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	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999
...	...	...	...	...	...	...	...	...	...	...
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN	22.150000
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN	18.540001
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN	NaN
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000

225120 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225120 entries, 0 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   date     225120 non-null  object
 1   BEN      68885 non-null   float64
 2   CO       206748 non-null  float64
 3   EBE      68883 non-null   float64
 4   MXY      26061 non-null   float64
 5   NMHC     86883 non-null   float64
 6   NO_2     223985 non-null  float64
 7   NOx      223972 non-null  float64
 8   OXY      26062 non-null   float64
 9   O_3      211850 non-null  float64
 10  PM10     222588 non-null  float64
 11  PM25     68870 non-null   float64
 12  PXY      26062 non-null   float64
 13  SO_2     224372 non-null  float64
 14  TCH      87026 non-null   float64
 15  TOL      68845 non-null   float64
 16  station  225120 non-null  int64
dtypes: float64(15), int64(1), object(1)
memory usage: 29.2+ MB
```

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

Out[4]:

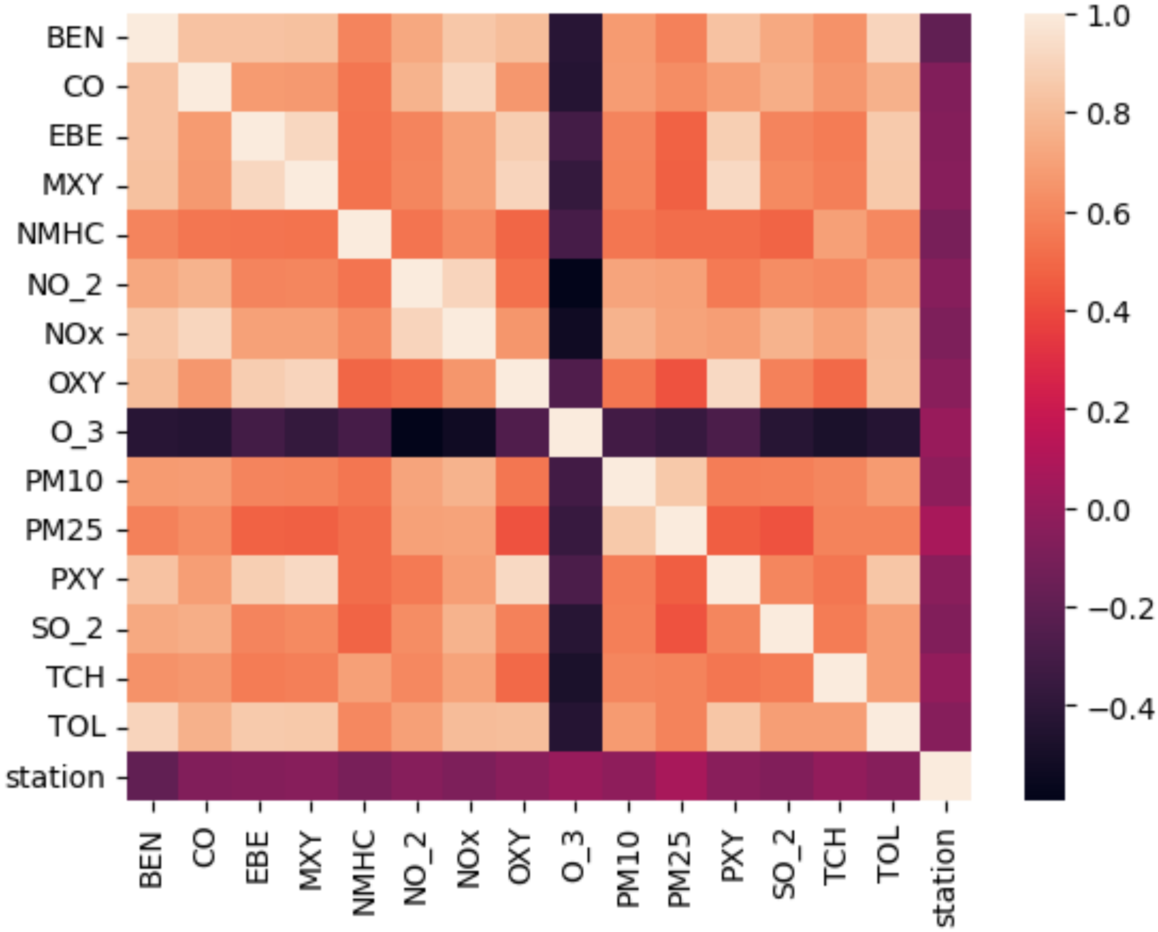
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999
21	2007-12-01 01:00:00	1.98	0.31	2.56	6.06	0.35	76.059998	208.899994	1.70	1.000000
25	2007-12-01 01:00:00	2.82	1.42	3.15	7.02	0.49	123.099998	402.399994	2.60	7.160000
30	2007-12-01 02:00:00	4.65	1.89	4.41	8.21	0.65	151.000000	622.700012	3.55	58.080002
47	2007-12-01 02:00:00	1.97	0.30	2.15	5.08	0.33	78.760002	189.800003	1.62	1.000000
...	...	...	...	...	...	...	...	...	...	...
225073	2007-02-28 23:00:00	2.12	0.47	2.51	4.99	0.05	43.560001	83.889999	2.57	13.090000
225094	2007-02-28 23:00:00	0.87	0.45	1.19	2.66	0.13	40.000000	61.959999	1.79	20.440001
225098	2007-03-01 00:00:00	0.95	0.41	1.55	3.11	0.05	36.090000	63.349998	1.74	17.160000
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000

25443 rows × 17 columns

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

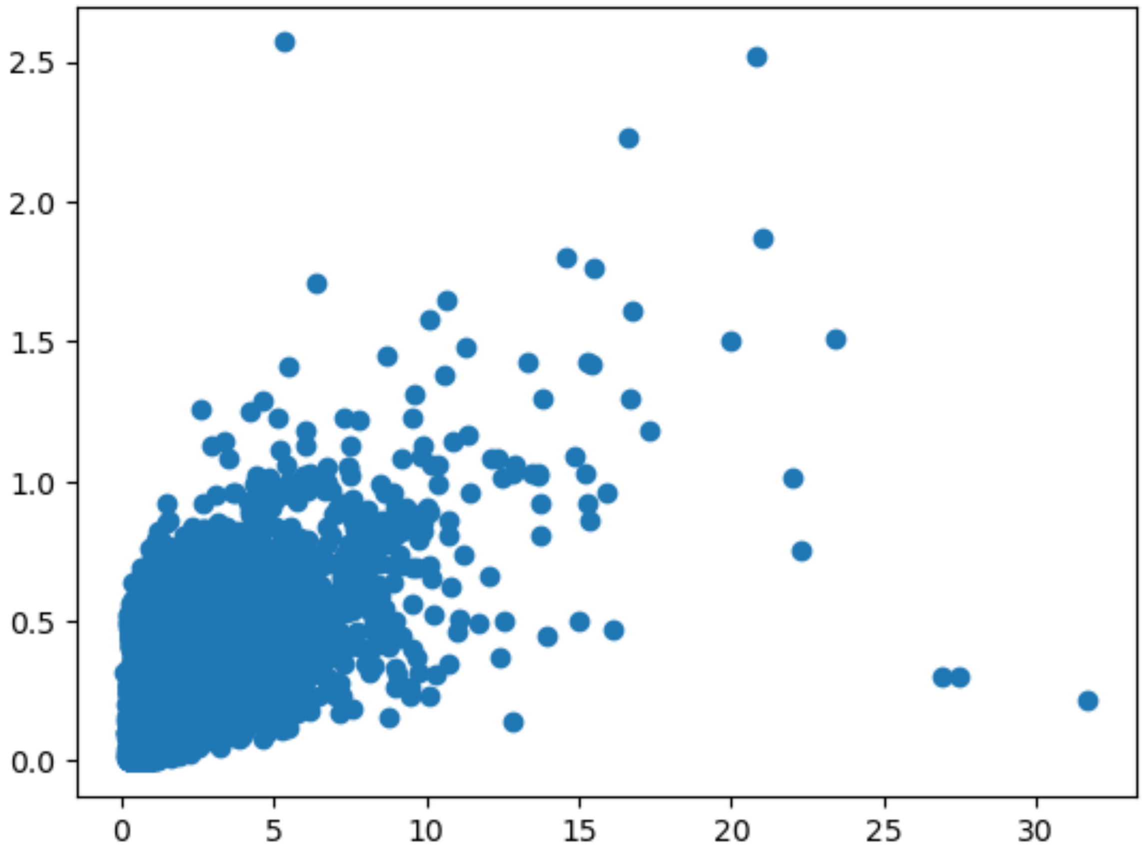
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <Axes: >



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

Out[7]:
```
[<matplotlib.lines.Line2D at 0x7fd5495b9150>]
```



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)
```

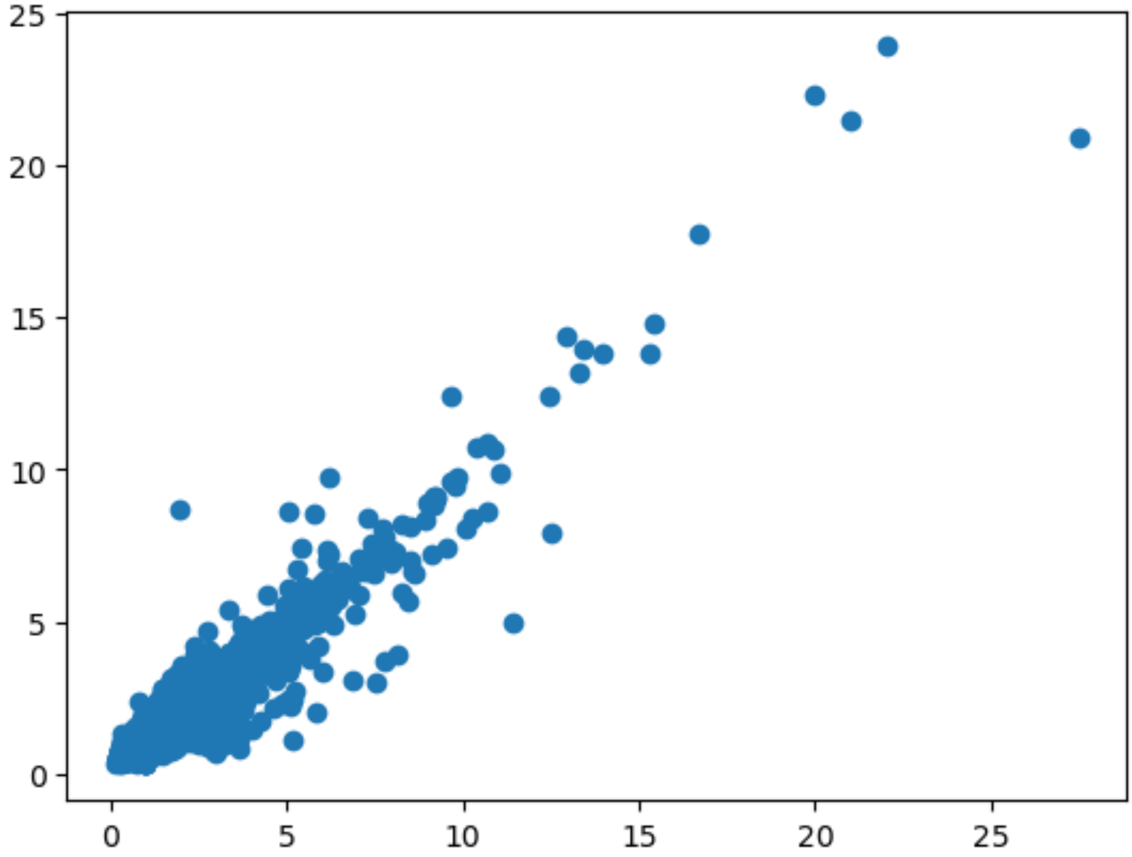
Linear

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 0x7fd5437eb850>



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

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

Out[13]:
```
1.34    1130
1.33    1067
1.35    1037
1.36    1002
1.32     991
        ...
3.03       1
4.07       1
3.70       1
2.52       1
0.58       1
Name: TCH, Length: 250, 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    14025
2.0    11418
Name: TCH, dtype: int64
```

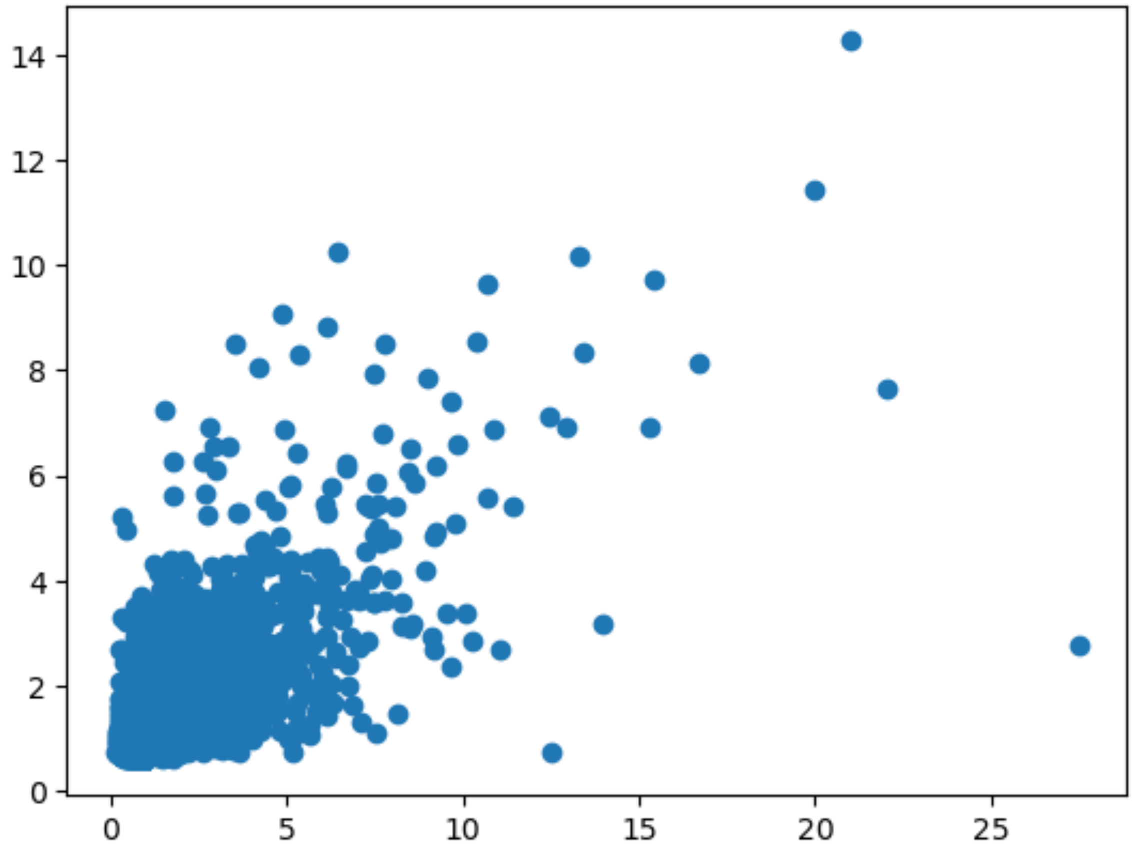
Lasso

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 0x7fd543867dc0>



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

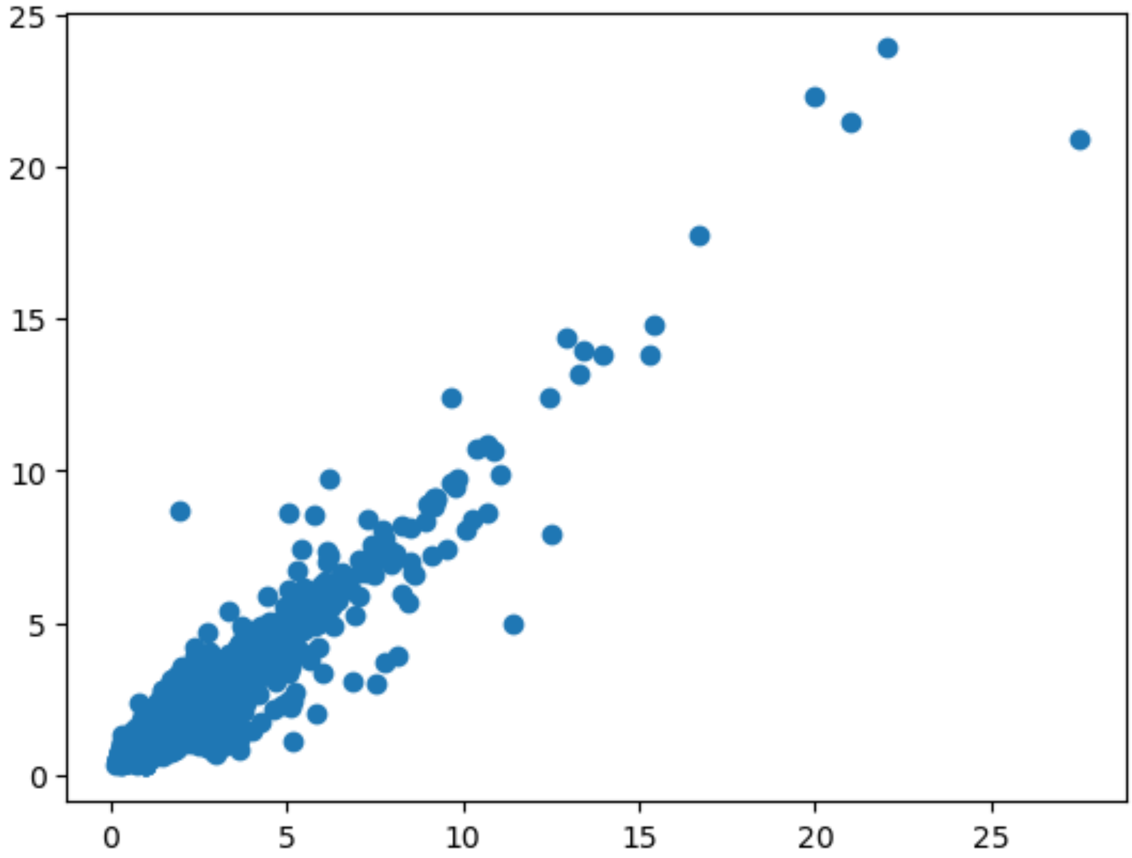
Ridge

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 0x7fd54384bee0>



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

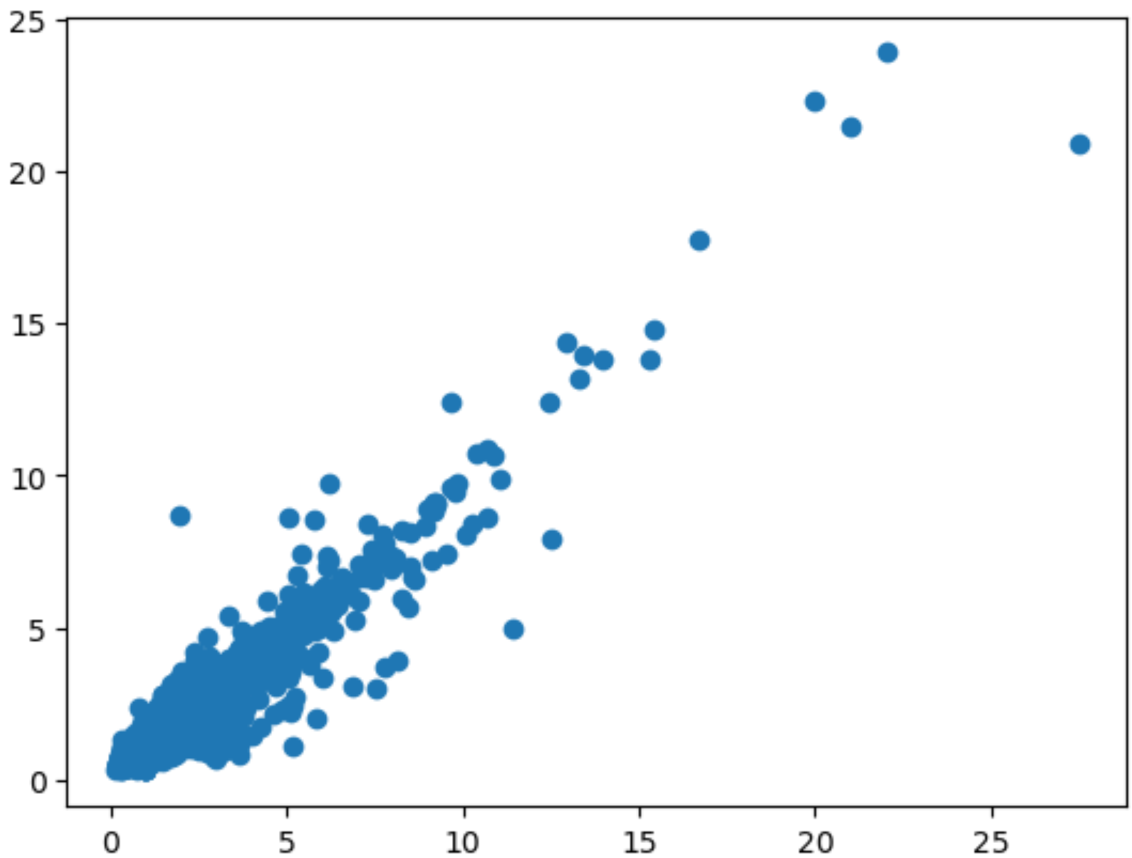
ElasticNet

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 0x7fd5313f3010>



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

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

```
0.9031092096516402
```

Out[24]: 0.8640132513733083

Logistic

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

Out[25]:
```
Low     14025
High    11418
Name: TCH, dtype: int64
```

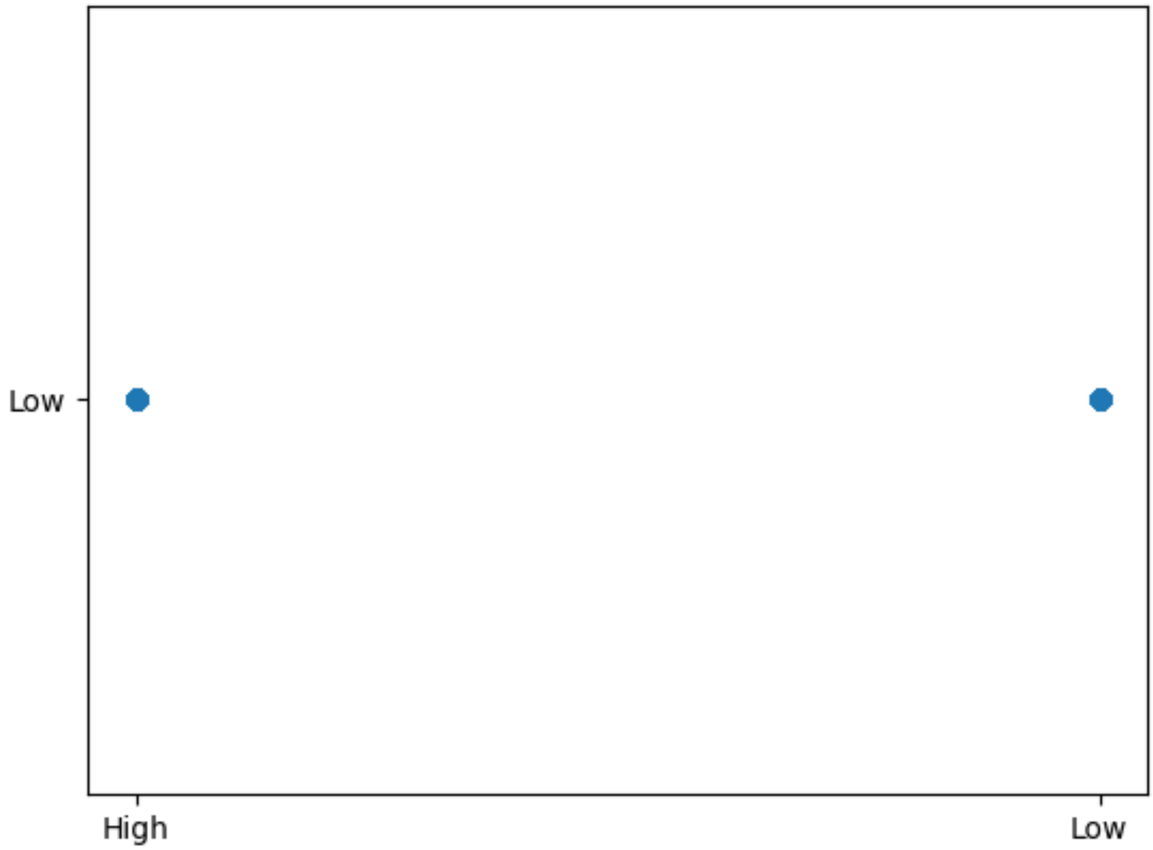
In [26]:
```
x=df1.drop(["TCH"],axis=1)
y=df1["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x7fd5314f5a50>



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(0.5206473214285714, 0.9285714285714286, 'O_3 <= 18.185\ngini
= 0.493\nsamples = 11219\nvalue = [9946, 7864]\nclass = Yes'),
 Text(0.26674107142857145, 0.7857142857142857, 'NMHC <= 0.225\ngin
i = 0.27\nsamples = 3787\nvalue = [972, 5075]\nclass = No'),
 Text(0.140625, 0.6428571428571429, 'NO_2 <= 91.22\ngini = 0.496\n
samples = 1005\nvalue = [740, 878]\nclass = No'),
 Text(0.07142857142857142, 0.5, 'station <= 28079015.0\ngini = 0.5
\nsamples = 844\nvalue = [701, 670]\nclass = Yes'),
 Text(0.03571428571428571, 0.35714285714285715, 'NO_2 <= 76.305\ng
ini = 0.377\nsamples = 395\nvalue = [465, 157]\nclass = Yes'),
 Text(0.017857142857142856, 0.21428571428571427, 'NOx <= 182.4\ngi
ni = 0.332\nsamples = 306\nvalue = [394, 105]\nclass = Yes'),
 Text(0.008928571428571428, 0.07142857142857142, 'gini = 0.314\nsa
mples = 294\nvalue = [387, 94]\nclass = Yes'),
 Text(0.026785714285714284, 0.07142857142857142, 'gini = 0.475\nsa
mples = 12\nvalue = [7, 11]\nclass = No'),
 Text(0.05357142857142857, 0.21428571428571427, 'PM10 <= 44.9\ngin
i = 0.488\nsamples = 89\nvalue = [71, 52]\nclass = Yes'),
 Text(0.044642857142857144, 0.07142857142857142, 'gini = 0.428\nsa
```

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

```
Linear: 0.903105908273488
Lasso: 0.49881871289711766
Ridge: 0.9031092096516402
ElasticNet: 0.8369295468530789
Logistic: 0.5470981265557447
Random Forest: 0.8717012914093206
```

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	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000
...	...	...	...	...	...	...	...	...	...	...
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000

226392 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 226392 entries, 0 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     226392 non-null  object 
 1   BEN      67047 non-null   float64
 2   CO       208109 non-null  float64
 3   EBE      67044 non-null   float64
 4   MXY      25867 non-null   float64
 5   NMHC     85079 non-null   float64
 6   NO_2     225315 non-null  float64
 7   NOx      225311 non-null  float64
 8   OXY      25878 non-null   float64
 9   O_3      215716 non-null  float64
 10  PM10     220179 non-null  float64
 11  PM25     67833 non-null   float64
 12  PXY      25877 non-null   float64
 13  SO_2     225405 non-null  float64
 14  TCH      85107 non-null   float64
 15  TOL      66940 non-null   float64
 16  station  226392 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.4+ MB
```

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

Out[42]:

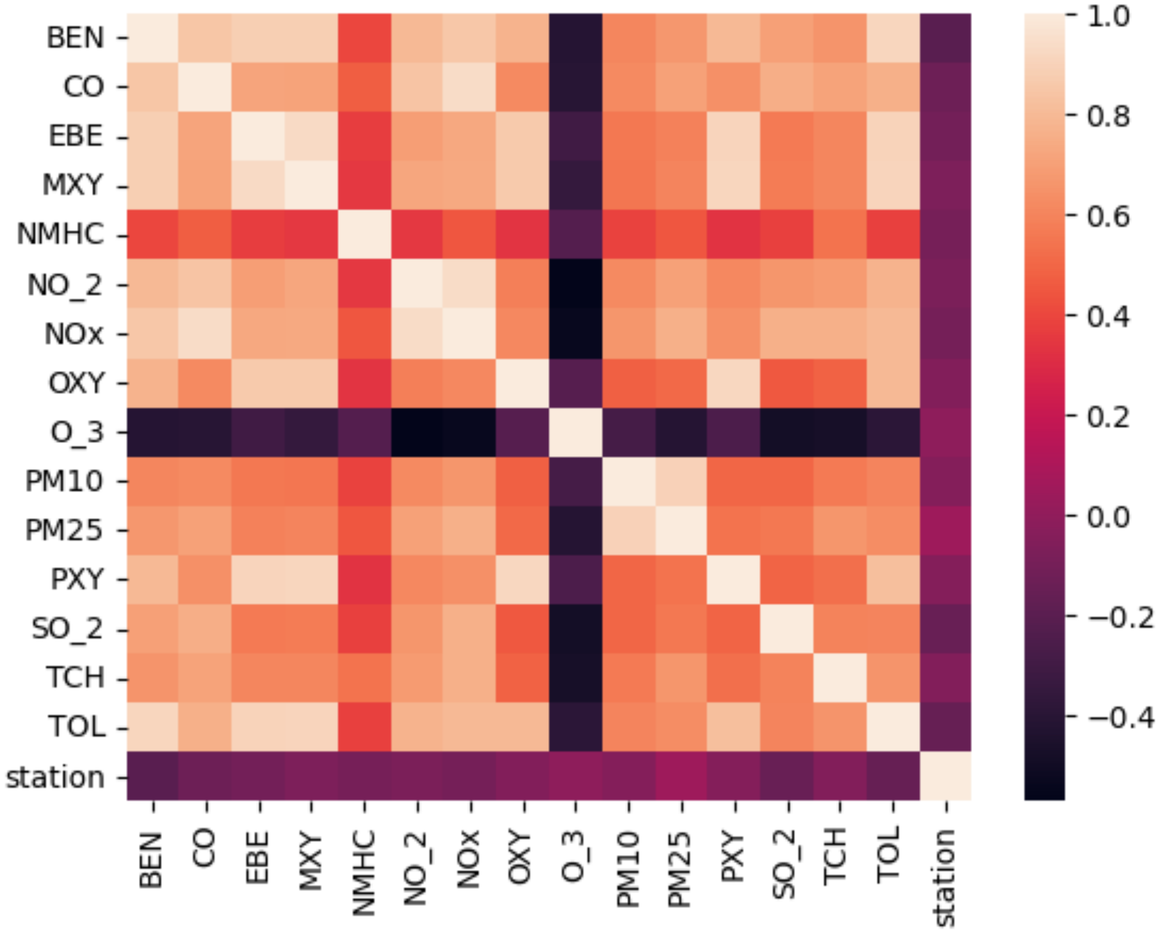
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000
21	2008-06-01 01:00:00	0.32	0.37	1.00	0.39	0.33	21.580000	22.180000	1.00	35.770000
25	2008-06-01 01:00:00	0.73	0.39	1.04	1.70	0.18	64.839996	86.709999	1.31	23.379999
30	2008-06-01 02:00:00	1.95	0.51	1.98	3.77	0.24	79.750000	143.399994	2.03	18.090000
47	2008-06-01 02:00:00	0.36	0.39	0.39	0.50	0.34	26.790001	27.389999	1.00	33.029999
...	...	...	...	...	...	...	...	...	...	...
226362	2008-10-31 23:00:00	0.47	0.35	0.65	1.00	0.33	22.480000	25.020000	1.00	33.509998
226366	2008-10-31 23:00:00	0.92	0.46	1.21	2.75	0.19	78.440002	106.199997	1.70	18.320000
226371	2008-11-01 00:00:00	1.83	0.53	2.22	4.51	0.17	93.260002	158.399994	2.38	18.770000
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000

25631 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)
```

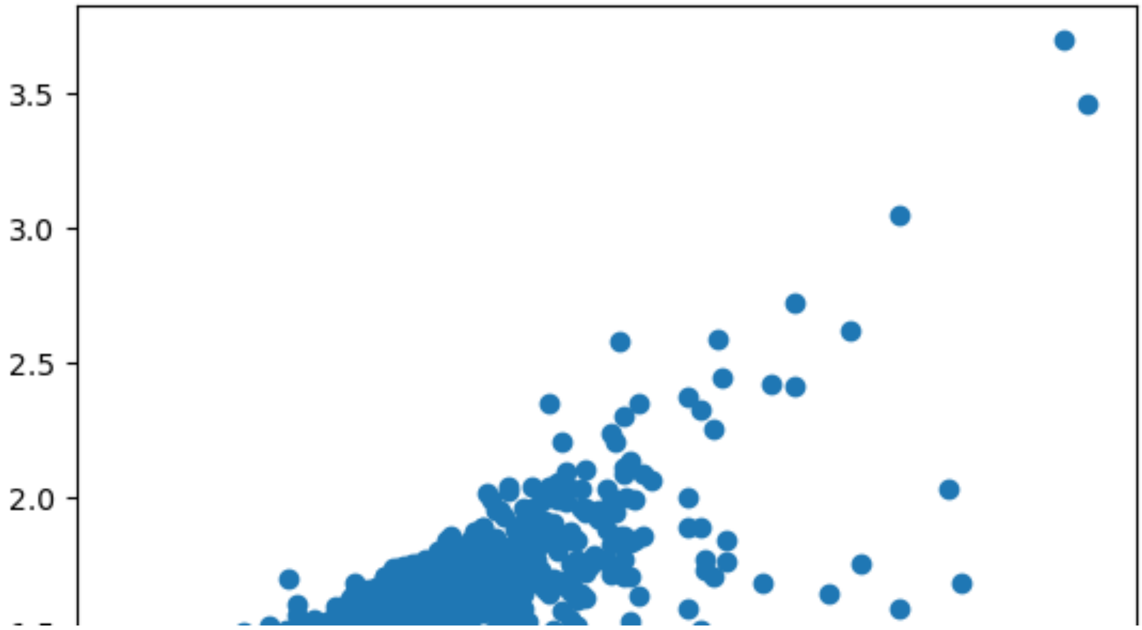
Linear

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 0x7fd54aacd2a0>
```



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

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

Out[49]:
```
1.38    1274
1.37    1246
1.36    1243
1.39    1242
1.35    1209
        ...
3.30       1
2.95       1
3.38       1
2.51       1
1.02       1
Name: TCH, Length: 177, 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]:
```
2.0    12904
1.0    12727
Name: TCH, dtype: int64
```

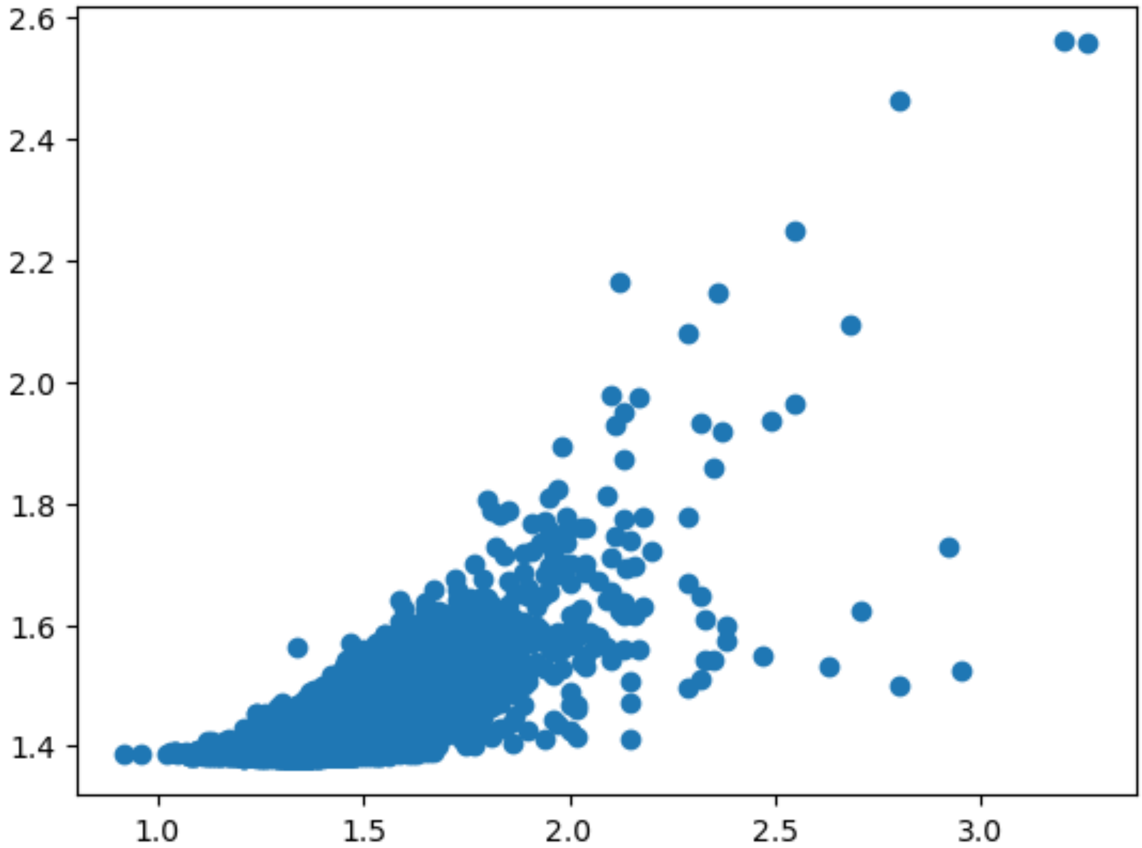
Lasso

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 0x7fd55143e290>



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

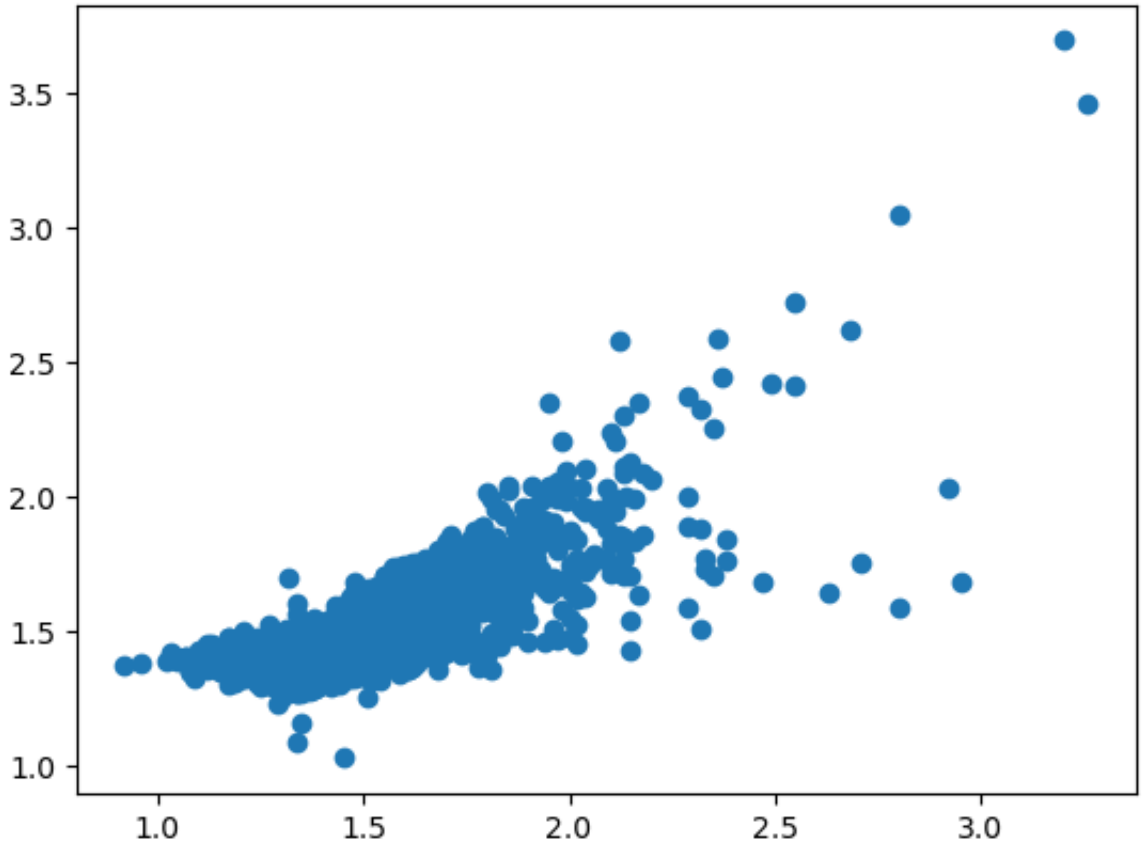
Ridge

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 0x7fd54457dea0>



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

ElasticNet

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

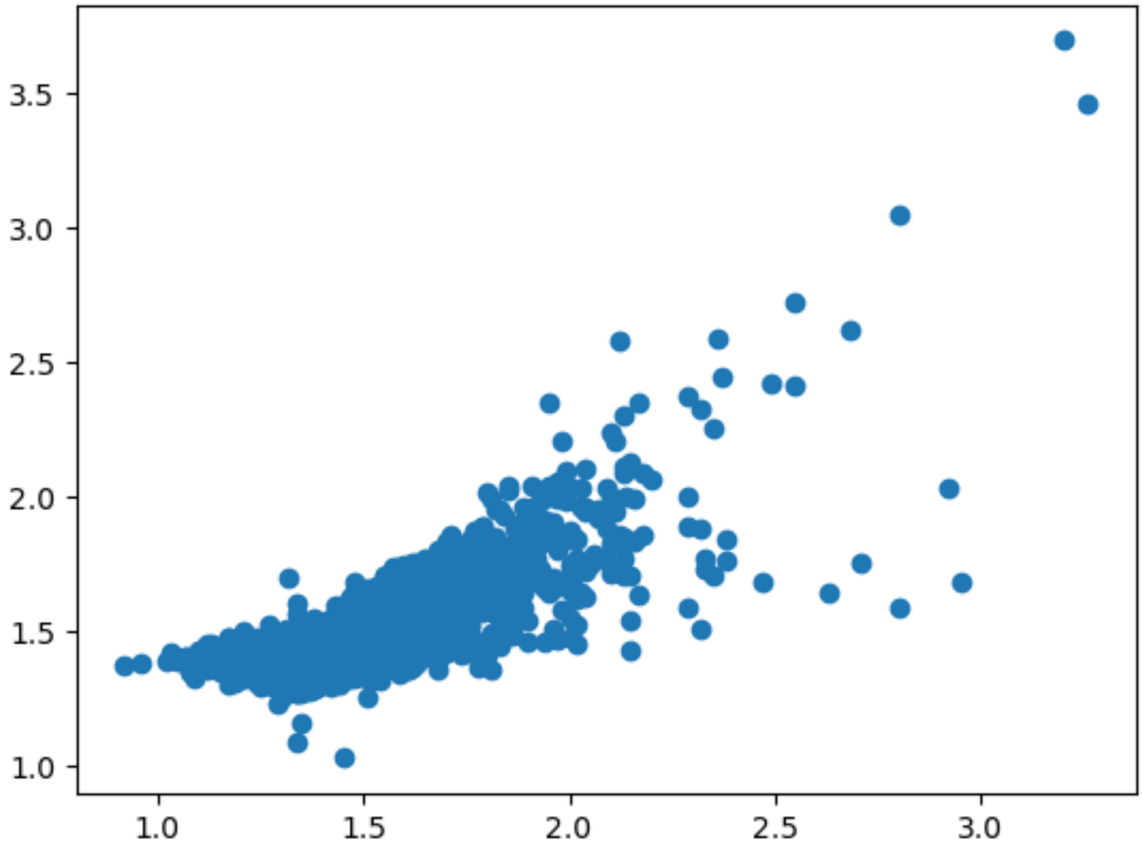
Out[57]:

▾ ElasticNet

ElasticNet()

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

Out[58]: <matplotlib.collections.PathCollection at 0x7fd5445fc940>



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

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

```
0.6686325972014862
```

Out[60]: 0.6552401520013147

Logistic

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

Out[61]:
```
High    12904
Low     12727
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 0x7fd5449f6890>

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(0.5114583333333333, 0.9285714285714286, 'TOL <= 4.835\ngini
= 0.5\nsamples = 11347\nvalue = [8960, 8981]\nclass = No'),
 Text(0.26666666666666666, 0.7857142857142857, 'O_3 <= 26.165\ngin
i = 0.436\nsamples = 7485\nvalue = [8047, 3809]\nclass = Yes'),
 Text(0.13333333333333333, 0.6428571428571429, 'SO_2 <= 11.4\ngini
= 0.435\nsamples = 1795\nvalue = [914, 1945]\nclass = No'),
 Text(0.06666666666666667, 0.5, 'MXY <= 1.325\ngini = 0.474\nsampl
es = 1055\nvalue = [647, 1029]\nclass = No'),
 Text(0.03333333333333333, 0.35714285714285715, 'TOL <= 3.185\ngin
i = 0.499\nsamples = 516\nvalue = [437, 410]\nclass = Yes'),
 Text(0.016666666666666666, 0.21428571428571427, 'EBE <= 0.605\ngi
ni = 0.496\nsamples = 451\nvalue = [404, 335]\nclass = Yes'),
 Text(0.008333333333333333, 0.07142857142857142, 'gini = 0.497\nsa
mples = 229\nvalue = [176, 204]\nclass = No'),
 Text(0.025, 0.07142857142857142, 'gini = 0.463\nsamples = 222\nva
lue = [228, 131]\nclass = Yes'),
 Text(0.05, 0.21428571428571427, 'NMHC <= 0.135\ngini = 0.424\nsam
ples = 65\nvalue = [33, 75]\nclass = No'),
 Text(0.041666666666666664, 0.07142857142857142, 'gini = 0.43\nsam
```

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

```
Linear: 0.6686214896300879
Lasso: 0.4702435204100489
Ridge: 0.6686325972014862
ElasticNet: 0.5832313700304619
Logistic: 0.5042912873862159
Random Forest: 0.8318935211402727
```

Best model is Random Forest