madrid_2005

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	PM10
0	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	14.720000	14.91
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.93
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	30.469999	14.60
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	21.379999	15.16
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	33.410000	5.00
...	...	...	...	...	...	...	...	...	...	...	...
236995	2006-01-01 00:00:00	1.08	0.36	1.01	NaN	0.11	21.990000	23.610001	NaN	43.349998	5.00
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.95
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	43.840000	4.31
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	NaN	5.00
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.67

237000 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      70370 non-null   float64
 2   CO       217656 non-null  float64
 3   EBE      68955 non-null   float64
 4   MXY      32549 non-null   float64
 5   NMHC     92854 non-null   float64
 6   NO_2     235022 non-null  float64
 7   NOx      235049 non-null  float64
 8   OXY      32555 non-null   float64
 9   O_3      223162 non-null  float64
 10  PM10     232142 non-null  float64
 11  PM25     69407 non-null   float64
 12  PXY      32549 non-null   float64
 13  SO_2     235277 non-null  float64
 14  TCH      93076 non-null   float64
 15  TOL      70255 non-null   float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ MB
```

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

Out[4]:

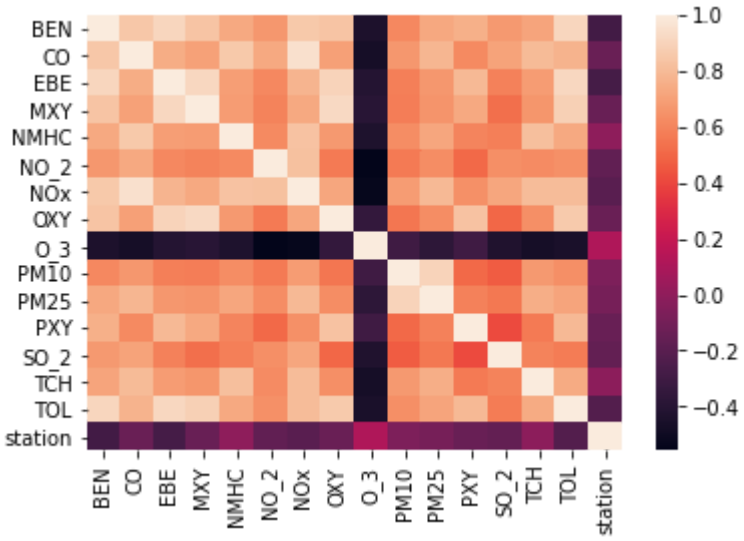
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
5	2005-11-01 01:00:00	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000	18.07
22	2005-11-01 01:00:00	0.30	0.22	0.25	0.59	0.11	18.540001	19.020000	0.67	46.799999	9.88
25	2005-11-01 01:00:00	0.67	0.49	0.94	3.44	0.17	48.740002	74.349998	1.57	23.430000	13.88
31	2005-11-01 02:00:00	3.10	0.84	3.21	6.82	0.22	89.919998	224.199997	3.72	12.390000	28.74
48	2005-11-01 02:00:00	0.39	0.20	0.29	0.68	0.11	16.639999	17.080000	0.40	47.689999	8.78
...	...	...	...	...	...	...	...	...	...	...	...
236970	2005-12-31 23:00:00	0.37	0.39	1.00	1.00	0.10	4.500000	5.550000	1.00	57.779999	8.26
236973	2005-12-31 23:00:00	0.92	0.45	1.26	3.42	0.14	37.250000	49.060001	2.57	31.889999	19.73
236979	2006-01-01 00:00:00	1.00	0.38	1.11	2.35	0.04	35.919998	59.480000	1.39	35.810001	4.22
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.95
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.67

20070 rows × 17 columns

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

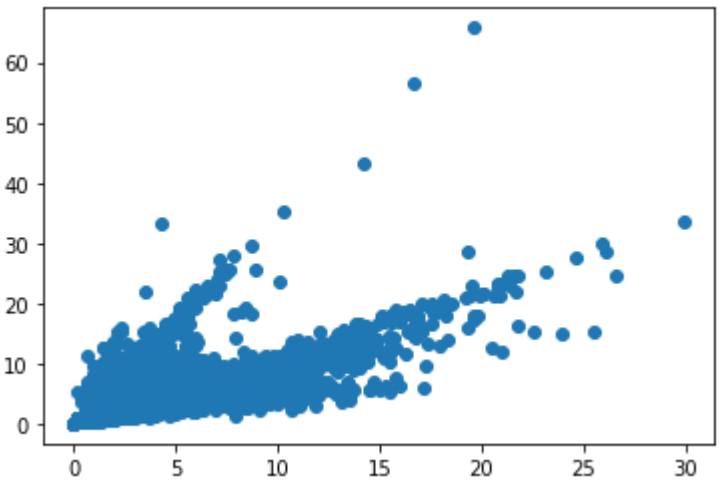
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <AxesSubplot:>



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

Out[7]: [<matplotlib.lines.Line2D at 0x19e808effa0>]



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

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

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

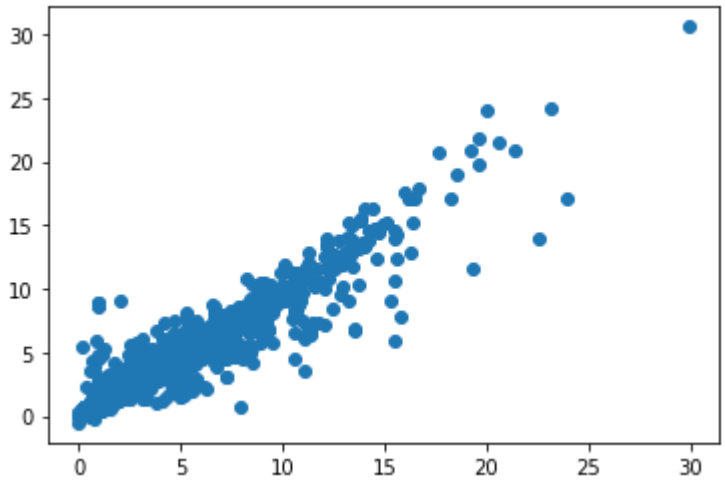
LINEAR

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 0x19e810efa90>
```



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

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

Out[14]:
```
1.31    845
1.33    820
1.28    812
1.30    806
1.34    794
       ...
3.04      1
3.22      1
2.79      1
2.68      1
3.37      1
Name: TCH, Length: 198, 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    12093
2.0     7977
Name: TCH, dtype: int64
```

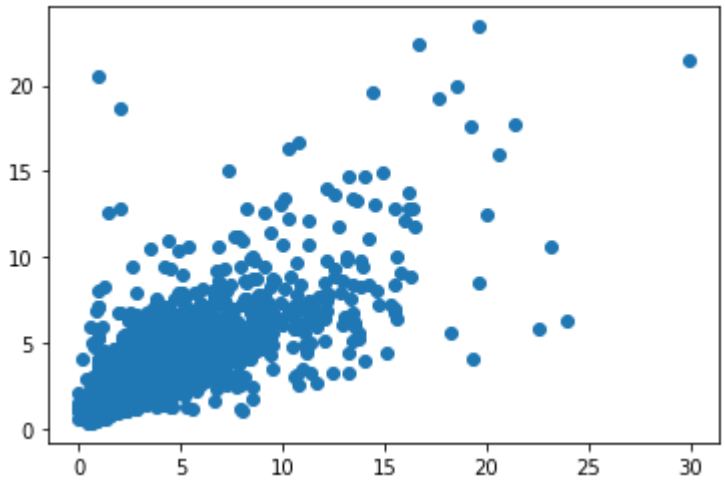
In [16]:
```
# Lasso
```

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 0x19e812c18e0>
```



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

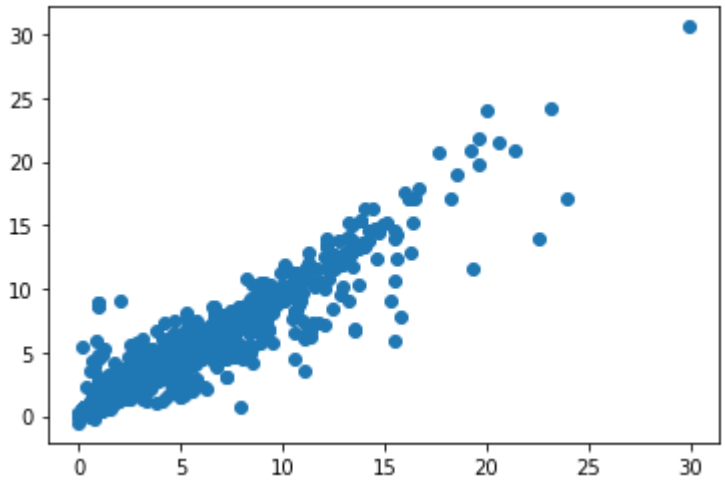
RIDGE

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 0x19e808d4250>
```



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

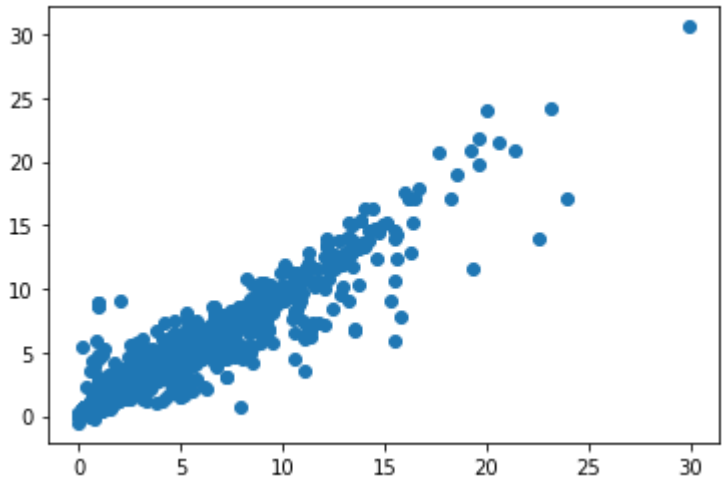
ElasticNet

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 0x19e81352580>
```



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

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

```
0.9154788730039654
```

Out[26]:
```
0.9283188352688306
```

LOGISTIC

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

Out[27]:
```
Low     12093
High     7977
Name: TCH, dtype: int64
```

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

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

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

Out[30]:
```
<matplotlib.collections.PathCollection at 0x19e80e9e550>
```

In [31]:
```
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]:
```
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(2226.8333333333335, 2019.0857142857144, 'NO_2 <= 76.595\ngini = 0.479
\nsamples = 8865\nvalue = [8456, 5593]\nclass = Yes'),
 Text(1260.6666666666667, 1708.457142857143, 'NOx <= 151.15\ngini = 0.315\n
samples = 5743\nvalue = [7342, 1785]\nclass = Yes'),
 Text(661.3333333333334, 1397.8285714285716, 'O_3 <= 13.475\ngini = 0.251\n
samples = 5320\nvalue = [7192, 1243]\nclass = Yes'),
 Text(330.6666666666667, 1087.2, 'CO <= 0.635\ngini = 0.478\nsamples = 518
\nvalue = [322, 495]\nclass = No'),
 Text(165.33333333333334, 776.5714285714287, 'station <= 28079015.0\ngini =
0.499\nsamples = 308\nvalue = [228, 244]\nclass = No'),
 Text(82.66666666666667, 465.9428571428573, 'NMHC <= 0.175\ngini = 0.295\ns
amples = 84\nvalue = [105, 23]\nclass = Yes'),
 Text(41.333333333333336, 155.3142857142857, 'gini = 0.212\nsamples = 77\nv
alue = [102, 14]\nclass = Yes'),
 Text(124.0, 155.3142857142857, 'gini = 0.375\nsamples = 7\nvalue = [3, 9]
\nclass = No'),
 Text(248.0, 465.9428571428573, 'PM25 <= 15.015\ngini = 0.459\nsamples = 22
4\nvalue = [123, 221]\nclass = No'),
 Text(206.66666666666669, 155.3142857142857, 'gini = 0.5\nsamples = 124\nva
```

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

```
Linear: 0.915480628504164
Lasso: 0.6694361110191274
Ridge: 0.9154788730039654
ElasticNet: 0.8859307850054954
Logistic: 0.5915960803853181
Random Forest: 0.9096021267195746
```

Best Model is Random Forest

madrid_2006

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	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.570
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.820
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.419
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.260
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.180
...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.120
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.469
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.680
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.360
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.490

230568 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230568 entries, 0 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     230568 non-null  object 
 1   BEN      73979 non-null   float64
 2   CO       211665 non-null  float64
 3   EBE      73948 non-null   float64
 4   MXY      33422 non-null   float64
 5   NMHC     90829 non-null   float64
 6   NO_2     228855 non-null  float64
 7   NOx      228855 non-null  float64
 8   OXY      33472 non-null   float64
 9   O_3      216511 non-null  float64
 10  PM10     227469 non-null  float64
 11  PM25     61758 non-null   float64
 12  PXY      33447 non-null   float64
 13  SO_2     229125 non-null  float64
 14  TCH      90887 non-null   float64
 15  TOL      73840 non-null   float64
 16  station  230568 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.9+ MB
```

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

Out[44]:

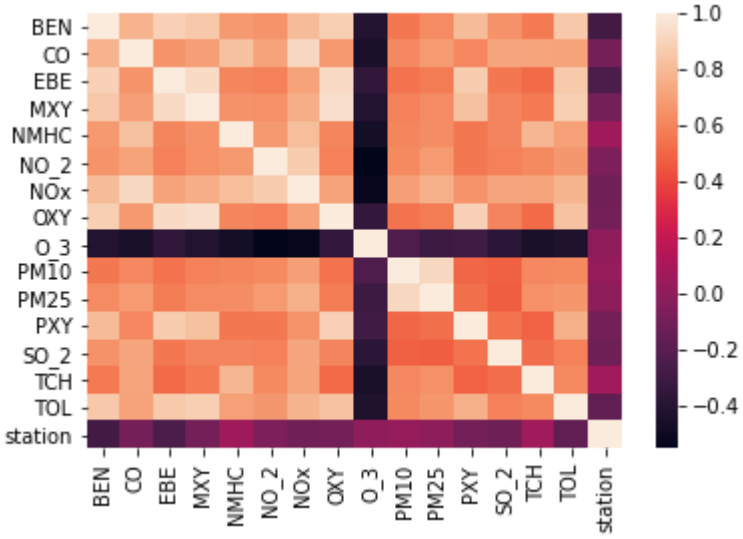
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
5	2006-02-01 01:00:00	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.990000
22	2006-02-01 01:00:00	1.69	0.79	1.24	2.670000	0.17	59.910000	120.199997	1.11	2.450000
25	2006-02-01 01:00:00	2.35	1.47	2.64	9.660000	0.40	117.699997	346.399994	5.15	4.780000
31	2006-02-01 02:00:00	4.39	0.85	7.92	17.139999	0.25	92.059998	237.000000	9.24	5.920000
48	2006-02-01 02:00:00	1.93	0.79	1.24	2.740000	0.16	60.189999	125.099998	1.11	2.280000
...	...	...	...	...	...	...	...	...	...	...
230538	2006-04-30 23:00:00	0.42	0.40	0.37	0.430000	0.10	49.259998	51.689999	1.00	64.599998
230541	2006-04-30 23:00:00	1.63	0.94	1.53	2.200000	0.33	63.220001	211.399994	1.35	17.670000
230547	2006-05-01 00:00:00	3.99	1.06	3.71	7.960000	0.26	202.399994	343.500000	3.92	11.130000
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.090000	0.08	51.900002	54.820000	0.61	48.410000
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.000000	0.24	107.300003	160.199997	2.01	17.730000

24758 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)
```

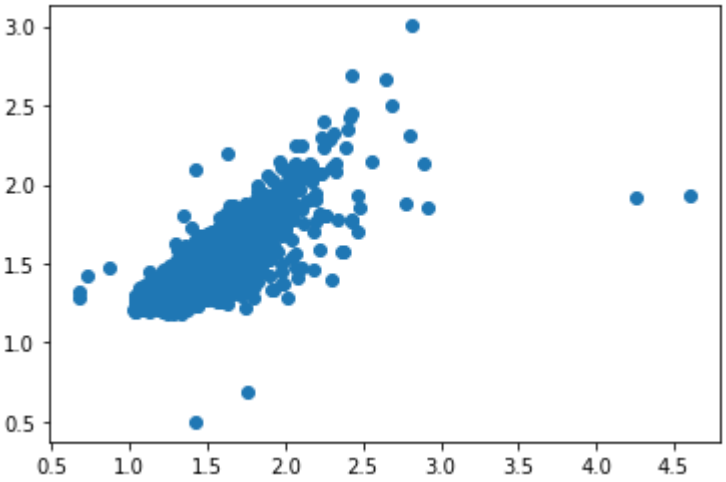
Linear

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 0x19e82febee0>



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

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

Out[51]:
```
1.35    921
1.30    916
1.36    914
1.33    909
1.31    908
       ...
0.94      1
0.81      1
0.72      1
3.33      1
2.91      1
Name: TCH, Length: 188, dtype: int64
```

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

Out[52]:
```
1.0    14706
2.0    10052
Name: TCH, dtype: int64
```

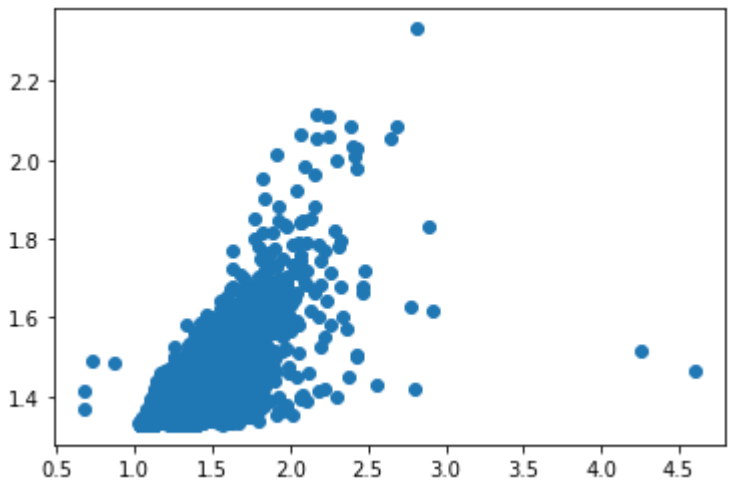
Lasso

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

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

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

Out[54]:
```
<matplotlib.collections.PathCollection at 0x19e813f4070>
```



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

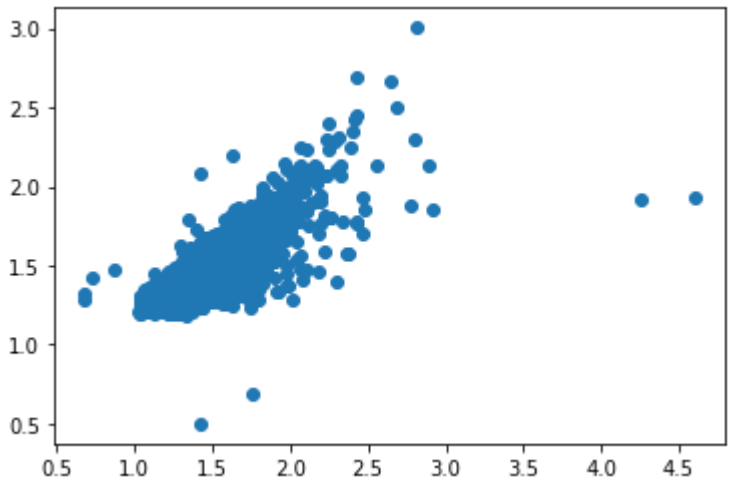
Ridge

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 0x19e814485e0>
```



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

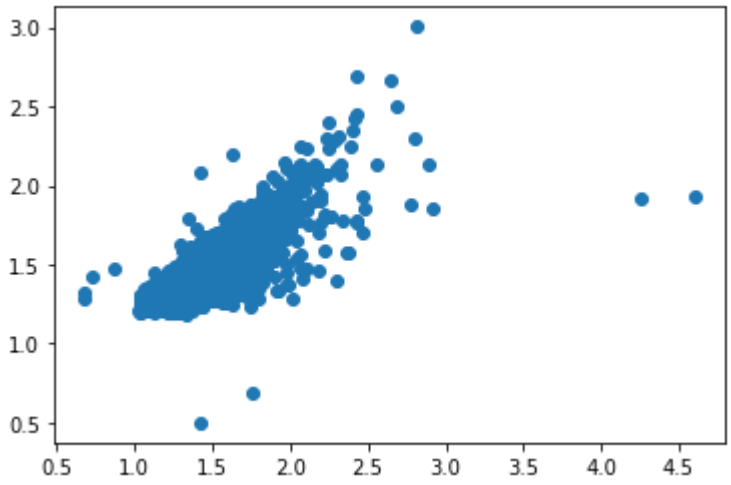
ElasticNet

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 0x19e814ab130>
```



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

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

```
0.6415748472378532
```

Out[62]:
```
0.6880908580848588
```

Logistic

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

Out[63]:
```
Low     14706
High    10052
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 0x19e80dc6430>
```

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')
```

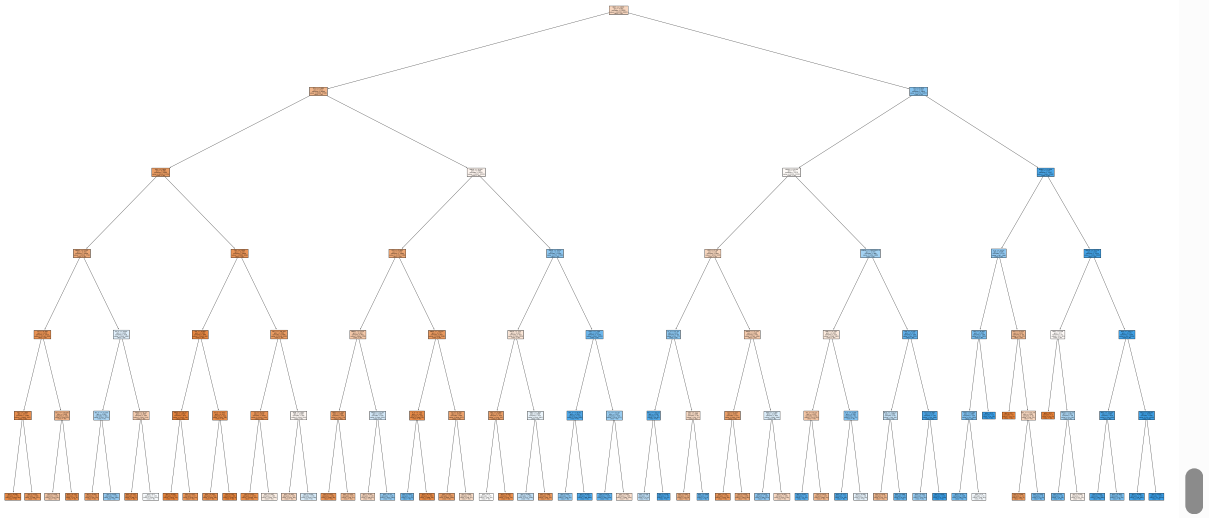
In [74]:
```
rfcs=grid_search.best_score_
```

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(4426.169491525424, 155.3142857142857, 'gini = 0.009\nsamples = 677\nv
alue = [5, 1053]\nclass = No')]
```



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

```
Linear: 0.6418610744783498
Lasso: 0.4174194874610826
Ridge: 0.6415748472378532
ElasticNet: 0.4894081124639743
Logistic: 0.5988152934841141
Random Forest: 0.8667628390075015
```

In [ ]: