In [13]:
```
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	PM10
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002
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	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002
...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000
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
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
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
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

217296 rows × 16 columns

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

32381 rows × 16 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
PXY        0
SO_2       0
TCH        0
TOL        0
station    0
dtype: int64
```

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

Out[6]:

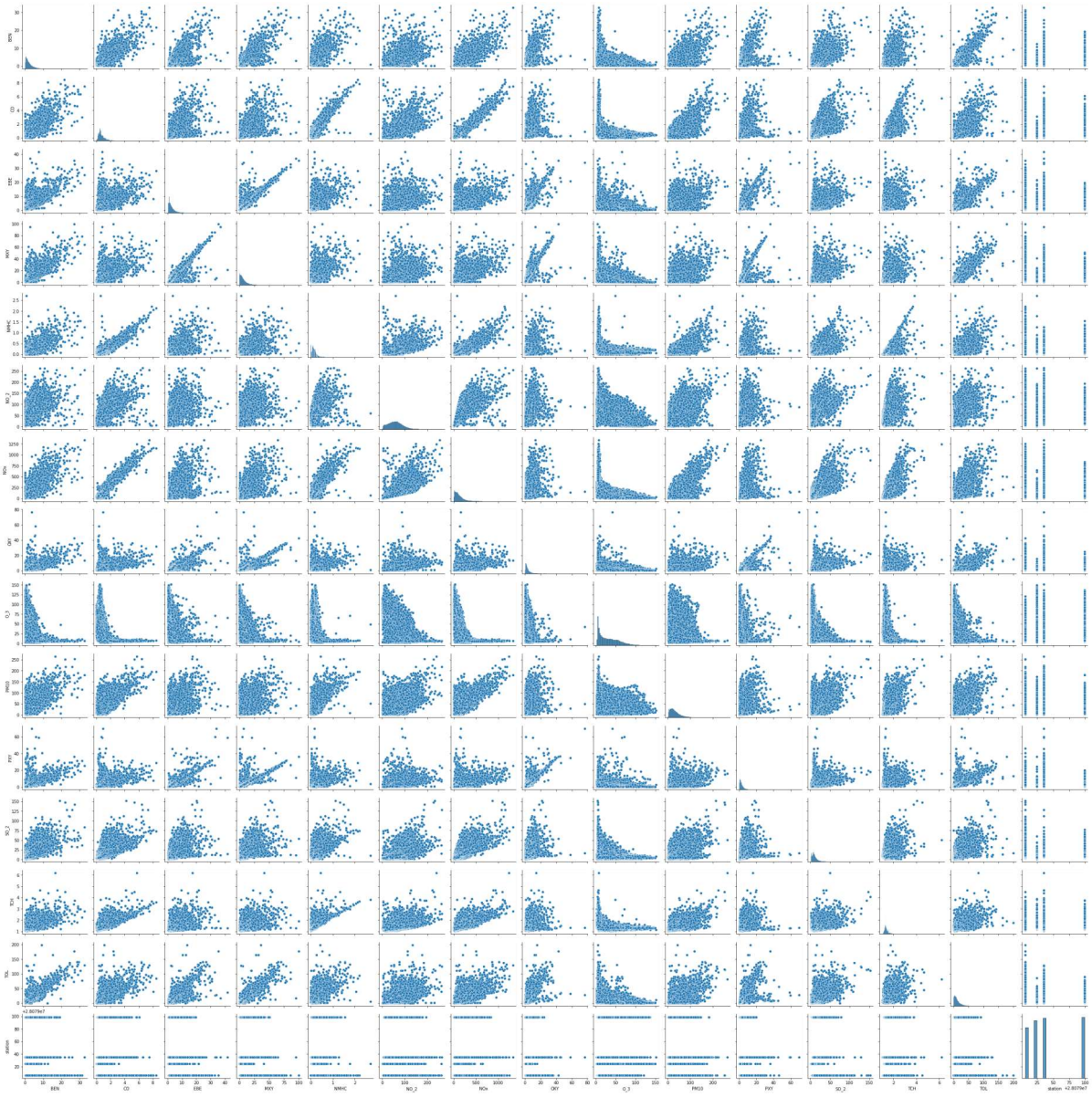
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	323
mean	2.479155	0.787323	2.914004	7.013636	0.155827	58.936796	1
std	2.280959	0.610810	2.667881	6.774365	0.135731	31.472733	1
min	0.180000	0.000000	0.180000	0.190000	0.000000	0.890000	
25%	0.970000	0.420000	1.140000	2.420000	0.080000	35.660000	
50%	1.840000	0.620000	2.130000	5.140000	0.130000	57.160000	
75%	3.250000	0.980000	3.830000	9.420000	0.200000	78.769997	1
max	32.660000	8.460000	41.740002	99.879997	2.700000	263.600006	13

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

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

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

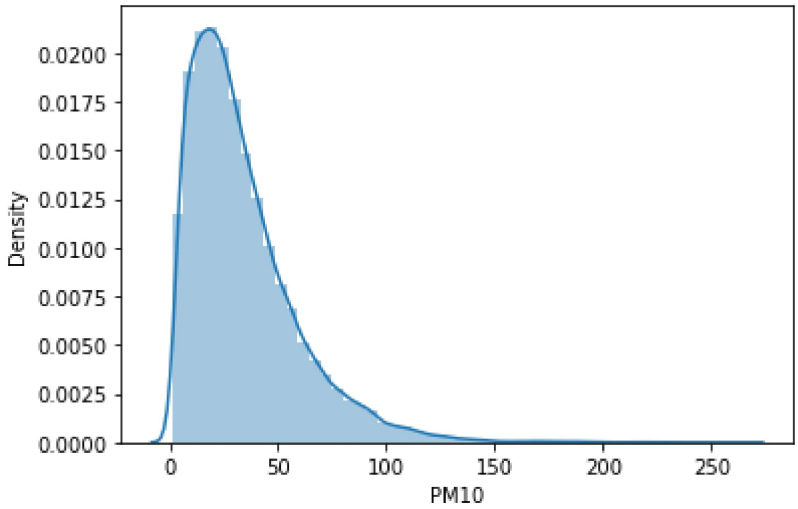
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1df74c12340>
```



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

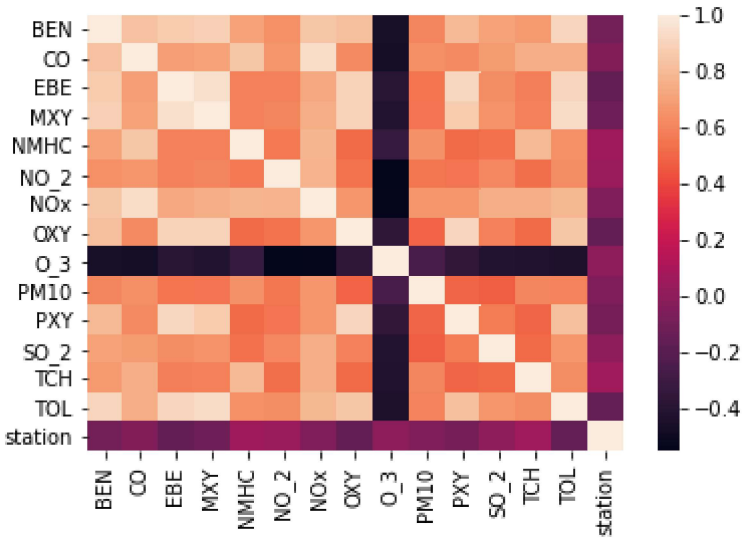
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function 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['NMHC']<1,'NMHC']=0
df.loc[df['NMHC']>1,'NMHC']=1
df['NMHC']=df['NMHC'].astype(int)
```

```
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-c5145d14383f>: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['NMHC']=df['NMHC'].astype(int)
```

LogisticRegression

In [31]:
```
x=df[[ 'BEN', 'CO', 'EBE', 'MXY', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
y=df['NMHC']
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[31]:
```
LogisticRegression()
```

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

Out[17]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

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

Out[18]:
```
0.9967061245496655
```

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

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

In [20]:
```
o=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
prediction=logr.predict(o)
print(prediction)
```

```
[0]
```

In [21]:
```
logr.classes_
```

Out[21]:
```
array([0, 1])
```

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

Out[22]:
```
0.9121773899252361
```

In [23]:
```
logr.predict_proba(o)[0][1]
```

Out[23]:
```
0.08782261007476391
```

LinearRegression

In [32]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

In [33]:
```
print(lr.intercept_)
```

```
-219.00391880080983
```

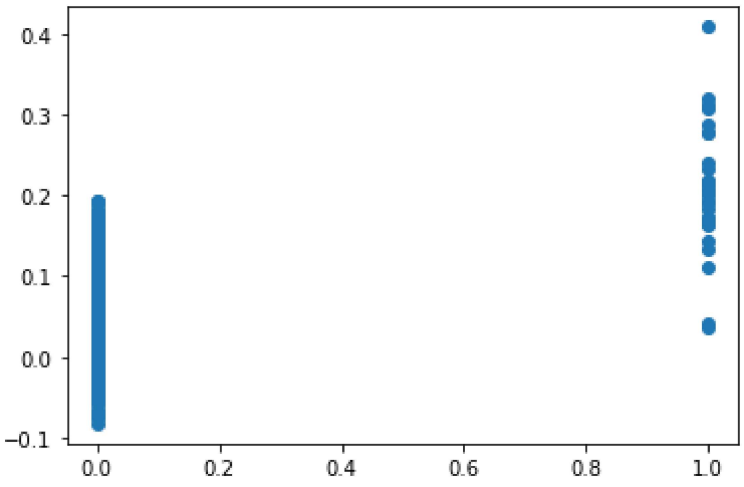
In [34]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[34]:

	Co-efficient
BEN	0.003820
CO	0.035804
EBE	0.000946
MXY	-0.001224
NO_2	-0.000463
NOx	0.000047
OXY	-0.001391
O_3	0.000249
PM10	-0.000032
PXY	0.000219
SO_2	0.000231
TCH	0.018479
TOL	0.000207
station	0.000008

In [35]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[35]:
```
<matplotlib.collections.PathCollection at 0x1df0e3ac850>
```



In [36]:
```
print(lr.score(x_test,y_test))
```

```
0.19891860476397782
```

Ridge,Lasso

In [37]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[37]:
```
Ridge(alpha=10)
```

In [38]:
```
rr.score(x_test,y_test)
```

Out[38]:
```
0.1987356164308257
```

In [39]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[39]:
```
Lasso(alpha=10)
```

In [40]:
```
la.score(x_test,y_test)
```

Out[40]:
```
-8.759608667974206e-06
```

ElasticNet

In [41]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[41]:
```
ElasticNet()
```

In [42]:
```
print(en.coef_)
```

```
[ 0.          0.          0.          0.         -0.          0.00012093
  0.          0.          0.          0.          0.          0.
  0.         -0.        ]
```

In [43]:
```
print(en.intercept_)
```

```
-0.012394759514420813
```

In [44]:
```
print(en.predict(x_train))
```

```
[-0.01052512  0.00116195 -0.00593325 ...  0.01623033  0.03632957
  0.00994176]
```

In [45]:
```
print(en.score(x_train,y_train))
```

```
0.11059506269439834
```

In [46]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 0.01616788249571067
```

In [47]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 0.0023841495745941764
```

In [48]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 0.12715298854415757
```

RandomForest

In [49]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[49]:
```
RandomForestClassifier()
```

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

In [51]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[51]:
```
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')
```

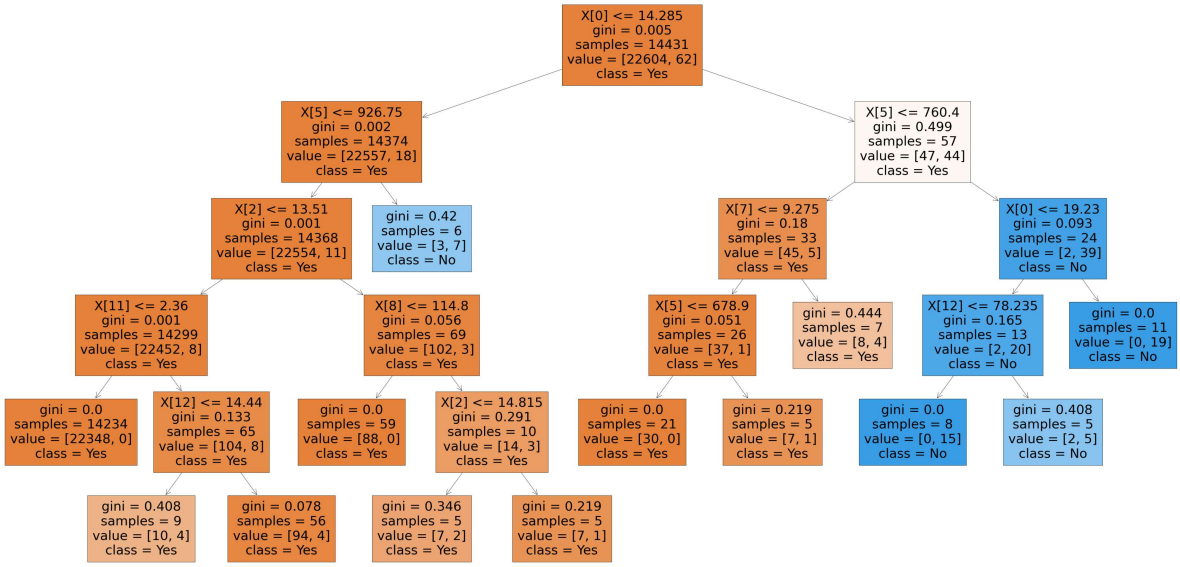
In [52]:
```
grid_search.best_score_
```

Out[52]:
```
0.9989411453278038
```

In [53]:
```
rfc_best=grid_search.best_estimator_
```

In [54]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],class_names=['Yes','No','Yes','No'],filled=T
```

Out[54]:
```
[Text(2363.2941176470586, 1993.2, 'X[0] <= 14.285\ngini = 0.005\nsamples = 14
431\nvalue = [22604, 62]\nclass = Yes'),
 Text(1312.941176470588, 1630.8000000000002, 'X[5] <= 926.75\ngini = 0.002\ns
amples = 14374\nvalue = [22557, 18]\nclass = Yes'),
 Text(1050.3529411764705, 1268.4, 'X[2] <= 13.51\ngini = 0.001\nsamples = 143
68\nvalue = [22554, 11]\nclass = Yes'),
 Text(525.1764705882352, 906.0, 'X[11] <= 2.36\ngini = 0.001\nsamples = 14299
\nvalue = [22452, 8]\nclass = Yes'),
 Text(262.5882352941176, 543.5999999999999, 'gini = 0.0\nsamples = 14234\nval
ue = [22348, 0]\nclass = Yes'),
 Text(787.7647058823529, 543.5999999999999, 'X[12] <= 14.44\ngini = 0.133\nsa
mples = 65\nvalue = [104, 8]\nclass = Yes'),
 Text(525.1764705882352, 181.19999999999982, 'gini = 0.408\nsamples = 9\nvalu
e = [10, 4]\nclass = Yes'),
 Text(1050.3529411764705, 181.19999999999982, 'gini = 0.078\nsamples = 56\nva
lue = [94, 4]\nclass = Yes'),
 Text(1575.5294117647059, 906.0, 'X[8] <= 114.8\ngini = 0.056\nsamples = 69\n
value = [102, 3]\nclass = Yes'),
 Text(1312.941176470588, 543.5999999999999, 'gini = 0.0\nsamples = 59\nvalue
= [88, 0]\nclass = Yes'),
 Text(1838.1176470588234, 543.5999999999999, 'X[2] <= 14.815\ngini = 0.291\ns
amples = 10\nvalue = [14, 3]\nclass = Yes'),
 Text(1575.5294117647059, 181.19999999999982, 'gini = 0.346\nsamples = 5\nval
ue = [7, 2]\nclass = Yes'),
 Text(2100.705882352941, 181.19999999999982, 'gini = 0.219\nsamples = 5\nvalu
e = [7, 1]\nclass = Yes'),
 Text(1575.5294117647059, 1268.4, 'gini = 0.42\nsamples = 6\nvalue = [3, 7]\n
class = No'),
 Text(3413.6470588235293, 1630.8000000000002, 'X[5] <= 760.4\ngini = 0.499\ns
amples = 57\nvalue = [47, 44]\nclass = Yes'),
 Text(2888.4705882352937, 1268.4, 'X[7] <= 9.275\ngini = 0.18\nsamples = 33\n
value = [45, 5]\nclass = Yes'),
 Text(2625.882352941176, 906.0, 'X[5] <= 678.9\ngini = 0.051\nsamples = 26\nv
alue = [37, 1]\nclass = Yes'),
 Text(2363.2941176470586, 543.5999999999999, 'gini = 0.0\nsamples = 21\nvalue
= [30, 0]\nclass = Yes'),
 Text(2888.4705882352937, 543.5999999999999, 'gini = 0.219\nsamples = 5\nvalu
e = [7, 1]\nclass = Yes'),
 Text(3151.0588235294117, 906.0, 'gini = 0.444\nsamples = 7\nvalue = [8, 4]\n
class = Yes'),
 Text(3938.8235294117644, 1268.4, 'X[0] <= 19.23\ngini = 0.093\nsamples = 24
\nvalue = [2, 39]\nclass = No'),
 Text(3676.235294117647, 906.0, 'X[12] <= 78.235\ngini = 0.165\nsamples = 13
\nvalue = [2, 20]\nclass = No'),
 Text(3413.6470588235293, 543.5999999999999, 'gini = 0.0\nsamples = 8\nvalue
= [0, 15]\nclass = No'),
 Text(3938.8235294117644, 543.5999999999999, 'gini = 0.408\nsamples = 5\nvalu
e = [2, 5]\nclass = No'),
 Text(4201.411764705882, 906.0, 'gini = 0.0\nsamples = 11\nvalue = [0, 19]\nc
lass = No')]
```



Best model:RandomForest

In [ ]: