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	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210091	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN	28
210092	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN	28
210093	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN	28
210094	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN	28
210095	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN	28

210096 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210096 entries, 0 to 210095
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210096 non-null  object 
 1   BEN      51039 non-null   float64
 2   CO       86827 non-null   float64
 3   EBE      50962 non-null   float64
 4   NMHC     25756 non-null   float64
 5   NO       208805 non-null  float64
 6   NO_2     208805 non-null  float64
 7   O_3      121574 non-null  float64
 8   PM10     102745 non-null  float64
 9   PM25     48798 non-null   float64
 10  SO_2     86898 non-null   float64
 11  TCH      25756 non-null   float64
 12  TOL      50626 non-null   float64
 13  station  210096 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	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28
6	2015-10-01 01:00:00	0.5	0.3	0.3	0.12	6.0	83.0	1.0	19.0	12.0	3.0	1.29	4.8	28
25	2015-10-01 02:00:00	1.6	0.7	1.3	0.38	81.0	105.0	4.0	36.0	19.0	13.0	1.93	6.9	28
30	2015-10-01 02:00:00	0.4	0.3	0.3	0.11	5.0	72.0	2.0	16.0	10.0	2.0	1.27	7.8	28
49	2015-10-01 03:00:00	2.2	0.8	1.8	0.41	111.0	104.0	4.0	35.0	20.0	14.0	2.05	13.9	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210030	2015-07-31 22:00:00	0.1	0.1	0.1	0.06	1.0	10.0	69.0	10.0	3.0	2.0	1.18	0.2	28
210049	2015-07-31 23:00:00	0.4	0.3	0.1	0.12	3.0	28.0	56.0	15.0	7.0	12.0	1.45	1.2	28
210054	2015-07-31 23:00:00	0.1	0.1	0.1	0.06	1.0	10.0	63.0	5.0	1.0	2.0	1.18	0.2	28
210073	2015-08-01 00:00:00	0.1	0.3	0.1	0.11	2.0	23.0	59.0	5.0	2.0	11.0	1.44	0.6	28
210078	2015-08-01 00:00:00	0.1	0.1	0.1	0.06	1.0	8.0	65.0	7.0	1.0	2.0	1.18	0.4	28

16026 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]:

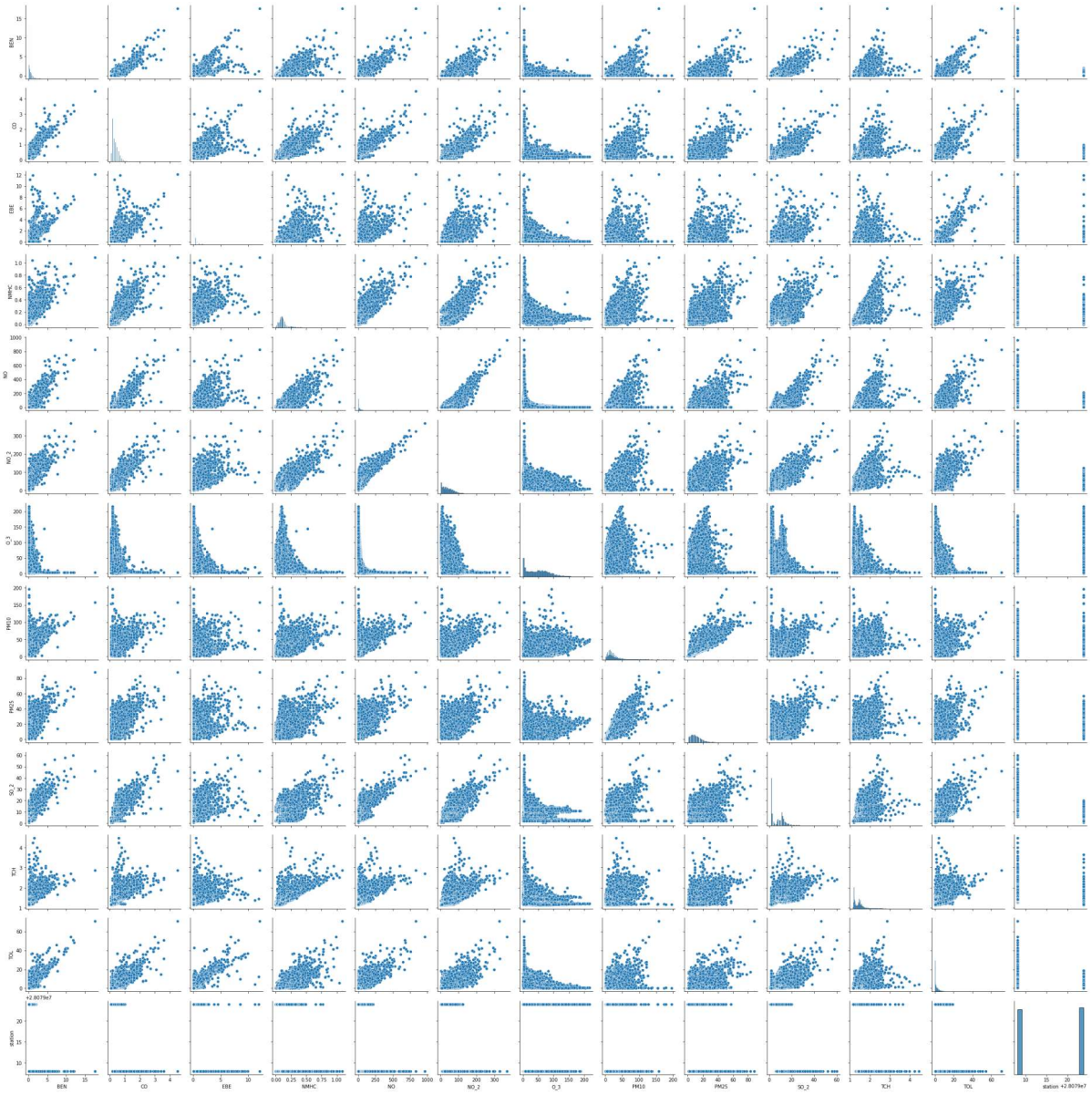
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16026.000000	16026.000000	16026.000000	16026.000000	16026.000000	16026.000000	160
mean	0.504823	0.380594	0.394247	0.123099	23.842256	40.948771	
std	0.716896	0.260805	0.678592	0.092368	51.255660	33.236098	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	
25%	0.100000	0.200000	0.100000	0.070000	1.000000	14.000000	
50%	0.200000	0.300000	0.100000	0.100000	6.000000	35.000000	
75%	0.700000	0.500000	0.400000	0.140000	24.000000	60.000000	
max	17.700001	4.500000	12.100000	1.090000	960.000000	369.000000	2

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)
```

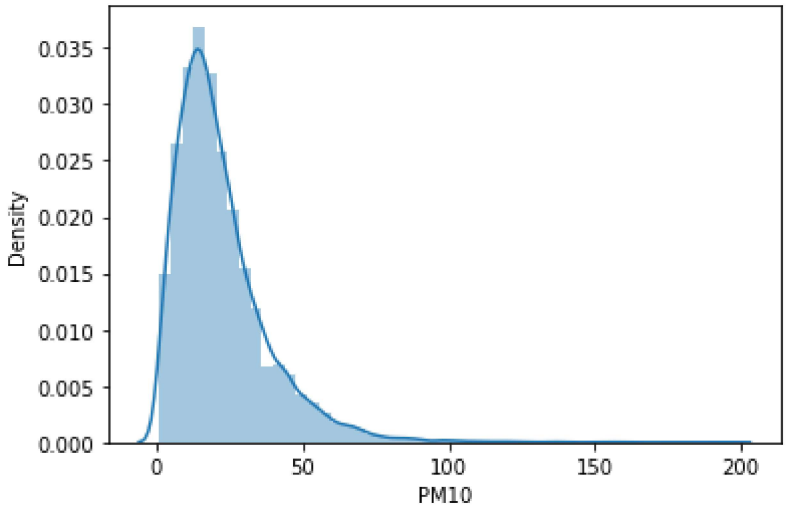
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1a5138711c0>
```



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

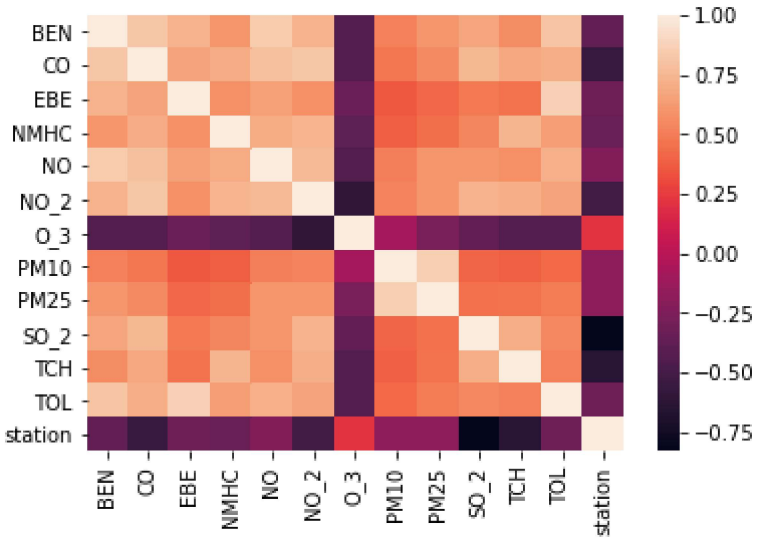
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function 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	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	0	8.3	28
6	2015-10-01 01:00:00	0.5	0.3	0.3	0.12	6.0	83.0	1.0	19.0	12.0	3.0	0	4.8	28
25	2015-10-01 02:00:00	1.6	0.7	1.3	0.38	81.0	105.0	4.0	36.0	19.0	13.0	0	6.9	28
30	2015-10-01 02:00:00	0.4	0.3	0.3	0.11	5.0	72.0	2.0	16.0	10.0	2.0	0	7.8	28
49	2015-10-01 03:00:00	2.2	0.8	1.8	0.41	111.0	104.0	4.0	35.0	20.0	14.0	1	13.9	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210030	2015-07-31 22:00:00	0.1	0.1	0.1	0.06	1.0	10.0	69.0	10.0	3.0	2.0	0	0.2	28
210049	2015-07-31 23:00:00	0.4	0.3	0.1	0.12	3.0	28.0	56.0	15.0	7.0	12.0	0	1.2	28
210054	2015-07-31 23:00:00	0.1	0.1	0.1	0.06	1.0	10.0	63.0	5.0	1.0	2.0	0	0.2	28
210073	2015-08-01 00:00:00	0.1	0.3	0.1	0.11	2.0	23.0	59.0	5.0	2.0	11.0	0	0.6	28
210078	2015-08-01 00:00:00	0.1	0.1	0.1	0.06	1.0	8.0	65.0	7.0	1.0	2.0	0	0.4	28

16026 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]:
```
0.9762895174708819
```

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]:
```
0.9999196786297907
```

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

Out[19]:
```
2.053888893534883e-14
```

LinearRegression

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

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

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

```
52904.227763502706
```

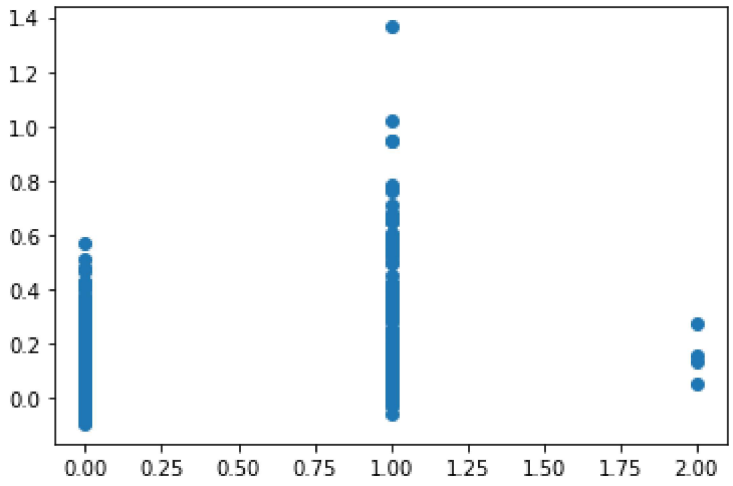
In [22]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[22]:

	Co-efficient
BEN	0.027972
CO	-0.074742
EBE	0.006364
NMHC	0.572787
NO	0.001284
NO_2	-0.001044
O_3	0.000014
PM10	0.000890
PM25	-0.000281
SO_2	-0.000795
TOL	-0.004117
station	-0.001884

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

Out[23]:
```
<matplotlib.collections.PathCollection at 0x1a5238feb50>
```



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

```
0.2556798222158476
```

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]:
```
0.2559853179539364
```

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]:
```
-3.4146404641610673e-06
```

ElasticNet

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

Out[29]:
```
ElasticNet()
```

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

```
[ 0.          0.          0.          0.          0.00127155  0.
  0.          0.          0.          0.          0.         -0.        ]
```

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

```
-0.006090643490251853
```

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

```
[-0.002276    0.06002987  0.02442652 ...  0.06257297  0.14268052
  0.00535329]
```

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

```
0.22078010833943218
```

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

```
Mean Absolytre Error: 0.05221831972592315
```

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

```
Mean Square Error: 0.019057567847766688
```

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

```
Root Mean Square Error: 0.22851328129000106
```

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_
```

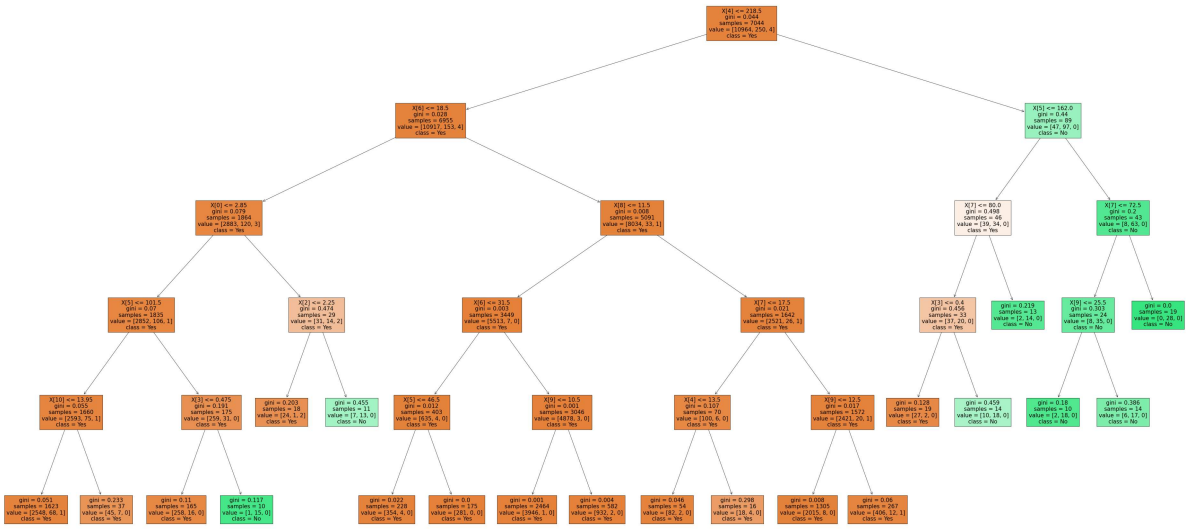
Out[40]:
```
0.9841326439650562
```

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(2773.5882352941176, 1993.2, 'X[4] <= 218.5\ngini = 0.044\nsamples = 7044\nvalue = [10964, 250, 4]\nclass = Yes'),
 Text(1608.3529411764705, 1630.8000000000002, 'X[6] <= 18.5\ngini = 0.028\nsamples = 6955\nvalue = [10917, 153, 4]\nclass = Yes'),
 Text(853.4117647058823, 1268.4, 'X[0] <= 2.85\ngini = 0.079\nsamples = 1864\nvalue = [2883, 120, 3]\nclass = Yes'),
 Text(525.1764705882352, 906.0, 'X[5] <= 101.5\ngini = 0.07\nsamples = 1835\nvalue = [2852, 106, 1]\nclass = Yes'),
 Text(262.5882352941176, 543.5999999999999, 'X[10] <= 13.95\ngini = 0.055\nsamples = 1660\nvalue = [2593, 75, 1]\nclass = Yes'),
 Text(131.2941176470588, 181.19999999999982, 'gini = 0.051\nsamples = 1623\nvalue = [2548, 68, 1]\nclass = Yes'),
 Text(393.88235294117646, 181.19999999999982, 'gini = 0.233\nsamples = 37\nvalue = [45, 7, 0]\nclass = Yes'),
 Text(787.7647058823529, 543.5999999999999, 'X[3] <= 0.475\ngini = 0.191\nsamples = 175\nvalue = [259, 31, 0]\nclass = Yes'),
 Text(656.470588235294, 181.19999999999982, 'gini = 0.11\nsamples = 165\nvalue = [258, 16, 0]\nclass = Yes'),
 Text(919.0588235294117, 181.19999999999982, 'gini = 0.117\nsamples = 10\nvalue = [1, 15, 0]\nclass = No'),
 Text(1181.6470588235293, 906.0, 'X[2] <= 2.25\ngini = 0.474\nsamples = 29\nvalue = [31, 14, 2]\nclass = Yes'),
 Text(1050.3529411764705, 543.5999999999999, 'gini = 0.203\nsamples = 18\nvalue = [24, 1, 2]\nclass = Yes'),
 Text(1312.941176470588, 543.5999999999999, 'gini = 0.455\nsamples = 11\nvalue = [7, 13, 0]\nclass = No'),
 Text(2363.2941176470586, 1268.4, 'X[8] <= 11.5\ngini = 0.008\nsamples = 5091\nvalue = [8034, 33, 1]\nclass = Yes'),
 Text(1838.1176470588234, 906.0, 'X[6] <= 31.5\ngini = 0.003\nsamples = 3449\nvalue = [5513, 7, 0]\nclass = Yes'),
 Text(1575.5294117647059, 543.5999999999999, 'X[5] <= 46.5\ngini = 0.012\nsamples = 403\nvalue = [635, 4, 0]\nclass = Yes'),
 Text(1444.2352941176468, 181.19999999999982, 'gini = 0.022\nsamples = 228\nvalue = [354, 4, 0]\nclass = Yes'),
 Text(1706.8235294117646, 181.19999999999982, 'gini = 0.0\nsamples = 175\nvalue = [281, 0, 0]\nclass = Yes'),
 Text(2100.705882352941, 543.5999999999999, 'X[9] <= 10.5\ngini = 0.001\nsamples = 3046\nvalue = [4878, 3, 0]\nclass = Yes'),
 Text(1969.4117647058822, 181.19999999999982, 'gini = 0.001\nsamples = 2464\nvalue = [3946, 1, 0]\nclass = Yes'),
 Text(2232.0, 181.19999999999982, 'gini = 0.004\nsamples = 582\nvalue = [932, 2, 0]\nclass = Yes'),
 Text(2888.4705882352937, 906.0, 'X[7] <= 17.5\ngini = 0.021\nsamples = 1642\nvalue = [2521, 26, 1]\nclass = Yes'),
 Text(2625.882352941176, 543.5999999999999, 'X[4] <= 13.5\ngini = 0.107\nsamples = 70\nvalue = [100, 6, 0]\nclass = Yes'),
 Text(2494.5882352941176, 181.19999999999982, 'gini = 0.046\nsamples = 54\nvalue = [82, 2, 0]\nclass = Yes'),
 Text(2757.176470588235, 181.19999999999982, 'gini = 0.298\nsamples = 16\nvalue = [18, 4, 0]\nclass = Yes'),
 Text(3151.0588235294117, 543.5999999999999, 'X[9] <= 12.5\ngini = 0.017\nsamples = 1572\nvalue = [2421, 20, 1]\nclass = Yes'),
 Text(3019.7647058823527, 181.19999999999982, 'gini = 0.008\nsamples = 1305\nvalue = [2015, 8, 0]\nclass = Yes'),
 Text(3282.3529411764703, 181.19999999999982, 'gini = 0.06\nsamples = 267\nvalue = [406, 12, 1]\nclass = Yes'),
 Text(3938.8235294117644, 1630.8000000000002, 'X[5] <= 162.0\ngini = 0.44\nsa
```

```
mples = 89\nvalue = [47, 97, 0]\nclass = No'),
 Text(3676.235294117647, 1268.4, 'X[7] <= 80.0\ngini = 0.498\nsamples = 46\nv
alue = [39, 34, 0]\nclass = Yes'),
 Text(3544.941176470588, 906.0, 'X[3] <= 0.4\ngini = 0.456\nsamples = 33\nval
ue = [37, 20, 0]\nclass = Yes'),
 Text(3413.6470588235293, 543.5999999999999, 'gini = 0.128\nsamples = 19\nval
ue = [27, 2, 0]\nclass = Yes'),
 Text(3676.235294117647, 543.5999999999999, 'gini = 0.459\nsamples = 14\nvalu
e = [10, 18, 0]\nclass = No'),
 Text(3807.5294117647054, 906.0, 'gini = 0.219\nsamples = 13\nvalue = [2, 14,
0]\nclass = No'),
 Text(4201.411764705882, 1268.4, 'X[7] <= 72.5\ngini = 0.2\nsamples = 43\nval
ue = [8, 63, 0]\nclass = No'),
 Text(4070.117647058823, 906.0, 'X[9] <= 25.5\ngini = 0.303\nsamples = 24\nva
lue = [8, 35, 0]\nclass = No'),
 Text(3938.8235294117644, 543.5999999999999, 'gini = 0.18\nsamples = 10\nvalu
e = [2, 18, 0]\nclass = No'),
 Text(4201.411764705882, 543.5999999999999, 'gini = 0.386\nsamples = 14\nvalu
e = [6, 17, 0]\nclass = No'),
 Text(4332.7058823529405, 906.0, 'gini = 0.0\nsamples = 19\nvalue = [0, 28,
0]\nclass = No')]
```



Best model:LogisticRegression

In [ ]: