

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"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs(Dataset)\ma
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[['BEN', 'TOL', 'TCH']]  
data
```

Out[6]:

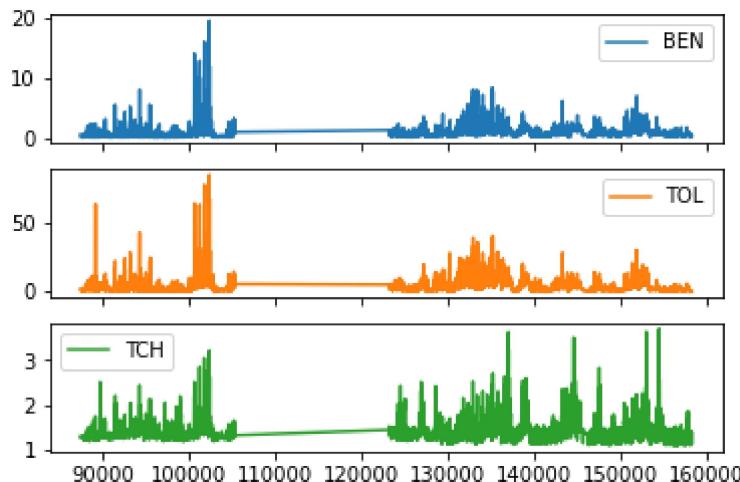
	BEN	TOL	TCH
87457	0.6	2.3	1.31
87462	0.2	1.1	1.27
87481	0.4	1.3	1.28
87486	0.2	0.8	1.26
87505	0.3	1.0	1.29
...
158238	0.3	0.2	1.14
158257	0.6	0.9	1.41
158262	0.3	0.2	1.14
158281	0.5	0.6	1.39
158286	0.3	0.2	1.14

4127 rows × 3 columns

Line chart

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

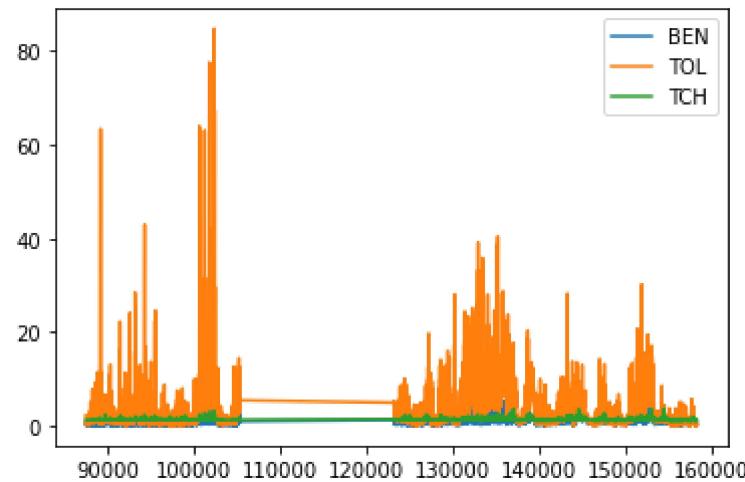
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

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

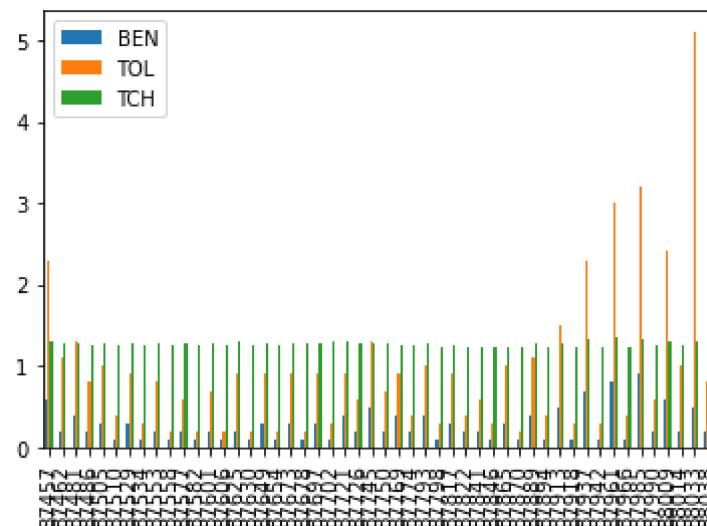


Bar chart

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

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

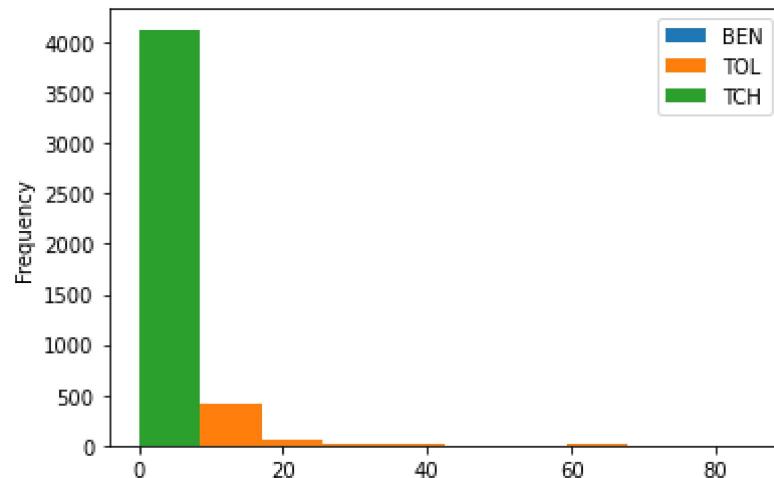
```
Out[10]: <AxesSubplot:>
```



Histogram

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

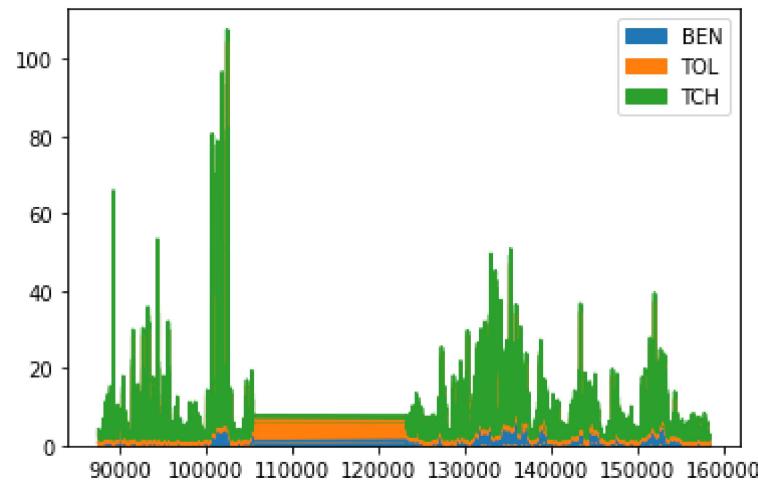
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

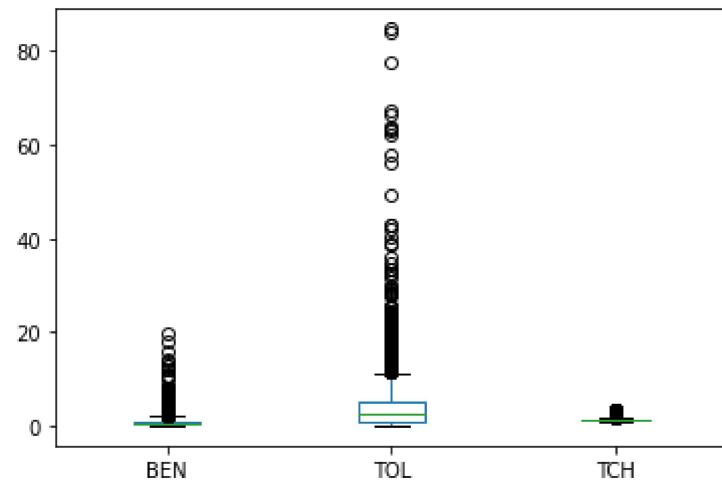
```
Out[12]: <AxesSubplot:>
```



Box chart

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

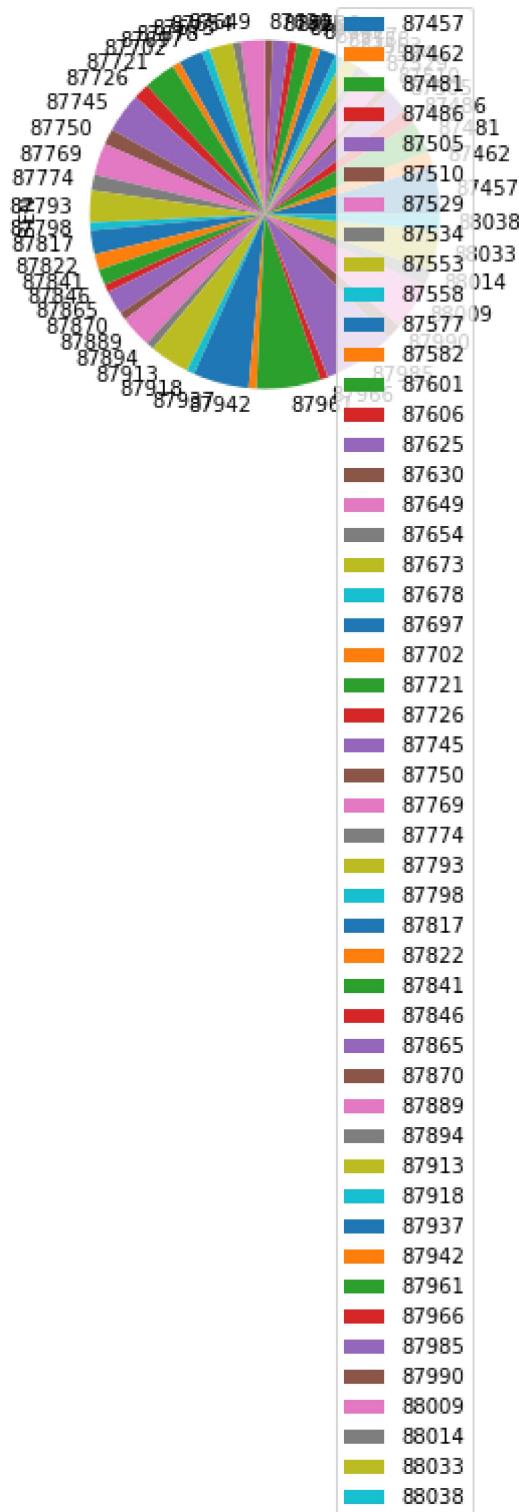
Out[13]: <AxesSubplot:>



Pie chart

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

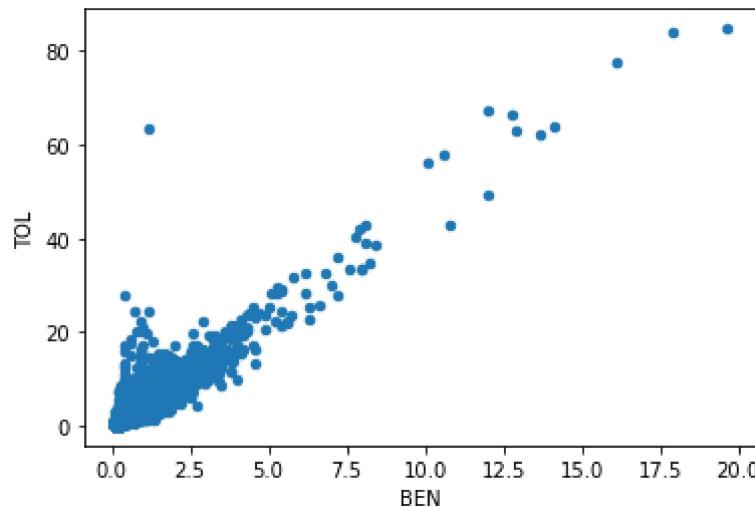
```
Out[14]: <AxesSubplot:ylabel='BEN'>
```



Scatter chart

```
In [15]: data.plot.scatter(x='BEN' ,y='TOL')
```

```
Out[15]: <AxesSubplot:xlabel='BEN', ylabel='TOL'>
```



```
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  TOL        4127 non-null   float64
 15  station    4127 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 548.1+ KB
```

In [17]: df.describe()

Out[17]:

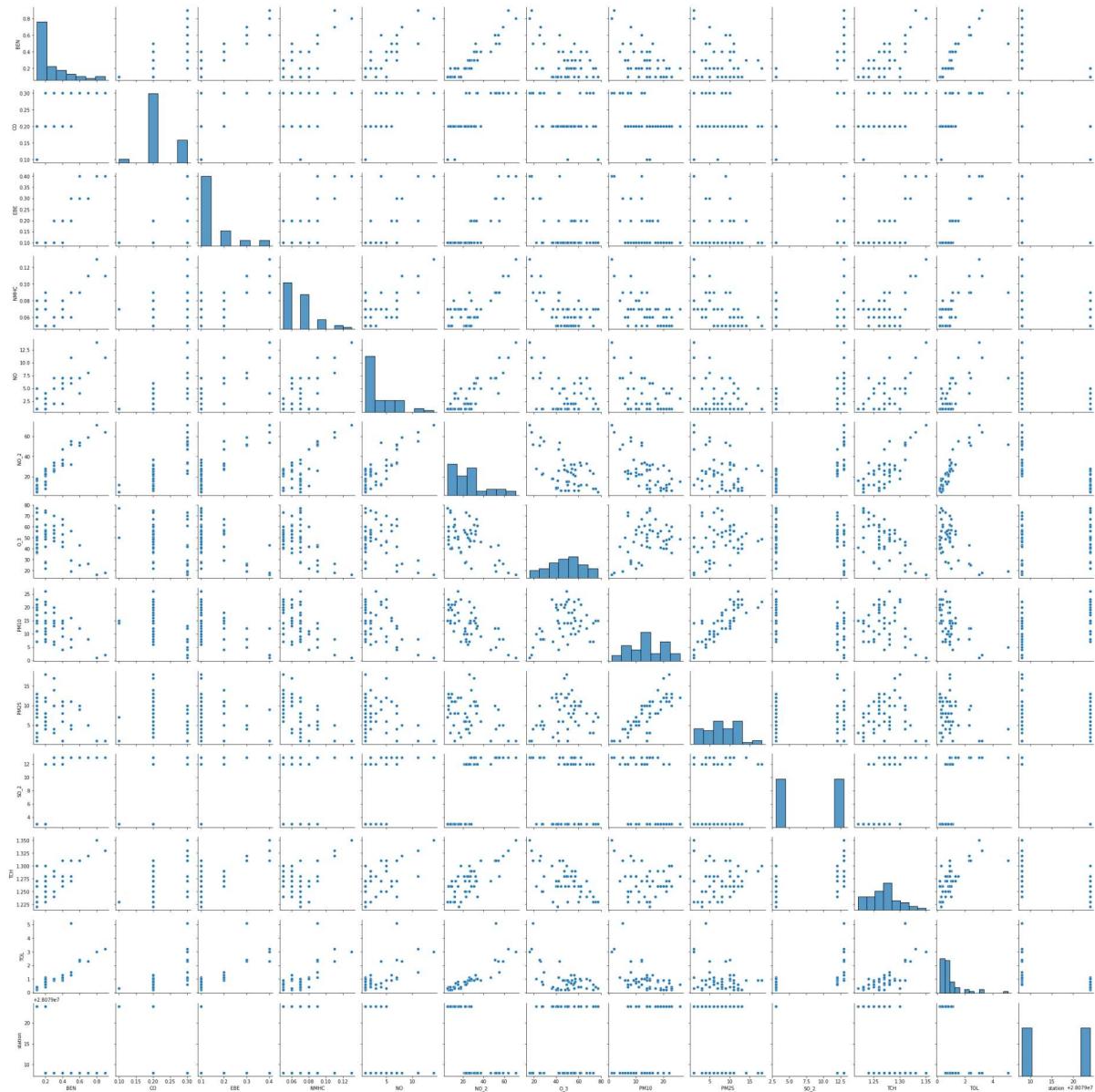
	BEN	CH4	CO	EBE	NMHC	NO	NO ₂
count	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000
mean	0.919918	1.323732	0.417858	0.578168	0.097269	41.785316	58.06905
std	1.123078	0.215742	0.342871	0.962000	0.094035	71.118499	38.9741
min	0.100000	1.100000	0.100000	0.100000	0.000000	1.000000	1.000000
25%	0.300000	1.180000	0.200000	0.100000	0.050000	3.000000	30.00000
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[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO₂', 'O₃', 'PM10', 'PM25', 'SO₂', 'TCH', 'TOL', 'station']]

EDA AND VISUALIZATION

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

```
Out[19]: <seaborn.axisgrid.PairGrid at 0x17eb5dbc220>
```

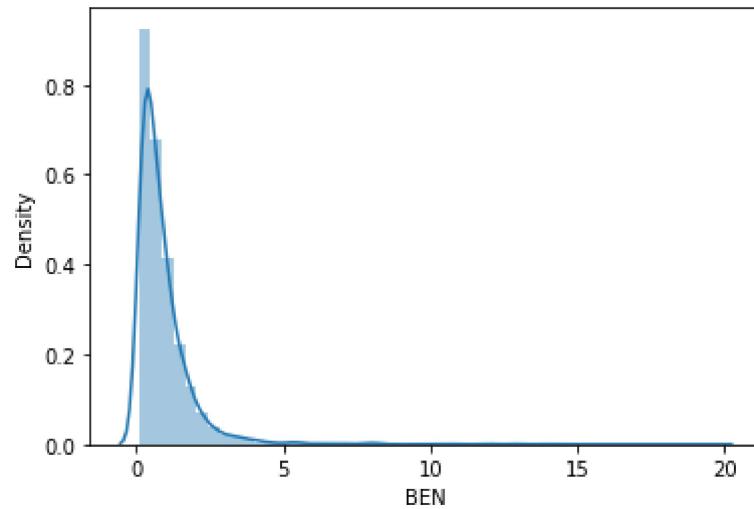


```
In [20]: sns.distplot(df1['BEN'])
```

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).

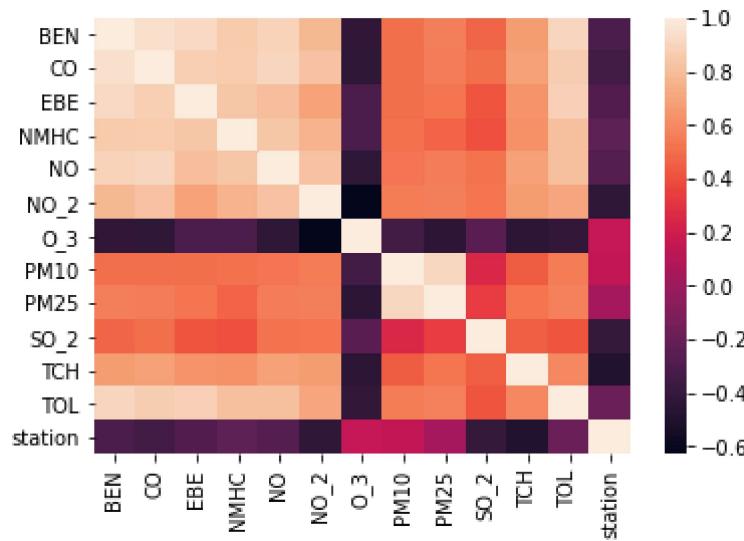
```
warnings.warn(msg, FutureWarning)
```

```
Out[20]: <AxesSubplot:xlabel='BEN', 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', 'NO_2', 'O_3', 'PM10', 'PM25',  
           '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]: 28079043.71596757
```

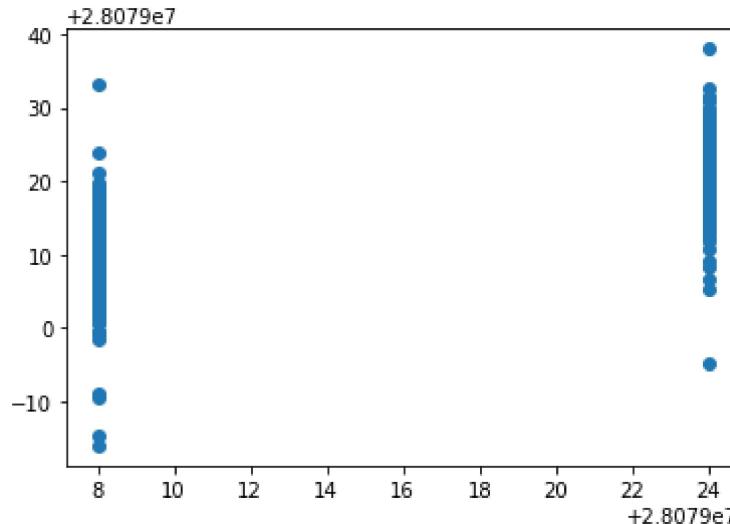
```
In [26]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[26]:
```

	Co-efficient
BEN	-0.131432
CO	-5.878558
EBE	-2.037298
NMHC	26.494987
NO	0.054856
NO_2	-0.179864
O_3	-0.089987
PM10	0.405238
PM25	-0.134231
SO_2	-0.246020
TCH	-14.958213
TOL	0.269782

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

```
Out[27]: <matplotlib.collections.PathCollection at 0x17ec1d3fd60>
```



ACCURACY

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

```
Out[28]: 0.6398945168135224
```

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

```
Out[29]: 0.6328582944214183
```

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]: 0.6333974711063022
```

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

```
Out[33]: 0.6234301862215097
```

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

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

```
In [35]: la.score(x_test,y_test)
```

```
Out[35]: 0.4213592202807822
```

Accuracy(Lasso)

```
In [36]: la.score(x_train,y_train)
```

```
Out[36]: 0.3936921139196006
```

Accuracy(Elastic Net)

```
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.          ,
 -0.2053017 , -0.08792252,  0.54764089, -0.41275344, -0.30528868,
 -0.          ,  0.          ])
```

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

```
Out[39]: 28079025.696305502
```

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

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

```
Out[41]: 0.5188724809294419
```

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)))
```

```
4.710186645608092  
30.685269693955842  
5.539428643276835
```

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="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_
```

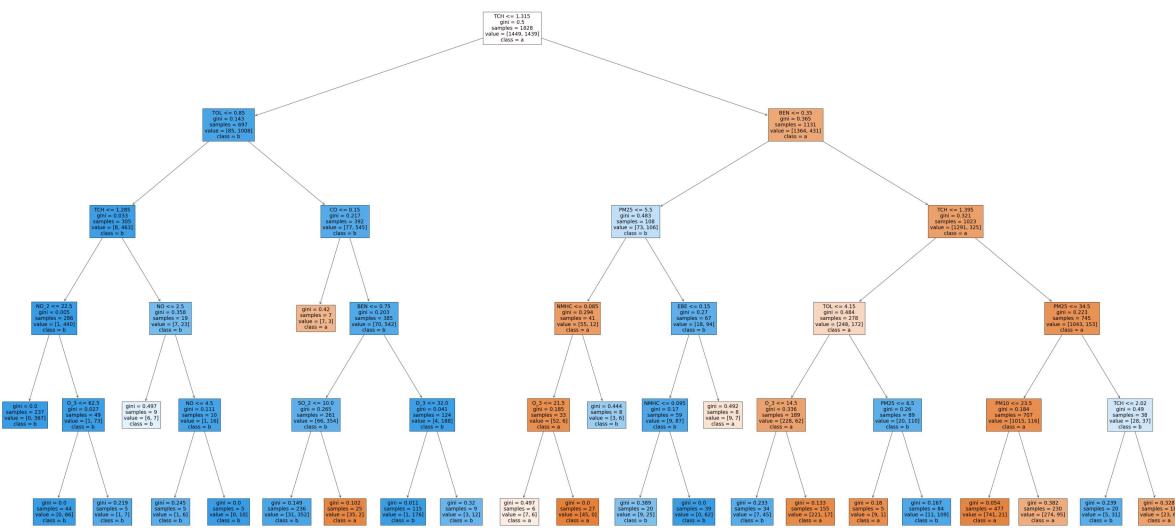
```
Out[60]: 0.971606648199446
```

```
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(1932.5853658536585, 1993.2, 'TCH <= 1.315\ngini = 0.5\nsamples = 1828\nvalue = [1449, 1439]\nclass = a'),  
Text(871.0243902439024, 1630.8000000000002, 'TOL <= 0.85\ngini = 0.143\nsamples = 697\nvalue = [85, 1008]\nclass = b'),  
Text(435.5121951219512, 1268.4, 'TCH <= 1.285\ngini = 0.033\nsamples = 305\nvalue = [8, 463]\nclass = b'),  
Text(217.7560975609756, 906.0, 'NO_2 <= 22.5\ngini = 0.005\nsamples = 286\nvalue = [1, 440]\nclass = b'),  
Text(108.8780487804878, 543.5999999999999, 'gini = 0.0\nsamples = 237\nvalue = [0, 367]\nclass = b'),  
Text(326.6341463414634, 543.5999999999999, 'O_3 <= 62.5\ngini = 0.027\nsamples = 49\nvalue = [1, 73]\nclass = b'),  
Text(217.7560975609756, 181.1999999999982, 'gini = 0.0\nsamples = 44\nvalue = [0, 66]\nclass = b'),  
Text(435.5121951219512, 181.1999999999982, 'gini = 0.219\nsamples = 5\nvalue = [1, 7]\nclass = b'),  
Text(653.2682926829268, 906.0, 'NO <= 2.5\ngini = 0.358\nsamples = 19\nvalue = [7, 23]\nclass = b'),  
Text(544.390243902439, 543.5999999999999, 'gini = 0.497\nsamples = 9\nvalue = [6, 7]\nclass = b'),  
Text(762.1463414634146, 543.5999999999999, 'NO <= 4.5\ngini = 0.111\nsamples = 10\nvalue = [1, 16]\nclass = b'),  
Text(653.2682926829268, 181.1999999999982, 'gini = 0.245\nsamples = 5\nvalue = [1, 6]\nclass = b'),  
Text(871.0243902439024, 181.1999999999982, 'gini = 0.0\nsamples = 5\nvalue = [0, 10]\nclass = b'),  
Text(1306.5365853658536, 1268.4, 'CO <= 0.15\ngini = 0.217\nsamples = 392\nvalue = [77, 545]\nclass = b'),  
Text(1197.6585365853657, 906.0, 'gini = 0.42\nsamples = 7\nvalue = [7, 3]\nclass = a'),  
Text(1415.4146341463415, 906.0, 'BEN <= 0.75\ngini = 0.203\nsamples = 385\nvalue = [70, 542]\nclass = b'),  
Text(1197.6585365853657, 543.5999999999999, 'SO_2 <= 10.0\ngini = 0.265\nsamples = 261\nvalue = [66, 354]\nclass = b'),  
Text(1088.780487804878, 181.1999999999982, 'gini = 0.149\nsamples = 236\nvalue = [31, 352]\nclass = b'),  
Text(1306.5365853658536, 181.1999999999982, 'gini = 0.102\nsamples = 25\nvalue = [35, 2]\nclass = a'),  
Text(1633.170731707317, 543.5999999999999, 'O_3 <= 32.0\ngini = 0.041\nsamples = 124\nvalue = [4, 188]\nclass = b'),  
Text(1524.2926829268292, 181.1999999999982, 'gini = 0.011\nsamples = 115\nvalue = [1, 176]\nclass = b'),  
Text(1742.0487804878048, 181.1999999999982, 'gini = 0.32\nsamples = 9\nvalue = [3, 12]\nclass = b'),  
Text(2994.1463414634145, 1630.8000000000002, 'BEN <= 0.35\ngini = 0.365\nsamples = 1131\nvalue = [1364, 431]\nclass = a'),  
Text(2395.3170731707314, 1268.4, 'PM25 <= 5.5\ngini = 0.483\nsamples = 108\nvalue = [73, 106]\nclass = b'),  
Text(2177.560975609756, 906.0, 'NMHC <= 0.085\ngini = 0.294\nsamples = 41\nvalue = [55, 12]\nclass = a'),  
Text(2068.682926829268, 543.5999999999999, 'O_3 <= 21.5\ngini = 0.185\nsamples = 33\nvalue = [52, 6]\nclass = a'),  
Text(1959.8048780487804, 181.1999999999982, 'gini = 0.497\nsamples = 6\nvalue = [7, 6]\nclass = a'),  
Text(2177.560975609756, 181.1999999999982, 'gini = 0.0\nsamples = 27\nvalue = [45, 0]\nclass = a'),  
Text(2286.439024390244, 543.5999999999999, 'gini = 0.444\nsamples = 8\nvalue = [45, 0]\nclass = a')]
```

```
= [3, 6]\nclass = b'),  
    Text(2613.0731707317073, 906.0, 'EBE <= 0.15\ngini = 0.27\nsamples = 67\nvalue = [18, 94]\nclass = b'),  
    Text(2504.1951219512193, 543.5999999999999, 'NMHC <= 0.095\ngini = 0.17\nsamples = 59\nvalue = [9, 87]\nclass = b'),  
    Text(2395.3170731707314, 181.1999999999982, 'gini = 0.389\nsamples = 20\nvalue = [9, 25]\nclass = b'),  
    Text(2613.0731707317073, 181.1999999999982, 'gini = 0.0\nsamples = 39\nvalue = [0, 62]\nclass = b'),  
    Text(2721.951219512195, 543.5999999999999, 'gini = 0.492\nsamples = 8\nvalue = [9, 7]\nclass = a'),  
    Text(3592.9756097560976, 1268.4, 'TCH <= 1.395\ngini = 0.321\nsamples = 1023\nvalue = [1291, 325]\nclass = a'),  
    Text(3157.4634146341464, 906.0, 'TOL <= 4.15\ngini = 0.484\nsamples = 278\nvalue = [248, 172]\nclass = a'),  
    Text(2939.7073170731705, 543.5999999999999, 'O_3 <= 14.5\ngini = 0.336\nsamples = 189\nvalue = [228, 62]\nclass = a'),  
    Text(2830.829268292683, 181.1999999999982, 'gini = 0.233\nsamples = 34\nvalue = [7, 45]\nclass = b'),  
    Text(3048.5853658536585, 181.1999999999982, 'gini = 0.133\nsamples = 155\nvalue = [221, 17]\nclass = a'),  
    Text(3375.2195121951218, 543.5999999999999, 'PM25 <= 6.5\ngini = 0.26\nsamples = 89\nvalue = [20, 110]\nclass = b'),  
    Text(3266.341463414634, 181.1999999999982, 'gini = 0.18\nsamples = 5\nvalue = [9, 1]\nclass = a'),  
    Text(3484.0975609756097, 181.1999999999982, 'gini = 0.167\nsamples = 84\nvalue = [11, 109]\nclass = b'),  
    Text(4028.487804878049, 906.0, 'PM25 <= 34.5\ngini = 0.223\nsamples = 745\nvalue = [1043, 153]\nclass = a'),  
    Text(3810.731707317073, 543.5999999999999, 'PM10 <= 23.5\ngini = 0.184\nsamples = 707\nvalue = [1015, 116]\nclass = a'),  
    Text(3701.8536585365855, 181.1999999999982, 'gini = 0.054\nsamples = 477\nvalue = [741, 21]\nclass = a'),  
    Text(3919.609756097561, 181.1999999999982, 'gini = 0.382\nsamples = 230\nvalue = [274, 95]\nclass = a'),  
    Text(4246.243902439024, 543.5999999999999, 'TCH <= 2.02\ngini = 0.49\ncounts = 38\nvalue = [28, 37]\nclass = b'),  
    Text(4137.365853658536, 181.1999999999982, 'gini = 0.239\ncounts = 20\nvalue = [5, 31]\nclass = b'),  
    Text(4355.121951219512, 181.1999999999982, 'gini = 0.328\ncounts = 18\nvalue = [23, 6]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression: 0.6328582944214183

Ridge Regression: 0.6234301862215097

Lasso Regression: 0.3936921139196006

ElasticNet Regression: 0.5188724809294419

Logistic Regression: 0.9437848315968016

Random Forest: 0.971606648199446

Random Forest is suitable for this dataset