20104028

Heamnath N

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [2]:
```
df=pd.read_csv("madrid_2013.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	stati
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	280790
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	280790
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	280790
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	280790
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	280790
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	280790
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	280790
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	280790

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	statio
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	280790
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	280790

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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	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

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
	00:00:00													

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

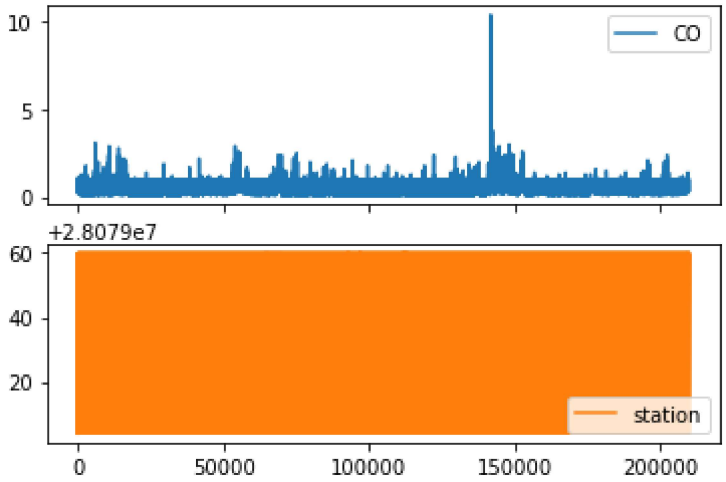
	CO	station
209879	1.0	28079060

209880 rows × 2 columns

Line chart

In [7]:
```
data.plot.line(subplots=True)
```

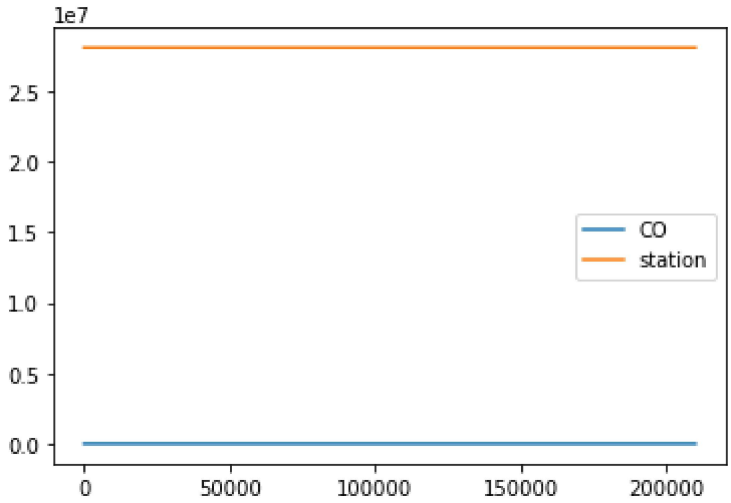
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:
```
data.plot.line()
```

Out[8]:
```
<AxesSubplot:>
```

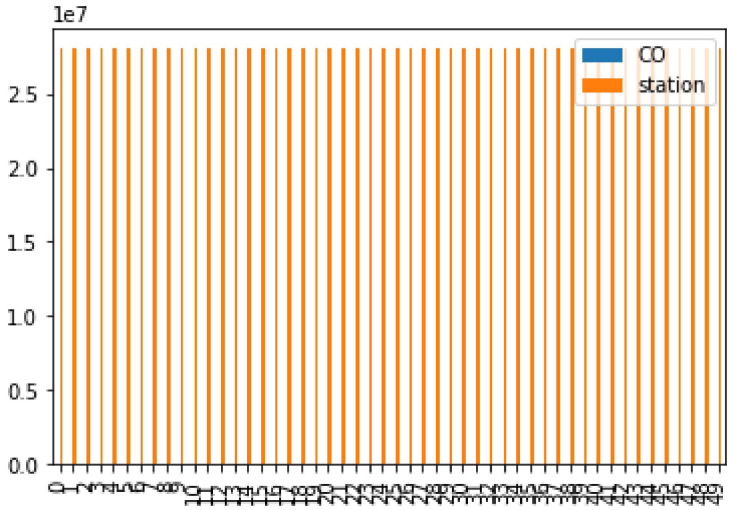


Bar chart

In [9]:
```
b=data[0:50]
```

In [10]:
```
b.plot.bar()
```

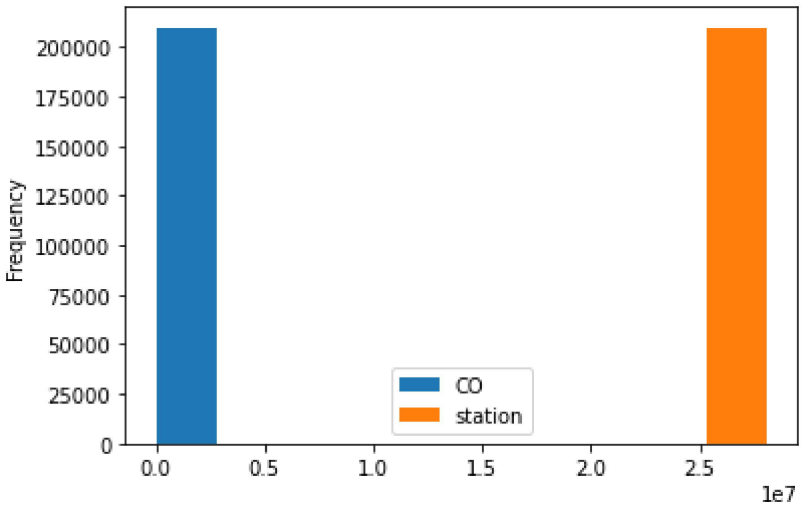
Out[10]: <AxesSubplot:>



Histogram

In [11]:
```
data.plot.hist()
```

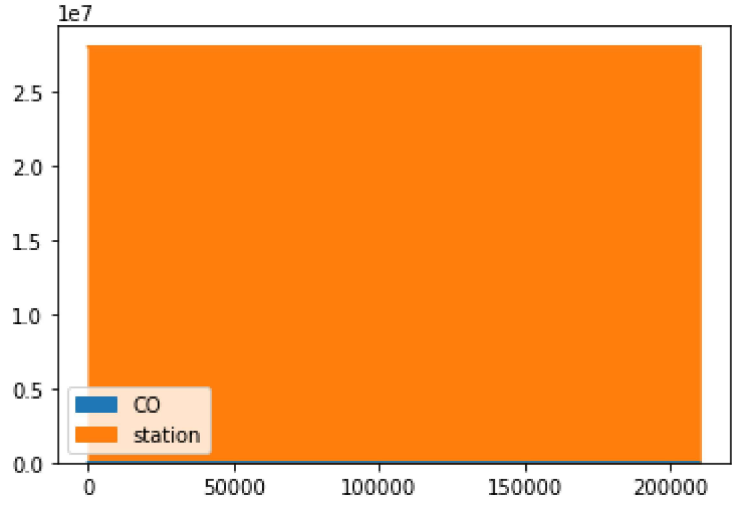
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:
```
data.plot.area()
```

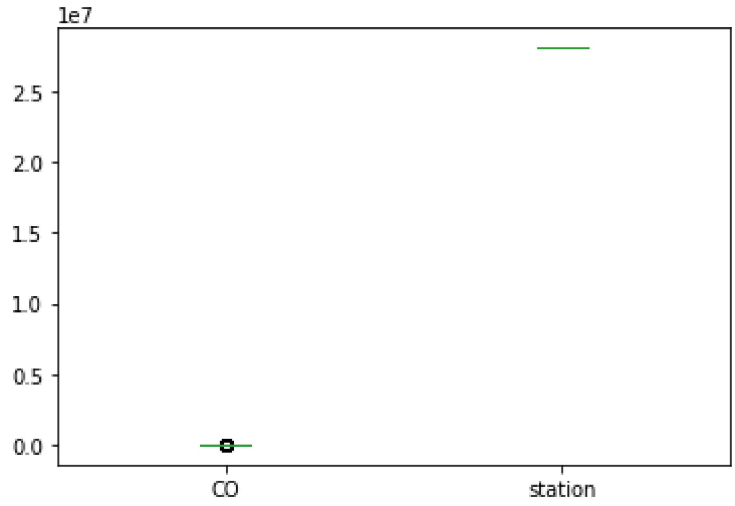
Out[12]: <AxesSubplot:>



Box chart

In [13]:
```
data.plot.box()
```

Out[13]: <AxesSubplot:>



Pie chart

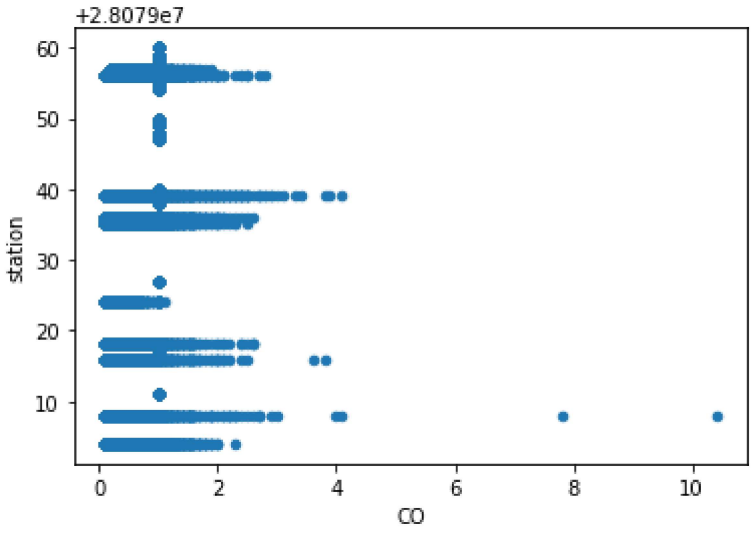
In [14]:
```
b.plot.pie(y='station' )
```

Out[14]: <AxesSubplot:ylabel='station'>

Scatter chart

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'>
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]:

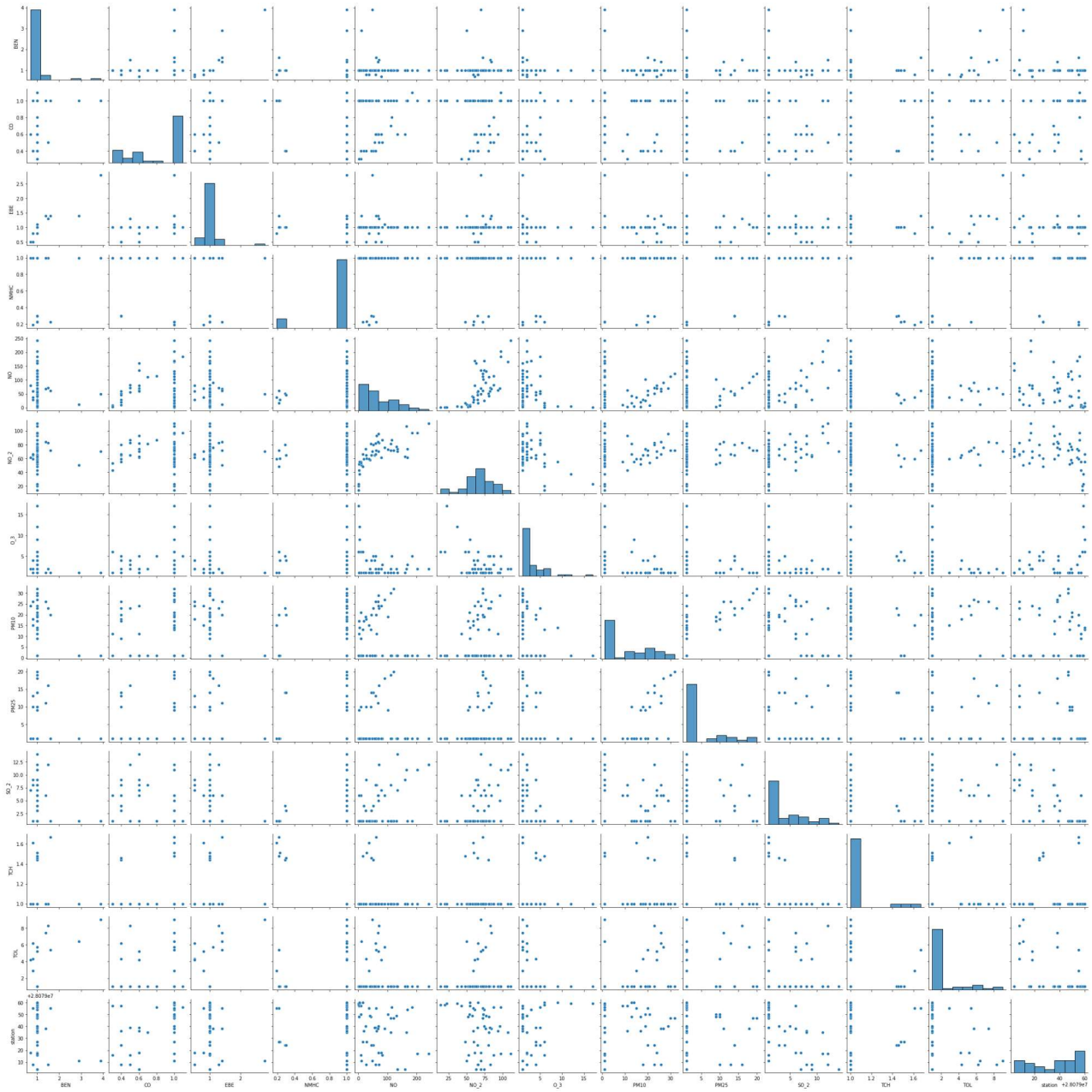
	BEN	CO	EBE	NMHC	NO	NO_2	
count	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	20988
mean	0.931014	0.721695	0.954744	0.900223	20.101401	34.586402	2
std	0.430684	0.361528	0.301074	0.267139	44.319112	27.866588	3
min	0.100000	0.100000	0.100000	0.040000	1.000000	1.000000	
25%	1.000000	0.300000	1.000000	1.000000	2.000000	14.000000	
50%	1.000000	1.000000	1.000000	1.000000	5.000000	27.000000	
75%	1.000000	1.000000	1.000000	1.000000	17.000000	48.000000	5

	BEN	CO	EBE	NMHC	NO	NO_2	
max	12.100000	10.400000	11.800000	1.000000	1081.000000	388.000000	22

EDA AND VISUALIZATION

In [18]:
```
sns.pairplot(df[0:50])
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x1e7b66b2160>

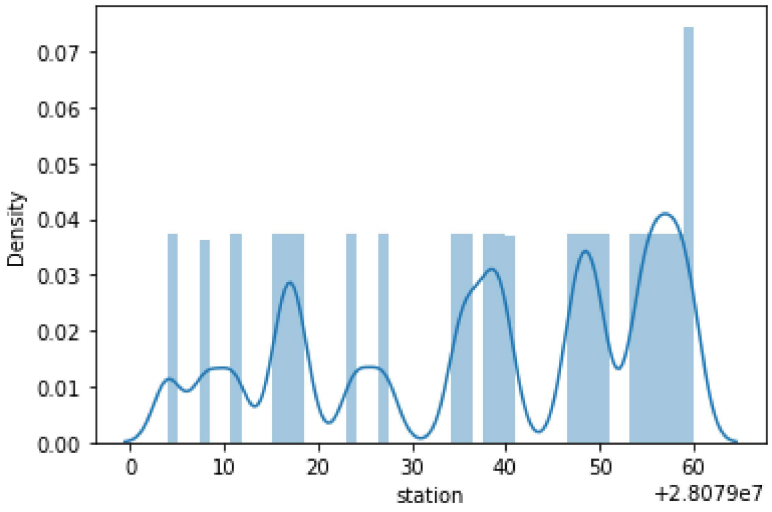


In [19]:
```
sns.distplot(df['station'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
```

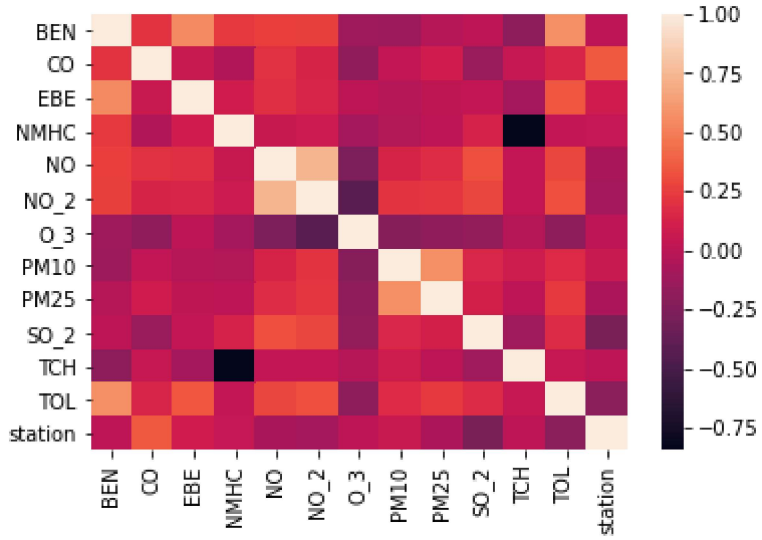
```
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[19]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [20]:
```
sns.heatmap(df.corr())
```

Out[20]: <AxesSubplot:>



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()
```

In [24]:
```
lr.intercept_
```

Out[24]:
```
28078976.944615297
```

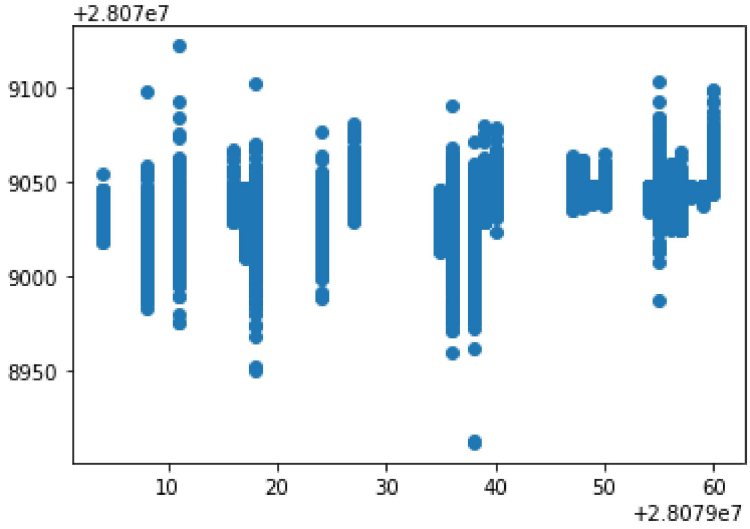
In [25]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[25]:

	Co-efficient
BEN	1.706080
CO	18.885080
EBE	10.105053
NMHC	17.957199
NO	-0.010396
NO_2	-0.037287
O_3	0.009598
PM10	0.276131
PM25	-0.366599
SO_2	-0.943005
TCH	25.241743
TOL	-3.468756

In [26]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[26]:
```
<matplotlib.collections.PathCollection at 0x1e7c2d268e0>
```



ACCURACY

In [27]:
```
lr.score(x_test,y_test)
```

Out[27]: 0.30590270278635223

In [28]:
```
lr.score(x_train,y_train)
```

Out[28]: 0.3078183537160195

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(alpha=10)

Accuracy(Ridge)

In [31]:
```
rr.score(x_test,y_test)
```

Out[31]: 0.3058864865329024

In [32]:
```
rr.score(x_train,y_train)
```

Out[32]: 0.3078155777316942

In [33]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[33]:
```
Lasso(alpha=10)
```

In [34]:
```
la.score(x_train,y_train)
```

Out[34]:
```
0.04557142315360363
```

Accuracy(Lasso)

In [35]:
```
la.score(x_test,y_test)
```

Out[35]:
```
0.04549542733747347
```

In [36]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[36]:
```
ElasticNet()
```

In [37]:
```
en.coef_
```

Out[37]:
```
array([ 0.30926835,  2.64272   ,  0.45457195,  0.        ,  0.03379406,
       -0.05357097, -0.01954829,  0.23461675, -0.34595082, -1.35916987,
       -0.        , -1.57010382])
```

In [38]:
```
en.intercept_
```

Out[38]:
```
28079041.185597092
```

In [39]:
```
prediction=en.predict(x_test)
```

In [40]:
```
en.score(x_test,y_test)
```

Out[40]:
```
0.1621738294701337
```

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)))
```

```
13.776042311056809
258.3654112310911
16.073749134258975
```

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(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],
      dtype=int64)
```

In [52]:
```
logr.score(fs,target_vector)
```

Out[52]:
```
0.9
```

In [53]:
```
logr.predict_proba(observation)[0][0]
```

Out[53]:
```
3.242461294380501e-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()
```

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(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 [59]:
```
grid_search.best_score_
```

Out[59]:
```
0.7503879768030711
```

In [60]:
```
rfc_best=grid_search.best_estimator_
```

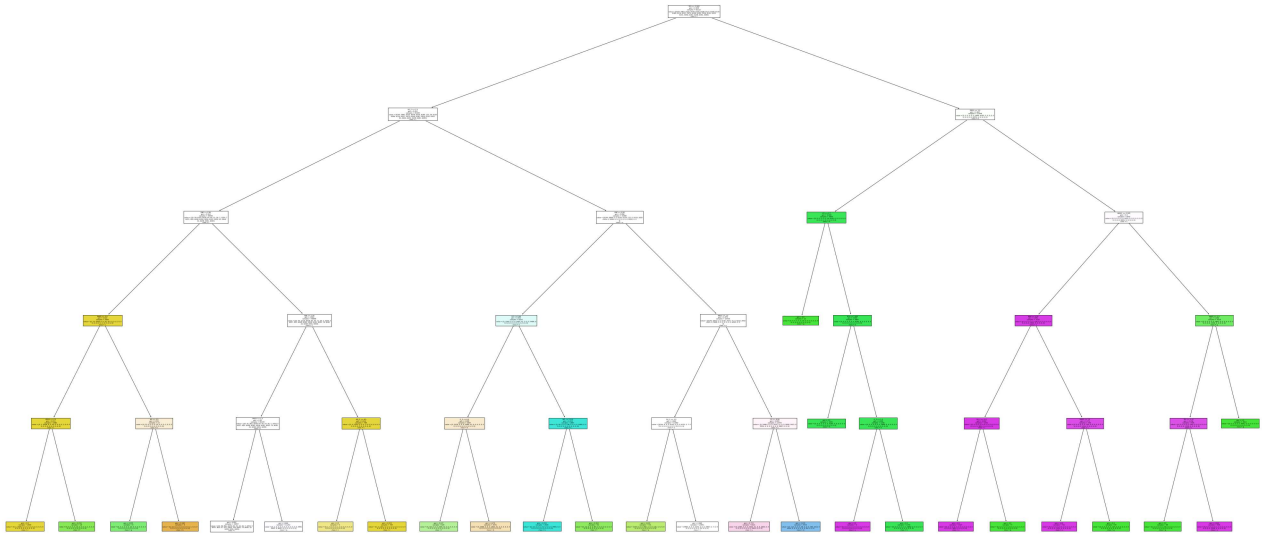
In [61]:
```
from sklearn.tree import plot_tree
```

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

Out[61]:
```
[Text(2448.3673469387754, 1993.2, 'TCH <= 1.005\ngini = 0.958\nsamples = 93132\nvalue =
[6164, 5867, 6325, 6078, 6200, 6189, 6127, 6209, 6130\n6086, 6116, 6037, 5974, 6048, 618
0, 5916, 6191, 6402\n6151, 6092, 6052, 6208, 6091, 6083]\nclass = r'),
 Text(1457.6326530612246, 1630.8000000000002, 'SO_2 <= 1.5\ngini = 0.953\nsamples = 8158
3\nvalue = [6164, 5867, 6325, 6078, 6200, 6189, 133, 26, 6130\n6086, 6116, 6037, 5974, 6
048, 6180, 5916, 6191, 6402\n81, 6092, 6052, 6208, 6091, 6083]\nclass = r'),
 Text(728.8163265306123, 1268.4, 'EBE <= 0.95\ngini = 0.924\nsamples = 49593\nvalue = [2
0, 59, 6325, 6078, 65, 65, 21, 26, 7, 3254, 7\n6037, 905, 6048, 6180, 5916, 6191, 6402,
81, 6092\n32, 6208, 6091, 6083]\nclass = r'),
 Text(364.40816326530614, 906.0, 'PM25 <= 3.0\ngini = 0.022\nsamples = 2695\nvalue = [0,
14, 4209, 0, 0, 20, 10, 0, 0, 0, 2, 0, 0\n0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0]\nclass = c'),
 Text(182.20408163265307, 543.5999999999999, 'PM10 <= 4.5\ngini = 0.011\nsamples = 2682
\nvalue = [0, 1, 4209, 0, 0, 20, 0, 0, 0, 0, 1, 0, 0, 0\n0, 0, 0, 0, 2, 0, 0, 0, 0, 0]\n
class = c'),
 Text(91.10204081632654, 181.19999999999982, 'gini = 0.0\nsamples = 2666\nvalue = [0, 1,
4209, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = c'),
 Text(273.30612244897964, 181.19999999999982, 'gini = 0.234\nsamples = 16\nvalue = [0,
0, 0, 0, 0, 20, 0, 0, 0, 0, 1, 0, 0, 0\n0, 0, 0, 0, 2, 0, 0, 0, 0, 0]\nclass = f'),
 Text(546.6122448979593, 543.5999999999999, 'NO <= 5.5\ngini = 0.531\nsamples = 13\nvalu
e = [0, 13, 0, 0, 0, 0, 10, 0, 0, 0, 1, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
b'),
 Text(455.51020408163265, 181.19999999999982, 'gini = 0.355\nsamples = 8\nvalue = [0, 3,
0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(637.7142857142858, 181.19999999999982, 'gini = 0.165\nsamples = 5\nvalue = [0, 10,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(1093.2244897959185, 906.0, 'EBE <= 1.05\ngini = 0.922\nsamples = 46898\nvalue = [2
0, 45, 2116, 6078, 65, 45, 11, 26, 7, 3254, 5\n6037, 905, 6048, 6180, 5916, 6191, 6402,
79, 6092\n32, 6208, 6091, 6083]\nclass = r'),
 Text(911.0204081632653, 543.5999999999999, 'PM25 <= 1.5\ngini = 0.92\nsamples = 46109\n
value = [20, 41, 861, 6078, 65, 45, 10, 26, 7, 3254, 5\n6037, 905, 6048, 6180, 5916, 619
1, 6402, 74, 6092\n32, 6208, 6091, 6083]\nclass = r'),
 Text(819.9183673469388, 181.19999999999982, 'gini = 0.897\nsamples = 34857\nvalue = [2
0, 39, 861, 6078, 65, 45, 10, 26, 7, 3254, 5\n6037, 905, 57, 537, 5916, 124, 6402, 74, 6
092, 32\n6208, 6091, 6083]\nclass = r'),
 Text(1002.1224489795919, 181.19999999999982, 'gini = 0.666\nsamples = 11252\nvalue =
[0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5991\n5643, 0, 6067, 0, 0, 0, 0, 0, 0, 0]\nclass
= q'),
 Text(1275.4285714285716, 543.5999999999999, 'NO_2 <= 8.5\ngini = 0.016\nsamples = 789\n
value = [0, 4, 1255, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5, 0, 0, 0, 0, 0]\ncla
ss = c'),
 Text(1184.326530612245, 181.19999999999982, 'gini = 0.5\nsamples = 5\nvalue = [0, 1, 4,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = c'),
 Text(1366.530612244898, 181.19999999999982, 'gini = 0.013\nsamples = 784\nvalue = [0,
3, 1251, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5, 0, 0, 0, 0, 0]\nclass = c'),
 Text(2186.448979591837, 1268.4, 'EBE <= 0.95\ngini = 0.886\nsamples = 31990\nvalue = [6
144, 5808, 0, 0, 6135, 6124, 112, 0, 6123, 2832\n6109, 0, 5069, 0, 0, 0, 0, 0, 0, 0, 602
0, 0, 0\n0]\nclass = a'),
 Text(1822.0408163265306, 906.0, 'CO <= 0.95\ngini = 0.652\nsamples = 6554\nvalue = [0,
3304, 0, 0, 0, 2587, 41, 0, 0, 0, 4489, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
k'),
 Text(1639.8367346938776, 543.5999999999999, 'O_3 <= 6.5\ngini = 0.498\nsamples = 3667\n
value = [0, 3278, 0, 0, 0, 2489, 41, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
\nclass = b'),
 Text(1548.734693877551, 181.19999999999982, 'gini = 0.447\nsamples = 626\nvalue = [0, 3
10, 0, 0, 0, 659, 10, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = f'),
 Text(1730.938775510204, 181.19999999999982, 'gini = 0.479\nsamples = 3041\nvalue = [0,
2968, 0, 0, 0, 1830, 31, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
b'),
 Text(2004.2448979591838, 543.5999999999999, 'NO <= 113.0\ngini = 0.053\nsamples = 2887
\nvalue = [0, 26, 0, 0, 0, 98, 0, 0, 0, 0, 4489, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
\nclass = k'),
 Text(1913.1428571428573, 181.19999999999982, 'gini = 0.012\nsamples = 2826\nvalue = [0,
```

14, 0, 0, 0, 14, 0, 0, 0, 0, 4483, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = k'),
 Text(2095.3469387755104, 181.19999999999982, 'gini = 0.304\nsamples = 61\nvalue = [0, 12, 0, 0, 0, 84, 0, 0, 0, 0, 6, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = f'),
 Text(2550.857142857143, 906.0, 'PM10 <= 1.5\ngini = 0.873\nsamples = 25436\nvalue = [6144, 2504, 0, 0, 6135, 3537, 71, 0, 6123, 2832\n1620, 0, 5069, 0, 0, 0, 0, 0, 0, 0, 6020, 0, 0\n0]\nclass = a'),
 Text(2368.65306122449, 543.5999999999999, 'SO_2 <= 2.5\ngini = 0.669\nsamples = 11682\nvalue = [6144, 6, 0, 0, 6135, 5, 0, 0, 6123, 3, 7, 0\n15, 0, 0, 0, 0, 0, 0, 0, 33, 0, 0, 0]\nclass = a'),
 Text(2277.5510204081634, 181.19999999999982, 'gini = 0.403\nsamples = 731\nvalue = [154, 0, 0, 0, 892, 2, 0, 0, 126, 3, 0, 0, 6\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = e'),
 Text(2459.7551020408164, 181.19999999999982, 'gini = 0.668\nsamples = 10951\nvalue = [5990, 6, 0, 0, 5243, 3, 0, 0, 5997, 0, 7, 0, 9\n0, 0, 0, 0, 0, 0, 0, 33, 0, 0, 0]\nclass = i'),
 Text(2733.061224489796, 543.5999999999999, 'CO <= 0.95\ngini = 0.805\nsamples = 13754\nvalue = [0, 2498, 0, 0, 0, 3532, 71, 0, 0, 2829, 1613, 0\n5054, 0, 0, 0, 0, 0, 0, 0, 5987, 0, 0, 0]\nclass = u'),
 Text(2641.9591836734694, 181.19999999999982, 'gini = 0.715\nsamples = 9221\nvalue = [0, 2385, 0, 0, 0, 3503, 71, 0, 0, 2630, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 5931, 0, 0, 0]\nclass = u'),
 Text(2824.1632653061224, 181.19999999999982, 'gini = 0.435\nsamples = 4533\nvalue = [0, 113, 0, 0, 0, 29, 0, 0, 0, 199, 1613, 0\n5054, 0, 0, 0, 0, 0, 0, 0, 56, 0, 0, 0]\nclass = m'),
 Text(3439.1020408163267, 1630.8000000000002, 'PM10 <= 1.5\ngini = 0.667\nsamples = 11549\nvalue = [0, 0, 0, 0, 0, 0, 5994, 6183, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 6070, 0, 0, 0, 0, 0]\nclass = h'),
 Text(2915.265306122449, 1268.4, 'CO <= 0.7\ngini = 0.009\nsamples = 3909\nvalue = [0, 0, 0, 0, 0, 0, 19, 6183, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 9, 0, 0, 0, 0, 0]\nclass = h'),
 Text(2824.1632653061224, 906.0, 'gini = 0.0\nsamples = 12\nvalue = [0, 0, 0, 0, 0, 0, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(3006.367346938776, 906.0, 'TCH <= 1.355\ngini = 0.003\nsamples = 3897\nvalue = [0, 0, 0, 0, 0, 0, 0, 6183, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 9, 0, 0, 0, 0, 0]\nclass = h'),
 Text(2915.265306122449, 543.5999999999999, 'gini = 0.0\nsamples = 2641\nvalue = [0, 0, 0, 0, 0, 0, 0, 4194, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(3097.469387755102, 543.5999999999999, 'O_3 <= 1.5\ngini = 0.009\nsamples = 1256\nvalue = [0, 0, 0, 0, 0, 0, 0, 1989, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 9, 0, 0, 0, 0, 0]\nclass = h'),
 Text(3006.367346938776, 181.19999999999982, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 9, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3188.571428571429, 181.19999999999982, 'gini = 0.0\nsamples = 1250\nvalue = [0, 0, 0, 0, 0, 0, 0, 1989, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(3962.9387755102043, 1268.4, 'NMHC <= 0.195\ngini = 0.5\nsamples = 7640\nvalue = [0, 0, 0, 0, 0, 0, 5975, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 6061, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3644.081632653061, 906.0, 'BEN <= 0.25\ngini = 0.051\nsamples = 3122\nvalue = [0, 0, 0, 0, 0, 0, 128, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 4802, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3461.877551020408, 543.5999999999999, 'SO_2 <= 1.5\ngini = 0.004\nsamples = 2332\nvalue = [0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3651, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3370.775510204082, 181.19999999999982, 'gini = 0.0\nsamples = 2327\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3651, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3552.979591836735, 181.19999999999982, 'gini = 0.0\nsamples = 5\nvalue = [0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(3826.2857142857147, 543.5999999999999, 'PM25 <= 1.5\ngini = 0.171\nsamples = 790\nvalue = [0, 0, 0, 0, 0, 0, 120, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1151, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3735.183673469388, 181.19999999999982, 'gini = 0.01\nsamples = 721\nvalue = [0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1151, 0, 0, 0, 0, 0]\nclass = s'),
 Text(3917.387755102041, 181.19999999999982, 'gini = 0.0\nsamples = 69\nvalue = [0, 0, 0, 0, 0, 0, 114, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(4281.795918367347, 906.0, 'PM25 <= 1.5\ngini = 0.292\nsamples = 4518\nvalue = [0, 0, 0, 0, 0, 0, 5847, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1259, 0, 0, 0, 0, 0]\nclass = g'),
 Text(4190.693877551021, 543.5999999999999, 'TCH <= 1.43\ngini = 0.196\nsamples = 906\nvalue = [0, 0, 0, 0, 0, 0, 156, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1259, 0, 0, 0, 0, 0]\ncl

```
ass = s'),
 Text(4099.591836734694, 181.19999999999982, 'gini = 0.0\nsamples = 98\nvalue = [0, 0,
0, 0, 0, 0, 155, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(4281.795918367347, 181.19999999999982, 'gini = 0.002\nsamples = 808\nvalue = [0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1259, 0, 0, 0, 0, 0]\nclass = s'),
 Text(4372.897959183674, 543.5999999999999, 'gini = 0.0\nsamples = 3612\nvalue = [0, 0,
0, 0, 0, 0, 5691, 0, 0, 0, 0, 0, 0, 0\n0, 0, 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: 0.30590270278635223
Ridge Regression: 0.3058864865329024
Lasso Regression 0.04549542733747347
ElasticNet Regression: 0.1621738294701337
Logistic Regression: 0.9
Random Forest: 0.7503879768030711
```

Logistic Regression is suitable for this dataset