

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_2014.csv")
2 df
```

Out[2]:

		date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0		2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN	28079004
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	28079008
2		2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1	28079011
3		2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN	28079016
4		2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN	28079017
...	
210019		2014-09-01 00:00:00	NaN	0.5	NaN	NaN	20.0	84.0	29.0	NaN	NaN	NaN	NaN	NaN	28079056
210020		2014-09-01 00:00:00	NaN	0.3	NaN	NaN	1.0	22.0	NaN	15.0	NaN	6.0	NaN	NaN	28079057
210021		2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	13.0	70.0	NaN	NaN	NaN	NaN	NaN	28079058
210022		2014-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	38.0	42.0	NaN	NaN	NaN	NaN	NaN	28079059
210023		2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	26.0	65.0	11.0	NaN	NaN	NaN	NaN	28079060

210024 rows × 14 columns

In [3]:

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

In [4]:

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

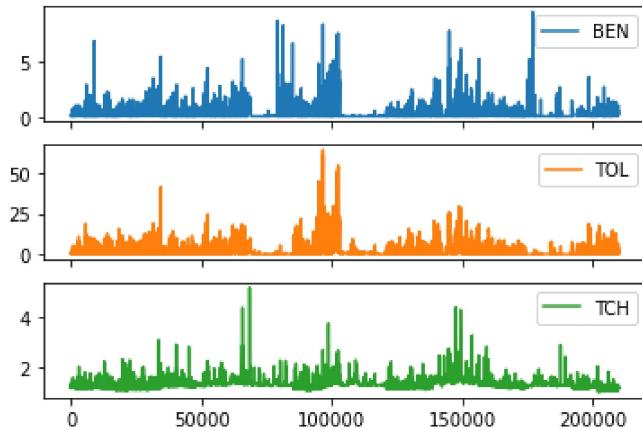
Out[6]:

	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

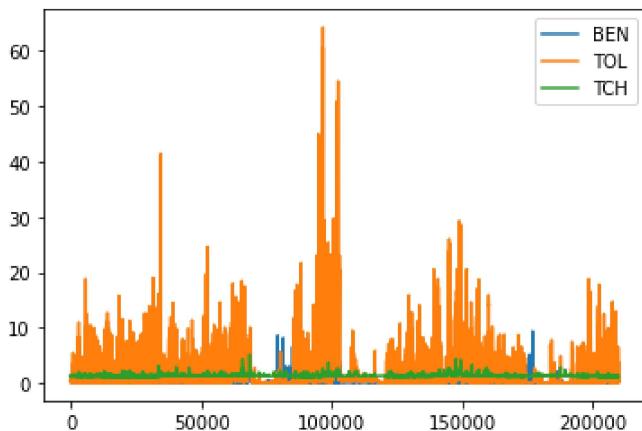
```
In [7]: 1 data.plot.line(subplots=True)
```

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



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

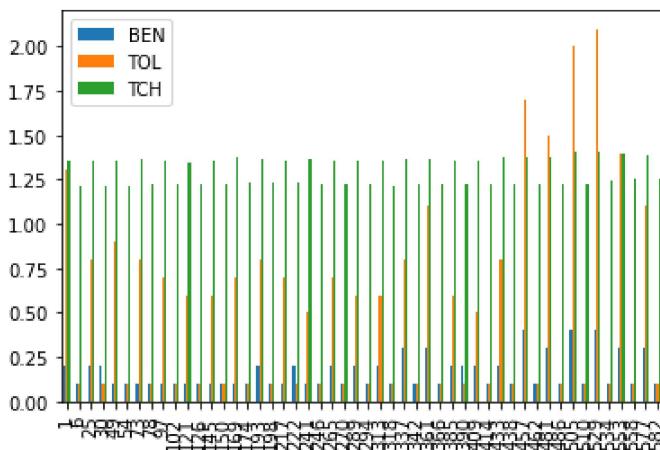
```
Out[8]: <AxesSubplot:>
```



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

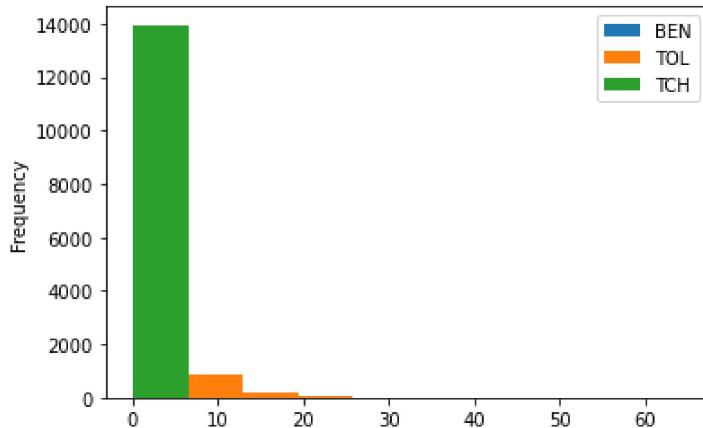
```
In [10]: 1 b.plot.bar()
```

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



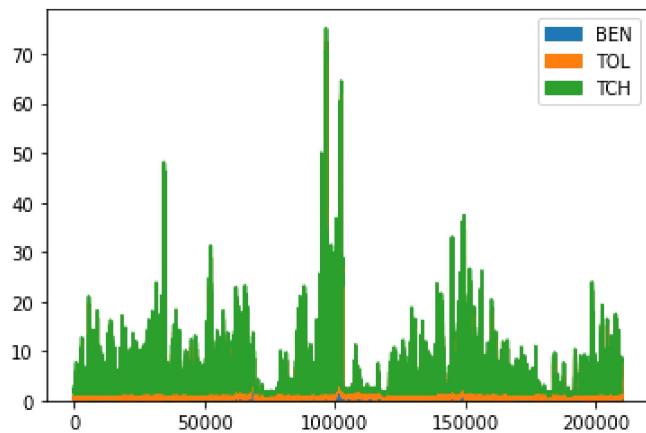
```
In [11]: 1 data.plot.hist()
```

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



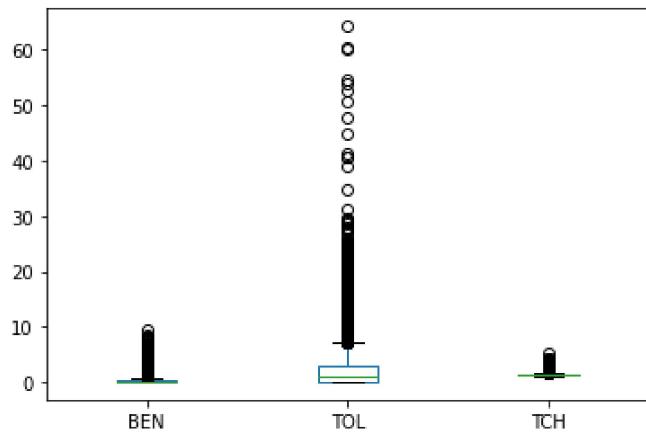
```
In [12]: 1 data.plot.area()
```

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



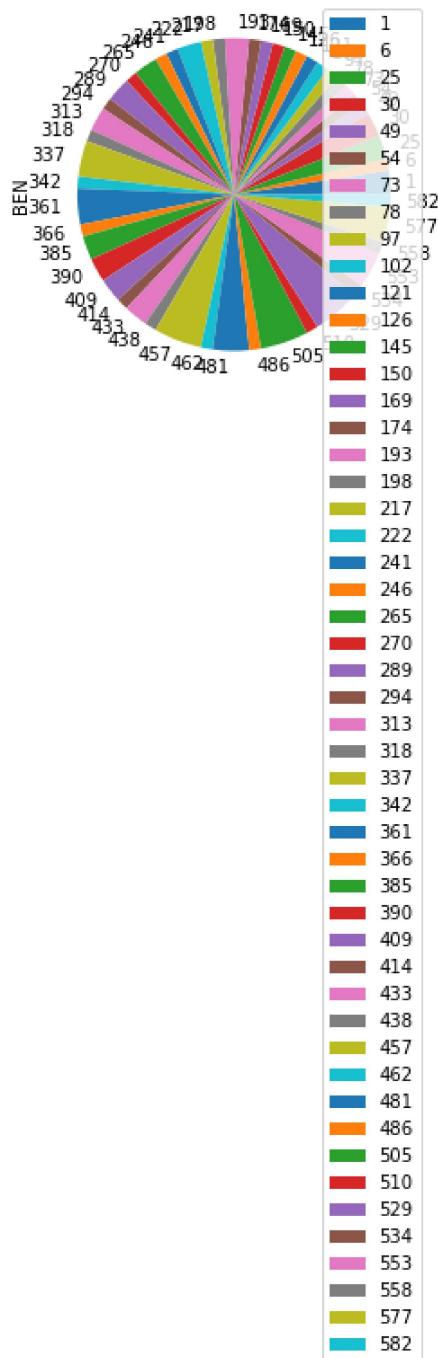
```
In [13]: 1 data.plot.box()
```

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



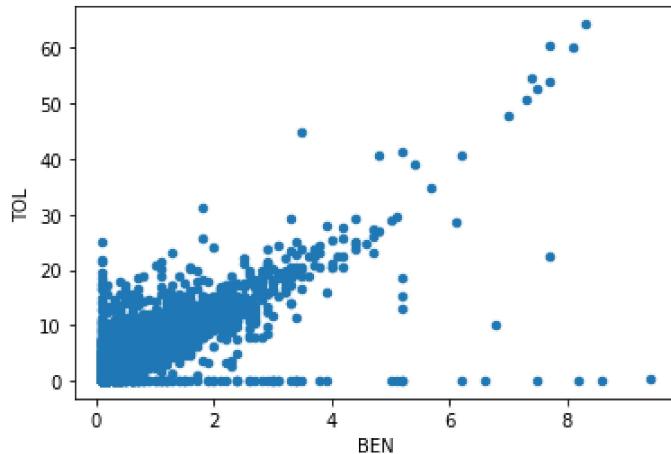
In [14]: 1 b.plot.pie(y='BEN')

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()

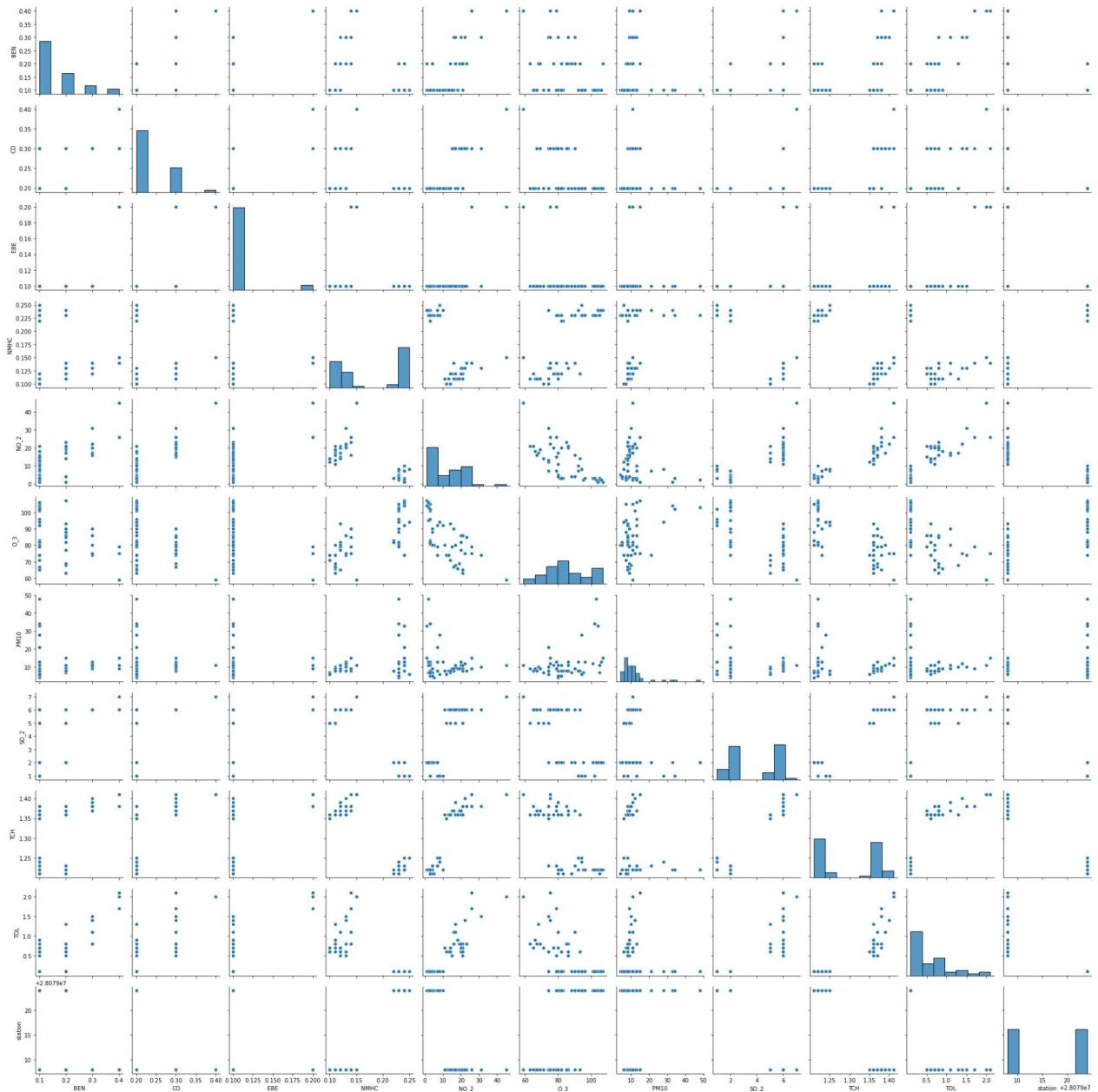
Out[16]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3
count	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000
mean	0.375921	0.314793	0.306016	0.222302	17.589129	34.240929	53.082389
std	0.555093	0.207375	0.635475	0.082403	39.432216	30.654229	33.488167
min	0.100000	0.100000	0.100000	0.060000	1.000000	1.000000	1.000000
25%	0.100000	0.200000	0.100000	0.160000	1.000000	10.000000	25.000000
50%	0.200000	0.300000	0.100000	0.230000	4.000000	27.000000	53.000000
75%	0.400000	0.400000	0.300000	0.260000	18.000000	51.000000	75.000000
max	9.400000	4.400000	16.200001	1.290000	725.000000	346.000000	220.000000

In [17]: 1 df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3', 'PM10',
2 'SO_2', 'TCH', 'TOL', 'station']]

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

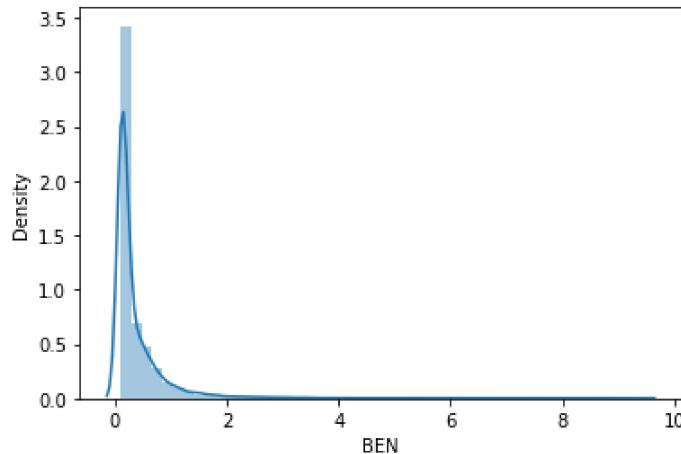
```
Out[18]: <seaborn.axisgrid.PairGrid at 0x1c1a6d61070>
```



In [19]: 1 sns.distplot(df1['BEN'])

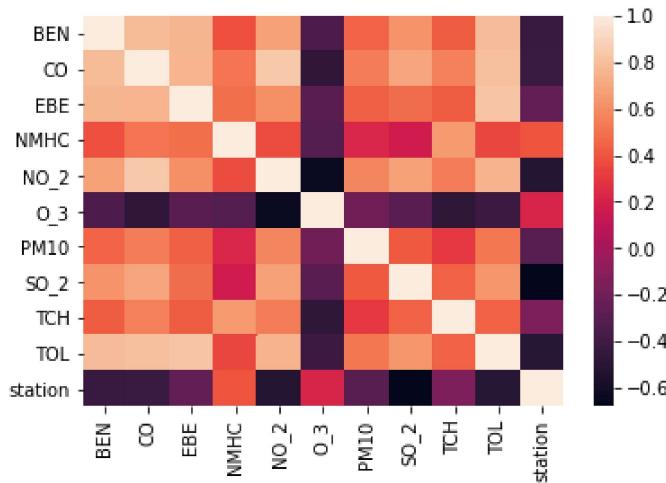
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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_2', 'O_3', 'PM10',
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]: 28079022.4478457
```

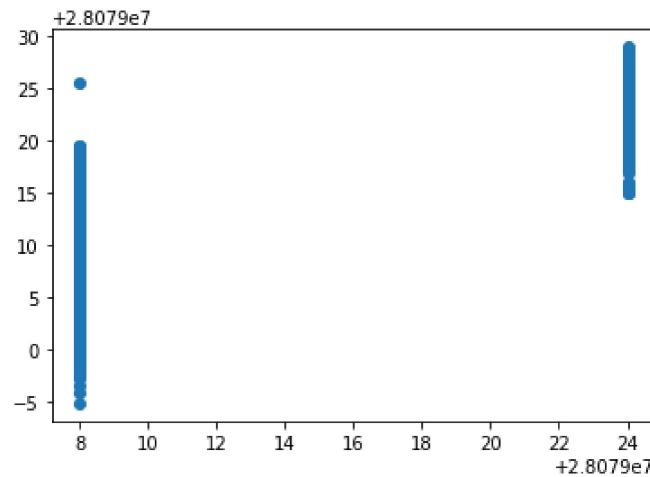
```
In [25]: 1 coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
2 coeff
```

```
Out[25]:
```

	Co-efficient
BEN	-1.293950
CO	-5.809428
EBE	0.424667
NMHC	83.373217
NO_2	-0.033397
O_3	0.002186
PM10	0.017884
SO_2	-0.879364
TCH	-11.780202
TOL	-0.441750

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

```
Out[26]: <matplotlib.collections.PathCollection at 0x1c1b0363400>
```



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

```
Out[27]: 0.8895083402137607
```

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

```
Out[28]: 0.881893030415985
```

```
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]: 0.8677110519288443
```

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

```
Out[32]: 0.8588863738901591
```

```
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]: 0.28025128201289573
```

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

```
Out[35]: 0.26893317064861
```

```
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.           ,  0.22449093,  0.           ,
   -0.0113182 ,  0.02390173, -1.24163916,  0.           ,
   -0.16995639])
```

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

```
Out[38]: 28079024.652497325
```

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

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

```
Out[40]: 0.4895268914063027
```

```
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)))
```

```
4.994734441387836
```

```
32.32841206961237
```

```
5.685807952227402
```

```
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]: (13946, 10)
```

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

```
Out[45]: (13946,)
```

```
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]: 0.9926143697117453
```

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

```
Out[53]: 1.0
```

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

```
Out[54]: array([[1.0000000e+00, 5.27113072e-18]])
```

```
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]: 0.99580004097521
```

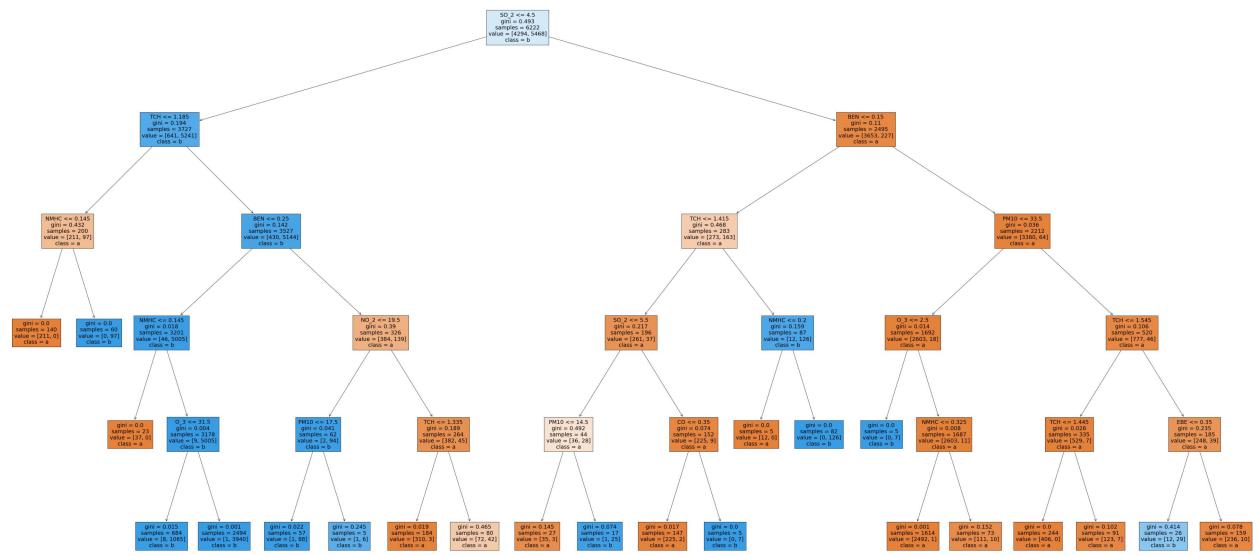
```
In [60]: 1 rfc_best=grid_search.best_estimator_
```

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'])
```

```
Out[61]: [Text(1827.4499999999998, 1993.2, 'SO_2 <= 4.5\ngini = 0.493\nsamples = 6222\nvalue = [429  
4, 5468]\nnclass = b'),  
Text(585.9, 1630.8000000000002, 'TCH <= 1.185\ngini = 0.194\nsamples = 3727\nvalue = [64  
1, 5241]\nnclass = b'),  
Text(223.2, 1268.4, 'NMHC <= 0.145\ngini = 0.432\nsamples = 200\nvalue = [211, 97]\nnclass  
= a'),  
Text(111.6, 906.0, 'gini = 0.0\nsamples = 140\nvalue = [211, 0]\nnclass = a'),  
Text(334.7999999999995, 906.0, 'gini = 0.0\nsamples = 60\nvalue = [0, 97]\nnclass = b'),  
Text(948.5999999999999, 1268.4, 'BEN <= 0.25\ngini = 0.142\nsamples = 3527\nvalue = [430,  
5144]\nnclass = b'),  
Text(558.0, 906.0, 'NMHC <= 0.145\ngini = 0.018\nsamples = 3201\nvalue = [46, 5005]\nclas  
s = b'),  
Text(446.4, 543.5999999999999, 'gini = 0.0\nsamples = 23\nvalue = [37, 0]\nnclass = a'),  
Text(669.5999999999999, 543.5999999999999, 'O_3 <= 31.5\ngini = 0.004\nsamples = 3178\nva  
lue = [9, 5005]\nnclass = b'),  
Text(558.0, 181.1999999999982, 'gini = 0.015\nsamples = 684\nvalue = [8, 1065]\nnclass =  
b'),  
Text(781.1999999999999, 181.1999999999982, 'gini = 0.001\nsamples = 2494\nvalue = [1, 39  
40]\nnclass = b'),  
Text(1339.1999999999998, 906.0, 'NO_2 <= 19.5\ngini = 0.39\nsamples = 326\nvalue = [384,  
139]\nnclass = a'),  
Text(1116.0, 543.5999999999999, 'PM10 <= 17.5\ngini = 0.041\nsamples = 62\nvalue = [2, 9  
4]\nnclass = b'),  
Text(1004.4, 181.1999999999982, 'gini = 0.022\nsamples = 57\nvalue = [1, 88]\nnclass =  
b'),  
Text(1227.6, 181.1999999999982, 'gini = 0.245\nsamples = 5\nvalue = [1, 6]\nnclass = b'),  
Text(1562.3999999999999, 543.5999999999999, 'TCH <= 1.335\ngini = 0.189\nsamples = 264\nva  
lue = [382, 45]\nnclass = a'),  
Text(1450.8, 181.1999999999982, 'gini = 0.019\nsamples = 184\nvalue = [310, 3]\nnclass =  
a'),  
Text(1674.0, 181.1999999999982, 'gini = 0.465\nsamples = 80\nvalue = [72, 42]\nnclass =  
a'),  
Text(3069.0, 1630.8000000000002, 'BEN <= 0.15\ngini = 0.11\nsamples = 2495\nvalue = [365  
3, 227]\nnclass = a'),  
Text(2511.0, 1268.4, 'TCH <= 1.415\ngini = 0.468\nsamples = 283\nvalue = [273, 163]\nclas  
s = a'),  
Text(2232.0, 906.0, 'SO_2 <= 5.5\ngini = 0.217\nsamples = 196\nvalue = [261, 37]\nnclass =  
a'),  
Text(2008.8, 543.5999999999999, 'PM10 <= 14.5\ngini = 0.492\nsamples = 44\nvalue = [36, 2  
8]\nnclass = a'),  
Text(1897.1999999999998, 181.1999999999982, 'gini = 0.145\nsamples = 27\nvalue = [35, 3]  
\nnclass = a'),  
Text(2120.4, 181.1999999999982, 'gini = 0.074\nsamples = 17\nvalue = [1, 25]\nnclass =  
b'),  
Text(2455.2, 543.5999999999999, 'CO <= 0.35\ngini = 0.074\nsamples = 152\nvalue = [225,  
9]\nnclass = a'),  
Text(2343.6, 181.1999999999982, 'gini = 0.017\nsamples = 147\nvalue = [225, 2]\nnclass =  
a'),  
Text(2566.7999999999997, 181.1999999999982, 'gini = 0.0\nsamples = 5\nvalue = [0, 7]\ncl  
ass = b'),  
Text(2790.0, 906.0, 'NMHC <= 0.2\ngini = 0.159\nsamples = 87\nvalue = [12, 126]\nnclass =  
b'),  
Text(2678.3999999999996, 543.5999999999999, 'gini = 0.0\nsamples = 5\nvalue = [12, 0]\ncl  
ass = a'),  
Text(2901.6, 543.5999999999999, 'gini = 0.0\nsamples = 82\nvalue = [0, 126]\nnclass = b'),  
Text(3627.0, 1268.4, 'PM10 <= 33.5\ngini = 0.036\nsamples = 2212\nvalue = [3380, 64]\ncl  
ass = a'),  
Text(3236.3999999999996, 906.0, 'O_3 <= 2.5\ngini = 0.014\nsamples = 1692\nvalue = [2603,  
18]\nnclass = a'),  
Text(3124.7999999999997, 543.5999999999999, 'gini = 0.0\nsamples = 5\nvalue = [0, 7]\ncl  
ass = b'),  
Text(3348.0, 543.5999999999999, 'NMHC <= 0.325\ngini = 0.008\nsamples = 1687\nvalue = [26  
03, 11]\nnclass = a'),
```

```
Text(3236.3999999999996, 181.19999999999982, 'gini = 0.001\nsamples = 1614\nvalue = [249  
2, 1]\nclass = a'),  
Text(3459.6, 181.19999999999982, 'gini = 0.152\nsamples = 73\nvalue = [111, 10]\nclass =  
a'),  
Text(4017.6, 906.0, 'TCH <= 1.545\n'gini = 0.106\nsamples = 520\nvalue = [777, 46]\nclasse  
= a'),  
Text(3794.3999999999996, 543.5999999999999, 'TCH <= 1.445\n'gini = 0.026\nsamples = 335\nnv  
alue = [529, 7]\nclass = a'),  
Text(3682.7999999999997, 181.19999999999982, 'gini = 0.0\nsamples = 244\nvalue = [406, 0]  
\nclass = a'),  
Text(3906.0, 181.19999999999982, 'gini = 0.102\nsamples = 91\nvalue = [123, 7]\nclass =  
a'),  
Text(4240.8, 543.5999999999999, 'EBE <= 0.35\n'gini = 0.235\nsamples = 185\nvalue = [248,  
39]\nclass = a'),  
Text(4129.2, 181.19999999999982, 'gini = 0.414\nsamples = 26\nvalue = [12, 29]\nclass =  
b'),  
Text(4352.4, 181.19999999999982, 'gini = 0.078\nsamples = 159\nvalue = [236, 10]\nclass =  
a')]
```



Conclusion

Linear Regression=0.8069102549582842

Ridge Regression=0.709408577546153

Lasso Regression=0.4113700257643299

ElasticNet Regression=0.4901906715101415

Logistic Regression=0.9888206926786497

Random Forest=0.9921700776675565

Random Forest is suitable for this dataset

