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	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.41(
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.67(
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.25(
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.15(

209448 rows × 17 columns

In [3]:
```
df.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 [4]:
```
df=df.dropna()
df
```

Out[4]:

	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	18.3700(
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	23.6399(
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	5.6000(
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	7.8900(
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	5.1300(
...	...	...	...	...	...	...	...	...	...	...	
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	20.3400(
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	14.3800(
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	22.8400(
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	10.7700(
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	15.3500(

6666 rows × 17 columns

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

Out[5]:
```
date       0
BEN        0
CO         0
EBE        0
MXY        0
NMHC       0
NO_2       0
NOx        0
OXY        0
O_3        0
PM10       0
PM25       0
PXY        0
SO_2       0
TCH        0
TOL        0
station    0
dtype: int64
```

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

Out[6]:

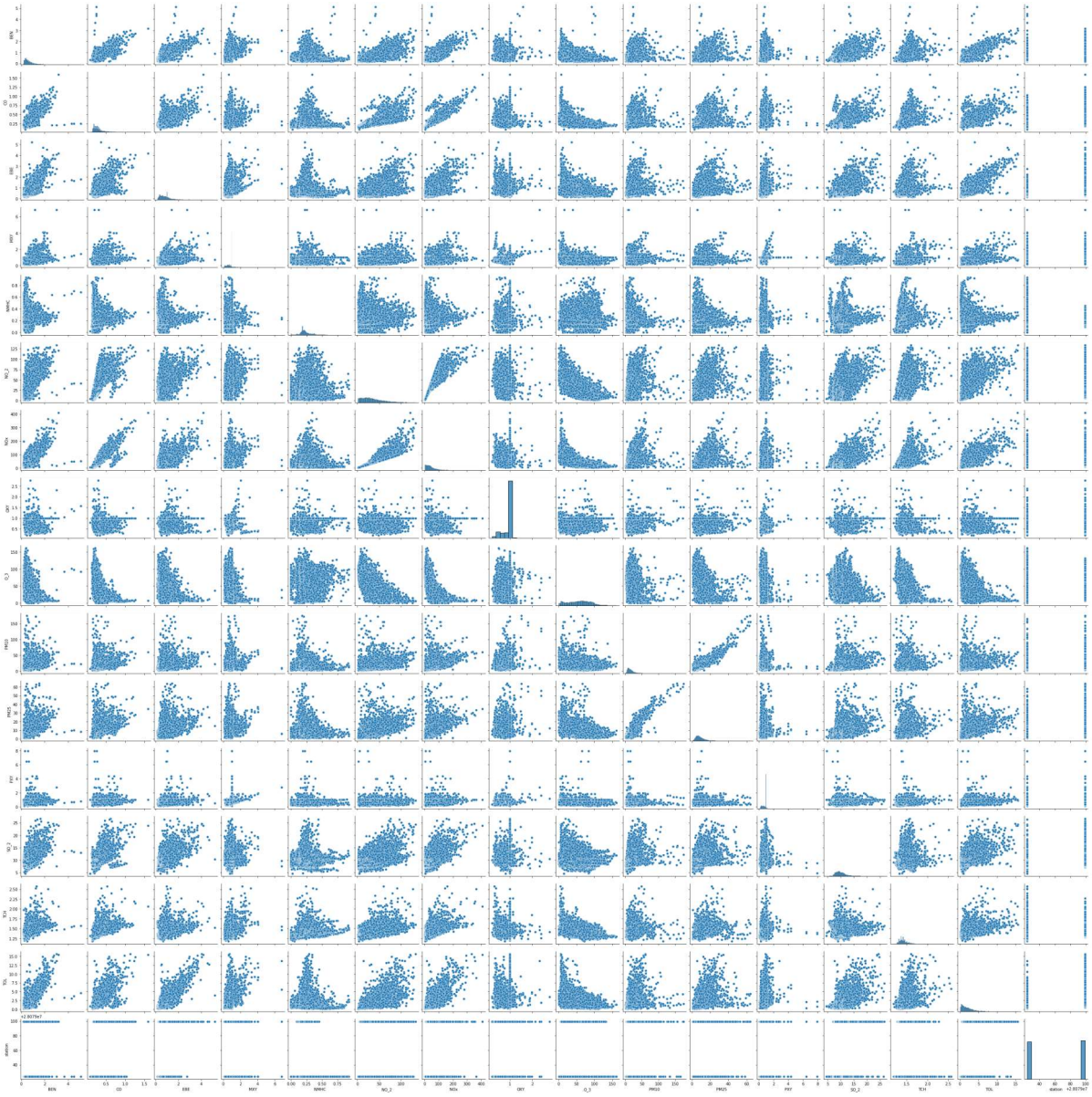
	BEN	CO	EBE	MXY	NMHC	NO_2	NC
count	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.00000
mean	0.648425	0.296280	0.840585	0.839959	0.243378	33.888744	47.5406
std	0.395346	0.133296	0.508031	0.382263	0.115730	23.465169	41.2305
min	0.170000	0.090000	0.140000	0.110000	0.000000	1.290000	2.7600
25%	0.380000	0.200000	0.470000	0.590000	0.180000	15.752500	19.4425
50%	0.540000	0.260000	0.755000	1.000000	0.220000	29.320000	36.7700
75%	0.810000	0.340000	1.000000	1.000000	0.280000	47.657500	62.1025
max	5.110000	1.590000	5.190000	6.810000	0.930000	133.399994	409.2999

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

Out[7]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

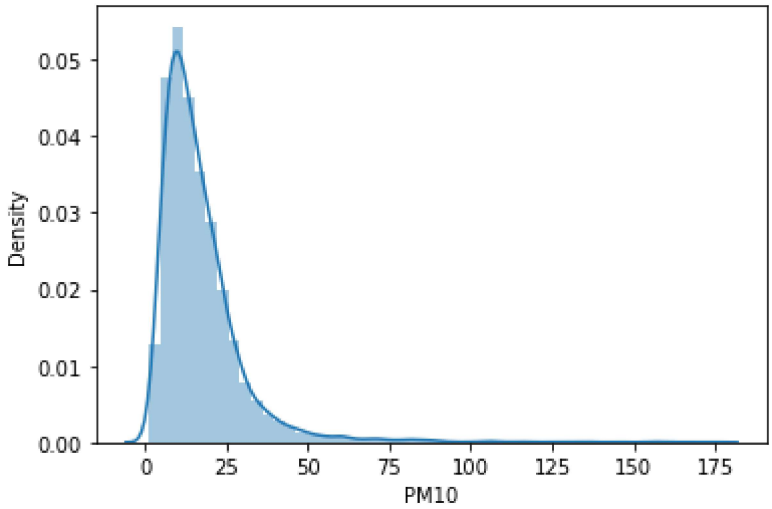
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x15458218dc0>
```



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

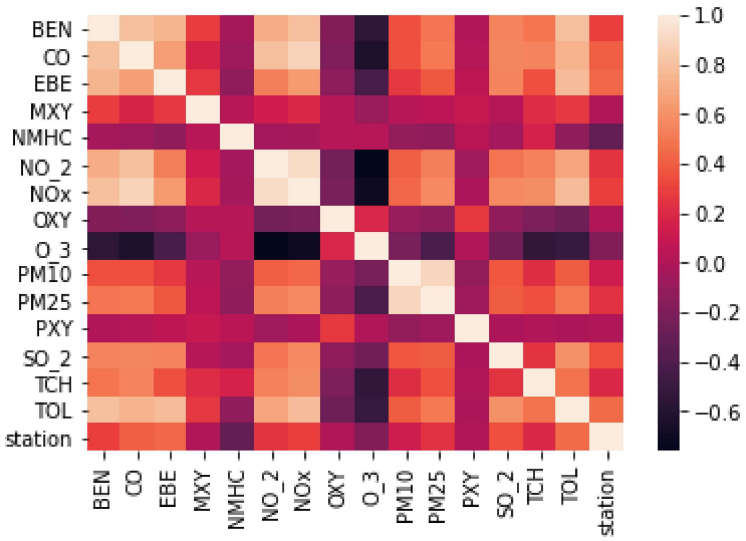
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[10]: <AxesSubplot:>



In [15]:
```
df.loc[df['OXY']<1,'OXY']=0
df.loc[df['OXY']>1,'OXY']=1
df['OXY']=df['OXY'].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-15-930207530bb0>: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['OXY']=df['OXY'].astype(int)
```

Out[15]:

	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	10.420000	11.900000	1	90.309998	18.3700
23	2010-03-01 01:00:00	0.70	0.23	1.00	0.73	0	17.820000	22.290001	1	70.550003	23.6399
35	2010-03-01 02:00:00	0.58	0.17	0.84	0.73	0	3.500000	4.950000	1	68.849998	5.6000
47	2010-03-01 02:00:00	0.33	0.21	0.84	0.73	0	10.810000	14.900000	1	74.750000	7.8900
59	2010-03-01 03:00:00	0.38	0.16	0.64	1.00	0	2.750000	4.200000	1	93.629997	5.1300
...	...	...	...	...	...	...	...	...	...	...	
191879	2010-05-31 22:00:00	0.60	0.26	0.82	0.13	0	33.360001	43.779999	1	38.459999	20.3400
191891	2010-05-31 23:00:00	0.41	0.16	0.71	0.19	0	24.299999	26.059999	1	50.290001	14.3800
191903	2010-05-31 23:00:00	0.57	0.28	0.64	0.19	0	35.540001	44.590000	1	34.020000	22.8400
191915	2010-06-01 00:00:00	0.34	0.16	0.69	0.22	0	23.559999	25.209999	1	45.930000	10.7700
191927	2010-06-01 00:00:00	0.43	0.25	0.79	0.22	0	34.910000	42.369999	1	29.540001	15.3500

6666 rows × 17 columns

LogisticRegression

In [16]:
```
x=df[[ 'BEN', 'CO', 'EBE', 'MXY','NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
y=df['OXY']
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[16]:
```
LogisticRegression()
```

In [17]:
```
lgr.predict(x_test)
```

Out[17]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

In [18]:
```
lgr.score(x_test,y_test)
```

Out[18]:
```
0.771
```

In [19]:
```
fs=StandardScaler().fit_transform(x)
logr=LogisticRegression()
logr.fit(fs,y)
```

Out[19]:
```
LogisticRegression()
```

In [21]:
```
o=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]]
prediction=logr.predict(o)
print(prediction)
```

```
[1]
```

In [22]:
```
logr.classes_
```

Out[22]:
```
array([0, 1])
```

In [23]:
```
logr.predict_proba(o)[0][0]
```

Out[23]:
```
0.0
```

In [24]:
```
logr.predict_proba(o)[0][1]
```

Out[24]:
```
1.0
```

LinearRegression

In [25]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
print(lr.intercept_)
```

```
-4.8707593514052405e-11
```

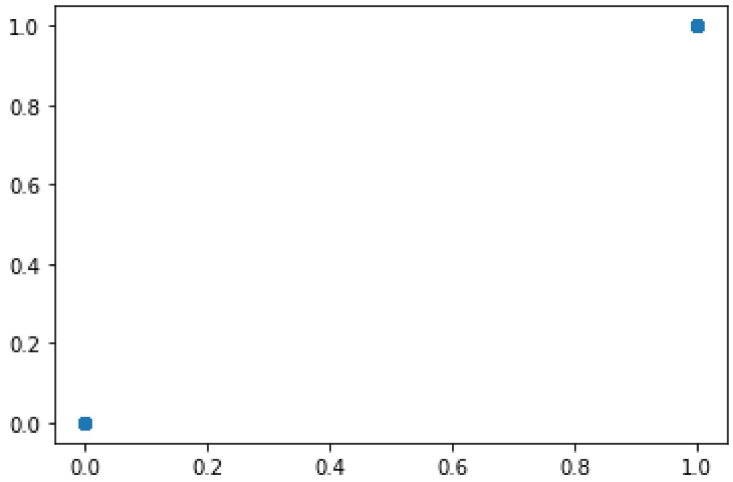
In [27]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[27]:

	Co-efficient
BEN	-4.795408e-16
CO	1.838807e-16
EBE	4.224763e-16
MXY	9.523649e-16
NMHC	-3.365364e-16
NO_2	-1.084202e-16
NOx	4.206704e-17
OXY	1.000000e+00
O_3	-3.144186e-17
PM10	1.173649e-16
PM25	-1.682953e-16
PXY	1.431435e-16
SO_2	-4.838252e-18
TCH	1.936677e-16
TOL	-1.372193e-16
station	1.734723e-18

In [28]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x1546a429d00>
```



In [29]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

Ridge,Lasso

In [30]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[30]:
```
Ridge(alpha=10)
```

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

Out[31]:
```
0.9998374825452105
```

In [32]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[32]:
```
Lasso(alpha=10)
```

In [33]:
```
la.score(x_test,y_test)
```

Out[33]:
```
-0.002273598775057506
```

ElasticNet

In [34]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[34]:
```
ElasticNet()
```

In [35]:
```
print(en.coef_)
```

```
[-0.         -0.         -0.          0.          0.         -0.00349874
 -0.          0.          0.00022191 -0.00253629 -0.          0.
 -0.         -0.         -0.          0.00029038]
```

In [36]:
```
print(en.intercept_)
```

```
-8152.612655176221
```

In [37]:
```
print(en.predict(x_train))
```

```
[0.7829464  0.75638005 0.83223565 ... 0.84603644 0.85150734 0.86588873]
```

In [38]:
```
print(en.score(x_train,y_train))
```

```
0.09195696972720324
```

In [39]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 8.45146130199697e-16
```

In [40]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 1.414160841089682e-30
```

In [41]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 2.9071397114684685e-08
```

RandomForest

In [42]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[42]:
```
RandomForestClassifier()
```

In [43]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [44]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[44]:
```
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')
```

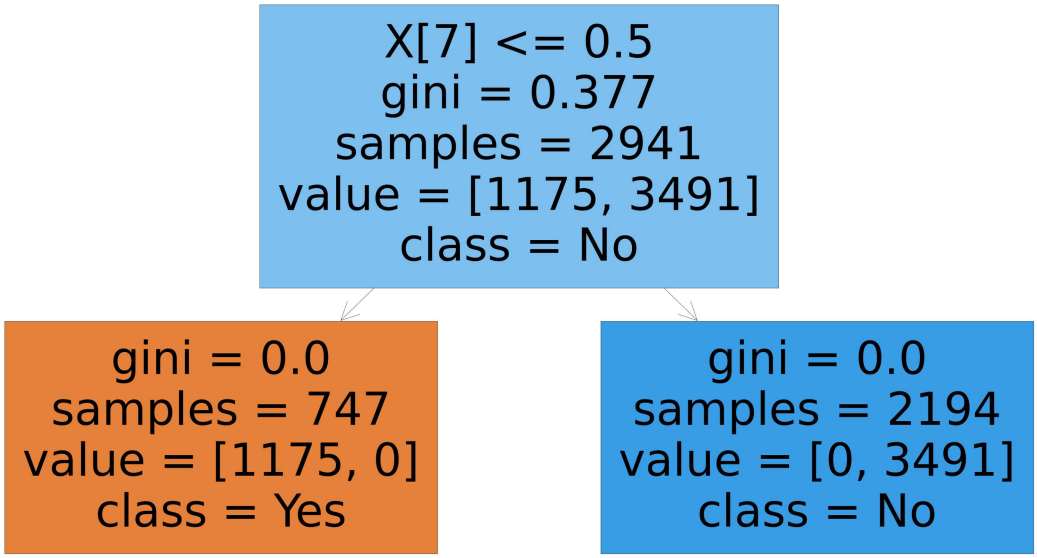
In [45]:
```
grid_search.best_score_
```

Out[45]:
```
1.0
```

In [46]:
```
rfc_best=grid_search.best_estimator_
```

In [47]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],class_names=['Yes','No','Yes','No'],filled=T
```

Out[47]:
```
[Text(2232.0, 1630.8000000000002, 'X[7] <= 0.5\ngini = 0.377\nsamples = 2941
\nvalue = [1175, 3491]\nclass = No'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 747\nvalue = [1175,
0]\nclass = Yes'),
 Text(3348.0, 543.5999999999999, 'gini = 0.0\nsamples = 2194\nvalue = [0, 349
1]\nclass = No')]
```



In [*]:
```
Best model:RandomForest
```

In [ ]:
```

```