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_200
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.889
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.040
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.270
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.850
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.450
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.020
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.540
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.910
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.690

226392 rows × 17 columns

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

In [4]:
```
df=df.head(1100)
```

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

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1100 entries, 4 to 9600
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1100 non-null   object 
 1   BEN      1100 non-null   float64
 2   CO       1100 non-null   float64
 3   EBE      1100 non-null   float64
 4   MXY      1100 non-null   float64
 5   NMHC     1100 non-null   float64
 6   NO_2     1100 non-null   float64
 7   NOx      1100 non-null   float64
 8   OXY      1100 non-null   float64
 9   O_3      1100 non-null   float64
 10  PM10     1100 non-null   float64
 11  PM25     1100 non-null   float64
 12  PXY      1100 non-null   float64
 13  SO_2     1100 non-null   float64
 14  TCH      1100 non-null   float64
 15  TOL      1100 non-null   float64
 16  station  1100 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 154.7+ KB
```

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

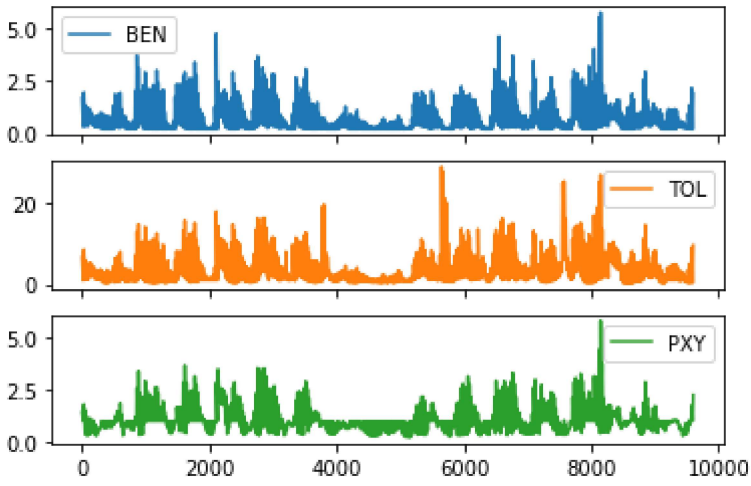
Out[7]:

	BEN	TOL	PXY
4	1.68	6.67	1.43
21	0.32	0.94	1.00
25	0.73	2.82	1.22
30	1.95	8.57	1.81
47	0.36	1.18	0.38
...	...	...	...
9569	0.75	2.86	1.04
9574	2.17	9.13	1.76
9591	0.20	0.39	1.00
9595	0.97	4.08	1.38
9600	1.90	9.59	2.26

1100 rows × 3 columns

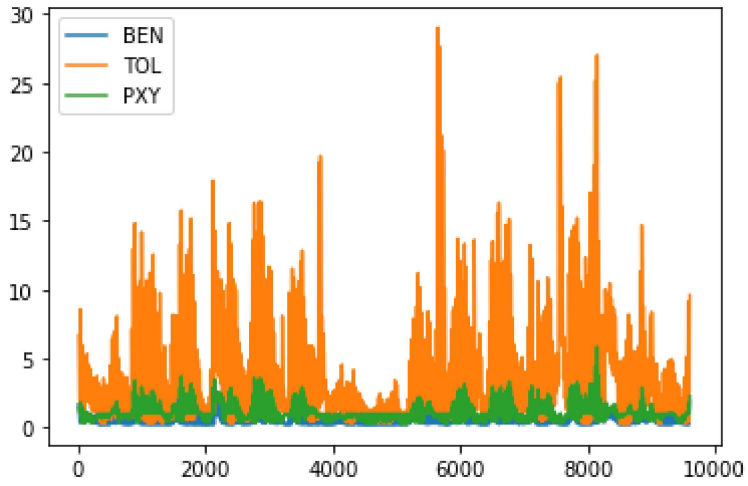
In [8]:
```
data.plot.line(subplots=True)
```

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



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

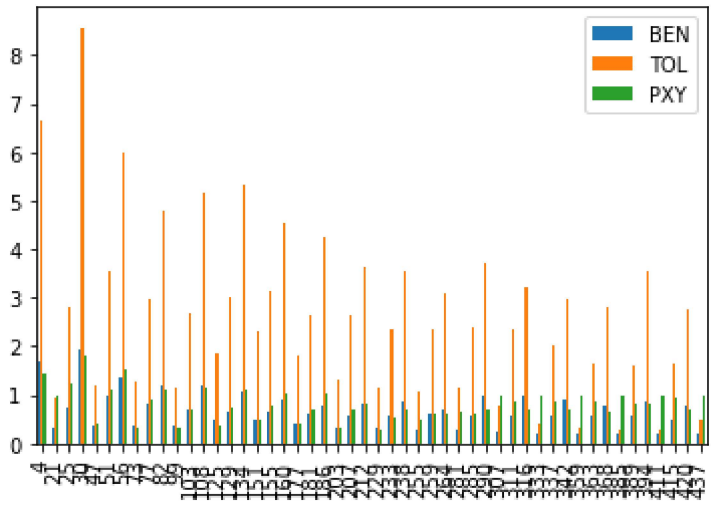
Out[9]: <AxesSubplot:>



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

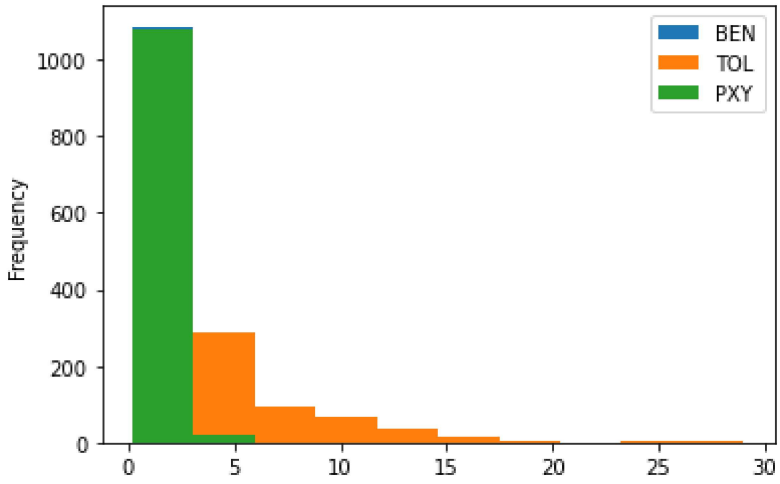
In [11]:
```
b.plot.bar()
```

Out[11]: <AxesSubplot:>



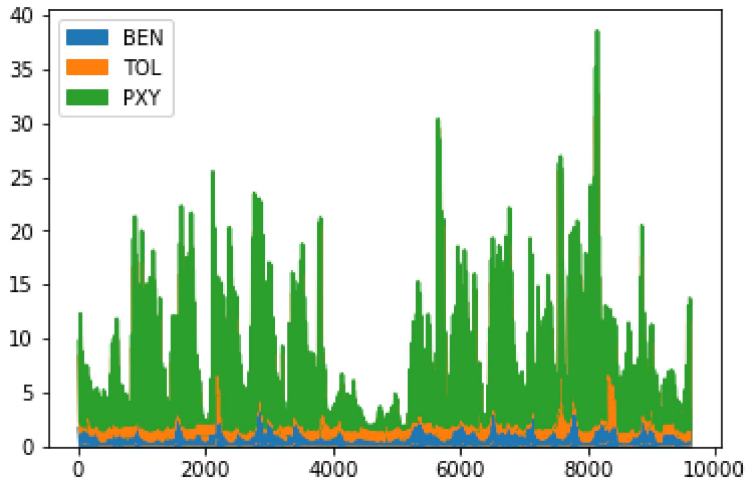
In [12]:
```
data.plot.hist()
```

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



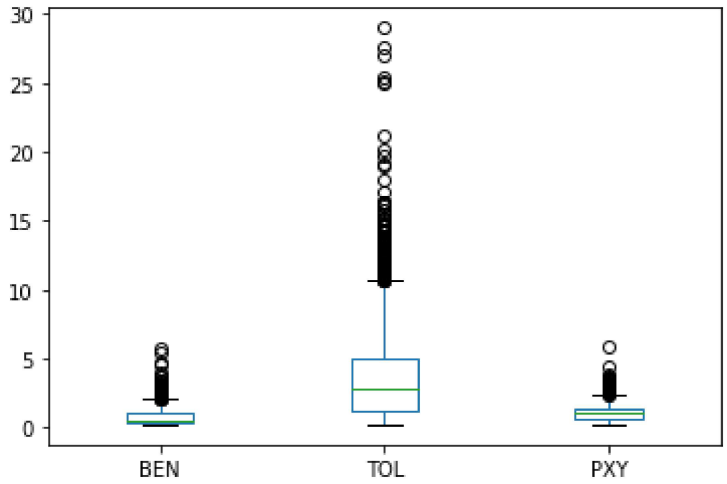
In [13]:
```
data.plot.area()
```

Out[13]: <AxesSubplot:>



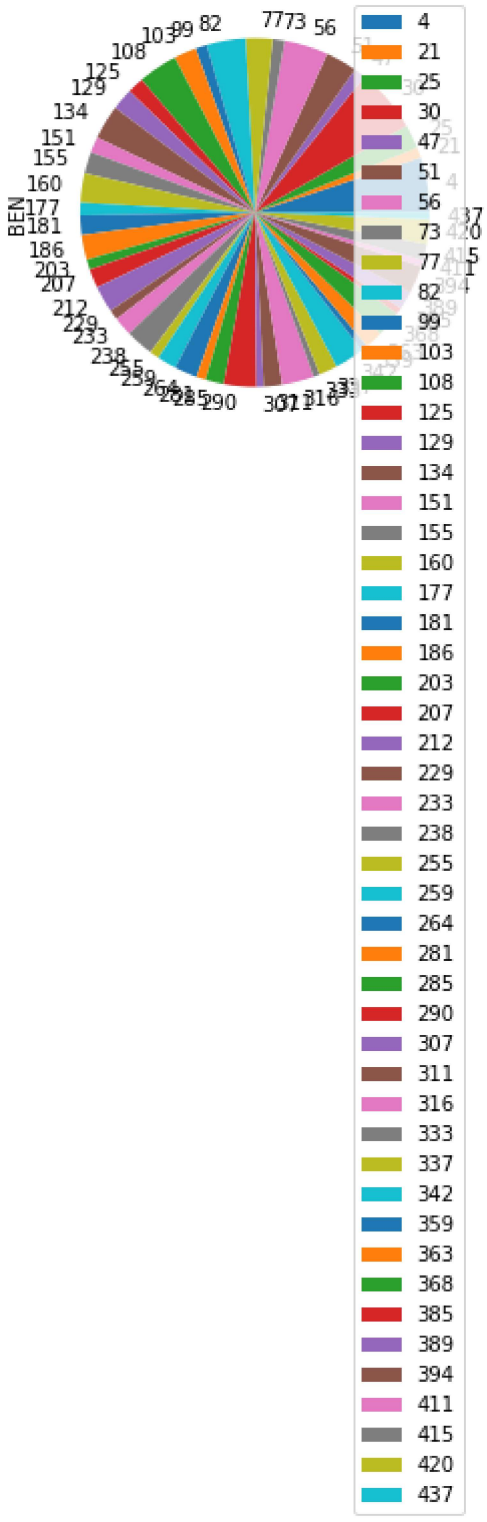
In [14]:
```
data.plot.box()
```

Out[14]: <AxesSubplot:>



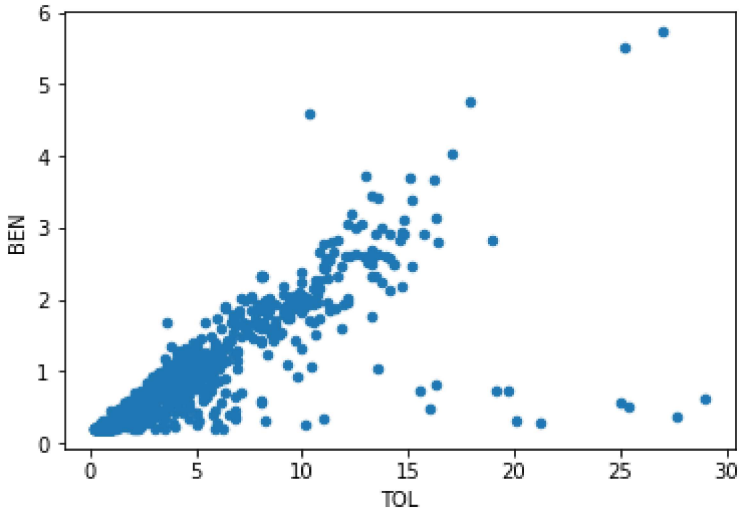
In [15]:
```
b.plot.pie(y='BEN' )
```

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



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

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



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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1100 entries, 4 to 9600
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1100 non-null   object 
 1   BEN      1100 non-null   float64
 2   CO       1100 non-null   float64
 3   EBE      1100 non-null   float64
 4   MXY      1100 non-null   float64
 5   NMHC     1100 non-null   float64
 6   NO_2     1100 non-null   float64
 7   NOx      1100 non-null   float64
 8   OXY      1100 non-null   float64
 9   O_3      1100 non-null   float64
 10  PM10     1100 non-null   float64
 11  PM25     1100 non-null   float64
 12  PXY      1100 non-null   float64
 13  SO_2     1100 non-null   float64
 14  TCH      1100 non-null   float64
 15  TOL      1100 non-null   float64
 16  station  1100 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 154.7+ KB
```

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

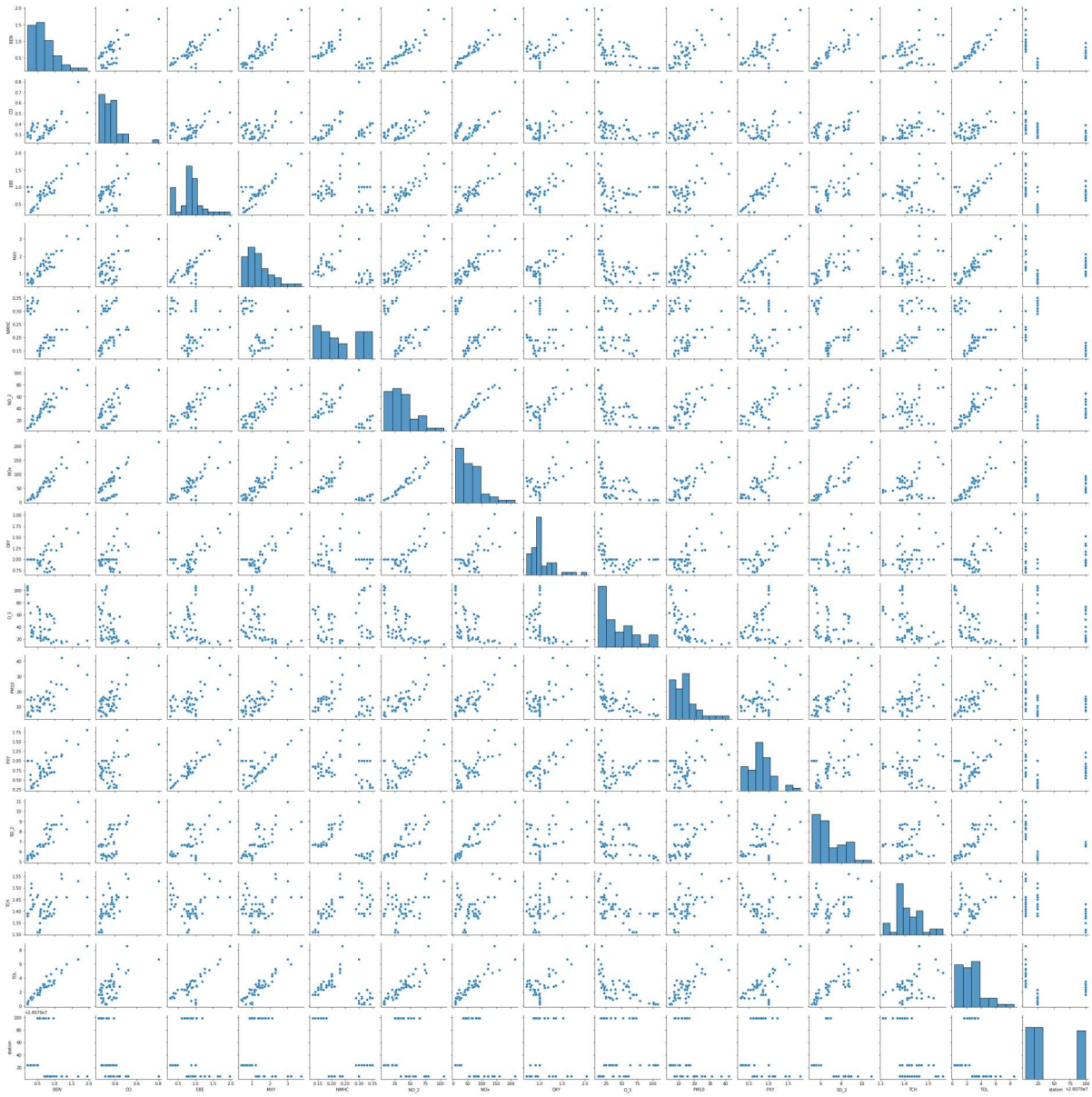
Out[18]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
count	1100.000000	1100.000000	1100.000000	1100.000000	1100.000000	1100.000000	1100.000000
mean	0.797373	0.380927	1.188373	2.004991	0.178736	41.265945	67.236136
std	0.733061	0.182232	0.819708	1.654855	0.074210	27.774774	59.691514
min	0.200000	0.090000	0.270000	0.260000	0.020000	5.270000	6.830000
25%	0.280000	0.270000	0.740000	0.870000	0.127500	17.472500	21.272501
50%	0.550000	0.340000	1.000000	1.300000	0.160000	36.494999	52.809999
75%	1.010000	0.430000	1.300000	2.790000	0.220000	60.067501	92.939999
max	5.730000	2.180000	9.770000	13.150000	0.540000	217.800003	694.900024

In [19]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

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

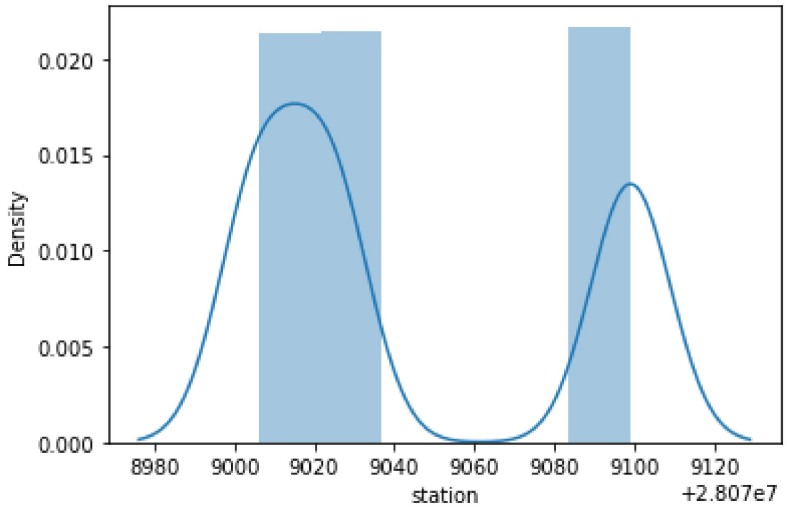
Out[20]: <seaborn.axisgrid.PairGrid at 0x1d8cc3bf460>



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

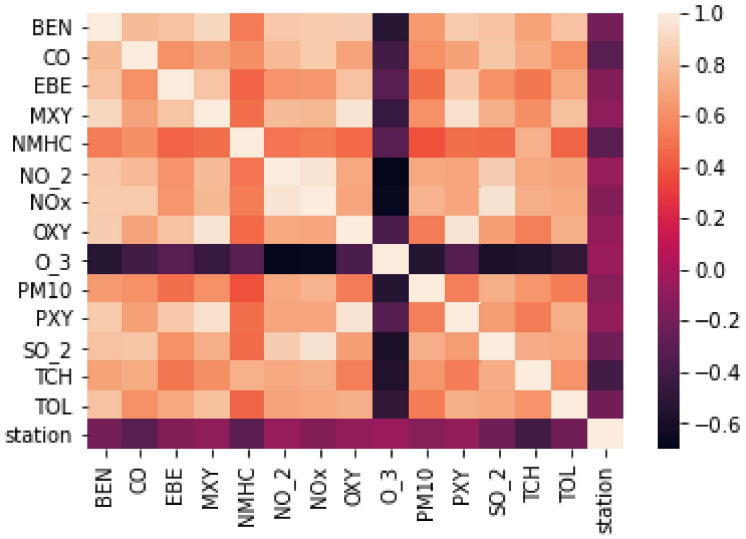
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[22]: <AxesSubplot:>



In [23]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', '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]:
```
28079454.909333665
```

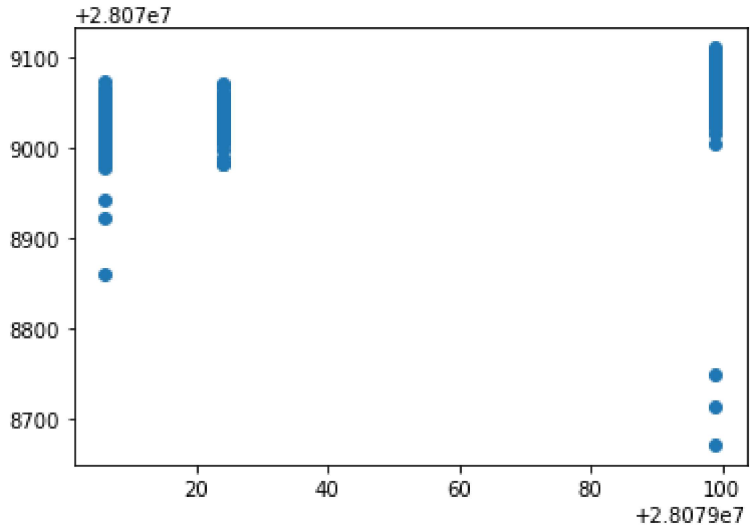
In [27]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[27]:

	Co-efficient
BEN	-22.016740
CO	-119.003147
EBE	-42.840255
MXY	-11.794412
NMHC	16.476359
NO_2	-0.141561
NOx	1.083389
OXY	44.107727
O_3	-0.176626
PM10	-0.127788
PXY	38.071233
SO_2	-16.481398
TCH	-228.832098
TOL	0.416695

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

Out[28]: <matplotlib.collections.PathCollection at 0x1d8da5987c0>



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

Out[29]: -0.3028224357287179

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

Out[30]: 0.478920116248428

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]: -0.1925874962934888

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

Out[34]: 0.39186650312521176

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]:
```
0.09710558105208578
```

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

Out[37]:
```
0.10893562584731231
```

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([-2.3617583 , -1.3276034 , -1.88581144,  1.93832014, -0.24216828,
        0.68208089, -0.18771792,  1.67021273, -0.25698309, -0.36351415,
        1.45141965, -4.17210571, -0.69180965, -2.49391497])
```

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

Out[40]:
```
28079086.156647407
```

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

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

Out[42]:
```
0.1469008555565141
```

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)))
```

```
35.33881585542677
1510.1265642920803
38.860346939934544
```

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

In [45]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

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

Out[46]: (1100, 14)

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

Out[47]: (1100,)

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

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

```
[28079006]
```

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

Out[53]: 0.95

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

Out[54]: 0.8281788232899833

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

Out[55]: array([[8.28178823e-01, 1.92073143e-07, 1.71820985e-01]])

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_
```

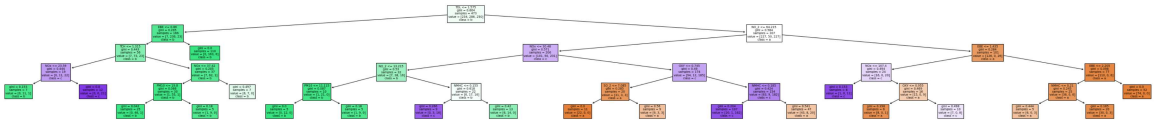
Out[60]:
```
0.9051948051948051
```

In [61]:
```
rfc_best=grid_search.best_estimator_
```

In [62]:
```
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[62]:
```
[Text(1133.4375, 249.15, 'TOL <= 1.575\ngini = 0.664\nsamples = 473\nvalue =
[234, 286, 250]\nclass = b'),
 Text(435.9375, 203.85000000000002, 'EBE <= 0.86\ngini = 0.205\nsamples = 166
\nvalue = [7, 236, 23]\nclass = b'),
 Text(348.75, 158.55, 'TCH <= 1.315\ngini = 0.443\nsamples = 56\nvalue = [7,
73, 23]\nclass = b'),
 Text(174.375, 113.25, 'NOx <= 23.59\ngini = 0.444\nsamples = 19\nvalue = [0,
11, 22]\nclass = c'),
 Text(87.1875, 67.94999999999999, 'gini = 0.153\nsamples = 7\nvalue = [0, 11,
1]\nclass = b'),
 Text(261.5625, 67.94999999999999, 'gini = 0.0\nsamples = 12\nvalue = [0, 0,
21]\nclass = c'),
 Text(523.125, 113.25, 'NOx <= 37.42\ngini = 0.205\nsamples = 37\nvalue = [7,
62, 1]\nclass = b'),
 Text(435.9375, 67.94999999999999, 'PM10 <= 19.8\ngini = 0.068\nsamples = 30
\nvalue = [1, 55, 1]\nclass = b'),
 Text(348.75, 22.649999999999977, 'gini = 0.042\nsamples = 25\nvalue = [0, 4
6, 1]\nclass = b'),
 Text(523.125, 22.649999999999977, 'gini = 0.18\nsamples = 5\nvalue = [1, 9,
0]\nclass = b'),
 Text(610.3125, 67.94999999999999, 'gini = 0.497\nsamples = 7\nvalue = [6, 7,
0]\nclass = b'),
 Text(523.125, 158.55, 'gini = 0.0\nsamples = 110\nvalue = [0, 163, 0]\nclass
= b'),
 Text(1830.9375, 203.85000000000002, 'NO_2 <= 64.215\ngini = 0.584\nsamples =
307\nvalue = [227, 50, 227]\nclass = a'),
 Text(1307.8125, 158.55, 'NOx <= 30.48\ngini = 0.571\nsamples = 206\nvalue =
[101, 50, 201]\nclass = c'),
 Text(959.0625, 113.25, 'NO_2 <= 13.215\ngini = 0.53\nsamples = 32\nvalue =
[7, 38, 16]\nclass = b'),
 Text(784.6875, 67.94999999999999, 'PM10 <= 11.115\ngini = 0.087\nsamples = 1
0\nvalue = [1, 21, 0]\nclass = b'),
 Text(697.5, 22.649999999999977, 'gini = 0.0\nsamples = 5\nvalue = [0, 12, 0]
\nclass = b'),
 Text(871.875, 22.649999999999977, 'gini = 0.18\nsamples = 5\nvalue = [1, 9,
0]\nclass = b'),
 Text(1133.4375, 67.94999999999999, 'NMHC <= 0.155\ngini = 0.618\nsamples = 2
2\nvalue = [6, 17, 16]\nclass = b'),
 Text(1046.25, 22.649999999999977, 'gini = 0.266\nsamples = 9\nvalue = [0, 3,
16]\nclass = c'),
 Text(1220.625, 22.649999999999977, 'gini = 0.42\nsamples = 13\nvalue = [6, 1
4, 0]\nclass = b'),
 Text(1656.5625, 113.25, 'OXY <= 0.745\ngini = 0.49\nsamples = 174\nvalue =
[94, 12, 185]\nclass = c'),
 Text(1482.1875, 67.94999999999999, 'SO_2 <= 7.085\ngini = 0.285\nsamples = 2
0\nvalue = [31, 3, 3]\nclass = a'),
 Text(1395.0, 22.649999999999977, 'gini = 0.0\nsamples = 11\nvalue = [22, 0,
0]\nclass = a'),
 Text(1569.375, 22.649999999999977, 'gini = 0.56\nsamples = 9\nvalue = [9, 3,
3]\nclass = a'),
 Text(1830.9375, 67.94999999999999, 'NMHC <= 0.185\ngini = 0.424\nsamples = 1
54\nvalue = [63, 9, 182]\nclass = c'),
 Text(1743.75, 22.649999999999977, 'gini = 0.204\nsamples = 107\nvalue = [20,
1, 162]\nclass = c'),
 Text(1918.125, 22.649999999999977, 'gini = 0.541\nsamples = 47\nvalue = [43,
8, 20]\nclass = a'),
 Text(2354.0625, 158.55, 'EBE <= 1.435\ngini = 0.284\nsamples = 101\nvalue =
```

```
[126, 0, 26]\nclass = a'),
 Text(2092.5, 113.25, 'NOx <= 107.4\ngini = 0.494\nsamples = 24\nvalue = [16,
0, 20]\nclass = c'),
 Text(2005.3125, 67.94999999999999, 'gini = 0.153\nsamples = 8\nvalue = [1,
0, 11]\nclass = c'),
 Text(2179.6875, 67.94999999999999, 'OXY <= 0.955\ngini = 0.469\nsamples = 16
\nvalue = [15, 0, 9]\nclass = a'),
 Text(2092.5, 22.649999999999977, 'gini = 0.198\nsamples = 6\nvalue = [8, 0,
1]\nclass = a'),
 Text(2266.875, 22.649999999999977, 'gini = 0.498\nsamples = 10\nvalue = [7,
0, 8]\nclass = c'),
 Text(2615.625, 113.25, 'EBE <= 2.205\ngini = 0.098\nsamples = 77\nvalue = [1
10, 0, 6]\nclass = a'),
 Text(2528.4375, 67.94999999999999, 'NMHC <= 0.21\ngini = 0.245\nsamples = 25
\nvalue = [36, 0, 6]\nclass = a'),
 Text(2441.25, 22.649999999999977, 'gini = 0.444\nsamples = 5\nvalue = [6, 0,
3]\nclass = a'),
 Text(2615.625, 22.649999999999977, 'gini = 0.165\nsamples = 20\nvalue = [30,
0, 3]\nclass = a'),
 Text(2702.8125, 67.94999999999999, 'gini = 0.0\nsamples = 52\nvalue = [74,
0, 0]\nclass = a')]
```



Conclusion

Linear Regression =0.478920116248428

Ridge Regression =0.478920116248428

Lasso Regression =0.1469008555565141

ElasticNet Regression =0.1469008555565141

Logistic Regression =0.8281788232899833

Randomforest =0.9051948051948051

Randomforest is suitable for this dataset

In [ ]: