In [31]:
```
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	PM
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240(
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779
...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380(
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400(
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.830(
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.570(
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350(

243984 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243984 entries, 0 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     243984 non-null  object 
 1   BEN      69745 non-null   float64
 2   CO       225340 non-null  float64
 3   EBE      61244 non-null   float64
 4   MXY      42045 non-null   float64
 5   NMHC     111951 non-null  float64
 6   NO_2     242625 non-null  float64
 7   NOx      242629 non-null  float64
 8   OXY      42072 non-null   float64
 9   O_3      234131 non-null  float64
 10  PM10     240896 non-null  float64
 11  PXY      42063 non-null   float64
 12  SO_2     242729 non-null  float64
 13  TCH      111991 non-null  float64
 14  TOL      69439 non-null   float64
 15  station  243984 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 29.8+ MB
```

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
5	2003-03-01 01:00:00	8.41	1.94	9.83	21.49	0.45	90.300003	384.899994	9.48	9.950000	95.1
23	2003-03-01 01:00:00	3.46	1.27	3.43	7.08	0.18	54.250000	173.300003	3.37	6.540000	53.0
27	2003-03-01 01:00:00	6.39	1.79	5.75	10.88	0.33	75.459999	281.100006	3.68	6.690000	63.8
33	2003-03-01 02:00:00	7.42	1.47	10.63	24.73	0.35	83.309998	277.200012	11.00	9.900000	58.8
51	2003-03-01 02:00:00	3.62	1.29	3.20	7.08	0.19	42.209999	166.300003	3.41	6.380000	47.5
...	...	...	...	...	...	...	...	...	...	...	
243955	2003-09-30 23:00:00	1.75	0.41	3.07	9.38	0.09	46.290001	77.709999	3.11	18.280001	7.5
243957	2003-10-01 00:00:00	2.35	0.60	3.88	10.86	0.11	61.240002	133.100006	0.89	10.900000	10.2
243961	2003-10-01 00:00:00	2.97	0.82	4.53	10.88	0.05	36.529999	131.300003	5.52	12.940000	25.6
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.3
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.3

33010 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]:

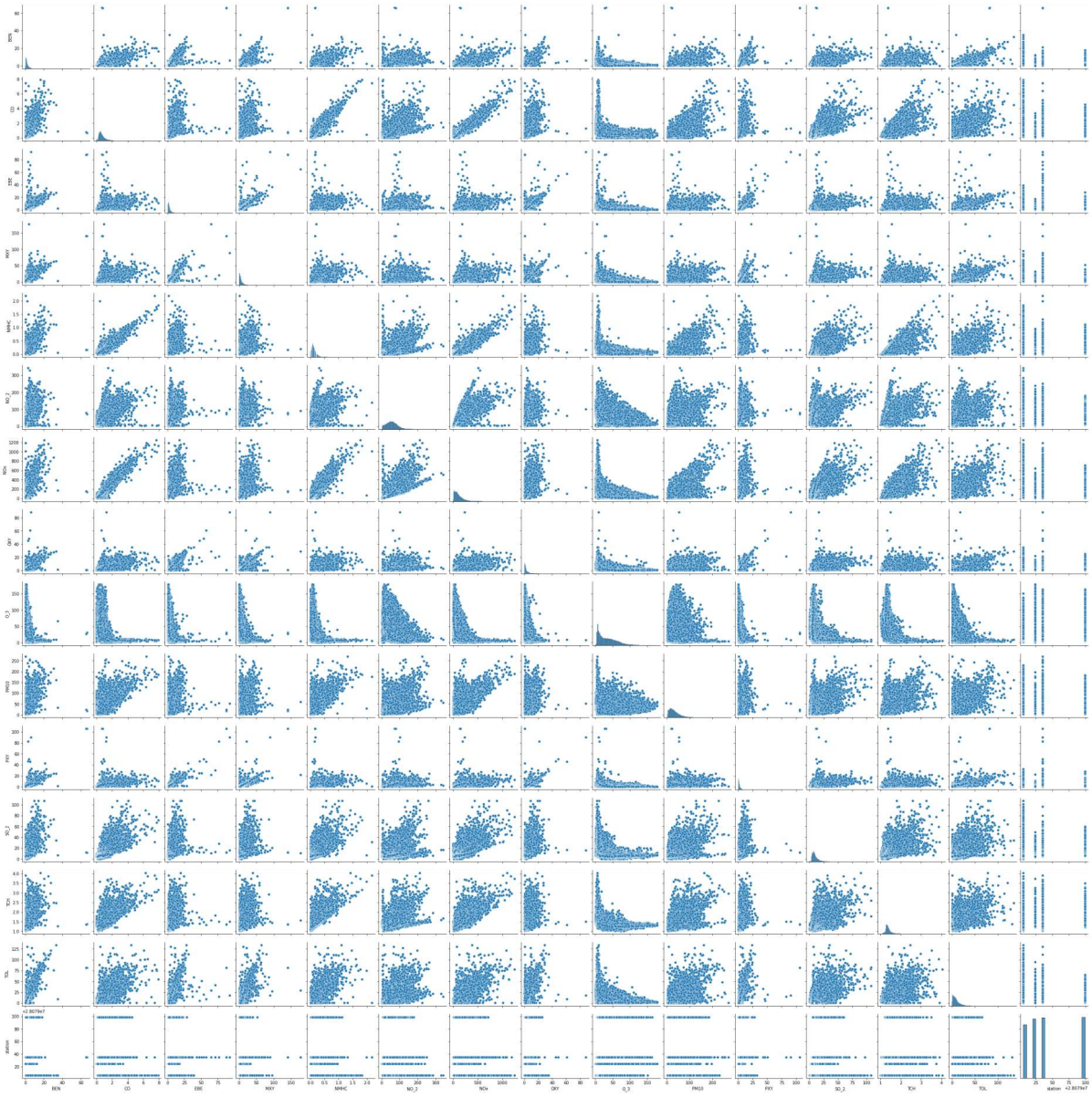
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	330
mean	2.192633	0.759868	2.639726	5.838414	0.137177	57.328049	1
std	2.064160	0.545999	2.825194	6.267296	0.127863	31.811082	1
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.900000	0.430000	1.010000	1.880000	0.060000	34.529999	
50%	1.610000	0.620000	1.890000	4.070000	0.110000	55.105000	
75%	2.810000	0.930000	3.300000	7.530000	0.170000	76.160004	1
max	66.389999	7.920000	92.589996	177.600006	2.180000	342.700012	12

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)
```

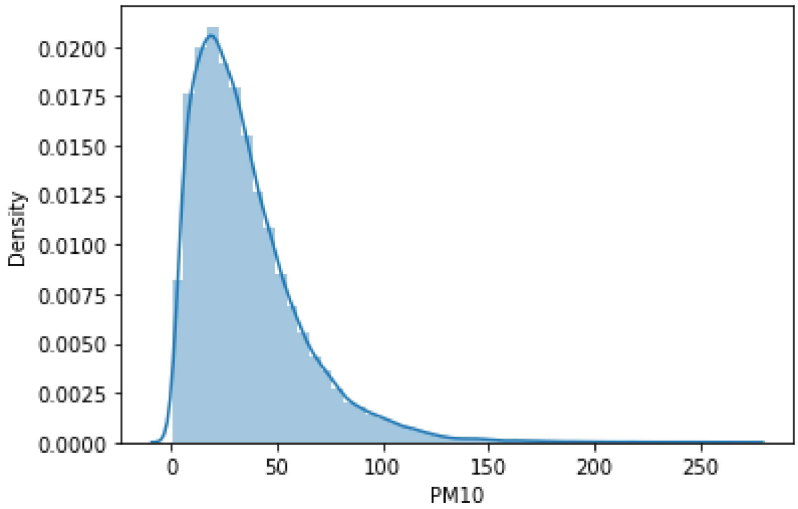
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1b8e68678b0>
```



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

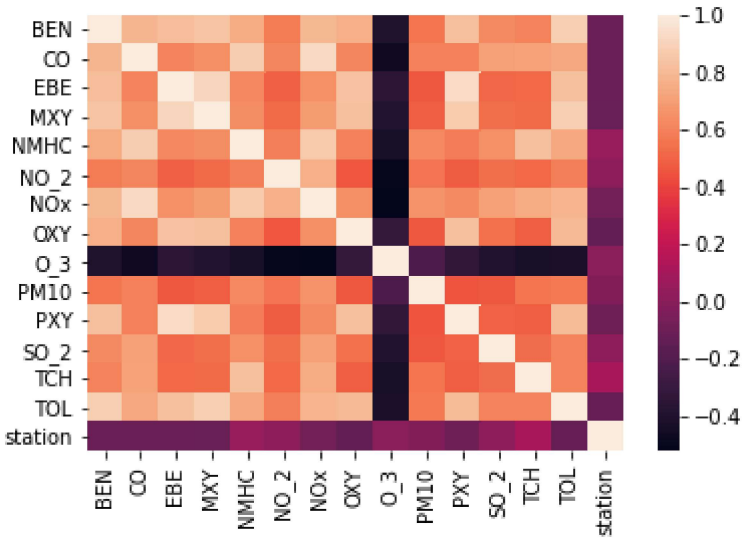
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[10]: <AxesSubplot:>



In [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]:
```
0.998081389477936
```

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)
```

```
[0]
```

In [17]:
```
logr.classes_
```

Out[17]:
```
array([0, 1])
```

In [18]:
```
logr.predict_proba(o)[0][0]
```

Out[18]:
```
0.9952057644483453
```

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

Out[19]:
```
0.004794235551654654
```

LinearRegression

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

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

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

```
54.51614752356137
```

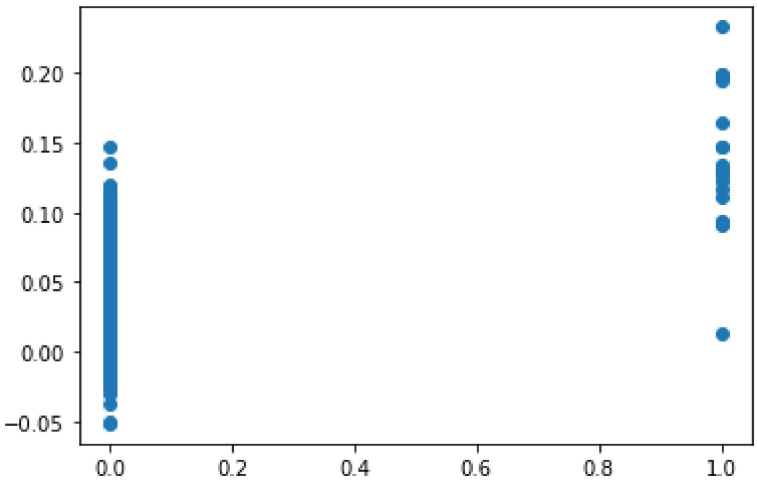
In [22]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[22]:

	Co-efficient
BEN	0.002046
CO	0.021282
EBE	0.001159
MXY	-0.000309
NO_2	-0.000254
NOx	0.000043
OXY	-0.000646
O_3	0.000143
PM10	0.000017
PXY	-0.001568
SO_2	0.000106
TCH	0.015362
TOL	0.000019
station	-0.000002

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

Out[23]:
```
<matplotlib.collections.PathCollection at 0x1b8ff5252b0>
```



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

```
0.1508392820800678
```

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]:
```
0.15072479822073515
```

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]:
```
-6.791998786304099e-05
```

ElasticNet

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

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

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

```
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 -0.00000000e+00  5.78617463e-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 [34]:
```
print(en.intercept_)
```

```
-0.005387629728312714
```

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

```
[ 1.68312809e-02  7.74120353e-05 -4.43811848e-03 ... -2.61373769e-03
 -4.47457139e-03 -4.76850905e-03]
```

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

```
0.06034003814908018
```

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

```
Mean Absolytre Error: 0.010284228831308242
```

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

```
Mean Square Error: 0.0016260828713925628
```

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

```
Root Mean Square Error: 0.10141118691400985
```

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')
```

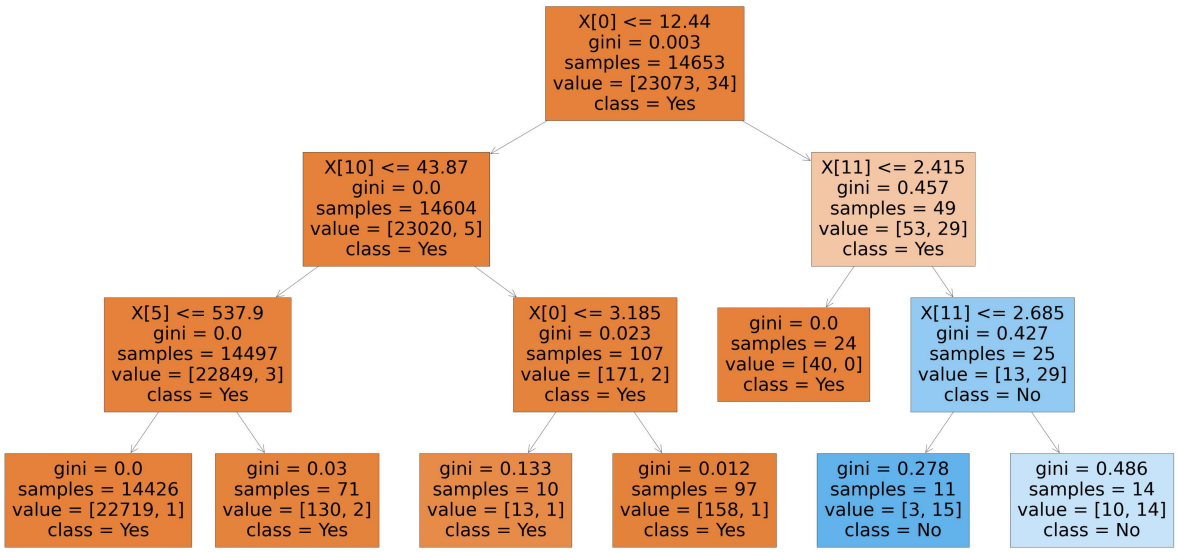
In [43]:
```
grid_search.best_score_
```

Out[43]:
```
0.9992643015689101
```

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(2418.0, 1902.6000000000001, 'X[0] <= 12.44\ngini = 0.003\nsamples = 146
53\nvalue = [23073, 34]\nclass = Yes'),
 Text(1488.0, 1359.0, 'X[10] <= 43.87\ngini = 0.0\nsamples = 14604\nvalue =
[23020, 5]\nclass = Yes'),
 Text(744.0, 815.4000000000001, 'X[5] <= 537.9\ngini = 0.0\nsamples = 14497\n
value = [22849, 3]\nclass = Yes'),
 Text(372.0, 271.79999999999995, 'gini = 0.0\nsamples = 14426\nvalue = [2271
9, 1]\nclass = Yes'),
 Text(1116.0, 271.79999999999995, 'gini = 0.03\nsamples = 71\nvalue = [130,
2]\nclass = Yes'),
 Text(2232.0, 815.4000000000001, 'X[0] <= 3.185\ngini = 0.023\nsamples = 107
\nvalue = [171, 2]\nclass = Yes'),
 Text(1860.0, 271.79999999999995, 'gini = 0.133\nsamples = 10\nvalue = [13,
1]\nclass = Yes'),
 Text(2604.0, 271.79999999999995, 'gini = 0.012\nsamples = 97\nvalue = [158,
1]\nclass = Yes'),
 Text(3348.0, 1359.0, 'X[11] <= 2.415\ngini = 0.457\nsamples = 49\nvalue = [5
3, 29]\nclass = Yes'),
 Text(2976.0, 815.4000000000001, 'gini = 0.0\nsamples = 24\nvalue = [40, 0]\n
class = Yes'),
 Text(3720.0, 815.4000000000001, 'X[11] <= 2.685\ngini = 0.427\nsamples = 25
\nvalue = [13, 29]\nclass = No'),
 Text(3348.0, 271.79999999999995, 'gini = 0.278\nsamples = 11\nvalue = [3, 1
5]\nclass = No'),
 Text(4092.0, 271.79999999999995, 'gini = 0.486\nsamples = 14\nvalue = [10, 1
4]\nclass = No')]
```



Best model:RandomForest

In [ ]:
```
```