madrid_2007

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,Ridg
from sklearn.model_selection import train_test_split
```

In [2]:
```
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	2007-12-01 01:00:00	NaN	2.86	NaN	NaN	NaN	282.200012	1054.000000	NaN	4.030000	156.1
1	2007-12-01 01:00:00	NaN	1.82	NaN	NaN	NaN	86.419998	354.600006	NaN	3.260000	80.8
2	2007-12-01 01:00:00	NaN	1.47	NaN	NaN	NaN	94.639999	319.000000	NaN	5.310000	53.0
3	2007-12-01 01:00:00	NaN	1.64	NaN	NaN	NaN	127.900002	476.700012	NaN	4.500000	105.3
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	106.5
...	...	...	...	...	...	...	...	...	...	...	
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999	6.7
225116	2007-03-01 00:00:00	NaN	0.16	NaN	NaN	NaN	46.820000	51.480000	NaN	22.150000	5.7
225117	2007-03-01 00:00:00	0.24	NaN	0.20	NaN	0.09	51.259998	66.809998	NaN	18.540001	13.0
225118	2007-03-01 00:00:00	0.11	NaN	1.00	NaN	0.05	24.240000	36.930000	NaN	NaN	6.6
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000	10.2

225120 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225120 entries, 0 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     225120 non-null  object 
 1   BEN      68885 non-null   float64
 2   CO       206748 non-null  float64
 3   EBE      68883 non-null   float64
 4   MXY      26061 non-null   float64
 5   NMHC     86883 non-null   float64
 6   NO_2     223985 non-null  float64
 7   NOx      223972 non-null  float64
 8   OXY      26062 non-null   float64
 9   O_3      211850 non-null  float64
 10  PM10     222588 non-null  float64
 11  PM25     68870 non-null   float64
 12  PXY      26062 non-null   float64
 13  SO_2     224372 non-null  float64
 14  TCH      87026 non-null   float64
 15  TOL      68845 non-null   float64
 16  station  225120 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.2+ MB
```

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

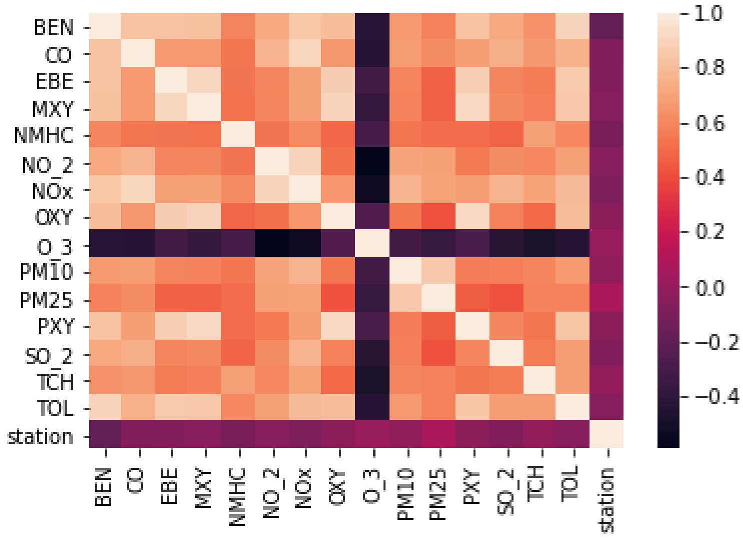
Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	I
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	106.50
21	2007-12-01 01:00:00	1.98	0.31	2.56	6.06	0.35	76.059998	208.899994	1.70	1.000000	37.79
25	2007-12-01 01:00:00	2.82	1.42	3.15	7.02	0.49	123.099998	402.399994	2.60	7.160000	70.80
30	2007-12-01 02:00:00	4.65	1.89	4.41	8.21	0.65	151.000000	622.700012	3.55	58.080002	117.09
47	2007-12-01 02:00:00	1.97	0.30	2.15	5.08	0.33	78.760002	189.800003	1.62	1.000000	34.74
...	...	...	...	...	...	...	...	...	...	...	
225073	2007-02-28 23:00:00	2.12	0.47	2.51	4.99	0.05	43.560001	83.889999	2.57	13.090000	21.86
225094	2007-02-28 23:00:00	0.87	0.45	1.19	2.66	0.13	40.000000	61.959999	1.79	20.440001	15.07
225098	2007-03-01 00:00:00	0.95	0.41	1.55	3.11	0.05	36.090000	63.349998	1.74	17.160000	9.21
225115	2007-03-01 00:00:00	0.30	0.45	1.00	0.30	0.26	8.690000	11.690000	1.00	42.209999	6.76
225119	2007-03-01 00:00:00	0.53	0.40	1.00	1.70	0.12	32.360001	47.860001	1.37	24.150000	10.26

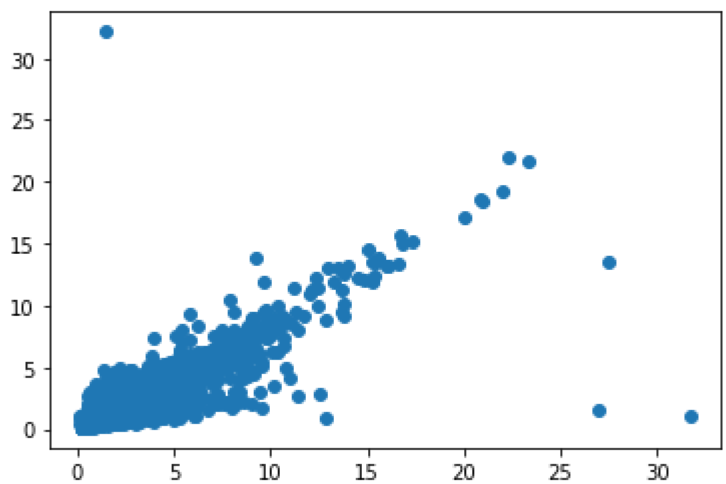
25443 rows × 17 columns

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 0x247361778e0>]



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)
```

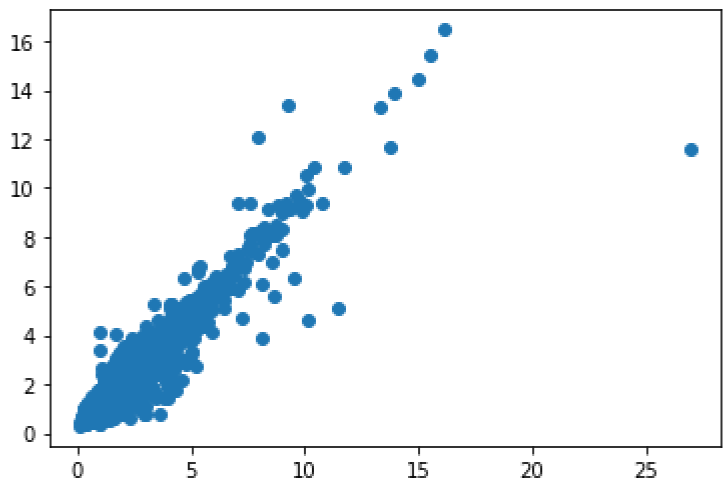
LINEAR

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 0x247361f8130>
```



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

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

Out[14]:
```
1.34    1130
1.33    1067
1.35    1037
1.36    1002
1.32     991
        ...
4.07       1
2.71       1
0.40       1
0.38       1
3.32       1
Name: TCH, Length: 250, 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    14025
2.0    11418
Name: TCH, dtype: int64
```

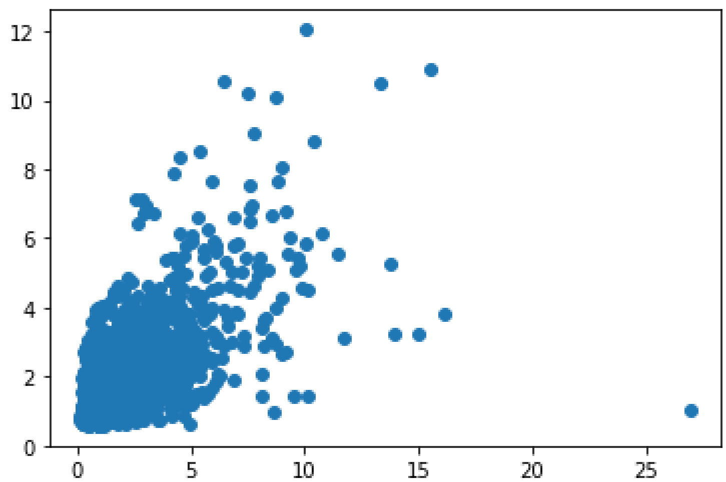
In [16]:
```
# Lasso
```

In [17]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[17]:
```
Lasso(alpha=5)
```

In [18]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x247370a44c0>
```



In [19]:
```
las=la.score(x_test,y_test)
```

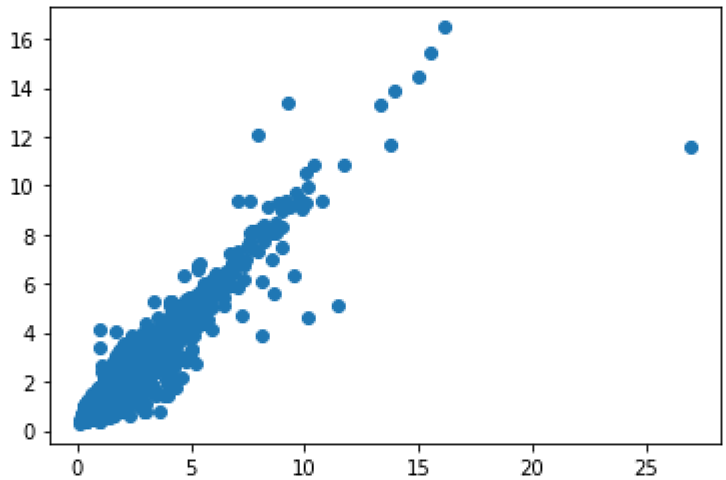
RIDGE

In [20]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[20]:
```
Ridge(alpha=1)
```

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

Out[21]: <matplotlib.collections.PathCollection at 0x2473614b460>



In [22]:
```
rrs=rr.score(x_test,y_test)
```

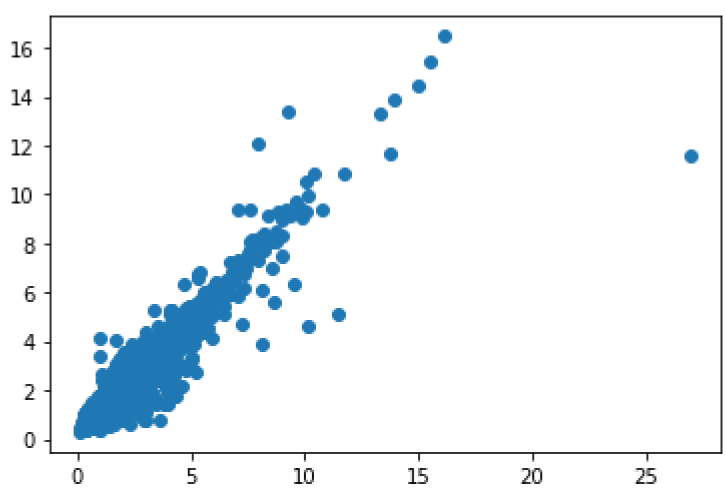
ElasticNet

In [23]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[23]: ElasticNet()

In [24]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[24]: <matplotlib.collections.PathCollection at 0x24737132c40>



In [25]:
```python
ens=en.score(x_test,y_test)
```

In [26]:
```python
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.8868504352876135
```

Out[26]:
```
0.8718454557494321
```

LOGISTIC

In [27]:
```python
g={"TCH":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["TCH"].value_counts()
```

Out[27]:
```
Low     14025
High    11418
Name: TCH, dtype: int64
```

In [28]:
```python
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 [29]:
```python
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[29]:
```
LogisticRegression()
```

In [30]:
```python
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[30]:
```
<matplotlib.collections.PathCollection at 0x24736b8cdf0>
```

In [31]:
```python
los=lo.score(x_test,y_test)
```

Random Forest

In [32]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [33]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df1=df1.replace(g1)
```

In [34]:
```
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 [35]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[35]:
```
RandomForestClassifier()
```

In [36]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [37]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accur
grid_search.fit(x_train,y_train)
```

Out[37]:
```
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 [38]:
```
rfcs=grid_search.best_score_
```

In [39]:
```
rfc_best=grid_search.best_estimator_
```

In [40]:
```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',"N
```

Out[40]:
```
[Text(2385.3243243243246, 2019.0857142857144, 'O_3 <= 16.525\ngini = 0.495
\nsamples = 11230\nvalue = [9794, 8016]\nclass = Yes'),
 Text(1211.5135135135135, 1708.457142857143, 'CO <= 0.795\ngini = 0.234\nsa
mples = 3587\nvalue = [768, 4915]\nclass = No'),
 Text(643.4594594594595, 1397.8285714285716, 'BEN <= 0.595\ngini = 0.325\ns
amples = 2209\nvalue = [719, 2806]\nclass = No'),
 Text(321.72972972972974, 1087.2, 'PXY <= 0.405\ngini = 0.495\nsamples = 26
5\nvalue = [176, 216]\nclass = No'),
 Text(160.86486486486487, 776.5714285714287, 'O_3 <= 8.09\ngini = 0.306\nsa
mples = 50\nvalue = [69, 16]\nclass = Yes'),
 Text(80.43243243243244, 465.9428571428573, 'PXY <= 0.345\ngini = 0.5\nsamp
les = 16\nvalue = [11, 11]\nclass = Yes'),
 Text(40.21621621621622, 155.3142857142857, 'gini = 0.346\nsamples = 7\nval
ue = [2, 7]\nclass = No'),
 Text(120.64864864864865, 155.3142857142857, 'gini = 0.426\nsamples = 9\nva
lue = [9, 4]\nclass = Yes'),
 Text(241.2972972972973, 465.9428571428573, 'PM25 <= 18.455\ngini = 0.146\n
samples = 34\nvalue = [58, 5]\nclass = Yes'),
 Text(201.0810810810811, 155.3142857142857, 'gini = 0.069\nsamples = 29\nva
```

In [41]:
```python
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.8868506867530143
Lasso: 0.47173015549407227
Ridge: 0.8868504352876135
ElasticNet: 0.8232884213400748
Logistic: 0.5465740862046378
Random Forest: 0.8695115103874228
```

Best Model is Random Forest

madrid_2008

In [42]:
```
df2=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_20
df2
```

Out[42]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.889
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.040
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.270
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.850
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.450
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.020
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.540
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.910
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.690

226392 rows × 17 columns

In [43]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 226392 entries, 0 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     226392 non-null  object 
 1   BEN      67047 non-null   float64
 2   CO       208109 non-null  float64
 3   EBE      67044 non-null   float64
 4   MXY      25867 non-null   float64
 5   NMHC     85079 non-null   float64
 6   NO_2     225315 non-null  float64
 7   NOx      225311 non-null  float64
 8   OXY      25878 non-null   float64
 9   O_3      215716 non-null  float64
 10  PM10     220179 non-null  float64
 11  PM25     67833 non-null   float64
 12  PXY      25877 non-null   float64
 13  SO_2     225405 non-null  float64
 14  TCH      85107 non-null   float64
 15  TOL      66940 non-null   float64
 16  station  226392 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.4+ MB
```

In [44]:
```
df3=df2.dropna()
df3
```

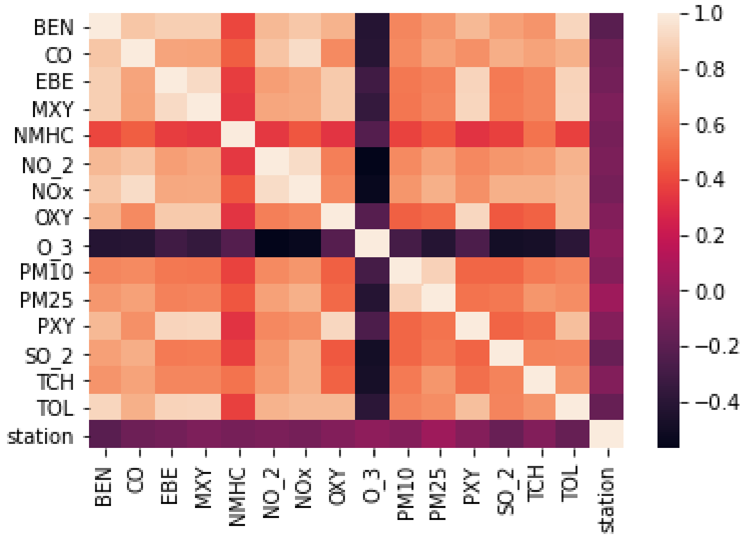
Out[44]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160
21	2008-06-01 01:00:00	0.32	0.37	1.00	0.39	0.33	21.580000	22.180000	1.00	35.770000	7.900
25	2008-06-01 01:00:00	0.73	0.39	1.04	1.70	0.18	64.839996	86.709999	1.31	23.379999	14.760
30	2008-06-01 02:00:00	1.95	0.51	1.98	3.77	0.24	79.750000	143.399994	2.03	18.090000	31.139
47	2008-06-01 02:00:00	0.36	0.39	0.39	0.50	0.34	26.790001	27.389999	1.00	33.029999	7.620
...	...	...	...	...	...	...	...	...	...	...	
226362	2008-10-31 23:00:00	0.47	0.35	0.65	1.00	0.33	22.480000	25.020000	1.00	33.509998	10.200
226366	2008-10-31 23:00:00	0.92	0.46	1.21	2.75	0.19	78.440002	106.199997	1.70	18.320000	14.140
226371	2008-11-01 00:00:00	1.83	0.53	2.22	4.51	0.17	93.260002	158.399994	2.38	18.770000	20.750
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.450
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.690

25631 rows × 17 columns

In [45]:
```
df3=df3.drop(["date"],axis=1)
```

In [46]:
```
sns.heatmap(df3.corr())
```
Out[46]:
```
<AxesSubplot:>
```

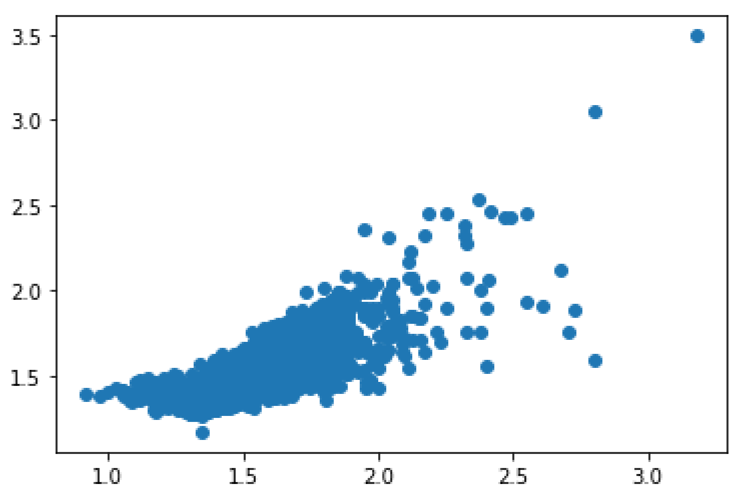


In [47]:
```
x=df3.drop(["TCH"],axis=1)
y=df3["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

Linear

In [48]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```
Out[48]:
```
LinearRegression()
```

In [49]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```
Out[49]:
```
<matplotlib.collections.PathCollection at 0x24737a11fa0>
```



In [50]:
```python
lis=li.score(x_test,y_test)
```

In [51]:
```python
df3["TCH"].value_counts()
```

Out[51]:
```
1.38    1274
1.37    1246
1.36    1243
1.39    1242
1.35    1209
        ...
2.41       1
2.95       1
0.98       1
2.64       1
2.61       1
Name: TCH, Length: 177, dtype: int64
```

In [52]:
```python
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[52]:
```
2.0    12904
1.0    12727
Name: TCH, dtype: int64
```

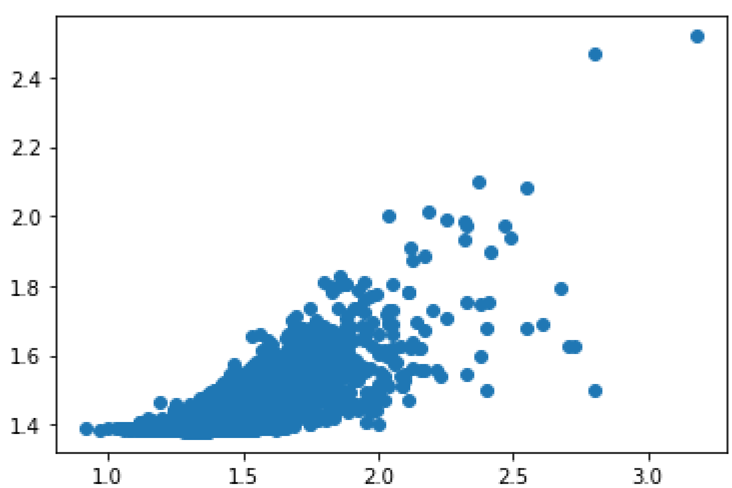
Lasso

In [53]:
```python
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[53]:
```
Lasso(alpha=5)
```

In [54]:
```python
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[54]:
```
<matplotlib.collections.PathCollection at 0x247371ca610>
```



In [55]:
```
las=la.score(x_test,y_test)
```

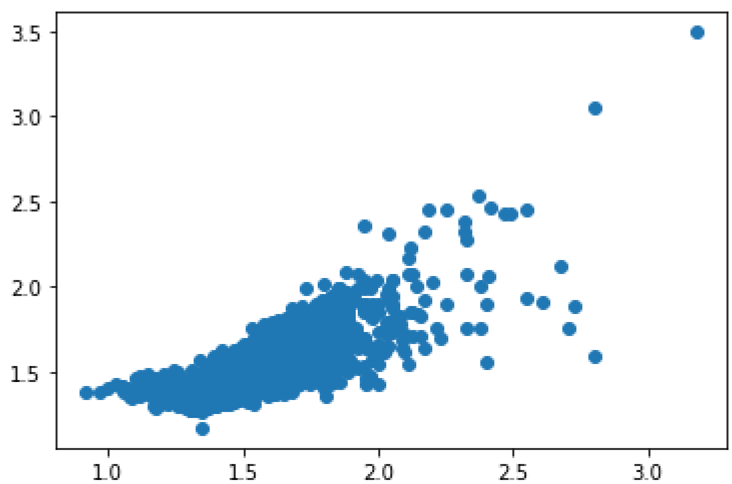
Ridge

In [56]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[56]:
```
Ridge(alpha=1)
```

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

Out[57]:
```
<matplotlib.collections.PathCollection at 0x24737218be0>
```



In [58]:
```
rrs=rr.score(x_test,y_test)
```

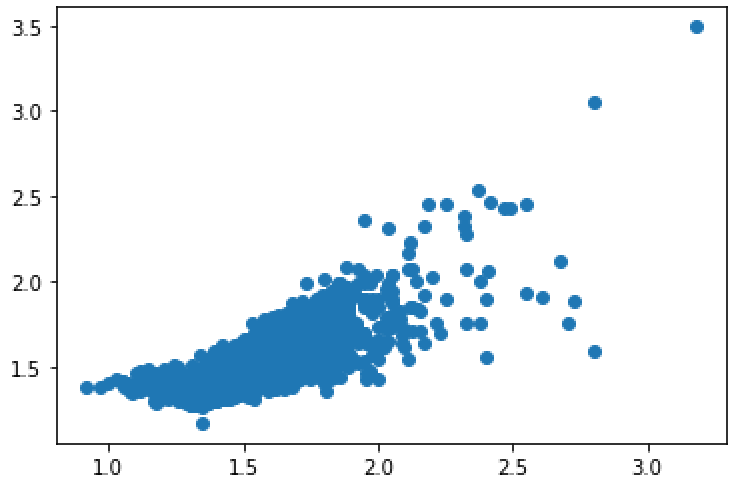
ElasticNet

In [59]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[59]:
```
ElasticNet()
```

In [60]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[60]: <matplotlib.collections.PathCollection at 0x2473726ac70>



In [61]:
```
ens=en.score(x_test,y_test)
```

In [62]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.6788306248393834
```

Out[62]: 0.6510323782465062

Logistic

In [63]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[63]:
```
High    12904
Low     12727
Name: TCH, dtype: int64
```

In [64]:
```
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 [65]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[65]: LogisticRegression()

In [66]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[66]: <matplotlib.collections.PathCollection at 0x24736a44be0>

In [67]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [68]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [69]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [70]:
```
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 [71]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[71]: RandomForestClassifier()

In [72]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [73]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accur
grid_search.fit(x_train,y_train)
```

Out[73]:
```
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')
```

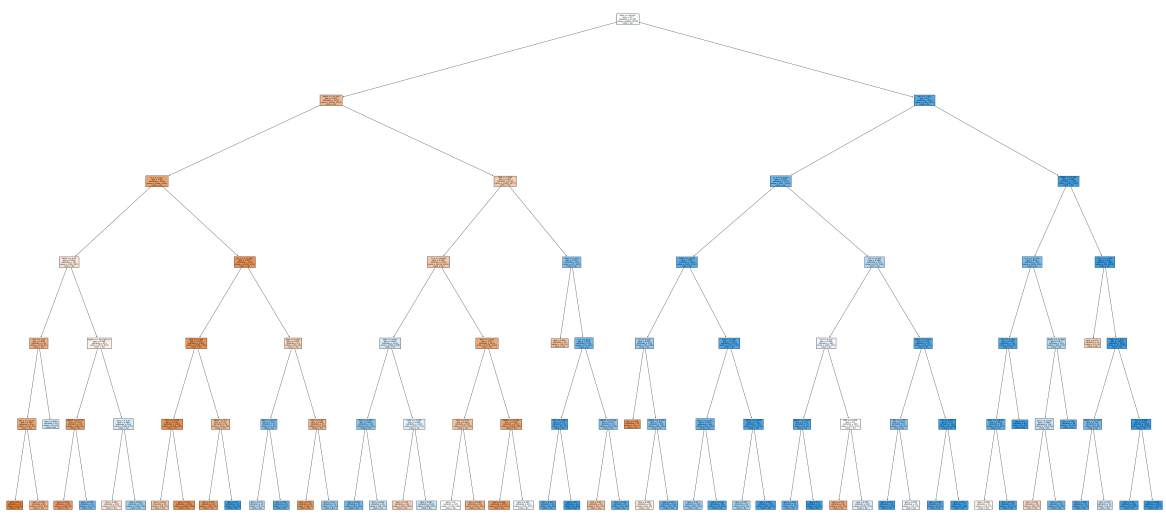
In [74]:
```
rfcs=grid_search.best_score_
```

In [75]:
```
rfc_best=grid_search.best_estimator_
```

In [76]:
```
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
```

```
 Text(4417.5, 155.3142857142857, 'gini = 0.006\nsamples = 1607\nvalue = [7,
2470]\nclass = No')]
```



In [77]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.678849766548985
Lasso: 0.4704282524306074
Ridge: 0.6788306248393834
ElasticNet: 0.5841605950723971
Logistic: 0.506892067620286
Random Forest: 0.8297755793565964
```

In [ ]: