MADRID 2004

In [3]:
```
df2=pd.read_csv(r"C:\Users\user\Downloads\FP1_air\csvs_per_year\csvs_per_year\r
df2
```

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.99(
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	22.95(
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.48(
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.07(
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.08(
...	...	...	...	...	...	...	...	...	...	...	
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	30.90(
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.68(
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.84(
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.63(
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	24.38(

245496 rows × 17 columns

In [4]:
```
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 [5]:
```
df3=df2.dropna()
df3
```

Out[5]:

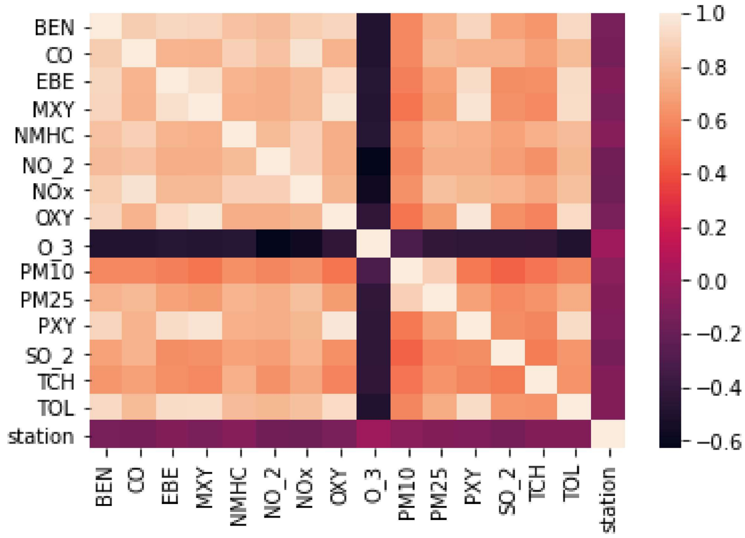
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.11(
22	2004-08-01 01:00:00	0.55	0.36	0.54	0.86	0.07	31.980000	32.799999	0.50	79.040001	43.54(
26	2004-08-01 01:00:00	1.80	0.46	2.28	4.62	0.21	62.259998	75.470001	2.47	54.419998	46.63(
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	86.93(
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	48.95(
...	...	...	...	...	...	...	...	...	...	...	
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	19.29(
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	31.80(
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	21.21(
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	30.90(
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	24.38(

19397 rows × 17 columns

In [6]:
```
df3=df3.drop(["date"],axis=1)
```

In [7]:
```
sns.heatmap(df3.corr())
```

Out[7]: <AxesSubplot:>



In [8]:
```
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

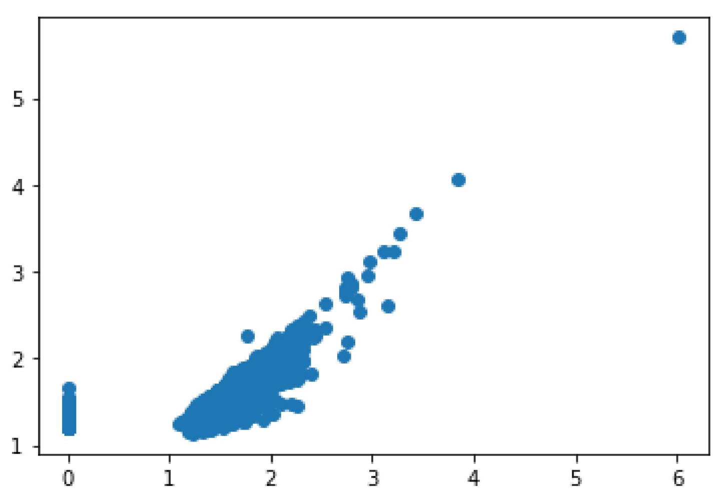
In [9]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[9]: LinearRegression()

In [ ]:

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

Out[10]: <matplotlib.collections.PathCollection at 0x238e4beb220>



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

In [12]:
```
df3["TCH"].value_counts()
```

Out[12]:
```
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 [13]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[13]:
```
1.0    11861
2.0     7536
Name: TCH, dtype: int64
```

In [ ]:

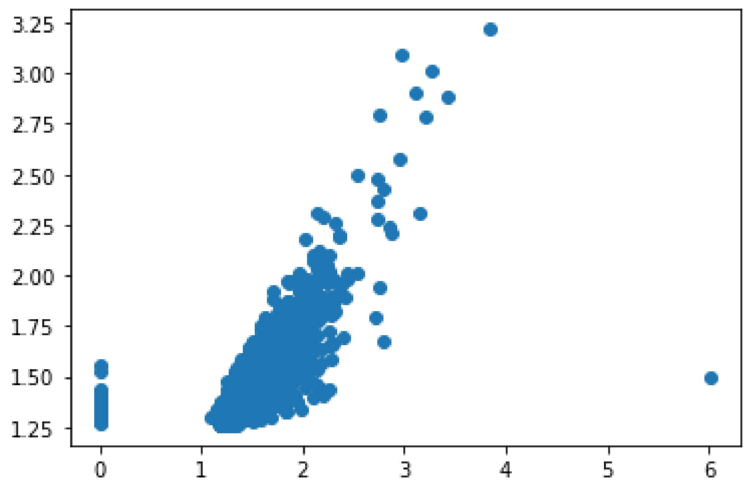
Lasso

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 0x238e4ca3e50>



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

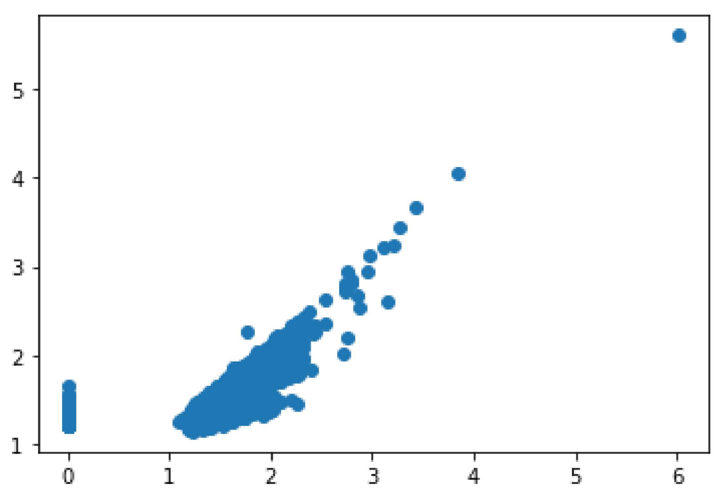
Ridge

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 0x238e5393f70>



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

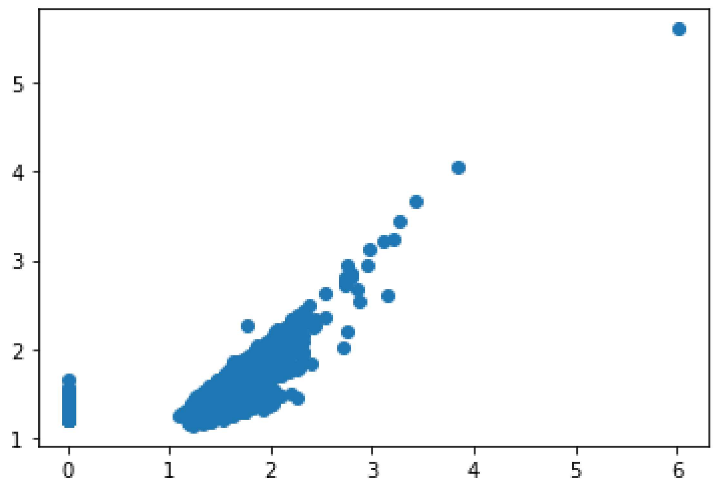
ElasticNet

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 0x238e4c587c0>



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

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

0.5859990677228715

Out[23]: 0.5914128037722362

Logistic

In [24]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[24]:
```
Low     11861
High     7536
Name: TCH, dtype: int64
```

In [25]:
```
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 [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 0x238e4ab94f0>

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}}
df3=df3.replace(g1)
```

In [31]:
```
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 [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="accu
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',"
```

```
841\nvalue = [4247, 205]\nclass = Yes'),
 Text(175.05882352941177, 776.5714285714287, 'O_3 <= 7.99\ngini = 0.043\nsa
mples = 2337\nvalue = [3587, 80]\nclass = Yes'),
 Text(87.52941176470588, 465.9428571428573, 'NO_2 <= 44.605\ngini = 0.406\n
samples = 43\nvalue = [43, 17]\nclass = Yes'),
 Text(43.76470588235294, 155.3142857142857, 'gini = 0.0\nsamples = 22\nvalu
e = [29, 0]\nclass = Yes'),
 Text(131.29411764705884, 155.3142857142857, 'gini = 0.495\nsamples = 21\nv
alue = [14, 17]\nclass = No'),
 Text(262.5882352941177, 465.9428571428573, 'NO_2 <= 34.065\ngini = 0.034\n
samples = 2294\nvalue = [3544, 63]\nclass = Yes'),
 Text(218.8235294117647, 155.3142857142857, 'gini = 0.011\nsamples = 1558\n
value = [2415, 14]\nclass = Yes'),
 Text(306.3529411764706, 155.3142857142857, 'gini = 0.08\nsamples = 736\nva
lue = [1129, 49]\nclass = Yes'),
 Text(525.1764705882354, 776.5714285714287, 'NMHC <= 0.135\ngini = 0.268\ns
amples = 504\nvalue = [660, 125]\nclass = Yes'),
 Text(437.6470588235294, 465.9428571428573, 'OXY <= 1.465\ngini = 0.201\nsa
mples = 461\nvalue = [641, 82]\nclass = Yes'),
 Text(393.88235294117646, 155.3142857142857, 'gini = 0.191\nsamples = 443\n
```

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

```
Linear: 0.5861025371823241
Lasso: 0.46524945960663167
Ridge: 0.5859990677228715
ElasticNet: 0.4893278610828047
Logistic: 0.6001718213058419
Random Forest: 0.8934226616021463
```

BEST MODEL IS RANDOM FOREST

In [ ]: