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,ElasticNe
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_2001.csv
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.000000	NaN	6
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	100.599998	1.73	8
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.099998	NaN	7
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.779999	NaN	6
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.180000	NaN	8
...	...	...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.880001	NaN	39
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.809999	NaN	13
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.770000	0.61	14
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.889999	4.31	39
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	35.369999	4.95	27

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]:

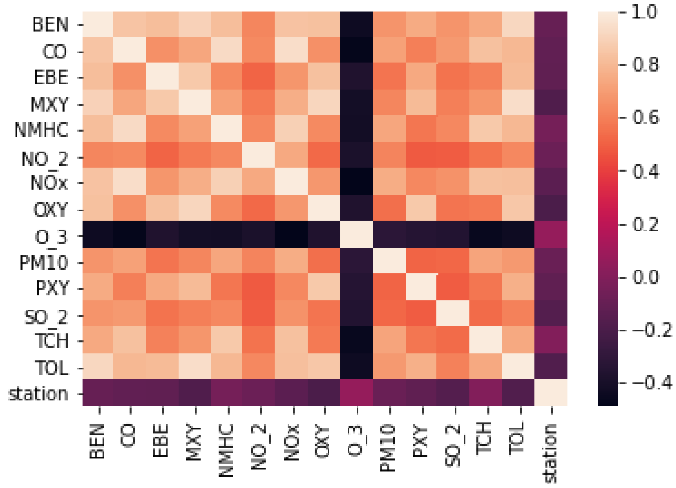
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	P
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.100000	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1
5	2001-08-01 01:00:00	2.11	0.63	2.48	5.940000	0.05	66.260002	118.099998	3.15	33.500000	122.699997	2
21	2001-08-01 01:00:00	0.80	0.43	0.71	1.200000	0.10	27.190001	29.700001	0.76	56.990002	114.300003	0
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	102.199997	1
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	56.290001	1
...	...	...	...	...	...	...	...	...	...	...	...	
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	77.180000	6
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	52.740002	3
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	17.570000	5
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	36.740002	9
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	35.369999	4

29669 rows × 16 columns

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

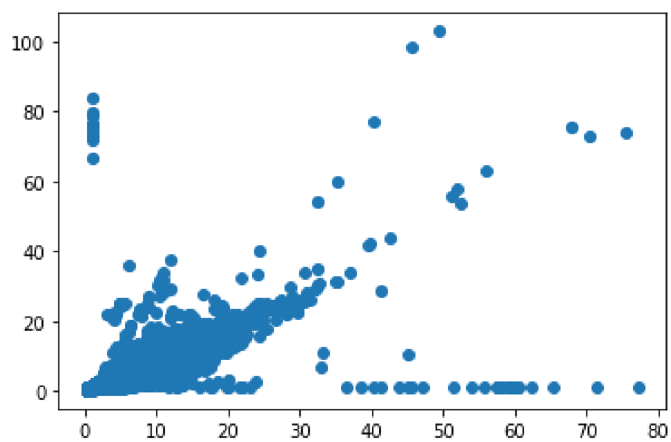
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <AxesSubplot:>



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

Out[7]: [<matplotlib.lines.Line2D at 0x23d88ab14f0>]



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

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

In [41]:
```
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)
```

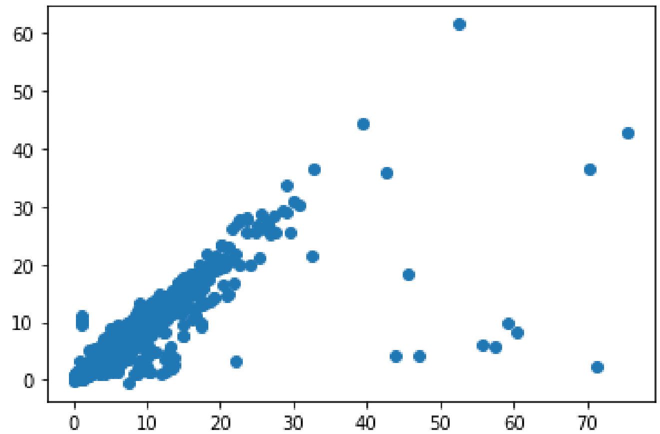
Linear

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 0x23d88b7d460>
```



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
```

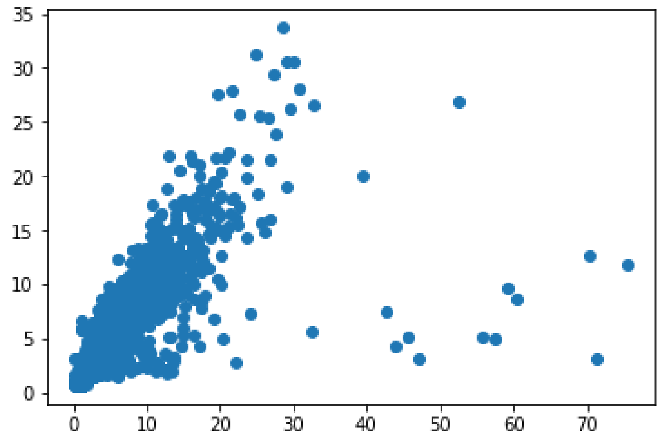
Lasso

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 0x23d88bec670>



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

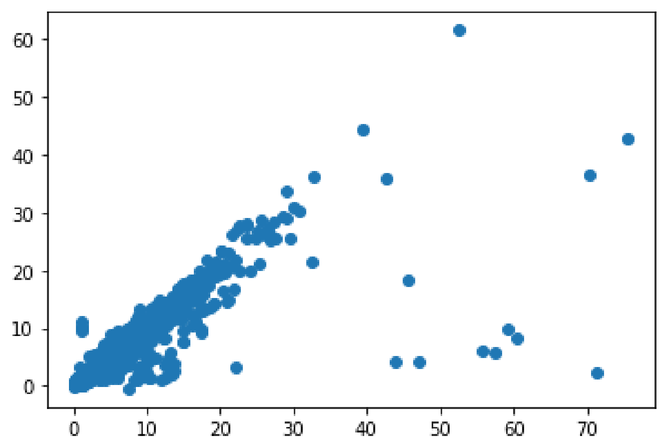
Ridge

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 0x23d88ac8ee0>



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

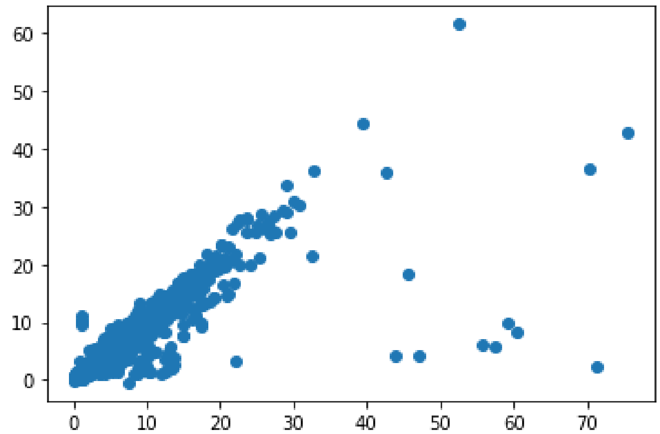
ElasticNet

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 0x23d89e36cd0>



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

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

0.7877953739336674

Out[25]: 0.759499092953202

Logistic

In [43]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["TCH"].value_counts()
```

Out[43]:
```
Low     17204
High    12465
Name: TCH, dtype: int64
```

In [44]:
```
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 [45]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[45]: LogisticRegression()

In [46]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[46]: <matplotlib.collections.PathCollection at 0x23d896d8490>

In [47]:
```
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 [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 [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=
```

```
 Text(1716.923076923077, 155.3142857142857, 'gini = 0.276\nsamples = 278\nvalue = [78,
394]\nclass = No'),
 Text(1793.2307692307693, 155.3142857142857, 'gini = 0.447\nsamples = 116\nvalue = [61,
120]\nclass = No'),
 Text(2136.6153846153848, 1087.2, 'station <= 28079068.0\ngini = 0.293\nsamples = 2294
\nvalue = [3027, 658]\nclass = Yes'),
 Text(1984.0, 776.5714285714287, 'NMHC <= 0.195\ngini = 0.178\nsamples = 1469\nvalue =
[2129, 233]\nclass = Yes'),
 Text(1907.6923076923076, 465.9428571428573, 'NMHC <= 0.135\ngini = 0.141\nsamples = 13
97\nvalue = [2075, 171]\nclass = Yes'),
 Text(1869.5384615384614, 155.3142857142857, 'gini = 0.089\nsamples = 1065\nvalue = [16
29, 80]\nclass = Yes'),
 Text(1945.8461538461538, 155.3142857142857, 'gini = 0.281\nsamples = 332\nvalue = [44
6, 91]\nclass = Yes'),
 Text(2060.3076923076924, 465.9428571428573, 'O_3 <= 33.825\ngini = 0.498\nsamples = 72
\nvalue = [54, 62]\nclass = No'),
 Text(2022.1538461538462, 155.3142857142857, 'gini = 0.284\nsamples = 22\nvalue = [6, 2
9]\nclass = No'),
 Text(2098.4615384615386, 155.3142857142857, 'gini = 0.483\nsamples = 50\nvalue = [48,
```

In [48]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.7877914082809474
Lasso: 0.6608359189741109
Ridge: 0.7877953739336674
ElasticNet: 0.7744562684843173
Logistic: 0.5802718795640939
Random Forest: 0.9163617103235747
```

Best Model is Random Forest

In [49]:
```
df2=pd.read_csv("C:/Users/user/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_2002.cs
df2
```

Out[49]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002	NaN	21.3200
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	2.53	11.6600
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001	NaN	13.6700
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000	NaN	16.9900
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002	NaN	15.2600
...	...	...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000	NaN	13.2100
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	NaN	15.6400
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	0.94	5.6200
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	5.52	24.2199
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	3.35	12.9100

217296 rows × 16 columns

In [50]:
```
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 [51]:
```
df3=df2.dropna()
df3
```

Out[51]:

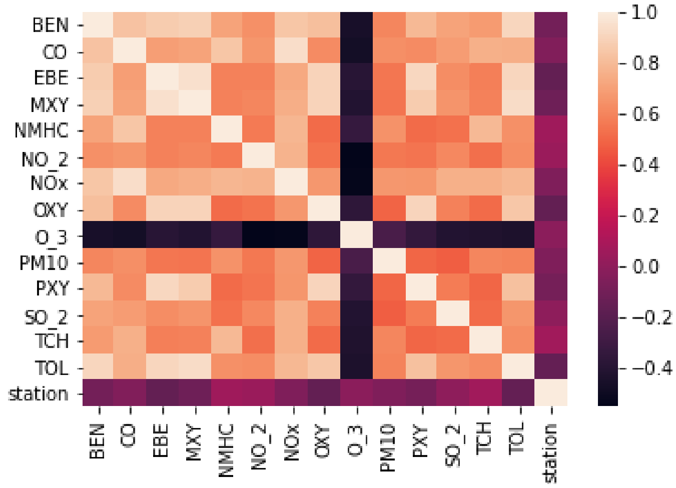
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	T
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.66	1
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.37	27.450001	2.98	14.78	1
22	2002-04-01 01:00:00	2.02	0.80	1.57	3.66	0.15	93.860001	101.300003	1.77	6.99	33.000000	1.48	1.98	1
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	2.18	15.91	1
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	2.45	10.15	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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	0.94	4.31	1
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	3.40	13.66	1
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	3.60	11.00	1
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	0.94	5.62	1
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	3.35	12.91	1

32381 rows × 16 columns

In [52]:
```
df3=df3.drop(["date"],axis=1)
```

In [53]:
```
sns.heatmap(df3.corr())
```

Out[53]: <AxesSubplot:>



In [54]:
```
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

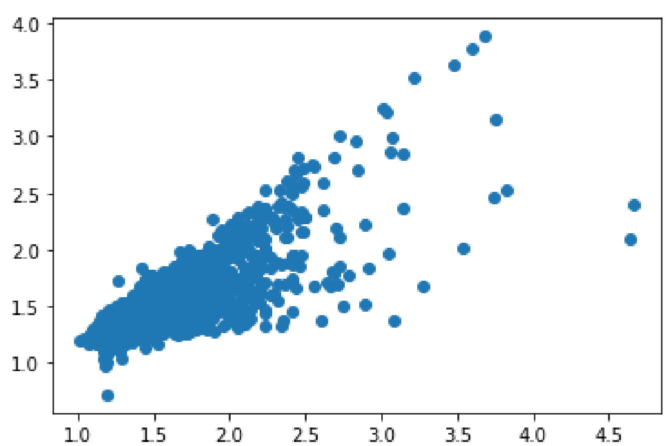
In [55]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[55]: LinearRegression()

In [ ]:

In [56]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[56]: <matplotlib.collections.PathCollection at 0x23d89e9d4f0>



In [57]:
```
lis=li.score(x_test,y_test)
```

In [58]:
```
df3["TCH"].value_counts()
```

Out[58]:
```
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 [59]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[59]:
```
1.0    21925
2.0    10456
Name: TCH, dtype: int64
```

In [ ]:

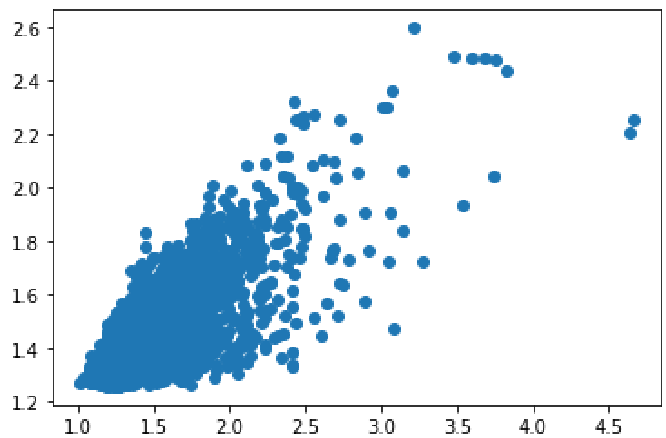
Lasso

In [60]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[60]:
```
Lasso(alpha=5)
```

In [61]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[61]:
```
<matplotlib.collections.PathCollection at 0x23d89ef3e20>
```



In [62]:
```
las=la.score(x_test,y_test)
```

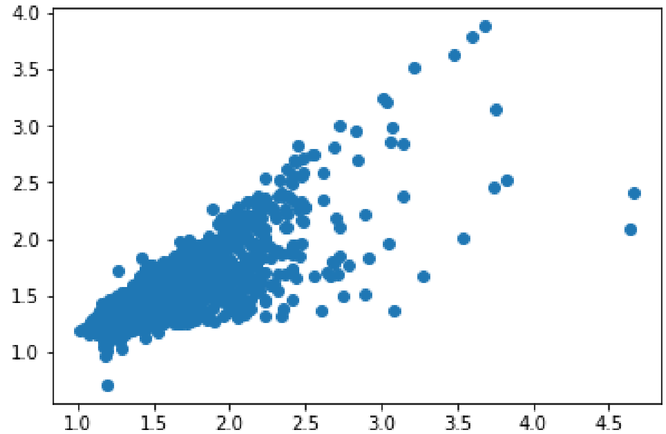
Ridge

In [63]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[63]:
```
Ridge(alpha=1)
```

In [64]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[64]:
```
<matplotlib.collections.PathCollection at 0x23d89f59250>
```



In [65]:
```
rrs=rr.score(x_test,y_test)
```

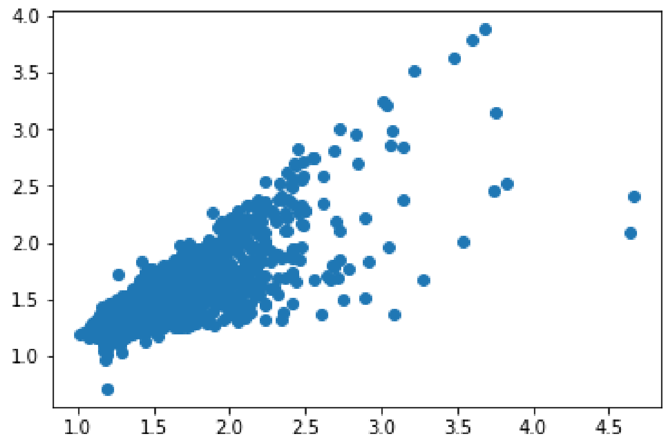
ElasticNet

In [66]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[66]:
```
ElasticNet()
```

In [67]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[67]:
```
<matplotlib.collections.PathCollection at 0x23d88c175b0>
```



In [68]:
```
ens=en.score(x_test,y_test)
```

In [69]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.7234772839066235
```

Out[69]:
```
0.7035718884047415
```

Logistic

In [75]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[75]:
```
Low     21925
High    10456
Name: TCH, dtype: int64
```

In [76]:
```
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 [77]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[77]:
```
LogisticRegression()
```

In [78]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[78]:
```
<matplotlib.collections.PathCollection at 0x23d89413ca0>
```

In [80]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [81]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [82]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [83]:
```
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 [84]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[84]:
```
RandomForestClassifier()
```

In [85]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [86]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[86]:
```
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')
```

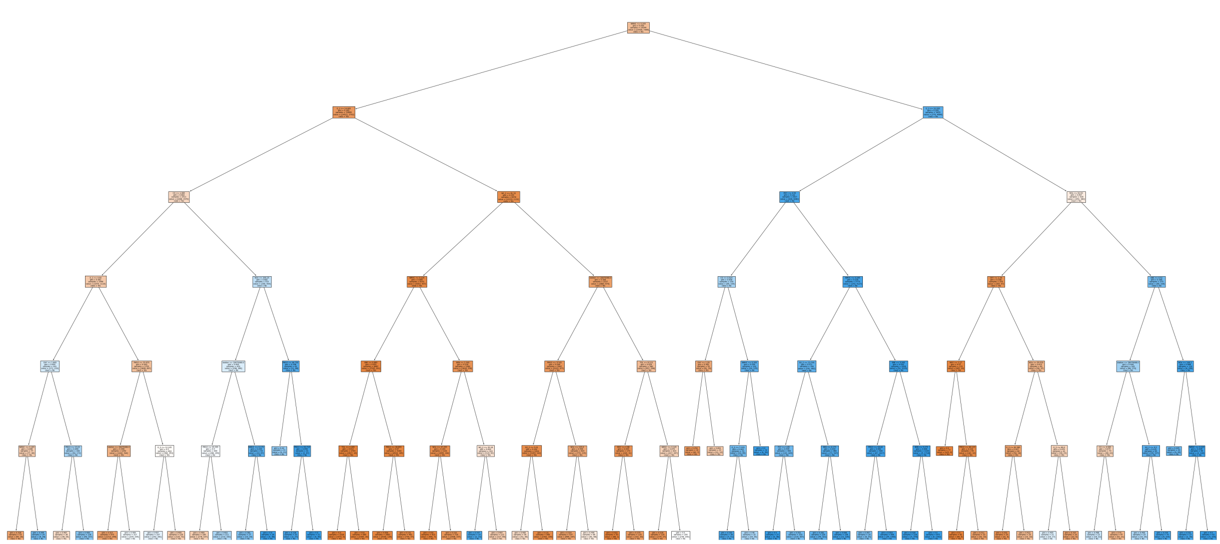
In [87]:
```
rfcs=grid_search.best_score_
```

In [88]:
```
rfc_best=grid_search.best_estimator_
```

In [89]:
```
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=
```

```
\nclass = No')]
```



In [90]:

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

```
Linear: 0.7234015760409874
Lasso: 0.5477773816966718
Ridge: 0.7234772839066235
ElasticNet: 0.6100678624001161
Logistic: 0.687596500257334
Random Forest: 0.8931439159975294
```

Best model is Random Forest

In [ ]: