MADRID 2001

In [5]:
```
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,ElasticNet
from sklearn.model_selection import train_test_split
```

In [6]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\FP1_air\csvs_per_year\csvs_per_year\madrid_2001.csv")
df
```

Out[6]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.000000	NaN	6.3400
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.1100
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.099998	NaN	7.8500
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.779999	NaN	6.4600
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.180000	NaN	8.8000
...	...	...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.880001	NaN	39.9100
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.809999	NaN	13.4500
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.770000	0.61	14.7000
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.889999	4.31	39.9199
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.369999	4.95	27.3400

217872 rows × 16 columns

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

In [8]:
```
df1=df.dropna()
df1
```

Out[8]:

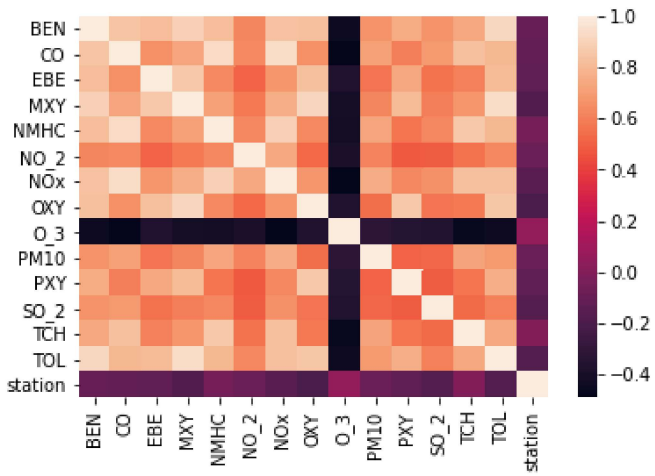
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.100000	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73
5	2001-08-01 01:00:00	2.11	0.63	2.48	5.940000	0.05	66.260002	118.099998	3.15	33.500000	122.699997	2.29
21	2001-08-01 01:00:00	0.80	0.43	0.71	1.200000	0.10	27.190001	29.700001	0.76	56.990002	114.300003	0.49
23	2001-08-01 01:00:00	1.29	0.34	1.41	3.090000	0.07	40.750000	51.570000	1.70	51.580002	102.199997	1.28
25	2001-08-01 02:00:00	0.87	0.06	0.88	2.410000	0.01	29.709999	31.440001	1.20	56.520000	56.290001	1.02
...	...	...	...	...	...	...	...	...	...	...	...	...
217829	2001-03-31 23:00:00	11.76	4.48	7.71	17.219999	0.89	103.900002	548.500000	7.62	9.680000	77.180000	6.14
217847	2001-03-31 23:00:00	9.79	2.65	7.59	9.730000	0.46	91.320000	315.899994	3.75	6.660000	52.740002	3.68
217849	2001-04-01 00:00:00	5.86	1.22	5.66	13.710000	0.25	64.370003	218.300003	6.46	7.480000	17.570000	5.48
217853	2001-04-01 00:00:00	14.47	1.83	11.39	26.059999	0.33	84.230003	259.200012	11.39	5.440000	36.740002	9.39
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.040000	0.18	76.809998	206.300003	5.20	8.340000	35.369999	4.95

29669 rows × 16 columns

In [9]:
```
df1=df1.drop(["date"],axis=1)
```

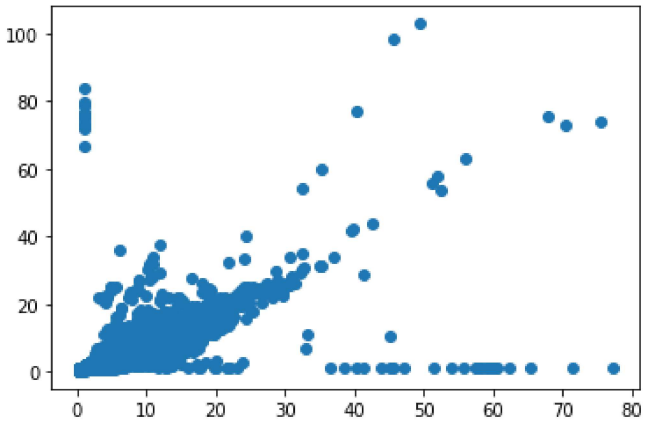
In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>



In [11]:
```
plt.plot(df1["EBE"],df1["PXY"],"o")
```

Out[11]: [<matplotlib.lines.Line2D at 0x2e4cd4d4e20>]



In [12]:
```
data=df[["EBE","PXY"]]
```

In [13]:
```
# sns.stripplot(x=df["EBE"],y=df["PXY"],jitter=True,marker='o',color='blue')
```

In [14]:
```
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)
```

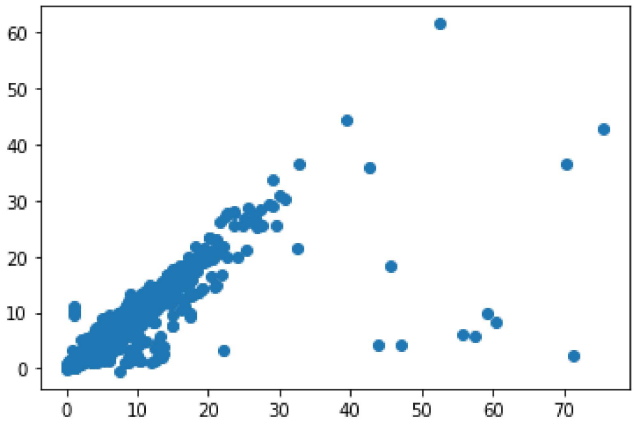
LINEAR REGRESSION

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 0x23d88b7d460>
```



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

In [14]:
```
df1["TCH"].value_counts()
```

Out[14]:
```
1.28    988
1.32    938
1.33    908
1.29    908
1.27    905
       ...
4.39      1
3.57      1
4.37      1
3.59      1
4.21      1
Name: TCH, Length: 269, 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    17204
2.0    12465
Name: TCH, dtype: int64
```

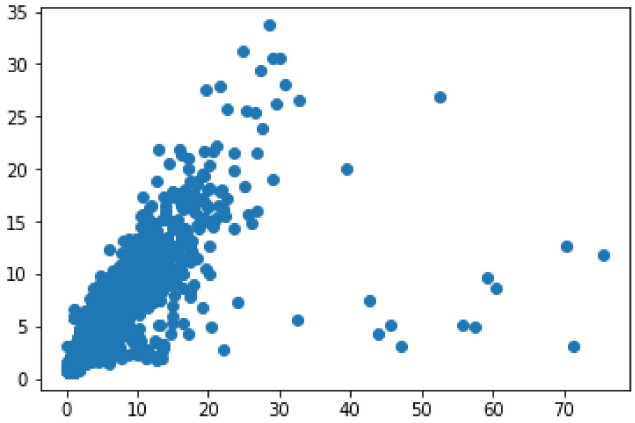
LASSO

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 0x23d88bec670>
```



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

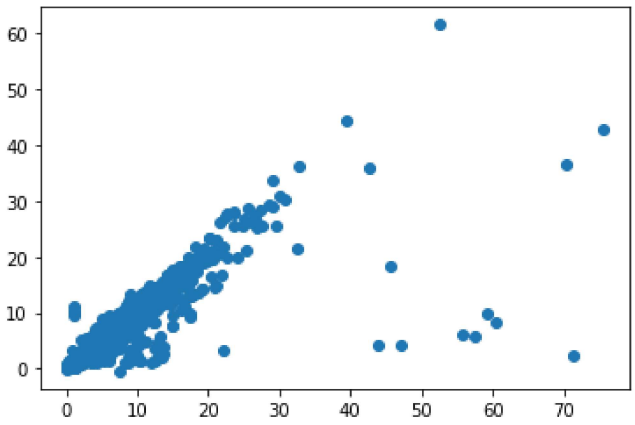
RIDGE

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 0x23d88ac8ee0>
```



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

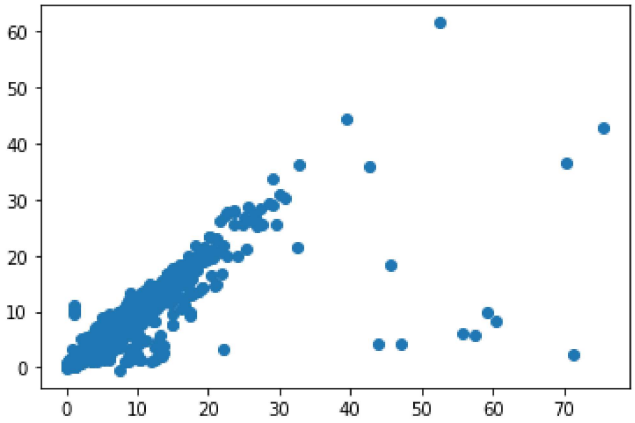
ELASTIC NET

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 0x23d89e36cd0>
```



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

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

```
0.7877953739336674
```

Out[25]:
```
0.759499092953202
```

LOGISTIC REGRESSION

In [43]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["TCH"].value_counts()
```

Out[43]:
```
Low     17204
High    12465
Name: TCH, dtype: int64
```

In [44]:
```
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 [45]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[45]:
```
LogisticRegression()
```

In [46]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[46]:
```
<matplotlib.collections.PathCollection at 0x23d896d8490>
```

In [47]:
```
los=lo.score(x_test,y_test)
```

RANDOM FOREST

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [31]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df1=df1.replace(g1)
```

In [32]:
```
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 [33]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[33]:
```
RandomForestClassifier()
```

In [34]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [35]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[35]:
```
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 [36]:
```
rfcs=grid_search.best_score_
```

In [37]:
```
rfc_best=grid_search.best_estimator_
```

In [38]:
```
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"],filled=True
```

Out[38]:
```
[Text(2387.0, 2019.0857142857144, 'CO <= 1.025\ngini = 0.488\nsamples = 13145\nvalue = [119
77, 8791]\nclass = Yes'),
 Text(1220.923076923077, 1708.457142857143, 'PM10 <= 25.05\ngini = 0.292\nsamples = 8587\nv
alue = [11173, 2407]\nclass = Yes'),
 Text(610.4615384615385, 1397.8285714285716, 'MXY <= 3.655\ngini = 0.169\nsamples = 5035\nv
alue = [7134, 732]\nclass = Yes'),
 Text(305.2307692307692, 1087.2, 'MXY <= 1.875\ngini = 0.074\nsamples = 2723\nvalue = [406
7, 163]\nclass = Yes'),
 Text(152.6153846153846, 776.5714285714287, 'SO_2 <= 19.11\ngini = 0.044\nsamples = 1310\nv
alue = [1978, 46]\nclass = Yes'),
 Text(76.3076923076923, 465.9428571428573, 'EBE <= 0.605\ngini = 0.039\nsamples = 1283\nval
ue = [1946, 40]\nclass = Yes'),
 Text(38.15384615384615, 155.3142857142857, 'gini = 0.021\nsamples = 651\nvalue = [1014, 1
1]\nclass = Yes'),
 Text(114.46153846153845, 155.3142857142857, 'gini = 0.059\nsamples = 632\nvalue = [932, 2
9]\nclass = Yes'),
 Text(228.9230769230769, 465.9428571428573, 'CO <= 0.81\ngini = 0.266\nsamples = 27\nvalue
= [32, 6]\nclass = Yes'),
 Text(190.76923076923077, 155.3142857142857, 'gini = 0.062\nsamples = 22\nvalue = [30, 1]\n
```

In [48]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.7877914082809474
Lasso: 0.6608359189741109
Ridge: 0.7877953739336674
ElasticNet: 0.7744562684843173
Logistic: 0.5802718795640939
Random Forest: 0.9163617103235747
```

BEST MODEL RANDOM FOREST

In [15]:
```
df2=pd.read_csv(r"C:\Users\user\Downloads\FP1_air\csvs_per_year\csvs_per_year\madrid_2002.csv")
df2
```

Out[15]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	T
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002	NaN	21.320000	N
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.660000	1
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001	NaN	13.670000	N
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000	NaN	16.990000	N
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002	NaN	15.260000	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000	NaN	13.210000	N
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999	NaN	15.640000	1
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.620000	1
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN	5.52	24.219999	N
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.910000	1

217296 rows × 16 columns

In [16]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217296 entries, 0 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217296 non-null  object 
 1   BEN      66747 non-null   float64
 2   CO       216637 non-null  float64
 3   EBE      58547 non-null   float64
 4   MXY      41255 non-null   float64
 5   NMHC     87045 non-null   float64
 6   NO_2     216439 non-null  float64
 7   NOx      216439 non-null  float64
 8   OXY      41314 non-null   float64
 9   O_3      216726 non-null  float64
 10  PM10     209113 non-null  float64
 11  PXY      41256 non-null   float64
 12  SO_2     216507 non-null  float64
 13  TCH      87115 non-null   float64
 14  TOL      66619 non-null   float64
 15  station  217296 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.5+ MB
```

In [17]:
```
df3=df2.dropna()
df3
```

Out[17]:

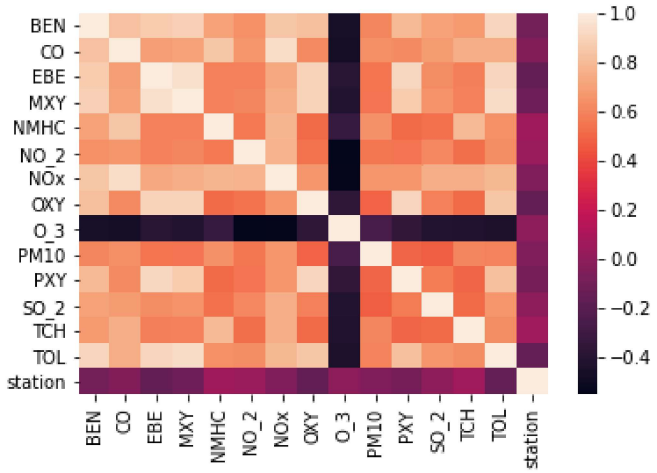
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2	TCH
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000	2.53	11.66	1.82
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.37	27.450001	2.98	14.78	1.83
22	2002-04-01 01:00:00	2.02	0.80	1.57	3.66	0.15	93.860001	101.300003	1.77	6.99	33.000000	1.48	1.98	1.54
24	2002-04-01 01:00:00	3.02	1.04	2.43	5.38	0.21	103.699997	195.399994	2.15	14.04	37.310001	2.18	15.91	1.79
26	2002-04-01 02:00:00	2.02	0.53	2.24	5.97	0.12	91.599998	136.199997	2.55	6.76	19.980000	2.45	10.15	2.08
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
217269	2002-10-31 23:00:00	1.24	0.28	1.26	2.64	0.11	60.080002	64.160004	1.23	15.64	13.910000	0.94	4.31	1.36
217271	2002-10-31 23:00:00	3.13	1.30	2.93	7.90	0.28	84.779999	184.000000	2.23	7.94	32.529999	3.40	13.66	1.53
217273	2002-11-01 00:00:00	2.50	0.97	3.63	9.95	0.19	61.759998	132.100006	4.46	5.45	29.500000	3.60	11.00	1.34
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000	0.94	5.62	1.43
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000	3.35	12.91	1.54

32381 rows × 16 columns

In [18]:
```
df3=df3.drop(["date"],axis=1)
```

In [19]:
```
sns.heatmap(df3.corr())
```

Out[19]: <AxesSubplot:>



In [20]:
```
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 REGRESSION

In [21]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

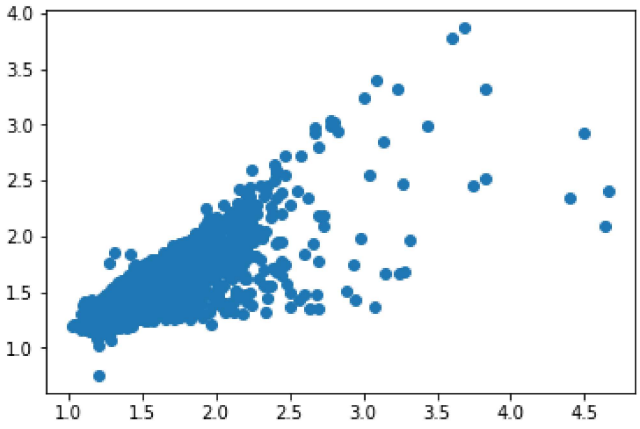
Out[21]: LinearRegression()

In [ ]:
```

```

In [22]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[22]: <matplotlib.collections.PathCollection at 0x2e4cd51cd30>



In [23]:
```
lis=li.score(x_test,y_test)
```

In [24]:
```
df3["TCH"].value_counts()
```

Out[24]:
```
1.29    1318
1.30    1253
1.27    1244
1.28    1232
1.31    1187
        ...
2.51       1
4.66       1
2.63       1
3.19       1
3.34       1
Name: TCH, Length: 232, dtype: int64
```

In [25]:
```
df3.loc[df3["TCH"]<1.40,"TCH"]=1
df3.loc[df3["TCH"]>1.40,"TCH"]=2
df3["TCH"].value_counts()
```

Out[25]:
```
1.0    21925
2.0    10456
Name: TCH, dtype: int64
```

In [ ]:
```

```

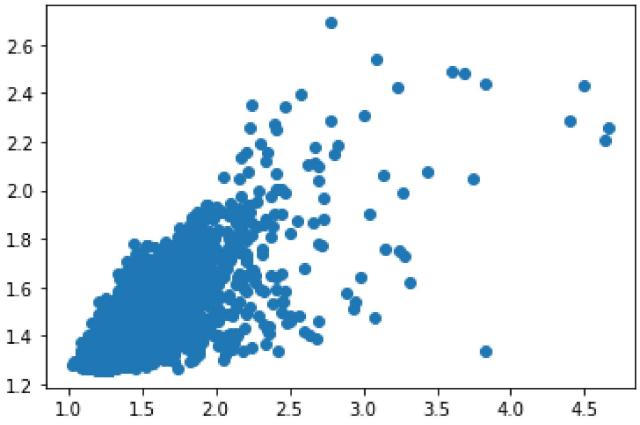
LASSO

In [26]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[26]:
```
Lasso(alpha=5)
```

In [27]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x2e4cda13490>
```



In [28]:
```
las=la.score(x_test,y_test)
```

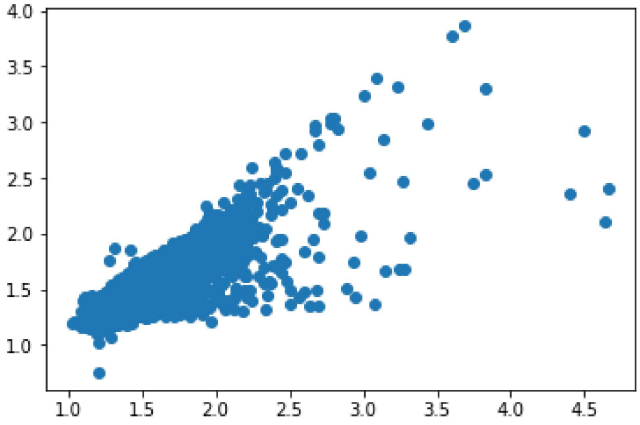
RIDGE

In [29]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[29]:
```
Ridge(alpha=1)
```

In [30]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[30]:
```
<matplotlib.collections.PathCollection at 0x2e4ce798100>
```



In [31]:
```
rrs=rr.score(x_test,y_test)
```

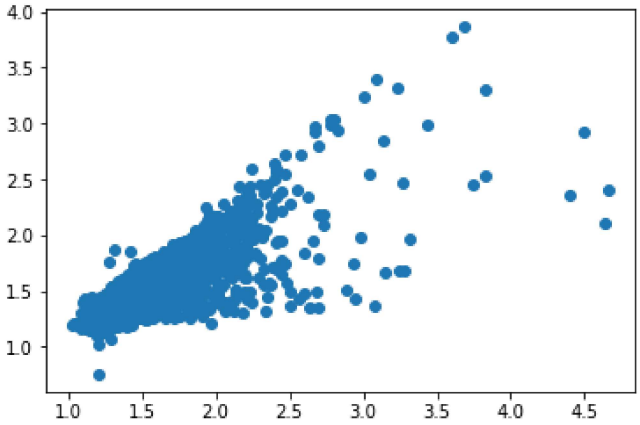
ELASTIC NET

In [32]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[32]:
```
ElasticNet()
```

In [33]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[33]:
```
<matplotlib.collections.PathCollection at 0x2e4ce7ef7f0>
```



In [34]:
```
ens=en.score(x_test,y_test)
```

In [35]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.7093266137603695
```

Out[35]:
```
0.7097231195252525
```

LOGISTIC

In [36]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df3=df3.replace(g)
df3["TCH"].value_counts()
```

Out[36]:
```
Low     21925
High    10456
Name: TCH, dtype: int64
```

In [37]:
```
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 [38]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[38]:
```
LogisticRegression()
```

In [39]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[39]:
```
<matplotlib.collections.PathCollection at 0x2e4ce8db340>
```

In [40]:
```
los=lo.score(x_test,y_test)
```

In [ ]:
```
# RANDOM FOREST
```

In [41]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [42]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df3=df3.replace(g1)
```

In [43]:
```
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 [44]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[44]:
```
RandomForestClassifier()
```

In [45]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [46]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[46]:
```
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 [47]:
```
rfcs=grid_search.best_score_
```

In [48]:
```
rfc_best=grid_search.best_estimator_
```

In [49]:
```
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"],filled=True
```

Out[49]:
```
[Text(2311.3706896551726, 2019.0857142857144, 'BEN <= 3.245\ngini = 0.436\nsamples = 14218
\nvalue = [15377, 7289]\nclass = Yes'),
 Text(1217.0172413793105, 1708.457142857143, 'CO <= 0.715\ngini = 0.303\nsamples = 10630\nv
alue = [13848, 3165]\nclass = Yes'),
 Text(615.7241379310345, 1397.8285714285716, 'NO_2 <= 55.045\ngini = 0.19\nsamples = 8096\n
value = [11563, 1378]\nclass = Yes'),
 Text(307.86206896551727, 1087.2, 'O_3 <= 14.995\ngini = 0.106\nsamples = 5972\nvalue = [89
53, 530]\nclass = Yes'),
 Text(153.93103448275863, 776.5714285714287, 'OXY <= 1.095\ngini = 0.455\nsamples = 478\nva
lue = [477, 256]\nclass = Yes'),
 Text(76.96551724137932, 465.9428571428573, 'PXY <= 1.275\ngini = 0.248\nsamples = 109\nval
ue = [130, 22]\nclass = Yes'),
 Text(38.48275862068966, 155.3142857142857, 'gini = 0.167\nsamples = 84\nvalue = [109, 11]
\nclass = Yes'),
 Text(115.44827586206898, 155.3142857142857, 'gini = 0.451\nsamples = 25\nvalue = [21, 11]
\nclass = Yes'),
 Text(230.89655172413796, 465.9428571428573, 'NO_2 <= 22.95\ngini = 0.481\nsamples = 369\nv
alue = [347, 234]\nclass = Yes'),
 Text(192.41379310344828, 155.3142857142857, 'gini = 0.245\nsamples = 10\nvalue = [2, 12]\n
class = No'),
```

In [50]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.7092997733375082
Lasso: 0.531980909734435
Ridge: 0.7093266137603695
ElasticNet: 0.589692144348577
Logistic: 0.6816263510036027
Random Forest: 0.8959675284567192
```

BEST MODEL RANDOM FOREST

MADRID 2003

In [54]:
```
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,ElasticNet
from sklearn.model_selection import train_test_split
```

In [55]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\FP1_air\csvs_per_year\csvs_per_year\madrid_2003.csv")
df
```

Out[55]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209999	NaN	24.299999
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389999	NaN	14.230000
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240002	NaN	17.879999
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839996	NaN	24.900000
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779999	NaN	18.750000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000	1.20	4.870000
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400000	0.50	8.360000
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.830000	NaN	5.330000
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.570000	NaN	6.830000
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000	2.43	6.060000

243984 rows × 16 columns

In [56]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243984 entries, 0 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     243984 non-null  object 
 1   BEN      69745 non-null   float64
 2   CO       225340 non-null  float64
 3   EBE      61244 non-null   float64
 4   MXY      42045 non-null   float64
 5   NMHC     111951 non-null  float64
 6   NO_2     242625 non-null  float64
 7   NOx      242629 non-null  float64
 8   OXY      42072 non-null   float64
 9   O_3      234131 non-null  float64
 10  PM10     240896 non-null  float64
 11  PXY      42063 non-null   float64
 12  SO_2     242729 non-null  float64
 13  TCH      111991 non-null  float64
 14  TOL      69439 non-null   float64
 15  station  243984 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 29.8+ MB
```

In [57]:
```
df1=df.dropna()
df1
```

Out[57]:

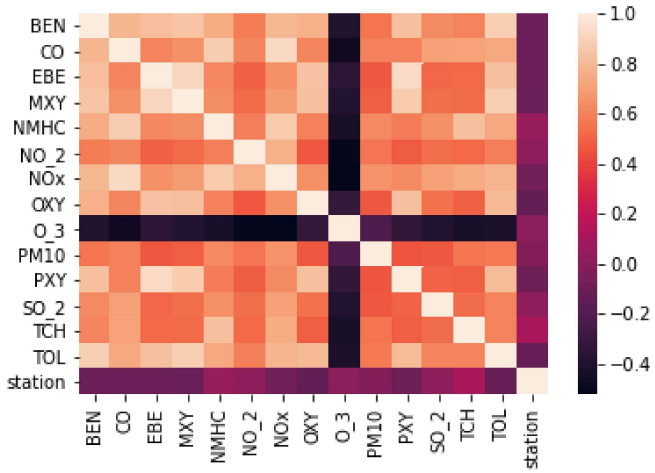
	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_
5	2003-03-01 01:00:00	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.950000	95.150002	7.94	29.2700
23	2003-03-01 01:00:00	3.46	1.27	3.43	7.08	0.18	54.250000	173.300003	3.37	6.540000	53.009998	2.62	8.8000
27	2003-03-01 01:00:00	6.39	1.79	5.75	10.88	0.33	75.459999	281.100006	3.68	6.690000	63.840000	4.24	18.4599
33	2003-03-01 02:00:00	7.42	1.47	10.63	24.73	0.35	83.309998	277.200012	11.00	9.900000	58.880001	8.93	24.7099
51	2003-03-01 02:00:00	3.62	1.29	3.20	7.08	0.19	42.209999	166.300003	3.41	6.380000	47.599998	2.70	8.4100
...	...	...	...	...	...	...	...	...	...	...	...	...	
243955	2003-09-30 23:00:00	1.75	0.41	3.07	9.38	0.09	46.290001	77.709999	3.11	18.280001	7.520000	3.48	6.4400
243957	2003-10-01 00:00:00	2.35	0.60	3.88	10.86	0.11	61.240002	133.100006	0.89	10.900000	10.240000	3.89	7.4000
243961	2003-10-01 00:00:00	2.97	0.82	4.53	10.88	0.05	36.529999	131.300003	5.52	12.940000	25.680000	4.13	5.6000
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380000	1.20	4.8700
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350000	2.43	6.0600

33010 rows × 16 columns

In [58]:
```
df1=df1.drop(["date"],axis=1)
```

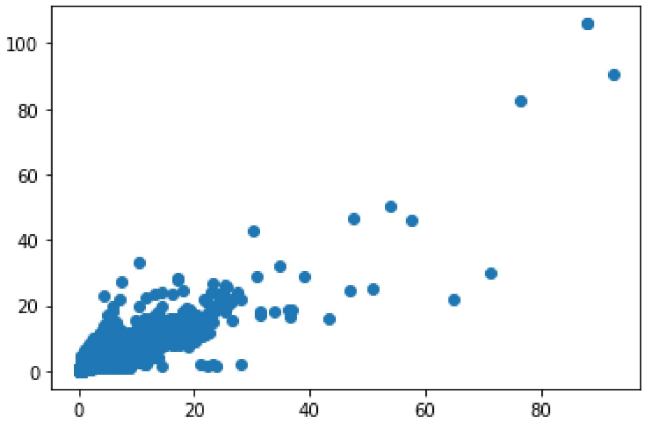
In [59]:
```
sns.heatmap(df1.corr())
```

Out[59]: <AxesSubplot:>



In [60]:
```
plt.plot(df1["EBE"],df1["PXY"],"o")
```

Out[60]: [<matplotlib.lines.Line2D at 0x2e4ced93220>]



In [61]:
```
data=df[["EBE","PXY"]]
```

In [64]:
```
x=df1.drop(["O_3"],axis=1)
y=df1["O_3"]
x_train,x_test,y_train,y_test=train_test_split(x,y)
```

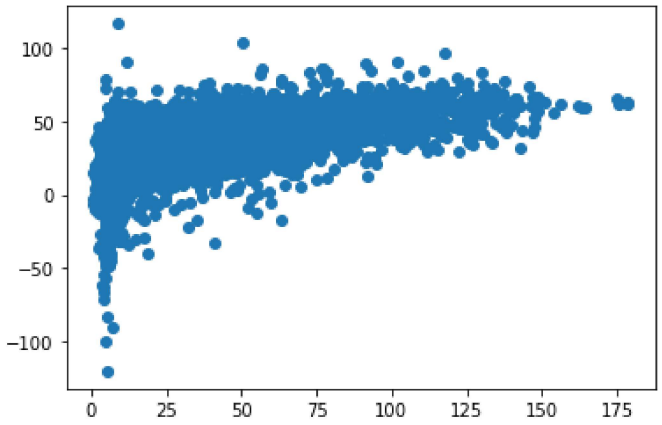
LINEAR REGRESSION

In [66]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[66]: LinearRegression()

In [67]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[67]:
```
<matplotlib.collections.PathCollection at 0x2e4cf361280>
```



In [68]:
```
lis=li.score(x_test,y_test)
```

In [69]:
```
df1["TCH"].value_counts()
```

Out[69]:
```
1.30    1344
1.31    1342
1.32    1281
1.27    1279
1.29    1262
        ...
3.50       1
3.87       1
3.21       1
3.14       1
1.01       1
Name: TCH, Length: 243, dtype: int64
```

In [70]:
```
df1.loc[df1["TCH"]<1.40,"TCH"]=1
df1.loc[df1["TCH"]>1.40,"TCH"]=2
df1["TCH"].value_counts()
```

Out[70]:
```
1.0    21614
2.0    11396
Name: TCH, dtype: int64
```

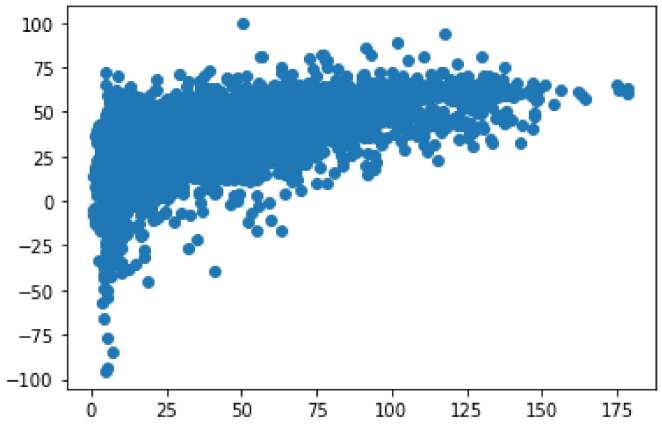
LASSO

In [71]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[71]:
```
Lasso(alpha=5)
```

In [72]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[72]: <matplotlib.collections.PathCollection at 0x2e4d061ffa0>



In [73]:
```
las=la.score(x_test,y_test)
```

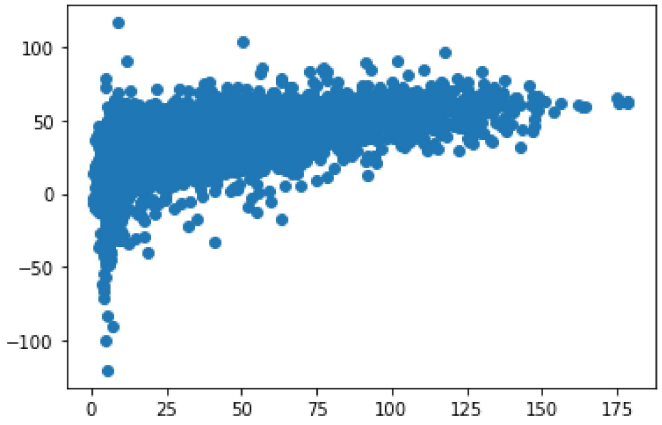
RIDGE

In [74]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[74]: Ridge(alpha=1)

In [75]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[75]: <matplotlib.collections.PathCollection at 0x2e4ce76e9a0>



In [76]:
```
rrs=rr.score(x_test,y_test)
```

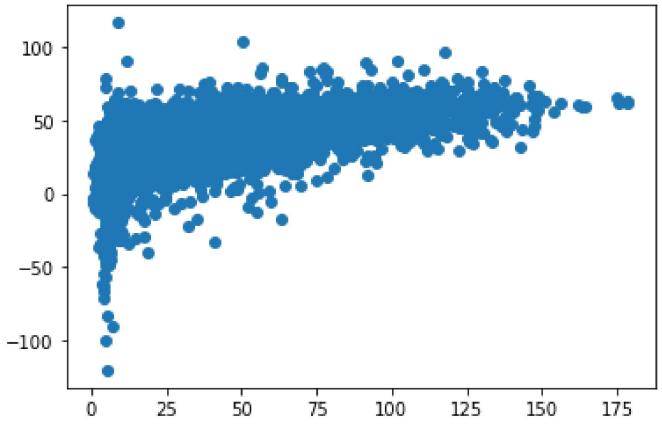
ELASTIC NET

In [77]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[77]:
```
ElasticNet()
```

In [78]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[78]:
```
<matplotlib.collections.PathCollection at 0x2e4ced4c250>
```



In [79]:
```
ens=en.score(x_test,y_test)
```

In [80]:
```
print(rr.score(x_test,y_test))
rr.score(x_train,y_train)
```

```
0.36861387107278976
```

Out[80]:
```
0.36161847914012046
```

LOGISTIC REGRESSION

In [81]:
```
g={"TCH":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["TCH"].value_counts()
```

Out[81]:
```
Low     21614
High    11396
Name: TCH, dtype: int64
```

In [82]:
```
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 [83]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[83]:
```
LogisticRegression()
```

In [84]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[84]:
```
<matplotlib.collections.PathCollection at 0x2e4ce735370>
```

In [85]:
```
los=lo.score(x_test,y_test)
```

RANDOM FOREST

In [86]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [87]:
```
g1={"TCH":{"Low":1.0,"High":2.0}}
df1=df1.replace(g1)
```

In [88]:
```
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 [89]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[89]:
```
RandomForestClassifier()
```

In [90]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [91]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[91]:
```
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 [92]:
```
rfcs=grid_search.best_score_
```

In [93]:
```
rfc_best=grid_search.best_estimator_
```

In [94]:
```
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"],filled=True
```

```
 Text(648.0, 465.9428571428573, 'PXY <= 0.745\ngini = 0.185\nsamples = 1340\nvalue = [1918,
220]\nclass = Yes'),
 Text(612.0, 155.3142857142857, 'gini = 0.09\nsamples = 440\nvalue = [682, 34]\nclass = Ye
s'),
 Text(684.0, 155.3142857142857, 'gini = 0.227\nsamples = 900\nvalue = [1236, 186]\nclass =
Yes'),
 Text(792.0, 465.9428571428573, 'station <= 28079030.0\ngini = 0.044\nsamples = 3647\nvalue
= [5632, 131]\nclass = Yes'),
 Text(756.0, 155.3142857142857, 'gini = 0.025\nsamples = 2037\nvalue = [3187, 41]\nclass =
Yes'),
 Text(828.0, 155.3142857142857, 'gini = 0.068\nsamples = 1610\nvalue = [2445, 90]\nclass =
Yes'),
 Text(1008.0, 776.5714285714287, 'O_3 <= 46.78\ngini = 0.247\nsamples = 1378\nvalue = [183
3, 310]\nclass = Yes'),
 Text(936.0, 465.9428571428573, 'NO_2 <= 22.29\ngini = 0.388\nsamples = 420\nvalue = [471,
168]\nclass = Yes'),
 Text(900.0, 155.3142857142857, 'gini = 0.408\nsamples = 13\nvalue = [6, 15]\nclass = No'),
 Text(972.0, 155.3142857142857, 'gini = 0.373\nsamples = 407\nvalue = [465, 153]\nclass = Y
es'),
 Text(1080.0, 465.9428571428573, 'SO_2 <= 8.845\ngini = 0.171\nsamples = 958\nvalue = [136
```

In [95]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.36862334532732444
Lasso: 0.35040943153584647
Ridge: 0.36861387107278976
ElasticNet: 0.35425816758891715
Logistic: 0.6554579420377663
Random Forest: 0.8834551305352947
```

In [ ]:
```

```