madrid_2003

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	PM
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.2099
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.3899
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.2400
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.8399
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.7799
...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.3800
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.4000
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.8300
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.5700
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.3500

243984 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243984 entries, 0 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     243984 non-null  object 
 1   BEN      69745 non-null   float64
 2   CO       225340 non-null  float64
 3   EBE      61244 non-null   float64
 4   MXY      42045 non-null   float64
 5   NMHC     111951 non-null  float64
 6   NO_2     242625 non-null  float64
 7   NOx      242629 non-null  float64
 8   OXY      42072 non-null   float64
 9   O_3      234131 non-null  float64
 10  PM10     240896 non-null  float64
 11  PXY      42063 non-null   float64
 12  SO_2     242729 non-null  float64
 13  TCH      111991 non-null  float64
 14  TOL      69439 non-null   float64
 15  station  243984 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 29.8+ MB
```

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

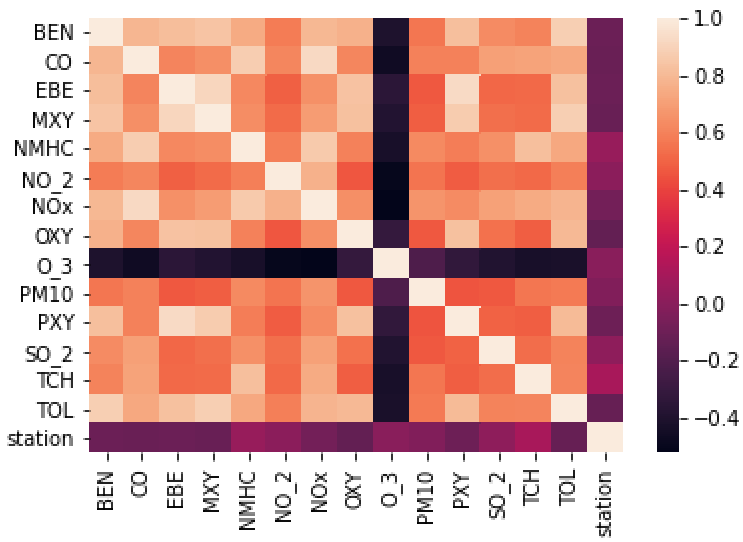
Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	F
5	2003-03-01 01:00:00	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.950000	95.15
23	2003-03-01 01:00:00	3.46	1.27	3.43	7.08	0.18	54.250000	173.300003	3.37	6.540000	53.00
27	2003-03-01 01:00:00	6.39	1.79	5.75	10.88	0.33	75.459999	281.100006	3.68	6.690000	63.84
33	2003-03-01 02:00:00	7.42	1.47	10.63	24.73	0.35	83.309998	277.200012	11.00	9.900000	58.88
51	2003-03-01 02:00:00	3.62	1.29	3.20	7.08	0.19	42.209999	166.300003	3.41	6.380000	47.59
...	...	...	...	...	...	...	...	...	...	...	
243955	2003-09-30 23:00:00	1.75	0.41	3.07	9.38	0.09	46.290001	77.709999	3.11	18.280001	7.52
243957	2003-10-01 00:00:00	2.35	0.60	3.88	10.86	0.11	61.240002	133.100006	0.89	10.900000	10.24
243961	2003-10-01 00:00:00	2.97	0.82	4.53	10.88	0.05	36.529999	131.300003	5.52	12.940000	25.68
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.38
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.35

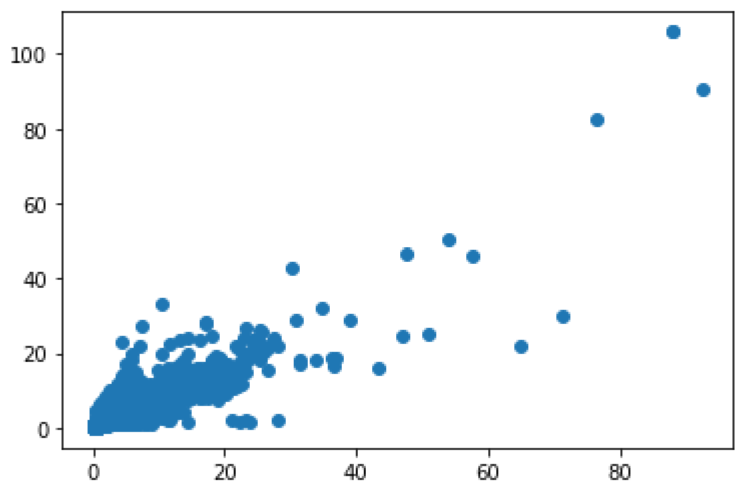
33010 rows × 16 columns

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 0x1d3355fe9d0>]



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)
```

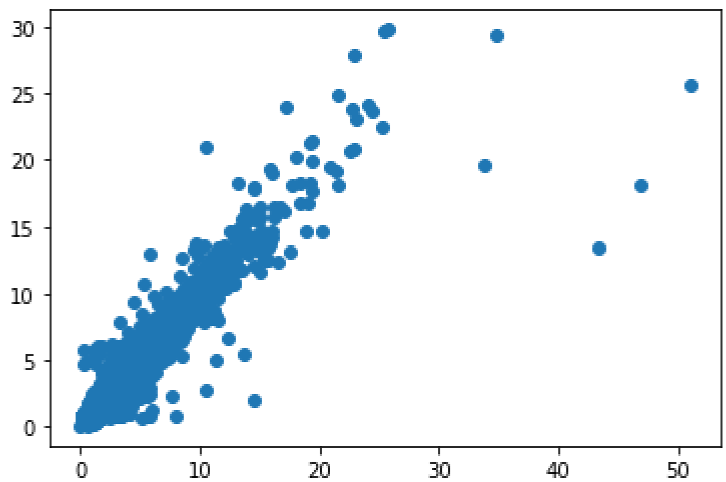
LINEAR

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 0x1d33628b460>
```



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

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

Out[14]:
```
1.30    1344
1.31    1342
1.32    1281
1.27    1279
1.29    1262
        ...
3.50       1
3.87       1
3.21       1
3.14       1
1.01       1
Name: TCH, Length: 243, 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    21614
2.0    11396
Name: TCH, dtype: int64
```

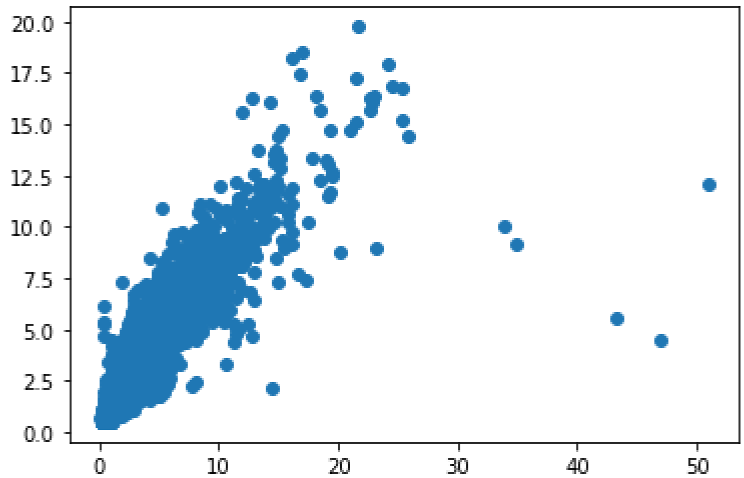
In [16]:
```
# Lasso
```

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 0x1d336dded30>



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

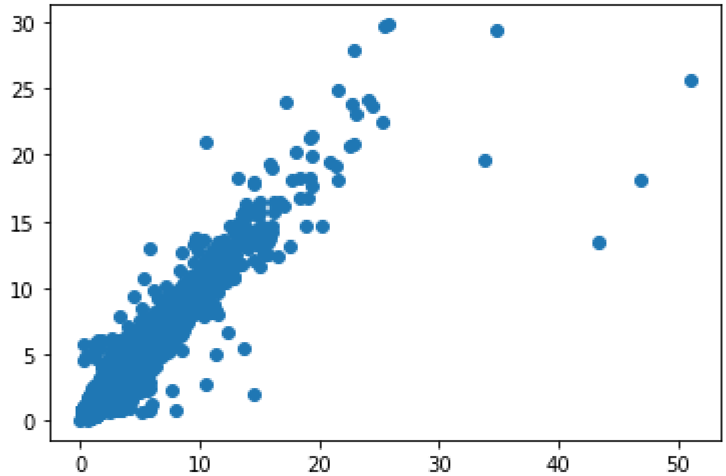
RIDGE

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 0x1d3355dd7f0>



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

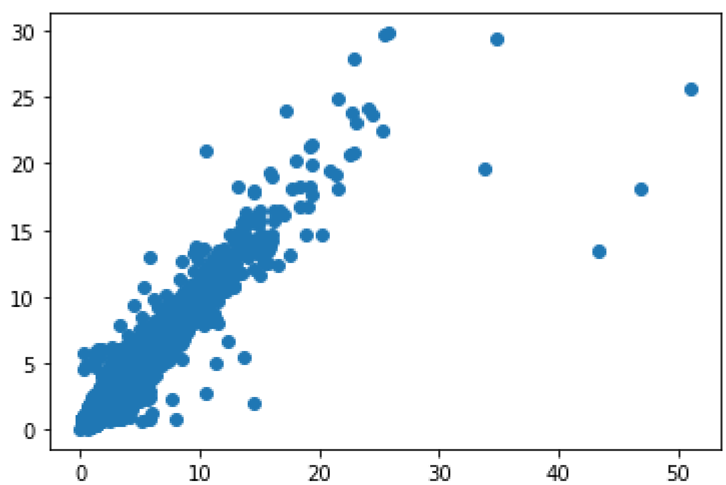
ElasticNet

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 0x1d336e787c0>



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

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

```
0.9058532823453943
```

Out[26]:
```
0.9183044295499866
```

LOGISTIC

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

Out[27]:
```
Low     21614
High    11396
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 0x1d336007df0>
```

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
```

```
 Text(1155.8571428571427, 155.3142857142857, 'gini = 0.477\nsamples = 47\nv
alue = [31, 48]\nclass = No'),
 Text(1275.4285714285713, 465.9428571428573, 'PXY <= 7.68\ngini = 0.202\nsa
mples = 20\nvalue = [4, 31]\nclass = No'),
 Text(1235.5714285714284, 155.3142857142857, 'gini = 0.0\nsamples = 8\nvalu
e = [0, 16]\nclass = No'),
 Text(1315.2857142857142, 155.3142857142857, 'gini = 0.332\nsamples = 12\nv
alue = [4, 15]\nclass = No'),
 Text(1514.5714285714284, 776.5714285714287, 'TOL <= 8.135\ngini = 0.275\ns
amples = 942\nvalue = [247, 1251]\nclass = No'),
 Text(1434.8571428571427, 465.9428571428573, 'NO_2 <= 60.92\ngini = 0.409\n
samples = 249\nvalue = [110, 274]\nclass = No'),
 Text(1395.0, 155.3142857142857, 'gini = 0.225\nsamples = 76\nvalue = [16,
108]\nclass = No'),
 Text(1474.7142857142856, 155.3142857142857, 'gini = 0.462\nsamples = 173\n
value = [94, 166]\nclass = No'),
 Text(1594.2857142857142, 465.9428571428573, 'NMHC <= 0.165\ngini = 0.216\n
samples = 693\nvalue = [137, 977]\nclass = No'),
 Text(1554.4285714285713, 155.3142857142857, 'gini = 0.454\nsamples = 175\n
```

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

```
Linear: 0.9058641997341937
Lasso: 0.777591570715501
Ridge: 0.9058532823453943
ElasticNet: 0.904694103254376
Logistic: 0.6520246389982833
Random Forest: 0.8822866253548514
```

Best Model is Random Forest

madrid_2004

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	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080
...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389

245496 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245496 entries, 0 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     245496 non-null  object 
 1   BEN      65158 non-null   float64
 2   CO       226043 non-null  float64
 3   EBE      56781 non-null   float64
 4   MXY      39867 non-null   float64
 5   NMHC     107630 non-null  float64
 6   NO_2     243280 non-null  float64
 7   NOx      243283 non-null  float64
 8   OXY      39882 non-null   float64
 9   O_3      233811 non-null  float64
 10  PM10     234655 non-null  float64
 11  PM25     58145 non-null   float64
 12  PXY      39891 non-null   float64
 13  SO_2     243402 non-null  float64
 14  TCH      107650 non-null  float64
 15  TOL      64914 non-null   float64
 16  station  245496 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 31.8+ MB
```

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

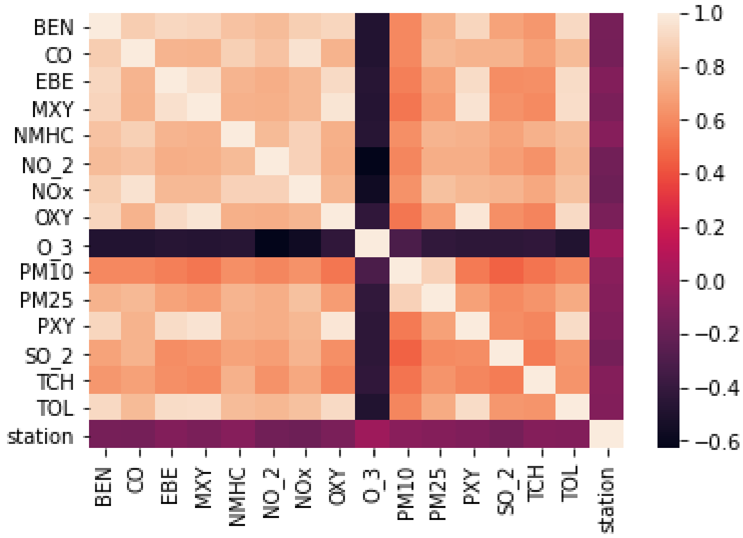
Out[44]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.110
22	2004-08-01 01:00:00	0.55	0.36	0.54	0.86	0.07	31.980000	32.799999	0.50	79.040001	43.549
26	2004-08-01 01:00:00	1.80	0.46	2.28	4.62	0.21	62.259998	75.470001	2.47	54.419998	46.630
32	2004-08-01 02:00:00	1.94	0.67	3.14	4.91	0.06	113.500000	165.800003	2.56	26.980000	86.930
49	2004-08-01 02:00:00	0.29	0.30	0.47	0.76	0.07	33.919998	34.840000	0.46	75.570000	48.959
...	...	...	...	...	...	...	...	...	...	...	
245463	2004-05-31 23:00:00	0.62	0.08	0.54	0.70	0.04	44.360001	45.450001	0.42	43.419998	19.290
245467	2004-05-31 23:00:00	2.39	0.67	2.49	3.92	0.20	89.809998	132.800003	2.09	14.740000	31.809
245473	2004-06-01 00:00:00	3.72	1.12	4.33	8.79	0.24	113.900002	253.600006	4.51	9.380000	21.219
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389

19397 rows × 17 columns

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

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

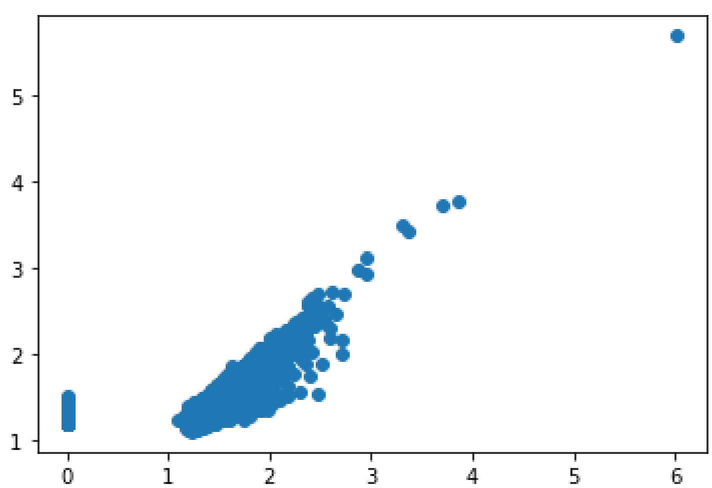


In [47]:
```
x=df3.drop(["TCH"],axis=1)
y=df3["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x1d336d1c310>



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

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

Out[51]:
```
1.34    740
1.33    714
1.35    708
1.37    688
1.36    679
       ...
2.95      1
3.65      1
3.59      1
2.58      1
3.86      1
Name: TCH, Length: 191, 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    11861
2.0     7536
Name: TCH, dtype: int64
```

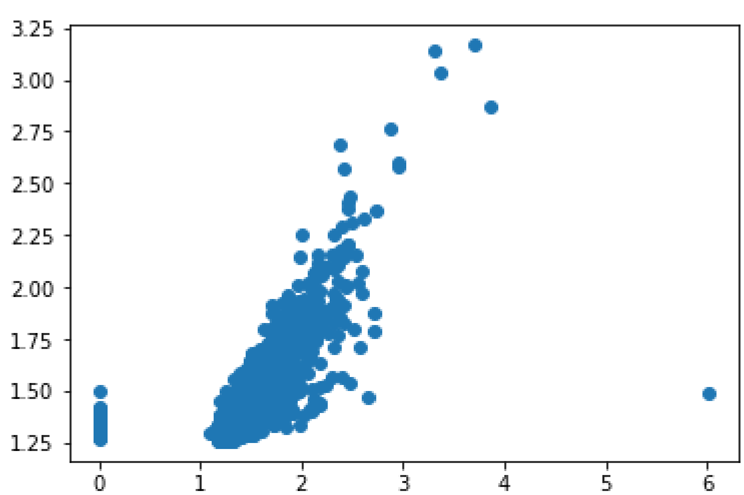
Lasso

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 0x1d336d41fd0>
```



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

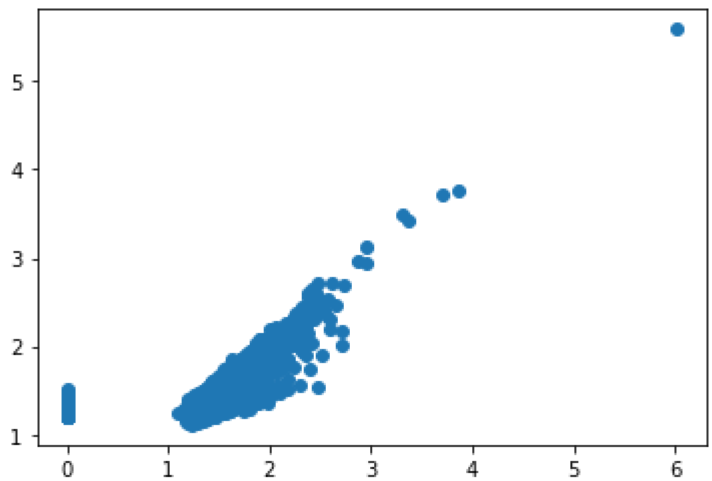
Ridge

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 0x1d336f304c0>



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

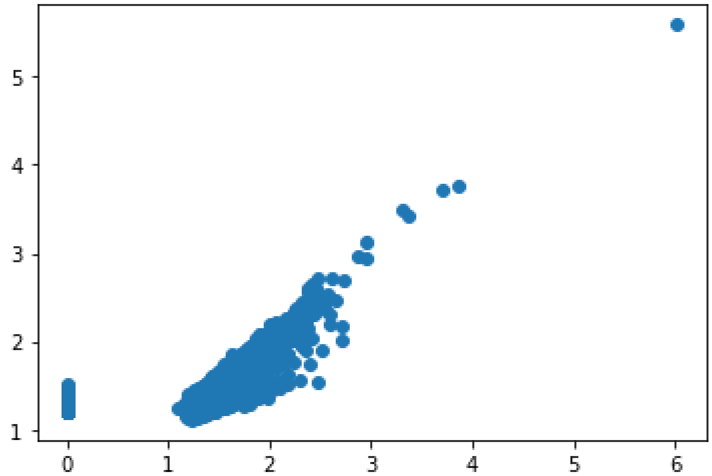
ElasticNet

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 0x1d336f80fd0>



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

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

0.6079595072031811

Out[62]: 0.5818228410220392

Logistic

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

Out[63]:
```
Low     11861
High     7536
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 0x1d336a96ca0>

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
```

Out[76]:
```
[Text(2269.2000000000003, 2019.0857142857144, 'NMHC <= 0.155\ngini = 0.475
\nsamples = 8525\nvalue = [8293, 5284]\nclass = Yes'),
 Text(1190.4, 1708.457142857143, 'TOL <= 9.185\ngini = 0.228\nsamples = 547
4\nvalue = [7583, 1147]\nclass = Yes'),
 Text(595.2, 1397.8285714285716, 'CO <= 0.605\ngini = 0.153\nsamples = 4712
\nvalue = [6872, 624]\nclass = Yes'),
 Text(297.6, 1087.2, 'NO_2 <= 45.965\ngini = 0.114\nsamples = 4150\nvalue =
[6224, 401]\nclass = Yes'),
 Text(148.8, 776.5714285714287, 'O_3 <= 16.285\ngini = 0.049\nsamples = 284
3\nvalue = [4415, 114]\nclass = Yes'),
 Text(74.4, 465.9428571428573, 'TOL <= 5.57\ngini = 0.326\nsamples = 108\nv
alue = [124, 32]\nclass = Yes'),
 Text(37.2, 155.3142857142857, 'gini = 0.172\nsamples = 66\nvalue = [86, 9]
\nclass = Yes'),
 Text(111.60000000000001, 155.3142857142857, 'gini = 0.47\nsamples = 42\nva
lue = [38, 23]\nclass = Yes'),
 Text(223.20000000000002, 465.9428571428573, 'O_3 <= 34.145\ngini = 0.037\n
samples = 2735\nvalue = [4291, 82]\nclass = Yes'),
 Text(186.0, 155.3142857142857, 'gini = 0.136\nsamples = 317\nvalue = [469,
```

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

```
Linear: 0.6080992534261667
Lasso: 0.4783072218444412
Ridge: 0.6079595072031811
ElasticNet: 0.501905323414281
Logistic: 0.6082474226804123
Random Forest: 0.8954112049779301
```

In [ ]: