In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression,LogisticRegressio
from sklearn.model_selection import train_test_split
```

In [2]:
```
df=pd.read_csv("/Users/bob/Downloads/FP1_air/csvs_per_year/csvs_per
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998
...	...	...	...	...	...	...	...	...	...	...
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000

217872 rows × 16 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

In [4]:
```
df1=df.dropna()
df1
```

Out[4]:

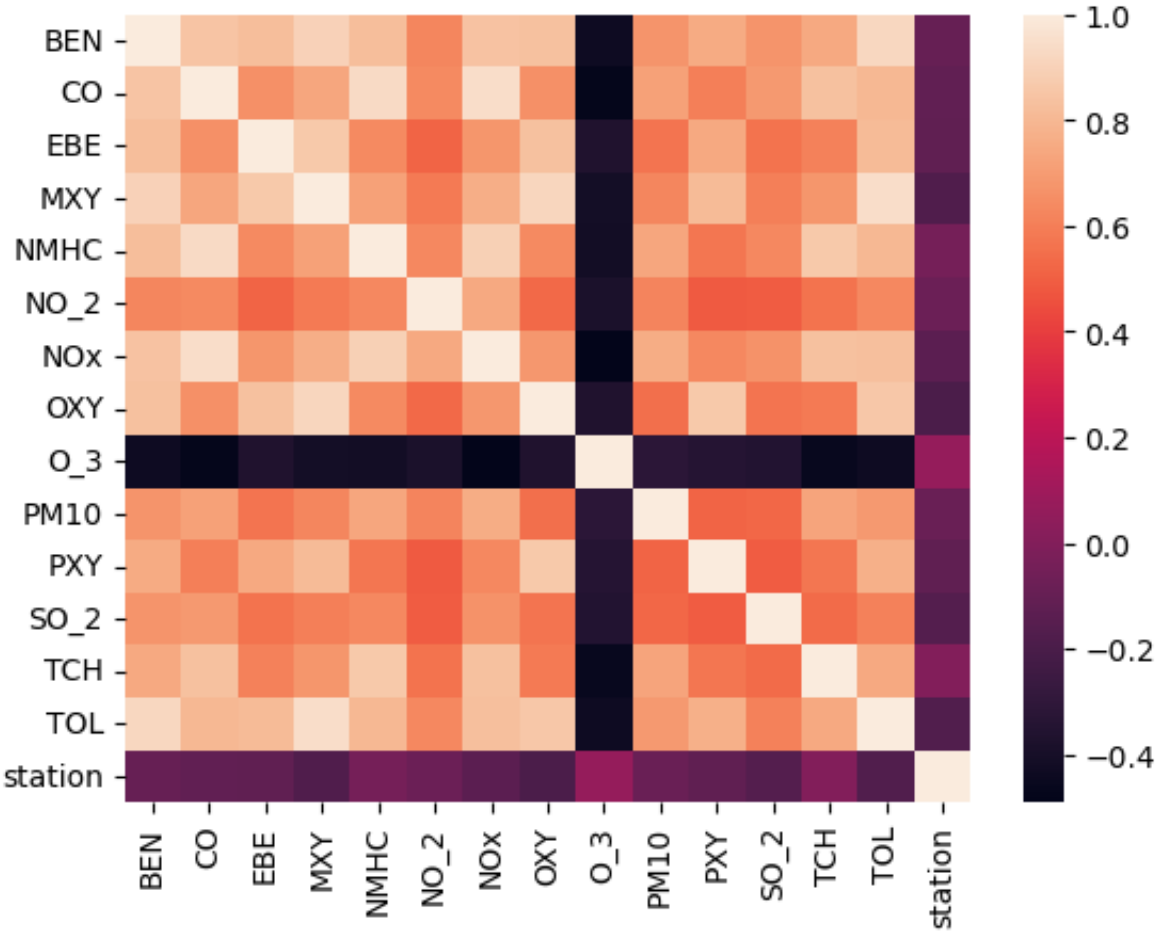
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.100000	0.07	56.250000	75.169998	2.11	42.
5	2001-08-01 01:00:00	2.11	0.63	2.48	5.940000	0.05	66.260002	118.099998	3.15	33.
21	2001-08-01 01:00:00	0.80	0.43	0.71	1.200000	0.10	27.190001	29.700001	0.76	56.
23	2001-08-01 01:00:00	1.29	0.34	1.41	3.090000	0.07	40.750000	51.570000	1.70	51.
25	2001-08-01 02:00:00	0.87	0.06	0.88	2.410000	0.01	29.709999	31.440001	1.20	56.
...	...	...	...	...	...	...	...	...	...	
217829	2001-03-31 23:00:00	11.76	4.48	7.71	17.219999	0.89	103.900002	548.500000	7.62	9.
217847	2001-03-31 23:00:00	9.79	2.65	7.59	9.730000	0.46	91.320000	315.899994	3.75	6.
217849	2001-04-01 00:00:00	5.86	1.22	5.66	13.710000	0.25	64.370003	218.300003	6.46	7.
217853	2001-04-01 00:00:00	14.47	1.83	11.39	26.059999	0.33	84.230003	259.200012	11.39	5.
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.040000	0.18	76.809998	206.300003	5.20	8.

29669 rows × 16 columns

In [5]:
```
df1=df1.drop(["date"],axis=1)
```

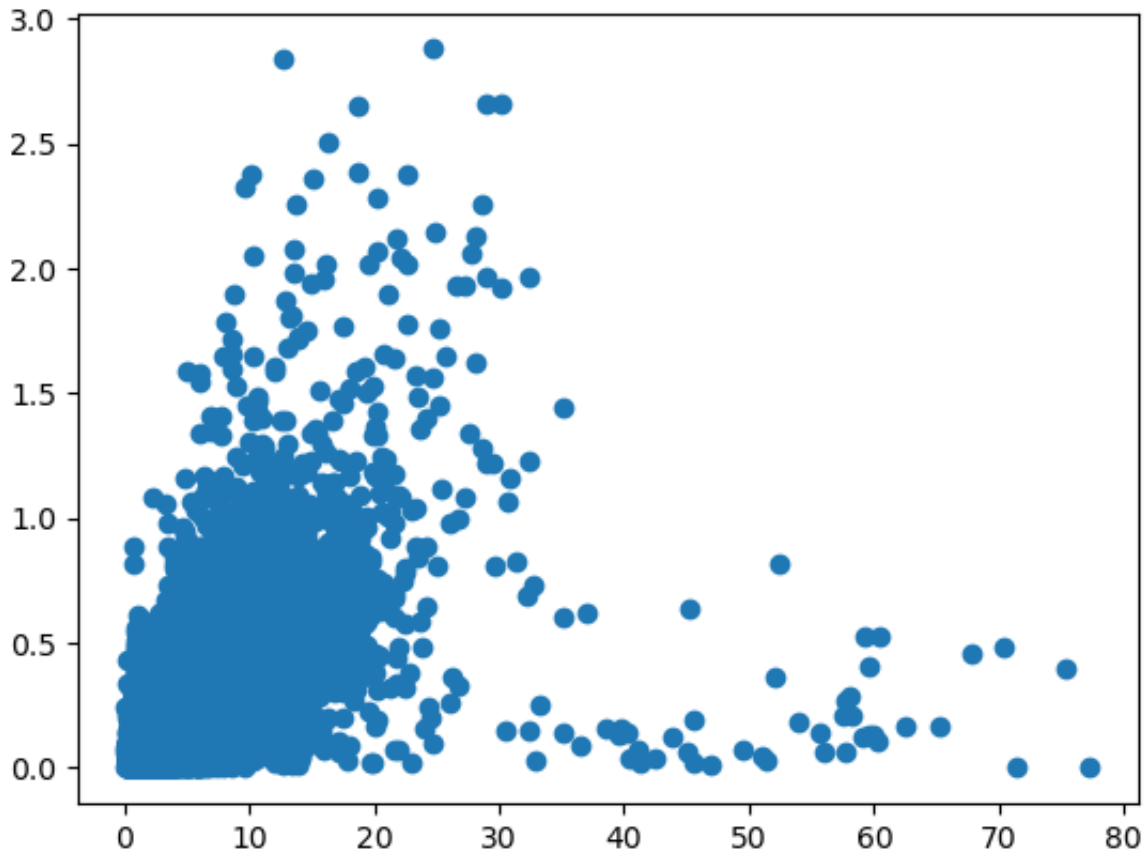
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <Axes: >



In [7]:
```
plt.plot(df1["EBE"],df1["NMHC"],"o")
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x7fafc990f310>]
```



In [8]:
```
data=df[["EBE","NMHC"]]
```

In [9]:
```
x=df1.drop(["EBE"],axis=1)
y=df1["EBE"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

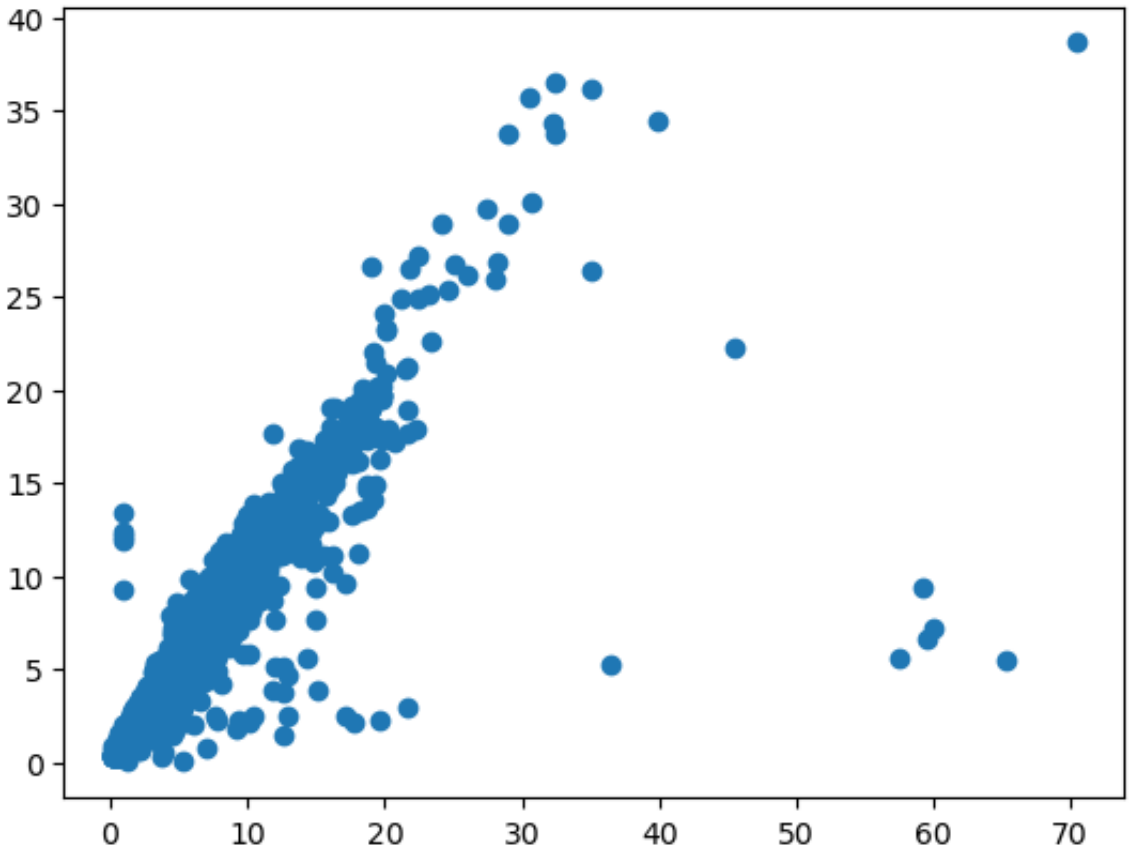
Linear

In [10]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[10]:
```
▾ LinearRegression
LinearRegression()
```

In [11]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[11]: <matplotlib.collections.PathCollection at 0x7fafc99c7310>



In [12]:
```
lis=li.score(x_test,y_test)
```

In [13]:
```
df1["TCH"].value_counts()
```

Out[13]:
```
1.28    988
1.32    938
1.29    908
1.33    908
1.27    905
       ...
4.51      1
3.88      1
4.13      1
3.22      1
3.57      1
Name: TCH, Length: 269, dtype: int64
```

In [14]:
```
df1.loc[df1["TCH"]<1.40,"TCH"]=1
df1.loc[df1["TCH"]>1.40,"TCH"]=2
df1["TCH"].value_counts()
```

Out[14]:
```
1.0    17204
2.0    12465
Name: TCH, dtype: int64
```

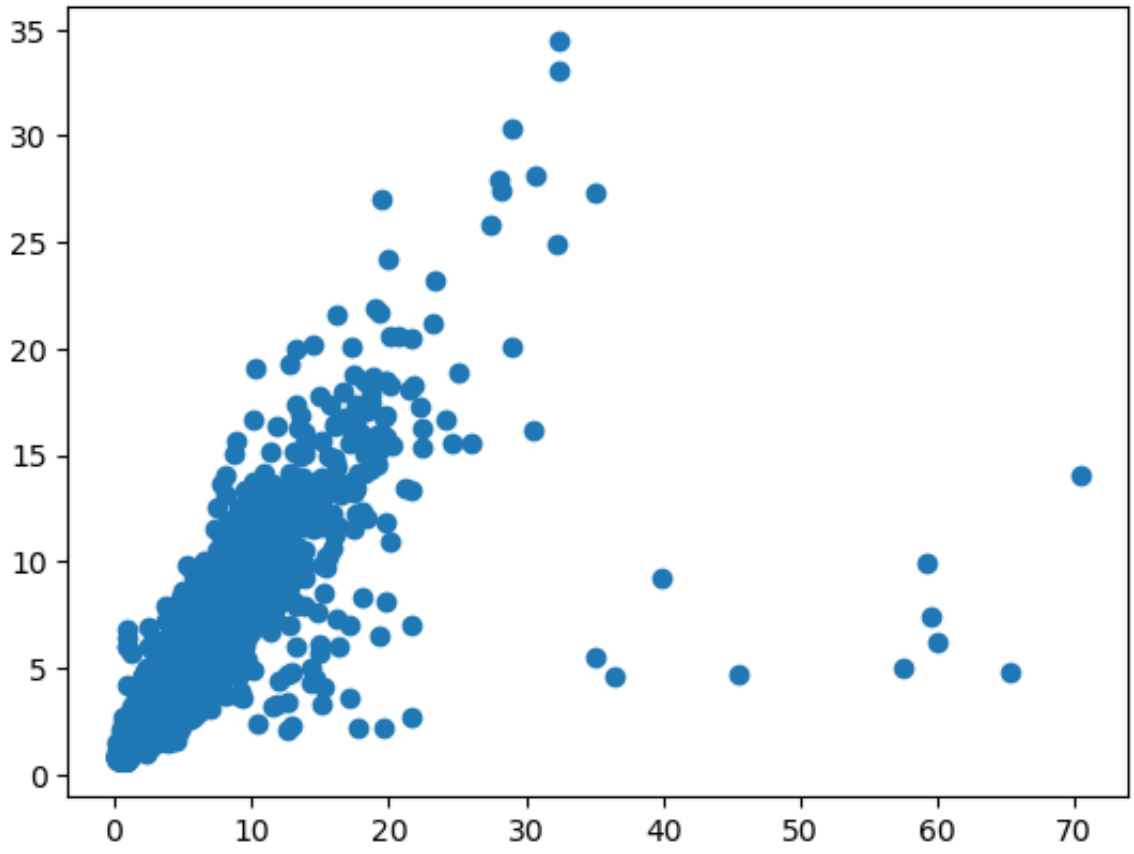
Lasso

In [15]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[15]:
```
Lasso
Lasso(alpha=5)
```

In [16]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x7fafb89eb370>
```



In [17]:
```
las=la.score(x_test,y_test)
```

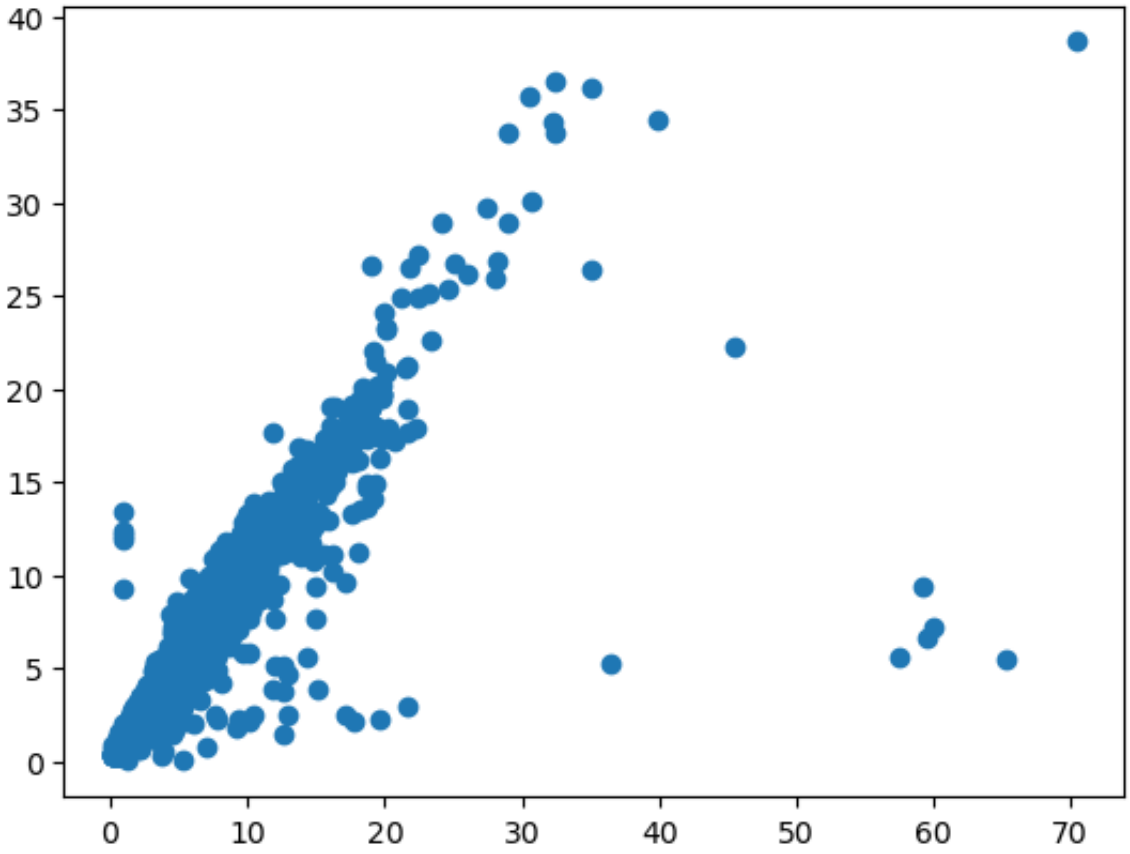
Ridge

In [18]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[18]:
```
Ridge
Ridge(alpha=1)
```

In [19]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[19]:
```
<matplotlib.collections.PathCollection at 0x7fafb8a54e80>
```



In [20]:
```
rrs=rr.score(x_test,y_test)
```

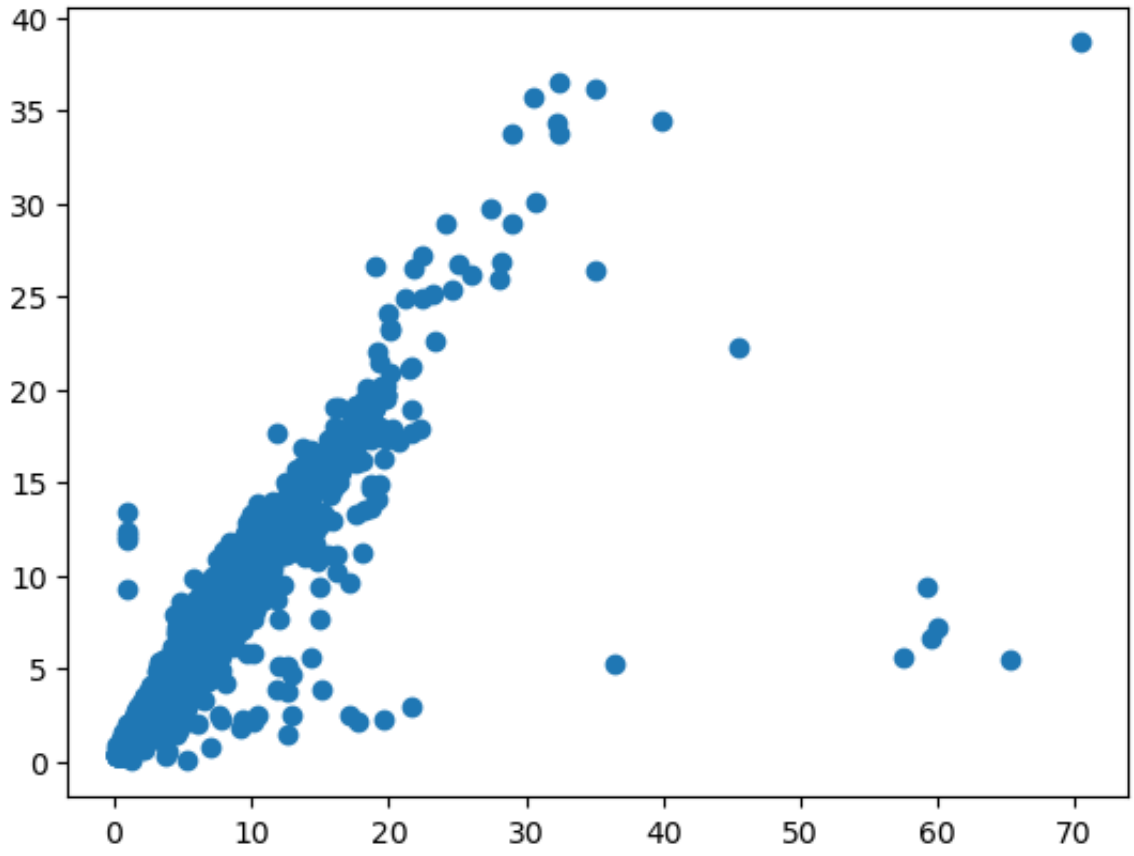
ElasticNet

In [21]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[21]:
```
ElasticNet
ElasticNet()
```

In [22]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x7fafc9a36d10>
```



In [23]:
```
ens=en.score(x_test,y_test)
```

In [24]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.802413483222234
```

Out[24]:
```
0.7554666374129868
```

Logistic

In [25]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["TCH"].value_counts()
```

Out[25]:
```
Low     17204
High    12465
Name: TCH, dtype: int64
```

In [26]:
```
x=df1.drop(["TCH"],axis=1)
y=df1["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

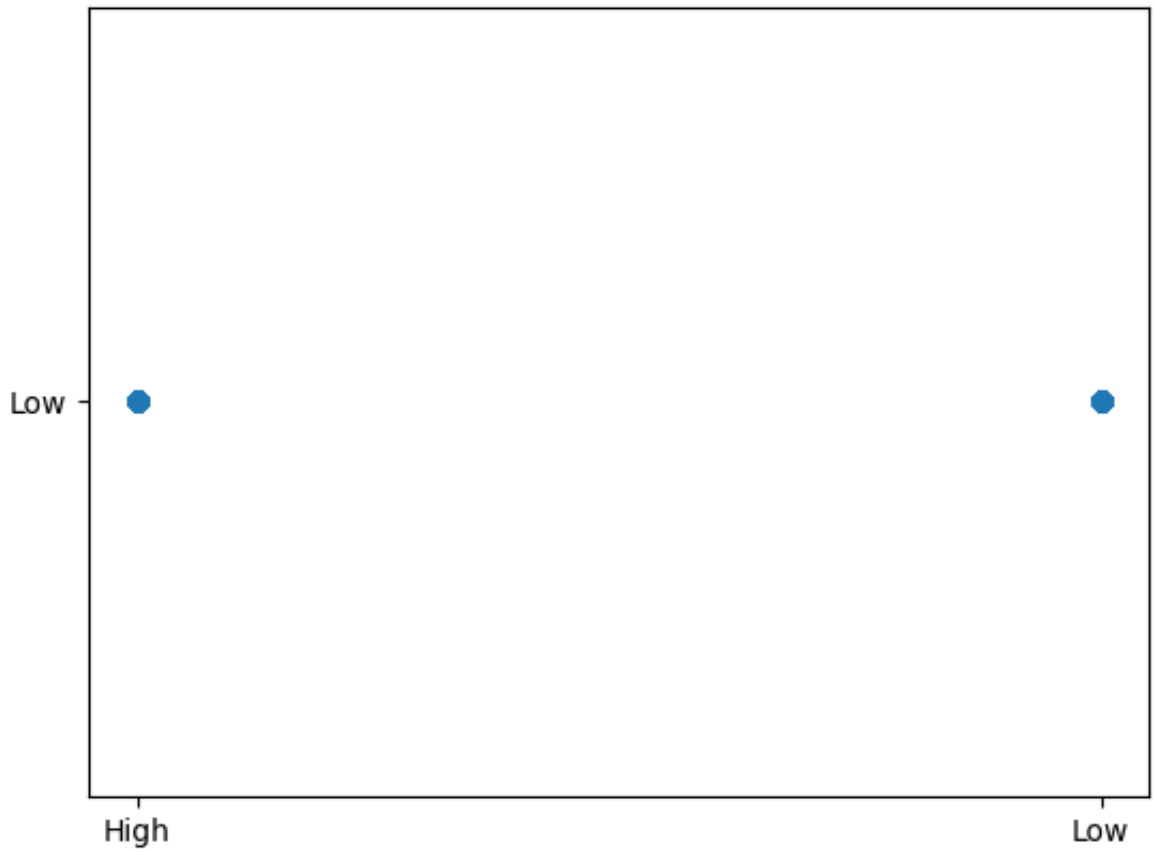
In [27]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[27]:
```
▾ LogisticRegression

LogisticRegression()
```

In [28]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x7faf98106d10>
```



In [29]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [31]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df1=df1.replace(g1)
```

In [32]:
```
x=df1.drop(["TCH"],axis=1)
y=df1["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [33]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[33]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [34]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [35]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,sc
grid_search.fit(x_train,y_train)
```

Out[35]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [36]:
```
rfcs=grid_search.best_score_
```

In [37]:
```
rfc_best=grid_search.best_estimator_
```

In [38]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_nam
```

Out[38]:
```
[Text(0.49318181818181817, 0.9285714285714286, 'NOx <= 167.25\ngin
i = 0.488\nsamples = 13134\nvalue = [12003, 8765]\nclass = Yes'),
 Text(0.23977272727272728, 0.7857142857142857, 'PXY <= 2.165\ngini
= 0.303\nsamples = 8414\nvalue = [10834, 2474]\nclass = Yes'),
 Text(0.13863636363636364, 0.6428571428571429, 'CO <= 0.785\ngini
= 0.214\nsamples = 5776\nvalue = [8000, 1112]\nclass = Yes'),
 Text(0.07272727272727272, 0.5, 'NMHC <= 0.155\ngini = 0.14\nsampl
es = 5176\nvalue = [7555, 618]\nclass = Yes'),
 Text(0.03636363636363636, 0.35714285714285715, 'CO <= 0.635\ngini
= 0.098\nsamples = 4890\nvalue = [7342, 398]\nclass = Yes'),
 Text(0.01818181818181818, 0.21428571428571427, 'O_3 <= 8.745\ngin
i = 0.086\nsamples = 4276\nvalue = [6459, 305]\nclass = Yes'),
 Text(0.00909090909090909, 0.07142857142857142, 'gini = 0.47\nsamp
les = 141\nvalue = [129, 78]\nclass = Yes'),
 Text(0.02727272727272727, 0.07142857142857142, 'gini = 0.067\nsam
ples = 4135\nvalue = [6330, 227]\nclass = Yes'),
 Text(0.05454545454545454, 0.21428571428571427, 'O_3 <= 10.615\ngi
ni = 0.172\nsamples = 614\nvalue = [883, 93]\nclass = Yes'),
 Text(0.045454545454545456, 0.07142857142857142, 'gini = 0.487\nsa
```

In [39]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.8024076584799325
Lasso: 0.7023527090369962
Ridge: 0.802413483222234
ElasticNet: 0.7942519438100765
Logistic: 0.5820694304010785
Random Forest: 0.9120762711864407
```

Best Model is Random Forest

In [41]:
```
df2=pd.read_csv("/Users/bob/Downloads/FP1_air/csvs_per_year/csvs_pe
df2
```

Out[41]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.9
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.9
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.4
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.1
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.3
...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.7
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.3
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.6
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.2

217296 rows × 16 columns

In [42]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217296 entries, 0 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217296 non-null  object 
 1   BEN      66747 non-null   float64
 2   CO       216637 non-null  float64
 3   EBE      58547 non-null   float64
 4   MXY      41255 non-null   float64
 5   NMHC     87045 non-null   float64
 6   NO_2     216439 non-null  float64
 7   NOx      216439 non-null  float64
 8   OXY      41314 non-null   float64
 9   O_3      216726 non-null  float64
 10  PM10     209113 non-null  float64
 11  PXY      41256 non-null   float64
 12  SO_2     216507 non-null  float64
 13  TCH      87115 non-null   float64
 14  TOL      66619 non-null   float64
 15  station  217296 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.5+ MB
```

In [43]:
```
df3=df2.dropna()
df3
```

Out[43]:

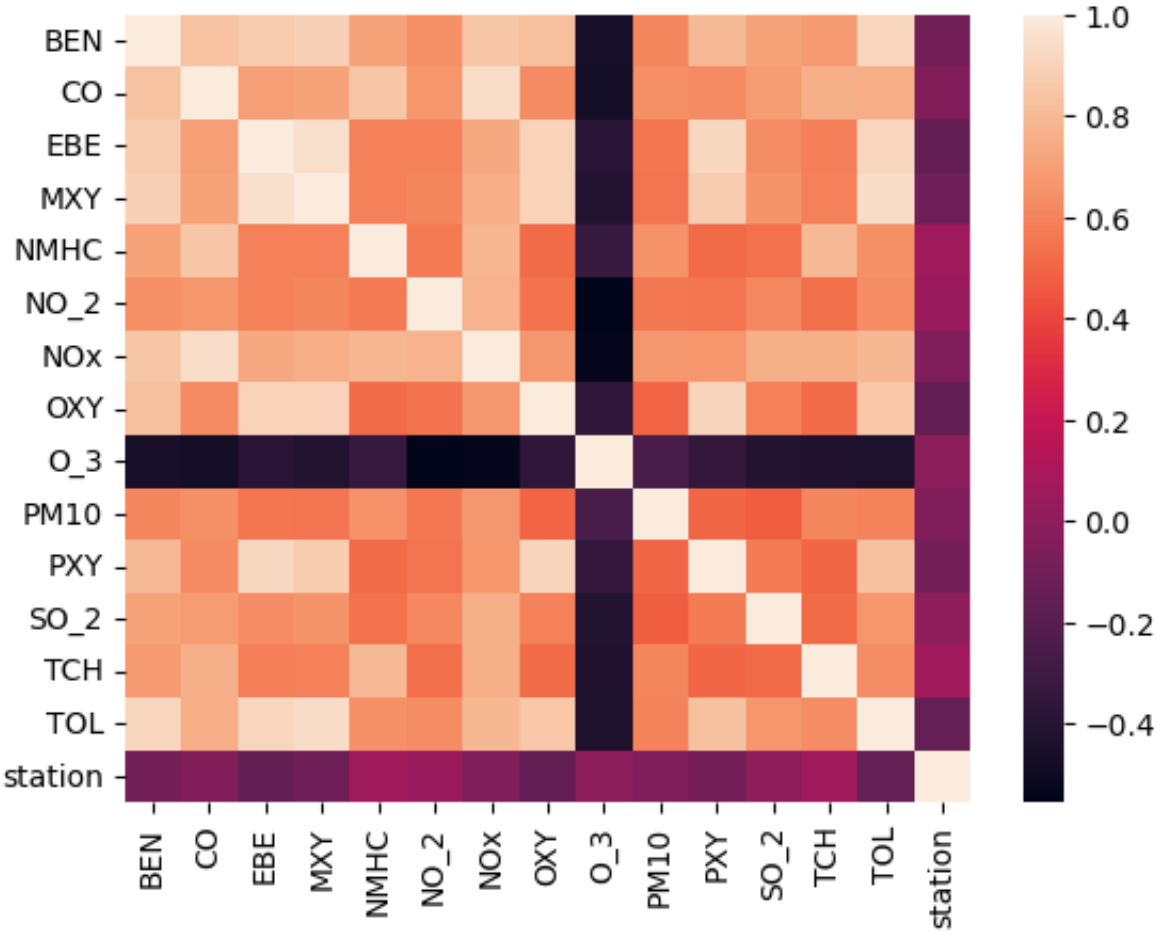
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.9
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.37	27.4
22	2002-04-01 01:00:00	2.02	0.80	1.57	3.66	0.15	93.860001	101.300003	1.77	6.99	33.0
24	2002-04-01 01:00:00	3.02	1.04	2.43	5.38	0.21	103.699997	195.399994	2.15	14.04	37.3
26	2002-04-01 02:00:00	2.02	0.53	2.24	5.97	0.12	91.599998	136.199997	2.55	6.76	19.9
...	...	...	...	...	...	...	...	...	...	...	
217269	2002-10-31 23:00:00	1.24	0.28	1.26	2.64	0.11	60.080002	64.160004	1.23	15.64	13.9
217271	2002-10-31 23:00:00	3.13	1.30	2.93	7.90	0.28	84.779999	184.000000	2.23	7.94	32.5
217273	2002-11-01 00:00:00	2.50	0.97	3.63	9.95	0.19	61.759998	132.100006	4.46	5.45	29.5
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.6
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.2

32381 rows × 16 columns

In [44]:
```
df3=df3.drop(["date"],axis=1)
```

In [45]:
```
sns.heatmap(df3.corr())
```

Out[45]: <Axes: >



In [46]:
```
x=df3.drop(["TCH"],axis=1)
y=df3["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

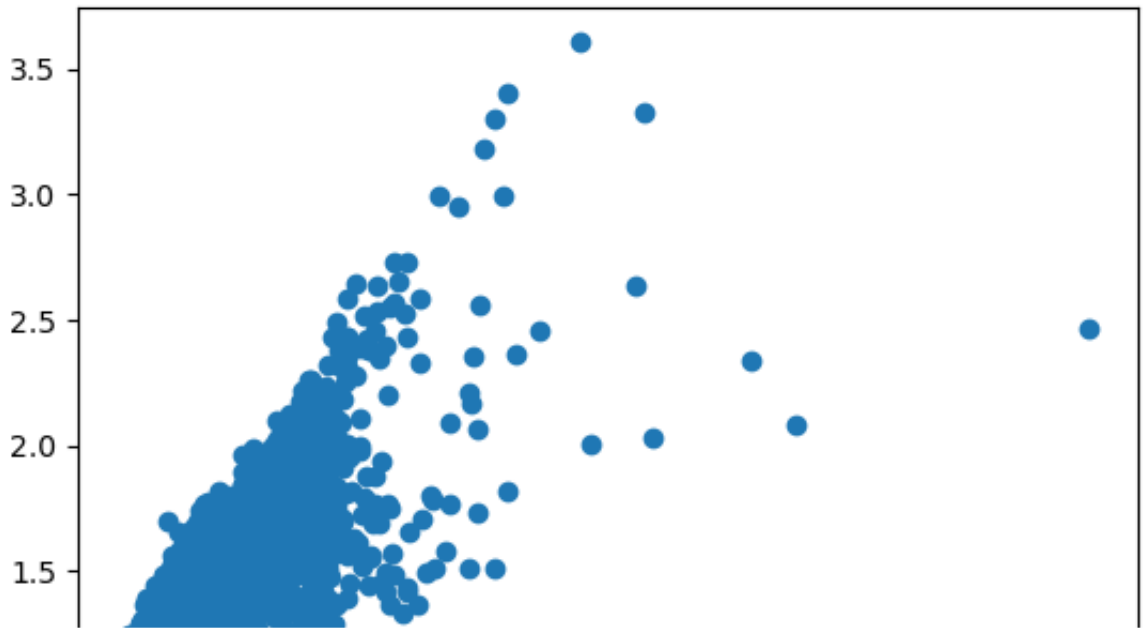
Linear

In [47]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[47]:
```
▾ LinearRegression
LinearRegression()
```

In [48]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[48]:
```
<matplotlib.collections.PathCollection at 0x7fafd9241c30>
```



In [49]:
```
lis=li.score(x_test,y_test)
```

In [50]:
```
df3["TCH"].value_counts()
```

Out[50]:
```
1.29    1318
1.30    1253
1.27    1244
1.28    1232
1.31    1187
        ...
3.68       1
3.48       1
3.41       1
3.01       1
2.51       1
Name: TCH, Length: 232, dtype: int64
```

In [51]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[51]:
```
1.0    21925
2.0    10456
Name: TCH, dtype: int64
```

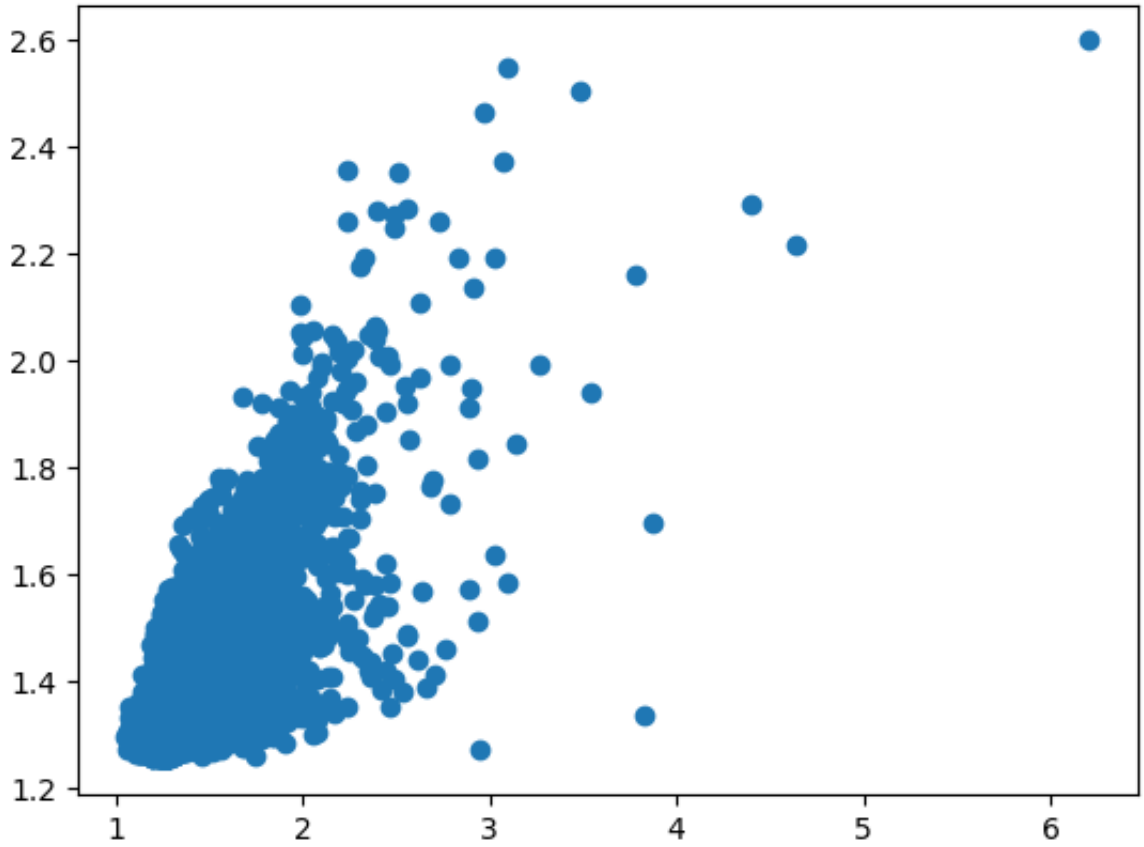
Lasso

In [52]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[52]:
```
Lasso
Lasso(alpha=5)
```

In [53]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x7fafd9293b50>
```



In [54]:
```
las=la.score(x_test,y_test)
```

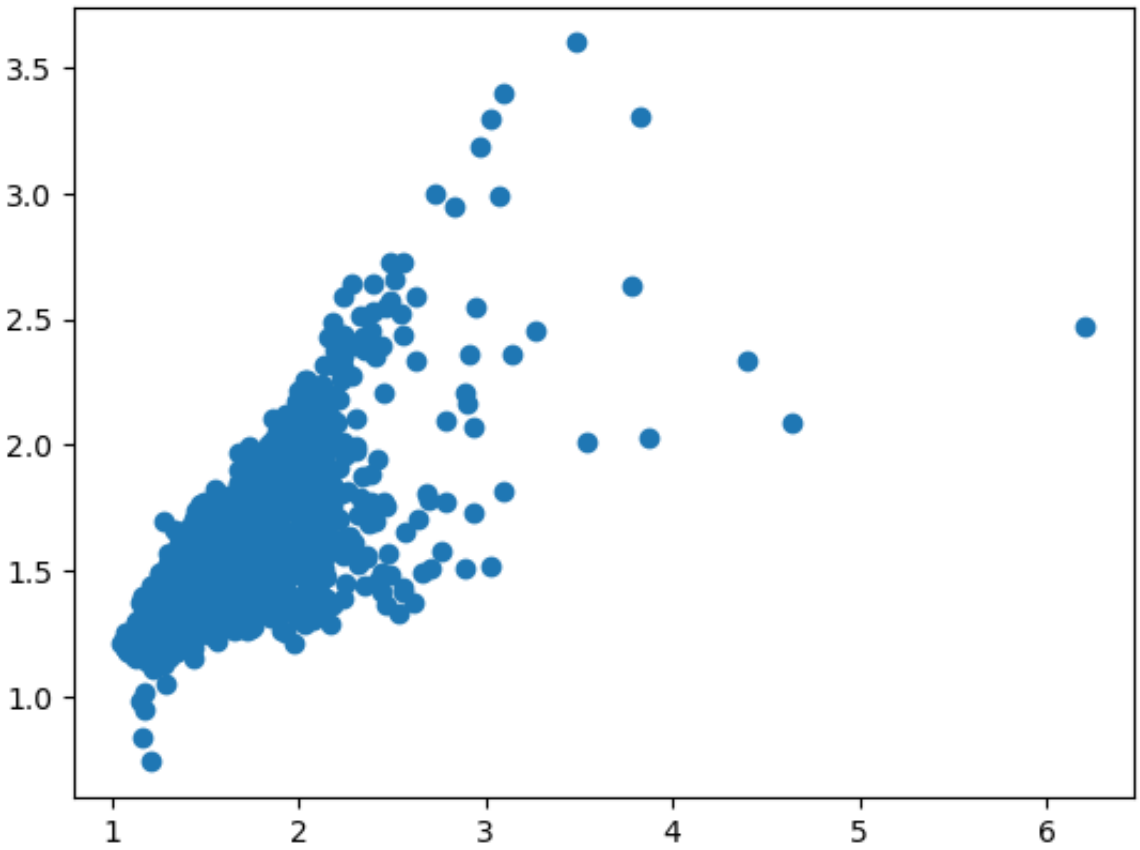
Ridge

In [55]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[55]:
```
Ridge
Ridge(alpha=1)
```

In [56]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[56]: <matplotlib.collections.PathCollection at 0x7fafd92e09d0>



In [57]:
```
rrs=rr.score(x_test,y_test)
```

ElasticNet

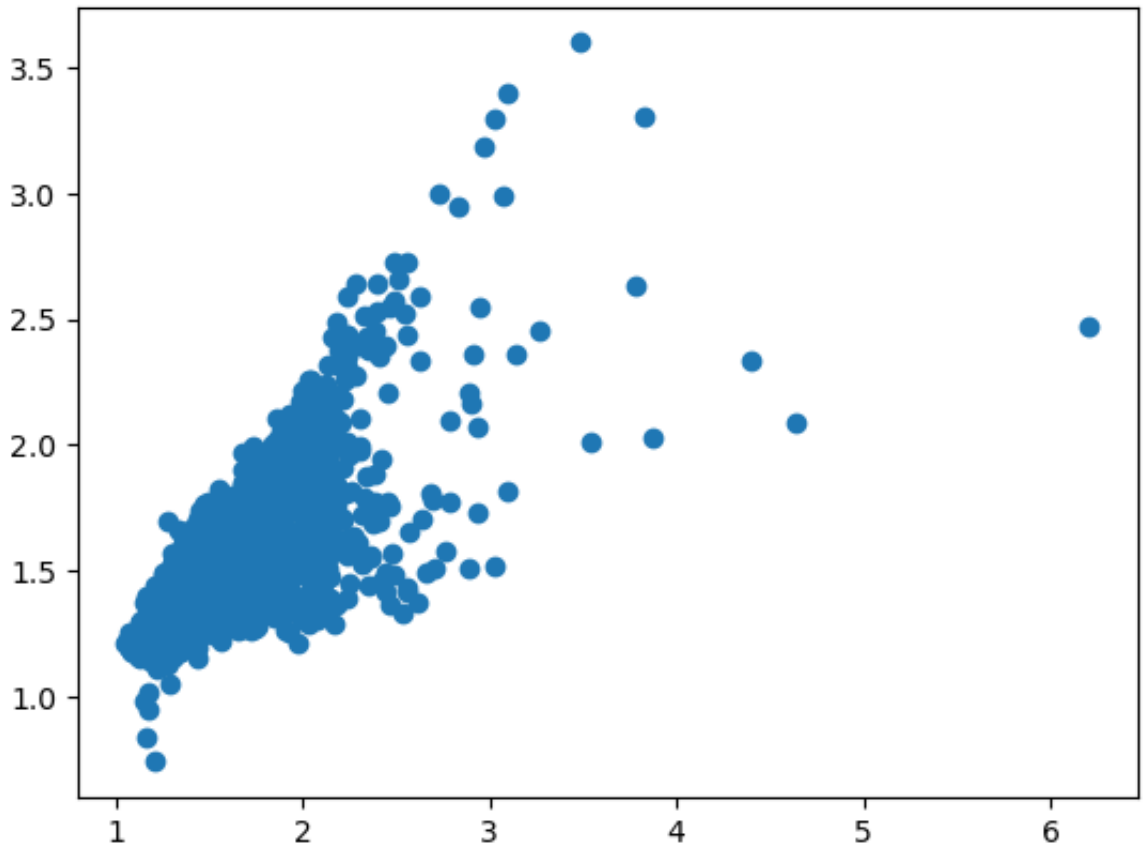
In [58]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[58]:
```
▾ ElasticNet

ElasticNet()
```

In [59]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[59]:
```
<matplotlib.collections.PathCollection at 0x7fafd95d52d0>
```



In [60]:
```
ens=en.score(x_test,y_test)
```

In [61]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.6823070107091946
```

Out[61]:
```
0.7220782029510737
```

Logistic

In [62]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[62]:
```
Low     21925
High    10456
Name: TCH, dtype: int64
```

In [63]:
```
x=df3.drop(["TCH"],axis=1)
y=df3["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

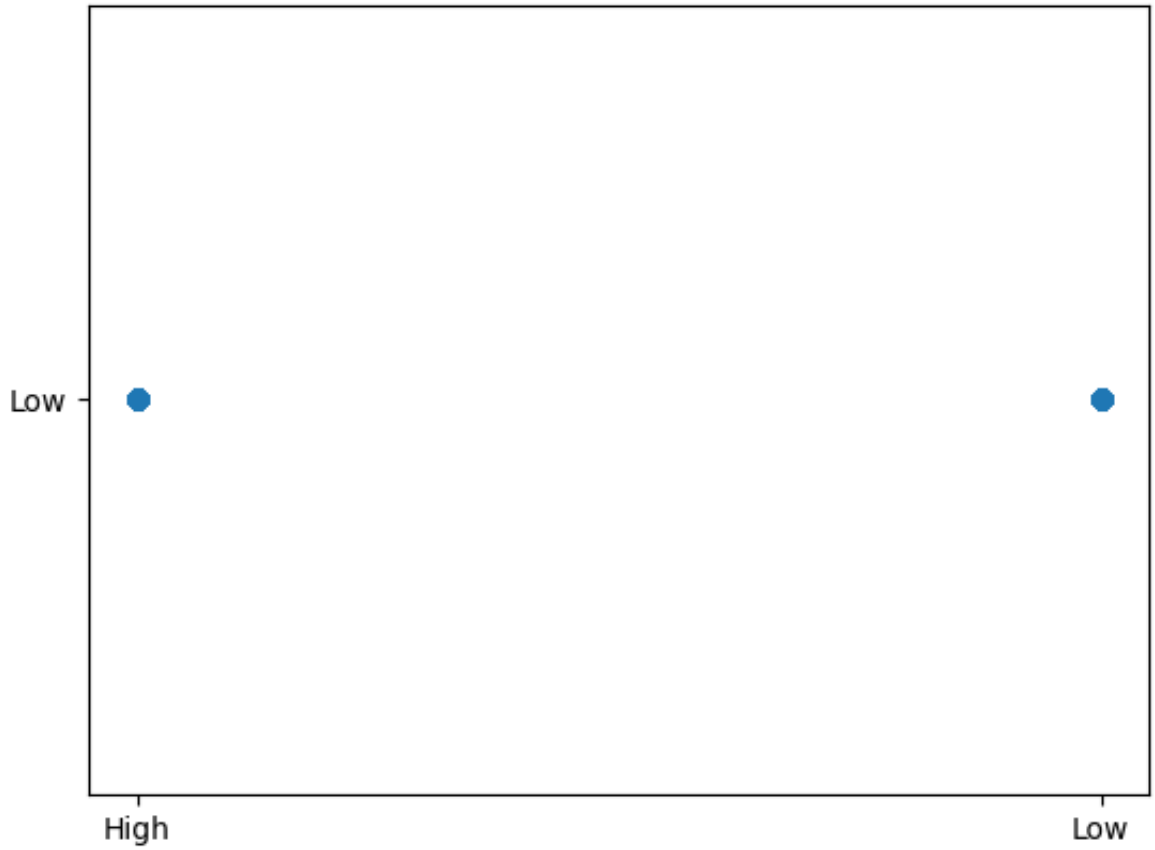
In [64]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[64]:
```
▾ LogisticRegression

LogisticRegression()
```

In [65]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[65]: <matplotlib.collections.PathCollection at 0x7fafd9698880>



In [66]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [67]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [68]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [69]:
```
x=df3.drop(["TCH"],axis=1)
y=df3["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [70]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[70]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [71]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [72]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,sc
grid_search.fit(x_train,y_train)
```

Out[72]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [73]:
```
rfcs=grid_search.best_score_
```

In [74]:
```
rfc_best=grid_search.best_estimator_
```

In [75]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_nam
```

Out[75]:
```
[Text(0.5026041666666666, 0.9285714285714286, 'NMHC <= 0.205\ngini
= 0.437\nsamples = 14311\nvalue = [15351, 7315]\nclass = Yes'),
 Text(0.2760416666666667, 0.7857142857142857, 'O_3 <= 16.315\ngini
= 0.27\nsamples = 10945\nvalue = [14572, 2794]\nclass = Yes'),
 Text(0.1328125, 0.6428571428571429, 'BEN <= 1.835\ngini = 0.481\n
samples = 2740\nvalue = [2606, 1764]\nclass = Yes'),
 Text(0.057291666666666664, 0.5, 'O_3 <= 4.615\ngini = 0.407\nsamp
les = 989\nvalue = [1140, 453]\nclass = Yes'),
 Text(0.03125, 0.35714285714285715, 'EBE <= 1.455\ngini = 0.489\ns
amples = 93\nvalue = [59, 79]\nclass = No'),
 Text(0.020833333333333332, 0.21428571428571427, 'NOx <= 70.655\ng
ini = 0.492\nsamples = 64\nvalue = [54, 42]\nclass = Yes'),
 Text(0.010416666666666666, 0.07142857142857142, 'gini = 0.408\nsa
mples = 32\nvalue = [30, 12]\nclass = Yes'),
 Text(0.03125, 0.07142857142857142, 'gini = 0.494\nsamples = 32\nv
alue = [24, 30]\nclass = No'),
 Text(0.041666666666666664, 0.21428571428571427, 'gini = 0.21\nsam
ples = 29\nvalue = [5, 37]\nclass = No'),
 Text(0.08333333333333333, 0.35714285714285715, 'TOL <= 4.795\ngin
```

In [76]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.6822690751065204
Lasso: 0.5096590774151815
Ridge: 0.6823070107091946
ElasticNet: 0.5625064551260017
Logistic: 0.6779207411219763
Random Forest: 0.8961881231800936
```

Best model is Random Forest