mk 3/08/2023

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_201
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210091	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN	28
210092	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN	28
210093	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN	28
210094	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN	28
210095	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN	28

210096 rows × 14 columns

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

Out[3]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28(
6	2015-10-01 01:00:00	0.5	0.3	0.3	0.12	6.0	83.0	1.0	19.0	12.0	3.0	1.29	4.8	28(
25	2015-10-01 02:00:00	1.6	0.7	1.3	0.38	81.0	105.0	4.0	36.0	19.0	13.0	1.93	6.9	28(
30	2015-10-01 02:00:00	0.4	0.3	0.3	0.11	5.0	72.0	2.0	16.0	10.0	2.0	1.27	7.8	28(
49	2015-10-01 03:00:00	2.2	0.8	1.8	0.41	111.0	104.0	4.0	35.0	20.0	14.0	2.05	13.9	28(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210030	2015-07-31 22:00:00	0.1	0.1	0.1	0.06	1.0	10.0	69.0	10.0	3.0	2.0	1.18	0.2	28(
210049	2015-07-31 23:00:00	0.4	0.3	0.1	0.12	3.0	28.0	56.0	15.0	7.0	12.0	1.45	1.2	28(
210054	2015-07-31 23:00:00	0.1	0.1	0.1	0.06	1.0	10.0	63.0	5.0	1.0	2.0	1.18	0.2	28(
210073	2015-08-01 00:00:00	0.1	0.3	0.1	0.11	2.0	23.0	59.0	5.0	2.0	11.0	1.44	0.6	28(
210078	2015-08-01 00:00:00	0.1	0.1	0.1	0.06	1.0	8.0	65.0	7.0	1.0	2.0	1.18	0.4	28(

16026 rows × 14 columns

In [4]:
```
df=df.head(2000)
df
```

Out[4]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	280
6	2015-10-01 01:00:00	0.5	0.3	0.3	0.12	6.0	83.0	1.0	19.0	12.0	3.0	1.29	4.8	280
25	2015-10-01 02:00:00	1.6	0.7	1.3	0.38	81.0	105.0	4.0	36.0	19.0	13.0	1.93	6.9	280
30	2015-10-01 02:00:00	0.4	0.3	0.3	0.11	5.0	72.0	2.0	16.0	10.0	2.0	1.27	7.8	280
49	2015-10-01 03:00:00	2.2	0.8	1.8	0.41	111.0	104.0	4.0	35.0	20.0	14.0	2.05	13.9	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
24942	2015-11-13 08:00:00	0.5	0.1	0.3	0.10	28.0	23.0	2.0	14.0	11.0	2.0	1.32	4.1	280
24961	2015-11-13 09:00:00	1.3	0.6	0.4	0.12	101.0	85.0	5.0	34.0	26.0	17.0	1.47	3.3	280
24966	2015-11-13 09:00:00	0.6	0.1	0.4	0.10	35.0	21.0	3.0	16.0	12.0	3.0	1.32	4.5	280
24985	2015-11-13 10:00:00	2.2	2.2	1.0	0.21	123.0	92.0	7.0	27.0	15.0	19.0	1.56	7.2	280
24990	2015-11-13 10:00:00	0.5	0.1	0.3	0.10	43.0	19.0	6.0	7.0	3.0	3.0	1.33	4.1	280

2000 rows × 14 columns

In [5]:
```
df.columns
```

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2000 entries, 1 to 24990
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     2000 non-null   object 
 1   BEN      2000 non-null   float64
 2   CO       2000 non-null   float64
 3   EBE      2000 non-null   float64
 4   NMHC     2000 non-null   float64
 5   NO       2000 non-null   float64
 6   NO_2     2000 non-null   float64
 7   O_3      2000 non-null   float64
 8   PM10     2000 non-null   float64
 9   PM25     2000 non-null   float64
 10  SO_2     2000 non-null   float64
 11  TCH      2000 non-null   float64
 12  TOL      2000 non-null   float64
 13  station  2000 non-null   int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 234.4+ KB
```

In [7]:
```
data=df[['BEN','TOL','TCH']]
data
```

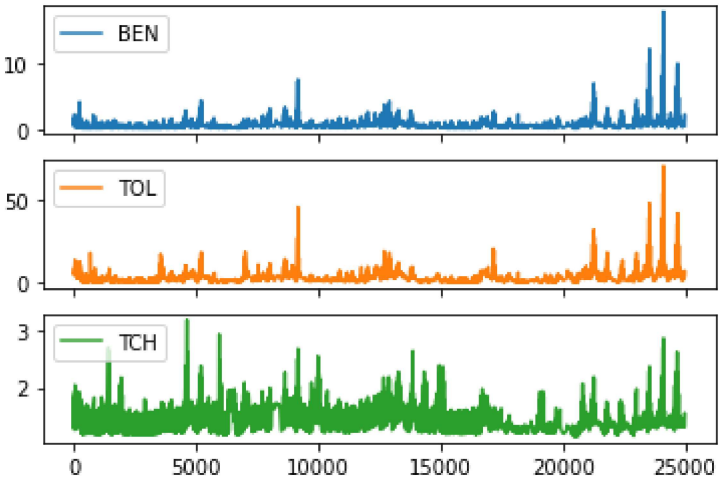
Out[7]:

	BEN	TOL	TCH
1	2.0	8.3	1.83
6	0.5	4.8	1.29
25	1.6	6.9	1.93
30	0.4	7.8	1.27
49	2.2	13.9	2.05
...	...	...	...
24942	0.5	4.1	1.32
24961	1.3	3.3	1.47
24966	0.6	4.5	1.32
24985	2.2	7.2	1.56
24990	0.5	4.1	1.33

2000 rows × 3 columns

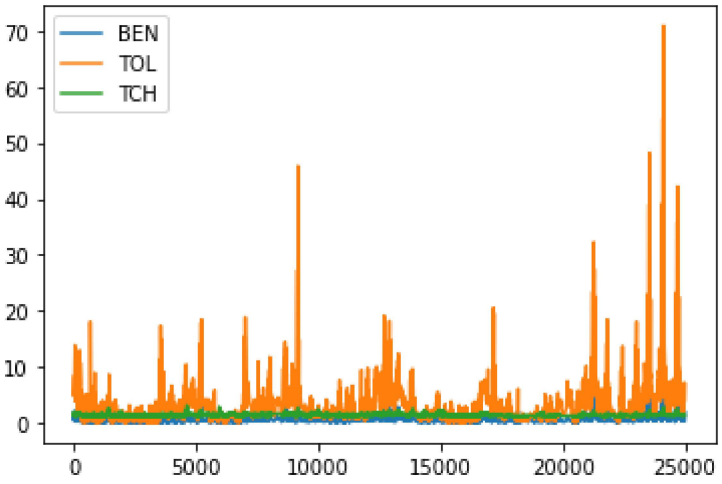
In [8]:
```
data.plot.line(subplots=True)
```

Out[8]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [9]:
```
data.plot.line()
```

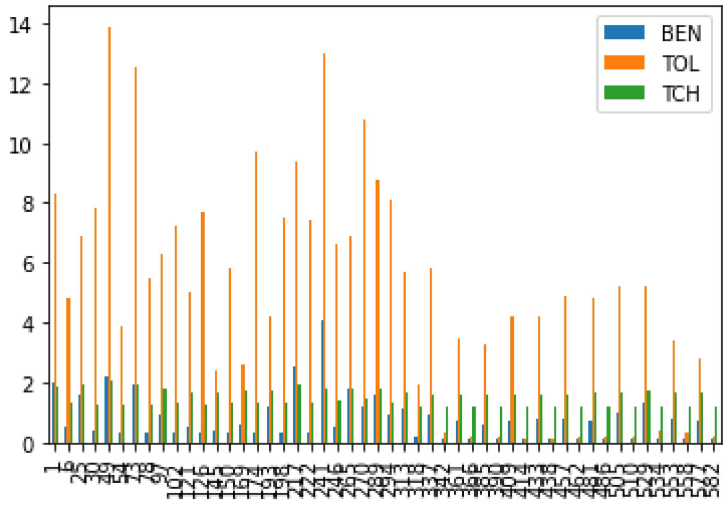
Out[9]:
```
<AxesSubplot:>
```



In [10]:
```
b=data[0:50]
```

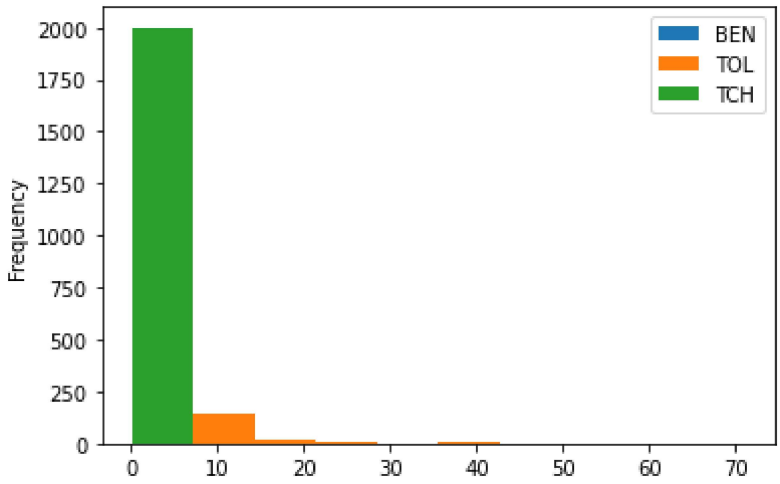
In [11]:
```
b.plot.bar()
```

Out[11]:
```
<AxesSubplot:>
```



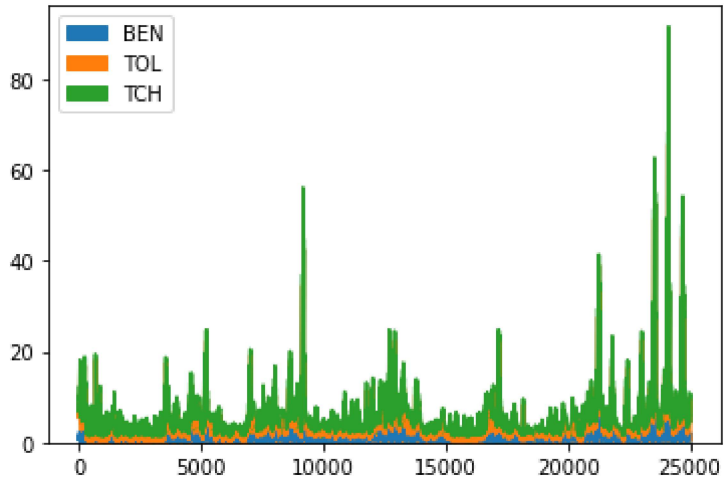
In [12]:
```
data.plot.hist()
```

Out[12]:
```
<AxesSubplot:ylabel='Frequency'>
```



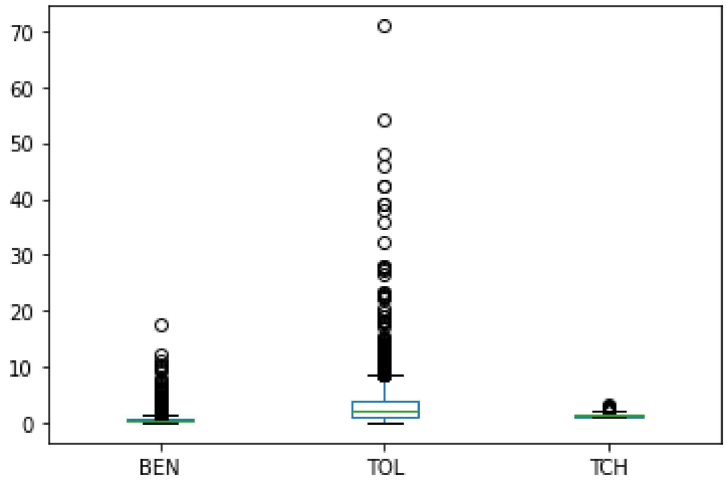
In [13]:
```
data.plot.area()
```

Out[13]: <AxesSubplot:>



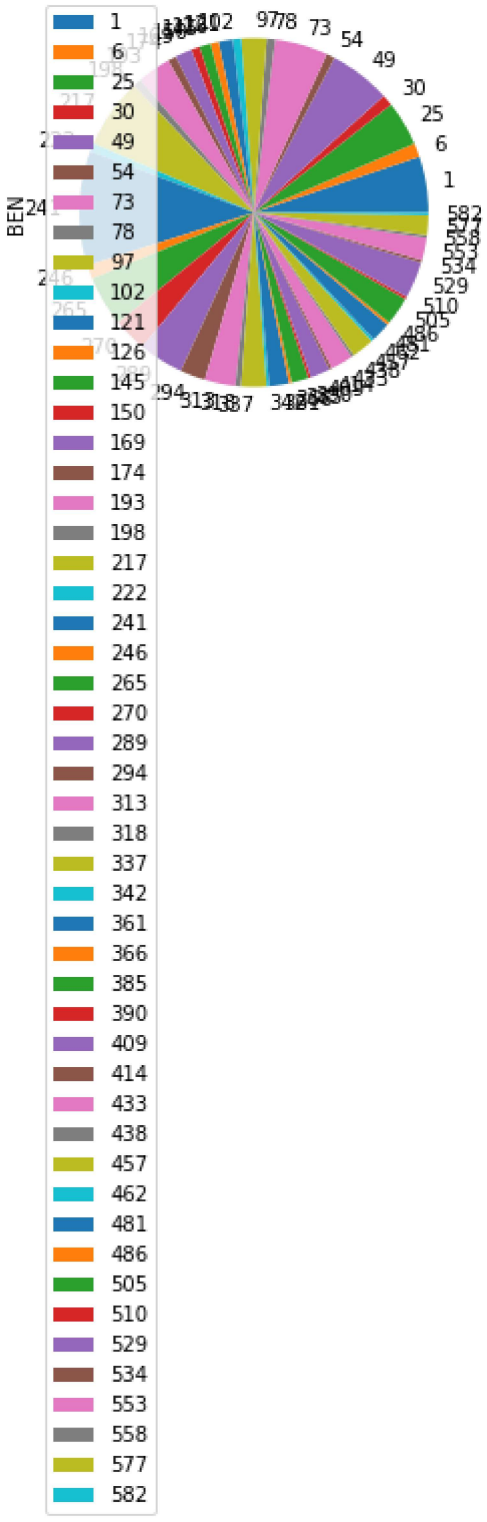
In [14]:
```
data.plot.box()
```

Out[14]: <AxesSubplot:>



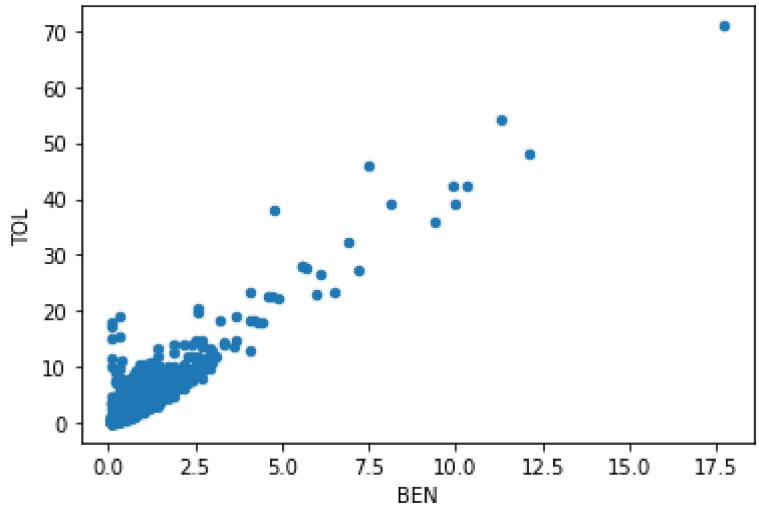
In [15]:
```
b.plot.pie(y='BEN' )
```

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



In [16]:
```
data.plot.scatter(x='BEN',y='TOL')
```

Out[16]:
```
<AxesSubplot:xlabel='BEN', ylabel='TOL'>
```



In [17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2000 entries, 1 to 24990
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     2000 non-null   object 
 1   BEN      2000 non-null   float64
 2   CO       2000 non-null   float64
 3   EBE      2000 non-null   float64
 4   NMHC     2000 non-null   float64
 5   NO       2000 non-null   float64
 6   NO_2     2000 non-null   float64
 7   O_3      2000 non-null   float64
 8   PM10     2000 non-null   float64
 9   PM25     2000 non-null   float64
 10  SO_2     2000 non-null   float64
 11  TCH      2000 non-null   float64
 12  TOL      2000 non-null   float64
 13  station  2000 non-null   int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 234.4+ KB
```

In [18]:
```
df.describe()
```

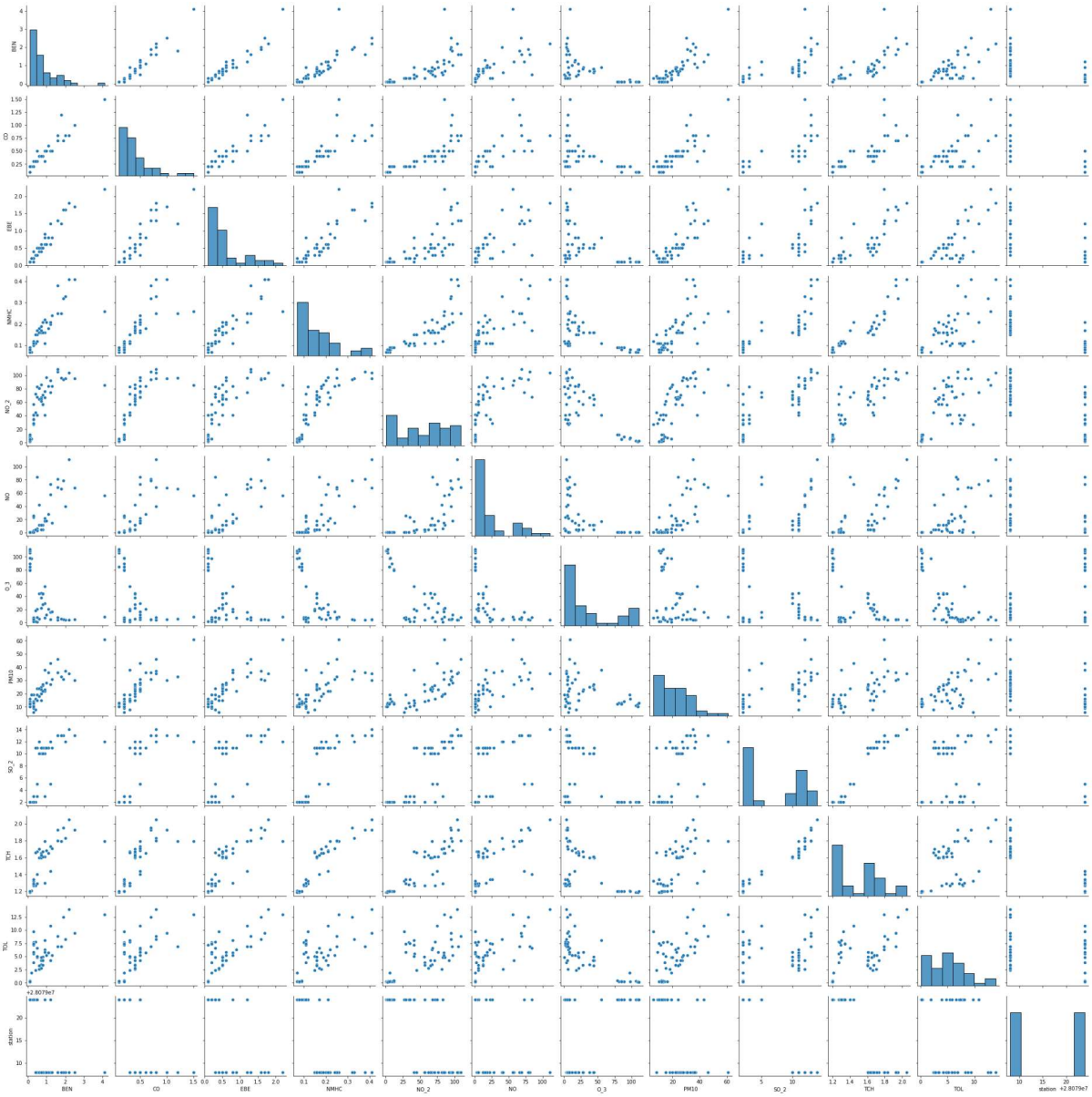
Out[18]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.00000
mean	0.723650	0.368000	0.442600	0.125890	31.691000	46.543000	27.49200
std	0.979983	0.303516	0.726091	0.089286	65.259819	35.883241	27.00825
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	1.00000
25%	0.300000	0.200000	0.100000	0.080000	3.000000	21.000000	6.00000
50%	0.500000	0.300000	0.300000	0.110000	12.000000	41.000000	17.00000
75%	0.800000	0.400000	0.500000	0.150000	34.000000	64.000000	43.00000
max	17.700001	4.500000	12.100000	1.090000	960.000000	369.000000	114.00000

In [19]:
```
df1=df[['date','BEN', 'CO', 'EBE','NMHC', 'NO_2', 'NO',  'O_3',
 'PM10','SO_2', 'TCH', 'TOL', 'station']]
```

In [20]:
```
sns.pairplot(df1[0:50])
```

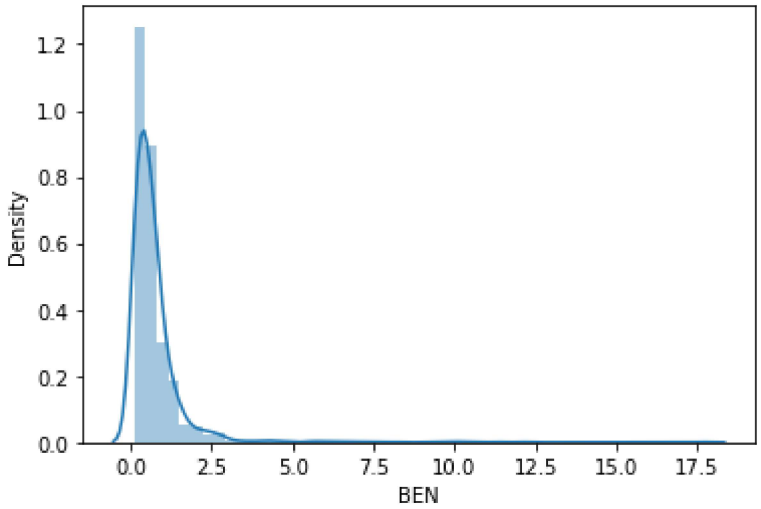
Out[20]: <seaborn.axisgrid.PairGrid at 0x1dd8c0201c0>



In [21]:
```
sns.distplot(df1['BEN'])
```

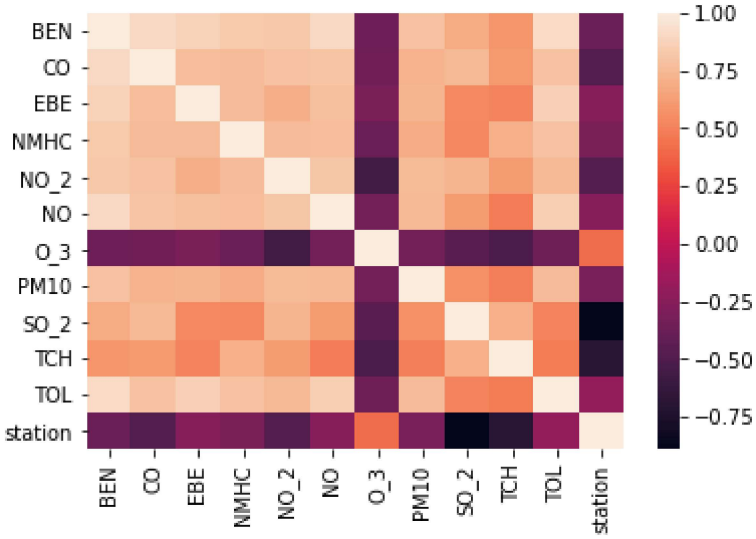
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[21]: <AxesSubplot:xlabel='BEN', ylabel='Density'>



In [22]:
```
sns.heatmap(df1.corr())
```

Out[22]: <AxesSubplot:>



In [23]:
```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [24]:
```
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)
```

In [25]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
lr.intercept_
```

Out[26]:
```
28079029.470978152
```

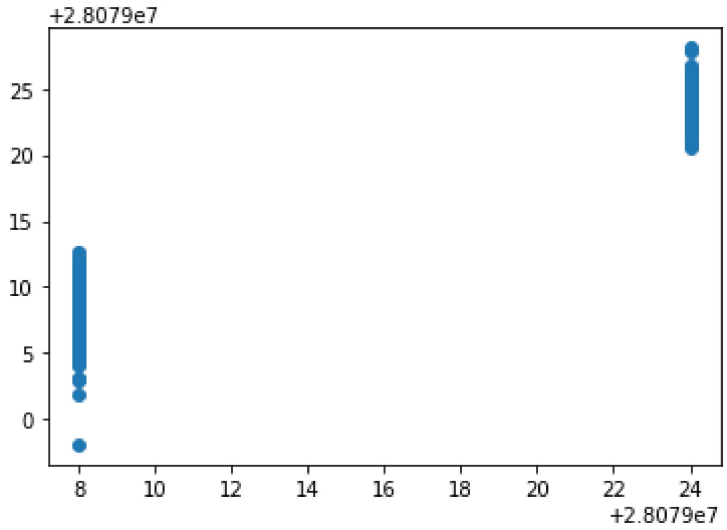
In [27]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[27]:

	Co-efficient
BEN	-0.520886
CO	1.192219
EBE	-0.098965
NMHC	-4.122017
NO	0.054842
NO_2	0.001621
O_3	0.017877
PM10	-0.015174
PM25	0.049981
SO_2	-1.451909
TCH	-3.181445
TOL	0.211179

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

Out[28]: <matplotlib.collections.PathCollection at 0x1dd960aaa90>



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

Out[29]: 0.9619693319707813

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

Out[30]: 0.9595251289143231

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

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

Out[32]: Ridge(alpha=10)

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

Out[33]: 0.9618823117116023

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

Out[34]: 0.9590779325246699

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

Out[35]: Lasso(alpha=10)

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

Out[36]:
```
0.8693202853038701
```

In [37]:
```
la.score(x_test,y_test)
```

Out[37]:
```
0.8698575598959035
```

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

Out[38]:
```
ElasticNet()
```

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

Out[39]:
```
array([-0.        , -0.        , -0.        , -0.        ,  0.05746809,
       -0.00808583,  0.02231745,  0.        ,  0.02088112, -1.45911087,
       -0.        ,  0.01797795])
```

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

Out[40]:
```
28079025.293215435
```

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

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

Out[42]:
```
0.9522528196094526
```

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

```
1.379812171322604
3.055276288187032
1.7479348638284644
```

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

In [45]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
 'PM10']]
target_vector=df[ 'station']
```

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

Out[46]: (2000, 7)

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

Out[47]: (2000,)

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

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

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

Out[50]: LogisticRegression(max_iter=10000)

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

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

```
[28079024]
```

In [53]:
```
logr.score(fs,target_vector)
```

Out[53]: 0.843

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

Out[54]: 0.07188798142584296

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

Out[55]: array([[0.07188798, 0.92811202]])

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="acc
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')
```

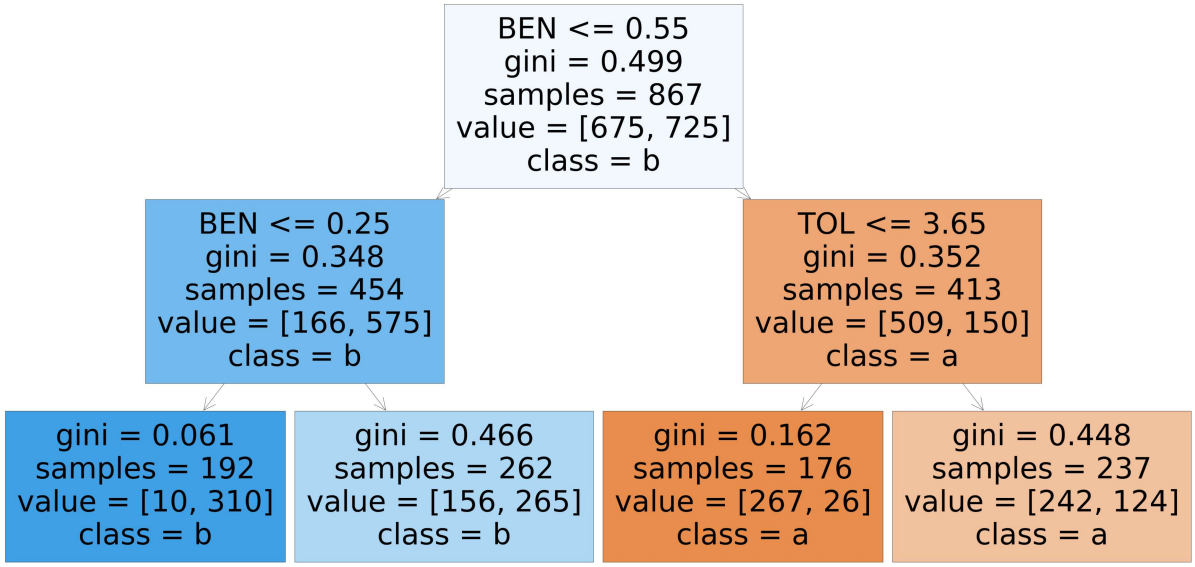
In [60]:
```
grid_search.best_score_
```

Out[60]:
```
1.0
```

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(2232.0, 1812.0, 'BEN <= 0.55\ngini = 0.499\nsamples = 867\nvalue = [67
5, 725]\nclass = b'),
 Text(1116.0, 1087.2, 'BEN <= 0.25\ngini = 0.348\nsamples = 454\nvalue = [16
6, 575]\nclass = b'),
 Text(558.0, 362.39999999999986, 'gini = 0.061\nsamples = 192\nvalue = [10, 3
10]\nclass = b'),
 Text(1674.0, 362.39999999999986, 'gini = 0.466\nsamples = 262\nvalue = [156,
265]\nclass = b'),
 Text(3348.0, 1087.2, 'TOL <= 3.65\ngini = 0.352\nsamples = 413\nvalue = [50
9, 150]\nclass = a'),
 Text(2790.0, 362.39999999999986, 'gini = 0.162\nsamples = 176\nvalue = [267,
26]\nclass = a'),
 Text(3906.0, 362.39999999999986, 'gini = 0.448\nsamples = 237\nvalue = [242,
124]\nclass = a')]
```



Conclusion

Linear Regression =0.9595251289143231

Ridge Regression =0.9590779325246699

Lasso Regression =0.8693202853038701

ElasticNet Regression =0.9522528196094526

Logistic Regression =0.07188798142584296

Randomforest =1.0

Randomforest is suitable for this dataset

In [ ]:

In [ ]: