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,LogisticRegression,Lasso,Ridge,ElasticNet
from sklearn.model_selection import train_test_split
```

In [2]:
```
df=pd.read_csv("/Users/bhoomish/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_2017.csv")
df
```

Out[2]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2017-06-01 01:00:00	NaN	NaN	0.3	NaN	NaN	4.0	38.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	28079004
1	2017-06-01 01:00:00	0.6	NaN	0.3	0.4	0.08	3.0	39.0	NaN	71.0	22.0	9.0	7.0	1.4	2.9	28079008
2	2017-06-01 01:00:00	0.2	NaN	NaN	0.1	NaN	1.0	14.0	NaN	NaN	NaN	NaN	NaN	NaN	0.9	28079011
3	2017-06-01 01:00:00	NaN	NaN	0.2	NaN	NaN	1.0	9.0	NaN	91.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2017-06-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	19.0	NaN	69.0	NaN	NaN	2.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210115	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	27.0	NaN	65.0	NaN	NaN	NaN	NaN	NaN	28079056
210116	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	14.0	NaN	NaN	73.0	NaN	7.0	NaN	NaN	28079057
210117	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	4.0	NaN	83.0	NaN	NaN	NaN	NaN	NaN	28079058
210118	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	11.0	NaN	78.0	NaN	NaN	NaN	NaN	NaN	28079059
210119	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	14.0	NaN	77.0	60.0	NaN	NaN	NaN	NaN	28079060

210120 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210120 entries, 0 to 210119
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210120 non-null  object 
 1   BEN      50201 non-null   float64
 2   CH4      6410 non-null    float64
 3   CO       87001 non-null   float64
 4   EBE      49973 non-null   float64
 5   NMHC     25472 non-null   float64
 6   NO       209065 non-null  float64
 7   NO_2     209065 non-null  float64
 8   NOx      52818 non-null   float64
 9   O_3      121398 non-null  float64
 10  PM10     104141 non-null  float64
 11  PM25     52023 non-null   float64
 12  SO_2     86803 non-null   float64
 13  TCH      25472 non-null   float64
 14  TOL      50117 non-null   float64
 15  station  210120 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 25.6+ MB
```

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

Out[4]:

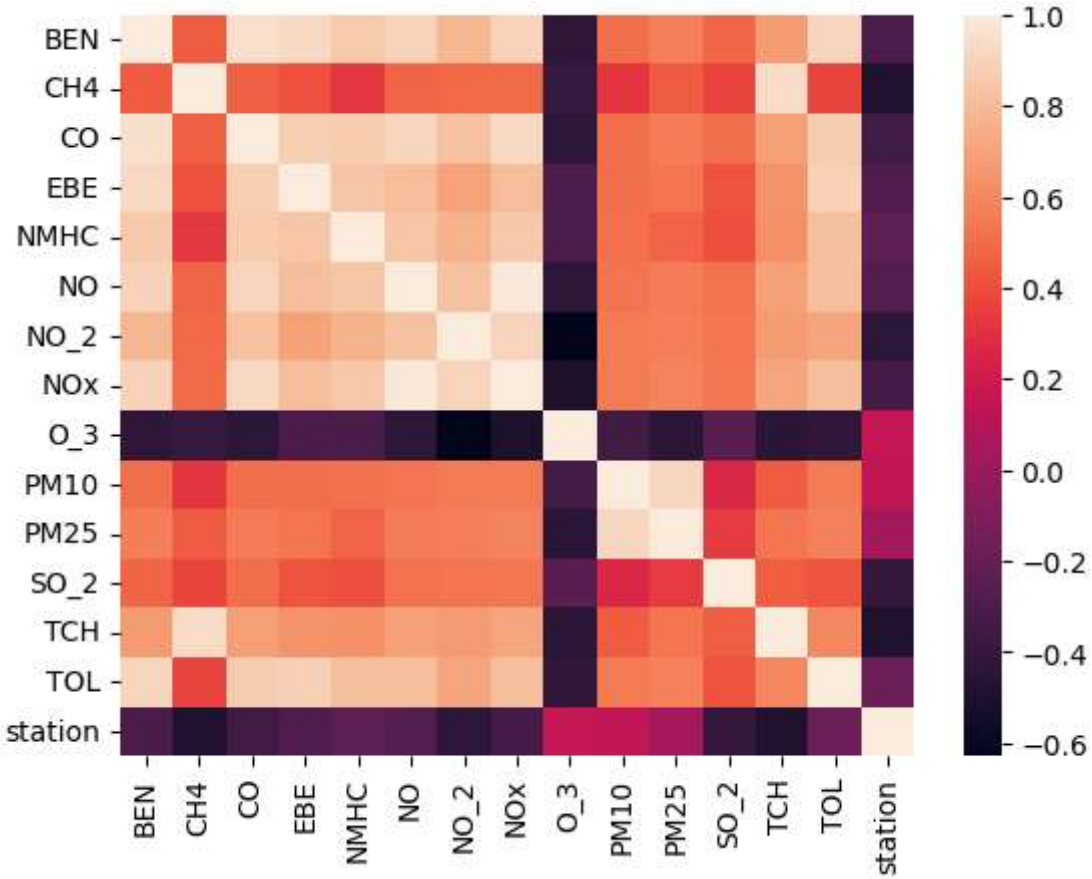
	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	station
87457	2017-10-01 01:00:00	0.6	1.22	0.3	0.4	0.09	4.0	54.0	60.0	43.0	12.0	9.0	13.0	1.31	2.3	28079008
87462	2017-10-01 01:00:00	0.2	1.18	0.2	0.1	0.09	1.0	26.0	28.0	42.0	14.0	6.0	3.0	1.27	1.1	28079024
87481	2017-10-01 02:00:00	0.4	1.22	0.2	0.2	0.06	2.0	32.0	36.0	53.0	14.0	10.0	13.0	1.28	1.3	28079008
87486	2017-10-01 02:00:00	0.2	1.19	0.2	0.1	0.07	1.0	15.0	17.0	51.0	18.0	8.0	3.0	1.26	0.8	28079024
87505	2017-10-01 03:00:00	0.3	1.23	0.2	0.2	0.06	2.0	27.0	29.0	57.0	15.0	10.0	13.0	1.29	1.0	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
158238	2017-12-31 22:00:00	0.3	1.11	0.2	0.1	0.03	1.0	8.0	9.0	73.0	3.0	1.0	3.0	1.14	0.2	28079024
158257	2017-12-31 23:00:00	0.6	1.38	0.3	0.1	0.03	6.0	42.0	51.0	47.0	7.0	4.0	3.0	1.41	0.9	28079008
158262	2017-12-31 23:00:00	0.3	1.11	0.2	0.1	0.03	1.0	6.0	8.0	72.0	6.0	3.0	3.0	1.14	0.2	28079024
158281	2018-01-01 00:00:00	0.5	1.38	0.2	0.1	0.02	2.0	20.0	23.0	69.0	4.0	2.0	3.0	1.39	0.6	28079008
158286	2018-01-01 00:00:00	0.3	1.11	0.2	0.1	0.03	1.0	1.0	3.0	83.0	8.0	5.0	3.0	1.14	0.2	28079024

4127 rows × 16 columns

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

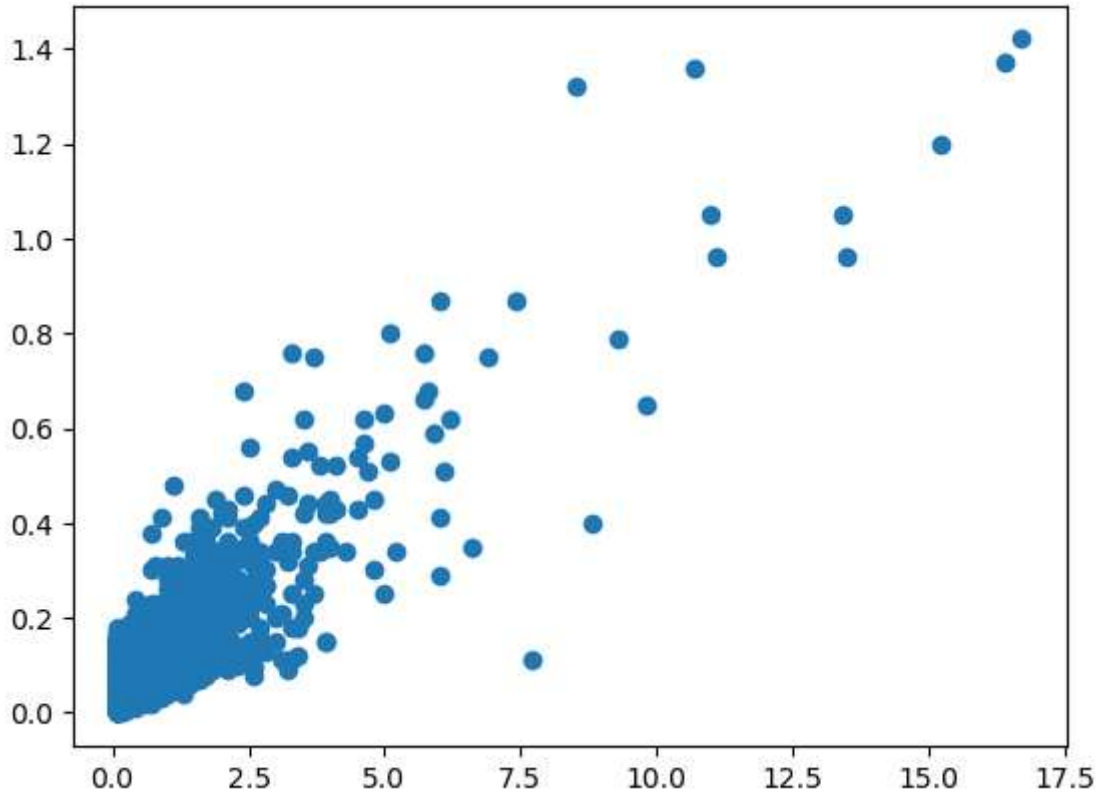
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <Axes: >



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

Out[7]:
```
[<matplotlib.lines.Line2D at 0x7fe3e8c2f910>]
```



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

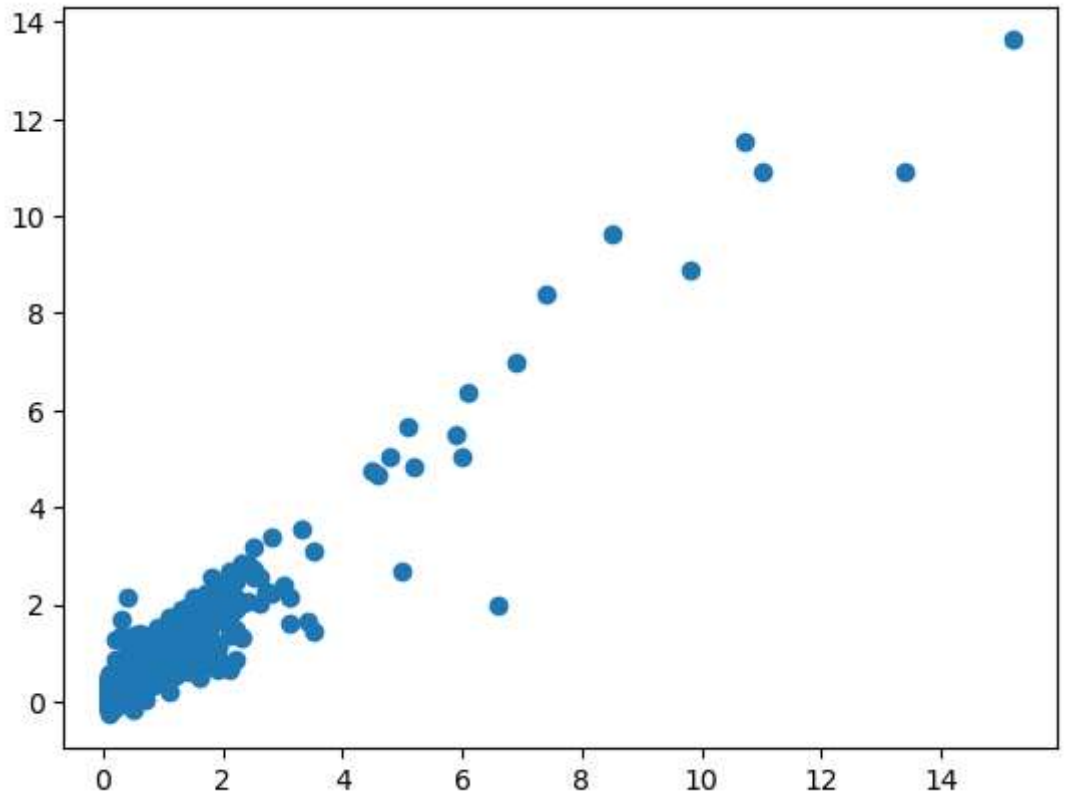
In [10]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

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 0x7fe409097e20>
```



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

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

Out[13]:
```
1.24    124
1.36    118
1.26    112
1.25    110
1.33    107
       ...
3.17      1
3.22      1
3.02      1
2.75      1
2.71      1
Name: TCH, Length: 164, 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    2428
2.0    1699
Name: TCH, dtype: int64
```

Lasso

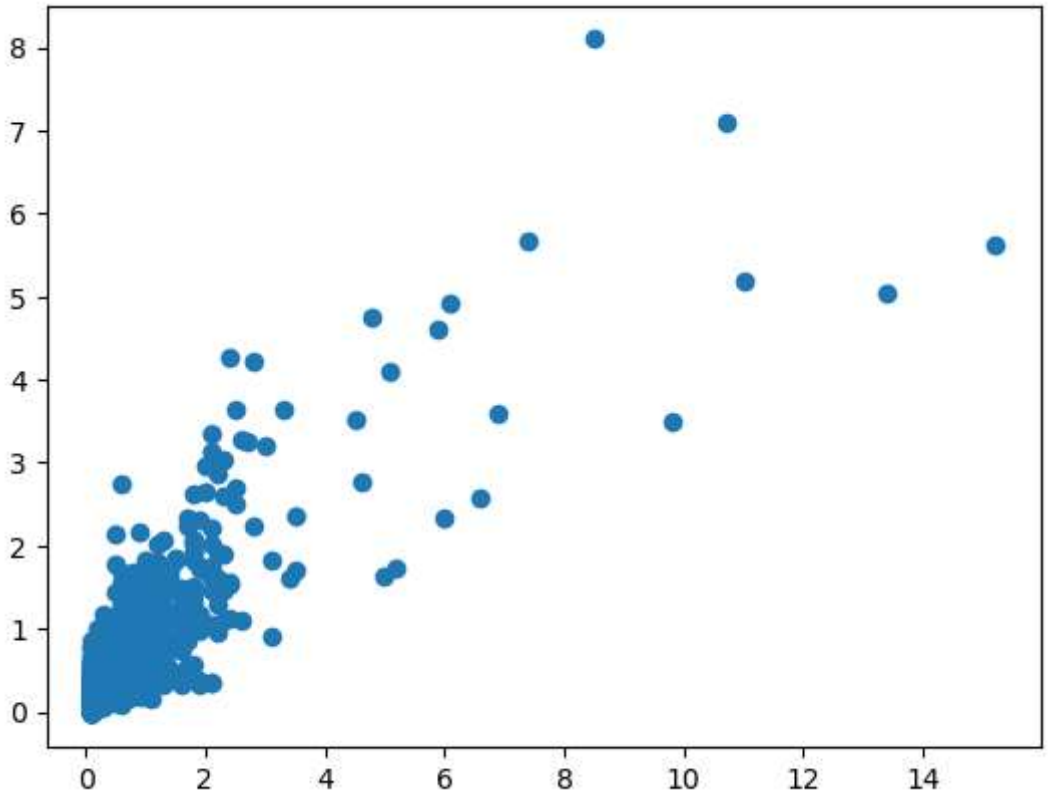
In [15]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

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 0x7fe4097671f0>
```



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

Ridge

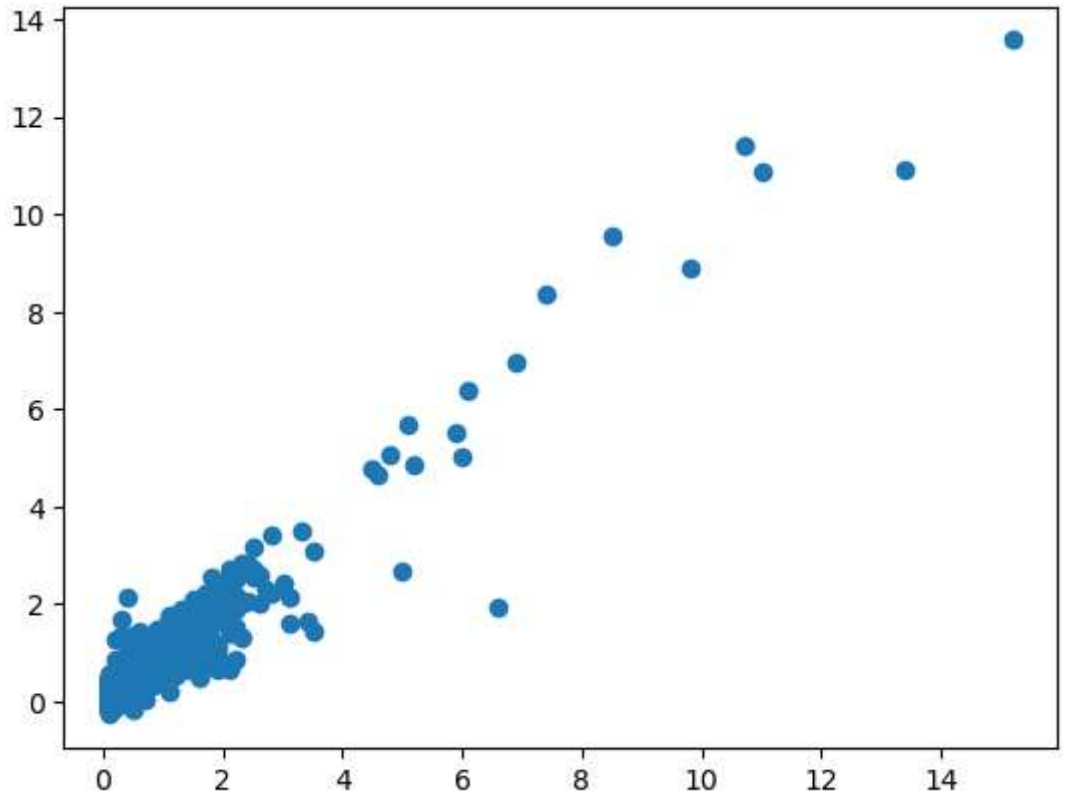
In [18]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

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 0x7fe4097fb3a0>
```



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

ElasticNet

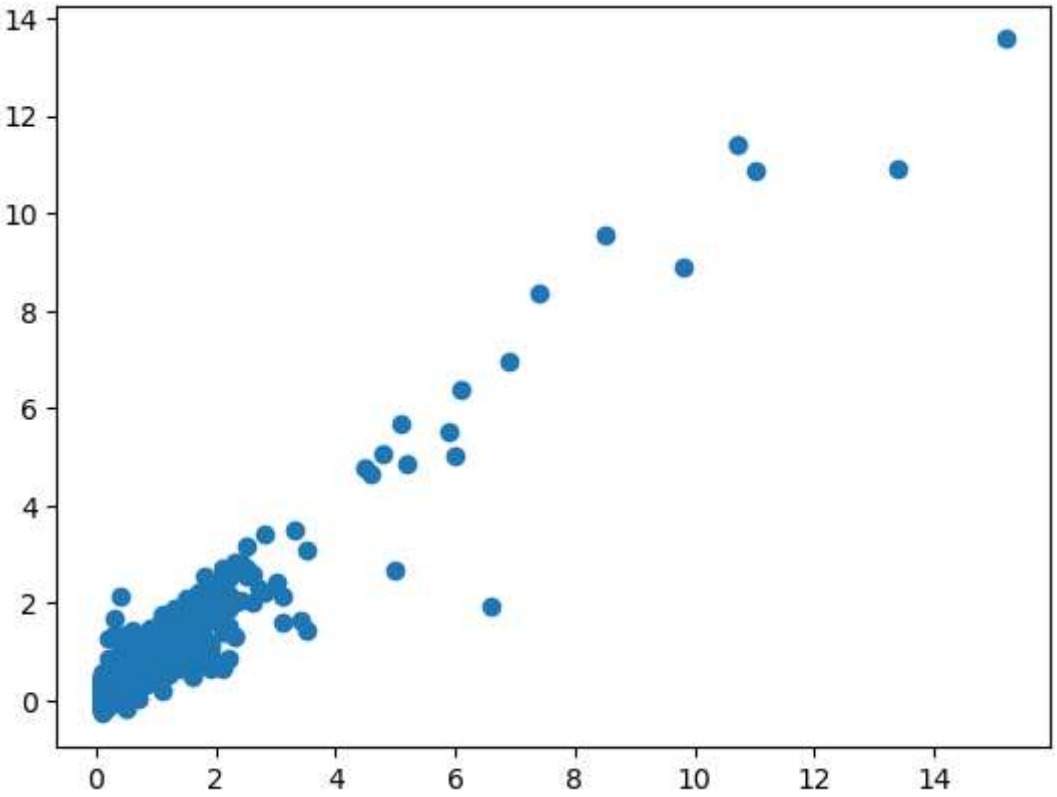
In [21]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

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 0x7fe409566980>



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

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

```
0.9059346080236226
```

Out[24]: 0.8762721659976673

Logistic

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

Out[25]:
```
Low     2428
High    1699
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)
```

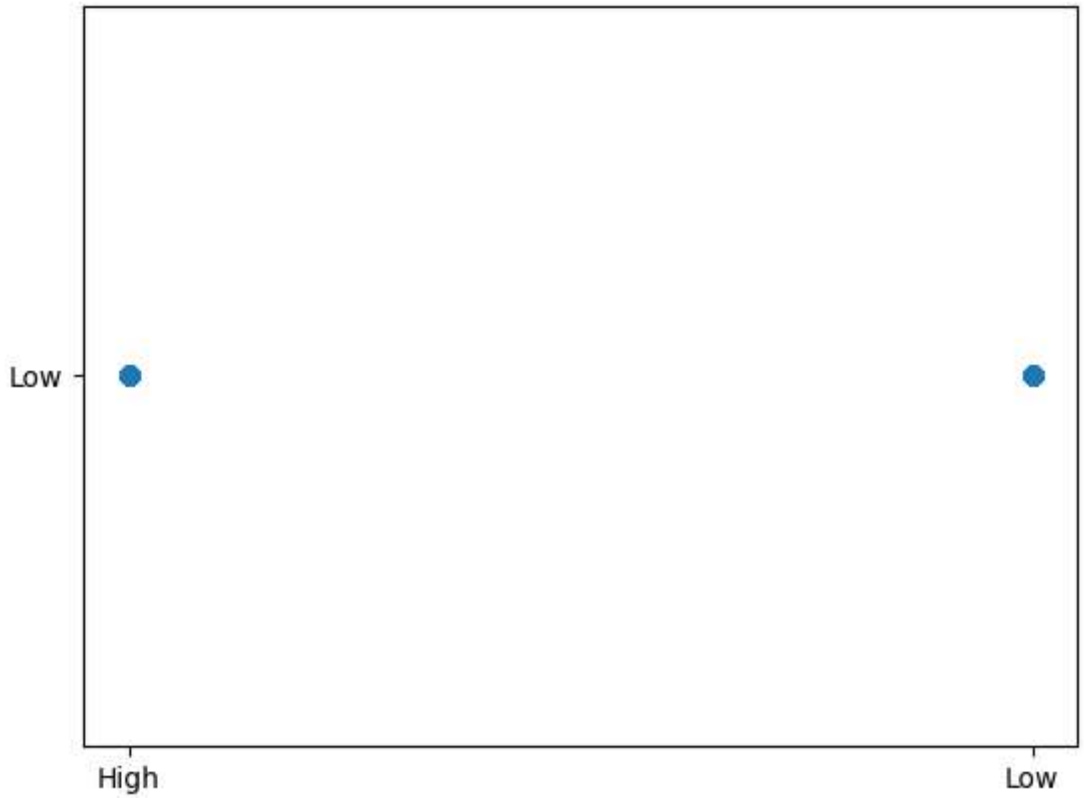
In [27]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

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 0x7fe409609bd0>
```



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)
```

```
 Text(0.26666666666666666, 0.07142857142857142, 'gini = 0.018\nsamples = 70\nvalue = [108, 1]\nclass = Ye
s'),
 Text(0.29333333333333333, 0.07142857142857142, 'gini = 0.0\nsamples = 228\nvalue = [360, 0]\nclass = Ye
s'),
 Text(0.3333333333333333, 0.21428571428571427, 'O_3 <= 54.5\ngini = 0.121\nsamples = 20\nvalue = [29, 2]
\nclass = Yes'),
 Text(0.32, 0.07142857142857142, 'gini = 0.346\nsamples = 5\nvalue = [7, 2]\nclass = Yes'),
 Text(0.3466666666666667, 0.07142857142857142, 'gini = 0.0\nsamples = 15\nvalue = [22, 0]\nclass = Yes'),
 Text(0.3333333333333333, 0.35714285714285715, 'gini = 0.496\nsamples = 6\nvalue = [6, 5]\nclass = Yes'),
 Text(0.4266666666666667, 0.5, 'NMHC <= 0.045\ngini = 0.208\nsamples = 90\nvalue = [127, 17]\nclass = Ye
s'),
 Text(0.4, 0.35714285714285715, 'PM25 <= 6.0\ngini = 0.444\nsamples = 26\nvalue = [24, 12]\nclass = Ye
s'),
 Text(0.38666666666666666, 0.21428571428571427, 'PM25 <= 3.5\ngini = 0.49\nsamples = 19\nvalue = [16, 12]
\nclass = Yes'),
 Text(0.37333333333333335, 0.07142857142857142, 'gini = 0.401\nsamples = 13\nvalue = [13, 5]\nclass = Ye
s'),
 Text(0.4, 0.07142857142857142, 'gini = 0.42\nsamples = 6\nvalue = [3, 7]\nclass = No'),
 Text(0.41333333333333333, 0.21428571428571427, 'gini = 0.0\nsamples = 7\nvalue = [8, 0]\nclass = Yes'),
 Text(0.4533333333333333, 0.35714285714285715, 'TOL <= 0.85\ngini = 0.088\nsamples = 64\nvalue = [103, 5]
```

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

```
Linear: 0.9056106323408553
Lasso: 0.6651452631235361
Ridge: 0.9059346080236226
ElasticNet: 0.8167044214155901
Logistic: 0.5819209039548022
Random Forest: 0.9639889196675899
```

Best Model is Random Forest

In [40]:
```
df2=pd.read_csv("/Users/bhoomish/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_2018.csv")
df2
```

Out[40]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	NaN	NaN	28079004
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.41	0.8	28079008
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	NaN	1.1	28079011
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69091	2018-02-01 00:00:00	NaN	NaN	0.5	NaN	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN	NaN	NaN	28079056
69092	2018-02-01 00:00:00	NaN	NaN	0.7	NaN	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0	NaN	NaN	28079057
69093	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN	NaN	NaN	28079058
69094	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN	NaN	NaN	28079059
69095	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN	NaN	NaN	28079060

69096 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69096 entries, 0 to 69095
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     69096 non-null  object 
 1   BEN      16950 non-null  float64
 2   CH4      8440 non-null   float64
 3   CO       28598 non-null  float64
 4   EBE      16949 non-null  float64
 5   NMHC     8440 non-null   float64
 6   NO       68826 non-null  float64
 7   NO_2     68826 non-null  float64
 8   NOx      68826 non-null  float64
 9   O_3      40049 non-null  float64
 10  PM10     36911 non-null  float64
 11  PM25     18912 non-null  float64
 12  SO_2     28586 non-null  float64
 13  TCH      8440 non-null   float64
 14  TOL      16950 non-null  float64
 15  station  69096 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 8.4+ MB
```

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

Out[42]:

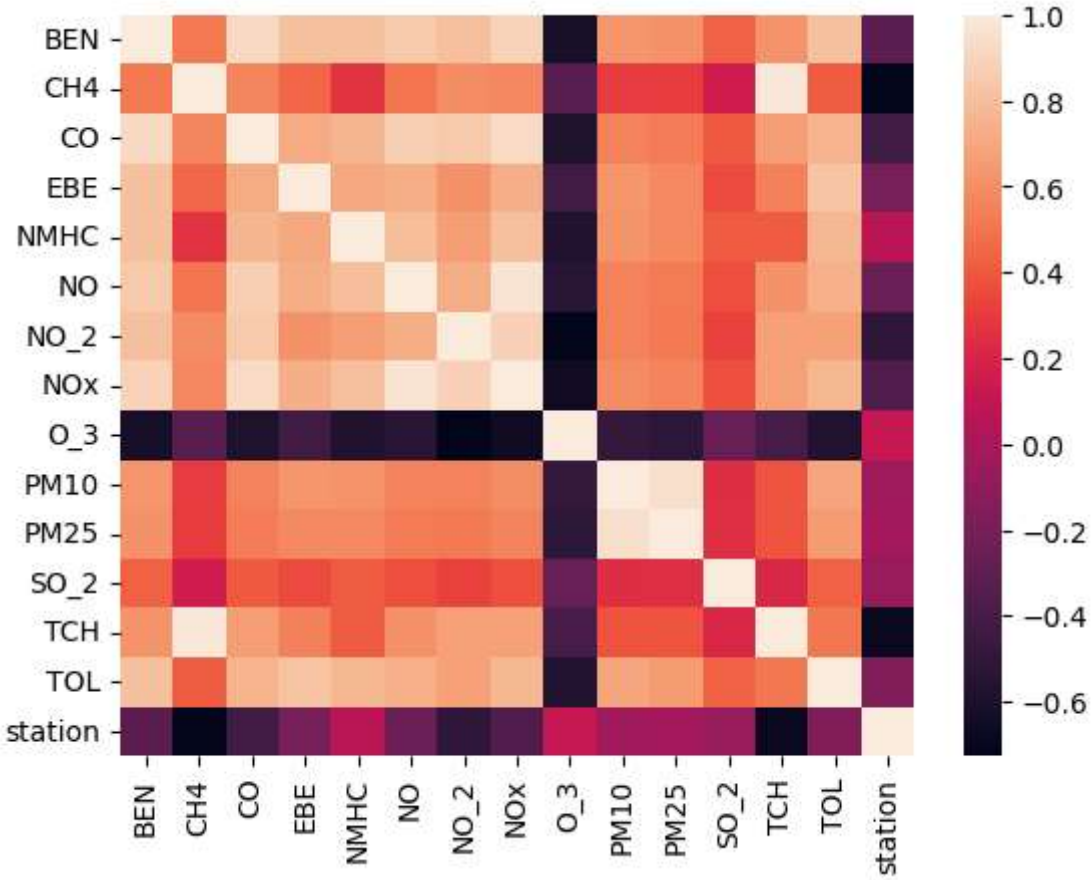
	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	station
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.41	0.8	28079008
6	2018-03-01 01:00:00	0.4	1.11	0.2	0.1	0.06	1.0	25.0	27.0	55.0	5.0	4.0	4.0	1.16	1.4	28079024
25	2018-03-01 02:00:00	0.4	1.42	0.2	0.1	0.01	4.0	26.0	32.0	64.0	4.0	4.0	3.0	1.44	0.7	28079008
30	2018-03-01 02:00:00	0.3	1.10	0.2	0.1	0.05	1.0	12.0	13.0	69.0	5.0	4.0	4.0	1.14	0.8	28079024
49	2018-03-01 03:00:00	0.3	1.41	0.2	0.1	0.01	3.0	16.0	20.0	68.0	3.0	2.0	3.0	1.42	0.4	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69030	2018-01-31 22:00:00	1.8	1.21	0.7	1.7	0.19	151.0	129.0	361.0	1.0	45.0	26.0	11.0	1.40	11.9	28079024
69049	2018-01-31 23:00:00	3.1	1.87	1.2	2.0	0.35	296.0	162.0	615.0	3.0	39.0	23.0	8.0	2.22	12.5	28079008
69054	2018-01-31 23:00:00	1.6	1.17	0.6	1.4	0.15	127.0	106.0	301.0	1.0	43.0	25.0	8.0	1.32	10.3	28079024
69073	2018-02-01 00:00:00	3.2	1.53	1.0	2.1	0.19	125.0	117.0	309.0	3.0	37.0	24.0	6.0	1.72	13.0	28079008
69078	2018-02-01 00:00:00	1.3	1.14	0.4	0.8	0.10	54.0	73.0	155.0	1.0	27.0	16.0	5.0	1.24	6.8	28079024

4562 rows × 16 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

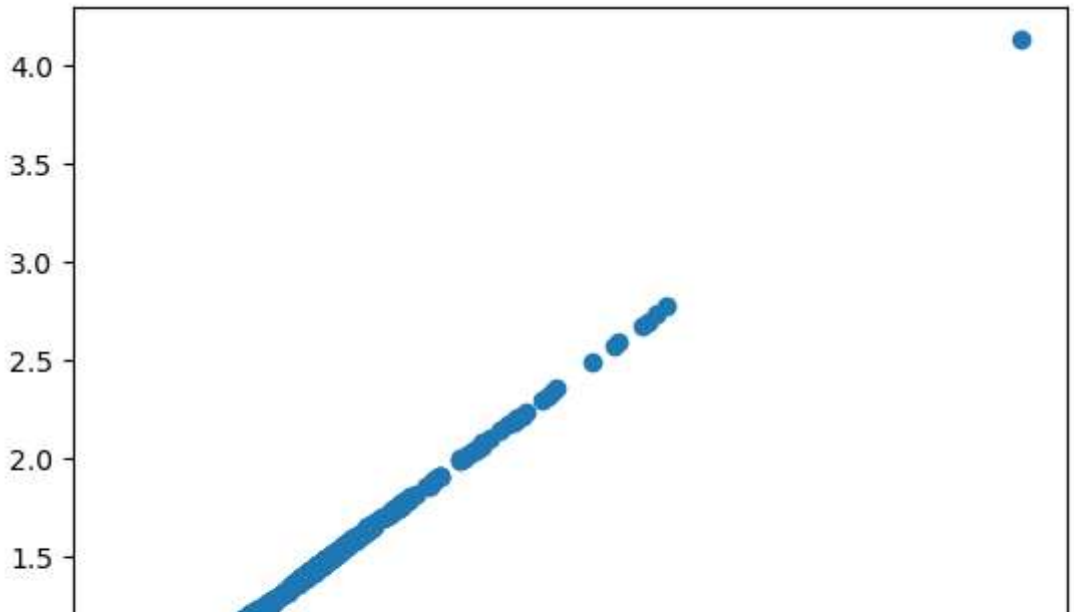
In [46]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

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

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 0x7fe419825b70>
```



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

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

Out[49]:
```
1.15    246
1.43    232
1.44    223
1.14    210
1.13    201
       ...
2.68      1
2.43      1
2.45      1
2.12      1
2.35      1
Name: TCH, Length: 143, 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    2477
1.0    2085
Name: TCH, dtype: int64
```

In [ ]:
```

```

Lasso

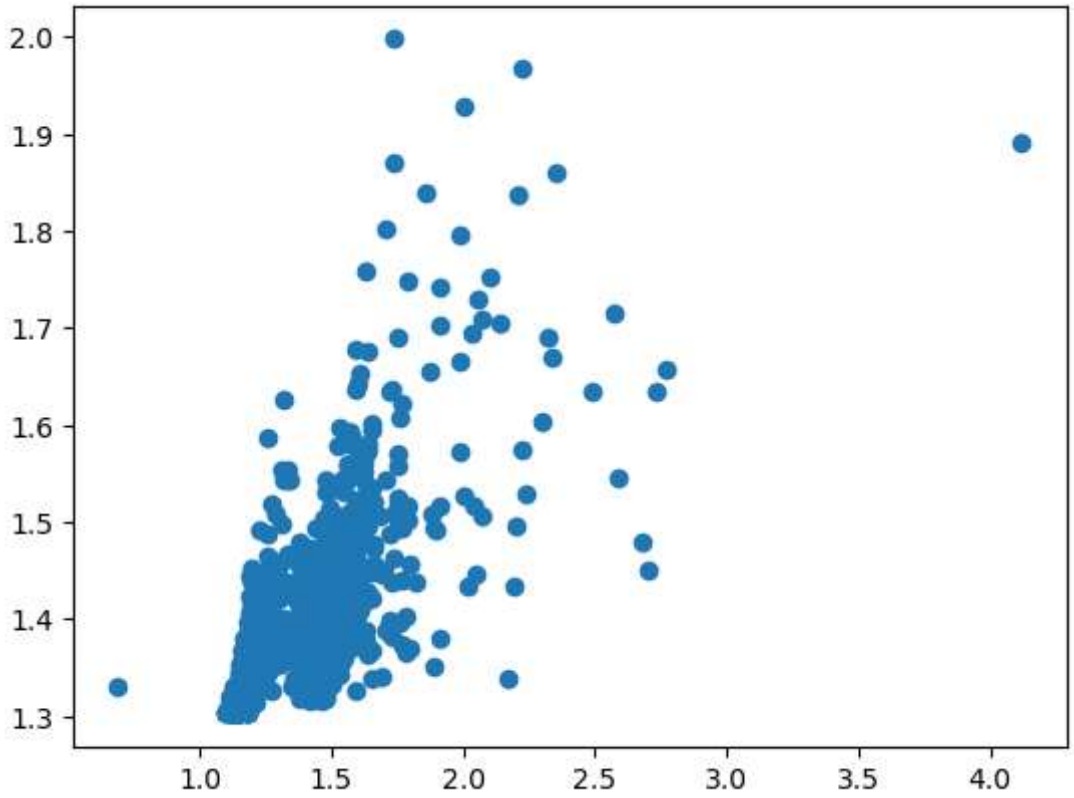
In [51]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

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 0x7fe419690670>



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

Ridge

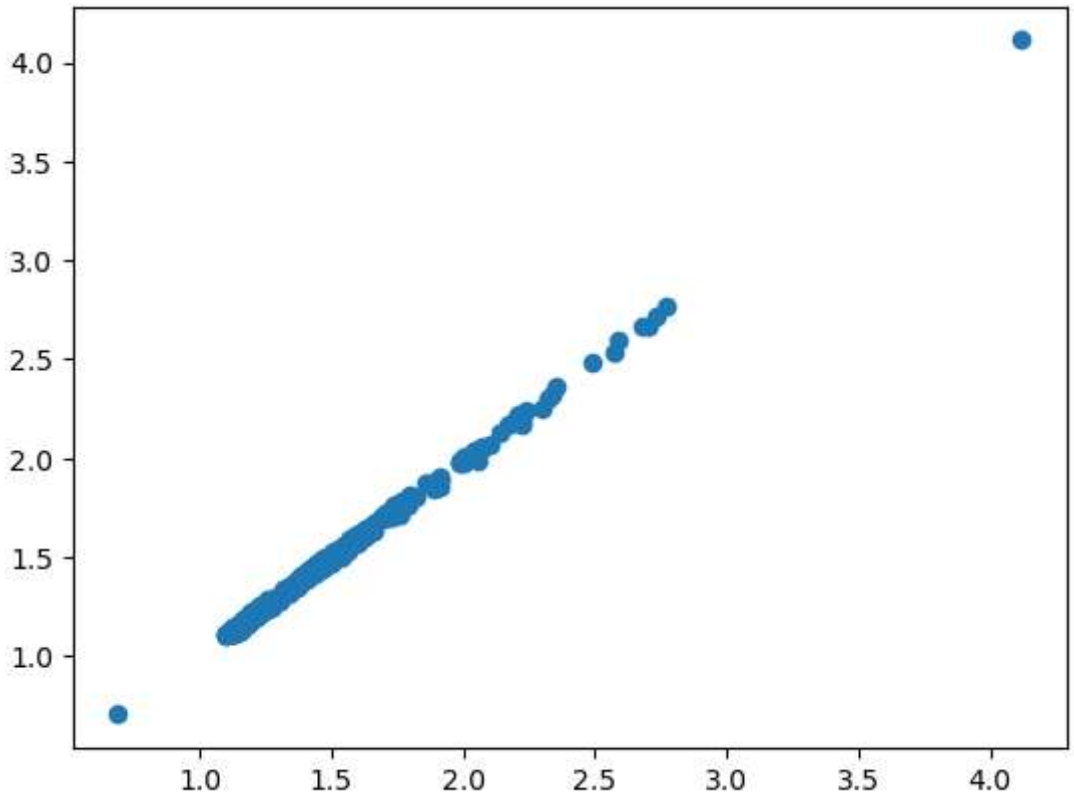
In [54]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

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 0x7fe3d852dde0>
```



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

ElasticNet

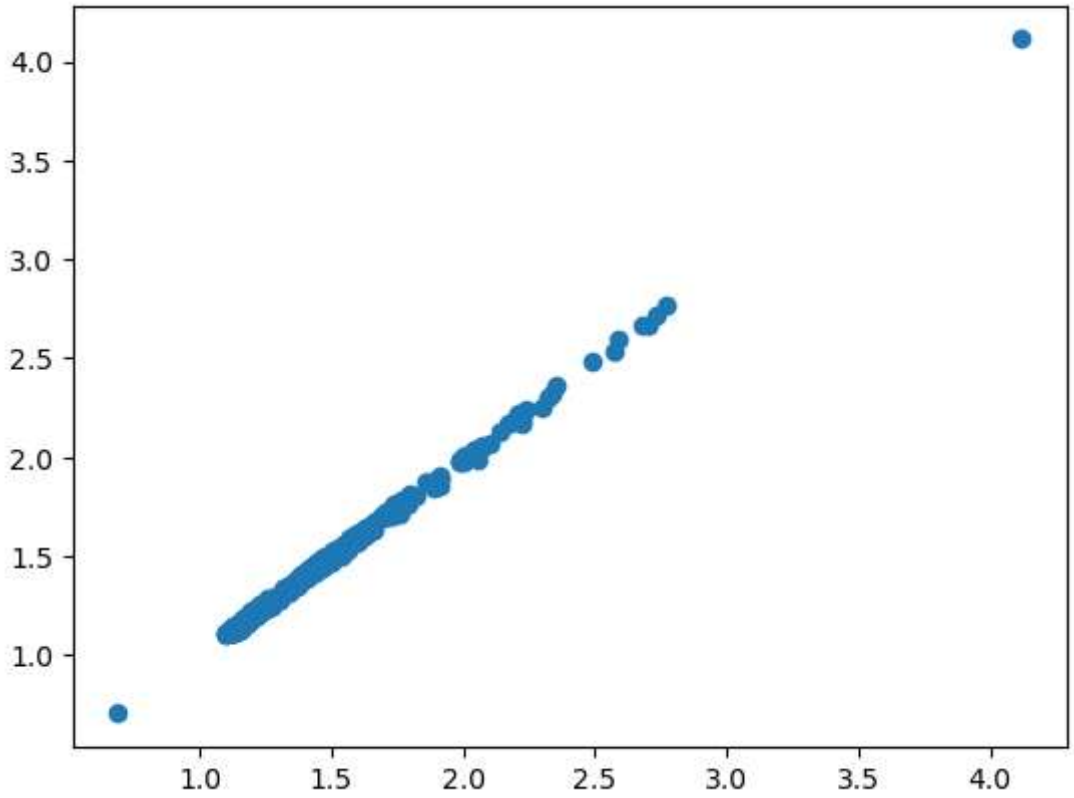
In [57]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

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 0x7fe4198f1510>



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

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

```
0.9983120614140699
```

Out[60]: 0.9981277627436985

Logistic

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

Out[61]:
```
High    2477
Low     2085
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)
```

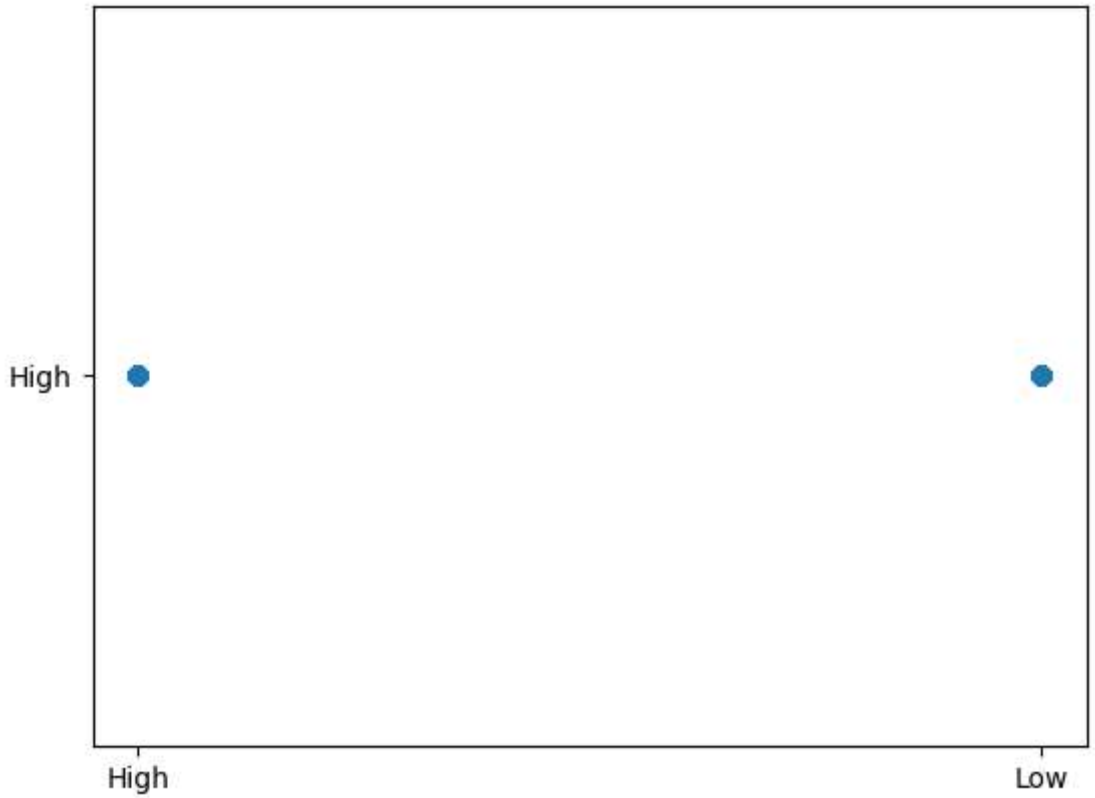
In [63]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

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 0x7fe419921510>
```



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]:
```
rfc_best=grid_search.best_estimator_
```

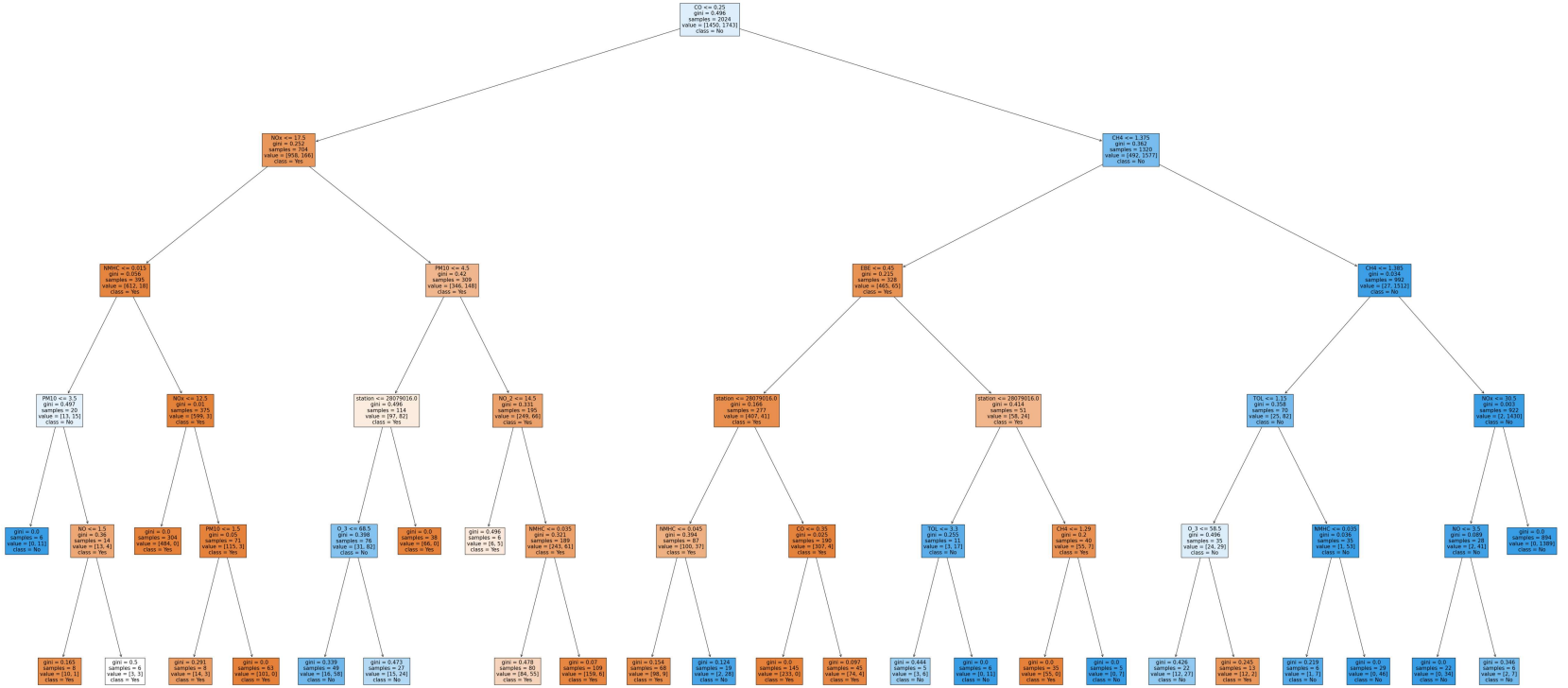
In [74]:

```
from sklearn.tree import plot_tree

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

Out[74]:
```
[Text(0.4557291666666667, 0.9166666666666666, 'CO <= 0.25\ngini = 0.496\nsamples = 2024\nvalue = [1450, 1743]\nclass = No'),
 Text(0.1875, 0.75, 'NOx <= 17.5\ngini = 0.252\nsamples = 704\nvalue = [958, 166]\nclass = Yes'),
 Text(0.08333333333333333, 0.5833333333333334, 'NMHC <= 0.015\ngini = 0.056\nsamples = 395\nvalue = [612, 18]\nclass = Yes'),
 Text(0.041666666666666664, 0.4166666666666667, 'PM10 <= 3.5\ngini = 0.497\nsamples = 20\nvalue = [13, 15]\nclass = No'),
 Text(0.020833333333333332, 0.25, 'gini = 0.0\nsamples = 6\nvalue = [0, 11]\nclass = No'),
 Text(0.0625, 0.25, 'NO <= 1.5\ngini = 0.36\nsamples = 14\nvalue = [13, 4]\nclass = Yes'),
 Text(0.041666666666666664, 0.08333333333333333, 'gini = 0.165\nsamples = 8\nvalue = [10, 1]\nclass = Yes'),
 Text(0.08333333333333333, 0.08333333333333333, 'gini = 0.5\nsamples = 6\nvalue = [3, 3]\nclass = Yes'),
 Text(0.125, 0.4166666666666667, 'NOx <= 12.5\ngini = 0.01\nsamples = 375\nvalue = [599, 3]\nclass = Yes'),
 Text(0.10416666666666667, 0.25, 'gini = 0.0\nsamples = 304\nvalue = [484, 0]\nclass = Yes'),
 Text(0.14583333333333334, 0.25, 'PM10 <= 1.5\ngini = 0.05\nsamples = 71\nvalue = [115, 3]\nclass = Yes'),
 Text(0.125, 0.08333333333333333, 'gini = 0.291\nsamples = 8\nvalue = [14, 3]\nclass = Yes'),
 Text(0.16666666666666666, 0.08333333333333333, 'gini = 0.0\nsamples = 63\nvalue = [101, 0]\nclass = Yes'),
 Text(0.2916666666666667, 0.5833333333333334, 'PM10 <= 4.5\ngini = 0.42\nsamples = 309\nvalue = [346, 148]\nclass = Yes'),
 Text(0.25, 0.4166666666666667, 'station <= 28079016.0\ngini = 0.496\nsamples = 114\nvalue = [97, 82]\nclass = Yes'),
 Text(0.22916666666666666, 0.25, 'O_3 <= 68.5\ngini = 0.398\nsamples = 76\nvalue = [31, 82]\nclass = No'),
 Text(0.20833333333333334, 0.08333333333333333, 'gini = 0.339\nsamples = 49\nvalue = [16, 58]\nclass = No'),
 Text(0.25, 0.08333333333333333, 'gini = 0.473\nsamples = 27\nvalue = [15, 24]\nclass = No'),
 Text(0.2708333333333333, 0.25, 'gini = 0.0\nsamples = 38\nvalue = [66, 0]\nclass = Yes'),
 Text(0.3333333333333333, 0.4166666666666667, 'NO_2 <= 14.5\ngini = 0.331\nsamples = 195\nvalue = [249, 66]\nclass = Yes'),
 Text(0.3125, 0.25, 'gini = 0.496\nsamples = 6\nvalue = [6, 5]\nclass = Yes'),
 Text(0.3541666666666667, 0.25, 'NMHC <= 0.035\ngini = 0.321\nsamples = 189\nvalue = [243, 61]\nclass = Yes'),
 Text(0.3333333333333333, 0.08333333333333333, 'gini = 0.478\nsamples = 80\nvalue = [84, 55]\nclass = Yes'),
 Text(0.375, 0.08333333333333333, 'gini = 0.07\nsamples = 109\nvalue = [159, 6]\nclass = Yes'),
 Text(0.7239583333333334, 0.75, 'CH4 <= 1.375\ngini = 0.362\nsamples = 1320\nvalue = [492, 1577]\nclass = No'),
 Text(0.5625, 0.5833333333333334, 'EBE <= 0.45\ngini = 0.215\nsamples = 328\nvalue = [465, 65]\nclass = Yes'),
 Text(0.4791666666666667, 0.4166666666666667, 'station <= 28079016.0\ngini = 0.166\nsamples = 277\nvalue = [407, 41]\nclass = Yes'),
 Text(0.4375, 0.25, 'NMHC <= 0.045\ngini = 0.394\nsamples = 87\nvalue = [100, 37]\nclass = Yes'),
 Text(0.4166666666666667, 0.08333333333333333, 'gini = 0.154\nsamples = 68\nvalue = [98, 9]\nclass = Yes'),
 Text(0.4583333333333333, 0.08333333333333333, 'gini = 0.124\nsamples = 19\nvalue = [2, 28]\nclass = No'),
 Text(0.5208333333333334, 0.25, 'CO <= 0.35\ngini = 0.025\nsamples = 190\nvalue = [307, 4]\nclass = Yes'),
 Text(0.5, 0.08333333333333333, 'gini = 0.0\nsamples = 145\nvalue = [233, 0]\nclass = Yes'),
 Text(0.5416666666666666, 0.08333333333333333, 'gini = 0.097\nsamples = 45\nvalue = [74, 4]\nclass = Yes'),
```

```
 Text(0.6458333333333334, 0.4166666666666667, 'station <= 28079016.0\ngini = 0.414\nsamples = 51\nvalue = [5
8, 24]\nclass = Yes'),
 Text(0.6041666666666666, 0.25, 'TOL <= 3.3\ngini = 0.255\nsamples = 11\nvalue = [3, 17]\nclass = No'),
 Text(0.5833333333333334, 0.08333333333333333, 'gini = 0.444\nsamples = 5\nvalue = [3, 6]\nclass = No'),
 Text(0.625, 0.08333333333333333, 'gini = 0.0\nsamples = 6\nvalue = [0, 11]\nclass = No'),
 Text(0.6875, 0.25, 'CH4 <= 1.29\ngini = 0.2\nsamples = 40\nvalue = [55, 7]\nclass = Yes'),
 Text(0.6666666666666666, 0.08333333333333333, 'gini = 0.0\nsamples = 35\nvalue = [55, 0]\nclass = Yes'),
 Text(0.7083333333333334, 0.08333333333333333, 'gini = 0.0\nsamples = 5\nvalue = [0, 7]\nclass = No'),
 Text(0.8854166666666666, 0.5833333333333334, 'CH4 <= 1.385\ngini = 0.034\nsamples = 992\nvalue = [27, 1512]
\nclass = No'),
 Text(0.8125, 0.4166666666666667, 'TOL <= 1.15\ngini = 0.358\nsamples = 70\nvalue = [25, 82]\nclass = No'),
 Text(0.7708333333333334, 0.25, 'O_3 <= 58.5\ngini = 0.496\nsamples = 35\nvalue = [24, 29]\nclass = No'),
 Text(0.75, 0.08333333333333333, 'gini = 0.426\nsamples = 22\nvalue = [12, 27]\nclass = No'),
 Text(0.7916666666666666, 0.08333333333333333, 'gini = 0.245\nsamples = 13\nvalue = [12, 2]\nclass = Yes'),
 Text(0.8541666666666666, 0.25, 'NMHC <= 0.035\ngini = 0.036\nsamples = 35\nvalue = [1, 53]\nclass = No'),
 Text(0.8333333333333334, 0.08333333333333333, 'gini = 0.219\nsamples = 6\nvalue = [1, 7]\nclass = No'),
 Text(0.875, 0.08333333333333333, 'gini = 0.0\nsamples = 29\nvalue = [0, 46]\nclass = No'),
 Text(0.9583333333333334, 0.4166666666666667, 'NOx <= 30.5\ngini = 0.003\nsamples = 922\nvalue = [2, 1430]\n
class = No'),
 Text(0.9375, 0.25, 'NO <= 3.5\ngini = 0.089\nsamples = 28\nvalue = [2, 41]\nclass = No'),
 Text(0.9166666666666666, 0.08333333333333333, 'gini = 0.0\nsamples = 22\nvalue = [0, 34]\nclass = No'),
 Text(0.9583333333333334, 0.08333333333333333, 'gini = 0.346\nsamples = 6\nvalue = [2, 7]\nclass = No'),
 Text(0.9791666666666666, 0.25, 'gini = 0.0\nsamples = 894\nvalue = [0, 1389]\nclass = No')]
```


CO <= 0.25
gini = 0.496
samples = 2024
value = [1450, 1743]
class = No
NOx <= 17.5
gini = 0.252
samples = 704
value = [958, 168]
class = Yes
CH4 <= 1.375
gini = 0.362
samples = 1320
value = [492, 1577]
class = No
NMHC <= 0.015
gini = 0.056
samples = 395
value = [612, 18]
class = Yes
PM10 <= 4.5
gini = 0.42
samples = 309
value = [346, 148]
class = Yes
EBE <= 0.45
gini = 0.215
samples = 328
value = [465, 65]
class = Yes
CH4 <= 1.385
gini = 0.034
samples = 992
value = [27, 1512]
class = No
PM10 <= 3.5
gini = 0.497
samples = 27
value = [20, 15]
class = No
NOx <= 12.5
gini = 0.01
samples = 368
value = [592, 3]
class = Yes
station <= 28079016.0
gini = 0.496
samples = 114
value = [76, 92]
class = No
NO_2 <= 14.5
gini = 0.131
samples = 195
value = [270, 56]
class = Yes
station <= 28079016.0
gini = 0.167
samples = 77
value = [111, 41]
class = Yes
station <= 28079016.0
gini = 0.414
samples = 51
value = [76, 24]
class = Yes
TOL <= 1.15
gini = 0.358
samples = 70
value = [24, 82]
class = No
NOx <= 30.5
gini = 0.003
samples = 922
value = [3, 1430]
class = No
gini = 0.0
samples = 6
value = [0, 11]
class = No
NO <= 1.5
gini = 0.36
samples = 14
value = [13, 4]
class = Yes
gini = 0.0
samples = 304
value = [484, 0]
class = Yes
PM10 <= 3.5
gini = 0.05
samples = 71
value = [115, 3]
class = Yes
station <= 28079016.0
gini = 0.496
samples = 38
value = [31, 27]
class = Yes
gini = 0.0
samples = 38
value = [66, 0]
class = Yes
gini = 0.496
samples = 6
value = [6, 5]
class = Yes
NMHC <= 0.025
gini = 0.321
samples = 189
value = [243, 61]
class = Yes
NMHC <= 0.045
gini = 0.394
samples = 67
value = [100, 37]
class = Yes
CO <= 0.35
gini = 0.025
samples = 190
value = [307, 4]
class = Yes
TOL <= 1.3
gini = 0.255
samples = 11
value = [3, 17]
class = No
CH4 <= 1.29
gini = 0.2
samples = 40
value = [53, 7]
class = Yes
O_3 <= 58.5
gini = 0.496
samples = 35
value = [24, 29]
class = No
NMHC <= 0.035
gini = 0.036
samples = 35
value = [1, 53]
class = No
NO <= 3.5
gini = 0.089
samples = 28
value = [2, 41]
class = No
gini = 0.0
samples = 894
value = [0, 1389]
class = No
gini = 0.185
samples = 9
value = [10, 1]
class = Yes
gini = 0.5
samples = 6
value = [3, 3]
class = Yes
gini = 0.291
samples = 13
value = [14, 3]
class = Yes
gini = 0.0
samples = 63
value = [101, 0]
class = Yes
gini = 0.339
samples = 49
value = [16, 50]
class = No
gini = 0.423
samples = 27
value = [15, 24]
class = No
gini = 0.479
samples = 80
value = [94, 55]
class = Yes
gini = 0.07
samples = 109
value = [156, 6]
class = Yes
gini = 0.164
samples = 48
value = [90, 9]
class = Yes
gini = 0.124
samples = 19
value = [10, 28]
class = No
gini = 0.0
samples = 145
value = [251, 0]
class = Yes
gini = 0.097
samples = 45
value = [56, 4]
class = Yes
gini = 0.444
samples = 5
value = [1, 6]
class = No
gini = 0.2
samples = 6
value = [1, 11]
class = No
gini = 0.0
samples = 30
value = [53, 0]
class = Yes
gini = 0.0
samples = 10
value = [0, 7]
class = No
gini = 0.426
samples = 22
value = [12, 27]
class = No
gini = 0.245
samples = 13
value = [12, 2]
class = Yes
gini = 0.219
samples = 7
value = [1, 7]
class = No
gini = 0.0
samples = 28
value = [0, 46]
class = No
gini = 0.0
samples = 22
value = [0, 34]
class = No
gini = 0.346
samples = 6
value = [2, 7]
class = No

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

```
Linear: 0.9996090192033011
Lasso: 0.3875640304973138
Ridge: 0.9983120614140699
ElasticNet: 0.609503466089908
Logistic: 0.5507669831994156
Random Forest: 0.9793305665541436
```

Best model is Random Forest