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	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.880000	97.57
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.100000	25.82
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.430000	34.41
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.830000	28.26
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.990000	54.18
...	...	...	...	...	...	...	...	...	...	...	
230563	2006-05-01 00:00:00	5.88	0.83	6.23	NaN	0.20	112.500000	218.000000	NaN	24.389999	93.12
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.09	0.08	51.900002	54.820000	0.61	48.410000	29.46
230565	2006-05-01 00:00:00	0.96	NaN	0.69	NaN	0.19	135.100006	179.199997	NaN	11.460000	64.68
230566	2006-05-01 00:00:00	0.50	NaN	0.67	NaN	0.10	82.599998	105.599998	NaN	NaN	94.36
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.00	0.24	107.300003	160.199997	2.01	17.730000	52.49

230568 rows × 17 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230568 entries, 0 to 230567
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     230568 non-null  object 
 1   BEN      73979 non-null   float64
 2   CO       211665 non-null  float64
 3   EBE      73948 non-null   float64
 4   MXY      33422 non-null   float64
 5   NMHC     90829 non-null   float64
 6   NO_2     228855 non-null  float64
 7   NOx      228855 non-null  float64
 8   OXY      33472 non-null   float64
 9   O_3      216511 non-null  float64
 10  PM10     227469 non-null  float64
 11  PM25     61758 non-null   float64
 12  PXY      33447 non-null   float64
 13  SO_2     229125 non-null  float64
 14  TCH      90887 non-null   float64
 15  TOL      73840 non-null   float64
 16  station  230568 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 29.9+ MB
```

In [4]:

```
df=df.dropna()
df
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
5	2006-02-01 01:00:00	9.41	1.69	9.98	19.959999	0.44	142.199997	453.500000	11.31	5.990000
22	2006-02-01 01:00:00	1.69	0.79	1.24	2.670000	0.17	59.910000	120.199997	1.11	2.450000
25	2006-02-01 01:00:00	2.35	1.47	2.64	9.660000	0.40	117.699997	346.399994	5.15	4.780000
31	2006-02-01 02:00:00	4.39	0.85	7.92	17.139999	0.25	92.059998	237.000000	9.24	5.920000
48	2006-02-01 02:00:00	1.93	0.79	1.24	2.740000	0.16	60.189999	125.099998	1.11	2.280000
...	...	...	...	...	...	...	...	...	...	...
230538	2006-04-30 23:00:00	0.42	0.40	0.37	0.430000	0.10	49.259998	51.689999	1.00	64.599998
230541	2006-04-30 23:00:00	1.63	0.94	1.53	2.200000	0.33	63.220001	211.399994	1.35	17.670000
230547	2006-05-01 00:00:00	3.99	1.06	3.71	7.960000	0.26	202.399994	343.500000	3.92	11.130000
230564	2006-05-01 00:00:00	0.76	0.32	0.48	1.090000	0.08	51.900002	54.820000	0.61	48.410000
230567	2006-05-01 00:00:00	1.95	0.74	1.99	4.000000	0.24	107.300003	160.199997	2.01	17.730000

24758 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	
count	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	24758.000000	247
mean	1.350624	0.600713	1.824534	3.835034	0.176546	58.333481	1
std	1.541636	0.419048	1.868939	4.069036	0.126683	40.529382	1
min	0.110000	0.000000	0.170000	0.150000	0.000000	1.680000	
25%	0.450000	0.360000	0.810000	1.060000	0.100000	28.450001	
50%	0.850000	0.500000	1.130000	2.500000	0.150000	52.959999	
75%	1.680000	0.720000	2.160000	5.090000	0.220000	79.347498	1
max	45.430000	7.250000	57.799999	66.900002	2.020000	461.299988	16

In [7]:

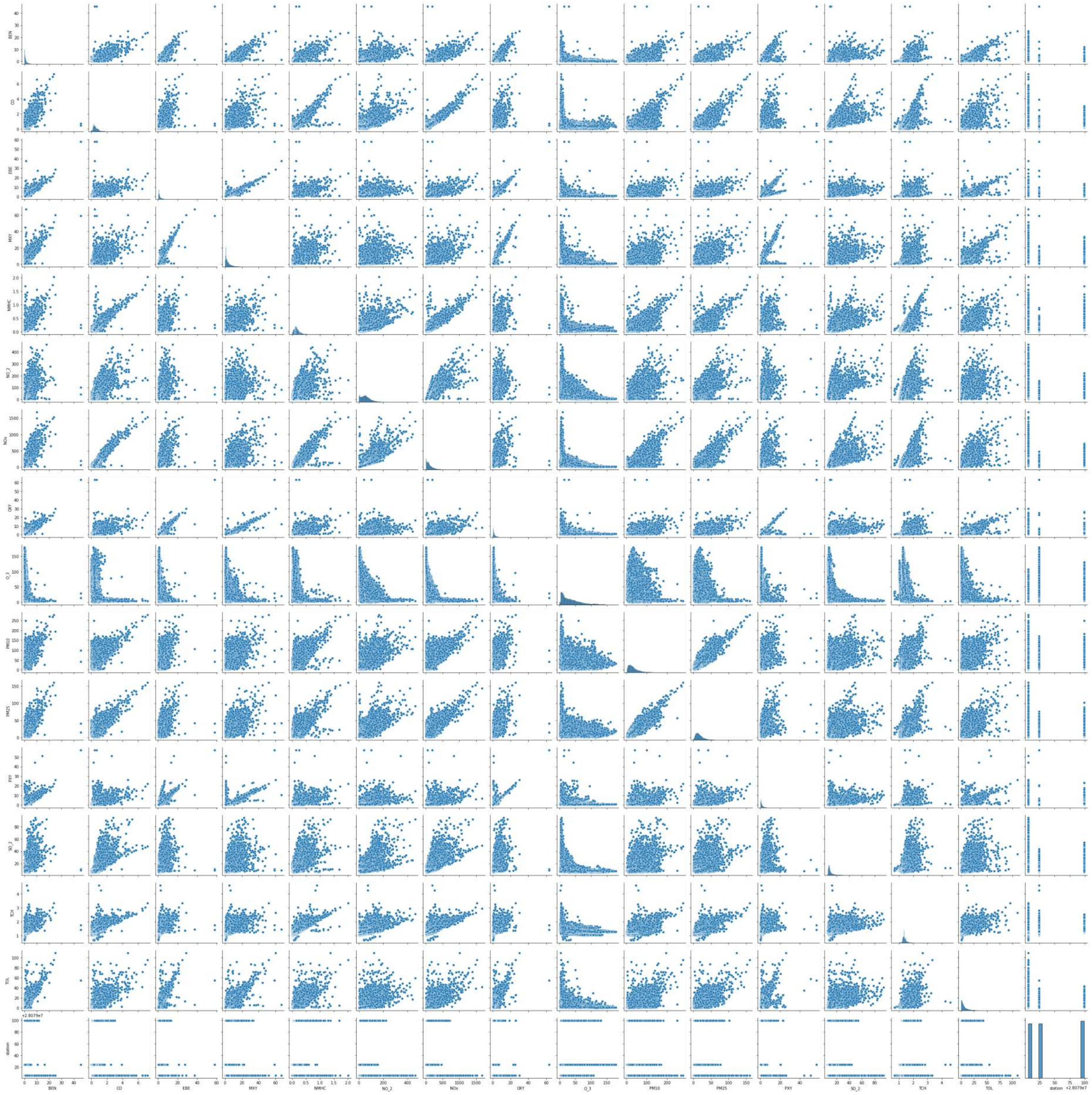
```
df.columns
```

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)
```

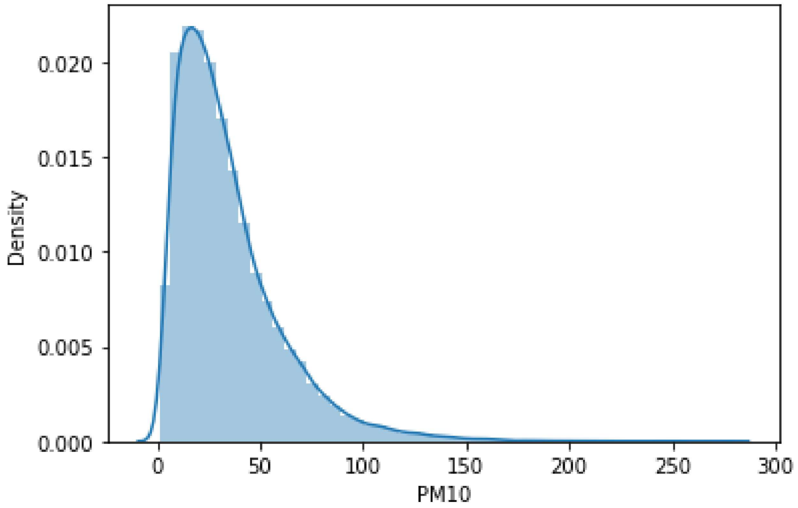
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x13f76e97c40>
```



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

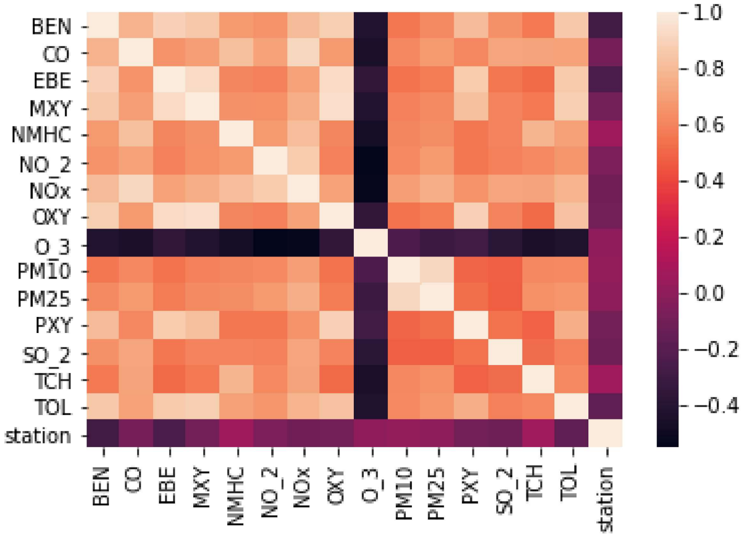
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  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['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-11-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 [12]:

```
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[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]:

```
0.9982498653742595
```

In [15]:

```
fs=StandardScaler().fit_transform(x)
logr=LogisticRegression()
logr.fit(fs,y)
```

Out[15]:

```
LogisticRegression()
```

In [16]:

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

```
[1]
```

In [17]:

```
logr.classes_
```

Out[17]:

```
array([0, 1])
```

In [18]:

```
logr.predict_proba(o)[0][0]
```

Out[18]:

```
0.11305170259168618
```

In [19]:

```
logr.predict_proba(o)[0][1]
```

Out[19]:

```
0.8869482974083138
```

LinearRegression

In [20]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[20]:

```
LinearRegression()
```

In [21]:

```
print(lr.intercept_)
```

```
-1444.599754329888
```

In [22]:

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

Out[22]:

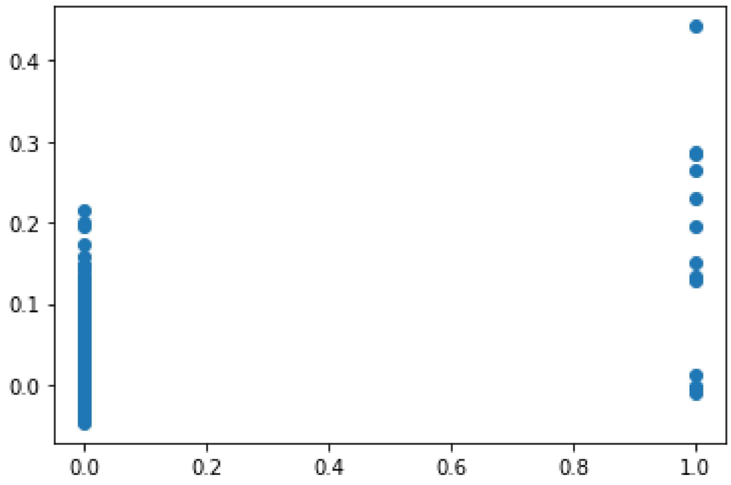
	Co-efficient
BEN	0.004637
CO	0.026814
EBE	0.001159
MXY	-0.000011
NO_2	-0.000513
NOx	0.000245
OXY	-0.003284
O_3	0.000179
PM10	-0.000016
PXY	-0.000786
SO_2	-0.000871
TCH	0.006763
TOL	-0.000158
station	0.000051

In [23]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[23]:

```
<matplotlib.collections.PathCollection at 0x13f0f882700>
```



In [24]:

```
print(lr.score(x_test,y_test))
```

```
0.09843150494198638
```

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]: 0.09852752809583232
```

```
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]: -4.1567792989916086e-05
```

ElasticNet

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

Out[29]: ElasticNet()
```

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

         [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
          -0.00000000e+00  9.43258244e-05  0.00000000e+00  0.00000000e+00
           0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
           0.00000000e+00 -0.00000000e+00]
```

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

         -0.009007361774978887
```

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

         [-0.00579745 -0.00150752 -0.00831407 ...  0.00987667 -0.00267244
           0.00586782]
```

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

         0.1093629893078445
```

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

         Mean Absolytre Error: 0.01177659098444107
```

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

```
Mean Square Error: 0.0015751047623351327
```

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

```
Root Mean Square Error: 0.10852000269278042
```

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)
```

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')
```

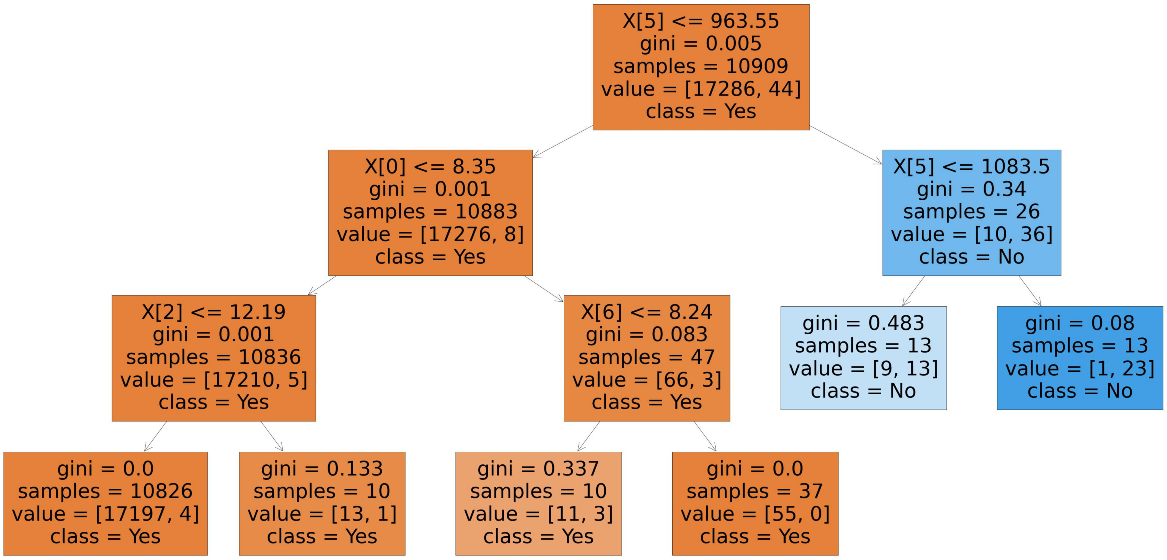
In [40]:
```
grid_search.best_score_
```

Out[40]:
```
0.9993652625504905
```

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(2637.818181818182, 1902.6000000000001, 'X[5] <= 963.55\ngini = 0.005\ns
amples = 10909\nvalue = [17286, 44]\nclass = Yes'),
 Text(1623.2727272727273, 1359.0, 'X[0] <= 8.35\ngini = 0.001\nsamples = 1088
3\nvalue = [17276, 8]\nclass = Yes'),
 Text(811.6363636363636, 815.4000000000001, 'X[2] <= 12.19\ngini = 0.001\nsam
ples = 10836\nvalue = [17210, 5]\nclass = Yes'),
 Text(405.8181818181818, 271.79999999999995, 'gini = 0.0\nsamples = 10826\nva
lue = [17197, 4]\nclass = Yes'),
 Text(1217.4545454545455, 271.79999999999995, 'gini = 0.133\nsamples = 10\nva
lue = [13, 1]\nclass = Yes'),
 Text(2434.909090909091, 815.4000000000001, 'X[6] <= 8.24\ngini = 0.083\nsamp
les = 47\nvalue = [66, 3]\nclass = Yes'),
 Text(2029.090909090909, 271.79999999999995, 'gini = 0.337\nsamples = 10\nval
ue = [11, 3]\nclass = Yes'),
 Text(2840.7272727272725, 271.79999999999995, 'gini = 0.0\nsamples = 37\nvalu
e = [55, 0]\nclass = Yes'),
 Text(3652.3636363636365, 1359.0, 'X[5] <= 1083.5\ngini = 0.34\nsamples = 26
\nvalue = [10, 36]\nclass = No'),
 Text(3246.5454545454545, 815.4000000000001, 'gini = 0.483\nsamples = 13\nval
ue = [9, 13]\nclass = No'),
 Text(4058.181818181818, 815.4000000000001, 'gini = 0.08\nsamples = 13\nvalue
= [1, 23]\nclass = No')]
```



Best model:RandomForest

In [ ]: