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_201
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN	2
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	2
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0	2
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN	2
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN	2
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN	2
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN	2
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN	2
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN	2

209496 rows × 14 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209496 entries, 0 to 209495
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209496 non-null  object 
 1   BEN      50755 non-null   float64
 2   CO       85999 non-null   float64
 3   EBE      50335 non-null   float64
 4   NMHC     25970 non-null   float64
 5   NO       208614 non-null  float64
 6   NO_2     208614 non-null  float64
 7   O_3      121197 non-null  float64
 8   PM10     102892 non-null  float64
 9   PM25     52165 non-null   float64
 10  SO_2     86023 non-null   float64
 11  TCH      25970 non-null   float64
 12  TOL      50662 non-null   float64
 13  station  209496 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ 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	
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	28
6	2016-11-01 01:00:00	0.7	0.8	0.4	0.13	57.0	66.0	3.0	23.0	15.0	4.0	1.35	5.0	28
25	2016-11-01 02:00:00	2.7	1.0	2.1	0.40	139.0	114.0	4.0	37.0	21.0	14.0	2.30	15.0	28
30	2016-11-01 02:00:00	0.7	0.7	0.4	0.13	48.0	59.0	3.0	23.0	15.0	3.0	1.35	5.0	28
49	2016-11-01 03:00:00	1.7	0.8	1.4	0.25	53.0	90.0	4.0	31.0	19.0	10.0	1.95	10.7	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209430	2016-06-30 22:00:00	0.1	0.2	0.1	0.02	1.0	5.0	97.0	19.0	12.0	2.0	1.15	0.2	28
209449	2016-06-30 23:00:00	0.6	0.4	0.3	0.15	14.0	63.0	54.0	29.0	13.0	16.0	1.48	1.9	28
209454	2016-06-30 23:00:00	0.1	0.2	0.1	0.02	1.0	7.0	91.0	16.0	9.0	2.0	1.15	0.3	28
209473	2016-07-01 00:00:00	0.6	0.4	0.3	0.16	11.0	68.0	45.0	24.0	14.0	16.0	1.50	1.9	28
209478	2016-07-01 00:00:00	0.1	0.2	0.1	0.02	1.0	6.0	89.0	16.0	9.0	2.0	1.15	0.2	28

16932 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]:

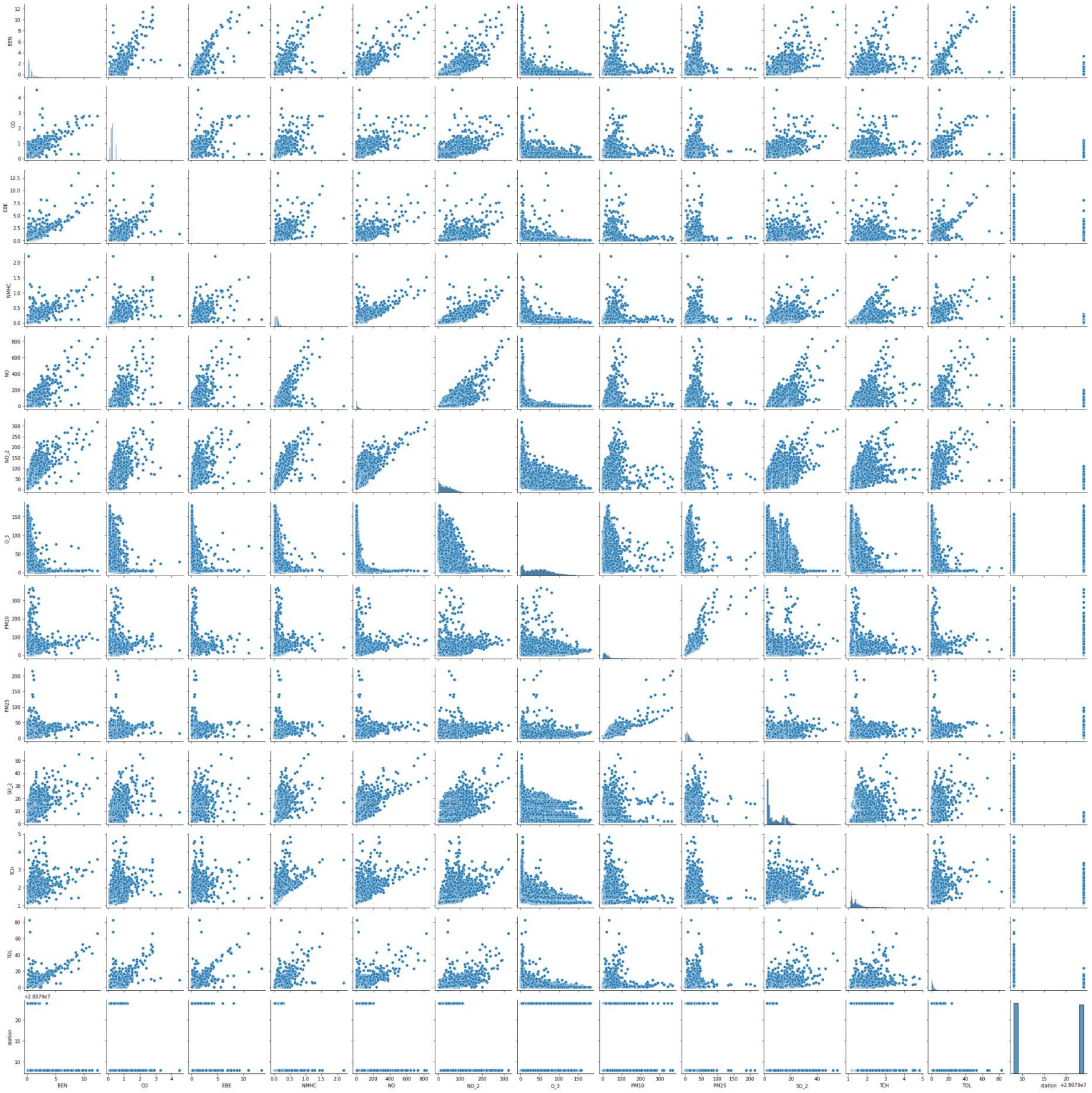
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	169
mean	0.537970	0.349941	0.298955	0.099913	20.815734	39.373376	
std	0.599479	0.203807	0.450204	0.079850	40.986063	31.170307	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	
25%	0.200000	0.200000	0.100000	0.050000	1.000000	14.000000	
50%	0.400000	0.300000	0.200000	0.090000	7.000000	34.000000	
75%	0.700000	0.400000	0.300000	0.120000	23.000000	58.000000	
max	12.300000	4.500000	13.500000	2.210000	829.000000	319.000000	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)
```

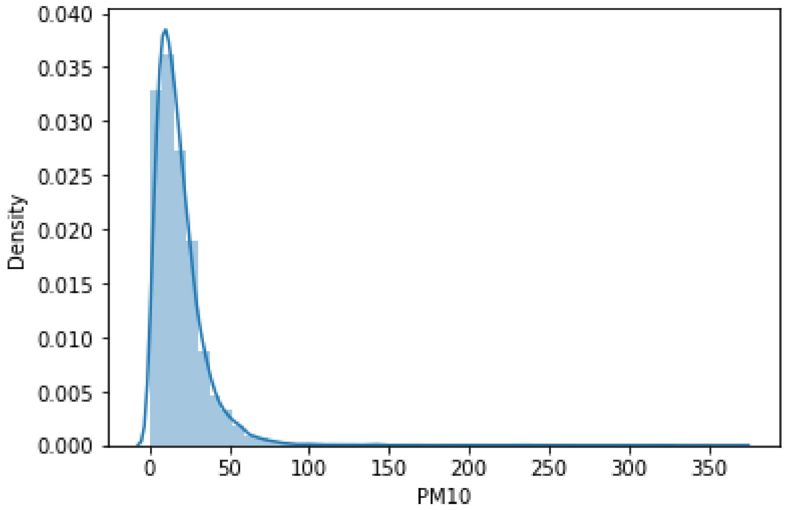
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x20f4d2c7220>
```



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

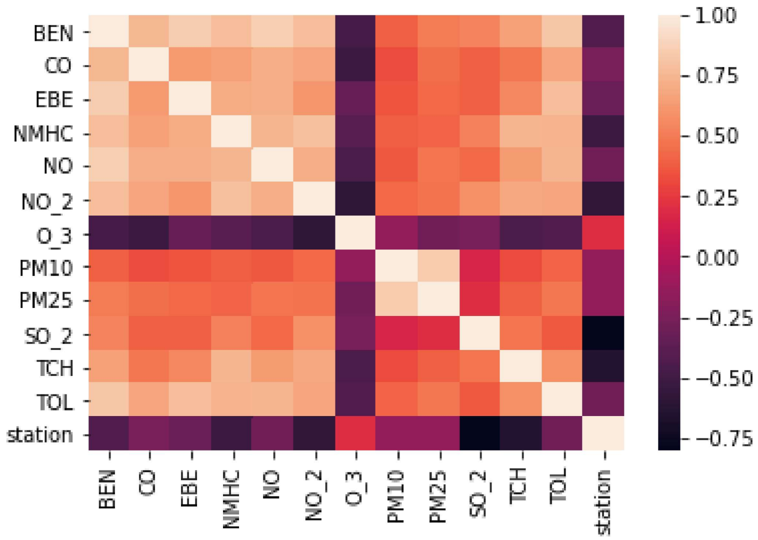
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function 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['TCH']<2,'TCH']=0
df.loc[df['TCH']>2,'TCH']=1
df['TCH']=df['TCH'].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-11-e3d36a273982>: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['TCH']=df['TCH'].astype(int)
```

Out[11]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	1	14.4	28
6	2016-11-01 01:00:00	0.7	0.8	0.4	0.13	57.0	66.0	3.0	23.0	15.0	4.0	0	5.0	28
25	2016-11-01 02:00:00	2.7	1.0	2.1	0.40	139.0	114.0	4.0	37.0	21.0	14.0	1	15.0	28
30	2016-11-01 02:00:00	0.7	0.7	0.4	0.13	48.0	59.0	3.0	23.0	15.0	3.0	0	5.0	28
49	2016-11-01 03:00:00	1.7	0.8	1.4	0.25	53.0	90.0	4.0	31.0	19.0	10.0	0	10.7	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209430	2016-06-30 22:00:00	0.1	0.2	0.1	0.02	1.0	5.0	97.0	19.0	12.0	2.0	0	0.2	28
209449	2016-06-30 23:00:00	0.6	0.4	0.3	0.15	14.0	63.0	54.0	29.0	13.0	16.0	0	1.9	28
209454	2016-06-30 23:00:00	0.1	0.2	0.1	0.02	1.0	7.0	91.0	16.0	9.0	2.0	0	0.3	28
209473	2016-07-01 00:00:00	0.6	0.4	0.3	0.16	11.0	68.0	45.0	24.0	14.0	16.0	0	1.9	28
209478	2016-07-01 00:00:00	0.1	0.2	0.1	0.02	1.0	6.0	89.0	16.0	9.0	2.0	0	0.2	28

16932 rows × 14 columns

LogisticRegression

In [12]:

```
x=df[[ 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TOL', 'station']]
y=df['TCH']
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]: 0.9702755905511811
```

```
In [15]: fs=StandardScaler().fit_transform(x)
         logr=LogisticRegression()
         logr.fit(fs,y)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

```
Out[15]: LogisticRegression()
```

```
In [16]: o=[[1,2,3,4,5,6,7,8,9,10,11,12]]
         prediction=logr.predict(o)
         print(prediction)
```

```
[0]
```

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

```
Out[17]: array([0, 1, 2])
```

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

```
Out[18]: 0.9999999999999951
```

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

```
Out[19]: 4.962724005677087e-15
```

LinearRegression

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

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

In [21]:

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

```
36395.73166804244
```

In [22]:

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

Out[22]:

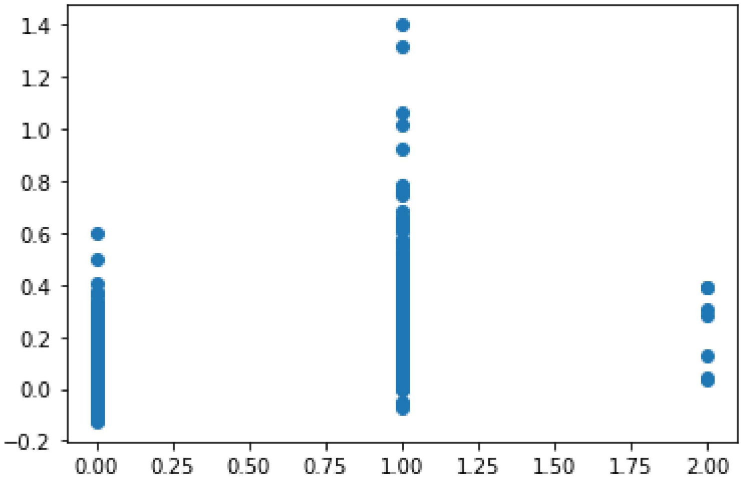
	Co-efficient
BEN	-0.012937
CO	-0.099476
EBE	0.021893
NMHC	1.082968
NO	0.001672
NO_2	-0.001474
O_3	-0.000293
PM10	-0.000666
PM25	0.002112
SO_2	-0.003539
TOL	-0.000923
station	-0.001296

In [23]:

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

Out[23]:

```
<matplotlib.collections.PathCollection at 0x20f5df91370>
```



In [24]:

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

```
0.31204558562684215
```

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]: 0.3113198549060787
```

```
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]: -1.220663748746631e-06
```

ElasticNet

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

Out[29]: ElasticNet()
```

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

         [ 0.          0.          0.          0.          0.00183732  0.
          -0.          0.          0.         -0.          0.         -0.        ]
```

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

         -0.007178338493906651
```

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

         [ 0.01119485 -0.00534102 -0.00534102 ...  0.00935753 -0.00534102
          -0.00534102]
```

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

         0.24517361988217445
```

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

         Mean Absolytre Error: 0.057938190734859646
```

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

```
Mean Square Error: 0.022627447265107833
```

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

```
Root Mean Square Error: 0.2407035328674252
```

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)
```

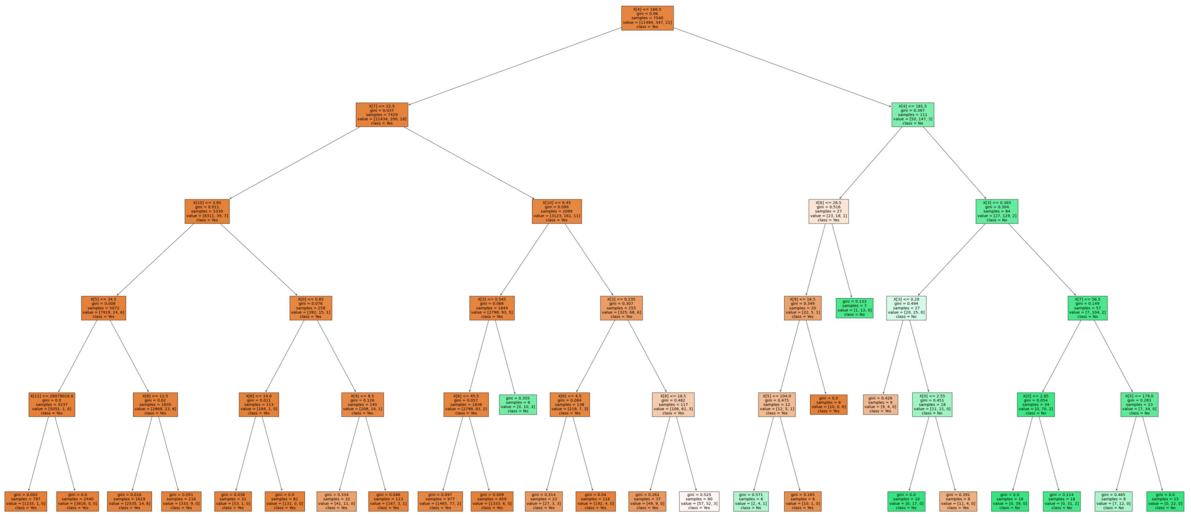
Out[39]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

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

Out[40]:
```
0.9785690178872763
```

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(2426.086956521739, 1993.2, 'X[4] <= 166.5\ngini = 0.06\nsamples = 7540
         \nvalue = [11484, 347, 21]\nclass = Yes'),
          Text(1431.391304347826, 1630.8000000000002, 'X[7] <= 22.5\ngini = 0.037\nsam
         ples = 7429\nvalue = [11434, 200, 18]\nclass = Yes'),
          Text(776.3478260869565, 1268.4, 'X[10] <= 3.85\ngini = 0.011\nsamples = 5330
         \nvalue = [8311, 39, 7]\nclass = Yes'),
          Text(388.17391304347825, 906.0, 'X[5] <= 34.5\ngini = 0.008\nsamples = 5072
         \nvalue = [7919, 24, 6]\nclass = Yes'),
          Text(194.08695652173913, 543.5999999999999, 'X[11] <= 28079016.0\ngini = 0.0
         \nsamples = 3237\nvalue = [5051, 1, 0]\nclass = Yes'),
          Text(97.04347826086956, 181.19999999999982, 'gini = 0.002\nsamples = 797\nva
         lue = [1235, 1, 0]\nclass = Yes'),
          Text(291.1304347826087, 181.19999999999982, 'gini = 0.0\nsamples = 2440\nval
         ue = [3816, 0, 0]\nclass = Yes'),
          Text(582.2608695652174, 543.5999999999999, 'X[8] <= 12.5\ngini = 0.02\nsampl
         es = 1835\nvalue = [2868, 23, 6]\nclass = Yes'),
          Text(485.2173913043478, 181.19999999999982, 'gini = 0.016\nsamples = 1619\nv
         alue = [2535, 14, 6]\nclass = Yes'),
          Text(679.304347826087, 181.19999999999982, 'gini = 0.051\nsamples = 216\nval
         ue = [333, 9, 0]\nclass = Yes'),
          Text(1164.5217391304348, 906.0, 'X[0] <= 0.85\ngini = 0.076\nsamples = 258\n
         value = [392, 15, 1]\nclass = Yes'),
          Text(970.4347826086956, 543.5999999999999, 'X[6] <= 14.0\ngini = 0.011\nsamp
         les = 113\nvalue = [184, 1, 0]\nclass = Yes'),
          Text(873.391304347826, 181.19999999999982, 'gini = 0.036\nsamples = 31\nvalu
         e = [53, 1, 0]\nclass = Yes'),
          Text(1067.4782608695652, 181.19999999999982, 'gini = 0.0\nsamples = 82\nvalu
         e = [131, 0, 0]\nclass = Yes'),
          Text(1358.608695652174, 543.5999999999999, 'X[9] <= 8.5\ngini = 0.126\nsampl
         es = 145\nvalue = [208, 14, 1]\nclass = Yes'),
          Text(1261.5652173913043, 181.19999999999982, 'gini = 0.334\nsamples = 32\nva
         lue = [41, 11, 0]\nclass = Yes'),
          Text(1455.6521739130435, 181.19999999999982, 'gini = 0.046\nsamples = 113\nv
         alue = [167, 3, 1]\nclass = Yes'),
          Text(2086.4347826086955, 1268.4, 'X[10] <= 6.45\ngini = 0.099\nsamples = 209
         9\nvalue = [3123, 161, 11]\nclass = Yes'),
          Text(1843.8260869565217, 906.0, 'X[3] <= 0.545\ngini = 0.066\nsamples = 1844
         \nvalue = [2798, 93, 5]\nclass = Yes'),
          Text(1746.782608695652, 543.5999999999999, 'X[6] <= 45.5\ngini = 0.057\nsamp
         les = 1836\nvalue = [2798, 83, 2]\nclass = Yes'),
          Text(1649.7391304347825, 181.19999999999982, 'gini = 0.097\nsamples = 977\nv
         alue = [1465, 77, 2]\nclass = Yes'),
          Text(1843.8260869565217, 181.19999999999982, 'gini = 0.009\nsamples = 859\nv
         alue = [1333, 6, 0]\nclass = Yes'),
          Text(1940.8695652173913, 543.5999999999999, 'gini = 0.355\nsamples = 8\nvalu
         e = [0, 10, 3]\nclass = No'),
          Text(2329.0434782608695, 906.0, 'X[3] <= 0.235\ngini = 0.307\nsamples = 255
         \nvalue = [325, 68, 6]\nclass = Yes'),
          Text(2134.9565217391305, 543.5999999999999, 'X[6] <= 4.5\ngini = 0.084\nsamp
         les = 138\nvalue = [219, 7, 3]\nclass = Yes'),
          Text(2037.9130434782608, 181.19999999999982, 'gini = 0.314\nsamples = 22\nva
         lue = [27, 3, 3]\nclass = Yes'),
          Text(2232.0, 181.19999999999982, 'gini = 0.04\nsamples = 116\nvalue = [192,
         4, 0]\nclass = Yes'),
          Text(2523.1304347826085, 543.5999999999999, 'X[8] <= 18.5\ngini = 0.482\nsam
         ples = 117\nvalue = [106, 61, 3]\nclass = Yes'),
          Text(2426.086956521739, 181.19999999999982, 'gini = 0.262\nsamples = 37\nval
ue = [49, 9, 0]\nclass = Yes'),
 Text(2620.173913043478, 181.19999999999982, 'gini = 0.525\nsamples = 80\nval
ue = [57, 52, 3]\nclass = Yes'),
 Text(3420.782608695652, 1630.8000000000002, 'X[4] <= 181.5\ngini = 0.397\nsa
mples = 111\nvalue = [50, 147, 3]\nclass = No'),
 Text(3105.391304347826, 1268.4, 'X[8] <= 28.5\ngini = 0.516\nsamples = 27\nv
alue = [23, 18, 1]\nclass = Yes'),
 Text(3008.3478260869565, 906.0, 'X[9] <= 16.5\ngini = 0.349\nsamples = 20\nv
alue = [22, 5, 1]\nclass = Yes'),
 Text(2911.304347826087, 543.5999999999999, 'X[5] <= 104.0\ngini = 0.475\nsam
ples = 12\nvalue = [12, 5, 1]\nclass = Yes'),
 Text(2814.2608695652175, 181.19999999999982, 'gini = 0.571\nsamples = 6\nval
ue = [2, 4, 1]\nclass = No'),
 Text(3008.3478260869565, 181.19999999999982, 'gini = 0.165\nsamples = 6\nval
ue = [10, 1, 0]\nclass = Yes'),
 Text(3105.391304347826, 543.5999999999999, 'gini = 0.0\nsamples = 8\nvalue =
[10, 0, 0]\nclass = Yes'),
 Text(3202.4347826086955, 906.0, 'gini = 0.133\nsamples = 7\nvalue = [1, 13,
0]\nclass = No'),
 Text(3736.173913043478, 1268.4, 'X[3] <= 0.365\ngini = 0.304\nsamples = 84\n
value = [27, 129, 2]\nclass = No'),
 Text(3396.5217391304345, 906.0, 'X[3] <= 0.28\ngini = 0.494\nsamples = 27\nv
alue = [20, 25, 0]\nclass = No'),
 Text(3299.478260869565, 543.5999999999999, 'gini = 0.426\nsamples = 9\nvalue
= [9, 4, 0]\nclass = Yes'),
 Text(3493.565217391304, 543.5999999999999, 'X[0] <= 2.55\ngini = 0.451\nsamp
les = 18\nvalue = [11, 21, 0]\nclass = No'),
 Text(3396.5217391304345, 181.19999999999982, 'gini = 0.0\nsamples = 10\nvalu
e = [0, 17, 0]\nclass = No'),
 Text(3590.608695652174, 181.19999999999982, 'gini = 0.391\nsamples = 8\nvalu
e = [11, 4, 0]\nclass = Yes'),
 Text(4075.8260869565215, 906.0, 'X[7] <= 56.5\ngini = 0.149\nsamples = 57\nv
alue = [7, 104, 2]\nclass = No'),
 Text(3881.7391304347825, 543.5999999999999, 'X[0] <= 2.85\ngini = 0.054\nsam
ples = 34\nvalue = [0, 70, 2]\nclass = No'),
 Text(3784.695652173913, 181.19999999999982, 'gini = 0.0\nsamples = 16\nvalue
= [0, 39, 0]\nclass = No'),
 Text(3978.782608695652, 181.19999999999982, 'gini = 0.114\nsamples = 18\nval
ue = [0, 31, 2]\nclass = No'),
 Text(4269.913043478261, 543.5999999999999, 'X[5] <= 179.0\ngini = 0.283\nsam
ples = 23\nvalue = [7, 34, 0]\nclass = No'),
 Text(4172.869565217391, 181.19999999999982, 'gini = 0.465\nsamples = 8\nvalu
e = [7, 12, 0]\nclass = No'),
 Text(4366.95652173913, 181.19999999999982, 'gini = 0.0\nsamples = 15\nvalue
= [0, 22, 0]\nclass = No')]
```



```
Best model:LogisticRegression
```

```
In [ ]:
```