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,Rid
from sklearn.model_selection import train_test_split
```

In [2]:
```
df=pd.read_csv("madrid_2007.csv")
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]:

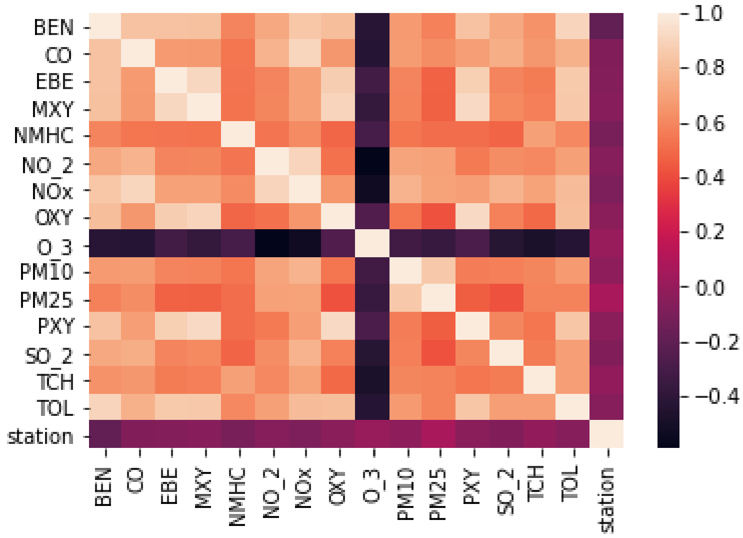
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
4	2007-12-01 01:00:00	4.64	1.86	4.26	7.98	0.57	145.100006	573.900024	3.49	52.689999	106.5
21	2007-12-01 01:00:00	1.98	0.31	2.56	6.06	0.35	76.059998	208.899994	1.70	1.000000	37.7
25	2007-12-01 01:00:00	2.82	1.42	3.15	7.02	0.49	123.099998	402.399994	2.60	7.160000	70.8
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	117.0
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	34.7
...	...	...	...	...	...	...	...	...	...	...	
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	21.8
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	15.0
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	9.2
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
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

25443 rows × 17 columns

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

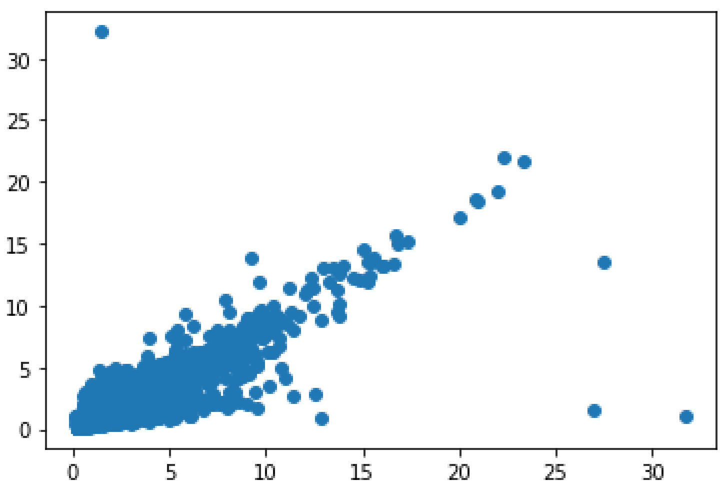
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <AxesSubplot:>



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

Out[7]: [<matplotlib.lines.Line2D at 0x2118cb47160>]



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)
```

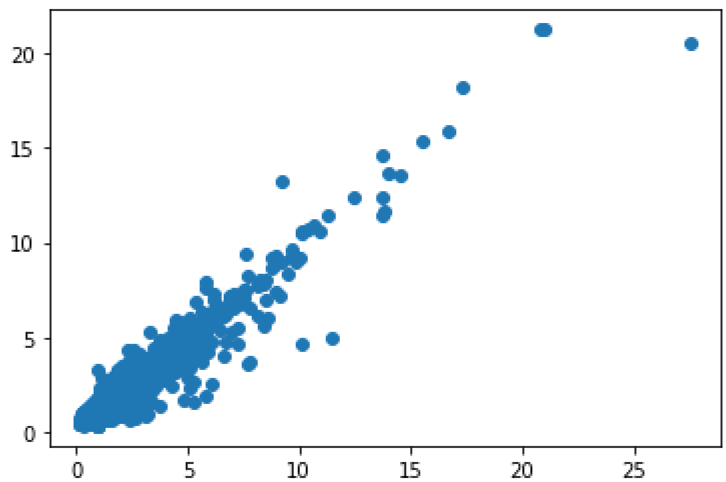
Linear

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 0x2118d9e5bb0>
```



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
```

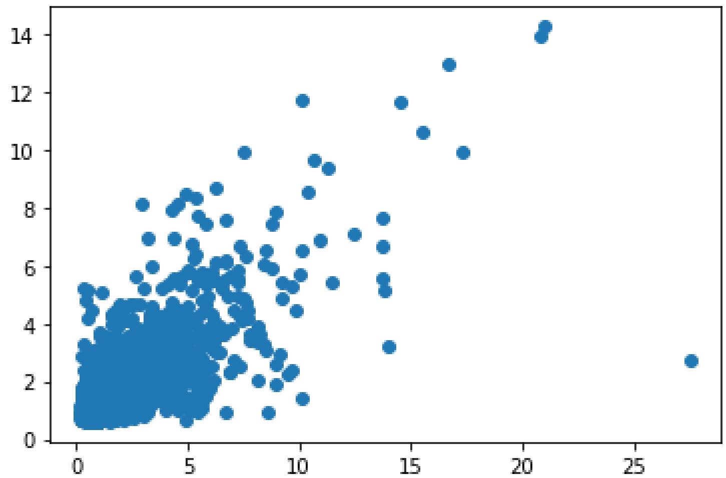
Lasso

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 0x2118da53880>



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

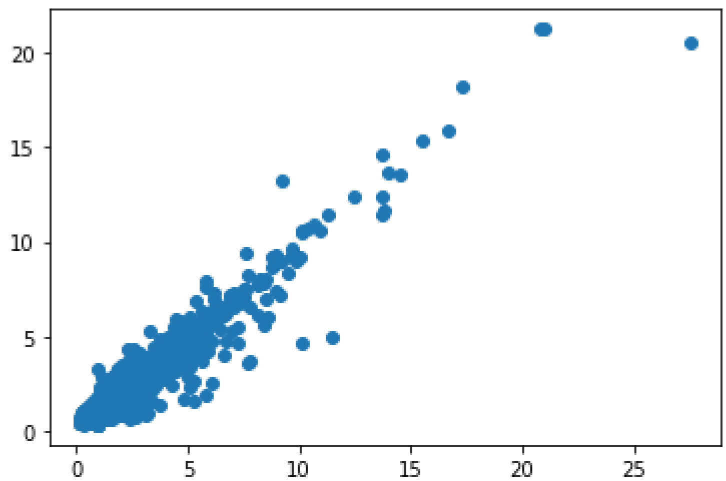
Ridge

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 0x2118cb1d6a0>



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

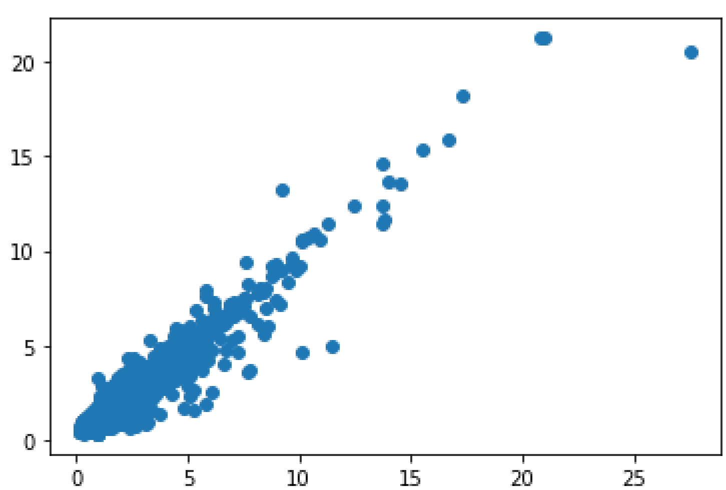
ElasticNet

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 0x2118dadef40>



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

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

```
0.9130065705546464
```

Out[25]:
```
0.8598350807252741
```

Logistic

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

Out[26]:
```
Low     14025
High    11418
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 0x2118db28d00>
```

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="accu
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',"
```

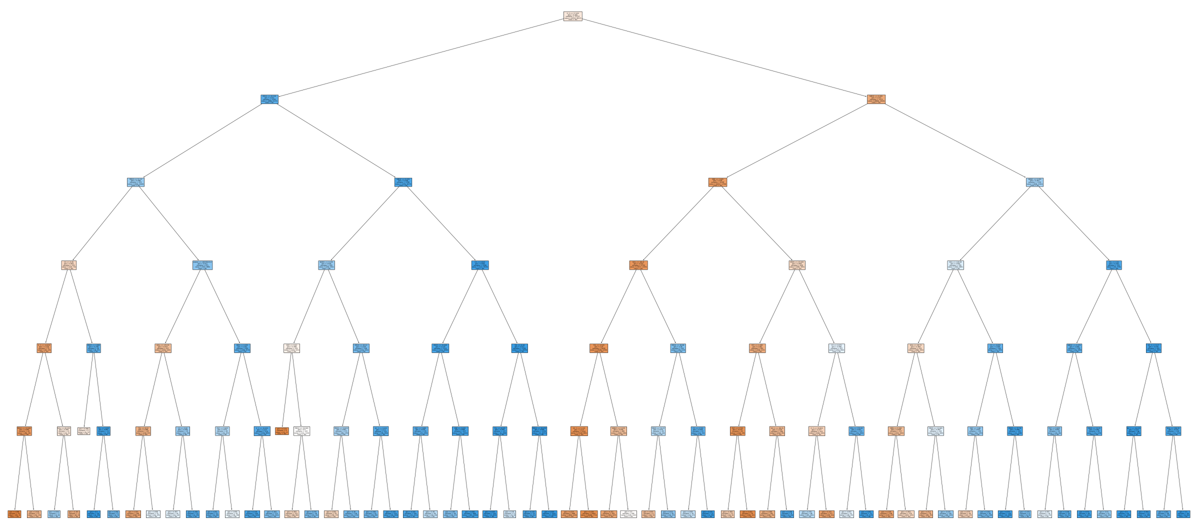
Out[39]:
```
[Text(2134.3500000000004, 2019.0857142857144, 'O_3 <= 17.425\ngini = 0.495\ns
amples = 11234\nvalue = [9753, 8057]\nclass = Yes'),
 Text(995.1, 1708.457142857143, 'PM10 <= 25.535\ngini = 0.251\nsamples = 3760
\nvalue = [888, 5137]\nclass = No'),
 Text(492.90000000000003, 1397.8285714285716, 'PM10 <= 11.12\ngini = 0.438\ns
amples = 974\nvalue = [509, 1060]\nclass = No'),
 Text(241.8, 1087.2, 'CO <= 0.45\ngini = 0.483\nsamples = 103\nvalue = [105,
72]\nclass = Yes'),
 Text(148.8, 776.5714285714287, 'O_3 <= 14.835\ngini = 0.326\nsamples = 74\nv
alue = [97, 25]\nclass = Yes'),
 Text(74.4, 465.9428571428573, 'PM10 <= 10.265\ngini = 0.265\nsamples = 60\nv
alue = [86, 16]\nclass = Yes'),
 Text(37.2, 155.3142857142857, 'gini = 0.098\nsamples = 33\nvalue = [55, 3]\n
class = Yes'),
 Text(111.60000000000001, 155.3142857142857, 'gini = 0.416\nsamples = 27\nval
ue = [31, 13]\nclass = Yes'),
 Text(223.20000000000002, 465.9428571428573, 'NOx <= 60.305\ngini = 0.495\nsa
mples = 14\nvalue = [11, 9]\nclass = Yes'),
 Text(186.0, 155.3142857142857, 'gini = 0.444\nsamples = 7\nvalue = [3, 6]\nc
lass = No'),
 Text(260.40000000000003, 155.3142857142857, 'gini = 0.397\nsamples = 7\nvalu
e = [8, 3]\nclass = Yes'),
 Text(334.8, 776.5714285714287, 'PM25 <= 4.895\ngini = 0.249\nsamples = 29\nv
alue = [8, 47]\nclass = No'),
 Text(297.6, 465.9428571428573, 'gini = 0.496\nsamples = 6\nvalue = [6, 5]\nc
lass = Yes'),
 Text(372.0, 465.9428571428573, 'TOL <= 5.025\ngini = 0.087\nsamples = 23\nva
lue = [2, 42]\nclass = No'),
 Text(334.8, 155.3142857142857, 'gini = 0.0\nsamples = 18\nvalue = [0, 35]\nc
lass = No'),
 Text(409.20000000000005, 155.3142857142857, 'gini = 0.346\nsamples = 5\nvalu
e = [2, 7]\nclass = No'),
 Text(744.0, 1087.2, 'station <= 28079015.0\ngini = 0.412\nsamples = 871\nval
ue = [404, 988]\nclass = No'),
 Text(595.2, 776.5714285714287, 'NOx <= 175.55\ngini = 0.437\nsamples = 227\n
value = [244, 116]\nclass = Yes'),
 Text(520.8000000000001, 465.9428571428573, 'TOL <= 8.14\ngini = 0.398\nsampl
es = 197\nvalue = [230, 87]\nclass = Yes'),
 Text(483.6, 155.3142857142857, 'gini = 0.354\nsamples = 166\nvalue = [208, 6
2]\nclass = Yes'),
 Text(558.0, 155.3142857142857, 'gini = 0.498\nsamples = 31\nvalue = [22, 25]
\nclass = No'),
 Text(669.6, 465.9428571428573, 'CO <= 0.82\ngini = 0.439\nsamples = 30\nvalu
e = [14, 29]\nclass = No'),
 Text(632.4000000000001, 155.3142857142857, 'gini = 0.497\nsamples = 18\nvalu
e = [12, 14]\nclass = No'),
 Text(706.8000000000001, 155.3142857142857, 'gini = 0.208\nsamples = 12\nvalu
e = [2, 15]\nclass = No'),
 Text(892.8000000000001, 776.5714285714287, 'BEN <= 0.455\ngini = 0.262\nsamp
les = 644\nvalue = [160, 872]\nclass = No'),
 Text(818.4000000000001, 465.9428571428573, 'PXY <= 0.625\ngini = 0.471\nsamp
les = 65\nvalue = [41, 67]\nclass = No'),
 Text(781.2, 155.3142857142857, 'gini = 0.291\nsamples = 21\nvalue = [6, 28]
\nclass = No'),
 Text(855.6, 155.3142857142857, 'gini = 0.499\nsamples = 44\nvalue = [35, 39]
\nclass = No'),
 Text(967.2, 465.9428571428573, 'O_3 <= 10.815\ngini = 0.224\nsamples = 579\n
```

```
value = [119, 805]\nclass = No'),
 Text(930.0000000000001, 155.3142857142857, 'gini = 0.139\nsamples = 405\nval
ue = [49, 604]\nclass = No'),
 Text(1004.4000000000001, 155.3142857142857, 'gini = 0.383\nsamples = 174\nva
lue = [70, 201]\nclass = No'),
 Text(1497.3000000000002, 1397.8285714285716, 'NMHC <= 0.225\ngini = 0.156\ns
amples = 2786\nvalue = [379, 4077]\nclass = No'),
 Text(1209.0, 1087.2, 'NMHC <= 0.155\ngini = 0.436\nsamples = 500\nvalue = [2
56, 542]\nclass = No'),
 Text(1078.8000000000002, 776.5714285714287, 'MXY <= 0.63\ngini = 0.498\nsamp
les = 152\nvalue = [124, 111]\nclass = Yes'),
 Text(1041.6000000000001, 465.9428571428573, 'gini = 0.133\nsamples = 9\nvalu
e = [13, 1]\nclass = Yes'),
 Text(1116.0, 465.9428571428573, 'PM10 <= 44.565\ngini = 0.5\nsamples = 143\n
value = [111, 110]\nclass = Yes'),
 Text(1078.8000000000002, 155.3142857142857, 'gini = 0.483\nsamples = 107\nva
lue = [96, 66]\nclass = Yes'),
 Text(1153.2, 155.3142857142857, 'gini = 0.379\nsamples = 36\nvalue = [15, 4
4]\nclass = No'),
 Text(1339.2, 776.5714285714287, 'PM10 <= 40.12\ngini = 0.359\nsamples = 348
\nvalue = [132, 431]\nclass = No'),
 Text(1264.8000000000002, 465.9428571428573, 'PM25 <= 14.615\ngini = 0.461\ns
amples = 173\nvalue = [96, 170]\nclass = No'),
 Text(1227.6000000000001, 155.3142857142857, 'gini = 0.481\nsamples = 55\nval
ue = [55, 37]\nclass = Yes'),
 Text(1302.0, 155.3142857142857, 'gini = 0.36\nsamples = 118\nvalue = [41, 13
3]\nclass = No'),
 Text(1413.6000000000001, 465.9428571428573, 'SO_2 <= 9.425\ngini = 0.213\nsa
mples = 175\nvalue = [36, 261]\nclass = No'),
 Text(1376.4, 155.3142857142857, 'gini = 0.348\nsamples = 52\nvalue = [20, 6
9]\nclass = No'),
 Text(1450.8000000000002, 155.3142857142857, 'gini = 0.142\nsamples = 123\nva
lue = [16, 192]\nclass = No'),
 Text(1785.6000000000001, 1087.2, 'CO <= 0.815\ngini = 0.065\nsamples = 2286
\nvalue = [123, 3535]\nclass = No'),
 Text(1636.8000000000002, 776.5714285714287, 'PM25 <= 15.605\ngini = 0.119\ns
amples = 1103\nvalue = [112, 1656]\nclass = No'),
 Text(1562.4, 465.9428571428573, 'OXY <= 2.695\ngini = 0.235\nsamples = 176\n
value = [37, 235]\nclass = No'),
 Text(1525.2, 155.3142857142857, 'gini = 0.165\nsamples = 150\nvalue = [21, 2
11]\nclass = No'),
 Text(1599.6000000000001, 155.3142857142857, 'gini = 0.48\nsamples = 26\nvalu
e = [16, 24]\nclass = No'),
 Text(1711.2, 465.9428571428573, 'EBE <= 0.635\ngini = 0.095\nsamples = 927\n
value = [75, 1421]\nclass = No'),
 Text(1674.0000000000002, 155.3142857142857, 'gini = 0.394\nsamples = 28\nval
ue = [10, 27]\nclass = No'),
 Text(1748.4, 155.3142857142857, 'gini = 0.085\nsamples = 899\nvalue = [65, 1
394]\nclass = No'),
 Text(1934.4, 776.5714285714287, 'TOL <= 6.18\ngini = 0.012\nsamples = 1183\n
value = [11, 1879]\nclass = No'),
 Text(1860.0000000000002, 465.9428571428573, 'TOL <= 6.025\ngini = 0.055\nsam
ples = 147\nvalue = [6, 208]\nclass = No'),
 Text(1822.8000000000002, 155.3142857142857, 'gini = 0.029\nsamples = 142\nva
lue = [3, 204]\nclass = No'),
 Text(1897.2, 155.3142857142857, 'gini = 0.49\nsamples = 5\nvalue = [3, 4]\nc
lass = No'),
```

```
 Text(2008.8000000000002, 465.9428571428573, 'PM10 <= 29.675\ngini = 0.006\ns
amples = 1036\nvalue = [5, 1671]\nclass = No'),
 Text(1971.6000000000001, 155.3142857142857, 'gini = 0.159\nsamples = 15\nval
ue = [2, 21]\nclass = No'),
 Text(2046.0000000000002, 155.3142857142857, 'gini = 0.004\nsamples = 1021\nv
alue = [3, 1650]\nclass = No'),
 Text(3273.6000000000004, 1708.457142857143, 'NMHC <= 0.285\ngini = 0.373\nsa
mples = 7474\nvalue = [8865, 2920]\nclass = Yes'),
 Text(2678.4, 1397.8285714285716, 'EBE <= 1.625\ngini = 0.29\nsamples = 6348
\nvalue = [8257, 1759]\nclass = Yes'),
 Text(2380.8, 1087.2, 'NOx <= 181.8\ngini = 0.255\nsamples = 5740\nvalue = [7
677, 1354]\nclass = Yes'),
 Text(2232.0, 776.5714285714287, 'TOL <= 4.525\ngini = 0.24\nsamples = 5638\n
value = [7638, 1240]\nclass = Yes'),
 Text(2157.6000000000004, 465.9428571428573, 'SO_2 <= 5.125\ngini = 0.214\nsa
mples = 5152\nvalue = [7129, 991]\nclass = Yes'),
 Text(2120.4, 155.3142857142857, 'gini = 0.324\nsamples = 688\nvalue = [863,
220]\nclass = Yes'),
 Text(2194.8, 155.3142857142857, 'gini = 0.195\nsamples = 4464\nvalue = [626
6, 771]\nclass = Yes'),
 Text(2306.4, 465.9428571428573, 'NMHC <= 0.225\ngini = 0.441\nsamples = 486
\nvalue = [509, 249]\nclass = Yes'),
 Text(2269.2000000000003, 155.3142857142857, 'gini = 0.351\nsamples = 301\nva
lue = [364, 107]\nclass = Yes'),
 Text(2343.6000000000004, 155.3142857142857, 'gini = 0.5\nsamples = 185\nvalu
e = [145, 142]\nclass = Yes'),
 Text(2529.6000000000004, 776.5714285714287, 'PM10 <= 38.89\ngini = 0.38\nsam
ples = 102\nvalue = [39, 114]\nclass = No'),
 Text(2455.2000000000003, 465.9428571428573, 'PM25 <= 10.23\ngini = 0.472\nsa
mples = 49\nvalue = [26, 42]\nclass = No'),
 Text(2418.0, 155.3142857142857, 'gini = 0.444\nsamples = 9\nvalue = [10, 5]
\nclass = Yes'),
 Text(2492.4, 155.3142857142857, 'gini = 0.422\nsamples = 40\nvalue = [16, 3
7]\nclass = No'),
 Text(2604.0, 465.9428571428573, 'CO <= 0.705\ngini = 0.259\nsamples = 53\nva
lue = [13, 72]\nclass = No'),
 Text(2566.8, 155.3142857142857, 'gini = 0.485\nsamples = 16\nvalue = [12, 1
7]\nclass = No'),
 Text(2641.2000000000003, 155.3142857142857, 'gini = 0.035\nsamples = 37\nval
ue = [1, 55]\nclass = No'),
 Text(2976.0, 1087.2, 'PM25 <= 15.645\ngini = 0.484\nsamples = 608\nvalue =
[580, 405]\nclass = Yes'),
 Text(2827.2000000000003, 776.5714285714287, 'PXY <= 1.285\ngini = 0.382\nsam
ples = 281\nvalue = [326, 113]\nclass = Yes'),
 Text(2752.8, 465.9428571428573, 'PXY <= 0.755\ngini = 0.233\nsamples = 92\nv
alue = [116, 18]\nclass = Yes'),
 Text(2715.6000000000004, 155.3142857142857, 'gini = 0.469\nsamples = 13\nval
ue = [10, 6]\nclass = Yes'),
 Text(2790.0, 155.3142857142857, 'gini = 0.183\nsamples = 79\nvalue = [106, 1
2]\nclass = Yes'),
 Text(2901.6000000000004, 465.9428571428573, 'NO_2 <= 98.25\ngini = 0.429\nsa
mples = 189\nvalue = [210, 95]\nclass = Yes'),
 Text(2864.4, 155.3142857142857, 'gini = 0.399\nsamples = 181\nvalue = [208,
79]\nclass = Yes'),
 Text(2938.8, 155.3142857142857, 'gini = 0.198\nsamples = 8\nvalue = [2, 16]
\nclass = No'),
 Text(3124.8, 776.5714285714287, 'CO <= 0.675\ngini = 0.498\nsamples = 327\nv
```

```
alue = [254, 292]\nclass = No'),
 Text(3050.4, 465.9428571428573, 'O_3 <= 34.815\ngini = 0.477\nsamples = 213
\nvalue = [220, 142]\nclass = Yes'),
 Text(3013.2000000000003, 155.3142857142857, 'gini = 0.476\nsamples = 96\nval
ue = [65, 101]\nclass = No'),
 Text(3087.6000000000004, 155.3142857142857, 'gini = 0.331\nsamples = 117\nva
lue = [155, 41]\nclass = Yes'),
 Text(3199.2000000000003, 465.9428571428573, 'NOx <= 174.65\ngini = 0.301\nsa
mples = 114\nvalue = [34, 150]\nclass = No'),
 Text(3162.0000000000005, 155.3142857142857, 'gini = 0.497\nsamples = 41\nval
ue = [28, 33]\nclass = No'),
 Text(3236.4, 155.3142857142857, 'gini = 0.093\nsamples = 73\nvalue = [6, 11
7]\nclass = No'),
 Text(3868.8, 1397.8285714285716, 'NMHC <= 0.455\ngini = 0.451\nsamples = 112
6\nvalue = [608, 1161]\nclass = No'),
 Text(3571.2000000000003, 1087.2, 'NOx <= 118.95\ngini = 0.495\nsamples = 795
\nvalue = [563, 685]\nclass = No'),
 Text(3422.4, 776.5714285714287, 'NOx <= 48.73\ngini = 0.484\nsamples = 533\n
value = [490, 342]\nclass = Yes'),
 Text(3348.0000000000005, 465.9428571428573, 'BEN <= 0.365\ngini = 0.439\nsam
ples = 317\nvalue = [337, 162]\nclass = Yes'),
 Text(3310.8, 155.3142857142857, 'gini = 0.339\nsamples = 138\nvalue = [170,
47]\nclass = Yes'),
 Text(3385.2000000000003, 155.3142857142857, 'gini = 0.483\nsamples = 179\nva
lue = [167, 115]\nclass = Yes'),
 Text(3496.8, 465.9428571428573, 'PM25 <= 12.065\ngini = 0.497\nsamples = 216
\nvalue = [153, 180]\nclass = No'),
 Text(3459.6000000000004, 155.3142857142857, 'gini = 0.413\nsamples = 76\nval
ue = [85, 35]\nclass = Yes'),
 Text(3534.0000000000005, 155.3142857142857, 'gini = 0.435\nsamples = 140\nva
lue = [68, 145]\nclass = No'),
 Text(3720.0000000000005, 776.5714285714287, 'CO <= 0.655\ngini = 0.289\nsamp
les = 262\nvalue = [73, 343]\nclass = No'),
 Text(3645.6000000000004, 465.9428571428573, 'MXY <= 1.86\ngini = 0.431\nsamp
les = 120\nvalue = [62, 135]\nclass = No'),
 Text(3608.4, 155.3142857142857, 'gini = 0.474\nsamples = 26\nvalue = [27, 1
7]\nclass = Yes'),
 Text(3682.8, 155.3142857142857, 'gini = 0.353\nsamples = 94\nvalue = [35, 11
8]\nclass = No'),
 Text(3794.4, 465.9428571428573, 'PM25 <= 37.79\ngini = 0.095\nsamples = 142
\nvalue = [11, 208]\nclass = No'),
 Text(3757.2000000000003, 155.3142857142857, 'gini = 0.074\nsamples = 134\nva
lue = [8, 199]\nclass = No'),
 Text(3831.6000000000004, 155.3142857142857, 'gini = 0.375\nsamples = 8\nvalu
e = [3, 9]\nclass = No'),
 Text(4166.400000000001, 1087.2, 'TOL <= 2.01\ngini = 0.158\nsamples = 331\nv
alue = [45, 476]\nclass = No'),
 Text(4017.6000000000004, 776.5714285714287, 'NMHC <= 0.475\ngini = 0.268\nsa
mples = 129\nvalue = [32, 169]\nclass = No'),
 Text(3943.2000000000003, 465.9428571428573, 'PXY <= 0.625\ngini = 0.404\nsam
ples = 44\nvalue = [18, 46]\nclass = No'),
 Text(3906.0000000000005, 155.3142857142857, 'gini = 0.497\nsamples = 19\nval
ue = [13, 15]\nclass = No'),
 Text(3980.4, 155.3142857142857, 'gini = 0.239\nsamples = 25\nvalue = [5, 31]
\nclass = No'),
 Text(4092.0000000000005, 465.9428571428573, 'PM25 <= 12.15\ngini = 0.183\nsa
mples = 85\nvalue = [14, 123]\nclass = No'),
```

```
 Text(4054.8, 155.3142857142857, 'gini = 0.073\nsamples = 64\nvalue = [4, 10
1]\nclass = No'),
 Text(4129.200000000001, 155.3142857142857, 'gini = 0.43\nsamples = 21\nvalue
= [10, 22]\nclass = No'),
 Text(4315.200000000001, 776.5714285714287, 'SO_2 <= 9.17\ngini = 0.078\nsamp
les = 202\nvalue = [13, 307]\nclass = No'),
 Text(4240.8, 465.9428571428573, 'NO_2 <= 54.075\ngini = 0.024\nsamples = 98
\nvalue = [2, 164]\nclass = No'),
 Text(4203.6, 155.3142857142857, 'gini = 0.0\nsamples = 61\nvalue = [0, 106]
\nclass = No'),
 Text(4278.0, 155.3142857142857, 'gini = 0.064\nsamples = 37\nvalue = [2, 58]
\nclass = No'),
 Text(4389.6, 465.9428571428573, 'NOx <= 156.35\ngini = 0.133\nsamples = 104
\nvalue = [11, 143]\nclass = No'),
 Text(4352.400000000001, 155.3142857142857, 'gini = 0.265\nsamples = 48\nvalu
e = [11, 59]\nclass = No'),
 Text(4426.8, 155.3142857142857, 'gini = 0.0\nsamples = 56\nvalue = [0, 84]\n
class = No')]
```



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

```
Linear: 0.9130054463649523
Lasso: 0.5416068633176341
Ridge: 0.9130065705546464
ElasticNet: 0.8500784330712808
Logistic: 0.5553517620856806
Random Forest: 0.8690061763054464
```

Best Model is Ridge Regression

2008

In [41]:
```
df2=pd.read_csv("madrid_2008.csv")
df2
```

Out[41]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.88
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.04
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.27
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.85
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	37.16
...	...	...	...	...	...	...	...	...	...	...	
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	5.45
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.02
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.54
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.91
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	12.69

226392 rows × 17 columns

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

Out[43]:

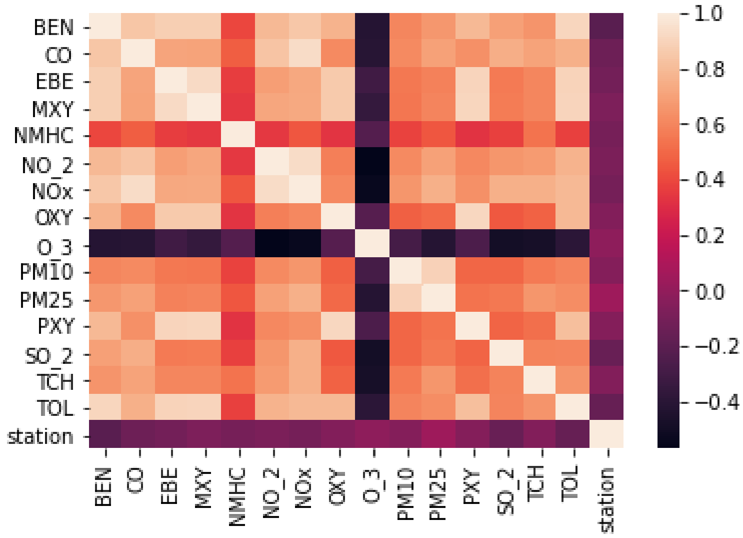
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160
21	2008-06-01 01:00:00	0.32	0.37	1.00	0.39	0.33	21.580000	22.180000	1.00	35.770000	7.900
25	2008-06-01 01:00:00	0.73	0.39	1.04	1.70	0.18	64.839996	86.709999	1.31	23.379999	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 [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

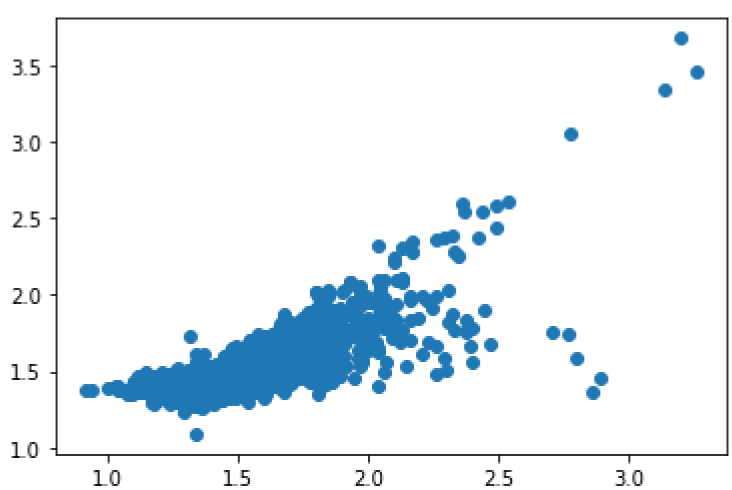
In [47]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[47]:
```
LinearRegression()
```

In [ ]:

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

Out[48]:
```
<matplotlib.collections.PathCollection at 0x2118db65550>
```



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

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

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

Out[51]:
```
2.0    12904
1.0    12727
Name: TCH, dtype: int64
```

In [ ]:

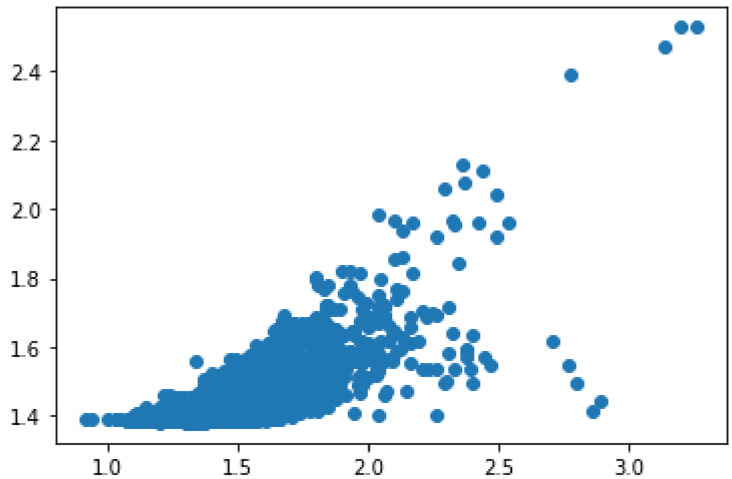
Lasso

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 0x2118dbbc100>



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

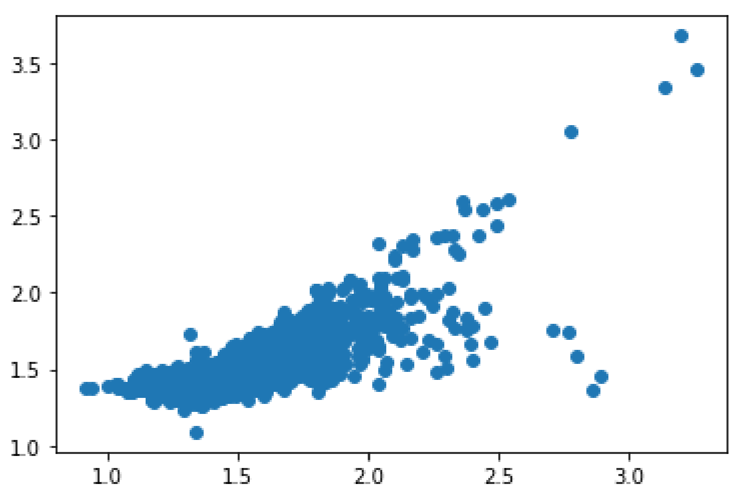
Ridge

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 0x2118dc096d0>



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

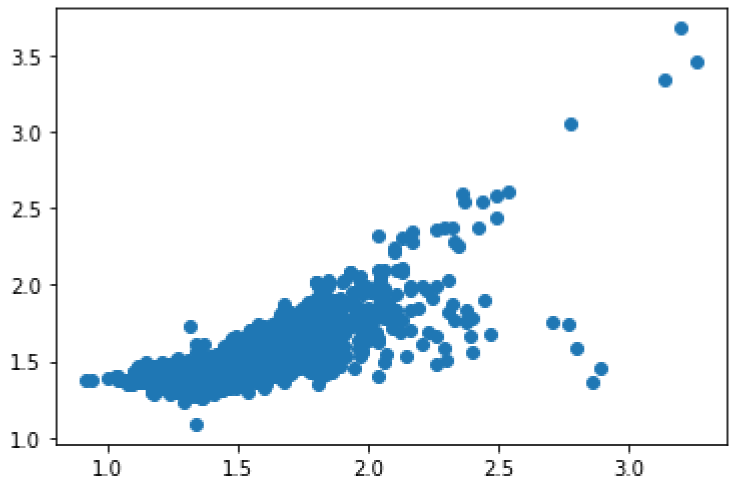
ElasticNet

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 0x2118dc5cca0>



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

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

0.6614787517671646

Out[61]: 0.6579137146941019

Logistic

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

Out[62]:
```
High    12904
Low     12727
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 0x2118d472880>

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="accu
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',"
```

Out[75]:
```
[Text(2217.053571428571, 2019.0857142857144, 'NOx <= 95.465\ngini = 0.5\nsa
mples = 11390\nvalue = [8921, 9020]\nclass = No'),
 Text(1101.0535714285713, 1708.457142857143, 'NO_2 <= 45.97\ngini = 0.411\n
samples = 7185\nvalue = [8055, 3272]\nclass = Yes'),
 Text(548.0357142857142, 1397.8285714285716, 'station <= 28079015.0\ngini =
0.364\nsamples = 5401\nvalue = [6447, 2024]\nclass = Yes'),
 Text(298.9285714285714, 1087.2, 'NMHC <= 0.255\ngini = 0.162\nsamples = 73
5\nvalue = [1047, 102]\nclass = Yes'),
 Text(159.42857142857142, 776.5714285714287, 'MXY <= 1.275\ngini = 0.107\ns
amples = 703\nvalue = [1043, 63]\nclass = Yes'),
 Text(79.71428571428571, 465.9428571428573, 'CO <= 0.425\ngini = 0.035\nsam
ples = 432\nvalue = [652, 12]\nclass = Yes'),
 Text(39.857142857142854, 155.3142857142857, 'gini = 0.024\nsamples = 426\n
value = [648, 8]\nclass = Yes'),
 Text(119.57142857142856, 155.3142857142857, 'gini = 0.5\nsamples = 6\nvalu
e = [4, 4]\nclass = Yes'),
 Text(239.1428571428571, 465.9428571428573, 'NO_2 <= 36.62\ngini = 0.204\ns
amples = 271\nvalue = [391, 51]\nclass = Yes'),
 Text(199.28571428571428, 155.3142857142857, 'gini = 0.105\nsamples = 130\n
```

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

```
Linear: 0.6614554468063345
Lasso: 0.4603230758526199
Ridge: 0.6614787517671646
ElasticNet: 0.5801002117823061
Logistic: 0.5009102730819246
Random Forest: 0.8317820819146347
```

Best model is Random Forest