madrid_2009

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	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.260C
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.580C
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.190C
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.530C
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.760C
...	...	...	...	...	...	...	...	...	...	...	
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.830C
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.920C
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.460C
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.030C
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.360C

215688 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215688 entries, 0 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     215688 non-null  object 
 1   BEN      60082 non-null   float64
 2   CO       190801 non-null  float64
 3   EBE      60081 non-null   float64
 4   MXY      24846 non-null   float64
 5   NMHC     74748 non-null   float64
 6   NO_2     214562 non-null  float64
 7   NOx      214565 non-null  float64
 8   OXY      24854 non-null   float64
 9   O_3      204482 non-null  float64
 10  PM10     196331 non-null  float64
 11  PM25     55822 non-null   float64
 12  PXY      24854 non-null   float64
 13  SO_2     212671 non-null  float64
 14  TCH      75213 non-null   float64
 15  TOL      59920 non-null   float64
 16  station  215688 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 28.0+ MB
```

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

Out[4]:

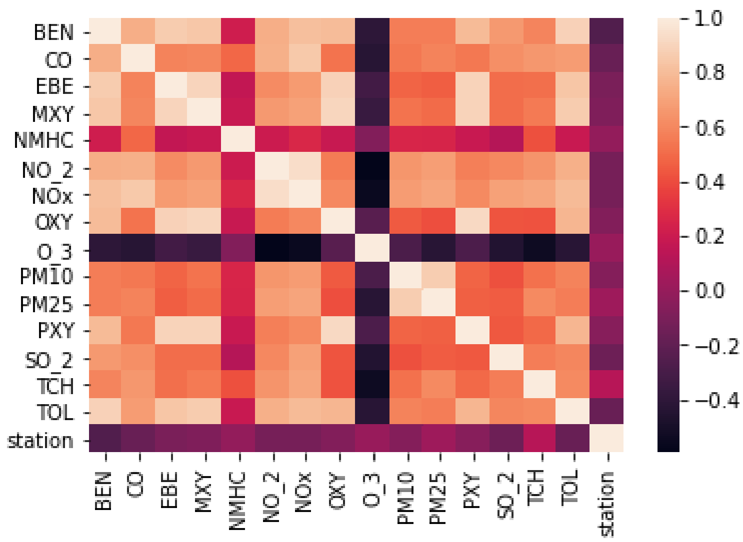
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM1
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.53000
20	2009-10-01 01:00:00	0.38	0.32	0.32	0.89	0.01	17.969999	19.240000	1.00	65.870003	10.52000
24	2009-10-01 01:00:00	0.55	0.24	0.65	1.79	0.18	36.619999	43.919998	1.28	48.070000	19.15000
28	2009-10-01 02:00:00	0.65	0.21	1.20	2.04	0.18	37.169998	48.869999	1.21	26.950001	32.20000
45	2009-10-01 02:00:00	0.38	0.30	0.50	1.15	0.00	17.889999	19.299999	1.00	60.009998	12.26000
...	...	...	...	...	...	...	...	...	...	...	.
215659	2009-05-31 23:00:00	0.54	0.27	1.00	0.69	0.09	28.280001	29.490000	0.86	78.750000	15.17000
215663	2009-05-31 23:00:00	0.74	0.35	1.13	1.65	0.15	56.410000	69.870003	1.26	56.799999	11.80000
215667	2009-06-01 00:00:00	0.78	0.29	0.99	1.96	0.04	64.870003	82.629997	1.13	58.000000	12.67000
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.83000
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.36000

24717 rows × 17 columns

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

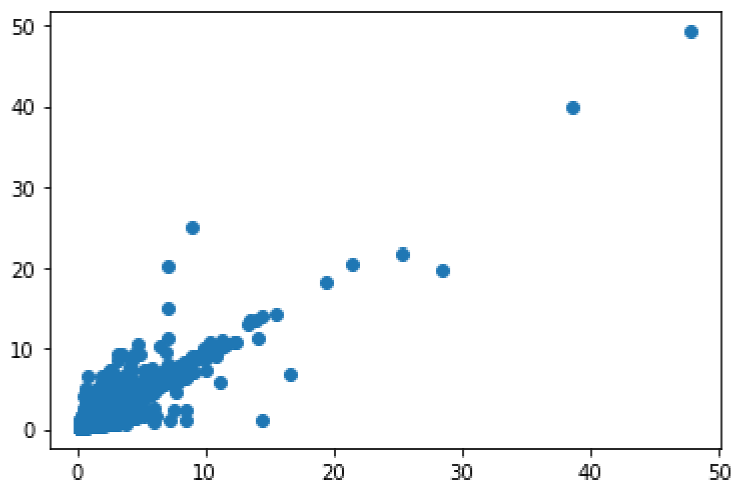
In [6]:
```python
sns.heatmap(df1.corr())
```

Out[6]:
```
<AxesSubplot:>
```



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

Out[7]:
```
[<matplotlib.lines.Line2D at 0x1b59f30f250>]
```



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

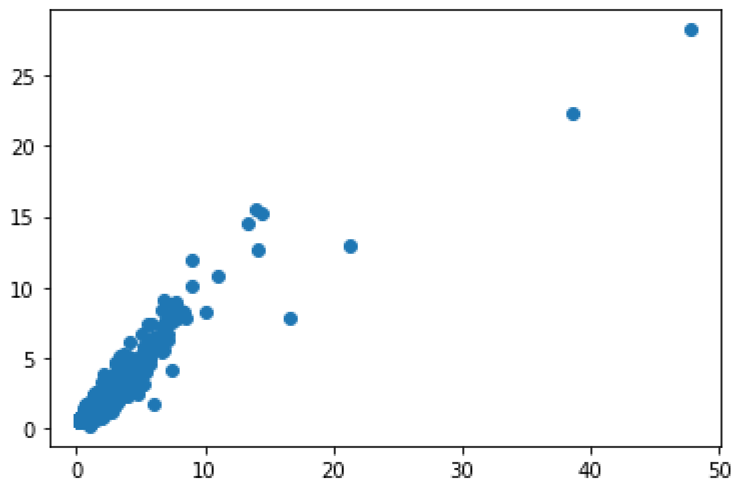
In [9]:
```python
# sns.stripplot(x=df["EBE"],y=df["PXY"],jitter=True,marker='o',color='blue')
```

In [10]:
```python
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 0x1b59f3c8f40>
```



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

In [14]:
```
df1["TCH"].value_counts()
```
Out[14]:
```
1.39    1091
1.36    1056
1.38    1046
1.40    1018
1.37    1017
        ...
2.52       1
1.16       1
2.41       1
1.13       1
2.79       1
Name: TCH, Length: 169, 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    12963
2.0    11754
Name: TCH, dtype: int64
```

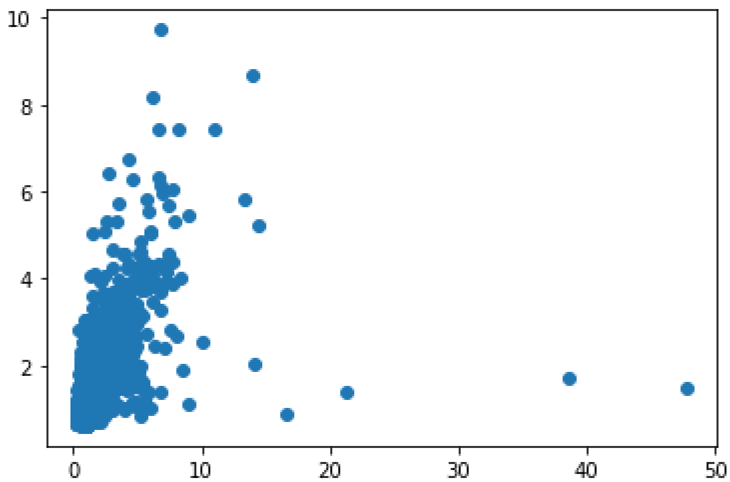
In [16]:
```
# Lasso
```

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 0x1b59ffde310>



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

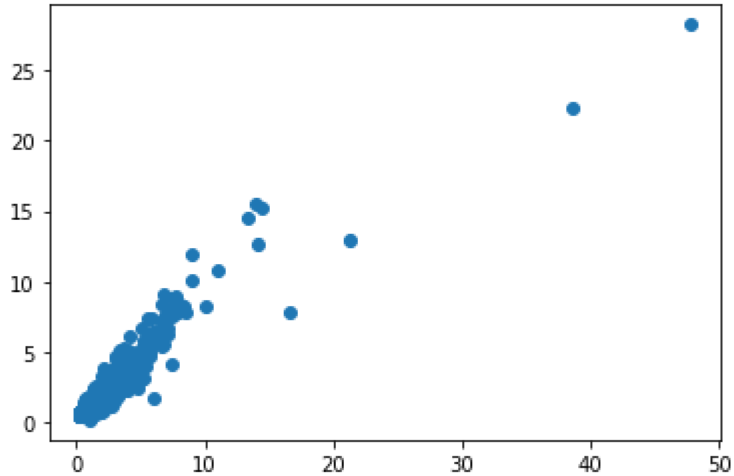
RIDGE

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 0x1b59f3326a0>



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

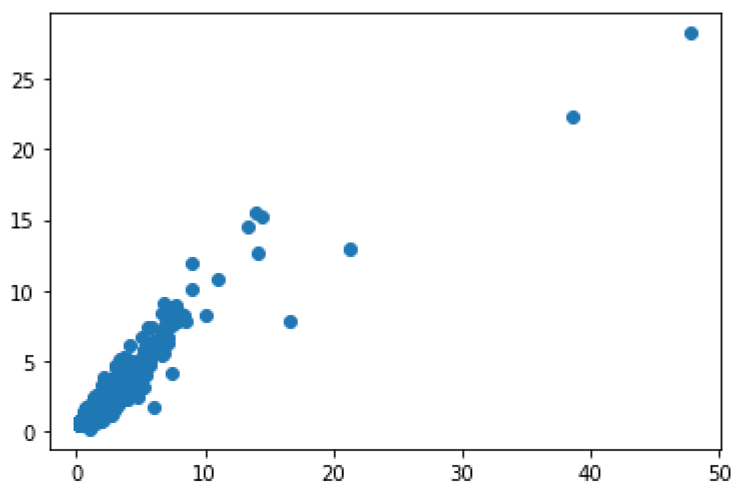
ElasticNet

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 0x1b5a005e2b0>



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

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

```
0.8655545107640262
```

Out[26]:
```
0.891088111281474
```

LOGISTIC

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

Out[27]:
```
Low     12963
High    11754
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 0x1b59fd047c0>
```

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')
```

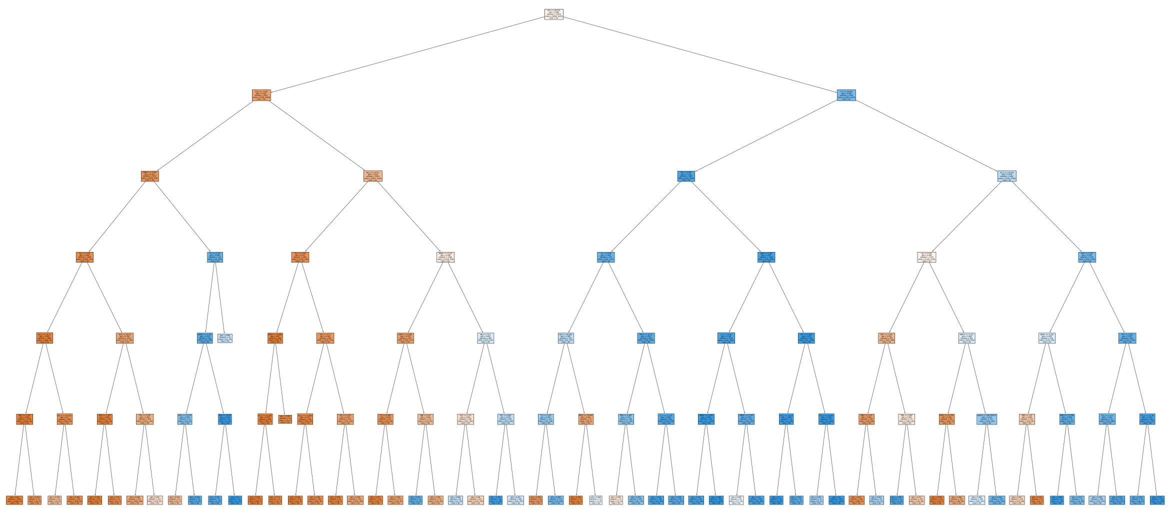
In [38]:
```
rfcs=grid_search.best_score_
```

In [39]:
```
rfc_best=grid_search.best_estimator_
```

In [40]:
```python
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(4425.517241379311, 155.3142857142857, 'gini = 0.031\nsamples = 118\nv
alue = [3, 189]\nclass = No')]
```



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

```
Linear: 0.8655713958460309
Lasso: 0.3459256380275074
Ridge: 0.8655545107640262
ElasticNet: 0.5761325583219384
Logistic: 0.5230582524271845
Random Forest: 0.8623202837321089
```

Best Model is Random Forest

madrid_2010

In [42]:
```
df2=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_2€
df2
```

Out[42]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	I
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	I
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	I
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.41C
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.67C
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	I
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	I
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	I
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.15C

209448 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209448 entries, 0 to 209447
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209448 non-null  object 
 1   BEN      60268 non-null   float64
 2   CO       94982 non-null   float64
 3   EBE      60253 non-null   float64
 4   MXY      6750 non-null    float64
 5   NMHC     51727 non-null   float64
 6   NO_2     208219 non-null  float64
 7   NOx      208210 non-null  float64
 8   OXY      6750 non-null    float64
 9   O_3      126684 non-null  float64
 10  PM10     106186 non-null  float64
 11  PM25     55514 non-null   float64
 12  PXY      6740 non-null    float64
 13  SO_2     93184 non-null   float64
 14  TCH      51730 non-null   float64
 15  TOL      60171 non-null   float64
 16  station  209448 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 27.2+ MB
```

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

Out[44]:

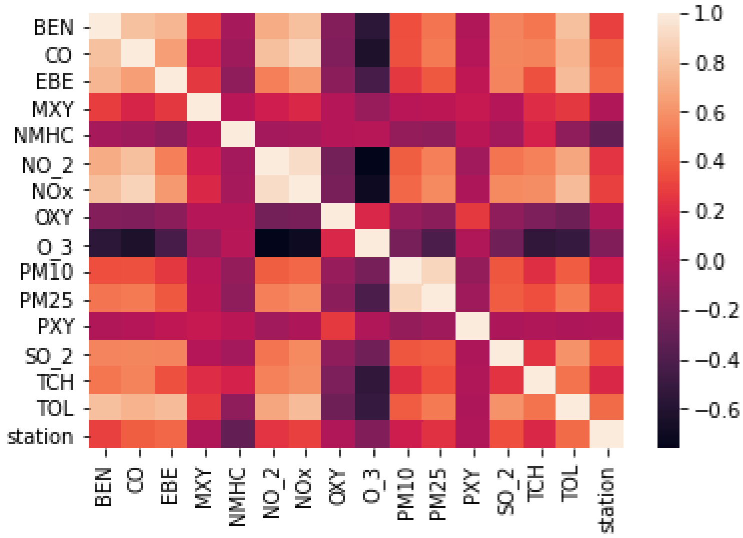
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM1
11	2010-03-01 01:00:00	0.78	0.18	0.84	0.73	0.28	10.420000	11.900000	1.0	90.309998	18.37000
23	2010-03-01 01:00:00	0.70	0.23	1.00	0.73	0.18	17.820000	22.290001	1.0	70.550003	23.63999
35	2010-03-01 02:00:00	0.58	0.17	0.84	0.73	0.28	3.500000	4.950000	1.0	68.849998	5.60000
47	2010-03-01 02:00:00	0.33	0.21	0.84	0.73	0.17	10.810000	14.900000	1.0	74.750000	7.89000
59	2010-03-01 03:00:00	0.38	0.16	0.64	1.00	0.26	2.750000	4.200000	1.0	93.629997	5.13000
...	...	...	...	...	...	...	...	...	...	...	.
191879	2010-05-31 22:00:00	0.60	0.26	0.82	0.13	0.16	33.360001	43.779999	1.0	38.459999	20.34000
191891	2010-05-31 23:00:00	0.41	0.16	0.71	0.19	0.10	24.299999	26.059999	1.0	50.290001	14.38000
191903	2010-05-31 23:00:00	0.57	0.28	0.64	0.19	0.18	35.540001	44.590000	1.0	34.020000	22.84000
191915	2010-06-01 00:00:00	0.34	0.16	0.69	0.22	0.10	23.559999	25.209999	1.0	45.930000	10.77000
191927	2010-06-01 00:00:00	0.43	0.25	0.79	0.22	0.18	34.910000	42.369999	1.0	29.540001	15.35000

6666 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)
```

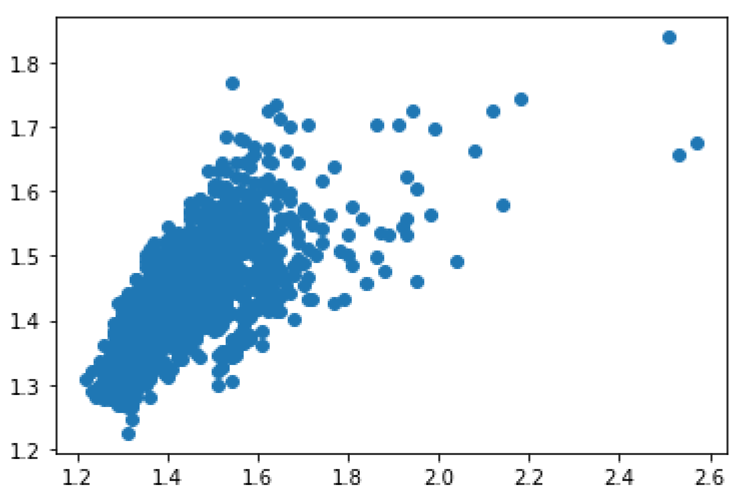
Linear

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 0x1b5a00dc4c0>
```



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

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

Out[51]:
```
1.36    364
1.38    351
1.39    324
1.35    323
1.37    321
       ...
2.07      1
2.17      1
2.53      1
2.12      1
2.05      1
Name: TCH, Length: 100, 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    3340
2.0    3326
Name: TCH, dtype: int64
```

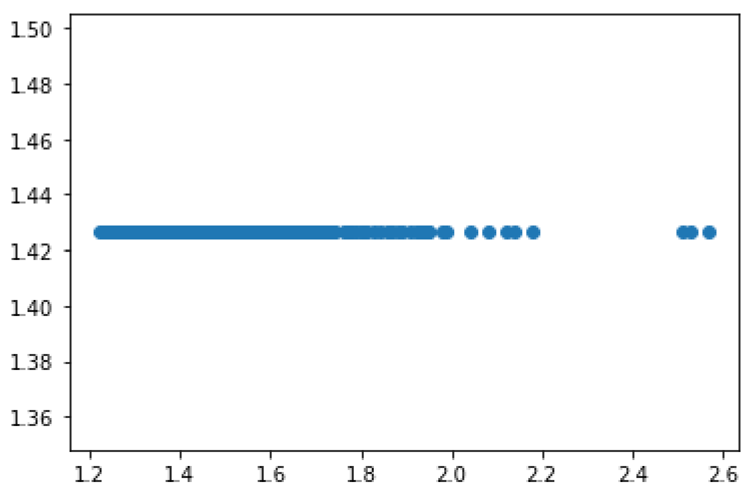
Lasso

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 0x1b5a01384c0>
```



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

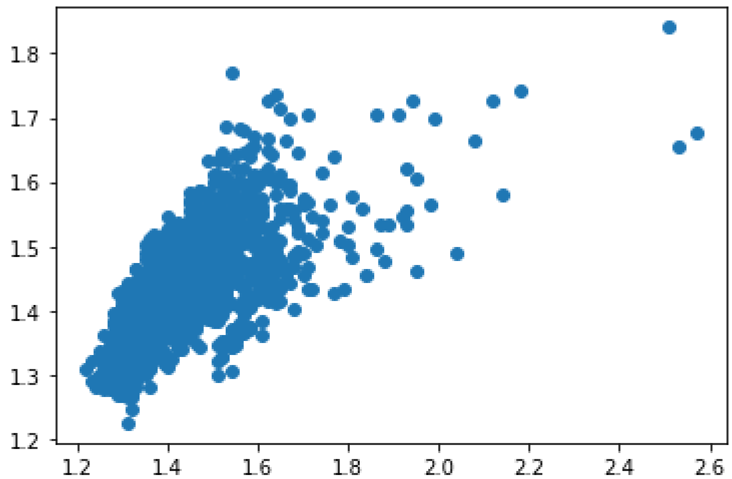
Ridge

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 0x1b5a01974c0>



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

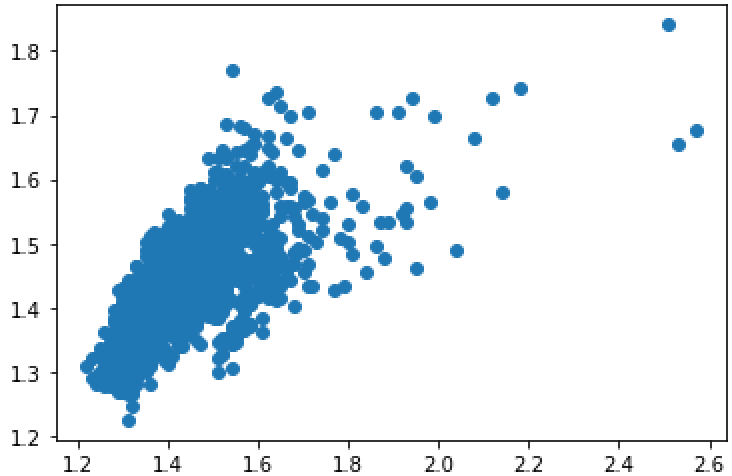
ElasticNet

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 0x1b5a01ebf40>



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

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

0.4588987162157072

Out[62]: 0.44755643852232574

Logistic

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

Out[63]:
```
Low     3340
High    3326
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 0x1b59fbd1ee0>

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(2080.734939759036, 2019.0857142857144, 'BEN <= 0.735\ngini = 0.5\nsam
ples = 2926\nvalue = [2353, 2313]\nclass = Yes'),
 Text(907.5903614457832, 1708.457142857143, 'O_3 <= 42.015\ngini = 0.464\ns
amples = 2045\nvalue = [2081, 1196]\nclass = Yes'),
 Text(255.4698795180723, 1397.8285714285716, 'MXY <= 0.185\ngini = 0.473\ns
amples = 387\nvalue = [239, 385]\nclass = No'),
 Text(107.56626506024097, 1087.2, 'CO <= 0.205\ngini = 0.108\nsamples = 21
\nvalue = [33, 2]\nclass = Yes'),
 Text(53.78313253012048, 776.5714285714287, 'gini = 0.0\nsamples = 10\nvalu
e = [18, 0]\nclass = Yes'),
 Text(161.34939759036143, 776.5714285714287, 'gini = 0.208\nsamples = 11\nv
alue = [15, 2]\nclass = Yes'),
 Text(403.3734939759036, 1087.2, 'PM10 <= 16.33\ngini = 0.455\nsamples = 36
6\nvalue = [206, 383]\nclass = No'),
 Text(268.9156626506024, 776.5714285714287, 'NOx <= 33.24\ngini = 0.489\nsa
mples = 218\nvalue = [150, 201]\nclass = No'),
 Text(161.34939759036143, 465.9428571428573, 'PM25 <= 5.665\ngini = 0.477\n
samples = 58\nvalue = [57, 37]\nclass = Yes'),
 Text(107.56626506024097, 155.3142857142857, 'gini = 0.26\nsamples = 15\nva
```

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

```
Linear: 0.45900314075246096
Lasso: -2.087909812176214e-05
Ridge: 0.4588987162157072
ElasticNet: 0.34488908831719434
Logistic: 0.487
Random Forest: 0.7801114444920703
```

In [ ]: