mk 3/08/2023

In [444]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [445]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_201
df
```

Out[445]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	28
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4	28
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	28
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	28
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	28
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	28
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	28
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	28
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	28

210720 rows × 14 columns

In [446]:
```
df=df.dropna()
```

In [447]:
```
df=df.head(2000)
```

In [448]:
```
df.columns
```

Out[448]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [449]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2000 entries, 6 to 36678
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 [450]:
```
data=df[['BEN', 'TOL']]
data
```

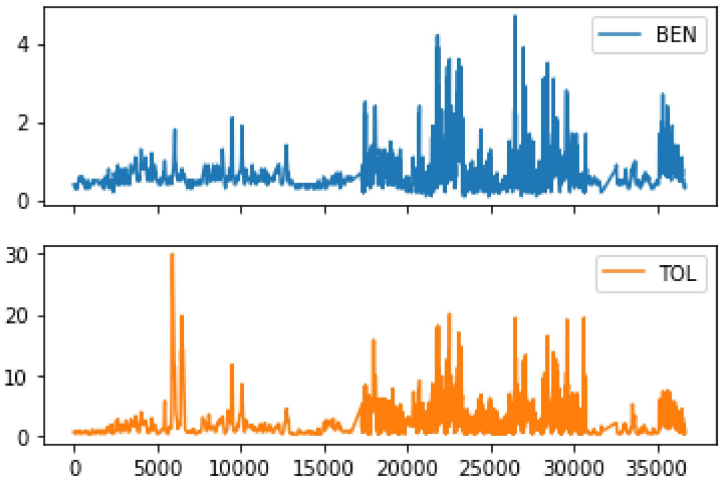
Out[450]:

	BEN	TOL
6	0.4	0.6
30	0.4	0.5
54	0.4	0.5
78	0.3	0.4
102	0.4	0.5
...	...	...
36630	0.4	0.3
36649	0.4	1.7
36654	0.4	0.3
36673	0.3	1.1
36678	0.4	0.4

2000 rows × 2 columns

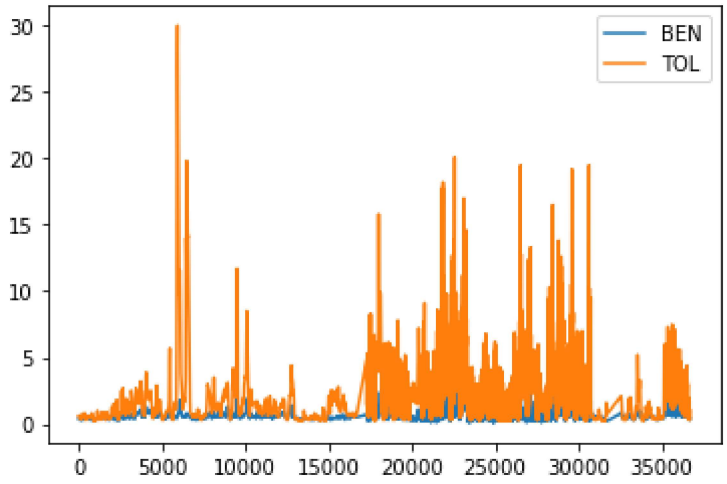
In [451]:
```
data.plot.line(subplots=True)
```

Out[451]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [452]:
```
data.plot.line()
```

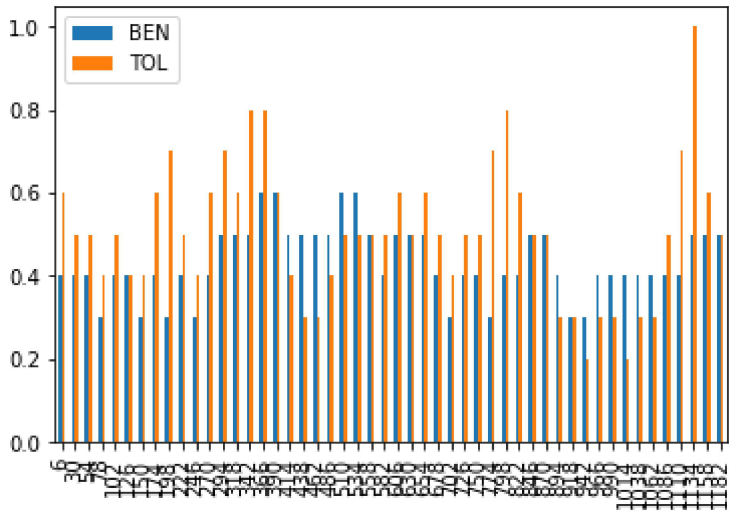
Out[452]: <AxesSubplot:>



In [453]:
```
b=data[0:50]
```

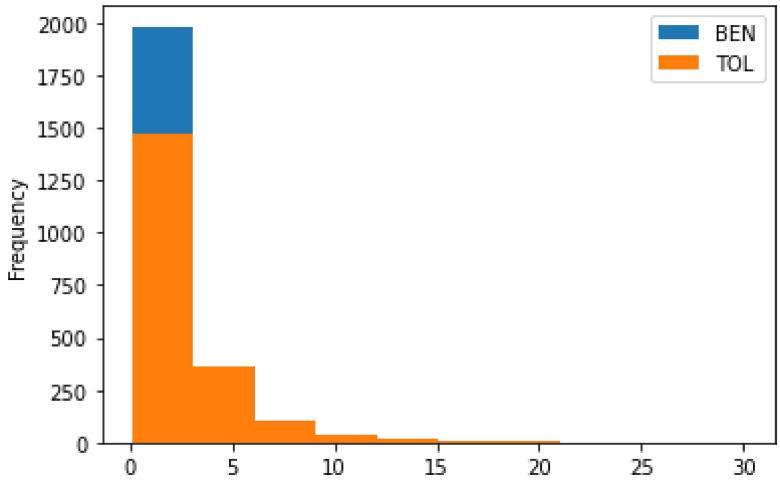
In [454]:
```
b.plot.bar()
```

Out[454]: <AxesSubplot:>



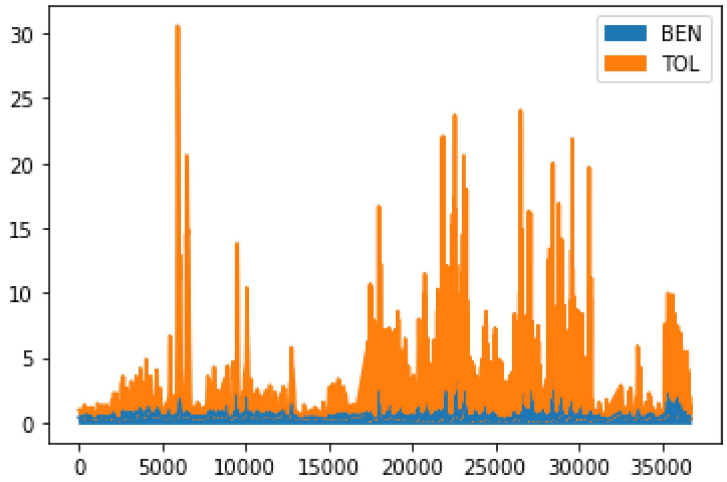
In [455]:
```
data.plot.hist()
```

Out[455]: <AxesSubplot:ylabel='Frequency'>



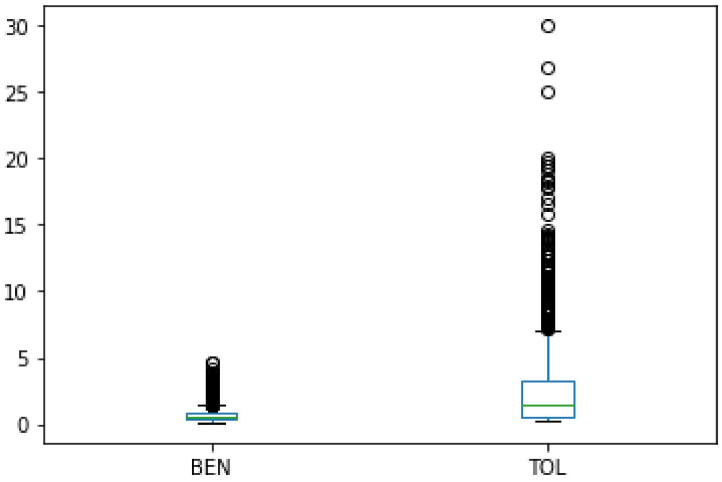
In [456]:
```
data.plot.area()
```

Out[456]: <AxesSubplot:>



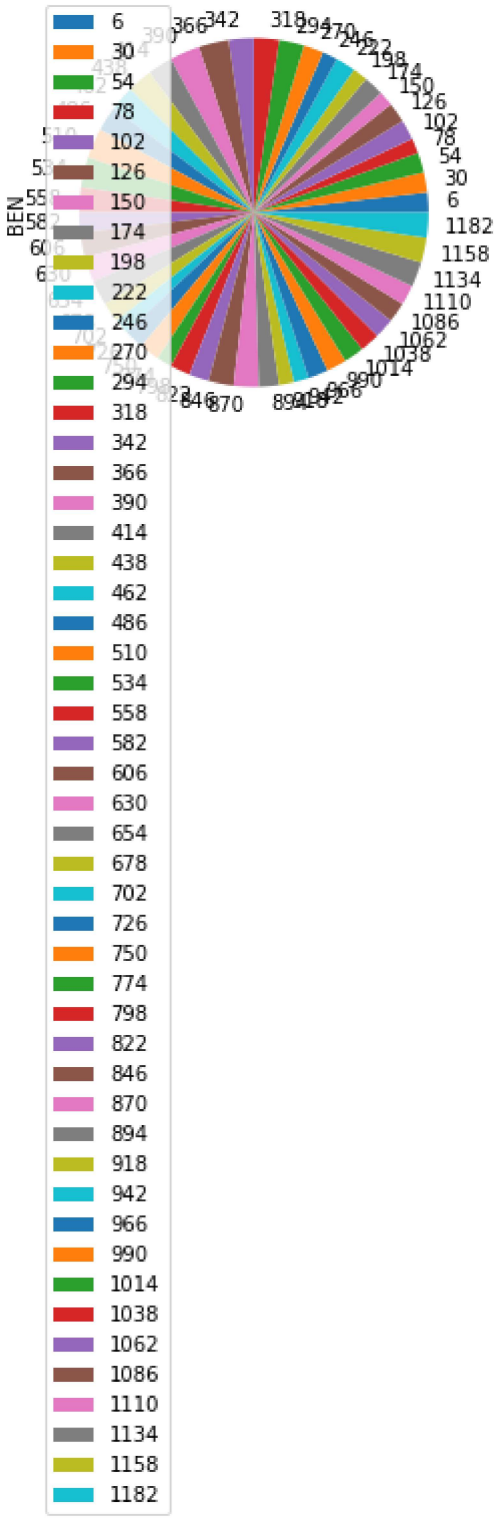
In [457]:
```
data.plot.box()
```

Out[457]: <AxesSubplot:>



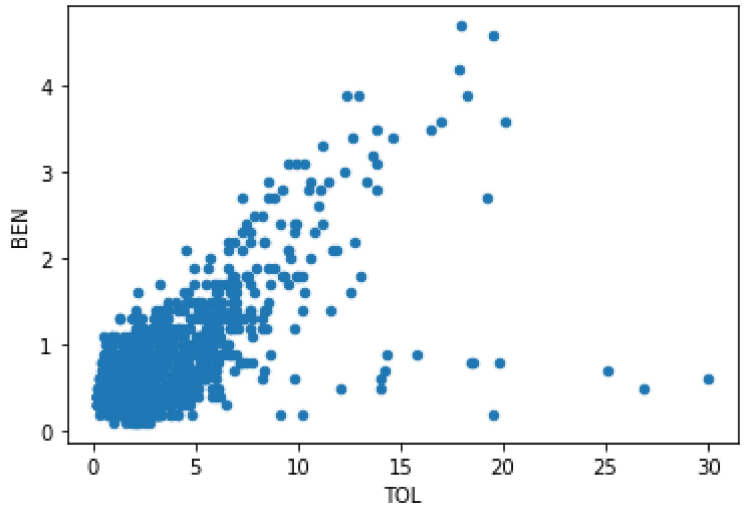
In [458]:
```
b.plot.pie(y='BEN' )
```

Out[458]: <AxesSubplot:ylabel='BEN'>



In [459]:
```
data.plot.scatter(x='TOL' ,y='BEN')
```

Out[459]: <AxesSubplot:xlabel='TOL', ylabel='BEN'>



In [460]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2000 entries, 6 to 36678
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 [461]:
```
df.describe()
```

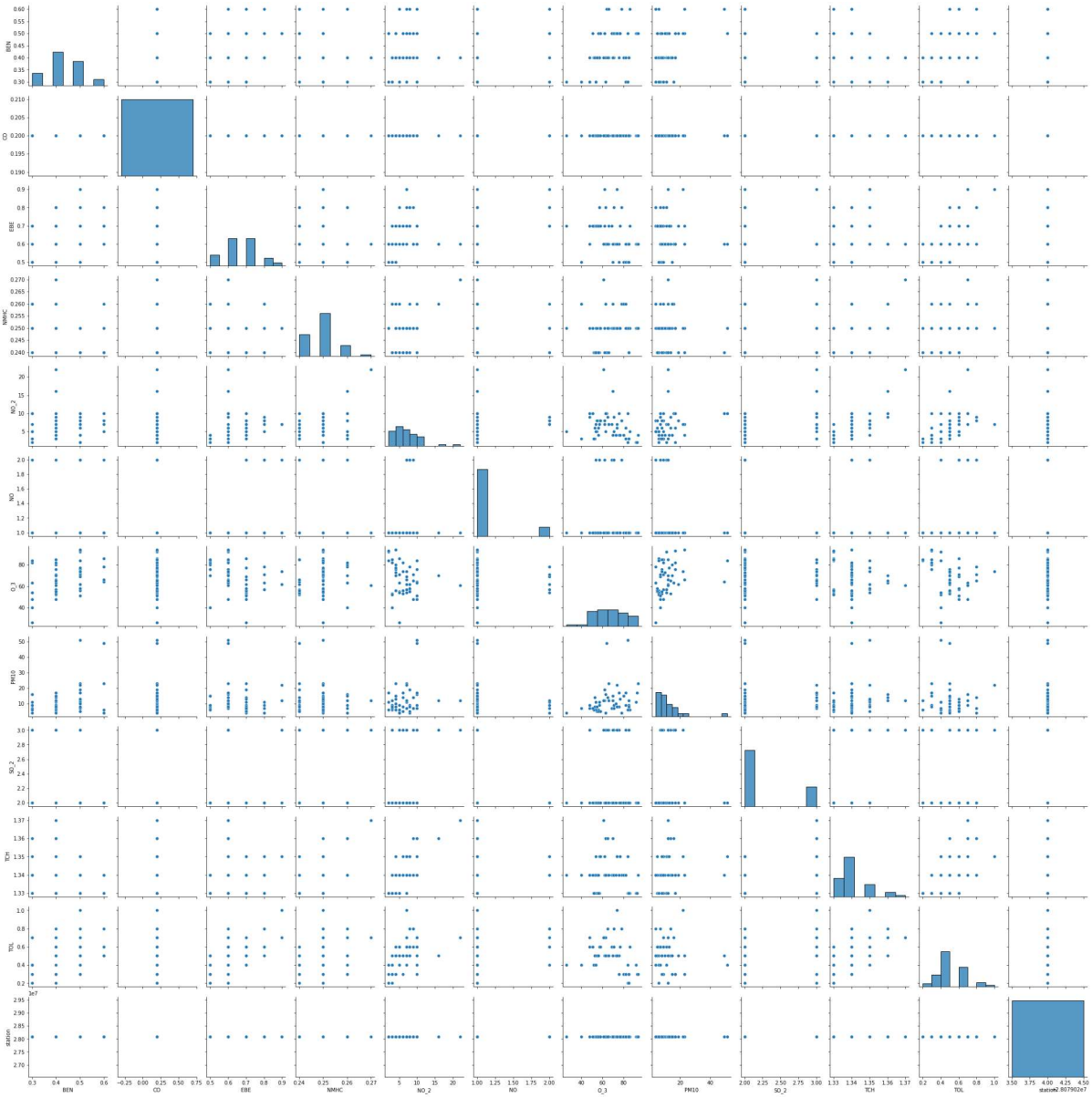
Out[461]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.00000
mean	0.681600	0.245550	1.333900	0.233980	8.079500	26.677500	54.61650
std	0.494675	0.113651	0.705102	0.055073	17.508954	25.422188	27.73402
min	0.100000	0.100000	0.300000	0.070000	1.000000	1.000000	2.00000
25%	0.400000	0.200000	0.900000	0.210000	1.000000	7.000000	34.00000
50%	0.500000	0.200000	1.200000	0.230000	1.000000	19.000000	55.00000
75%	0.800000	0.300000	1.600000	0.250000	8.000000	39.000000	75.00000
max	4.700000	1.500000	9.300000	0.580000	227.000000	225.000000	146.00000

In [462]:
```
df1=df[['BEN', 'CO', 'EBE','NMHC', 'NO_2', 'NO',  'O_3',
 'PM10','SO_2', 'TCH', 'TOL', 'station']]
```

In [463]:
```
sns.pairplot(df1[0:50])
```

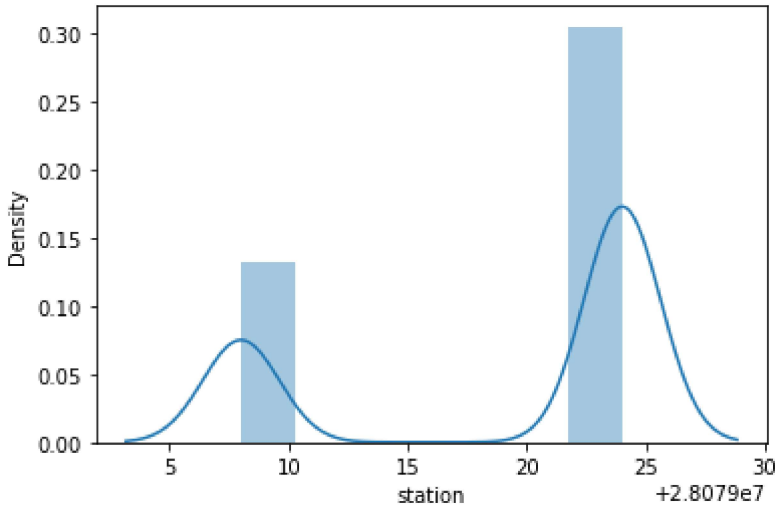
Out[463]: <seaborn.axisgrid.PairGrid at 0x1d8a3d874f0>



In [464]:
```
sns.distplot(df1['station'])
```

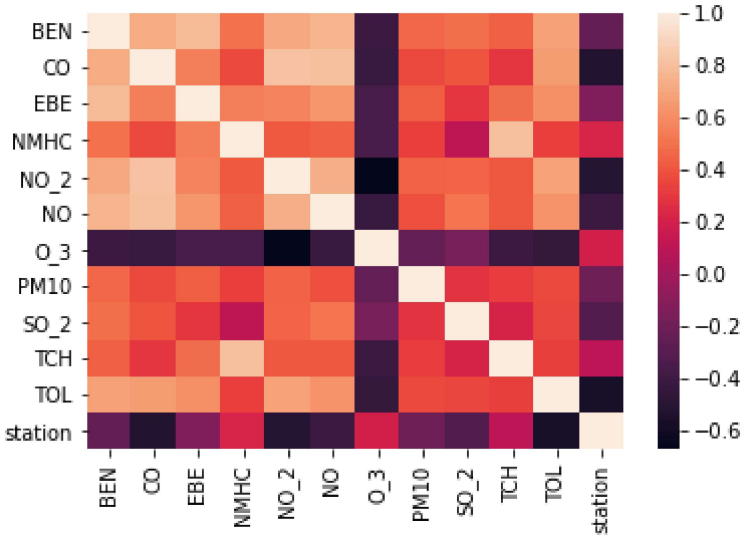
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[464]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [465]:
```
sns.heatmap(df1.corr())
```

Out[465]: <AxesSubplot:>



In [466]:
```
x=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'NO',  'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [467]:
```
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 [468]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[468]:
```
LinearRegression()
```

In [469]:
```
lr.intercept_
```

Out[469]:
```
28079017.487666864
```

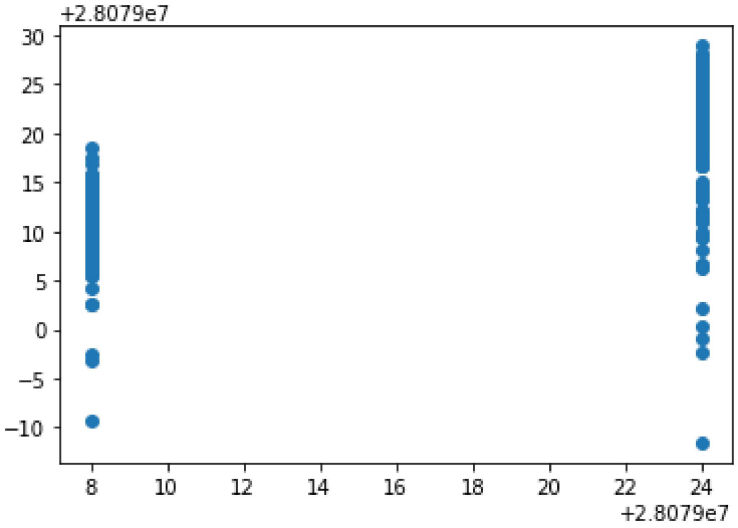
In [470]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[470]:

	Co-efficient
BEN	6.492013
CO	-17.512409
EBE	0.599825
NMHC	67.713439
NO_2	-0.104454
NO	-0.039934
O_3	-0.033816
PM10	-0.014981
SO_2	-0.287359
TCH	-3.897132
TOL	-1.535314

In [471]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[471]: <matplotlib.collections.PathCollection at 0x1d8a7c70820>



In [472]:
```
lr.score(x_test,y_test)
```

Out[472]: 0.5882526401349736

In [473]:
```
lr.score(x_train,y_train)
```

Out[473]: 0.7047913890866866

In [474]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [475]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[475]: Ridge(alpha=10)

In [476]:
```
rr.score(x_test,y_test)
```

Out[476]: 0.48257661667834184

In [477]:
```
rr.score(x_train,y_train)
```

Out[477]: 0.6393695335957623

In [478]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[478]: Lasso(alpha=10)

In [479]:
```
la.score(x_train,y_train)
```

Out[479]:
```
0.2962095362275843
```

In [480]:
```
la.score(x_test,y_test)
```

Out[480]:
```
0.2409254172561317
```

In [481]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[481]:
```
ElasticNet()
```

In [482]:
```
en.coef_
```

Out[482]:
```
array([ 0.15853487, -0.        ,  0.7535189 ,  0.        , -0.14387226,
        0.00736633, -0.06613462,  0.01643248, -0.10230426,  0.        ,
       -1.03229496])
```

In [483]:
```
en.intercept_
```

Out[483]:
```
28079027.480865043
```

In [484]:
```
prediction=en.predict(x_test)
```

In [485]:
```
en.score(x_test,y_test)
```

Out[485]:
```
0.33729842205370375
```

In [486]:
```
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)))
```

```
4.491311779568592
34.93126525418045
5.910267781935135
```

In [487]:
```
from sklearn.linear_model import LogisticRegression
```

In [488]:
```
feature_matrix=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'NO',  'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [489]:
```
feature_matrix.shape
```

Out[489]: (2000, 11)

In [490]:
```
target_vector.shape
```

Out[490]: (2000,)

In [491]:
```
from sklearn.preprocessing import StandardScaler
```

In [492]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [493]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[493]: LogisticRegression(max_iter=10000)

In [494]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11]]
```

In [495]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [496]:
```
logr.score(fs,target_vector)
```

Out[496]: 0.94

In [497]:
```
logr.predict_proba(observation)[0][0]
```

Out[497]: 1.0

In [498]:
```
logr.predict_proba(observation)
```

Out[498]: array([[1.00000000e+00, 2.67597975e-17]])

In [499]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [500]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[500]: RandomForestClassifier()

In [501]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]}
```

In [502]:
```
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[502]:
```
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 [503]:
```
grid_search.best_score_
```

Out[503]:
```
0.9821428571428571
```

In [504]:
```
rfc_best=grid_search.best_estimator_
```

In [505]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(50,5))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b',
```

Out[505]:
```
[Text(1031.086956521739, 249.15, 'NMHC <= 0.215\ngini = 0.433\nsamples = 885
\nvalue = [444, 956]\nclass = b'),
 Text(242.6086956521739, 203.85000000000002, 'TOL <= 1.25\ngini = 0.427\nsamp
les = 265\nvalue = [290, 130]\nclass = a'),
 Text(121.30434782608695, 158.55, 'gini = 0.0\nsamples = 76\nvalue = [0, 107]
\nclass = b'),
 Text(363.9130434782609, 158.55, 'O_3 <= 16.5\ngini = 0.136\nsamples = 189\nv
alue = [290, 23]\nclass = a'),
 Text(242.6086956521739, 113.25, 'gini = 0.198\nsamples = 6\nvalue = [1, 8]\n
class = b'),
 Text(485.2173913043478, 113.25, 'TCH <= 1.205\ngini = 0.094\nsamples = 183\n
value = [289, 15]\nclass = a'),
 Text(363.9130434782609, 67.94999999999999, 'gini = 0.18\nsamples = 7\nvalue
= [1, 9]\nclass = b'),
 Text(606.5217391304348, 67.94999999999999, 'O_3 <= 100.5\ngini = 0.04\nsampl
es = 176\nvalue = [288, 6]\nclass = a'),
 Text(485.2173913043478, 22.649999999999977, 'gini = 0.028\nsamples = 169\nva
lue = [281, 4]\nclass = a'),
 Text(727.8260869565217, 22.649999999999977, 'gini = 0.346\nsamples = 7\nvalu
e = [7, 2]\nclass = a'),
 Text(1819.5652173913043, 203.85000000000002, 'TOL <= 3.65\ngini = 0.265\nsam
ples = 620\nvalue = [154, 826]\nclass = b'),
 Text(1455.6521739130435, 158.55, 'TCH <= 1.75\ngini = 0.027\nsamples = 509\n
value = [11, 785]\nclass = b'),
 Text(1334.3478260869565, 113.25, 'NO_2 <= 54.5\ngini = 0.018\nsamples = 502
\nvalue = [7, 779]\nclass = b'),
 Text(1091.7391304347825, 67.94999999999999, 'TCH <= 1.295\ngini = 0.003\nsam
ples = 476\nvalue = [1, 741]\nclass = b'),
 Text(970.4347826086956, 22.649999999999977, 'gini = 0.024\nsamples = 53\nval
ue = [1, 80]\nclass = b'),
 Text(1213.0434782608695, 22.649999999999977, 'gini = 0.0\nsamples = 423\nval
ue = [0, 661]\nclass = b'),
 Text(1576.9565217391305, 67.94999999999999, 'TCH <= 1.375\ngini = 0.236\nsam
ples = 26\nvalue = [6, 38]\nclass = b'),
 Text(1455.6521739130435, 22.649999999999977, 'gini = 0.49\nsamples = 5\nvalu
e = [4, 3]\nclass = a'),
 Text(1698.2608695652173, 22.649999999999977, 'gini = 0.102\nsamples = 21\nva
lue = [2, 35]\nclass = b'),
 Text(1576.9565217391305, 113.25, 'gini = 0.48\nsamples = 7\nvalue = [4, 6]\n
class = b'),
 Text(2183.478260869565, 158.55, 'TCH <= 1.385\ngini = 0.346\nsamples = 111\n
value = [143, 41]\nclass = a'),
 Text(2062.173913043478, 113.25, 'gini = 0.0\nsamples = 47\nvalue = [74, 0]\n
class = a'),
 Text(2304.782608695652, 113.25, 'TOL <= 6.65\ngini = 0.468\nsamples = 64\nva
lue = [69, 41]\nclass = a'),
 Text(2062.173913043478, 67.94999999999999, 'TOL <= 3.85\ngini = 0.422\nsampl
es = 27\nvalue = [13, 30]\nclass = b'),
 Text(1940.8695652173913, 22.649999999999977, 'gini = 0.469\nsamples = 5\nval
ue = [5, 3]\nclass = a'),
 Text(2183.478260869565, 22.649999999999977, 'gini = 0.353\nsamples = 22\nval
ue = [8, 27]\nclass = b'),
 Text(2547.391304347826, 67.94999999999999, 'PM10 <= 72.5\ngini = 0.274\nsamp
les = 37\nvalue = [56, 11]\nclass = a'),
 Text(2426.086956521739, 22.649999999999977, 'gini = 0.437\nsamples = 17\nval
ue = [21, 10]\nclass = a'),
```

```
 Text(2668.695652173913, 22.649999999999977, 'gini = 0.054\nsamples = 20\nval
ue = [35, 1]\nclass = a')]
```

Conclusion

```
Linear Regression =0.7047913890866866

Ridge Regression =0.6393695335957623

Lasso Regression =0.2962095362275843

ElasticNet Regression =0.33729842205370375

Logistic Regression =0.94

Randomforest =0.9821428571428571
```

```
Logistic Regression is suitable for this dataset
```

In [ ]: