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	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN

209928 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209928 entries, 0 to 209927
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209928 non-null  object 
 1   BEN      51393 non-null   float64
 2   CO       87127 non-null   float64
 3   EBE      51350 non-null   float64
 4   NMHC     43517 non-null   float64
 5   NO       208954 non-null  float64
 6   NO_2     208973 non-null  float64
 7   O_3      122049 non-null  float64
 8   PM10     103743 non-null  float64
 9   PM25     51079 non-null   float64
 10  SO_2     87131 non-null   float64
 11  TCH      43519 non-null   float64
 12  TOL      51175 non-null   float64
 13  station  209928 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

Out[4]:

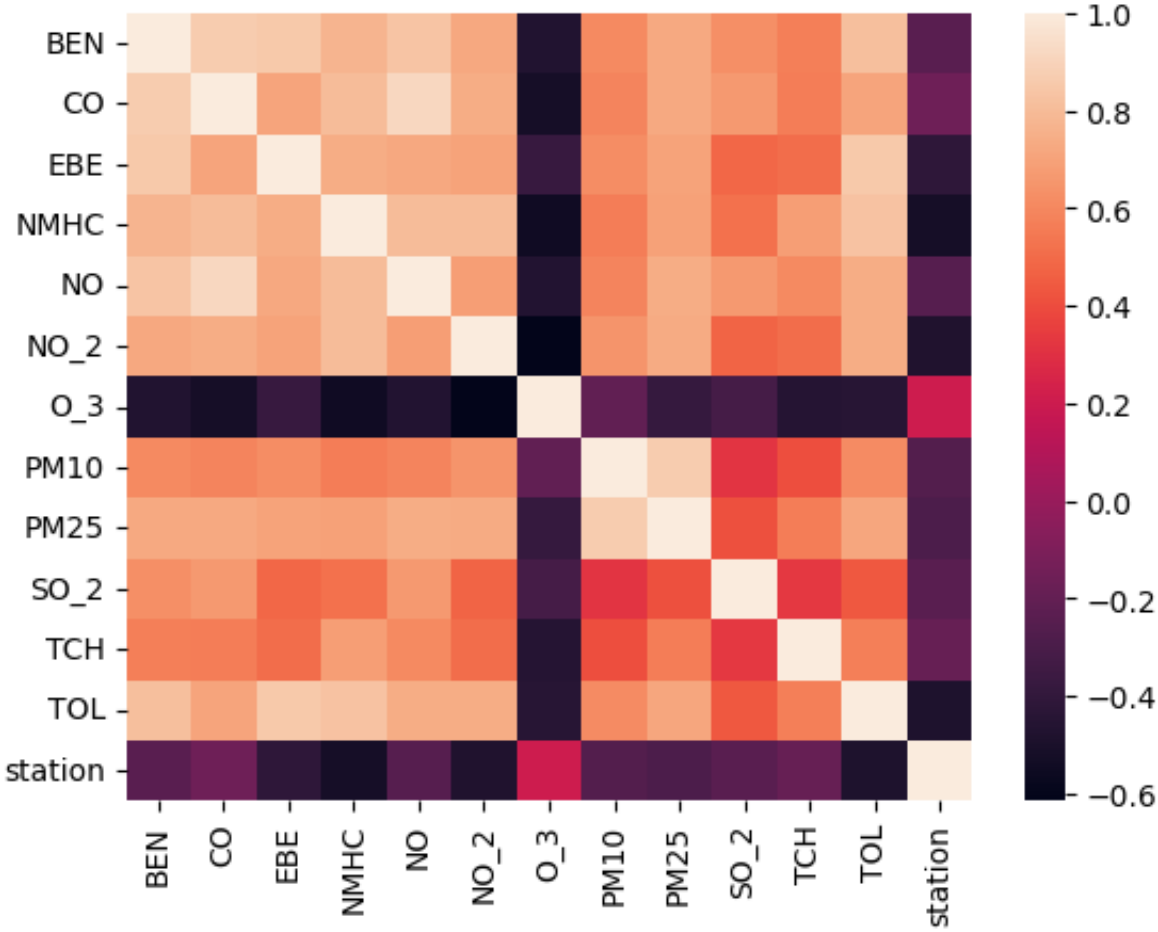
	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TO
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.
6	2011-11-01 01:00:00	0.7	0.3	1.1	0.16	17.0	66.0	7.0	22.0	16.0	2.0	1.36	1.
25	2011-11-01 02:00:00	1.8	0.3	2.8	0.20	34.0	76.0	3.0	34.0	21.0	8.0	1.71	7.
30	2011-11-01 02:00:00	1.0	0.4	1.3	0.18	31.0	67.0	5.0	25.0	18.0	3.0	1.40	2.
49	2011-11-01 03:00:00	1.3	0.2	2.4	0.22	29.0	72.0	3.0	33.0	20.0	8.0	1.75	6.
...	...	...	...	...	...	...	...	...	...	...	...	...	.
209862	2011-08-31 22:00:00	0.4	0.1	1.0	0.06	1.0	13.0	33.0	21.0	6.0	5.0	1.26	0.
209881	2011-08-31 23:00:00	0.9	0.1	1.8	0.16	11.0	45.0	30.0	32.0	17.0	3.0	1.34	4.
209886	2011-08-31 23:00:00	0.6	0.1	1.1	0.05	1.0	12.0	48.0	19.0	7.0	5.0	1.26	0.
209905	2011-09-01 00:00:00	0.6	0.1	1.3	0.15	6.0	35.0	34.0	21.0	12.0	3.0	1.32	3.
209910	2011-09-01 00:00:00	0.7	0.1	1.1	0.04	1.0	12.0	46.0	8.0	5.0	5.0	1.25	0.

16460 rows × 14 columns

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

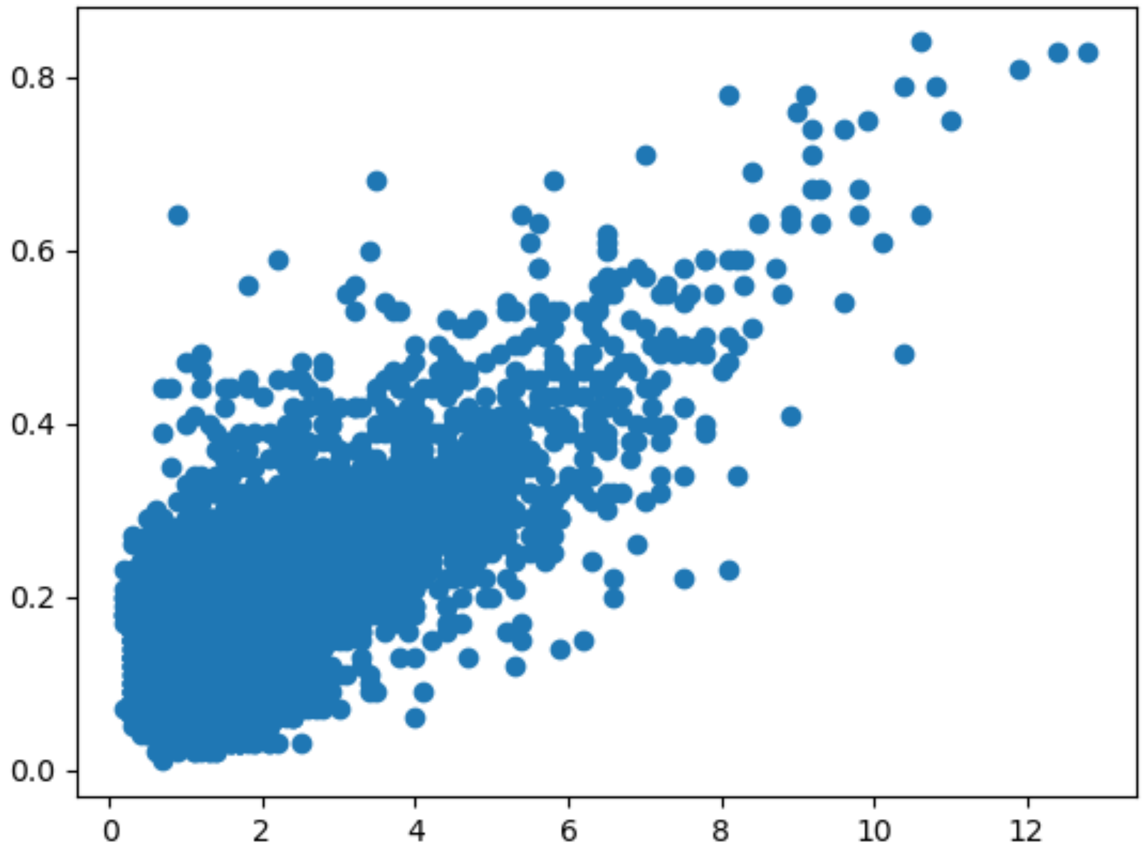
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <Axes: >



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

Out[7]: [<matplotlib.lines.Line2D at 0x7fcca2d3e080>]



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)
```

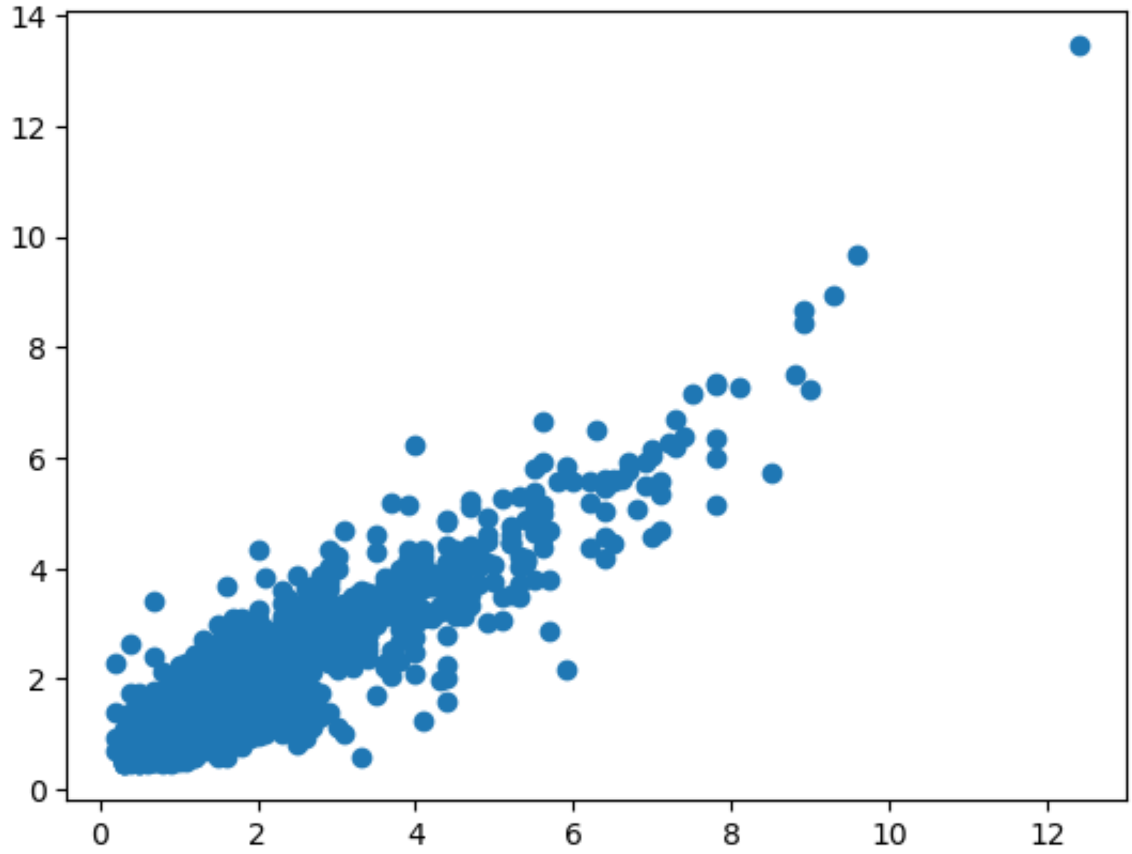
Linear

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 0x7fccb1523fd0>



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

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

Out[13]:
```
1.30    897
1.29    878
1.28    856
1.31    827
1.27    820
       ...
2.89      1
3.06      1
3.36      1
2.99      1
3.49      1
Name: TCH, Length: 171, 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    12828
2.0     3632
Name: TCH, dtype: int64
```

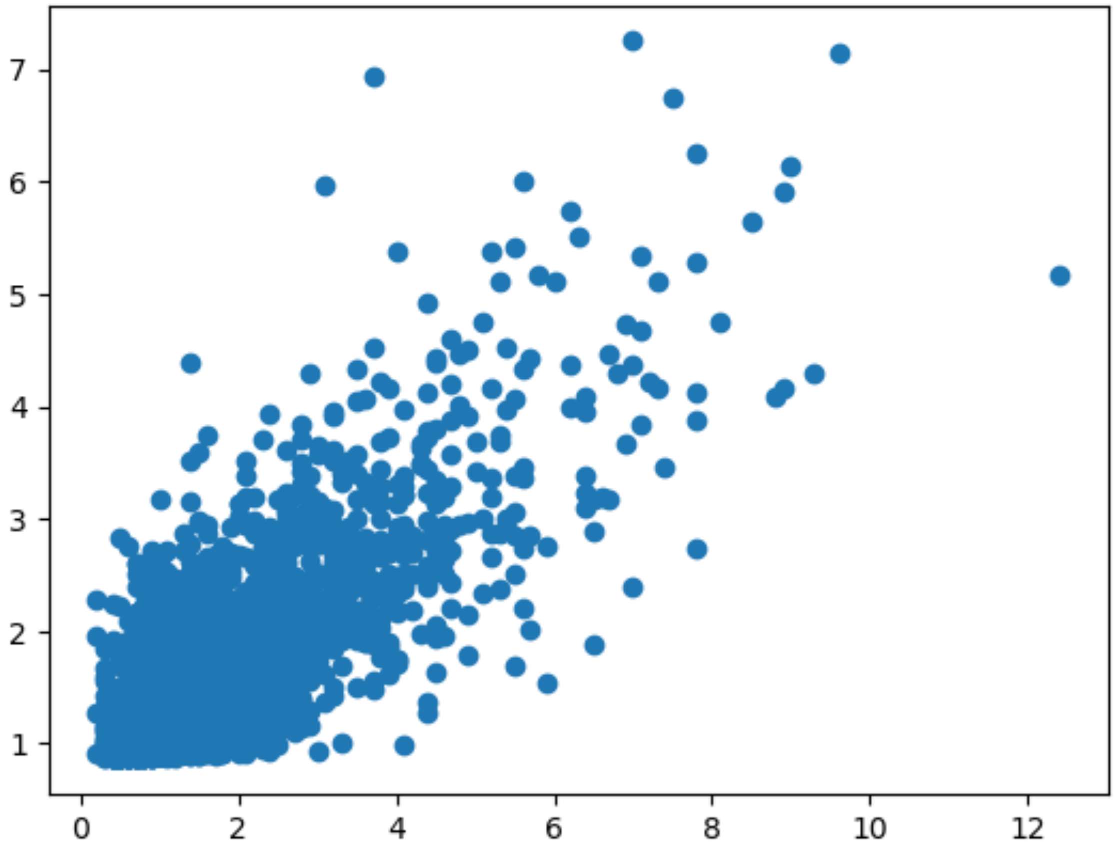
Lasso

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 0x7fccb159c430>



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

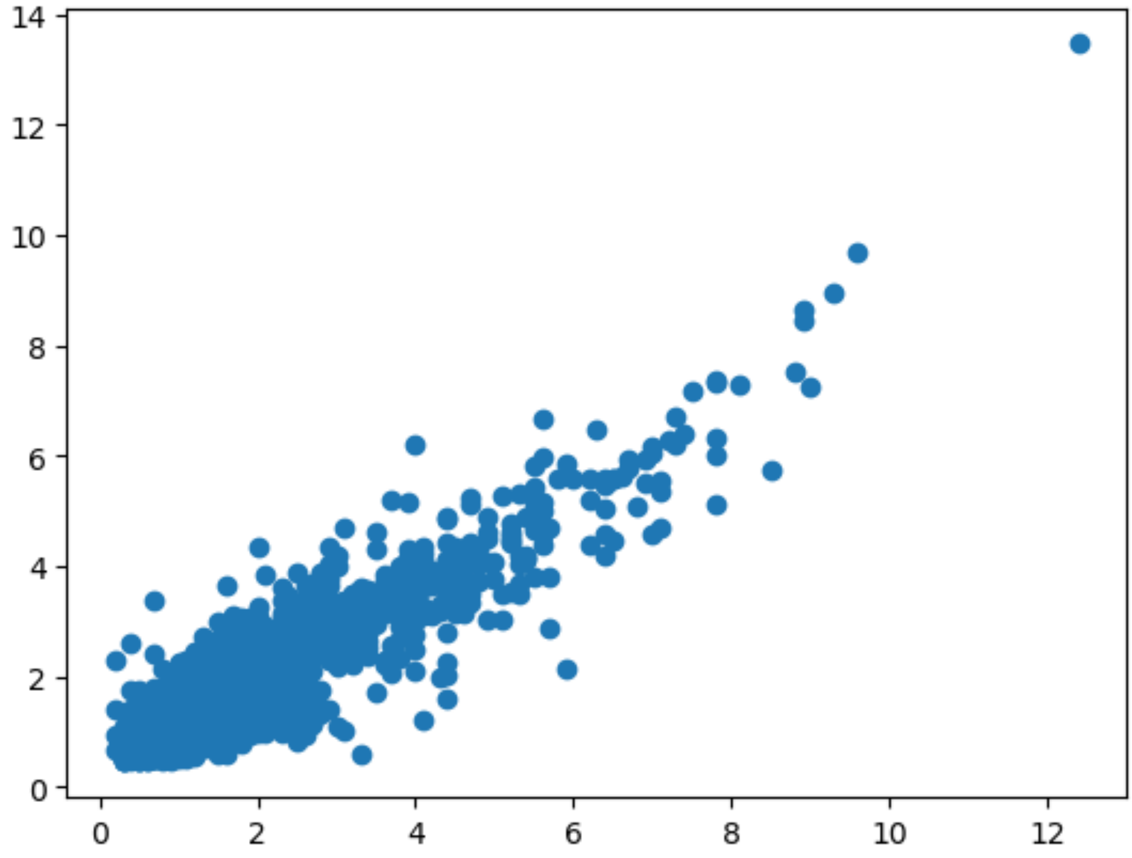
Ridge

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 0x7fcca30222c0>



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

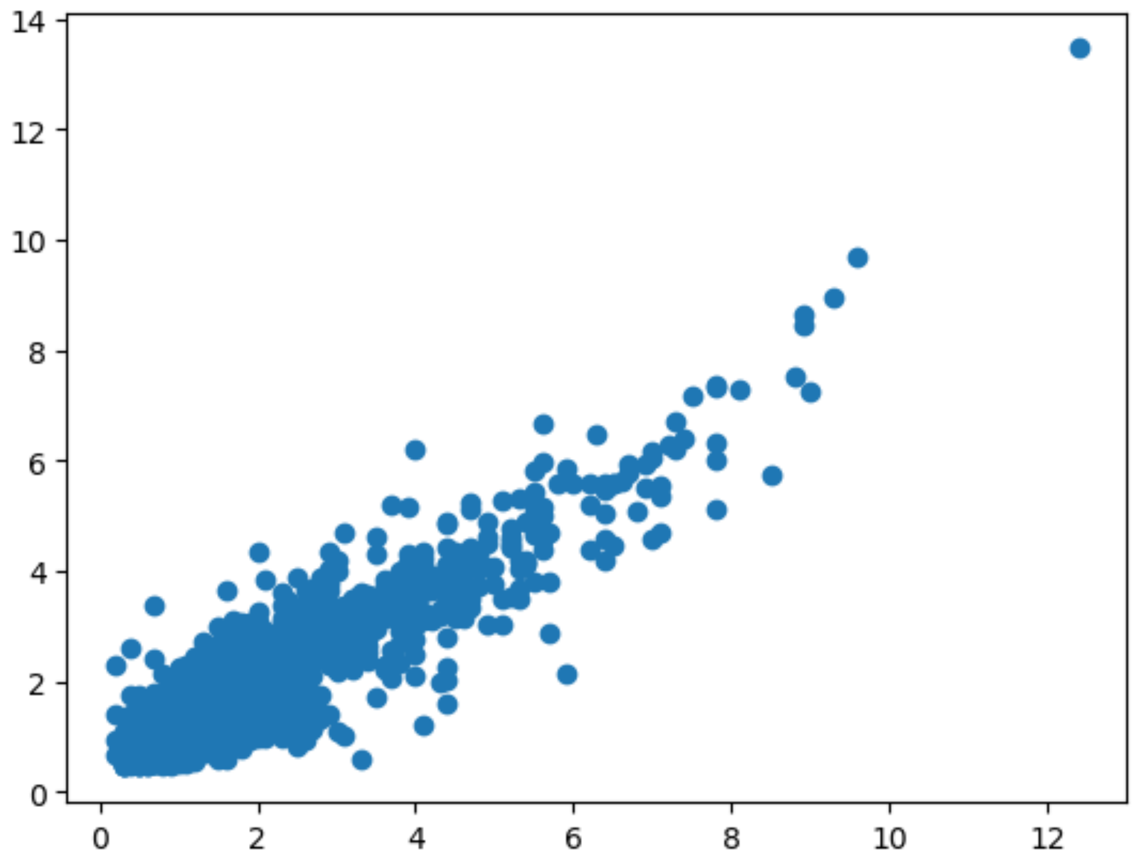
ElasticNet

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 0x7fcca2fdb850>
```



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

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

```
0.8279312677210368
```

Out[24]:
```
0.8141050323866895
```

Logistic

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

Out[25]:
```
Low     12828
High     3632
Name: TCH, dtype: int64
```

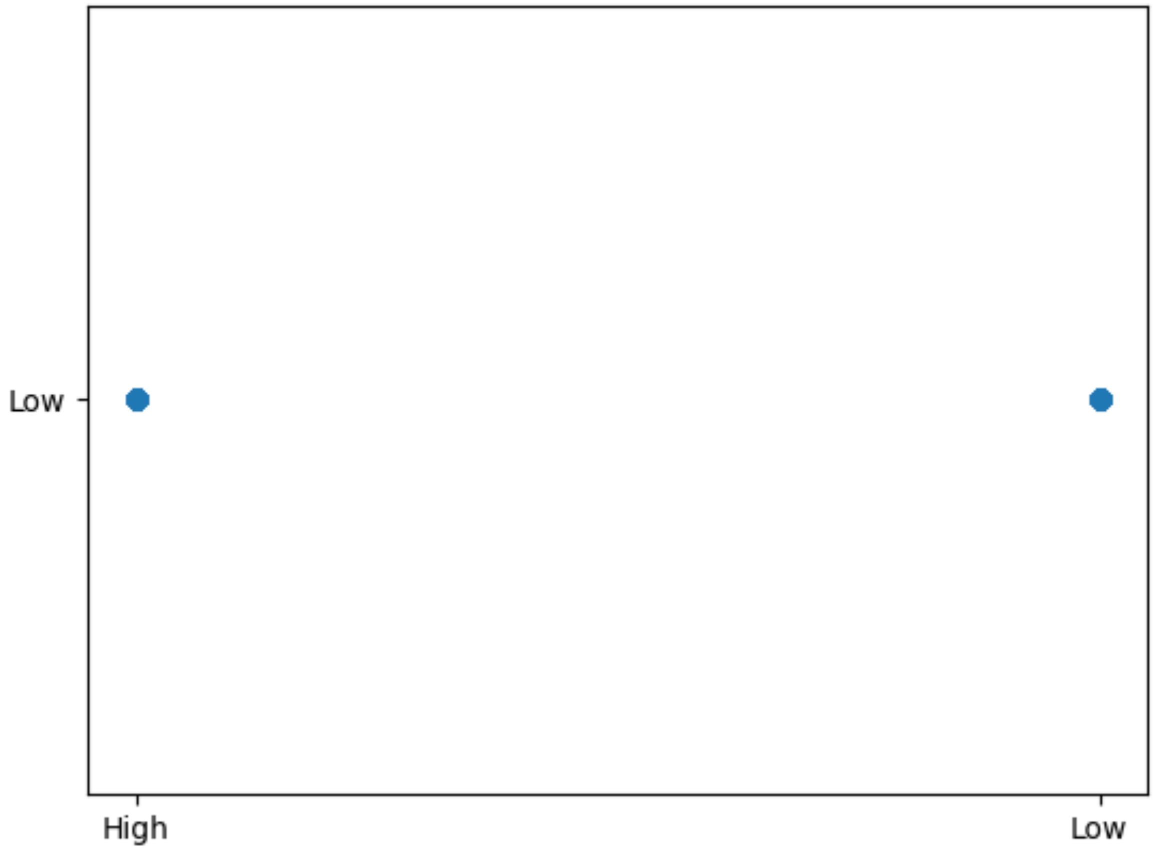
In [26]:
```
x=df1.drop(["TCH"],axis=1)
y=df1["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x7fcca2251600>



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
```

```
 Text(0.5350877192982456, 0.35714285714285715, 'O_3 <= 5.5\ngini =
0.489\nsamples = 276\nvalue = [246, 182]\nclass = Yes'),
 Text(0.5175438596491229, 0.21428571428571427, 'NMHC <= 0.205\ngin
i = 0.427\nsamples = 92\nvalue = [47, 105]\nclass = No'),
 Text(0.5087719298245614, 0.07142857142857142, 'gini = 0.499\nsamp
les = 41\nvalue = [33, 30]\nclass = Yes'),
 Text(0.5263157894736842, 0.07142857142857142, 'gini = 0.265\nsamp
les = 51\nvalue = [14, 75]\nclass = No'),
 Text(0.5526315789473685, 0.21428571428571427, 'PM10 <= 17.5\ngini
= 0.402\nsamples = 184\nvalue = [199, 77]\nclass = Yes'),
 Text(0.543859649122807, 0.07142857142857142, 'gini = 0.064\nsampl
es = 19\nvalue = [29, 1]\nclass = Yes'),
 Text(0.5614035087719298, 0.07142857142857142, 'gini = 0.427\nsamp
les = 165\nvalue = [170, 76]\nclass = Yes'),
 Text(0.6052631578947368, 0.35714285714285715, 'NO <= 82.5\ngini =
0.439\nsamples = 167\nvalue = [89, 184]\nclass = No'),
 Text(0.5877192982456141, 0.21428571428571427, 'NO_2 <= 67.5\ngini
= 0.426\nsamples = 153\nvalue = [76, 171]\nclass = No'),
 Text(0.5789473684210527, 0.07142857142857142, 'gini = 0.113\nsamp
les = 49\nvalue = [5, 78]\nclass = No'),
```

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

```
Linear: 0.8279633125878545
Lasso: 0.5823338875996914
Ridge: 0.8279312677210368
ElasticNet: 0.7138887521604254
Logistic: 0.7806804374240583
Random Forest: 0.8900364520048603
```

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	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	T
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	N
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	[illegible]
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	[illegible]
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	N
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	N
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	N
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	N
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	N
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	N

210720 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210720 entries, 0 to 210719
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210720 non-null  object 
 1   BEN      51511 non-null   float64
 2   CO       87097 non-null   float64
 3   EBE      51482 non-null   float64
 4   NMHC     30736 non-null   float64
 5   NO       209871 non-null  float64
 6   NO_2     209872 non-null  float64
 7   O_3      122339 non-null  float64
 8   PM10     104838 non-null  float64
 9   PM25     52164 non-null   float64
 10  SO_2     87333 non-null   float64
 11  TCH      30736 non-null   float64
 12  TOL      51373 non-null   float64
 13  station  210720 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.5+ MB
```

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

Out[42]:

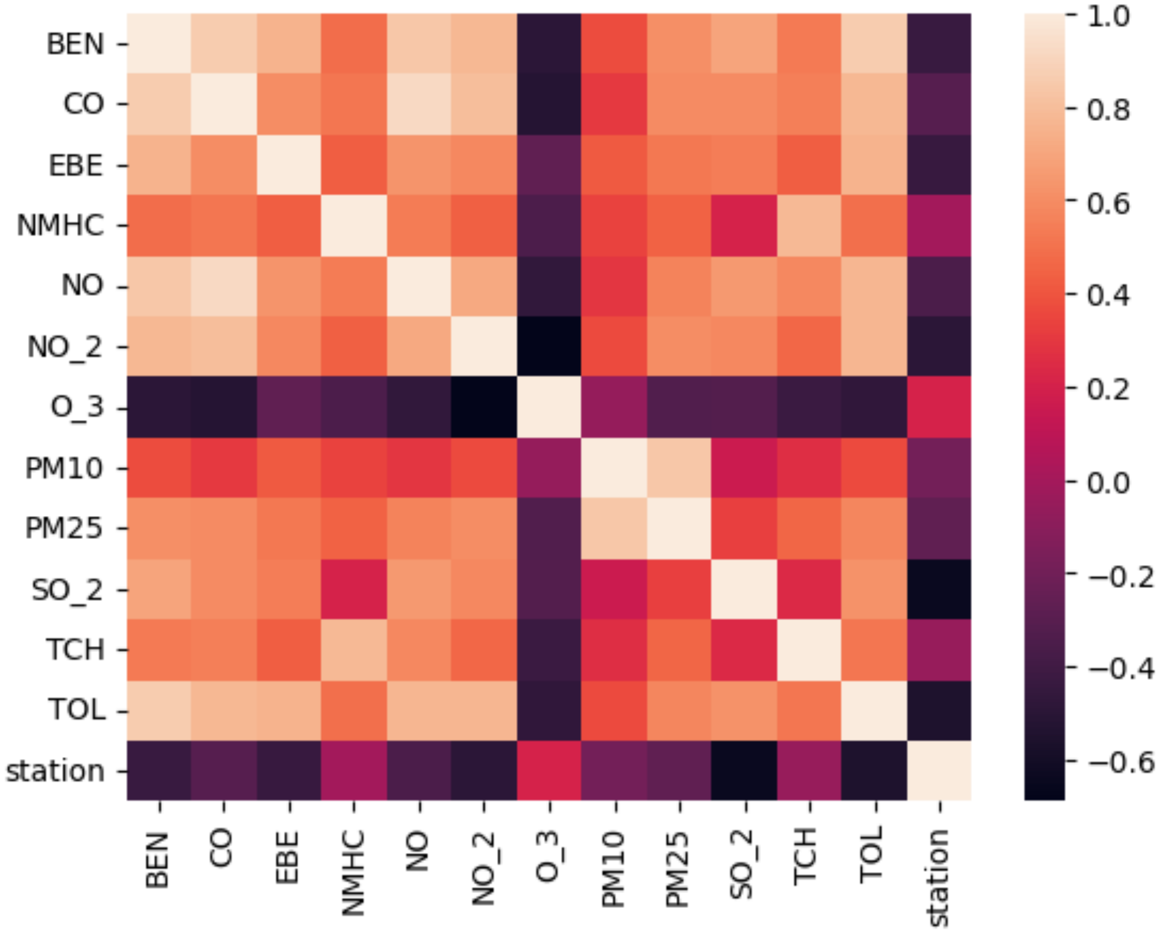
	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TO
6	2012-09-01 01:00:00	0.4	0.2	0.8	0.24	1.0	7.0	57.0	11.0	7.0	2.0	1.33	0.
30	2012-09-01 02:00:00	0.4	0.2	0.7	0.24	1.0	5.0	55.0	5.0	5.0	2.0	1.33	0.
54	2012-09-01 03:00:00	0.4	0.2	0.7	0.24	1.0	4.0	56.0	6.0	4.0	2.0	1.33	0.
78	2012-09-01 04:00:00	0.3	0.2	0.7	0.25	1.0	5.0	54.0	6.0	5.0	2.0	1.34	0.
102	2012-09-01 05:00:00	0.4	0.2	0.7	0.24	1.0	3.0	53.0	8.0	5.0	2.0	1.33	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	.
210654	2012-02-29 22:00:00	0.6	0.3	0.5	0.09	1.0	35.0	57.0	25.0	21.0	3.0	1.12	2.
210673	2012-02-29 23:00:00	2.0	0.4	2.4	0.21	16.0	79.0	20.0	37.0	25.0	12.0	1.33	6.
210678	2012-02-29 23:00:00	0.7	0.3	0.6	0.09	1.0	27.0	63.0	22.0	18.0	3.0	1.11	1.
210697	2012-03-01 00:00:00	1.5	0.4	1.7	0.21	16.0	79.0	17.0	28.0	21.0	11.0	1.34	4.
210702	2012-03-01 00:00:00	0.6	0.3	0.5	0.09	1.0	23.0	61.0	18.0	16.0	3.0	1.11	1.

10916 rows × 14 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)
```

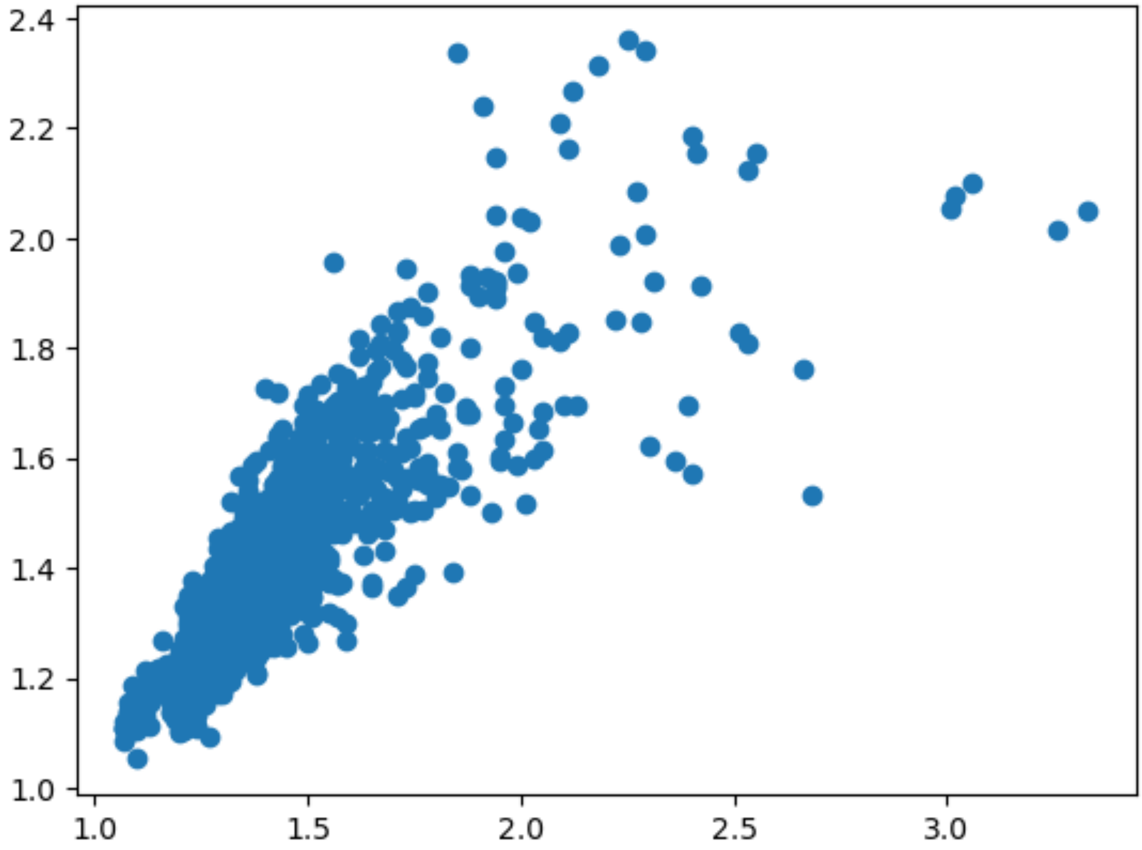
Linear

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 0x7fccc36d3010>



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

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

Out[49]:
```
1.30    737
1.31    676
1.32    644
1.33    552
1.29    529
       ...
2.39      1
2.20      1
2.72      1
3.11      1
2.70      1
Name: TCH, Length: 167, 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    8772
2.0    2144
Name: TCH, dtype: int64
```

In [ ]:

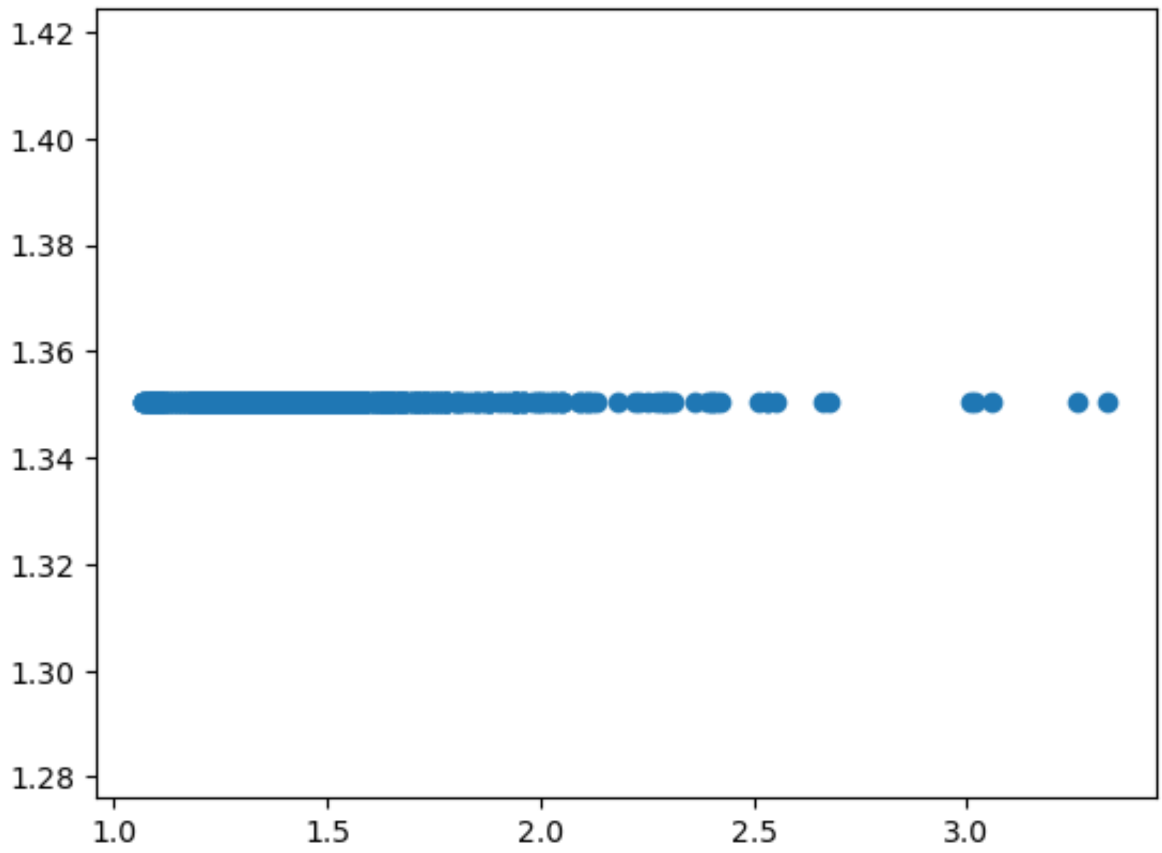
Lasso

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 0x7fccc3747370>



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

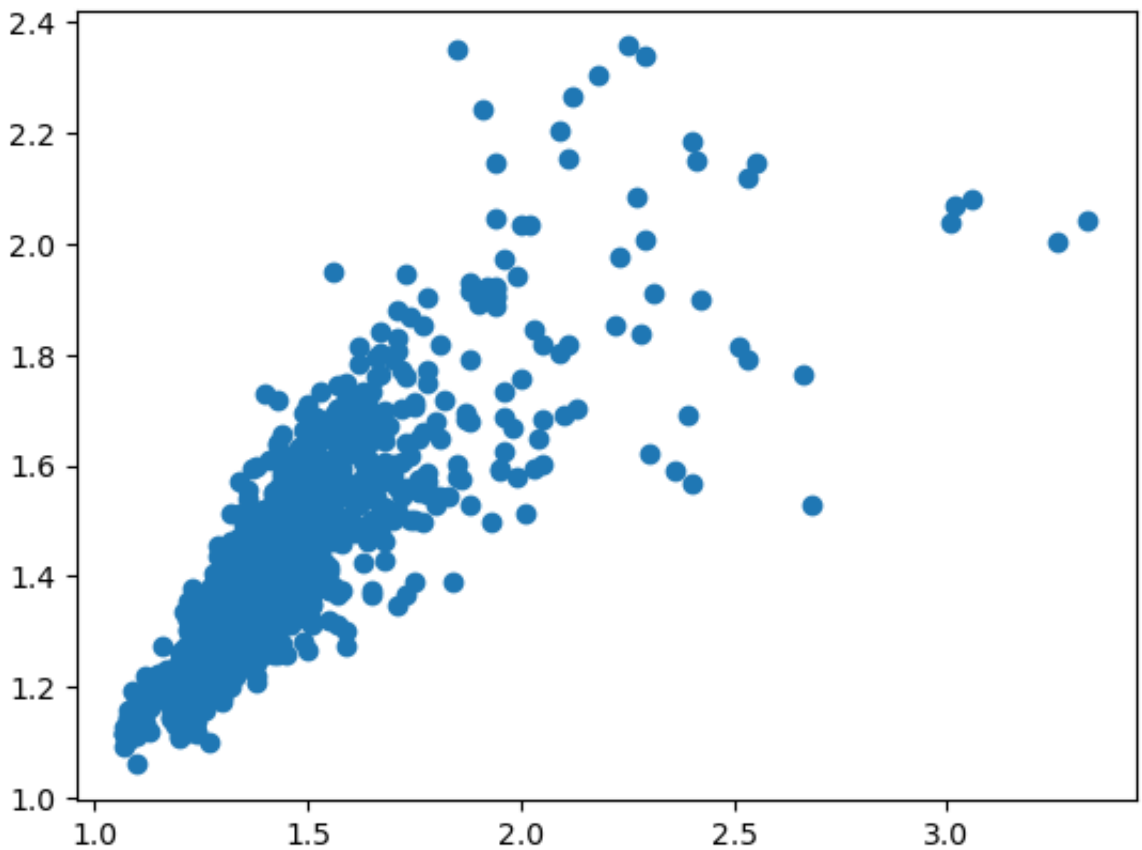
Ridge

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 0x7fccc37477f0>



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

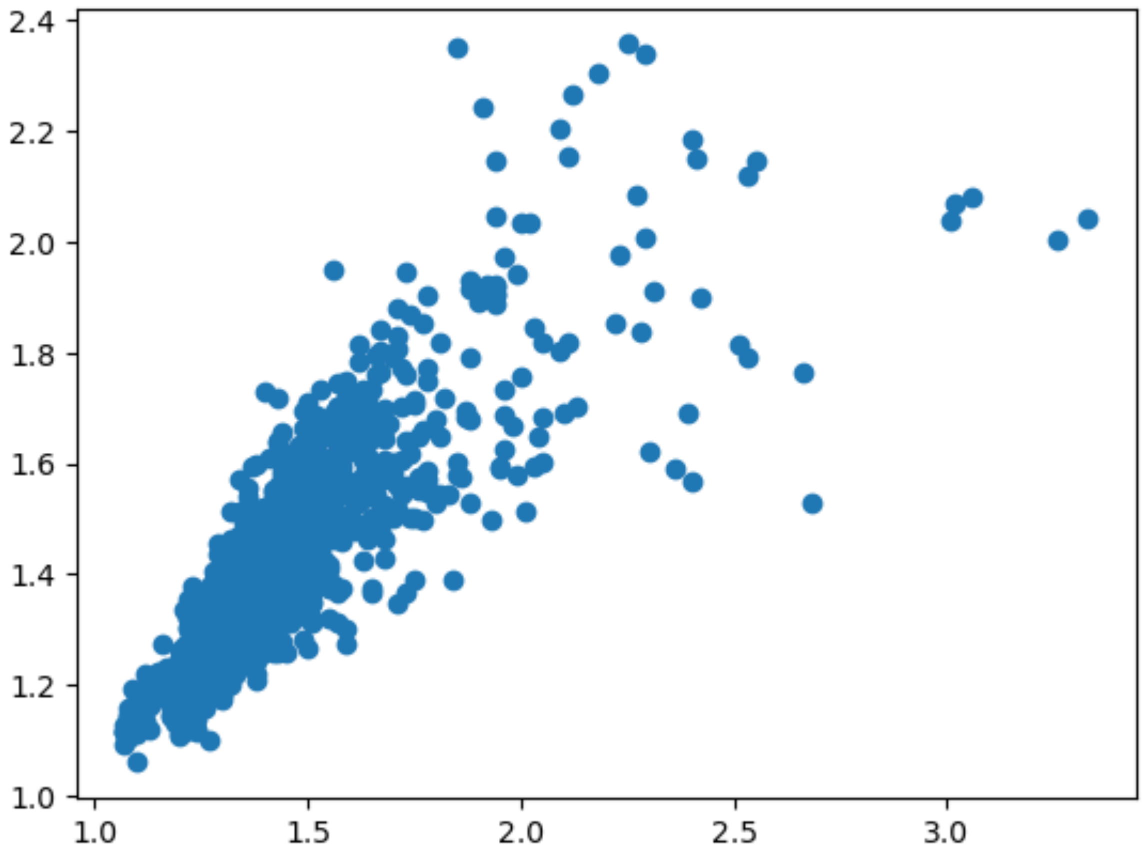
ElasticNet

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 0x7fccb2023b50>



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

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

```
0.7027769553239515
```

Out[60]: 0.6820735566707625

Logistic

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

Out[61]:
```
Low     8772
High    2144
Name: TCH, dtype: int64
```

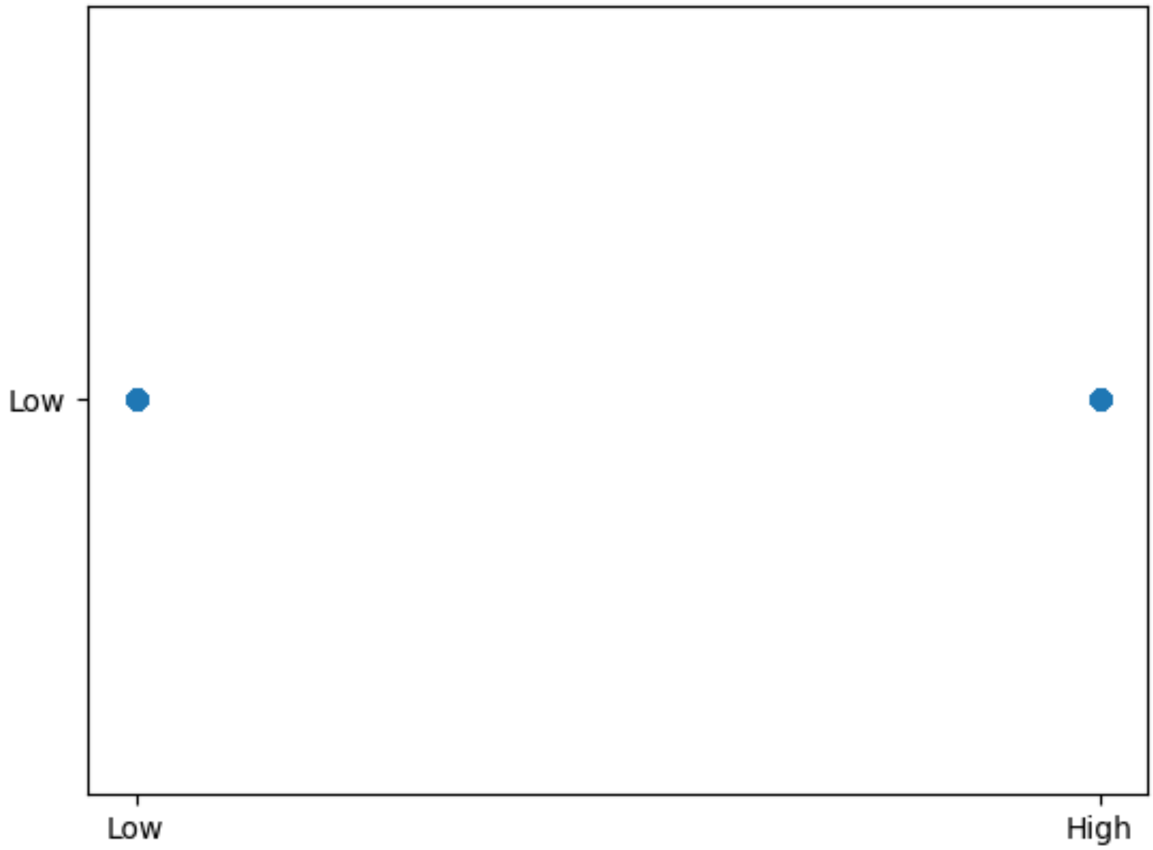
In [62]:
```
x=df3.drop(["TCH"],axis=1)
y=df3["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x7fccc36a6020>



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(0.5377604166666666, 0.9285714285714286, 'NMHC <= 0.275\ngini
= 0.315\nsamples = 4831\nvalue = [6147, 1494]\nclass = Yes'),
 Text(0.2838541666666667, 0.7857142857142857, 'CO <= 0.35\ngini =
0.132\nsamples = 4074\nvalue = [6002, 460]\nclass = Yes'),
 Text(0.15104166666666666, 0.6428571428571429, 'PM25 <= 11.5\ngini
= 0.084\nsamples = 3548\nvalue = [5413, 248]\nclass = Yes'),
 Text(0.08333333333333333, 0.5, 'NO_2 <= 13.5\ngini = 0.046\nsampl
es = 2726\nvalue = [4256, 103]\nclass = Yes'),
 Text(0.041666666666666664, 0.35714285714285715, 'PM10 <= 10.5\ngi
ni = 0.01\nsamples = 1437\nvalue = [2290, 12]\nclass = Yes'),
 Text(0.020833333333333332, 0.21428571428571427, 'NMHC <= 0.265\ng
ini = 0.003\nsamples = 806\nvalue = [1291, 2]\nclass = Yes'),
 Text(0.010416666666666666, 0.07142857142857142, 'gini = 0.0\nsamp
les = 787\nvalue = [1265, 0]\nclass = Yes'),
 Text(0.03125, 0.07142857142857142, 'gini = 0.133\nsamples = 19\nv
alue = [26, 2]\nclass = Yes'),
 Text(0.0625, 0.21428571428571427, 'EBE <= 2.2\ngini = 0.02\nsampl
es = 631\nvalue = [999, 10]\nclass = Yes'),
 Text(0.052083333333333336, 0.07142857142857142, 'gini = 0.012\nsa
```

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

```
Linear: 0.7025995094680152
Lasso: -7.83493655487355e-06
Ridge: 0.7027769553239515
ElasticNet: 0.38196670444638303
Logistic: 0.8003053435114503
Random Forest: 0.9339087106113775
```

Best model is Random Forest