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	2013-11-01 01:00:00	NaN	0.6	NaN	NaN	135.0	74.0	NaN	NaN	NaN	7.0	NaN	NaN
1	2013-11-01 01:00:00	1.5	0.5	1.3	NaN	71.0	83.0	2.0	23.0	16.0	12.0	NaN	8.3
2	2013-11-01 01:00:00	3.9	NaN	2.8	NaN	49.0	70.0	NaN	NaN	NaN	NaN	NaN	9.0
3	2013-11-01 01:00:00	NaN	0.5	NaN	NaN	82.0	87.0	3.0	NaN	NaN	NaN	NaN	NaN
4	2013-11-01 01:00:00	NaN	NaN	NaN	NaN	242.0	111.0	2.0	NaN	NaN	12.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	8.0	39.0	52.0	NaN	NaN	NaN	NaN	NaN
209876	2013-03-01 00:00:00	NaN	0.4	NaN	NaN	1.0	11.0	NaN	6.0	NaN	2.0	NaN	NaN
209877	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	4.0	75.0	NaN	NaN	NaN	NaN	NaN
209878	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	11.0	52.0	NaN	NaN	NaN	NaN	NaN
209879	2013-03-01 00:00:00	NaN	NaN	NaN	NaN	1.0	10.0	75.0	3.0	NaN	NaN	NaN	NaN

209880 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      50462 non-null   float64
 2   CO       87018 non-null   float64
 3   EBE      50463 non-null   float64
 4   NMHC     25935 non-null   float64
 5   NO       209108 non-null  float64
 6   NO_2     209108 non-null  float64
 7   O_3      121858 non-null  float64
 8   PM10     104339 non-null  float64
 9   PM25     51980 non-null   float64
 10  SO_2     86970 non-null   float64
 11  TCH      25935 non-null   float64
 12  TOL      50317 non-null   float64
 13  station  209880 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	st
17286	2013-08-01 01:00:00	0.4	0.2	0.8	0.28	1.0	24.0	79.0	35.0	8.0	3.0	1.49	1.3	2807
17310	2013-08-01 02:00:00	0.5	0.2	0.9	0.28	1.0	16.0	93.0	60.0	18.0	3.0	1.61	4.0	2807
17334	2013-08-01 03:00:00	0.5	0.2	1.1	0.29	1.0	14.0	90.0	38.0	12.0	3.0	1.71	2.8	2807
17358	2013-08-01 04:00:00	0.6	0.2	1.2	0.26	1.0	12.0	84.0	30.0	8.0	3.0	1.44	2.8	2807
17382	2013-08-01 05:00:00	0.3	0.2	0.8	0.25	1.0	15.0	72.0	25.0	7.0	3.0	1.40	1.7	2807
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209622	2013-02-28 14:00:00	1.1	0.3	0.3	0.27	3.0	17.0	64.0	5.0	5.0	2.0	1.41	0.9	2807
209646	2013-02-28 15:00:00	1.3	0.4	0.3	0.27	2.0	16.0	66.0	6.0	5.0	1.0	1.40	0.9	2807
209670	2013-02-28 16:00:00	1.1	0.3	0.3	0.27	1.0	17.0	65.0	5.0	4.0	1.0	1.40	0.7	2807
209694	2013-02-28 17:00:00	1.0	0.3	0.4	0.27	1.0	18.0	64.0	5.0	5.0	1.0	1.39	0.7	2807
209718	2013-02-28 18:00:00	1.0	0.3	0.4	0.27	1.0	22.0	62.0	6.0	6.0	1.0	1.39	0.7	2807

7315 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]:

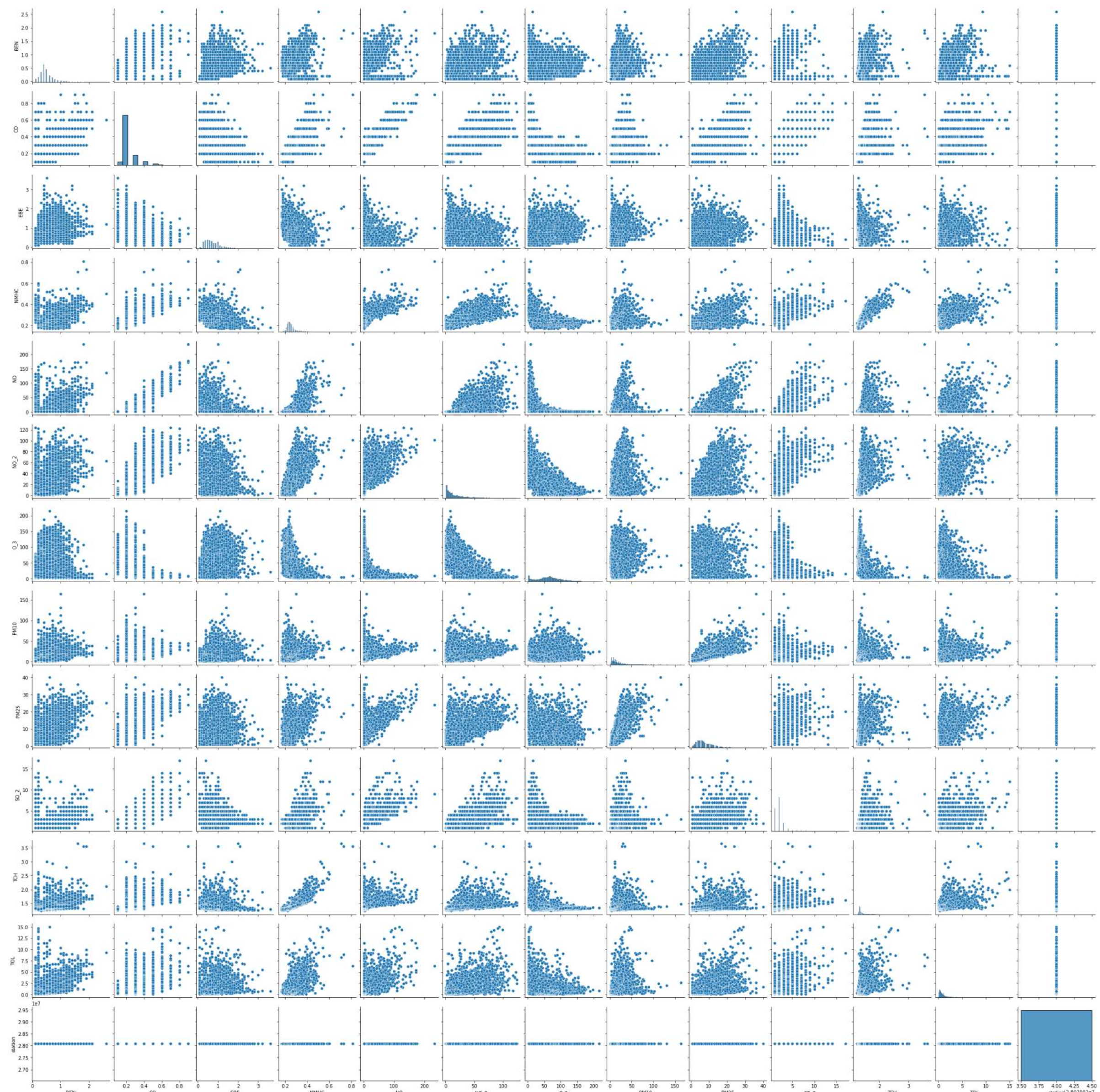
	BEN	CO	EBE	NMHC	NO	NO_2	O_
count	7315.000000	7315.000000	7315.000000	7315.000000	7315.000000	7315.000000	7315.00000
mean	0.501928	0.236008	0.753247	0.255133	7.486808	19.742584	62.6539
std	0.275264	0.092865	0.386968	0.046754	18.386879	20.984539	35.8224
min	0.100000	0.100000	0.100000	0.170000	1.000000	1.000000	2.0000
25%	0.300000	0.200000	0.500000	0.230000	1.000000	5.000000	38.0000
50%	0.400000	0.200000	0.700000	0.240000	1.000000	12.000000	63.0000
75%	0.600000	0.200000	1.000000	0.270000	3.000000	27.000000	85.0000
max	2.600000	0.900000	3.600000	0.810000	234.000000	124.000000	215.0000

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)
```

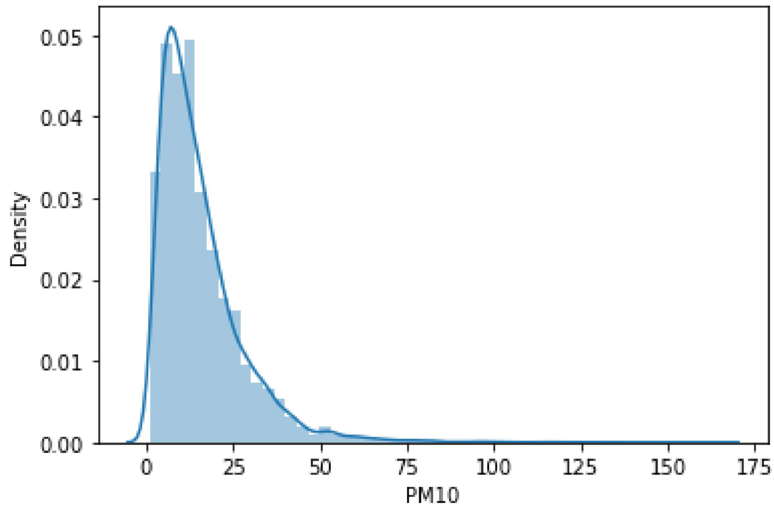
Out[8]: <seaborn.axisgrid.PairGrid at 0x229665aa9a0>



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

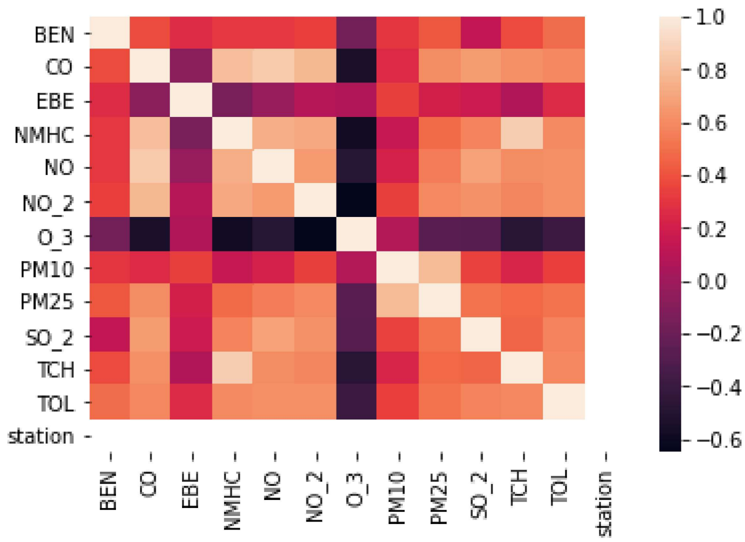
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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['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	st
17286	2013-08-01 01:00:00	0.4	0.2	0.8	0.28	1.0	24.0	79.0	35.0	8.0	3.0	0	1.3	2807
17310	2013-08-01 02:00:00	0.5	0.2	0.9	0.28	1.0	16.0	93.0	60.0	18.0	3.0	0	4.0	2807
17334	2013-08-01 03:00:00	0.5	0.2	1.1	0.29	1.0	14.0	90.0	38.0	12.0	3.0	0	2.8	2807
17358	2013-08-01 04:00:00	0.6	0.2	1.2	0.26	1.0	12.0	84.0	30.0	8.0	3.0	0	2.8	2807
17382	2013-08-01 05:00:00	0.3	0.2	0.8	0.25	1.0	15.0	72.0	25.0	7.0	3.0	0	1.7	2807
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209622	2013-02-28 14:00:00	1.1	0.3	0.3	0.27	3.0	17.0	64.0	5.0	5.0	2.0	0	0.9	2807
209646	2013-02-28 15:00:00	1.3	0.4	0.3	0.27	2.0	16.0	66.0	6.0	5.0	1.0	0	0.9	2807
209670	2013-02-28 16:00:00	1.1	0.3	0.3	0.27	1.0	17.0	65.0	5.0	4.0	1.0	0	0.7	2807
209694	2013-02-28 17:00:00	1.0	0.3	0.4	0.27	1.0	18.0	64.0	5.0	5.0	1.0	0	0.7	2807
209718	2013-02-28 18:00:00	1.0	0.3	0.4	0.27	1.0	22.0	62.0	6.0	6.0	1.0	0	0.7	2807

7315 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]:
```
0.9931662870159453
```

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

In [17]:
```
o=[[1,2,3,4,5,6,7,8,9,10,11,12]]
prediction=logr.predict(o)
print(prediction)
```
```
[0]
```

In [18]:
```
logr.classes_
```
Out[18]:
```
array([0, 1, 2])
```

In [19]:
```
logr.predict_proba(o)[0][0]
```
Out[19]:
```
0.999965076528319
```

In [20]:
```
logr.predict_proba(o)[0][1]
```
Out[20]:
```
3.492292250767436e-05
```

LinearRegression

In [21]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```
Out[21]:
```
LinearRegression()
```

In [22]:
```
print(lr.intercept_)
```
```
-0.3781578699604019
```

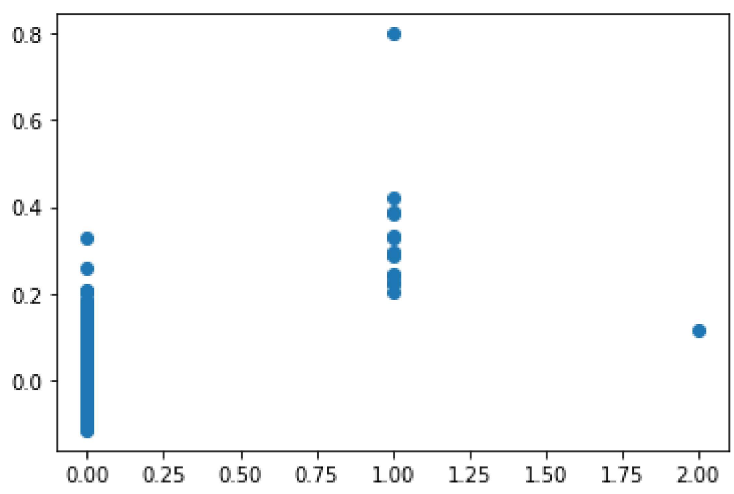
In [23]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[23]:

	Co-efficient
BEN	-0.004576
CO	-0.428683
EBE	0.035687
NMHC	1.830873
NO	0.000592
NO_2	-0.000606
O_3	0.000235
PM10	0.000776
PM25	-0.001487
SO_2	-0.006708
TOL	0.002402
station	0.000000

In [24]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[24]: <matplotlib.collections.PathCollection at 0x22973e395b0>



In [25]:
```
print(lr.score(x_test,y_test))
```

```
0.2173651816259501
```

Ridge,Lasso

In [26]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```
Out[26]:
```
Ridge(alpha=10)
```
In [27]:
```
rr.score(x_test,y_test)
```
Out[27]:
```
0.12953059882910944
```
In [28]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[28]:
```
Lasso(alpha=10)
```
In [29]:
```
la.score(x_test,y_test)
```
Out[29]:
```
-0.000533938669069034
```

ElasticNet

In [30]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[30]:
```
ElasticNet()
```
In [31]:
```
print(en.coef_)
```
```
[ 0.  0.  0.  0.  0.  0. -0.  0.  0.  0.  0.  0.]
```
In [32]:
```
print(en.intercept_)
```
```
0.009375
```
In [33]:
```
print(en.predict(x_train))
```
```
[0.009375 0.009375 0.009375 ... 0.009375 0.009375 0.009375]
```
In [34]:
```
print(en.score(x_train,y_train))
```
```
0.0
```
In [35]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```
```
Mean Absolytre Error: 0.03533414583388424
```
In [36]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```
```
Mean Square Error: 0.006376377698425174
```

In [37]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 0.18797379028440173
```

RandomForest

In [38]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[38]:
```
RandomForestClassifier()
```

In [39]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [40]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[40]:
```
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')
```

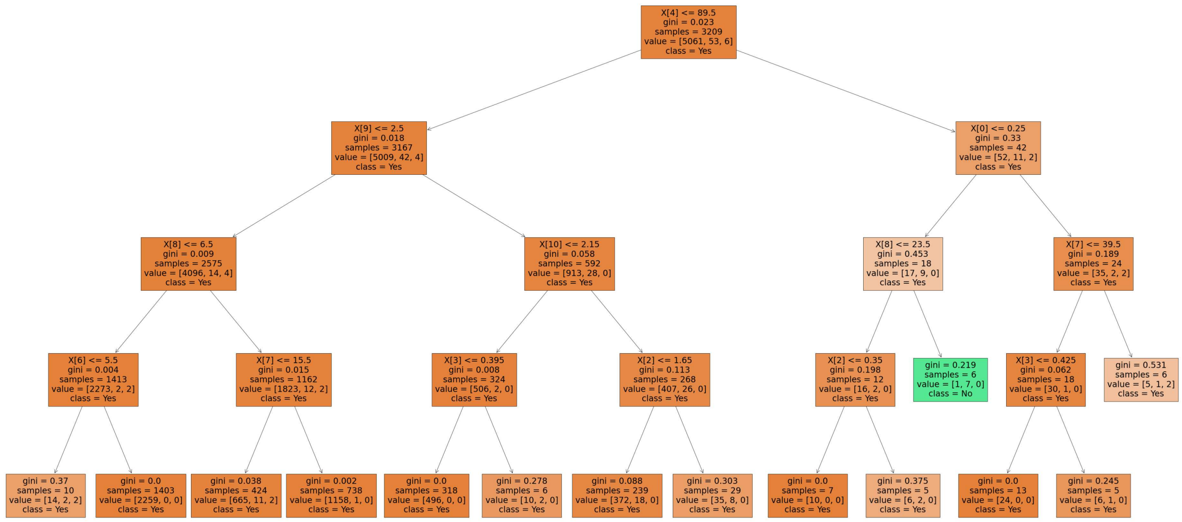
In [41]:
```
grid_search.best_score_
```

Out[41]:
```
0.99296875
```

In [42]:
```
rfc_best=grid_search.best_estimator_
```

In [43]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],class_names=['Yes','No','Yes','No'],filled=Tr
```

Out[43]:
```
[Text(2589.12, 1956.96, 'X[4] <= 89.5\ngini = 0.023\nsamples = 3209\nvalue =
[5061, 53, 6]\nclass = Yes'),
 Text(1428.48, 1522.0800000000002, 'X[9] <= 2.5\ngini = 0.018\nsamples = 3167
\nvalue = [5009, 42, 4]\nclass = Yes'),
 Text(714.24, 1087.2, 'X[8] <= 6.5\ngini = 0.009\nsamples = 2575\nvalue = [40
96, 14, 4]\nclass = Yes'),
 Text(357.12, 652.3200000000002, 'X[6] <= 5.5\ngini = 0.004\nsamples = 1413\n
value = [2273, 2, 2]\nclass = Yes'),
 Text(178.56, 217.44000000000005, 'gini = 0.37\nsamples = 10\nvalue = [14, 2,
2]\nclass = Yes'),
 Text(535.6800000000001, 217.44000000000005, 'gini = 0.0\nsamples = 1403\nval
ue = [2259, 0, 0]\nclass = Yes'),
 Text(1071.3600000000001, 652.3200000000002, 'X[7] <= 15.5\ngini = 0.015\nsam
ples = 1162\nvalue = [1823, 12, 2]\nclass = Yes'),
 Text(892.8, 217.44000000000005, 'gini = 0.038\nsamples = 424\nvalue = [665,
11, 2]\nclass = Yes'),
 Text(1249.92, 217.44000000000005, 'gini = 0.002\nsamples = 738\nvalue = [115
8, 1, 0]\nclass = Yes'),
 Text(2142.7200000000003, 1087.2, 'X[10] <= 2.15\ngini = 0.058\nsamples = 592
\nvalue = [913, 28, 0]\nclass = Yes'),
 Text(1785.6, 652.3200000000002, 'X[3] <= 0.395\ngini = 0.008\nsamples = 324
\nvalue = [506, 2, 0]\nclass = Yes'),
 Text(1607.04, 217.44000000000005, 'gini = 0.0\nsamples = 318\nvalue = [496,
0, 0]\nclass = Yes'),
 Text(1964.16, 217.44000000000005, 'gini = 0.278\nsamples = 6\nvalue = [10,
2, 0]\nclass = Yes'),
 Text(2499.84, 652.3200000000002, 'X[2] <= 1.65\ngini = 0.113\nsamples = 268
\nvalue = [407, 26, 0]\nclass = Yes'),
 Text(2321.28, 217.44000000000005, 'gini = 0.088\nsamples = 239\nvalue = [37
2, 18, 0]\nclass = Yes'),
 Text(2678.4, 217.44000000000005, 'gini = 0.303\nsamples = 29\nvalue = [35,
8, 0]\nclass = Yes'),
 Text(3749.76, 1522.0800000000002, 'X[0] <= 0.25\ngini = 0.33\nsamples = 42\n
value = [52, 11, 2]\nclass = Yes'),
 Text(3392.64, 1087.2, 'X[8] <= 23.5\ngini = 0.453\nsamples = 18\nvalue = [1
7, 9, 0]\nclass = Yes'),
 Text(3214.08, 652.3200000000002, 'X[2] <= 0.35\ngini = 0.198\nsamples = 12\n
value = [16, 2, 0]\nclass = Yes'),
 Text(3035.52, 217.44000000000005, 'gini = 0.0\nsamples = 7\nvalue = [10, 0,
0]\nclass = Yes'),
 Text(3392.64, 217.44000000000005, 'gini = 0.375\nsamples = 5\nvalue = [6, 2,
0]\nclass = Yes'),
 Text(3571.2, 652.3200000000002, 'gini = 0.219\nsamples = 6\nvalue = [1, 7,
0]\nclass = No'),
 Text(4106.88, 1087.2, 'X[7] <= 39.5\ngini = 0.189\nsamples = 24\nvalue = [3
5, 2, 2]\nclass = Yes'),
 Text(3928.32, 652.3200000000002, 'X[3] <= 0.425\ngini = 0.062\nsamples = 18
\nvalue = [30, 1, 0]\nclass = Yes'),
 Text(3749.76, 217.44000000000005, 'gini = 0.0\nsamples = 13\nvalue = [24, 0,
0]\nclass = Yes'),
 Text(4106.88, 217.44000000000005, 'gini = 0.245\nsamples = 5\nvalue = [6, 1,
0]\nclass = Yes'),
 Text(4285.440000000005, 652.3200000000002, 'gini = 0.531\nsamples = 6\nvalu
e = [5, 1, 2]\nclass = Yes')]
```



Best model:LogisticRegression

In [ ]: