

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("2009.csv")
df
```

Out[2]:

| | date | BEN | CO | EBE | MXV | NMHC | NO_2 | NOx | OXY | O_3 | |
|--------|---------------------|------|------|------|------|------|-----------|------------|------|-----------|------|
| 0 | 2009-10-01 01:00:00 | NaN | 0.27 | NaN | NaN | NaN | 39.889999 | 48.150002 | NaN | 50.680000 | 18.0 |
| 1 | 2009-10-01 01:00:00 | NaN | 0.22 | NaN | NaN | NaN | 21.230000 | 24.260000 | NaN | 55.880001 | 10.0 |
| 2 | 2009-10-01 01:00:00 | NaN | 0.18 | NaN | NaN | NaN | 31.230000 | 34.880001 | NaN | 49.060001 | 25.0 |
| 3 | 2009-10-01 01:00:00 | 0.95 | 0.33 | 1.43 | 2.68 | 0.25 | 55.180000 | 81.360001 | 1.57 | 36.669998 | 26.0 |
| 4 | 2009-10-01 01:00:00 | NaN | 0.41 | NaN | NaN | 0.12 | 61.349998 | 76.260002 | NaN | 38.090000 | 23.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 215683 | 2009-06-01 00:00:00 | 0.50 | 0.22 | 0.39 | 0.75 | 0.09 | 22.000000 | 24.510000 | 1.00 | 82.239998 | 10.0 |
| 215684 | 2009-06-01 00:00:00 | NaN | 0.31 | NaN | NaN | NaN | 76.110001 | 101.099998 | NaN | 41.220001 | 9.0 |
| 215685 | 2009-06-01 00:00:00 | 0.13 | NaN | 0.86 | NaN | 0.23 | 81.050003 | 99.849998 | NaN | 24.830000 | 12.0 |
| 215686 | 2009-06-01 00:00:00 | 0.21 | NaN | 2.96 | NaN | 0.10 | 72.419998 | 82.959999 | NaN | NaN | 13.0 |
| 215687 | 2009-06-01 00:00:00 | 0.37 | 0.32 | 0.99 | 1.36 | 0.14 | 54.290001 | 64.480003 | 1.06 | 56.919998 | 15.0 |

215688 rows × 17 columns

Data Cleaning and Data Preprocessing

In [3]:

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

In [4]:

```
df.columns
```

Out[4]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
      'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24717 entries, 3 to 215687
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  -
0   date        24717 non-null  object
1   BEN         24717 non-null  float64
2   CO          24717 non-null  float64
3   EBE         24717 non-null  float64
4   MXY         24717 non-null  float64
5   NMHC        24717 non-null  float64
6   NO_2        24717 non-null  float64
7   NOx         24717 non-null  float64
8   OXY         24717 non-null  float64
9   O_3         24717 non-null  float64
10  PM10        24717 non-null  float64
11  PM25        24717 non-null  float64
12  PXY         24717 non-null  float64
13  SO_2        24717 non-null  float64
14  TCH         24717 non-null  float64
15  TOL         24717 non-null  float64
16  station     24717 non-null  int64
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

In [6]:

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

Out[6]:

| | CO | station |
|--------|------|----------|
| 3 | 0.33 | 28079006 |
| 20 | 0.32 | 28079024 |
| 24 | 0.24 | 28079099 |
| 28 | 0.21 | 28079006 |
| 45 | 0.30 | 28079024 |
| ... | ... | ... |
| 215659 | 0.27 | 28079024 |
| 215663 | 0.35 | 28079099 |
| 215667 | 0.29 | 28079006 |
| 215683 | 0.22 | 28079024 |
| 215687 | 0.32 | 28079099 |

24717 rows × 2 columns

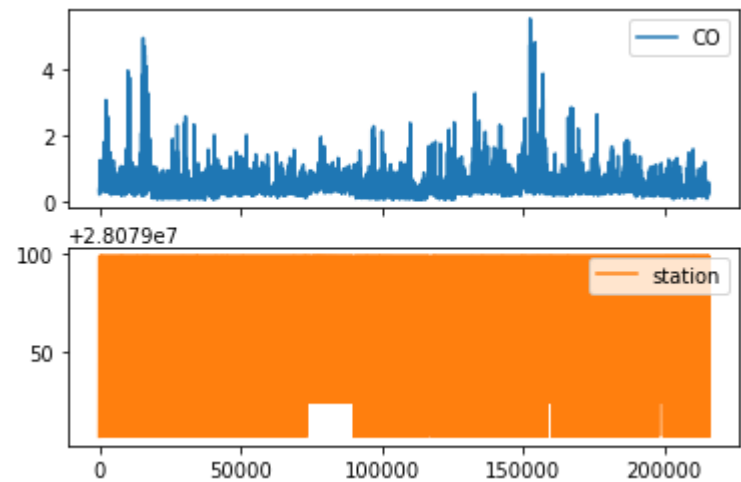
Line chart

In [7]:

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

Out[7]:

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



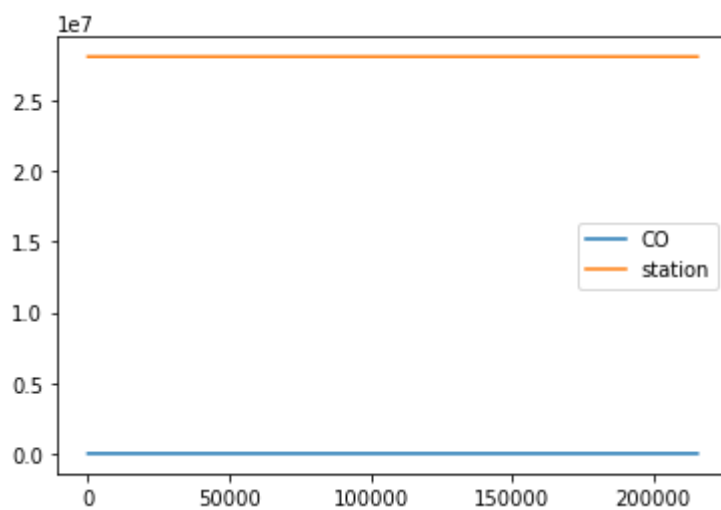
Line chart

In [8]:

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

Out[8]:

<AxesSubplot:>



Bar chart

In [9]:

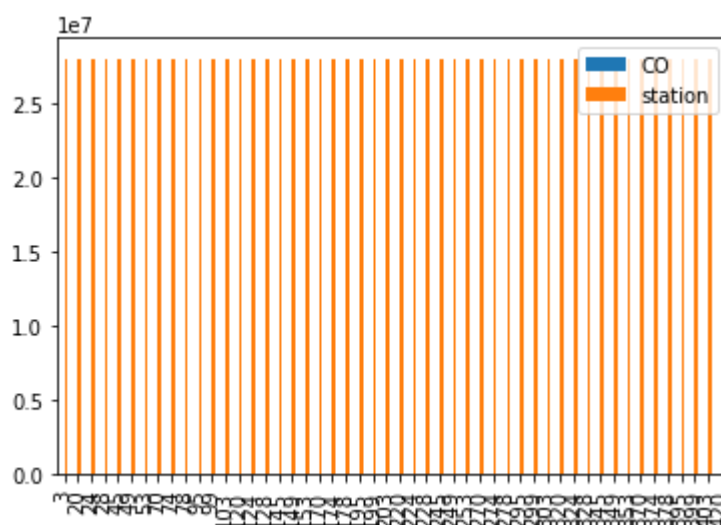
```
b=data[0:50]
```

In [10]:

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

Out[10]:

<AxesSubplot:>



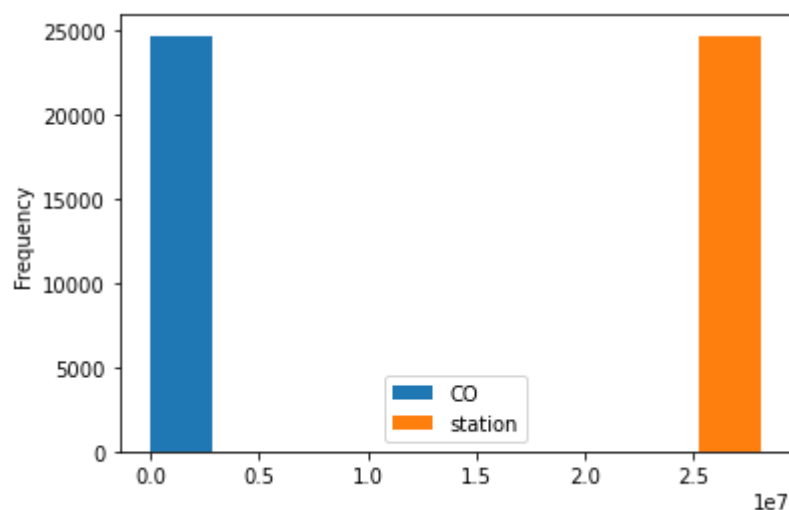
Histogram

In [11]:

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

Out[11]:

<AxesSubplot:ylabel='Frequency'>



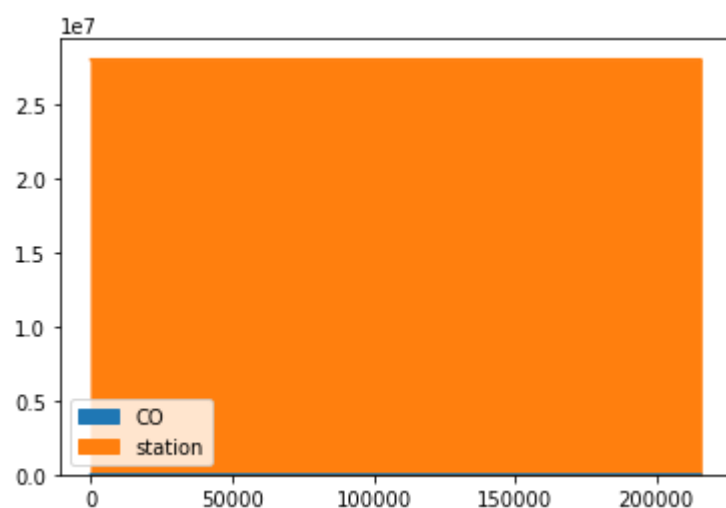
Area chart

In [12]:

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

Out[12]:

<AxesSubplot:>



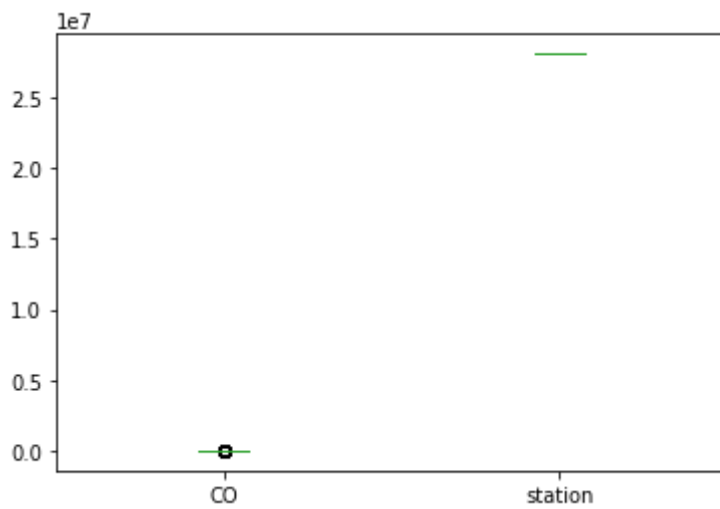
Box chart

In [13]:

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

Out[13]:

<AxesSubplot:>



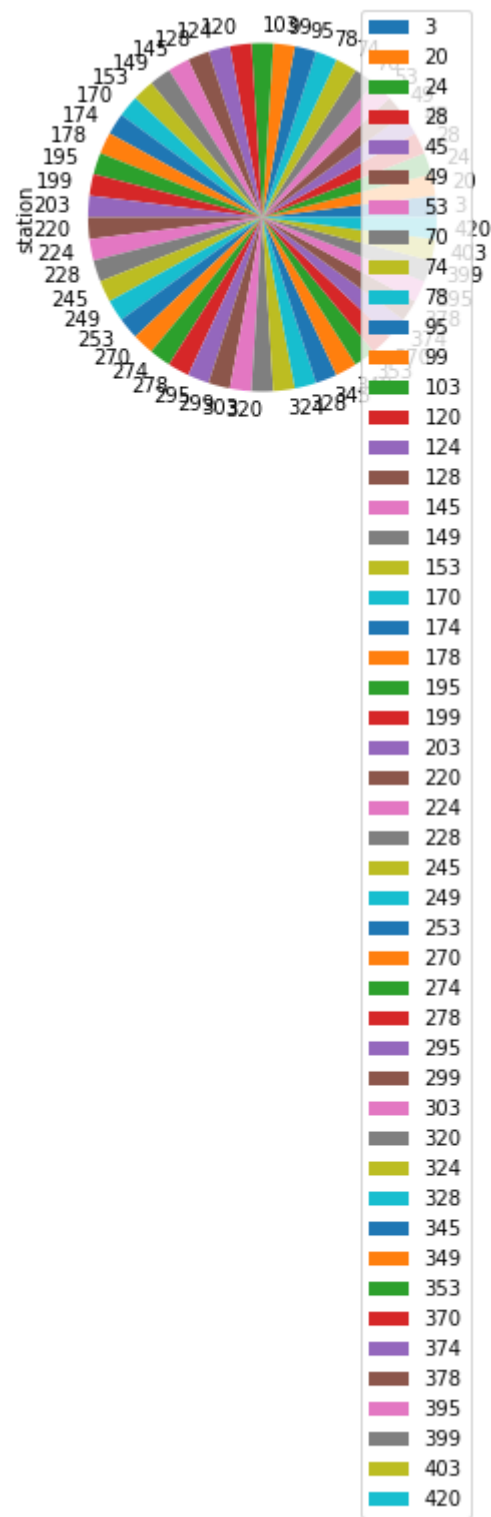
Pie chart

In [14]:

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

Out[14]:

<AxesSubplot:ylabel='station'>



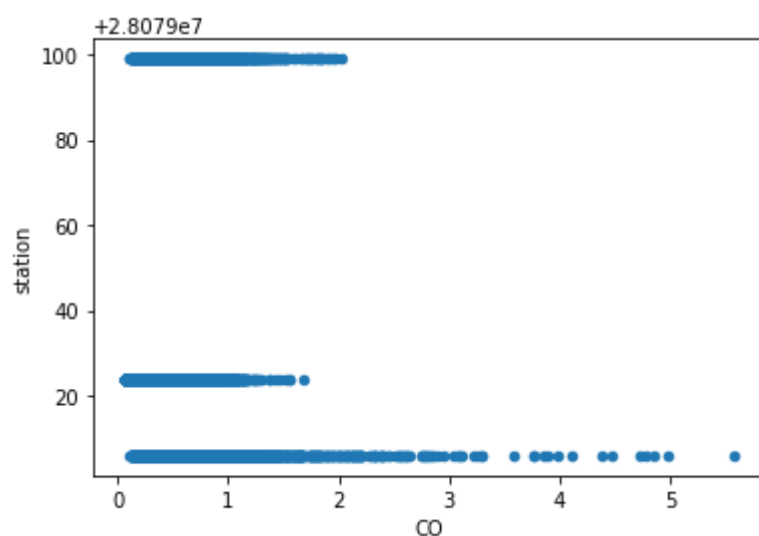
Scatter chart

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'>
Int64Index: 24717 entries, 3 to 215687
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   date        24717 non-null  object  
 1   BEN         24717 non-null  float64 
 2   CO          24717 non-null  float64 
 3   EBE         24717 non-null  float64 
 4   MXY         24717 non-null  float64 
 5   NMHC        24717 non-null  float64 
 6   NO_2        24717 non-null  float64 
 7   NOx         24717 non-null  float64 
 8   OXY         24717 non-null  float64 
 9   O_3         24717 non-null  float64 
10  PM10        24717 non-null  float64 
11  PM25        24717 non-null  float64 
12  PXY         24717 non-null  float64 
13  SO_2        24717 non-null  float64 
14  TSP         24717 non-null  float64 
15  TSP_25      24717 non-null  float64 
16  TSP_250     24717 non-null  float64 
17  TSP_500     24717 non-null  float64 
18  TSP_1000    24717 non-null  float64 
19  TSP_2000    24717 non-null  float64 
20  TSP_4000    24717 non-null  float64 
21  TSP_8000    24717 non-null  float64 
22  TSP_16000   24717 non-null  float64 
23  TSP_32000   24717 non-null  float64 
24  TSP_64000   24717 non-null  float64 
25  TSP_128000  24717 non-null  float64 
26  TSP_256000  24717 non-null  float64 
27  TSP_512000  24717 non-null  float64 
28  TSP_1024000 24717 non-null  float64 
29  TSP_2048000 24717 non-null  float64 
30  TSP_4096000 24717 non-null  float64 
31  TSP_8192000 24717 non-null  float64 
32  TSP_16384000 24717 non-null  float64 
33  TSP_32768000 24717 non-null  float64 
34  TSP_65536000 24717 non-null  float64 
35  TSP_131072000 24717 non-null  float64 
36  TSP_262144000 24717 non-null  float64 
37  TSP_524288000 24717 non-null  float64 
38  TSP_1048576000 24717 non-null  float64 
39  TSP_2097152000 24717 non-null  float64 
40  TSP_4194304000 24717 non-null  float64 
41  TSP_8388608000 24717 non-null  float64 
42  TSP_16777216000 24717 non-null  float64 
43  TSP_33554432000 24717 non-null  float64 
44  TSP_67108864000 24717 non-null  float64 
45  TSP_134217728000 24717 non-null  float64 
46  TSP_268435456000 24717 non-null  float64 
47  TSP_536870912000 24717 non-null  float64 
48  TSP_1073741824000 24717 non-null  float64 
49  TSP_2147483648000 24717 non-null  float64 
50  TSP_4294967296000 24717 non-null  float64 
51  TSP_8589934592000 24717 non-null  float64 
52  TSP_17179869184000 24717 non-null  float64 
53  TSP_34359738368000 24717 non-null  float64 
54  TSP_68719476736000 24717 non-null  float64 
55  TSP_137438953472000 24717 non-null  float64 
56  TSP_274877906944000 24717 non-null  float64 
57  TSP_549755813888000 24717 non-null  float64 
58  TSP_1099511627776000 24717 non-null  float64 
59  TSP_2199023255552000 24717 non-null  float64 
60  TSP_4398046511104000 24717 non-null  float64 
61  TSP_8796093022208000 24717 non-null  float64 
62  TSP_17592186044416000 24717 non-null  float64 
63  TSP_35184372088832000 24717 non-null  float64 
64  TSP_70368744177664000 24717 non-null  float64 
65  TSP_140737488355328000 24717 non-null  float64 
66  TSP_281474976710656000 24717 non-null  float64 
67  TSP_562949953421312000 24717 non-null  float64 
68  TSP_1125899906842624000 24717 non-null  float64 
69  TSP_2251799813685248000 24717 non-null  float64 
70  TSP_4503599627370496000 24717 non-null  float64 
71  TSP_9007199254740992000 24717 non-null  float64 
72  TSP_18014398509481984000 24717 non-null  float64 
73  TSP_36028797018963968000 24717 non-null  float64 
74  TSP_72057594037927936000 24717 non-null  float64 
75  TSP_144115188075855872000 24717 non-null  float64 
76  TSP_288230376151711744000 24717 non-null  float64 
77  TSP_576460752303423488000 24717 non-null  float64 
78  TSP_1152921504606846976000 24717 non-null  float64 
79  TSP_2305843009213693952000 24717 non-null  float64 
80  TSP_4611686018427387904000 24717 non-null  float64 
81  TSP_9223372036854775808000 24717 non-null  float64 
82  TSP_18446744073709551616000 24717 non-null  float64 
83  TSP_36893488147419103232000 24717 non-null  float64 
84  TSP_73786976294838206464000 24717 non-null  float64 
85  TSP_147573952589676412928000 24717 non-null  float64 
86  TSP_295147905179352825856000 24717 non-null  float64 
87  TSP_590295810358705651712000 24717 non-null  float64 
88  TSP_1180591620717411303424000 24717 non-null  float64 
89  TSP_2361183241434822606848000 24717 non-null  float64 
90  TSP_4722366482869645213696000 24717 non-null  float64 
91  TSP_9444732965739290427392000 24717 non-null  float64 
92  TSP_18889465931478580854784000 24717 non-null  float64 
93  TSP_37778931862957161709568000 24717 non-null  float64 
94  TSP_75557863725914323419136000 24717 non-null  float64 
95  TSP_151115727451828646838272000 24717 non-null  float64 
96  TSP_302231454903657293676544000 24717 non-null  float64 
97  TSP_604462909807314587353088000 24717 non-null  float64 
98  TSP_1208925819614629174706176000 24717 non-null  float64 
99  TSP_2417851639229258349412352000 24717 non-null  float64 
100 TSP_4835703278458516698824704000 24717 non-null  float64 
101 TSP_9671406556917033397649408000 24717 non-null  float64 
102 TSP_19342813113834066795298816000 24717 non-null  float64 
103 TSP_38685626227668133590597632000 24717 non-null  float64 
104 TSP_77371252455336267181195264000 24717 non-null  float64 
105 TSP_154742504910672534362390528000 24717 non-null  float64 
106 TSP_309485009821345068724781056000 24717 non-null  float64 
107 TSP_618970019642690137449562112000 24717 non-null  float64 
108 TSP_1237940039285380274899124224000 24717 non-null  float64 
109 TSP_2475880078570760549798248448000 24717 non-null  float64 
110 TSP_4951760157141521099596496896000 24717 non-null  float64 
111 TSP_9903520314283042199192993792000 24717 non-null  float64 
112 TSP_19807040628566084398385987584000 24717 non-null  float64 
113 TSP_39614081257132168796771975168000 24717 non-null  float64 
114 TSP_79228162514264337593543950336000 24717 non-null  float64 
115 TSP_158456325028528675187087900672000 24717 non-null  float64 
116 TSP_316912650057057350374175801344000 24717 non-null  float64 
117 TSP_633825300114114700748351602688000 24717 non-null  float64 
118 TSP_1267650600228229401496703205376000 24717 non-null  float64 
119 TSP_2535301200456458802993406410752000 24717 non-null  float64 
120 TSP_5070602400912917605986812821504000 24717 non-null  float64 
121 TSP_10141204801825835211973625643008000 24717 non-null  float64 
122 TSP_20282409603651670423947251286016000 24717 non-null  float64 
123 TSP_40564819207303340847894502572032000 24717 non-null  float64 
124 TSP_81129638414606681695789005144064000 24717 non-null  float64 
125 TSP_162259276829213363391578010288128000 24717 non-null  float64 
126 TSP_324518553658426726783156020576256000 24717 non-null  float64 
127 TSP_649037107316853453566312041152512000 24717 non-null  float64 
128 TSP_1298074214633706907132624082305024000 24717 non-null  float64 
129 TSP_2596148429267413814265248164610048000 24717 non-null  float64 
130 TSP_5192296858534827628530496329220096000 24717 non-null  float64 
131 TSP_10384593717069655257060992658440192000 24717 non-null  float64 
132 TSP_20769187434139310514121985316880384000 24717 non-null  float64 
133 TSP_41538374868278621028243970633760768000 24717 non-null  float64 
134 TSP_83076749736557242056487941267521536000 24717 non-null  float64 
135 TSP_166153499473114484112975882535043072000 24717 non-null  float64 
136 TSP_332306998946228968225951765070086144000 24717 non-null  float64 
137 TSP_664613997892457936451903530140172288000 24717 non-null  float64 
138 TSP_1329227995784915872903807060280344576000 24717 non-null  float64 
139 TSP_2658455991569831745807614120560689152000 24717 non-null  float64 
140 TSP_5316911983139663491615228241121378304000 24717 non-null  float64 
141 TSP_10633823966279326983230456482242756608000 24717 non-null  float64 
142 TSP_21267647932558653966460912964485513216000 24717 non-null  float64 
143 TSP_42535295865117307932921825928971026432000 24717 non-null  float64 
144 TSP_85070591730234615865843651857942052864000 24717 non-null  float64 
145 TSP_170141183460469231731687303715884105728000 24717 non-null  float64 
146 TSP_340282366920938463463374607431768211456000 24717 non-null  float64 
147 TSP_680564733841876926926749214863536422912000 24717 non-null  float64 
148 TSP_1361129467683753853853498429727072845824000 24717 non-null  float64 
149 TSP_2722258935367507707706996859454145691648000 24717 non-null  float64 
150 TSP_5444517870735015415413993718908291383296000 24717 non-null  float64 
151 TSP_10889035741470030830827987437816582766592000 24717 non-null  float64 
152 TSP_21778071482940061661655974875633165533184000 24717 non-null  float64 
153 TSP_43556142965880123323311949751266331066368000 24717 non-null  float64 
154 TSP_87112285931760246646623899502532662132736000 24717 non-null  float64 
155 TSP_174224571863520493293247799005065324265472000 24717 non-null  float64 
156 TSP_348449143727040986586495598010130648530944000 24717 non-null  float64 
157 TSP_696898287454081973172991196020261297061888000 24717 non-null  float64 
158 TSP_1393796574908163946345982392040522594123776000 24717 non-null  float64 
159 TSP_2787593149816327892691964784081045188247552000 24717 non-null  float64 
160 TSP_5575186299632655785383929568162090376495104000 24717 non-null  float64 
161 TSP_11150372599265311570767859136324180752990208000 24717 non-null  float64 
162 TSP_22300745198530623141535718272648361505980416000 24717 non-null  float64 
163 TSP_44601490397061246283071436545296723011960832000 24717 non-null  float64 
164 TSP_89202980794122492566142873090593446023921664000 24717 non-null  float64 
165 TSP_178405961588244985132285746181186892047843328000 24717 non-null  float64 
166 TSP_356811923176489970264571492362373784095686656000 24717 non-null  float64 
167 TSP_713623846352979940529142984724747568191373312000 24717 non-null  float64 
168 TSP_1427247692705959881058285969449495136382746624000 24717 non-null  float64 
169 TSP_2854495385411919762116571938898990272765493248000 24717 non-null  float64 
170 TSP_5708990770823839524233143877797980545530986496000 24717 non-null  float64 
171 TSP_11417981541647679048466287755595961091061972992000 24717 non-null  float64 
172 TSP_22835963083295358096932575511191922182123945984000 24717 non-null  float64 
173 TSP_45671926166590716193865151022383844364247891968000 24717 non-null  float64 
174 TSP_91343852333181432387730302044767688728495783936000 24717 non-null  float64 
175 TSP_182687704666362864775460604089535377456991567872000 24717 non-null  float64 
176 TSP_365375409332725729550921208179070754913983135744000 24717 non-null  float64 
177 TSP_730750818665451459101842416358141509827966271488000 24717 non-null  float64 
178 TSP_1461501637330902918203684832716283019655932542976000 24717 non-null  float64 
179 TSP_2923003274661805836407369665432566039311865085952000 24717 non-null  float64 
180 TSP_5846006549323611672814739330865132078623730171904000 24717 non-null  float64 
181 TSP_11692013098647223345629478661730264157247460343808000 24717 non-null  float64 
182 TSP_23384026197294446691258957323460528314494920687616000 24717 non-null  float64 
183 TSP_46768052394588893382517914646921056628989841375232000 24717 non-null  float64 
184 TSP_93536104789177786765035829293842113257979682750464000 24717 non-null  float64 
185 TSP_187072209578355573530071658587684226515959365500928000 24717 non-null  float64 
186 TSP_374144419156711147060143317175368453031918731001856000 24717 non-null  float64 
187 TSP_748288838313422294120286634350736906063837462003712000 24717 non-null  float64 
188 TSP_1496577676626844588240573268701473812127674924007424000 24717 non-null  float64 
189 TSP_2993155353253689176481146537402947624255349848014848000 24717 non-null  float64 
190 TSP_5986310706507378352962293074805895248510699696029696000 24717 non-null  float64 
191 TSP_11972621413014756705924586149611790497021399392059392000 24717 non-null  float64 
192 TSP_23945242826029513411849172299223580994042798784118784000 24717 non-null  float64 
193 TSP_47890485652059026823698344598447161988085597568237568000 24717 non-null  float64 
194 TSP_95780971304118053647396689196894323976171195136475136000 24717 non-null  float64 
195 TSP_191561942608236107294793378393788647952342390272950272000 24717 non-null  float64 
196 TSP_383123885216472214589586756787577295904684780545900544000 24717 non-null  float64 
197 TSP_766247770432944429179173513575154591809369561091801088000 24717 non-null  float64 
198 TSP_1532495540865888858358347027150309183618739122183602176000 24717 non-null  float64 
199 TSP_3064991081731777716716694054300618367237478244367204352000 24717 non-null  float64 
200 TSP_6129982163463555433433388108601236734474956488734408704000 24717 non-null  float64 
201 TSP_12259964326927110866866776217202473468949912977468817408000 24717 non-null  float64 
202 TSP_24519928653854221733733552434404946937899825954937634816000 24717 non-null  float64 
203 TSP_49039857307708443467467104868809893875799651909875269632000 24717 non-null  float64 
204 TSP_98079714615416886934934209737619787751599303819750539264000 24717 non-null  float64 
205 TSP_196159429230833773869868419475239575503198607639501078528000 24717 non-null  float64 
206 TSP_392318858461667547739736838950479151006397215279002157056000 24717 non-null  float64 
207 TSP_784637716923335095479473677900958302012794430558004314112000 24717 non-null  float64 
208 TSP_1569275433846670190958947355801916604025588861116008628224000 24717 non-null  float64 
209 TSP_3138550867693340381917894711603833208051177722232017256448000 24717 non-null  float64 
210 TSP_6277101735386680763835789423207666416102355444464034512896000 24717 non-null  float64 
211 TSP_12554203470773361527671578846415332832204710888928069025792000 24717 non-null  float64 
212 TSP_25108406941546723055343157692830665664409421777856138051584000 24717 non-null  float64 
213 TSP_50216813883093446110686315385661331328818843555712276103168000 24717 non-null  float64 
214 TSP_100433627766186892221372630771322662657637687111424552206336000 24717 non-null  float64 
215 TSP_200867255532373784442745261542645325315275374222849104412672000 24717 non-null  float64 
216 TSP_401734511064747568885490523085290650630550748445698208825344000 24717 non-null  float64 
217 TSP_803469022129495137770981046170581301261101496891396417650688000 24717 non-null  float64 
218 TSP_1606938044258990275541962092341162602522202993782792835301376000 24717 non-null  float64 
219 TSP_3213876088517980551083924184682325205044405987565585670602752000 24717 non-null  float64 
220 TSP_6427752177035961102167848369364650410088811975131171341205504000 24717 non-null  float64 
221 TSP_12855504354071922204335696738729300820177623950262342682411008000 24717 non-null  float64 
222 TSP_25711008708143844408671393477458601640355247900524685364822016000 24717 non-null  float64 
223 TSP_51422017416287688817342786954917203280710495801049370729644032000 24717 non-null  float64 
224 TSP_102844034832575377634685573909834406561420991602098741459288064000 24717 non-null  float64 
225 TSP_205688069665150755269371147819668813122841983204197482918576128000 24717 non-null  float64 
226 TSP_411376139330301510538742295639337626245683966408394965837152256000 24717 non-null  float64 
227 TSP_822752278660603021077484591278675252491367932816789931674304512000 24717 non-null  float64 
228 TSP_1645504557321206042154969182557350504982735865633579863348609024000 24717 non-null  float64 
229 TSP_32910091146424120843099383651147
```

In [17]:

```
df.describe()
```

Out[17]:

| | BEN | CO | EBE | MXY | NMHC | NO_2 |
|-------|--------------|--------------|--------------|--------------|--------------|--------------|
| count | 24717.000000 | 24717.000000 | 24717.000000 | 24717.000000 | 24717.000000 | 24717.000000 |
| mean | 1.010583 | 0.448056 | 1.262430 | 2.244469 | 0.219582 | 55.563929 |
| std | 1.007345 | 0.291706 | 1.074768 | 2.242214 | 0.141661 | 38.911677 |
| min | 0.170000 | 0.060000 | 0.250000 | 0.240000 | 0.000000 | 0.600000 |
| 25% | 0.460000 | 0.270000 | 0.720000 | 0.990000 | 0.140000 | 26.510000 |
| 50% | 0.670000 | 0.370000 | 1.000000 | 1.490000 | 0.190000 | 47.930000 |
| 75% | 1.180000 | 0.570000 | 1.430000 | 2.820000 | 0.260000 | 76.269997 |
| max | 22.379999 | 5.570000 | 47.669998 | 56.500000 | 2.580000 | 477.399994 |

In [18]:

```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',  
        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

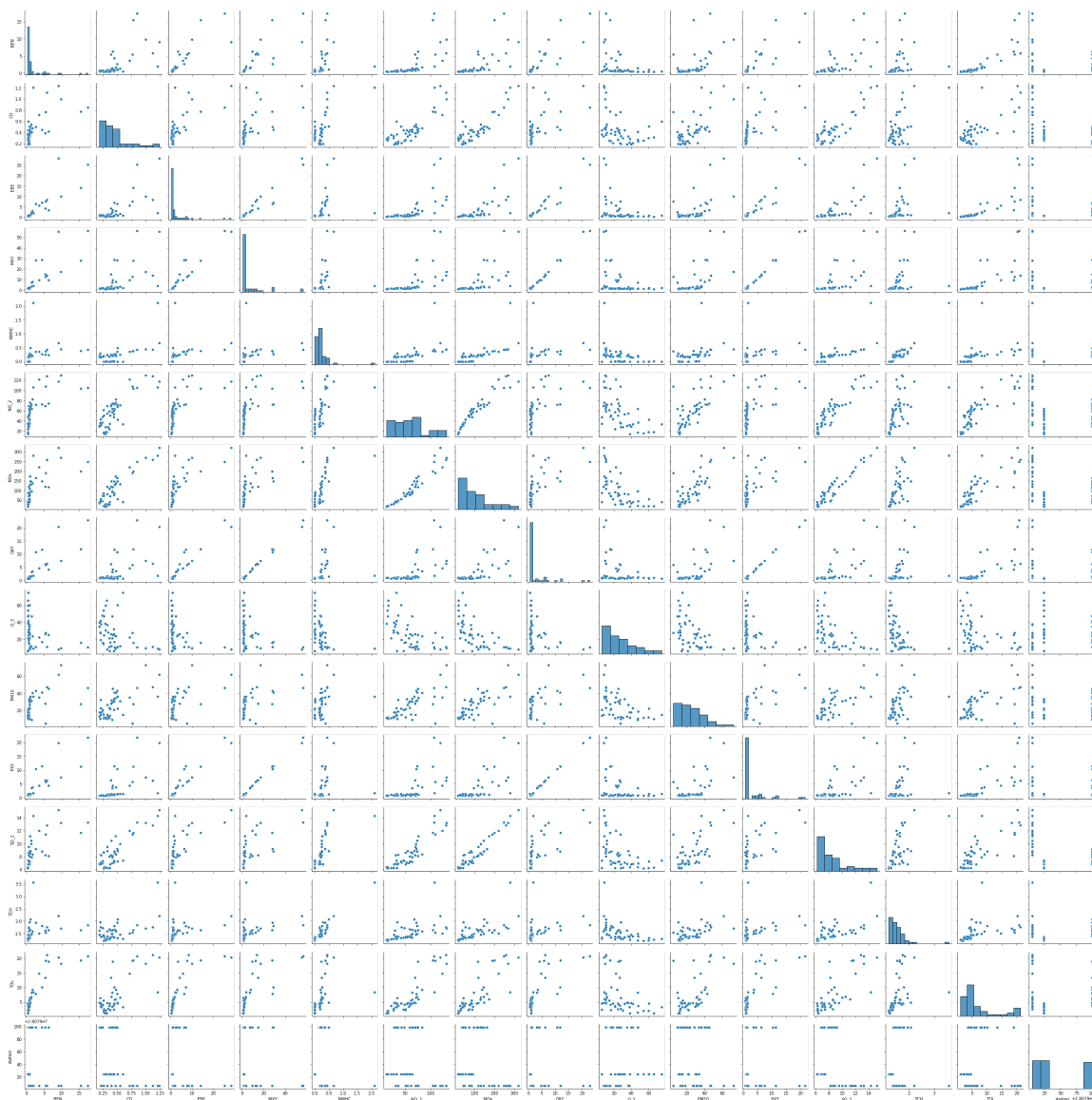
EDA AND VISUALIZATION

In [19]:

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

Out[19]:

<seaborn.axisgrid.PairGrid at 0x1c686fd3880>



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)
```

Linear Regression

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]:

28078897.431059588

In [26]:

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

Out[26]:

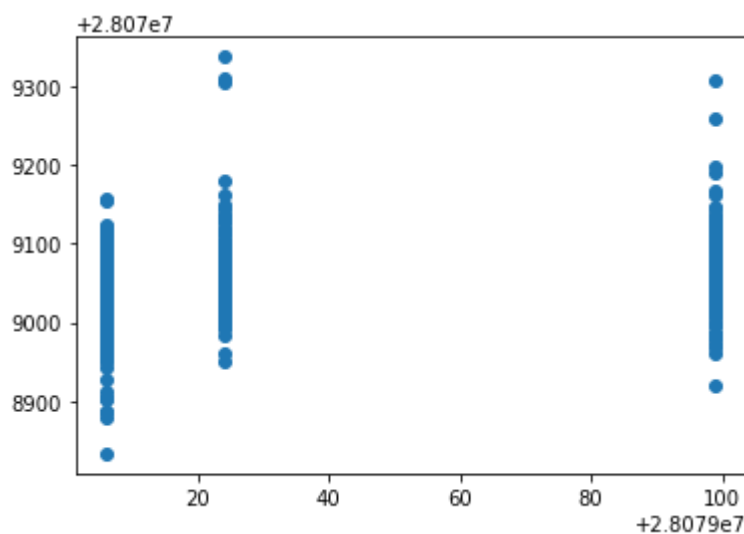
| | Co-efficient |
|-------------|--------------|
| BEN | -35.490221 |
| CO | -28.569434 |
| EBE | 6.280297 |
| MXY | -0.514729 |
| NMHC | -17.687543 |
| NO_2 | -0.182322 |
| NOx | 0.202518 |
| OXY | 13.116387 |
| O_3 | 0.018908 |
| PM10 | -0.054712 |
| PXY | 2.181191 |
| SO_2 | -0.365033 |
| TCH | 121.852124 |
| TOL | -1.192129 |

In [27]:

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

Out[27]:

<matplotlib.collections.PathCollection at 0x1c695866550>



ACCURACY

In [28]:

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

Out[28]:

0.28907770904191143

In [29]:

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

Out[29]:

0.2866577907132045

Ridge and Lasso

In [30]:

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

In [31]:

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

Out[31]:

Ridge(alpha=10)

Accuracy(Ridge)

In [32]:

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

Out[32]:

```
0.2895379984598364
```

In [33]:

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

Out[33]:

```
0.2863332584799254
```

In [34]:

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

Out[34]:

```
Lasso(alpha=10)
```

In [35]:

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

Out[35]:

```
0.036780247807414734
```

Accuracy(Lasso)

In [36]:

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

Out[36]:

```
0.03515746933141528
```

Elastic Net

In [37]:

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

Out[37]:

```
ElasticNet()
```

In [38]:

```
en.coef_
```

Out[38]:

```
array([-7.07154932, -0.65489885,  0.31774118,  2.09529124,  0.         ,
        -0.22374956,  0.12687927,  1.1911961 , -0.14587028,  0.08206418,
         1.72109062, -0.78899961,  1.494183   , -1.93521422])
```

In [39]:

```
en.intercept_
```

Out[39]:

```
28079064.228753425
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.10785348119567184
```

Evaluation Metrics

In [42]:

```
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)))
```

```
35.8310913665405
```

```
1454.217905501406
```

```
38.13420912384844
```

Logistic Regression

In [43]:

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

In [44]:

```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXV', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
                  'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```


In [45]:

```
feature_matrix.shape
```

Out[45]:

```
(24717, 14)
```

In [46]:

```
target_vector.shape
```

Out[46]:

```
(24717,)
```

In [47]:

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

In [48]:

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

In [49]:

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

Out[49]:

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

In [50]:

```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [51]:

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

```
[28079099]
```

In [52]:

```
logr.classes_
```

Out[52]:

```
array([28079006, 28079024, 28079099], dtype=int64)
```

In [53]:

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

Out[53]:

```
0.8951733624630821
```

In [54]:

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

Out[54]:

```
5.447205522232353e-13
```

In [55]:

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

Out[55]:

```
array([[5.44720552e-13, 8.28692830e-44, 1.00000000e+00]])
```

Random Forest

In [56]:

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

In [57]:

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

Out[57]:

```
RandomForestClassifier()
```

In [58]:

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

In [59]:

```
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[59]:

```
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 [60]:

```
grid_search.best_score_
```

Out[60]:

```
0.8911047204272553
```

In [61]:

```
rfc_best=grid_search.best_estimator_
```

In [62]:

```
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'],f
```

Out[62]:

```

[Text(2241.3, 1993.2, 'EBE <= 1.005\ngini = 0.665\nsamples = 10912\nvalue
= [5173, 5941, 6187]\nclass = c'),
Text(1190.4, 1630.8000000000002, 'SO_2 <= 6.815\ngini = 0.585\nsamples =
6041\nvalue = [1564, 5348, 2714]\nclass = b'),
Text(595.2, 1268.4, 'NMHC <= 0.145\ngini = 0.111\nsamples = 1813\nvalue =
[25, 2711, 143]\nclass = b'),
Text(297.6, 906.0, 'SO_2 <= 6.395\ngini = 0.321\nsamples = 443\nvalue =
[24, 555, 108]\nclass = b'),
Text(148.8, 543.5999999999999, 'NMHC <= 0.035\ngini = 0.039\nsamples = 27
0\nvalue = [6, 400, 2]\nclass = b'),
Text(74.4, 181.19999999999982, 'gini = 0.223\nsamples = 31\nvalue = [6, 4
1, 0]\nclass = b'),
Text(223.20000000000002, 181.19999999999982, 'gini = 0.011\nsamples = 239
\nvalue = [0, 359, 2]\nclass = b'),
Text(446.40000000000003, 543.5999999999999, 'OXY <= 1.005\ngini = 0.543\n
samples = 173\nvalue = [18, 155, 106]\nclass = b'),
Text(372.0, 181.19999999999982, 'gini = 0.513\nsamples = 151\nvalue = [1
8, 152, 73]\nclass = b'),
Text(520.80000000000001, 181.19999999999982, 'gini = 0.153\nsamples = 22\n
value = [0, 3, 33]\nclass = c'),
Text(892.80000000000001, 906.0, 'SO_2 <= 6.695\ngini = 0.032\nsamples = 13
70\nvalue = [1, 2156, 35]\nclass = b'),
Text(744.0, 543.5999999999999, 'OXY <= 0.815\ngini = 0.008\nsamples = 124
8\nvalue = [0, 1989, 8]\nclass = b'),
Text(669.6, 181.19999999999982, 'gini = 0.035\nsamples = 212\nvalue = [0,
328, 6]\nclass = b'),
Text(818.40000000000001, 181.19999999999982, 'gini = 0.002\nsamples = 1036
\nvalue = [0, 1661, 2]\nclass = b'),
Text(1041.60000000000001, 543.5999999999999, 'EBE <= 0.685\ngini = 0.247\n
samples = 122\nvalue = [1, 167, 27]\nclass = b'),
Text(967.2, 181.19999999999982, 'gini = 0.022\nsamples = 54\nvalue = [1,
88, 0]\nclass = b'),
Text(1116.0, 181.19999999999982, 'gini = 0.38\nsamples = 68\nvalue = [0,
79, 27]\nclass = b'),
Text(1785.60000000000001, 1268.4, 'NMHC <= 0.105\ngini = 0.65\nsamples = 4
228\nvalue = [1539, 2637, 2571]\nclass = b'),
Text(1488.0, 906.0, 'NOx <= 23.105\ngini = 0.433\nsamples = 873\nvalue =
[982, 365, 45]\nclass = a'),
Text(1339.2, 543.5999999999999, 'TOL <= 1.54\ngini = 0.37\nsamples = 137
\nvalue = [38, 174, 13]\nclass = b'),
Text(1264.80000000000002, 181.19999999999982, 'gini = 0.225\nsamples = 114
\nvalue = [10, 162, 13]\nclass = b'),
Text(1413.60000000000001, 181.19999999999982, 'gini = 0.42\nsamples = 23\n
value = [28, 12, 0]\nclass = a'),
Text(1636.80000000000002, 543.5999999999999, 'TCH <= 1.435\ngini = 0.318\n
samples = 736\nvalue = [944, 191, 32]\nclass = a'),
Text(1562.4, 181.19999999999982, 'gini = 0.231\nsamples = 666\nvalue = [9
21, 105, 32]\nclass = a'),
Text(1711.2, 181.19999999999982, 'gini = 0.333\nsamples = 70\nvalue = [2
3, 86, 0]\nclass = b'),
Text(2083.20000000000003, 906.0, 'CO <= 0.565\ngini = 0.587\nsamples = 335
5\nvalue = [557, 2272, 2526]\nclass = c'),
Text(1934.4, 543.5999999999999, 'NMHC <= 0.275\ngini = 0.565\nsamples = 2
778\nvalue = [495, 1446, 2495]\nclass = c'),
Text(1860.00000000000002, 181.19999999999982, 'gini = 0.546\nsamples = 256
3\nvalue = [494, 1150, 2478]\nclass = c'),
Text(2008.80000000000002, 181.19999999999982, 'gini = 0.108\nsamples = 215
\nvalue = [1, 296, 17]\nclass = b'),
Text(2232.0, 543.5999999999999, 'NO_2 <= 79.245\ngini = 0.186\nsamples =
577\nvalue = [62, 826, 31]\nclass = b'),
Text(2157.60000000000004, 181.19999999999982, 'gini = 0.04\nsamples = 451

```

```

\value = [10, 715, 5]\nclclass = b'),
Text(2306.4, 181.1999999999982, 'gini = 0.56\nsamples = 126\nvalue = [5
2, 111, 26]\nclclass = b'),
Text(3292.20000000000003, 1630.8000000000002, 'NOx <= 112.5\ngini = 0.568
\nsamples = 4871\nvalue = [3609, 593, 3473]\nclclass = a'),
Text(2715.60000000000004, 1268.4, 'NMHC <= 0.125\ngini = 0.503\nsamples =
2143\nvalue = [849, 325, 2189]\nclclass = c'),
Text(2455.20000000000003, 906.0, 'NOx <= 28.335\ngini = 0.222\nsamples = 4
76\nvalue = [637, 52, 37]\nclclass = a'),
Text(2380.8, 543.5999999999999, 'gini = 0.648\nsamples = 29\nvalue = [11,
19, 13]\nclclass = b'),
Text(2529.60000000000004, 543.5999999999999, 'NOx <= 57.965\ngini = 0.156
\nsamples = 447\nvalue = [626, 33, 24]\nclclass = a'),
Text(2455.20000000000003, 181.1999999999982, 'gini = 0.355\nsamples = 109
\nvalue = [130, 13, 22]\nclclass = a'),
Text(2604.0, 181.1999999999982, 'gini = 0.082\nsamples = 338\nvalue = [4
96, 20, 2]\nclclass = a'),
Text(2976.0, 906.0, 'NMHC <= 0.355\ngini = 0.317\nsamples = 1667\nvalue =
[212, 273, 2152]\nclclass = c'),
Text(2827.20000000000003, 543.5999999999999, 'CO <= 0.555\ngini = 0.276\ns
amples = 1613\nvalue = [212, 186, 2148]\nclclass = c'),
Text(2051.1999999999998, 'gini = 0.248\nsamples = 1579\nvalue =
[208, 138, 2145]\nclclass = c'),
Text(2901.60000000000004, 181.1999999999982, 'gini = 0.23\nsamples = 34\n
value = [4, 48, 3]\nclclass = b'),
Text(3124.8, 543.5999999999999, 'O_3 <= 31.03\ngini = 0.084\nsamples = 54
\nvalue = [0, 87, 4]\nclclass = b'),
Text(3050.4, 181.1999999999982, 'gini = 0.0\nsamples = 26\nvalue = [0, 5
0, 0]\nclclass = b'),
Text(3199.20000000000003, 181.1999999999982, 'gini = 0.176\nsamples = 28
\nvalue = [0, 37, 4]\nclclass = b'),
Text(3386.8, 1268.4, 'TCH <= 1.495\ngini = 0.498\nsamples = 2728\nvalue =
[2760, 268, 1284]\nclclass = a'),
Text(3571.20000000000003, 906.0, 'BEN <= 1.025\ngini = 0.265\nsamples = 11
33\nvalue = [1518, 60, 211]\nclclass = a'),
Text(3422.4, 543.5999999999999, 'OXY <= 1.565\ngini = 0.516\nsamples = 15
0\nvalue = [289, 70, 219, 143, 147]\nclclass = c'),
Text(3348.00000000000005, 181.1999999999982, 'gini = 0.549\nsamples = 112
\nvalue = [91, 10, 80]\nclclass = a'),
Text(3496.8, 181.1999999999982, 'gini = 0.106\nsamples = 43\nvalue = [4,
0, 67]\nclclass = c'),
In [64]: Ir.Score(x_train, y_train)
Text(3720.00000000000005, 543.5999999999999, 'O_3 <= 6.07\ngini = 0.14\nsa
mples = 978\nvalue = [1423, 50, 64]\nclclass = a'),
Text(3645.60000000000004, 181.1999999999982, 'gini = 0.365\nsamples = 35
0\nvalue = [286, 72, 90, 13, 245, 2]\nclclass = b'),
Text(3794.4, 181.1999999999982, 'gini = 0.092\nsamples = 943\nvalue = [1
413, 9, 62]\nclclass = a'),
Text(4166.40000000000001, 906.0, 'MXV <= 4.005\ngini = 0.57\nsamples = 1595
\nvalue = [1242, 208, 1073]\nclclass = a'),
Text(4017.60000000000004, 543.5999999999999, 'O_3 <= 5.31\ngini = 0.569\ns
amples = 698\nvalue = [337, 146, 636]\nclclass = c'),
Text(3943.20000000000003, 181.1999999999982, 'gini = 0.49\nsamples = 89\n
value = [80, 50, 0]\nclclass = a'),
Text(4092.00000000000005, 181.1999999999982, 'gini = 0.501\nsamples = 609
\nvalue = [257, 86, 636]\nclclass = c'),
Text(4315.20000000000001, 543.5999999999999, 'O_3 <= 5.76\ngini = 0.486\nsa
mples = 897\nvalue = [905, 62, 437]\nclclass = a'),
Text(4240.8, 181.1999999999982, 'gini = 0.296\nsamples = 143\nvalue = [1
83, 39, 1]\nclclass = a'),
Text(4099.6, 181.1999999999982, 'gini = 0.49\nsamples = 754\nvalue = [72
2, 23, 436]\nclclass = a')]
```

Conclusion

Scores

Linear Regression

Out[63]:

In [64]:

Out[64]:

Lasso

In [65]:

Out[65]:

Ridge

In [66]:

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

Out[66]:

0.2895379984598364

In [67]:

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

Out[67]:

0.2863332584799254

Elastic Net

In [68]:

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

Out[68]:

0.10785348119567184

Logistic Regression

In [69]:

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

Out[69]:

0.8951733624630821

Random Forest

In [70]:

```
grid_search.best_score_
```

Out[70]:

0.8911047204272553

From the above data, we can conclude that logistic regression is preferable to other regression types

In []:

