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	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	28
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4	28
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	28
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	28
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	28
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	28
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	28
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	28
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	28

210720 rows × 14 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210720 entries, 0 to 210719
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210720 non-null  object 
 1   BEN      51511 non-null   float64
 2   CO       87097 non-null   float64
 3   EBE      51482 non-null   float64
 4   NMHC     30736 non-null   float64
 5   NO       209871 non-null  float64
 6   NO_2     209872 non-null  float64
 7   O_3      122339 non-null  float64
 8   PM10     104838 non-null  float64
 9   PM25     52164 non-null   float64
 10  SO_2     87333 non-null   float64
 11  TCH      30736 non-null   float64
 12  TOL      51373 non-null   float64
 13  station  210720 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.5+ MB
```

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

Out[4]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
6	2012-09-01 01:00:00	0.4	0.2	0.8	0.24	1.0	7.0	57.0	11.0	7.0	2.0	1.33	0.6	280
30	2012-09-01 02:00:00	0.4	0.2	0.7	0.24	1.0	5.0	55.0	5.0	5.0	2.0	1.33	0.5	280
54	2012-09-01 03:00:00	0.4	0.2	0.7	0.24	1.0	4.0	56.0	6.0	4.0	2.0	1.33	0.5	280
78	2012-09-01 04:00:00	0.3	0.2	0.7	0.25	1.0	5.0	54.0	6.0	5.0	2.0	1.34	0.4	280
102	2012-09-01 05:00:00	0.4	0.2	0.7	0.24	1.0	3.0	53.0	8.0	5.0	2.0	1.33	0.5	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210654	2012-02-29 22:00:00	0.6	0.3	0.5	0.09	1.0	35.0	57.0	25.0	21.0	3.0	1.12	2.3	280
210673	2012-02-29 23:00:00	2.0	0.4	2.4	0.21	16.0	79.0	20.0	37.0	25.0	12.0	1.33	6.2	280
210678	2012-02-29 23:00:00	0.7	0.3	0.6	0.09	1.0	27.0	63.0	22.0	18.0	3.0	1.11	1.9	280
210697	2012-03-01 00:00:00	1.5	0.4	1.7	0.21	16.0	79.0	17.0	28.0	21.0	11.0	1.34	4.9	280
210702	2012-03-01 00:00:00	0.6	0.3	0.5	0.09	1.0	23.0	61.0	18.0	16.0	3.0	1.11	1.2	280

10916 rows × 14 columns

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

Out[5]:
```
date       0
BEN        0
CO         0
EBE        0
NMHC       0
NO         0
NO_2       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]:

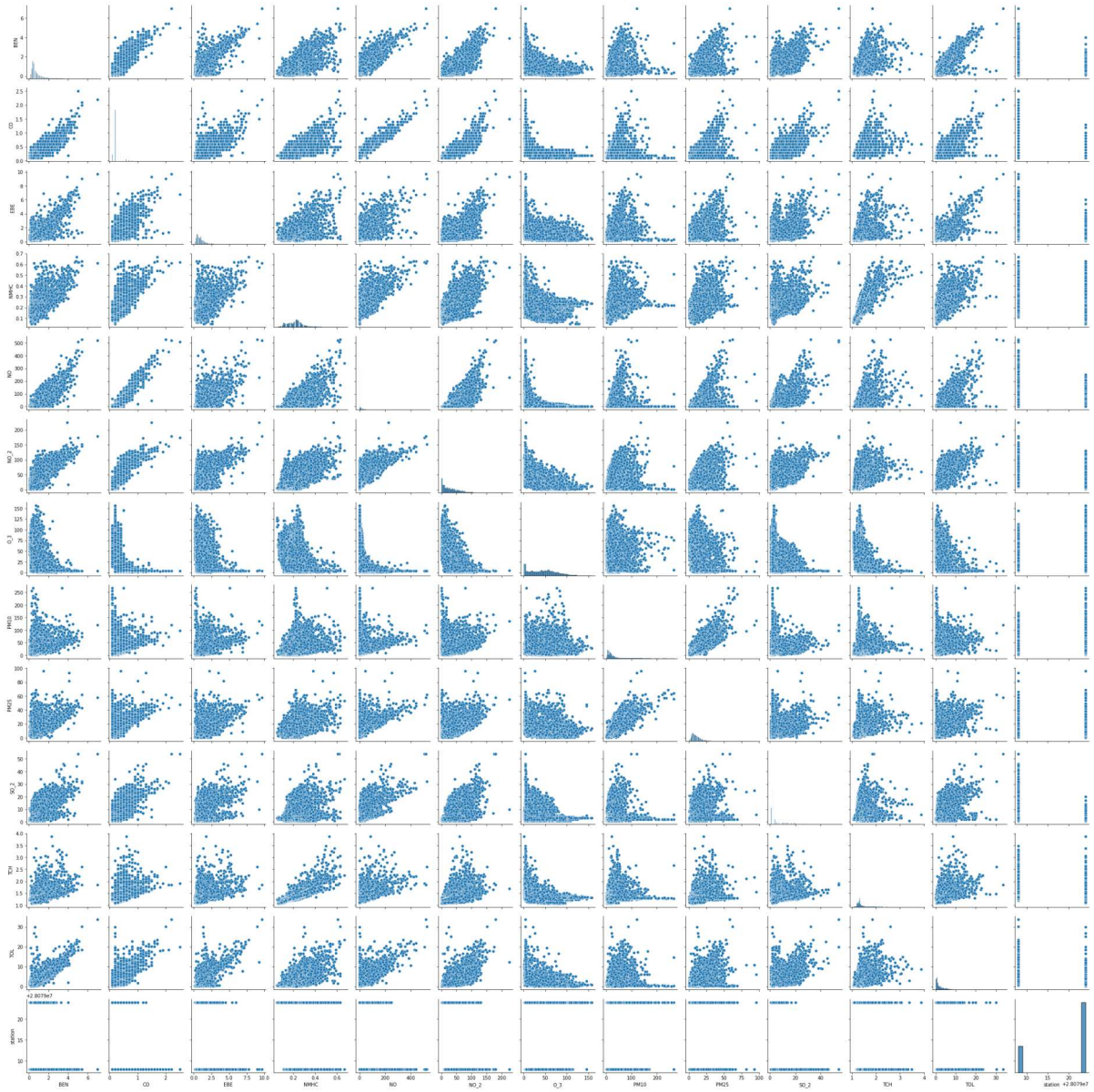
	BEN	CO	EBE	NMHC	NO	NO_2	
count	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	109
mean	0.784014	0.279333	0.992213	0.215755	18.795529	31.262642	
std	0.632755	0.167922	0.804554	0.075169	40.038872	27.234732	
min	0.100000	0.100000	0.100000	0.050000	0.000000	1.000000	
25%	0.400000	0.200000	0.500000	0.160000	1.000000	9.000000	
50%	0.600000	0.200000	0.800000	0.220000	3.000000	24.000000	
75%	0.900000	0.300000	1.200000	0.250000	18.000000	47.000000	
max	7.000000	2.500000	9.700000	0.670000	525.000000	225.000000	1

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

Out[7]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

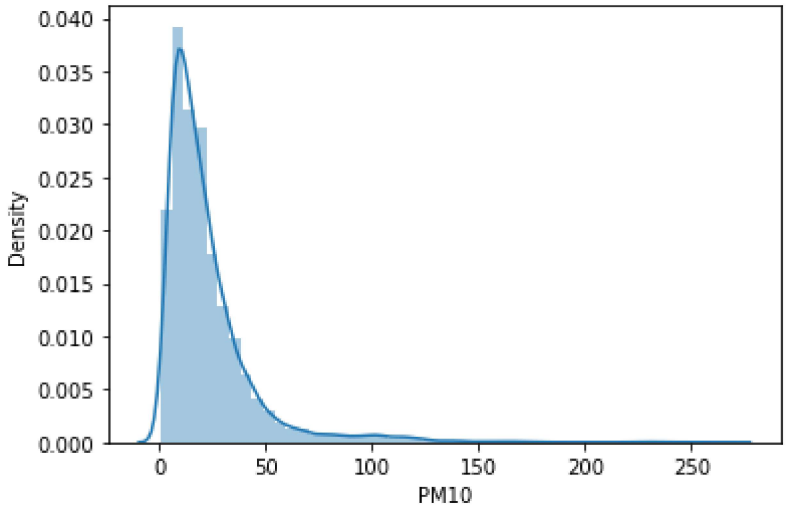
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x18b87c084f0>
```



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

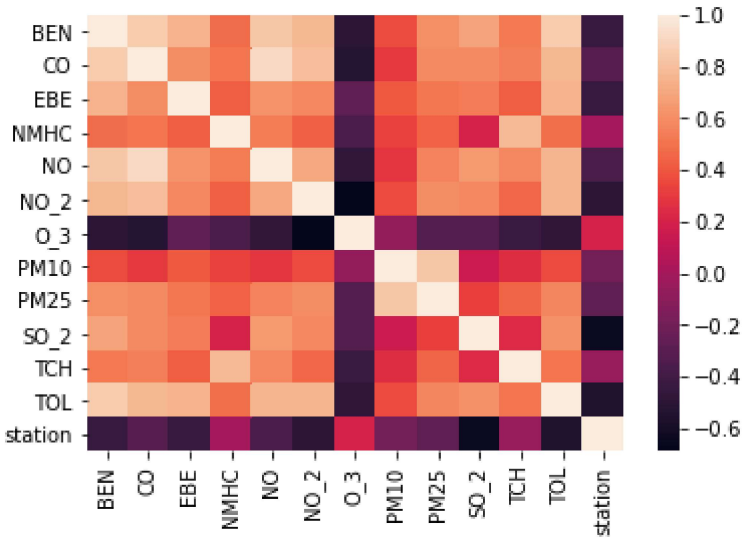
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[10]: <AxesSubplot:>



In [12]:
```
df.loc[df['CO']<1,'CO']=0
df.loc[df['CO']>1,'CO']=1
df['CO']=df['CO'].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-12-dc2e98cdf216>: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['CO']=df['CO'].astype(int)
```

Out[12]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
6	2012-09-01 01:00:00	0.4	0	0.8	0.24	1.0	7.0	57.0	11.0	7.0	2.0	1.33	0.6	280
30	2012-09-01 02:00:00	0.4	0	0.7	0.24	1.0	5.0	55.0	5.0	5.0	2.0	1.33	0.5	280
54	2012-09-01 03:00:00	0.4	0	0.7	0.24	1.0	4.0	56.0	6.0	4.0	2.0	1.33	0.5	280
78	2012-09-01 04:00:00	0.3	0	0.7	0.25	1.0	5.0	54.0	6.0	5.0	2.0	1.34	0.4	280
102	2012-09-01 05:00:00	0.4	0	0.7	0.24	1.0	3.0	53.0	8.0	5.0	2.0	1.33	0.5	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210654	2012-02-29 22:00:00	0.6	0	0.5	0.09	1.0	35.0	57.0	25.0	21.0	3.0	1.12	2.3	280
210673	2012-02-29 23:00:00	2.0	0	2.4	0.21	16.0	79.0	20.0	37.0	25.0	12.0	1.33	6.2	280
210678	2012-02-29 23:00:00	0.7	0	0.6	0.09	1.0	27.0	63.0	22.0	18.0	3.0	1.11	1.9	280
210697	2012-03-01 00:00:00	1.5	0	1.7	0.21	16.0	79.0	17.0	28.0	21.0	11.0	1.34	4.9	280
210702	2012-03-01 00:00:00	0.6	0	0.5	0.09	1.0	23.0	61.0	18.0	16.0	3.0	1.11	1.2	280

10916 rows × 14 columns

LogisticRegression

In [13]:
```
x=df[[ 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
y=df['CO']
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[13]:
```
LogisticRegression()
```

In [14]:
```
lgr.predict(x_test)
```

Out[14]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [15]:
```
lgr.score(x_test,y_test)
```

Out[15]:
```
0.9905343511450382
```

In [16]:
```
fs=StandardScaler().fit_transform(x)
logr=LogisticRegression()
logr.fit(fs,y)
```

Out[16]:
```
LogisticRegression()
```

In [19]:
```
o=[[1,2,3,4,5,6,7,8,9,10,11,12,13]]
prediction=logr.predict(o)
print(prediction)
```

```
[0]
```

In [20]:
```
logr.classes_
```

Out[20]:
```
array([0, 1])
```

In [21]:
```
logr.predict_proba(o)[0][0]
```

Out[21]:
```
0.8372786799394607
```

In [22]:
```
logr.predict_proba(o)[0][1]
```

Out[22]:
```
0.1627213200605393
```

LinearRegression

In [23]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[23]:
```
LinearRegression()
```

In [24]:
```
print(lr.intercept_)
```

```
-4.6761178401633074e-09
```

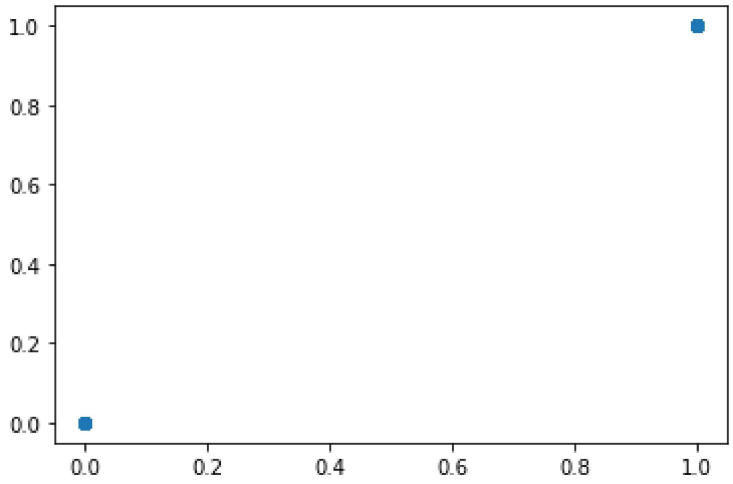
In [25]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[25]:

	Co-efficient
BEN	3.203140e-14
CO	1.000000e+00
EBE	-2.695760e-15
NMHC	2.361685e-14
NO	-1.110223e-16
NO_2	-1.665335e-16
O_3	5.551115e-17
PM10	2.775558e-17
PM25	-8.326673e-17
SO_2	-8.326673e-17
TCH	3.642919e-16
TOL	-3.816392e-16
station	1.665335e-16

In [26]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[26]:
```
<matplotlib.collections.PathCollection at 0x18b931cef70>
```



In [27]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

Ridge,Lasso

In [28]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[28]:
```
Ridge(alpha=10)
```

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

Out[29]:
```
0.980929845032987
```

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

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

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

Out[31]:
```
-0.0003415119011236367
```

ElasticNet

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

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

In [33]:
```
print(en.coef_)
```

```
[ 0.          0.          0.          0.          0.00133254 -0.
  0.          0.          0.          0.          0.          0.
  0.        ]
```

In [34]:
```
print(en.intercept_)
```

```
-0.014341719307344859
```

In [35]:
```
print(en.predict(x_train))
```

```
[ 0.00697894 -0.01300918 -0.00501393 ... -0.01300918  0.01763927
  0.07493855]
```

In [36]:
```
print(en.score(x_train,y_train))
```

```
0.3857680979696393
```

In [37]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 7.630040014222296e-15
```

In [38]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 1.2524196977447458e-28
```

In [39]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 8.735010025307525e-08
```

RandomForest

In [40]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[40]:
```
RandomForestClassifier()
```

In [41]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [42]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[42]:
```
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')
```

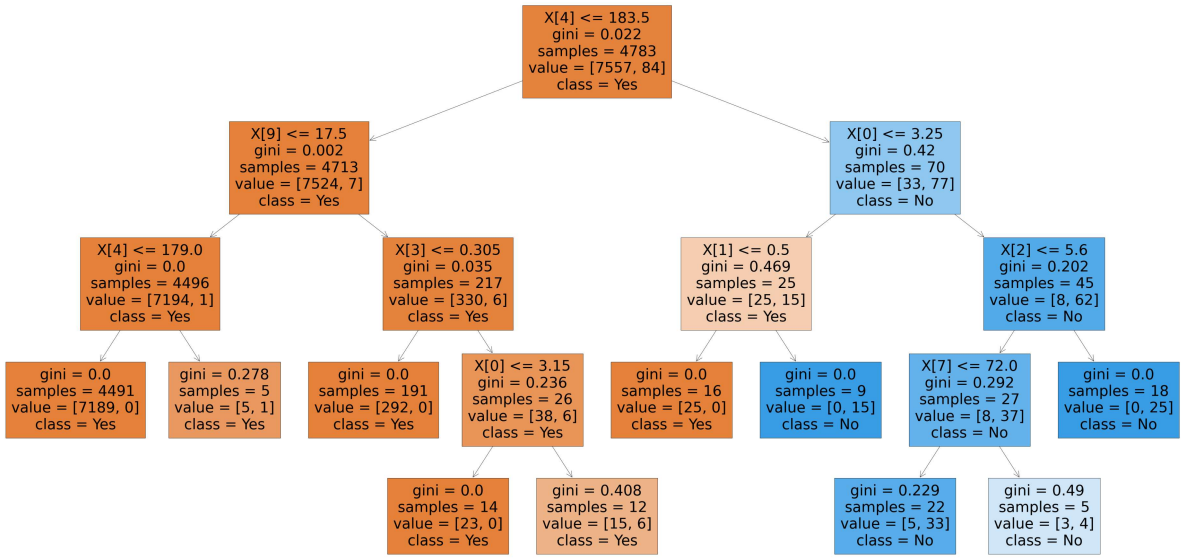
In [43]:
```
grid_search.best_score_
```

Out[43]:
```
1.0
```

In [44]:
```
rfc_best=grid_search.best_estimator_
```

In [45]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],class_names=['Yes','No','Yes','No'],filled=T
```

Out[45]:
```
[Text(2232.0, 1956.96, 'X[4] <= 183.5\ngini = 0.022\nsamples = 4783\nvalue =
[7557, 84]\nclass = Yes'),
 Text(1116.0, 1522.0800000000002, 'X[9] <= 17.5\ngini = 0.002\nsamples = 4713
\nvalue = [7524, 7]\nclass = Yes'),
 Text(558.0, 1087.2, 'X[4] <= 179.0\ngini = 0.0\nsamples = 4496\nvalue = [719
4, 1]\nclass = Yes'),
 Text(279.0, 652.3200000000002, 'gini = 0.0\nsamples = 4491\nvalue = [7189,
0]\nclass = Yes'),
 Text(837.0, 652.3200000000002, 'gini = 0.278\nsamples = 5\nvalue = [5, 1]\nc
lass = Yes'),
 Text(1674.0, 1087.2, 'X[3] <= 0.305\ngini = 0.035\nsamples = 217\nvalue = [3
30, 6]\nclass = Yes'),
 Text(1395.0, 652.3200000000002, 'gini = 0.0\nsamples = 191\nvalue = [292, 0]
\nclass = Yes'),
 Text(1953.0, 652.3200000000002, 'X[0] <= 3.15\ngini = 0.236\nsamples = 26\nv
alue = [38, 6]\nclass = Yes'),
 Text(1674.0, 217.44000000000005, 'gini = 0.0\nsamples = 14\nvalue = [23, 0]
\nclass = Yes'),
 Text(2232.0, 217.44000000000005, 'gini = 0.408\nsamples = 12\nvalue = [15,
6]\nclass = Yes'),
 Text(3348.0, 1522.0800000000002, 'X[0] <= 3.25\ngini = 0.42\nsamples = 70\nv
alue = [33, 77]\nclass = No'),
 Text(2790.0, 1087.2, 'X[1] <= 0.5\ngini = 0.469\nsamples = 25\nvalue = [25,
15]\nclass = Yes'),
 Text(2511.0, 652.3200000000002, 'gini = 0.0\nsamples = 16\nvalue = [25, 0]\n
class = Yes'),
 Text(3069.0, 652.3200000000002, 'gini = 0.0\nsamples = 9\nvalue = [0, 15]\nc
lass = No'),
 Text(3906.0, 1087.2, 'X[2] <= 5.6\ngini = 0.202\nsamples = 45\nvalue = [8, 6
2]\nclass = No'),
 Text(3627.0, 652.3200000000002, 'X[7] <= 72.0\ngini = 0.292\nsamples = 27\nv
alue = [8, 37]\nclass = No'),
 Text(3348.0, 217.44000000000005, 'gini = 0.229\nsamples = 22\nvalue = [5, 3
3]\nclass = No'),
 Text(3906.0, 217.44000000000005, 'gini = 0.49\nsamples = 5\nvalue = [3, 4]\n
class = No'),
 Text(4185.0, 652.3200000000002, 'gini = 0.0\nsamples = 18\nvalue = [0, 25]\n
class = No')]
```



In [ ]:
```
best model:RandomForest
```