mk 3/08/2023

In [316]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [317]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_201
df
```

Out[317]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	2
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	2
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	2
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	2
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	2
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	2
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	2
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	2
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	2

209928 rows × 14 columns

In [318]:
```
df=df.dropna()
```

In [319]:
```
df=df.head(800)
```

In [320]:
```
df.columns
```

Out[320]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [321]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 1 to 9721
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     800 non-null    object 
 1   BEN      800 non-null    float64
 2   CO       800 non-null    float64
 3   EBE      800 non-null    float64
 4   NMHC     800 non-null    float64
 5   NO       800 non-null    float64
 6   NO_2     800 non-null    float64
 7   O_3      800 non-null    float64
 8   PM10     800 non-null    float64
 9   PM25     800 non-null    float64
 10  SO_2     800 non-null    float64
 11  TCH      800 non-null    float64
 12  TOL      800 non-null    float64
 13  station  800 non-null    int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 93.8+ KB
```

In [322]:
```
data=df[['BEN', 'TOL']]
data
```

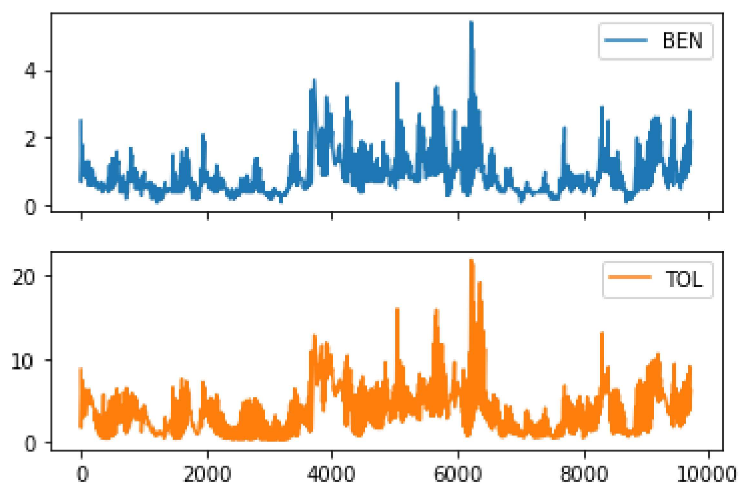
Out[322]:

	BEN	TOL
1	2.5	8.7
6	0.7	1.7
25	1.8	7.4
30	1.0	2.9
49	1.3	6.2
...	...	...
9673	2.4	8.1
9678	1.0	3.4
9697	2.8	9.1
9702	1.2	3.8
9721	1.9	6.2

800 rows × 2 columns

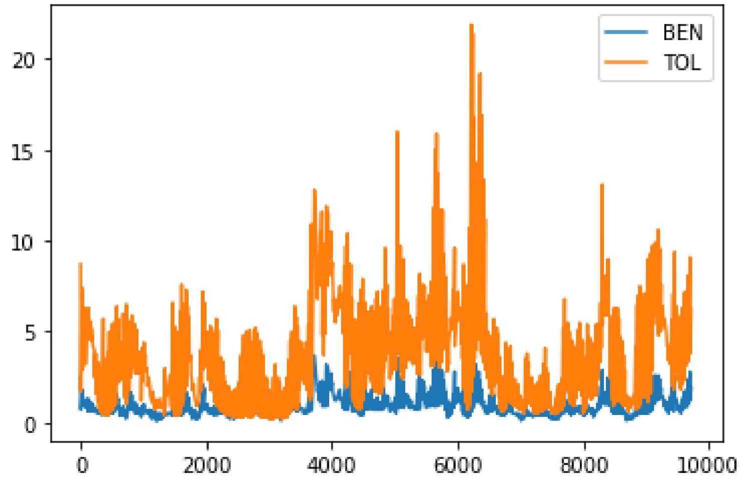
In [323]:
```
data.plot.line(subplots=True)
```

Out[323]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [324]:
```
data.plot.line()
```

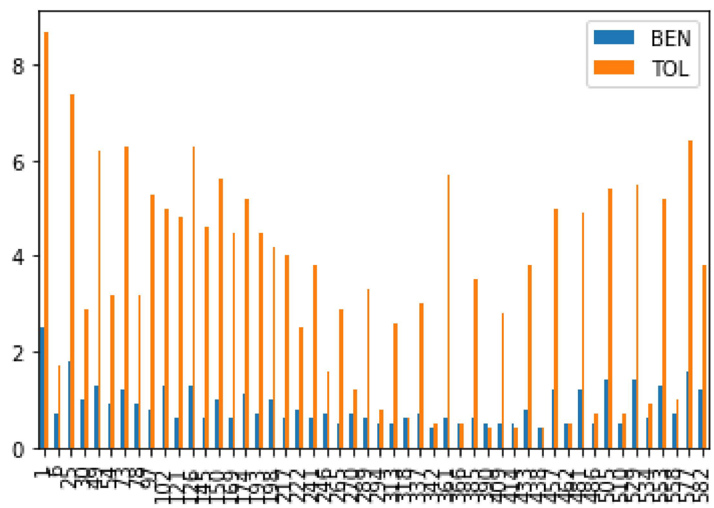
Out[324]:
```
<AxesSubplot:>
```



In [325]:
```
b=data[0:50]
```

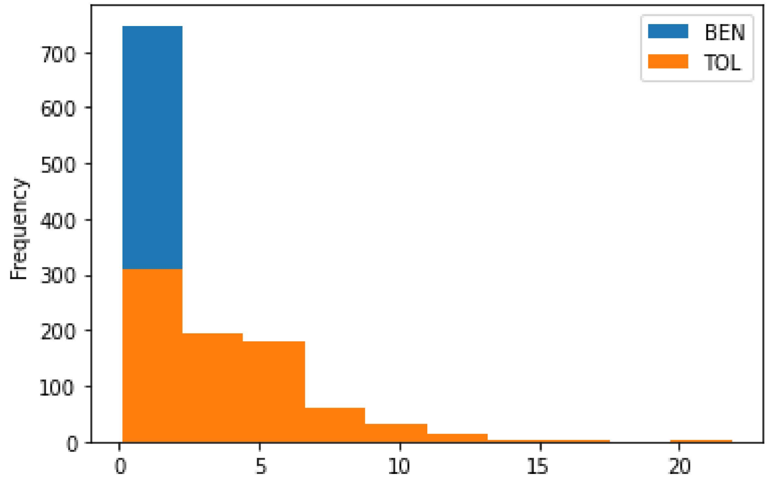
In [326]:
```
b.plot.bar()
```

Out[326]:
```
<AxesSubplot:>
```



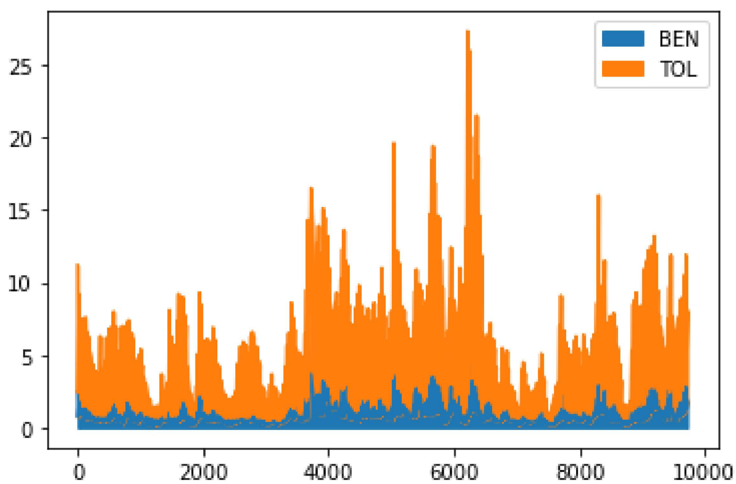
In [327]:
```
data.plot.hist()
```

Out[327]: <AxesSubplot:ylabel='Frequency'>



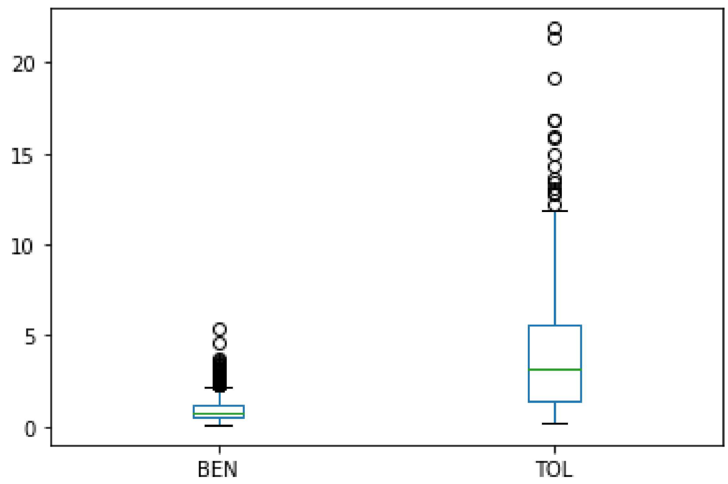
In [328]:
```
data.plot.area()
```

Out[328]: <AxesSubplot:>



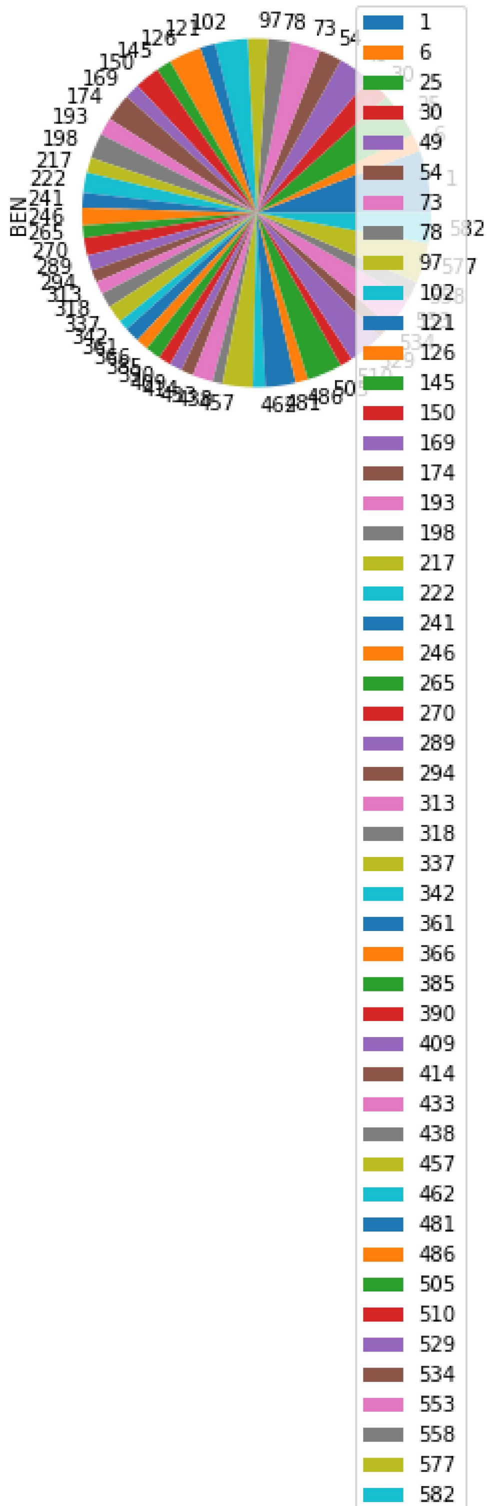
In [329]:
```
data.plot.box()
```

Out[329]: <AxesSubplot:>



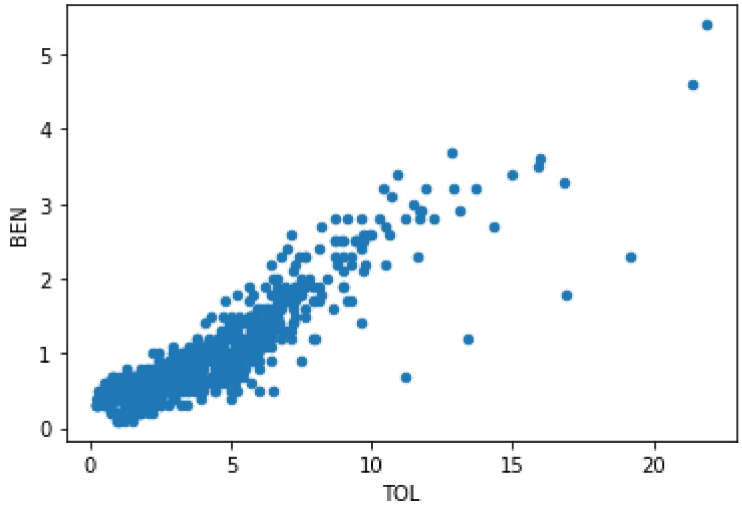
In [330]:
```
b.plot.pie(y='BEN' )
```

Out[330]:
```
<AxesSubplot:ylabel='BEN'>
```



In [331]:
```
data.plot.scatter(x='TOL' ,y='BEN')
```

Out[331]:
```
<AxesSubplot:xlabel='TOL', ylabel='BEN'>
```



In [332]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 1 to 9721
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     800 non-null    object 
 1   BEN      800 non-null    float64
 2   CO       800 non-null    float64
 3   EBE      800 non-null    float64
 4   NMHC     800 non-null    float64
 5   NO       800 non-null    float64
 6   NO_2     800 non-null    float64
 7   O_3      800 non-null    float64
 8   PM10     800 non-null    float64
 9   PM25     800 non-null    float64
 10  SO_2     800 non-null    float64
 11  TCH      800 non-null    float64
 12  TOL      800 non-null    float64
 13  station  800 non-null    int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 93.8+ KB
```

In [333]:
```
df.describe()
```

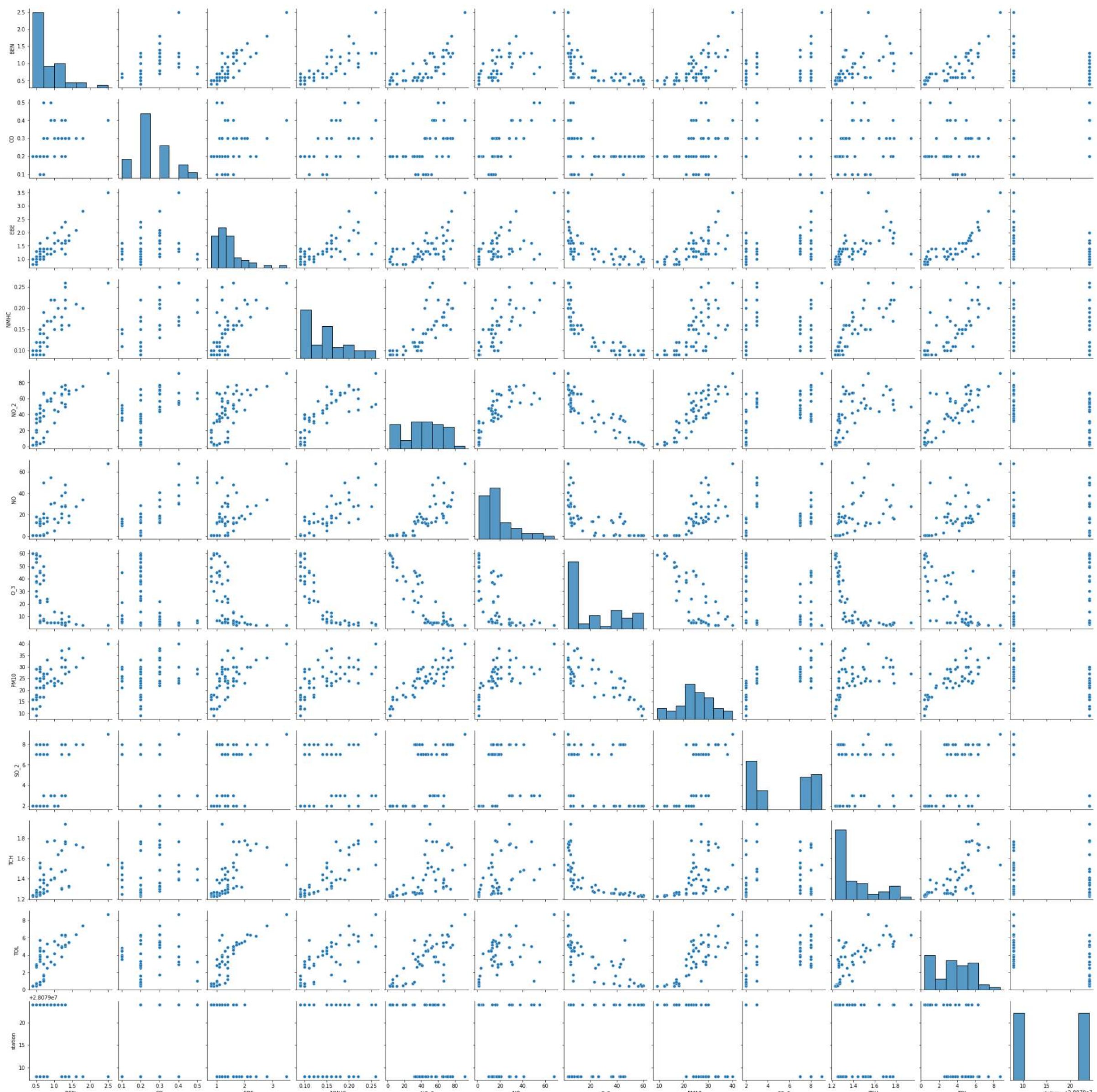
Out[333]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	
count	800.00000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.0(
mean	0.94225	0.296750	1.475750	0.161412	25.791250	42.945000	22.772500	18.7(
std	0.68275	0.162704	1.041041	0.053453	35.160995	25.994213	17.267917	10.4:
min	0.10000	0.100000	0.300000	0.080000	1.000000	1.000000	2.000000	2.0(
25%	0.50000	0.200000	0.800000	0.120000	2.000000	22.000000	6.000000	11.0(
50%	0.70000	0.300000	1.100000	0.150000	14.000000	43.500000	19.000000	18.0(
75%	1.20000	0.400000	1.700000	0.190000	36.000000	61.000000	37.000000	25.0(
max	5.40000	1.900000	8.200000	0.490000	365.000000	175.000000	64.000000	89.0(

In [334]:
```
df1=df[['BEN', 'CO', 'EBE','NMHC', 'NO_2', 'NO',  'O_3',
 'PM10','SO_2', 'TCH', 'TOL', 'station']]
```

In [335]:
```
sns.pairplot(df1[0:50])
```

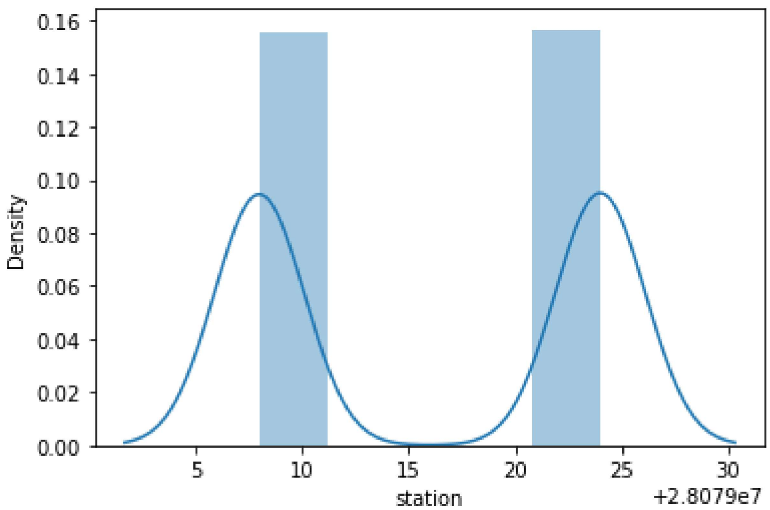
Out[335]: <seaborn.axisgrid.PairGrid at 0x1d88a3f7430>



In [336]:
```
sns.distplot(df1['station'])
```

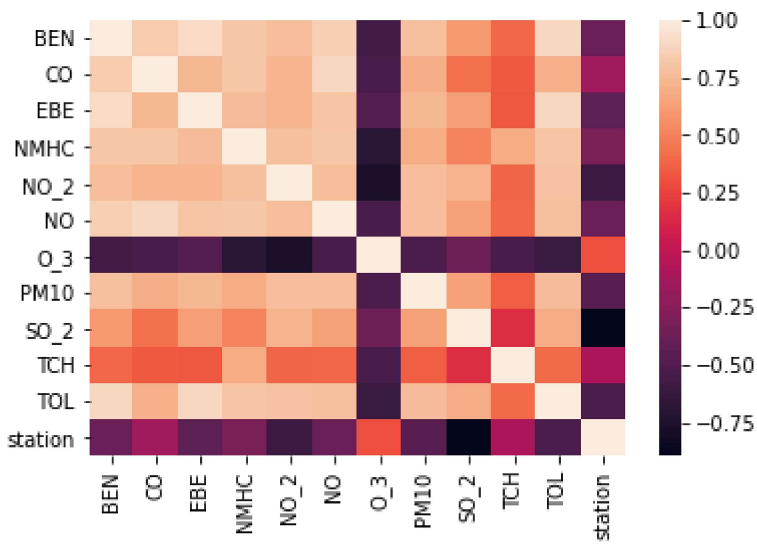
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[336]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [337]:
```
sns.heatmap(df1.corr())
```

Out[337]: <AxesSubplot:>



In [338]:
```
x=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'NO',  'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [339]:
```
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 [340]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[340]:
```
LinearRegression()
```

In [341]:
```
lr.intercept_
```

Out[341]:
```
28079021.82965764
```

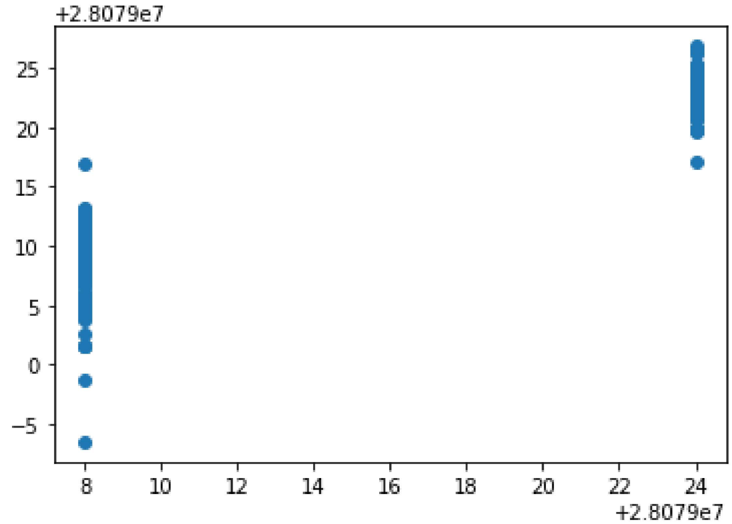
In [342]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[342]:

	Co-efficient
BEN	-0.034243
CO	22.339344
EBE	-0.741200
NMHC	8.754248
NO_2	-0.075296
NO	-0.012261
O_3	0.032894
PM10	0.051771
SO_2	-2.097788
TCH	0.133014
TOL	0.149400

In [343]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[343]: <matplotlib.collections.PathCollection at 0x1d893640550>



In [344]:
```
lr.score(x_test,y_test)
```

Out[344]: 0.8895515441584051

In [345]:
```
lr.score(x_train,y_train)
```

Out[345]: 0.9063734845521797

In [346]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [347]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[347]: Ridge(alpha=10)

In [348]:
```
rr.score(x_test,y_test)
```

Out[348]: 0.8687242338146928

In [349]:
```
rr.score(x_train,y_train)
```

Out[349]: 0.8888695976199741

In [350]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[350]: Lasso(alpha=10)

In [351]:
```
la.score(x_train,y_train)
```
Out[351]: 0.6073128952517146

In [352]:
```
la.score(x_test,y_test)
```
Out[352]: 0.5343332803911878

In [353]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[353]: ElasticNet()

In [354]:
```
en.coef_
```
Out[354]:
```
array([ 0.        ,  0.        ,  0.        ,  0.        , -0.06491198,
        0.06997964, -0.        ,  0.03644468, -2.11361822,  0.        ,
        0.        ])
```

In [355]:
```
en.intercept_
```
Out[355]: 28079028.09886945

In [356]:
```
prediction=en.predict(x_test)
```

In [357]:
```
en.score(x_test,y_test)
```
Out[357]: 0.8407158041429669

In [358]:
```
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)))
```
```
2.5001350118933865
10.136846224341589
3.183841425753109
```

In [359]:
```
from sklearn.linear_model import LogisticRegression
```

In [360]:
```
feature_matrix=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'NO',  'O_3',
 'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [361]:
```
feature_matrix.shape
```
Out[361]: (800, 11)

In [362]:
```
target_vector.shape
```
Out[362]: (800,)

In [363]:
```
from sklearn.preprocessing import StandardScaler
```

In [364]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [365]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```
Out[365]: LogisticRegression(max_iter=10000)

In [370]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11]]
```

In [371]:
```
prediction=logr.predict(observation)
print(prediction)
```
```
[28079008]
```

In [372]:
```
logr.score(fs,target_vector)
```
Out[372]: 1.0

In [373]:
```
logr.predict_proba(observation)[0][0]
```
Out[373]: 1.0

In [374]:
```
logr.predict_proba(observation)
```
Out[374]: array([[1.00000000e+00, 4.27828389e-22]])

In [375]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [376]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```
Out[376]: RandomForestClassifier()

In [377]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]}
```

In [378]:
```
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[378]:
```
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')
```

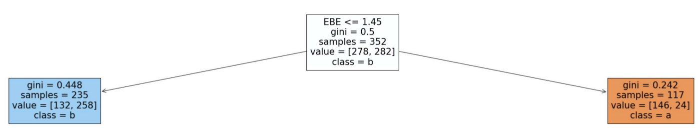
In [379]:
```
grid_search.best_score_
```

Out[379]:
```
0.9964285714285714
```

In [380]:
```
rfc_best=grid_search.best_estimator_
```

In [381]:
```
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[381]:
```
[Text(1395.0, 203.85000000000002, 'EBE <= 1.45\ngini = 0.5\nsamples = 352\nva
lue = [278, 282]\nclass = b'),
 Text(697.5, 67.94999999999999, 'gini = 0.448\nsamples = 235\nvalue = [132, 2
58]\nclass = b'),
 Text(2092.5, 67.94999999999999, 'gini = 0.242\nsamples = 117\nvalue = [146,
24]\nclass = a')]
```



Conclusion

Linear Regression =0.9063734845521797

Ridge Regression =0.8888695976199741

Lasso Regression =0.6073128952517146

ElasticNet Regression =0.8407158041429669

Logistic Regression =1.0

Randomforest =0.9964285714285714

Logistic Regression is suitable for this dataset

In [ ]: