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_2013.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	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	2
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	2
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	1
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	2
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	2013-11-02 18:00:00	NaN	0.3	NaN	NaN	9.0	25.0	44.0	NaN	NaN	NaN	NaN	NaN	2
996	2013-11-02 18:00:00	NaN	NaN	NaN	NaN	3.0	15.0	NaN	7.0	NaN	1.0	NaN	NaN	2
997	2013-11-02 18:00:00	NaN	NaN	NaN	NaN	2.0	15.0	NaN	7.0	4.0	NaN	NaN	NaN	2
998	2013-11-02 18:00:00	NaN	NaN	NaN	NaN	9.0	34.0	NaN	9.0	3.0	NaN	NaN	NaN	2
999	2013-11-02 18:00:00	NaN	NaN	NaN	NaN	1.0	17.0	50.0	NaN	NaN	NaN	NaN	NaN	2

1000 rows × 14 columns

In [16]:

```
df=df.fillna('8.0')
```

In [17]:

```
df.columns
```

Out[17]:

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

In [18]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 0 entries
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     0 non-null      object 
 1   BEN      0 non-null      float64
 2   CO       0 non-null      float64
 3   EBE      0 non-null      float64
 4   NMHC     0 non-null      float64
 5   NO       0 non-null      float64
 6   NO_2     0 non-null      float64
 7   O_3      0 non-null      float64
 8   PM10     0 non-null      float64
 9   PM25     0 non-null      float64
 10  SO_2     0 non-null      float64
 11  TCH      0 non-null      float64
 12  TOL      0 non-null      float64
 13  station  0 non-null      int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 0.0+ bytes
```

In [20]:

```
data=df[['TOL' ,'SO_2']]
data
```

Out[20]:

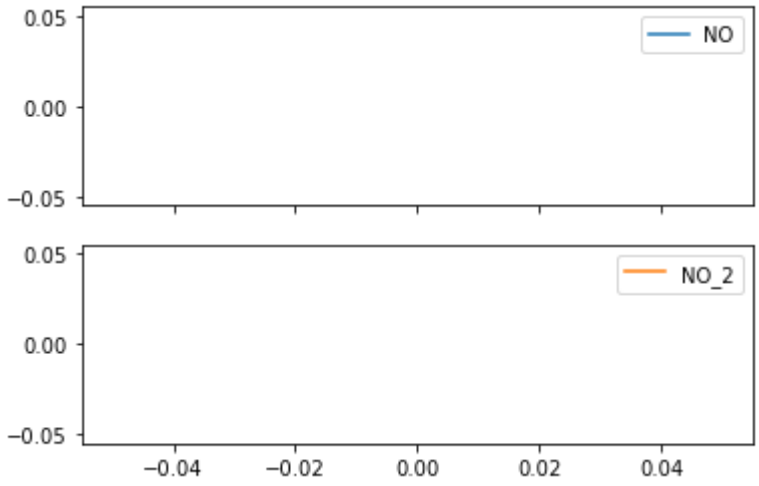
TOL	SO_2

In [15]:

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

Out[15]:

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

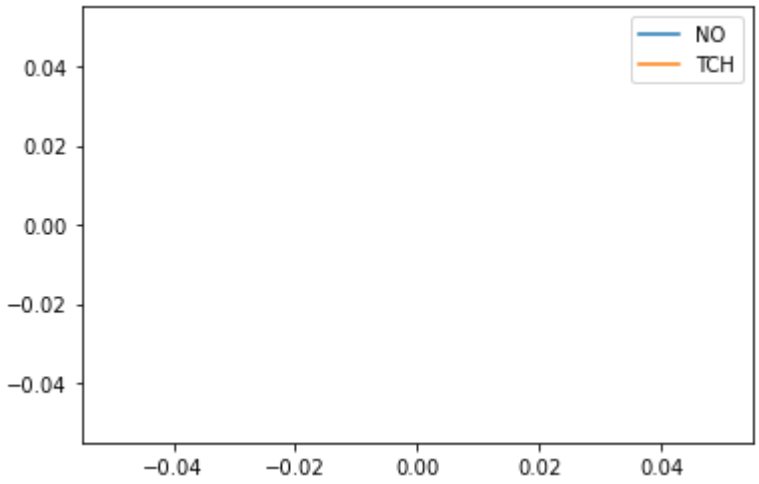


In [11]:

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

Out[11]:

```
<AxesSubplot:>
```



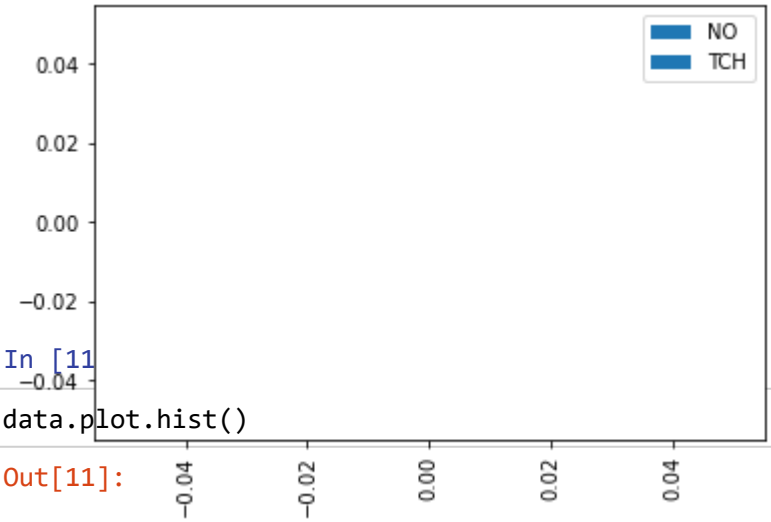
In [12]:

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

In [13]:

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

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-13-bf0fada621b6> in <module>
----> 1 x.plot.bar()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\plotting\_core.py in bar(self, x, y, **kwargs)
   1111         other axis represents a measured value.
   1112         """
-> 1113         return self(kind="bar", x=x, y=y, **kwargs)
   1114
   1115     @Appender(

C:\ProgramData\Anaconda3\lib\site-packages\pandas\plotting\_core.py in __call__(self, *args, **kwargs)
    953                     data.columns = label_name
    954
--> 955         return plot_backend.plot(data, kind=kind, **kwargs)
    956
    957     __call__.__doc__ = __doc__

C:\ProgramData\Anaconda3\lib\site-packages\pandas\plotting\_matplotlib\__init__.py in plot(data, kind, **kwargs)
     59             kwargs["ax"] = getattr(ax, "left_ax", ax)
     60     plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 61     plot_obj.generate()
     62     plot_obj.draw()
     63     return plot_obj.result

C:\ProgramData\Anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py in generate(self)
    285         for ax in self.axes:
    286             self._post_plot_logic_common(ax, self.data)
--> 287             self._post_plot_logic(ax, self.data)
    288
    289     def _args_adjust(self):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py in _post_plot_logic(self, ax, data)
   1492         name = self._get_index_name()
   1493
-> 1494         s_edge = self.ax_pos[0] - 0.25 + self.lim_offset
   1495         e_edge = self.ax_pos[-1] + 0.25 + self.bar_width + self.lim_offset
   1496

IndexError: index 0 is out of bounds for axis 0 with size 0
```

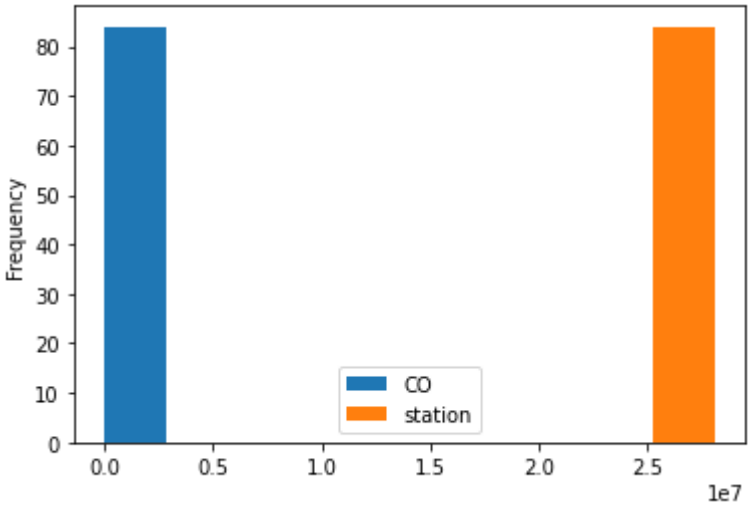


In [11]:

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

Out[11]:

<AxesSubplot:ylabel='Frequency'>

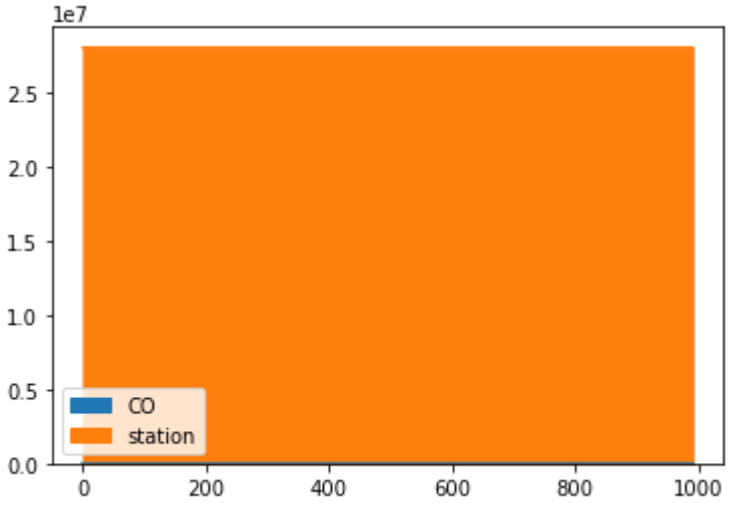


In [12]:

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

Out[12]:

<AxesSubplot:>

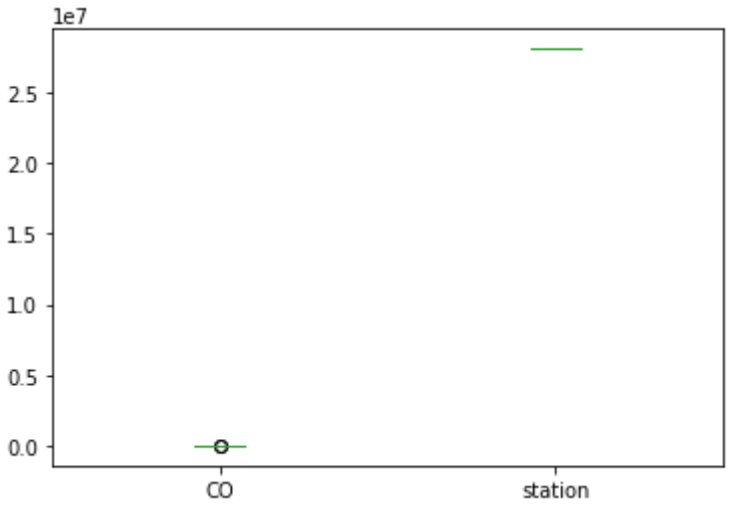


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

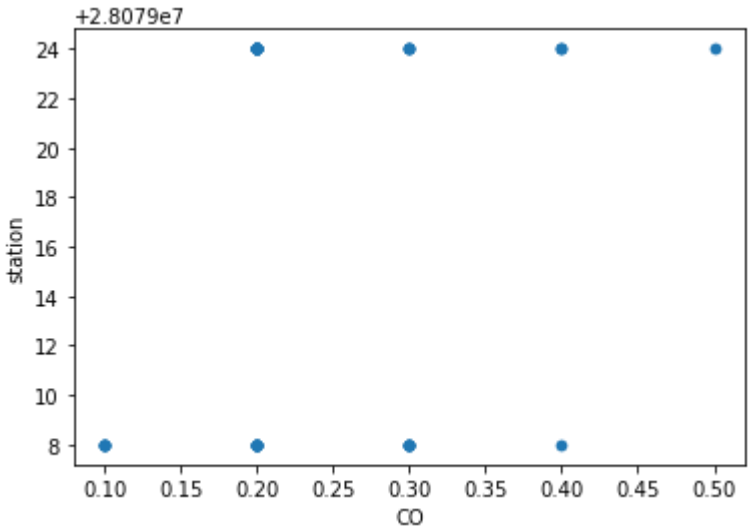
```
<AxesSubplot:ylabel='station'>
```

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	PM1
count	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.00000
mean	0.810714	0.234524	1.347619	0.150476	16.178571	42.107143	22.404762	25.88095
std	0.388391	0.089838	0.494908	0.043490	14.639574	19.695280	16.990514	6.95147
min	0.200000	0.100000	0.500000	0.090000	1.000000	2.000000	3.000000	9.00000
25%	0.500000	0.200000	1.000000	0.120000	3.750000	32.000000	6.750000	22.00000
50%	0.700000	0.200000	1.300000	0.150000	13.500000	41.000000	20.500000	25.00000
75%	1.000000	0.300000	1.600000	0.170000	23.250000	55.250000	37.250000	30.00000
max	2.500000	0.500000	3.500000	0.260000	68.000000	92.000000	60.000000	41.00000

In [18]:

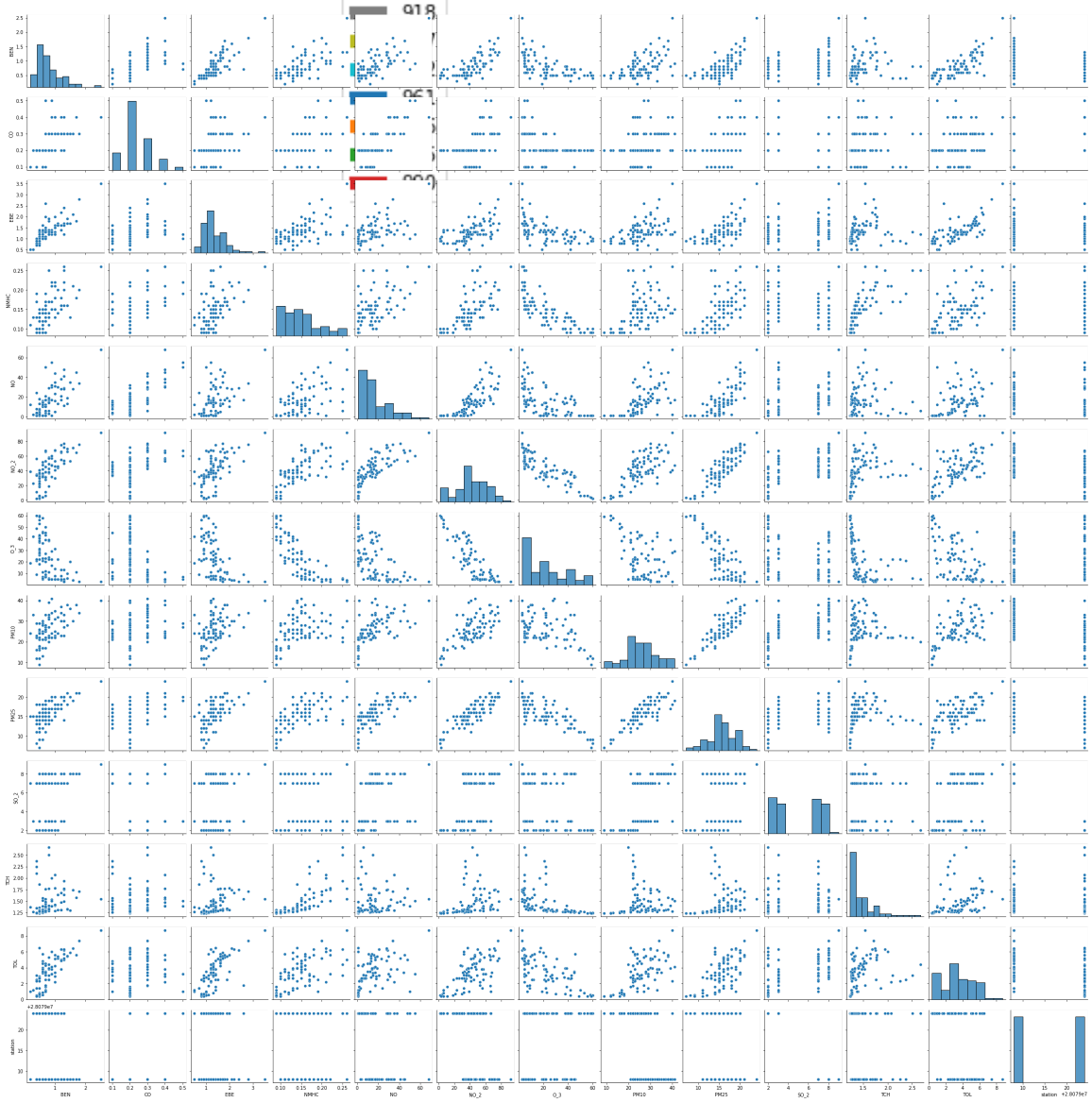
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x29966eec160>
```



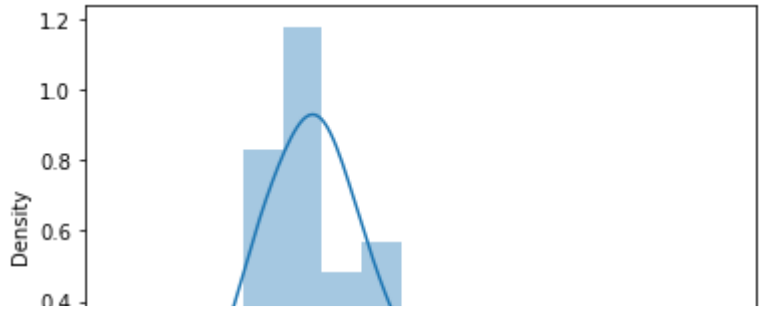
In [20]:

```
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'>
```

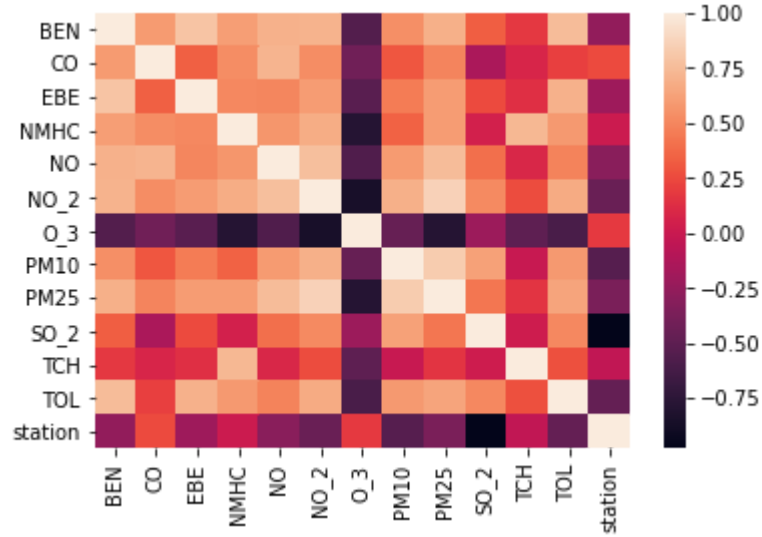


In [21]:

```
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]:

```
1.1175870895385742e-08
```

In [26]:

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

Out[26]:

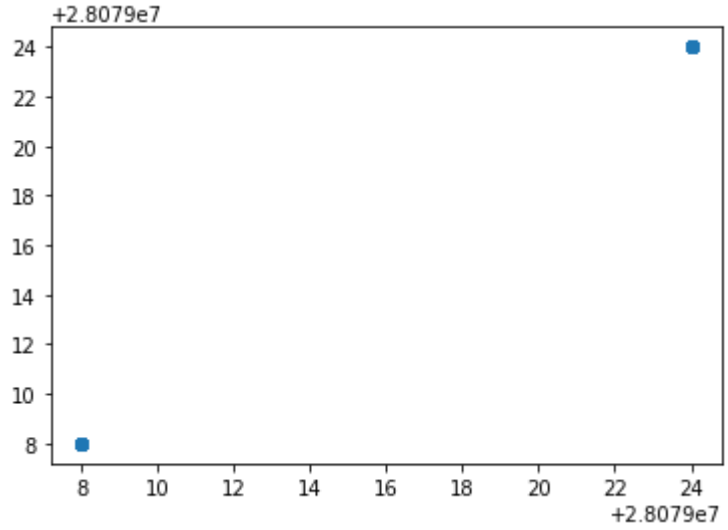
	Co-efficient
BEN	-7.359959e-16
CO	-1.019861e-14
EBE	3.705224e-15
NMHC	-5.754017e-14
NO	1.508851e-15
NO_2	3.046926e-15
O_3	-2.344186e-15
PM10	-1.839644e-16
PM25	1.704662e-16
SO_2	1.947208e-15
TCH	3.585601e-15
TOL	-8.594264e-17
station	1.000000e+00

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x29970b7c280>
```



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]:

```
0.999934625738826
```

In [33]:

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

Out[33]:

```
0.9999448285040875
```

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]:

```
0.9746297850229715
```

In [36]:

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

Out[36]:

```
0.9690649117201806
```

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([-0.00000000e+00,  0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00, -9.42149800e-04, -0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
        9.83469669e-01])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
464155.4993694611
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.9997198999331961
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
0.01622922990617703
```

In [44]:

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

```
0.12739399478066865
```

In [45]:

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

```
0.12457750794979242
```

In [46]:

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

In [52]:

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

In [53]:

```
feature_matrix.shape
```

Out[53]:

```
(84, 13)
```

In [54]:

```
target_vector.shape
```

Out[54]:

```
(84,)
```

In [55]:

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

In [56]:

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

In [57]:

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

Out[57]:

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

In [61]:

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

In [62]:

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

```
[28079008]
```

In [63]:

```
logr.classes_
```

Out[63]:

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

In [64]:

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

Out[64]:

```
1.0
```

In [65]:

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

Out[65]:

```
0.9012473007112899
```

In [66]:

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

Out[66]:

```
array([[0.9012473, 0.0987527]])
```

In [67]:

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

In [68]:

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

Out[68]:

```
RandomForestClassifier()
```

In [69]:

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

In [70]:

```
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[70]:

```
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 [71]:

```
grid_search.best_score_
```

Out[71]:

```
1.0
```

In [72]:

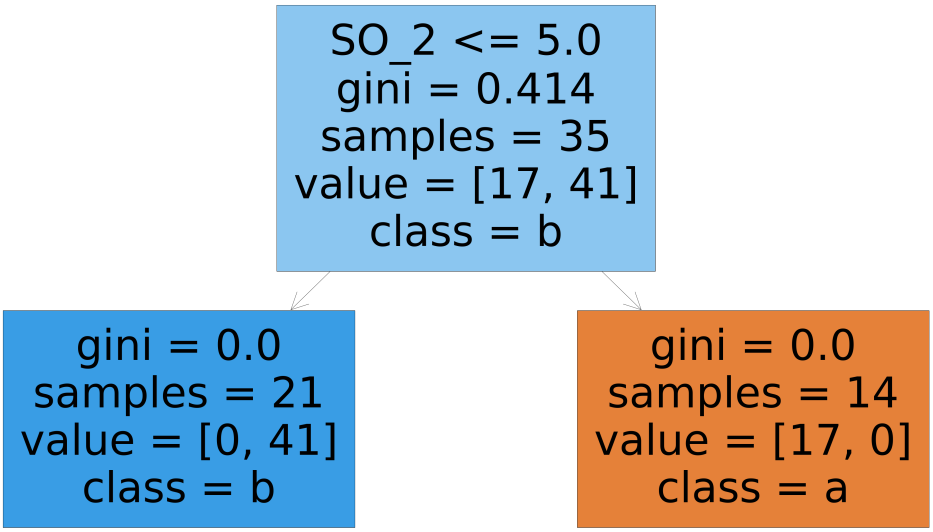
```
rfc_best=grid_search.best_estimator_
```

In [73]:

```
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[73]:

```
[Text(2232.0, 1630.8000000000002, 'SO_2 <= 5.0\ngini = 0.414\nsamples = 35
\nvalue = [17, 41]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 21\nvalue = [0, 4
1]\nclass = b'),
 Text(3348.0, 543.5999999999999, 'gini = 0.0\nsamples = 14\nvalue = [17,
0]\nclass = a')]
```



random forest is best suitable for this dataset

In [ ]: