20104169 - SUMESH R

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
df=pd.read_csv("/content/drive/MyDrive/mydatasets/csvs_per_year/madrid_2013.csv")
df
```

Mounted at /content/drive

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
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	28079008
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	28079011
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	28079056
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	28079057
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	28079058
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	28079059
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	28079060

209880 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:

```
df=df.fillna(1)
df
```

Out[3]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2013-11-01 01:00:00	1.0	0.6	1.0	1.0	135.0	74.0	1.0	1.0	1.0	7.0	1.0	1.0	28079004
1	2013-11-01 01:00:00	1.5	0.5	1.3	1.0	71.0	83.0	2.0	23.0	16.0	12.0	1.0	8.3	28079008
2	2013-11-01 01:00:00	3.9	1.0	2.8	1.0	49.0	70.0	1.0	1.0	1.0	1.0	1.0	9.0	28079011
3	2013-11-01 01:00:00	1.0	0.5	1.0	1.0	82.0	87.0	3.0	1.0	1.0	1.0	1.0	1.0	28079016
4	2013-11-01 01:00:00	1.0	1.0	1.0	1.0	242.0	111.0	2.0	1.0	1.0	12.0	1.0	1.0	28079017

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
209875	2013-03-01 00:00:00	1.0	0.4	1.0	1.0	8.0	39.0	52.0	1.0	1.0	1.0	1.0	1.0	28079056
209876	2013-03-01 00:00:00	1.0	0.4	1.0	1.0	1.0	11.0	1.0	6.0	1.0	2.0	1.0	1.0	28079057
209877	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	2.0	4.0	75.0	1.0	1.0	1.0	1.0	1.0	28079058
209878	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	2.0	11.0	52.0	1.0	1.0	1.0	1.0	1.0	28079059
209879	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	1.0	10.0	75.0	3.0	1.0	1.0	1.0	1.0	28079060

209880 rows × 14 columns

In [4]:

```
df.columns
```

Out[4]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [6]:

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

Out[6]:

	CO	station
0	0.6	28079004
1	0.5	28079008
2	1.0	28079011
3	0.5	28079016
4	1.0	28079017
...	...	...
209875	0.4	28079056
209876	0.4	28079057
209877	1.0	28079058
209878	1.0	28079059

	CO	station
209878	1.0	28079059
209879	1.0	28079060

209880 rows × 2 columns

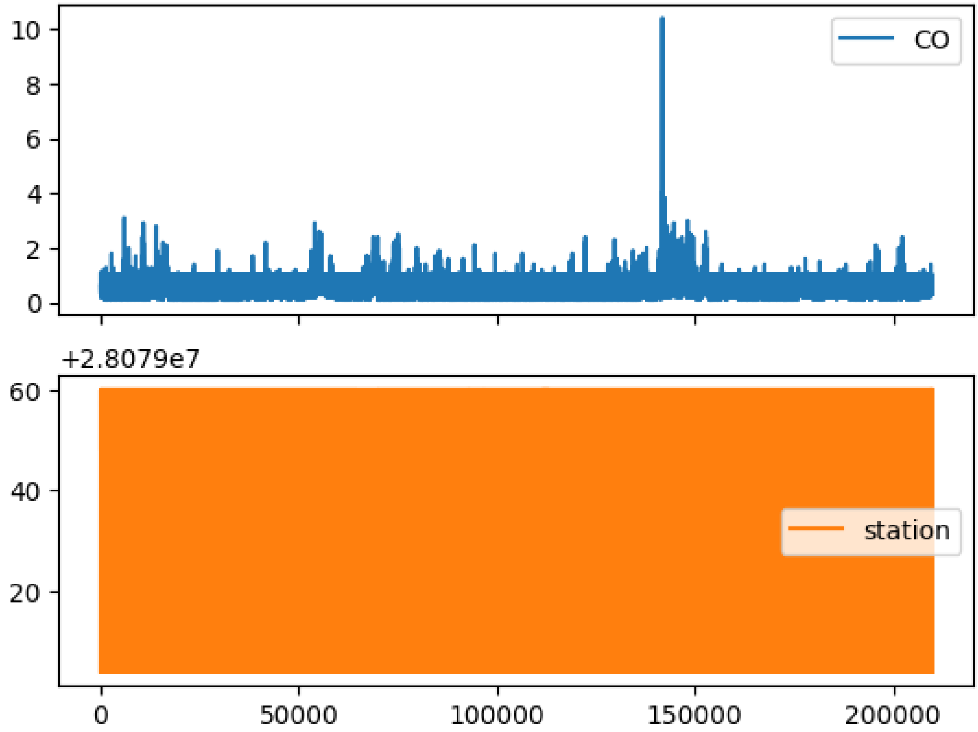
Line chart

In [7]:

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

Out[7]:

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



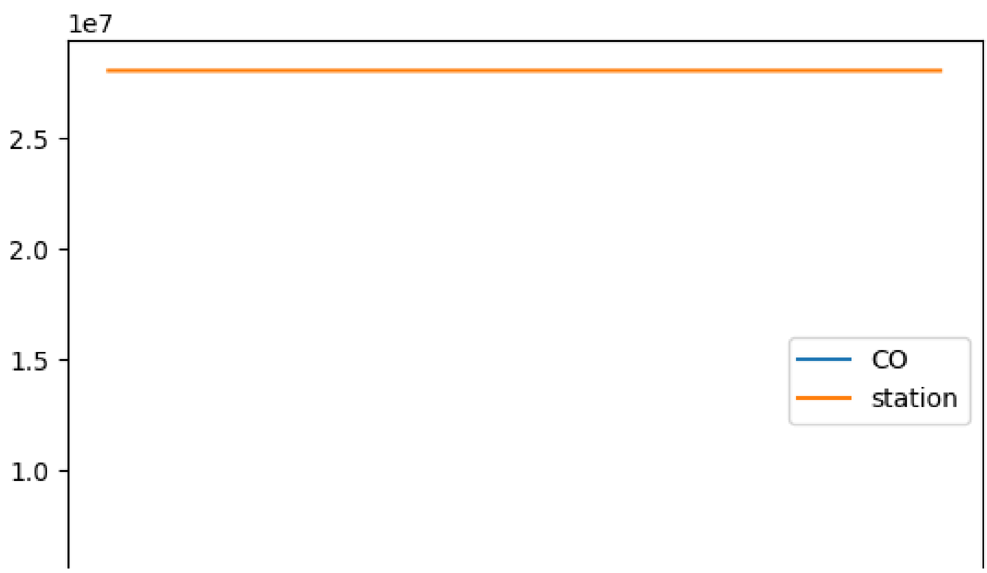
Line chart

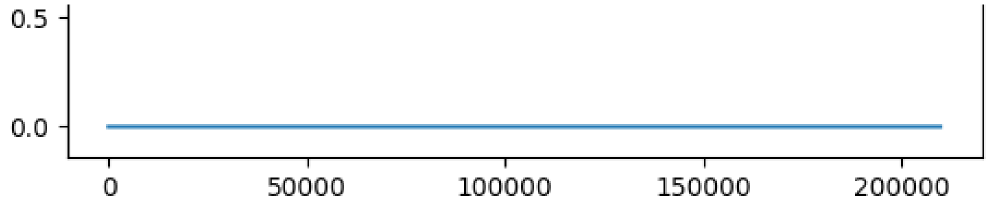
In [8]:

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

Out[8]:

```
<Axes: >
```





Bar chart

In [9]:

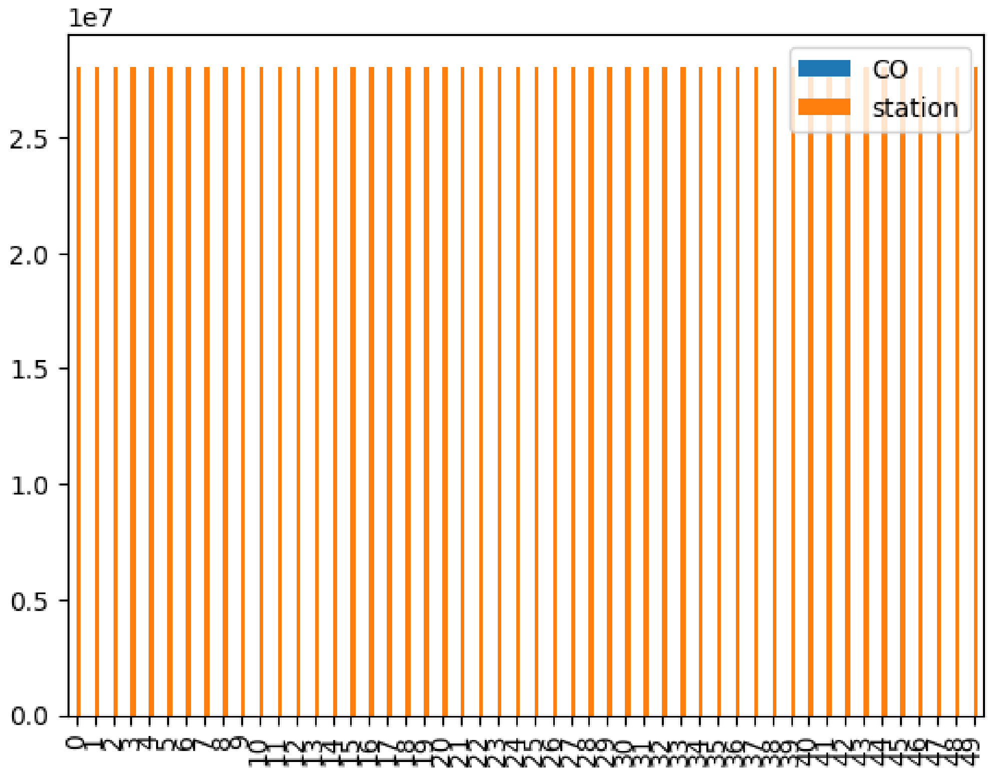
```
b=data[0:50]
```

In [10]:

```
b.plot.bar()
```

Out[10]:

```
<Axes: >
```



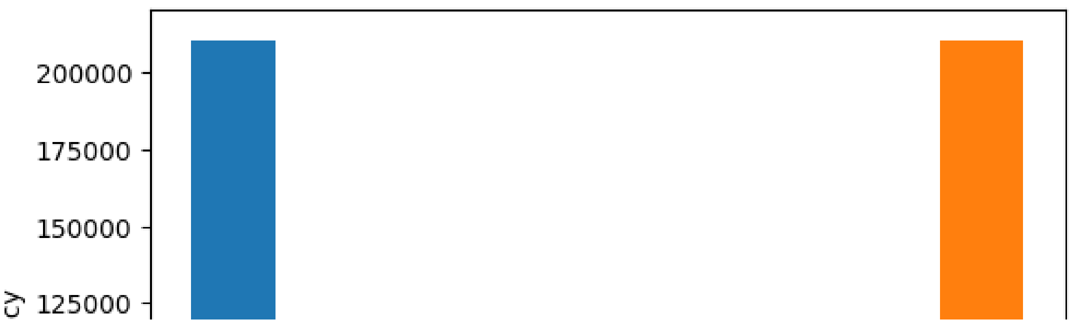
Histogram

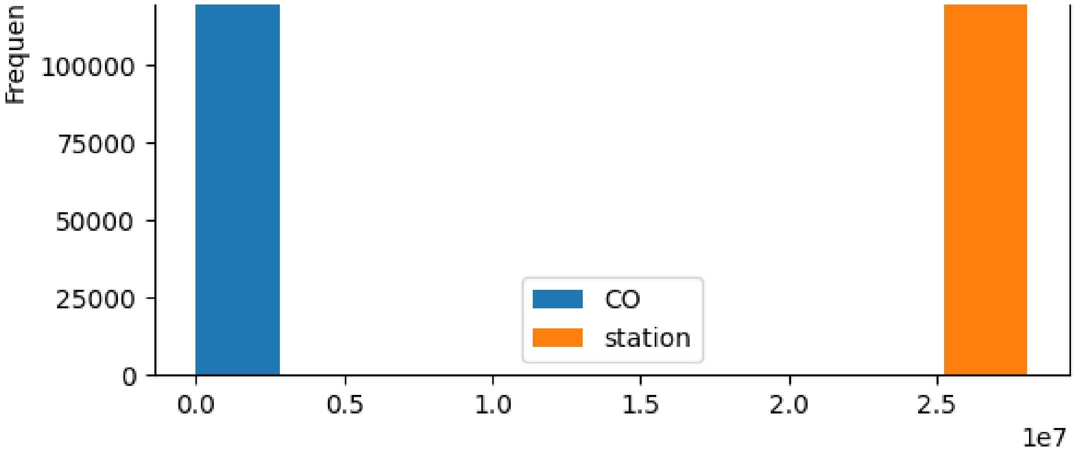
In [11]:

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

Out[11]:

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





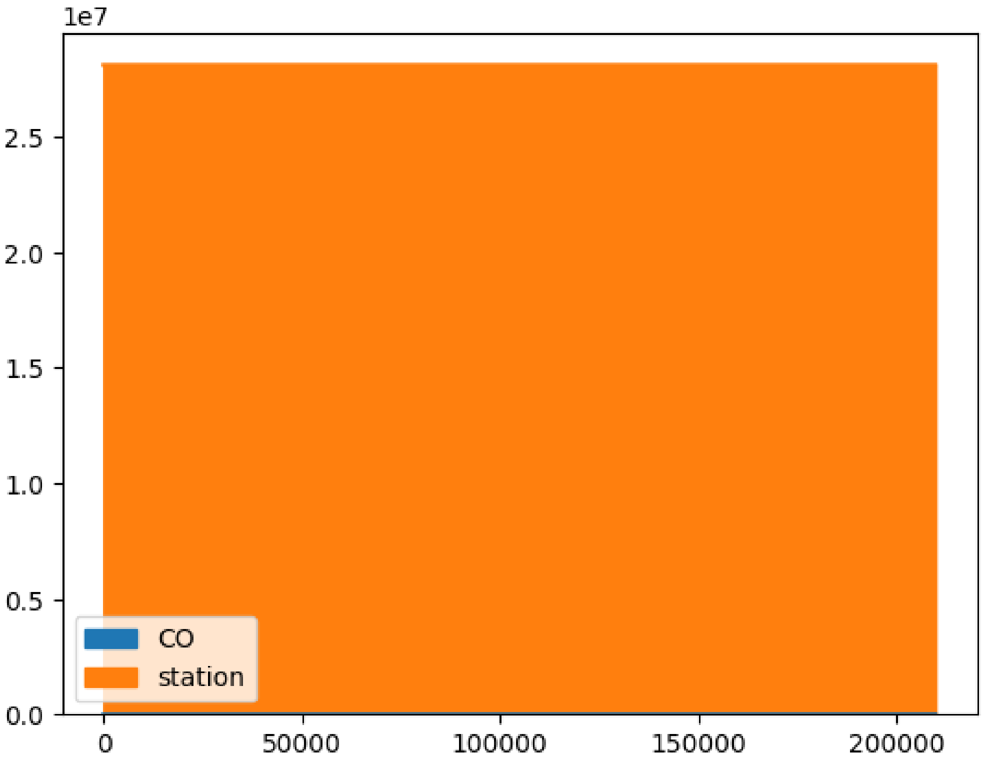
Area chart

In [12]:

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

Out[12]:

<Axes: >



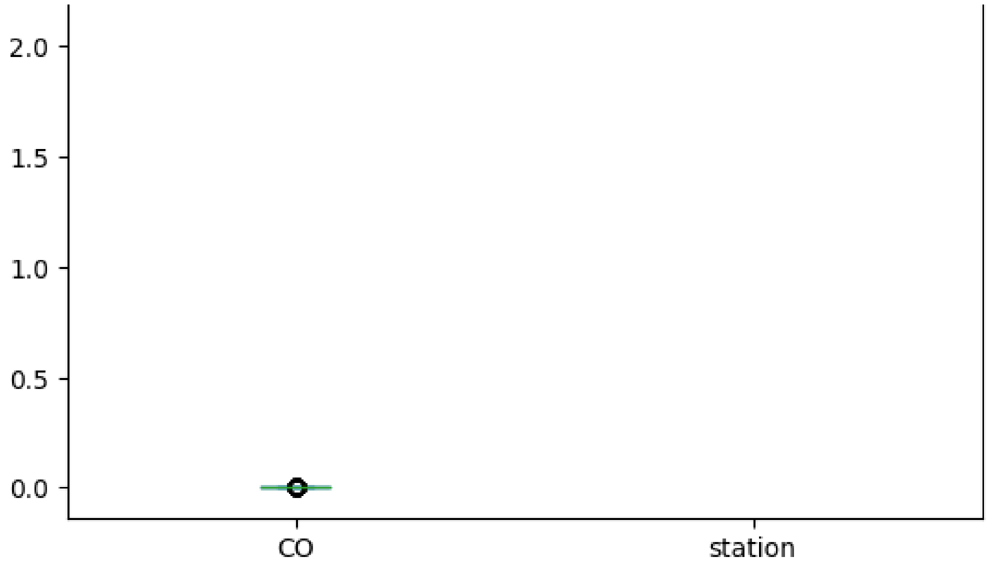
Box chart

In [13]:

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

Out[13]:

<Axes: >



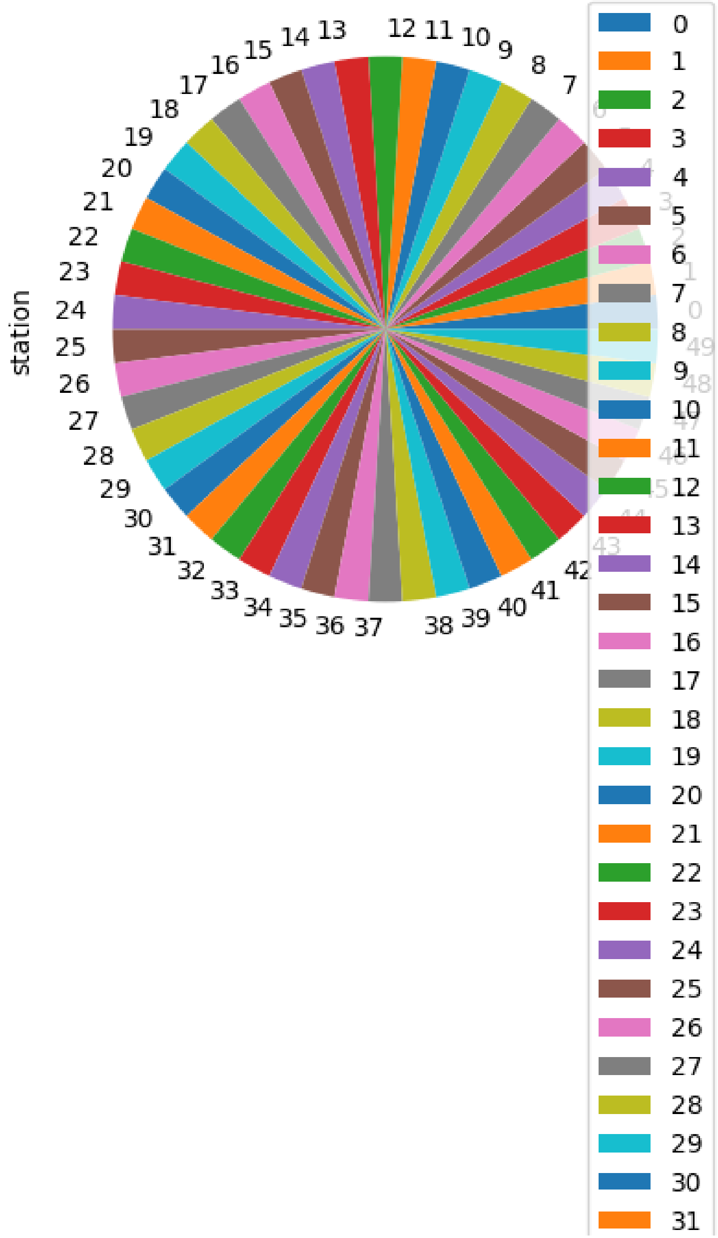
Pie chart

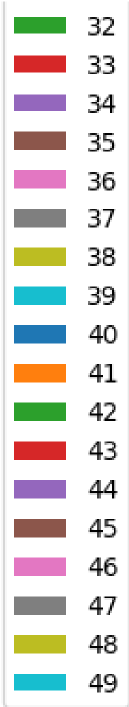
In [14]:

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

Out[14]:

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





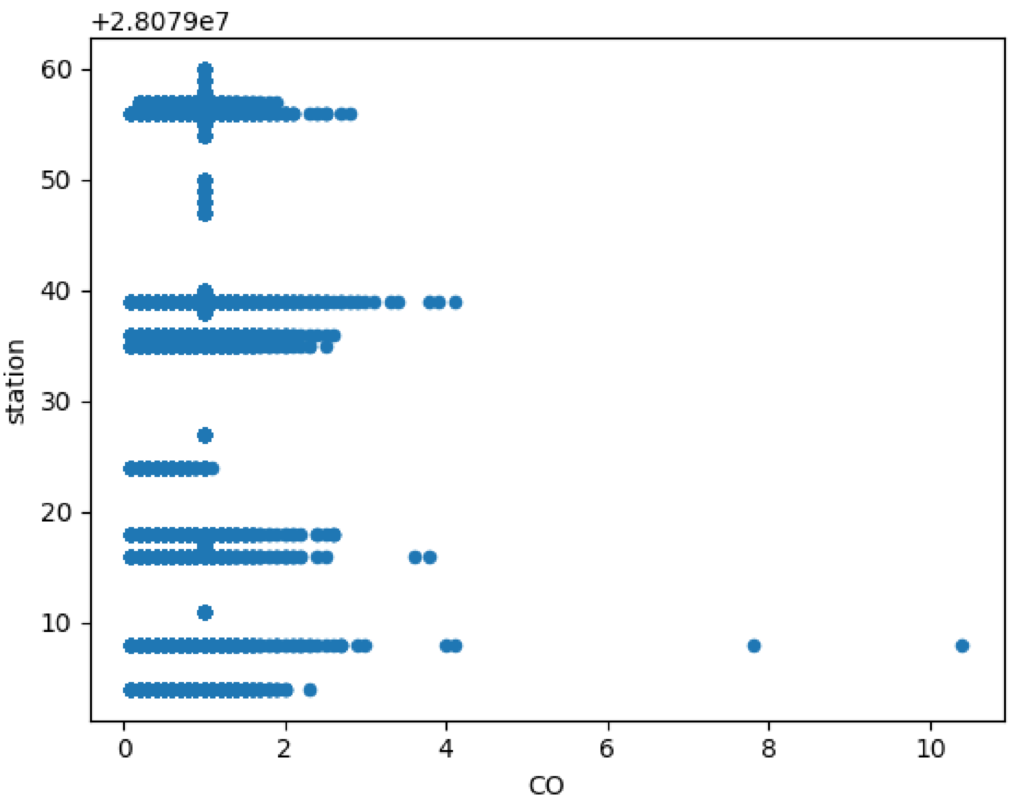
Scatter chart

In [15]:

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

Out[15]:

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



In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
```

```
 0   date     209880 non-null  object
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [17]:

```
df.describe()
```

Out[17]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PI
count	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000
mean	0.931014	0.721695	0.954744	0.900223	20.101401	34.586402	29.461235	9.636
std	0.430684	0.361528	0.301074	0.267139	44.319112	27.866588	35.362880	13.492
min	0.100000	0.100000	0.100000	0.040000	1.000000	1.000000	1.000000	1.000
25%	1.000000	0.300000	1.000000	1.000000	2.000000	14.000000	1.000000	1.000
50%	1.000000	1.000000	1.000000	1.000000	5.000000	27.000000	8.000000	1.000
75%	1.000000	1.000000	1.000000	1.000000	17.000000	48.000000	54.000000	14.000
max	12.100000	10.400000	11.800000	1.000000	1081.000000	388.000000	226.000000	232.000

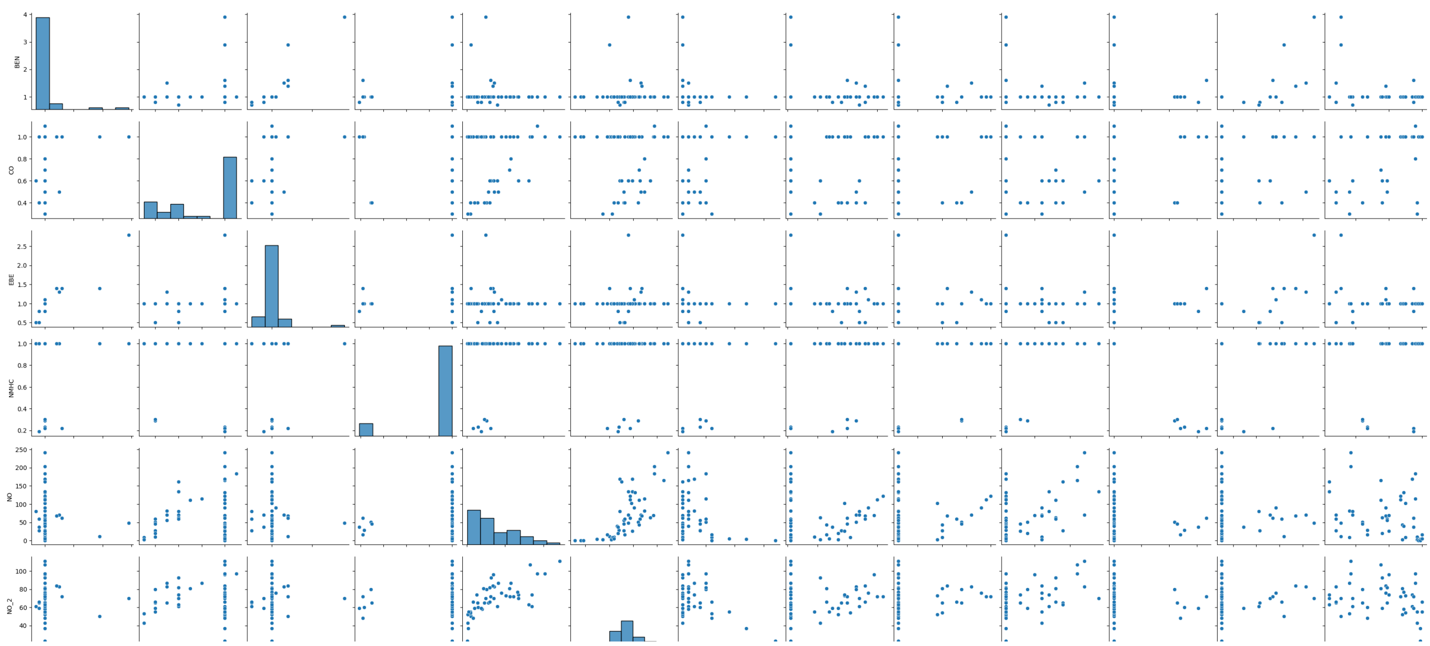
EDA AND VISUALIZATION

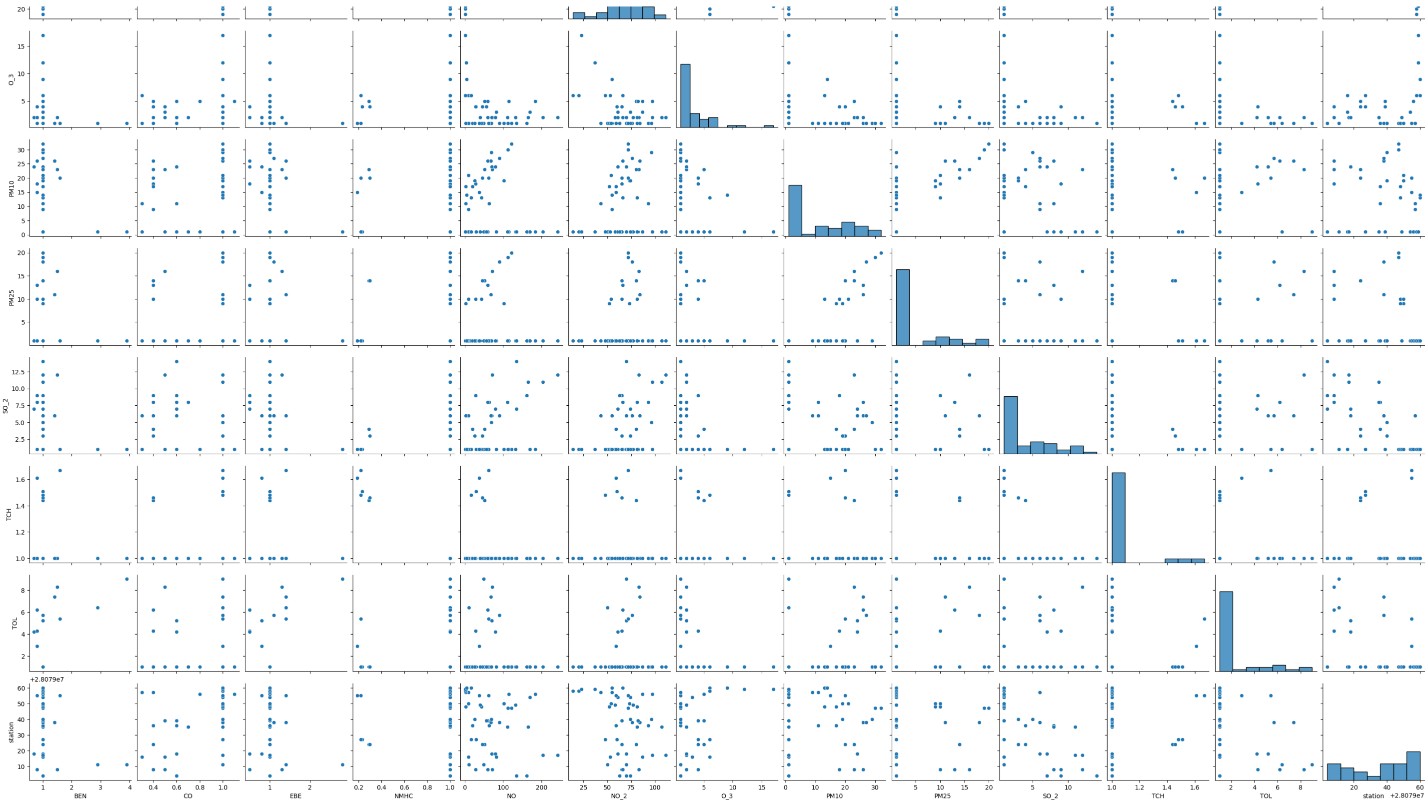
In [18]:

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

Out[18]:

<seaborn.axisgrid.PairGrid at 0x7d1a405fcbb0>





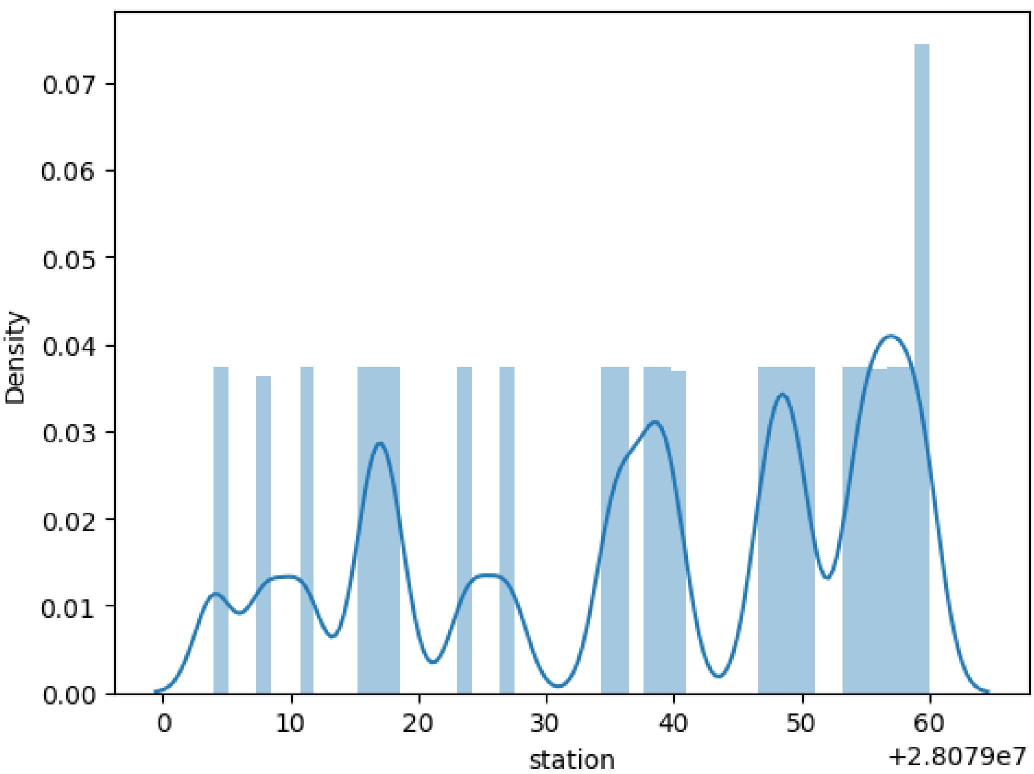
In [19]:

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

```
<ipython-input-19-6e2460d4583e>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

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

Out[19]:

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



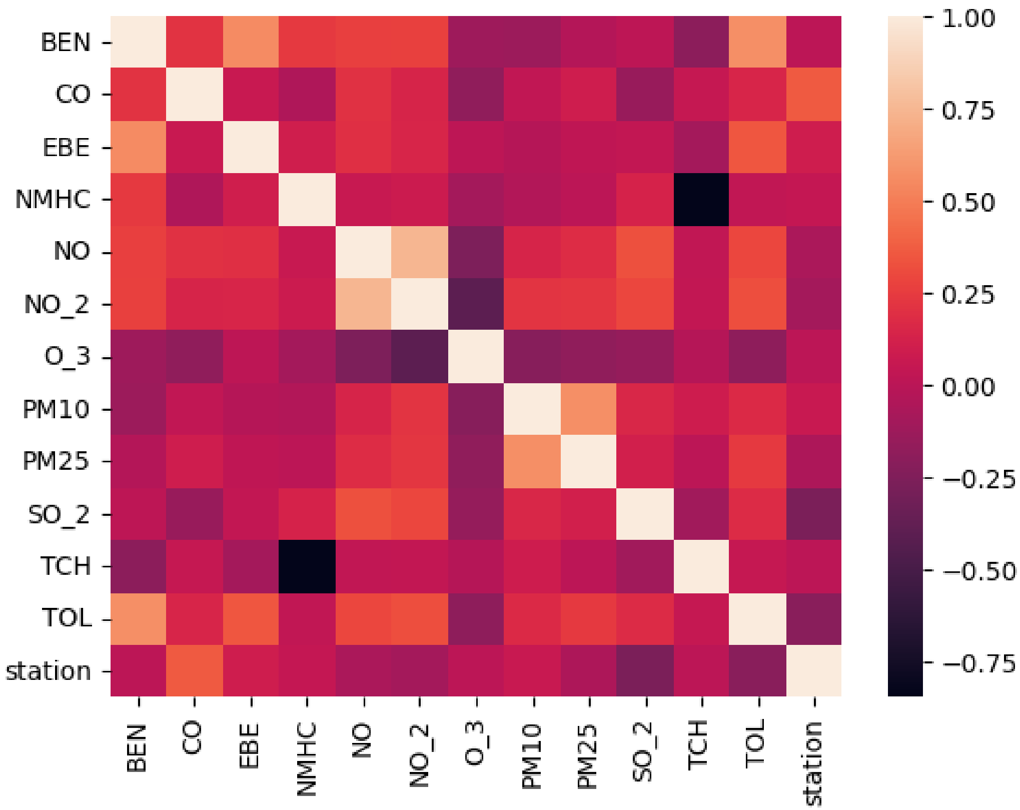
In [20]:

```
sns.heatmap(df.corr())
```

```
<ipython-input-20-aa4f4450a243>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr())
```

Out[20]:

```
<Axes: >
```



TO TRAIN THE MODEL AND MODEL BULDING

In [21]:

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

In [22]:

```
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 [23]:

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

Out[23]:

```
▾ LinearRegression
LinearRegression()
```

In [24]:

```
lr.intercept_
```

Out[24]:

```
28078977.103350364
```

In [25]:

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

Out[25]:

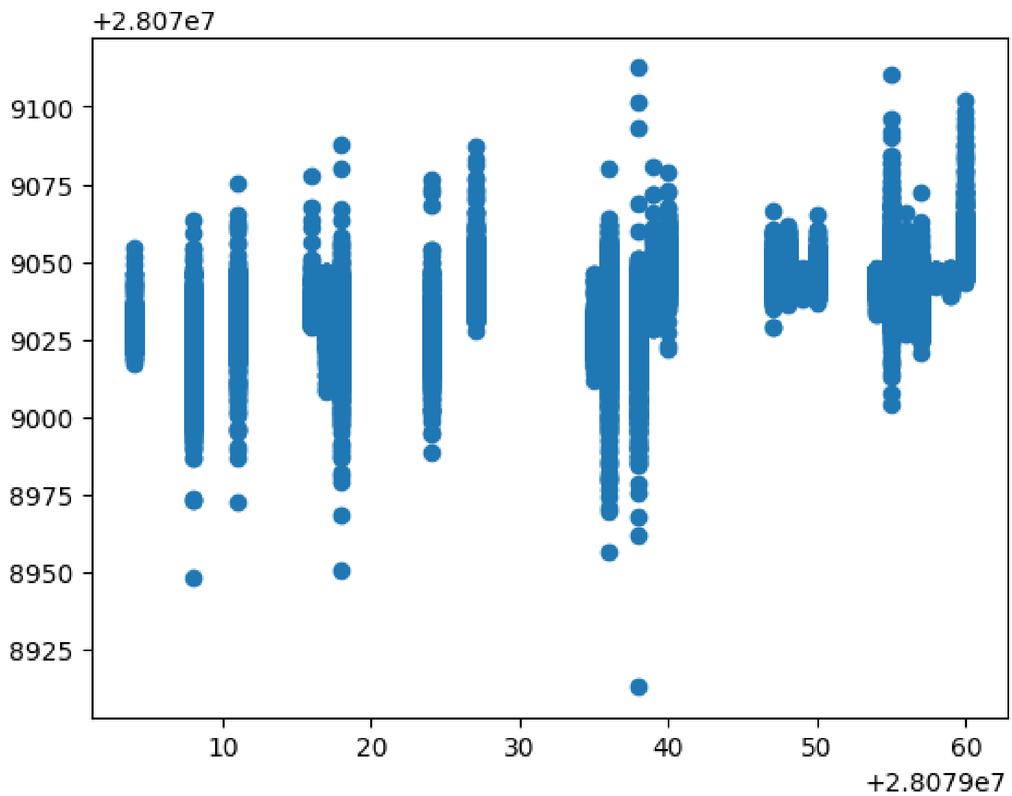
	Co-efficient
BEN	1.675295
CO	18.812592
EBE	10.008459
NMHC	18.092884
NO	-0.009336
NO_2	-0.039330
O_3	0.008942
PM10	0.279728
PM25	-0.371620
SO_2	-0.945909
TCH	25.161095
TOL	-3.445599

In [26]:

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

Out[26]:

```
<matplotlib.collections.PathCollection at 0x7d1a7f509660>
```



ACCURACY

In [27]:

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

Out[27]:

```
0.30543236748624947
```

In [28]:

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

Out[28]:

```
0.30799712227942355
```

Ridge and Lasso

In [29]:

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

In [30]:

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

Out[30]:

```
▾ Ridge
Ridge(alpha=10)
```

Accuracy(Ridge)

In [31]:

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

Out[31]:

```
0.3054128646053008
```

In [32]:

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

Out[32]:

```
0.3079943077378543
```

In [33]:

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

Out[33]:

```
▾ Lasso
Lasso(alpha=10)
```

In [34]:

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

Out[34]:

```
0.0451475937957041
```

Accuracy(Lasso)

In [35]:

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

Out[35]:

```
0.04531888865719813
```

In [36]:

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

Out[36]:

```
▾ ElasticNet
ElasticNet()
```

In [37]:

```
en.coef_
```

Out[37]:

```
array([ 0.33177767,  2.64130733,  0.45537669,  0.        ,  0.03569569,
       -0.05608388, -0.01983977,  0.23701462, -0.35214909, -1.35812245,
       -0.        , -1.5728494 ])
```

In [38]:

```
en.intercept_
```

Out[38]:

```
28079041.198403195
```

In [39]:

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

In [40]:

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

Out[40]:

```
0.1609692866422956
```

Evaluation Metrics

In [41]:

```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
13.81701650598825
259.6923434931426
16.11497264946927
```

Logistic Regression

In [42]:

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

In [43]:

```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']][0:50]
target_vector=df[ 'station'][0:50]
```

In [44]:

```
feature_matrix.shape
```

Out[44]:

```
(50, 12)
```

In [45]:

```
target_vector.shape
```

Out[45]:

```
(50,)
```

In [46]:

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

In [47]:

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

In [48]:

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

Out[48]:

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

In [49]:

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

In [50]:

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

```
[28079038]
```

In [51]:

```
logr.classes_
```

Out[51]:

```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060])
```

In [52]:

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

Out[52]:

```
0.9
```

In [53]:

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

Out[53]:

```
3.2424612943857784e-12
```

In [54]:

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

Out[54]:

```
array([[3.24246129e-12, 1.98879790e-01, 2.68546600e-11, 9.18595327e-17,
        2.85595544e-05, 1.13436010e-07, 1.11387803e-06, 1.83744227e-14,
        4.41788720e-09, 2.60721644e-13, 8.01089957e-01, 6.75335556e-16,
        4.31028393e-12, 3.50004736e-07, 1.45256021e-14, 1.06842862e-19,
        1.45341132e-13, 1.18084324e-14, 1.11311169e-07, 4.24582423e-10,
        4.14877181e-17, 1.07641028e-18, 8.68787036e-11, 5.39240034e-11]])
```

Random Forest

In [55]:

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

In [56]:

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

Out[56]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [57]:

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

In [58]:

```
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[58]:

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
▸ RandomForestClassifier
```

In [59]:

```
grid_search.best_score_
```

Out[59]:

```
0.7496052165863487
```

In [60]:

```python
rfc_best=grid_search.best_estimator_
```

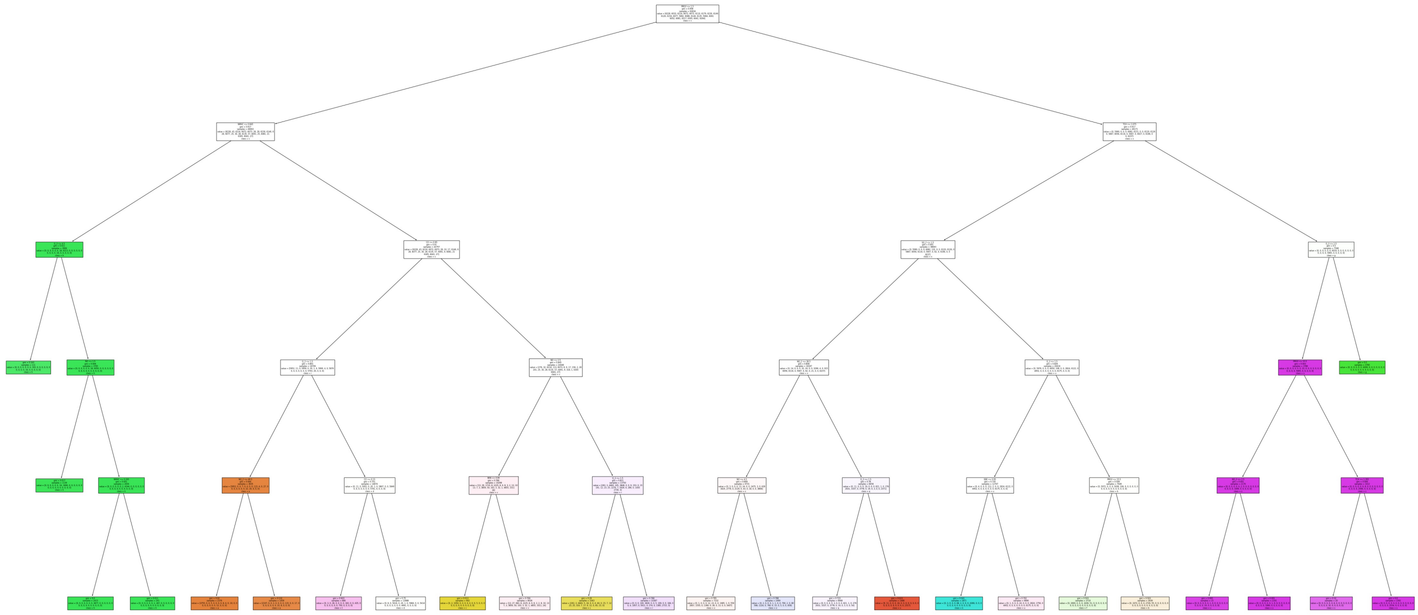
In [61]:

```python
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','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'],filled=True)
```

Out[61]:

```
[Text(0.483695652173913, 0.9166666666666666, 'PM10 <= 1.5\ngini = 0.958\nsamples = 92834\nvalue = [6228, 6033, 6216, 6072, 6073, 6110, 6179, 6230, 6146\n6126, 6154, 6077, 5892, 6086, 6144, 6135, 5964, 6091\n6052, 6081, 6217, 6305, 6041, 6264]\nclass = v'),
 Text(0.17391304347826086, 0.75, 'NMHC <= 0.845\ngini = 0.917\nsamples = 46663\nvalue = [6228, 43, 6216, 6072, 6073, 28, 28, 6230, 6146, 6\n28, 6077, 25, 30, 28, 6135, 57, 6091, 25, 6081, 21\n6305, 6041, 27]\nclass = v'),
 Text(0.043478260869565216, 0.5833333333333334, 'O_3 <= 4.5\ngini = 0.011\nsamples = 3906\nvalue = [0, 0, 0, 0, 0, 0, 18, 6213, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 16, 0, 0, 0, 0, 0]\nclass = h'),
 Text(0.021739130434782608, 0.4166666666666667, 'gini = 0.163\nsamples = 111\nvalue = [0, 0, 0, 0, 0, 0, 0, 163, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 16, 0, 0, 0, 0, 0]\nclass = h'),
 Text(0.06521739130434782, 0.4166666666666667, 'NO <= 1.5\ngini = 0.006\nsamples = 3795\nvalue = [0, 0, 0, 0, 0, 0, 18, 6050, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(0.043478260869565216, 0.25, 'gini = 0.017\nsamples = 1139\nvalue = [0, 0, 0, 0, 0, 0, 16, 1806, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(0.08695652173913043, 0.25, 'NMHC <= 0.265\ngini = 0.001\nsamples = 2656\nvalue = [0, 0, 0, 0, 0, 0, 2, 4244, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(0.06521739130434782, 0.08333333333333333, 'gini = 0.0\nsamples = 2415\nvalue = [0, 0, 0, 0, 0, 0, 0, 3877, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(0.10869565217391304, 0.08333333333333333, 'gini = 0.011\nsamples = 241\nvalue = [0, 0, 0, 0, 0, 0, 2, 367, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(0.30434782608695654, 0.5833333333333334, 'CO <= 0.95\ngini = 0.91\nsamples = 42757\nvalue = [6228, 43, 6216, 6072, 6073, 28, 10, 17, 6146, 6\n28, 6077, 25, 30, 28, 6135, 57, 6091, 9, 6081, 21\n6305, 6041, 27]\nclass = v'),
 Text(0.21739130434782608, 0.4166666666666667, 'O_3 <= 1.5\ngini = 0.801\nsamples = 18709\nvalue = [5952, 11, 0, 5959, 0, 20, 1, 0, 5990, 4, 0, 5876\n0, 0, 0, 0, 0, 0, 5763, 20, 0, 0, 0]\nclass = i'),
 Text(0.17391304347826086, 0.25, 'NO_2 <= 46.5\ngini = 0.064\nsamples = 3835\nvalue = [5952, 0, 0, 7, 0, 0, 0, 0, 123, 4, 0, 27, 0\n0, 0, 0, 0, 0, 22, 20, 0, 0, 0]\nclass = a')
,
 Text(0.15217391304347827, 0.08333333333333333, 'gini = 0.02\nsamples = 2376\nvalue = [3755, 0, 0, 5, 0, 0, 0, 0, 8, 4, 0, 10, 0, 0\n0, 0, 0, 0, 0, 12, 0, 0, 0]\nclass = a'),
 Text(0.1956521739130435, 0.08333333333333333, 'gini = 0.132\nsamples = 1459\nvalue = [2197, 0, 0, 2, 0, 0, 0, 0, 115, 0, 0, 17, 0\n0, 0, 0, 0, 0, 22, 8, 0, 0, 0]\nclass = a'),
 Text(0.2608695652173913, 0.25, 'CO <= 0.15\ngini = 0.751\nsamples = 14874\nvalue = [0, 11, 0, 5952, 0, 20, 1, 0, 5867, 0, 0, 5849\n0, 0, 0, 0, 0, 0, 5741, 0, 0, 0, 0]\nclass = d'),
 Text(0.2391304347826087, 0.08333333333333333, 'gini = 0.603\nsamples = 886\nvalue = [0, 3, 0, 36, 0, 0, 0, 0, 183, 0, 0, 435, 0\n0, 0, 0, 0, 0, 750, 0, 0, 0, 0]\nclass = t'),
 Text(0.2826086956521739, 0.08333333333333333, 'gini = 0.75\nsamples = 13988\nvalue = [0, 8, 0, 5916, 0, 20, 1, 0, 5684, 0, 0, 5414\n0, 0, 0, 0, 0, 0, 4991, 0, 0, 0, 0]\nclass = d'),
 Text(0.391304347826087, 0.4166666666666667, 'NO <= 2.5\ngini = 0.845\nsamples = 24048\nvalue = [276, 32, 6216, 113, 6073, 8, 9, 17, 156, 2, 28\n201, 25, 30, 28, 6135, 57, 6091, 9, 318, 1, 6305\n6041, 27]\nclass = v'),
 Text(0.34782608695652173, 0.25, 'BEN <= 0.95\ngini = 0.786\nsamples = 10298\nvalue = [13, 29, 1733, 8, 2225, 8, 9, 8, 3, 2, 13, 10\n13, 7, 3, 3859, 50, 163, 3, 32, 1, 4855, 3311\n24]\nclass = v'),
 Text(0.32608695652173914, 0.08333333333333333, 'gini = 0.013\nsamples = 662\nvalue = [0, 2, 1050, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = c'),
 Text(0.3695652173913043, 0.08333333333333333, 'gini = 0.766\nsamples = 9636\nvalue = [13, 27, 683, 8, 2225, 8, 9, 8, 3, 2, 8, 10, 13\n7, 3, 3859, 50, 163, 3, 32, 1, 4855, 3311,
```

```
24]\nclass = v'),
 Text(0.43478260869565216, 0.25, 'O_3 <= 1.5\ngini = 0.821\nsamples = 13750\nvalue = [263
, 3, 4483, 105, 3848, 0, 0, 9, 153, 0, 15\n191, 12, 23, 25, 2276, 7, 5928, 6, 286, 0, 145
0\n2730, 3]\nclass = r'),
 Text(0.41304347826086957, 0.08333333333333333, 'gini = 0.29\nsamples = 3363\nvalue = [26
3, 3, 4483, 5, 18, 0, 0, 4, 44, 0, 15, 7, 12\n23, 25, 319, 7, 17, 6, 12, 0, 65, 15, 0]\ncl
ass = c'),
 Text(0.45652173913043476, 0.08333333333333333, 'gini = 0.768\nsamples = 10387\nvalue = [
0, 0, 0, 100, 3830, 0, 0, 5, 109, 0, 0, 184, 0\n0, 0, 1957, 0, 5911, 0, 274, 0, 1385, 271
5, 3]\nclass = r'),
 Text(0.7934782608695652, 0.75, 'TCH <= 1.075\ngini = 0.917\nsamples = 46171\nvalue = [0,
5990, 0, 0, 0, 6082, 6151, 0, 0, 6120, 6126\n0, 5867, 6056, 6116, 0, 5907, 0, 6027, 0, 61
96, 0\n0, 6237]\nclass = x'),
 Text(0.6521739130434783, 0.5833333333333334, 'SO_2 <= 1.5\ngini = 0.901\nsamples = 38603
\nvalue = [0, 5990, 0, 0, 0, 6082, 132, 0, 0, 6120, 6126, 0\n5867, 6056, 6116, 0, 5907, 0
, 62, 0, 6196, 0, 0\n6237]\nclass = x'),
 Text(0.5652173913043478, 0.4166666666666667, 'NO_2 <= 28.5\ngini = 0.806\nsamples = 1818
7\nvalue = [0, 14, 0, 0, 0, 32, 24, 0, 0, 3296, 4, 0, 915\n6056, 6116, 0, 5907, 0, 62, 0,
21, 0, 0, 6237]\nclass = x'),
 Text(0.5217391304347826, 0.25, 'NO <= 4.5\ngini = 0.809\nsamples = 9731\nvalue = [0, 3,
0, 0, 0, 13, 24, 0, 0, 2475, 3, 0, 639\n3425, 2779, 0, 2129, 0, 43, 0, 18, 0, 0, 3866]\ncl
ass = x'),
 Text(0.5, 0.08333333333333333, 'gini = 0.795\nsamples = 7232\nvalue = [0, 1, 0, 0, 0, 12
, 23, 0, 0, 1685, 1, 0, 559\n2857, 1555, 0, 1369, 0, 28, 0, 13, 0, 0, 3407]\nclass = x'),
 Text(0.5434782608695652, 0.08333333333333333, 'gini = 0.788\nsamples = 2499\nvalue = [0,
2, 0, 0, 0, 1, 1, 0, 0, 790, 2, 0, 80\n568, 1224, 0, 760, 0, 15, 0, 5, 0, 0, 459]\nclass =
o'),
 Text(0.6086956521739131, 0.25, 'O_3 <= 1.5\ngini = 0.78\nsamples = 8456\nvalue = [0, 11,
0, 0, 0, 19, 0, 0, 0, 821, 1, 0, 276\n2631, 3337, 0, 3778, 0, 19, 0, 3, 0, 0, 2371]\nclass
= q'),
 Text(0.5869565217391305, 0.08333333333333333, 'gini = 0.723\nsamples = 6968\nvalue = [0,
0, 0, 0, 0, 3, 0, 0, 0, 821, 1, 0, 276\n2631, 3337, 0, 3778, 0, 19, 0, 3, 0, 0, 54]\nclass
= q'),
 Text(0.6304347826086957, 0.08333333333333333, 'gini = 0.023\nsamples = 1488\nvalue = [0,
11, 0, 0, 0, 16, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 2317]\nclass = x'),
 Text(0.7391304347826086, 0.4166666666666667, 'O_3 <= 1.5\ngini = 0.826\nsamples = 20416\
nvalue = [0, 5976, 0, 0, 0, 6050, 108, 0, 0, 2824, 6122, 0\n4952, 0, 0, 0, 0, 0, 0, 0, 617
5, 0, 0, 0]\nclass = u'),
 Text(0.6956521739130435, 0.25, 'EBE <= 0.95\ngini = 0.735\nsamples = 12767\nvalue = [0,
4, 0, 0, 0, 112, 2, 0, 0, 2824, 6122, 0\n4952, 0, 0, 0, 0, 0, 0, 0, 6175, 0, 0, 0]\nclass
= u'),
 Text(0.6739130434782609, 0.08333333333333333, 'gini = 0.041\nsamples = 2871\nvalue = [0,
3, 0, 0, 0, 88, 2, 0, 0, 0, 4366, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = k'),
 Text(0.717391304347826, 0.08333333333333333, 'gini = 0.702\nsamples = 9896\nvalue = [0,
1, 0, 0, 0, 24, 0, 0, 0, 2824, 1756, 0\n4952, 0, 0, 0, 0, 0, 0, 0, 6175, 0, 0, 0]\nclass =
u'),
 Text(0.782608695652174, 0.25, 'PM10 <= 15.5\ngini = 0.509\nsamples = 7649\nvalue = [0, 5
972, 0, 0, 0, 5938, 106, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(0.7608695652173914, 0.08333333333333333, 'gini = 0.503\nsamples = 3710\nvalue = [0,
2600, 0, 0, 0, 3209, 51, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = f'),
 Text(0.8043478260869565, 0.08333333333333333, 'gini = 0.503\nsamples = 3939\nvalue = [0,
3372, 0, 0, 0, 2729, 55, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(0.9347826086956522, 0.5833333333333334, 'O_3 <= 1.5\ngini = 0.5\nsamples = 7568\nva
lue = [0, 0, 0, 0, 0, 0, 6019, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5965, 0, 0, 0, 0, 0]\nclas
s = g'),
 Text(0.9130434782608695, 0.4166666666666667, 'PM10 <= 15.5\ngini = 0.003\nsamples = 3788
\nvalue = [0, 0, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 5965, 0, 0, 0, 0, 0]\ncl
ass = s'),
 Text(0.8695652173913043, 0.25, 'NO_2 <= 2.5\ngini = 0.001\nsamples = 2176\nvalue = [0, 0
, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3399, 0, 0, 0, 0, 0]\nclass = s'),
 Text(0.8478260869565217, 0.08333333333333333, 'gini = 0.095\nsamples = 15\nvalue = [0, 0
, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 19, 0, 0, 0, 0, 0]\nclass = s'),
 Text(0.8913043478260869, 0.08333333333333333, 'gini = 0.001\nsamples = 2161\nvalue = [0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3380, 0, 0, 0, 0, 0]\nclass = s'),
 Text(0.9565217391304348, 0.25, 'TCH <= 1.355\ngini = 0.006\nsamples = 1612\nvalue = [0,
0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2566, 0, 0, 0, 0, 0]\nclass = s'),
 Text(0.9347826086956522, 0.08333333333333333, 'gini = 0.32\nsamples = 20\nvalue = [0, 0,
0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 32, 0, 0, 0, 0, 0]\nclass = s'),
 Text(0.9782608695652174, 0.08333333333333333, 'gini = 0.0\nsamples = 1592\nvalue = [0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2534, 0, 0, 0, 0, 0]\nclass = s'),
 Text(0.9565217391304348, 0.4166666666666667, 'gini = 0.0\nsamples = 3780\nvalue = [0, 0,
0, 0, 0, 0, 6009, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g')]
```



Conclusion

Accuracy

In [62]:

```
print("Linear Regression:",lr.score(x_test,y_test))
print("Ridge Regression:",rr.score(x_test,y_test))
print("Lasso Regression",la.score(x_test,y_test))
print("ElasticNet Regression:",en.score(x_test,y_test))
print("Logistic Regression:",logr.score(fs,target_vector))
print("Random Forest:",grid_search.best_score_)
```

```
Linear Regression: 0.30543236748624947
Ridge Regression: 0.3054128646053008
Lasso Regression 0.04531888865719813
ElasticNet Regression: 0.1609692866422956
Logistic Regression: 0.9
Random Forest: 0.7496052165863487
```

Logistic Regression is suitable for this dataset