mk 3/08/2023

In [615]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [616]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_200
df
```

Out[616]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.419
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180
...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490

230568 rows × 17 columns

In [617]:
```
df=df.dropna()
```

In [618]:
```
df=df.head(1200)
```

In [619]:
```
df.columns
```

Out[619]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [620]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1200 entries, 5 to 10485
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1200 non-null   object 
 1   BEN      1200 non-null   float64
 2   CO       1200 non-null   float64
 3   EBE      1200 non-null   float64
 4   MXY      1200 non-null   float64
 5   NMHC     1200 non-null   float64
 6   NO_2     1200 non-null   float64
 7   NOx      1200 non-null   float64
 8   OXY      1200 non-null   float64
 9   O_3      1200 non-null   float64
 10  PM10     1200 non-null   float64
 11  PM25     1200 non-null   float64
 12  PXY      1200 non-null   float64
 13  SO_2     1200 non-null   float64
 14  TCH      1200 non-null   float64
 15  TOL      1200 non-null   float64
 16  station  1200 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 168.8+ KB
```

In [621]:
```
data=df[['BEN', 'TOL', 'PXY']]
data
```

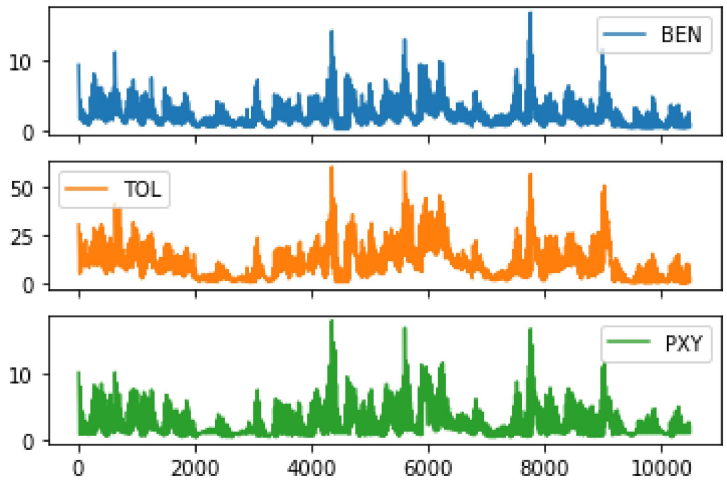
Out[621]:

	BEN	TOL	PXY
5	9.41	30.450001	10.11
22	1.69	5.350000	0.79
25	2.35	11.340000	4.46
31	4.39	26.219999	8.06
48	1.93	6.130000	0.82
...	...	...	...
10457	0.44	0.700000	1.00
10460	0.79	3.170000	1.78
10465	2.66	9.970000	2.66
10482	0.47	0.950000	1.00
10485	0.94	3.170000	2.01

1200 rows × 3 columns

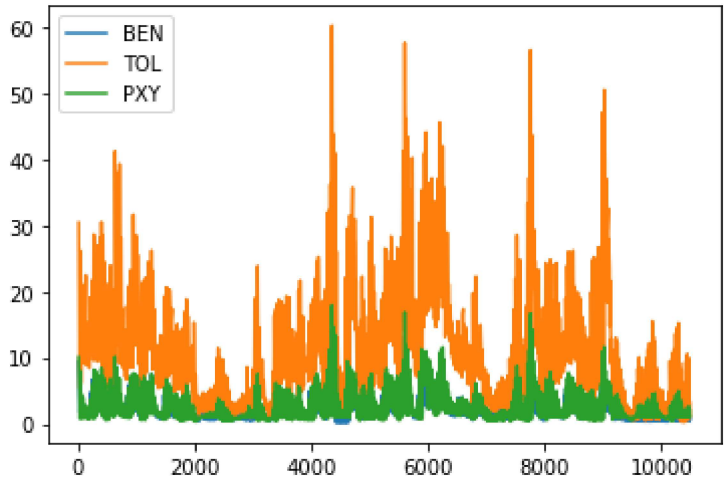
In [622]:
```
data.plot.line(subplots=True)
```

Out[622]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [623]:
```
data.plot.line()
```

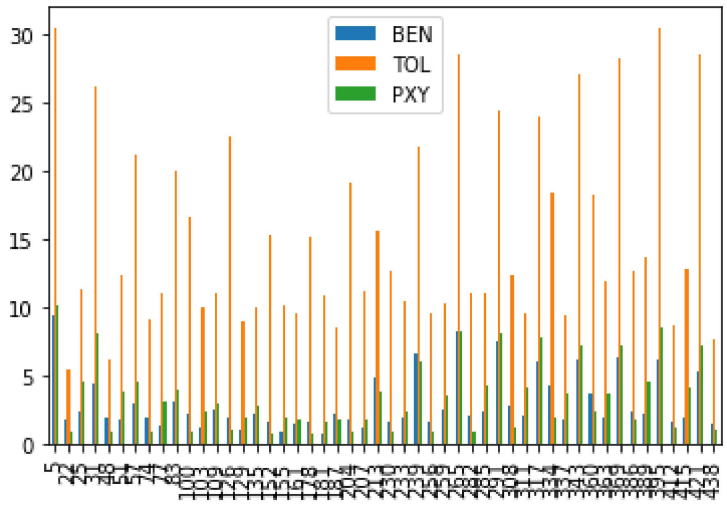
Out[623]: <AxesSubplot:>



In [624]:
```
b=data[0:50]
```

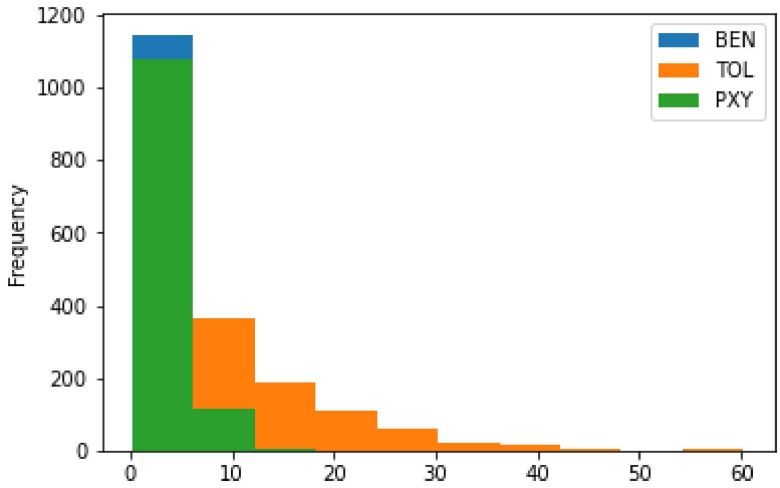
In [625]:
```
b.plot.bar()
```

Out[625]: <AxesSubplot:>



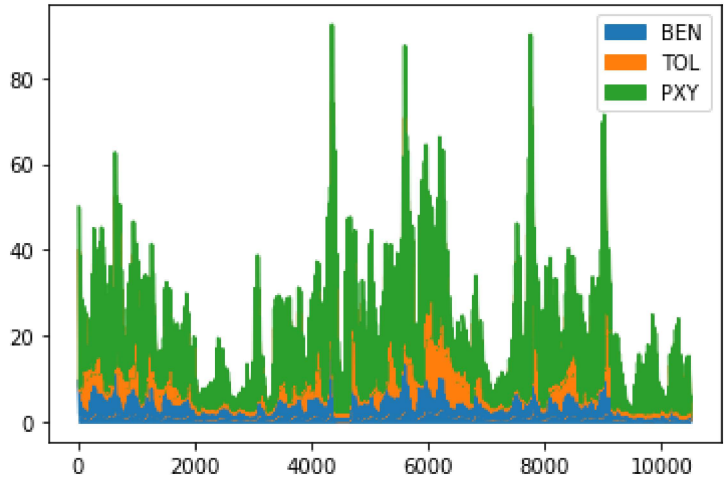
In [626]:
```
data.plot.hist()
```

Out[626]: <AxesSubplot:ylabel='Frequency'>



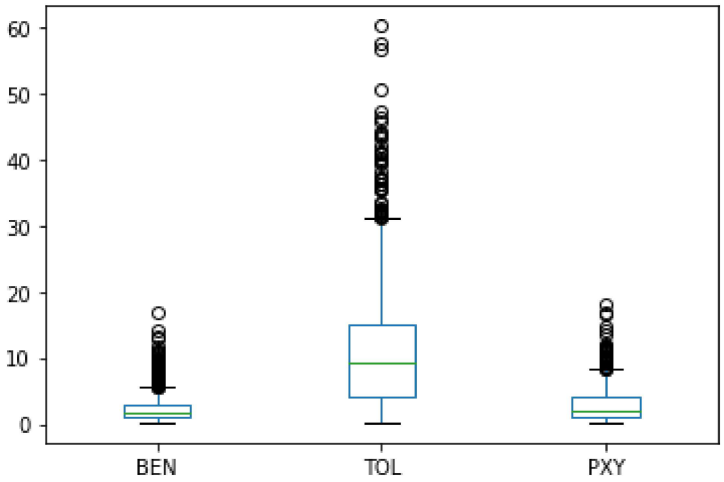
In [627]:
```
data.plot.area()
```

Out[627]: <AxesSubplot:>



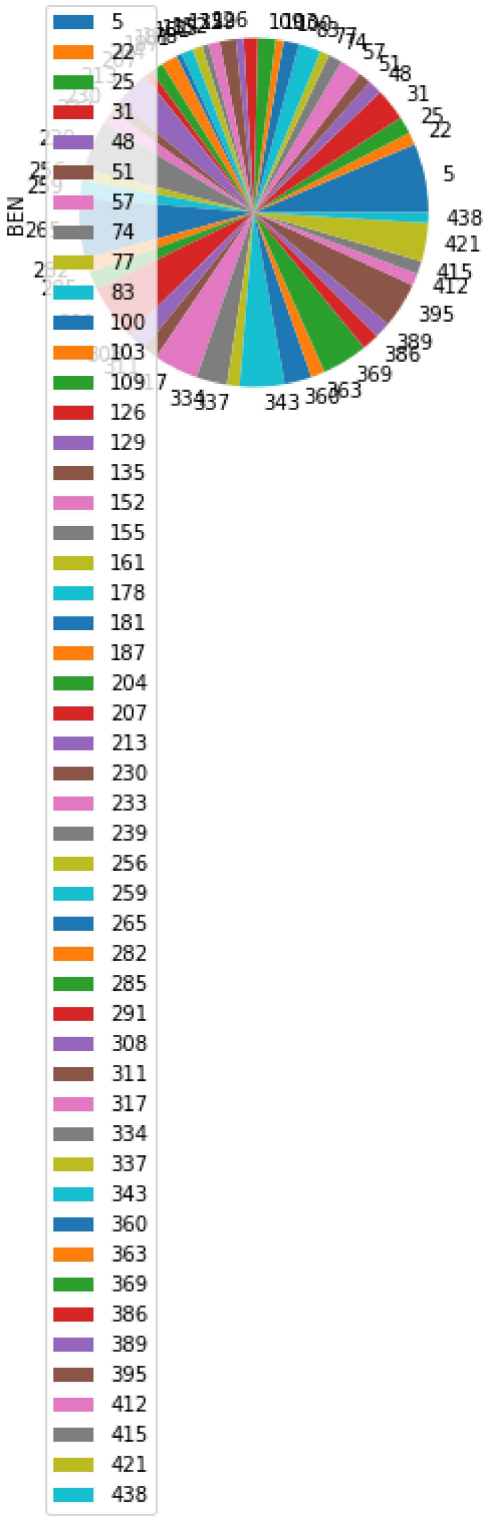
In [628]:
```
data.plot.box()
```

Out[628]: <AxesSubplot:>



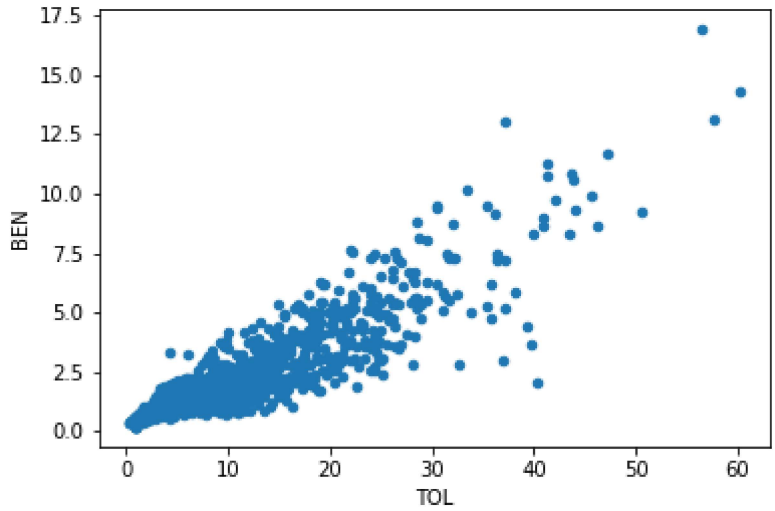
In [629]:
```
b.plot.pie(y='BEN' )
```

Out[629]: <AxesSubplot:ylabel='BEN'>



In [630]:
```
data.plot.scatter(x='TOL' ,y='BEN')
```

Out[630]: <AxesSubplot:xlabel='TOL', ylabel='BEN'>



In [631]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1200 entries, 5 to 10485
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     1200 non-null   object 
 1   BEN      1200 non-null   float64
 2   CO       1200 non-null   float64
 3   EBE      1200 non-null   float64
 4   MXY      1200 non-null   float64
 5   NMHC     1200 non-null   float64
 6   NO_2     1200 non-null   float64
 7   NOx      1200 non-null   float64
 8   OXY      1200 non-null   float64
 9   O_3      1200 non-null   float64
 10  PM10     1200 non-null   float64
 11  PM25     1200 non-null   float64
 12  PXY      1200 non-null   float64
 13  SO_2     1200 non-null   float64
 14  TCH      1200 non-null   float64
 15  TOL      1200 non-null   float64
 16  station  1200 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 168.8+ KB
```

In [632]:
```
df.describe()
```

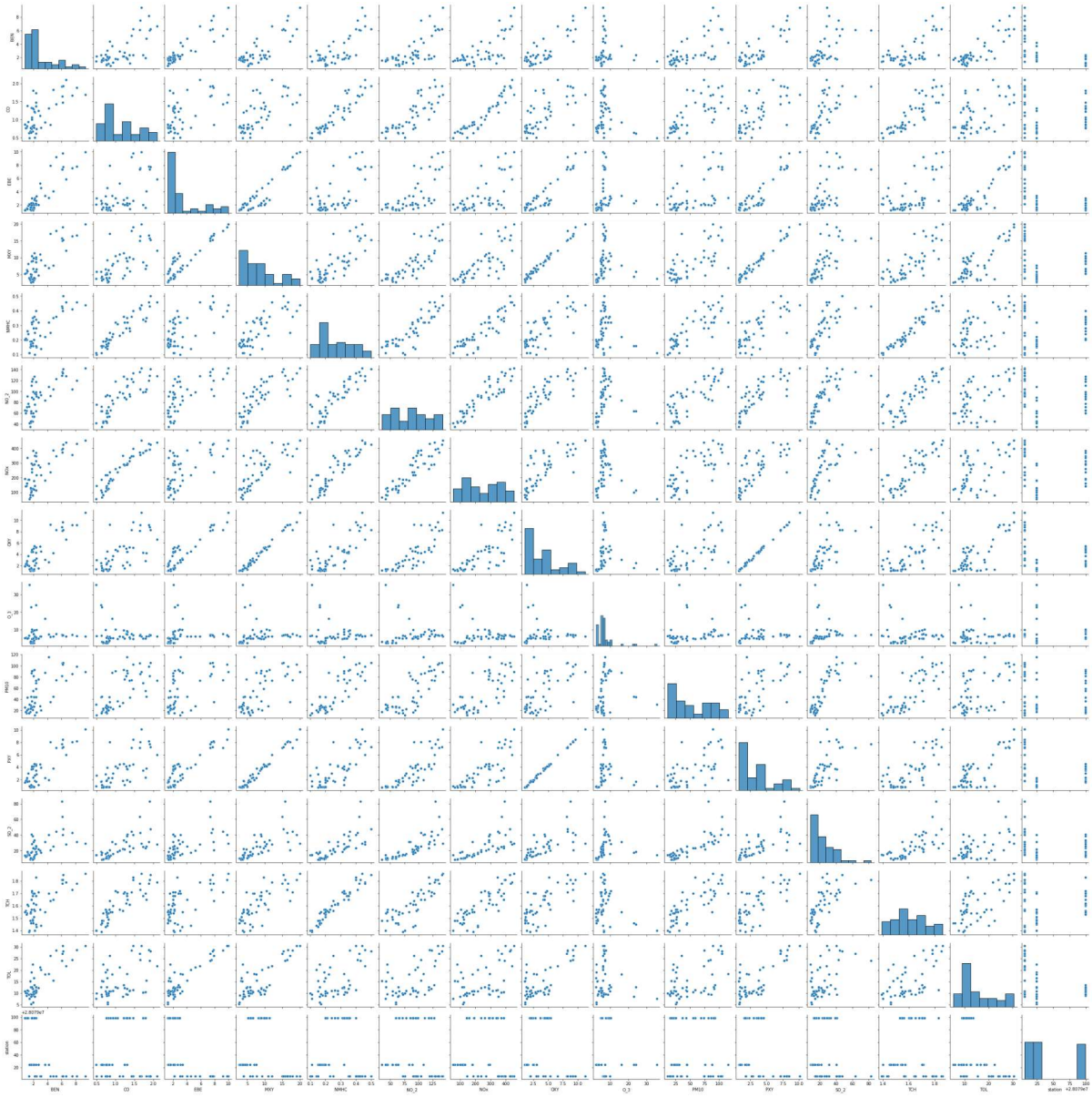
Out[632]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NO
count	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.00000
mean	2.305467	0.960658	2.867808	6.877733	0.226750	80.715525	198.47935
std	1.940453	0.552812	2.566230	5.573970	0.148778	43.627412	159.64830
min	0.200000	0.220000	0.240000	0.250000	0.010000	3.750000	4.10000
25%	1.030000	0.590000	1.060000	2.577500	0.120000	46.712501	81.70750
50%	1.700000	0.810000	1.985000	5.535000	0.190000	77.815002	160.94999
75%	2.912500	1.200000	3.875000	9.662500	0.300000	108.699997	274.82501
max	16.900000	4.790000	17.209999	37.290001	1.160000	332.200012	1279.00000

In [633]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [634]:
```
sns.pairplot(df1[0:50])
```

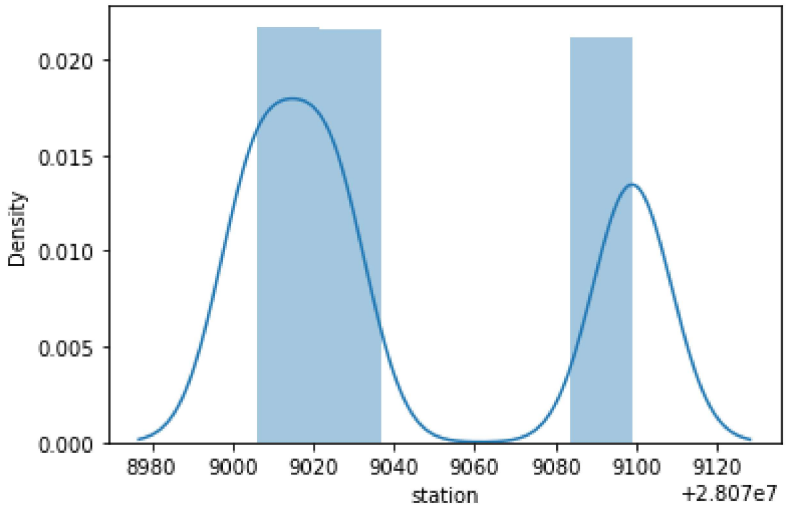
Out[634]: <seaborn.axisgrid.PairGrid at 0x19199ca9ac0>



In [635]:
```
sns.distplot(df1['station'])
```

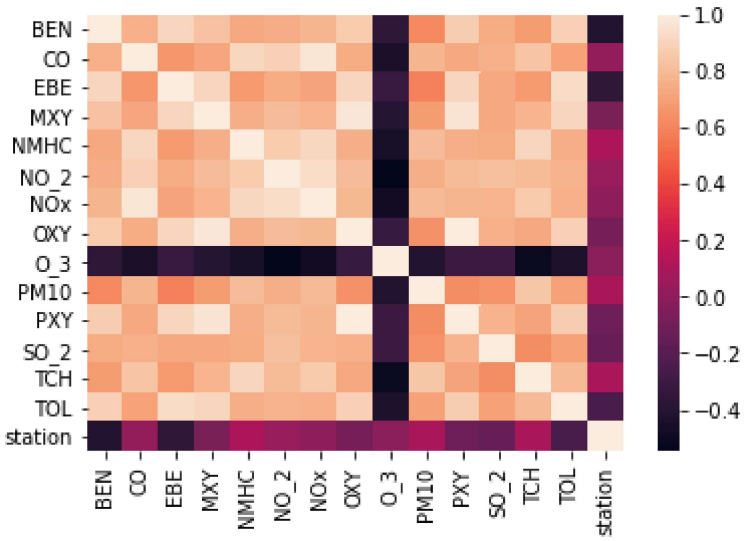
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[635]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [636]:
```
sns.heatmap(df1.corr())
```

Out[636]: <AxesSubplot:>



In [637]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [638]:
```
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 [639]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[639]:
```
LinearRegression()
```

In [640]:
```
lr.intercept_
```

Out[640]:
```
28078976.370691862
```

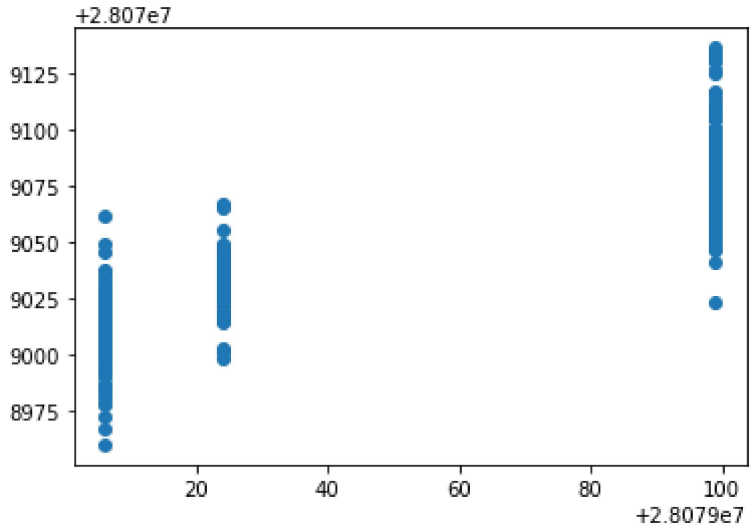
In [641]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[641]:

	Co-efficient
BEN	-19.739142
CO	41.029333
EBE	-13.569603
MXY	0.650562
NMHC	96.200254
NO_2	0.317863
NOx	-0.183311
OXY	24.982099
O_3	0.140500
PM10	0.071323
PXY	-9.987189
SO_2	-0.602394
TCH	32.726405
TOL	-0.280711

In [642]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[642]: <matplotlib.collections.PathCollection at 0x191a86d2730>



In [643]:
```
lr.score(x_test,y_test)
```

Out[643]: 0.7478612503224795

In [644]:
```
lr.score(x_train,y_train)
```

Out[644]: 0.7548303061675042

In [645]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [646]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[646]: Ridge(alpha=10)

In [647]:
```
rr.score(x_test,y_test)
```

Out[647]: 0.730455761410577

In [648]:
```
rr.score(x_train,y_train)
```

Out[648]: 0.7360079138814519

In [649]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[649]: Lasso(alpha=10)

In [650]:
```
la.score(x_train,y_train)
```

Out[650]:
```
0.5155679352303824
```

In [651]:
```
la.score(x_test,y_test)
```

Out[651]:
```
0.49401180765363784
```

In [652]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[652]:
```
ElasticNet()
```

In [653]:
```
en.coef_
```

Out[653]:
```
array([-9.74325309,  0.42713135, -9.54667112,  5.19822817,  0.15762478,
        0.38788312,  0.01367113,  2.68518929,  0.1707266 ,  0.33814657,
        0.7952624 , -0.72867231,  0.33362304, -2.11259189])
```

In [654]:
```
en.intercept_
```

Out[654]:
```
28079031.167656854
```

In [655]:
```
prediction=en.predict(x_test)
```

In [656]:
```
en.score(x_test,y_test)
```

Out[656]:
```
0.6237301983555076
```

In [657]:
```
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)))
```

```
19.856055700872094
603.5177131790576
24.56659750920053
```

In [658]:
```
from sklearn.linear_model import LogisticRegression
```

In [659]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [660]:
```
feature_matrix.shape
```

Out[660]: (1200, 14)

In [661]:
```
target_vector.shape
```

Out[661]: (1200,)

In [662]:
```
from sklearn.preprocessing import StandardScaler
```

In [663]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [664]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[664]: LogisticRegression(max_iter=10000)

In [665]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [666]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [667]:
```
logr.score(fs,target_vector)
```

Out[667]: 0.9716666666666667

In [668]:
```
logr.predict_proba(observation)[0][0]
```

Out[668]: 0.0024892676597109433

In [669]:
```
logr.predict_proba(observation)
```

Out[669]: array([[2.48926766e-03, 4.86609702e-23, 9.97510732e-01]])

In [670]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [671]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[671]: RandomForestClassifier()

In [672]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]}
```

In [673]:
```
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[673]:
```
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 [674]:
```
grid_search.best_score_
```

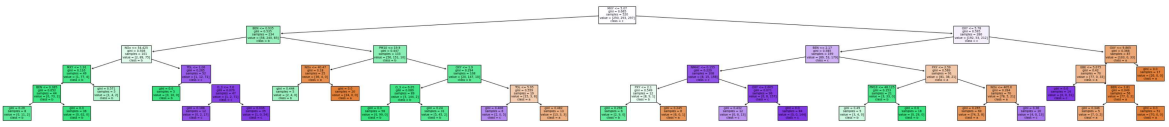
Out[674]:
```
0.9369047619047619
```

In [675]:
```
rfc_best=grid_search.best_estimator_
```

In [676]:
```
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[676]:
```
[Text(1473.46875, 249.15, 'MXY <= 5.07\ngini = 0.665\nsamples = 520\nvalue =
[250, 293, 297]\nclass = c'),
 Text(645.1875, 203.85000000000002, 'BEN <= 0.935\ngini = 0.535\nsamples = 23
4\nvalue = [58, 240, 85]\nclass = b'),
 Text(348.75, 158.55, 'NOx <= 54.425\ngini = 0.508\nsamples = 101\nvalue =
[2, 89, 75]\nclass = b'),
 Text(209.25, 113.25, 'MXY <= 1.91\ngini = 0.116\nsamples = 49\nvalue = [1, 7
7, 4]\nclass = b'),
 Text(139.5, 67.94999999999999, 'BEN <= 0.385\ngini = 0.052\nsamples = 44\nva
lue = [0, 73, 2]\nclass = b'),
 Text(69.75, 22.649999999999977, 'gini = 0.26\nsamples = 8\nvalue = [0, 11,
2]\nclass = b'),
 Text(209.25, 22.649999999999977, 'gini = 0.0\nsamples = 36\nvalue = [0, 62,
0]\nclass = b'),
 Text(279.0, 67.94999999999999, 'gini = 0.571\nsamples = 5\nvalue = [1, 4, 2]
\nclass = b'),
 Text(488.25, 113.25, 'TOL <= 1.06\ngini = 0.265\nsamples = 52\nvalue = [1, 1
2, 71]\nclass = c'),
 Text(418.5, 67.94999999999999, 'gini = 0.0\nsamples = 5\nvalue = [0, 10, 0]
\nclass = b'),
 Text(558.0, 67.94999999999999, 'O_3 <= 7.6\ngini = 0.079\nsamples = 47\nvalu
e = [1, 2, 71]\nclass = c'),
 Text(488.25, 22.649999999999977, 'gini = 0.188\nsamples = 12\nvalue = [0, 2,
17]\nclass = c'),
 Text(627.75, 22.649999999999977, 'gini = 0.036\nsamples = 35\nvalue = [1, 0,
54]\nclass = c'),
 Text(941.625, 158.55, 'PM10 <= 19.9\ngini = 0.447\nsamples = 133\nvalue = [5
6, 151, 10]\nclass = b'),
 Text(767.25, 113.25, 'NOx <= 40.47\ngini = 0.18\nsamples = 25\nvalue = [36,
4, 0]\nclass = a'),
 Text(697.5, 67.94999999999999, 'gini = 0.444\nsamples = 5\nvalue = [2, 4, 0]
\nclass = b'),
 Text(837.0, 67.94999999999999, 'gini = 0.0\nsamples = 20\nvalue = [34, 0, 0]
\nclass = a'),
 Text(1116.0, 113.25, 'OXY <= 1.9\ngini = 0.294\nsamples = 108\nvalue = [20,
147, 10]\nclass = b'),
 Text(976.5, 67.94999999999999, 'O_3 <= 6.05\ngini = 0.089\nsamples = 89\nval
ue = [5, 144, 2]\nclass = b'),
 Text(906.75, 22.649999999999977, 'gini = 0.0\nsamples = 58\nvalue = [0, 99,
0]\nclass = b'),
 Text(1046.25, 22.649999999999977, 'gini = 0.24\nsamples = 31\nvalue = [5, 4
5, 2]\nclass = b'),
 Text(1255.5, 67.94999999999999, 'TOL <= 5.95\ngini = 0.559\nsamples = 19\nva
lue = [15, 3, 8]\nclass = a'),
 Text(1185.75, 22.649999999999977, 'gini = 0.408\nsamples = 6\nvalue = [2, 0,
5]\nclass = c'),
 Text(1325.25, 22.649999999999977, 'gini = 0.482\nsamples = 13\nvalue = [13,
3, 3]\nclass = a'),
 Text(2301.75, 203.85000000000002, 'OXY <= 5.78\ngini = 0.595\nsamples = 286
\nvalue = [192, 53, 212]\nclass = c'),
 Text(1953.0, 158.55, 'BEN <= 2.17\ngini = 0.585\nsamples = 199\nvalue = [89,
53, 179]\nclass = c'),
 Text(1674.0, 113.25, 'NMHC <= 0.155\ngini = 0.229\nsamples = 108\nvalue =
[8, 15, 158]\nclass = c'),
 Text(1534.5, 67.94999999999999, 'PXY <= 2.1\ngini = 0.549\nsamples = 12\nval
ue = [8, 9, 1]\nclass = b'),
 Text(1464.75, 22.649999999999977, 'gini = 0.298\nsamples = 6\nvalue = [2, 9,
```

```
0]\nclass = b'),
 Text(1604.25, 22.649999999999977, 'gini = 0.245\nsamples = 6\nvalue = [6, 0,
1]\nclass = a'),
 Text(1813.5, 67.94999999999999, 'OXY <= 2.605\ngini = 0.071\nsamples = 96\nv
alue = [0, 6, 157]\nclass = c'),
 Text(1743.75, 22.649999999999977, 'gini = 0.432\nsamples = 12\nvalue = [0,
6, 13]\nclass = c'),
 Text(1883.25, 22.649999999999977, 'gini = 0.0\nsamples = 84\nvalue = [0, 0,
144]\nclass = c'),
 Text(2232.0, 113.25, 'PXY <= 2.59\ngini = 0.569\nsamples = 91\nvalue = [81,
38, 21]\nclass = a'),
 Text(2092.5, 67.94999999999999, 'PM10 <= 48.125\ngini = 0.153\nsamples = 21
\nvalue = [3, 33, 0]\nclass = b'),
 Text(2022.75, 22.649999999999977, 'gini = 0.49\nsamples = 5\nvalue = [3, 4,
0]\nclass = b'),
 Text(2162.25, 22.649999999999977, 'gini = 0.0\nsamples = 16\nvalue = [0, 29,
0]\nclass = b'),
 Text(2371.5, 67.94999999999999, 'NOx <= 405.6\ngini = 0.394\nsamples = 70\nv
alue = [78, 5, 21]\nclass = a'),
 Text(2301.75, 22.649999999999977, 'gini = 0.265\nsamples = 60\nvalue = [74,
5, 8]\nclass = a'),
 Text(2441.25, 22.649999999999977, 'gini = 0.36\nsamples = 10\nvalue = [4, 0,
13]\nclass = c'),
 Text(2650.5, 158.55, 'OXY <= 9.865\ngini = 0.368\nsamples = 87\nvalue = [10
3, 0, 33]\nclass = a'),
 Text(2580.75, 113.25, 'EBE <= 5.075\ngini = 0.42\nsamples = 70\nvalue = [77,
0, 33]\nclass = a'),
 Text(2511.0, 67.94999999999999, 'gini = 0.0\nsamples = 14\nvalue = [0, 0, 3
1]\nclass = c'),
 Text(2650.5, 67.94999999999999, 'BEN <= 3.81\ngini = 0.049\nsamples = 56\nva
lue = [77, 0, 2]\nclass = a'),
 Text(2580.75, 22.649999999999977, 'gini = 0.346\nsamples = 5\nvalue = [7, 0,
2]\nclass = a'),
 Text(2720.25, 22.649999999999977, 'gini = 0.0\nsamples = 51\nvalue = [70, 0,
0]\nclass = a'),
 Text(2720.25, 113.25, 'gini = 0.0\nsamples = 17\nvalue = [26, 0, 0]\nclass =
a')]
```



Conclusion

Linear Regression =0.7548303061675042

Ridge Regression =0.7360079138814519

Lasso Regression =0.5155679352303824

ElasticNet Regression =0.6237301983555076

Logistic Regression =0.9716666666666667

Randomforest =0.9369047619047619

Logistic Regression is suitable for this dataset

In [ ]: