In [2]:

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

In [3]:

```
df1 = pd.read_csv(r"C:\Users\user\Desktop\c10\madrid_2018.csv")
df = df1.head(1000)
df
```

Out[3]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	2018-03-02 18:00:00	NaN	NaN	0.4	NaN	NaN	13.0	62.0	83.0	31.0	NaN	NaN	NaN
996	2018-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	19.0	70.0	99.0	NaN	9.0	NaN	4.0
997	2018-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	42.0	88.0	152.0	NaN	3.0	2.0	NaN
998	2018-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	20.0	69.0	100.0	NaN	11.0	10.0	NaN
999	2018-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	19.0	76.0	105.0	14.0	NaN	NaN	NaN

1000 rows × 16 columns

In [4]:

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

In [5]:

```
df.columns
```

Out[5]:

```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_
3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83 entries, 1 to 990
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     83 non-null     object 
 1   BEN      83 non-null     float64
 2   CH4      83 non-null     float64
 3   CO       83 non-null     float64
 4   EBE      83 non-null     float64
 5   NMHC     83 non-null     float64
 6   NO       83 non-null     float64
 7   NO_2     83 non-null     float64
 8   NOx      83 non-null     float64
 9   O_3      83 non-null     float64
 10  PM10     83 non-null     float64
 11  PM25     83 non-null     float64
 12  SO_2     83 non-null     float64
 13  TCH      83 non-null     float64
 14  TOL      83 non-null     float64
 15  station  83 non-null     int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 11.0+ KB
```

In [7]:

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

Out[7]:

	CO	station
1	0.3	28079008
6	0.2	28079024
25	0.2	28079008
30	0.2	28079024
49	0.2	28079008
...	...	...
942	0.3	28079024
961	0.6	28079008
966	0.2	28079024
985	0.5	28079008
990	0.2	28079024

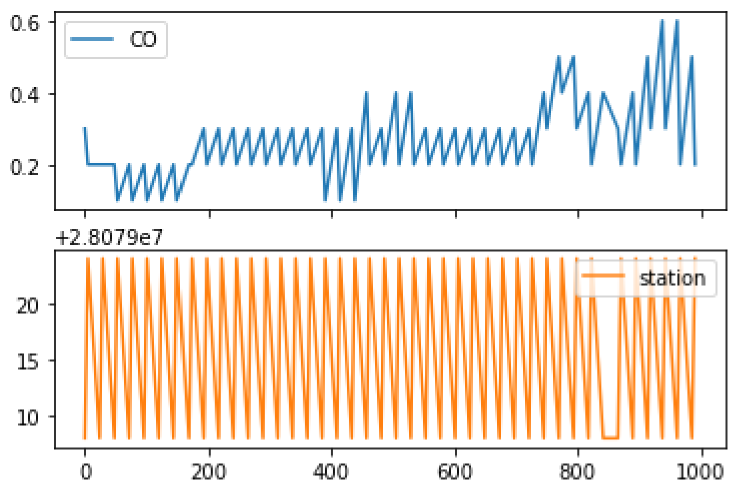
83 rows × 2 columns

In [8]:

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

Out[8]:

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

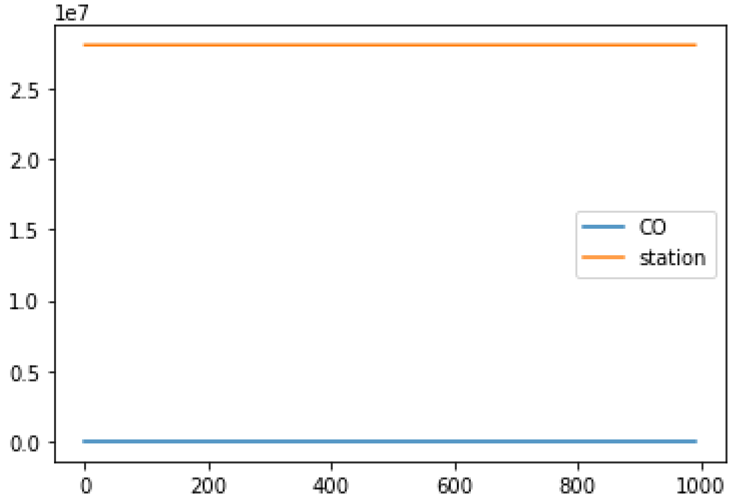


In [9]:

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

Out[9]:

<AxesSubplot:>



In [10]:

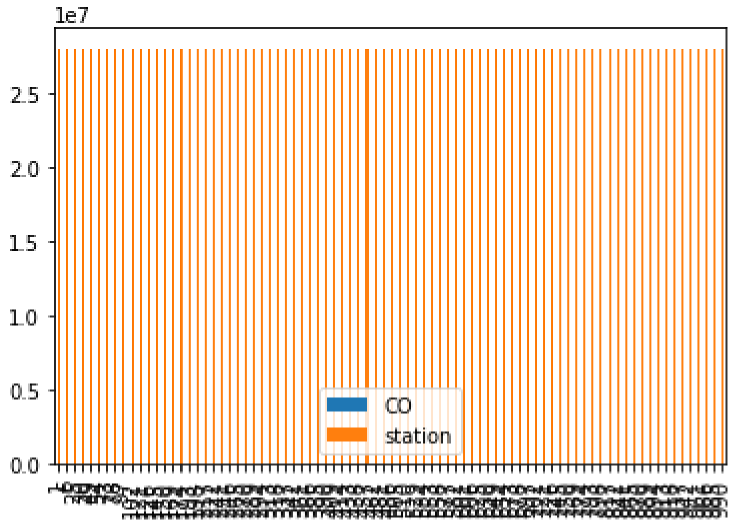
```
x = data[0:100]
```

In [11]:

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

Out[11]:

<AxesSubplot:>

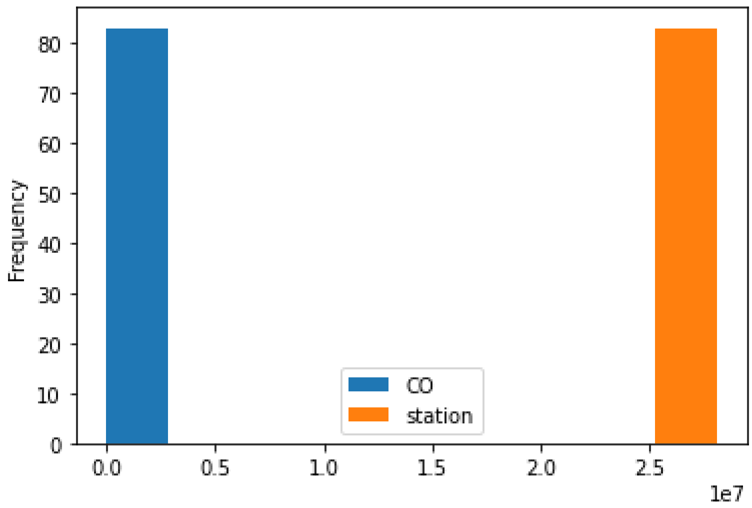


In [12]:

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

Out[12]:

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

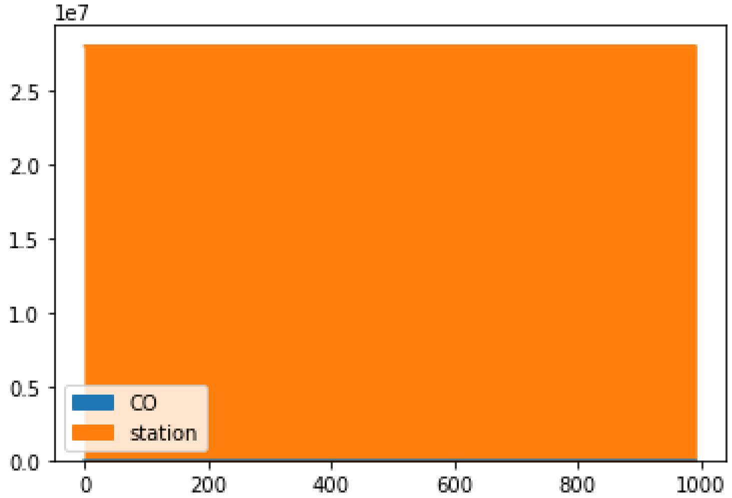


In [13]:

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

Out[13]:

```
<AxesSubplot:>
```

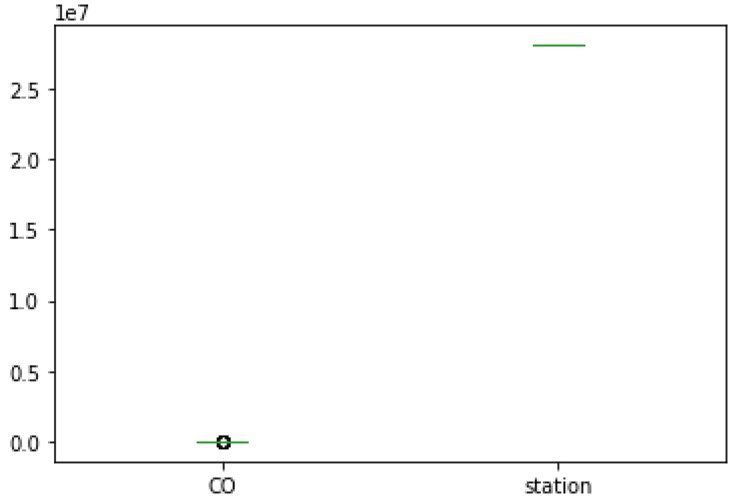


In [14]:

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

Out[14]:

<AxesSubplot:>



In [15]:

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

Out[15]:

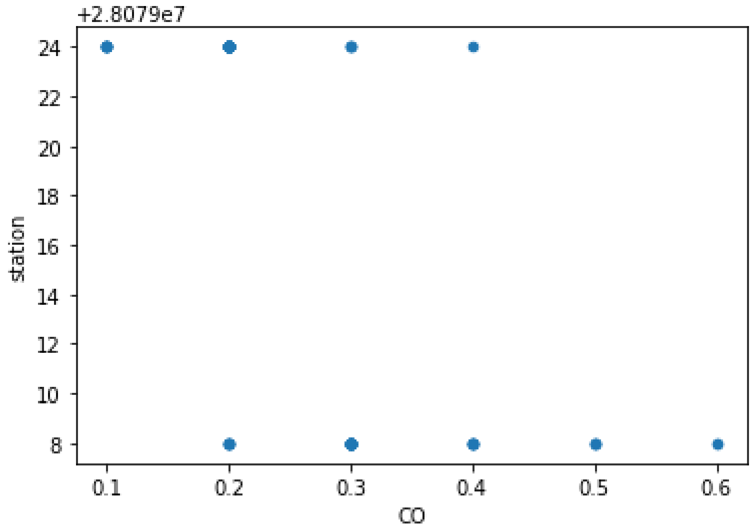
```
<AxesSubplot:ylabel='station'>
```

In [16]:

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

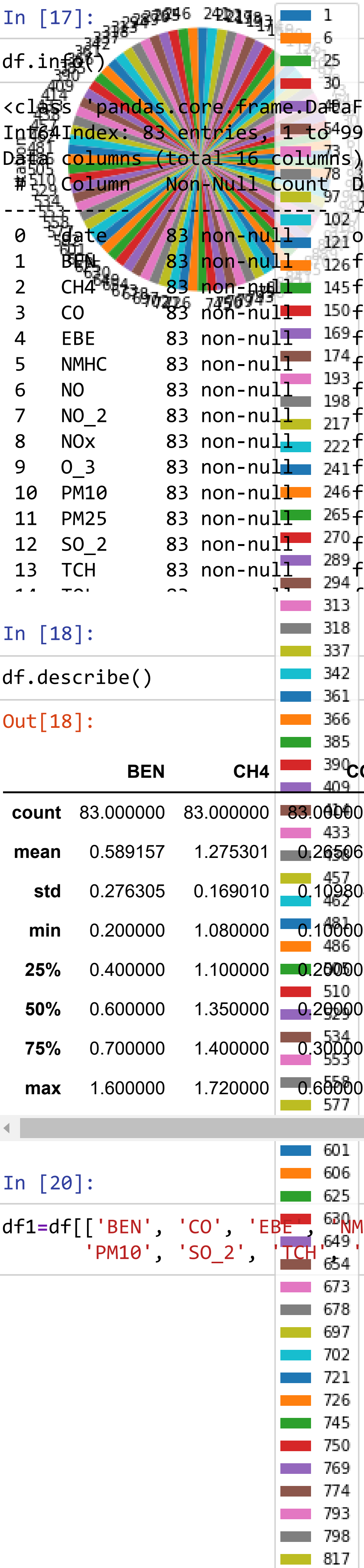
Out[16]:

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



In [17]:

```
df.info()
```




```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83 entries, 1 to 990
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     83 non-null     object
 1   BEN      83 non-null     float64
 2   CH4      83 non-null     float64
 3   CO       83 non-null     float64
 4   EBE      83 non-null     float64
 5   NMHC     83 non-null     float64
 6   NO       83 non-null     float64
 7   NO_2     83 non-null     float64
 8   NOx      83 non-null     float64
 9   O_3      83 non-null     float64
 10  PM10     83 non-null     float64
 11  PM25     83 non-null     float64
 12  SO_2     83 non-null     float64
 13  TCH      83 non-null     float64
```

In [18]:

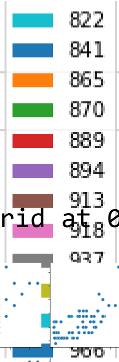
```
df.describe()
```

Out[18]:

	BEN	CH4	CO	EBE	NMHC	NO	NO_2	N
count	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000
mean	0.589157	1.275301	0.265060	0.185542	0.044096	12.783133	36.843373	56.385
std	0.276305	0.169010	0.109804	0.123103	0.020184	15.798308	27.034974	50.024
min	0.200000	1.080000	0.100000	0.100000	0.010000	1.000000	2.000000	3.000
25%	0.400000	1.100000	0.200000	0.100000	0.030000	1.000000	14.000000	17.000
50%	0.600000	1.350000	0.200000	0.100000	0.040000	4.000000	34.000000	40.000
75%	0.700000	1.400000	0.300000	0.200000	0.050000	19.500000	55.000000	89.000
max	1.600000	1.720000	0.600000	0.600000	0.110000	71.000000	102.000000	208.000

In [20]:

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

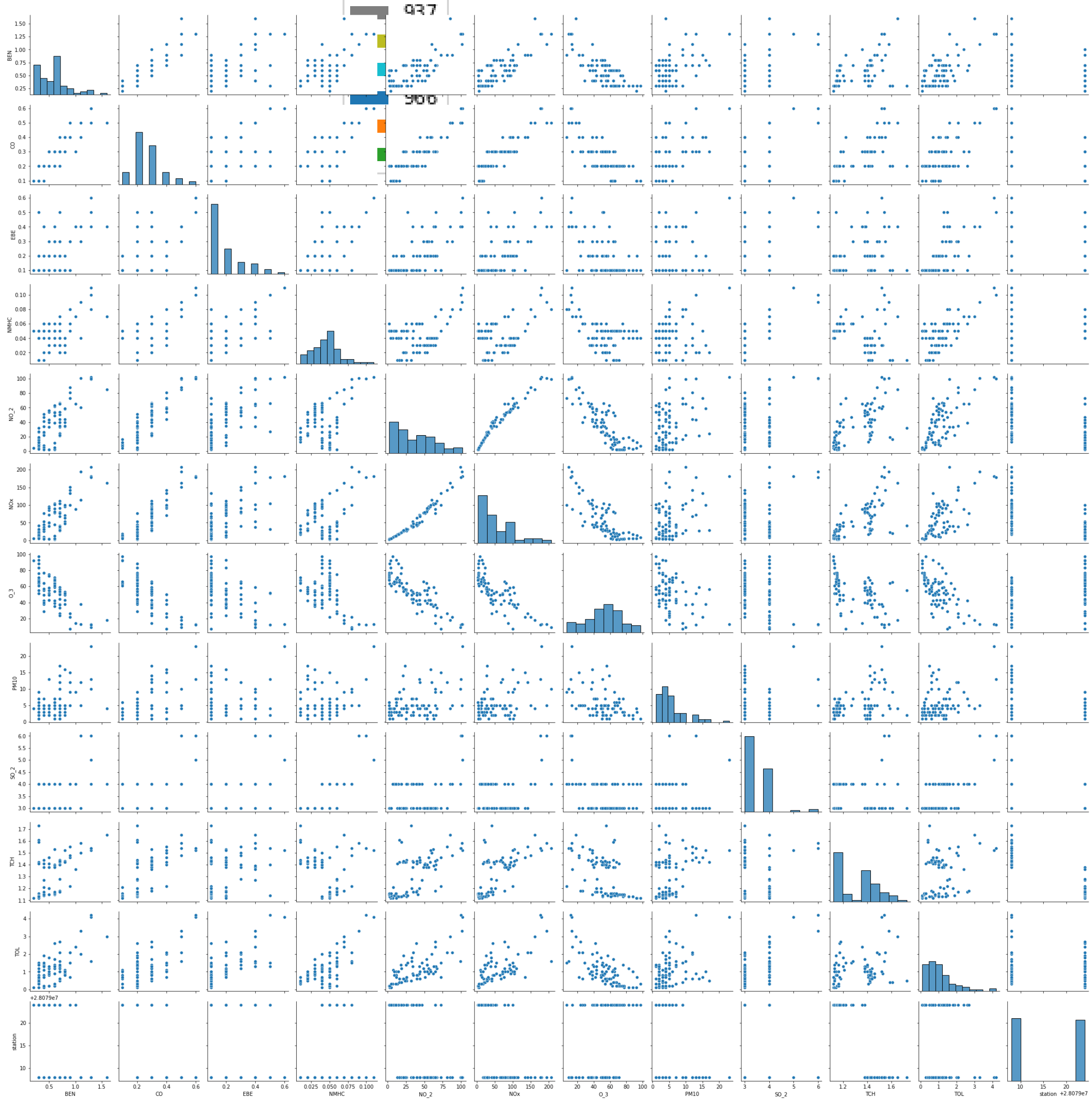


In [21]:

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

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x20e3ec8e250>
```



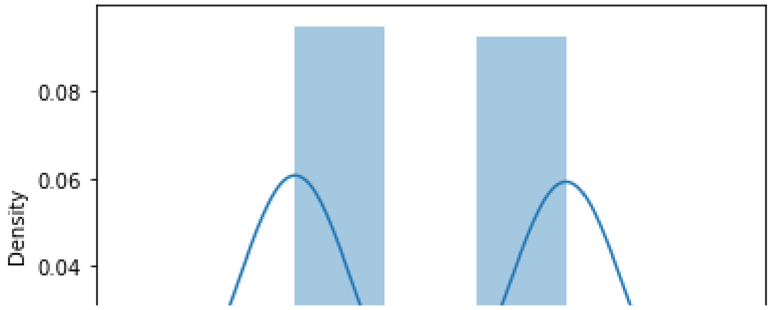
In [22]:

```
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[22]:

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



In [ ]:

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

In [23]:

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

In [24]:

```
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 [25]:

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

Out[25]:

```
LinearRegression()
```

In [26]:

```
lr.intercept_
```

Out[26]:

```
28079031.416664694
```

In [27]:

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

Out[27]:

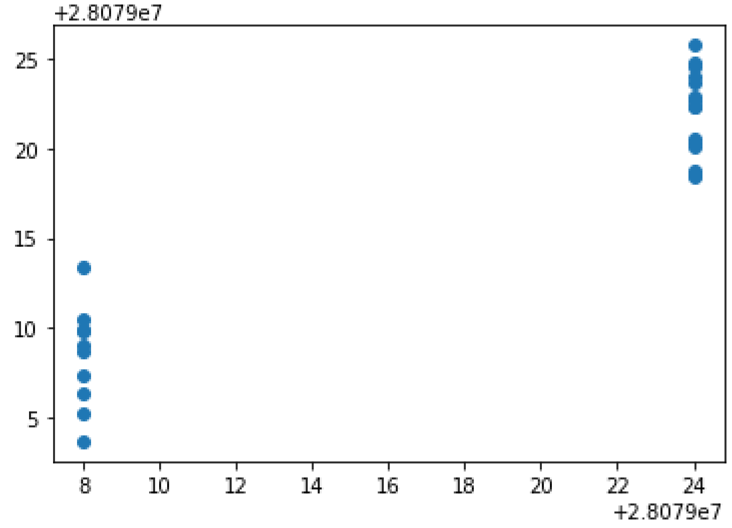
	Co-efficient
BEN	2.850632
CO	-33.764002
EBE	2.894612
NMHC	263.064382
NO_2	0.093230
NOx	-0.121530
O_3	-0.043294
PM10	0.019408
SO_2	-0.305820
TCH	-10.166375
TOL	-0.616478

In [28]:

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

Out[28]:

```
<matplotlib.collections.PathCollection at 0x20e47e3f070>
```



In [29]:

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

Out[29]:

```
0.8593006551325477
```

In [30]:

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

Out[30]:

```
0.9685777662792087
```

In [31]:

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

In [32]:

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

Out[32]:

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

In [33]:

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

Out[33]:

```
0.6843527368959668
```

In [34]:

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

Out[34]:

```
0.6805566756117155
```

In [35]:

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

Out[35]:

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

In [36]:

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

Out[36]:

```
0.43550728759496293
```

In [37]:

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

Out[37]:

```
0.3738345569498982
```

In [38]:

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

Out[38]:

```
ElasticNet()
```

In [39]:

```
e.coef_
```

Out[39]:

```
array([ 0.        , -0.        ,  0.        ,  0.        , -0.05044841,
       -0.14127213, -0.16530384, -0.21242968,  1.74123806, -0.0257283 ,
        0.640385  ])
```

In [40]:

```
e.intercept_
```

Out[40]:

```
28079028.52312789
```

In [41]:

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

In [42]:

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

Out[42]:

```
0.5566407016409418
```

In [43]:

```
from sklearn import metrics
```

In [44]:

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

```
27.239995291180534
```

In [45]:

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

```
5.219194889174051
```

In [46]:

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

```
4.456924369931221
```

In [47]:

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

In [49]:

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

In [50]:

```
feature_matrix.shape
```

Out[50]:

```
(83, 11)
```

In [51]:

```
target_vector.shape
```

Out[51]:

```
(83,)
```

In [52]:

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

In [53]:

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

In [54]:

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

Out[54]:

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

In [60]:

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

In [61]:

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

```
[28079008]
```

In [62]:

```
logr.classes_
```

Out[62]:

```
array([28079008, 28079024], dtype=int64)
```

In [63]:

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

Out[63]:

```
1.0
```

In [64]:

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

Out[64]:

```
0.9999999988636115
```

In [65]:

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

Out[65]:

```
array([[9.99999999e-01, 1.13638856e-09]])
```

In [66]:

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

In [67]:

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

Out[67]:

```
RandomForestClassifier()
```

In [68]:

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

In [69]:

```
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[69]:

```
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 [70]:

```
grid_search.best_score_
```

Out[70]:

```
0.9827586206896552
```

In [71]:

```
rfc_best=grid_search.best_estimator_
```

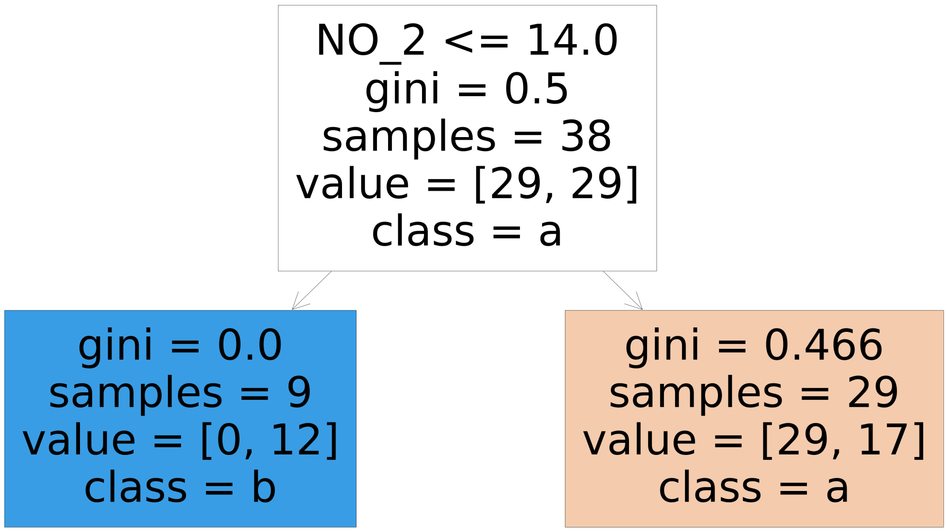
In [72]:

```
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[72]:

```
[Text(2232.0, 1630.8000000000002, 'NO_2 <= 14.0\ngini = 0.5\nsamples = 38
\nvalue = [29, 29]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 9\nvalue = [0, 12]
\nclass = b'),
 Text(3348.0, 543.5999999999999, 'gini = 0.466\nsamples = 29\nvalue = [29,
17]\nclass = a')]
```



random forest is best suitable for this dataset

In [ ]:

```

```