mk 3/08/2023

In [85]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [86]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_200
df
```

Out[86]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.2600
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.5800
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.1900
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.5300
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.7600
...	...	...	...	...	...	...	...	...	...	...	
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.8300
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.9200
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.4600
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.0300
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.3600

215688 rows × 17 columns

In [87]:
```
df=df.dropna()
```

In [88]:
```
df=df.head(500)
```

In [89]:
```
df.columns
```

Out[89]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [90]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 500 entries, 3 to 4228
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     500 non-null    object 
 1   BEN      500 non-null    float64
 2   CO       500 non-null    float64
 3   EBE      500 non-null    float64
 4   MXY      500 non-null    float64
 5   NMHC     500 non-null    float64
 6   NO_2     500 non-null    float64
 7   NOx      500 non-null    float64
 8   OXY      500 non-null    float64
 9   O_3      500 non-null    float64
 10  PM10     500 non-null    float64
 11  PM25     500 non-null    float64
 12  PXY      500 non-null    float64
 13  SO_2     500 non-null    float64
 14  TCH      500 non-null    float64
 15  TOL      500 non-null    float64
 16  station  500 non-null    int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 70.3+ KB
```

In [91]:
```
data=df[['BEN', 'TOL', 'PXY']]
data
```

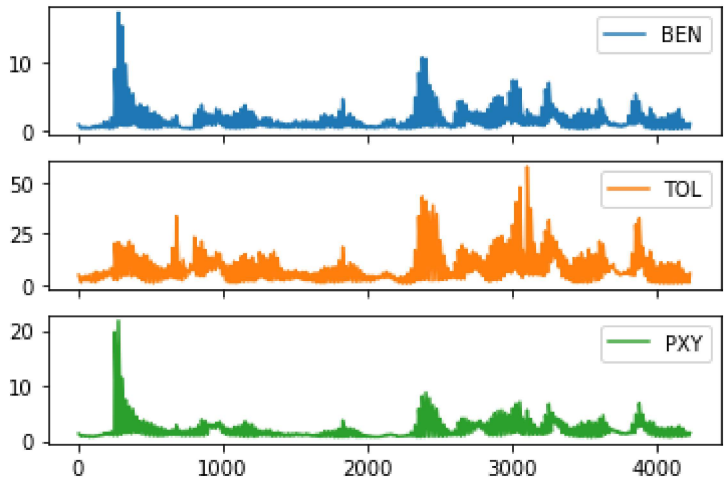
Out[91]:

	BEN	TOL	PXY
3	0.95	4.62	1.30
20	0.38	1.10	0.84
24	0.55	3.64	1.07
28	0.65	3.82	1.04
45	0.38	1.94	0.88
...	...	...	...
4199	0.59	3.73	1.58
4203	1.16	5.12	1.62
4220	0.43	0.89	0.91
4224	0.44	2.44	1.26
4228	1.12	5.46	1.43

500 rows × 3 columns

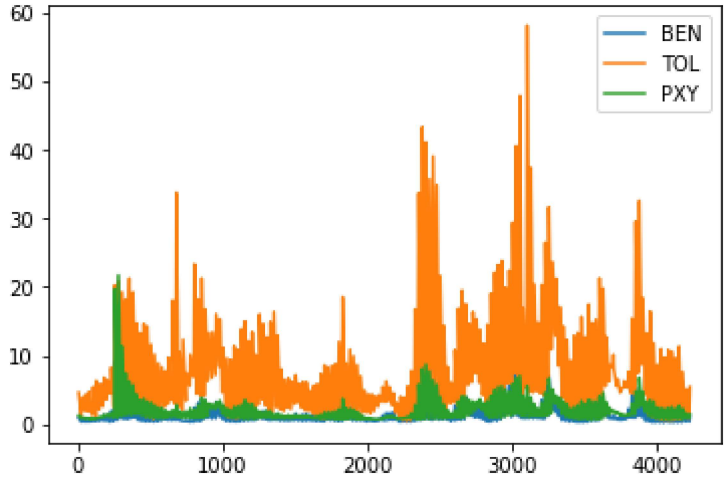
In [92]:
```
data.plot.line(subplots=True)
```

Out[92]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [93]:
```
data.plot.line()
```

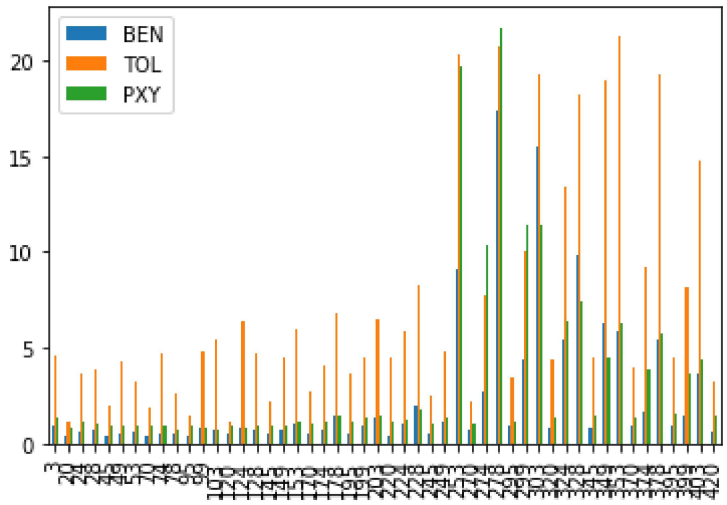
Out[93]: <AxesSubplot:>



In [94]:
```
b=data[0:50]
```

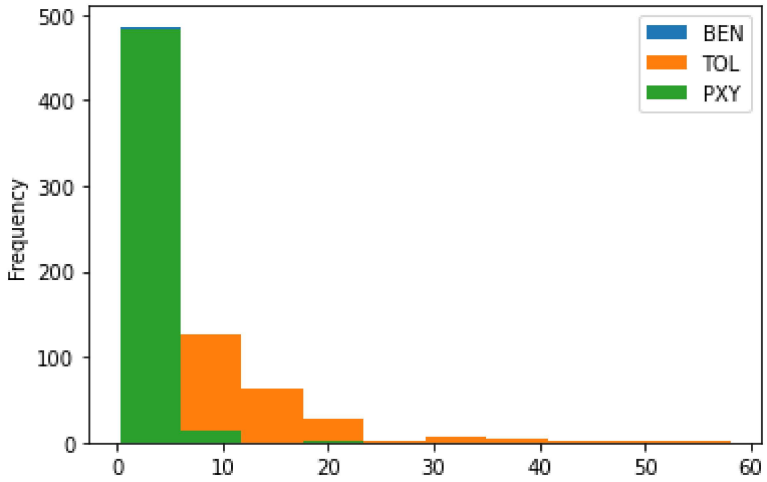
In [95]:
```
b.plot.bar()
```

Out[95]: <AxesSubplot:>



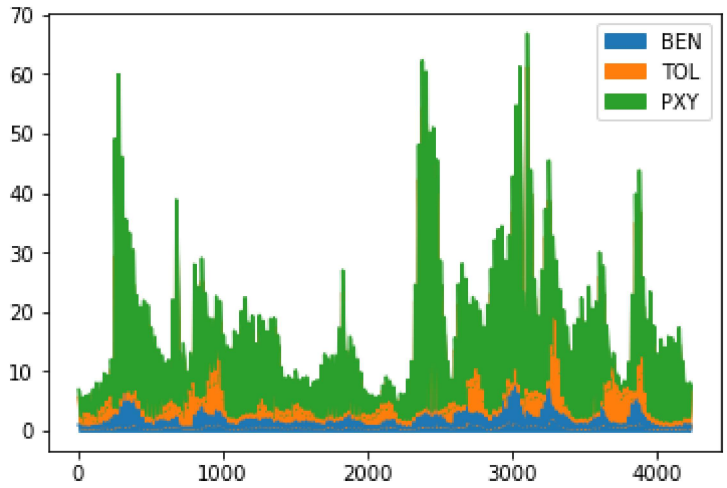
In [96]:
```
data.plot.hist()
```

Out[96]: <AxesSubplot:ylabel='Frequency'>



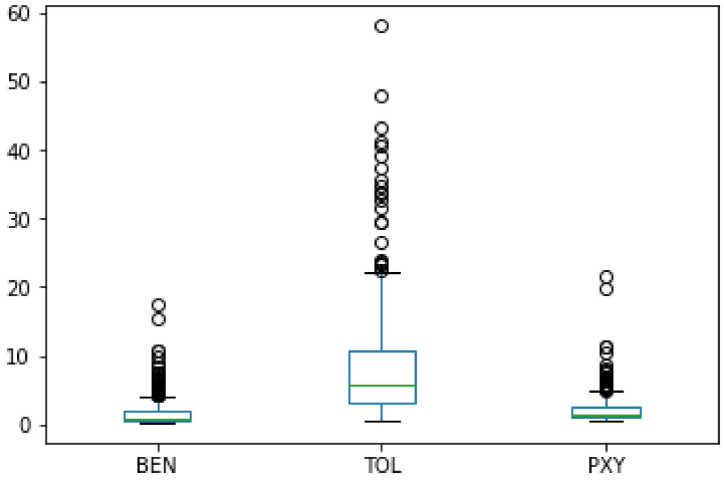
In [97]:
```
data.plot.area()
```

Out[97]: <AxesSubplot:>



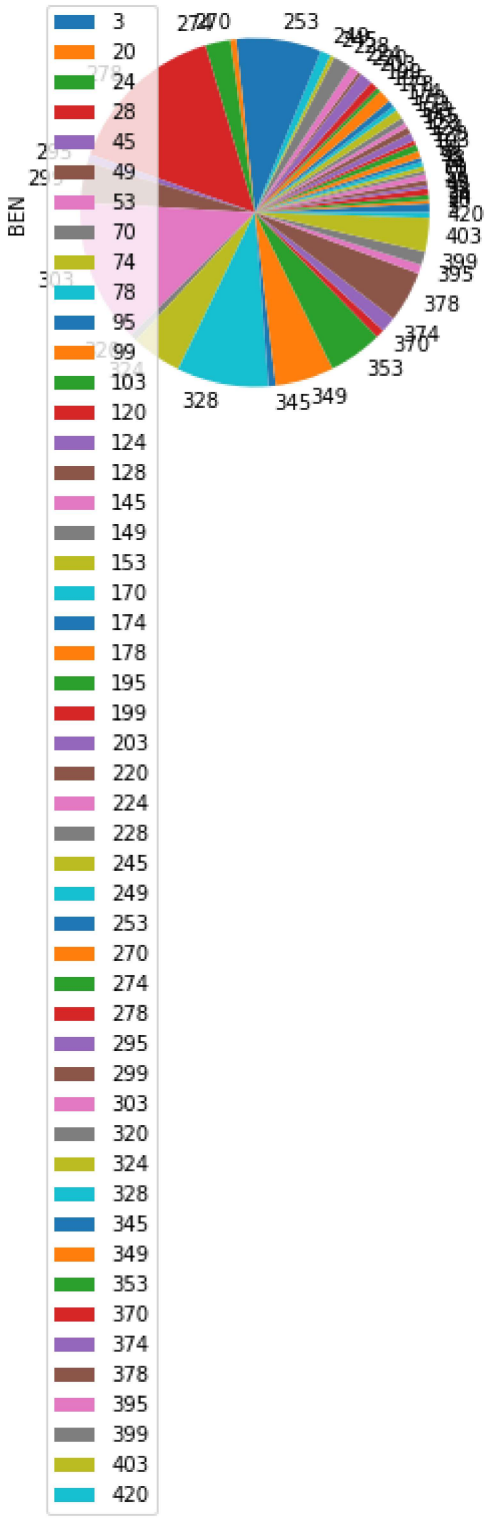
In [98]:
```
data.plot.box()
```

Out[98]: <AxesSubplot:>



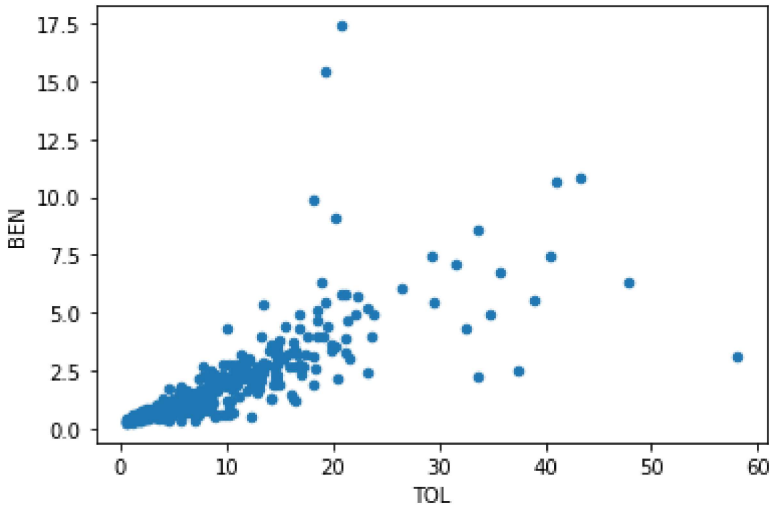
In [99]:
```
b.plot.pie(y='BEN' )
```

Out[99]: <AxesSubplot:ylabel='BEN'>



In [100]:
```
data.plot.scatter(x='TOL' ,y='BEN')
```

Out[100]: <AxesSubplot:xlabel='TOL', ylabel='BEN'>



In [101]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 500 entries, 3 to 4228
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     500 non-null    object 
 1   BEN      500 non-null    float64
 2   CO       500 non-null    float64
 3   EBE      500 non-null    float64
 4   MXY      500 non-null    float64
 5   NMHC     500 non-null    float64
 6   NO_2     500 non-null    float64
 7   NOx      500 non-null    float64
 8   OXY      500 non-null    float64
 9   O_3      500 non-null    float64
 10  PM10     500 non-null    float64
 11  PM25     500 non-null    float64
 12  PXY      500 non-null    float64
 13  SO_2     500 non-null    float64
 14  TCH      500 non-null    float64
 15  TOL      500 non-null    float64
 16  station  500 non-null    int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 70.3+ KB
```

In [102]:
```
df.describe()
```

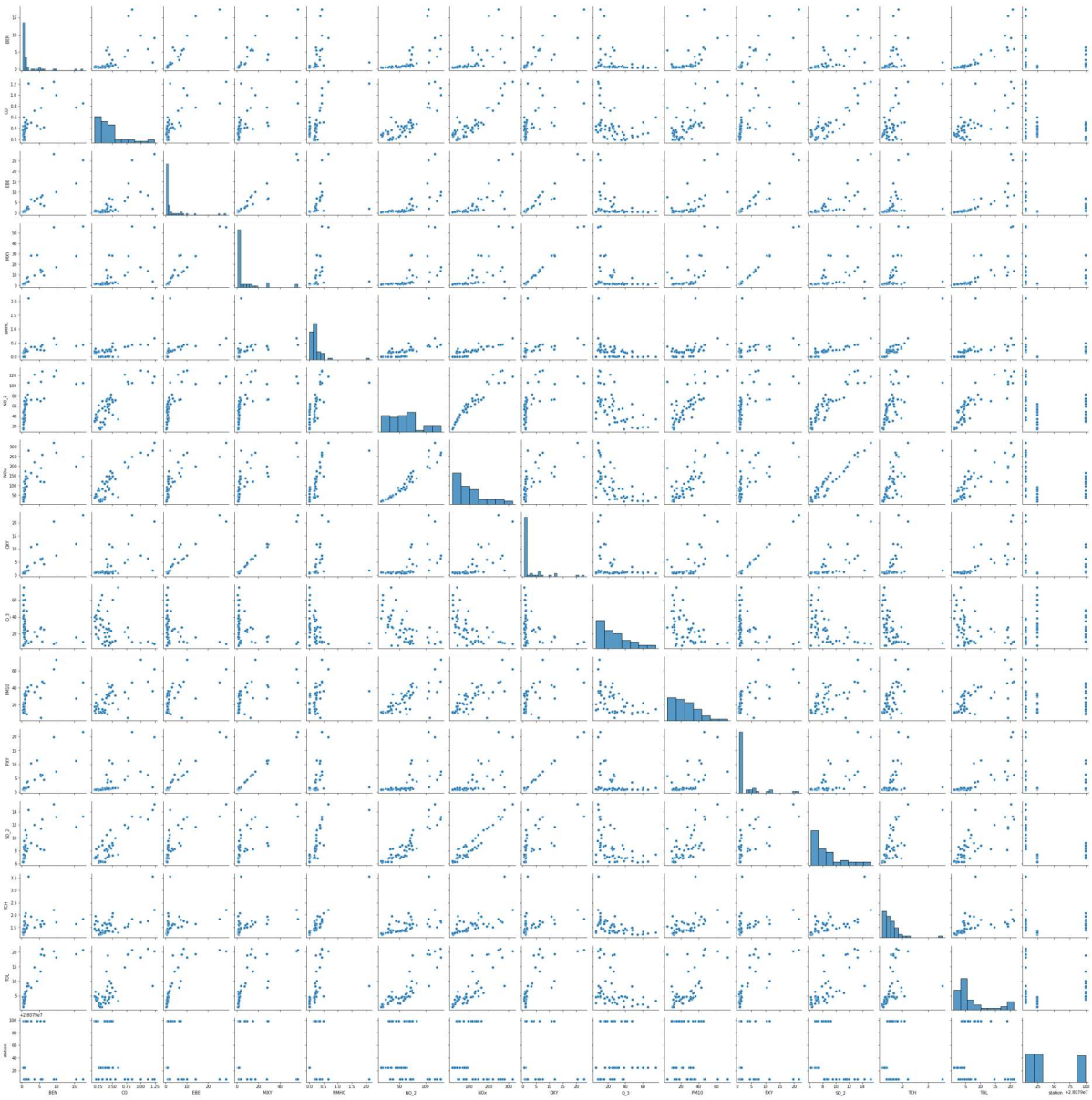
Out[102]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.(
mean	1.617720	0.629740	2.019740	4.501660	0.381560	80.619260	145.999140	2.2
std	1.808122	0.351552	2.360526	5.064028	0.216003	48.576316	131.500811	2.1
min	0.270000	0.190000	0.290000	0.470000	0.000000	3.260000	5.500000	0.4
25%	0.600000	0.407500	0.707500	1.560000	0.240000	48.157500	62.592499	1.(
50%	0.950000	0.590000	1.340000	3.110000	0.330000	73.549999	113.950001	1.5
75%	1.995000	0.740000	2.600000	5.582500	0.490000	107.150000	188.725006	2.6
max	17.400000	3.080000	28.410000	56.500000	2.110000	323.700012	991.000000	22.9

In [103]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [104]:
```
sns.pairplot(df1[0:50])
```

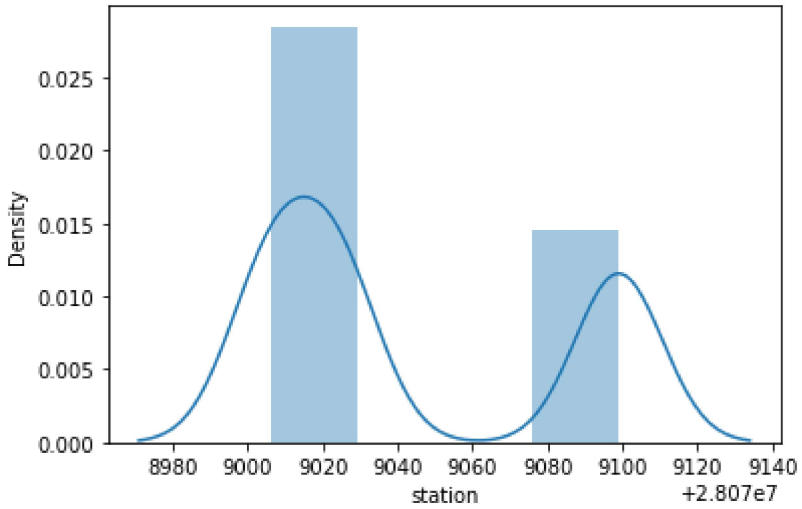
Out[104]:
```
<seaborn.axisgrid.PairGrid at 0x1d8eb50d0a0>
```



In [105]:
```
sns.distplot(df1['station'])
```

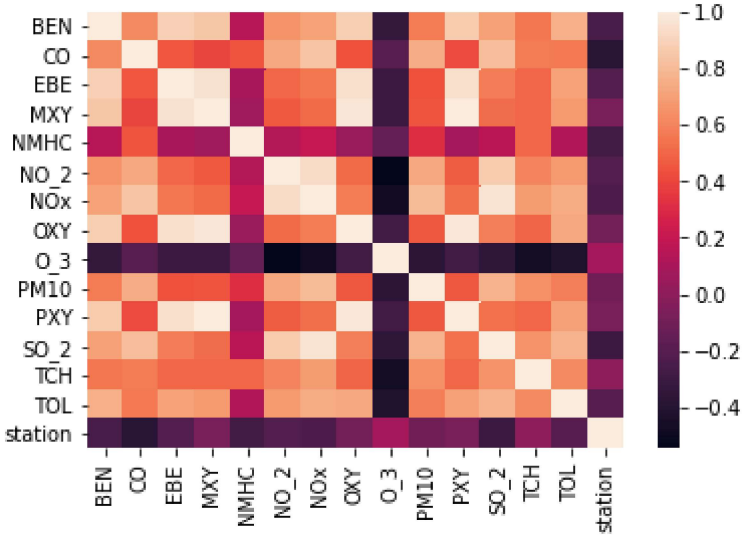
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[105]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [106]:
```
sns.heatmap(df1.corr())
```

Out[106]: <AxesSubplot:>



In [107]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [108]:
```
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 [109]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[109]:
```
LinearRegression()
```

In [110]:
```
lr.intercept_
```

Out[110]:
```
28079035.114567645
```

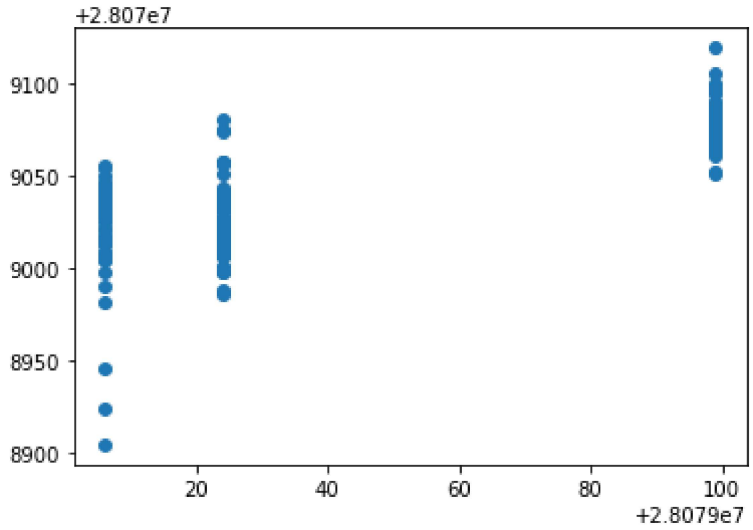
In [111]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[111]:

	Co-efficient
BEN	-1.529029
CO	-80.072373
EBE	-14.460490
MXY	-5.494208
NMHC	-70.172628
NO_2	-0.365132
NOx	0.714227
OXY	10.585323
O_3	0.458734
PM10	0.561037
PXY	18.086535
SO_2	-17.869992
TCH	99.511060
TOL	0.054121

In [112]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[112]: <matplotlib.collections.PathCollection at 0x1d8d94465b0>



In [113]:
```
lr.score(x_test,y_test)
```

Out[113]: 0.5650070997769747

In [114]:
```
lr.score(x_train,y_train)
```

Out[114]: 0.6363409296416573

In [115]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [116]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[116]: Ridge(alpha=10)

In [117]:
```
rr.score(x_test,y_test)
```

Out[117]: 0.5550176255059163

In [118]:
```
rr.score(x_train,y_train)
```

Out[118]: 0.5536340427117608

In [119]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[119]: Lasso(alpha=10)

In [120]:
```
la.score(x_train,y_train)
```

Out[120]:
```
0.1473133779060921
```

In [121]:
```
la.score(x_test,y_test)
```

Out[121]:
```
0.050305076369285606
```

In [122]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[122]:
```
ElasticNet()
```

In [123]:
```
en.coef_
```

Out[123]:
```
array([-5.14918708, -4.12863031, -7.81885611,  3.8410681 , -3.47471994,
       -0.18070396,  0.25314536,  1.90276967,  0.07700695,  0.37065497,
        0.88634726, -8.75372749,  0.76613778,  0.08028506])
```

In [124]:
```
en.intercept_
```

Out[124]:
```
28079099.174885258
```

In [125]:
```
prediction=en.predict(x_test)
```

In [126]:
```
en.score(x_test,y_test)
```

Out[126]:
```
0.2762737078871572
```

In [127]:
```
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)))
```

```
31.212575071404377
1103.791443906419
33.223356903034635
```

In [128]:
```
from sklearn.linear_model import LogisticRegression
```

In [129]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [130]:
```
feature_matrix.shape
```

Out[130]: (500, 14)

In [131]:
```
target_vector.shape
```

Out[131]: (500,)

In [132]:
```
from sklearn.preprocessing import StandardScaler
```

In [133]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [134]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[134]: LogisticRegression(max_iter=10000)

In [135]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [136]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [137]:
```
logr.score(fs,target_vector)
```

Out[137]: 0.98

In [138]:
```
logr.predict_proba(observation)[0][0]
```

Out[138]: 0.19142108470373131

In [139]:
```
logr.predict_proba(observation)
```

Out[139]: array([[1.91421085e-01, 1.87975019e-46, 8.08578915e-01]])

In [140]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [141]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[141]: RandomForestClassifier()

In [142]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]}
```

In [143]:
```
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[143]:
```
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')
```

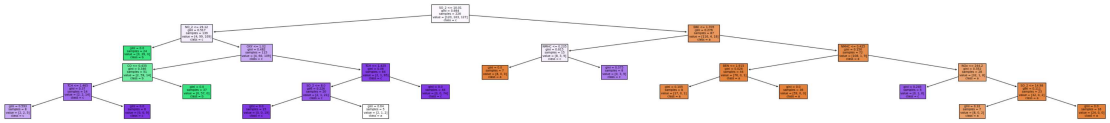
In [144]:
```
grid_search.best_score_
```

Out[144]:
```
0.9057142857142857
```

In [145]:
```
rfc_best=grid_search.best_estimator_
```

In [146]:
```
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[146]:
```
[Text(1150.875, 249.15, 'SO_2 <= 10.01\ngini = 0.664\nsamples = 226\nvalue =
[120, 103, 127]\nclass = c'),
 Text(558.0, 203.85000000000002, 'NO_2 <= 29.12\ngini = 0.517\nsamples = 139
\nvalue = [4, 99, 109]\nclass = c'),
 Text(418.5, 158.55, 'gini = 0.0\nsamples = 24\nvalue = [0, 39, 0]\nclass =
b'),
 Text(697.5, 158.55, 'OXY <= 1.02\ngini = 0.482\nsamples = 115\nvalue = [4, 6
0, 109]\nclass = c'),
 Text(418.5, 113.25, 'CO <= 0.435\ngini = 0.346\nsamples = 51\nvalue = [2, 5
9, 14]\nclass = b'),
 Text(279.0, 67.94999999999999, 'TCH <= 1.465\ngini = 0.37\nsamples = 14\nval
ue = [2, 2, 14]\nclass = c'),
 Text(139.5, 22.649999999999977, 'gini = 0.593\nsamples = 8\nvalue = [2, 2,
5]\nclass = c'),
 Text(418.5, 22.649999999999977, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 9]
\nclass = c'),
 Text(558.0, 67.94999999999999, 'gini = 0.0\nsamples = 37\nvalue = [0, 57, 0]
\nclass = b'),
 Text(976.5, 113.25, 'TCH <= 1.425\ngini = 0.06\nsamples = 64\nvalue = [2, 1,
95]\nclass = c'),
 Text(837.0, 67.94999999999999, 'SO_2 <= 8.0\ngini = 0.226\nsamples = 20\nval
ue = [2, 1, 21]\nclass = c'),
 Text(697.5, 22.649999999999977, 'gini = 0.0\nsamples = 15\nvalue = [0, 0, 1
9]\nclass = c'),
 Text(976.5, 22.649999999999977, 'gini = 0.64\nsamples = 5\nvalue = [2, 1, 2]
\nclass = a'),
 Text(1116.0, 67.94999999999999, 'gini = 0.0\nsamples = 44\nvalue = [0, 0, 7
4]\nclass = c'),
 Text(1743.75, 203.85000000000002, 'EBE <= 1.705\ngini = 0.276\nsamples = 87
\nvalue = [116, 4, 18]\nclass = a'),
 Text(1395.0, 158.55, 'NMHC <= 0.235\ngini = 0.615\nsamples = 15\nvalue = [8,
3, 9]\nclass = c'),
 Text(1255.5, 113.25, 'gini = 0.0\nsamples = 7\nvalue = [8, 0, 0]\nclass =
a'),
 Text(1534.5, 113.25, 'gini = 0.375\nsamples = 8\nvalue = [0, 3, 9]\nclass =
c'),
 Text(2092.5, 158.55, 'NMHC <= 0.425\ngini = 0.156\nsamples = 72\nvalue = [10
8, 1, 9]\nclass = a'),
 Text(1813.5, 113.25, 'BEN <= 1.915\ngini = 0.026\nsamples = 44\nvalue = [76,
0, 1]\nclass = a'),
 Text(1674.0, 67.94999999999999, 'gini = 0.105\nsamples = 8\nvalue = [17, 0,
1]\nclass = a'),
 Text(1953.0, 67.94999999999999, 'gini = 0.0\nsamples = 36\nvalue = [59, 0,
0]\nclass = a'),
 Text(2371.5, 113.25, 'NOx <= 244.2\ngini = 0.352\nsamples = 28\nvalue = [32,
1, 8]\nclass = a'),
 Text(2232.0, 67.94999999999999, 'gini = 0.245\nsamples = 5\nvalue = [0, 1,
6]\nclass = c'),
 Text(2511.0, 67.94999999999999, 'SO_2 <= 14.98\ngini = 0.111\nsamples = 23\n
value = [32, 0, 2]\nclass = a'),
 Text(2371.5, 22.649999999999977, 'gini = 0.32\nsamples = 7\nvalue = [8, 0,
2]\nclass = a'),
 Text(2650.5, 22.649999999999977, 'gini = 0.0\nsamples = 16\nvalue = [24, 0,
0]\nclass = a')]
```



Conclusion

```
Linear Regression =0.6363409296416573

Ridge Regression =0.5536340427117608

Lasso Regression =0.1473133779060921

ElasticNet Regression =0.2762737078871572

Logistic Regression =0.19142108470373131

Randomforest =0.9057142857142857
```

```
Randomforest is suitable for this dataset
```

In [ ]: