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	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	2
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	2
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	2
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	2
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	2
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	2
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	2
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	2
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	2

209928 rows × 14 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209928 entries, 0 to 209927
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209928 non-null  object 
 1   BEN      51393 non-null   float64
 2   CO       87127 non-null   float64
 3   EBE      51350 non-null   float64
 4   NMHC     43517 non-null   float64
 5   NO       208954 non-null  float64
 6   NO_2     208973 non-null  float64
 7   O_3      122049 non-null  float64
 8   PM10     103743 non-null  float64
 9   PM25     51079 non-null   float64
 10  SO_2     87131 non-null   float64
 11  TCH      43519 non-null   float64
 12  TOL      51175 non-null   float64
 13  station  209928 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	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	280
6	2011-11-01 01:00:00	0.7	0.3	1.1	0.16	17.0	66.0	7.0	22.0	16.0	2.0	1.36	1.7	280
25	2011-11-01 02:00:00	1.8	0.3	2.8	0.20	34.0	76.0	3.0	34.0	21.0	8.0	1.71	7.4	280
30	2011-11-01 02:00:00	1.0	0.4	1.3	0.18	31.0	67.0	5.0	25.0	18.0	3.0	1.40	2.9	280
49	2011-11-01 03:00:00	1.3	0.2	2.4	0.22	29.0	72.0	3.0	33.0	20.0	8.0	1.75	6.2	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209862	2011-08-31 22:00:00	0.4	0.1	1.0	0.06	1.0	13.0	33.0	21.0	6.0	5.0	1.26	0.7	280
209881	2011-08-31 23:00:00	0.9	0.1	1.8	0.16	11.0	45.0	30.0	32.0	17.0	3.0	1.34	4.9	280
209886	2011-08-31 23:00:00	0.6	0.1	1.1	0.05	1.0	12.0	48.0	19.0	7.0	5.0	1.26	0.9	280
209905	2011-09-01 00:00:00	0.6	0.1	1.3	0.15	6.0	35.0	34.0	21.0	12.0	3.0	1.32	3.8	280
209910	2011-09-01 00:00:00	0.7	0.1	1.1	0.04	1.0	12.0	46.0	8.0	5.0	5.0	1.25	0.9	280

16460 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 [21]:
```
df.describe()
```

Out[21]:

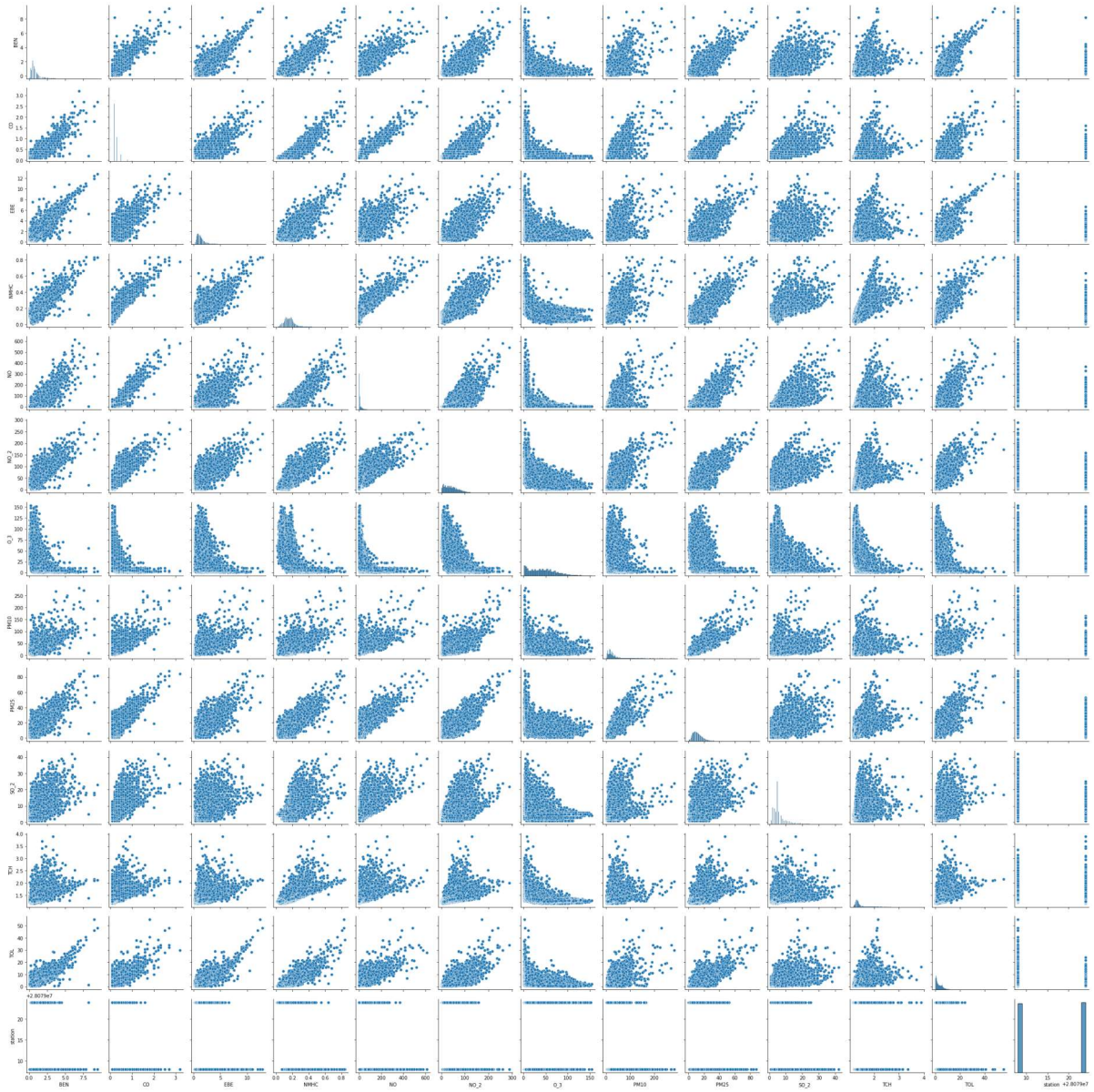
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	164
mean	0.900680	0.015188	1.471871	0.167043	23.671810	44.583961	
std	0.768892	0.122305	1.051004	0.075068	44.362859	31.569185	
min	0.100000	0.000000	0.200000	0.010000	1.000000	1.000000	
25%	0.500000	0.000000	0.800000	0.120000	2.000000	19.000000	
50%	0.700000	0.000000	1.200000	0.160000	7.000000	40.000000	
75%	1.100000	0.000000	1.700000	0.200000	25.000000	63.000000	
max	9.500000	1.000000	12.800000	0.840000	615.000000	289.000000	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)
```

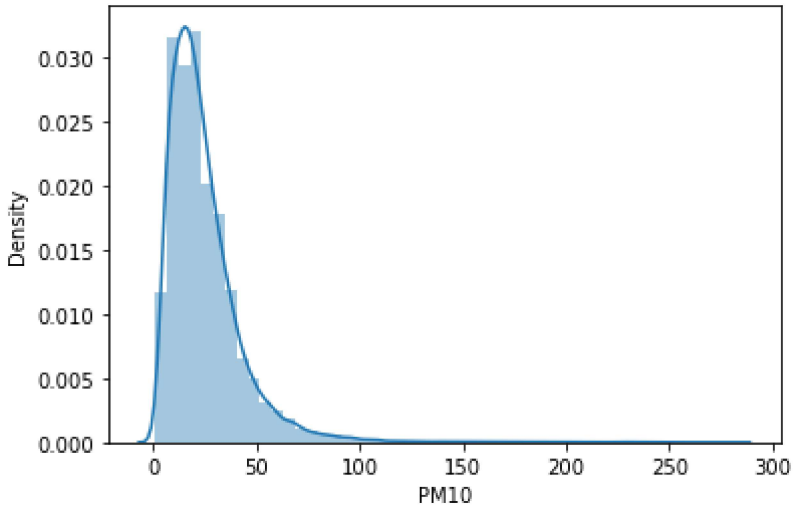
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x2ad9467b490>
```



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

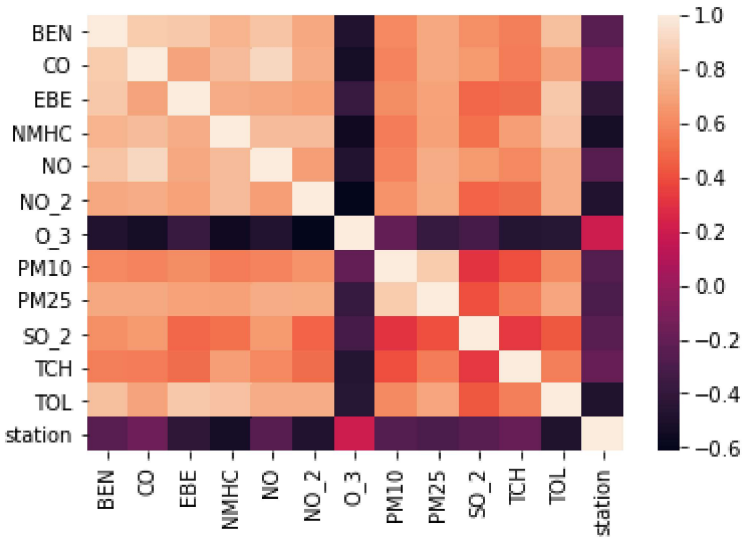
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[10]:
```
<AxesSubplot:>
```



In [18]:
```
df.loc[df['CO']<1,'CO']=0
df.loc[df['CO']>1,'CO']=1
df['CO']=df['CO'].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-18-dc2e98cdf216>: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['CO']=df['CO'].astype(int)
```

Out[18]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	s
1	2011-11-01 01:00:00	2.5	0	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	0	8.7	280
6	2011-11-01 01:00:00	0.7	0	1.1	0.16	17.0	66.0	7.0	22.0	16.0	2.0	0	1.7	280
25	2011-11-01 02:00:00	1.8	0	2.8	0.20	34.0	76.0	3.0	34.0	21.0	8.0	0	7.4	280
30	2011-11-01 02:00:00	1.0	0	1.3	0.18	31.0	67.0	5.0	25.0	18.0	3.0	0	2.9	280
49	2011-11-01 03:00:00	1.3	0	2.4	0.22	29.0	72.0	3.0	33.0	20.0	8.0	0	6.2	280
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209862	2011-08-31 22:00:00	0.4	0	1.0	0.06	1.0	13.0	33.0	21.0	6.0	5.0	0	0.7	280
209881	2011-08-31 23:00:00	0.9	0	1.8	0.16	11.0	45.0	30.0	32.0	17.0	3.0	0	4.9	280
209886	2011-08-31 23:00:00	0.6	0	1.1	0.05	1.0	12.0	48.0	19.0	7.0	5.0	0	0.9	280
209905	2011-09-01 00:00:00	0.6	0	1.3	0.15	6.0	35.0	34.0	21.0	12.0	3.0	0	3.8	280
209910	2011-09-01 00:00:00	0.7	0	1.1	0.04	1.0	12.0	46.0	8.0	5.0	5.0	0	0.9	280

16460 rows × 14 columns

LogisticRegression

In [22]:
```
x=df[[ 'BEN', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
y=df['CO']
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[22]:
```
LogisticRegression()
```

In [23]:
```
lgr.predict(x_test)
```

Out[23]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

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

Out[24]:
```
0.98440664236533
```

In [25]:
```
fs=StandardScaler().fit_transform(x)
logr=LogisticRegression()
logr.fit(fs,y)
```

Out[25]:
```
LogisticRegression()
```

In [28]:
```
o=[[1,2,3,4,5,6,7,8,9,10,11,12]]
prediction=logr.predict(o)
print(prediction)
```

```
[1]
```

In [29]:
```
logr.classes_
```

Out[29]:
```
array([0, 1])
```

In [30]:
```
logr.predict_proba(o)[0][0]
```

Out[30]:
```
0.016059072791087203
```

In [31]:
```
logr.predict_proba(o)[0][1]
```

Out[31]:
```
0.9839409272089128
```

LinearRegression

In [32]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

In [33]:
```
print(lr.intercept_)
```

```
-41097.06967040013
```

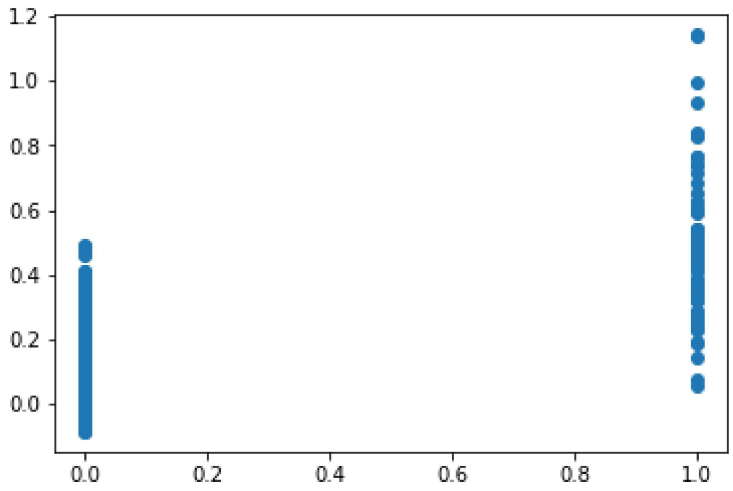
In [34]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[34]:

	Co-efficient
BEN	-1.785725e-02
EBE	2.619629e-02
NMHC	1.958372e-01
NO	1.989550e-03
NO_2	-3.988913e-04
O_3	6.990414e-04
PM10	2.781695e-04
PM25	-7.692662e-05
SO_2	-2.268107e-03
TCH	7.285839e-16
TOL	-9.990132e-04
station	1.463619e-03

In [35]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[35]:
```
<matplotlib.collections.PathCollection at 0x2ad94ce4340>
```



In [36]:
```
print(lr.score(x_test,y_test))
```

```
0.45150106570675164
```

Ridge,Lasso

In [37]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[37]:
```
Ridge(alpha=10)
```

In [38]:
```
rr.score(x_test,y_test)
```

Out[38]:
```
0.45001875708548655
```

In [39]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[39]:
```
Lasso(alpha=10)
```

In [40]:
```
la.score(x_test,y_test)
```

Out[40]:
```
-2.1809593200528e-05
```

ElasticNet

In [41]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[41]:
```
ElasticNet()
```

In [42]:
```
print(en.coef_)
```

```
[ 0.          0.          0.          0.00157729 -0.          0.
  0.          0.         -0.          0.          0.          0.        ]
```

In [43]:
```
print(en.intercept_)
```

```
-0.022243373742065482
```

In [44]:
```
print(en.predict(x_train))
```

```
[ 0.26955547 -0.02066608 -0.01435692 ...  0.04715743 -0.02066608
  0.00614787]
```

In [45]:
```
print(en.score(x_train,y_train))
```

```
0.43705840003491625
```

In [46]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 0.04175415033672663
```

In [47]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 0.008419570991725502
```

In [48]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 0.2043383232208942
```

RandomForest

In [49]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[49]:
```
RandomForestClassifier()
```

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

In [51]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[51]:
```
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')
```

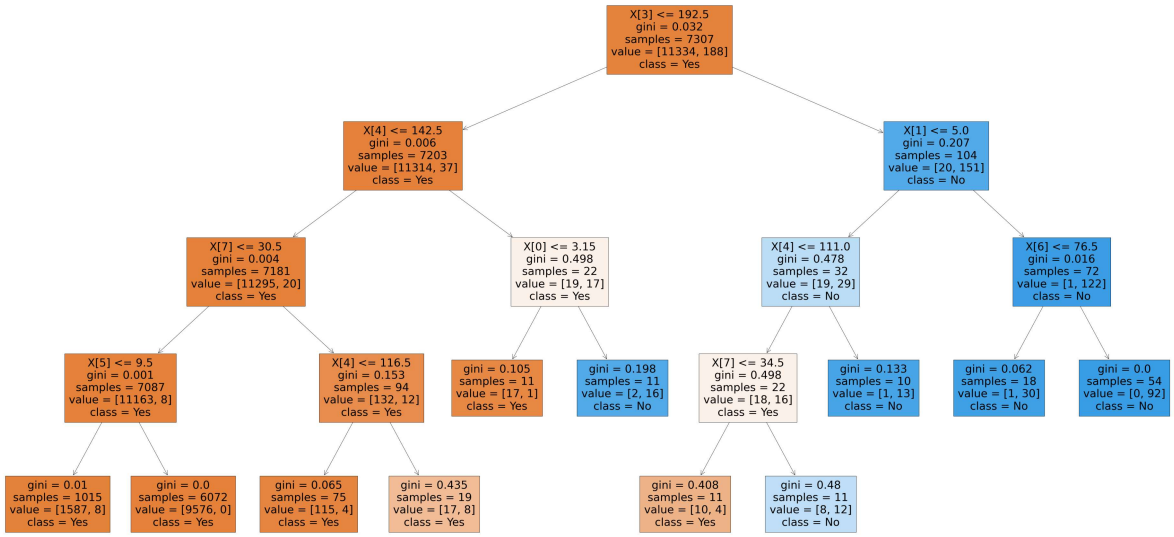
In [52]:
```
grid_search.best_score_
```

Out[52]:
```
0.9961812185384482
```

In [53]:
```
rfc_best=grid_search.best_estimator_
```

In [54]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],class_names=['Yes','No','Yes','No'],filled=T
```

Out[54]:
```
[Text(2525.684210526316, 1956.96, 'X[3] <= 192.5\ngini = 0.032\nsamples = 730
7\nvalue = [11334, 188]\nclass = Yes'),
 Text(1527.157894736842, 1522.0800000000002, 'X[4] <= 142.5\ngini = 0.006\nsa
mples = 7203\nvalue = [11314, 37]\nclass = Yes'),
 Text(939.7894736842105, 1087.2, 'X[7] <= 30.5\ngini = 0.004\nsamples = 7181
\nvalue = [11295, 20]\nclass = Yes'),
 Text(469.89473684210526, 652.3200000000002, 'X[5] <= 9.5\ngini = 0.001\nsamp
les = 7087\nvalue = [11163, 8]\nclass = Yes'),
 Text(234.94736842105263, 217.44000000000005, 'gini = 0.01\nsamples = 1015\nv
alue = [1587, 8]\nclass = Yes'),
 Text(704.8421052631579, 217.44000000000005, 'gini = 0.0\nsamples = 6072\nval
ue = [9576, 0]\nclass = Yes'),
 Text(1409.6842105263158, 652.3200000000002, 'X[4] <= 116.5\ngini = 0.153\nsa
mples = 94\nvalue = [132, 12]\nclass = Yes'),
 Text(1174.7368421052631, 217.44000000000005, 'gini = 0.065\nsamples = 75\nva
lue = [115, 4]\nclass = Yes'),
 Text(1644.6315789473683, 217.44000000000005, 'gini = 0.435\nsamples = 19\nva
lue = [17, 8]\nclass = Yes'),
 Text(2114.5263157894738, 1087.2, 'X[0] <= 3.15\ngini = 0.498\nsamples = 22\n
value = [19, 17]\nclass = Yes'),
 Text(1879.578947368421, 652.3200000000002, 'gini = 0.105\nsamples = 11\nvalu
e = [17, 1]\nclass = Yes'),
 Text(2349.4736842105262, 652.3200000000002, 'gini = 0.198\nsamples = 11\nval
ue = [2, 16]\nclass = No'),
 Text(3524.2105263157896, 1522.0800000000002, 'X[1] <= 5.0\ngini = 0.207\nsam
ples = 104\nvalue = [20, 151]\nclass = No'),
 Text(3054.315789473684, 1087.2, 'X[4] <= 111.0\ngini = 0.478\nsamples = 32\n
value = [19, 29]\nclass = No'),
 Text(2819.3684210526317, 652.3200000000002, 'X[7] <= 34.5\ngini = 0.498\nsam
ples = 22\nvalue = [18, 16]\nclass = Yes'),
 Text(2584.4210526315787, 217.44000000000005, 'gini = 0.408\nsamples = 11\nva
lue = [10, 4]\nclass = Yes'),
 Text(3054.315789473684, 217.44000000000005, 'gini = 0.48\nsamples = 11\nvalu
e = [8, 12]\nclass = No'),
 Text(3289.2631578947367, 652.3200000000002, 'gini = 0.133\nsamples = 10\nval
ue = [1, 13]\nclass = No'),
 Text(3994.1052631578946, 1087.2, 'X[6] <= 76.5\ngini = 0.016\nsamples = 72\n
value = [1, 122]\nclass = No'),
 Text(3759.157894736842, 652.3200000000002, 'gini = 0.062\nsamples = 18\nvalu
e = [1, 30]\nclass = No'),
 Text(4229.0526315789475, 652.3200000000002, 'gini = 0.0\nsamples = 54\nvalue
= [0, 92]\nclass = No')]
```



In [ ]:
```
Best model:RandomForest
```