mk 3/08/2023

In [494]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [495]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_200
df
```

Out[495]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080
...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389

245496 rows × 17 columns

In [496]:
```
df=df.dropna()
```

In [556]:
```
df=df.head(1000)
```

In [557]:
```
df.columns
```

Out[557]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [558]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20 entries, 5 to 184
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     20 non-null     object 
 1   BEN      20 non-null     float64
 2   CO       20 non-null     float64
 3   EBE      20 non-null     float64
 4   MXY      20 non-null     float64
 5   NMHC     20 non-null     float64
 6   NO_2     20 non-null     float64
 7   NOx      20 non-null     float64
 8   OXY      20 non-null     float64
 9   O_3      20 non-null     float64
 10  PM10     20 non-null     float64
 11  PM25     20 non-null     float64
 12  PXY      20 non-null     float64
 13  SO_2     20 non-null     float64
 14  TCH      20 non-null     float64
 15  TOL      20 non-null     float64
 16  station  20 non-null     int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.8+ KB
```

In [559]:

```
data=df[['BEN', 'TOL', 'PXY']]
data
```

Out[559]:

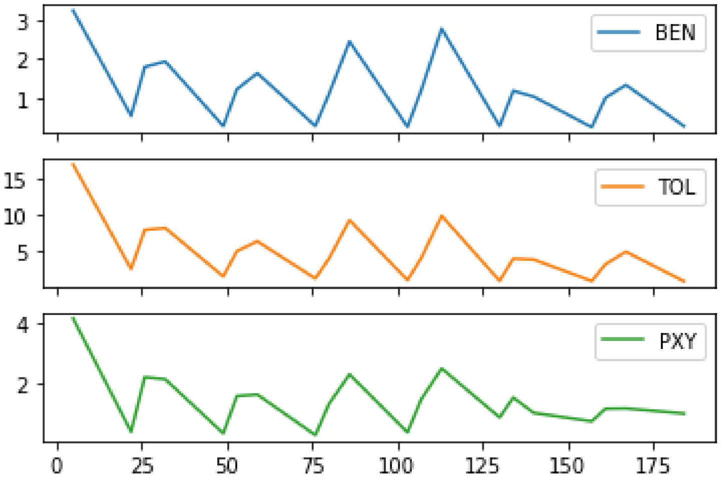
	BEN	TOL	PXY
5	3.24	16.93	4.16
22	0.55	2.53	0.39
26	1.80	7.92	2.21
32	1.94	8.18	2.14
49	0.29	1.52	0.34
53	1.23	5.00	1.58
59	1.64	6.35	1.63
76	0.29	1.23	0.28
80	1.11	3.98	1.32
86	2.45	9.26	2.30
103	0.27	1.01	0.37
107	1.20	4.03	1.47
113	2.78	9.86	2.50
130	0.29	0.91	0.87
134	1.19	3.97	1.53
140	1.04	3.84	1.02
157	0.26	0.88	0.74
161	1.01	3.19	1.16
167	1.34	4.92	1.17
184	0.29	0.86	1.00

In [560]:

```
data.plot.line(subplots=True)
```

Out[560]:

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

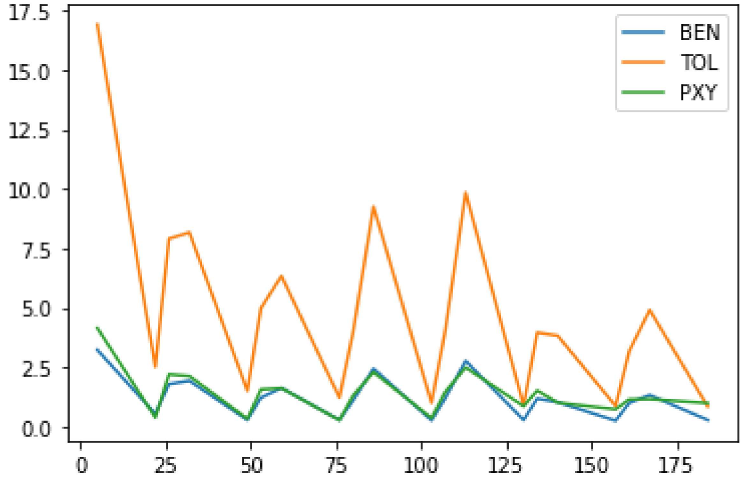


In [561]:

```
data.plot.line()
```

Out[561]:

```
<AxesSubplot:>
```



In [562]:

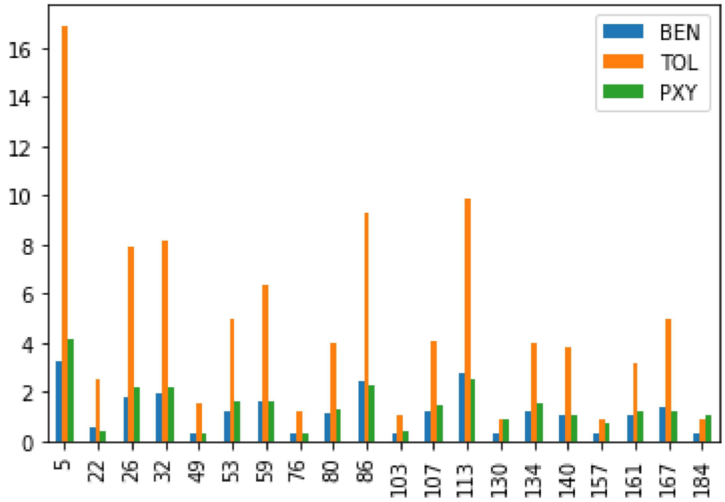
```
b=data[0:50]
```

In [563]:

```
b.plot.bar()
```

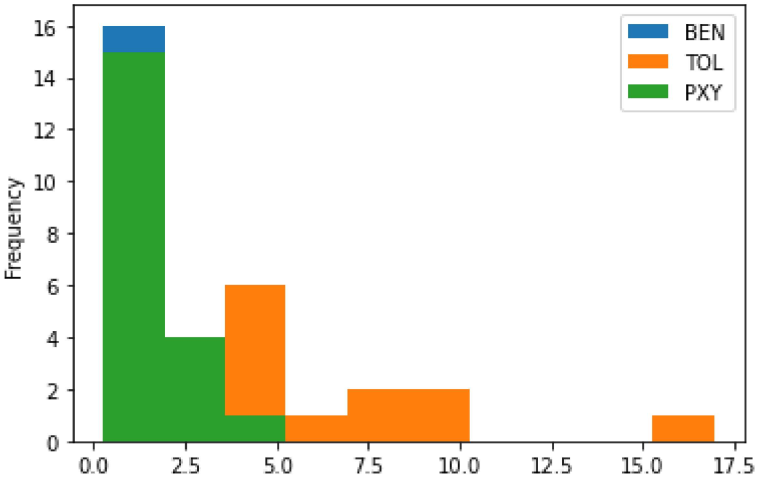
Out[563]:

```
<AxesSubplot:>
```



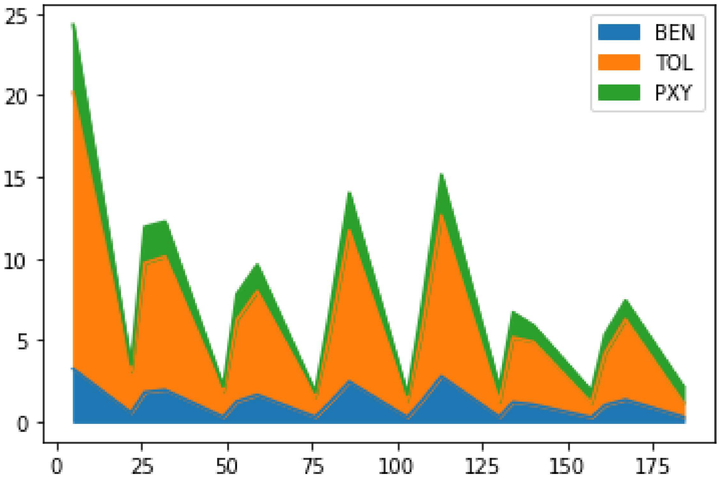
In [564]:
```
data.plot.hist()
```

Out[564]:
```
<AxesSubplot:ylabel='Frequency'>
```



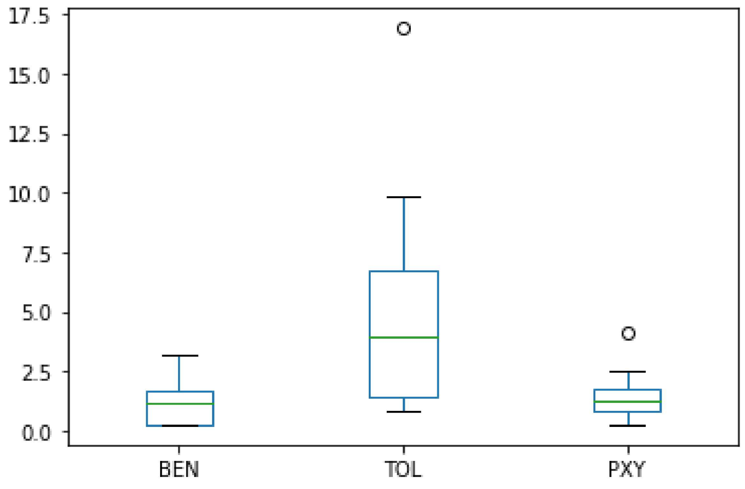
In [565]:
```

data.plot.area()
```

Out[565]:
```
<AxesSubplot:>
```



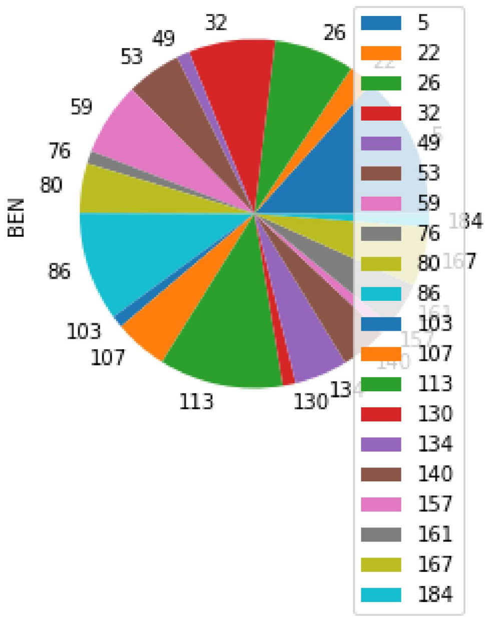
```
In [566]: data.plot.box()
```

```
Out[566]: <AxesSubplot:>
```



```
In [567]: b.plot.pie(y='BEN' )
```

```
Out[567]: <AxesSubplot:ylabel='BEN'>
```

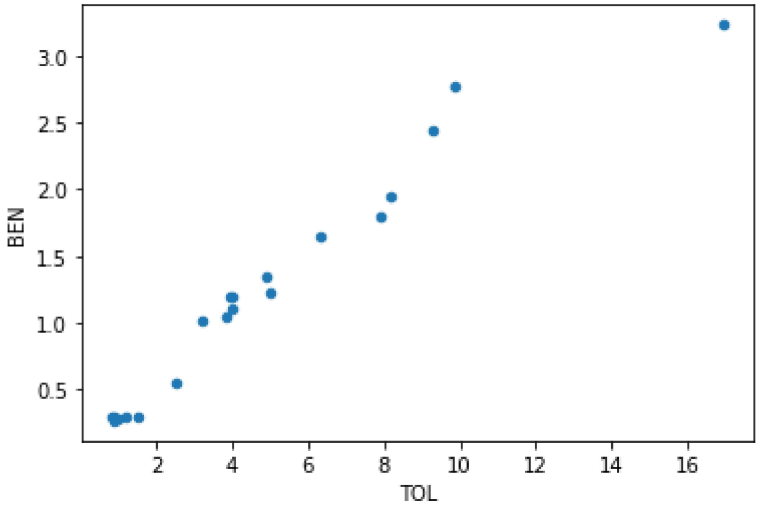


In [568]:

```
data.plot.scatter(x='TOL' ,y='BEN')
```

Out[568]:

```
<AxesSubplot:xlabel='TOL', ylabel='BEN'>
```



In [569]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20 entries, 5 to 184
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     20 non-null     object 
 1   BEN      20 non-null     float64
 2   CO       20 non-null     float64
 3   EBE      20 non-null     float64
 4   MXY      20 non-null     float64
 5   NMHC     20 non-null     float64
 6   NO_2     20 non-null     float64
 7   NOx      20 non-null     float64
 8   OXY      20 non-null     float64
 9   O_3      20 non-null     float64
 10  PM10     20 non-null     float64
 11  PM25     20 non-null     float64
 12  PXY      20 non-null     float64
 13  SO_2     20 non-null     float64
 14  TCH      20 non-null     float64
 15  TOL      20 non-null     float64
 16  station  20 non-null     int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.8+ KB
```

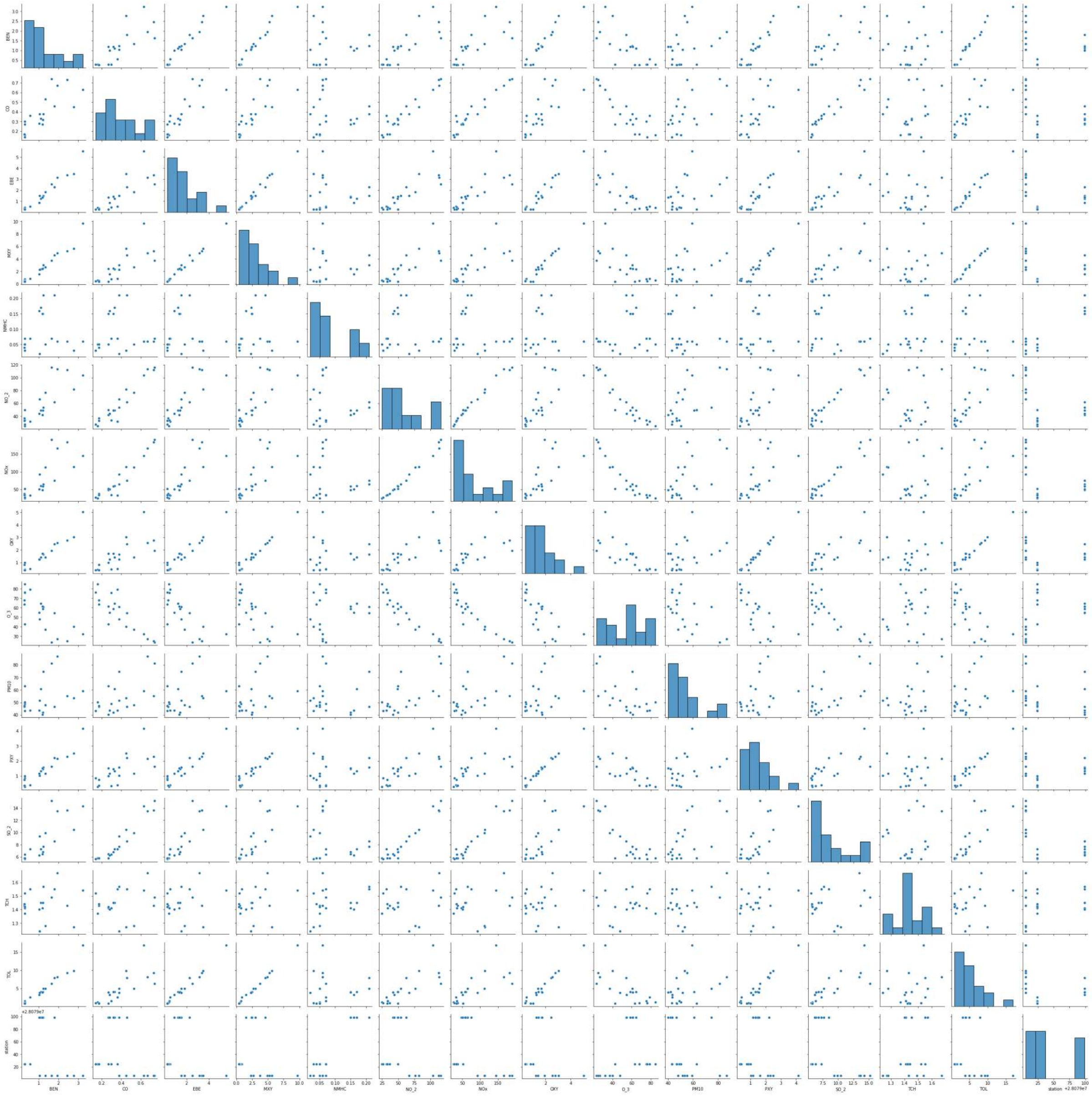
In [570]:
```
df.describe()
```

Out[570]:
```
             BEN        CO        EBE        MXY      NMHC        NO_2        NOx        OXY
count  20.000000  20.00000  20.000000  20.000000  20.00000   20.000000   20.000000  20.000000  2
 mean   1.210500   0.38700   1.635000   2.736500   0.08800   60.458500   80.153500   1.611500  5
  std   0.885357   0.18818   1.405394   2.382576   0.06161   30.317645   53.404650   1.152641  1
  min   0.260000   0.14000   0.270000   0.390000   0.02000   24.870001   25.530001   0.380000  2
  25%   0.290000   0.27000   0.452500   0.722500   0.05000   36.297500   37.044999   0.732500  3
  50%   1.150000   0.34500   1.390000   2.390000   0.06000   49.594999   58.955000   1.420000  5
  75%   1.680000   0.47750   2.345000   3.975000   0.15000   78.012499  112.849998   2.072500  6
  max   3.240000   0.74000   5.550000   9.720000   0.21000  115.800003  189.899994   5.040000  8
```

In [571]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [572]:
```
sns.pairplot(df1[0:50])
```

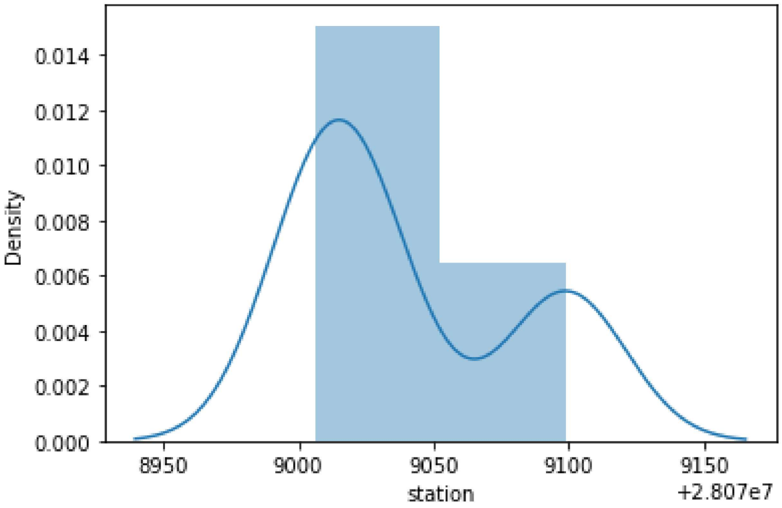
Out[572]:
```
<seaborn.axisgrid.PairGrid at 0x1918e072eb0>
```



```
In [573]: sns.distplot(df1['station'])
```

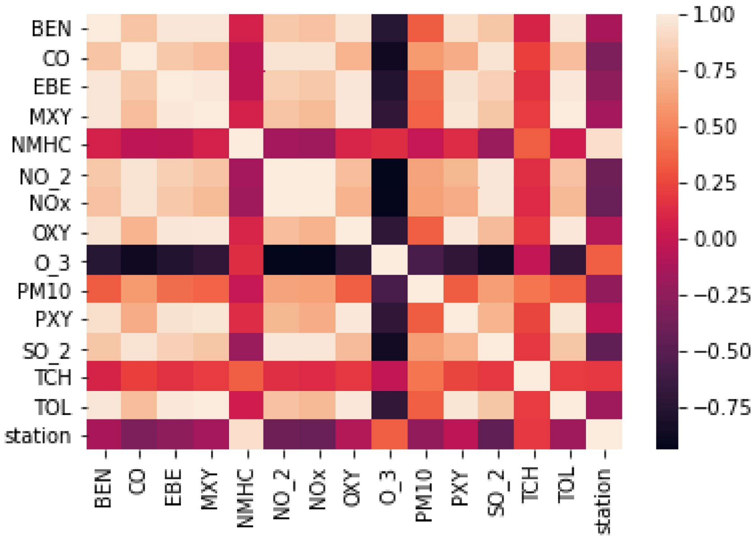
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

```
Out[573]: <AxesSubplot:xlabel='station', ylabel='Density'>
```



```
In [574]: sns.heatmap(df1.corr())
```

```
Out[574]: <AxesSubplot:>
```



```
In [575]: x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
           'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
          y=df['station']
```

In [576]:

```
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 [577]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[577]:

```
LinearRegression()
```

In [578]:

```
lr.intercept_
```

Out[578]:

```
28078611.61393362
```

In [579]:

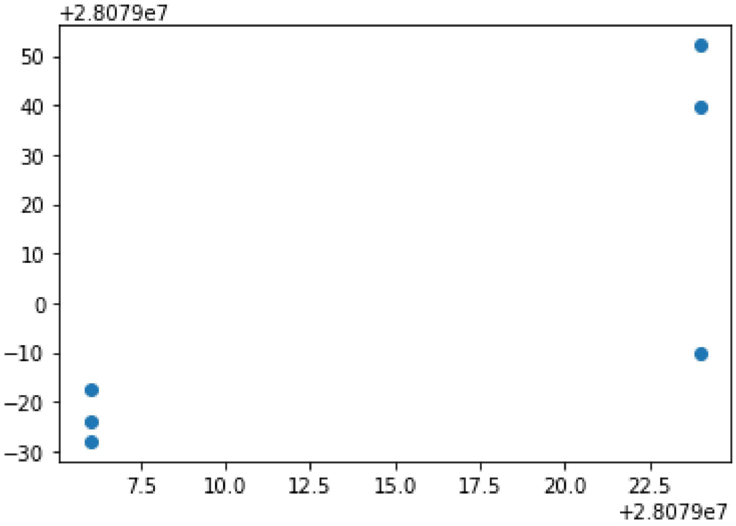
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[579]:

	Co-efficient
BEN	46.289259
CO	-222.315735
EBE	42.375662
MXY	-84.578916
NMHC	799.168387
NO_2	3.605769
NOx	-1.045082
OXY	58.701244
O_3	2.973096
PM10	-0.831940
PXY	0.828973
SO_2	13.869036
TCH	44.506946
TOL	3.758407

In [580]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[580]:
```
<matplotlib.collections.PathCollection at 0x19198e7bca0>
```



In [581]:
```
lr.score(x_test,y_test)
```

Out[581]:
```
-8.913696645528033
```

In [582]:
```
lr.score(x_train,y_train)
```

Out[582]:
```
1.0
```

In [583]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [584]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[584]:
```
Ridge(alpha=10)
```

In [585]:
```
rr.score(x_test,y_test)
```

Out[585]:
```
-25.680273390727546
```

In [586]:
```
rr.score(x_train,y_train)
```

Out[586]:
```
0.4757690329552654
```

In [587]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[587]:
```
Lasso(alpha=10)
```

In [588]:
```
la.score(x_train,y_train)
```

Out[588]:
```
0.397166976827539
```

In [589]:
```
la.score(x_test,y_test)
```

Out[589]:
```
-23.39084298600196
```

In [590]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[590]:
```
ElasticNet()
```

In [591]:
```
en.coef_
```

Out[591]:
```
array([ 0.35276679,  0.59873896, -0.7033591 ,  5.17705657,  2.24177463,
       -2.27215567, -0.13267477,  3.76843052, -1.97925011,  0.99602759,
        3.50473413, -3.50906271,  0.42418908,  5.22961332])
```

In [592]:
```
en.intercept_
```

Out[592]:
```
28079231.929840643
```

In [593]:
```
prediction=en.predict(x_test)
```

In [594]:
```
en.score(x_test,y_test)
```

Out[594]:
```
-25.999562917054273
```

In [595]:
```
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)))
```

```
39.17938381433487
2186.964596281396
46.7649932778932
```

In [596]:
```
from sklearn.linear_model import LogisticRegression
```

In [597]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [598]:
```
feature_matrix.shape
```

Out[598]:
```
(20, 14)
```

In [599]:
```
target_vector.shape
```

Out[599]:
```
(20,)
```

In [600]:
```
from sklearn.preprocessing import StandardScaler
```

In [601]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [602]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[602]:
```
LogisticRegression(max_iter=10000)
```

In [603]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [604]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [605]:
```
logr.score(fs,target_vector)
```

Out[605]:
```
1.0
```

In [606]:
```
logr.predict_proba(observation)[0][0]
```

Out[606]:
```
0.23544436221308845
```

In [607]:
```
logr.predict_proba(observation)
```

Out[607]:
```
array([[2.35444362e-01, 1.79916389e-11, 7.64555638e-01]])
```

In [608]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [609]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[609]:
```
RandomForestClassifier()
```

In [610]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]}
```

In [611]:
```
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[611]:
```
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')
```

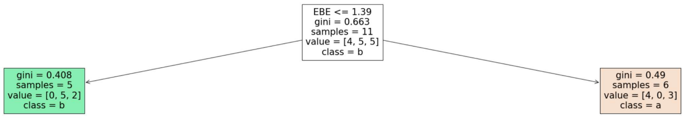
In [612]:
```
grid_search.best_score_
```

Out[612]:
```
0.42857142857142855
```

In [613]:
```
rfc_best=grid_search.best_estimator_
```

In [614]:
```
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[614]:
```
[Text(1395.0, 203.85000000000002, 'EBE <= 1.39\ngini = 0.663\nsamples = 11\nv
alue = [4, 5, 5]\nclass = b'),
 Text(697.5, 67.94999999999999, 'gini = 0.408\nsamples = 5\nvalue = [0, 5, 2]
\nclass = b'),
 Text(2092.5, 67.94999999999999, 'gini = 0.49\nsamples = 6\nvalue = [4, 0, 3]
\nclass = a')]
```



Conclusion

Linear Regression =1.0

Ridge Regression =0.8505902210559991

Lasso Regression =0.7818422293632583

ElasticNet Regression =0.6639225233261004

Logistic Regression =0.8991524766814838

Randomforest =0.3571428571428571

Logistic Regression is suitable for this dataset

In [ ]: