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,Elasti
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_2003
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209999	NaN	2
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389999	NaN	1
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240002	NaN	1
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839996	NaN	2
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779999	NaN	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	7.380000	1.20	
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400000	0.50	
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.830000	NaN	
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.570000	NaN	
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	12.350000	2.43	

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]:

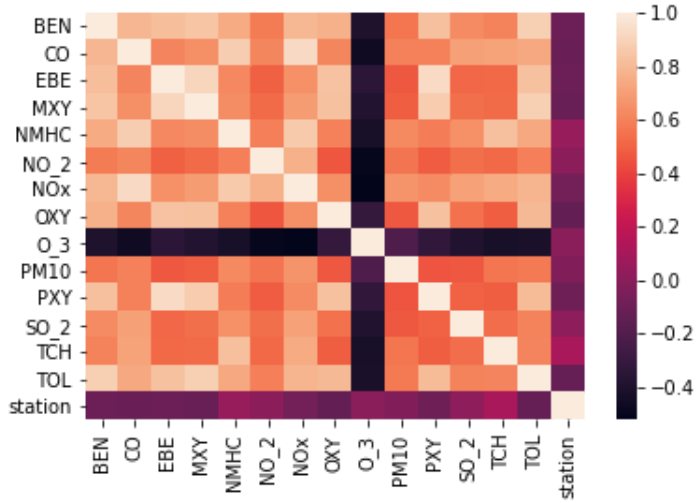
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY
5	2003-03-01 01:00:00	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.950000	95.150002	7.94
23	2003-03-01 01:00:00	3.46	1.27	3.43	7.08	0.18	54.250000	173.300003	3.37	6.540000	53.009998	2.62
27	2003-03-01 01:00:00	6.39	1.79	5.75	10.88	0.33	75.459999	281.100006	3.68	6.690000	63.840000	4.24
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	58.880001	8.93
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	47.599998	2.70
...	...	...	...	...	...	...	...	...	...	...	...	...
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	7.520000	3.48
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	10.240000	3.89
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	25.680000	4.13
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	7.380000	1.20
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	12.350000	2.43

33010 rows × 16 columns

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

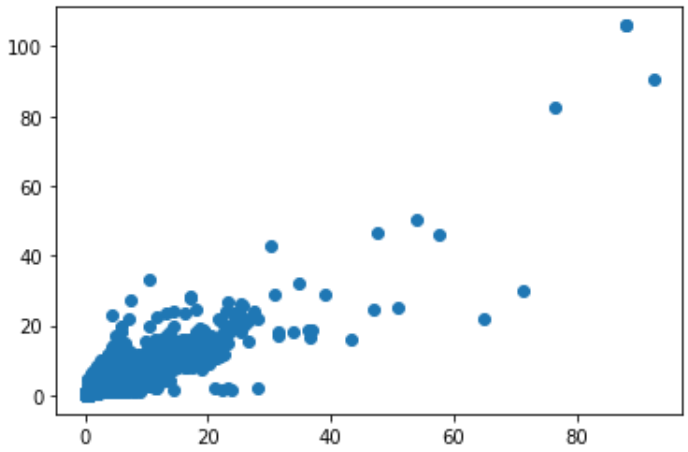
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <AxesSubplot:>



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

Out[7]: [<matplotlib.lines.Line2D at 0x255505b0490>]



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)
```

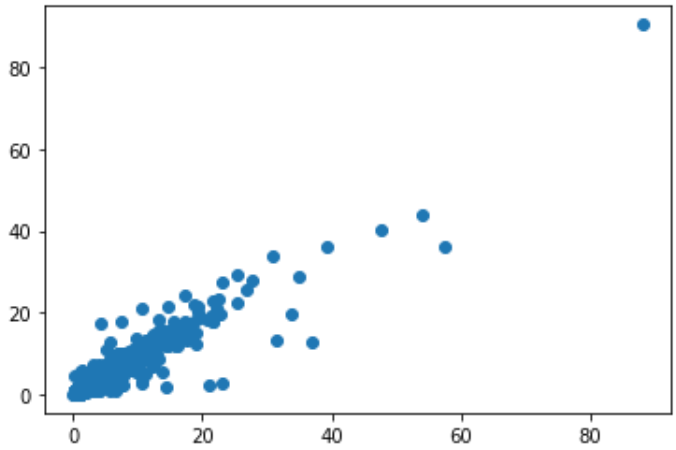
Linear

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 0x25550a8dee0>
```



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
```

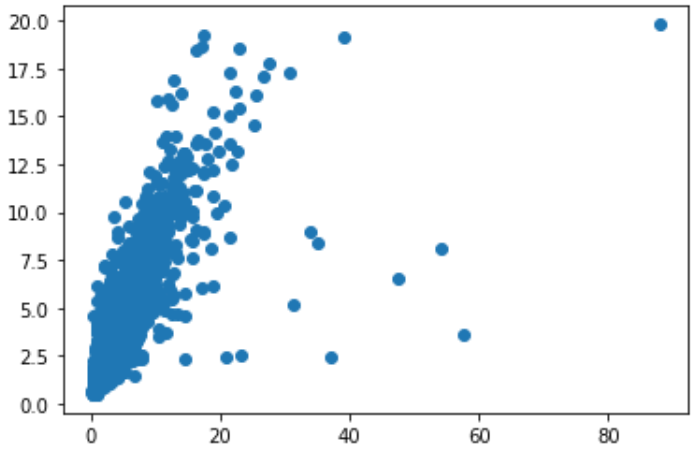
Lasso

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 0x25550acbc70>
```



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

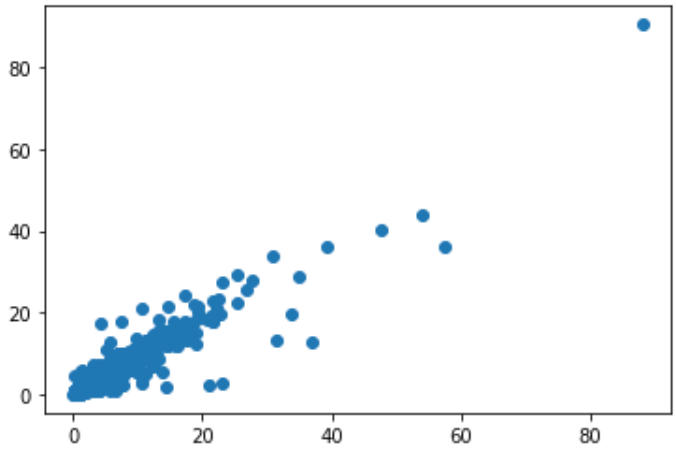
Ridge

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 0x255504ca8e0>
```



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

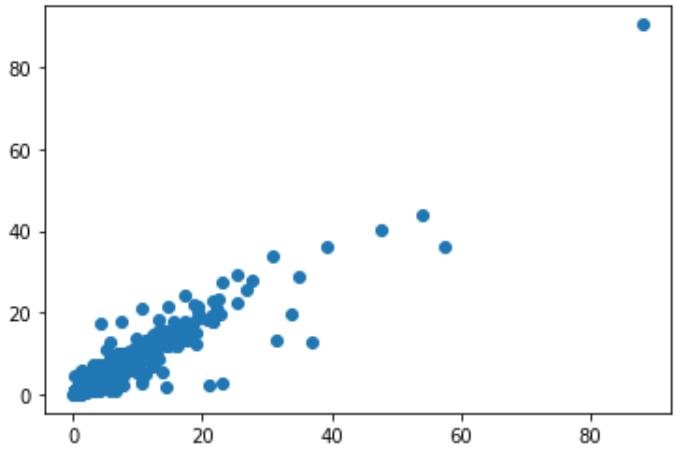
ElasticNet

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 0x25551ce3cd0>
```



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

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

```
0.9195176980660054
```

Out[25]:
```
0.9131593670000937
```

Logistic

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

Out[26]:
```
Low     21614
High    11396
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 0x25551986520>
```

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="accuracy")
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',"No"],fill
```

```
 Text(1675.9999999999998, 1397.8285714285716, 'NMHC <= 0.155\ngini = 0.497\nsamples
= 1496\nvalue = [1278, 1106]\nclass = Yes'),
 Text(1355.142857142857, 1087.2, 'OXY <= 3.16\ngini = 0.296\nsamples = 845\nvalue =
[1080, 238]\nclass = Yes'),
 Text(1195.7142857142856, 776.5714285714287, 'NOx <= 101.4\ngini = 0.418\nsamples =
314\nvalue = [350, 148]\nclass = Yes'),
 Text(1116.0, 465.9428571428573, 'O_3 <= 13.42\ngini = 0.309\nsamples = 186\nvalue =
[241, 57]\nclass = Yes'),
 Text(1076.142857142857, 155.3142857142857, 'gini = 0.481\nsamples = 53\nvalue = [5
5, 37]\nclass = Yes'),
 Text(1155.8571428571427, 155.3142857142857, 'gini = 0.175\nsamples = 133\nvalue =
[186, 20]\nclass = Yes'),
 Text(1275.4285714285713, 465.9428571428573, 'station <= 28079015.0\ngini = 0.496\ns
amples = 128\nvalue = [109, 91]\nclass = Yes'),
 Text(1235.5714285714284, 155.3142857142857, 'gini = 0.26\nsamples = 31\nvalue = [4
4, 8]\nclass = Yes'),
 Text(1315.2857142857142, 155.3142857142857, 'gini = 0.493\nsamples = 97\nvalue = [6
5, 83]\nclass = No'),
 Text(1514.5714285714284, 776.5714285714287, 'O_3 <= 13.285\ngini = 0.195\nsamples =
531\nvalue = [730, 90]\nclass = Yes'),
```

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

```
Linear: 0.9195103306695078
Lasso: 0.6724055165668984
Ridge: 0.9195176980660054
ElasticNet: 0.8935055393696031
Logistic: 0.6567706755528627
Random Forest: 0.8821569724921972
```

Best Model is Ridge Regression

In [41]:
```
df2=pd.read_csv("C:/Users/user/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_200
df2
```

Out[41]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	P
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002	25.86(
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	22.950001	
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000	
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000	
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002	
...	...	...	...	...	...	...	...	...	...	...	...	...
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	30.900000	14.86(
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689999	
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840000	12.04(
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630001	
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	24.389999	17.95(

245496 rows × 17 columns

In [42]:
```
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 [43]:
```
df3=df2.dropna()
df3
```

Out[43]:

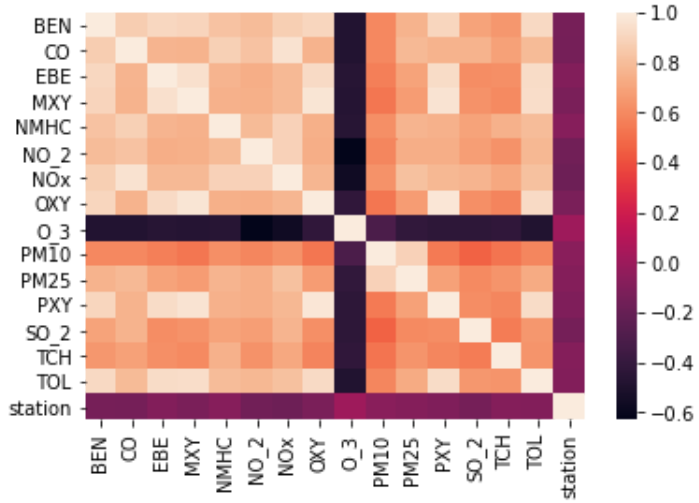
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	P
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.110001	38.049
22	2004-08-01 01:00:00	0.55	0.36	0.54	0.86	0.07	31.980000	32.799999	0.50	79.040001	43.549999	22.780
26	2004-08-01 01:00:00	1.80	0.46	2.28	4.62	0.21	62.259998	75.470001	2.47	54.419998	46.630001	29.459
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	86.930000	45.639
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	48.959999	25.959
...	...	...	...	...	...	...	...	...	...	...	...	...
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	19.290001	13.190
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	31.809999	20.370
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	21.219999	16.660
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	30.900000	14.860
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	24.389999	17.959

19397 rows × 17 columns

In [44]:
```
df3=df3.drop(["date"],axis=1)
```

In [45]:
```
sns.heatmap(df3.corr())
```

Out[45]: <AxesSubplot:>



In [46]:
```
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

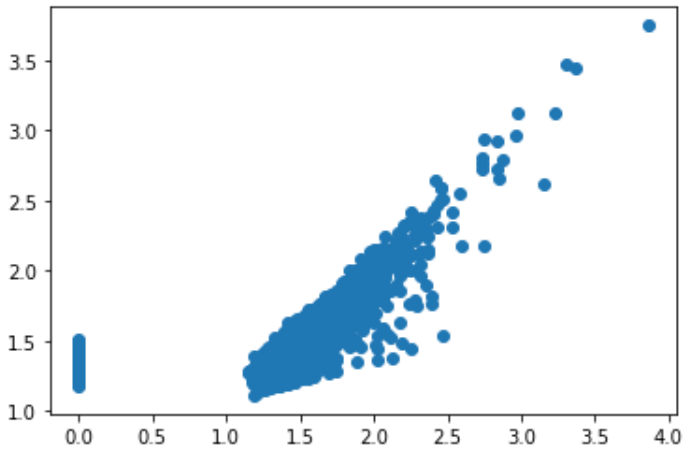
In [47]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[47]: LinearRegression()

In [ ]:

In [48]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[48]: <matplotlib.collections.PathCollection at 0x25551c2ebb0>



In [49]:
```
lis=li.score(x_test,y_test)
```

In [50]:
```
df3["TCH"].value_counts()
```

Out[50]:
```
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 [51]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[51]:
```
1.0    11861
2.0     7536
Name: TCH, dtype: int64
```

In [ ]:
```

```

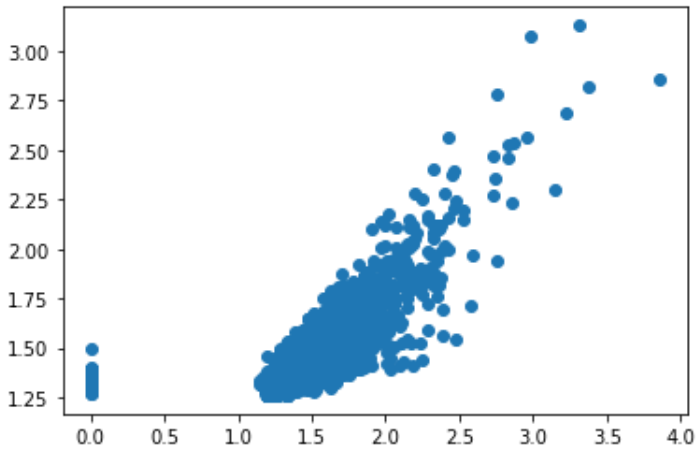
Lasso

In [52]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[52]:
```
Lasso(alpha=5)
```

In [53]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x25551d39cd0>
```



In [54]:
```
las=la.score(x_test,y_test)
```

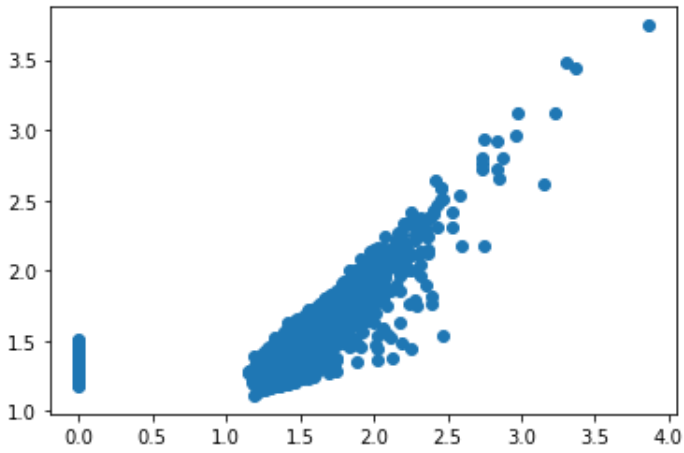
Ridge

In [55]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[55]:
```
Ridge(alpha=1)
```

In [56]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[56]:
```
<matplotlib.collections.PathCollection at 0x25551da22e0>
```



In [57]:
```
rrs=rr.score(x_test,y_test)
```

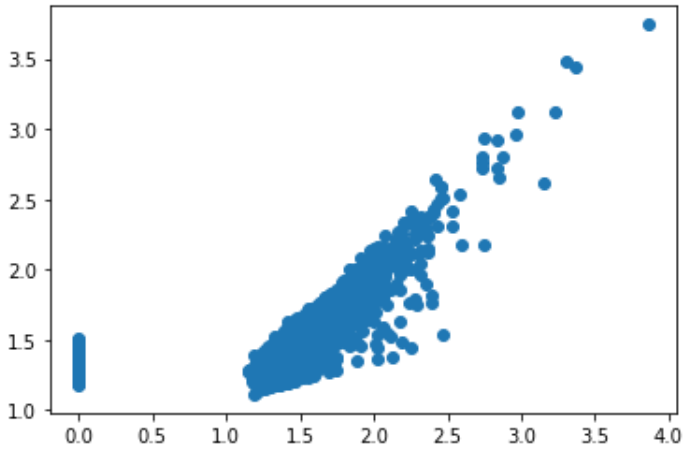
ElasticNet

In [58]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[58]:
```
ElasticNet()
```

In [59]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[59]:
```
<matplotlib.collections.PathCollection at 0x25551df9df0>
```



In [60]:
```
ens=en.score(x_test,y_test)
```

In [61]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.5759339018894761
```

Out[61]:
```
0.5960803646994859
```

Logistic

In [62]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[62]:
```
Low     11861
High     7536
Name: TCH, dtype: int64
```

In [63]:
```
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 [64]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[64]:
```
LogisticRegression()
```

In [65]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[65]:
```
<matplotlib.collections.PathCollection at 0x25551710670>
```

In [66]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [67]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [68]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [69]:
```
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 [70]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[70]:
```
RandomForestClassifier()
```

In [71]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [72]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[72]:
```
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 [73]:
```
rfcs=grid_search.best_score_
```

In [74]:
```
rfc_best=grid_search.best_estimator_
```

In [75]:
```
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"],fill
```

```
347\nvalue = [559, 36]\nclass = Yes'),
 Text(192.41379310344828, 155.3142857142857, 'gini = 0.08\nsamples = 340\nvalue = [5
54, 24]\nclass = Yes'),
 Text(269.3793103448276, 155.3142857142857, 'gini = 0.415\nsamples = 7\nvalue = [5,
12]\nclass = No'),
 Text(461.79310344827593, 776.5714285714287, 'TOL <= 7.055\ngini = 0.236\nsamples =
1265\nvalue = [1709, 270]\nclass = Yes'),
 Text(384.82758620689657, 465.9428571428573, 'PM25 <= 15.78\ngini = 0.202\nsamples =
1128\nvalue = [1559, 201]\nclass = Yes'),
 Text(346.3448275862069, 155.3142857142857, 'gini = 0.111\nsamples = 777\nvalue = [1
129, 71]\nclass = Yes'),
 Text(423.3103448275862, 155.3142857142857, 'gini = 0.357\nsamples = 351\nvalue = [4
30, 130]\nclass = Yes'),
 Text(538.7586206896552, 465.9428571428573, 'SO_2 <= 8.61\ngini = 0.432\nsamples = 1
37\nvalue = [150, 69]\nclass = Yes'),
 Text(500.2758620689656, 155.3142857142857, 'gini = 0.488\nsamples = 83\nvalue = [7
8, 57]\nclass = Yes'),
 Text(577.2413793103449, 155.3142857142857, 'gini = 0.245\nsamples = 54\nvalue = [7
2, 12]\nclass = Yes'),
 Text(923.5862068965519, 1087.2, 'BEN <= 2.135\ngini = 0.347\nsamples = 1327\nvalue
```

In [76]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.5758911744718893
Lasso: 0.49886828067074285
Ridge: 0.5759339018894761
ElasticNet: 0.5250372976356534
Logistic: 0.6163230240549828
Random Forest: 0.8953377842749368
```

Best model is Random Forest

In [ ]:
```

```