [280  
2, 147]\nclass = a'),  
    Text(3794.399999999996, 543.5999999999999, 'TOL <= 4.05\ngini = 0.441\nnsamp  
les = 252\nvalue = [284, 139]\nclass = a'),  
    Text(3682.799999999997, 181.1999999999982, 'gini = 0.118\nsamples = 135\nv  
alue = [223, 15]\nclass = a'),  
    Text(3906.0, 181.1999999999982, 'gini = 0.442\nsamples = 117\nvalue = [61,  
124]\nclass = b'),  
    Text(4240.8, 543.5999999999999, 'CO <= 1.05\ngini = 0.006\nsamples = 1595\nv  
alue = [2518, 8]\nclass = a'),  
    Text(4129.2, 181.1999999999982, 'gini = 0.0\nsamples = 1541\nvalue = [2436,  
0]\nclass = a'),  
    Text(4352.4, 181.1999999999982, 'gini = 0.162\nsamples = 54\nvalue = [82,  
8]\nclass = a')]
```



Conclusion

Accuracy

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

```
Out[63]: 0.797504987278778
```

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

```
Out[64]: 0.7972435890975537
```

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

```
Out[65]: 0.6211989444968682
```

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

```
Out[66]: 0.6733182949330822
```

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

```
Out[67]: 0.9923812898653437
```

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

```
Out[68]: 0.9956125548430645
```

Random Forest is suitable for this dataset

```
In [ ]:
```