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,Ridg
from sklearn.model_selection import train_test_split
```

In [2]:
```python
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_200
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.00
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.59
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.09
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.77
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.18
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.88
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.80
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.77
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.88
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.36

217872 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

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

Out[4]:

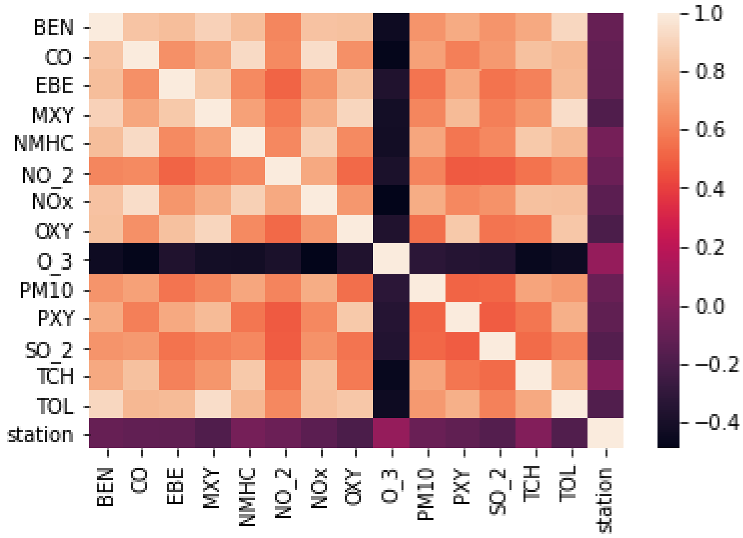
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.100000	0.07	56.250000	75.169998	2.11	42.160000
5	2001-08-01 01:00:00	2.11	0.63	2.48	5.940000	0.05	66.260002	118.099998	3.15	33.500000
21	2001-08-01 01:00:00	0.80	0.43	0.71	1.200000	0.10	27.190001	29.700001	0.76	56.990002
23	2001-08-01 01:00:00	1.29	0.34	1.41	3.090000	0.07	40.750000	51.570000	1.70	51.580002
25	2001-08-01 02:00:00	0.87	0.06	0.88	2.410000	0.01	29.709999	31.440001	1.20	56.520000
...	...	...	...	...	...	...	...	...	...	...
217829	2001-03-31 23:00:00	11.76	4.48	7.71	17.219999	0.89	103.900002	548.500000	7.62	9.680000
217847	2001-03-31 23:00:00	9.79	2.65	7.59	9.730000	0.46	91.320000	315.899994	3.75	6.660000
217849	2001-04-01 00:00:00	5.86	1.22	5.66	13.710000	0.25	64.370003	218.300003	6.46	7.480000
217853	2001-04-01 00:00:00	14.47	1.83	11.39	26.059999	0.33	84.230003	259.200012	11.39	5.440000
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.040000	0.18	76.809998	206.300003	5.20	8.340000

29669 rows × 16 columns

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

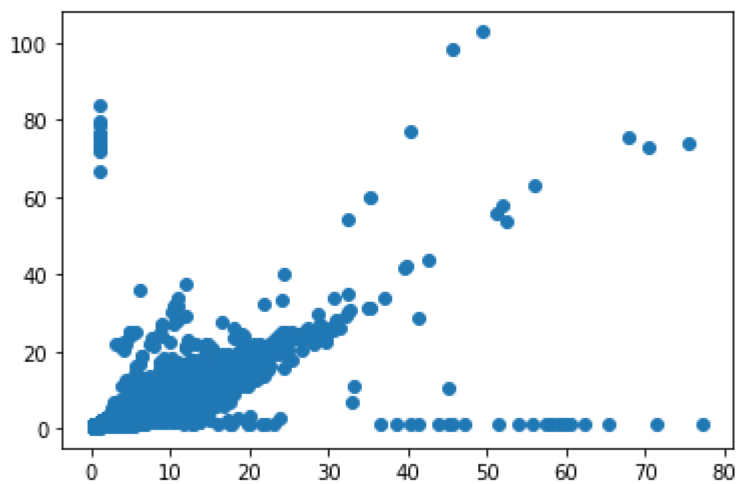
In [6]:
```python
sns.heatmap(df1.corr())
```

Out[6]: <AxesSubplot:>



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

Out[7]: [<matplotlib.lines.Line2D at 0x1ec59f55370>]



In [8]:
```python
data=df[["EBE","PXY"]]
```

In [9]:
```python
# sns.stripplot(x=df["EBE"],y=df["PXY"],jitter=True,marker='o',color='blue')
```

In [10]:
```python
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)
```

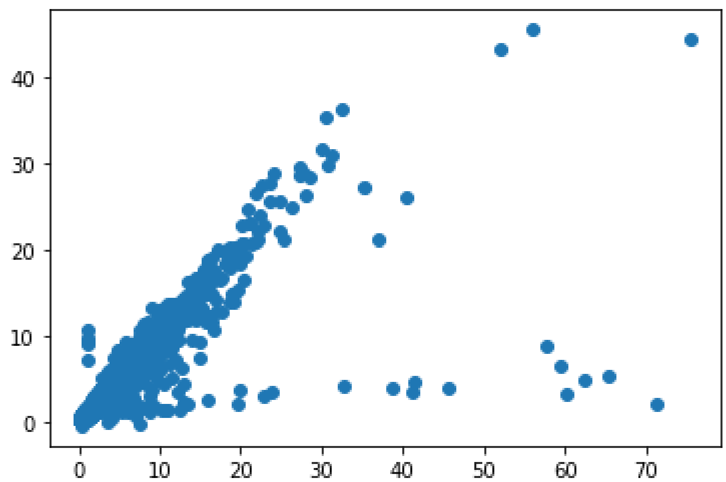
LINEAR

In [11]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[11]:
```
LinearRegression()
```

In [12]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[12]:
```
<matplotlib.collections.PathCollection at 0x1ec5a01f340>
```



In [13]:
```
lis=li.score(x_test,y_test)
```

In [14]:
```
df1["TCH"].value_counts()
```

Out[14]:
```
1.28    988
1.32    938
1.33    908
1.29    908
1.27    905
       ...
4.39      1
3.57      1
4.37      1
3.59      1
4.21      1
Name: TCH, Length: 269, dtype: int64
```

In [15]:
```
df1.loc[df1["TCH"]<1.40,"TCH"]=1
df1.loc[df1["TCH"]>1.40,"TCH"]=2
df1["TCH"].value_counts()
```

Out[15]:
```
1.0    17204
2.0    12465
Name: TCH, dtype: int64
```

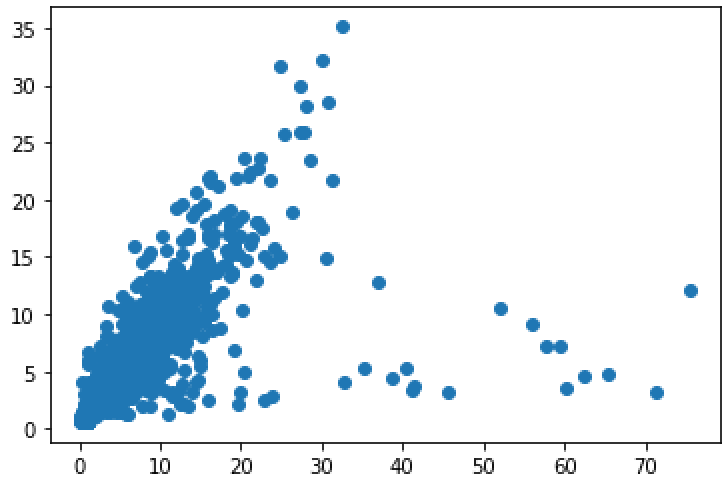
In [ ]:
```
# Lasso
```

In [16]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[16]:
```
Lasso(alpha=5)
```

In [17]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1ec5a085a90>
```



In [18]:
```
las=la.score(x_test,y_test)
```

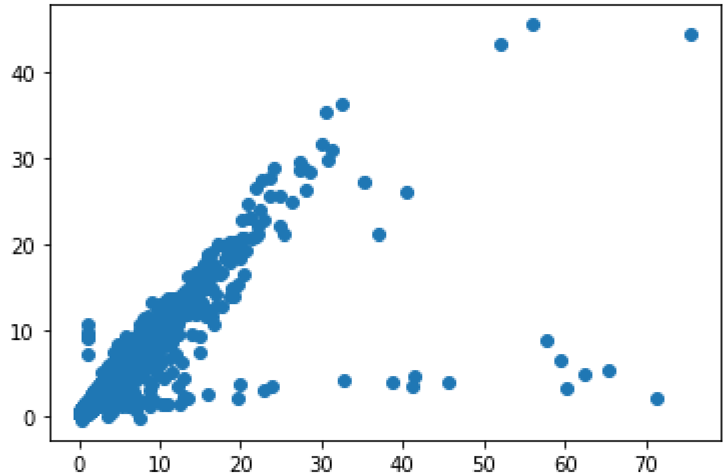
RIDGE

In [19]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[19]:
```
Ridge(alpha=1)
```

In [20]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[20]: <matplotlib.collections.PathCollection at 0x1ec59f2c370>



In [21]:
```
rrs=rr.score(x_test,y_test)
```

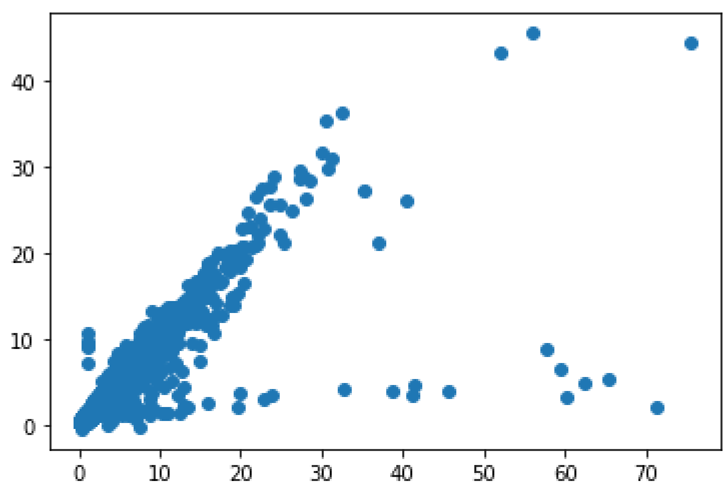
ElasticNet

In [22]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[22]: ElasticNet()

In [23]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[23]: <matplotlib.collections.PathCollection at 0x1ec5a504730>



In [24]:
```
ens=en.score(x_test,y_test)
```

In [25]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.7275996609918448
```

Out[25]:
```
0.7874511769269894
```

LOGISTIC

In [26]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["TCH"].value_counts()
```

Out[26]:
```
Low     17204
High    12465
Name: TCH, dtype: int64
```

In [27]:
```
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 [28]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[28]:
```
LogisticRegression()
```

In [29]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x1ec5a280f70>
```

In [30]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [31]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [32]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df1=df1.replace(g1)
```

In [33]:
```
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 [34]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[34]:
```
RandomForestClassifier()
```

In [35]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [36]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accur
grid_search.fit(x_train,y_train)
```

Out[36]:
```
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 [37]:
```
rfcs=grid_search.best_score_
```

In [38]:
```
rfc_best=grid_search.best_estimator_
```

In [39]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes',"N
```

Out[39]:
```
[Text(2147.4545454545455, 2019.0857142857144, 'NMHC <= 0.195\ngini = 0.486
\nsamples = 13189\nvalue = [12124, 8644]\nclass = Yes'),
 Text(868.0, 1708.457142857143, 'O_3 <= 11.065\ngini = 0.235\nsamples = 856
1\nvalue = [11653, 1834]\nclass = Yes'),
 Text(248.0, 1397.8285714285716, 'NO_2 <= 30.61\ngini = 0.499\nsamples = 11
74\nvalue = [942, 883]\nclass = Yes'),
 Text(90.18181818181819, 1087.2, 'O_3 <= 2.755\ngini = 0.287\nsamples = 41
\nvalue = [13, 62]\nclass = No'),
 Text(45.09090909090909, 776.5714285714287, 'gini = 0.0\nsamples = 26\nvalu
e = [0, 53]\nclass = No'),
 Text(135.27272727272728, 776.5714285714287, 'gini = 0.483\nsamples = 15\nv
alue = [13, 9]\nclass = Yes'),
 Text(405.81818181818187, 1087.2, 'station <= 28079015.0\ngini = 0.498\nsam
ples = 1133\nvalue = [929, 821]\nclass = Yes'),
 Text(225.45454545454547, 776.5714285714287, 'MXY <= 12.15\ngini = 0.408\ns
amples = 241\nvalue = [262, 105]\nclass = Yes'),
 Text(135.27272727272728, 465.9428571428573, 'NO_2 <= 55.815\ngini = 0.292
\nsamples = 144\nvalue = [181, 39]\nclass = Yes'),
 Text(90.18181818181819, 155.3142857142857, 'gini = 0.464\nsamples = 26\nva
```

In [40]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.7276001378416694
Lasso: 0.5989676598608343
Ridge: 0.7275996609918448
ElasticNet: 0.7108933071115187
Logistic: 0.5810583080552747
Random Forest: 0.9137134052388289
```

Best Model is Random Forest

madrid_2002

In [43]:
```
df2=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_20
df2
```

Out[43]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002
...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000

217296 rows × 16 columns

In [44]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217296 entries, 0 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217296 non-null  object 
 1   BEN      66747 non-null   float64
 2   CO       216637 non-null  float64
 3   EBE      58547 non-null   float64
 4   MXY      41255 non-null   float64
 5   NMHC     87045 non-null   float64
 6   NO_2     216439 non-null  float64
 7   NOx      216439 non-null  float64
 8   OXY      41314 non-null   float64
 9   O_3      216726 non-null  float64
 10  PM10     209113 non-null  float64
 11  PXY      41256 non-null   float64
 12  SO_2     216507 non-null  float64
 13  TCH      87115 non-null   float64
 14  TOL      66619 non-null   float64
 15  station  217296 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.5+ MB
```

In [45]:
```
df3=df2.dropna()
df3
```

Out[45]:

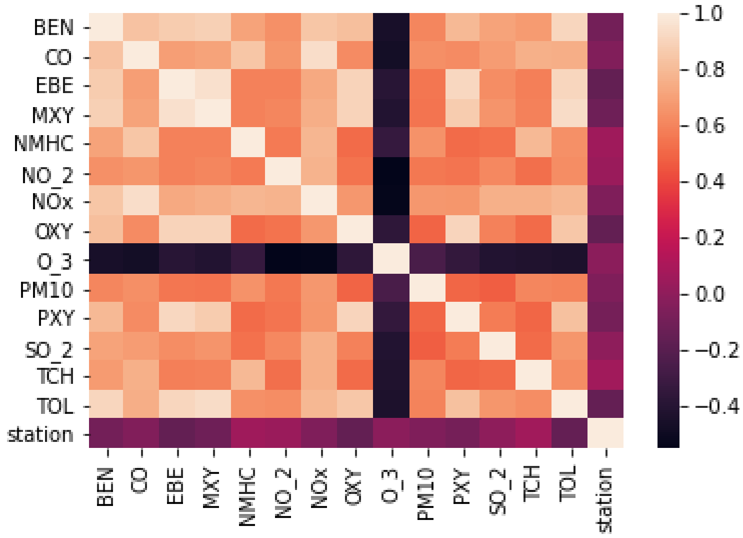
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.37	27.450001
22	2002-04-01 01:00:00	2.02	0.80	1.57	3.66	0.15	93.860001	101.300003	1.77	6.99	33.000000
24	2002-04-01 01:00:00	3.02	1.04	2.43	5.38	0.21	103.699997	195.399994	2.15	14.04	37.310001
26	2002-04-01 02:00:00	2.02	0.53	2.24	5.97	0.12	91.599998	136.199997	2.55	6.76	19.980000
...	...	...	...	...	...	...	...	...	...	...	...
217269	2002-10-31 23:00:00	1.24	0.28	1.26	2.64	0.11	60.080002	64.160004	1.23	15.64	13.910000
217271	2002-10-31 23:00:00	3.13	1.30	2.93	7.90	0.28	84.779999	184.000000	2.23	7.94	32.529999
217273	2002-11-01 00:00:00	2.50	0.97	3.63	9.95	0.19	61.759998	132.100006	4.46	5.45	29.500000
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000

32381 rows × 16 columns

In [46]:
```
df3=df3.drop(["date"],axis=1)
```

In [47]:
```
sns.heatmap(df3.corr())
```

Out[47]: <AxesSubplot:>



In [48]:
```
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)
```

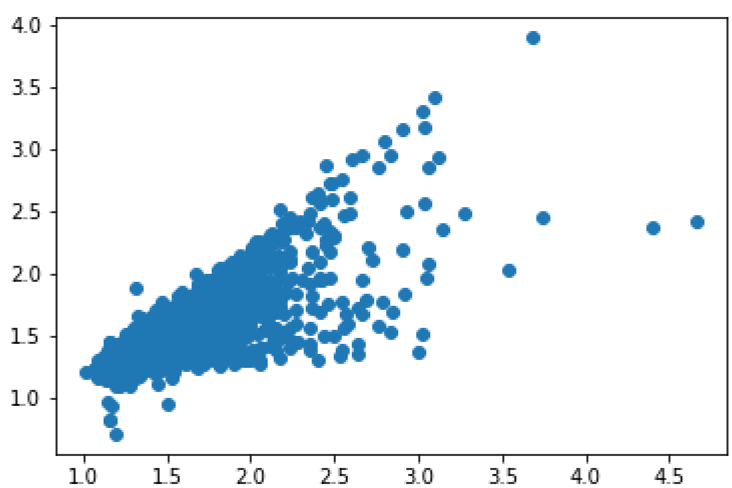
Linear

In [49]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[49]: LinearRegression()

In [50]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[50]: <matplotlib.collections.PathCollection at 0x1ec5b6674c0>



In [51]:
```
lis=li.score(x_test,y_test)
```

In [52]:
```
df3["TCH"].value_counts()
```

Out[52]:
```
1.29    1318
1.30    1253
1.27    1244
1.28    1232
1.31    1187
        ...
2.51       1
4.66       1
2.63       1
3.19       1
3.34       1
Name: TCH, Length: 232, dtype: int64
```

In [53]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[53]:
```
1.0    21925
2.0    10456
Name: TCH, dtype: int64
```

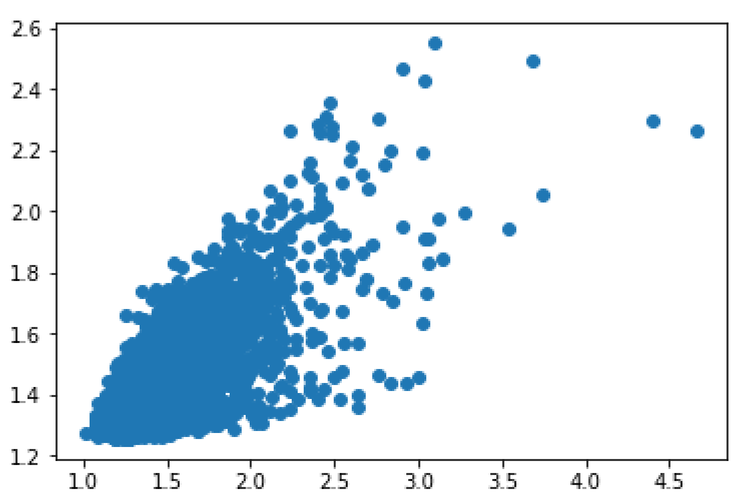
Lasso

In [54]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[54]:
```
Lasso(alpha=5)
```

In [55]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[55]:
```
<matplotlib.collections.PathCollection at 0x1ec5b6b8df0>
```



In [56]:
```
las=la.score(x_test,y_test)
```

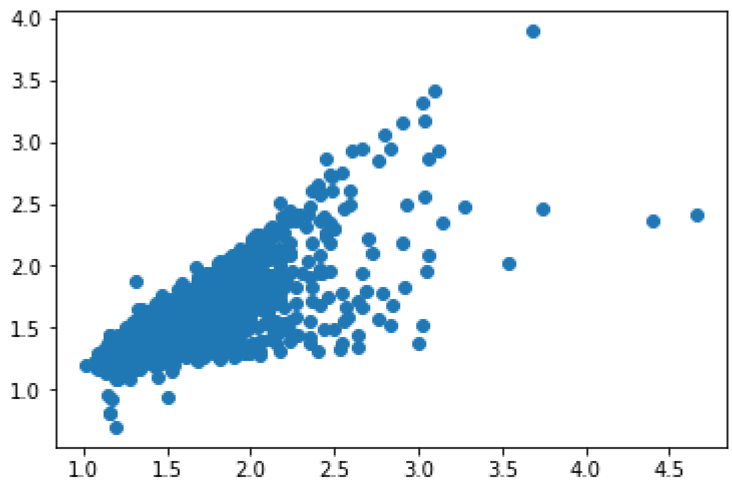
Ridge

In [57]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[57]: Ridge(alpha=1)

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

Out[58]: <matplotlib.collections.PathCollection at 0x1ec5b715a60>



In [59]:
```
rrs=rr.score(x_test,y_test)
```

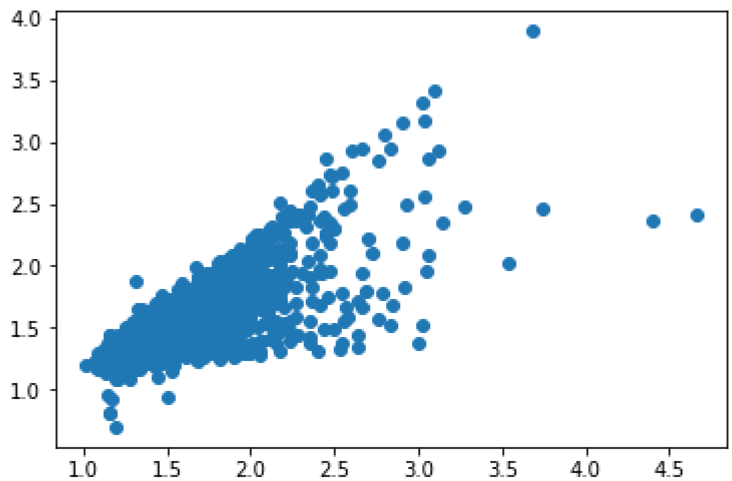
ElasticNet

In [60]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[60]: ElasticNet()

In [61]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[61]: <matplotlib.collections.PathCollection at 0x1ec5b778130>



In [62]:
```
ens=en.score(x_test,y_test)
```

In [63]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.7027639850503987
```

Out[63]: 0.7125089793152224

Logistic

In [64]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[64]:
```
Low     21925
High    10456
Name: TCH, dtype: int64
```

In [65]:
```
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 [66]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[66]: LogisticRegression()

In [67]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[67]: <matplotlib.collections.PathCollection at 0x1ec5b7d8400>

In [68]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [69]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [70]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [71]:
```
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 [72]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[72]: RandomForestClassifier()

In [73]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [74]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accur
grid_search.fit(x_train,y_train)
```

Out[74]:
```
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 [75]:
```
rfcs=grid_search.best_score_
```

In [76]:
```
rfc_best=grid_search.best_estimator_
```

In [77]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes',"N
```

```
6\nvalue = [148, 103]\nclass = Yes'),
 Text(180.0, 155.3142857142857, 'gini = 0.498\nsamples = 96\nvalue = [77, 8
7]\nclass = No'),
 Text(252.0, 155.3142857142857, 'gini = 0.3\nsamples = 60\nvalue = [71, 16]
\nclass = Yes'),
 Text(432.0, 776.5714285714287, 'station <= 28079030.0\ngini = 0.372\nsampl
es = 441\nvalue = [165, 504]\nclass = No'),
 Text(360.0, 465.9428571428573, 'BEN <= 2.435\ngini = 0.224\nsamples = 334
\nvalue = [65, 442]\nclass = No'),
 Text(324.0, 155.3142857142857, 'gini = 0.269\nsamples = 236\nvalue = [58,
305]\nclass = No'),
 Text(396.0, 155.3142857142857, 'gini = 0.092\nsamples = 98\nvalue = [7, 13
7]\nclass = No'),
 Text(504.0, 465.9428571428573, 'NMHC <= 0.245\ngini = 0.472\nsamples = 107
\nvalue = [100, 62]\nclass = Yes'),
 Text(468.0, 155.3142857142857, 'gini = 0.417\nsamples = 92\nvalue = [100,
42]\nclass = Yes'),
 Text(540.0, 155.3142857142857, 'gini = 0.0\nsamples = 15\nvalue = [0, 20]
\nclass = No'),
 Text(864.0, 1087.2, 'TOL <= 14.08\ngini = 0.479\nsamples = 432\nvalue = [2
```

In [78]:
```python
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.7027689416160163
Lasso: 0.5332187392608769
Ridge: 0.7027639850503987
ElasticNet: 0.587420509826319
Logistic: 0.680905815748842
Random Forest: 0.8953939821759463
```

In [ ]: