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	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001
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
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000
...	...	...	...	...	...	...	...	...	...	...
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
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN
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

215688 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215688 entries, 0 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     215688 non-null  object 
 1   BEN      60082 non-null   float64
 2   CO       190801 non-null  float64
 3   EBE      60081 non-null   float64
 4   MXY      24846 non-null   float64
 5   NMHC     74748 non-null   float64
 6   NO_2     214562 non-null  float64
 7   NOx      214565 non-null  float64
 8   OXY      24854 non-null   float64
 9   O_3      204482 non-null  float64
 10  PM10     196331 non-null  float64
 11  PM25     55822 non-null   float64
 12  PXY      24854 non-null   float64
 13  SO_2     212671 non-null  float64
 14  TCH      75213 non-null   float64
 15  TOL      59920 non-null   float64
 16  station  215688 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 28.0+ MB
```

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

Out[4]:

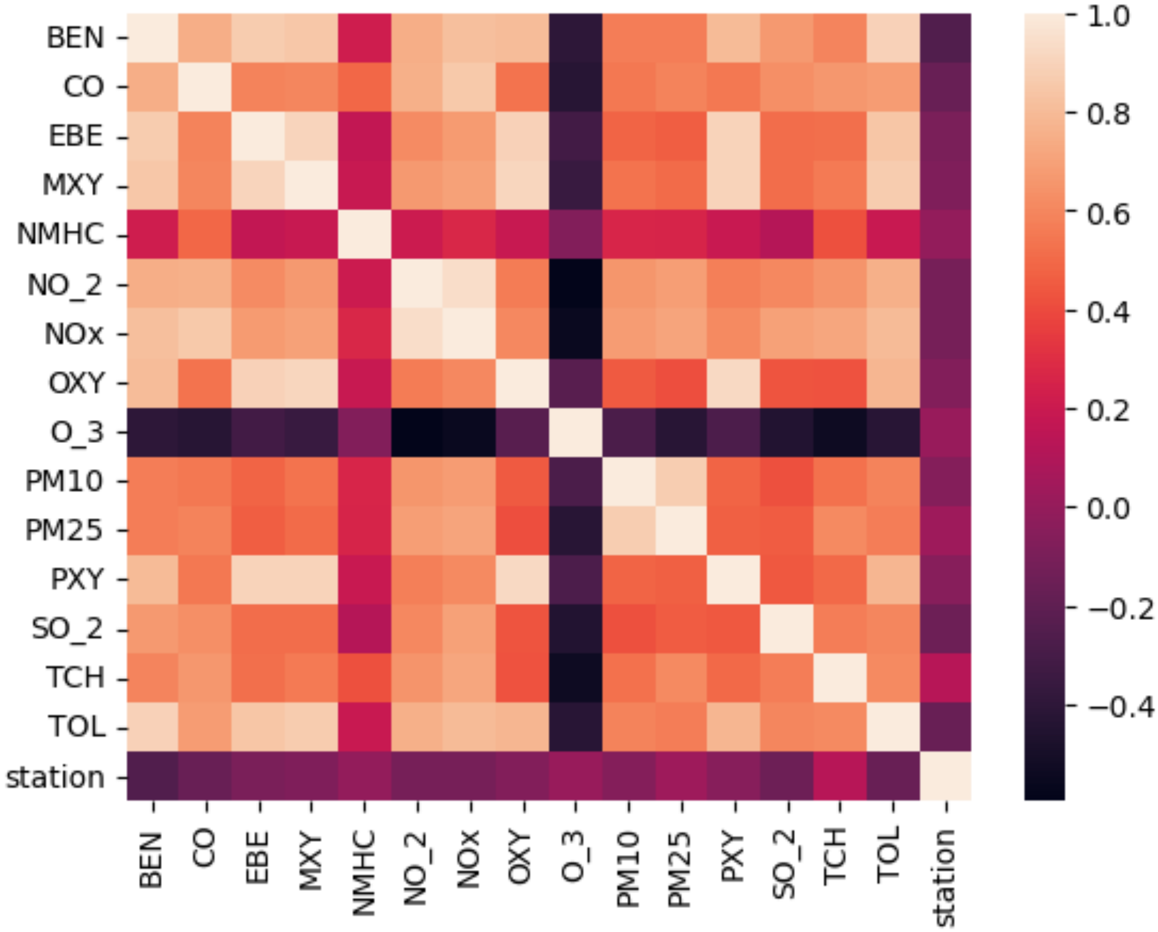
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	2
20	2009-10-01 01:00:00	0.38	0.32	0.32	0.89	0.01	17.969999	19.240000	1.00	65.870003	1
24	2009-10-01 01:00:00	0.55	0.24	0.65	1.79	0.18	36.619999	43.919998	1.28	48.070000	1
28	2009-10-01 02:00:00	0.65	0.21	1.20	2.04	0.18	37.169998	48.869999	1.21	26.950001	3
45	2009-10-01 02:00:00	0.38	0.30	0.50	1.15	0.00	17.889999	19.299999	1.00	60.009998	1
...	...	...	...	...	...	...	...	...	...	...	
215659	2009-05-31 23:00:00	0.54	0.27	1.00	0.69	0.09	28.280001	29.490000	0.86	78.750000	1
215663	2009-05-31 23:00:00	0.74	0.35	1.13	1.65	0.15	56.410000	69.870003	1.26	56.799999	1
215667	2009-06-01 00:00:00	0.78	0.29	0.99	1.96	0.04	64.870003	82.629997	1.13	58.000000	1
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	1
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	1

24717 rows × 17 columns

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

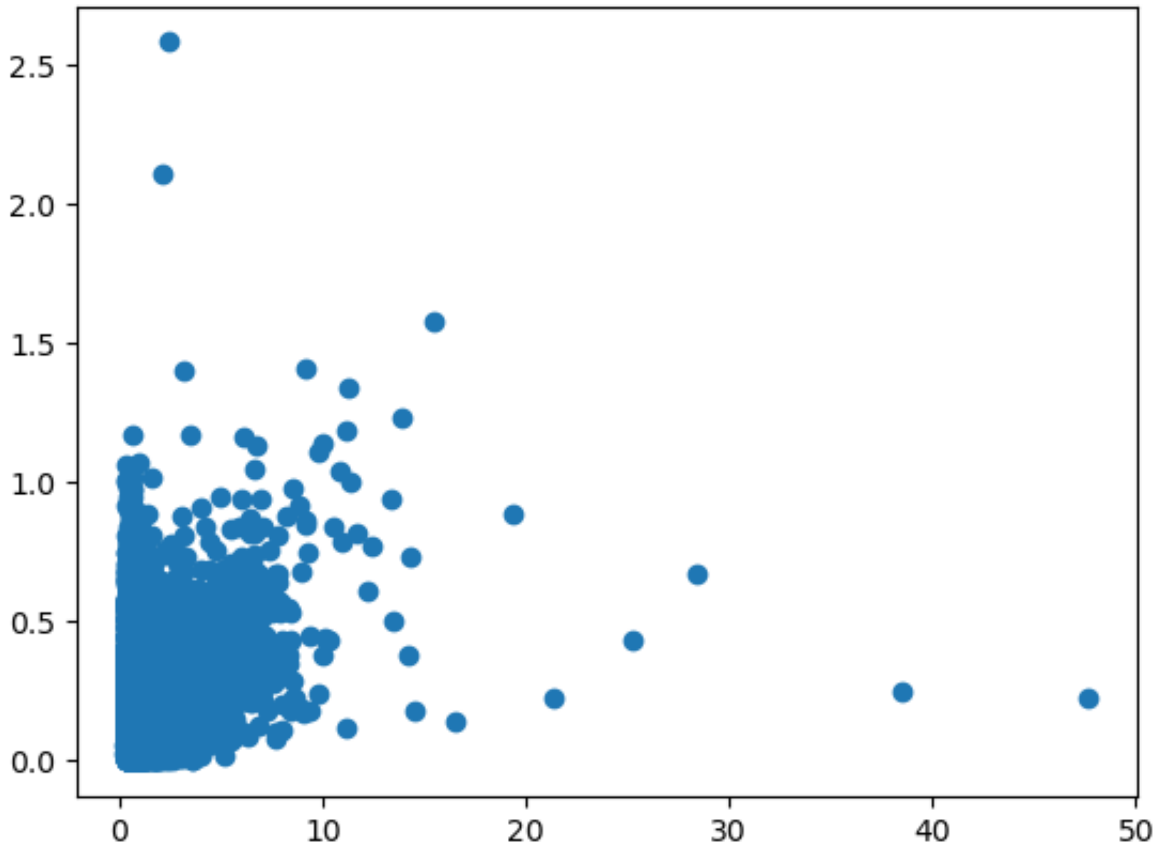
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <Axes: >



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

Out[7]: [<matplotlib.lines.Line2D at 0x7fdd01918730>]



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)
```

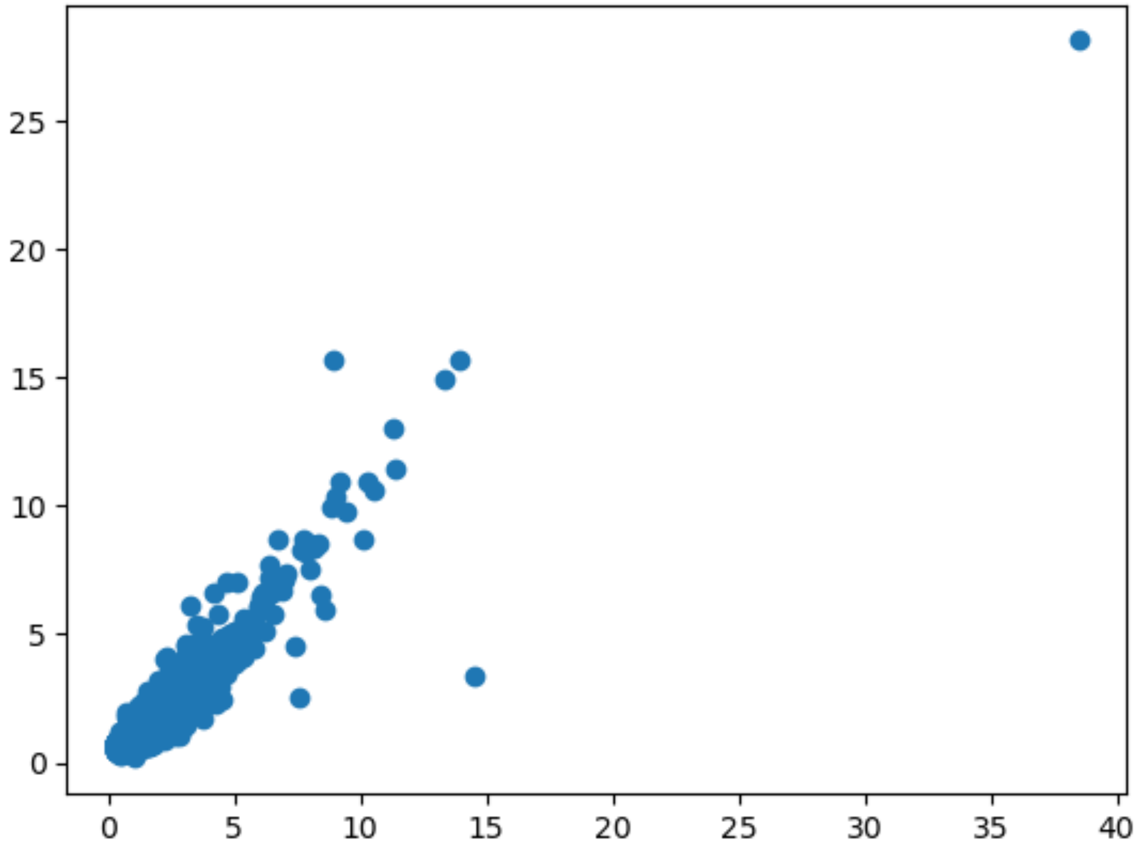
Linear

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 0x7fdd11fa0640>



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

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

Out[13]:
```
1.39    1091
1.36    1056
1.38    1046
1.40    1018
1.37    1017
        ...
4.03       1
3.84       1
3.29       1
2.79       1
3.94       1
Name: TCH, Length: 169, 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    12963
2.0    11754
Name: TCH, dtype: int64
```

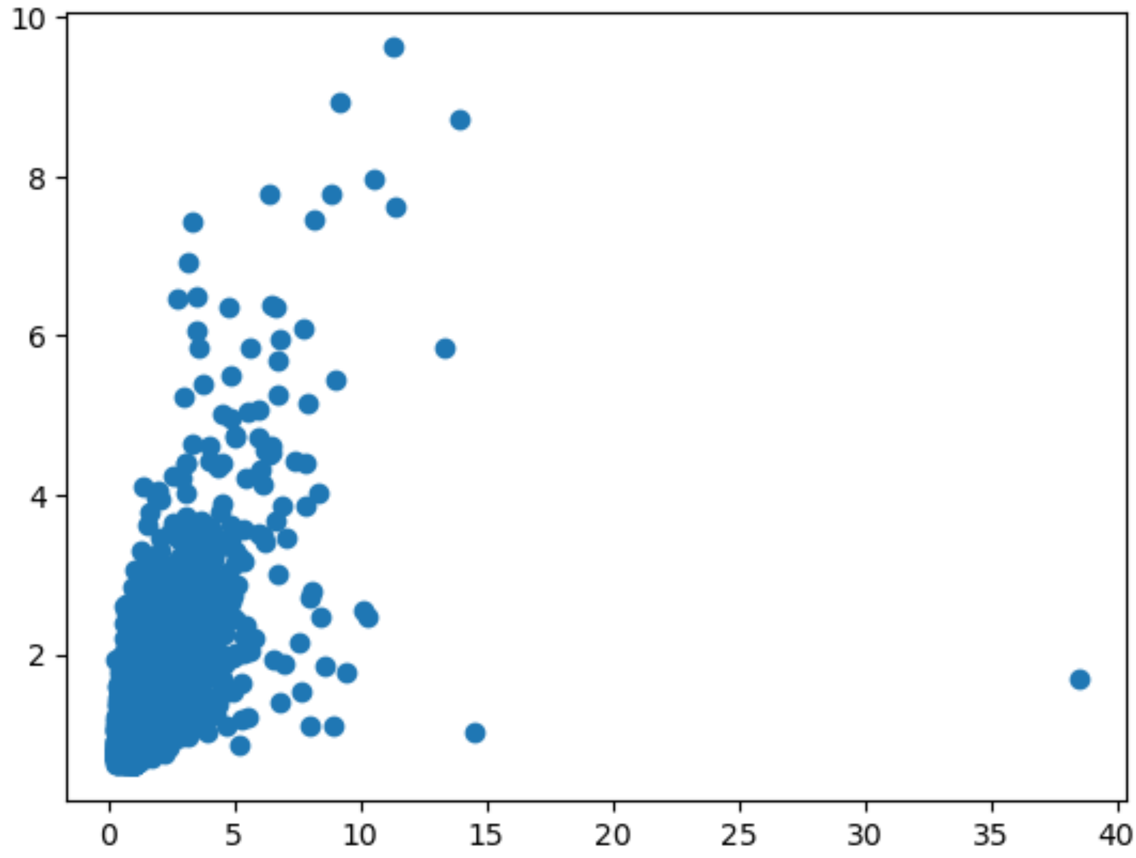
Lasso

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 0x7fdcf92b15d0>



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

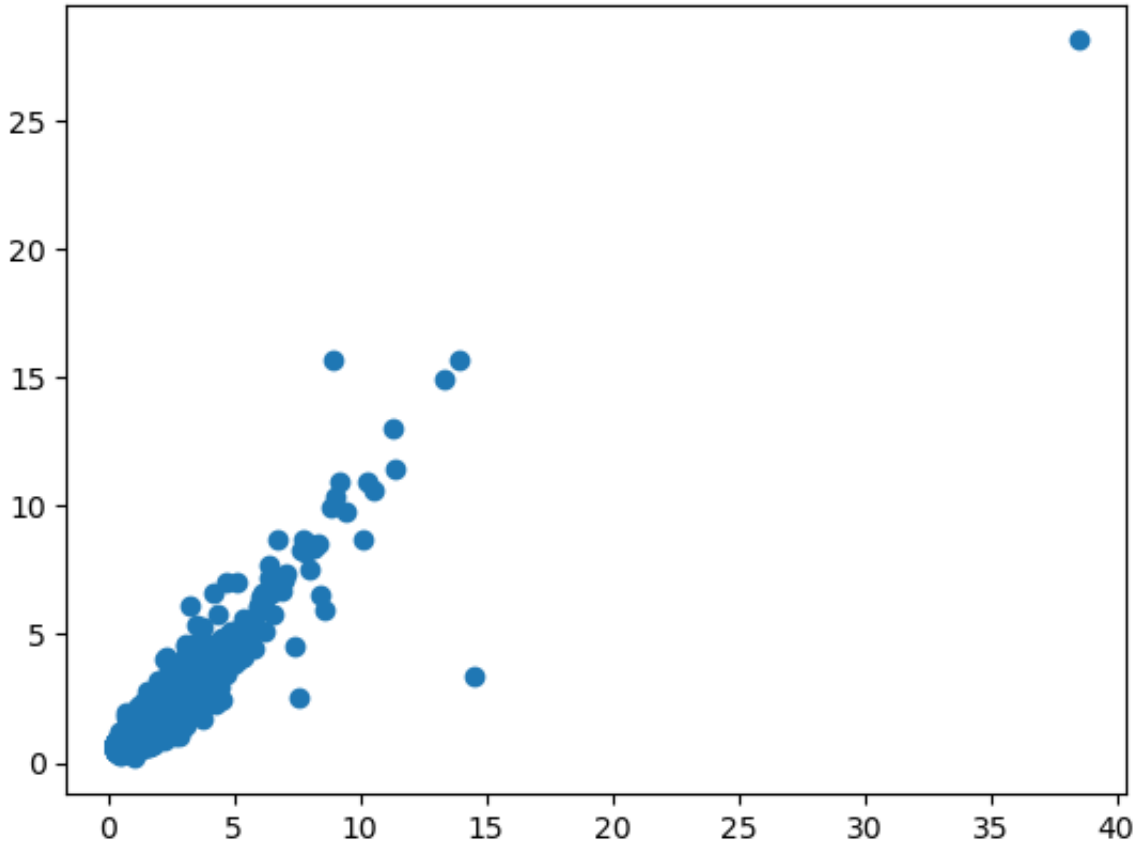
Ridge

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 0x7fdcfba07c10>



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

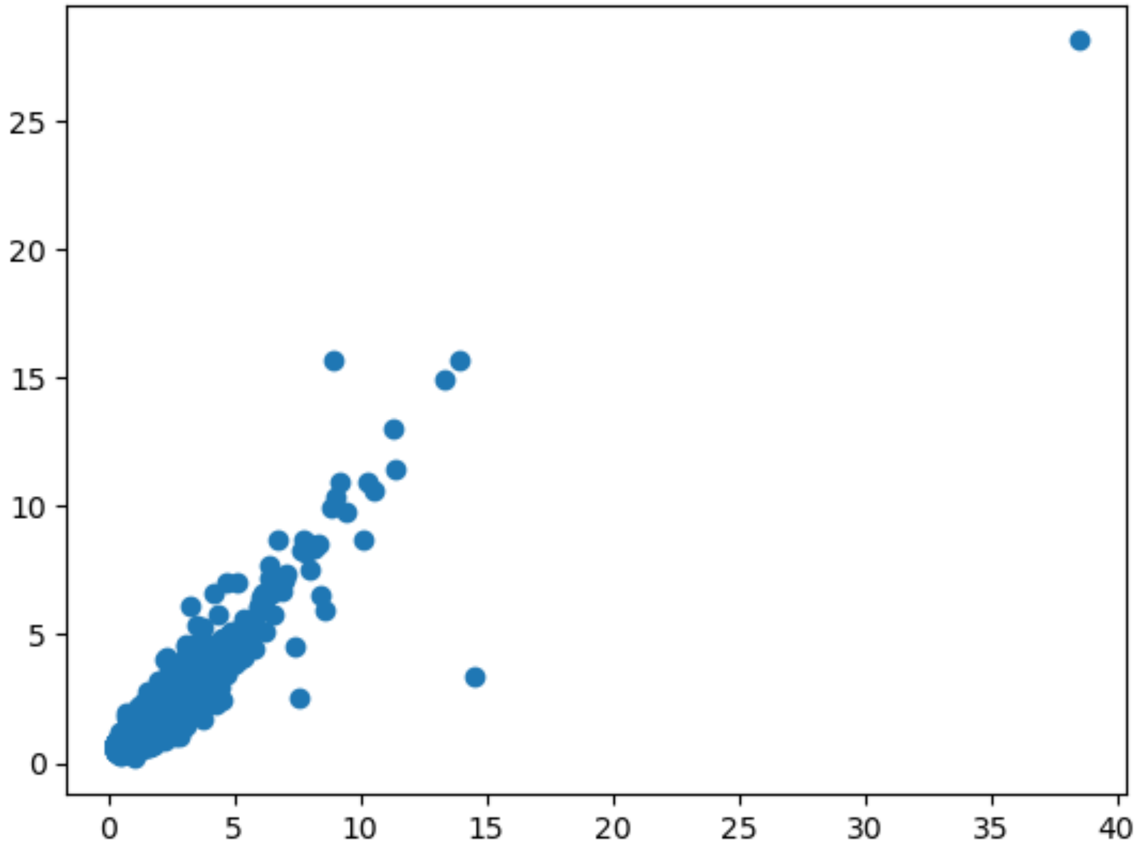
ElasticNet

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 0x7fdcf01d5ea0>



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

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

```
0.8795101137449133
```

Out[24]: 0.8929278460152889

Logistic

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

Out[25]:
```
Low     12963
High    11754
Name: TCH, dtype: int64
```

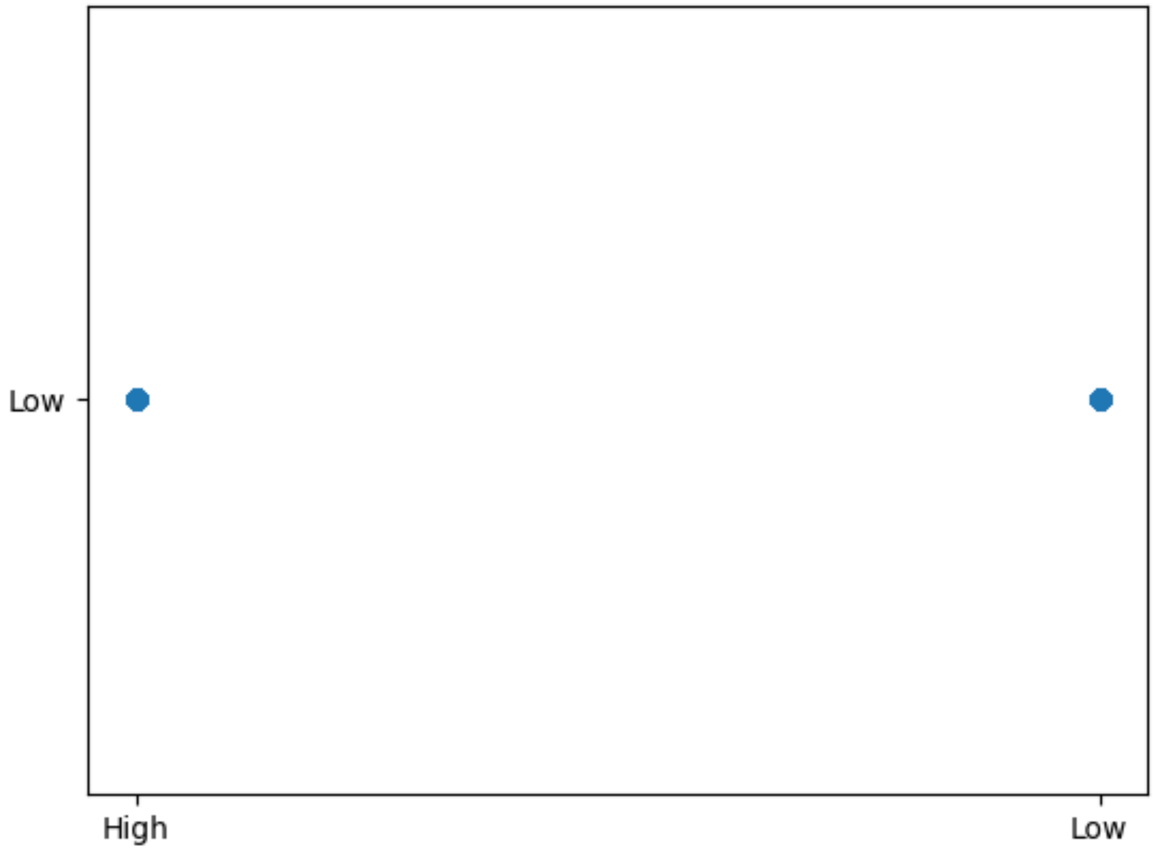
In [26]:
```
x=df1.drop(["TCH"],axis=1)
y=df1["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x7fdd01b01750>



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(0.4756944444444444, 0.9285714285714286, 'BEN <= 0.755\ngini
= 0.499\nsamples = 10885\nvalue = [8964, 8337]\nclass = Yes'),
 Text(0.22916666666666666, 0.7857142857142857, 'NO_2 <= 39.815\ngi
ni = 0.404\nsamples = 6200\nvalue = [7047, 2747]\nclass = Yes'),
 Text(0.08796296296296297, 0.6428571428571429, 'NOx <= 25.395\ngin
i = 0.229\nsamples = 4027\nvalue = [5512, 838]\nclass = Yes'),
 Text(0.027777777777777776, 0.5, 'O_3 <= 21.875\ngini = 0.077\nsam
ples = 2045\nvalue = [3077, 128]\nclass = Yes'),
 Text(0.018518518518518517, 0.35714285714285715, 'gini = 0.375\nsa
mples = 6\nvalue = [2, 6]\nclass = No'),
 Text(0.037037037037037035, 0.35714285714285715, 'CO <= 0.425\ngin
i = 0.073\nsamples = 2039\nvalue = [3075, 122]\nclass = Yes'),
 Text(0.018518518518518517, 0.21428571428571427, 'MXY <= 1.175\ngi
ni = 0.024\nsamples = 1581\nvalue = [2432, 30]\nclass = Yes'),
 Text(0.009259259259259259, 0.07142857142857142, 'gini = 0.019\nsa
mples = 1488\nvalue = [2279, 22]\nclass = Yes'),
 Text(0.027777777777777776, 0.07142857142857142, 'gini = 0.094\nsa
mples = 93\nvalue = [153, 8]\nclass = Yes'),
 Text(0.05555555555555555, 0.21428571428571427, 'NO_2 <= 11.88\ngi
```

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

```
Linear: 0.8795129171288489
Lasso: 0.4570177776335941
Ridge: 0.8795101137449133
ElasticNet: 0.737810583117062
Logistic: 0.5242718446601942
Random Forest: 0.8623781005637359
```

Best Model is Random Forest

In [40]:
```
df2=pd.read_csv("/Users/bob/Downloads/FP1_air/csvs_per_year/csvs_pe
df2
```

Out[40]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998

209448 rows × 17 columns

In [41]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209448 entries, 0 to 209447
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209448 non-null  object 
 1   BEN      60268 non-null   float64
 2   CO       94982 non-null   float64
 3   EBE      60253 non-null   float64
 4   MXY      6750 non-null    float64
 5   NMHC     51727 non-null   float64
 6   NO_2     208219 non-null  float64
 7   NOx      208210 non-null  float64
 8   OXY      6750 non-null    float64
 9   O_3      126684 non-null  float64
 10  PM10     106186 non-null  float64
 11  PM25     55514 non-null   float64
 12  PXY      6740 non-null    float64
 13  SO_2     93184 non-null   float64
 14  TCH      51730 non-null   float64
 15  TOL      60171 non-null   float64
 16  station  209448 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 27.2+ MB
```

In [42]:
```
df3=df2.dropna()
df3
```

Out[42]:

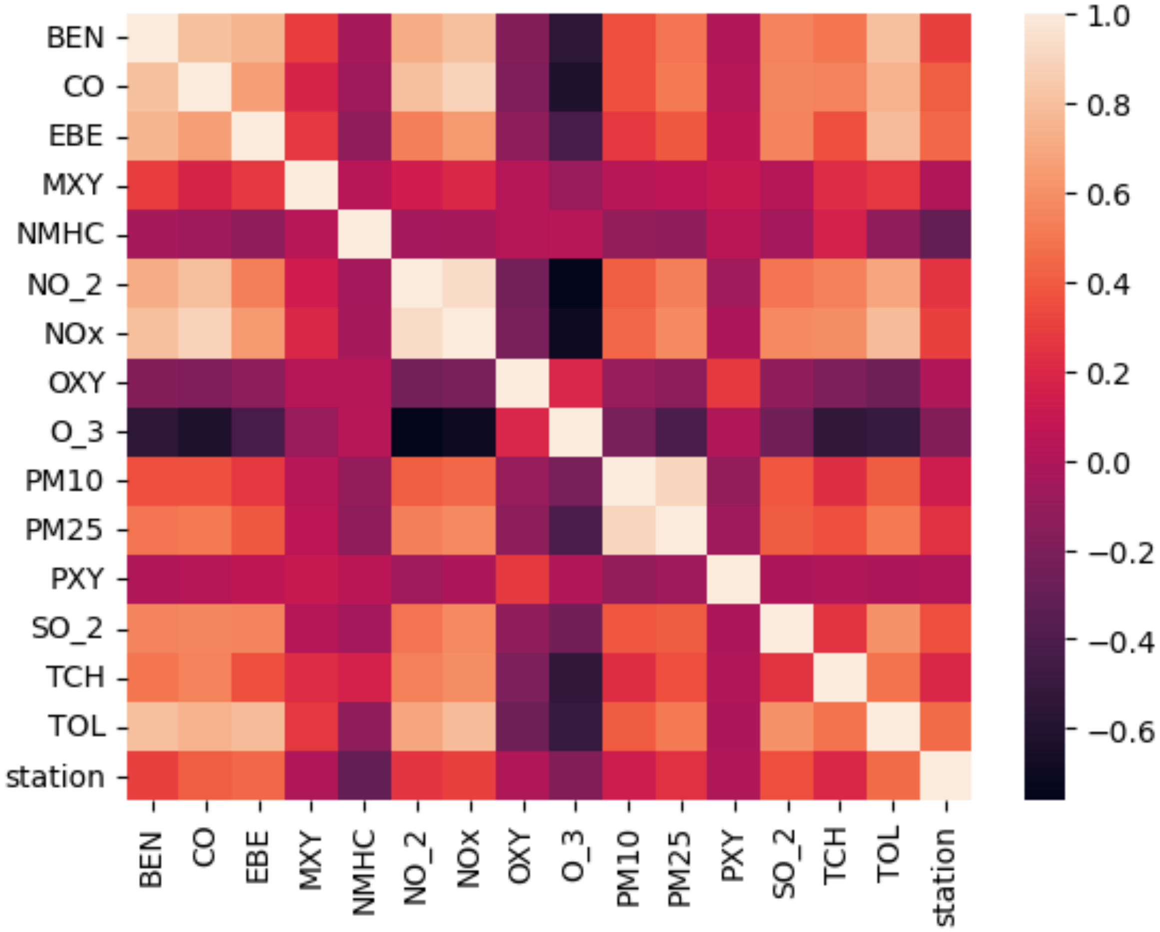
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
11	2010-03-01 01:00:00	0.78	0.18	0.84	0.73	0.28	10.420000	11.900000	1.0	90.309998
23	2010-03-01 01:00:00	0.70	0.23	1.00	0.73	0.18	17.820000	22.290001	1.0	70.550003
35	2010-03-01 02:00:00	0.58	0.17	0.84	0.73	0.28	3.500000	4.950000	1.0	68.849998
47	2010-03-01 02:00:00	0.33	0.21	0.84	0.73	0.17	10.810000	14.900000	1.0	74.750000
59	2010-03-01 03:00:00	0.38	0.16	0.64	1.00	0.26	2.750000	4.200000	1.0	93.629997
...	...	...	...	...	...	...	...	...	...	...
191879	2010-05-31 22:00:00	0.60	0.26	0.82	0.13	0.16	33.360001	43.779999	1.0	38.459999
191891	2010-05-31 23:00:00	0.41	0.16	0.71	0.19	0.10	24.299999	26.059999	1.0	50.290001
191903	2010-05-31 23:00:00	0.57	0.28	0.64	0.19	0.18	35.540001	44.590000	1.0	34.020000
191915	2010-06-01 00:00:00	0.34	0.16	0.69	0.22	0.10	23.559999	25.209999	1.0	45.930000
191927	2010-06-01 00:00:00	0.43	0.25	0.79	0.22	0.18	34.910000	42.369999	1.0	29.540001

6666 rows × 17 columns

In [43]:
```
df3=df3.drop(["date"],axis=1)
```

In [44]:
```
sns.heatmap(df3.corr())
```

Out[44]: <Axes: >



In [45]:
```
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)
```

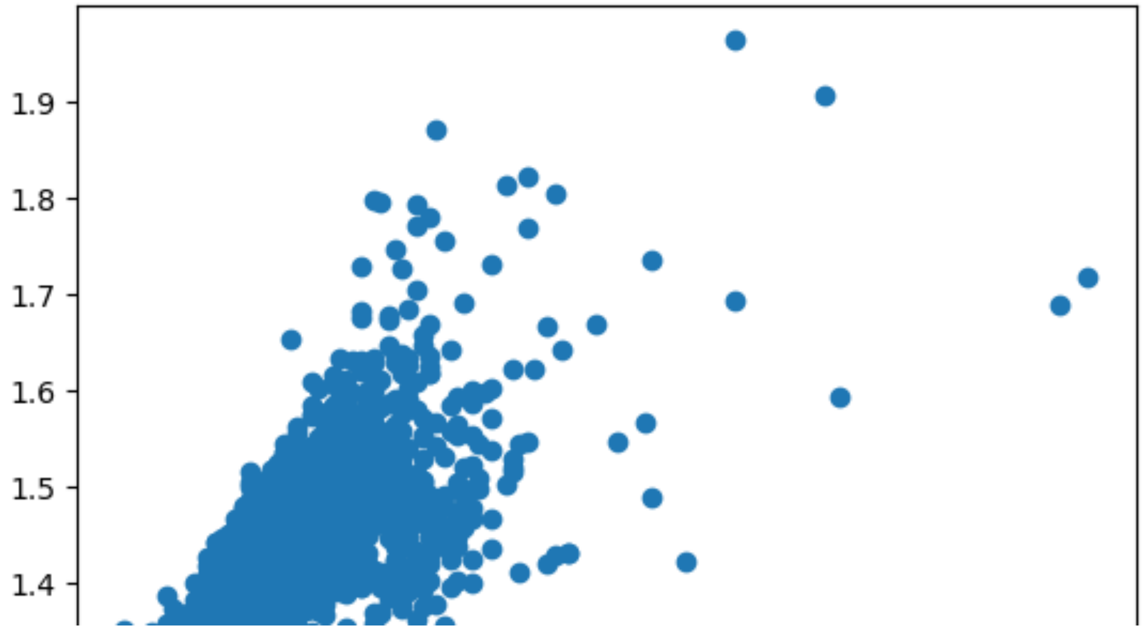
Linear

In [46]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[46]:
```
▾ LinearRegression
LinearRegression()
```

In [47]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[47]: <matplotlib.collections.PathCollection at 0x7fdcf1c1c670>



In [48]:
```
lis=li.score(x_test,y_test)
```

In [49]:
```
df3["TCH"].value_counts()
```

Out[49]:
```
1.36    364
1.38    351
1.39    324
1.35    323
1.37    321
       ...
2.11      1
1.98      1
1.18      1
2.18      1
2.04      1
Name: TCH, Length: 100, dtype: int64
```

In [50]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[50]:
```
1.0    3340
2.0    3326
Name: TCH, dtype: int64
```

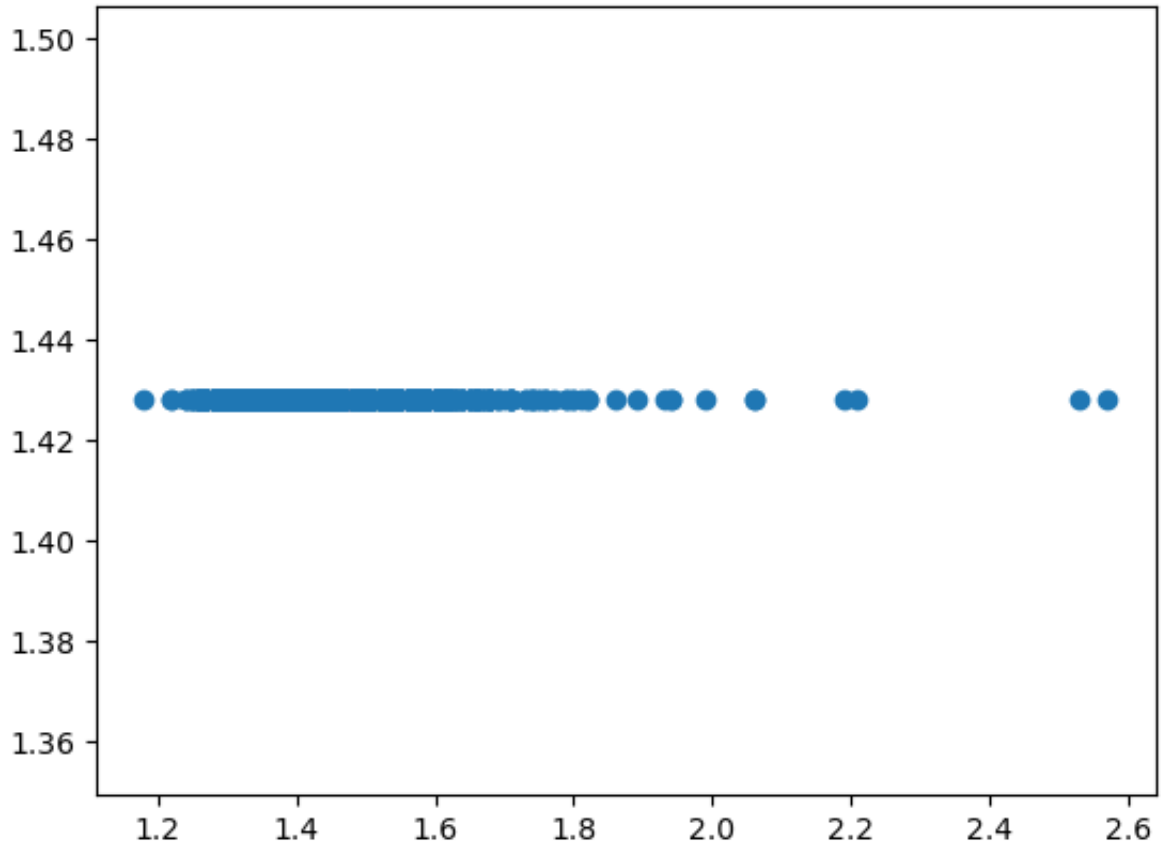
Lasso

In [51]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[51]:
```
▾ Lasso
Lasso(alpha=5)
```

In [52]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[52]: <matplotlib.collections.PathCollection at 0x7fdcf2aa4160>



In [53]:
```
las=la.score(x_test,y_test)
```

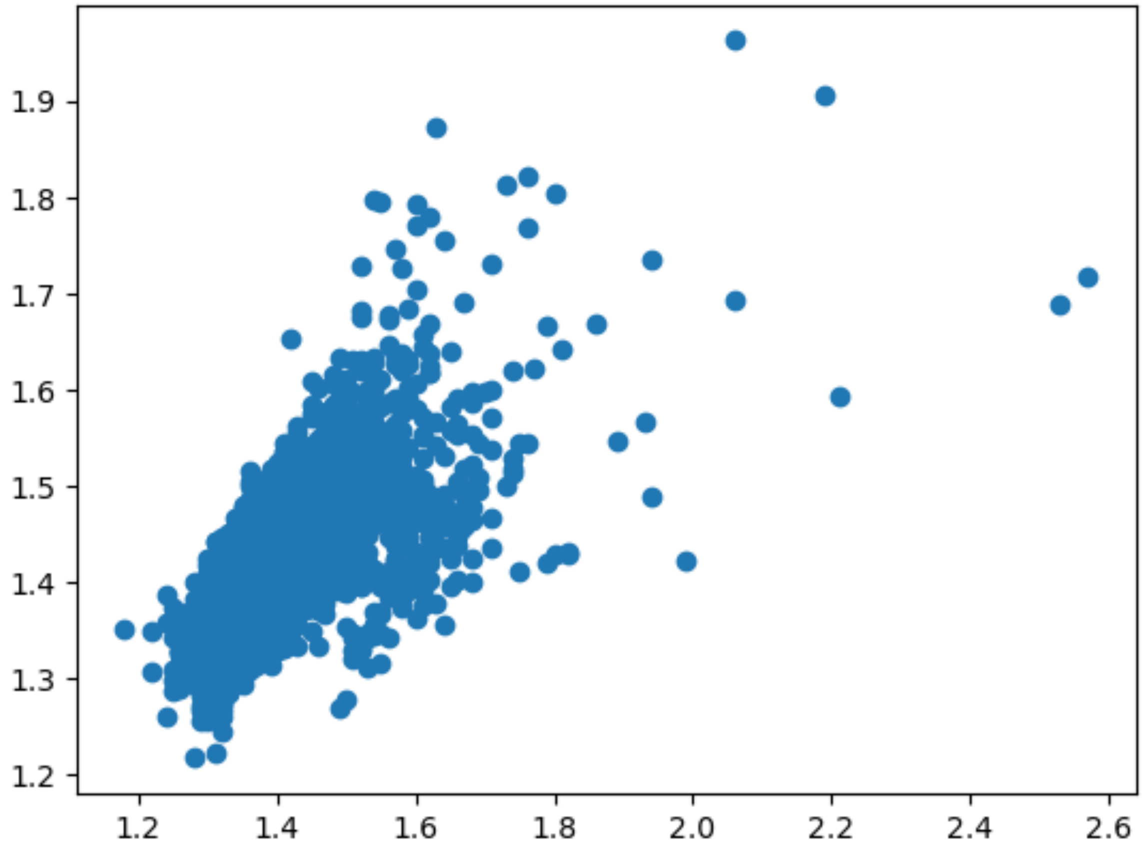
Ridge

In [54]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[54]:
```
▾ Ridge
Ridge(alpha=1)
```

In [55]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[55]: <matplotlib.collections.PathCollection at 0x7fdd026d5210>



In [56]:
```
rrs=rr.score(x_test,y_test)
```

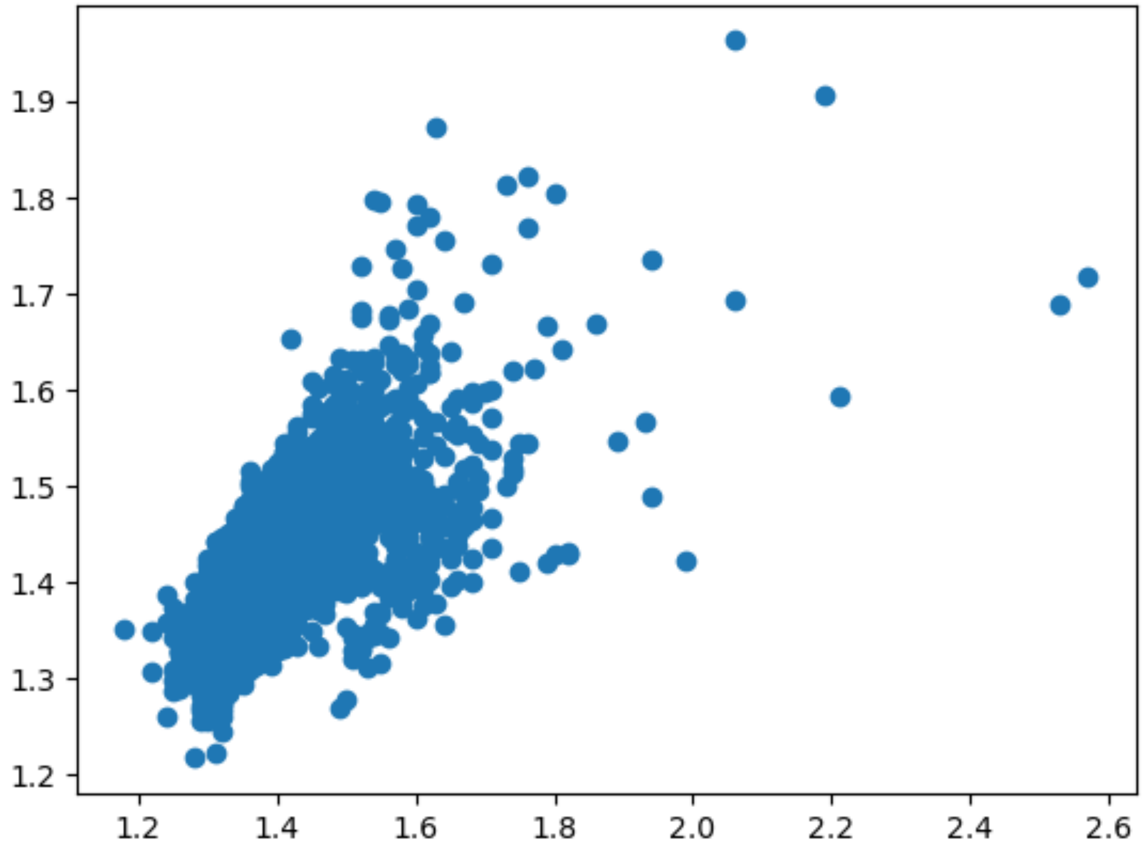
ElasticNet

In [57]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[57]:
```
▾ ElasticNet
ElasticNet()
```

In [58]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[58]: <matplotlib.collections.PathCollection at 0x7fdd0268bc40>



In [59]:
```
ens=en.score(x_test,y_test)
```

In [60]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.4495448643204282
```

Out[60]: 0.45195948083792425

Logistic

In [61]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[61]:
```
Low     3340
High    3326
Name: TCH, dtype: int64
```

In [62]:
```
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 [63]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[63]:
```
▾ LogisticRegression
LogisticRegression()
```

In [64]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[64]: <matplotlib.collections.PathCollection at 0x7fdd0268bd90>

In [65]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [66]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [67]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [68]:
```
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 [69]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[69]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [70]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [71]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,sc
grid_search.fit(x_train,y_train)
```

Out[71]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

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

In [73]:
```
rfc_best=grid_search.best_estimator_
```

In [74]:
```
from sklearn.tree import plot_tree

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

Out[74]:
```
[Text(0.5733418367346939, 0.9285714285714286, 'NOx <= 59.28\ngini
= 0.5\nsamples = 2944\nvalue = [2307, 2359]\nclass = No'),
 Text(0.30739795918367346, 0.7857142857142857, 'O_3 <= 80.6\ngini
= 0.47\nsamples = 2149\nvalue = [2110, 1284]\nclass = Yes'),
 Text(0.16326530612244897, 0.6428571428571429, 'NMHC <= 0.195\ngin
i = 0.497\nsamples = 1453\nvalue = [1260, 1066]\nclass = Yes'),
 Text(0.08163265306122448, 0.5, 'NOx <= 37.9\ngini = 0.429\nsample
s = 534\nvalue = [578, 262]\nclass = Yes'),
 Text(0.04081632653061224, 0.35714285714285715, 'CO <= 0.255\ngini
= 0.337\nsamples = 303\nvalue = [355, 97]\nclass = Yes'),
 Text(0.02040816326530612, 0.21428571428571427, 'CO <= 0.205\ngini
= 0.261\nsamples = 218\nvalue = [280, 51]\nclass = Yes'),
 Text(0.01020408163265306, 0.07142857142857142, 'gini = 0.098\nsam
ples = 84\nvalue = [110, 6]\nclass = Yes'),
 Text(0.030612244897959183, 0.07142857142857142, 'gini = 0.331\nsa
mples = 134\nvalue = [170, 45]\nclass = Yes'),
 Text(0.061224489795918366, 0.21428571428571427, 'BEN <= 0.495\ngi
ni = 0.471\nsamples = 85\nvalue = [75, 46]\nclass = Yes'),
 Text(0.05102040816326531, 0.07142857142857142, 'gini = 0.473\nsam
```

In [75]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.4496582984833126
Lasso: -0.0013142147893867584
Ridge: 0.4495448643204282
ElasticNet: 0.35229329191658254
Logistic: 0.4985
Random Forest: 0.7730390055722246
```

Best model is Random Forest