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,Elasti
from sklearn.model_selection import train_test_split
```

In [2]:
```
df=pd.read_csv("C:/Users/user/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_2011
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	28079004
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	8.7	28079008
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	28079011
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	28079056
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	28079057
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	28079058
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	28079059
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	28079060

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]:

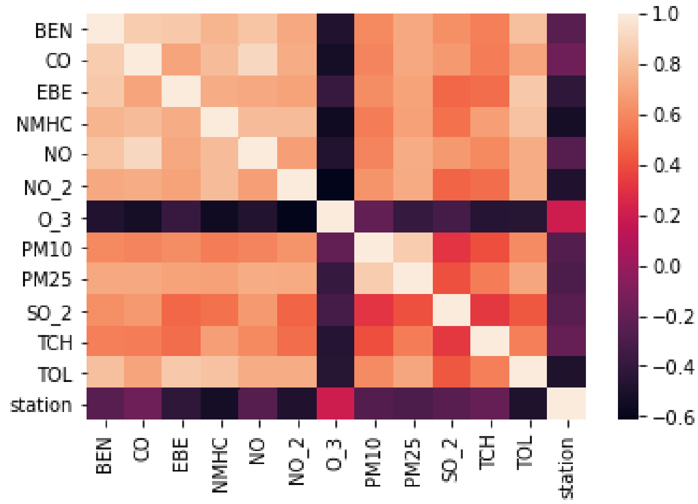
	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	28079008
6	2011-11-01 01:00:00	0.7	0.3	1.1	0.16	17.0	66.0	7.0	22.0	16.0	2.0	1.36	1.7	28079024
25	2011-11-01 02:00:00	1.8	0.3	2.8	0.20	34.0	76.0	3.0	34.0	21.0	8.0	1.71	7.4	28079008
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	2.9	28079024
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	6.2	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	0.7	28079024
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	4.9	28079008
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	0.9	28079024
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	3.8	28079008
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	0.9	28079024

16460 rows × 14 columns

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

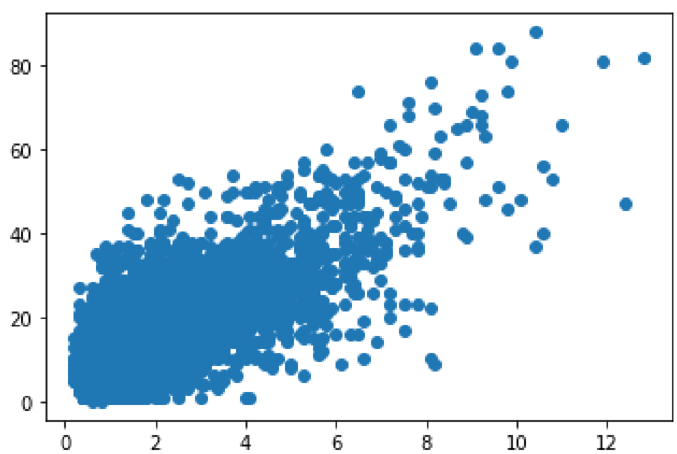
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <AxesSubplot:>



In [7]:
```
plt.plot(df1["EBE"],df1["PM25"],"o")
```

Out[7]: [<matplotlib.lines.Line2D at 0x1aa70adb160>]



In [8]:
```
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)
```

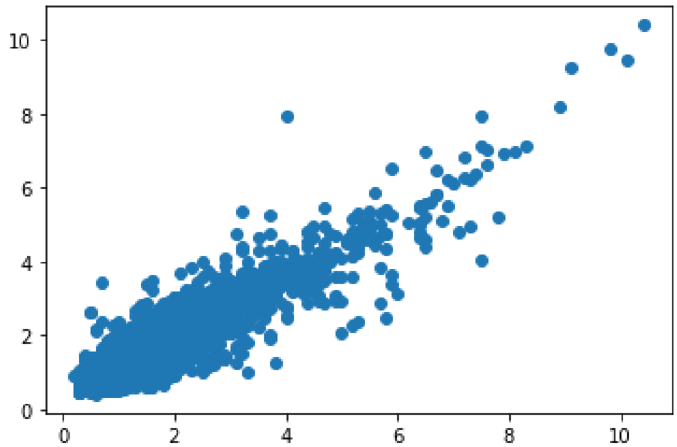
Linear

In [9]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[9]: LinearRegression()

In [10]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[10]: <matplotlib.collections.PathCollection at 0x1aa70cb4cd0>



In [11]:
```
lis=li.score(x_test,y_test)
```

In [12]:
```
df1["TCH"].value_counts()
```

Out[12]:
```
1.30    897
1.29    878
1.28    856
1.31    827
1.27    820
       ...
3.41      1
2.88      1
2.41      1
2.80      1
2.49      1
Name: TCH, Length: 171, dtype: int64
```

In [13]:
```
df1.loc[df1["TCH"]<1.40,"TCH"]=1
df1.loc[df1["TCH"]>1.40,"TCH"]=2
df1["TCH"].value_counts()
```

Out[13]:
```
1.0    12828
2.0     3632
Name: TCH, dtype: int64
```

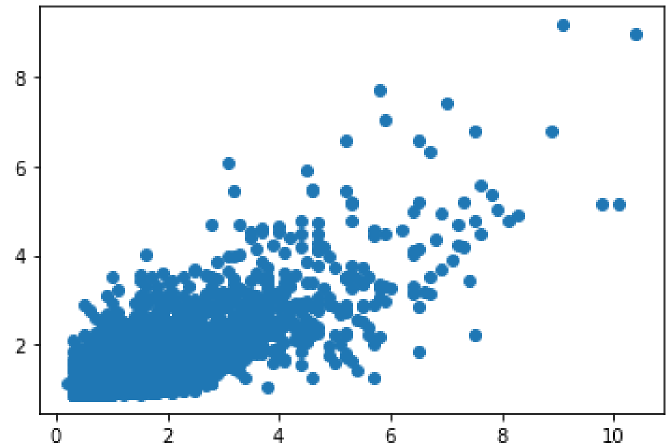
Lasso

In [14]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[14]: Lasso(alpha=5)

In [15]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[15]: <matplotlib.collections.PathCollection at 0x1aa70d1ee80>



In [16]:
```
las=la.score(x_test,y_test)
```

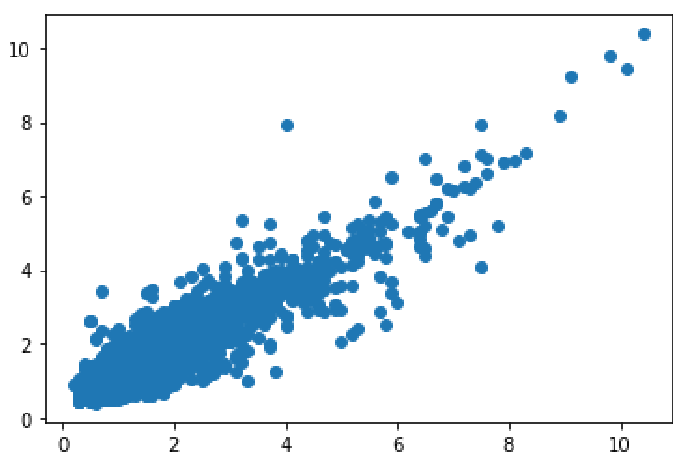
Ridge

In [17]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[17]: Ridge(alpha=1)

In [18]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[18]: <matplotlib.collections.PathCollection at 0x1aa70b271f0>



In [19]:
```
rrs=rr.score(x_test,y_test)
```

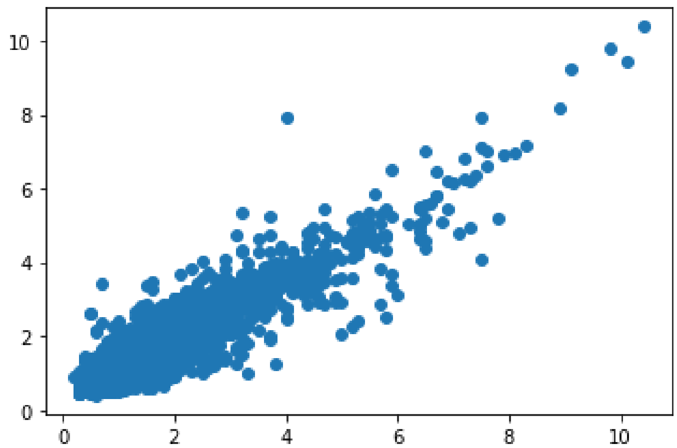
ElasticNet

In [20]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[20]:
```
ElasticNet()
```

In [21]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x1aa70d9ca30>
```



In [22]:
```
ens=en.score(x_test,y_test)
```

In [23]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.8305645679872002
```

Out[23]:
```
0.8132155812013233
```

Logistic

In [24]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["TCH"].value_counts()
```

Out[24]:
```
Low     12828
High     3632
Name: TCH, dtype: int64
```

In [25]:
```
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 [26]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[26]:
```
LogisticRegression()
```

In [27]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[27]: <matplotlib.collections.PathCollection at 0x1aa70b57cd0>

In [28]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [29]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [30]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df1=df1.replace(g1)
```

In [31]:
```
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 [32]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[32]: RandomForestClassifier()

In [33]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [34]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[34]:
```
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 [35]:
```
rfcs=grid_search.best_score_
```

In [36]:
```
rfc_best=grid_search.best_estimator_
```

In [37]:
```
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"],fill
```

```
\nvalue = [130, 3]\nclass = Yes'),
 Text(40.95412844036697, 155.3142857142857, 'gini = 0.117\nsamples = 35\nvalue = [4
5, 3]\nclass = Yes'),
 Text(122.86238532110092, 155.3142857142857, 'gini = 0.0\nsamples = 56\nvalue = [85,
0]\nclass = Yes'),
 Text(245.72477064220183, 465.9428571428573, 'NO_2 <= 23.5\ngini = 0.348\nsamples =
128\nvalue = [152, 44]\nclass = Yes'),
 Text(204.77064220183485, 155.3142857142857, 'gini = 0.408\nsamples = 12\nvalue =
[4, 10]\nclass = No'),
 Text(286.6788990825688, 155.3142857142857, 'gini = 0.304\nsamples = 116\nvalue = [1
48, 34]\nclass = Yes'),
 Text(491.44954128440367, 776.5714285714287, 'SO_2 <= 3.5\ngini = 0.431\nsamples = 6
96\nvalue = [761, 349]\nclass = Yes'),
 Text(409.5412844036697, 465.9428571428573, 'EBE <= 1.85\ngini = 0.495\nsamples = 19
8\nvalue = [171, 141]\nclass = Yes'),
 Text(368.58715596330273, 155.3142857142857, 'gini = 0.485\nsamples = 181\nvalue =
[168, 119]\nclass = Yes'),
 Text(450.4954128440367, 155.3142857142857, 'gini = 0.211\nsamples = 17\nvalue = [3,
22]\nclass = No'),
 Text(573.3577981651376, 465.9428571428573, 'BEN <= 1.75\ngini = 0.385\nsamples = 49
```

In [38]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.8305073316874275
Lasso: 0.5846802065548421
Ridge: 0.8305645679872002
ElasticNet: 0.7133987542437089
Logistic: 0.7778452814904819
Random Forest: 0.8899496615170978
```

Best Model is Random Forest

In [39]:
```
df2=pd.read_csv("C:/Users/user/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_201
df2
```

Out[39]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	28079004
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	2.4	28079008
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	28079011
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	28079056
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	28079057
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	28079058
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	28079059
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	28079060

210720 rows × 14 columns

In [40]:
```
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 [41]:
```
df3=df2.dropna()
df3
```

Out[41]:

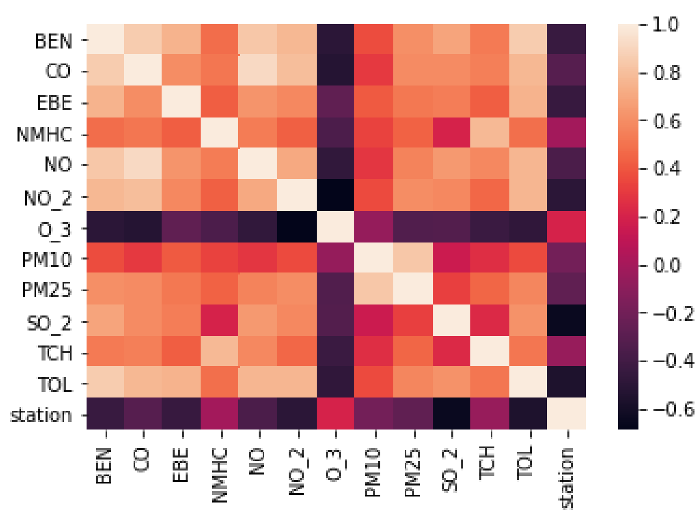
	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
6	2012-09-01 01:00:00	0.4	0.2	0.8	0.24	1.0	7.0	57.0	11.0	7.0	2.0	1.33	0.6	28079024
30	2012-09-01 02:00:00	0.4	0.2	0.7	0.24	1.0	5.0	55.0	5.0	5.0	2.0	1.33	0.5	28079024
54	2012-09-01 03:00:00	0.4	0.2	0.7	0.24	1.0	4.0	56.0	6.0	4.0	2.0	1.33	0.5	28079024
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	0.4	28079024
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	0.5	28079024
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	2.3	28079024
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	6.2	28079008
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	1.9	28079024
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	4.9	28079008
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	1.2	28079024

10916 rows × 14 columns

In [42]:
```
df3=df3.drop(["date"],axis=1)
```

In [43]:
```
sns.heatmap(df3.corr())
```

Out[43]: <AxesSubplot:>



In [44]:
```
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

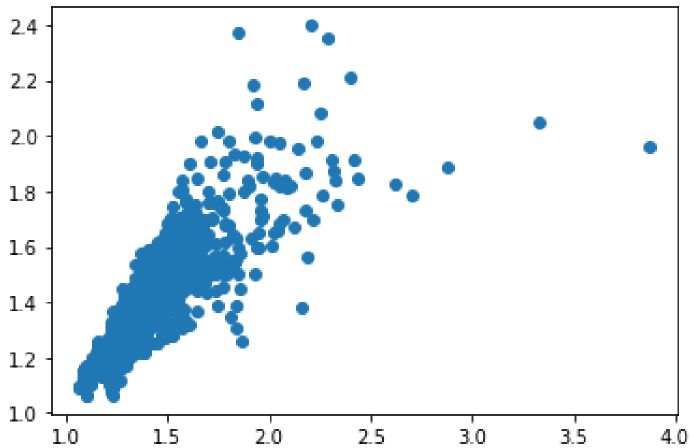
In [45]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[45]:
```
LinearRegression()
```

In [ ]:

In [46]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[46]:
```
<matplotlib.collections.PathCollection at 0x1aa72e176a0>
```



In [47]:
```
lis=li.score(x_test,y_test)
```

In [48]:
```
df3["TCH"].value_counts()
```

Out[48]:
```
1.30    737
1.31    676
1.32    644
1.33    552
1.29    529
       ... 
3.03      1
3.01      1
2.47      1
2.33      1
2.07      1
Name: TCH, Length: 167, dtype: int64
```

In [49]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[49]:
```
1.0    8772
2.0    2144
Name: TCH, dtype: int64
```

In [ ]:

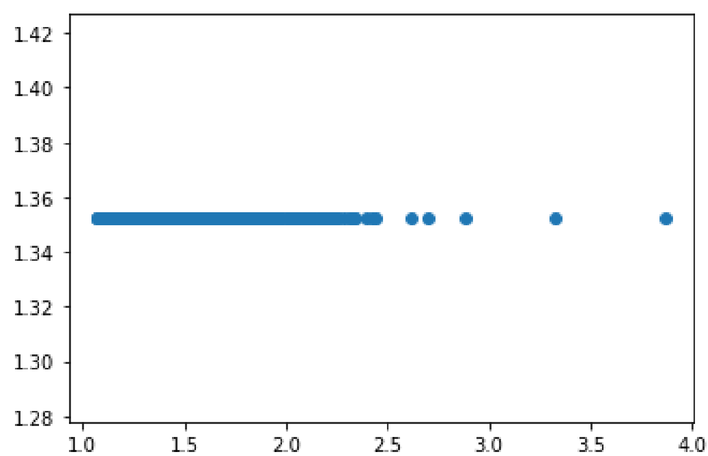
Lasso

In [50]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[50]:
```
Lasso(alpha=5)
```

In [51]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[51]:
```
<matplotlib.collections.PathCollection at 0x1aa71741790>
```



In [52]:
```
las=la.score(x_test,y_test)
```

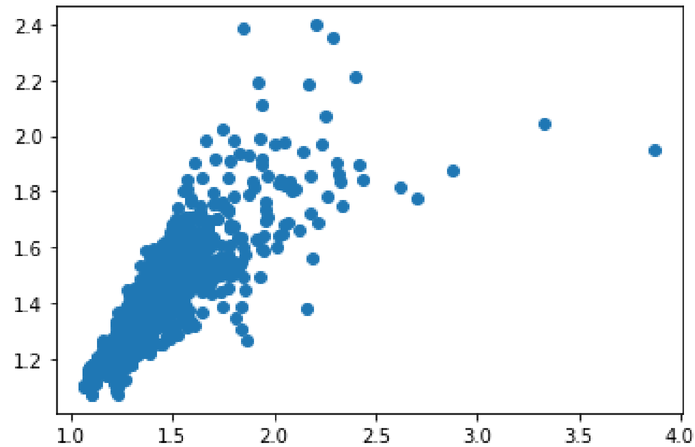
Ridge

In [53]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[53]:
```
Ridge(alpha=1)
```

In [54]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[54]:
```
<matplotlib.collections.PathCollection at 0x1aa717a3310>
```



In [55]:
```
rrs=rr.score(x_test,y_test)
```

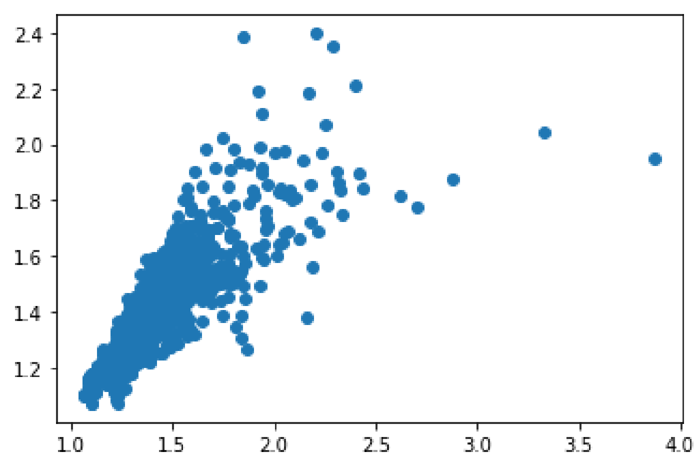
ElasticNet

In [56]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[56]:
```
ElasticNet()
```

In [57]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[57]:
```
<matplotlib.collections.PathCollection at 0x1aa717f88e0>
```



In [58]:
```
ens=en.score(x_test,y_test)
```

In [59]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```
```
0.6854937601485451
```
Out[59]:
```
0.6889077427743225
```

Logistic

In [60]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```
Out[60]:
```
Low     8772
High    2144
Name: TCH, dtype: int64
```

In [61]:
```
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 [62]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```
Out[62]:
```
LogisticRegression()
```

In [63]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```
Out[63]:
```
<matplotlib.collections.PathCollection at 0x1aa716bc580>
```

In [64]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [65]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [66]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [67]:
```
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 [68]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[68]:
```
RandomForestClassifier()
```

In [69]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [70]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[70]:
```
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 [71]:
```
rfcs=grid_search.best_score_
```

In [72]:
```
rfc_best=grid_search.best_estimator_
```

In [73]:
```
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"],fill
```

Out[73]:
```
[Text(2493.5625, 2019.0857142857144, 'TOL <= 7.15\ngini = 0.308\nsamples = 4812\nval
ue = [6188, 1453]\nclass = Yes'),
 Text(1488.0, 1708.457142857143, 'NO <= 22.5\ngini = 0.259\nsamples = 4518\nvalue =
[6094, 1098]\nclass = Yes'),
 Text(744.0, 1397.8285714285716, 'O_3 <= 23.5\ngini = 0.178\nsamples = 3782\nvalue =
[5456, 597]\nclass = Yes'),
 Text(372.0, 1087.2, 'NMHC <= 0.275\ngini = 0.452\nsamples = 574\nvalue = [603, 317]
\nclass = Yes'),
 Text(186.0, 776.5714285714287, 'PM10 <= 16.5\ngini = 0.258\nsamples = 406\nvalue =
[556, 100]\nclass = Yes'),
 Text(93.0, 465.9428571428573, 'station <= 28079016.0\ngini = 0.18\nsamples = 248\nv
alue = [361, 40]\nclass = Yes'),
 Text(46.5, 155.3142857142857, 'gini = 0.0\nsamples = 44\nvalue = [73, 0]\nclass = Y
es'),
 Text(139.5, 155.3142857142857, 'gini = 0.214\nsamples = 204\nvalue = [288, 40]\ncla
ss = Yes'),
 Text(279.0, 465.9428571428573, 'TOL <= 1.05\ngini = 0.36\nsamples = 158\nvalue = [1
95, 60]\nclass = Yes'),
 Text(232.5, 155.3142857142857, 'gini = 0.5\nsamples = 23\nvalue = [20, 20]\nclass =
```

In [74]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.6851032563985955
Lasso: -0.0014194828338582877
Ridge: 0.6854937601485451
ElasticNet: 0.34195759108298374
Logistic: 0.8006106870229007
Random Forest: 0.9331240896615699
```

Best model is Random Forest

In [ ]: