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("2013.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	28C
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	28C
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	28C
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	28C
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	28C
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	28C
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	28C
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	28C
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	28C
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	28C

209880 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.fillna(1)
```

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]:

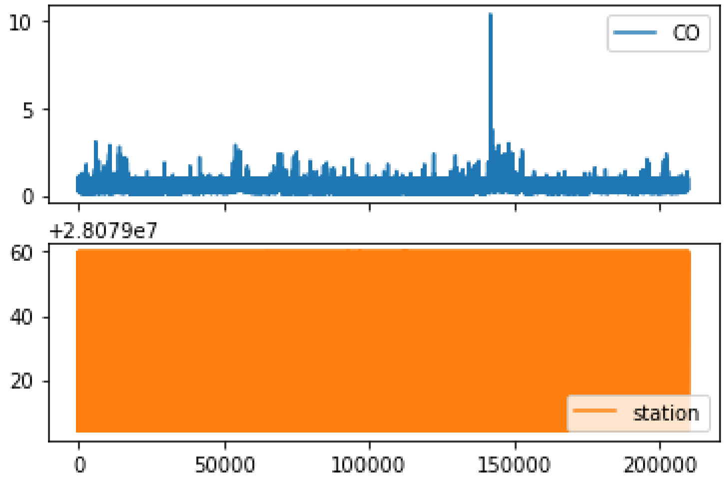
	CO	station
0	0.6	28079004
1	0.5	28079008
2	1.0	28079011
3	0.5	28079016
4	1.0	28079017
...	...	...
209875	0.4	28079056
209876	0.4	28079057
209877	1.0	28079058
209878	1.0	28079059
209879	1.0	28079060

209880 rows × 2 columns

Line chart

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

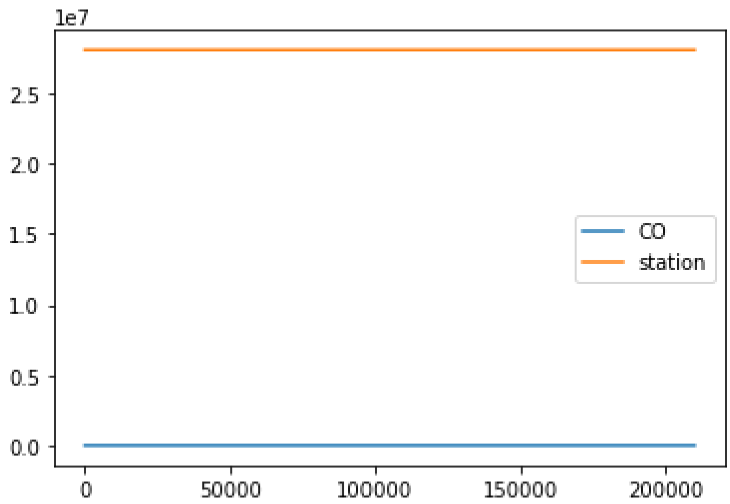
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

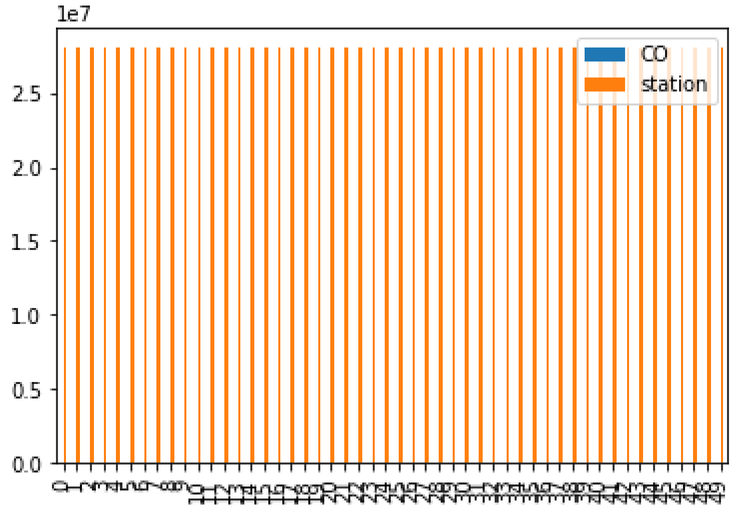


Bar chart

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

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

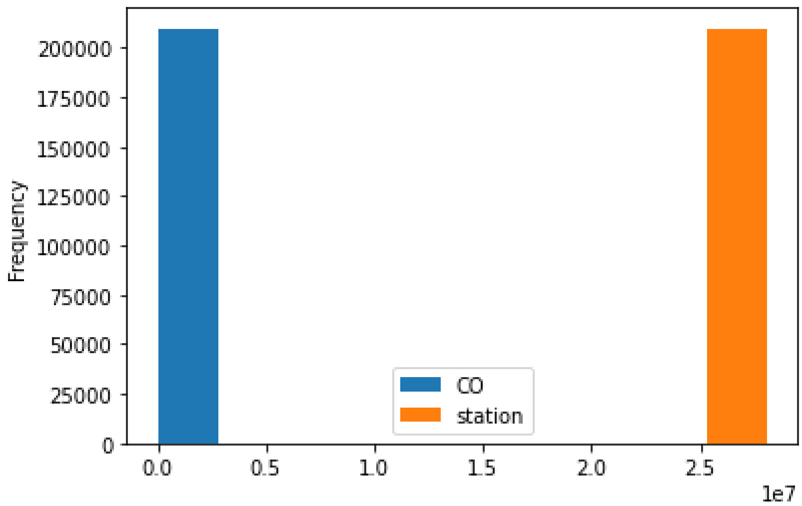
Out[10]:
```
<AxesSubplot:>
```



Histogram

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

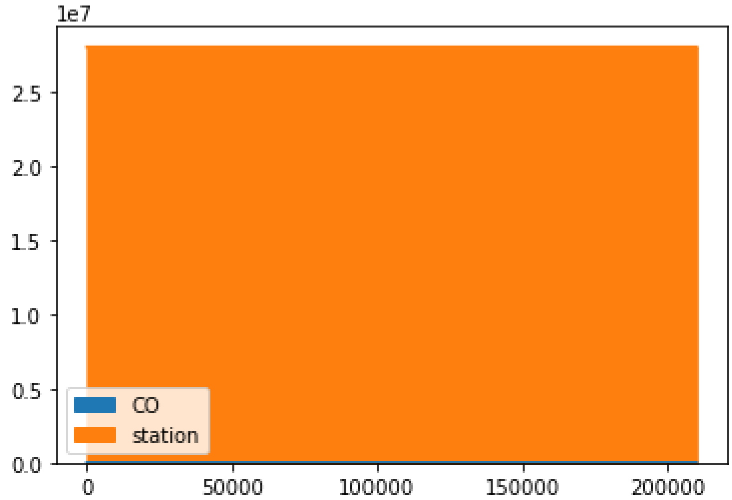
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

Out[12]: <AxesSubplot:>

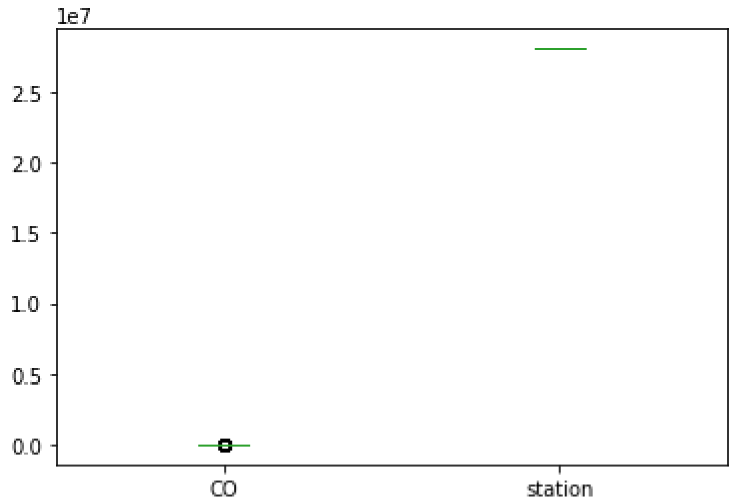


Box chart

In [13]:

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

Out[13]: <AxesSubplot:>



Pie chart

In [14]:

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

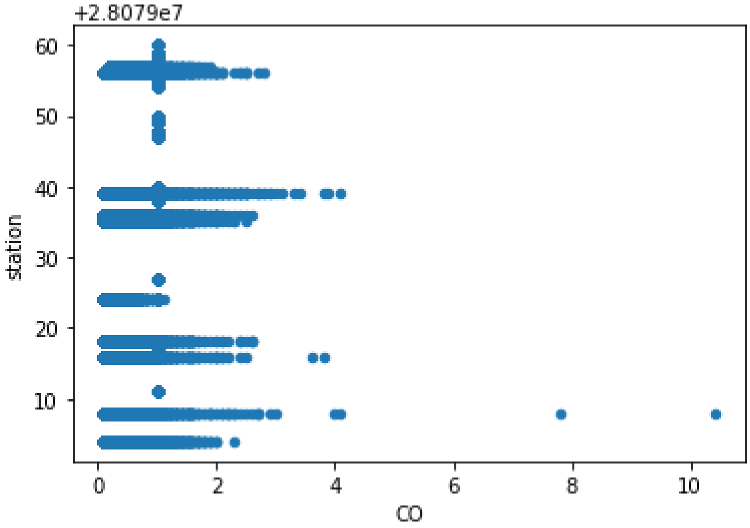
Out[14]: <AxesSubplot:ylabel='station'>

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.columns
```

Out[17]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [18]:
```
df.describe()
```

Out[18]:

	BEN	CO	EBE	NMHC	NO	NO_2	
count	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	2(
mean	0.931014	0.721695	0.954744	0.900223	20.101401	34.586402	
std	0.430684	0.361528	0.301074	0.267139	44.319112	27.866588	
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	

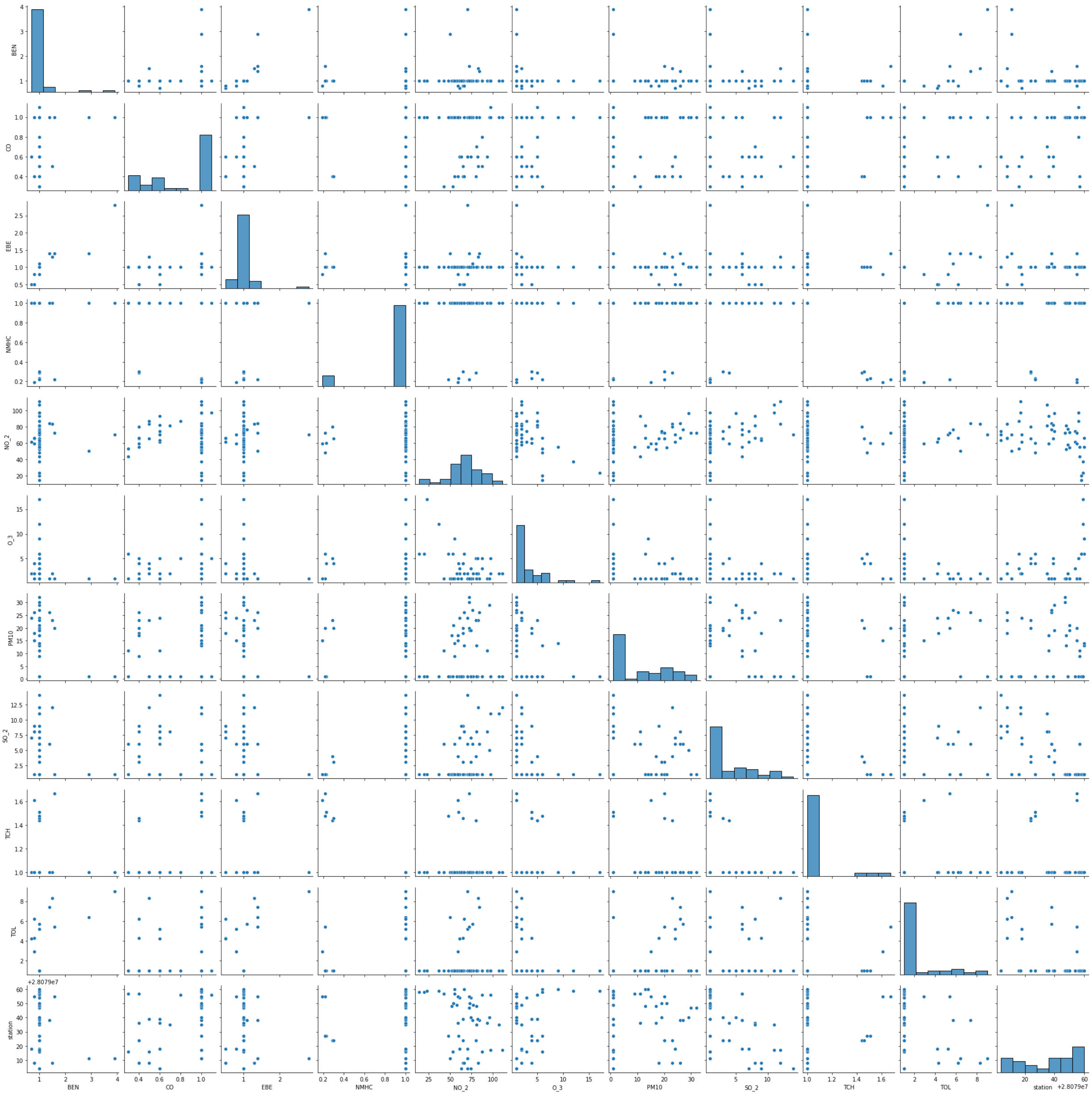
	BEN	CO	EBE	NMHC	NO	NO_2
max	12.100000	10.400000	11.800000	1.000000	1081.000000	388.000000

In [19]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2','O_3',
        'PM10','SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [20]:
```
sns.pairplot(df1[0:50])
```

Out[20]:
```
<seaborn.axisgrid.PairGrid at 0x1ecb809d880>
```

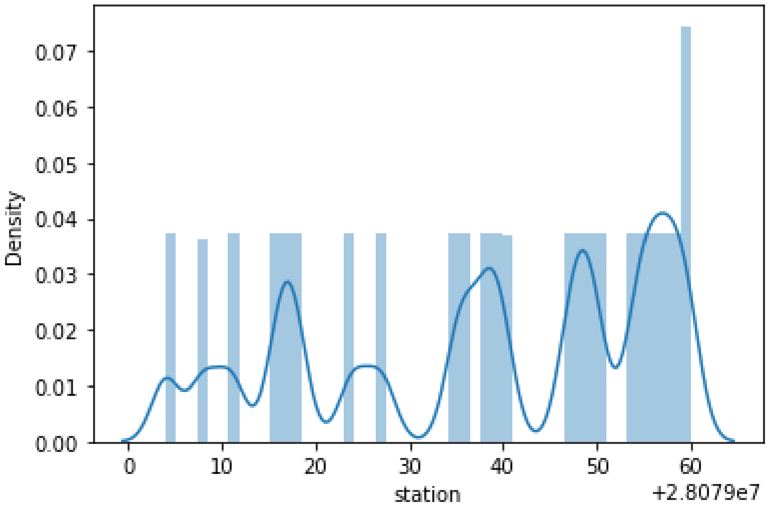


In [21]:
```
sns.distplot(df1['station'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
```

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

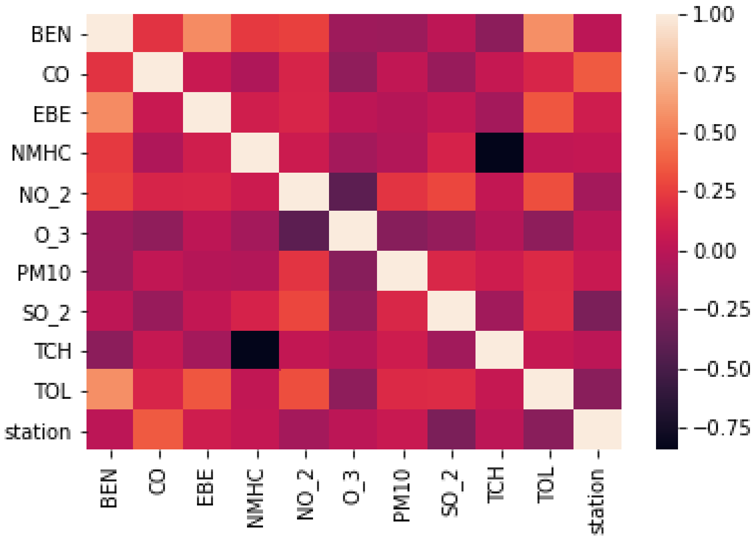
Out[21]: `<AxesSubplot:xlabel='station', ylabel='Density'>`



In [22]:

```
sns.heatmap(df1.corr())
```

Out[22]: `<AxesSubplot:>`



TO TRAIN THE MODEL AND MODEL BULDING

In [23]:

```
x=df[['BEN', 'CO', 'EBE','NMHC', 'NO_2','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)
```

Linear Regression

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]: 28078975.337922096

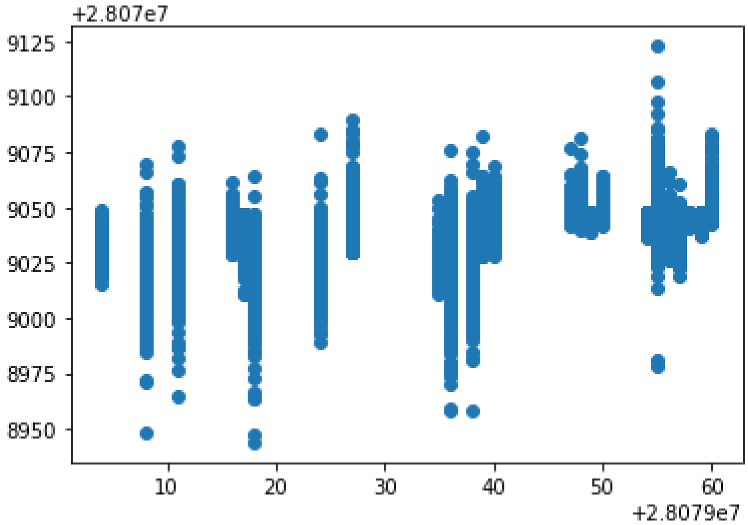
In [27]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[27]:

	Co-efficient
BEN	2.183759
CO	18.267284
EBE	9.859581
NMHC	18.421704
NO_2	-0.055654
O_3	0.008958
PM10	0.207631
SO_2	-0.948829
TCH	26.754017
TOL	-3.629545

In [28]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[28]: <matplotlib.collections.PathCollection at 0x1ecc41da8b0>



ACCURACY

In [29]:
```
lr.score(x_test,y_test)
```
Out[29]: 0.30346977495834404

In [30]:
```
lr.score(x_train,y_train)
```
Out[30]: 0.2982019580998285

Ridge and Lasso

In [31]:
```
from sklearn.linear_model import Ridge,Lasso
```
In [32]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```
Out[32]: Ridge(alpha=10)

Accuracy(Ridge)

In [33]:
```
rr.score(x_test,y_test)
```
Out[33]: 0.30344964354068593

In [34]:
```
rr.score(x_train,y_train)
```
Out[34]: 0.29819884757718973

In [35]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[35]: Lasso(alpha=10)

In [36]:
```
la.score(x_train,y_train)
```
Out[36]: 0.04551248665873586

Accuracy(Lasso)

In [37]:
```
la.score(x_test,y_test)
```
Out[37]: 0.04508362771666252

Elastic Net

In [38]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[38]:
```
ElasticNet()
```

In [39]:
```
en.coef_
```

Out[39]:
```
array([ 0.41231749,  2.69315984,  0.54016161,  0.        , -0.01987354,
       -0.01656965,  0.16339604, -1.27816588, -0.        , -1.62970521])
```

In [40]:
```
en.intercept_
```

Out[40]:
```
28079039.858748578
```

In [41]:
```
prediction=en.predict(x_test)
```

In [42]:
```
en.score(x_test,y_test)
```

Out[42]:
```
0.15378968239891944
```

Evaluation Metrics

In [43]:
```
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)))
```

```
13.726294274893448
261.85861400455855
16.182046038883914
```

Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression
```

In [45]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2','O_3',
       'PM10','SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [46]:
```
feature_matrix.shape
```

Out[46]:
```
(209880, 10)
```

In [47]:
```
target_vector.shape
```

Out[47]:
```
(209880,)
```

In [48]:
```
from sklearn.preprocessing import StandardScaler
```

In [49]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [50]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[50]:
```
LogisticRegression(max_iter=10000)
```

In [51]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [52]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [53]:
```
logr.classes_
```

Out[53]:
```
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 [54]:
```
logr.score(fs,target_vector)
```

Out[54]:
```
0.6612921669525443
```

In [55]:
```
logr.predict_proba(observation)[0][0]
```

Out[55]:
```
9.49253547859177e-217
```

In [56]:
```
logr.predict_proba(observation)
```

Out[56]:
```
array([[9.49253548e-217, 6.03969072e-001, 1.69773000e-169,
        1.44179094e-134, 1.71060740e-074, 3.96021369e-001,
        9.55808997e-006, 5.22717178e-089, 5.48319507e-081,
        1.32436170e-079, 1.07294134e-076, 3.50636612e-129,
        1.69529056e-079, 3.82520459e-158, 4.22872970e-161,
        3.57928159e-187, 2.10845766e-164, 8.33937392e-188,
        1.12752042e-082, 7.42692411e-129, 7.66872499e-080,
        6.30044443e-191, 4.32093567e-191, 3.26054498e-071]])
```

Random Forest

In [57]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [58]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[58]:
```
RandomForestClassifier()
```

In [59]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [60]:
```
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[60]:
```
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 [61]:
```
grid_search.best_score_
```

Out[61]:
```
0.6929265702850609
```

In [62]:
```
rfc_best=grid_search.best_estimator_
```

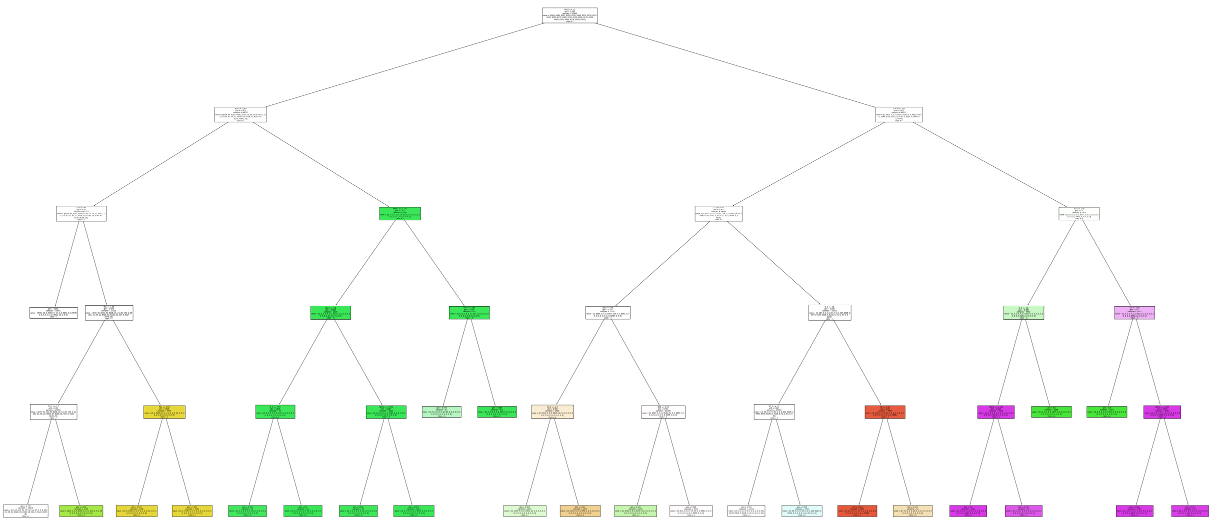
In [64]:
```
from sklearn.tree import plot_tree

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

Out[64]:
```
[Text(2092.5, 1993.2, 'PM10 <= 1.5\ngini = 0.958\nsamples = 92849\nvalue = [6006, 60
86, 6241, 6000, 6235, 6266, 6219, 6150, 6011\n6102, 6050, 6176, 5985, 6212, 6126, 60
28, 6179, 6238\n6068, 6165, 6096, 6124, 6049, 6104]\nclass = f'),
 Text(887.7272727272727, 1630.8000000000002, 'TCH <= 1.005\ngini = 0.918\nsamples =
46637\nvalue = [6006, 64, 6241, 6000, 6235, 25, 34, 6150, 6011, 11\n25, 6176, 41, 3
6, 21, 6028, 63, 6238, 35, 6165, 47\n6124, 6049, 25]\nclass = c'),
 Text(304.3636363636364, 1268.4, 'CO <= 0.95\ngini = 0.91\nsamples = 42729\nvalue =
[6006, 64, 6241, 6000, 6235, 25, 14, 25, 6011, 11\n25, 6176, 41, 36, 21, 6028, 63, 6
238, 18, 6165, 47\n6124, 6049, 25]\nclass = c'),
 Text(202.9090909090909, 906.0, 'gini = 0.801\nsamples = 18601\nvalue = [5734, 18,
0, 5907, 0, 14, 0, 0, 5841, 8, 0, 5975\n0, 0, 0, 0, 0, 0, 5861, 38, 0, 0, 0]\ncla
ss = l'),
 Text(405.8181818181818, 906.0, 'TOL <= 1.05\ngini = 0.845\nsamples = 24128\nvalue =
[272, 46, 6241, 93, 6235, 11, 14, 25, 170, 3, 25\n201, 41, 36, 21, 6028, 63, 6238, 1
8, 304, 9, 6124\n6049, 25]\nclass = c'),
 Text(202.9090909090909, 543.5999999999999, 'SO_2 <= 1.5\ngini = 0.82\nsamples = 203
91\nvalue = [272, 45, 311, 93, 6235, 10, 14, 25, 170, 3, 11\n201, 41, 36, 21, 6028,
63, 6238, 18, 304, 9, 6124\n6049, 25]\nclass = r'),
 Text(101.45454545454545, 181.19999999999982, 'gini = 0.774\nsamples = 16215\nvalue
= [11, 45, 311, 93, 79, 10, 14, 25, 9, 3, 8, 201\n21, 36, 21, 6028, 63, 6238, 18, 30
4, 9, 6124, 6049\n25]\nclass = r'),
 Text(304.3636363636364, 181.19999999999982, 'gini = 0.128\nsamples = 4176\nvalue =
[261, 0, 0, 0, 6156, 0, 0, 0, 161, 0, 3, 0, 20\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\ncl
ass = e'),
 Text(608.7272727272727, 543.5999999999999, 'TOL <= 3.65\ngini = 0.005\nsamples = 37
37\nvalue = [0, 1, 5930, 0, 0, 1, 0, 0, 0, 0, 14, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0,
0, 0]\nclass = c'),
 Text(507.27272727272725, 181.19999999999982, 'gini = 0.011\nsamples = 1385\nvalue =
[0, 0, 2205, 0, 0, 0, 0, 0, 0, 0, 12, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass
= c'),
 Text(710.1818181818181, 181.19999999999982, 'gini = 0.002\nsamples = 2352\nvalue =
[0, 1, 3725, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
```

```
c'),
 Text(1471.090909090909, 1268.4, 'NMHC <= 0.225\ngini = 0.012\nsamples = 3908\nvalue
= [0, 0, 0, 0, 0, 0, 20, 6125, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 17, 0, 0, 0, 0, 0]\ncla
ss = h'),
 Text(1217.4545454545455, 906.0, 'NO_2 <= 2.5\ngini = 0.008\nsamples = 3316\nvalue =
[0, 0, 0, 0, 0, 0, 3, 5211, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 17, 0, 0, 0, 0, 0]\nclass
= h'),
 Text(1014.5454545454545, 543.5999999999999, 'O_3 <= 73.5\ngini = 0.035\nsamples = 7
4\nvalue = [0, 0, 0, 0, 0, 0, 2, 111, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0,
0]\nclass = h'),
 Text(913.0909090909091, 181.19999999999982, 'gini = 0.102\nsamples = 20\nvalue =
[0, 0, 0, 0, 0, 0, 2, 35, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
h'),
 Text(1116.0, 181.19999999999982, 'gini = 0.0\nsamples = 54\nvalue = [0, 0, 0, 0, 0,
0, 0, 76, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = h'),
 Text(1420.3636363636363, 543.5999999999999, 'NMHC <= 0.075\ngini = 0.007\nsamples =
3242\nvalue = [0, 0, 0, 0, 0, 0, 1, 5100, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 17, 0, 0, 0,
0, 0]\nclass = h'),
 Text(1318.909090909091, 181.19999999999982, 'gini = 0.044\nsamples = 25\nvalue =
[0, 0, 0, 0, 0, 0, 0, 43, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 1, 0, 0, 0, 0, 0]\nclass =
h'),
 Text(1521.8181818181818, 181.19999999999982, 'gini = 0.007\nsamples = 3217\nvalue =
[0, 0, 0, 0, 0, 0, 1, 5057, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 16, 0, 0, 0, 0, 0]\nclass
= h'),
 Text(1724.7272727272727, 906.0, 'TCH <= 1.385\ngini = 0.036\nsamples = 592\nvalue =
[0, 0, 0, 0, 0, 0, 17, 914, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
h'),
 Text(1623.2727272727273, 543.5999999999999, 'gini = 0.473\nsamples = 21\nvalue =
[0, 0, 0, 0, 0, 0, 15, 24, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
h'),
 Text(1826.1818181818182, 543.5999999999999, 'gini = 0.004\nsamples = 571\nvalue =
[0, 0, 0, 0, 0, 0, 2, 890, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
h'),
 Text(3297.272727272727, 1630.8000000000002, 'TCH <= 1.105\ngini = 0.917\nsamples =
46212\nvalue = [0, 6022, 0, 0, 0, 6241, 6185, 0, 0, 6091, 6025\n0, 5944, 6176, 6105,
0, 6116, 0, 6033, 0, 6049, 0\n0, 6079]\nclass = f'),
 Text(2637.818181818182, 1268.4, 'CO <= 0.95\ngini = 0.901\nsamples = 38649\nvalue =
[0, 6022, 0, 0, 0, 6241, 108, 0, 0, 6091, 6025, 0\n5944, 6176, 6105, 0, 6116, 0, 70,
0, 6049, 0, 0\n6079]\nclass = f'),
 Text(2232.0, 906.0, 'EBE <= 0.95\ngini = 0.752\nsamples = 15107\nvalue = [0, 5858,
0, 0, 0, 6094, 106, 0, 0, 5861, 0, 0\n0, 0, 0, 0, 0, 0, 0, 5995, 0, 0, 0]\nclass
= f'),
 Text(2029.090909090909, 543.5999999999999, 'SO_2 <= 4.5\ngini = 0.499\nsamples = 37
29\nvalue = [0, 3311, 0, 0, 0, 2549, 46, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]\nclass = b'),
 Text(1927.6363636363635, 181.19999999999982, 'gini = 0.505\nsamples = 1857\nvalue =
[0, 1239, 0, 0, 0, 1653, 44, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\ncla
ss = f'),
 Text(2130.5454545454545, 181.19999999999982, 'gini = 0.422\nsamples = 1872\nvalue =
[0, 2072, 0, 0, 0, 896, 2, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass
= b'),
 Text(2434.909090909091, 543.5999999999999, 'BEN <= 0.95\ngini = 0.724\nsamples = 11
378\nvalue = [0, 2547, 0, 0, 0, 3545, 60, 0, 0, 5861, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0,
5995, 0, 0, 0]\nclass = u'),
 Text(2333.4545454545455, 181.19999999999982, 'gini = 0.472\nsamples = 3456\nvalue =
[0, 2034, 0, 0, 0, 3472, 30, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\ncla
ss = f'),
 Text(2536.3636363636365, 181.19999999999982, 'gini = 0.546\nsamples = 7922\nvalue =
[0, 513, 0, 0, 0, 73, 30, 0, 0, 5861, 0, 0, 0\n0, 0, 0, 0, 0, 0, 5995, 0, 0, 0]\n
class = u'),
 Text(3043.6363636363635, 906.0, 'O_3 <= 2.5\ngini = 0.839\nsamples = 23542\nvalue =
[0, 164, 0, 0, 0, 147, 2, 0, 0, 230, 6025, 0\n5944, 6176, 6105, 0, 6116, 0, 70, 0, 5
4, 0, 0\n6079]\nclass = n'),
 Text(2840.7272727272725, 543.5999999999999, 'SO_2 <= 1.5\ngini = 0.807\nsamples = 1
9633\nvalue = [0, 58, 0, 0, 0, 65, 0, 0, 0, 230, 6025, 0\n5944, 6176, 6105, 0, 6116,
0, 70, 0, 54, 0, 0\n93]\nclass = n'),
 Text(2739.272727272727, 181.19999999999982, 'gini = 0.701\nsamples = 12314\nvalue =
[0, 0, 0, 0, 0, 6, 0, 0, 0, 4, 2, 0, 911\n6176, 6105, 0, 6116, 0, 70, 0, 9, 0, 0, 9
3]\nclass = n'),
```

```
 Text(2942.181818181818, 181.19999999999982, 'gini = 0.529\nsamples = 7319\nvalue =
[0, 58, 0, 0, 0, 59, 0, 0, 0, 226, 6023, 0\n5033, 0, 0, 0, 0, 0, 0, 0, 45, 0, 0, 0]
\nclass = k'),
 Text(3246.5454545454545, 543.5999999999999, 'CO <= 1.05\ngini = 0.06\nsamples = 390
9\nvalue = [0, 106, 0, 0, 0, 82, 2, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0,
5986]\nclass = x'),
 Text(3145.090909090909, 181.19999999999982, 'gini = 0.024\nsamples = 3835\nvalue =
[0, 34, 0, 0, 0, 37, 2, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 5986]\nclass
= x'),
 Text(3348.0, 181.19999999999982, 'gini = 0.473\nsamples = 74\nvalue = [0, 72, 0, 0,
0, 45, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(3956.7272727272725, 1268.4, 'TOL <= 1.05\ngini = 0.5\nsamples = 7563\nvalue =
[0, 0, 0, 0, 0, 0, 6077, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 5963, 0, 0, 0, 0, 0]\nclas
s = g'),
 Text(3753.818181818182, 906.0, 'O_3 <= 1.5\ngini = 0.485\nsamples = 4593\nvalue =
[0, 0, 0, 0, 0, 0, 4282, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 3013, 0, 0, 0, 0, 0]\nclas
s = g'),
 Text(3652.3636363636365, 543.5999999999999, 'NMHC <= 0.225\ngini = 0.003\nsamples =
1925\nvalue = [0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 3013, 0, 0, 0,
0, 0]\nclass = s'),
 Text(3550.909090909091, 181.19999999999982, 'gini = 0.0\nsamples = 1900\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 2981, 0, 0, 0, 0, 0]\nclass =
s'),
 Text(3753.818181818182, 181.19999999999982, 'gini = 0.234\nsamples = 25\nvalue =
[0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 32, 0, 0, 0, 0, 0]\nclass =
s'),
 Text(3855.272727272727, 543.5999999999999, 'gini = 0.0\nsamples = 2668\nvalue = [0,
0, 0, 0, 0, 0, 4277, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
g'),
 Text(4159.636363636364, 906.0, 'CO <= 0.95\ngini = 0.47\nsamples = 2970\nvalue =
[0, 0, 0, 0, 0, 0, 1795, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 2950, 0, 0, 0, 0, 0]\nclas
s = s'),
 Text(4058.181818181818, 543.5999999999999, 'gini = 0.0\nsamples = 1097\nvalue = [0,
0, 0, 0, 0, 0, 1791, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass =
g'),
 Text(4261.090909090909, 543.5999999999999, 'NMHC <= 0.355\ngini = 0.003\nsamples =
1873\nvalue = [0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 2950, 0, 0, 0,
0, 0]\nclass = s'),
 Text(4159.636363636364, 181.19999999999982, 'gini = 0.001\nsamples = 1790\nvalue =
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 2837, 0, 0, 0, 0, 0]\nclass =
s'),
 Text(4362.545454545454, 181.19999999999982, 'gini = 0.05\nsamples = 83\nvalue = [0,
0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 113, 0, 0, 0, 0, 0]\nclass = s')]
```



Conclusion

Scores

Linear Regression

In [65]:
```
lr.score(x_test,y_test)
```
Out[65]: 0.30346977495834404

In [66]:
```
lr.score(x_train,y_train)
```
Out[66]: 0.2982019580998285

Lasso

In [67]:
```
la.score(x_test,y_test)
```
Out[67]: 0.04508362771666252

Ridge

In [68]:
```
rr.score(x_test,y_test)
```
Out[68]: 0.30344964354068593

In [69]:
```
rr.score(x_train,y_train)
```
Out[69]: 0.29819884757718973

Elastic Net

In [70]:
```
en.score(x_test,y_test)
```
Out[70]: 0.15378968239891944

Logistic Regression

In [71]:
```
logr.score(fs,target_vector)
```
Out[71]: 0.6612921669525443

Random Forest

In [72]:
```
grid_search.best_score_
```

Out[72]: 0.6929265702850609

From the above data, we can conclude that random forest regression is preferrable to other regression types

In [ ]:
```

```