mk 3/08/2023

In [147]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [148]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_201
df
```

Out[148]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	I
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	I
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	I
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.410
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.670
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	I
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	I
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	I
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.150

209448 rows × 17 columns

In [149]:
```
df=df.dropna()
```

In [150]:
```
df=df.head(100)
```

In [151]:
```
df.columns
```

Out[151]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [152]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 11 to 1199
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     100 non-null    object 
 1   BEN      100 non-null    float64
 2   CO       100 non-null    float64
 3   EBE      100 non-null    float64
 4   MXY      100 non-null    float64
 5   NMHC     100 non-null    float64
 6   NO_2     100 non-null    float64
 7   NOx      100 non-null    float64
 8   OXY      100 non-null    float64
 9   O_3      100 non-null    float64
 10  PM10     100 non-null    float64
 11  PM25     100 non-null    float64
 12  PXY      100 non-null    float64
 13  SO_2     100 non-null    float64
 14  TCH      100 non-null    float64
 15  TOL      100 non-null    float64
 16  station  100 non-null    int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 14.1+ KB
```

In [153]:
```
data=df[['BEN', 'TOL', 'PXY']]
data
```

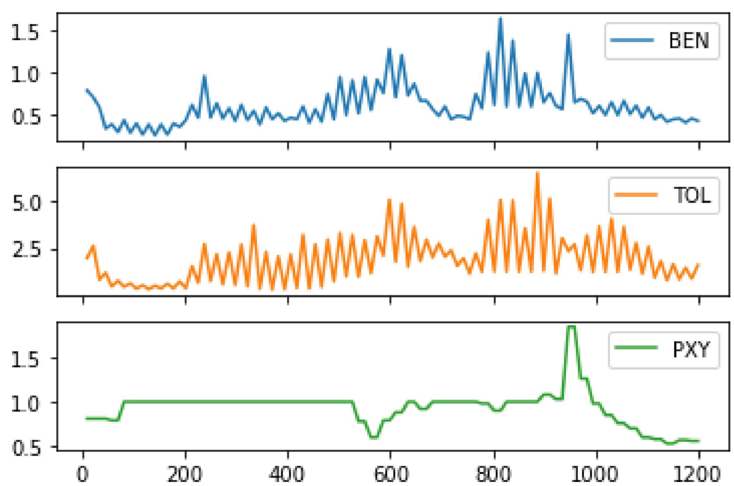
Out[153]:

	BEN	TOL	PXY
11	0.78	1.99	0.81
23	0.70	2.62	0.81
35	0.58	0.84	0.81
47	0.33	1.21	0.81
59	0.38	0.49	0.79
...	...	...	...
1151	0.44	1.66	0.53
1163	0.45	0.86	0.57
1175	0.40	1.48	0.57
1187	0.45	0.91	0.56
1199	0.42	1.60	0.56

100 rows × 3 columns

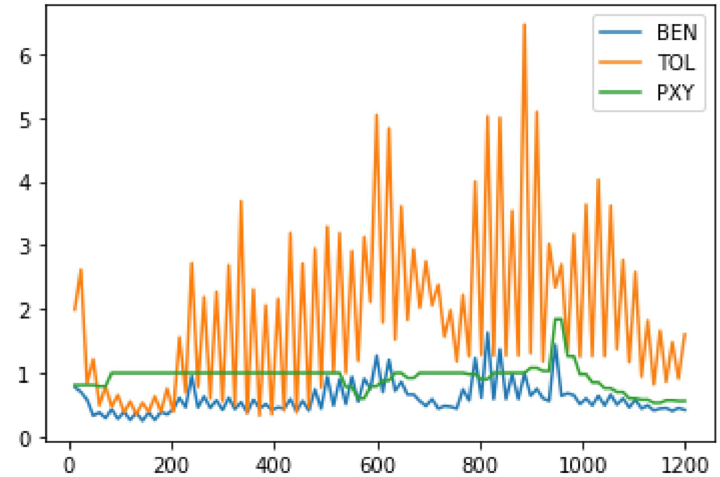
In [154]:
```
data.plot.line(subplots=True)
```

Out[154]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [155]:
```
data.plot.line()
```

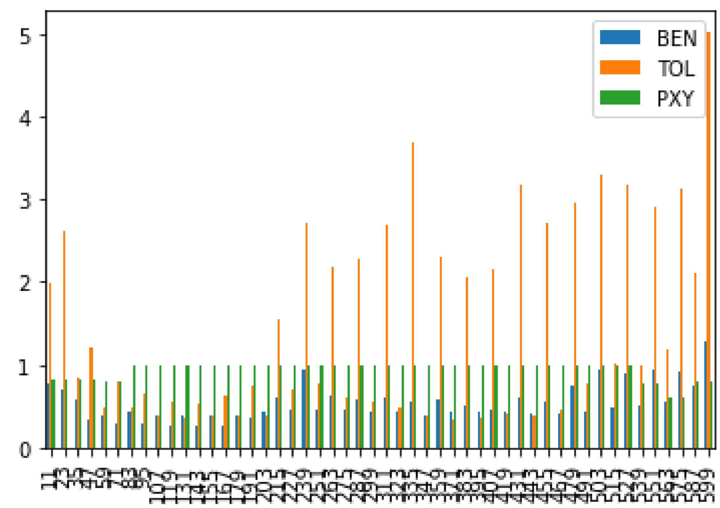
Out[155]: <AxesSubplot:>



In [156]:
```
b=data[0:50]
```

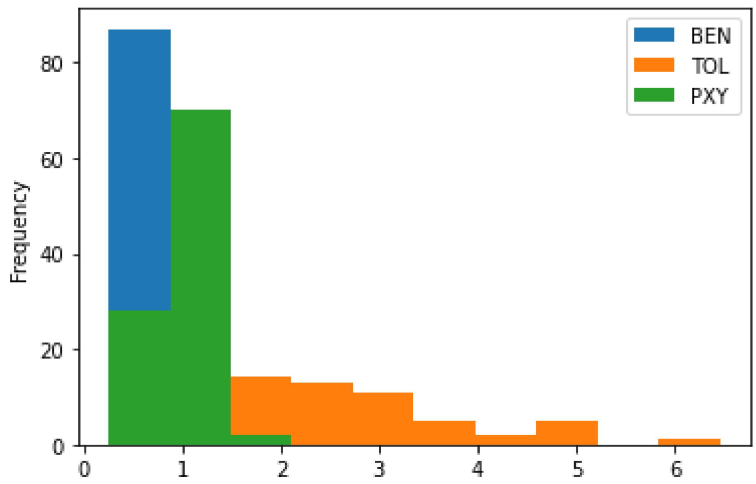
In [157]:
```
b.plot.bar()
```

Out[157]: <AxesSubplot:>



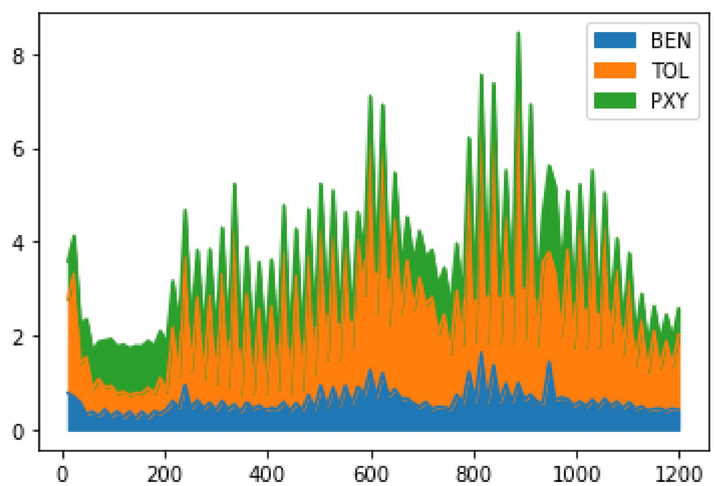
In [158]:
```
data.plot.hist()
```

Out[158]: <AxesSubplot:ylabel='Frequency'>



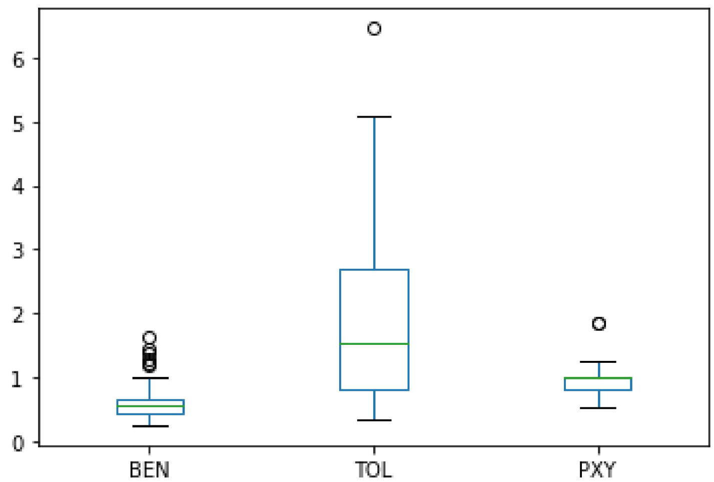
In [159]:
```
data.plot.area()
```

Out[159]: <AxesSubplot:>



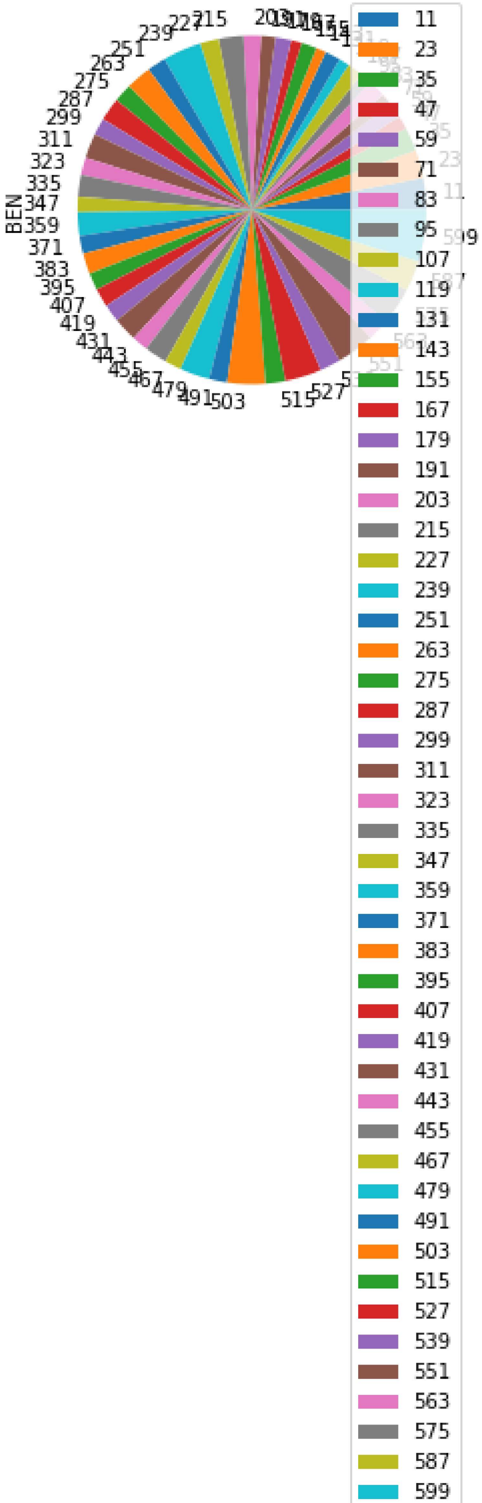
In [160]:
```
data.plot.box()
```

Out[160]: <AxesSubplot:>



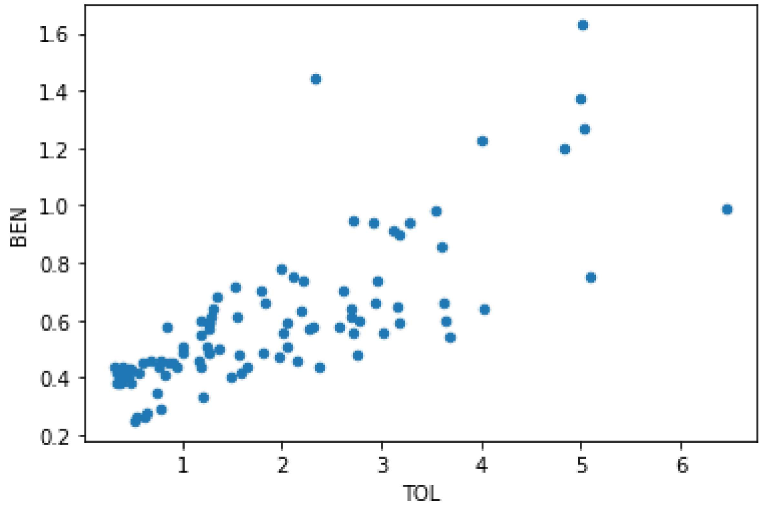
In [161]:
```
b.plot.pie(y='BEN' )
```

Out[161]: <AxesSubplot:ylabel='BEN'>



In [162]:
```
data.plot.scatter(x='TOL' ,y='BEN')
```

Out[162]: <AxesSubplot:xlabel='TOL', ylabel='BEN'>



In [163]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 11 to 1199
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     100 non-null    object
 1   BEN      100 non-null    float64
 2   CO       100 non-null    float64
 3   EBE      100 non-null    float64
 4   MXY      100 non-null    float64
 5   NMHC     100 non-null    float64
 6   NO_2     100 non-null    float64
 7   NOx      100 non-null    float64
 8   OXY      100 non-null    float64
 9   O_3      100 non-null    float64
 10  PM10     100 non-null    float64
 11  PM25     100 non-null    float64
 12  PXY      100 non-null    float64
 13  SO_2     100 non-null    float64
 14  TCH      100 non-null    float64
 15  TOL      100 non-null    float64
 16  station  100 non-null    int64
dtypes: float64(15), int64(1), object(1)
memory usage: 14.1+ KB
```

In [164]:
```
df.describe()
```

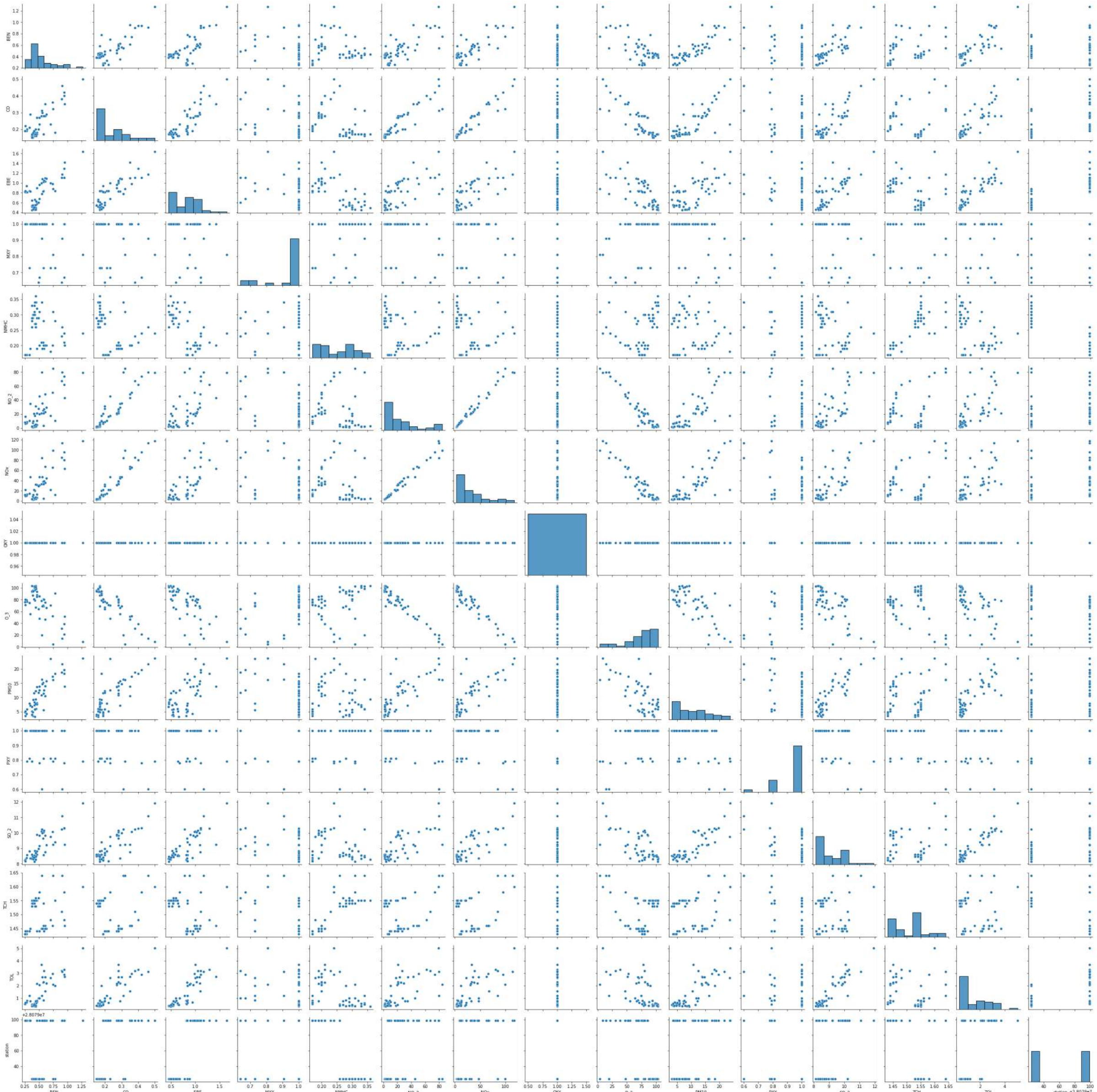
Out[164]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.(
mean	0.597600	0.258900	0.920100	0.842200	0.275100	29.911300	38.454500	1.(
std	0.254626	0.083217	0.352931	0.284575	0.079181	20.908565	27.822058	0.:
min	0.250000	0.150000	0.340000	0.380000	0.170000	1.290000	2.780000	0.!
25%	0.440000	0.190000	0.655000	0.610000	0.210000	16.910000	20.057500	1.(
50%	0.555000	0.240000	0.880000	0.935000	0.260000	27.100000	33.234999	1.(
75%	0.660000	0.310000	1.102500	1.000000	0.330000	40.415001	50.852500	1.(
max	1.630000	0.500000	2.180000	1.780000	0.470000	84.629997	117.300003	1.8

In [165]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [166]:
```
sns.pairplot(df1[0:50])
```

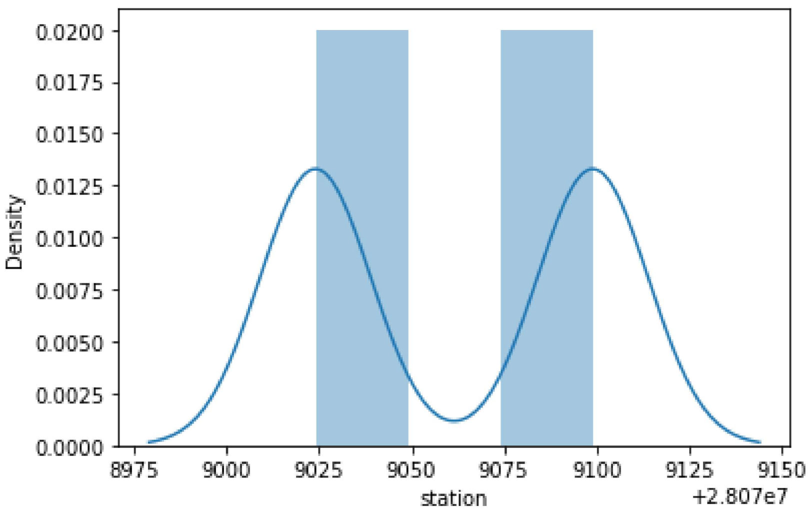
Out[166]: <seaborn.axisgrid.PairGrid at 0x1d8f8c34760>



In [167]:
```
sns.distplot(df1['station'])
```

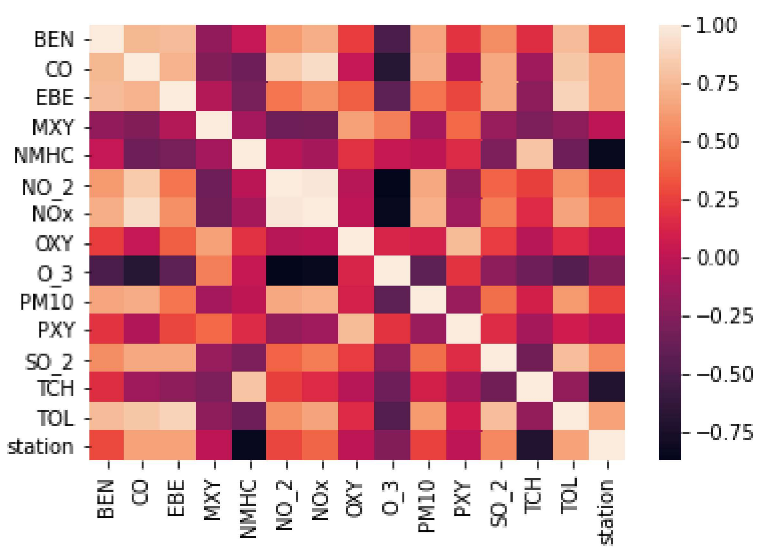
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[167]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [168]:
```
sns.heatmap(df1.corr())
```

Out[168]: <AxesSubplot:>



In [169]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [170]:
```
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 [171]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[171]:
```
LinearRegression()
```

In [172]:
```
lr.intercept_
```

Out[172]:
```
28079197.48878474
```

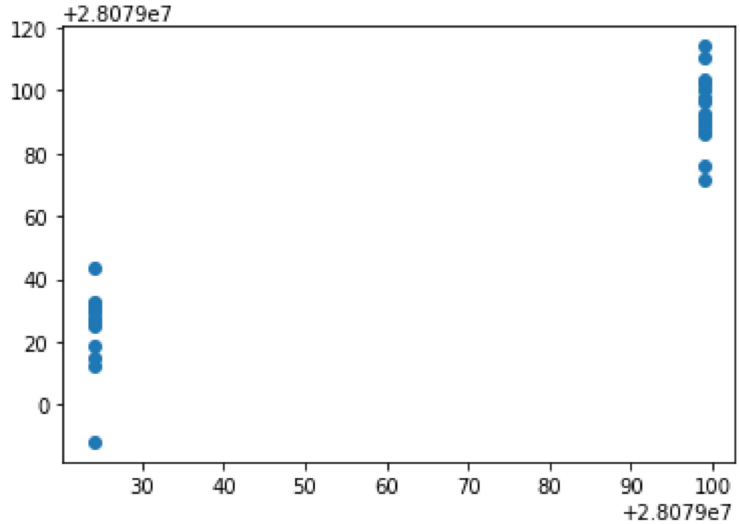
In [173]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[173]:

	Co-efficient
BEN	-65.477903
CO	562.027959
EBE	27.200743
MXY	-10.248607
NMHC	-111.487476
NO_2	-0.741432
NOx	-0.447744
OXY	16.357217
O_3	-0.235314
PM10	0.116936
PXY	-3.133652
SO_2	-3.087070
TCH	-106.991085
TOL	2.717068

In [174]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[174]: <matplotlib.collections.PathCollection at 0x1d886e84640>



In [175]:
```
lr.score(x_test,y_test)
```

Out[175]: 0.8896517498224122

In [176]:
```
lr.score(x_train,y_train)
```

Out[176]: 0.949306857592614

In [177]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [178]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[178]: Ridge(alpha=10)

In [179]:
```
rr.score(x_test,y_test)
```

Out[179]: 0.4697103996800025

In [180]:
```
rr.score(x_train,y_train)
```

Out[180]: 0.6520503785699647

In [181]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[181]: Lasso(alpha=10)

In [182]:
```
la.score(x_train,y_train)
```
Out[182]: 0.4692794984392259

In [183]:
```
la.score(x_test,y_test)
```
Out[183]: 0.36782468523872713

In [184]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[184]: ElasticNet()

In [185]:
```
en.coef_
```
Out[185]:
```
array([-1.92676588,  0.        ,  1.79373525,  3.68565428, -2.03130748,
       -3.14780078,  2.35863109, -0.        , -0.35659703, -1.21336016,
       -0.39958075,  2.97652582, -1.38750317,  6.95817564])
```

In [186]:
```
en.intercept_
```
Out[186]: 28079057.24657358

In [187]:
```
prediction=en.predict(x_test)
```

In [188]:
```
en.score(x_test,y_test)
```
Out[188]: 0.3895631825806143

In [189]:
```
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)))
```
```
25.601336059346796
843.1658540605265
29.037318300086294
```

In [190]:
```
from sklearn.linear_model import LogisticRegression
```

In [191]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [192]:
```
feature_matrix.shape
```
Out[192]: (100, 14)

In [193]:
```
target_vector.shape
```
Out[193]: (100,)

In [194]:
```
from sklearn.preprocessing import StandardScaler
```

In [195]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [196]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```
Out[196]: LogisticRegression(max_iter=10000)

In [197]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [198]:
```
prediction=logr.predict(observation)
print(prediction)
```
```
[28079024]
```

In [199]:
```
logr.score(fs,target_vector)
```
Out[199]: 1.0

In [200]:
```
logr.predict_proba(observation)[0][0]
```
Out[200]: 0.9993168651234103

In [201]:
```
logr.predict_proba(observation)
```
Out[201]: array([[9.99316865e-01, 6.83134877e-04]])

In [202]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [203]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```
Out[203]: RandomForestClassifier()

In [204]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]}
```

In [205]:
```
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[205]:
```
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')
```

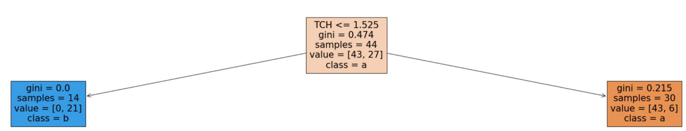
In [206]:
```
grid_search.best_score_
```

Out[206]:
```
1.0
```

In [207]:
```
rfc_best=grid_search.best_estimator_
```

In [208]:
```
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[208]:
```
[Text(1395.0, 203.85000000000002, 'TCH <= 1.525\ngini = 0.474\nsamples = 44\n
value = [43, 27]\nclass = a'),
 Text(697.5, 67.94999999999999, 'gini = 0.0\nsamples = 14\nvalue = [0, 21]\nc
lass = b'),
 Text(2092.5, 67.94999999999999, 'gini = 0.215\nsamples = 30\nvalue = [43, 6]
\nclass = a')]
```



Conclusion

Linear Regression =0.949306857592614

Ridge Regression =0.6520503785699647

Lasso Regression =0.4692794984392259

ElasticNet Regression =0.3895631825806143

Logistic Regression =0.9993168651234103

Randomforest =1.0

Randomforest is suitable for this dataset

In [ ]: