In [1]:

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

In [2]:

```
df1 = pd.read_csv(r"C:\Users\user\Desktop\c10\madrid_2005.csv")
df = df1.head(1000)
df
```

Out[2]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2005-11-01 01:00:00 | NaN | 0.77 | NaN | NaN | NaN | 57.130001 | 128.699997 | NaN | 14.720000 | 14.91( |
| **1** | 2005-11-01 01:00:00 | 1.52 | 0.65 | 1.49 | 4.57 | 0.25 | 86.559998 | 181.699997 | 1.27 | 11.680000 | 30.93( |
| **2** | 2005-11-01 01:00:00 | NaN | 0.40 | NaN | NaN | NaN | 46.119999 | 53.000000 | NaN | 30.469999 | 14.60( |
| **3** | 2005-11-01 01:00:00 | NaN | 0.42 | NaN | NaN | NaN | 37.220001 | 52.009998 | NaN | 21.379999 | 15.16( |
| **4** | 2005-11-01 01:00:00 | NaN | 0.57 | NaN | NaN | NaN | 32.160000 | 36.680000 | NaN | 33.410000 | 5.00( |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **995** | 2005-11-02 15:00:00 | 0.20 | 0.74 | 1.00 | NaN | 0.42 | 83.660004 | 182.600006 | NaN | 13.290000 | 76.16( |
| **996** | 2005-11-02 15:00:00 | NaN | 1.19 | NaN | NaN | NaN | NaN | NaN | NaN | 15.400000 | 66.70 |
| **997** | 2005-11-02 15:00:00 | NaN | 0.86 | NaN | NaN | NaN | 99.360001 | 163.100006 | NaN | 14.250000 | 54.36( |
| **998** | 2005-11-02 15:00:00 | NaN | 0.95 | NaN | NaN | 0.28 | 79.699997 | 168.199997 | NaN | 21.709999 | 47.88 |
| **999** | 2005-11-02 15:00:00 | NaN | 1.09 | NaN | NaN | NaN | 123.199997 | 259.299988 | NaN | 15.500000 | 60.93( |

1000 rows × 17 columns

In [3]:

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

In [4]:

```
df.columns
```

Out[4]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 81 entries, 5 to 993
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     81 non-null     object 
 1   BEN      81 non-null     float64
 2   CO       81 non-null     float64
 3   EBE      81 non-null     float64
 4   MXY      81 non-null     float64
 5   NMHC     81 non-null     float64
 6   NO_2     81 non-null     float64
 7   NOx      81 non-null     float64
 8   OXY      81 non-null     float64
 9   O_3      81 non-null     float64
 10  PM10     81 non-null     float64
 11  PM25     81 non-null     float64
 12  PXY      81 non-null     float64
 13  SO_2     81 non-null     float64
 14  TCH      81 non-null     float64
 15  TOL      81 non-null     float64
 16  station  81 non-null     int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 11.4+ KB
```

In [6]:

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

Out[6]:

| | CO | station |
|---|---|---|
| **5** | 0.88 | 28079006 |
| **22** | 0.22 | 28079024 |
| **25** | 0.49 | 28079099 |
| **31** | 0.84 | 28079006 |
| **48** | 0.20 | 28079024 |
| **...** | ... | ... |
| **941** | 1.36 | 28079006 |
| **961** | 0.81 | 28079099 |
| **967** | 1.07 | 28079006 |
| **987** | 0.78 | 28079099 |
| **993** | 1.11 | 28079006 |

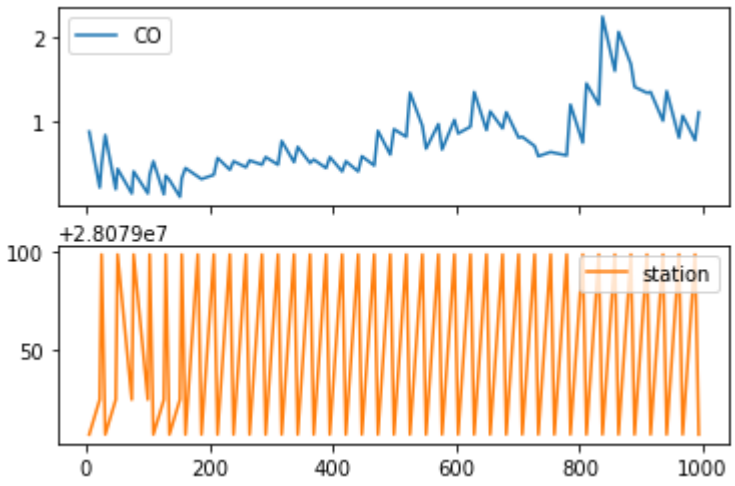
81 rows × 2 columns

In [7]:

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

Out[7]:

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

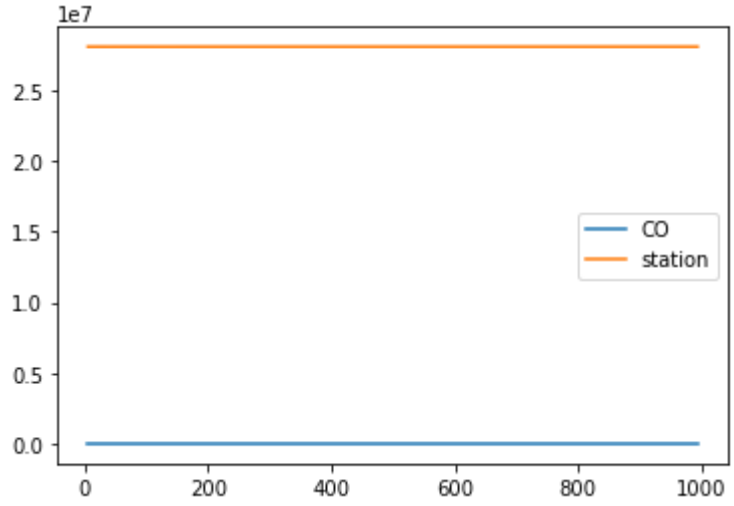


In [8]:

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

Out[8]:

<AxesSubplot:>




In [9]:

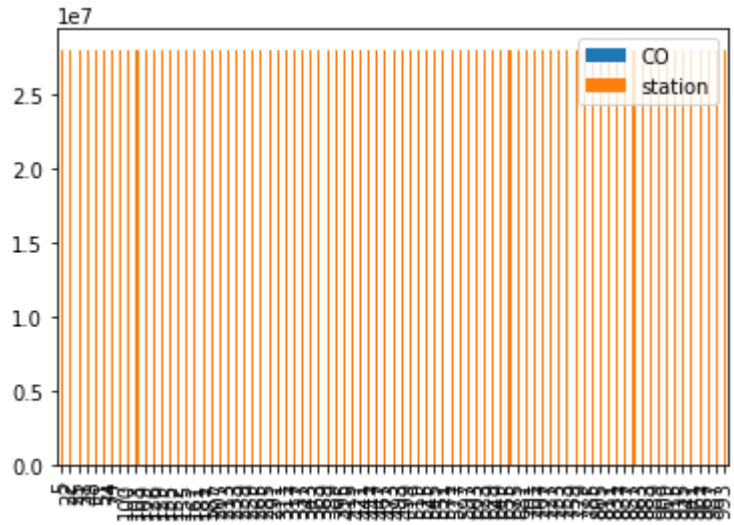
```
x = data[0:100]
```

In [10]:

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

Out[10]:

<AxesSubplot:>

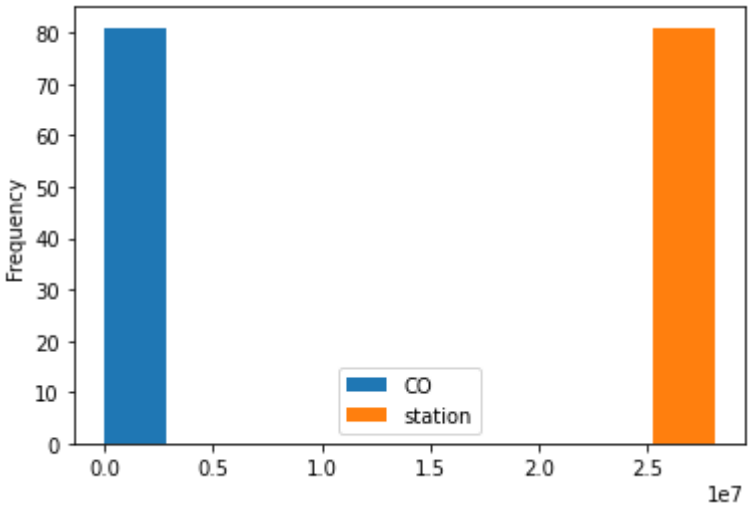


In [11]:

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

Out[11]:

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

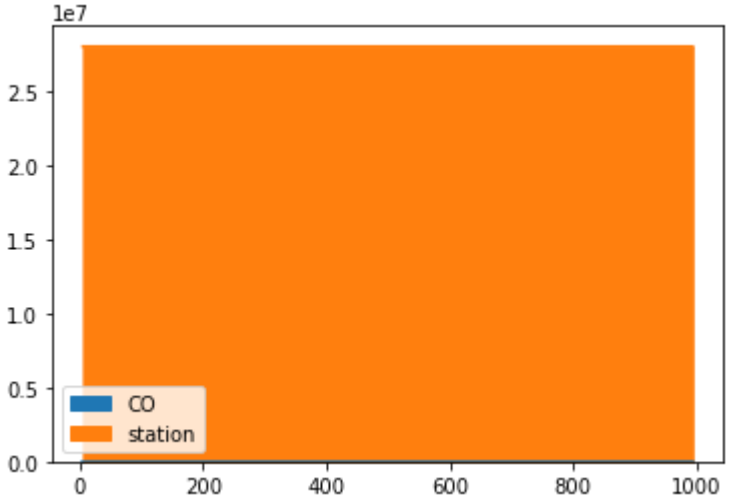



In [12]:

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

Out[12]:

```
<AxesSubplot:>
```

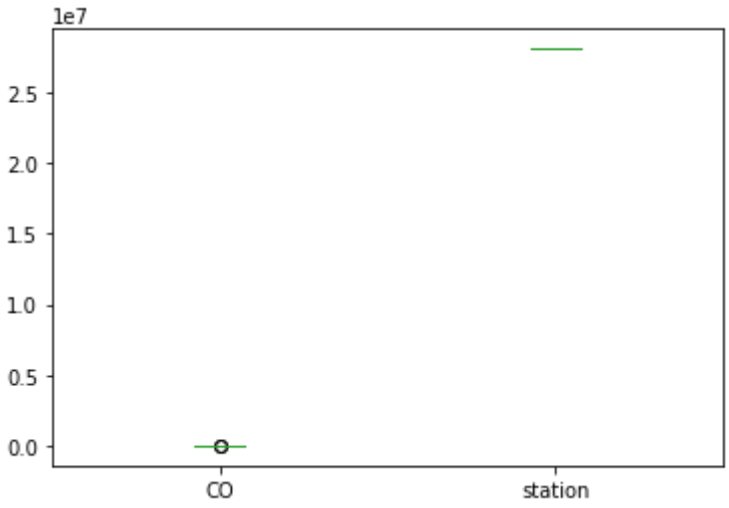


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

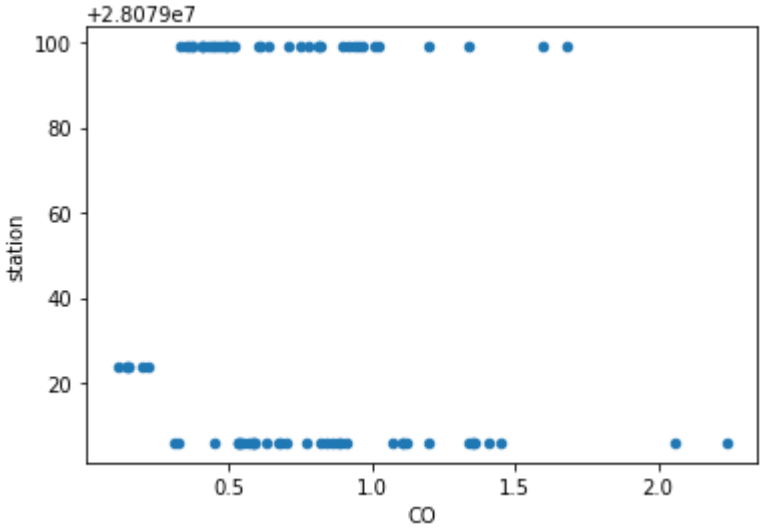
```
<AxesSubplot:ylabel='station'>
```

In [15]:

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

Out[15]:

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



In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 81 entries, 5 to 1993
Data columns (total 17 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      81 non-null     object
 1   BEN       81 non-null     float64
 2   CO        81 non-null     float64
 3   EBE       81 non-null     float64
 4   MXY       81 non-null     float64
 5   NMHC      81 non-null     float64
 6   NO_2      81 non-null     float64
 7   NOx       81 non-null     float64
 8   OXY       81 non-null     float64
 9   O_3       81 non-null     float64
 10  PM10      81 non-null     float64
 11  PM25      81 non-null     float64
 12  PXY       81 non-null     float64
 13  SO_2      81 non-null     float64
```

In [17]:

```
df.describe()
```

Out[17]:

| | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | C |
|---|---|---|---|---|---|---|---|---|
| count | 81.000000 | 81.000000 | 81.000000 | 81.000000 | 81.000000 | 81.000000 | 81.000000 | 81.000 |
| mean | 1.959877 | 0.756667 | 2.209506 | 6.021358 | 0.220123 | 64.633704 | 158.060864 | 2.967 |
| std | 1.635275 | 0.424935 | 1.827817 | 4.107748 | 0.111271 | 30.821367 | 111.029997 | 2.123 |
| min | 0.180000 | 0.110000 | 0.230000 | 0.480000 | 0.080000 | 3.110000 | 3.270000 | 0.350 |
| 25% | 0.810000 | 0.460000 | 1.020000 | 3.310000 | 0.140000 | 41.820000 | 67.330002 | 1.590 |
| 50% | 1.610000 | 0.640000 | 1.710000 | 5.060000 | 0.190000 | 62.470001 | 141.600006 | 2.420 |
| 75% | 2.510000 | 0.950000 | 2.520000 | 7.280000 | 0.280000 | 88.919998 | 207.699997 | 3.540 |
| max | 9.200000 | 2.240000 | 8.850000 | 18.469999 | 0.620000 | 125.099998 | 502.000000 | 10.210 |

In [18]:

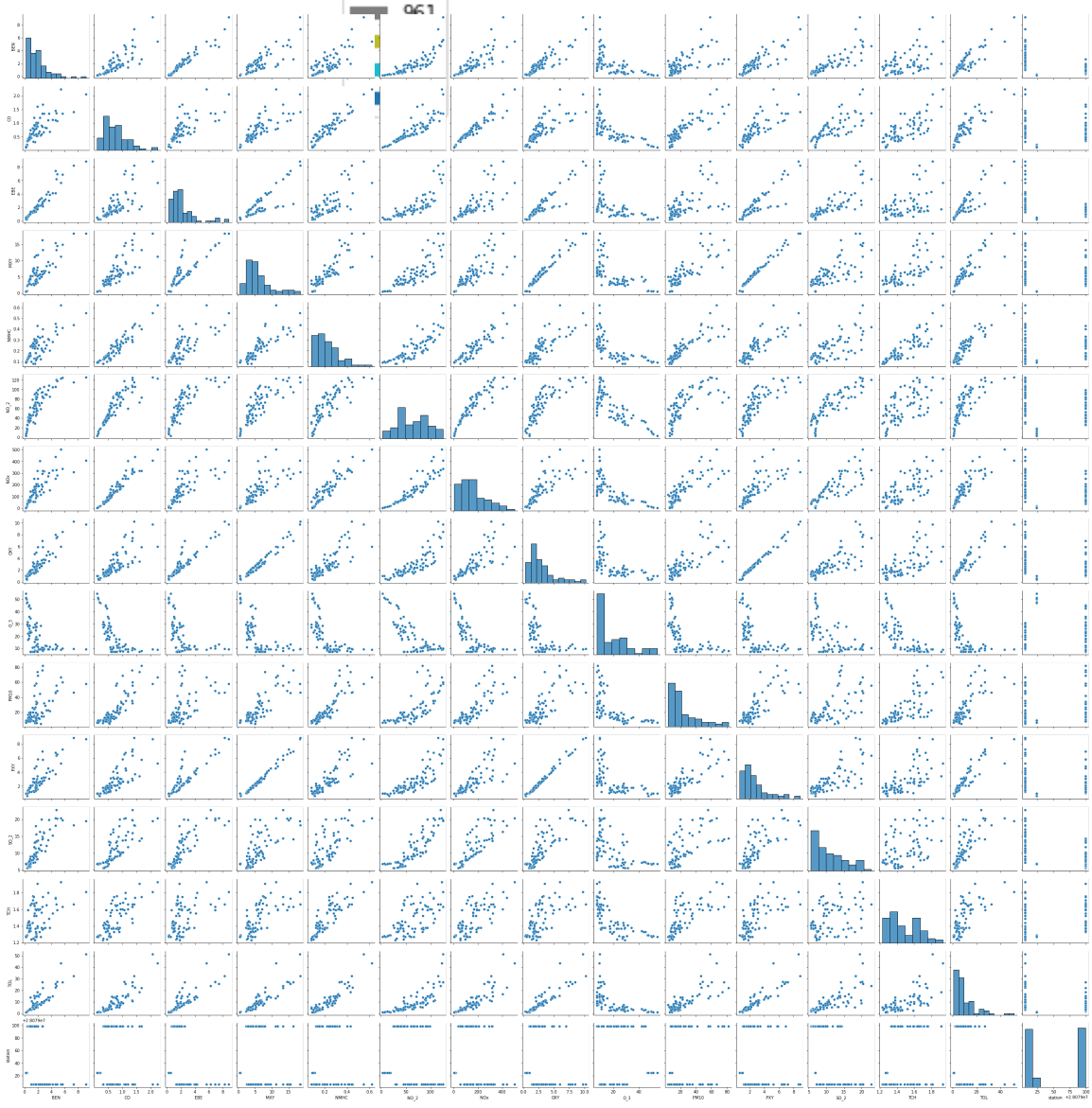
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

```
sb.pairplot(df1[0:100])
```

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x1d7f48bc430>
```



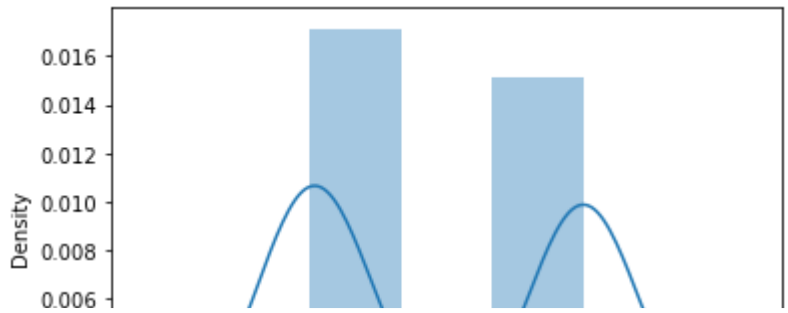
In [20]:

```
sb.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[20]:

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

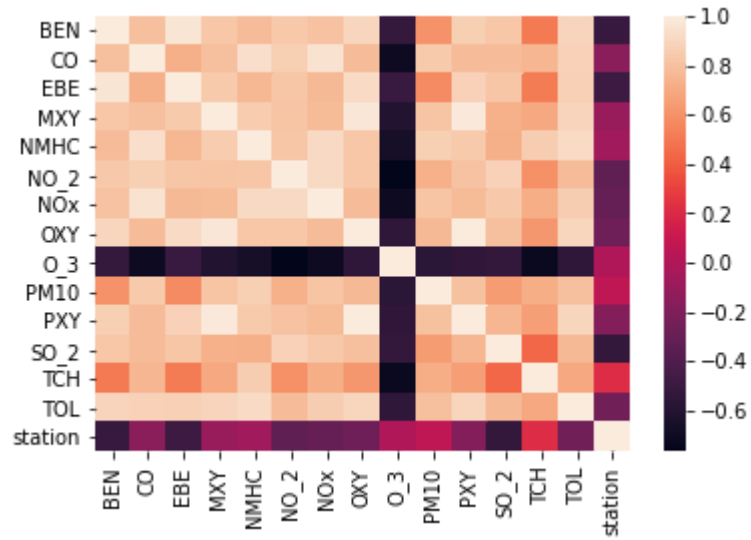



In [21]:

```
sb.heatmap(df1.corr())
```

Out[21]:

```
<AxesSubplot:>
```




In [22]:

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

In [23]:

```
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 [24]:

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

Out[24]:

```
LinearRegression()
```

In [25]:

```
lr.intercept_
```

Out[25]:

```
28079093.564677972
```

In [26]:

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

Out[26]:

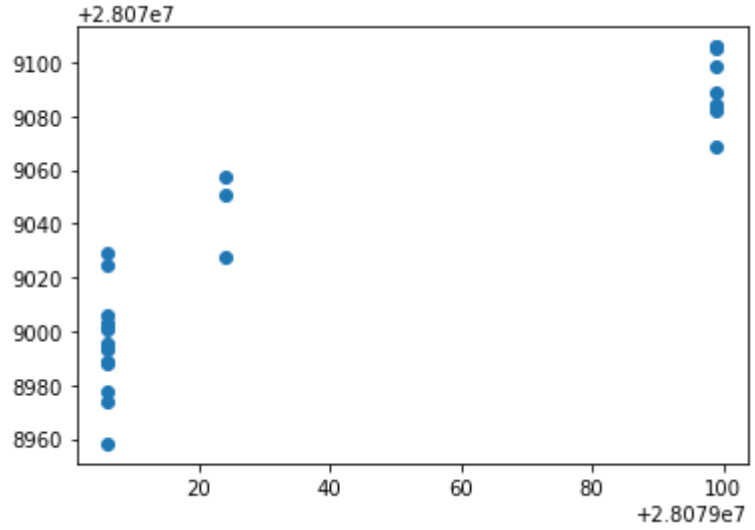
| | Co-efficient |
|---|---|
| **BEN** | -15.535247 |
| **CO** | 40.298445 |
| **EBE** | 9.310757 |
| **MXY** | 25.029354 |
| **NMHC** | 448.917491 |
| **NO_2** | -0.211860 |
| **NOx** | -0.316229 |
| **OXY** | -57.567245 |
| **O_3** | -0.209929 |
| **PM10** | 0.126558 |
| **PXY** | 15.362682 |
| **SO_2** | -2.965668 |
| **TCH** | -39.175654 |
| **TOL** | -3.089950 |

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x1d7832e5bb0>
```




In [28]:

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

Out[28]:

```
0.784710489508994
```

In [29]:

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

Out[29]:

```
0.8850679868106206
```

In [30]:

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

In [31]:

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

Out[31]:

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

In [32]:

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

Out[32]:

```
0.478858330840365
```

In [33]:

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

Out[33]:

```
0.771448803092762
```

In [34]:

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

Out[34]:

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

In [35]:

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

Out[35]:

```
0.6467128266950037
```

In [36]:

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

Out[36]:

```
0.5575472242436266
```

In [37]:

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

Out[37]:

```
ElasticNet()
```

In [38]:

```
e.coef_
```

Out[38]:

```
array([-6.16127131,  0.52067731, -3.77362299,  6.73127071,  0.        ,
       -0.14120895, -0.13577461, -2.21852862, -0.94357252,  1.54194123,
       -0.        , -6.04281372,  0.56856527, -1.8365477 ])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
28079132.366635006
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.4939516286450115
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
904.6493137943091
```

In [44]:

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

```
30.077388746270994
```

In [45]:

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

```
24.531105111837388
```

In [46]:

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

In [47]:

```
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 [48]:

```
feature_matrix.shape
```

Out[48]:

```
(81, 14)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(81,)
```

In [50]:

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

In [51]:

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

In [52]:

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

Out[52]:

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

In [53]:

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

In [54]:

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

```
[28079099]
```

In [55]:

```
logr.classes_
```

Out[55]:

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

In [56]:

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

Out[56]:

```
1.0
```

In [57]:

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

Out[57]:

```
0.03551946821940964
```

In [58]:

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

Out[58]:

```
array([[3.55194682e-02, 1.02444630e-15, 9.64480532e-01]])
```

In [59]:

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

In [60]:

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

Out[60]:

```
RandomForestClassifier()
```

In [61]:

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

In [62]:

```
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[62]:

```
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 [63]:

```
grid_search.best_score_
```

Out[63]:

```
0.7857142857142858
```

In [64]:

```
rfc_best=grid_search.best_estimator_
```

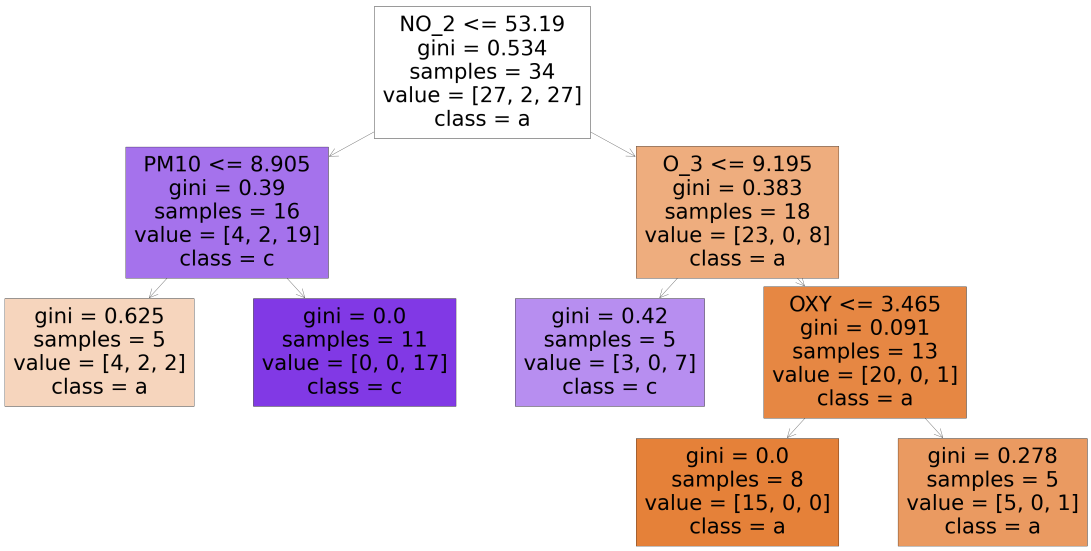
In [65]:

```
from sklearn.tree import plot_tree

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

Out[65]:

```
[Text(1984.0, 1902.6000000000001, 'NO_2 <= 53.19\ngini = 0.534\nsamples =
34\nvalue = [27, 2, 27]\nclass = a'),
 Text(992.0, 1359.0, 'PM10 <= 8.905\ngini = 0.39\nsamples = 16\nvalue =
[4, 2, 19]\nclass = c'),
 Text(496.0, 815.4000000000001, 'gini = 0.625\nsamples = 5\nvalue = [4, 2,
2]\nclass = a'),
 Text(1488.0, 815.4000000000001, 'gini = 0.0\nsamples = 11\nvalue = [0, 0,
17]\nclass = c'),
 Text(2976.0, 1359.0, 'O_3 <= 9.195\ngini = 0.383\nsamples = 18\nvalue =
[23, 0, 8]\nclass = a'),
 Text(2480.0, 815.4000000000001, 'gini = 0.42\nsamples = 5\nvalue = [3, 0,
7]\nclass = c'),
 Text(3472.0, 815.4000000000001, 'OXY <= 3.465\ngini = 0.091\nsamples = 13
\nvalue = [20, 0, 1]\nclass = a'),
 Text(2976.0, 271.79999999999995, 'gini = 0.0\nsamples = 8\nvalue = [15,
0, 0]\nclass = a'),
 Text(3968.0, 271.79999999999995, 'gini = 0.278\nsamples = 5\nvalue = [5,
0, 1]\nclass = a')]
```




# logistic regression is best suitable for this dataset

In [ ]: