In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge,Lasso
from sklearn.linear_model import ElasticNet
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.tree import plot_tree
```

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_20
df
```

Out[2]:

| | date | BEN | CH4 | CO | EBE | NMHC | NO | NO_2 | NOx | O_3 | PM10 | PM25 | SO_2 | TC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2017-06-01 01:00:00 | NaN | NaN | 0.3 | NaN | NaN | 4.0 | 38.0 | NaN | NaN | NaN | NaN | 5.0 | Na |
| **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. |
| **2** | 2017-06-01 01:00:00 | 0.2 | NaN | NaN | 0.1 | NaN | 1.0 | 14.0 | NaN | NaN | NaN | NaN | NaN | Na |
| **3** | 2017-06-01 01:00:00 | NaN | NaN | 0.2 | NaN | NaN | 1.0 | 9.0 | NaN | 91.0 | NaN | NaN | NaN | Na |
| **4** | 2017-06-01 01:00:00 | NaN | NaN | NaN | NaN | NaN | 1.0 | 19.0 | NaN | 69.0 | NaN | NaN | 2.0 | Na |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **210115** | 2017-08-01 00:00:00 | NaN | NaN | 0.2 | NaN | NaN | 1.0 | 27.0 | NaN | 65.0 | NaN | NaN | NaN | Na |
| **210116** | 2017-08-01 00:00:00 | NaN | NaN | 0.2 | NaN | NaN | 1.0 | 14.0 | NaN | NaN | 73.0 | NaN | 7.0 | Na |
| **210117** | 2017-08-01 00:00:00 | NaN | NaN | NaN | NaN | NaN | 1.0 | 4.0 | NaN | 83.0 | NaN | NaN | NaN | Na |
| **210118** | 2017-08-01 00:00:00 | NaN | NaN | NaN | NaN | NaN | 1.0 | 11.0 | NaN | 78.0 | NaN | NaN | NaN | Na |
| **210119** | 2017-08-01 00:00:00 | NaN | NaN | NaN | NaN | NaN | 1.0 | 14.0 | NaN | 77.0 | 60.0 | NaN | NaN | Na |

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]:
```
df=df.dropna()
df
```

Out[4]:

| | date | BEN | CH4 | CO | EBE | NMHC | NO | NO_2 | NOx | O_3 | PM10 | PM25 | SO_2 | TCH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 |
| **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 |
| **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 |
| **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 |
| **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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **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 |
| **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 |
| **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 |
| **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 |
| **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 |

4127 rows × 16 columns

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
date       0
BEN        0
CH4        0
CO         0
EBE        0
NMHC       0
NO         0
NO_2       0
NOx        0
O_3        0
PM10       0
PM25       0
SO_2       0
TCH        0
TOL        0
station    0
dtype: int64
```

In [6]:
```
df.describe()
```

Out[6]:

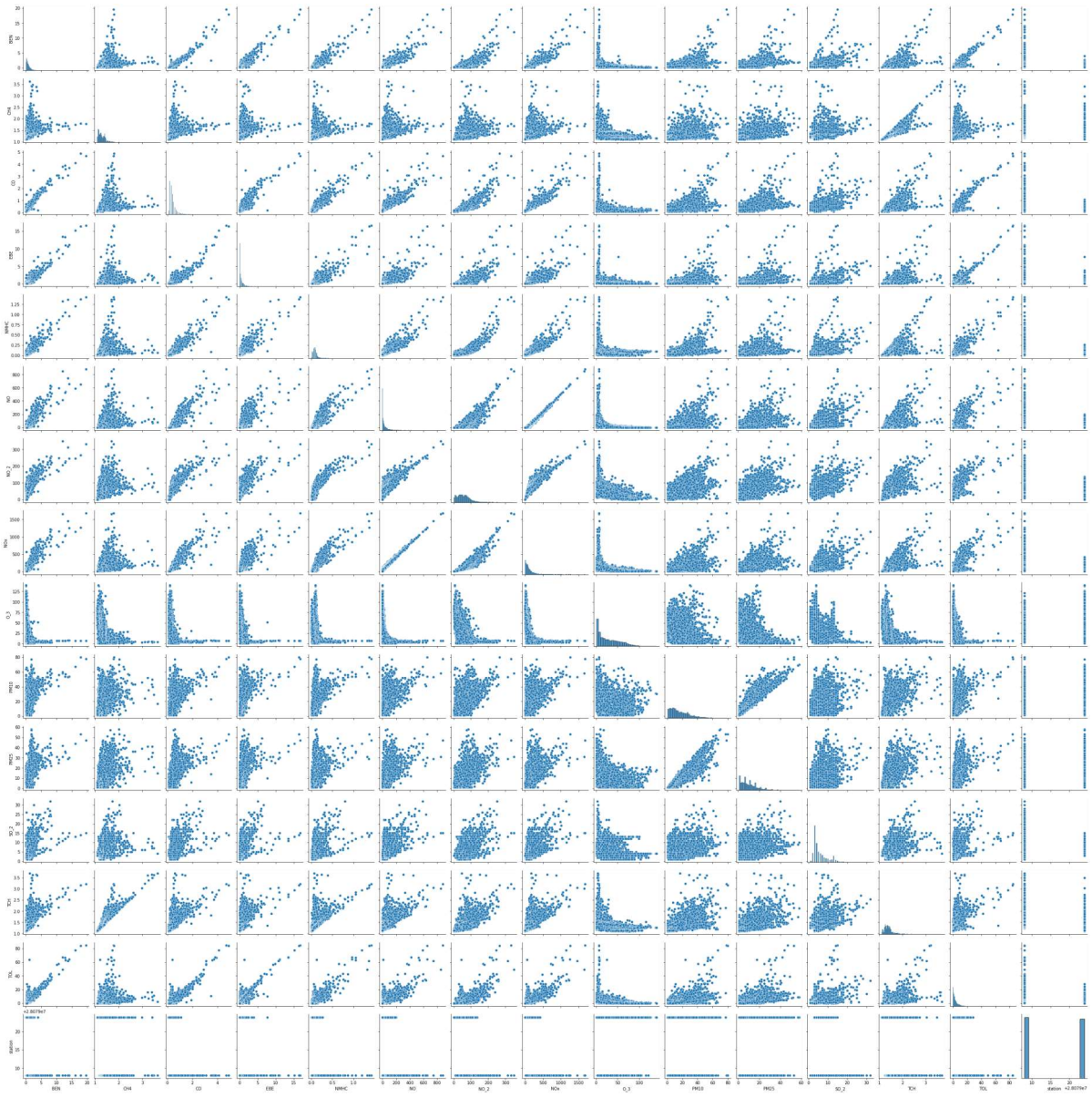
| | BEN | CH4 | CO | EBE | NMHC | NO | NO_ |
|---|---|---|---|---|---|---|---|
| **count** | 4127.000000 | 4127.000000 | 4127.000000 | 4127.000000 | 4127.000000 | 4127.000000 | 4127.00000 |
| **mean** | 0.919918 | 1.323732 | 0.417858 | 0.578168 | 0.097269 | 41.785316 | 58.06905 |
| **std** | 1.123078 | 0.215742 | 0.342871 | 0.962000 | 0.094035 | 71.118499 | 38.97411 |
| **min** | 0.100000 | 1.100000 | 0.100000 | 0.100000 | 0.000000 | 1.000000 | 1.00000 |
| **25%** | 0.300000 | 1.180000 | 0.200000 | 0.100000 | 0.050000 | 3.000000 | 30.00000 |
| **50%** | 0.600000 | 1.270000 | 0.300000 | 0.300000 | 0.080000 | 16.000000 | 54.00000 |
| **75%** | 1.100000 | 1.400000 | 0.500000 | 0.700000 | 0.110000 | 50.000000 | 78.00000 |
| **max** | 19.600000 | 3.630000 | 4.900000 | 16.700001 | 1.420000 | 879.000000 | 349.00000 |

In [7]:
```
df.columns
```

Out[7]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [8]:
```
sns.pairplot(df)
```

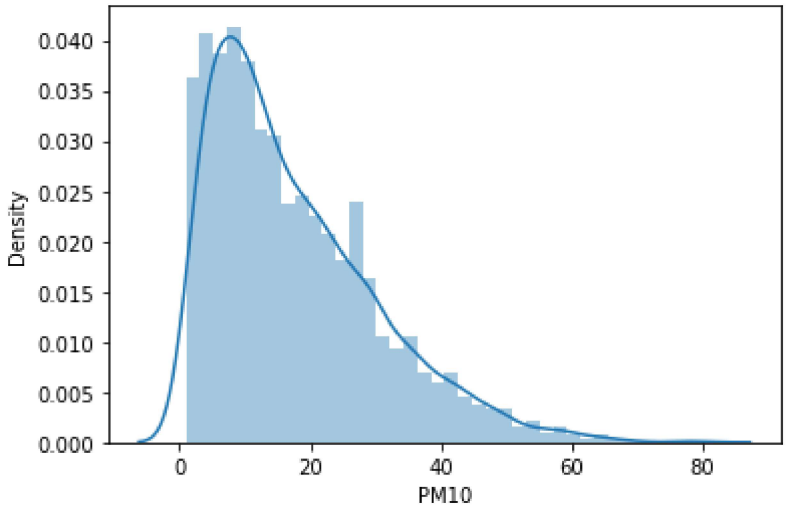
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1311ccfe5e0>
```



In [9]:
```
sns.distplot(df['PM10'])
```

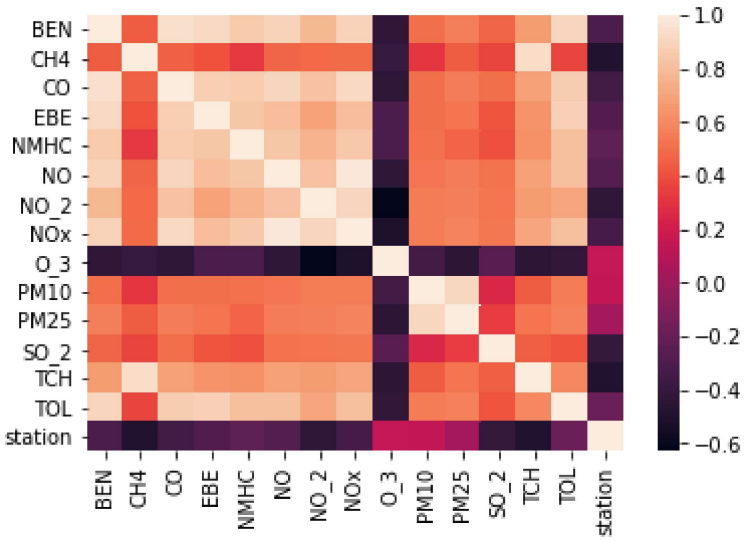
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]: <AxesSubplot:xlabel='PM10', ylabel='Density'>




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

Out[10]: <AxesSubplot:>



In [11]:
```
df.loc[df['TCH']<2,'TCH']=0
df.loc[df['TCH']>2,'TCH']=1
df['TCH']=df['TCH'].astype(int)
df
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:1720: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  self._setitem_single_column(loc, value, pi)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:1720: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  self._setitem_single_column(loc, value, pi)
<ipython-input-11-e3d36a273982>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df['TCH']=df['TCH'].astype(int)
```

Out[11]:

| | date | BEN | CH4 | CO | EBE | NMHC | NO | NO_2 | NOx | O_3 | PM10 | PM25 | SO_2 | TCH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | 0 |
| **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 | 0 |
| **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 | 0 |
| **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 | 0 |
| **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 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **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 | 0 |
| **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 | 0 |
| **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 | 0 |
| **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 | 0 |
| **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 | 0 |

4127 rows × 16 columns

# LogisticRegression

In [12]:
```
x=df[[ 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TOL', 'station']]
y=df['TCH']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
lgr=LogisticRegression()
lgr.fit(x_train,y_train)
```

Out[12]:
```
LogisticRegression()
```

In [13]:
```
lgr.predict(x_test)
```

Out[13]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [14]:
```
lgr.score(x_test,y_test)
```

Out[14]:
```
0.9741727199354318
```

In [15]:
```
fs=StandardScaler().fit_transform(x)
logr=LogisticRegression()
logr.fit(fs,y)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[15]:
```
LogisticRegression()
```

In [16]:
```
o=[[1,2,3,4,5,6,7,8,9,10,11,12]]
prediction=logr.predict(o)
print(prediction)
```

```
[0]
```

In [17]:
```
logr.classes_
```

Out[17]:
```
array([0, 1, 2])
```

In [18]:
```
logr.predict_proba(o)[0][0]
```

Out[18]:
```
0.9998820050961933
```

In [19]:
```
logr.predict_proba(o)[0][1]
```

Out[19]:
```
2.960805477134634e-12
```

# LinearRegression

In [20]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

In [21]:
```
print(lr.intercept_)
```

```
26853.65546002316
```

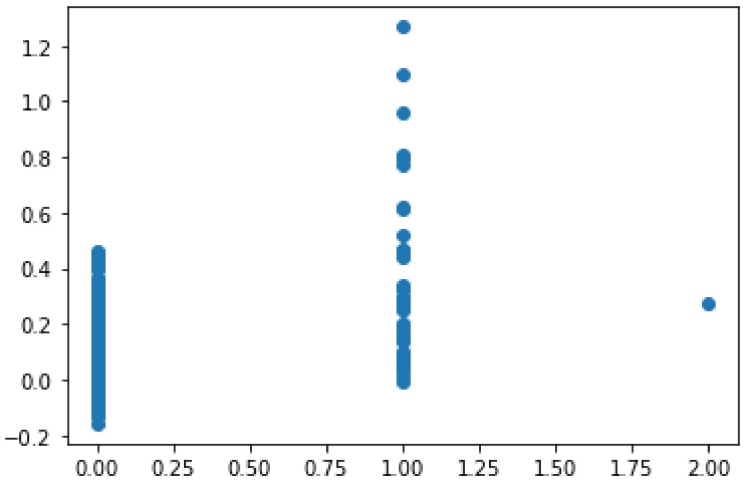
In [22]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[22]:

| | Co-efficient |
|---|---|
| **BEN** | -0.014217 |
| **CO** | -0.072624 |
| **EBE** | 0.067624 |
| **NMHC** | 0.543903 |
| **NO** | 0.001124 |
| **NO_2** | -0.001157 |
| **O_3** | -0.000119 |
| **PM10** | -0.003988 |
| **PM25** | 0.009482 |
| **SO_2** | 0.004584 |
| **TOL** | -0.006439 |
| **station** | -0.000956 |

In [23]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x13131691940>
```




In [24]:
```
print(lr.score(x_test,y_test))
```

```
0.2576700195782905
```

# Ridge,Lasso

In [25]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[25]:
```
Ridge(alpha=10)
```

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

Out[26]:
```
0.25561120867895626
```

In [27]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[27]:
```
Lasso(alpha=10)
```

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

Out[28]:
```
-0.001862074246179457
```

# ElasticNet

In [29]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[29]:
```
ElasticNet()
```

In [30]:
```
print(en.coef_)
```

```
[ 0.         0.         0.         0.         0.0013631 -0.
  0.         0.         0.         0.         0.        -0.       ]
```

In [31]:
```
print(en.intercept_)
```

```
-0.01564627430519628
```

In [32]:
```
print(en.predict(x_train))
```

```
[-0.00883077 -0.00610457  0.15201504 ...  0.00752643  0.02252053
  0.08386003]
```

In [33]:
```
print(en.score(x_train,y_train))
```

```
0.2680061856692486
```

In [34]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 0.07080928821097447
```

In [35]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 0.02550899318503624
```

In [36]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 0.266100146957822
```

# RandomForest

In [37]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[37]:
```
RandomForestClassifier()
```

In [38]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [39]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:
666: UserWarning: The least populated class in y has only 1 members, which is
less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[39]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

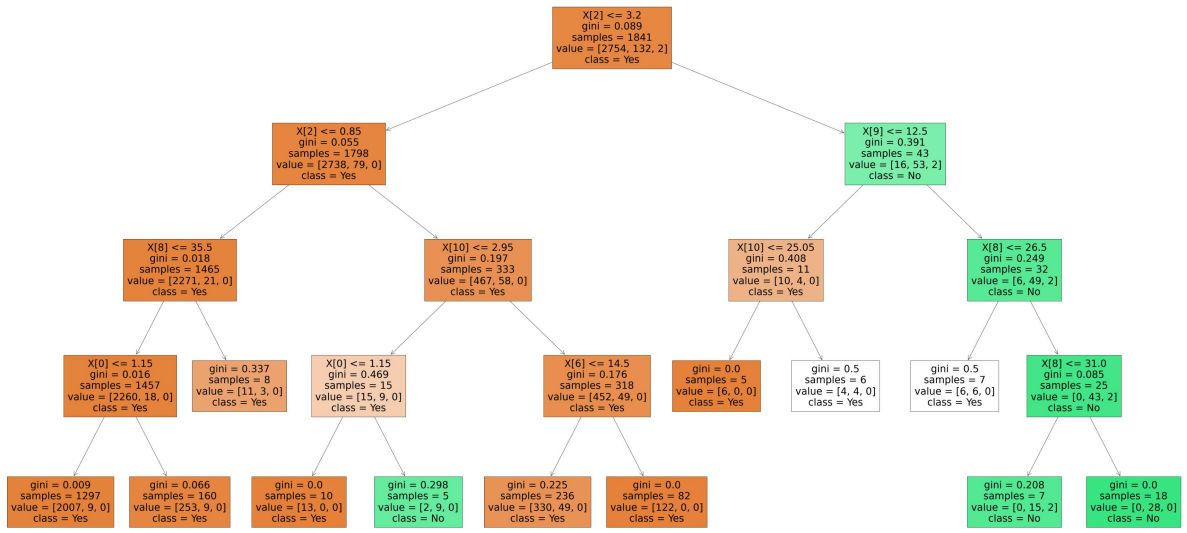
In [40]:
```
grid_search.best_score_
```

Out[40]:
```
0.9695290858725762
```

In [41]:
```
rfc_best=grid_search.best_estimator_
```

In [42]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],class_names=['Yes','No','Yes','No'],filled=T
```

Out[42]:
```
[Text(2287.7999999999997, 1956.96, 'X[2] <= 3.2\ngini = 0.089\nsamples = 1841
\nvalue = [2754, 132, 2]\nclass = Yes'),
 Text(1227.6, 1522.0800000000002, 'X[2] <= 0.85\ngini = 0.055\nsamples = 1798
\nvalue = [2738, 79, 0]\nclass = Yes'),
 Text(669.5999999999999, 1087.2, 'X[8] <= 35.5\ngini = 0.018\nsamples = 1465
\nvalue = [2271, 21, 0]\nclass = Yes'),
 Text(446.4, 652.3200000000002, 'X[0] <= 1.15\ngini = 0.016\nsamples = 1457\n
value = [2260, 18, 0]\nclass = Yes'),
 Text(223.2, 217.44000000000005, 'gini = 0.009\nsamples = 1297\nvalue = [200
7, 9, 0]\nclass = Yes'),
 Text(669.5999999999999, 217.44000000000005, 'gini = 0.066\nsamples = 160\nva
lue = [253, 9, 0]\nclass = Yes'),
 Text(892.8, 652.3200000000002, 'gini = 0.337\nsamples = 8\nvalue = [11, 3,
0]\nclass = Yes'),
 Text(1785.6, 1087.2, 'X[10] <= 2.95\ngini = 0.197\nsamples = 333\nvalue = [4
67, 58, 0]\nclass = Yes'),
 Text(1339.1999999999998, 652.3200000000002, 'X[0] <= 1.15\ngini = 0.469\nsam
ples = 15\nvalue = [15, 9, 0]\nclass = Yes'),
 Text(1116.0, 217.44000000000005, 'gini = 0.0\nsamples = 10\nvalue = [13, 0,
0]\nclass = Yes'),
 Text(1562.3999999999999, 217.44000000000005, 'gini = 0.298\nsamples = 5\nval
ue = [2, 9, 0]\nclass = No'),
 Text(2232.0, 652.3200000000002, 'X[6] <= 14.5\ngini = 0.176\nsamples = 318\n
value = [452, 49, 0]\nclass = Yes'),
 Text(2008.8, 217.44000000000005, 'gini = 0.225\nsamples = 236\nvalue = [330,
49, 0]\nclass = Yes'),
 Text(2455.2, 217.44000000000005, 'gini = 0.0\nsamples = 82\nvalue = [122, 0,
0]\nclass = Yes'),
 Text(3348.0, 1522.0800000000002, 'X[9] <= 12.5\ngini = 0.391\nsamples = 43\n
value = [16, 53, 2]\nclass = No'),
 Text(2901.6, 1087.2, 'X[10] <= 25.05\ngini = 0.408\nsamples = 11\nvalue = [1
0, 4, 0]\nclass = Yes'),
 Text(2678.3999999999996, 652.3200000000002, 'gini = 0.0\nsamples = 5\nvalue
= [6, 0, 0]\nclass = Yes'),
 Text(3124.7999999999997, 652.3200000000002, 'gini = 0.5\nsamples = 6\nvalue
= [4, 4, 0]\nclass = Yes'),
 Text(3794.3999999999996, 1087.2, 'X[8] <= 26.5\ngini = 0.249\nsamples = 32\n
value = [6, 49, 2]\nclass = No'),
 Text(3571.2, 652.3200000000002, 'gini = 0.5\nsamples = 7\nvalue = [6, 6, 0]
\nclass = Yes'),
 Text(4017.6, 652.3200000000002, 'X[8] <= 31.0\ngini = 0.085\nsamples = 25\nv
alue = [0, 43, 2]\nclass = No'),
 Text(3794.3999999999996, 217.44000000000005, 'gini = 0.208\nsamples = 7\nval
ue = [0, 15, 2]\nclass = No'),
 Text(4240.8, 217.44000000000005, 'gini = 0.0\nsamples = 18\nvalue = [0, 28,
0]\nclass = No')]
```




Best model:LinearRegression

In [ ]: