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	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	14.720000
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	30.469999
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	21.379999
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	33.410000
...	...	...	...	...	...	...	...	...	...	...
236995	2006-01-01 00:00:00	1.08	0.36	1.01	NaN	0.11	21.990000	23.610001	NaN	43.349998
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	43.840000
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	NaN
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998

237000 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      70370 non-null   float64
 2   CO       217656 non-null  float64
 3   EBE      68955 non-null   float64
 4   MXY      32549 non-null   float64
 5   NMHC     92854 non-null   float64
 6   NO_2     235022 non-null  float64
 7   NOx      235049 non-null  float64
 8   OXY      32555 non-null   float64
 9   O_3      223162 non-null  float64
 10  PM10     232142 non-null  float64
 11  PM25     69407 non-null   float64
 12  PXY      32549 non-null   float64
 13  SO_2     235277 non-null  float64
 14  TCH      93076 non-null   float64
 15  TOL      70255 non-null   float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ MB
```

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

Out[4]:

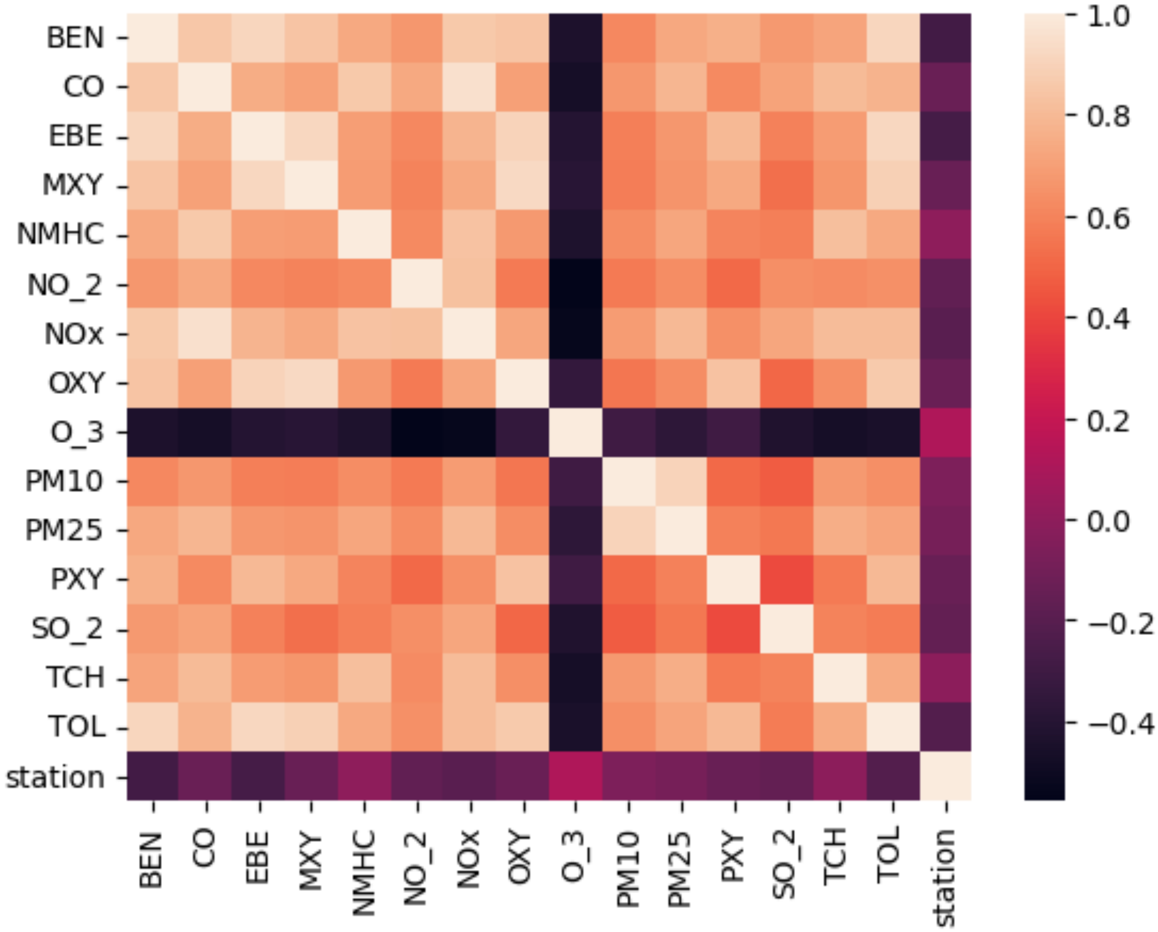
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
5	2005-11-01 01:00:00	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000
22	2005-11-01 01:00:00	0.30	0.22	0.25	0.59	0.11	18.540001	19.020000	0.67	46.799999
25	2005-11-01 01:00:00	0.67	0.49	0.94	3.44	0.17	48.740002	74.349998	1.57	23.430000
31	2005-11-01 02:00:00	3.10	0.84	3.21	6.82	0.22	89.919998	224.199997	3.72	12.390000
48	2005-11-01 02:00:00	0.39	0.20	0.29	0.68	0.11	16.639999	17.080000	0.40	47.689999
...	...	...	...	...	...	...	...	...	...	...
236970	2005-12-31 23:00:00	0.37	0.39	1.00	1.00	0.10	4.500000	5.550000	1.00	57.779999
236973	2005-12-31 23:00:00	0.92	0.45	1.26	3.42	0.14	37.250000	49.060001	2.57	31.889999
236979	2006-01-01 00:00:00	1.00	0.38	1.11	2.35	0.04	35.919998	59.480000	1.39	35.810001
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998

20070 rows × 17 columns

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

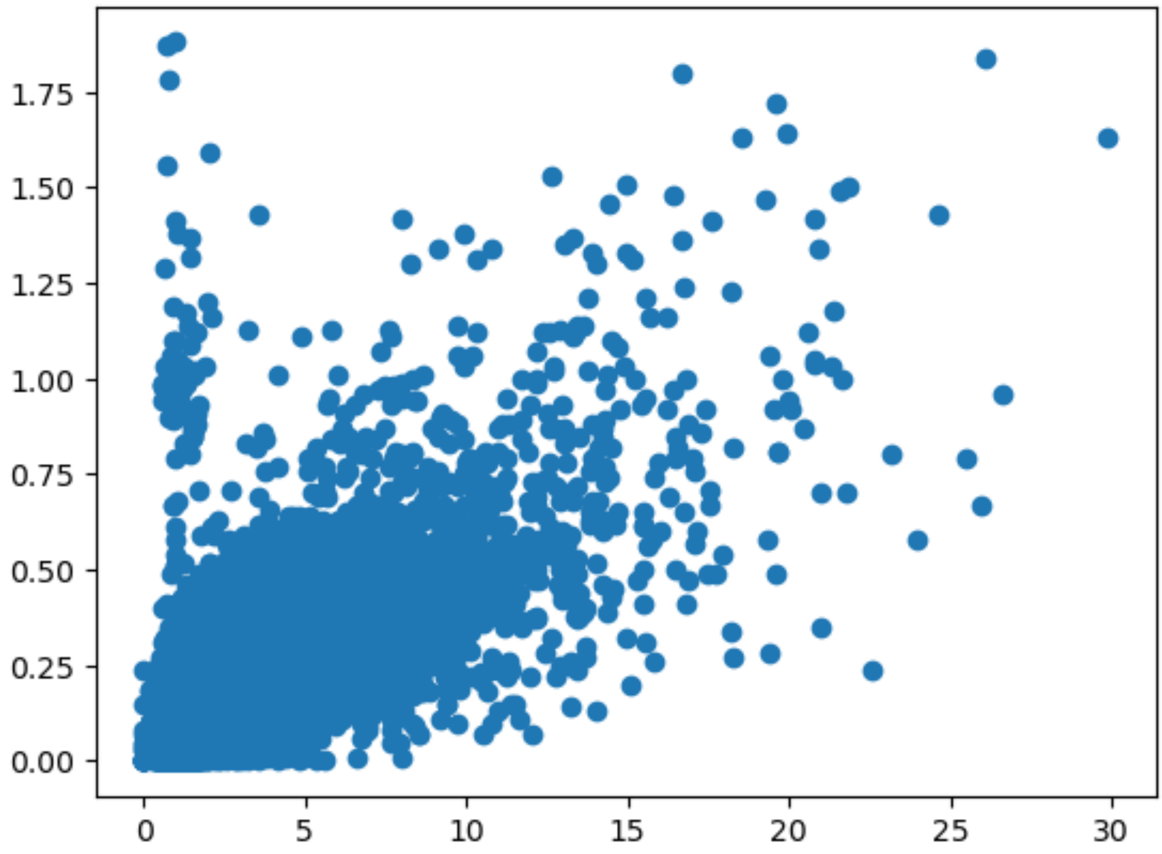
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <Axes: >



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

Out[7]:
```
[<matplotlib.lines.Line2D at 0x7fe2492811e0>]
```



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)
```

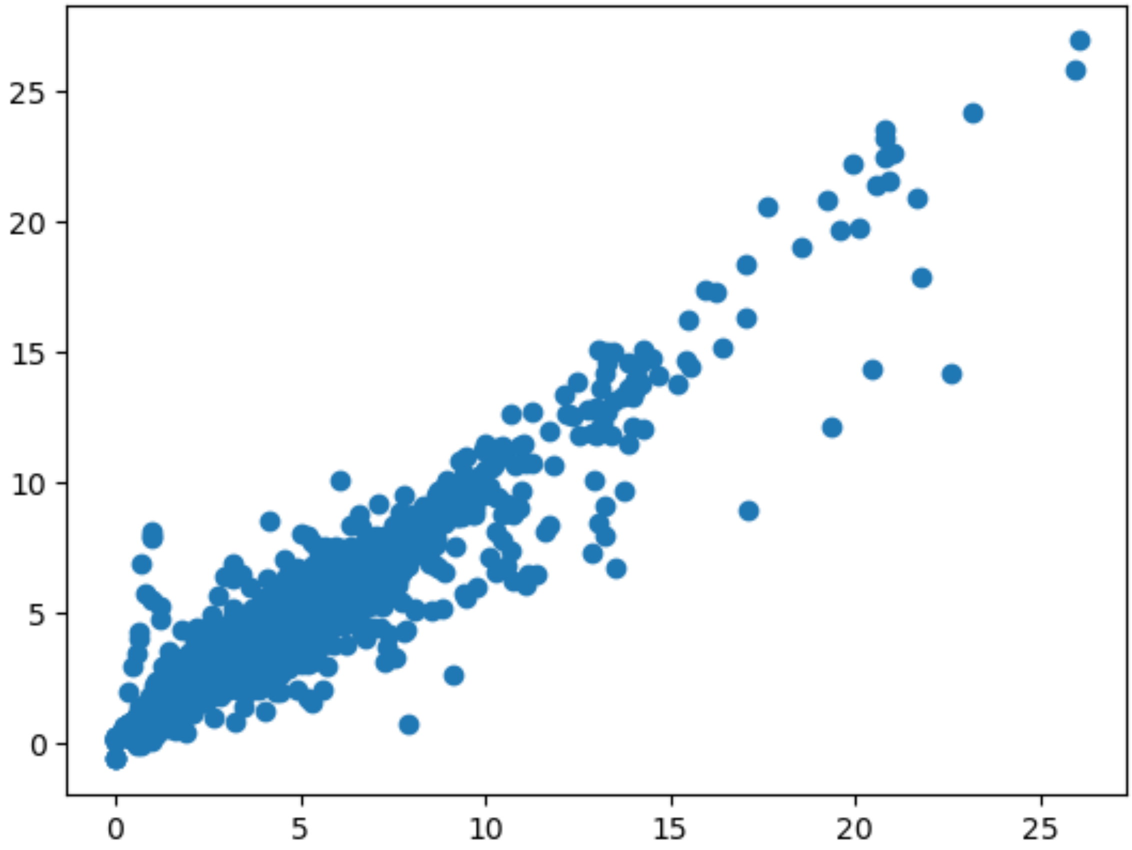
Linear

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 0x7fe22a96a0b0>



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

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

Out[13]:
```
1.31    845
1.33    820
1.28    812
1.30    806
1.34    794
       ...
2.99      1
3.37      1
3.38      1
3.22      1
2.48      1
Name: TCH, Length: 198, 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    12093
2.0     7977
Name: TCH, dtype: int64
```

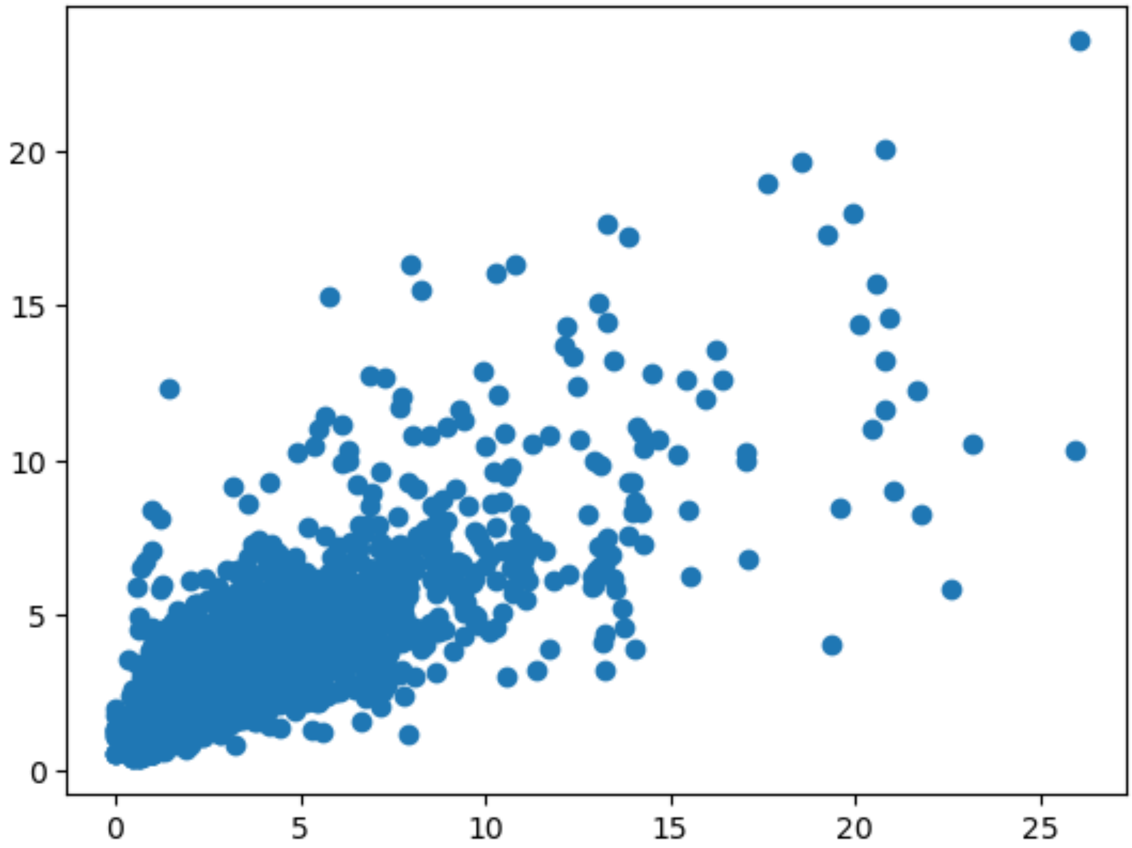
Lasso

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 0x7fe2491a3d90>



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

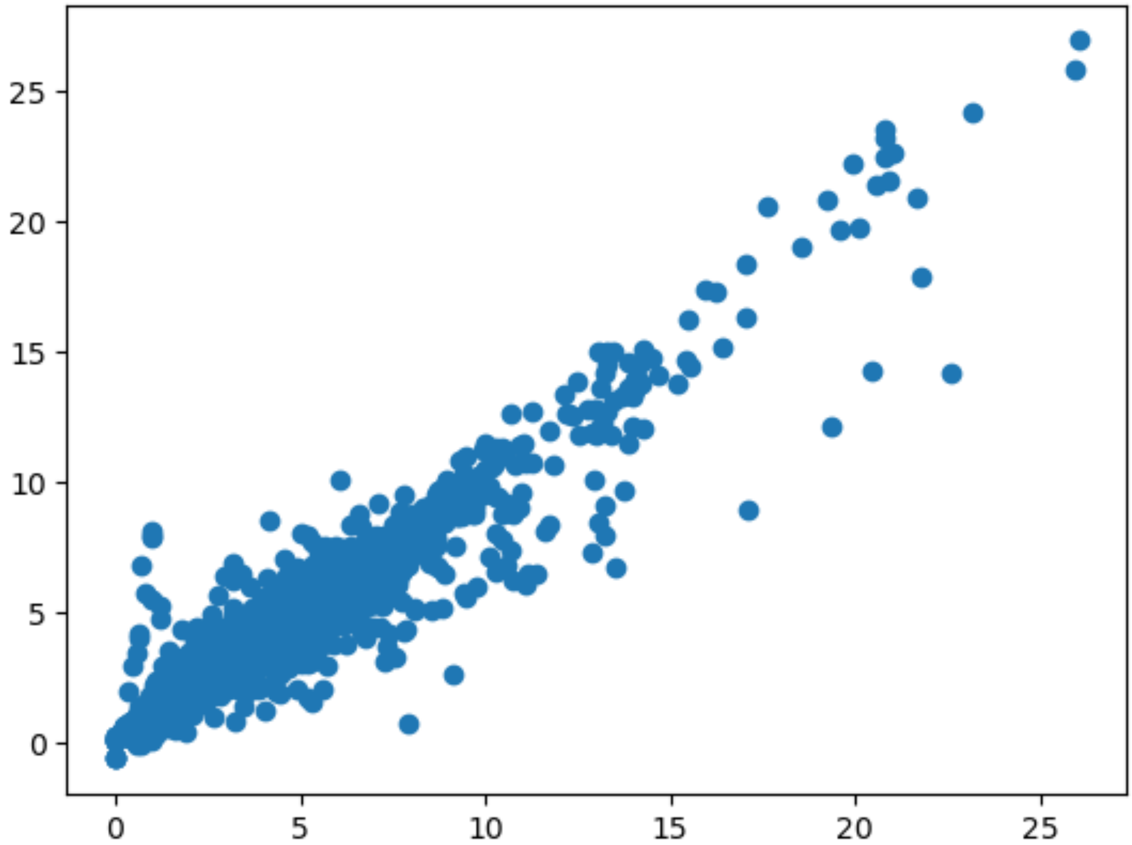
Ridge

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 0x7fe2596c59f0>



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

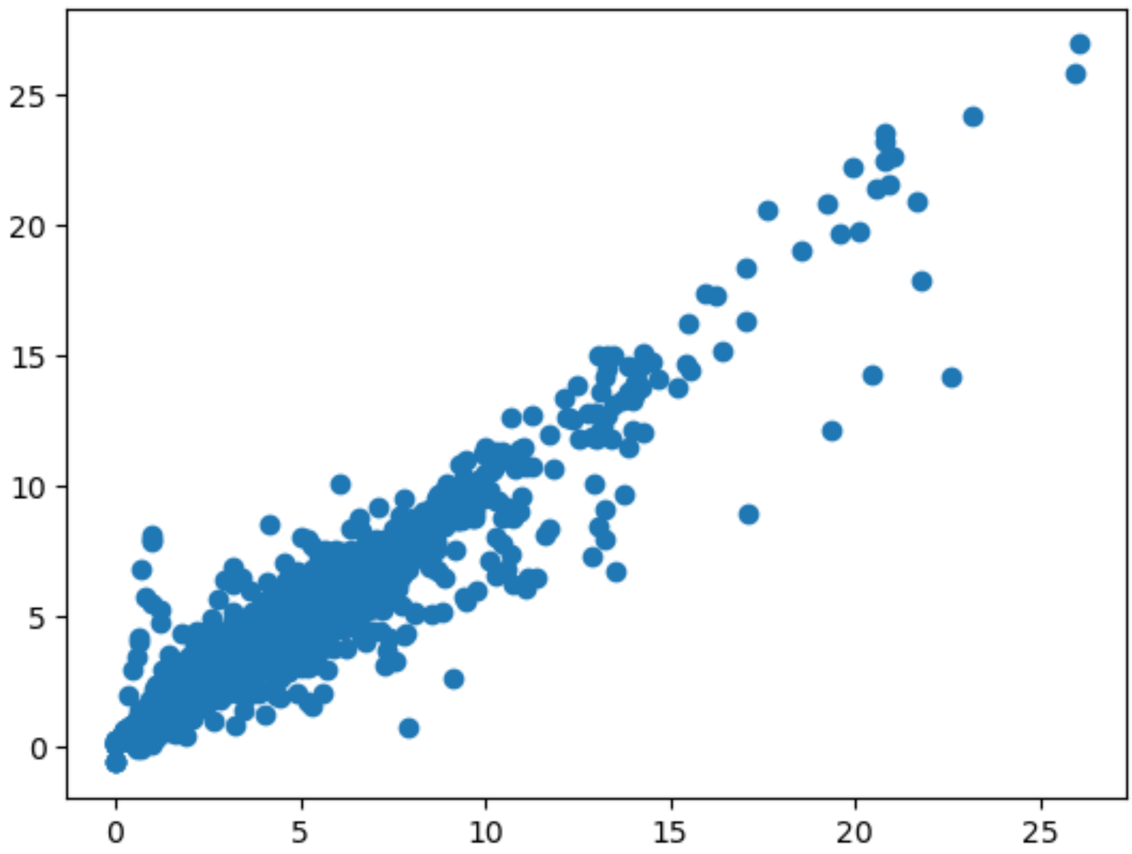
ElasticNet

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 0x7fe2596ec1c0>



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

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

```
0.9292398466663178
```

Out[24]: 0.9224085235667279

Logistic

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

Out[25]:
```
Low     12093
High     7977
Name: TCH, dtype: int64
```

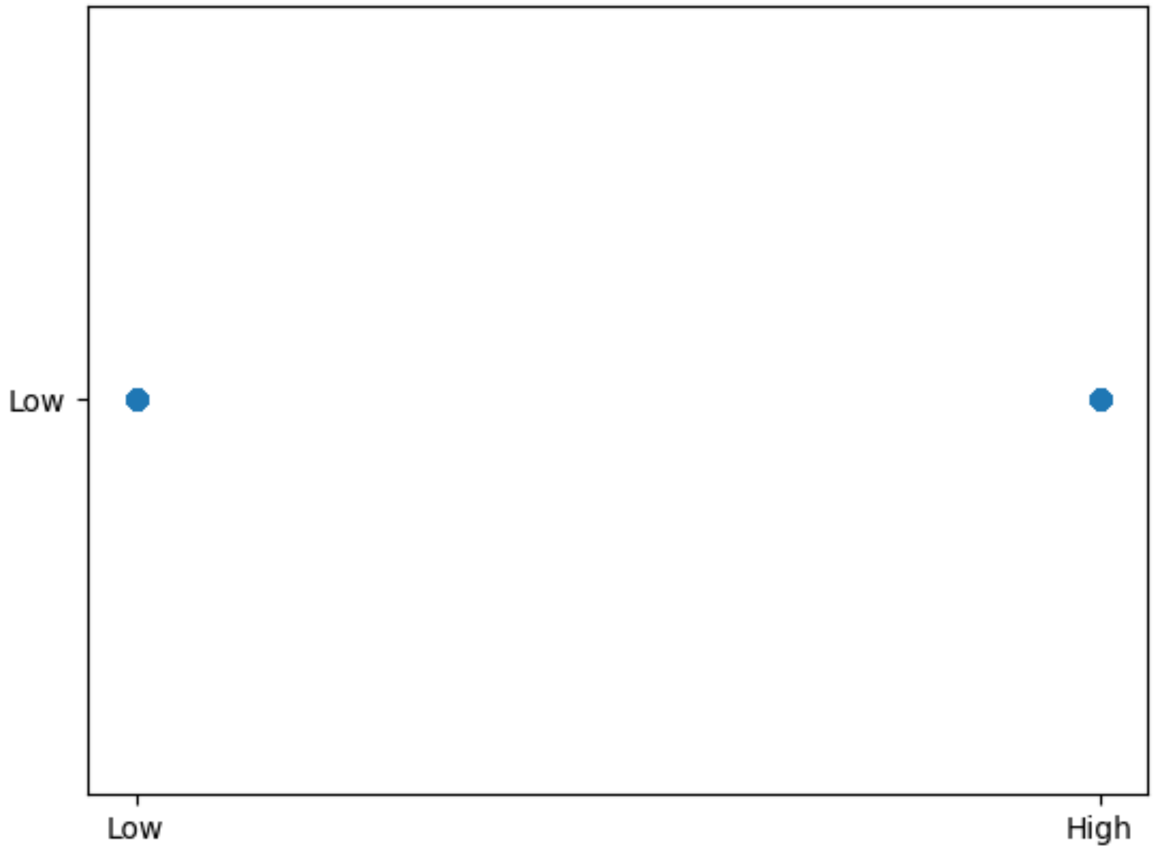
In [26]:
```
x=df1.drop(["TCH"],axis=1)
y=df1["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x7fe22abef970>



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(0.5346467391304348, 0.9285714285714286, 'O_3 <= 16.915\ngini
= 0.478\nsamples = 8844\nvalue = [8489, 5560]\nclass = Yes'),
 Text(0.2717391304347826, 0.7857142857142857, 'PM25 <= 17.745\ngin
i = 0.344\nsamples = 3169\nvalue = [1107, 3911]\nclass = No'),
 Text(0.14945652173913043, 0.6428571428571429, 'PM10 <= 17.95\ngin
i = 0.495\nsamples = 917\nvalue = [794, 651]\nclass = Yes'),
 Text(0.08152173913043478, 0.5, 'OXY <= 2.185\ngini = 0.353\nsampl
es = 370\nvalue = [448, 133]\nclass = Yes'),
 Text(0.043478260869565216, 0.35714285714285715, 'PM10 <= 10.22\ng
ini = 0.309\nsamples = 248\nvalue = [322, 76]\nclass = Yes'),
 Text(0.021739130434782608, 0.21428571428571427, 'NMHC <= 0.125\ng
ini = 0.057\nsamples = 63\nvalue = [99, 3]\nclass = Yes'),
 Text(0.010869565217391304, 0.07142857142857142, 'gini = 0.0\nsamp
les = 37\nvalue = [56, 0]\nclass = Yes'),
 Text(0.03260869565217391, 0.07142857142857142, 'gini = 0.122\nsam
ples = 26\nvalue = [43, 3]\nclass = Yes'),
 Text(0.06521739130434782, 0.21428571428571427, 'NOx <= 74.02\ngin
i = 0.372\nsamples = 185\nvalue = [223, 73]\nclass = Yes'),
 Text(0.05434782608695652, 0.07142857142857142, 'gini = 0.462\nsam
ples = 59\nvalue = [58, 33]\nclass = Yes'),
```

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

```
Linear: 0.9292382757190922
Lasso: 0.7070213696894748
Ridge: 0.9292398466663178
ElasticNet: 0.9076959882566026
Logistic: 0.5992360073077562
Random Forest: 0.9078939720652729
```

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	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000
...	...	...	...	...	...	...	...	...	...	...
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000

230568 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230568 entries, 0 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     230568 non-null  object 
 1   BEN      73979 non-null   float64
 2   CO       211665 non-null  float64
 3   EBE      73948 non-null   float64
 4   MXY      33422 non-null   float64
 5   NMHC     90829 non-null   float64
 6   NO_2     228855 non-null  float64
 7   NOx      228855 non-null  float64
 8   OXY      33472 non-null   float64
 9   O_3      216511 non-null  float64
 10  PM10     227469 non-null  float64
 11  PM25     61758 non-null   float64
 12  PXY      33447 non-null   float64
 13  SO_2     229125 non-null  float64
 14  TCH      90887 non-null   float64
 15  TOL      73840 non-null   float64
 16  station  230568 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.9+ MB
```

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

Out[42]:

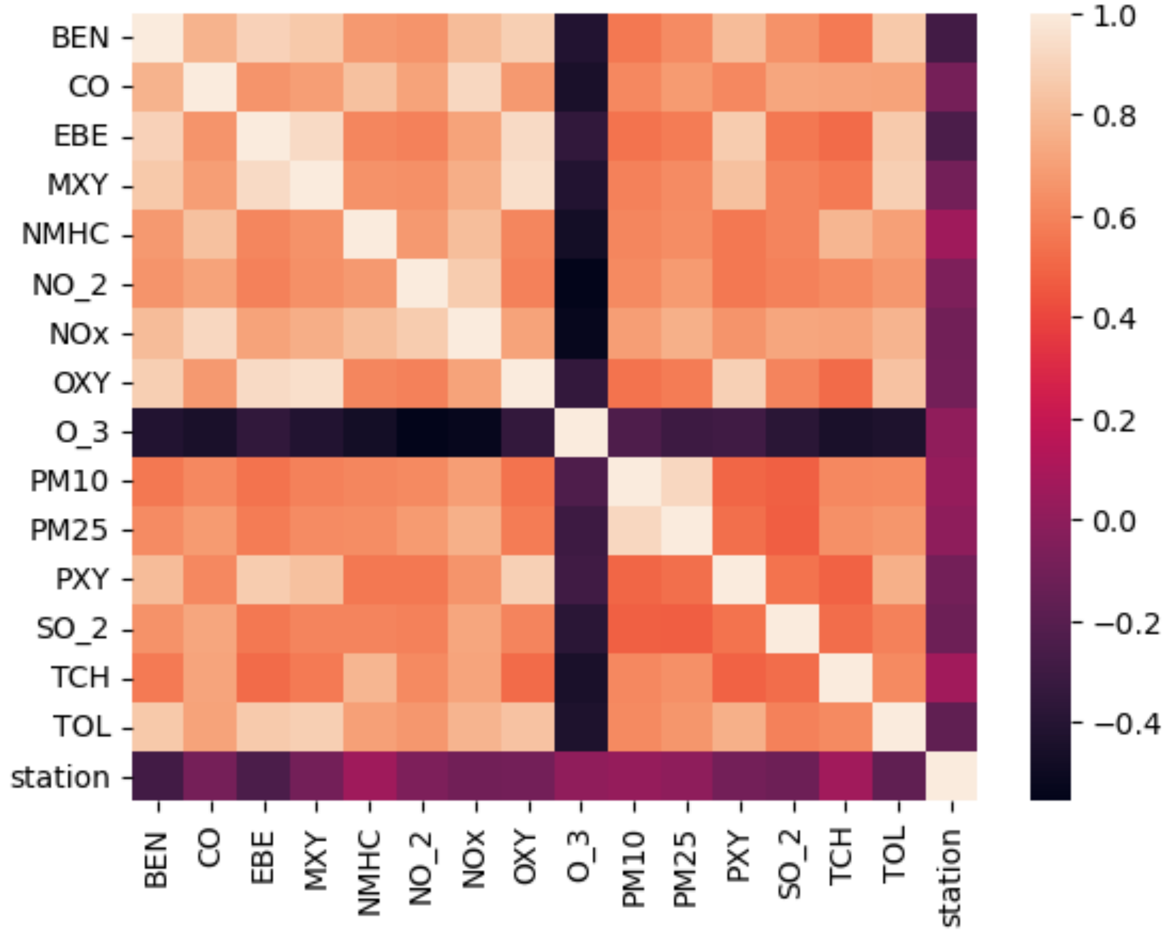
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
5	2006-02-01 01:00:00	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.99
22	2006-02-01 01:00:00	1.69	0.79	1.24	2.670000	0.17	59.910000	120.199997	1.11	2.45
25	2006-02-01 01:00:00	2.35	1.47	2.64	9.660000	0.40	117.699997	346.399994	5.15	4.78
31	2006-02-01 02:00:00	4.39	0.85	7.92	17.139999	0.25	92.059998	237.000000	9.24	5.92
48	2006-02-01 02:00:00	1.93	0.79	1.24	2.740000	0.16	60.189999	125.099998	1.11	2.28
...	...	...	...	...	...	...	...	...	...	
230538	2006-04-30 23:00:00	0.42	0.40	0.37	0.430000	0.10	49.259998	51.689999	1.00	64.59
230541	2006-04-30 23:00:00	1.63	0.94	1.53	2.200000	0.33	63.220001	211.399994	1.35	17.67
230547	2006-05-01 00:00:00	3.99	1.06	3.71	7.960000	0.26	202.399994	343.500000	3.92	11.13
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.090000	0.08	51.900002	54.820000	0.61	48.41
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.000000	0.24	107.300003	160.199997	2.01	17.73

24758 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 0x7fe24a2ce800>

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

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

Out[49]:
```
1.35    921
1.30    916
1.36    914
1.33    909
1.31    908
       ...
3.08      1
3.25      1
2.91      1
2.43      1
0.72      1
Name: TCH, Length: 188, 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    14706
2.0    10052
Name: TCH, dtype: int64
```

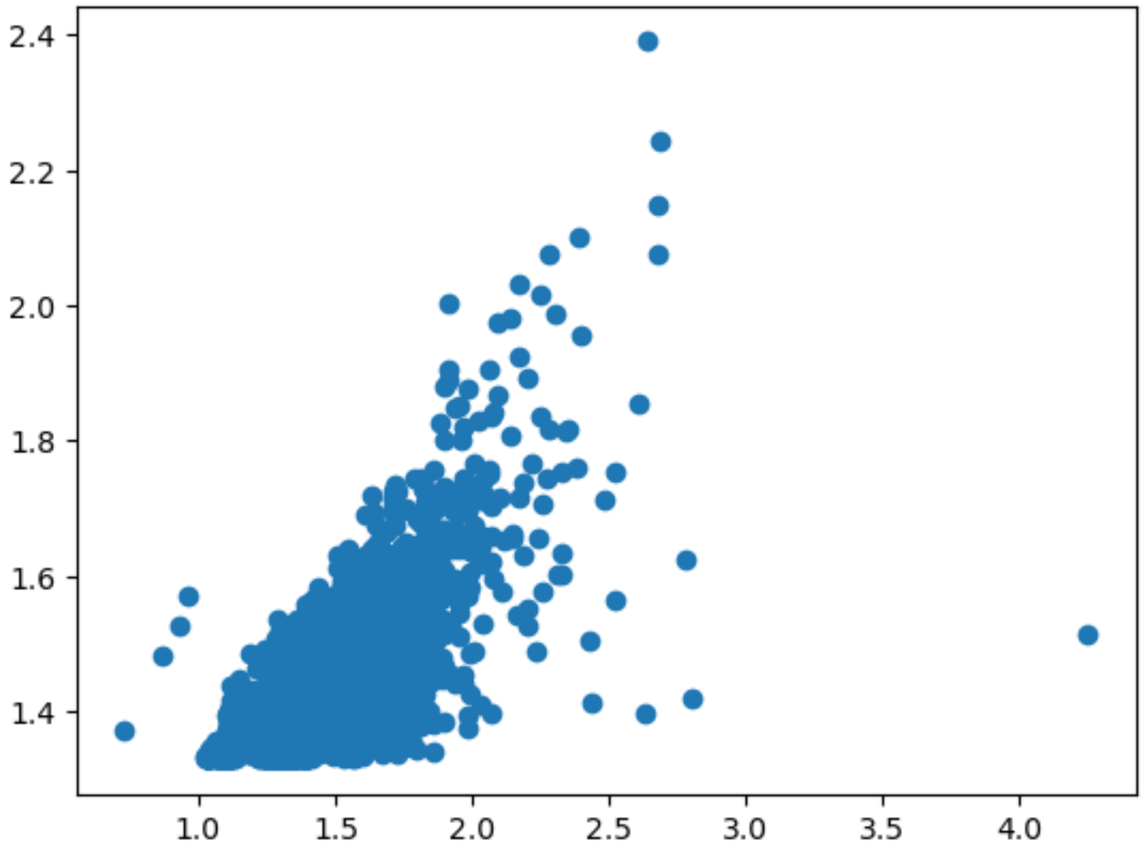
Lasso

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 0x7fe24a343c40>



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

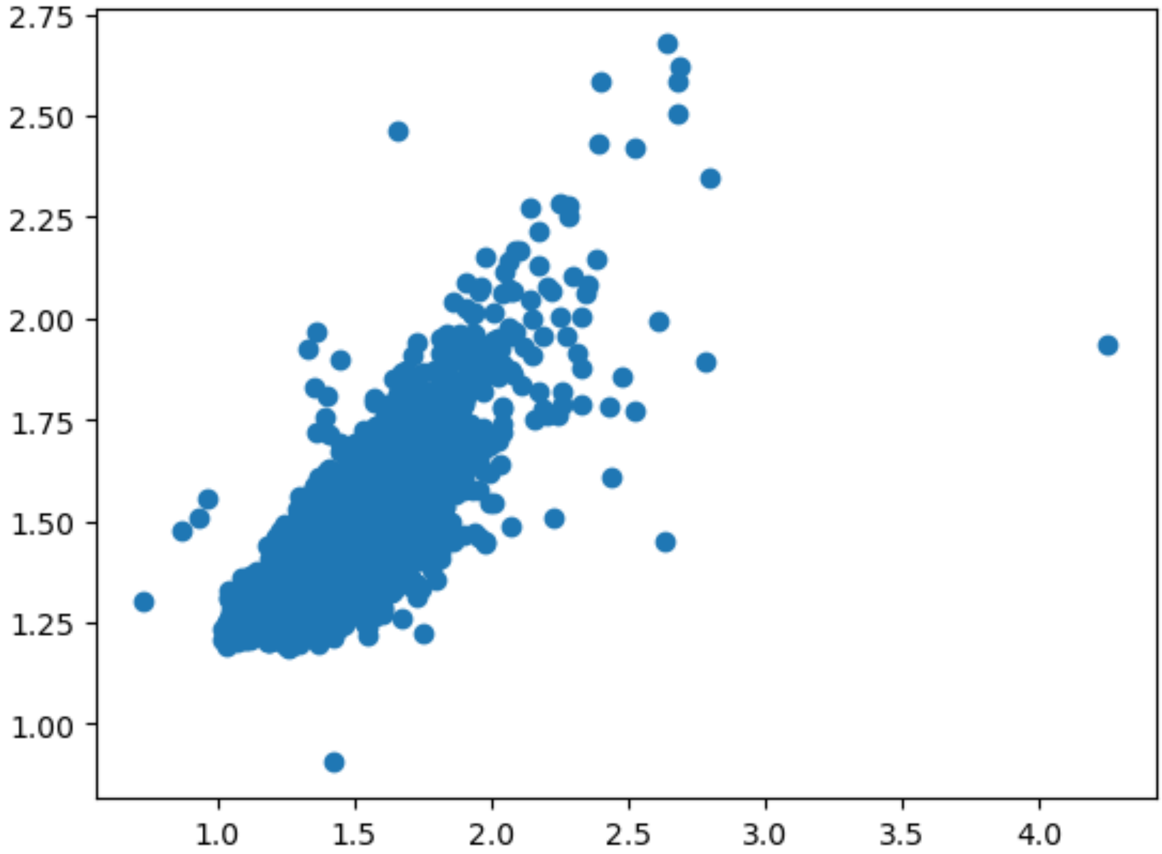
Ridge

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 0x7fe24a3b3c40>



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

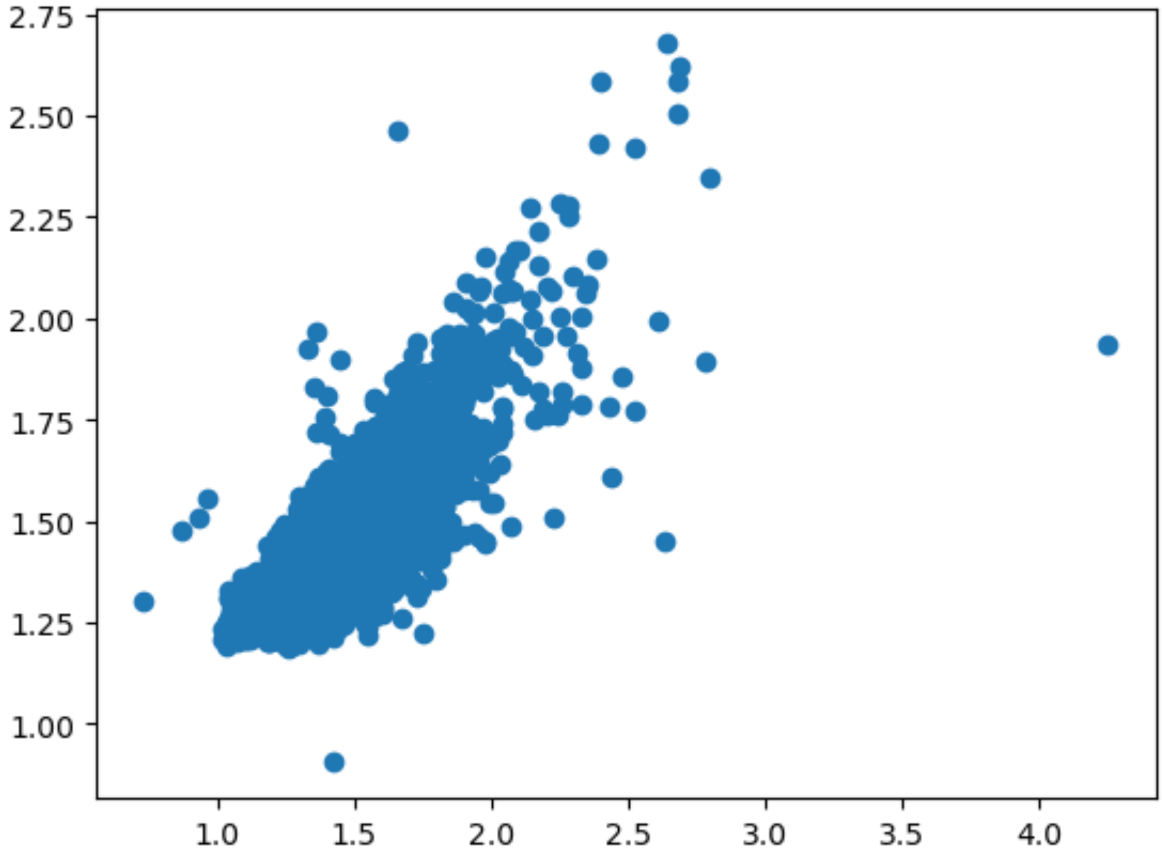
ElasticNet

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 0x7fe24a31eb60>



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

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

```
0.6705159604191635
```

Out[60]: 0.6767940774302723

Logistic

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

Out[61]:
```
Low     14706
High    10052
Name: TCH, dtype: int64
```

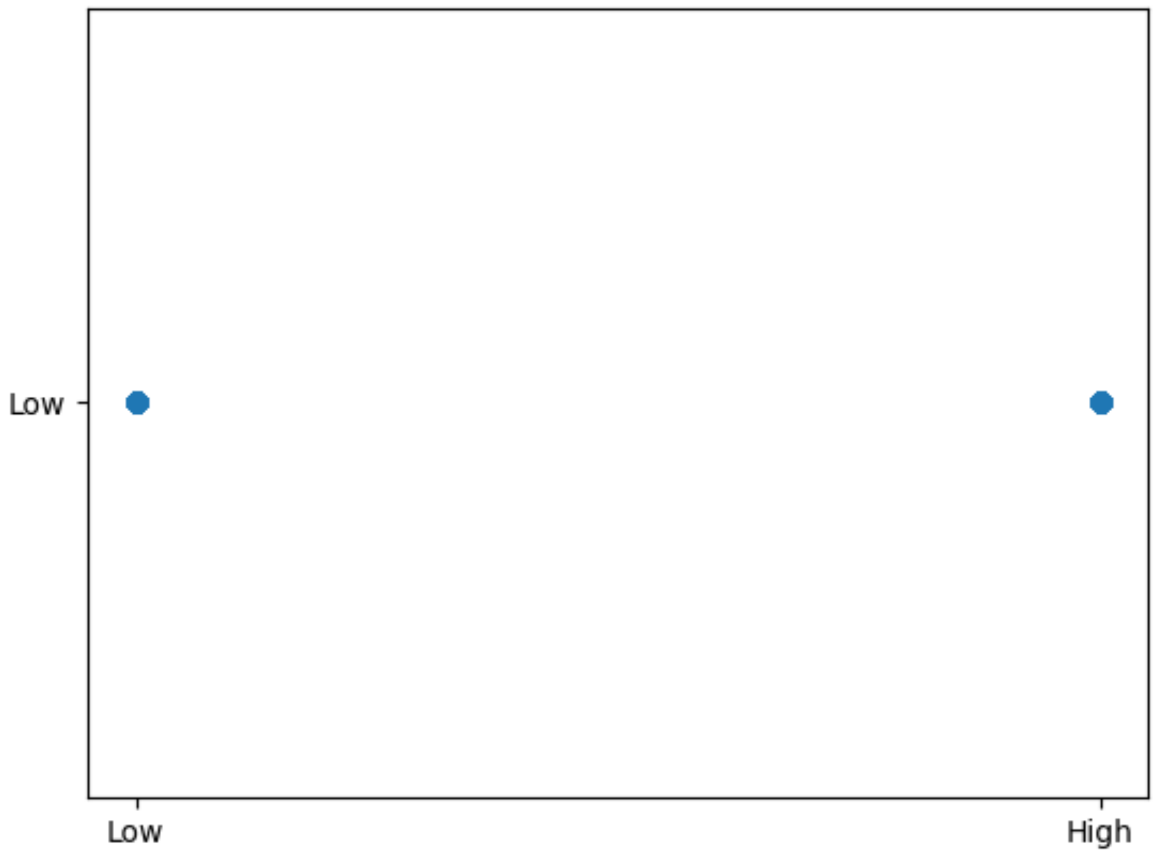
In [62]:
```
x=df3.drop(["TCH"],axis=1)
y=df3["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x7fe259e04550>



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(0.5373831775700935, 0.9285714285714286, 'NOx <= 92.36\ngini
= 0.484\nsamples = 10940\nvalue = [10226, 7104]\nclass = Yes'),
 Text(0.28154205607476634, 0.7857142857142857, 'TOL <= 2.805\ngini
= 0.315\nsamples = 5779\nvalue = [7333, 1786]\nclass = Yes'),
 Text(0.14953271028037382, 0.6428571428571429, 'O_3 <= 26.12\ngini
= 0.21\nsamples = 3607\nvalue = [5030, 681]\nclass = Yes'),
 Text(0.07476635514018691, 0.5, 'NMHC <= 0.135\ngini = 0.476\nsamp
les = 457\nvalue = [423, 271]\nclass = Yes'),
 Text(0.037383177570093455, 0.35714285714285715, 'BEN <= 0.615\ngi
ni = 0.274\nsamples = 235\nvalue = [286, 56]\nclass = Yes'),
 Text(0.018691588785046728, 0.21428571428571427, 'NO_2 <= 47.57\ng
ini = 0.349\nsamples = 142\nvalue = [155, 45]\nclass = Yes'),
 Text(0.009345794392523364, 0.07142857142857142, 'gini = 0.271\nsa
mples = 115\nvalue = [140, 27]\nclass = Yes'),
 Text(0.028037383177570093, 0.07142857142857142, 'gini = 0.496\nsa
mples = 27\nvalue = [15, 18]\nclass = No'),
 Text(0.056074766355140186, 0.21428571428571427, 'NMHC <= 0.115\ng
ini = 0.143\nsamples = 93\nvalue = [131, 11]\nclass = Yes'),
 Text(0.04672897196261682, 0.07142857142857142, 'gini = 0.026\nsam
```

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

```
Linear: 0.6704000731025829
Lasso: 0.4481089030955524
Ridge: 0.6705159604191635
ElasticNet: 0.5331870062979789
Logistic: 0.5965266558966075
Random Forest: 0.8661858049624929
```

Best model is Random Forest