

Vijay(P2) 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_2000.csv")
2 df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002	NaN	21.320000
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.660000
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001	NaN	13.670000
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000	NaN	16.990000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002	NaN	15.260000
...
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000	NaN	13.210000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999	NaN	15.640000
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.620000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN	5.52	24.219996
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.910000

217296 rows × 16 columns



```
In [3]: 1 df=df.dropna()
```

```
In [4]: 1 df.columns
```

```
Out[4]: Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3', 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
              dtype='object')
```

In [5]: 1 df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32381 entries, 1 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   date      32381 non-null   object  
 1   BEN        32381 non-null   float64 
 2   CO         32381 non-null   float64 
 3   EBE        32381 non-null   float64 
 4   MXY        32381 non-null   float64 
 5   NMHC       32381 non-null   float64 
 6   NO_2       32381 non-null   float64 
 7   NOx        32381 non-null   float64 
 8   OXY        32381 non-null   float64 
 9   O_3         32381 non-null   float64 
 10  PM10       32381 non-null   float64 
 11  PXY        32381 non-null   float64 
 12  SO_2       32381 non-null   float64 
 13  TCH        32381 non-null   float64 
 14  TOL        32381 non-null   float64 
 15  station    32381 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.2+ MB
```

In [6]: 1 data=df[['BEN', 'TOL', 'TCH']]
2 data

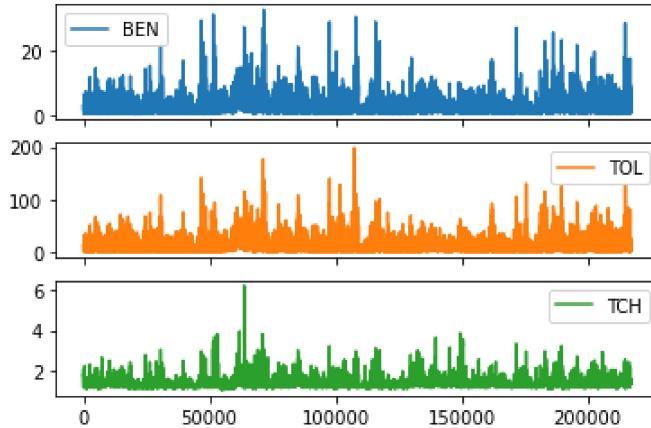
Out[6]:

	BEN	TOL	TCH
1	1.93	10.98	1.82
5	3.19	15.60	1.83
22	2.02	7.32	1.54
24	3.02	11.42	1.79
26	2.02	10.60	2.08
...
217269	1.24	4.45	1.36
217271	3.13	15.10	1.53
217273	2.50	16.65	1.34
217293	1.37	4.33	1.43
217295	3.11	15.51	1.54

32381 rows × 3 columns

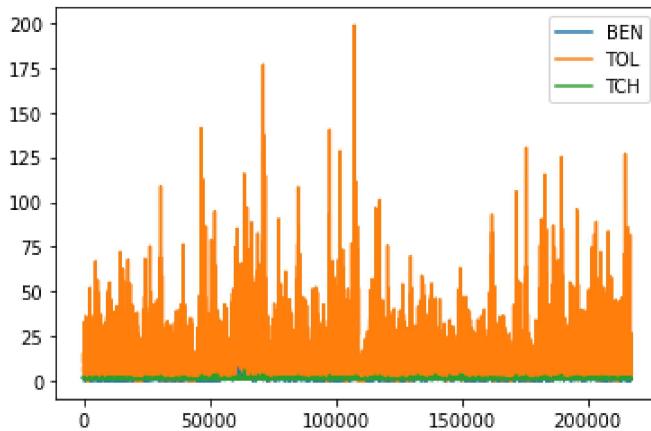
```
In [7]: 1 data.plot.line(subplots=True)
```

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



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

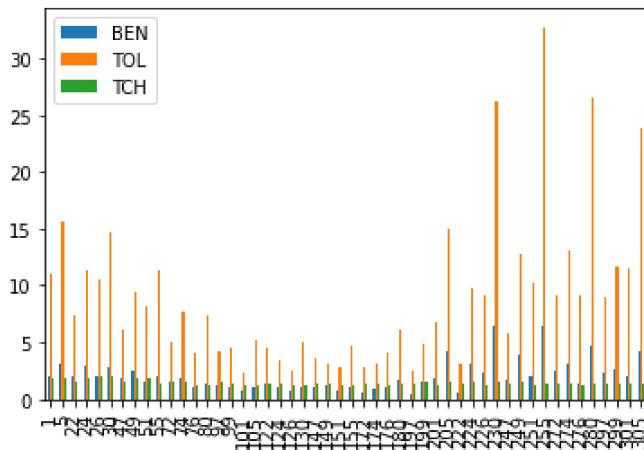
```
Out[8]: <AxesSubplot:>
```



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

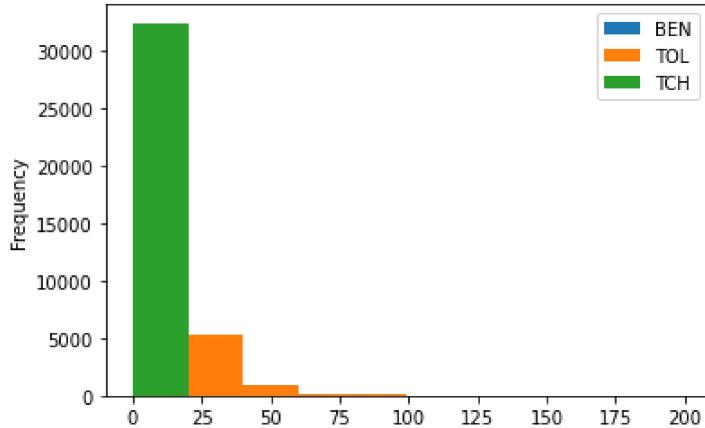
```
In [10]: 1 b.plot.bar()
```

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



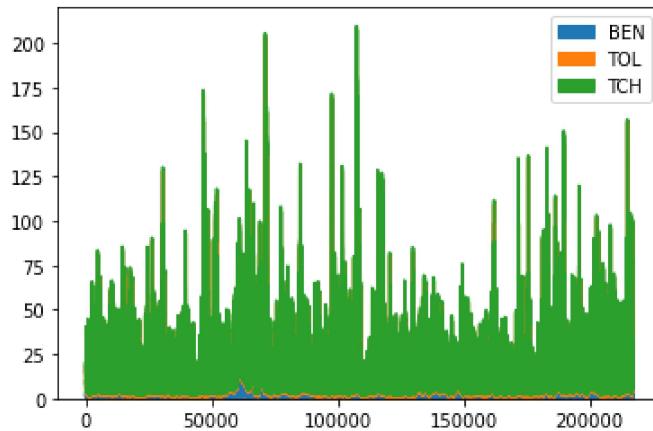
```
In [11]: 1 data.plot.hist()
```

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



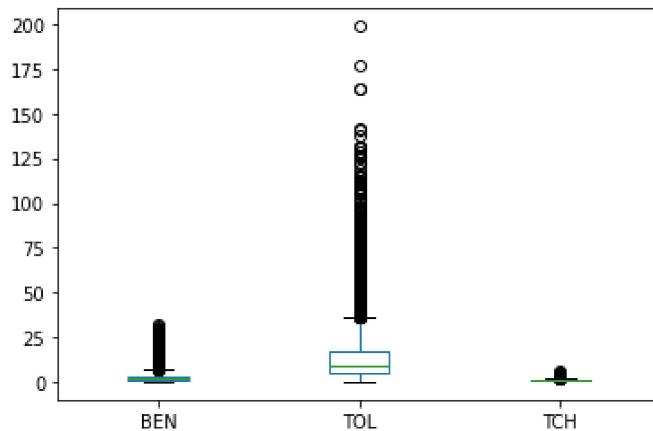
```
In [12]: 1 data.plot.area()
```

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



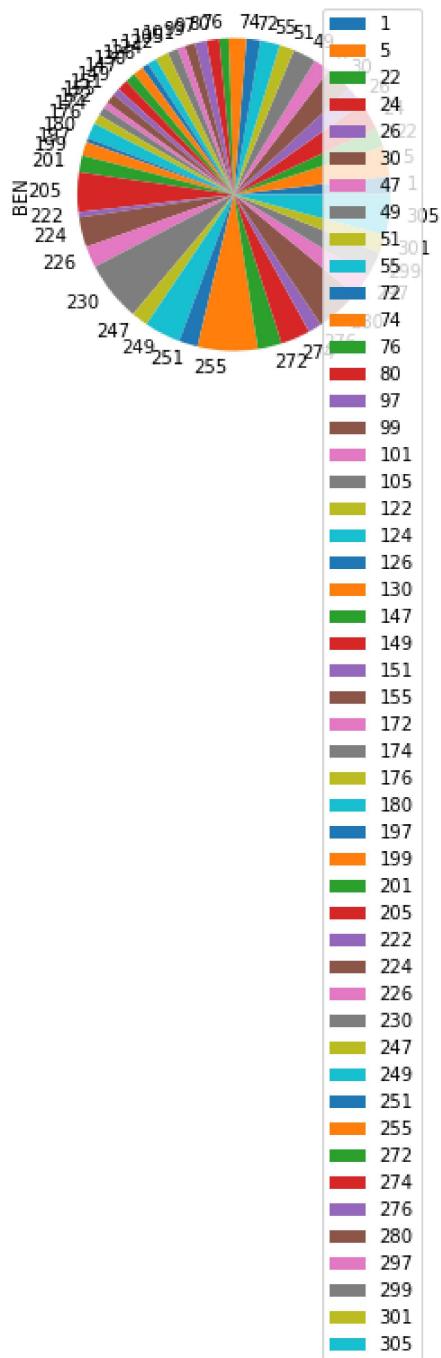
```
In [13]: 1 data.plot.box()
```

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



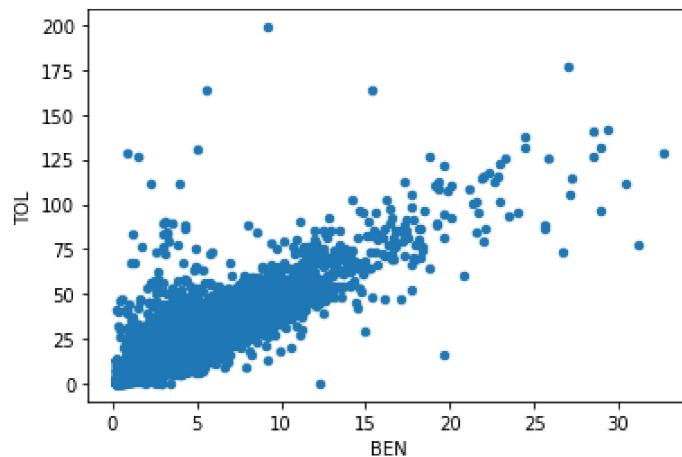
```
In [14]: 1 b.plot.pie(y='BEN' )
```

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()

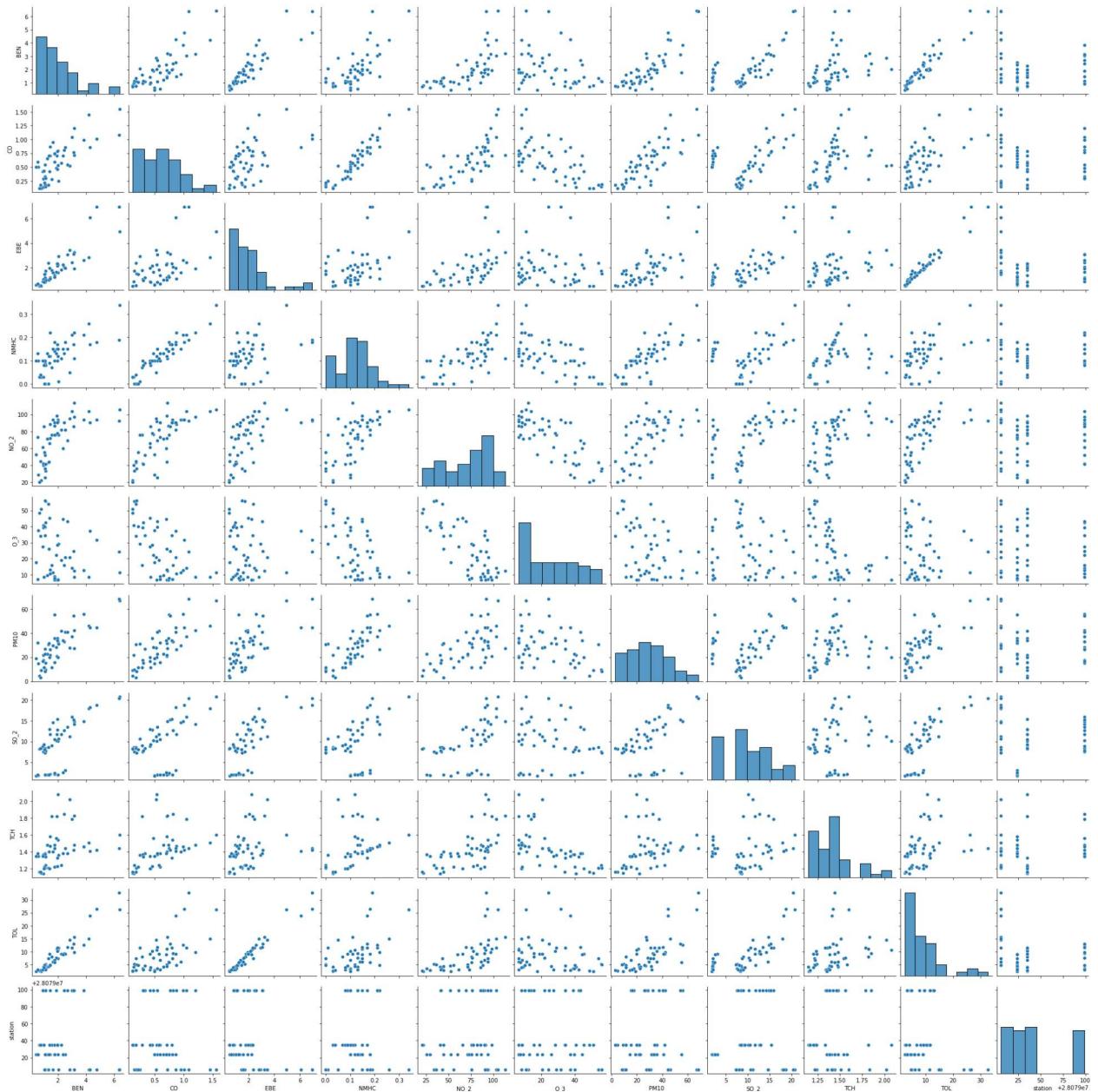
Out[16]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
count	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000
mean	2.479155	0.787323	2.914004	7.013636	0.155827	58.936796	126.009340
std	2.280959	0.610810	2.667881	6.774365	0.135731	31.472733	114.035078
min	0.180000	0.000000	0.180000	0.190000	0.000000	0.890000	1.710000
25%	0.970000	0.420000	1.140000	2.420000	0.080000	35.660000	48.720001
50%	1.840000	0.620000	2.130000	5.140000	0.130000	57.160000	96.830002
75%	3.250000	0.980000	3.830000	9.420000	0.200000	78.769997	167.500000
max	32.660000	8.460000	41.740002	99.879997	2.700000	263.600006	1336.000000

In [18]: 1 df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3', 'PM10',
2 'SO_2', 'TCH', 'TOL', 'station']]

```
In [19]: 1 sns.pairplot(df1[0:50])
```

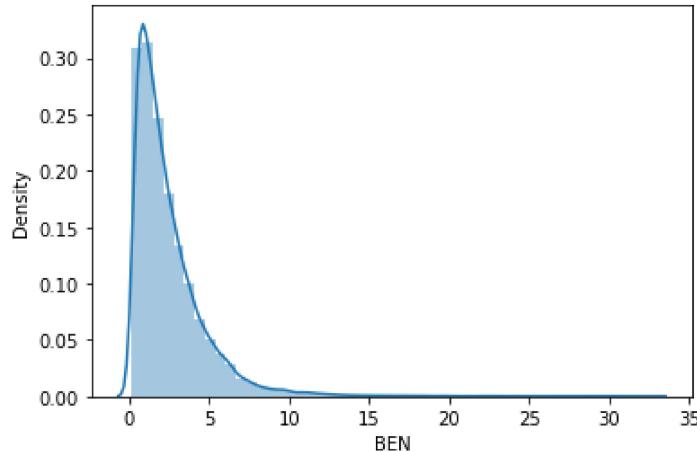
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x260f3977520>
```



In [20]: 1 sns.distplot(df1['BEN'])

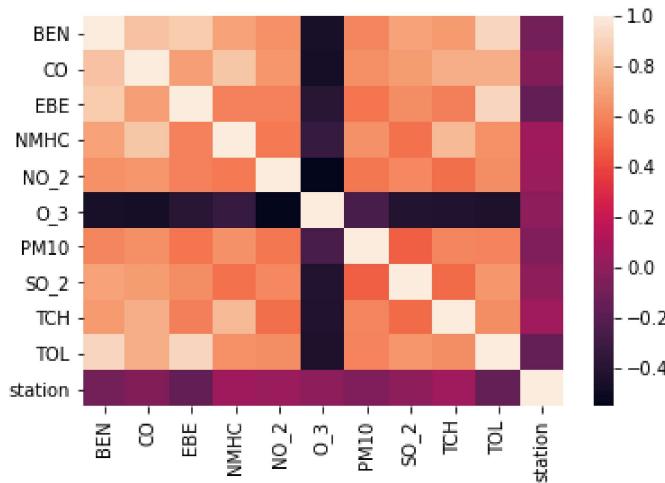
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[20]: <AxesSubplot:xlabel='BEN', ylabel='Density'>



In [21]: 1 sns.heatmap(df1.corr())

Out[21]: <AxesSubplot:>



In [23]: 1 x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3', 'PM10',
2 'SO_2', 'TCH', 'TOL']]
3 y=df['station']

In [24]: 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 [25]: 1 from sklearn.linear_model import LinearRegression
2 lr=LinearRegression()
3 lr.fit(x_train,y_train)

Out[25]: LinearRegression()

In [26]: 1 lr.intercept_

Out[26]: 28079002.638303623

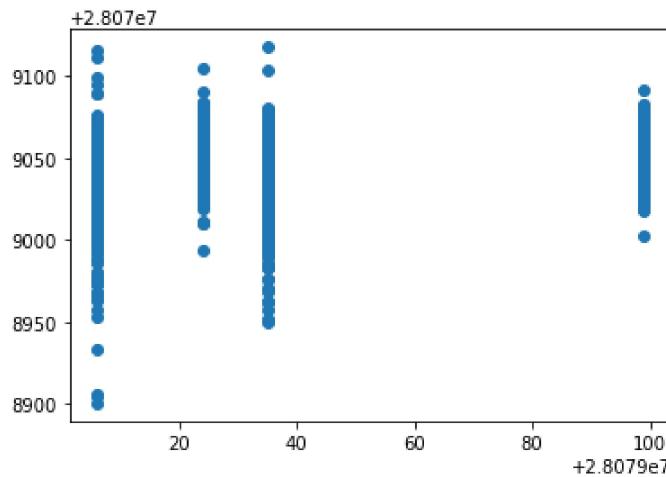
In [27]: 1 coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
2 coeff

Out[27]:

	Co-efficient
BEN	2.594159
CO	-21.875852
EBE	-2.073236
NMHC	78.858291
NO_2	0.229989
O_3	0.000040
PM10	-0.160225
SO_2	0.467244
TCH	31.525011
TOL	-0.998069

In [28]: 1 prediction =lr.predict(x_test)
2 plt.scatter(y_test,prediction)

Out[28]: <matplotlib.collections.PathCollection at 0x26081c04190>



In [29]: 1 lr.score(x_test,y_test)

Out[29]: 0.1362998219995376

In [30]: 1 lr.score(x_train,y_train)

Out[30]: 0.12314091359623547

In [31]: 1 from sklearn.linear_model import Ridge,Lasso

```
In [32]: 1 rr=Ridge(alpha=10)
          2 rr.fit(x_train,y_train)
```

Out[32]: Ridge(alpha=10)

```
In [33]: 1 rr.score(x_test,y_test)
```

Out[33]: 0.1353672419009878

```
In [34]: 1 rr.score(x_train,y_train)
```

Out[34]: 0.122953205809053

```
In [35]: 1 la=Lasso(alpha=10)
          2 la.fit(x_train,y_train)
```

Out[35]: Lasso(alpha=10)

```
In [36]: 1 la.score(x_test,y_test)
```

Out[36]: 0.05884379733131284

```
In [37]: 1 la.score(x_train,y_train)
```

Out[37]: 0.056550800067747375

```
In [38]: 1 from sklearn.linear_model import ElasticNet
          2 en=ElasticNet()
          3 en.fit(x_train,y_train)
```

Out[38]: ElasticNet()

```
In [39]: 1 en.coef_
```

Out[39]: array([0.80360807, 0. , -1.04511452, 0.05341397, 0.22009593,
 -0.00419588, -0.0224454 , 0.33499834, 0.85961246, -0.89121427])

```
In [40]: 1 en.intercept_
```

Out[40]: 28079038.08829956

```
In [41]: 1 prediction=en.predict(x_test)
```

```
In [42]: 1 en.score(x_test,y_test)
```

Out[42]: 0.06896982483407044

```
In [43]: 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)))
```

29.19207120339817
1163.9954947718343
34.117378193112

```
In [44]: 1 from sklearn.linear_model import LogisticRegression
```

```
In [45]: 1 feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
2 'PM10', 'SO_2', 'TCH', 'TOL']]
3 target_vector=df[ 'station']
```

```
In [46]: 1 feature_matrix.shape
```

```
Out[46]: (32381, 10)
```

```
In [47]: 1 target_vector.shape
```

```
Out[47]: (32381,)
```

```
In [48]: 1 from sklearn.preprocessing import StandardScaler
```

```
In [49]: 1 fs=StandardScaler().fit_transform(feature_matrix)
```

```
In [50]: 1 logr=LogisticRegression(max_iter=10000)
2 logr.fit(fs,target_vector)
```

```
Out[50]: LogisticRegression(max_iter=10000)
```

```
In [51]: 1 observation=[[1,2,3,4,5,6,7,8,9,10]]
```

```
In [52]: 1 prediction=logr.predict(observation)
2 print(prediction)
```

```
[28079006]
```

```
In [53]: 1 logr.classes_
```

```
Out[53]: array([28079006, 28079024, 28079035, 28079099], dtype=int64)
```

```
In [54]: 1 logr.score(fs,target_vector)
```

```
Out[54]: 0.7608782928260399
```

```
In [55]: 1 logr.predict_proba(observation)[0][0]
```

```
Out[55]: 0.9998759853067379
```

```
In [56]: 1 logr.predict_proba(observation)
```

```
Out[56]: array([[9.99875985e-01, 2.34979379e-50, 1.24014421e-04, 2.71856788e-10]])
```

```
In [57]: 1 from sklearn.ensemble import RandomForestClassifier
```

```
In [58]: 1 rfc=RandomForestClassifier()
2 rfc.fit(x_train,y_train)
```

```
Out[58]: RandomForestClassifier()
```

```
In [59]: 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 [60]: 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[60]: 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 [61]: 1 grid_search.best_score_
```

```
Out[61]: 0.7857584046589605
```

```
In [62]: 1 rfc_best=grid_search.best_estimator_
```

In [63]:

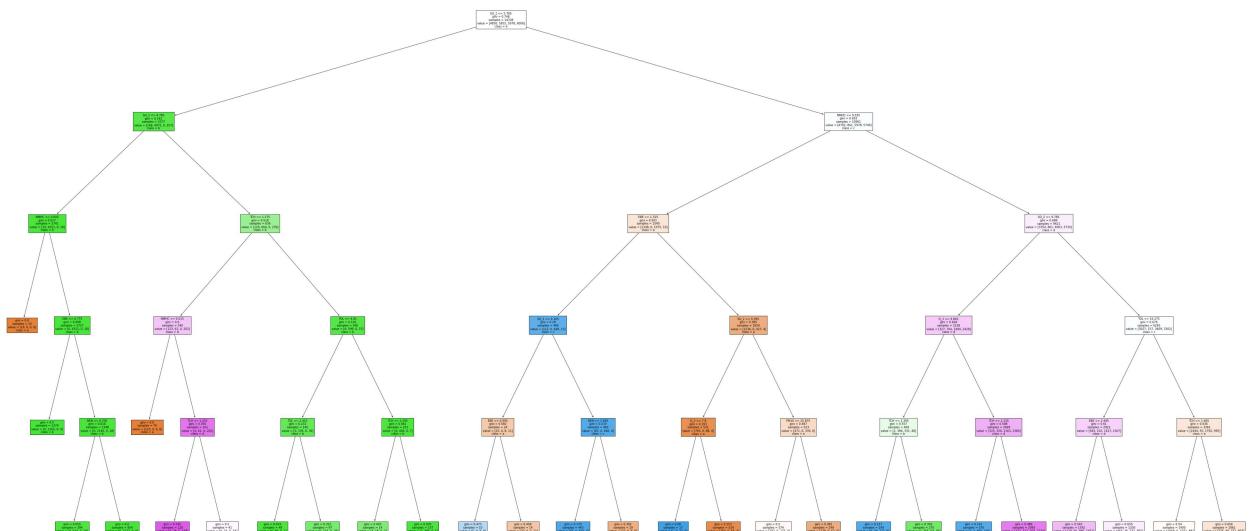
```
1 from sklearn.tree import plot_tree
2 plt.figure(figsize=(80,40))
3 plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d'])
```

```
Out[63]: [Text(1796.76, 1993.2, 'SO_2 <= 5.785\ngini = 0.748\nsamples = 14338\nvalue = [4850, 5832, 5978, 6006]\nclass = d'),  
 Text(558.0, 1630.8000000000002, 'SO_2 <= 4.785\ngini = 0.142\nsamples = 3377\nvalue = [148, 4971, 0, 257]\nclass = b'),  
 Text(178.56, 1268.4, 'NMHC <= 0.005\ngini = 0.017\nsamples = 2741\nvalue = [19, 4311, 0, 18]\nclass = b'),  
 Text(89.28, 906.0, 'gini = 0.0\nsamples = 14\nvalue = [19, 0, 0, 0]\nclass = a'),  
 Text(267.8400000000003, 906.0, 'EBE <= 0.775\ngini = 0.008\nsamples = 2727\nvalue = [0, 4311, 0, 18]\nclass = b'),  
 Text(178.56, 543.5999999999999, 'gini = 0.0\nsamples = 1379\nvalue = [0, 2165, 0, 0]\nclass = b'),  
 Text(357.12, 543.5999999999999, 'BEN <= 0.705\ngini = 0.016\nsamples = 1348\nvalue = [0, 2146, 0, 18]\nclass = b'),  
 Text(267.8400000000003, 181.1999999999982, 'gini = 0.055\nsamples = 394\nvalue = [0, 624, 0, 18]\nclass = b'),  
 Text(446.4, 181.1999999999982, 'gini = 0.0\nsamples = 954\nvalue = [0, 1522, 0, 0]\nclass = b'),  
 Text(937.44, 1268.4, 'TCH <= 1.275\ngini = 0.518\nsamples = 636\nvalue = [129, 660, 0, 239]\nclass = b'),  
 Text(624.96, 906.0, 'NMHC <= 0.015\ngini = 0.6\nsamples = 240\nvalue = [123, 61, 0, 202]\nclass = d'),  
 Text(535.6800000000001, 543.5999999999999, 'gini = 0.0\nsamples = 79\nvalue = [123, 0, 0, 0]\nclass = a'),  
 Text(714.24, 543.5999999999999, 'TCH <= 1.255\ngini = 0.356\nsamples = 161\nvalue = [0, 61, 0, 202]\nclass = d'),  
 Text(624.96, 181.1999999999982, 'gini = 0.245\nsamples = 120\nvalue = [0, 28, 0, 168]\nclass = d'),  
 Text(803.52, 181.1999999999982, 'gini = 0.5\nsamples = 41\nvalue = [0, 33, 0, 34]\nclass = d'),  
 Text(1249.92, 906.0, 'TOL <= 4.85\ngini = 0.126\nsamples = 396\nvalue = [6, 599, 0, 37]\nclass = b'),  
 Text(1071.3600000000001, 543.5999999999999, 'TOL <= 2.415\ngini = 0.231\nsamples = 145\nvalue = [0, 195, 0, 30]\nclass = b'),  
 Text(982.08, 181.1999999999982, 'gini = 0.024\nsamples = 48\nvalue = [0, 80, 0, 1]\nclass = b'),  
 Text(1160.64, 181.1999999999982, 'gini = 0.322\nsamples = 97\nvalue = [0, 115, 0, 29]\nclass = b'),  
 Text(1428.48, 543.5999999999999, 'TCH <= 1.295\ngini = 0.061\nsamples = 251\nvalue = [6, 404, 0, 7]\nclass = b'),  
 Text(1339.2, 181.1999999999982, 'gini = 0.403\nsamples = 14\nvalue = [4, 18, 0, 2]\nclass = b'),  
 Text(1517.76, 181.1999999999982, 'gini = 0.035\nsamples = 237\nvalue = [2, 386, 0, 5]\nclass = b'),  
 Text(3035.52, 1630.8000000000002, 'NMHC <= 0.035\ngini = 0.693\nsamples = 10961\nvalue = [4702, 861, 5978, 5749]\nclass = c'),  
 Text(2321.28, 1268.4, 'EBE <= 1.325\ngini = 0.501\nsamples = 1540\nvalue = [1348, 0, 1075, 19]\nclass = a'),  
 Text(1964.16, 906.0, 'SO_2 <= 6.365\ngini = 0.28\nsamples = 486\nvalue = [112, 0, 648, 15]\nclass = c'),  
 Text(1785.6, 543.5999999999999, 'EBE <= 0.995\ngini = 0.582\nsamples = 24\nvalue = [25, 0, 8, 11]\nclass = a'),  
 Text(1696.32, 181.1999999999982, 'gini = 0.473\nsamples = 10\nvalue = [5, 0, 8, 0]\nclass = c'),  
 Text(1874.88, 181.1999999999982, 'gini = 0.458\nsamples = 14\nvalue = [20, 0, 0, 11]\nclass = a'),  
 Text(2142.7200000000003, 543.5999999999999, 'BEN <= 1.695\ngini = 0.219\nsamples = 462\nvalue = [87, 0, 640, 4]\nclass = c'),  
 Text(2053.44, 181.1999999999982, 'gini = 0.179\nsamples = 443\nvalue = [65, 0, 632, 4]\nclass = c'),  
 Text(2232.0, 181.1999999999982, 'gini = 0.391\nsamples = 19\nvalue = [22, 0, 8, 0]\nclass = a'),  
 Text(2678.4, 906.0, 'SO_2 <= 9.395\ngini = 0.385\nsamples = 1054\nvalue = [1236, 0, 427, 4]\nclass = a')]
```

```

Text(2499.84, 543.5999999999999, 'O_3 <= 7.8\ngini = 0.193\nsamples = 531\nvalue = [765,
0, 88, 4]\nclass = a'),
Text(2410.56, 181.1999999999982, 'gini = 0.08\nsamples = 17\nvalue = [1, 0, 23, 0]\nclas
s = c'),
Text(2589.12, 181.1999999999982, 'gini = 0.153\nsamples = 514\nvalue = [764, 0, 65, 4]\n
class = a'),
Text(2856.96, 543.5999999999999, 'PM10 <= 25.875\ngini = 0.487\nsamples = 523\nvalue = [4
71, 0, 339, 0]\nclass = a'),
Text(2767.68, 181.1999999999982, 'gini = 0.5\nsamples = 374\nvalue = [292, 0, 277, 0]\n
class = a'),
Text(2946.240000000002, 181.1999999999982, 'gini = 0.382\nsamples = 149\nvalue = [179,
0, 62, 0]\nclass = a'),
Text(3749.76, 1268.4, 'SO_2 <= 9.785\ngini = 0.688\nsamples = 9421\nvalue = [3354, 861, 4
903, 5730]\nclass = d'),
Text(3392.64, 906.0, 'O_3 <= 9.845\ngini = 0.644\nsamples = 3138\nvalue = [327, 704, 149
4, 2428]\nclass = d'),
Text(3214.08, 543.5999999999999, 'TCH <= 1.395\ngini = 0.557\nsamples = 469\nvalue = [2,
384, 331, 48]\nclass = b'),
Text(3124.8, 181.1999999999982, 'gini = 0.217\nsamples = 199\nvalue = [0, 29, 277, 9]\n
class = c'),
Text(3303.36, 181.1999999999982, 'gini = 0.356\nsamples = 270\nvalue = [2, 355, 54, 39]
\nclass = b'),
Text(3571.2, 543.5999999999999, 'TCH <= 1.235\ngini = 0.588\nsamples = 2669\nvalue = [32
5, 320, 1163, 2380]\nclass = d'),
Text(3481.92, 181.1999999999982, 'gini = 0.216\nsamples = 576\nvalue = [3, 3, 805, 106]
\nclass = c'),
Text(3660.48, 181.1999999999982, 'gini = 0.486\nsamples = 2093\nvalue = [322, 317, 358,
2274]\nclass = d'),
Text(4106.88, 906.0, 'TOL <= 16.275\ngini = 0.676\nsamples = 6283\nvalue = [3027, 157, 34
09, 3302]\nclass = c'),
Text(3928.32, 543.5999999999999, 'EBE <= 2.445\ngini = 0.61\nsamples = 2922\nvalue = [58
3, 102, 1617, 2307]\nclass = d'),
Text(3839.04, 181.1999999999982, 'gini = 0.543\nsamples = 1592\nvalue = [119, 66, 886, 1
452]\nclass = d'),
Text(4017.6, 181.1999999999982, 'gini = 0.659\nsamples = 1330\nvalue = [464, 36, 731, 85]
\nclass = d'),
Text(4285.440000000005, 543.5999999999999, 'TCH <= 1.465\ngini = 0.636\nsamples = 3361\n
value = [2444, 55, 1792, 995]\nclass = a'),
Text(4196.16, 181.1999999999982, 'gini = 0.54\nsamples = 1400\nvalue = [1069, 6, 1021, 8
8]\nclass = a'),
Text(4374.72, 181.1999999999982, 'gini = 0.656\nsamples = 1961\nvalue = [1375, 49, 771,
907]\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