In [19]:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as pp
```

In [20]:

```
df1 = pd.read_csv(r"C:\Users\user\Desktop\c10\madrid_2001.csv")
df = df1.head(1000)
df
```

Out[20]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2001-08-01 01:00:00 | NaN | 0.37 | NaN | NaN | NaN | 58.400002 | 87.150002 | NaN | 34.529999 | 105.000 |
| **1** | 2001-08-01 01:00:00 | 1.50 | 0.34 | 1.49 | 4.1 | 0.07 | 56.250000 | 75.169998 | 2.11 | 42.160000 | 100.599 |
| **2** | 2001-08-01 01:00:00 | NaN | 0.28 | NaN | NaN | NaN | 50.660000 | 61.380001 | NaN | 46.310001 | 100.099 |
| **3** | 2001-08-01 01:00:00 | NaN | 0.47 | NaN | NaN | NaN | 69.790001 | 73.449997 | NaN | 40.650002 | 69.779 |
| **4** | 2001-08-01 01:00:00 | NaN | 0.39 | NaN | NaN | NaN | 22.830000 | 24.799999 | NaN | 66.309998 | 75.180 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **995** | 2001-08-02 18:00:00 | NaN | 0.09 | NaN | NaN | 0.09 | 27.670000 | 33.189999 | NaN | 93.559998 | 30.309 |
| **996** | 2001-08-02 18:00:00 | NaN | 0.41 | NaN | NaN | NaN | 45.639999 | 62.180000 | NaN | 86.820000 | 44.279 |
| **997** | 2001-08-02 18:00:00 | 1.28 | 0.35 | 1.68 | NaN | 0.10 | 51.560001 | 84.430000 | NaN | 56.520000 | 50.509 |
| **998** | 2001-08-02 18:00:00 | NaN | 0.11 | NaN | NaN | NaN | 33.270000 | 42.939999 | NaN | 93.910004 | 25.760 |
| **999** | 2001-08-02 18:00:00 | NaN | 0.05 | NaN | NaN | 0.04 | 11.520000 | 12.950000 | NaN | 83.019997 | 35.660 |

1000 rows × 16 columns

In [21]:

```
df=df.dropna()
```

In [22]:

```
df.columns
```

Out[22]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [23]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 166 entries, 1 to 989
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     166 non-null    object 
 1   BEN      166 non-null    float64
 2   CO       166 non-null    float64
 3   EBE      166 non-null    float64
 4   MXY      166 non-null    float64
 5   NMHC     166 non-null    float64
 6   NO_2     166 non-null    float64
 7   NOx      166 non-null    float64
 8   OXY      166 non-null    float64
 9   O_3      166 non-null    float64
 10  PM10     166 non-null    float64
 11  PXY      166 non-null    float64
 12  SO_2     166 non-null    float64
 13  TCH      166 non-null    float64
 14  TOL      166 non-null    float64
 15  station  166 non-null    int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 22.0+ KB
```

In [24]:

```
data=df[['CO' ,'station']]
data
```

Out[24]:

| | CO | station |
|---|---|---|
| **1** | 0.34 | 28079035 |
| **5** | 0.63 | 28079006 |
| **21** | 0.43 | 28079024 |
| **23** | 0.34 | 28079099 |
| **25** | 0.06 | 28079035 |
| **...** | ... | ... |
| **965** | 0.77 | 28079006 |
| **981** | 0.23 | 28079024 |
| **983** | 0.35 | 28079099 |
| **985** | 0.61 | 28079035 |
| **989** | 0.85 | 28079006 |

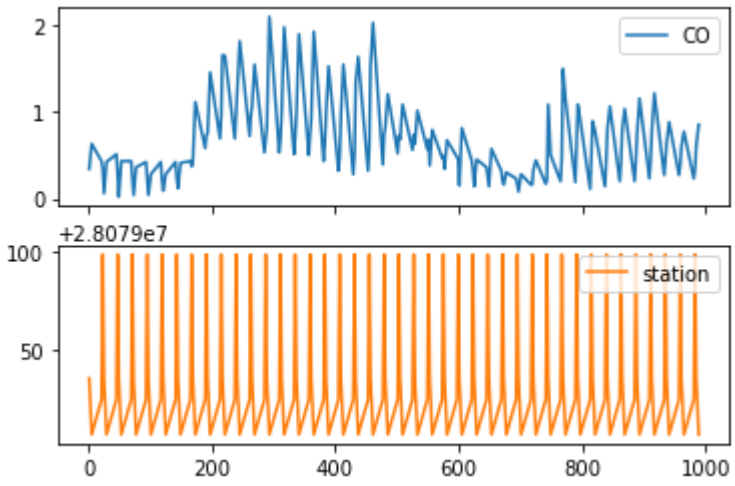
166 rows × 2 columns

In [25]:

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

Out[25]:

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

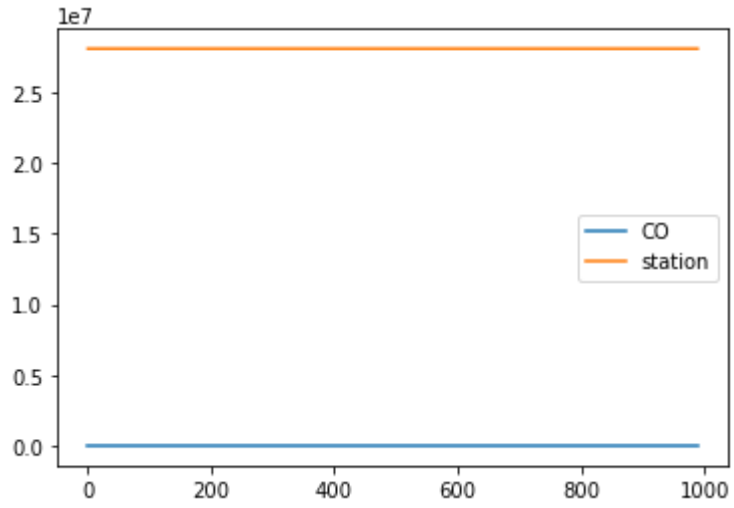


In [26]:

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

Out[26]:

<AxesSubplot:>




In [27]:

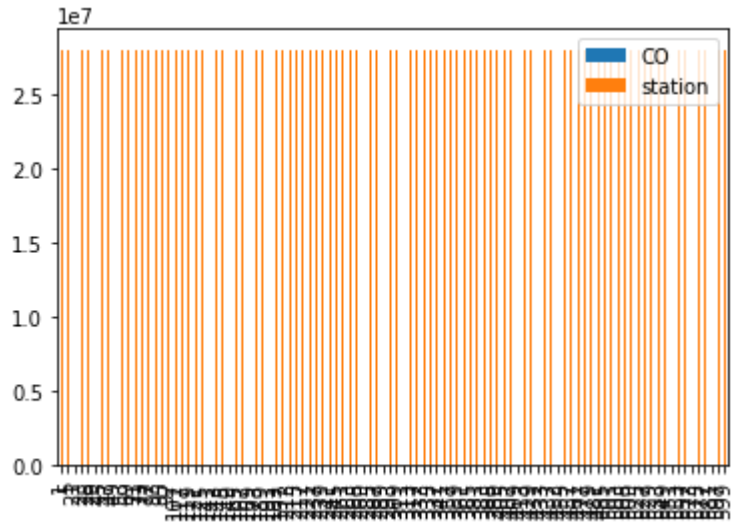
```
x = data[0:100]
```

In [28]:

```
x.plot.bar()
```

Out[28]:

<AxesSubplot:>

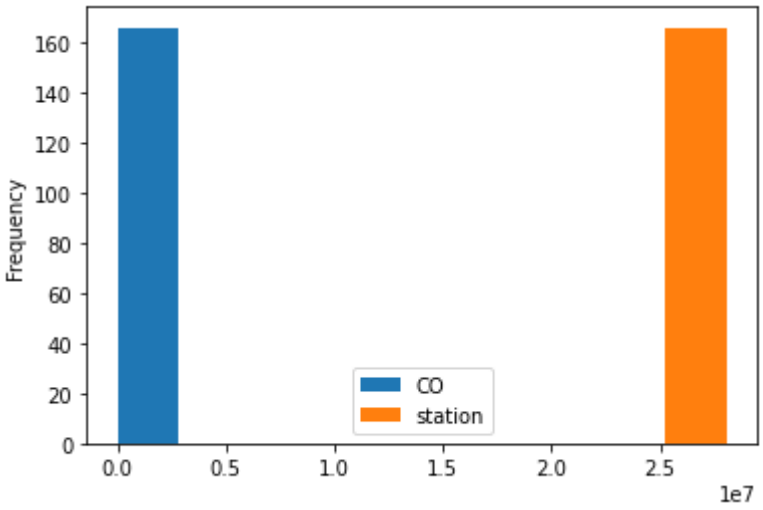


In [29]:

```
data.plot.hist()
```

Out[29]:

```
<AxesSubplot:ylabel='Frequency'>
```

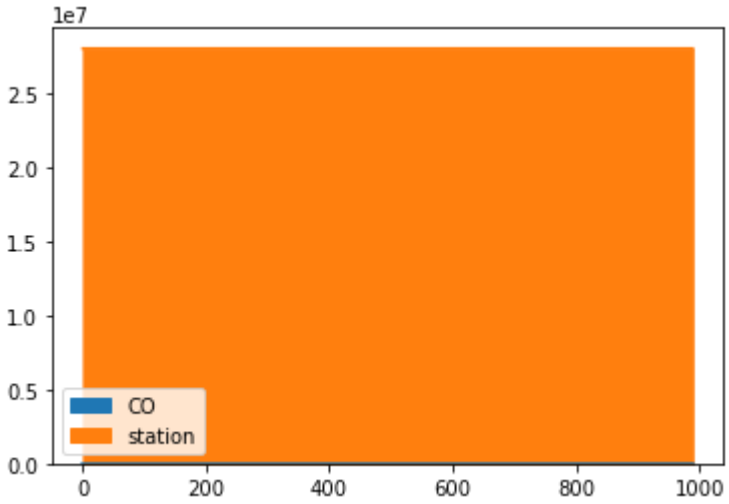



In [30]:

```
data.plot.area()
```

Out[30]:

```
<AxesSubplot:>
```

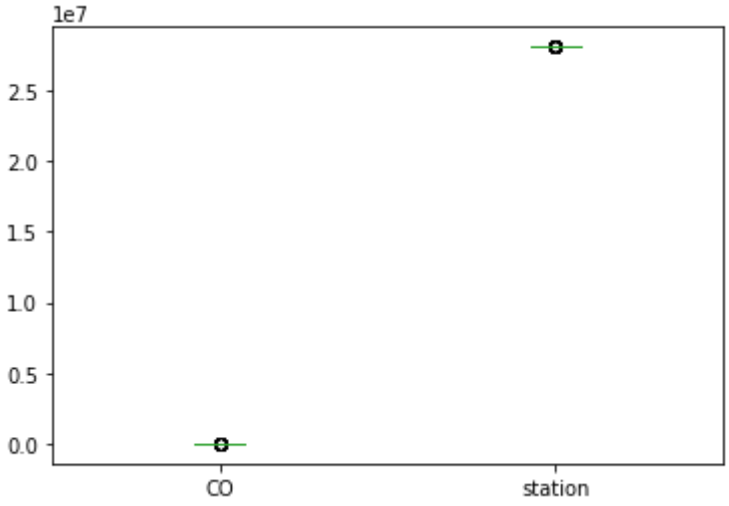


In [31]:

```
data.plot.box()
```

Out[31]:

<AxesSubplot:>



In [32]:

```
b.plot.pie(y='station' )
```

Out[32]:

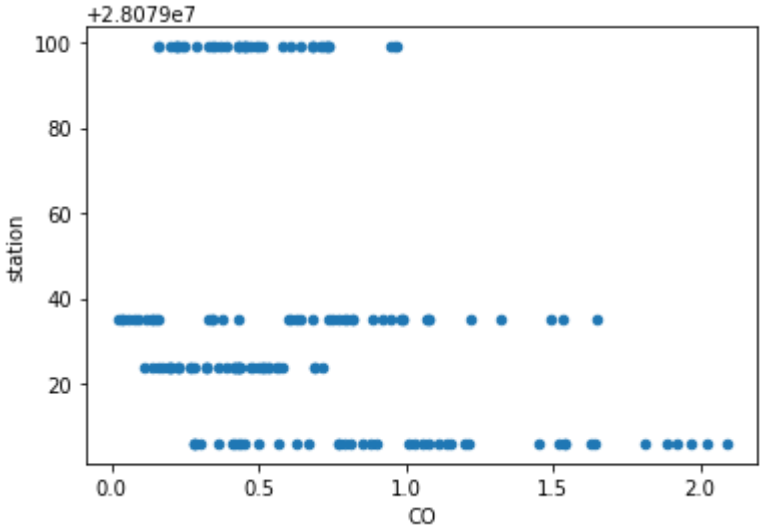
```
<AxesSubplot:ylabel='station'>
```

In [33]:

```
data.plot.scatter(x='CO' ,y='station')
```

Out[33]:

```
<AxesSubplot:xlabel='CO', ylabel='station'>
```




In [34]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 166 entries, 1 to 989
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     166 non-null    object 
 1   BEN      166 non-null    float64
 2   CO       166 non-null    float64
 3   EBE      166 non-null    float64
 4   MXY      166 non-null    float64
 5   NMHC     166 non-null    float64
 6   NO_2     166 non-null    float64
 7   NOx      166 non-null    float64
 8   OXY      166 non-null    float64
 9   O_3      166 non-null    float64
 10  PM10     166 non-null    float64
 11  PXY      166 non-null    float64
 12  SO_2     166 non-null    float64
 13  TCH      166 non-null    float64
```

In [35]:

```
df.describe()
```

Out[35]:

| | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx |
|---|---|---|---|---|---|---|---|
| **count** | 166.000000 | 166.000000 | 166.000000 | 166.000000 | 166.000000 | 166.000000 | 166.000000 |
| **mean** | 2.179639 | 0.641566 | 2.435542 | 5.874518 | 0.134398 | 61.698193 | 109.326385 |
| **std** | 1.697713 | 0.449958 | 1.876533 | 5.129837 | 0.099364 | 35.642849 | 87.084126 |
| **min** | 0.430000 | 0.020000 | 0.250000 | 0.530000 | 0.000000 | 4.190000 | 7.190000 |
| **25%** | 0.782500 | 0.322500 | 0.940000 | 1.985000 | 0.060000 | 32.182500 | 35.682501 |
| **50%** | 1.645000 | 0.510000 | 2.020000 | 4.740000 | 0.120000 | 61.730000 | 91.309998 |
| **75%** | 3.080000 | 0.820000 | 3.262500 | 8.185000 | 0.180000 | 90.977503 | 159.450005 |
| **max** | 7.840000 | 2.090000 | 8.900000 | 24.180000 | 0.510000 | 138.600006 | 389.299988 |

In [36]:

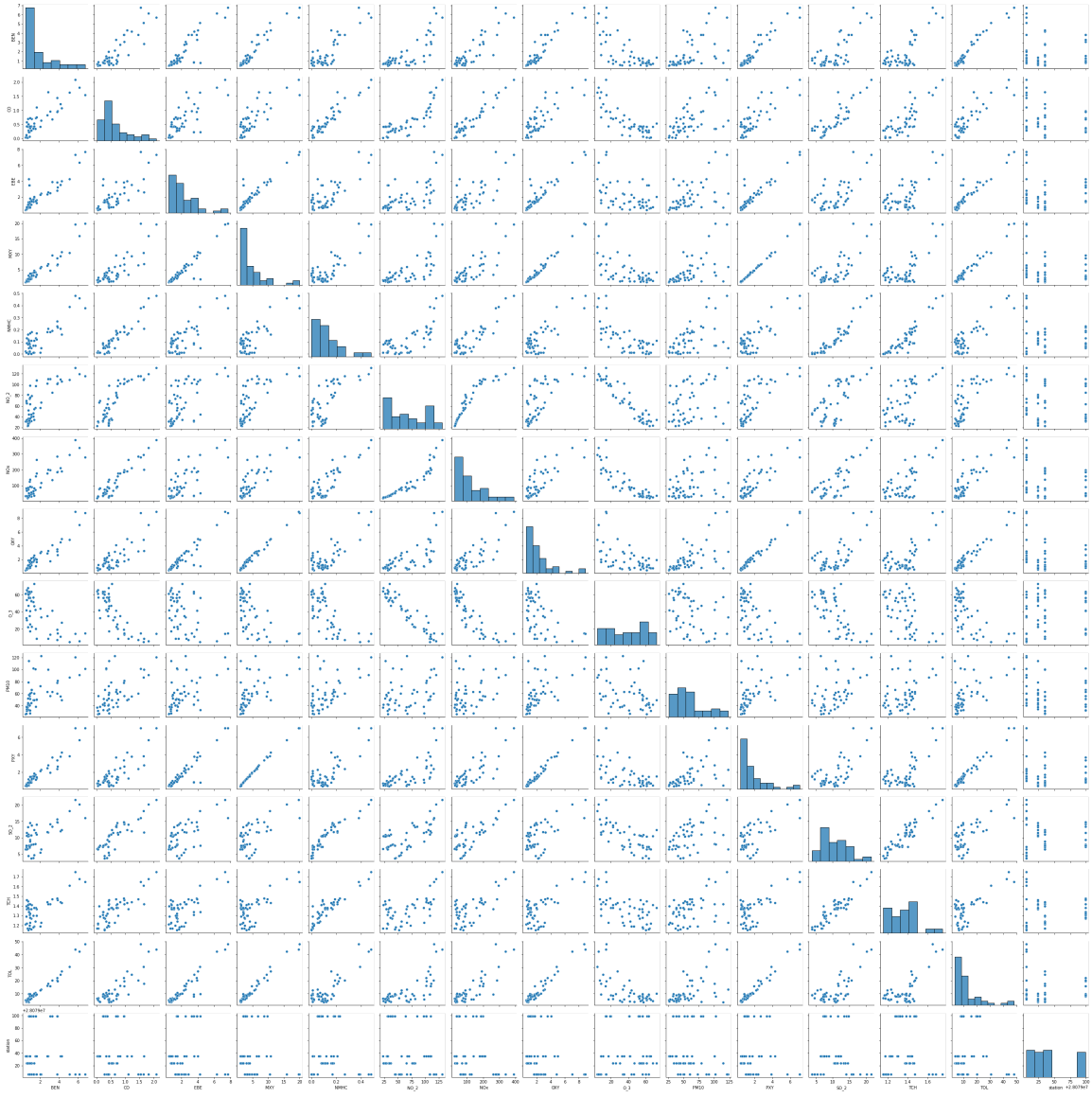
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [37]:

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

Out[37]:

```
<seaborn.axisgrid.PairGrid at 0x176f05bff40>
```



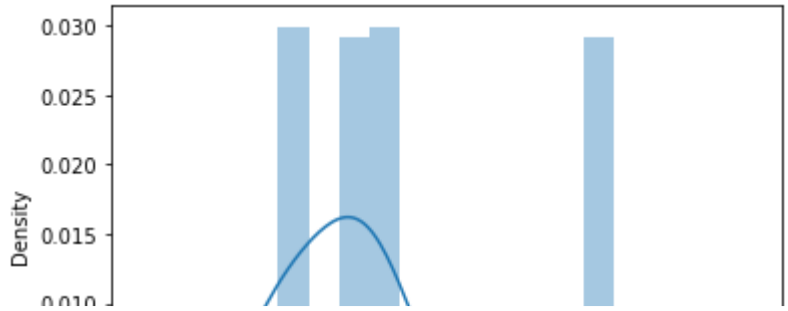
In [38]:

```
sns.distplot(df1['station'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:255
7: FutureWarning: `distplot` is a deprecated function and will be remove
d in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-l
evel function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[38]:

```
<AxesSubplot:xlabel='station', ylabel='Density'>
```

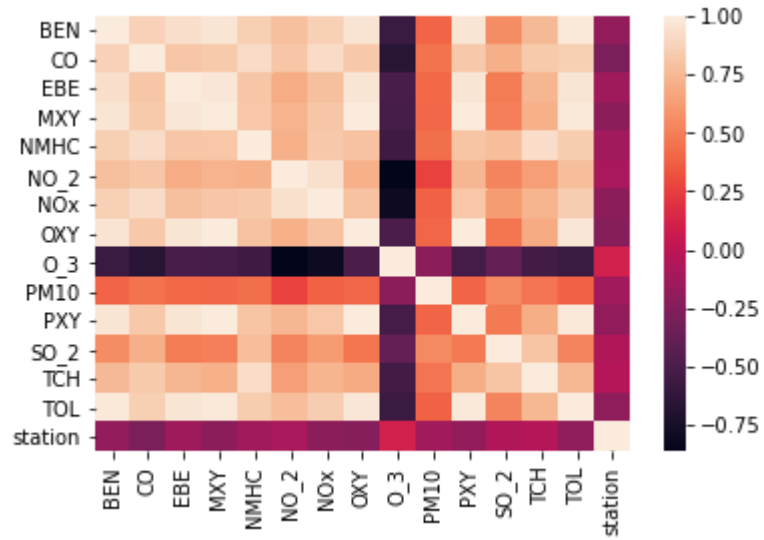



In [39]:

```
sns.heatmap(df1.corr())
```

Out[39]:

```
<AxesSubplot:>
```




In [40]:

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

In [41]:

```
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)
```

# Linear Regression

In [42]:

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

Out[42]:

```
LinearRegression()
```

In [43]:

```
lr.intercept_
```

Out[43]:

```
28078924.650785644
```

In [44]:

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

Out[44]:

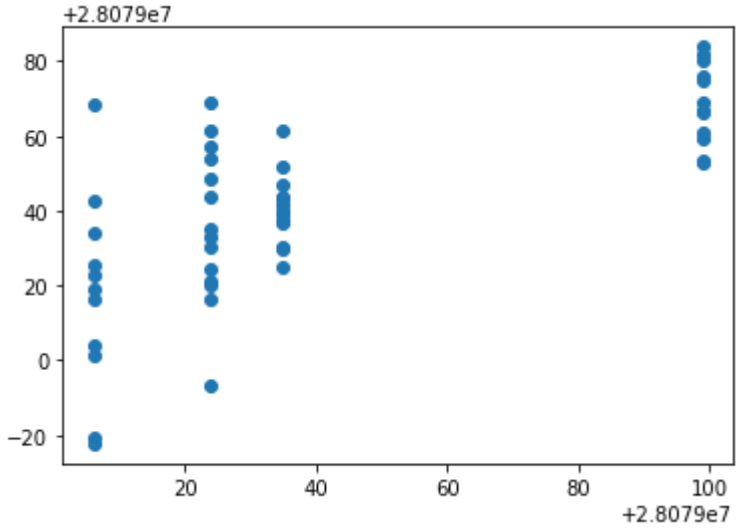
| | Co-efficient |
|---|---|
| **BEN** | 7.601835 |
| **CO** | -75.818249 |
| **EBE** | 26.947553 |
| **MXY** | -28.704971 |
| **NMHC** | 311.556604 |
| **NO_2** | 0.409014 |
| **NOx** | -0.062479 |
| **OXY** | -51.313632 |
| **O_3** | 0.021585 |
| **PM10** | 0.164274 |
| **PXY** | 140.334383 |
| **SO_2** | -2.038764 |
| **TCH** | 88.321620 |
| **TOL** | -6.460025 |

In [45]:

```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[45]:

```
<matplotlib.collections.PathCollection at 0x17682314a30>
```




In [46]:

```
lr.score(x_test,y_test)
```

Out[46]:

```
0.4872767878254125
```

In [47]:

```
lr.score(x_train,y_train)
```

Out[47]:

```
0.6299345764794582
```

In [48]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [54]:

```
r=Ridge(alpha=10)
r.fit(x_train,y_train)
```

Out[54]:

```
Ridge(alpha=10)
```

In [55]:

```
r.score(x_test,y_test)
```

Out[55]:

```
0.2631926973406742
```

In [56]:

```
r.score(x_train,y_train)
```

Out[56]:

```
0.396161240870166
```

In [57]:

```
l=Lasso(alpha=10)
l.fit(x_train,y_train)
```

Out[57]:

```
Lasso(alpha=10)
```

In [58]:

```
l.score(x_train,y_train)
```

Out[58]:

```
0.19468447289278856
```

In [59]:

```
l.score(x_test,y_test)
```

Out[59]:

```
0.17123023752648703
```

In [60]:

```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(x_train,y_train)
```

Out[60]:

```
ElasticNet()
```

In [61]:

```
e.coef_
```

Out[61]:

```
array([ 1.71173836, -0.56560351,  6.48100498, -0.81225867,  0.        ,
        1.2649111 , -0.61371316, -3.32224475,  0.18148639,  0.06472336,
        0.        ,  2.71840775,  0.        , -0.        ])
```

In [62]:

```
e.intercept_
```

Out[62]:

```
28078983.823303495
```

In [63]:

```
prediction=e.predict(x_test)
```

In [64]:

```
e.score(x_test,y_test)
```

Out[64]:

```
0.17899369923948127
```

In [67]:

```
from sklearn import metrics
```

In [68]:

```
print(metrics.mean_squared_error(y_test,prediction))
```

```
958.2210838066244
```

In [69]:

```
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
30.955146321841614
```

In [70]:

```
print(metrics.mean_absolute_error(y_test,prediction))
```

```
25.12592246942222
```

In [71]:

```
from sklearn.linear_model import LogisticRegression
```

In [72]:

```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [73]:

```
feature_matrix.shape
```

Out[73]:

```
(166, 14)
```

In [74]:

```
target_vector.shape
```

Out[74]:

```
(166,)
```

In [75]:

```
from sklearn.preprocessing import StandardScaler
```

In [76]:

```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [77]:

```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[77]:

```
LogisticRegression(max_iter=10000)
```

In [78]:

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

In [79]:

```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079035]
```

In [80]:

```
logr.classes_
```

Out[80]:

```
array([28079006, 28079024, 28079035, 28079099], dtype=int64)
```

In [81]:

```
logr.score(fs,target_vector)
```

Out[81]:

```
0.9397590361445783
```

In [82]:

```
logr.predict_proba(observation)[0][0]
```

Out[82]:

```
8.732936483311502e-18
```

In [83]:

```
logr.predict_proba(observation)
```

Out[83]:

```
array([[8.73293648e-18, 2.78894484e-10, 9.99999901e-01, 9.82938257e-08]])
```

In [84]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [85]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[85]:

```
RandomForestClassifier()
```

In [86]:

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

In [87]:

```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[87]:

```
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 [88]:

```
grid_search.best_score_
```

Out[88]:

```
0.6982758620689655
```

In [89]:

```
rfc_best=grid_search.best_estimator_
```

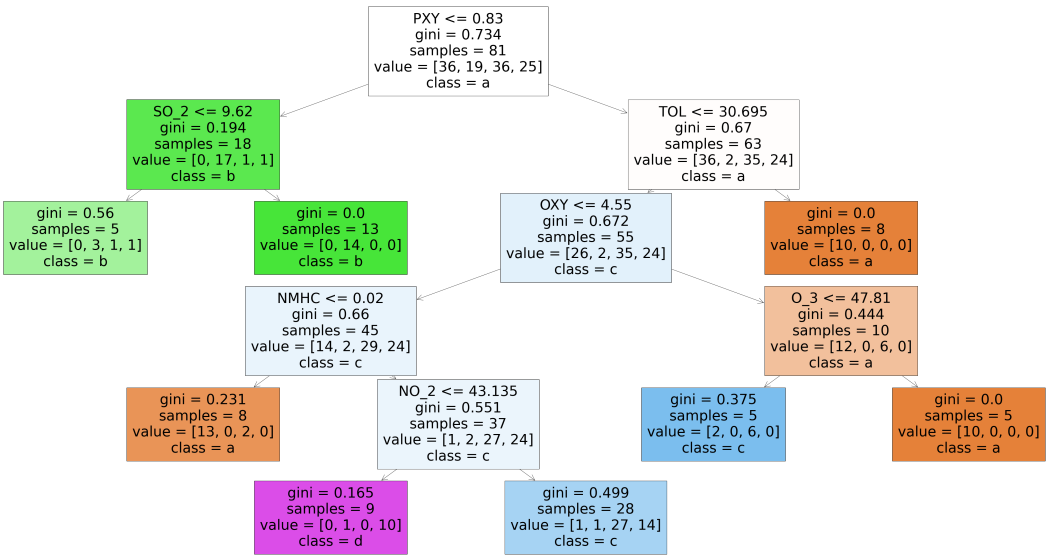
In [90]:

```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d'],f
```

Out[90]:

```
[Text(1984.0, 1993.2, 'PXY <= 0.83\ngini = 0.734\nsamples = 81\nvalue = [3
6, 19, 36, 25]\nclass = a'),
 Text(992.0, 1630.8000000000002, 'SO_2 <= 9.62\ngini = 0.194\nsamples = 18
\nvalue = [0, 17, 1, 1]\nclass = b'),
 Text(496.0, 1268.4, 'gini = 0.56\nsamples = 5\nvalue = [0, 3, 1, 1]\nclas
s = b'),
 Text(1488.0, 1268.4, 'gini = 0.0\nsamples = 13\nvalue = [0, 14, 0, 0]\ncl
ass = b'),
 Text(2976.0, 1630.8000000000002, 'TOL <= 30.695\ngini = 0.67\nsamples = 6
3\nvalue = [36, 2, 35, 24]\nclass = a'),
 Text(2480.0, 1268.4, 'OXY <= 4.55\ngini = 0.672\nsamples = 55\nvalue = [2
6, 2, 35, 24]\nclass = c'),
 Text(1488.0, 906.0, 'NMHC <= 0.02\ngini = 0.66\nsamples = 45\nvalue = [1
4, 2, 29, 24]\nclass = c'),
 Text(992.0, 543.5999999999999, 'gini = 0.231\nsamples = 8\nvalue = [13,
0, 2, 0]\nclass = a'),
 Text(1984.0, 543.5999999999999, 'NO_2 <= 43.135\ngini = 0.551\nsamples =
37\nvalue = [1, 2, 27, 24]\nclass = c'),
 Text(1488.0, 181.19999999999982, 'gini = 0.165\nsamples = 9\nvalue = [0,
1, 0, 10]\nclass = d'),
 Text(2480.0, 181.19999999999982, 'gini = 0.499\nsamples = 28\nvalue = [1,
1, 27, 14]\nclass = c'),
 Text(3472.0, 906.0, 'O_3 <= 47.81\ngini = 0.444\nsamples = 10\nvalue = [1
2, 0, 6, 0]\nclass = a'),
 Text(2976.0, 543.5999999999999, 'gini = 0.375\nsamples = 5\nvalue = [2,
0, 6, 0]\nclass = c'),
 Text(3968.0, 543.5999999999999, 'gini = 0.0\nsamples = 5\nvalue = [10, 0,
0, 0]\nclass = a'),
 Text(3472.0, 1268.4, 'gini = 0.0\nsamples = 8\nvalue = [10, 0, 0, 0]\ncla
ss = a')]
```




# logistic regression is best suitable for this dataset

In [ ]: