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

In [2]:
```python
df=pd.read_csv("/Users/bhoomish/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_2013.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3	28079008
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0	28079011
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN	28079056
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN	28079057
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN	28079058
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN	28079059
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN	28079060

209880 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      50462 non-null   float64
 2   CO       87018 non-null   float64
 3   EBE      50463 non-null   float64
 4   NMHC     25935 non-null   float64
 5   NO       209108 non-null  float64
 6   NO_2     209108 non-null  float64
 7   O_3      121858 non-null  float64
 8   PM10     104339 non-null  float64
 9   PM25     51980 non-null   float64
 10  SO_2     86970 non-null   float64
 11  TCH      25935 non-null   float64
 12  TOL      50317 non-null   float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

Out[4]:

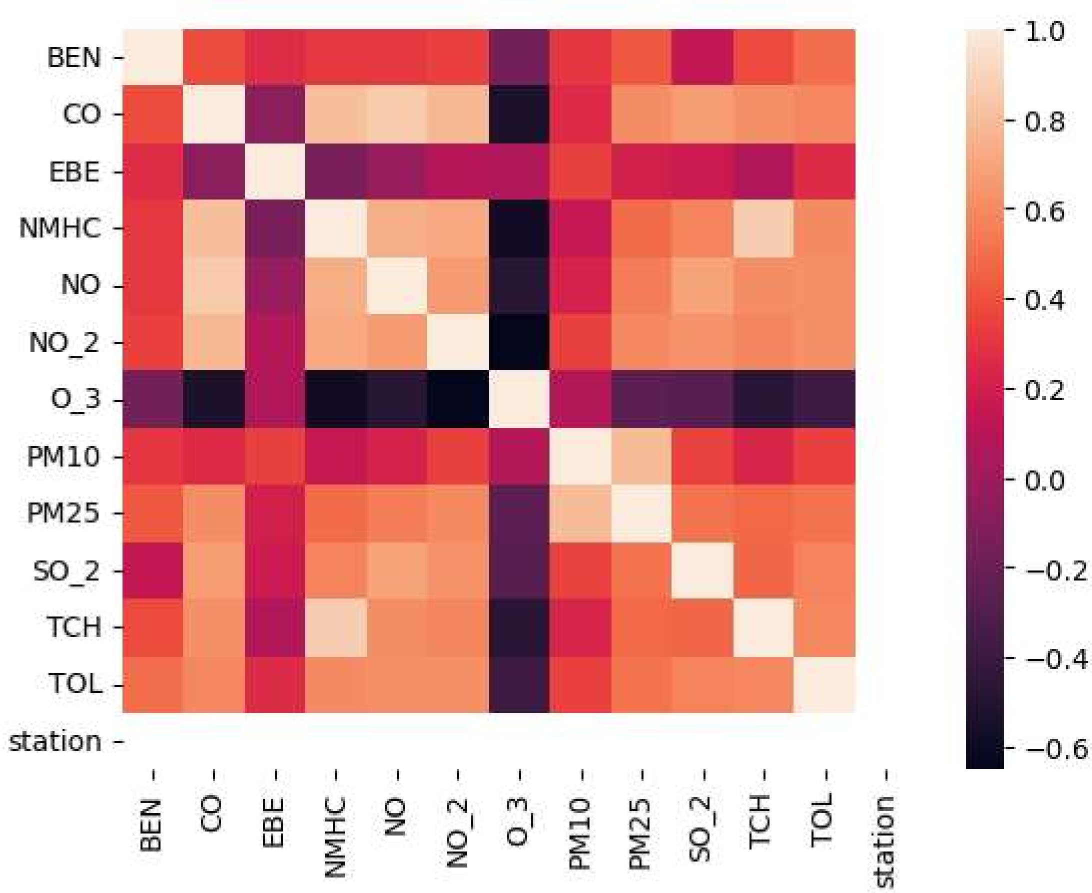
	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
17286	2013-08-01 01:00:00	0.4	0.2	0.8	0.28	1.0	24.0	79.0	35.0	8.0	3.0	1.49	1.3	28079024
17310	2013-08-01 02:00:00	0.5	0.2	0.9	0.28	1.0	16.0	93.0	60.0	18.0	3.0	1.61	4.0	28079024
17334	2013-08-01 03:00:00	0.5	0.2	1.1	0.29	1.0	14.0	90.0	38.0	12.0	3.0	1.71	2.8	28079024
17358	2013-08-01 04:00:00	0.6	0.2	1.2	0.26	1.0	12.0	84.0	30.0	8.0	3.0	1.44	2.8	28079024
17382	2013-08-01 05:00:00	0.3	0.2	0.8	0.25	1.0	15.0	72.0	25.0	7.0	3.0	1.40	1.7	28079024
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209622	2013-02-28 14:00:00	1.1	0.3	0.3	0.27	3.0	17.0	64.0	5.0	5.0	2.0	1.41	0.9	28079024
209646	2013-02-28 15:00:00	1.3	0.4	0.3	0.27	2.0	16.0	66.0	6.0	5.0	1.0	1.40	0.9	28079024
209670	2013-02-28 16:00:00	1.1	0.3	0.3	0.27	1.0	17.0	65.0	5.0	4.0	1.0	1.40	0.7	28079024
209694	2013-02-28 17:00:00	1.0	0.3	0.4	0.27	1.0	18.0	64.0	5.0	5.0	1.0	1.39	0.7	28079024
209718	2013-02-28 18:00:00	1.0	0.3	0.4	0.27	1.0	22.0	62.0	6.0	6.0	1.0	1.39	0.7	28079024

7315 rows × 14 columns

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

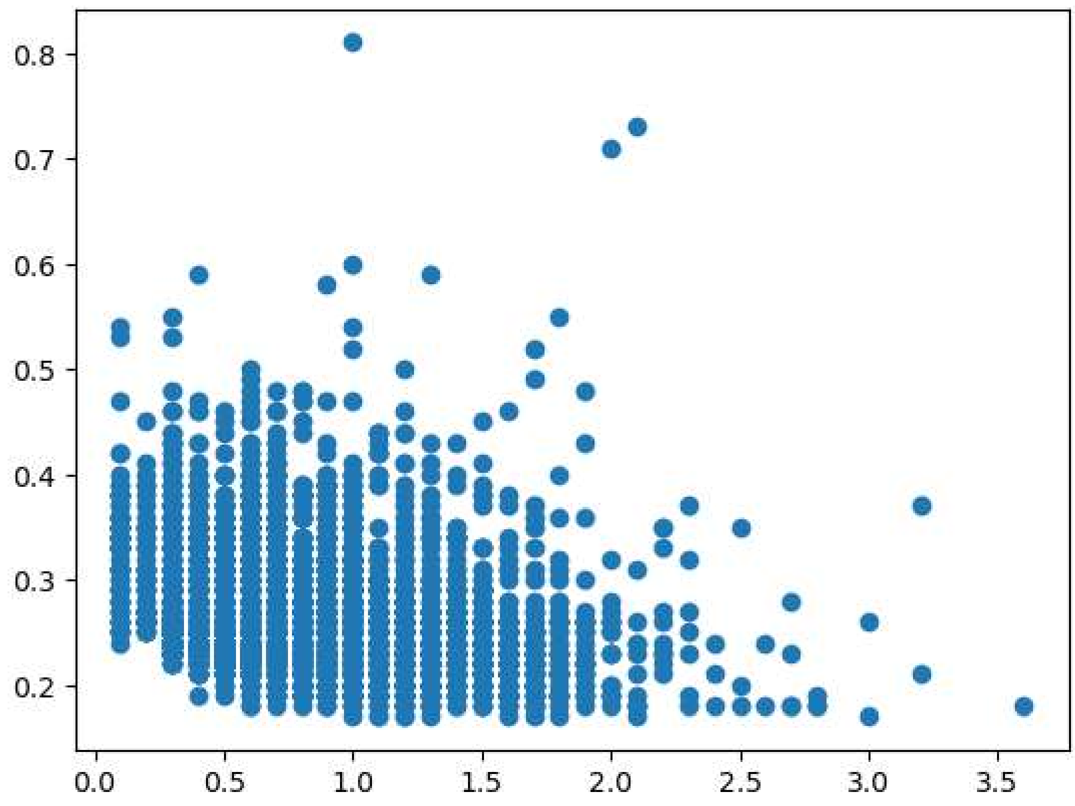
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <Axes: >



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

Out[7]: [<matplotlib.lines.Line2D at 0x7f9d822f5150>]



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

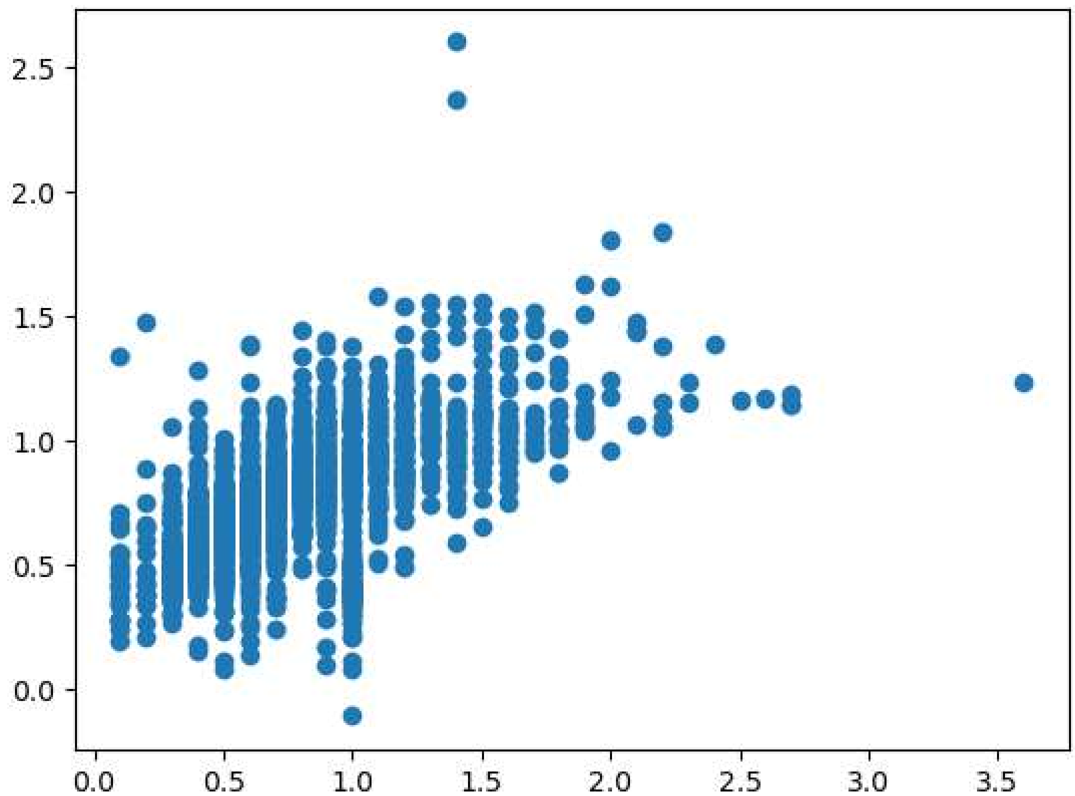
In [10]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[11]:
```
<matplotlib.collections.PathCollection at 0x7f9d92470940>
```



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

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

Out[13]:
```
1.32    888
1.33    843
1.34    729
1.31    719
1.35    556
       ...
2.39      1
2.22      1
2.29      1
2.38      1
2.80      1
Name: TCH, Length: 114, 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    5718
2.0    1597
Name: TCH, dtype: int64
```

Lasso

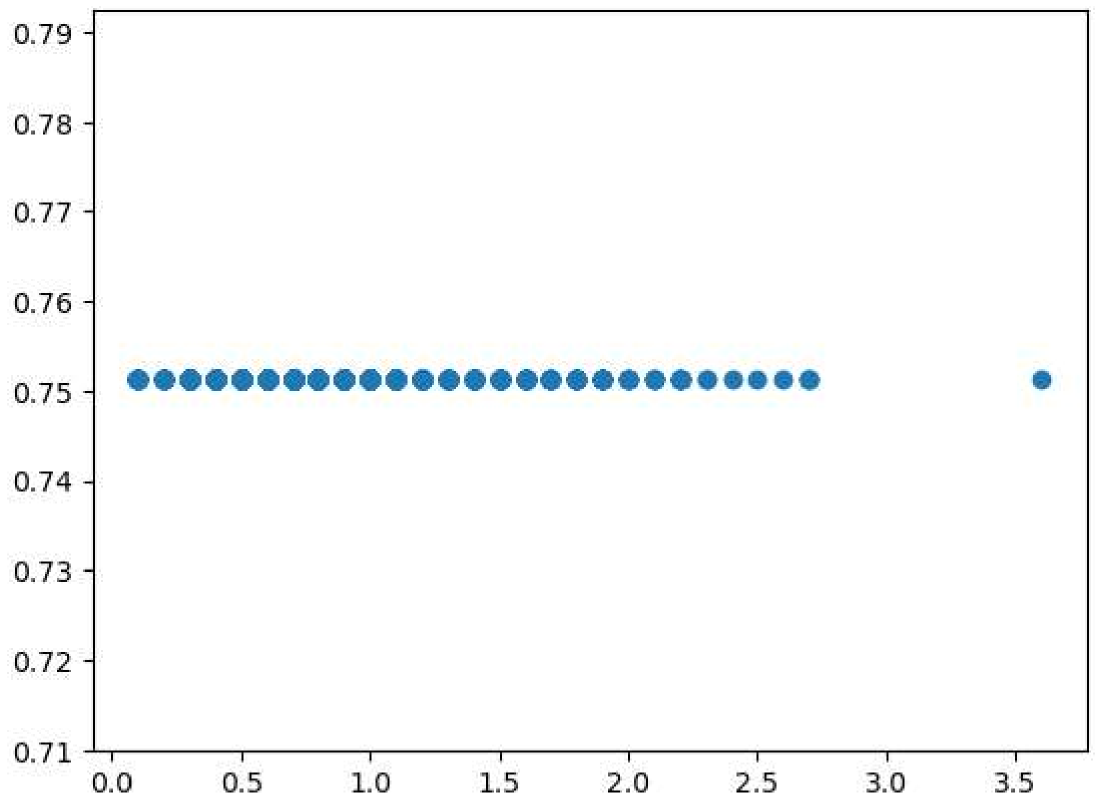
In [15]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[16]: <matplotlib.collections.PathCollection at 0x7f9d924dab30>



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

Ridge

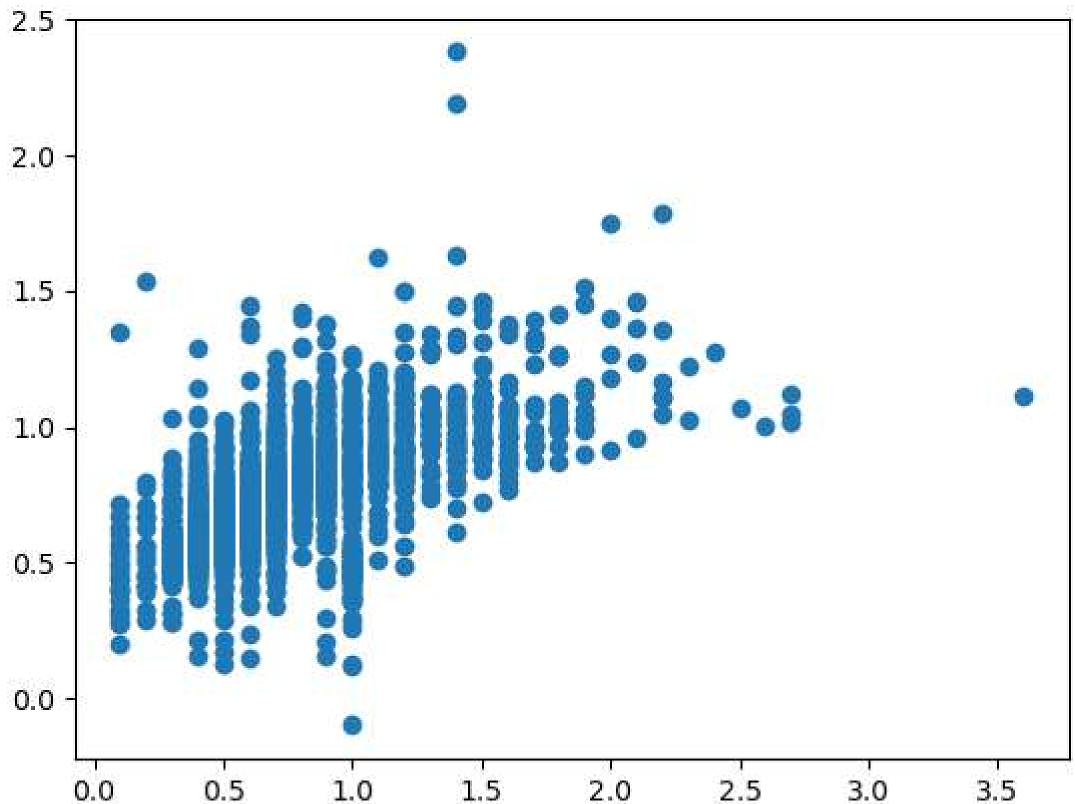
In [18]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[18]: Ridge(alpha=1)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[19]: <matplotlib.collections.PathCollection at 0x7f9d9256b550>



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

ElasticNet

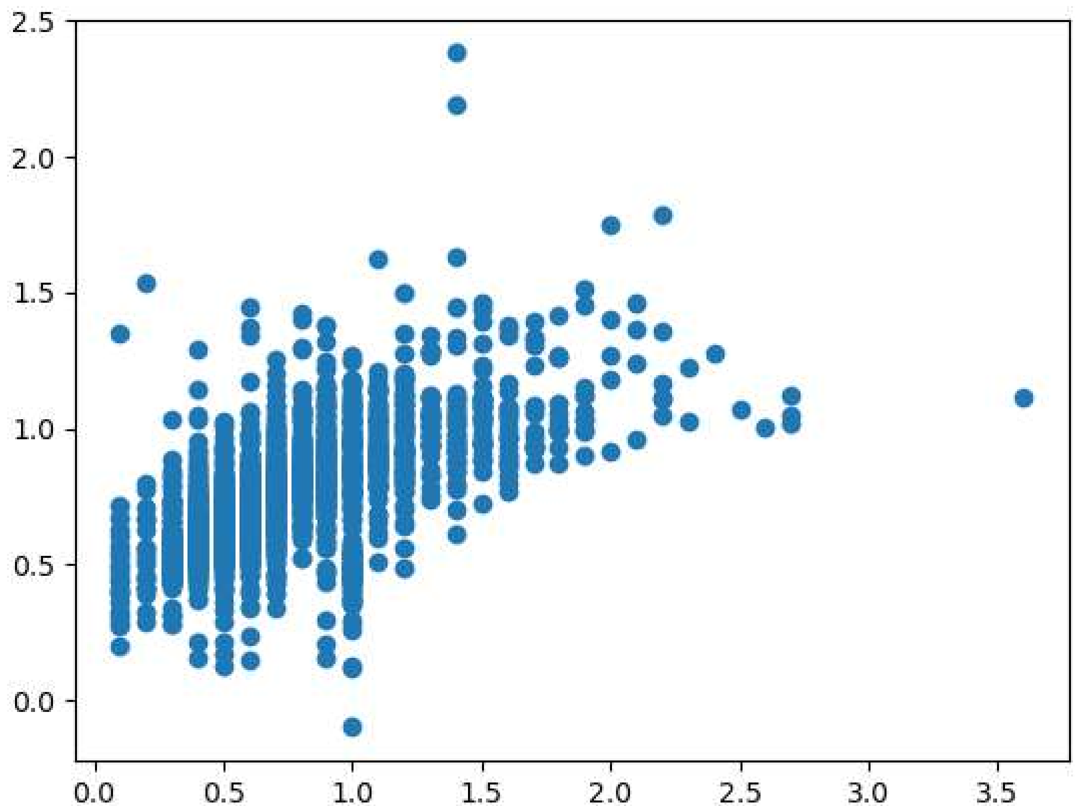
In [21]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[22]: <matplotlib.collections.PathCollection at 0x7f9d9240b4f0>



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

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

```
0.3923674796819595
```

Out[24]:
```
0.3927671300897322
```

Logistic

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

Out[25]:
```
Low     5718
High    1597
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)
```

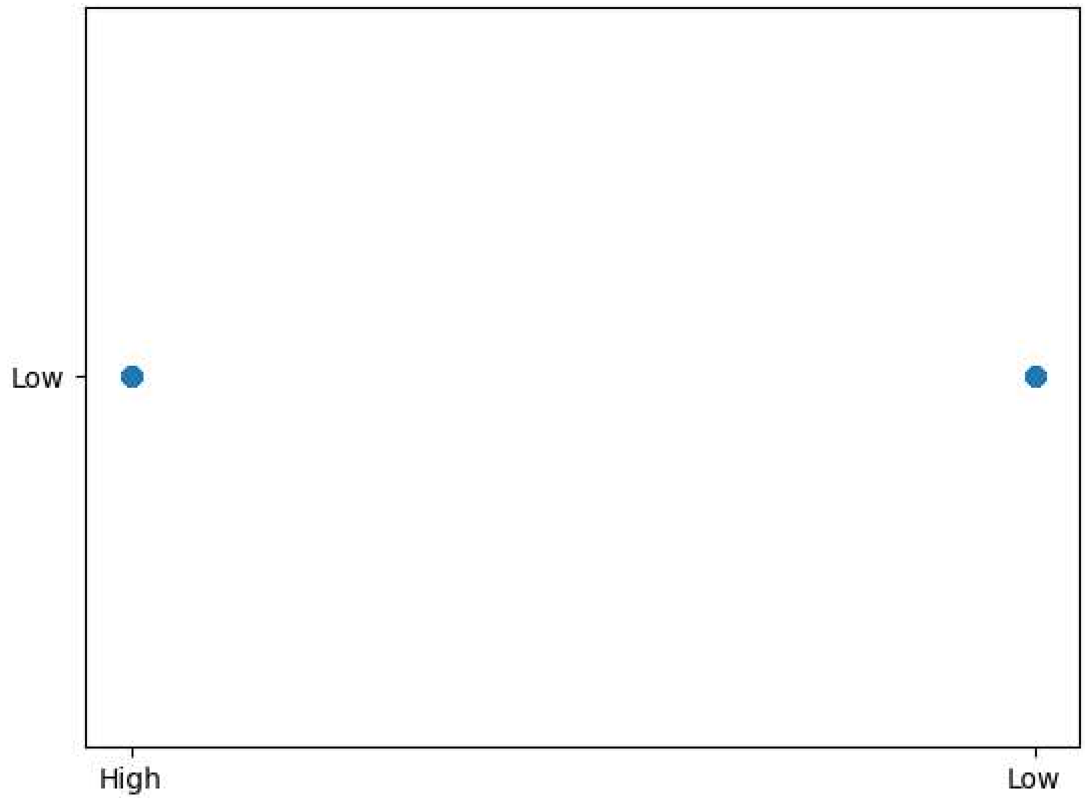
In [27]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[27]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[28]: <matplotlib.collections.PathCollection at 0x7f9d30433c70>



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()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[35]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 4, 5, 6],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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_names=['Yes',"No"],filled=True)
```

Out[38]:

```
[Text(0.5375, 0.9285714285714286, 'NMHC <= 0.275\ngini = 0.328\nsamples = 3238\nvalue = [4060, 1060]\ncla
ss = Yes'),
 Text(0.3390625, 0.7857142857142857, 'TOL <= 1.05\ngini = 0.09\nsamples = 2599\nvalue = [3927, 194]\nclas
s = Yes'),
 Text(0.196875, 0.6428571428571429, 'O_3 <= 45.5\ngini = 0.031\nsamples = 1960\nvalue = [3090, 50]\nclass
= Yes'),
 Text(0.1, 0.5, 'PM10 <= 17.5\ngini = 0.174\nsamples = 215\nvalue = [309, 33]\nclass = Yes'),
 Text(0.05, 0.35714285714285715, 'O_3 <= 20.5\ngini = 0.123\nsamples = 191\nvalue = [283, 20]\nclass = Ye
s'),
 Text(0.025, 0.21428571428571427, 'SO_2 <= 1.5\ngini = 0.335\nsamples = 30\nvalue = [37, 10]\nclass = Ye
s'),
 Text(0.0125, 0.07142857142857142, 'gini = 0.497\nsamples = 9\nvalue = [7, 6]\nclass = Yes'),
 Text(0.0375, 0.07142857142857142, 'gini = 0.208\nsamples = 21\nvalue = [30, 4]\nclass = Yes'),
 Text(0.075, 0.21428571428571427, 'PM10 <= 5.5\ngini = 0.075\nsamples = 161\nvalue = [246, 10]\nclass = Y
es'),
 Text(0.0625, 0.07142857142857142, 'gini = 0.0\nsamples = 46\nvalue = [75, 0]\nclass = Yes'),
 Text(0.0875, 0.07142857142857142, 'gini = 0.104\nsamples = 115\nvalue = [171, 10]\nclass = Yes'),
 Text(0.15, 0.35714285714285715, 'O_3 <= 37.5\ngini = 0.444\nsamples = 24\nvalue = [26, 13]\nclass = Ye
s'),
```

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

```
Linear: 0.40767907025541705
Lasso: -0.00029951584872067727
Ridge: 0.3923674796819595
ElasticNet: 0.10978052070818145
Logistic: 0.7890660592255125
Random Forest: 0.946484375
```

Best Model is Random Forest

In [40]:
```
df2=pd.read_csv("/Users/bob/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_2014.csv")
df2
```

Out[40]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN	28079004
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	28079008
2	2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1	28079011
3	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210019	2014-09-01 00:00:00	NaN	0.5	NaN	NaN	20.0	84.0	29.0	NaN	NaN	NaN	NaN	NaN	28079056
210020	2014-09-01 00:00:00	NaN	0.3	NaN	NaN	1.0	22.0	NaN	15.0	NaN	6.0	NaN	NaN	28079057
210021	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	13.0	70.0	NaN	NaN	NaN	NaN	NaN	28079058
210022	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	38.0	42.0	NaN	NaN	NaN	NaN	NaN	28079059
210023	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	26.0	65.0	11.0	NaN	NaN	NaN	NaN	28079060

210024 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210024 entries, 0 to 210023
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210024 non-null  object 
 1   BEN      46703 non-null   float64
 2   CO       87023 non-null   float64
 3   EBE      46722 non-null   float64
 4   NMHC     25021 non-null   float64
 5   NO       209154 non-null  float64
 6   NO_2     209154 non-null  float64
 7   O_3      121681 non-null  float64
 8   PM10     104311 non-null  float64
 9   PM25     51954 non-null   float64
 10  SO_2     87141 non-null   float64
 11  TCH      25021 non-null   float64
 12  TOL      46570 non-null   float64
 13  station  210024 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

Out[42]:

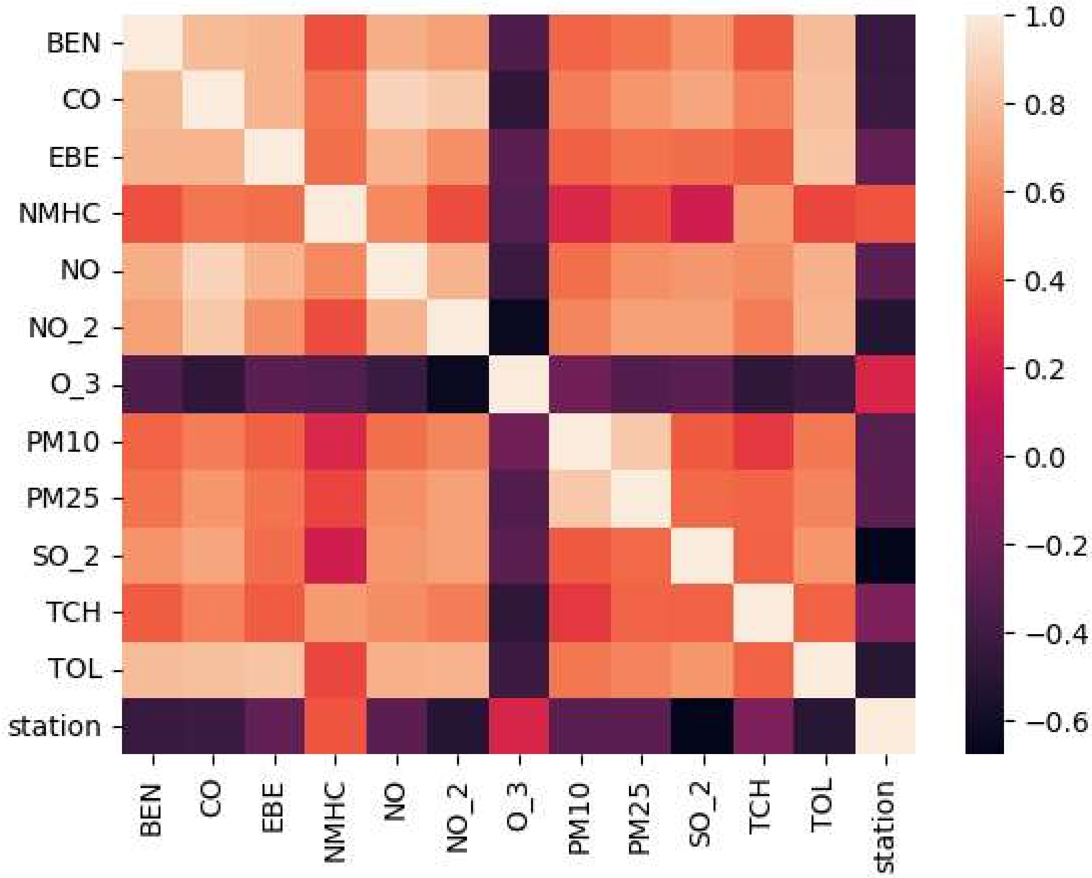
	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	28079008
6	2014-06-01 01:00:00	0.1	0.2	0.1	0.23	1.0	5.0	80.0	4.0	3.0	2.0	1.21	0.1	28079024
25	2014-06-01 02:00:00	0.2	0.2	0.1	0.11	4.0	21.0	63.0	9.0	6.0	5.0	1.36	0.8	28079008
30	2014-06-01 02:00:00	0.2	0.2	0.1	0.23	1.0	4.0	88.0	7.0	5.0	2.0	1.21	0.1	28079024
49	2014-06-01 03:00:00	0.1	0.2	0.1	0.11	4.0	18.0	66.0	9.0	7.0	6.0	1.36	0.9	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209958	2014-08-31 22:00:00	0.2	0.2	0.1	0.22	1.0	28.0	96.0	61.0	15.0	3.0	1.28	0.1	28079024
209977	2014-08-31 23:00:00	1.1	0.7	0.7	0.19	36.0	118.0	23.0	60.0	25.0	9.0	1.27	6.5	28079008
209982	2014-08-31 23:00:00	0.2	0.2	0.1	0.21	1.0	17.0	90.0	28.0	14.0	3.0	1.27	0.2	28079024
210001	2014-09-01 00:00:00	0.6	0.4	0.4	0.12	6.0	63.0	41.0	26.0	15.0	8.0	1.19	4.1	28079008
210006	2014-09-01 00:00:00	0.2	0.2	0.1	0.23	1.0	30.0	69.0	18.0	13.0	3.0	1.30	0.1	28079024

13946 rows × 14 columns

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

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

Out[44]:
```
<Axes: >
```



In [45]:
```python
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

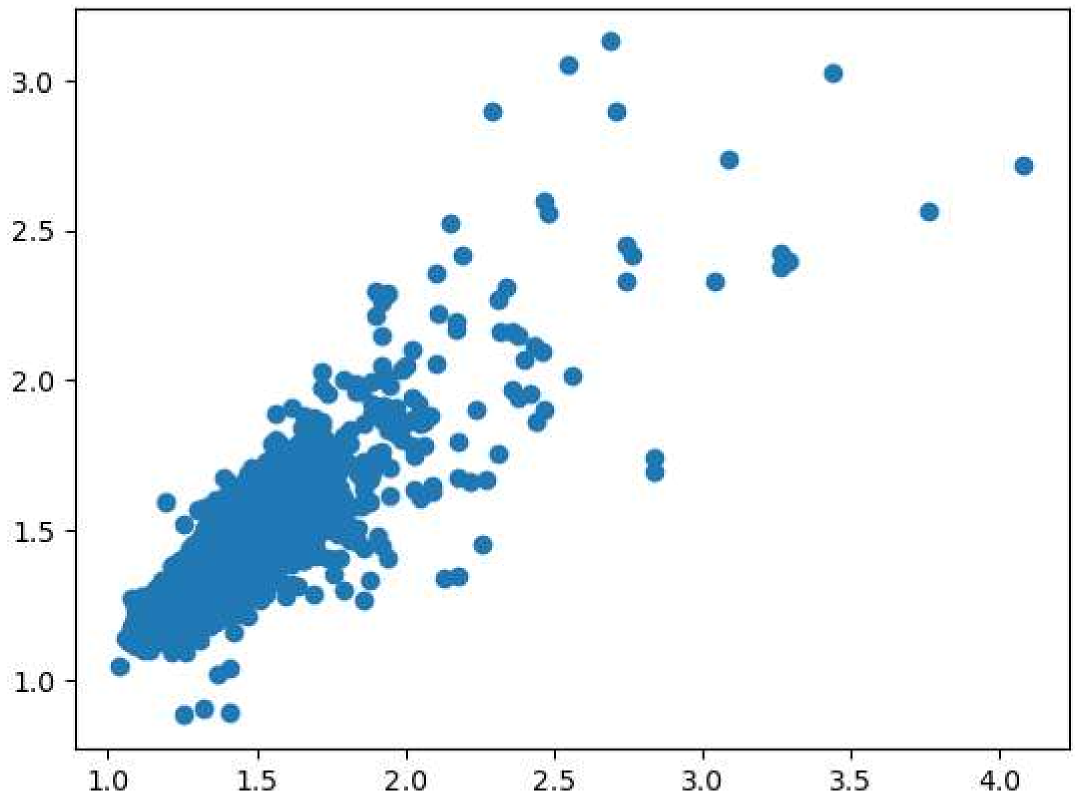
In [46]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[46]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[47]: <matplotlib.collections.PathCollection at 0x7f9d82aff730>



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

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

Out[49]:
```
1.37    601
1.36    598
1.34    529
1.35    528
1.38    515
       ...
4.39      1
4.08      1
3.42      1
2.98      1
2.69      1
Name: TCH, Length: 184, 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    9997
2.0    3949
Name: TCH, dtype: int64
```

In [ ]:
```

```

Lasso

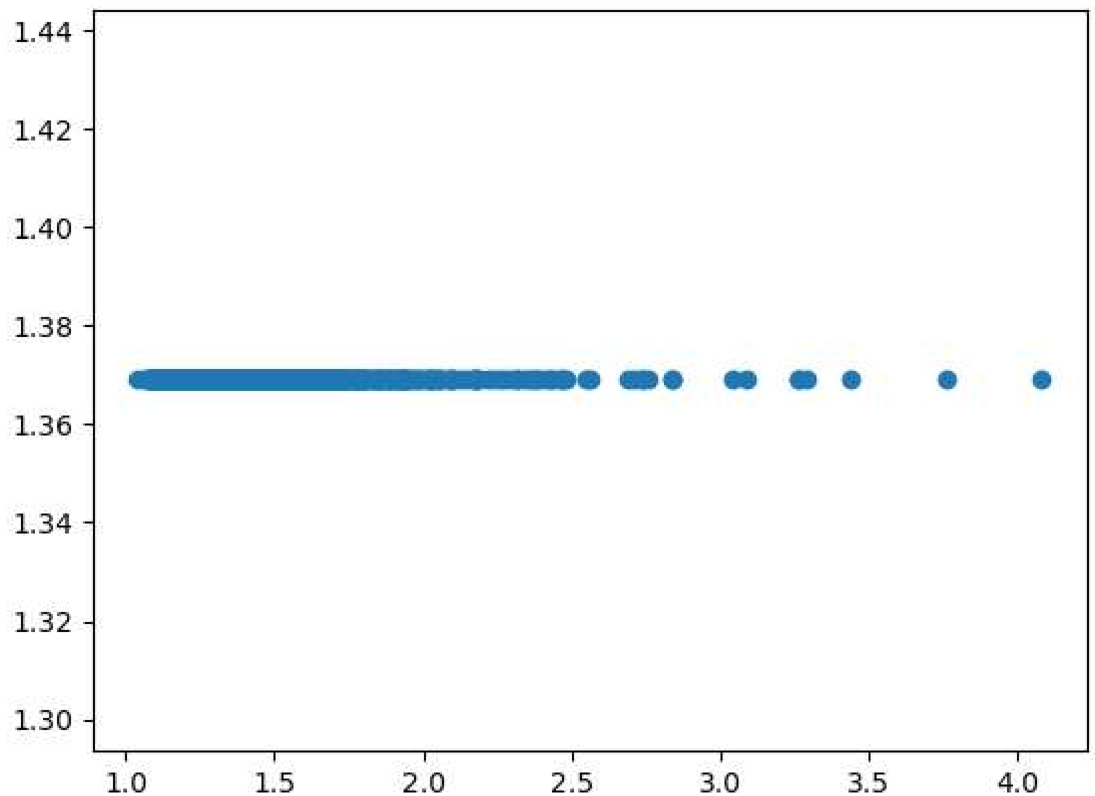
In [51]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[52]: <matplotlib.collections.PathCollection at 0x7f9d8297ceb0>



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

Ridge

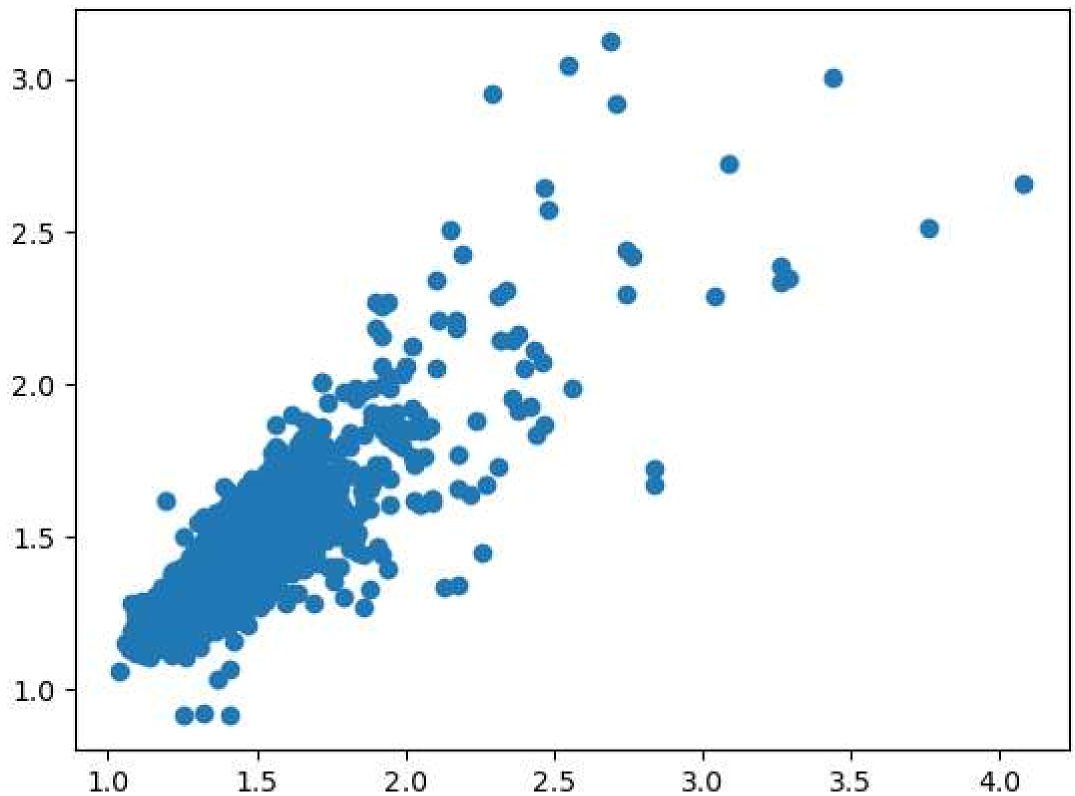
In [54]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[54]: Ridge(alpha=1)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[55]: <matplotlib.collections.PathCollection at 0x7f9d829cc9d0>



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

ElasticNet

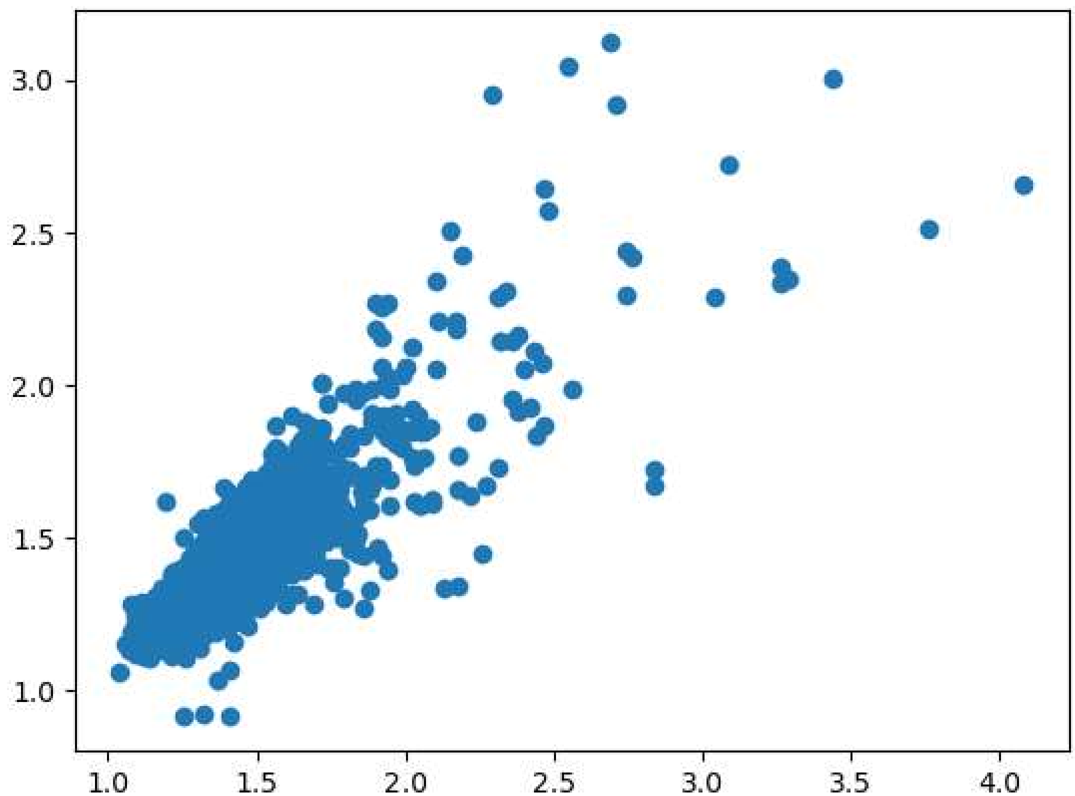
In [57]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[58]: <matplotlib.collections.PathCollection at 0x7f9d82b49030>



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

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

```
0.7091386251261051
```

Out[60]: 0.7047426795616607

Logistic

In [61]:

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

Out[61]:

```
Low     9997
High    3949
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()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Out[64]: <matplotlib.collections.PathCollection at 0x7f9d3046cb80>

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()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[71]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 4, 5, 6],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

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

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

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes',"No"],filled=True)
```

Out[74]:
```
[Text(0.5191326530612245, 0.9285714285714286, 'PM25 <= 12.5\ngini = 0.402\nsamples = 6171\nvalue = [7045,
2717]\nclass = Yes'),
 Text(0.2755102040816326, 0.7857142857142857, 'NMHC <= 0.285\ngini = 0.289\nsamples = 4464\nvalue = [583
6, 1237]\nclass = Yes'),
 Text(0.14540816326530612, 0.6428571428571429, 'SO_2 <= 8.5\ngini = 0.233\nsamples = 4206\nvalue = [5776,
898]\nclass = Yes'),
 Text(0.08163265306122448, 0.5, 'NO_2 <= 25.5\ngini = 0.211\nsamples = 4094\nvalue = [5722, 779]\nclass =
Yes'),
 Text(0.04081632653061224, 0.35714285714285715, 'NO_2 <= 11.5\ngini = 0.084\nsamples = 2607\nvalue = [395
3, 182]\nclass = Yes'),
 Text(0.02040816326530612, 0.21428571428571427, 'NO_2 <= 4.5\ngini = 0.023\nsamples = 1603\nvalue = [253
6, 30]\nclass = Yes'),
 Text(0.01020408163265306, 0.07142857142857142, 'gini = 0.008\nsamples = 800\nvalue = [1275, 5]\nclass =
Yes'),
 Text(0.030612244897959183, 0.07142857142857142, 'gini = 0.038\nsamples = 803\nvalue = [1261, 25]\nclass
= Yes'),
 Text(0.061224489795918366, 0.21428571428571427, 'NMHC <= 0.135\ngini = 0.175\nsamples = 1004\nvalue = [1
417, 152]\nclass = Yes'),
 Text(0.05102040816326531, 0.07142857142857142, 'gini = 0.05\nsamples = 279\nvalue = [418, 11]\nclass = Y
```

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

```
Linear: 0.7105902856424091
Lasso: -0.0001259044549823951
Ridge: 0.7091386251261051
ElasticNet: 0.43942661468297217
Logistic: 0.7127151051625239
Random Forest: 0.8929522638803524
```

Best model is Random Forest