

mk 03/08/2023

In [1]:

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
```

In [2]:

```
1 df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2017.csv")
2 df
```

Out[2]:

| | date | BEN | CH4 | CO | EBE | NMHC | NO | NO_2 | NOx | O_3 | PM10 | PM25 | SO_2 | TCH | TOL | station |
|--------|---------------------|-----|-----|-----|-----|------|-----|------|-----|------|------|------|------|-----|-----|----------|
| 0 | 2017-06-01 01:00:00 | NaN | NaN | 0.3 | NaN | NaN | 4.0 | 38.0 | NaN | NaN | NaN | NaN | 5.0 | NaN | NaN | 28079004 |
| 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 | 1.4 | 2.9 | 28079008 |
| 2 | 2017-06-01 01:00:00 | 0.2 | NaN | NaN | 0.1 | NaN | 1.0 | 14.0 | NaN | NaN | NaN | NaN | NaN | NaN | 0.9 | 28079011 |
| 3 | 2017-06-01 01:00:00 | NaN | NaN | 0.2 | NaN | NaN | 1.0 | 9.0 | NaN | 91.0 | NaN | NaN | NaN | NaN | NaN | 28079016 |
| 4 | 2017-06-01 01:00:00 | NaN | NaN | NaN | NaN | NaN | 1.0 | 19.0 | NaN | 69.0 | NaN | NaN | 2.0 | NaN | NaN | 28079017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 210115 | 2017-08-01 00:00:00 | NaN | NaN | 0.2 | NaN | NaN | 1.0 | 27.0 | NaN | 65.0 | NaN | NaN | NaN | NaN | NaN | 28079056 |
| 210116 | 2017-08-01 00:00:00 | NaN | NaN | 0.2 | NaN | NaN | 1.0 | 14.0 | NaN | NaN | 73.0 | NaN | 7.0 | NaN | NaN | 28079057 |
| 210117 | 2017-08-01 00:00:00 | NaN | NaN | NaN | NaN | NaN | 1.0 | 4.0 | NaN | 83.0 | NaN | NaN | NaN | NaN | NaN | 28079058 |
| 210118 | 2017-08-01 00:00:00 | NaN | NaN | NaN | NaN | NaN | 1.0 | 11.0 | NaN | 78.0 | NaN | NaN | NaN | NaN | NaN | 28079059 |
| 210119 | 2017-08-01 00:00:00 | NaN | NaN | NaN | NaN | NaN | 1.0 | 14.0 | NaN | 77.0 | 60.0 | NaN | NaN | NaN | NaN | 28079060 |

210120 rows × 16 columns

In [3]:

```
1 df=df.dropna()
```

In [4]:

```
1 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]:

```
1 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]: 1 data=df[['BEN', 'TOL', 'TCH']]  
2 data
```

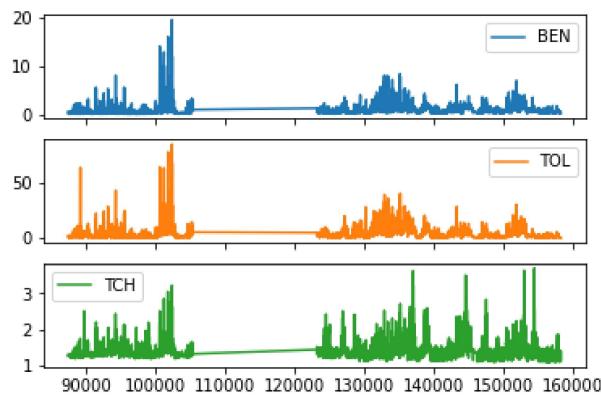
Out[6]:

| | 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

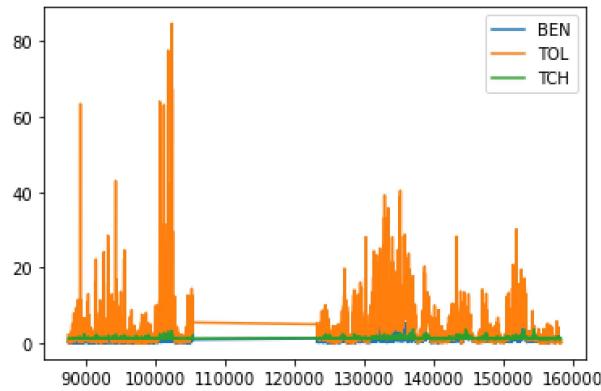
```
In [7]: 1 data.plot.line(subplots=True)
```

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



```
In [8]: 1 data.plot.line()
```

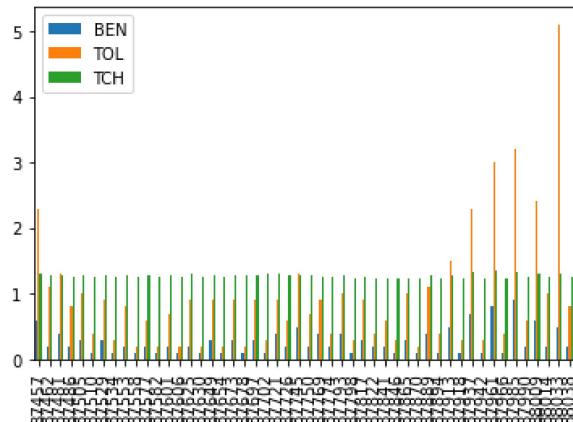
Out[8]: <AxesSubplot:>



```
In [9]: 1 b=data[0:50]
```

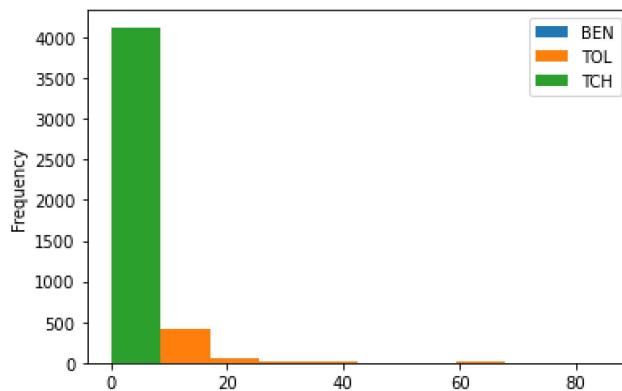
```
In [10]: 1 b.plot.bar()
```

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



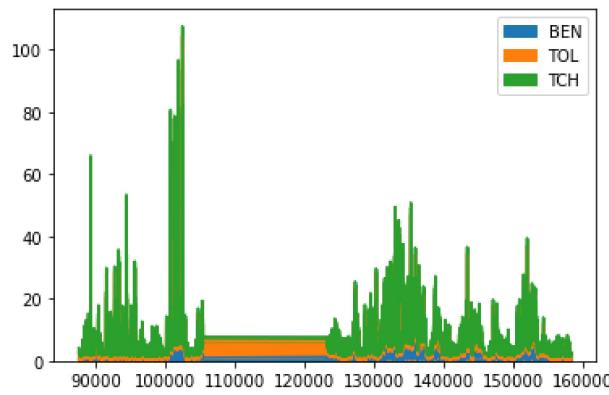
```
In [11]: 1 data.plot.hist()
```

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



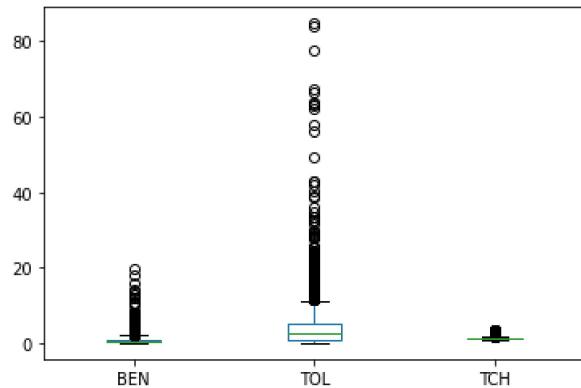
```
In [12]: 1 data.plot.area()
```

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



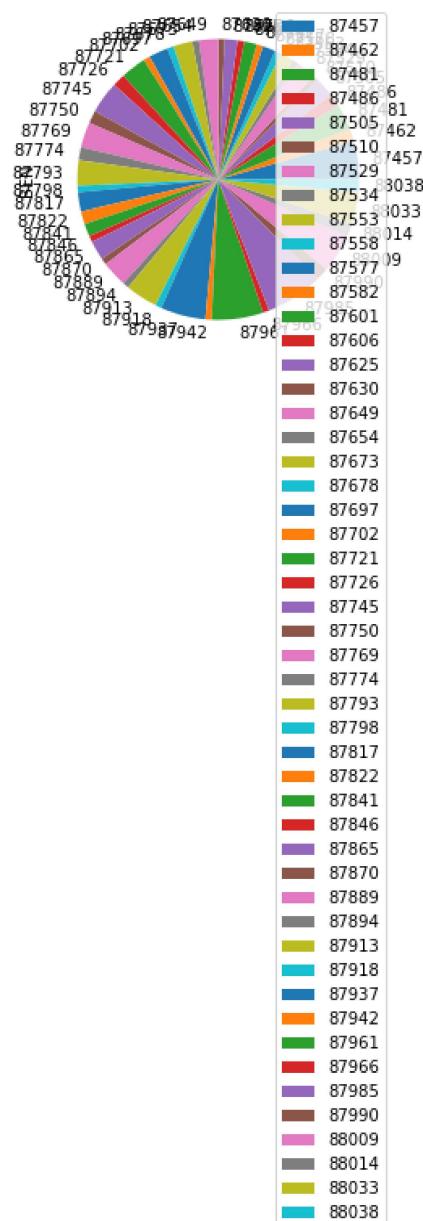
```
In [13]: 1 data.plot.box()
```

```
Out[13]: <AxesSubplot:>
```



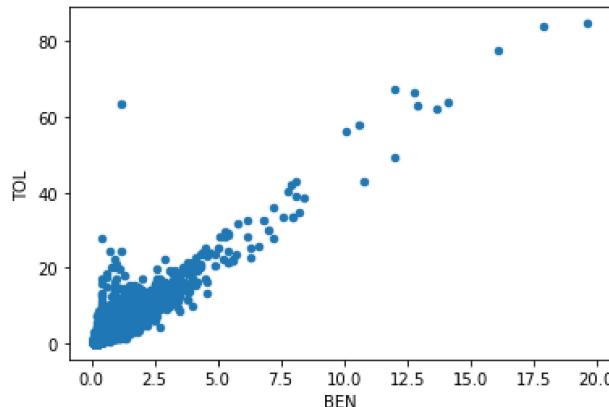
```
In [14]: 1 b.plot.pie(y='BEN' )
```

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



In [15]: 1 data.plot.scatter(x='BEN' ,y='TOL')

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



In [16]: 1 df.describe()

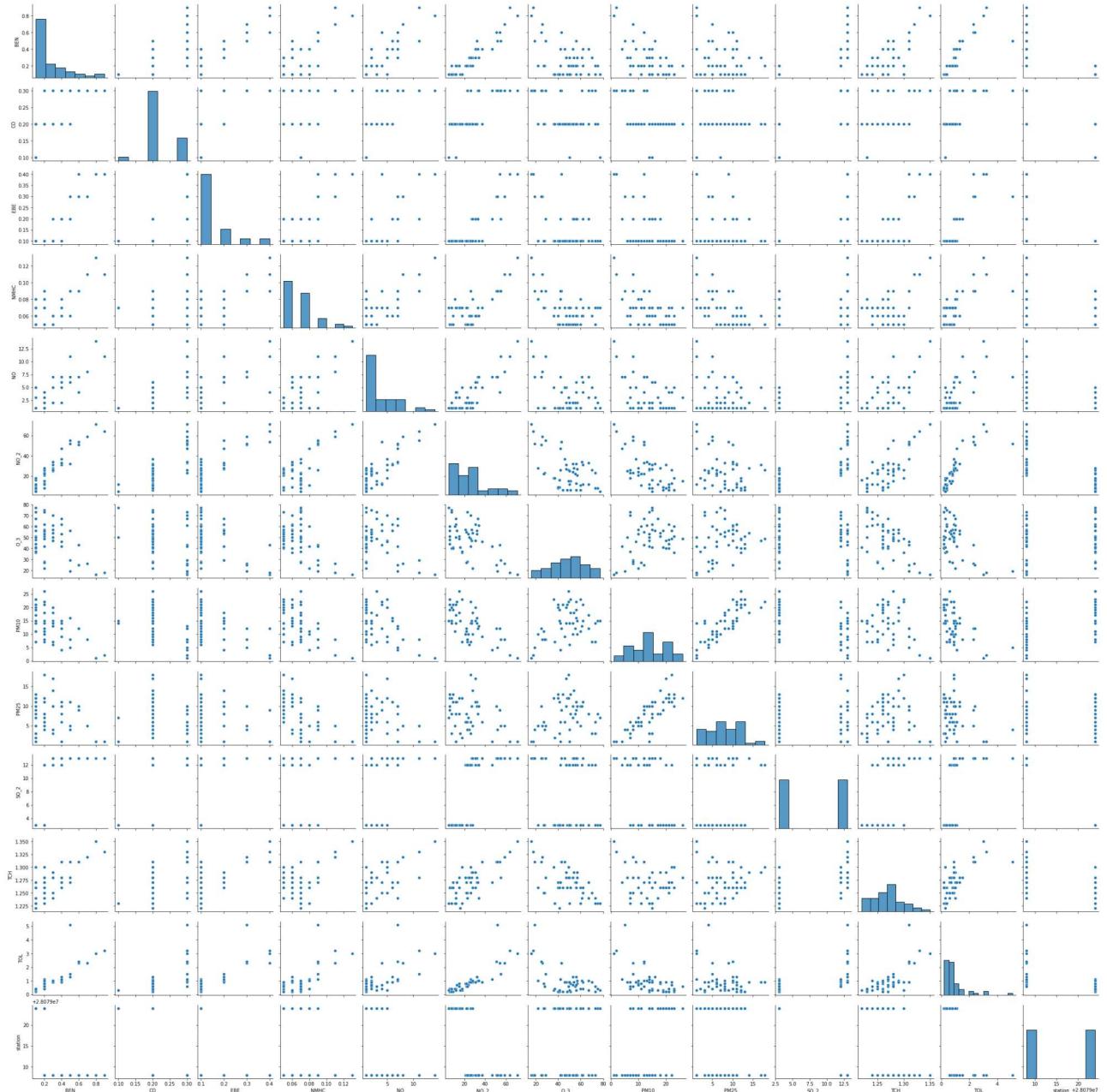
Out[16]:

| | BEN | CH4 | CO | EBE | NMHC | NO | NO_2 | NOx | O_3 |
|-------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| count | 4127.000000 | 4127.000000 | 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 | 58.069057 | 122.125515 | 28.716501 |
| std | 1.123078 | 0.215742 | 0.342871 | 0.962000 | 0.094035 | 71.118499 | 38.974112 | 142.828344 | 25.304909 |
| min | 0.100000 | 1.100000 | 0.100000 | 0.100000 | 0.000000 | 1.000000 | 1.000000 | 2.000000 | 1.000000 |
| 25% | 0.300000 | 1.180000 | 0.200000 | 0.100000 | 0.050000 | 3.000000 | 30.000000 | 37.000000 | 6.000000 |
| 50% | 0.600000 | 1.270000 | 0.300000 | 0.300000 | 0.080000 | 16.000000 | 54.000000 | 80.000000 | 22.000000 |
| 75% | 1.100000 | 1.400000 | 0.500000 | 0.700000 | 0.110000 | 50.000000 | 78.000000 | 153.000000 | 46.000000 |
| max | 19.600000 | 3.630000 | 4.900000 | 16.700001 | 1.420000 | 879.000000 | 349.000000 | 1681.000000 | 140.000000 |

In [17]: 1 df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
2 'SO_2', 'TCH', 'TOL', 'station']]

```
In [18]: 1 sns.pairplot(df1[0:50])
```

```
Out[18]: <seaborn.axisgrid.PairGrid at 0x2135c7472b0>
```

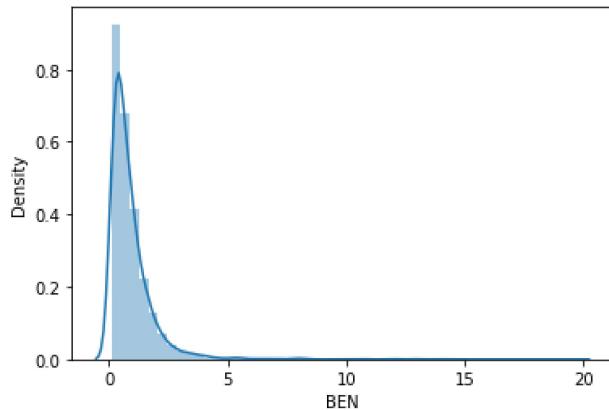


In [19]:

```
1 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)

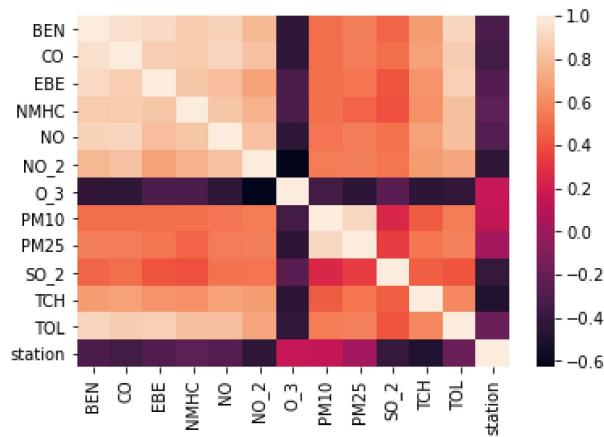
Out[19]: <AxesSubplot:xlabel='BEN', ylabel='Density'>



In [20]:

```
1 sns.heatmap(df1.corr())
```

Out[20]: <AxesSubplot:>



In [21]:

```
1 x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
2 'SO_2', 'TCH', 'TOL']]
3 y=df['station']
```

In [22]:

```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [23]:

```
1 from sklearn.linear_model import LinearRegression
2 lr=LinearRegression()
3 lr.fit(x_train,y_train)
```

Out[23]: LinearRegression()

In [24]:

```
1 lr.intercept_
```

Out[24]: 28079042.124266382

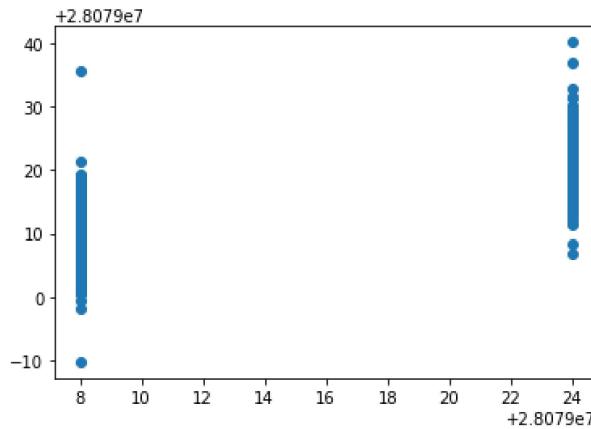
```
In [25]: 1 coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
2 coeff
```

Out[25]:

| | Co-efficient |
|------|--------------|
| BEN | -0.582697 |
| CO | -4.505782 |
| EBE | -1.900084 |
| NMHC | 25.454120 |
| NO | 0.050906 |
| NO_2 | -0.183283 |
| O_3 | -0.090455 |
| PM10 | 0.418976 |
| PM25 | -0.153112 |
| SO_2 | -0.249782 |
| TCH | -13.825667 |
| TOL | 0.312383 |

```
In [26]: 1 prediction =lr.predict(x_test)
2 plt.scatter(y_test,prediction)
```

Out[26]: <matplotlib.collections.PathCollection at 0x21367703a90>



```
In [27]: 1 lr.score(x_test,y_test)
```

Out[27]: 0.6509798297639944

```
In [28]: 1 lr.score(x_train,y_train)
```

Out[28]: 0.6277182624521873

```
In [29]: 1 from sklearn.linear_model import Ridge,Lasso
```

```
In [30]: 1 rr=Ridge(alpha=10)
2 rr.fit(x_train,y_train)
```

Out[30]: Ridge(alpha=10)

```
In [31]: 1 rr.score(x_test,y_test)
```

Out[31]: 0.6388430689805904

```
In [32]: 1 rr.score(x_train,y_train)
```

Out[32]: 0.6191897548217457

```
In [33]: 1 la=Lasso(alpha=10)
          2 la.fit(x_train,y_train)
```

```
Out[33]: Lasso(alpha=10)
```

```
In [34]: 1 la.score(x_test,y_test)
```

```
Out[34]: 0.4106090579902061
```

```
In [35]: 1 la.score(x_train,y_train)
```

```
Out[35]: 0.3977464968218405
```

```
In [36]: 1 from sklearn.linear_model import ElasticNet
          2 en=ElasticNet()
          3 en.fit(x_train,y_train)
```

```
Out[36]: ElasticNet()
```

```
In [37]: 1 en.coef_
```

```
Out[37]: array([-0.         , -0.         , -0.         ,  0.         ,  0.03190586,
       -0.20368023, -0.08462839,  0.55000139, -0.40450207, -0.30078072,
       -0.         ,  0.         ])
```

```
In [38]: 1 en.intercept_
```

```
Out[38]: 28079025.257014
```

```
In [39]: 1 prediction=en.predict(x_test)
```

```
In [40]: 1 en.score(x_test,y_test)
```

```
Out[40]: 0.5205114372790913
```

```
In [41]: 1 from sklearn import metrics
          2 print(metrics.mean_absolute_error(y_test,prediction))
          3 print(metrics.mean_squared_error(y_test,prediction))
          4 print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
4.723383129836693
30.683889682205763
5.539304079232856
```

```
In [42]: 1 from sklearn.linear_model import LogisticRegression
```

```
In [43]: 1 feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
          2 'PM10', 'SO_2', 'TCH', 'TOL']]
          3 target_vector=df['station']
```

```
In [44]: 1 feature_matrix.shape
```

```
Out[44]: (4127, 10)
```

```
In [45]: 1 target_vector.shape
```

```
Out[45]: (4127,)
```

```
In [46]: 1 from sklearn.preprocessing import StandardScaler
```

```
In [47]: 1 fs=StandardScaler().fit_transform(feature_matrix)
```

```
In [48]: 1 logr=LogisticRegression(max_iter=10000)
          2 logr.fit(fs,target_vector)
```

```
Out[48]: LogisticRegression(max_iter=10000)
```

```
In [49]: 1 observation=[[1,2,3,4,5,6,7,8,9,10]]
```

```
In [50]: 1 prediction=logr.predict(observation)
          2 print(prediction)
```

```
[28079008]
```

```
In [51]: 1 logr.classes_
```

```
Out[51]: array([28079008, 28079024], dtype=int64)
```

```
In [52]: 1 logr.score(fs,target_vector)
```

```
Out[52]: 0.9437848315968016
```

```
In [53]: 1 logr.predict_proba(observation)[0][0]
```

```
Out[53]: 0.999999999725541
```

```
In [54]: 1 logr.predict_proba(observation)
```

```
Out[54]: array([[1.00000000e+00, 2.74458959e-11]])
```

```
In [55]: 1 from sklearn.ensemble import RandomForestClassifier
```

```
In [56]: 1 rfc=RandomForestClassifier()
          2 rfc.fit(x_train,y_train)
```

```
Out[56]: RandomForestClassifier()
```

```
In [57]: 1 parameters={'max_depth':[1,2,3,4,5],
          2 'min_samples_leaf':[5,10,15,20,25],
          3 'n_estimators':[10,20,30,40,50]
          4 }
```

```
In [58]: 1 from sklearn.model_selection import GridSearchCV
          2 grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
          3 grid_search.fit(x_train,y_train)
```

```
Out[58]: 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 [59]: 1 grid_search.best_score_
```

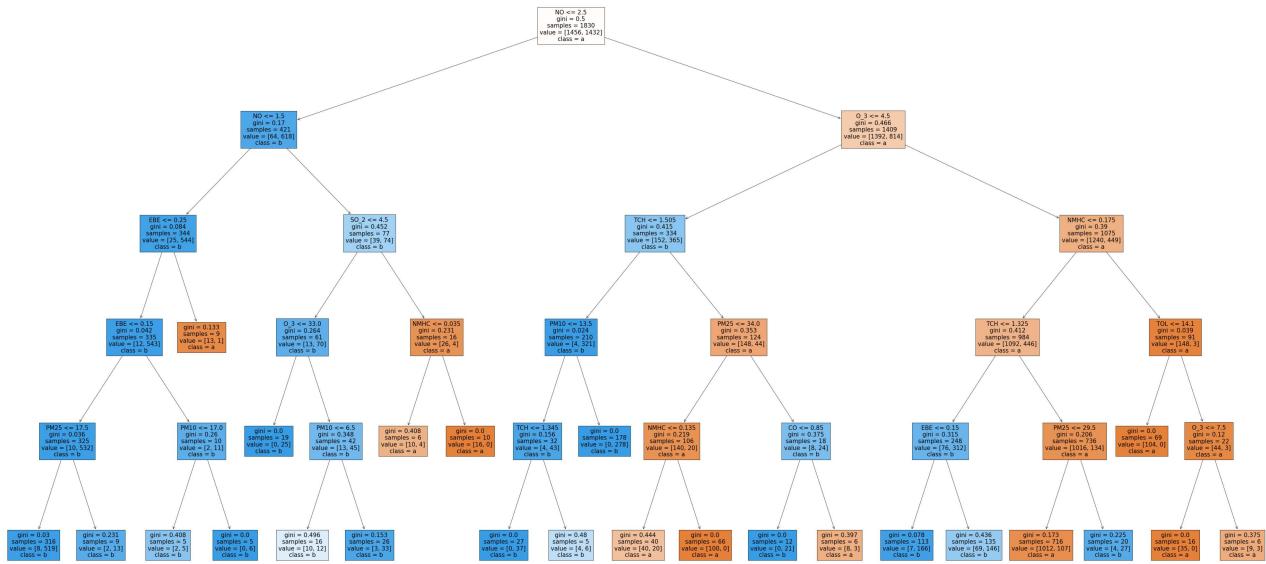
```
Out[59]: 0.9653739612188366
```

```
In [60]: 1 rfc_best=grid_search.best_estimator_
```

```
In [61]: 1 from sklearn.tree import plot_tree
2 plt.figure(figsize=(80,40))
3 plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d'],filled=True)
```

```
Out[61]: [Text(1997.0526315789473, 1993.2, 'NO <= 2.5\ngini = 0.5\nsamples = 1830\nvalue = [1456, 1432]\nnclass = a'),
Text(939.7894736842105, 1630.8000000000002, 'NO <= 1.5\ngini = 0.17\nsamples = 421\nvalue = [64, 618]\nnclass = b'),
Text(587.3684210526316, 1268.4, 'EBE <= 0.25\ngini = 0.084\nsamples = 344\nvalue = [25, 544]\nnclass = b'),
Text(469.89473684210526, 906.0, 'EBE <= 0.15\ngini = 0.042\nsamples = 335\nvalue = [12, 543]\nnclass = b'),
Text(234.94736842105263, 543.5999999999999, 'PM25 <= 17.5\ngini = 0.036\nsamples = 325\nvalue = [10, 532]\nnclass = b'),
Text(117.47368421052632, 181.1999999999982, 'gini = 0.03\nsamples = 316\nvalue = [8, 519]\nnclass = b'),
Text(352.42105263157896, 181.1999999999982, 'gini = 0.231\nsamples = 9\nvalue = [2, 13]\nnclass = b'),
Text(704.8421052631579, 543.5999999999999, 'PM10 <= 17.0\ngini = 0.26\nsamples = 10\nvalue = [2, 11]\nnclass = b'),
Text(587.3684210526316, 181.1999999999982, 'gini = 0.408\nsamples = 5\nvalue = [2, 5]\nnclass = b'),
Text(822.3157894736842, 181.1999999999982, 'gini = 0.0\nsamples = 5\nvalue = [0, 6]\nnclass = b'),
Text(704.8421052631579, 906.0, 'gini = 0.133\nsamples = 9\nvalue = [13, 1]\nnclass = a'),
Text(1292.2105263157894, 1268.4, 'SO_2 <= 4.5\ngini = 0.452\nsamples = 77\nvalue = [39, 74]\nnclass = b'),
Text(1057.2631578947369, 906.0, 'O_3 <= 33.0\ngini = 0.264\nsamples = 61\nvalue = [13, 70]\nnclass = b'),
Text(939.7894736842105, 543.5999999999999, 'gini = 0.0\nsamples = 19\nvalue = [0, 25]\nnclass = b'),
Text(1174.7368421052631, 543.5999999999999, 'PM10 <= 6.5\ngini = 0.348\nsamples = 42\nvalue = [13, 45]\nnclass = b'),
Text(1057.2631578947369, 181.1999999999982, 'gini = 0.496\nsamples = 16\nvalue = [10, 12]\nnclass = b'),
Text(1292.2105263157894, 181.1999999999982, 'gini = 0.153\nsamples = 26\nvalue = [3, 33]\nnclass = b'),
Text(1527.157894736842, 906.0, 'NMHC <= 0.035\ngini = 0.231\nsamples = 16\nvalue = [26, 4]\nnclass = a'),
Text(1409.6842105263158, 543.5999999999999, 'gini = 0.408\nsamples = 6\nvalue = [10, 4]\nnclass = a'),
Text(1644.6315789473683, 543.5999999999999, 'gini = 0.0\nsamples = 10\nvalue = [16, 0]\nnclass = a'),
Text(3054.315789473684, 1630.8000000000002, 'O_3 <= 4.5\ngini = 0.466\nsamples = 1409\nvalue = [1392, 814]\nnclass = a'),
Text(2290.7368421052633, 1268.4, 'TCH <= 1.505\ngini = 0.415\nsamples = 334\nvalue = [152, 365]\nnclass = b'),
Text(1997.0526315789473, 906.0, 'PM10 <= 13.5\ngini = 0.024\nsamples = 210\nvalue = [4, 321]\nnclass = b'),
Text(1879.578947368421, 543.5999999999999, 'TCH <= 1.345\ngini = 0.156\nsamples = 32\nvalue = [4, 43]\nnclass = b'),
Text(1762.1052631578948, 181.1999999999982, 'gini = 0.0\nsamples = 27\nvalue = [0, 37]\nnclass = b'),
Text(1997.0526315789473, 181.1999999999982, 'gini = 0.48\nsamples = 5\nvalue = [4, 6]\nnclass = b'),
Text(2114.5263157894738, 543.5999999999999, 'gini = 0.0\nsamples = 178\nvalue = [0, 278]\nnclass = b'),
Text(2584.4210526315787, 906.0, 'PM25 <= 34.0\ngini = 0.353\nsamples = 124\nvalue = [148, 44]\nnclass = a'),
Text(2349.4736842105262, 543.5999999999999, 'NMHC <= 0.135\ngini = 0.219\nsamples = 106\nvalue = [140, 20]\nnclass = a'),
Text(2232.0, 181.1999999999982, 'gini = 0.444\nsamples = 40\nvalue = [40, 20]\nnclass = a'),
Text(2466.9473684210525, 181.1999999999982, 'gini = 0.0\nsamples = 66\nvalue = [100, 0]\nnclass = a'),
Text(2819.3684210526317, 543.5999999999999, 'CO <= 0.85\ngini = 0.375\nsamples = 18\nvalue = [8, 24]\nnclass = b'),
Text(2701.8947368421054, 181.1999999999982, 'gini = 0.0\nsamples = 12\nvalue = [0, 21]\nnclass = b'),
Text(2936.842105263158, 181.1999999999982, 'gini = 0.397\nsamples = 6\nvalue = [8, 3]\nnclass = a'),
Text(3817.8947368421054, 1268.4, 'NMHC <= 0.175\ngini = 0.39\nsamples = 1075\nvalue = [1240, 449]\nnclass = a'),
Text(3524.2105263157896, 906.0, 'TCH <= 1.325\ngini = 0.412\nsamples = 984\nvalue = [1092, 446]\nnclass = a'),
Text(3289.2631578947367, 543.5999999999999, 'EBE <= 0.15\ngini = 0.315\nsamples = 248\nvalue = [76, 312]\nnclass = b'),
Text(3171.7894736842104, 181.1999999999982, 'gini = 0.078\nsamples = 113\nvalue = [7, 166]\nnclass = b'),
Text(3406.7368421052633, 181.1999999999982, 'gini = 0.436\nsamples = 135\nvalue = [69, 146]\nnclass = b'),
Text(3759.157894736842, 543.5999999999999, 'PM25 <= 29.5\ngini = 0.206\nsamples = 736\nvalue = [1016, 134]\nnclass = a'),
Text(3641.684210526316, 181.1999999999982, 'gini = 0.173\nsamples = 716\nvalue = [1012, 107]\nnclass = a'),
Text(3876.6315789473683, 181.1999999999982, 'gini = 0.225\nsamples = 20\nvalue = [4, 27]\nnclass = b'),
Text(4111.578947368421, 906.0, 'TOL <= 14.1\ngini = 0.039\nsamples = 91\nvalue = [148, 3]\nnclass =
```

```
a'),
Text(3994.1052631578946, 543.5999999999999, 'gini = 0.0\nsamples = 69\nvalue = [104, 0]\nclass = a'),
Text(4229.0526315789475, 543.5999999999999, '0_3 <= 7.5\nngini = 0.12\nsamples = 22\nvalue = [44, 3]\nclass = a'),
Text(4111.578947368421, 181.1999999999982, 'gini = 0.0\nsamples = 16\nvalue = [35, 0]\nclass = a'),
Text(4346.526315789473, 181.1999999999982, 'gini = 0.375\nsamples = 6\nvalue = [9, 3]\nclass = a')
```



Conclusion

Linear Regression=0.6277182624521873

Ridge Regression=0.6191897548217457

Lasso Regression=0.3977464968218405

ElasticNet Regression=0.5205114372790913

Logistic Regression=0.9437848315968016

Random Forest=0.9653739612188366

Random Forest is suitable for this dataset