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_2015.csv")
df = df1.head(1000)
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	2015-10-02 18:00:00	NaN	0.2	NaN	NaN	9.0	28.0	79.0	NaN	NaN	NaN	NaN	NaN	28
996	2015-10-02 18:00:00	NaN	NaN	NaN	NaN	6.0	22.0	NaN	18.0	NaN	3.0	NaN	NaN	28
997	2015-10-02 18:00:00	NaN	NaN	NaN	NaN	5.0	5.0	NaN	11.0	2.0	NaN	NaN	NaN	28
998	2015-10-02 18:00:00	NaN	NaN	NaN	NaN	8.0	33.0	NaN	19.0	9.0	NaN	NaN	NaN	28
999	2015-10-02 18:00:00	NaN	NaN	NaN	NaN	1.0	16.0	82.0	NaN	NaN	NaN	NaN	NaN	28

1000 rows × 14 columns

In [3]:

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

In [4]:

```
df.columns
```

Out[4]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'P
M25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 84 entries, 1 to 990
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     84 non-null     object 
 1   BEN      84 non-null     float64
 2   CO       84 non-null     float64
 3   EBE      84 non-null     float64
 4   NMHC     84 non-null     float64
 5   NO       84 non-null     float64
 6   NO_2     84 non-null     float64
 7   O_3      84 non-null     float64
 8   PM10     84 non-null     float64
 9   PM25     84 non-null     float64
 10  SO_2     84 non-null     float64
 11  TCH      84 non-null     float64
 12  TOL      84 non-null     float64
 13  station  84 non-null     int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 9.8+ KB
```

In [6]:

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

Out[6]:

	CO	station
1	0.8	28079008
6	0.3	28079024
25	0.7	28079008
30	0.3	28079024
49	0.8	28079008
...	...	...
942	0.1	28079024
961	0.4	28079008
966	0.1	28079024
985	0.4	28079008
990	0.1	28079024

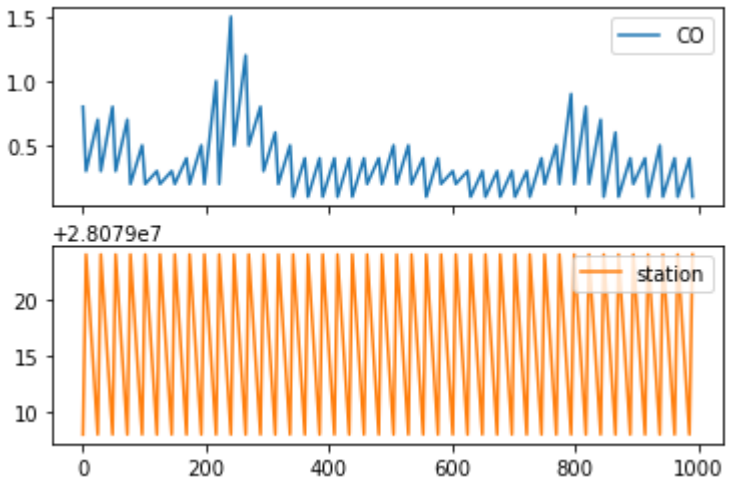
84 rows × 2 columns

In [7]:

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

Out[7]:

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

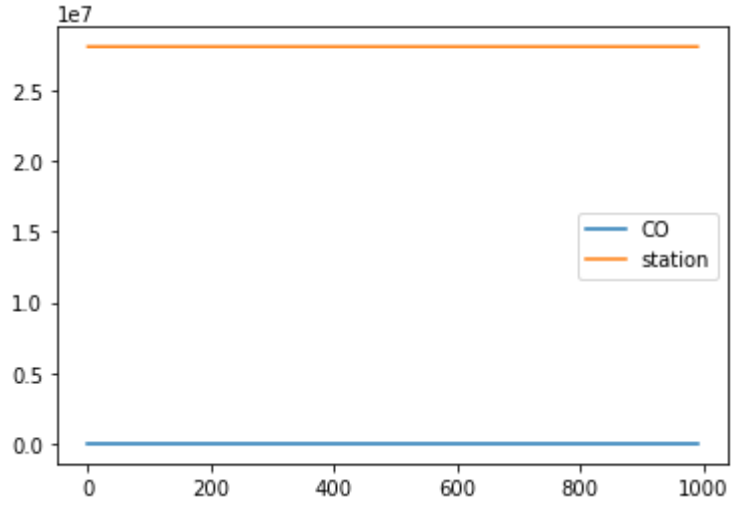


In [8]:

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

Out[8]:

```
<AxesSubplot:>
```



In [9]:

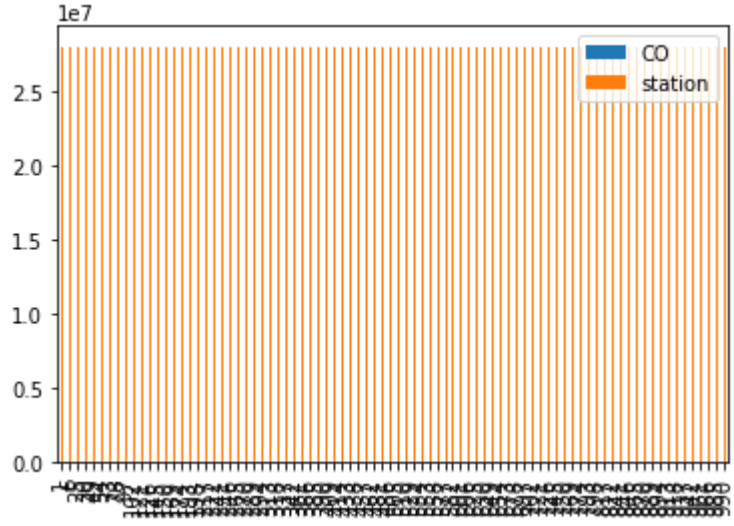
```
x = data[0:100]
```

In [10]:

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

Out[10]:

```
<AxesSubplot:>
```

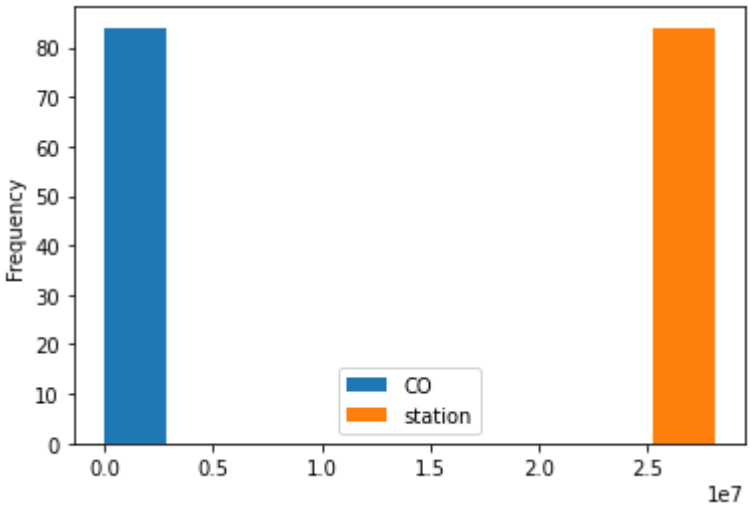


In [11]:

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

Out[11]:

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

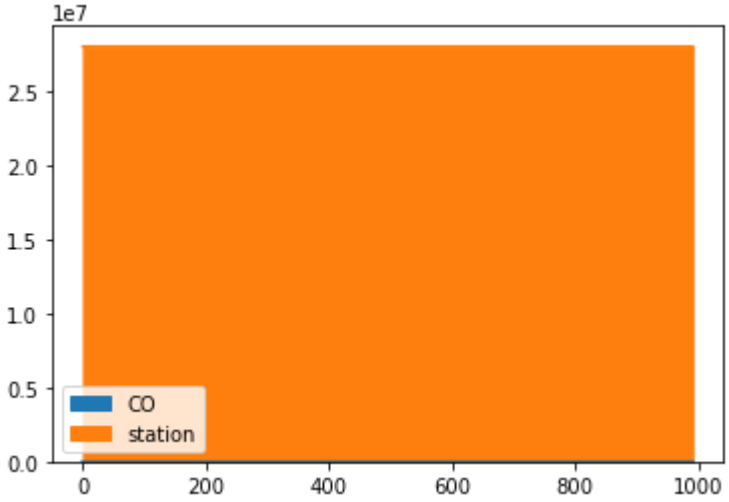


In [12]:

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

Out[12]:

```
<AxesSubplot:>
```

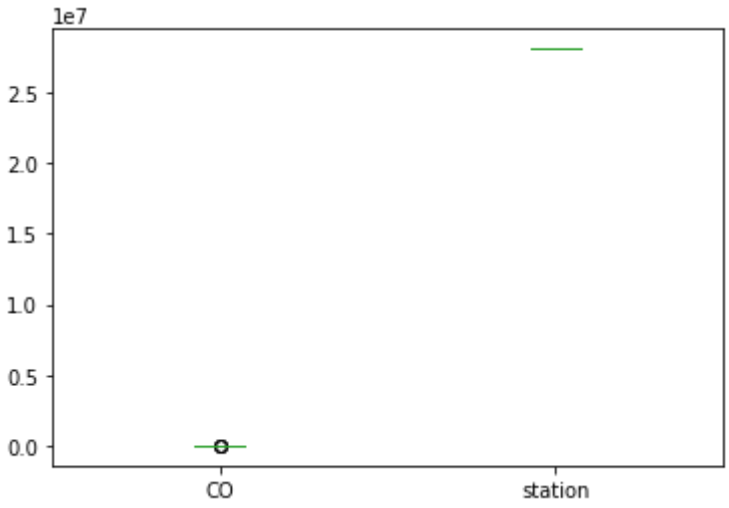


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

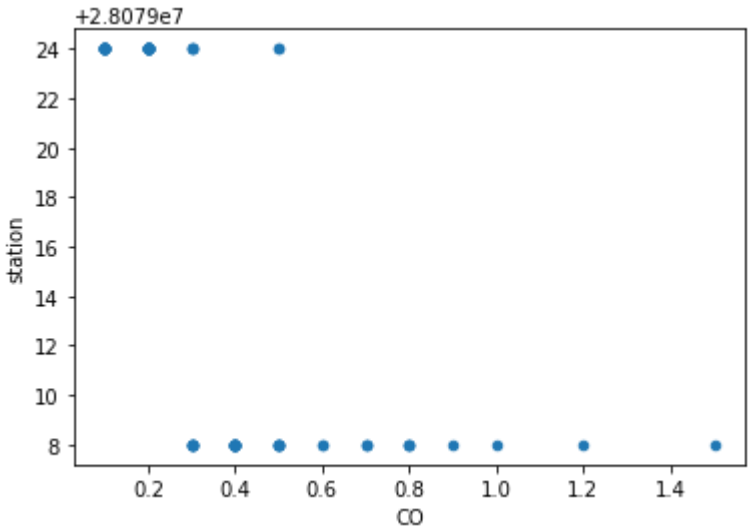
```
<AxesSubplot:ylabel='station'>
```

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: 84 entries, 1 to 990
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   date      84 non-null     object 
 1   BEN       84 non-null     float64
 2   CO        84 non-null     float64
 3   EBE       84 non-null     float64
 4   NMHC      84 non-null     float64
 5   NO        84 non-null     float64
 6   NO_2      84 non-null     float64
 7   O_3       84 non-null     float64
 8   PM10      84 non-null     float64
 9   PM25      84 non-null     float64
 10  SO_2      84 non-null     float64
 11  TCH       84 non-null     float64
 12  TOL       84 non-null     float64
 13  station   84 non-null     int64  
```

In [17]:

```
df.describe()
```

Out[17]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	P
count	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000
mean	0.651190	0.360714	0.470238	0.147976	18.083333	47.202381	42.464286	20.285
std	0.704802	0.264811	0.476630	0.077658	25.962071	35.225189	36.616688	11.169
min	0.100000	0.100000	0.100000	0.070000	1.000000	1.000000	1.000000	3.000
25%	0.100000	0.200000	0.100000	0.090000	1.000000	7.000000	8.000000	12.000
50%	0.400000	0.300000	0.300000	0.125000	5.000000	45.000000	32.000000	18.500
75%	0.900000	0.500000	0.600000	0.172500	19.250000	76.000000	79.500000	26.000
max	4.100000	1.500000	2.200000	0.410000	111.000000	111.000000	111.000000	61.000

In [18]:

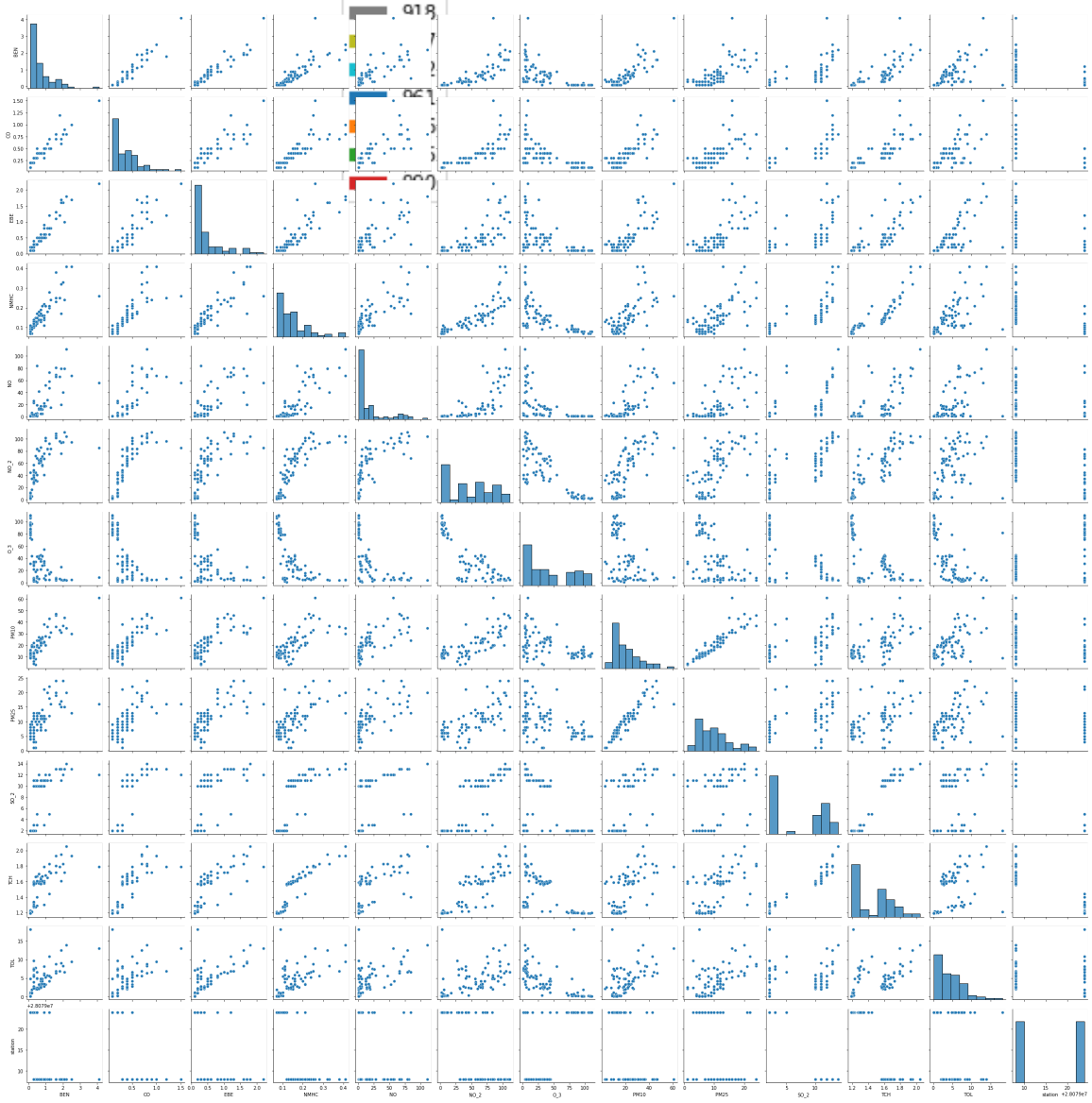
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x2166e4878e0>
```



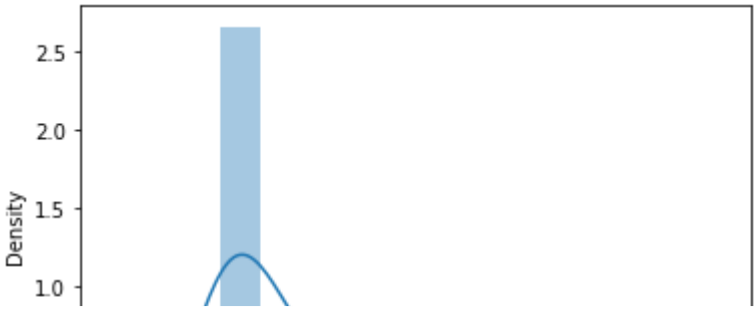
In [20]:

```
sb.distplot(df1['EBE'])
```

```
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='EBE', ylabel='Density'>
```

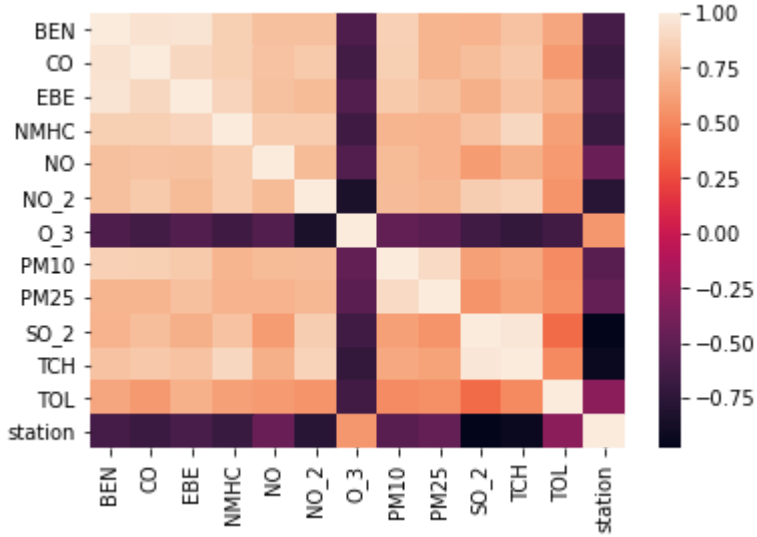


In [21]:

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

Out[21]:

```
<AxesSubplot:>
```



In [22]:

```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
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]:

```
-2.9802322387695312e-08
```

In [26]:

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

Out[26]:

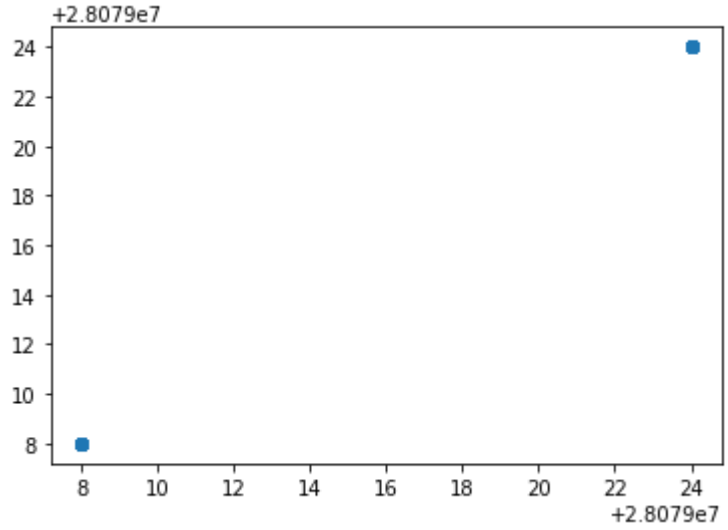
	Co-efficient
BEN	-1.535694e-16
CO	8.298219e-17
EBE	-3.913297e-16
NMHC	5.335089e-15
NO	-8.793682e-17
NO_2	3.026791e-16
O_3	3.029680e-16
PM10	-1.112615e-15
PM25	2.196117e-15
SO_2	1.137477e-15
TCH	-1.839824e-15
TOL	2.259574e-16
station	1.000000e+00

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x2167924bee0>
```



In [28]:

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

Out[28]:

```
1.0
```

In [29]:

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

Out[29]:

```
1.0
```

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]:

```
0.9998764664905837
```

In [33]:

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

Out[33]:

```
0.9998600238518036
```

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]:

```
0.9613234080647435
```

In [36]:

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

Out[36]:

```
0.9687740448522263
```

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([-0.        , -0.        , -0.        , -0.        ,  0.        ,
       -0.00490868, -0.        , -0.        ,  0.        , -0.        ,
       -0.        ,  0.        ,  0.96890461])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
873128.2097956985
```

In [40]:

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

In [53]:

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

Out[53]:

```
0.9996537376792173
```

In [54]:

```
from sklearn import metrics
```

In [55]:

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

```
0.021636272825239784
```

In [56]:

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

```
0.14709273546045631
```

In [57]:

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

```
0.11281863685983878
```

In [58]:

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

In [59]:

```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
target_vector=df['station']
```

In [60]:

```
feature_matrix.shape
```

Out[60]:

```
(84, 13)
```

In [61]:

```
target_vector.shape
```

Out[61]:

```
(84,)
```

In [62]:

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

In [63]:

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

In [64]:

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

Out[64]:

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

In [65]:

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

In [66]:

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

```
[28079008]
```

In [67]:

```
logr.classes_
```

Out[67]:

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

In [68]:

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

Out[68]:

```
1.0
```

In [69]:

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

Out[69]:

```
0.8659776048533394
```

In [70]:

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

Out[70]:

```
array([[0.8659776, 0.1340224]])
```

In [71]:

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

In [72]:

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

Out[72]:

```
RandomForestClassifier()
```

In [73]:

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

In [74]:

```
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[74]:

```
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 [75]:

```
grid_search.best_score_
```

Out[75]:

```
1.0
```

In [76]:

```
rfc_best=grid_search.best_estimator_
```

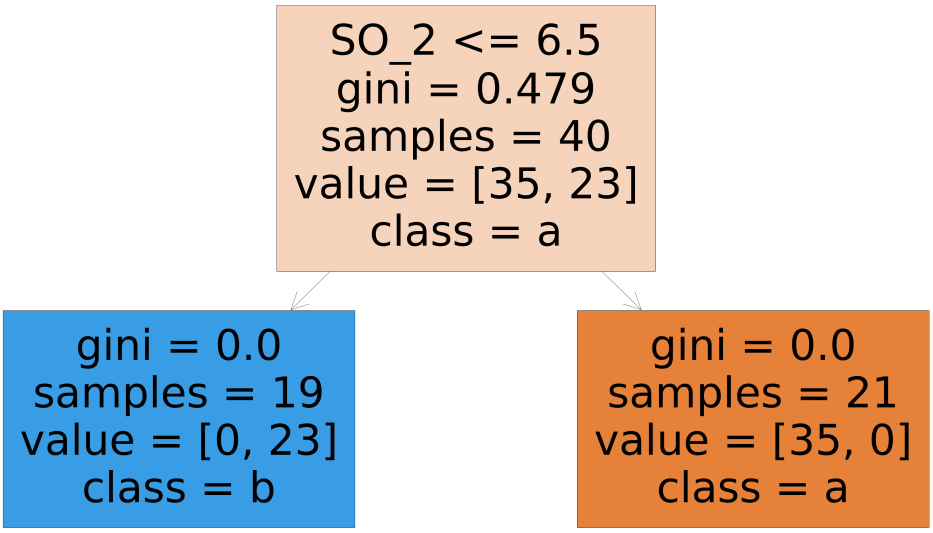
In [77]:

```
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[77]:

```
[Text(2232.0, 1630.8000000000002, 'SO_2 <= 6.5\ngini = 0.479\nsamples = 40
\nvalue = [35, 23]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 19\nvalue = [0, 2
3]\nclass = b'),
 Text(3348.0, 543.5999999999999, 'gini = 0.0\nsamples = 21\nvalue = [35,
0]\nclass = a')]
```



Elastic Net is suitable (0.9996537376792173)

logistic regression is best suitable for this dataset

In [ ]: