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     2014-06-01   NaN  0.2  NaN   NaN   3.0  10.0   NaN   NaN   NaN   3.0   NaN  NaN  28
             01:00:00
     1     2014-06-01   0.2  0.2  0.1  0.11   3.0  17.0  68.0  10.0   5.0   5.0  1.36  1.3  28
             01:00:00
     2     2014-06-01   0.3  NaN  0.1   NaN   2.0   6.0   NaN   NaN   NaN   NaN   NaN  1.1  28
             01:00:00
     3     2014-06-01   NaN  0.2  NaN   NaN   1.0   6.0  79.0   NaN   NaN   NaN   NaN  NaN  28
             01:00:00
     4     2014-06-01   NaN  NaN  NaN   NaN   1.0   6.0  75.0   NaN   NaN   4.0   NaN  NaN  28
             01:00:00
   ...            ...   ...  ...  ...   ...   ...   ...   ...   ...   ...   ...   ...  ...
210019     2014-09-01   NaN  0.5  NaN   NaN  20.0  84.0  29.0   NaN   NaN   NaN   NaN  NaN  28
             00:00:00
210020     2014-09-01   NaN  0.3  NaN   NaN   1.0  22.0   NaN  15.0   NaN   6.0   NaN  NaN  28
             00:00:00
210021     2014-09-01   NaN  NaN  NaN   NaN   1.0  13.0  70.0   NaN   NaN   NaN   NaN  NaN  28
             00:00:00
210022     2014-09-01   NaN  NaN  NaN   NaN   3.0  38.0  42.0   NaN   NaN   NaN   NaN  NaN  28
             00:00:00
210023     2014-09-01   NaN  NaN  NaN   NaN   1.0  26.0  65.0  11.0   NaN   NaN   NaN  NaN  28
             00:00:00

210024 rows × 14 columns
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210024 entries, 0 to 210023
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210024 non-null  object 
 1   BEN      46703 non-null   float64
 2   CO       87023 non-null   float64
 3   EBE      46722 non-null   float64
 4   NMHC     25021 non-null   float64
 5   NO       209154 non-null  float64
 6   NO_2     209154 non-null  float64
 7   O_3      121681 non-null  float64
 8   PM10     104311 non-null  float64
 9   PM25     51954 non-null   float64
 10  SO_2     87141 non-null   float64
 11  TCH      25021 non-null   float64
 12  TOL      46570 non-null   float64
 13  station  210024 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	s
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	280
6	2014-06-01 01:00:00	0.1	0.2	0.1	0.23	1.0	5.0	80.0	4.0	3.0	2.0	1.21	0.1	280
25	2014-06-01 02:00:00	0.2	0.2	0.1	0.11	4.0	21.0	63.0	9.0	6.0	5.0	1.36	0.8	280
30	2014-06-01 02:00:00	0.2	0.2	0.1	0.23	1.0	4.0	88.0	7.0	5.0	2.0	1.21	0.1	280
49	2014-06-01 03:00:00	0.1	0.2	0.1	0.11	4.0	18.0	66.0	9.0	7.0	6.0	1.36	0.9	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209958	2014-08-31 22:00:00	0.2	0.2	0.1	0.22	1.0	28.0	96.0	61.0	15.0	3.0	1.28	0.1	280
209977	2014-08-31 23:00:00	1.1	0.7	0.7	0.19	36.0	118.0	23.0	60.0	25.0	9.0	1.27	6.5	280
209982	2014-08-31 23:00:00	0.2	0.2	0.1	0.21	1.0	17.0	90.0	28.0	14.0	3.0	1.27	0.2	280
210001	2014-09-01 00:00:00	0.6	0.4	0.4	0.12	6.0	63.0	41.0	26.0	15.0	8.0	1.19	4.1	280
210006	2014-09-01 00:00:00	0.2	0.2	0.1	0.23	1.0	30.0	69.0	18.0	13.0	3.0	1.30	0.1	280

13946 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]:

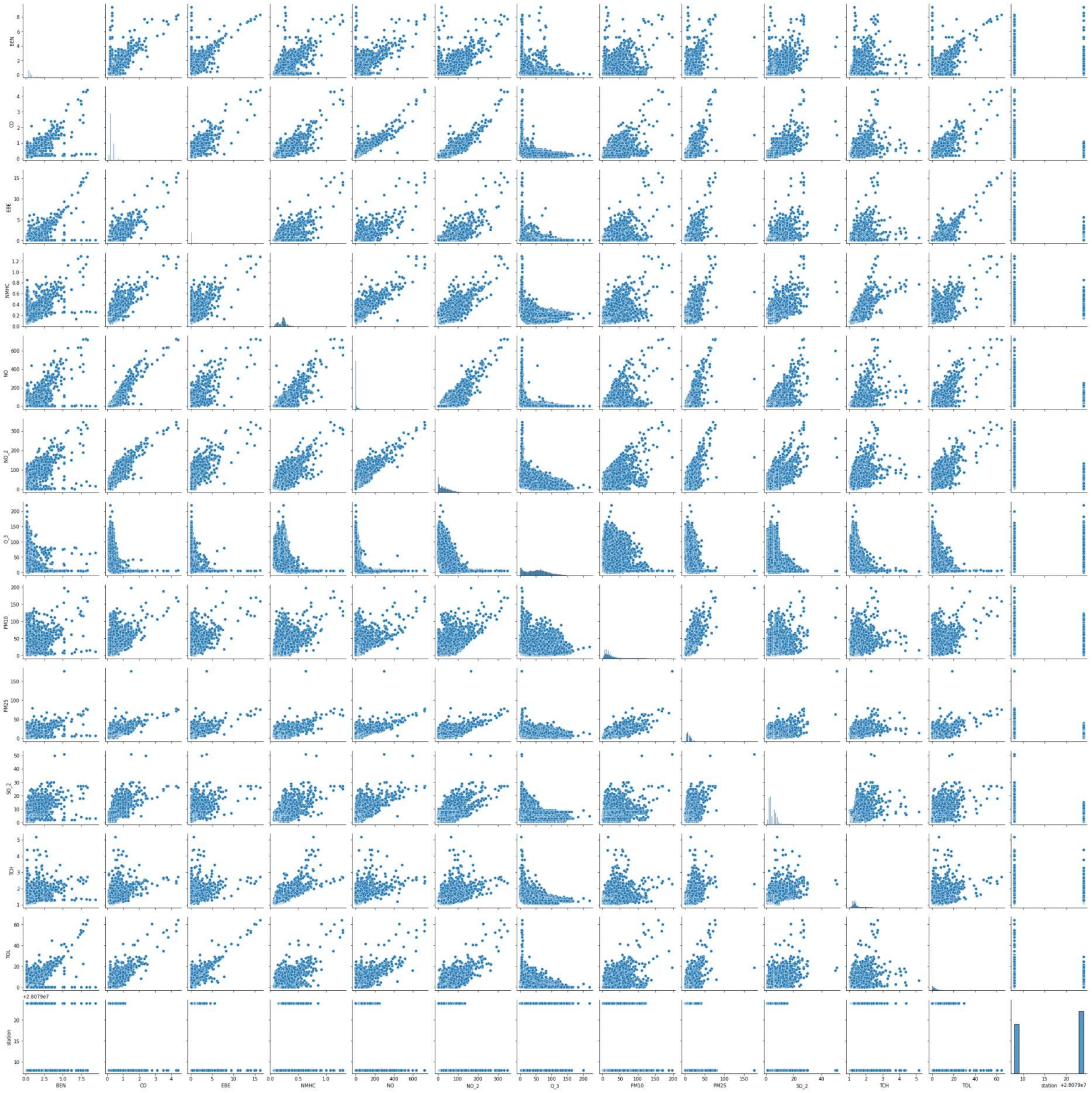
	BEN	CO	EBE	NMHC	NO	NO_2	
count	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000	13946.000000	139
mean	0.375921	0.314793	0.306016	0.222302	17.589129	34.240929	
std	0.555093	0.207375	0.635475	0.082403	39.432216	30.654229	
min	0.100000	0.100000	0.100000	0.060000	1.000000	1.000000	
25%	0.100000	0.200000	0.100000	0.160000	1.000000	10.000000	
50%	0.200000	0.300000	0.100000	0.230000	4.000000	27.000000	
75%	0.400000	0.400000	0.300000	0.260000	18.000000	51.000000	
max	9.400000	4.400000	16.200001	1.290000	725.000000	346.000000	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)
```

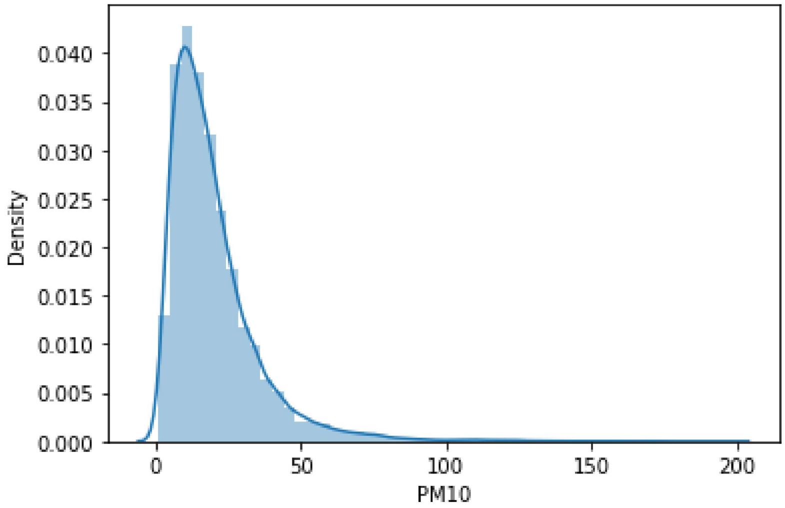
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x288c9a79d60>
```



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

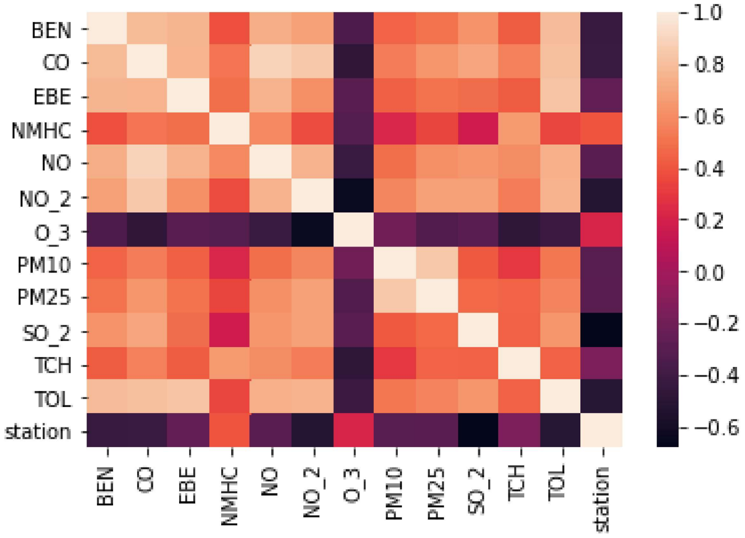
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function 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	s
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	0	1.3	280
6	2014-06-01 01:00:00	0.1	0.2	0.1	0.23	1.0	5.0	80.0	4.0	3.0	2.0	0	0.1	280
25	2014-06-01 02:00:00	0.2	0.2	0.1	0.11	4.0	21.0	63.0	9.0	6.0	5.0	0	0.8	280
30	2014-06-01 02:00:00	0.2	0.2	0.1	0.23	1.0	4.0	88.0	7.0	5.0	2.0	0	0.1	280
49	2014-06-01 03:00:00	0.1	0.2	0.1	0.11	4.0	18.0	66.0	9.0	7.0	6.0	0	0.9	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209958	2014-08-31 22:00:00	0.2	0.2	0.1	0.22	1.0	28.0	96.0	61.0	15.0	3.0	0	0.1	280
209977	2014-08-31 23:00:00	1.1	0.7	0.7	0.19	36.0	118.0	23.0	60.0	25.0	9.0	0	6.5	280
209982	2014-08-31 23:00:00	0.2	0.2	0.1	0.21	1.0	17.0	90.0	28.0	14.0	3.0	0	0.2	280
210001	2014-09-01 00:00:00	0.6	0.4	0.4	0.12	6.0	63.0	41.0	26.0	15.0	8.0	0	4.1	280
210006	2014-09-01 00:00:00	0.2	0.2	0.1	0.23	1.0	30.0	69.0	18.0	13.0	3.0	0	0.1	280

13946 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]:

```
0.9861376673040153
```

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]:

```
0.9999999999997202
```

In [19]:

```
logr.predict_proba(o)[0][1]
```

Out[19]:

```
7.704648678920763e-15
```

LinearRegression

In [20]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[20]:

```
LinearRegression()
```

In [21]:

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

```
223968.26402332023
```

In [22]:

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

Out[22]:

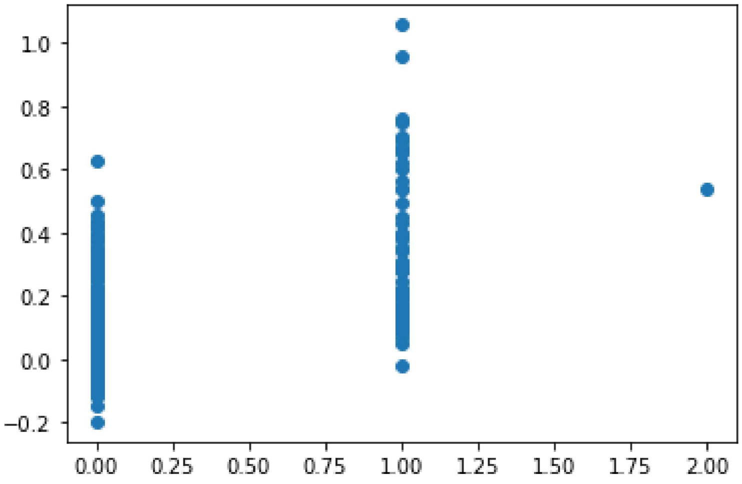
	Co-efficient
BEN	-0.015361
CO	-0.207433
EBE	0.021501
NMHC	0.892985
NO	0.001524
NO_2	-0.000524
O_3	0.000280
PM10	-0.000286
PM25	0.002470
SO_2	-0.004989
TOL	-0.004160
station	-0.007976

In [23]:

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

Out[23]:

```
<matplotlib.collections.PathCollection at 0x288d959f340>
```



In [24]:

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

```
0.322971710303267
```

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]:

```
0.3053442466079368
```

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]:

```
-1.3761787409638515e-05
```

ElasticNet

In [29]:

```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[29]:

```
ElasticNet()
```

In [30]:

```
print(en.coef_)
```

```
[0.         0.         0.         0.         0.00111633 0.
 0.         0.         0.         0.         0.         0.        ]
```

In [31]:

```
print(en.intercept_)
```

```
-0.005319777935996429
```

In [32]:

```
print(en.predict(x_train))
```

```
[-0.00308712  0.00026186  0.0270537  ...  0.01254145  0.00584349
 -0.00420345]
```

In [33]:

```
print(en.score(x_train,y_train))
```

```
0.20531499203311165
```

In [34]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 0.03859831113532434
```

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

```
Mean Square Error: 0.009736006781566388
```

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

```
Root Mean Square Error: 0.19646452894943742
```

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')
```

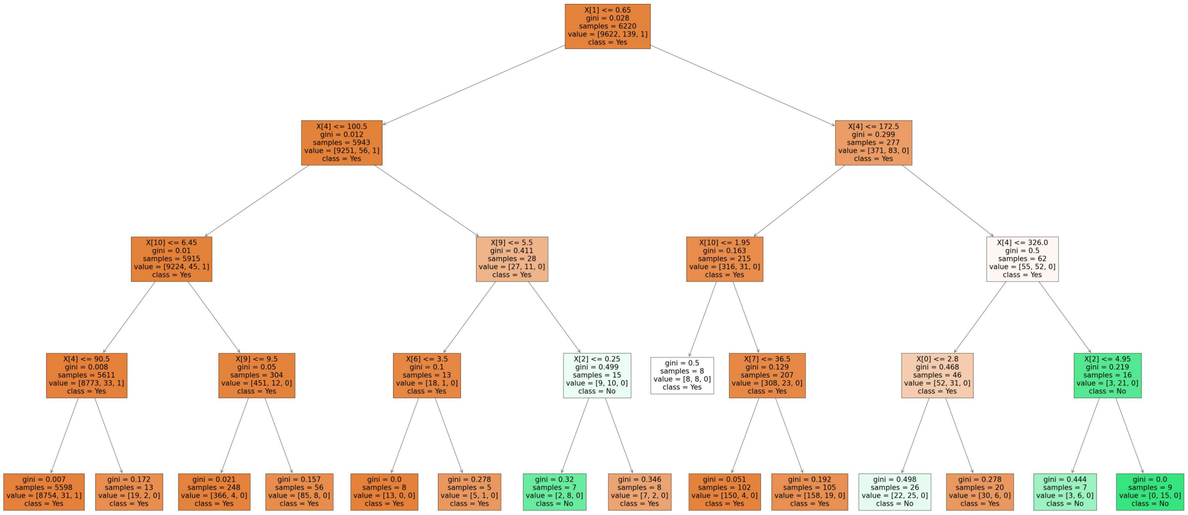
In [40]:
```
grid_search.best_score_
```

Out[40]:
```
0.9905757017004713
```

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(2271.8571428571427, 1956.96, 'X[1] <= 0.65\ngini = 0.028\nsamples = 622
         0\nvalue = [9622, 139, 1]\nclass = Yes'),
          Text(1275.4285714285713, 1522.0800000000002, 'X[4] <= 100.5\ngini = 0.012\ns
         amples = 5943\nvalue = [9251, 56, 1]\nclass = Yes'),
          Text(637.7142857142857, 1087.2, 'X[10] <= 6.45\ngini = 0.01\nsamples = 5915
         \nvalue = [9224, 45, 1]\nclass = Yes'),
          Text(318.85714285714283, 652.3200000000002, 'X[4] <= 90.5\ngini = 0.008\nsam
         ples = 5611\nvalue = [8773, 33, 1]\nclass = Yes'),
          Text(159.42857142857142, 217.44000000000005, 'gini = 0.007\nsamples = 5598\n
         value = [8754, 31, 1]\nclass = Yes'),
          Text(478.2857142857142, 217.44000000000005, 'gini = 0.172\nsamples = 13\nval
         ue = [19, 2, 0]\nclass = Yes'),
          Text(956.5714285714284, 652.3200000000002, 'X[9] <= 9.5\ngini = 0.05\nsample
         s = 304\nvalue = [451, 12, 0]\nclass = Yes'),
          Text(797.1428571428571, 217.44000000000005, 'gini = 0.021\nsamples = 248\nva
         lue = [366, 4, 0]\nclass = Yes'),
          Text(1116.0, 217.44000000000005, 'gini = 0.157\nsamples = 56\nvalue = [85,
         8, 0]\nclass = Yes'),
          Text(1913.1428571428569, 1087.2, 'X[9] <= 5.5\ngini = 0.411\nsamples = 28\nv
         alue = [27, 11, 0]\nclass = Yes'),
          Text(1594.2857142857142, 652.3200000000002, 'X[6] <= 3.5\ngini = 0.1\nsample
         s = 13\nvalue = [18, 1, 0]\nclass = Yes'),
          Text(1434.8571428571427, 217.44000000000005, 'gini = 0.0\nsamples = 8\nvalue
         = [13, 0, 0]\nclass = Yes'),
          Text(1753.7142857142856, 217.44000000000005, 'gini = 0.278\nsamples = 5\nval
         ue = [5, 1, 0]\nclass = Yes'),
          Text(2232.0, 652.3200000000002, 'X[2] <= 0.25\ngini = 0.499\nsamples = 15\nv
         alue = [9, 10, 0]\nclass = No'),
          Text(2072.5714285714284, 217.44000000000005, 'gini = 0.32\nsamples = 7\nvalu
         e = [2, 8, 0]\nclass = No'),
          Text(2391.428571428571, 217.44000000000005, 'gini = 0.346\nsamples = 8\nvalu
         e = [7, 2, 0]\nclass = Yes'),
          Text(3268.285714285714, 1522.0800000000002, 'X[4] <= 172.5\ngini = 0.299\nsa
         mples = 277\nvalue = [371, 83, 0]\nclass = Yes'),
          Text(2710.285714285714, 1087.2, 'X[10] <= 1.95\ngini = 0.163\nsamples = 215
         \nvalue = [316, 31, 0]\nclass = Yes'),
          Text(2550.8571428571427, 652.3200000000002, 'gini = 0.5\nsamples = 8\nvalue
         = [8, 8, 0]\nclass = Yes'),
          Text(2869.7142857142853, 652.3200000000002, 'X[7] <= 36.5\ngini = 0.129\nsam
         ples = 207\nvalue = [308, 23, 0]\nclass = Yes'),
          Text(2710.285714285714, 217.44000000000005, 'gini = 0.051\nsamples = 102\nva
         lue = [150, 4, 0]\nclass = Yes'),
          Text(3029.142857142857, 217.44000000000005, 'gini = 0.192\nsamples = 105\nva
         lue = [158, 19, 0]\nclass = Yes'),
          Text(3826.2857142857138, 1087.2, 'X[4] <= 326.0\ngini = 0.5\nsamples = 62\nv
         alue = [55, 52, 0]\nclass = Yes'),
          Text(3507.428571428571, 652.3200000000002, 'X[0] <= 2.8\ngini = 0.468\nsampl
         es = 46\nvalue = [52, 31, 0]\nclass = Yes'),
          Text(3347.9999999999995, 217.44000000000005, 'gini = 0.498\nsamples = 26\nva
         lue = [22, 25, 0]\nclass = No'),
          Text(3666.8571428571427, 217.44000000000005, 'gini = 0.278\nsamples = 20\nva
         lue = [30, 6, 0]\nclass = Yes'),
          Text(4145.142857142857, 652.3200000000002, 'X[2] <= 4.95\ngini = 0.219\nsamp
         les = 16\nvalue = [3, 21, 0]\nclass = No'),
          Text(3985.7142857142853, 217.44000000000005, 'gini = 0.444\nsamples = 7\nval
         ue = [3, 6, 0]\nclass = No'),
 Text(4304.571428571428, 217.44000000000005, 'gini = 0.0\nsamples = 9\nvalue
= [0, 15, 0]\nclass = No')]
```



Best model:LogisticRegression

```
In [ ]:
```