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_2017
df
```

Out[2]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2017-06-01 01:00:00	NaN	NaN	0.3	NaN	NaN	4.0	38.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	28
1	2017-06-01 01:00:00	0.6	NaN	0.3	0.4	0.08	3.0	39.0	NaN	71.0	22.0	9.0	7.0	1.4	2.9	28
2	2017-06-01 01:00:00	0.2	NaN	NaN	0.1	NaN	1.0	14.0	NaN	NaN	NaN	NaN	NaN	NaN	0.9	28
3	2017-06-01 01:00:00	NaN	NaN	0.2	NaN	NaN	1.0	9.0	NaN	91.0	NaN	NaN	NaN	NaN	NaN	28
4	2017-06-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	19.0	NaN	69.0	NaN	NaN	2.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210115	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	27.0	NaN	65.0	NaN	NaN	NaN	NaN	NaN	28
210116	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	14.0	NaN	NaN	73.0	NaN	7.0	NaN	NaN	28
210117	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	4.0	NaN	83.0	NaN	NaN	NaN	NaN	NaN	28
210118	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	11.0	NaN	78.0	NaN	NaN	NaN	NaN	NaN	28
210119	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	14.0	NaN	77.0	60.0	NaN	NaN	NaN	NaN	28

210120 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210120 entries, 0 to 210119
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210120 non-null  object 
 1   BEN      50201 non-null   float64
 2   CH4      6410 non-null    float64
 3   CO       87001 non-null   float64
 4   EBE      49973 non-null   float64
 5   NMHC     25472 non-null   float64
 6   NO       209065 non-null  float64
 7   NO_2     209065 non-null  float64
 8   NOx      52818 non-null   float64
 9   O_3      121398 non-null  float64
 10  PM10     104141 non-null  float64
 11  PM25     52023 non-null   float64
 12  SO_2     86803 non-null   float64
 13  TCH      25472 non-null   float64
 14  TOL      50117 non-null   float64
 15  station  210120 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 25.6+ MB
```

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

Out[4]:

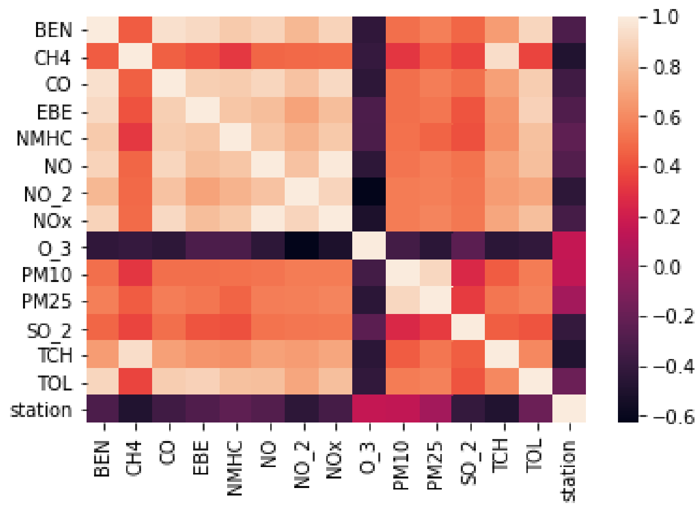
	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	s
87457	2017-10-01 01:00:00	0.6	1.22	0.3	0.4	0.09	4.0	54.0	60.0	43.0	12.0	9.0	13.0	1.31	2.3	280
87462	2017-10-01 01:00:00	0.2	1.18	0.2	0.1	0.09	1.0	26.0	28.0	42.0	14.0	6.0	3.0	1.27	1.1	280
87481	2017-10-01 02:00:00	0.4	1.22	0.2	0.2	0.06	2.0	32.0	36.0	53.0	14.0	10.0	13.0	1.28	1.3	280
87486	2017-10-01 02:00:00	0.2	1.19	0.2	0.1	0.07	1.0	15.0	17.0	51.0	18.0	8.0	3.0	1.26	0.8	280
87505	2017-10-01 03:00:00	0.3	1.23	0.2	0.2	0.06	2.0	27.0	29.0	57.0	15.0	10.0	13.0	1.29	1.0	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
158238	2017-12-31 22:00:00	0.3	1.11	0.2	0.1	0.03	1.0	8.0	9.0	73.0	3.0	1.0	3.0	1.14	0.2	280
158257	2017-12-31 23:00:00	0.6	1.38	0.3	0.1	0.03	6.0	42.0	51.0	47.0	7.0	4.0	3.0	1.41	0.9	280
158262	2017-12-31 23:00:00	0.3	1.11	0.2	0.1	0.03	1.0	6.0	8.0	72.0	6.0	3.0	3.0	1.14	0.2	280
158281	2018-01-01 00:00:00	0.5	1.38	0.2	0.1	0.02	2.0	20.0	23.0	69.0	4.0	2.0	3.0	1.39	0.6	280
158286	2018-01-01 00:00:00	0.3	1.11	0.2	0.1	0.03	1.0	1.0	3.0	83.0	8.0	5.0	3.0	1.14	0.2	280

4127 rows × 16 columns

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

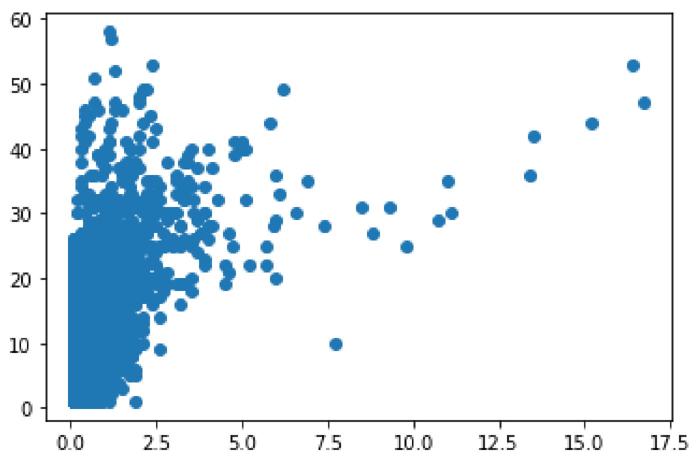
In [6]:
```
sns.heatmap(df1.corr())
```

Out[6]: <AxesSubplot:>



In [7]:
```
plt.plot(df1["EBE"],df1["PM25"],"o")
```

Out[7]: [<matplotlib.lines.Line2D at 0x25f5f6d7fd0>]



In [8]:
```
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)
```

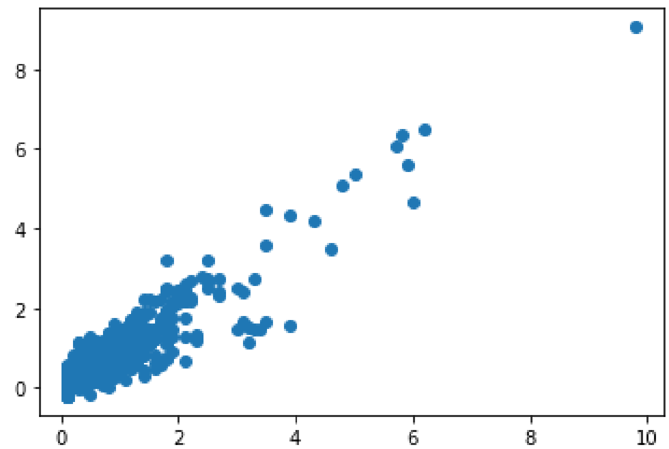
Linear

In [9]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[9]: LinearRegression()

In [10]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[10]: <matplotlib.collections.PathCollection at 0x25f5f7a3c70>



In [11]:
```
lis=li.score(x_test,y_test)
```

In [12]:
```
df1["TCH"].value_counts()
```

Out[12]:
```
1.24    124
1.36    118
1.26    112
1.25    110
1.41    107
       ... 
3.23      1
2.47      1
2.35      1
2.61      1
2.94      1
Name: TCH, Length: 164, dtype: int64
```

In [13]:
```
df1.loc[df1["TCH"]<1.40,"TCH"]=1
df1.loc[df1["TCH"]>1.40,"TCH"]=2
df1["TCH"].value_counts()
```

Out[13]:
```
1.0    2428
2.0    1699
Name: TCH, dtype: int64
```

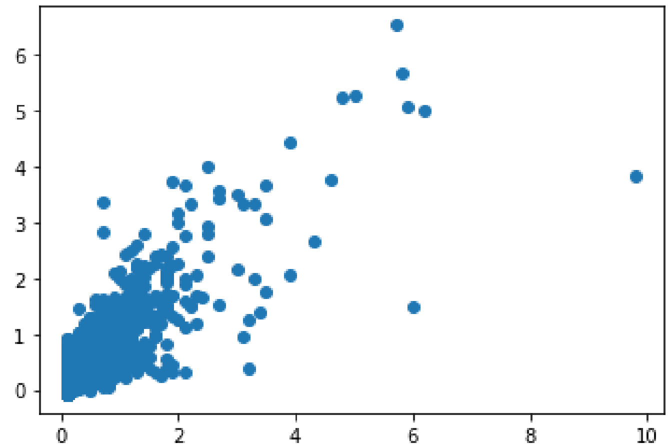
Lasso

In [14]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[14]: Lasso(alpha=5)

In [15]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[15]: <matplotlib.collections.PathCollection at 0x25f5f80f820>



In [16]:
```
las=la.score(x_test,y_test)
```

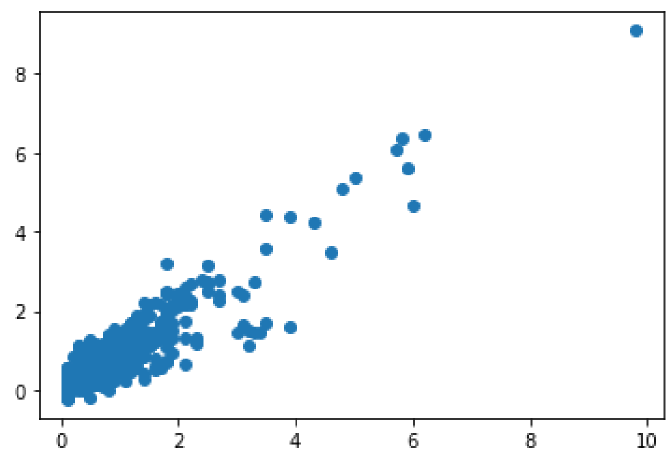
Ridge

In [17]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[17]: Ridge(alpha=1)

In [18]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[18]: <matplotlib.collections.PathCollection at 0x25f5f64b2e0>



In [19]:
```
rrs=rr.score(x_test,y_test)
```

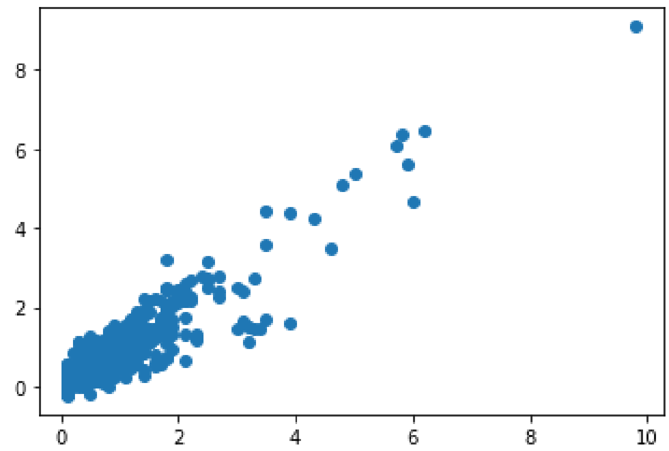
ElasticNet

In [20]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[20]:
```
ElasticNet()
```

In [21]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x25f60140640>
```



In [22]:
```
ens=en.score(x_test,y_test)
```

In [23]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.8446472469016151
```

Out[23]:
```
0.8976558488919304
```

Logistic

In [24]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["TCH"].value_counts()
```

Out[24]:
```
Low     2428
High    1699
Name: TCH, dtype: int64
```

In [25]:
```
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 [26]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[26]:
```
LogisticRegression()
```

In [27]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[27]: <matplotlib.collections.PathCollection at 0x25f601ad700>

In [28]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [29]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [30]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df1=df1.replace(g1)
```

In [31]:
```
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 [32]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[32]: RandomForestClassifier()

In [33]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [34]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[34]:
```
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 [35]:
```
rfcs=grid_search.best_score_
```

In [36]:
```
rfc_best=grid_search.best_estimator_
```

In [37]:
```
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
```

Out[37]:
```
[Text(1962.6206896551726, 2019.0857142857144, 'BEN <= 0.65\ngini = 0.477\nsamples =
1820\nvalue = [1757, 1131]\nclass = Yes'),
 Text(837.0000000000001, 1708.457142857143, 'NO_2 <= 24.5\ngini = 0.239\nsamples = 9
40\nvalue = [1312, 211]\nclass = Yes'),
 Text(384.82758620689657, 1397.8285714285716, 'CH4 <= 1.315\ngini = 0.069\nsamples =
362\nvalue = [567, 21]\nclass = Yes'),
 Text(230.89655172413796, 1087.2, 'CO <= 0.15\ngini = 0.004\nsamples = 342\nvalue =
[555, 1]\nclass = Yes'),
 Text(153.93103448275863, 776.5714285714287, 'CH4 <= 1.215\ngini = 0.021\nsamples =
60\nvalue = [93, 1]\nclass = Yes'),
 Text(76.96551724137932, 465.9428571428573, 'gini = 0.0\nsamples = 55\nvalue = [89,
0]\nclass = Yes'),
 Text(230.89655172413796, 465.9428571428573, 'gini = 0.32\nsamples = 5\nvalue = [4,
1]\nclass = Yes'),
 Text(307.86206896551727, 776.5714285714287, 'gini = 0.0\nsamples = 282\nvalue = [46
2, 0]\nclass = Yes'),
 Text(538.7586206896552, 1087.2, 'NMHC <= 0.025\ngini = 0.469\nsamples = 20\nvalue =
[12, 20]\nclass = No'),
 Text(461.79310344827593, 776.5714285714287, 'TOL <= 0.75\ngini = 0.245\nsamples = 1
```

In [38]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.8441268043977549
Lasso: 0.6393010463991731
Ridge: 0.8446472469016151
ElasticNet: 0.7899757472729623
Logistic: 0.579499596448749
Random Forest: 0.9681440443213296
```

Best Model is Random Forest

In [39]:
```
df2=pd.read_csv("C:/Users/user/Downloads/FP1_air/csvs_per_year/csvs_per_year/madrid_201
df2
```

Out[39]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	NaN	NaN
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.41	0.8
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	NaN	1.1
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN	NaN	NaN
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69091	2018-02-01 00:00:00	NaN	NaN	0.5	NaN	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN	NaN	NaN
69092	2018-02-01 00:00:00	NaN	NaN	0.7	NaN	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0	NaN	NaN
69093	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN	NaN	NaN
69094	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN	NaN	NaN
69095	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN	NaN	NaN

69096 rows × 16 columns

In [40]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69096 entries, 0 to 69095
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     69096 non-null  object 
 1   BEN      16950 non-null  float64
 2   CH4      8440 non-null   float64
 3   CO       28598 non-null  float64
 4   EBE      16949 non-null  float64
 5   NMHC     8440 non-null   float64
 6   NO       68826 non-null  float64
 7   NO_2     68826 non-null  float64
 8   NOx      68826 non-null  float64
 9   O_3      40049 non-null  float64
 10  PM10     36911 non-null  float64
 11  PM25     18912 non-null  float64
 12  SO_2     28586 non-null  float64
 13  TCH      8440 non-null   float64
 14  TOL      16950 non-null  float64
 15  station  69096 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 8.4+ MB
```

In [41]:
```
df3=df2.dropna()
df3
```

Out[41]:

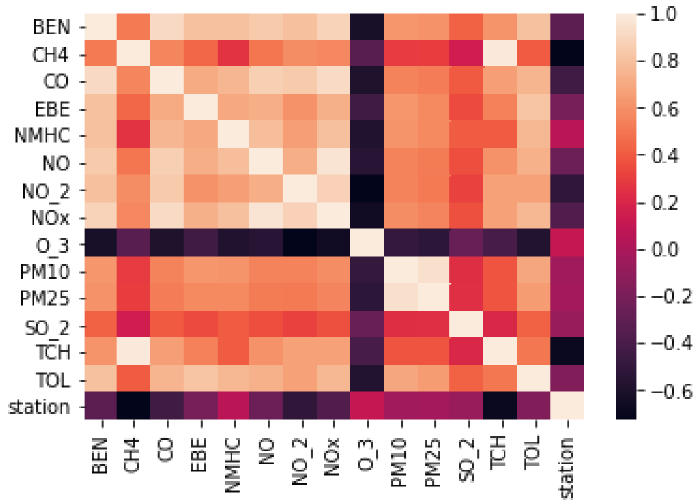
	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL	
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.41	0.8	28
6	2018-03-01 01:00:00	0.4	1.11	0.2	0.1	0.06	1.0	25.0	27.0	55.0	5.0	4.0	4.0	1.16	1.4	28
25	2018-03-01 02:00:00	0.4	1.42	0.2	0.1	0.01	4.0	26.0	32.0	64.0	4.0	4.0	3.0	1.44	0.7	28
30	2018-03-01 02:00:00	0.3	1.10	0.2	0.1	0.05	1.0	12.0	13.0	69.0	5.0	4.0	4.0	1.14	0.8	28
49	2018-03-01 03:00:00	0.3	1.41	0.2	0.1	0.01	3.0	16.0	20.0	68.0	3.0	2.0	3.0	1.42	0.4	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
69030	2018-01-31 22:00:00	1.8	1.21	0.7	1.7	0.19	151.0	129.0	361.0	1.0	45.0	26.0	11.0	1.40	11.9	28
69049	2018-01-31 23:00:00	3.1	1.87	1.2	2.0	0.35	296.0	162.0	615.0	3.0	39.0	23.0	8.0	2.22	12.5	28
69054	2018-01-31 23:00:00	1.6	1.17	0.6	1.4	0.15	127.0	106.0	301.0	1.0	43.0	25.0	8.0	1.32	10.3	28
69073	2018-02-01 00:00:00	3.2	1.53	1.0	2.1	0.19	125.0	117.0	309.0	3.0	37.0	24.0	6.0	1.72	13.0	28
69078	2018-02-01 00:00:00	1.3	1.14	0.4	0.8	0.10	54.0	73.0	155.0	1.0	27.0	16.0	5.0	1.24	6.8	28

4562 rows × 16 columns

In [42]:
```
df3=df3.drop(["date"],axis=1)
```

In [43]:
```
sns.heatmap(df3.corr())
```

Out[43]: <AxesSubplot:>



In [44]:
```
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

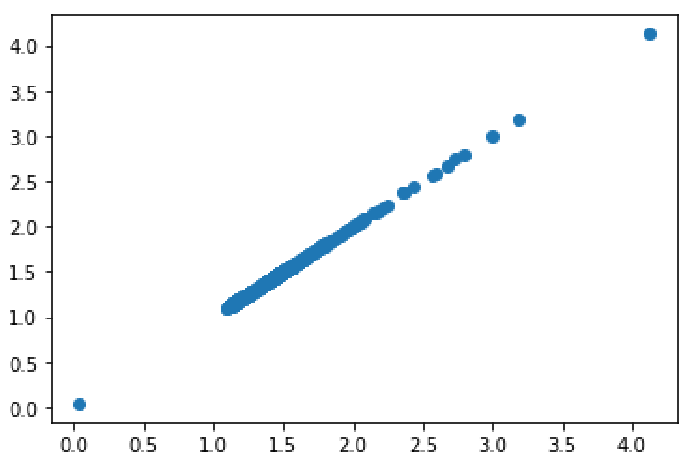
In [45]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[45]: LinearRegression()

In [ ]:

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

Out[46]: <matplotlib.collections.PathCollection at 0x25f605d5070>



In [47]:
```
lis=li.score(x_test,y_test)
```

In [48]:
```
df3["TCH"].value_counts()
```

Out[48]:
```
1.15    246
1.43    232
1.44    223
1.14    210
1.13    201
       ...
2.35      1
2.58      1
2.73      1
2.12      1
1.96      1
Name: TCH, Length: 143, dtype: int64
```

In [49]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[49]:
```
2.0    2477
1.0    2085
Name: TCH, dtype: int64
```

In [ ]:

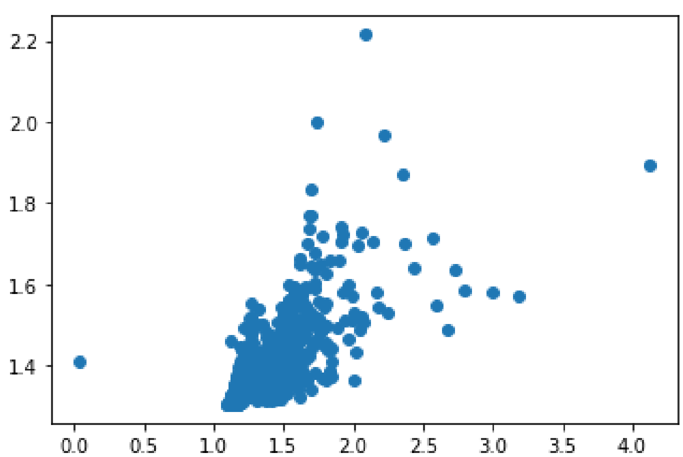
Lasso

In [50]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[50]:
```
Lasso(alpha=5)
```

In [51]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[51]:
```
<matplotlib.collections.PathCollection at 0x25f60639190>
```



In [52]:
```
las=la.score(x_test,y_test)
```

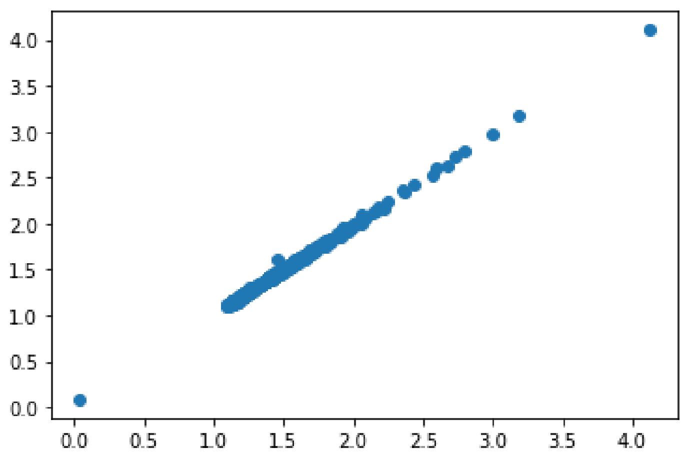
Ridge

In [53]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[53]: Ridge(alpha=1)

In [54]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[54]: <matplotlib.collections.PathCollection at 0x25f6068d760>



In [55]:
```
rrs=rr.score(x_test,y_test)
```

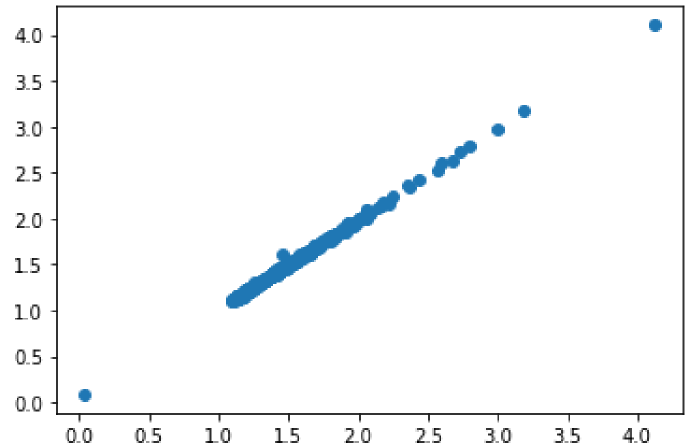
ElasticNet

In [56]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[56]: ElasticNet()

In [57]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[57]: <matplotlib.collections.PathCollection at 0x25f606f32b0>



In [58]:
```
ens=en.score(x_test,y_test)
```

In [59]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

0.9979283454674264

Out[59]: 0.998113461617783

Logistic

In [60]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[60]:
```
High    2477
Low     2085
Name: TCH, dtype: int64
```

In [61]:
```
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 [62]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[62]: LogisticRegression()

In [63]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[63]:
```
<matplotlib.collections.PathCollection at 0x25f6075c7f0>
```

In [64]:
```
los=lo.score(x_test,y_test)
```

Random Forest

In [65]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [66]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [67]:
```
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 [68]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[68]:
```
RandomForestClassifier()
```

In [69]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [70]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[70]:
```
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 [71]:
```
rfcs=grid_search.best_score_
```

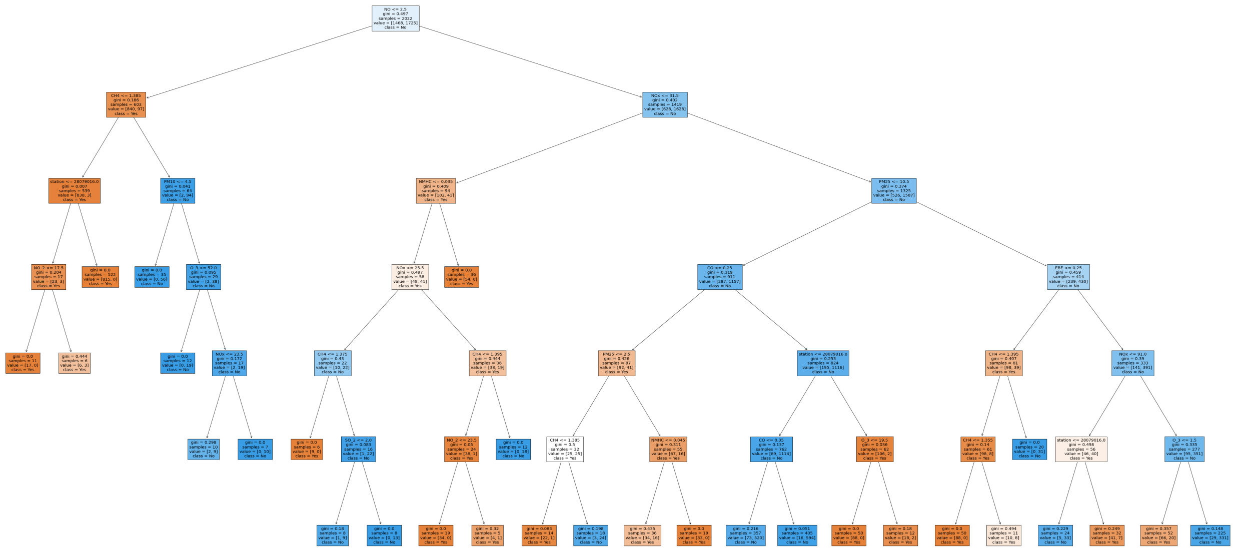
In [72]:
```
rfc_best=grid_search.best_estimator_
```

In [73]:
```
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
```

Out[73]:
```
[Text(1435.6875, 2019.0857142857144, 'NO <= 2.5\ngini = 0.497\nsamples = 2022\nvalue =
[1468, 1725]\nclass = No'),
 Text(465.0, 1708.457142857143, 'CH4 <= 1.385\ngini = 0.186\nsamples = 603\nvalue = [8
40, 97]\nclass = Yes'),
 Text(279.0, 1397.8285714285716, 'station <= 28079016.0\ngini = 0.007\nsamples = 539\n
value = [838, 3]\nclass = Yes'),
 Text(186.0, 1087.2, 'NO_2 <= 17.5\ngini = 0.204\nsamples = 17\nvalue = [23, 3]\nclass
 = Yes'),
 Text(93.0, 776.5714285714287, 'gini = 0.0\nsamples = 11\nvalue = [17, 0]\nclass = Ye
s'),
 Text(279.0, 776.5714285714287, 'gini = 0.444\nsamples = 6\nvalue = [6, 3]\nclass = Ye
s'),
 Text(372.0, 1087.2, 'gini = 0.0\nsamples = 522\nvalue = [815, 0]\nclass = Yes'),
 Text(651.0, 1397.8285714285716, 'PM10 <= 4.5\ngini = 0.041\nsamples = 64\nvalue = [2,
94]\nclass = No'),
 Text(558.0, 1087.2, 'gini = 0.0\nsamples = 35\nvalue = [0, 56]\nclass = No'),
 Text(744.0, 1087.2, 'O_3 <= 52.0\ngini = 0.095\nsamples = 29\nvalue = [2, 38]\nclass
 = No'),
 Text(651.0, 776.5714285714287, 'gini = 0.0\nsamples = 12\nvalue = [0, 19]\nclass = N
o'),
 Text(837.0, 776.5714285714287, 'NOx <= 23.5\ngini = 0.172\nsamples = 17\nvalue = [2,
19]\nclass = No'),
 Text(744.0, 465.9428571428573, 'gini = 0.298\nsamples = 10\nvalue = [2, 9]\nclass = N
o'),
 Text(930.0, 465.9428571428573, 'gini = 0.0\nsamples = 7\nvalue = [0, 10]\nclass = N
o'),
 Text(2406.375, 1708.457142857143, 'NOx <= 31.5\ngini = 0.402\nsamples = 1419\nvalue =
[628, 1628]\nclass = No'),
 Text(1581.0, 1397.8285714285716, 'NMHC <= 0.035\ngini = 0.409\nsamples = 94\nvalue =
[102, 41]\nclass = Yes'),
 Text(1488.0, 1087.2, 'NOx <= 25.5\ngini = 0.497\nsamples = 58\nvalue = [48, 41]\nclas
s = Yes'),
 Text(1209.0, 776.5714285714287, 'CH4 <= 1.375\ngini = 0.43\nsamples = 22\nvalue = [1
0, 22]\nclass = No'),
 Text(1116.0, 465.9428571428573, 'gini = 0.0\nsamples = 6\nvalue = [9, 0]\nclass = Ye
s'),
 Text(1302.0, 465.9428571428573, 'SO_2 <= 2.0\ngini = 0.083\nsamples = 16\nvalue = [1,
22]\nclass = No'),
 Text(1209.0, 155.3142857142857, 'gini = 0.18\nsamples = 8\nvalue = [1, 9]\nclass = N
o'),
 Text(1395.0, 155.3142857142857, 'gini = 0.0\nsamples = 8\nvalue = [0, 13]\nclass = N
o'),
 Text(1767.0, 776.5714285714287, 'CH4 <= 1.395\ngini = 0.444\nsamples = 36\nvalue = [3
8, 19]\nclass = Yes'),
 Text(1674.0, 465.9428571428573, 'NO_2 <= 23.5\ngini = 0.05\nsamples = 24\nvalue = [3
8, 1]\nclass = Yes'),
 Text(1581.0, 155.3142857142857, 'gini = 0.0\nsamples = 19\nvalue = [34, 0]\nclass = Y
es'),
 Text(1767.0, 155.3142857142857, 'gini = 0.32\nsamples = 5\nvalue = [4, 1]\nclass = Ye
s'),
 Text(1860.0, 465.9428571428573, 'gini = 0.0\nsamples = 12\nvalue = [0, 18]\nclass = N
o'),
 Text(1674.0, 1087.2, 'gini = 0.0\nsamples = 36\nvalue = [54, 0]\nclass = Yes'),
 Text(3231.75, 1397.8285714285716, 'PM25 <= 10.5\ngini = 0.374\nsamples = 1325\nvalue
= [526, 1587]\nclass = No'),
 Text(2604.0, 1087.2, 'CO <= 0.25\ngini = 0.319\nsamples = 911\nvalue = [287, 1157]\nc
lass = No'),
 Text(2232.0, 776.5714285714287, 'PM25 <= 2.5\ngini = 0.426\nsamples = 87\nvalue = [9
2, 41]\nclass = Yes'),
 Text(2046.0, 465.9428571428573, 'CH4 <= 1.385\ngini = 0.5\nsamples = 32\nvalue = [25,
25]\nclass = Yes'),
```

```
 Text(1953.0, 155.3142857142857, 'gini = 0.083\nsamples = 14\nvalue = [22, 1]\nclass =
Yes'),
 Text(2139.0, 155.3142857142857, 'gini = 0.198\nsamples = 18\nvalue = [3, 24]\nclass =
No'),
 Text(2418.0, 465.9428571428573, 'NMHC <= 0.045\ngini = 0.311\nsamples = 55\nvalue =
[67, 16]\nclass = Yes'),
 Text(2325.0, 155.3142857142857, 'gini = 0.435\nsamples = 36\nvalue = [34, 16]\nclass
= Yes'),
 Text(2511.0, 155.3142857142857, 'gini = 0.0\nsamples = 19\nvalue = [33, 0]\nclass = Y
es'),
 Text(2976.0, 776.5714285714287, 'station <= 28079016.0\ngini = 0.253\nsamples = 824\n
value = [195, 1116]\nclass = No'),
 Text(2790.0, 465.9428571428573, 'CO <= 0.35\ngini = 0.137\nsamples = 762\nvalue = [8
9, 1114]\nclass = No'),
 Text(2697.0, 155.3142857142857, 'gini = 0.216\nsamples = 357\nvalue = [73, 520]\nclas
s = No'),
 Text(2883.0, 155.3142857142857, 'gini = 0.051\nsamples = 405\nvalue = [16, 594]\nclas
s = No'),
 Text(3162.0, 465.9428571428573, 'O_3 <= 19.5\ngini = 0.036\nsamples = 62\nvalue = [10
6, 2]\nclass = Yes'),
 Text(3069.0, 155.3142857142857, 'gini = 0.0\nsamples = 50\nvalue = [88, 0]\nclass = Y
es'),
 Text(3255.0, 155.3142857142857, 'gini = 0.18\nsamples = 12\nvalue = [18, 2]\nclass =
Yes'),
 Text(3859.5, 1087.2, 'EBE <= 0.25\ngini = 0.459\nsamples = 414\nvalue = [239, 430]\nc
lass = No'),
 Text(3627.0, 776.5714285714287, 'CH4 <= 1.395\ngini = 0.407\nsamples = 81\nvalue = [9
8, 39]\nclass = Yes'),
 Text(3534.0, 465.9428571428573, 'CH4 <= 1.355\ngini = 0.14\nsamples = 61\nvalue = [9
8, 8]\nclass = Yes'),
 Text(3441.0, 155.3142857142857, 'gini = 0.0\nsamples = 50\nvalue = [88, 0]\nclass = Y
es'),
 Text(3627.0, 155.3142857142857, 'gini = 0.494\nsamples = 11\nvalue = [10, 8]\nclass =
Yes'),
 Text(3720.0, 465.9428571428573, 'gini = 0.0\nsamples = 20\nvalue = [0, 31]\nclass = N
o'),
 Text(4092.0, 776.5714285714287, 'NOx <= 91.0\ngini = 0.39\nsamples = 333\nvalue = [14
1, 391]\nclass = No'),
 Text(3906.0, 465.9428571428573, 'station <= 28079016.0\ngini = 0.498\nsamples = 56\nv
alue = [46, 40]\nclass = Yes'),
 Text(3813.0, 155.3142857142857, 'gini = 0.229\nsamples = 24\nvalue = [5, 33]\nclass =
No'),
 Text(3999.0, 155.3142857142857, 'gini = 0.249\nsamples = 32\nvalue = [41, 7]\nclass =
Yes'),
 Text(4278.0, 465.9428571428573, 'O_3 <= 1.5\ngini = 0.335\nsamples = 277\nvalue = [9
5, 351]\nclass = No'),
 Text(4185.0, 155.3142857142857, 'gini = 0.357\nsamples = 52\nvalue = [66, 20]\nclass
= Yes'),
 Text(4371.0, 155.3142857142857, 'gini = 0.148\nsamples = 225\nvalue = [29, 331]\nclas
s = No')]
```



In [74]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.9996222720077558
Lasso: 0.3728582371674869
Ridge: 0.9979283454674264
ElasticNet: 0.5889752318704515
Logistic: 0.5536888239590942
Random Forest: 0.9802696314989101
```

Best model is Linear Regression

In [ ]: