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_2015
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28079004
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28079008
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28079011
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210091	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN	28079056
210092	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN	28079057
210093	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN	28079058
210094	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN	28079059
210095	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN	28079060

210096 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210096 entries, 0 to 210095
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210096 non-null  object 
 1   BEN      51039 non-null   float64
 2   CO       86827 non-null   float64
 3   EBE      50962 non-null   float64
 4   NMHC     25756 non-null   float64
 5   NO       208805 non-null  float64
 6   NO_2     208805 non-null  float64
 7   O_3      121574 non-null  float64
 8   PM10     102745 non-null  float64
 9   PM25     48798 non-null   float64
 10  SO_2     86898 non-null   float64
 11  TCH      25756 non-null   float64
 12  TOL      50626 non-null   float64
 13  station  210096 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

Out[4]:

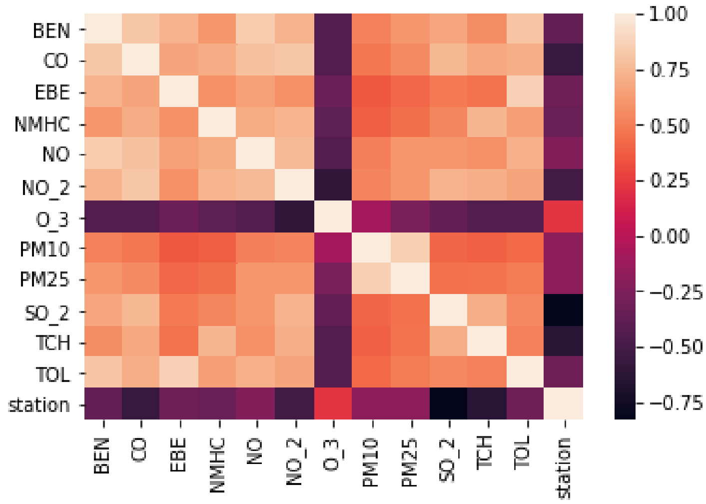
	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28079008
6	2015-10-01 01:00:00	0.5	0.3	0.3	0.12	6.0	83.0	1.0	19.0	12.0	3.0	1.29	4.8	28079024
25	2015-10-01 02:00:00	1.6	0.7	1.3	0.38	81.0	105.0	4.0	36.0	19.0	13.0	1.93	6.9	28079008
30	2015-10-01 02:00:00	0.4	0.3	0.3	0.11	5.0	72.0	2.0	16.0	10.0	2.0	1.27	7.8	28079024
49	2015-10-01 03:00:00	2.2	0.8	1.8	0.41	111.0	104.0	4.0	35.0	20.0	14.0	2.05	13.9	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210030	2015-07-31 22:00:00	0.1	0.1	0.1	0.06	1.0	10.0	69.0	10.0	3.0	2.0	1.18	0.2	28079024
210049	2015-07-31 23:00:00	0.4	0.3	0.1	0.12	3.0	28.0	56.0	15.0	7.0	12.0	1.45	1.2	28079008
210054	2015-07-31 23:00:00	0.1	0.1	0.1	0.06	1.0	10.0	63.0	5.0	1.0	2.0	1.18	0.2	28079024
210073	2015-08-01 00:00:00	0.1	0.3	0.1	0.11	2.0	23.0	59.0	5.0	2.0	11.0	1.44	0.6	28079008
210078	2015-08-01 00:00:00	0.1	0.1	0.1	0.06	1.0	8.0	65.0	7.0	1.0	2.0	1.18	0.4	28079024

16026 rows × 14 columns

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

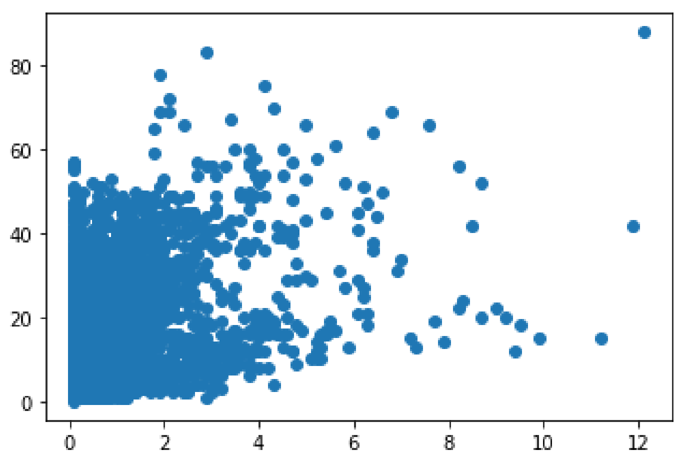
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <AxesSubplot:>



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

Out[7]: [<matplotlib.lines.Line2D at 0x23a1638ebb0>]



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)
```

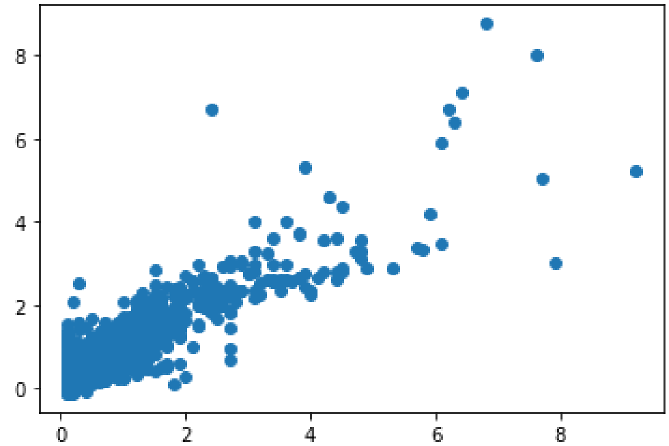
Linear

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 0x23a16575700>
```



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

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

Out[12]:
```
1.20    905
1.19    873
1.21    793
1.22    638
1.18    465
       ...
2.79      1
4.46      1
2.48      1
3.43      1
2.63      1
Name: TCH, Length: 184, 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]:
```
2.0    8290
1.0    7736
Name: TCH, dtype: int64
```

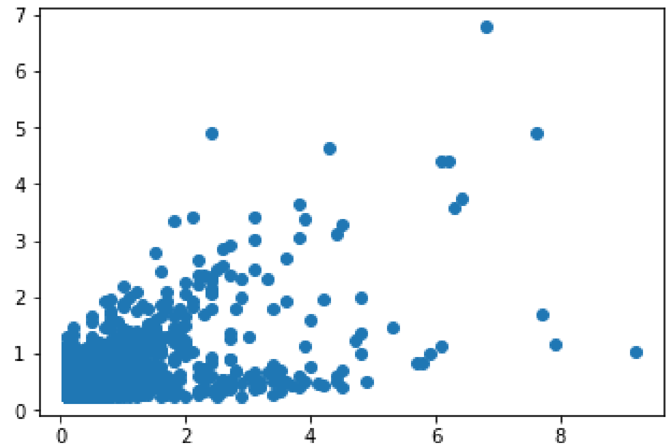
Lasso

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 0x23a165d8e50>



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

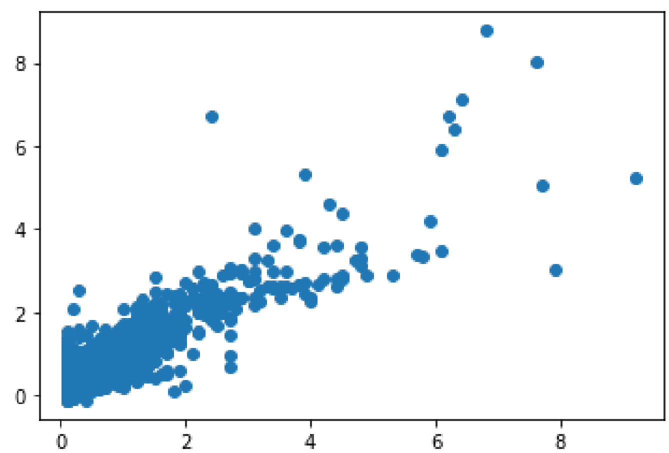
Ridge

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 0x23a163df9a0>



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

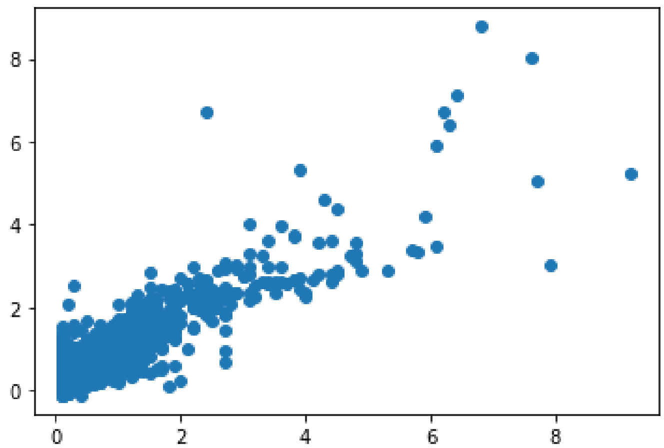
ElasticNet

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 0x23a16657ee0>
```



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

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

```
0.8030115171965686
```

Out[23]:
```
0.7576572353945755
```

Logistic

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

Out[24]:
```
High    8290
Low     7736
Name: TCH, dtype: int64
```

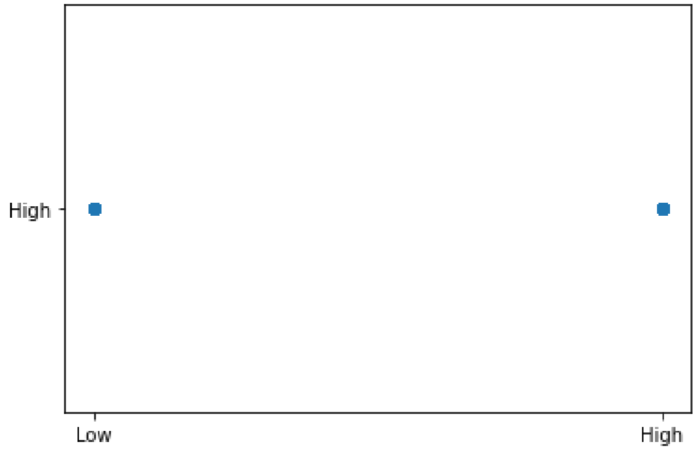
In [25]:
```
x=df1.drop(["TCH"],axis=1)
y=df1["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x23a16414a60>
```



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
```

Out[37]:
```
[Text(2200.56338028169, 2019.0857142857144, 'SO_2 <= 5.5\ngini = 0.5\nsamples = 7059
\nvalue = [5445, 5773]\nclass = No'),
 Text(1257.4647887323945, 1708.457142857143, 'CO <= 0.35\ngini = 0.168\nsamples = 34
58\nvalue = [4990, 509]\nclass = Yes'),
 Text(723.0422535211268, 1397.8285714285716, 'NO <= 2.5\ngini = 0.109\nsamples = 315
9\nvalue = [4720, 291]\nclass = Yes'),
 Text(345.80281690140845, 1087.2, 'CO <= 0.25\ngini = 0.065\nsamples = 2303\nvalue =
[3540, 123]\nclass = Yes'),
 Text(125.74647887323944, 776.5714285714287, 'CO <= 0.15\ngini = 0.048\nsamples = 21
64\nvalue = [3359, 84]\nclass = Yes'),
 Text(62.87323943661972, 465.9428571428573, 'gini = 0.0\nsamples = 368\nvalue = [58
8, 0]\nclass = Yes'),
 Text(188.61971830985917, 465.9428571428573, 'O_3 <= 50.5\ngini = 0.057\nsamples = 1
796\nvalue = [2771, 84]\nclass = Yes'),
 Text(125.74647887323944, 155.3142857142857, 'gini = 0.169\nsamples = 366\nvalue =
[544, 56]\nclass = Yes'),
 Text(251.49295774647888, 155.3142857142857, 'gini = 0.025\nsamples = 1430\nvalue =
[2227, 28]\nclass = Yes'),
 Text(565.8591549295775, 776.5714285714287, 'BEN <= 0.15\ngini = 0.292\nsamples = 13
```

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

```
Linear: 0.8029944802406465
Lasso: 0.4077775855828065
Ridge: 0.8030115171965686
ElasticNet: 0.7001824428916614
Logistic: 0.5187188019966722
Random Forest: 0.9561419147798181
```

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	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN	28079004
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	28079008
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0	28079011
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN	28079016
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN	28079056
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN	28079057
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN	28079058
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN	28079059
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN	28079060

209496 rows × 14 columns

In [40]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209496 entries, 0 to 209495
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209496 non-null  object 
 1   BEN      50755 non-null   float64
 2   CO       85999 non-null   float64
 3   EBE      50335 non-null   float64
 4   NMHC     25970 non-null   float64
 5   NO       208614 non-null  float64
 6   NO_2     208614 non-null  float64
 7   O_3      121197 non-null  float64
 8   PM10     102892 non-null  float64
 9   PM25     52165 non-null   float64
 10  SO_2     86023 non-null   float64
 11  TCH      25970 non-null   float64
 12  TOL      50662 non-null   float64
 13  station  209496 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

Out[41]:

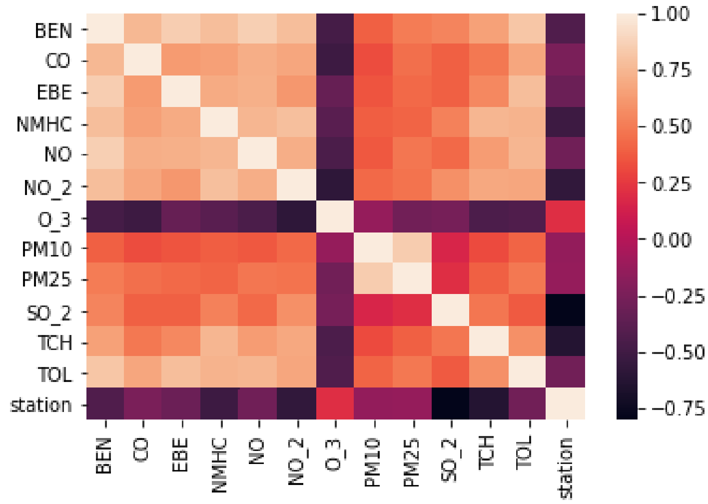
	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	28079008
6	2016-11-01 01:00:00	0.7	0.8	0.4	0.13	57.0	66.0	3.0	23.0	15.0	4.0	1.35	5.0	28079024
25	2016-11-01 02:00:00	2.7	1.0	2.1	0.40	139.0	114.0	4.0	37.0	21.0	14.0	2.30	15.0	28079008
30	2016-11-01 02:00:00	0.7	0.7	0.4	0.13	48.0	59.0	3.0	23.0	15.0	3.0	1.35	5.0	28079024
49	2016-11-01 03:00:00	1.7	0.8	1.4	0.25	53.0	90.0	4.0	31.0	19.0	10.0	1.95	10.7	28079008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209430	2016-06-30 22:00:00	0.1	0.2	0.1	0.02	1.0	5.0	97.0	19.0	12.0	2.0	1.15	0.2	28079024
209449	2016-06-30 23:00:00	0.6	0.4	0.3	0.15	14.0	63.0	54.0	29.0	13.0	16.0	1.48	1.9	28079008
209454	2016-06-30 23:00:00	0.1	0.2	0.1	0.02	1.0	7.0	91.0	16.0	9.0	2.0	1.15	0.3	28079024
209473	2016-07-01 00:00:00	0.6	0.4	0.3	0.16	11.0	68.0	45.0	24.0	14.0	16.0	1.50	1.9	28079008
209478	2016-07-01 00:00:00	0.1	0.2	0.1	0.02	1.0	6.0	89.0	16.0	9.0	2.0	1.15	0.2	28079024

16932 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

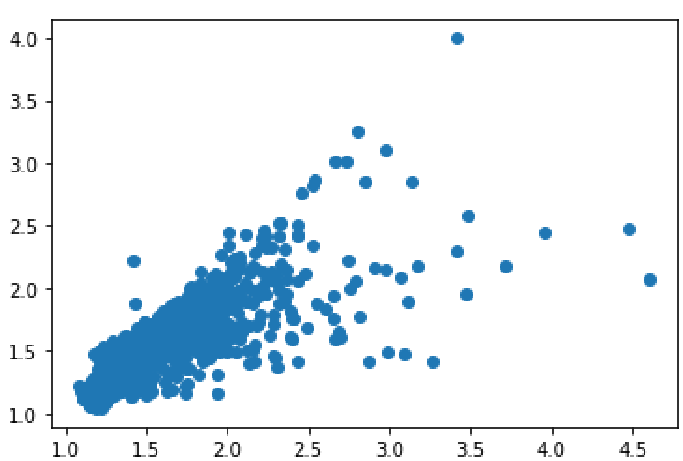
In [45]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[45]: LinearRegression()

In [ ]:

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

Out[46]: <matplotlib.collections.PathCollection at 0x23a1724cd60>



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

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

Out[48]:
```
1.16    757
1.18    701
1.17    683
1.19    618
1.15    577
       ...
4.82      1
2.78      1
3.59      1
3.10      1
4.07      1
Name: TCH, Length: 217, 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    10002
2.0     6930
Name: TCH, dtype: int64
```

In [ ]:

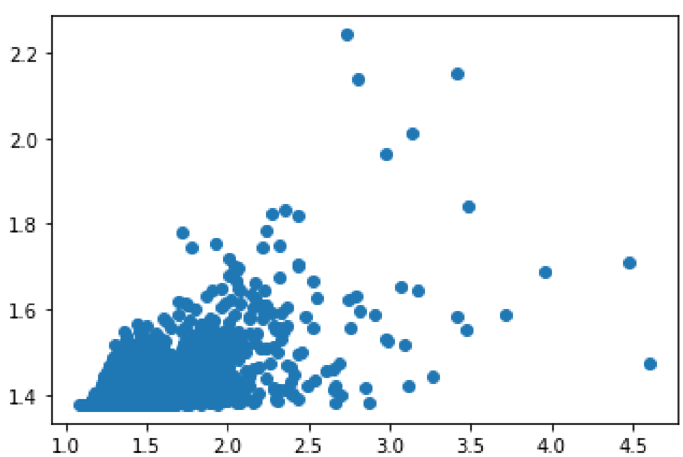
Lasso

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 0x23a16fa9e80>
```



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

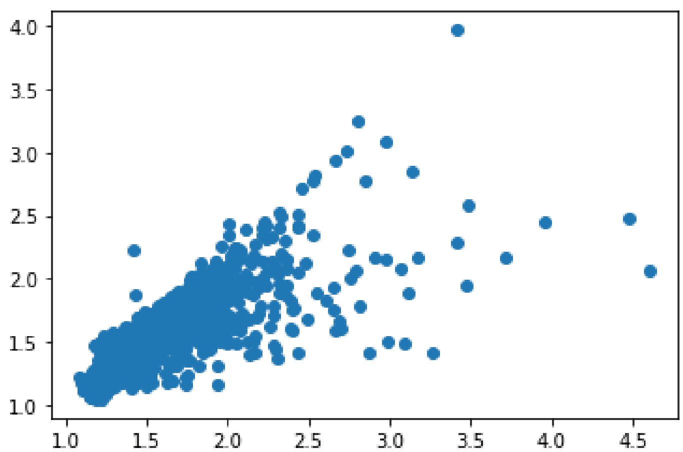
Ridge

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 0x23a16ffef40>



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

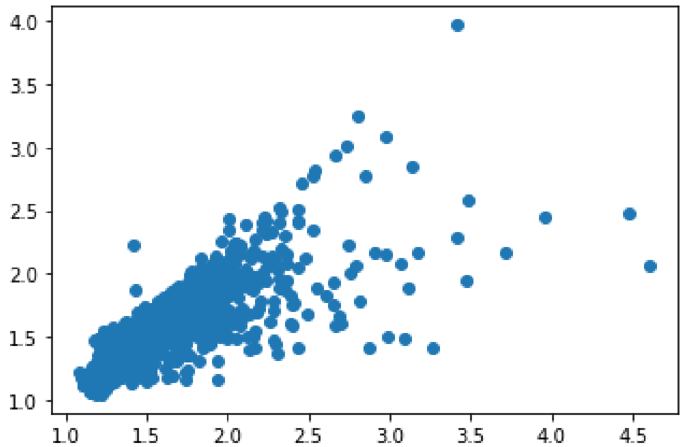
ElasticNet

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 0x23a17022f40>



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

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

```
0.7500531919104193
```

Out[59]: 0.7540860619941899

Logistic

In [60]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[60]:
```
Low     10002
High     6930
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 0x23a170c0ca0>

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(2187.8019801980195, 2019.0857142857144, 'NO_2 <= 37.5\ngini = 0.484\nsamples =
7481\nvalue = [6970, 4882]\nclass = Yes'),
 Text(999.980198019802, 1708.457142857143, 'TOL <= 0.55\ngini = 0.283\nsamples = 407
9\nvalue = [5368, 1102]\nclass = Yes'),
 Text(453.02970297029697, 1397.8285714285716, 'SO_2 <= 3.5\ngini = 0.075\nsamples =
1882\nvalue = [2887, 117]\nclass = Yes'),
 Text(220.99009900990097, 1087.2, 'TOL <= 0.35\ngini = 0.01\nsamples = 1702\nvalue =
[2710, 13]\nclass = Yes'),
 Text(88.39603960396039, 776.5714285714287, 'NO_2 <= 11.5\ngini = 0.003\nsamples = 1
243\nvalue = [2015, 3]\nclass = Yes'),
 Text(44.198019801980195, 465.9428571428573, 'gini = 0.0\nsamples = 1100\nvalue = [1
803, 0]\nclass = Yes'),
 Text(132.59405940594058, 465.9428571428573, 'TOL <= 0.25\ngini = 0.028\nsamples = 1
43\nvalue = [212, 3]\nclass = Yes'),
 Text(88.39603960396039, 155.3142857142857, 'gini = 0.0\nsamples = 67\nvalue = [105,
0]\nclass = Yes'),
 Text(176.79207920792078, 155.3142857142857, 'gini = 0.053\nsamples = 76\nvalue = [1
07, 3]\nclass = Yes'),
 Text(353.58415841584156, 776.5714285714287, 'NO_2 <= 29.5\ngini = 0.028\nsamples =
```

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

```
Linear: 0.7499272637946903
Lasso: 0.19751618918213187
Ridge: 0.7500531919104193
ElasticNet: 0.5703555451049473
Logistic: 0.5938976377952756
Random Forest: 0.9190010124873439
```

Best model is Random Forest

In [ ]: