MADRID 2014

In [2]:
```
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 [3]:
```
df2=pd.read_csv(r"C:\Users\user\Downloads\FP1_air\csvs_per_year\csvs_per_year/
df2
```

Out[3]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN	28
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	28
2	2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1	28
3	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN	28
4	2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210019	2014-09-01 00:00:00	NaN	0.5	NaN	NaN	20.0	84.0	29.0	NaN	NaN	NaN	NaN	NaN	28
210020	2014-09-01 00:00:00	NaN	0.3	NaN	NaN	1.0	22.0	NaN	15.0	NaN	6.0	NaN	NaN	28
210021	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	13.0	70.0	NaN	NaN	NaN	NaN	NaN	28
210022	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	38.0	42.0	NaN	NaN	NaN	NaN	NaN	28
210023	2014-09-01 00:00:00	NaN	NaN	NaN	NaN	1.0	26.0	65.0	11.0	NaN	NaN	NaN	NaN	28

210024 rows × 14 columns

In [4]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210024 entries, 0 to 210023
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210024 non-null  object 
 1   BEN      46703 non-null   float64
 2   CO       87023 non-null   float64
 3   EBE      46722 non-null   float64
 4   NMHC     25021 non-null   float64
 5   NO       209154 non-null  float64
 6   NO_2     209154 non-null  float64
 7   O_3      121681 non-null  float64
 8   PM10     104311 non-null  float64
 9   PM25     51954 non-null   float64
 10  SO_2     87141 non-null   float64
 11  TCH      25021 non-null   float64
 12  TOL      46570 non-null   float64
 13  station  210024 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

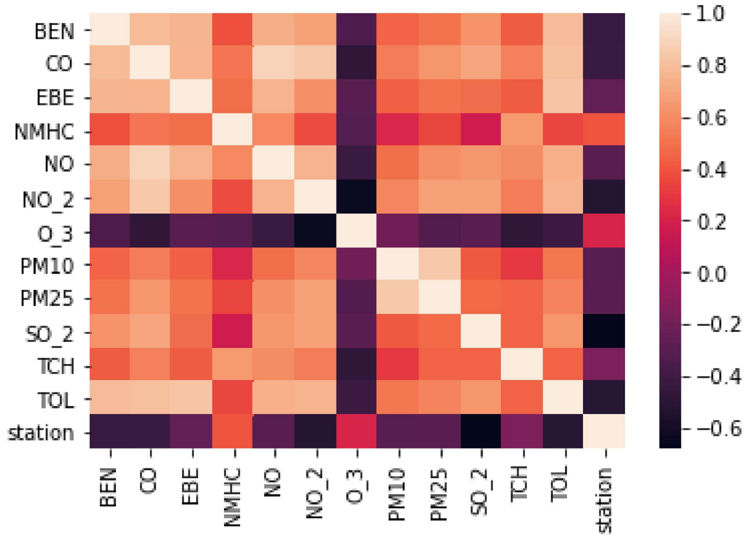
Out[5]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	280
6	2014-06-01 01:00:00	0.1	0.2	0.1	0.23	1.0	5.0	80.0	4.0	3.0	2.0	1.21	0.1	280
25	2014-06-01 02:00:00	0.2	0.2	0.1	0.11	4.0	21.0	63.0	9.0	6.0	5.0	1.36	0.8	280
30	2014-06-01 02:00:00	0.2	0.2	0.1	0.23	1.0	4.0	88.0	7.0	5.0	2.0	1.21	0.1	280
49	2014-06-01 03:00:00	0.1	0.2	0.1	0.11	4.0	18.0	66.0	9.0	7.0	6.0	1.36	0.9	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209958	2014-08-31 22:00:00	0.2	0.2	0.1	0.22	1.0	28.0	96.0	61.0	15.0	3.0	1.28	0.1	280
209977	2014-08-31 23:00:00	1.1	0.7	0.7	0.19	36.0	118.0	23.0	60.0	25.0	9.0	1.27	6.5	280
209982	2014-08-31 23:00:00	0.2	0.2	0.1	0.21	1.0	17.0	90.0	28.0	14.0	3.0	1.27	0.2	280
210001	2014-09-01 00:00:00	0.6	0.4	0.4	0.12	6.0	63.0	41.0	26.0	15.0	8.0	1.19	4.1	280
210006	2014-09-01 00:00:00	0.2	0.2	0.1	0.23	1.0	30.0	69.0	18.0	13.0	3.0	1.30	0.1	280

13946 rows × 14 columns

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

In [7]:
```
sns.heatmap(df3.corr())
```
Out[7]: <AxesSubplot:>



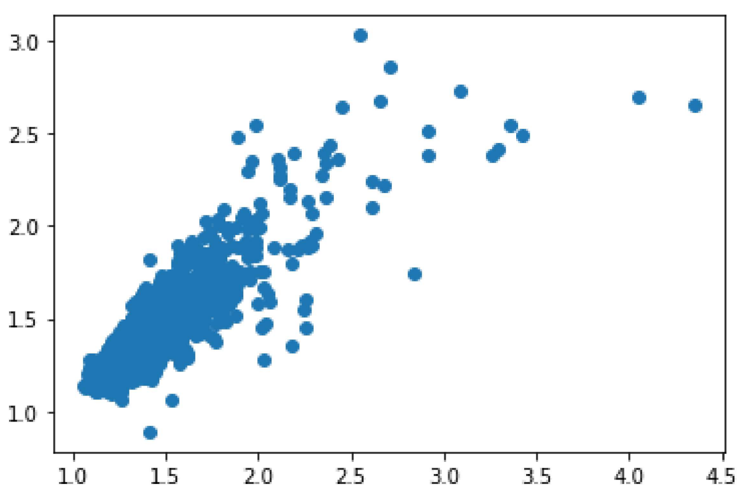
In [8]:
```
x=df3.drop(["TCH"],axis=1)
y=df3["TCH"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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 0x23e34647b20>



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

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

Out[12]:
```
1.37    601
1.36    598
1.34    529
1.35    528
1.38    515
       ...
2.50      1
2.86      1
2.70      1
3.04      1
4.37      1
Name: TCH, Length: 184, 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    9997
2.0    3949
Name: TCH, dtype: int64
```

In [ ]:

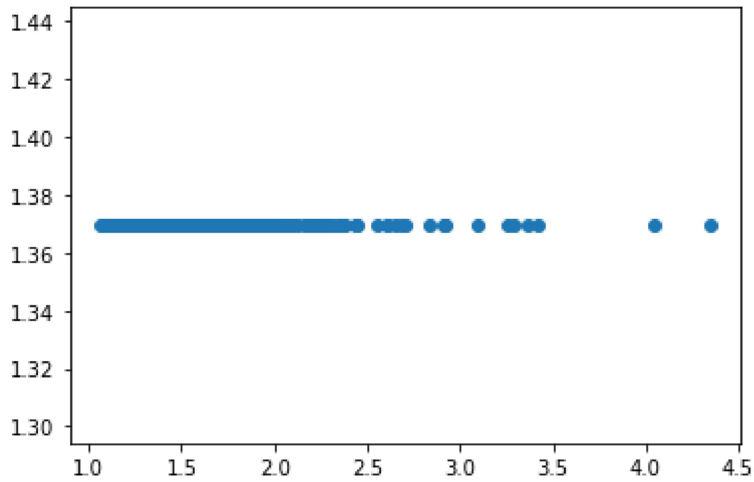
Lasso

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 0x23e34da5b50>



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

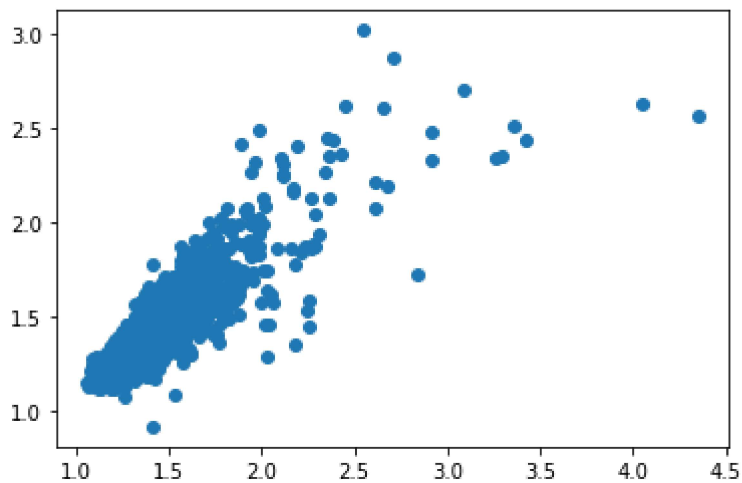
Ridge

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 0x23e34e097f0>



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

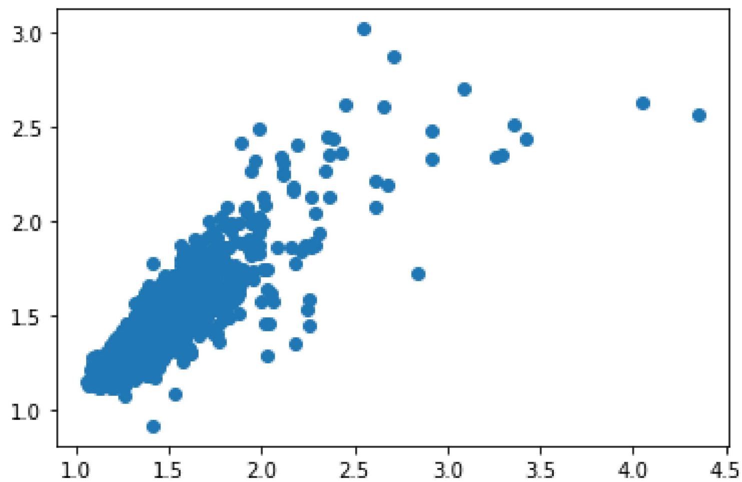
ElasticNet

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 0x23e34d86fa0>



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

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

```
0.7200667721423921
```

Out[23]: 0.7012758788869609

Logistic

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

Out[24]:
```
Low     9997
High    3949
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 0x23e34043e50>

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',"
```

Out[37]:
```
[Text(2469.446808510638, 2019.0857142857144, 'NMHC <= 0.275\ngini = 0.408\n
samples = 6178\nvalue = [6972, 2790]\nclass = Yes'),
 Text(1489.9787234042553, 1708.457142857143, 'BEN <= 0.35\ngini = 0.295\nsa
mples = 5152\nvalue = [6711, 1473]\nclass = Yes'),
 Text(759.8297872340426, 1397.8285714285716, 'NO_2 <= 20.5\ngini = 0.177\ns
amples = 3659\nvalue = [5221, 569]\nclass = Yes'),
 Text(379.9148936170213, 1087.2, 'NO <= 3.5\ngini = 0.054\nsamples = 2367\n
value = [3644, 104]\nclass = Yes'),
 Text(189.95744680851064, 776.5714285714287, 'NMHC <= 0.245\ngini = 0.045\n
samples = 2198\nvalue = [3413, 81]\nclass = Yes'),
 Text(94.97872340425532, 465.9428571428573, 'O_3 <= 31.5\ngini = 0.028\nsam
ples = 1640\nvalue = [2541, 36]\nclass = Yes'),
 Text(47.48936170212766, 155.3142857142857, 'gini = 0.272\nsamples = 26\nva
lue = [31, 6]\nclass = Yes'),
 Text(142.46808510638297, 155.3142857142857, 'gini = 0.023\nsamples = 1614
\nvalue = [2510, 30]\nclass = Yes'),
 Text(284.93617021276594, 465.9428571428573, 'NO_2 <= 16.5\ngini = 0.093\ns
amples = 558\nvalue = [872, 45]\nclass = Yes'),
 Text(237.4468085106383, 155.3142857142857, 'gini = 0.068\nsamples = 500\nv
```

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

```
Linear: 0.7193673186557659
Lasso: -2.9617947867599526e-06
Ridge: 0.7200667721423921
ElasticNet: 0.4410938925779051
Logistic: 0.7050669216061185
Random Forest: 0.8899815611555009
```

Best model is Random Forest

In [ ]: