

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	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3
2	2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1
3	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	28
4	2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN
...
210019	2014-09-01 00:00:00	NaN	0.5	NaN	NaN	20.0	84.0	29.0	NaN	NaN	NaN	NaN	28
210020	2014-09-01 00:00:00	NaN	0.3	NaN	NaN	1.0	22.0	NaN	15.0	NaN	6.0	NaN	NaN
210021	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	13.0	70.0	NaN	NaN	NaN	NaN	28
210022	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	38.0	42.0	NaN	NaN	NaN	NaN	28
210023	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	26.0	65.0	11.0	NaN	NaN	NaN	28

210024 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: 13946 entries, 1 to 210006
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   date      13946 non-null   object 
 1   BEN        13946 non-null   float64
 2   CO         13946 non-null   float64
 3   EBE        13946 non-null   float64
 4   NMHC       13946 non-null   float64
 5   NO         13946 non-null   float64
 6   NO_2       13946 non-null   float64
 7   O_3        13946 non-null   float64
 8   PM10       13946 non-null   float64
 9   PM25       13946 non-null   float64
 10  SO_2       13946 non-null   float64
 11  TCH        13946 non-null   float64
 12  TOL        13946 non-null   float64
 13  station    13946 non-null   int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.6+ MB
```

```
In [6]: data=df[['BEN', 'TOL', 'TCH']]  
data
```

Out[6]:

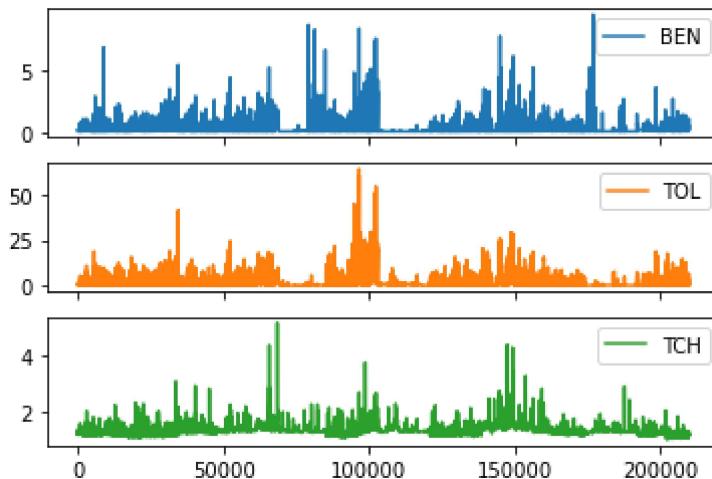
	BEN	TOL	TCH
1	0.2	1.3	1.36
6	0.1	0.1	1.21
25	0.2	0.8	1.36
30	0.2	0.1	1.21
49	0.1	0.9	1.36
...
209958	0.2	0.1	1.28
209977	1.1	6.5	1.27
209982	0.2	0.2	1.27
210001	0.6	4.1	1.19
210006	0.2	0.1	1.30

13946 rows × 3 columns

Line chart

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

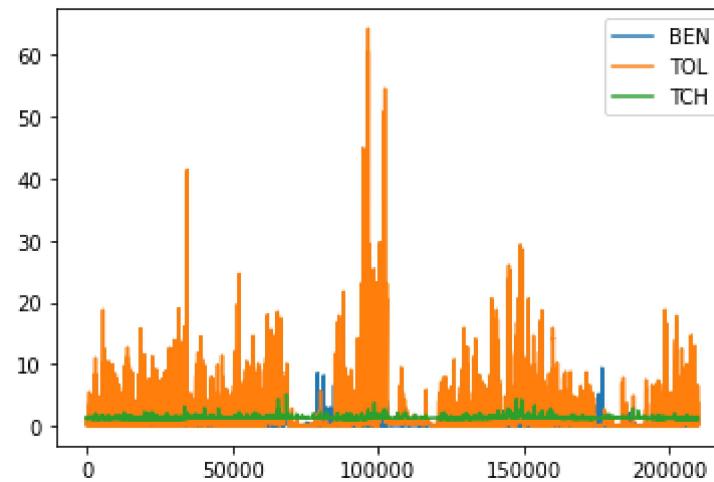
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

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

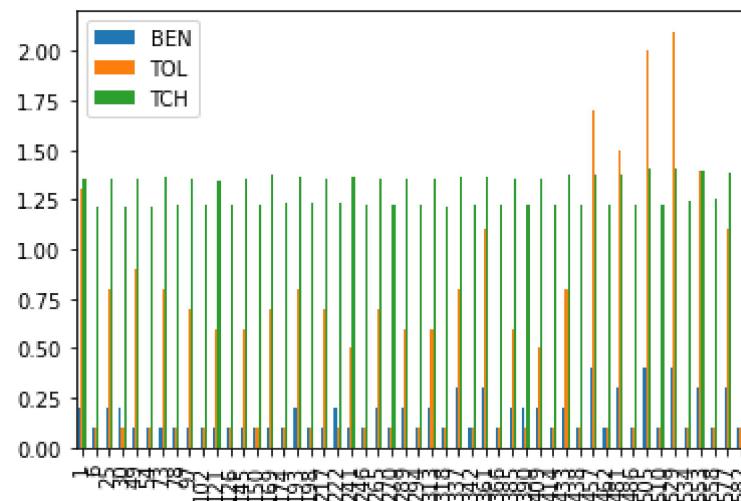


Bar chart

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

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

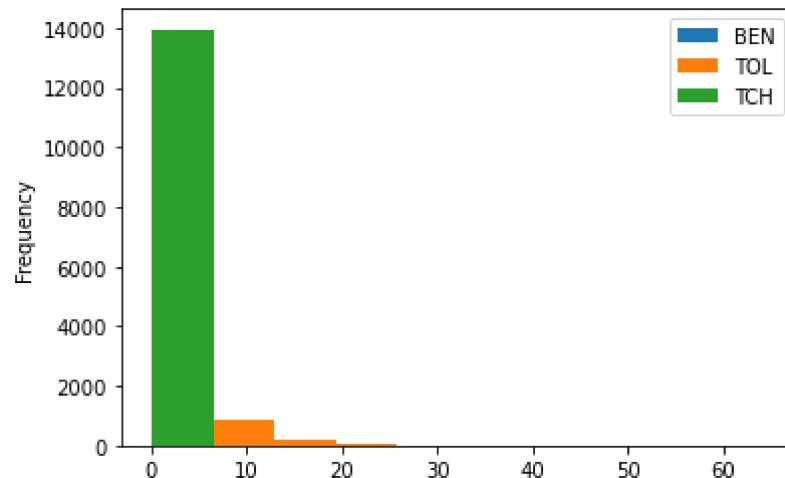
```
Out[10]: <AxesSubplot:>
```



Histogram

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

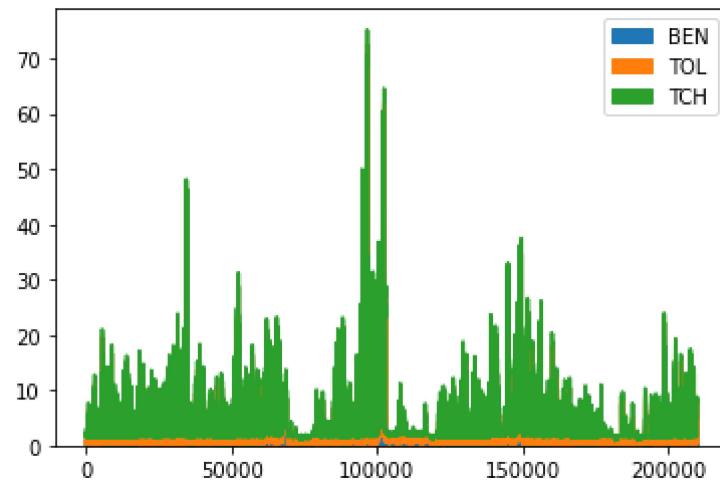
```
Out[11]: <AxesSubplot:ylabel='Frequency'>
```



Area chart

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

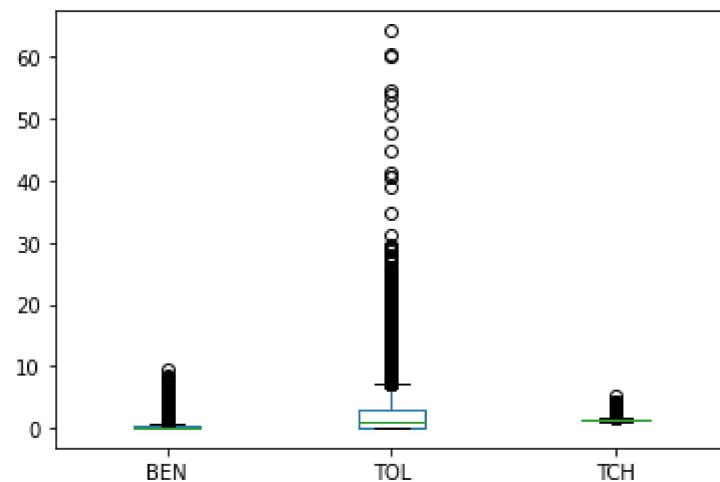
```
Out[12]: <AxesSubplot:>
```



Box chart

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

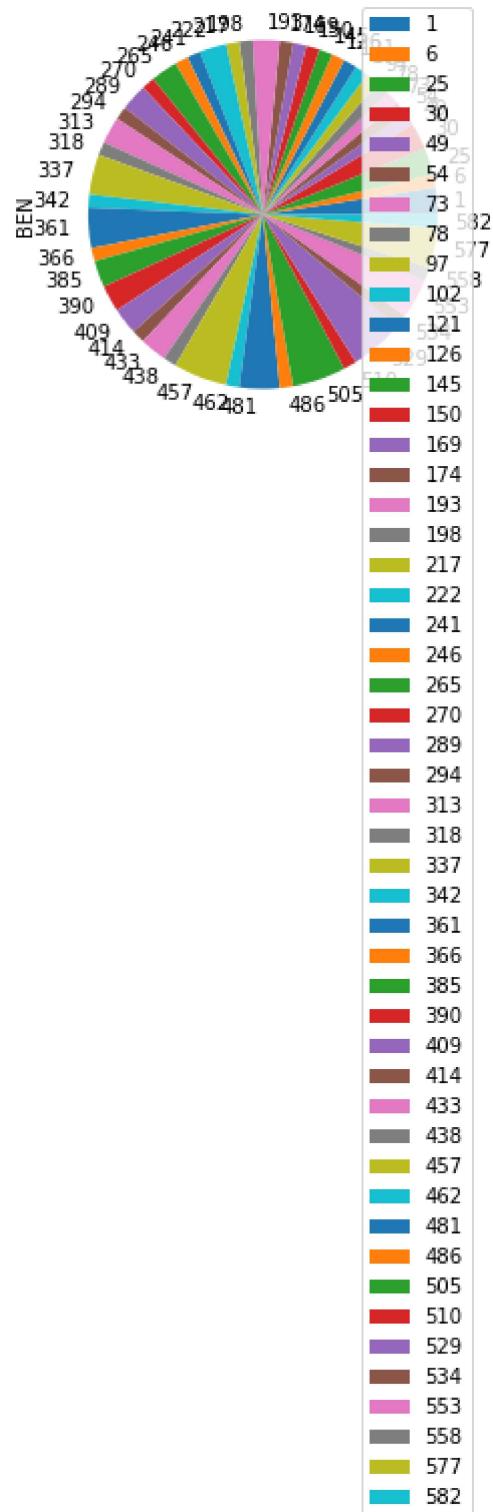
Out[13]: <AxesSubplot:>



Pie chart

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

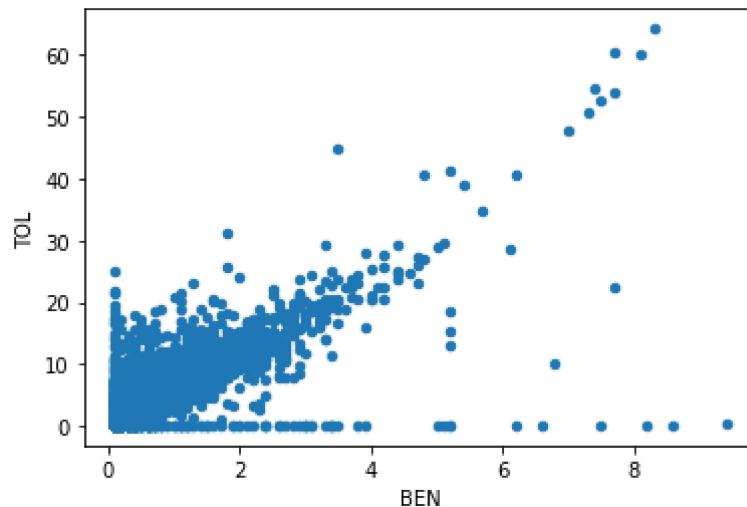
```
Out[14]: <AxesSubplot:ylabel='BEN'>
```



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: 13946 entries, 1 to 210006
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   date        13946 non-null   object 
 1   BEN         13946 non-null   float64
 2   CO          13946 non-null   float64
 3   EBE         13946 non-null   float64
 4   NMHC        13946 non-null   float64
 5   NO          13946 non-null   float64
 6   NO_2         13946 non-null   float64
 7   O_3          13946 non-null   float64
 8   PM10        13946 non-null   float64
 9   PM25        13946 non-null   float64
 10  SO_2         13946 non-null   float64
 11  TCH          13946 non-null   float64
 12  TOL          13946 non-null   float64
 13  station     13946 non-null   int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.6+ MB
```

In [17]: df.describe()

Out[17]:

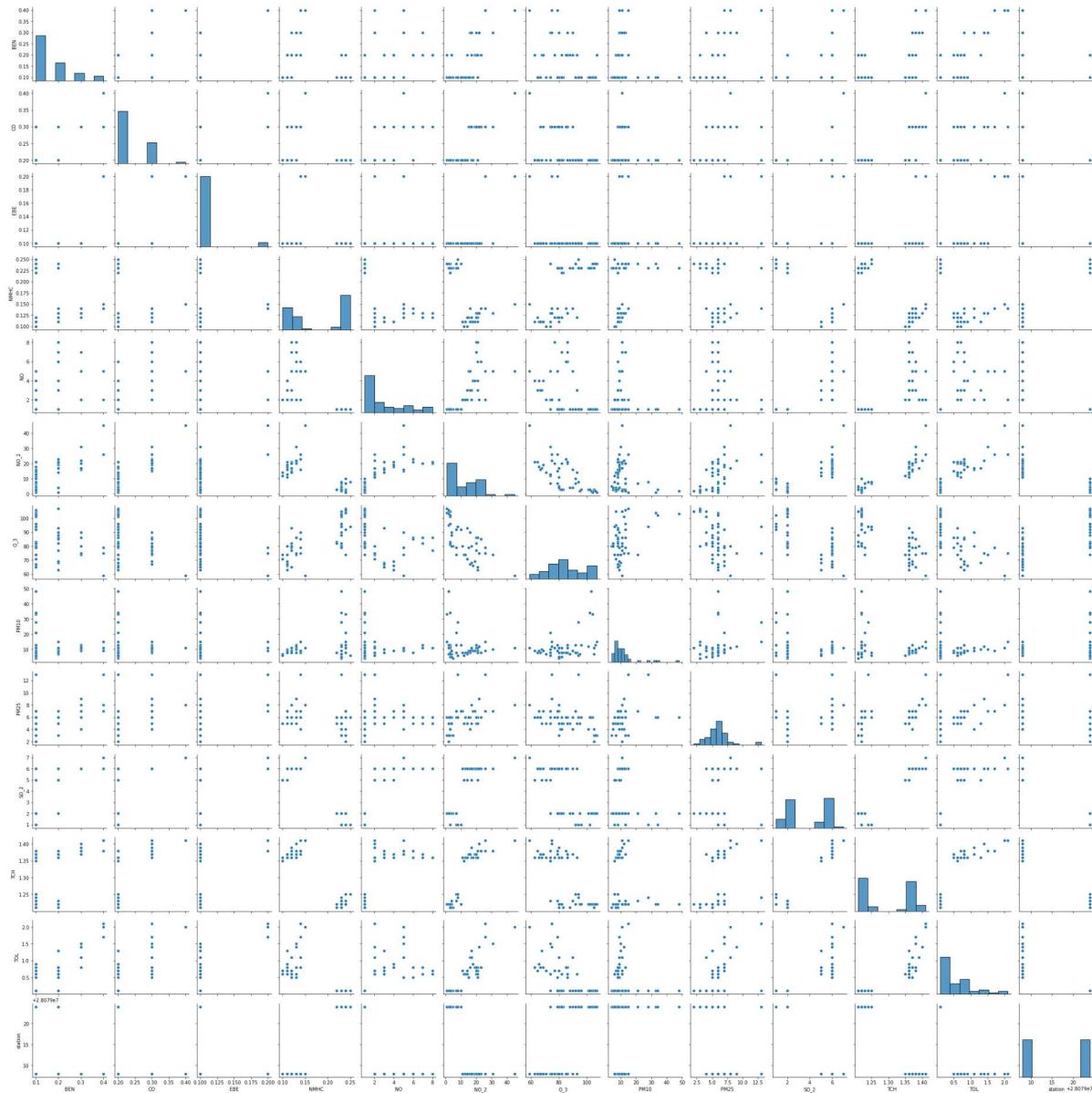
	BEN	CO	EBE	NMHC	NO	NO_2	
count	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000	139
mean	0.375921	0.314793	0.306016	0.222302	17.589129	34.240929	
std	0.555093	0.207375	0.635475	0.082403	39.432216	30.654229	
min	0.100000	0.100000	0.100000	0.060000	1.000000	1.000000	
25%	0.100000	0.200000	0.100000	0.160000	1.000000	10.000000	
50%	0.200000	0.300000	0.100000	0.230000	4.000000	27.000000	
75%	0.400000	0.400000	0.300000	0.260000	18.000000	51.000000	
max	9.400000	4.400000	16.200001	1.290000	725.000000	346.000000	2

In [18]: df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station']]

EDA AND VISUALIZATION

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

```
Out[19]: <seaborn.axisgrid.PairGrid at 0x22581899c10>
```

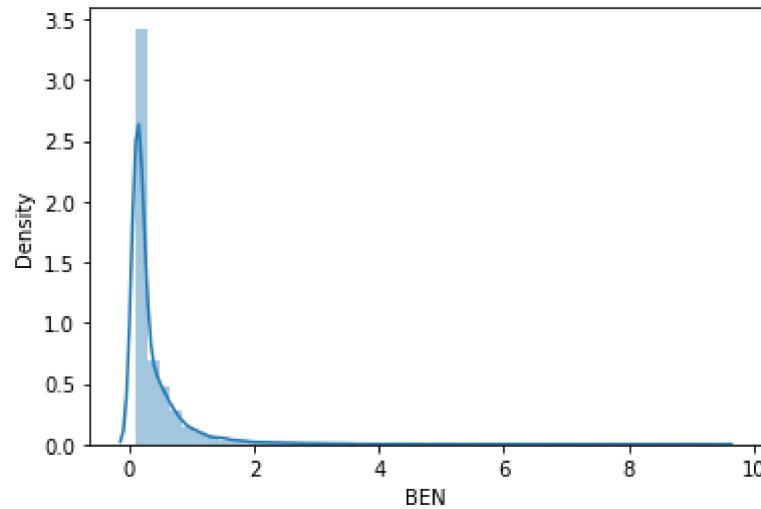


```
In [20]: sns.distplot(df1['BEN'])
```

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).

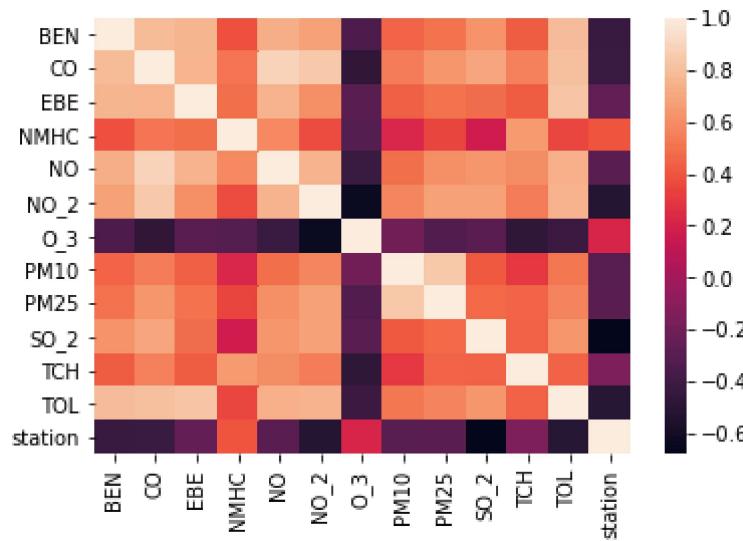
```
warnings.warn(msg, FutureWarning)
```

```
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]: 28079024.75309375
```

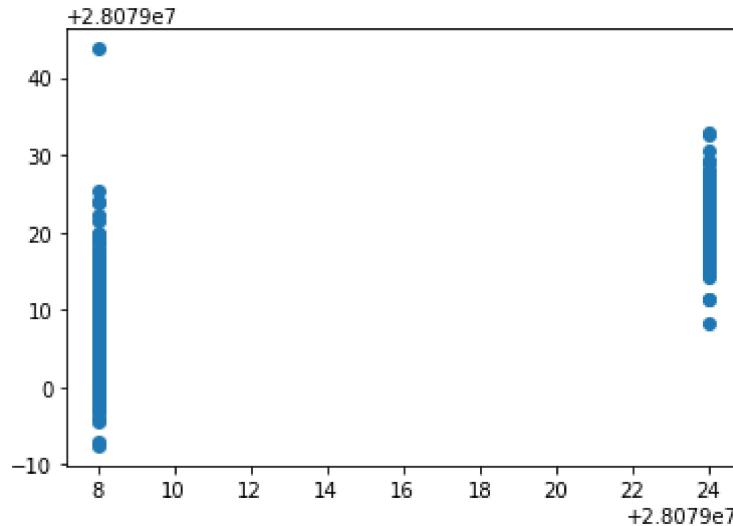
```
In [26]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[26]:
```

	Co-efficient
BEN	-1.526053
CO	-9.582083
EBE	0.169485
NMHC	81.813711
NO	0.027040
NO_2	-0.039069
O_3	0.001044
PM10	-0.021125
PM25	0.135383
SO_2	-0.886189
TCH	-12.883515
TOL	-0.399896

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

```
Out[27]: <matplotlib.collections.PathCollection at 0x2258d91ddf0>
```



ACCURACY

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

```
Out[28]: 0.8856068356416521
```

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

```
Out[29]: 0.8933824684760232
```

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]: 0.8685900527611394
```

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

```
Out[33]: 0.8700021760005732
```

```
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]: 0.2984542671759195
```

Accuracy(Lasso)

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

```
Out[36]: 0.29803877281876634
```

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.          ,
 -0.10952204, -0.01681879, -0.00967349,  0.09959627, -1.41820342,
 0.          , -0.5323858 ])
```

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

```
Out[39]: 28079026.65375127
```

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

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

```
Out[41]: 0.5963524316522861
```

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)))
```

```
4.1518782354627755
25.440529593890414
5.043860584303497
```

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]: (13946, 10)
```

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

```
Out[46]: (13946,)
```

```
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]: 0.9926143697117453
```

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

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

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

```
Out[55]: array([[1.0, 5.27113072e-18]])
```

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_
```

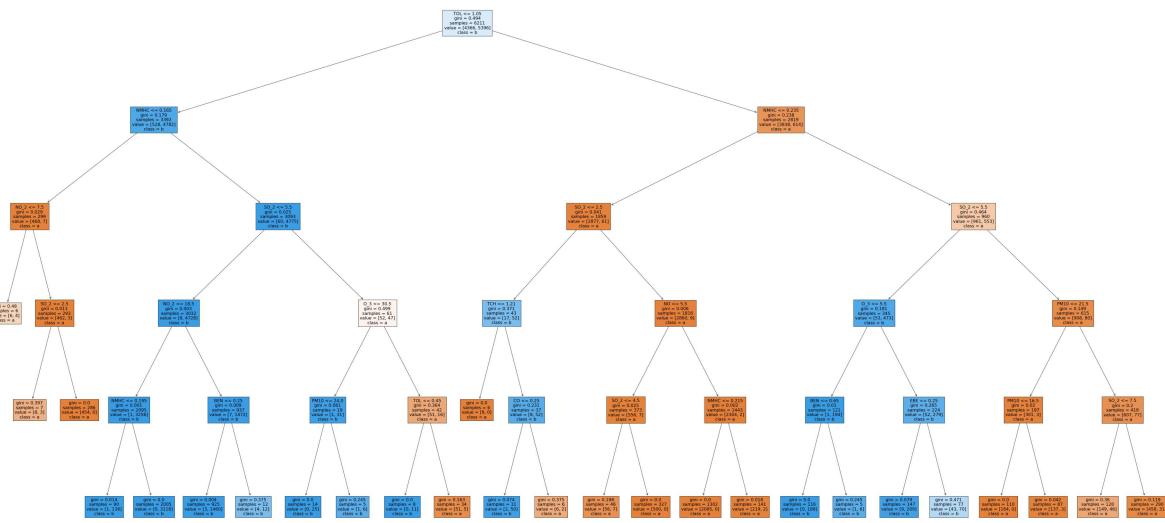
```
Out[60]: 0.99580004097521
```

```
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(1825.125, 1993.2, 'TOL <= 1.05\n gini = 0.494\n samples = 6211\n value = [4366, 5396]\n class = b'),  
 Text(651.0, 1630.800000000002, 'NMHC <= 0.165\n gini = 0.179\n samples = 3392\n value = [528, 4782]\n class = b'),  
 Text(186.0, 1268.4, 'NO_2 <= 7.5\n gini = 0.029\n samples = 299\n value = [468, 7]\n class = a'),  
 Text(93.0, 906.0, 'gini = 0.48\n samples = 6\n value = [6, 4]\n class = a'),  
 Text(279.0, 906.0, 'SO_2 <= 2.5\n gini = 0.013\n samples = 293\n value = [462, 3]\n class = a'),  
 Text(186.0, 543.599999999999, 'gini = 0.397\n samples = 7\n value = [8, 3]\n class = a'),  
 Text(372.0, 543.599999999999, 'gini = 0.0\n samples = 286\n value = [454, 0]\n class = a'),  
 Text(1116.0, 1268.4, 'SO_2 <= 5.5\n gini = 0.025\n samples = 3093\n value = [60, 4775]\n class = b'),  
 Text(744.0, 906.0, 'NO_2 <= 18.5\n gini = 0.003\n samples = 3032\n value = [8, 4728]\n class = b'),  
 Text(558.0, 543.599999999999, 'NMHC <= 0.195\n gini = 0.001\n samples = 2095\n value = [1, 3256]\n class = b'),  
 Text(465.0, 181.199999999982, 'gini = 0.014\n samples = 90\n value = [1, 138]\n class = b'),  
 Text(651.0, 181.199999999982, 'gini = 0.0\n samples = 2005\n value = [0, 3118]\n class = b'),  
 Text(930.0, 543.599999999999, 'BEN <= 0.25\n gini = 0.009\n samples = 937\n value = [7, 1472]\n class = b'),  
 Text(837.0, 181.199999999982, 'gini = 0.004\n samples = 925\n value = [3, 1460]\n class = b'),  
 Text(1023.0, 181.199999999982, 'gini = 0.375\n samples = 12\n value = [4, 12]\n class = b'),  
 Text(1488.0, 906.0, 'O_3 <= 30.5\n gini = 0.499\n samples = 61\n value = [52, 47]\n class = a'),  
 Text(1302.0, 543.599999999999, 'PM10 <= 24.0\n gini = 0.061\n samples = 19\n value = [1, 31]\n class = b'),  
 Text(1209.0, 181.199999999982, 'gini = 0.0\n samples = 14\n value = [0, 25]\n class = b'),  
 Text(1395.0, 181.199999999982, 'gini = 0.245\n samples = 5\n value = [1, 6]\n class = b'),  
 Text(1674.0, 543.599999999999, 'TOL <= 0.45\n gini = 0.364\n samples = 42\n value = [51, 16]\n class = a'),  
 Text(1581.0, 181.199999999982, 'gini = 0.0\n samples = 8\n value = [0, 11]\n class = b'),  
 Text(1767.0, 181.199999999982, 'gini = 0.163\n samples = 34\n value = [51, 5]\n class = a'),  
 Text(2999.25, 1630.800000000002, 'NMHC <= 0.235\n gini = 0.238\n samples = 2819\n value = [3838, 614]\n class = a'),  
 Text(2278.5, 1268.4, 'SO_2 <= 2.5\n gini = 0.041\n samples = 1859\n value = [2877, 61]\n class = a'),  
 Text(1953.0, 906.0, 'TCH <= 1.21\n gini = 0.371\n samples = 43\n value = [17, 52]\n class = b'),  
 Text(1860.0, 543.599999999999, 'gini = 0.0\n samples = 6\n value = [9, 0]\n class = a'),  
 Text(2046.0, 543.599999999999, 'CO <= 0.25\n gini = 0.231\n samples = 37\n value = [8, 52]\n class = b'),  
 Text(1953.0, 181.199999999982, 'gini = 0.074\n samples = 31\n value = [2, 50]\n class = b'),  
 Text(2139.0, 181.199999999982, 'gini = 0.375\n samples = 6\n value = [6, 2]\n class = a'),
```

```
Text(2604.0, 906.0, 'NO <= 5.5\ngini = 0.006\nsamples = 1816\nvalue = [2860, 9]\nnclass = a'),  
Text(2418.0, 543.5999999999999, 'SO_2 <= 4.5\ngini = 0.025\nsamples = 373\nvalue = [556, 7]\nnclass = a'),  
Text(2325.0, 181.1999999999982, 'gini = 0.198\nsamples = 46\nvalue = [56, 7]\nnclass = a'),  
Text(2511.0, 181.1999999999982, 'gini = 0.0\nsamples = 327\nvalue = [500, 0]\nnclass = a'),  
Text(2790.0, 543.5999999999999, 'NMHC <= 0.215\ngini = 0.002\nsamples = 1443\nvalue = [2304, 2]\nnclass = a'),  
Text(2697.0, 181.1999999999982, 'gini = 0.0\nsamples = 1302\nvalue = [2085, 0]\nnclass = a'),  
Text(2883.0, 181.1999999999982, 'gini = 0.018\nsamples = 141\nvalue = [219, 2]\nnclass = a'),  
Text(3720.0, 1268.4, 'SO_2 <= 5.5\ngini = 0.464\nsamples = 960\nvalue = [961, 553]\nnclass = a'),  
Text(3348.0, 906.0, 'O_3 <= 5.5\ngini = 0.181\nsamples = 345\nvalue = [53, 473]\nnclass = b'),  
Text(3162.0, 543.5999999999999, 'BEN <= 0.65\ngini = 0.01\nsamples = 121\nvalue = [1, 194]\nnclass = b'),  
Text(3069.0, 181.1999999999982, 'gini = 0.0\nsamples = 116\nvalue = [0, 188]\nnclass = b'),  
Text(3255.0, 181.1999999999982, 'gini = 0.245\nsamples = 5\nvalue = [1, 6]\nnclass = b'),  
Text(3534.0, 543.5999999999999, 'EBE <= 0.25\ngini = 0.265\nsamples = 224\nvalue = [52, 279]\nnclass = b'),  
Text(3441.0, 181.1999999999982, 'gini = 0.079\nsamples = 147\nvalue = [9, 209]\nnclass = b'),  
Text(3627.0, 181.1999999999982, 'gini = 0.471\nsamples = 77\nvalue = [43, 70]\nnclass = b'),  
Text(4092.0, 906.0, 'PM10 <= 21.5\ngini = 0.149\nsamples = 615\nvalue = [908, 80]\nnclass = a'),  
Text(3906.0, 543.5999999999999, 'PM10 <= 16.5\ngini = 0.02\nsamples = 197\nvalue = [301, 3]\nnclass = a'),  
Text(3813.0, 181.1999999999982, 'gini = 0.0\nsamples = 110\nvalue = [164, 0]\nnclass = a'),  
Text(3999.0, 181.1999999999982, 'gini = 0.042\nsamples = 87\nvalue = [137, 3]\nnclass = a'),  
Text(4278.0, 543.5999999999999, 'SO_2 <= 7.5\ngini = 0.2\nsamples = 418\nvalue = [607, 77]\nnclass = a'),  
Text(4185.0, 181.1999999999982, 'gini = 0.36\nsamples = 128\nvalue = [149, 46]\nnclass = a'),  
Text(4371.0, 181.1999999999982, 'gini = 0.119\nsamples = 290\nvalue = [458, 31]\nnclass = a')]
```



Conclusion

Accuracy

Linear Regression:0.8933824684760232

Ridge Regression:0.8700021760005732

Lasso Regression:0.29803877281876634

ElasticNet Regression:0.5963524316522861

Logistic Regression:0.9926143697117453

Random Forest:0.99580004097521

Random Forest is suitable for this dataset