mk 3/08/2023

In [677]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [678]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_200
df
```

Out[678]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000	156.1
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000	80.8
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000	53.0
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000	105.3
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	106.5
...	...	...	...	...	...	...	...	...	...	...	
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999	6.7
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN	22.150000	5.7
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN	18.540001	13.0
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN	NaN	6.6
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000	10.2

225120 rows × 17 columns

In [679]:
```
df=df.dropna()
```

In [680]:
```
df=df.head(1100)
```

In [681]:
```
df.columns
```

Out[681]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [682]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1100 entries, 4 to 9931
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 [683]:
```
data=df[['BEN', 'TOL', 'PXY']]
data
```

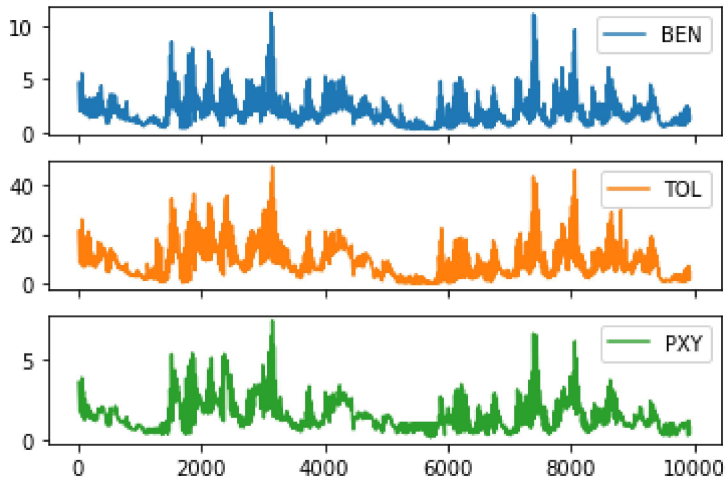
Out[683]:

	BEN	TOL	PXY
4	4.64	21.200001	3.56
21	1.98	8.440000	1.78
25	2.82	15.010000	2.67
30	4.65	21.330000	3.57
47	1.97	8.400000	1.59
...	...	...	...
9901	0.96	1.600000	0.26
9905	1.32	3.750000	0.68
9910	2.26	7.240000	1.18
9927	1.10	1.760000	0.32
9931	1.48	4.630000	0.75

1100 rows × 3 columns

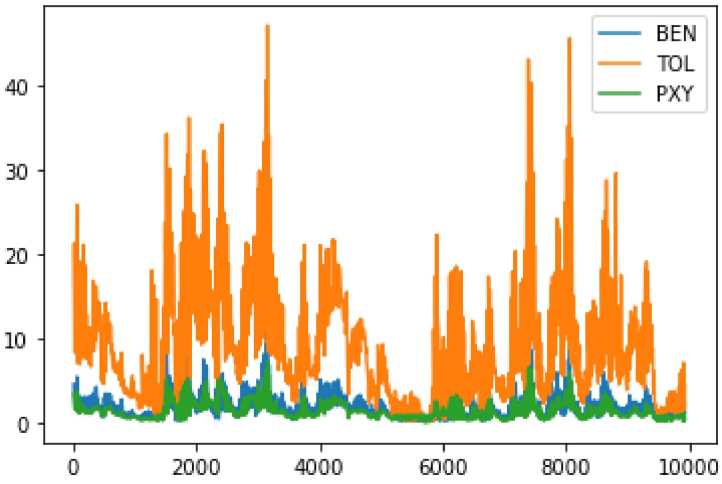
In [684]:
```
data.plot.line(subplots=True)
```

Out[684]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [685]:
```
data.plot.line()
```

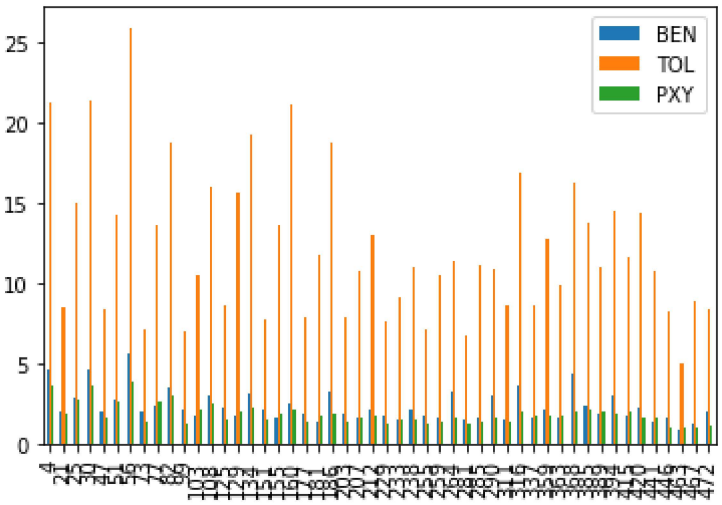
Out[685]: <AxesSubplot:>



In [686]:
```
b=data[0:50]
```

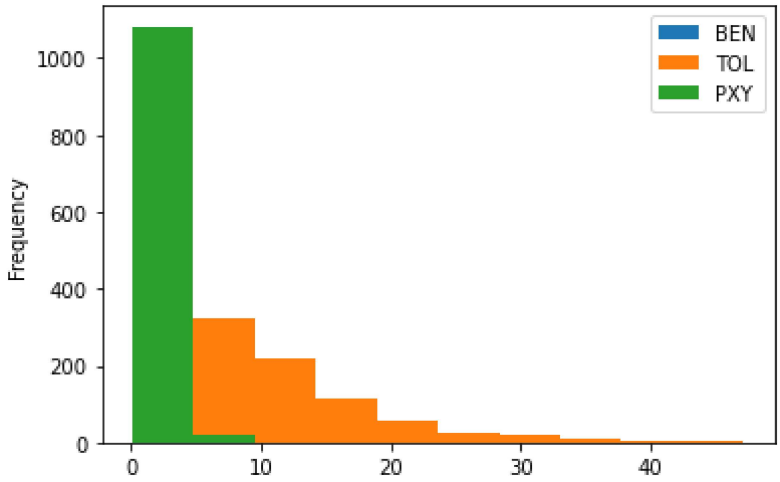
In [687]:
```
b.plot.bar()
```

Out[687]: <AxesSubplot:>



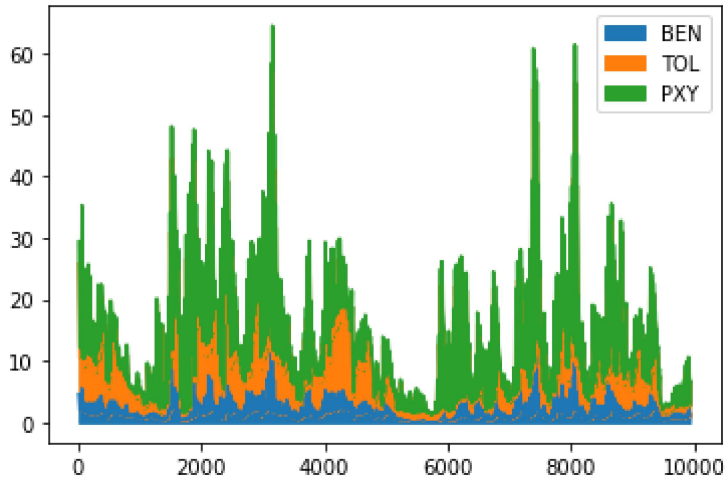
In [688]:
```
data.plot.hist()
```

Out[688]: <AxesSubplot:ylabel='Frequency'>



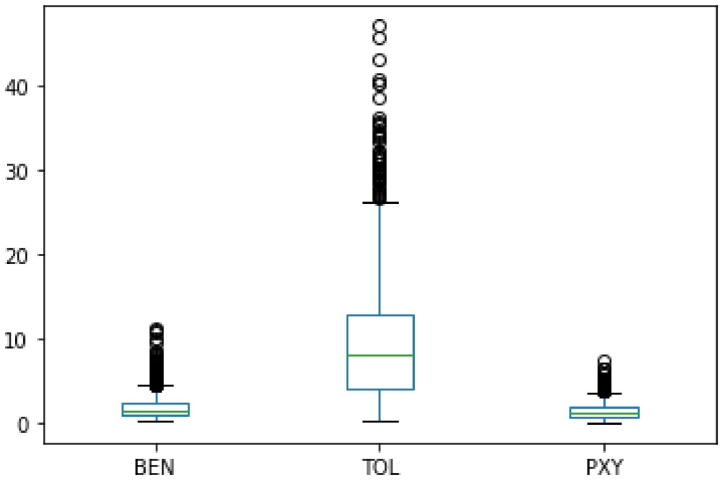
In [689]:
```
data.plot.area()
```

Out[689]: <AxesSubplot:>



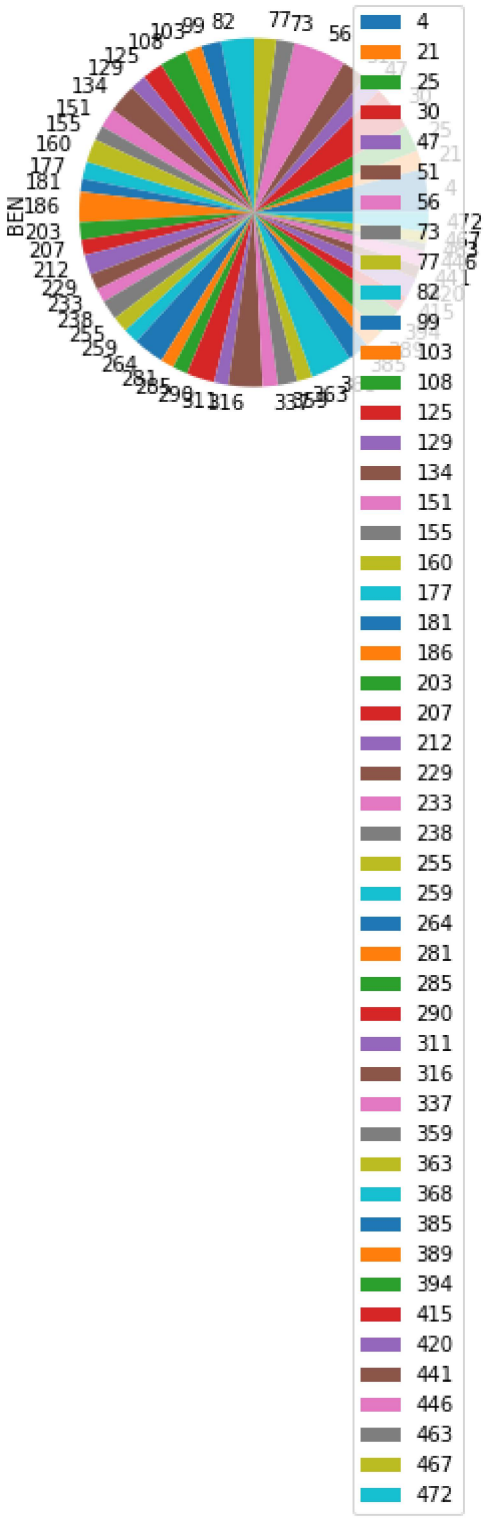
In [690]:
```
data.plot.box()
```

Out[690]: <AxesSubplot:>



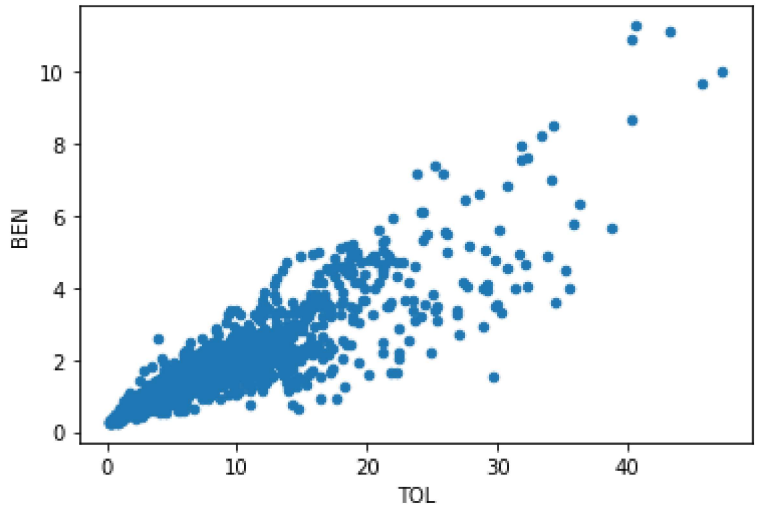
In [691]:
```
b.plot.pie(y='BEN' )
```

Out[691]: <AxesSubplot:ylabel='BEN'>



In [692]:
```
data.plot.scatter(x='TOL' ,y='BEN')
```

Out[692]:
```
<AxesSubplot:xlabel='TOL', ylabel='BEN'>
```



In [693]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1100 entries, 4 to 9931
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 [694]:
```
df.describe()
```

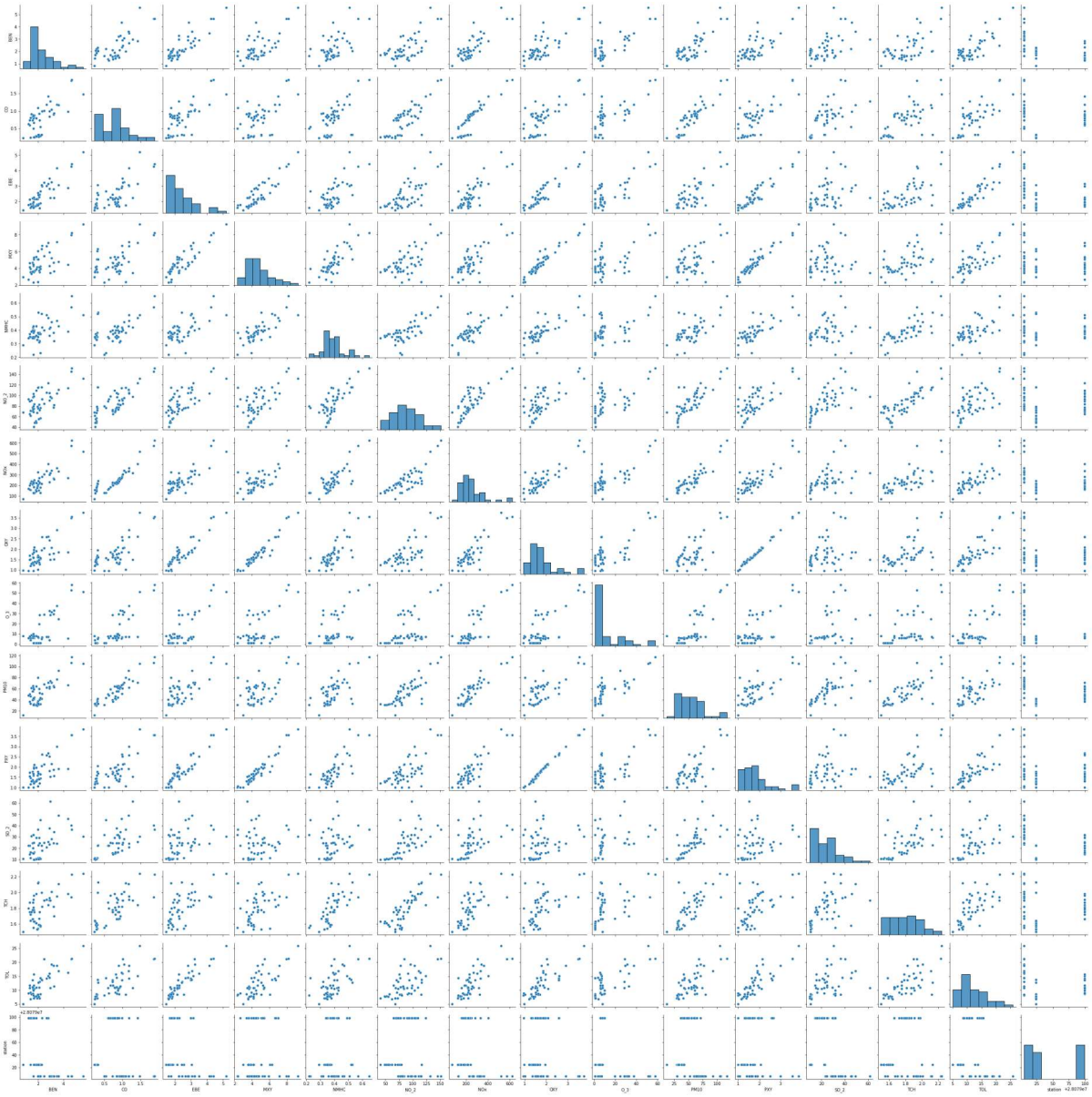
Out[694]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
count	1100.000000	1100.000000	1100.000000	1100.000000	1100.000000	1100.000000	1100.000000
mean	1.979718	0.686445	2.010536	3.834764	0.353191	79.928964	206.406473
std	1.448805	0.525607	1.413268	2.784414	0.132502	36.948851	147.403760
min	0.260000	0.150000	0.190000	0.270000	0.100000	6.620000	9.620000
25%	0.990000	0.290000	1.000000	1.770000	0.260000	54.072501	101.700003
50%	1.620000	0.520000	1.680000	3.280000	0.320000	76.154999	177.599998
75%	2.430000	0.932500	2.560000	5.305000	0.420000	103.549999	274.900009
max	11.280000	3.820000	9.210000	21.540001	1.220000	254.000000	1141.000000

In [695]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [696]:
```
sns.pairplot(df1[0:50])
```

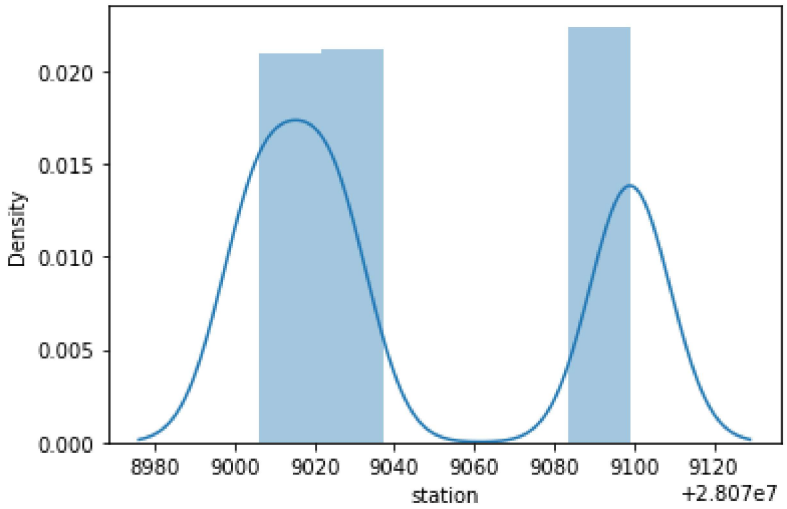
Out[696]: <seaborn.axisgrid.PairGrid at 0x191aad3d790>



In [697]:
```
sns.distplot(df1['station'])
```

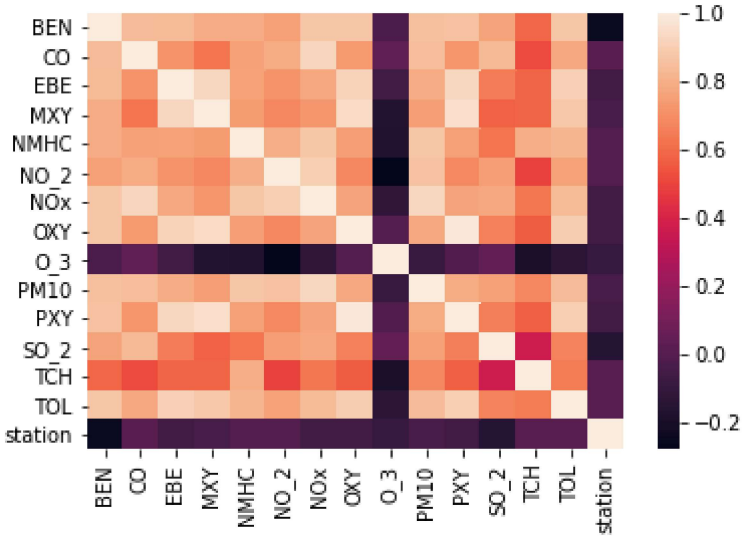
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[697]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [698]:
```
sns.heatmap(df1.corr())
```

Out[698]: <AxesSubplot:>



In [699]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [700]:
```
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 [701]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[701]:
```
LinearRegression()
```

In [702]:
```
lr.intercept_
```

Out[702]:
```
28079028.67812601
```

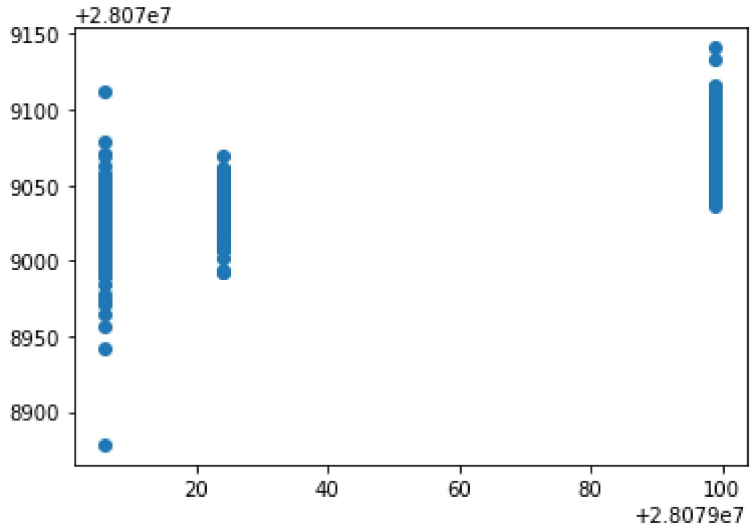
In [703]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[703]:

	Co-efficient
BEN	-41.375766
CO	109.199108
EBE	-5.774170
MXY	0.397242
NMHC	47.860998
NO_2	0.395781
NOx	-0.328913
OXY	31.269348
O_3	-0.203209
PM10	0.689245
PXY	-16.194702
SO_2	-1.333811
TCH	2.334178
TOL	2.810878

In [704]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[704]: <matplotlib.collections.PathCollection at 0x191b772e2b0>



In [705]:
```
lr.score(x_test,y_test)
```

Out[705]: 0.4858544821589984

In [706]:
```
lr.score(x_train,y_train)
```

Out[706]: 0.5121489707205813

In [707]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [708]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[708]: Ridge(alpha=10)

In [709]:
```
rr.score(x_test,y_test)
```

Out[709]: 0.4667273496830383

In [710]:
```
rr.score(x_train,y_train)
```

Out[710]: 0.4857018052262091

In [711]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[711]: Lasso(alpha=10)

In [712]:
```
la.score(x_train,y_train)
```

Out[712]:
```
0.09061057903831926
```

In [713]:
```
la.score(x_test,y_test)
```

Out[713]:
```
0.09081436055191594
```

In [714]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[714]:
```
ElasticNet()
```

In [715]:
```
en.coef_
```

Out[715]:
```
array([-13.09982565,   3.61685282,  -1.79280908,  -1.29204422,
         0.        ,   0.2485598 ,  -0.01844147,   0.        ,
         0.0631072 ,   0.09523907,   0.        ,  -0.71565223,
        -0.        ,   2.77213426])
```

In [716]:
```
en.intercept_
```

Out[716]:
```
28079043.459119305
```

In [717]:
```
prediction=en.predict(x_test)
```

In [718]:
```
en.score(x_test,y_test)
```

Out[718]:
```
0.24254622175843044
```

In [719]:
```
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)))
```

```
32.4511535066321
1272.77956873301
35.676036337197125
```

In [720]:
```
from sklearn.linear_model import LogisticRegression
```

In [721]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [722]:
```
feature_matrix.shape
```

Out[722]: (1100, 14)

In [723]:
```
target_vector.shape
```

Out[723]: (1100,)

In [724]:
```
from sklearn.preprocessing import StandardScaler
```

In [725]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [726]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[726]: LogisticRegression(max_iter=10000)

In [727]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [728]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [729]:
```
logr.score(fs,target_vector)
```

Out[729]: 0.9563636363636364

In [730]:
```
logr.predict_proba(observation)[0][0]
```

Out[730]: 3.0173663637141116e-06

In [731]:
```
logr.predict_proba(observation)
```

Out[731]: array([[3.01736636e-06, 7.18998734e-30, 9.99996983e-01]])

In [732]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [733]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[733]: RandomForestClassifier()

In [734]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]}
```

In [735]:
```
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[735]:
```
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 [736]:
```
grid_search.best_score_
```

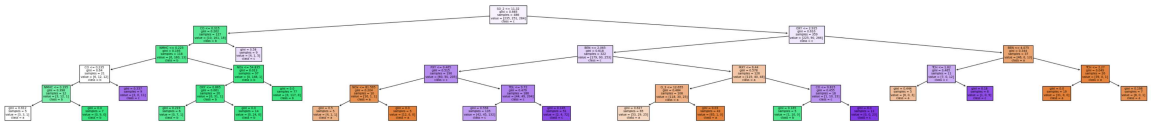
Out[736]:
```
0.9064935064935065
```

In [737]:
```
rfc_best=grid_search.best_estimator_
```

In [738]:
```
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[738]:
```
[Text(1237.5, 249.15, 'SO_2 <= 11.32\ngini = 0.665\nsamples = 486\nvalue = [2
35, 251, 284]\nclass = c'),
 Text(540.0, 203.85000000000002, 'CO <= 0.315\ngini = 0.262\nsamples = 127\nv
alue = [10, 161, 18]\nclass = b'),
 Text(450.0, 158.55, 'NMHC <= 0.225\ngini = 0.195\nsamples = 118\nvalue = [6,
160, 13]\nclass = b'),
 Text(270.0, 113.25, 'CO <= 0.235\ngini = 0.64\nsamples = 21\nvalue = [6, 12,
12]\nclass = b'),
 Text(180.0, 67.94999999999999, 'NMHC <= 0.195\ngini = 0.398\nsamples = 12\nv
alue = [3, 12, 1]\nclass = b'),
 Text(90.0, 22.649999999999977, 'gini = 0.612\nsamples = 5\nvalue = [3, 3, 1]
\nclass = a'),
 Text(270.0, 22.649999999999977, 'gini = 0.0\nsamples = 7\nvalue = [0, 9, 0]
\nclass = b'),
 Text(360.0, 67.94999999999999, 'gini = 0.337\nsamples = 9\nvalue = [3, 0, 1
1]\nclass = c'),
 Text(630.0, 113.25, 'NOx <= 54.835\ngini = 0.013\nsamples = 97\nvalue = [0,
148, 1]\nclass = b'),
 Text(540.0, 67.94999999999999, 'OXY <= 0.865\ngini = 0.061\nsamples = 20\nva
lue = [0, 31, 1]\nclass = b'),
 Text(450.0, 22.649999999999977, 'gini = 0.219\nsamples = 6\nvalue = [0, 7,
1]\nclass = b'),
 Text(630.0, 22.649999999999977, 'gini = 0.0\nsamples = 14\nvalue = [0, 24,
0]\nclass = b'),
 Text(720.0, 67.94999999999999, 'gini = 0.0\nsamples = 77\nvalue = [0, 117,
0]\nclass = b'),
 Text(630.0, 158.55, 'gini = 0.58\nsamples = 9\nvalue = [4, 1, 5]\nclass =
c'),
 Text(1935.0, 203.85000000000002, 'OXY <= 2.915\ngini = 0.616\nsamples = 359
\nvalue = [225, 90, 266]\nclass = c'),
 Text(1440.0, 158.55, 'BEN <= 2.065\ngini = 0.618\nsamples = 322\nvalue = [17
9, 90, 253]\nclass = c'),
 Text(1080.0, 113.25, 'PXY <= 0.465\ngini = 0.515\nsamples = 196\nvalue = [6
0, 50, 205]\nclass = c'),
 Text(900.0, 67.94999999999999, 'NOx <= 81.595\ngini = 0.204\nsamples = 10\nv
alue = [16, 1, 1]\nclass = a'),
 Text(810.0, 22.649999999999977, 'gini = 0.5\nsamples = 5\nvalue = [4, 1, 1]
\nclass = a'),
 Text(990.0, 22.649999999999977, 'gini = 0.0\nsamples = 5\nvalue = [12, 0, 0]
\nclass = a'),
 Text(1260.0, 67.94999999999999, 'TOL <= 9.72\ngini = 0.479\nsamples = 186\nv
alue = [44, 49, 204]\nclass = c'),
 Text(1170.0, 22.649999999999977, 'gini = 0.558\nsamples = 135\nvalue = [42,
45, 132]\nclass = c'),
 Text(1350.0, 22.649999999999977, 'gini = 0.145\nsamples = 51\nvalue = [2, 4,
72]\nclass = c'),
 Text(1800.0, 113.25, 'MXY <= 6.44\ngini = 0.578\nsamples = 126\nvalue = [11
9, 40, 48]\nclass = a'),
 Text(1620.0, 67.94999999999999, 'O_3 <= 12.655\ngini = 0.484\nsamples = 108
\nvalue = [118, 30, 25]\nclass = a'),
 Text(1530.0, 22.649999999999977, 'gini = 0.627\nsamples = 65\nvalue = [53, 2
9, 25]\nclass = a'),
 Text(1710.0, 22.649999999999977, 'gini = 0.03\nsamples = 43\nvalue = [65, 1,
0]\nclass = a'),
 Text(1980.0, 67.94999999999999, 'CO <= 0.815\ngini = 0.455\nsamples = 18\nva
lue = [1, 10, 23]\nclass = c'),
 Text(1890.0, 22.649999999999977, 'gini = 0.165\nsamples = 5\nvalue = [1, 10,
```

```
0]\nclass = b'),
 Text(2070.0, 22.649999999999977, 'gini = 0.0\nsamples = 13\nvalue = [0, 0, 2
3]\nclass = c'),
 Text(2430.0, 158.55, 'BEN <= 4.075\ngini = 0.344\nsamples = 37\nvalue = [46,
0, 13]\nclass = a'),
 Text(2250.0, 113.25, 'TCH <= 1.82\ngini = 0.465\nsamples = 11\nvalue = [7,
0, 12]\nclass = c'),
 Text(2160.0, 67.94999999999999, 'gini = 0.444\nsamples = 5\nvalue = [6, 0,
3]\nclass = a'),
 Text(2340.0, 67.94999999999999, 'gini = 0.18\nsamples = 6\nvalue = [1, 0, 9]
\nclass = c'),
 Text(2610.0, 113.25, 'TCH <= 2.27\ngini = 0.049\nsamples = 26\nvalue = [39,
0, 1]\nclass = a'),
 Text(2520.0, 67.94999999999999, 'gini = 0.0\nsamples = 19\nvalue = [31, 0,
0]\nclass = a'),
 Text(2700.0, 67.94999999999999, 'gini = 0.198\nsamples = 7\nvalue = [8, 0,
1]\nclass = a')]
```



Conclusion

Linear Regression =0.5121489707205813

Ridge Regression =0.4857018052262091

Lasso Regression =0.5121489707205813

ElasticNet Regression =0.24254622175843044

Logistic Regression =0.9563636363636364

Randomforest =0.9064935064935065

Logistic Regression is suitable for this dataset

In [ ]: