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	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.99(
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.95(
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.48(
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.07(
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.08(
...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.90(
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.68!
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.84(
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.63(
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.38!

245496 rows × 17 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245496 entries, 0 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     245496 non-null  object 
 1   BEN      65158 non-null   float64
 2   CO       226043 non-null  float64
 3   EBE      56781 non-null   float64
 4   MXY      39867 non-null   float64
 5   NMHC     107630 non-null  float64
 6   NO_2     243280 non-null  float64
 7   NOx      243283 non-null  float64
 8   OXY      39882 non-null   float64
 9   O_3      233811 non-null  float64
 10  PM10     234655 non-null  float64
 11  PM25     58145 non-null   float64
 12  PXY      39891 non-null   float64
 13  SO_2     243402 non-null  float64
 14  TCH      107650 non-null  float64
 15  TOL      64914 non-null   float64
 16  station  245496 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 31.8+ MB
```

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
5	2004-08-01 01:00:00	3.24	0.63	5.55	9.72	0.06	103.800003	144.800003	5.04	32.480000	59.11
22	2004-08-01 01:00:00	0.55	0.36	0.54	0.86	0.07	31.980000	32.799999	0.50	79.040001	43.54
26	2004-08-01 01:00:00	1.80	0.46	2.28	4.62	0.21	62.259998	75.470001	2.47	54.419998	46.63
32	2004-08-01 02:00:00	1.94	0.67	3.14	4.91	0.06	113.500000	165.800003	2.56	26.980000	86.93
49	2004-08-01 02:00:00	0.29	0.30	0.47	0.76	0.07	33.919998	34.840000	0.46	75.570000	48.95
...	...	...	...	...	...	...	...	...	...	...	
245463	2004-05-31 23:00:00	0.62	0.08	0.54	0.70	0.04	44.360001	45.450001	0.42	43.419998	19.29
245467	2004-05-31 23:00:00	2.39	0.67	2.49	3.92	0.20	89.809998	132.800003	2.09	14.740000	31.80
245473	2004-06-01 00:00:00	3.72	1.12	4.33	8.79	0.24	113.900002	253.600006	4.51	9.380000	21.21
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.90
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.38

19397 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]:

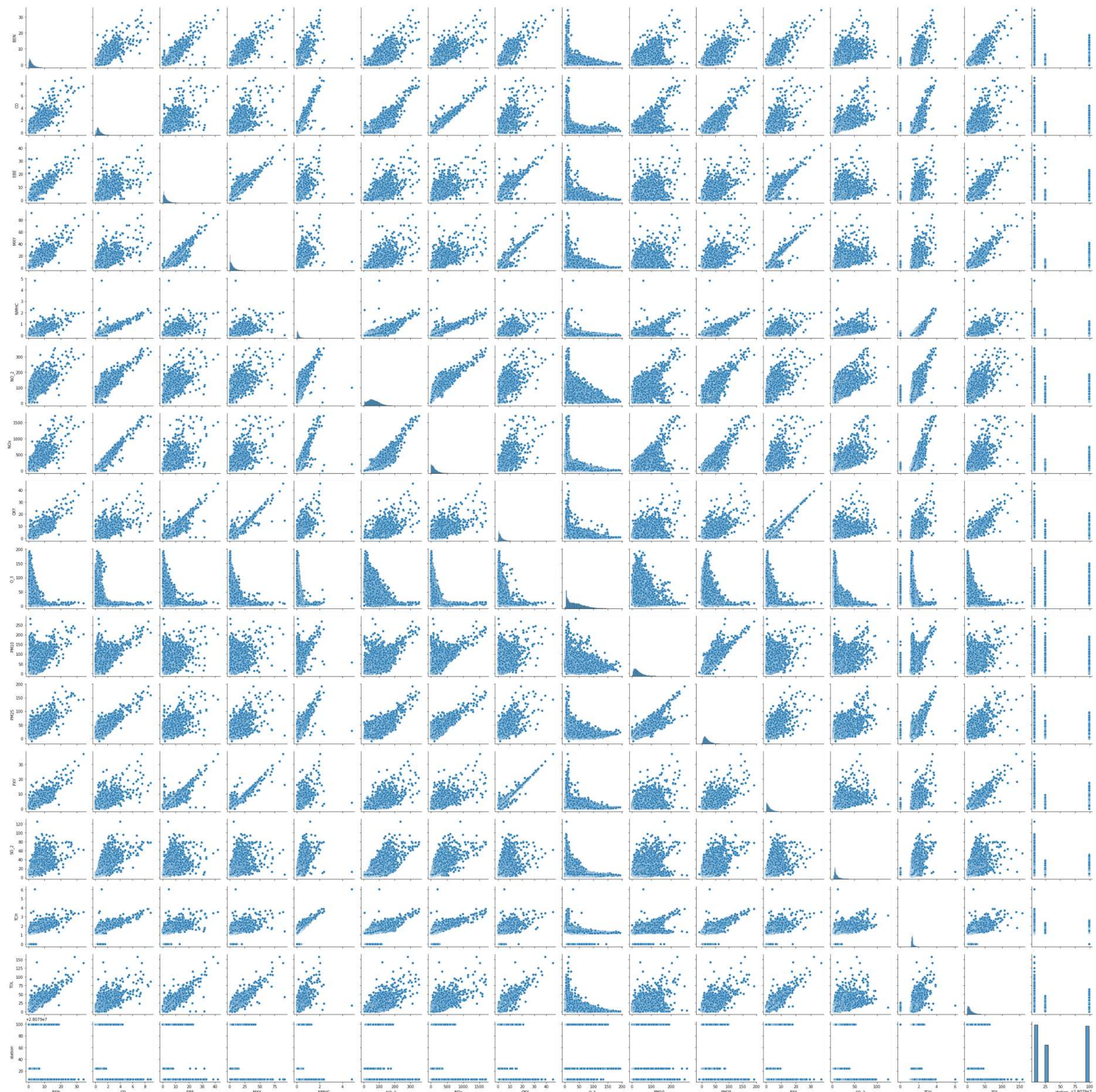
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	193
mean	2.250781	0.675347	2.775913	5.424809	0.151024	62.887023	1
std	2.184724	0.591026	2.729622	5.554358	0.158603	37.952255	1
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.090000	
25%	0.870000	0.320000	1.020000	1.780000	0.060000	35.150002	
50%	1.620000	0.520000	1.970000	3.800000	0.110000	58.310001	
75%	2.910000	0.860000	3.580000	7.260000	0.200000	85.730003	1
max	34.180000	8.900000	41.880001	91.599998	4.810000	355.100006	17

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)
```

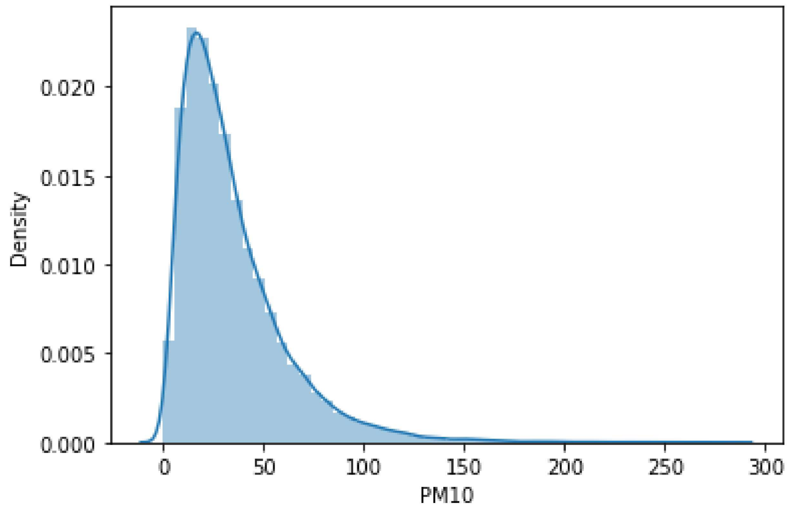
Out[8]: <seaborn.axisgrid.PairGrid at 0x1f8d84b0c70>



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

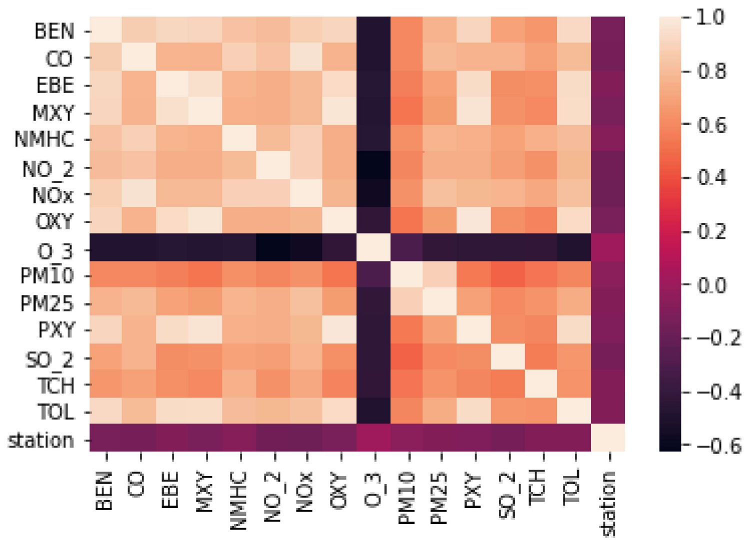
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
0.9962199312714777
```

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]:
```
1.2229580181877253e-07
```

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

Out[19]:
```
0.9999998777041982
```

LinearRegression

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

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

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

```
-1528.6478634991358
```

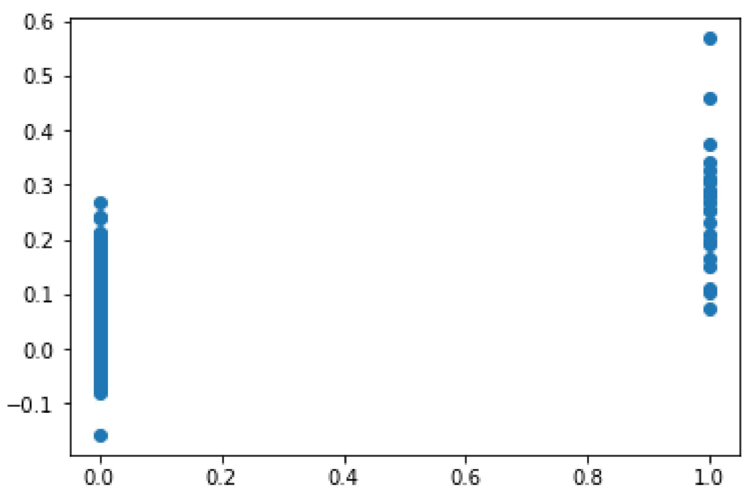
In [22]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[22]:

	Co-efficient
BEN	-0.002369
CO	0.031898
EBE	-0.005493
MXY	0.001408
NO_2	-0.000597
NOx	0.000320
OXY	0.004470
O_3	0.000380
PM10	0.000030
PXY	-0.003739
SO_2	-0.000747
TCH	0.039217
TOL	0.000005
station	0.000054

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

Out[23]: <matplotlib.collections.PathCollection at 0x1f8e41dee80>



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

```
0.22426765937058513
```

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]:
```
0.2251016207877765
```

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]:
```
-1.9487786558247677e-05
```

ElasticNet

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

In [30]:
```
print(en.coef_)
```
```
[ 0.          0.         -0.          0.         -0.          0.00018651
  0.          0.          0.          0.          0.          0.
  0.          0.        ]
```

In [31]:
```
print(en.intercept_)
```
```
-0.019955314751395007
```

In [32]:
```
print(en.predict(x_train))
```
```
[-0.00696309 -0.01493262 -0.01212007 ... -0.0058832   0.0128703
 -0.01508743]
```

In [33]:
```
print(en.score(x_train,y_train))
```
```
0.1896264668946248
```

In [34]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```
```
Mean Absolytre Error: 0.020022805597023283
```

In [35]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```
```
Mean Square Error: 0.002921237185476247
```

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

```
Root Mean Square Error: 0.14150196322674566
```

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')
```

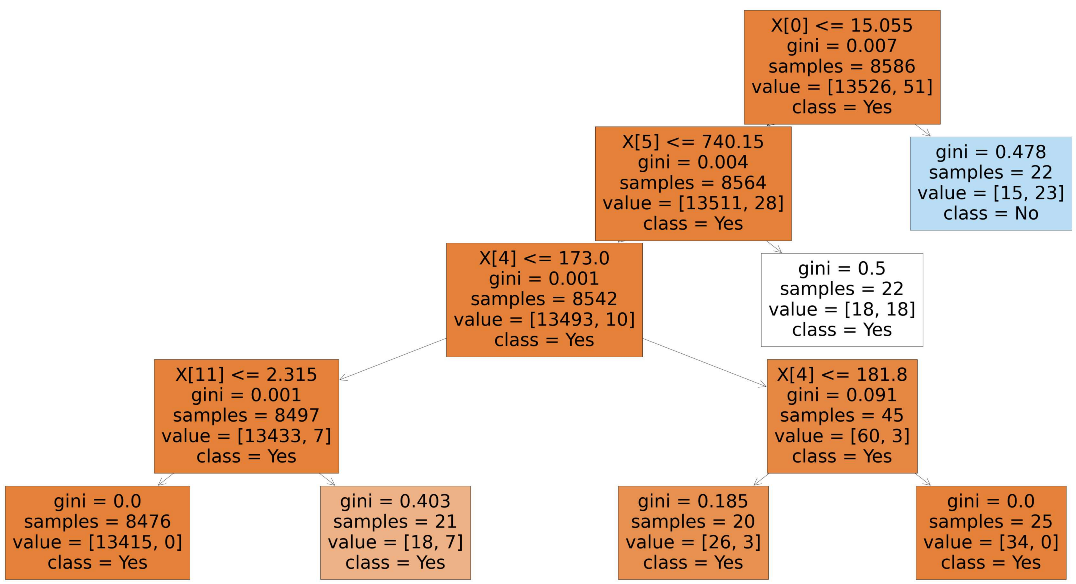
In [40]:
```
grid_search.best_score_
```

Out[40]:
```
0.9987479204158205
```

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=Tr
```

Out[42]:
```
[Text(3348.0, 1956.96, 'X[0] <= 15.055\ngini = 0.007\nsamples = 8586\nvalue =
[13526, 51]\nclass = Yes'),
 Text(2790.0, 1522.0800000000002, 'X[5] <= 740.15\ngini = 0.004\nsamples = 85
64\nvalue = [13511, 28]\nclass = Yes'),
 Text(2232.0, 1087.2, 'X[4] <= 173.0\ngini = 0.001\nsamples = 8542\nvalue =
[13493, 10]\nclass = Yes'),
 Text(1116.0, 652.3200000000002, 'X[11] <= 2.315\ngini = 0.001\nsamples = 849
7\nvalue = [13433, 7]\nclass = Yes'),
 Text(558.0, 217.44000000000005, 'gini = 0.0\nsamples = 8476\nvalue = [13415,
0]\nclass = Yes'),
 Text(1674.0, 217.44000000000005, 'gini = 0.403\nsamples = 21\nvalue = [18,
7]\nclass = Yes'),
 Text(3348.0, 652.3200000000002, 'X[4] <= 181.8\ngini = 0.091\nsamples = 45\n
value = [60, 3]\nclass = Yes'),
 Text(2790.0, 217.44000000000005, 'gini = 0.185\nsamples = 20\nvalue = [26,
3]\nclass = Yes'),
 Text(3906.0, 217.44000000000005, 'gini = 0.0\nsamples = 25\nvalue = [34, 0]
\nclass = Yes'),
 Text(3348.0, 1087.2, 'gini = 0.5\nsamples = 22\nvalue = [18, 18]\nclass = Ye
s'),
 Text(3906.0, 1522.0800000000002, 'gini = 0.478\nsamples = 22\nvalue = [15, 2
3]\nclass = No')]
```



Best model:RandomForest

In [ ]: