In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [74]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\csvs_per_year\csvs_per_year\madrid_200
df
```

Out[74]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.0
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.5
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.0
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.7
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.1
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.8
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.8
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.7
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.8
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.3

217872 rows × 16 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

In [75]:
```
df=df.dropna()
df
```

Out[75]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.100000	0.07	56.250000	75.169998	2.11	42.160000
5	2001-08-01 01:00:00	2.11	0.63	2.48	5.940000	0.05	66.260002	118.099998	3.15	33.500000
21	2001-08-01 01:00:00	0.80	0.43	0.71	1.200000	0.10	27.190001	29.700001	0.76	56.990002
23	2001-08-01 01:00:00	1.29	0.34	1.41	3.090000	0.07	40.750000	51.570000	1.70	51.580002
25	2001-08-01 02:00:00	0.87	0.06	0.88	2.410000	0.01	29.709999	31.440001	1.20	56.520000
...	...	...	...	...	...	...	...	...	...	...
217829	2001-03-31 23:00:00	11.76	4.48	7.71	17.219999	0.89	103.900002	548.500000	7.62	9.680000
217847	2001-03-31 23:00:00	9.79	2.65	7.59	9.730000	0.46	91.320000	315.899994	3.75	6.660000
217849	2001-04-01 00:00:00	5.86	1.22	5.66	13.710000	0.25	64.370003	218.300003	6.46	7.480000
217853	2001-04-01 00:00:00	14.47	1.83	11.39	26.059999	0.33	84.230003	259.200012	11.39	5.440000
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.040000	0.18	76.809998	206.300003	5.20	8.340000

29669 rows × 16 columns

In [76]:
```
df.isnull().sum()
```

Out[76]:
```
date       0
BEN        0
CO         0
EBE        0
MXY        0
NMHC       0
NO_2       0
NOx        0
OXY        0
O_3        0
PM10       0
PXY        0
SO_2       0
TCH        0
TOL        0
station    0
dtype: int64
```

In [77]:
```
df.describe()
```

Out[77]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	296
mean	3.361895	1.005413	3.580229	8.113086	0.195222	67.652292	1
std	3.176669	0.863135	3.744496	7.909701	0.192585	34.003120	1
min	0.100000	0.000000	0.140000	0.210000	0.000000	1.180000	
25%	1.280000	0.470000	1.390000	3.040000	0.080000	44.299999	
50%	2.510000	0.760000	2.600000	5.830000	0.140000	64.449997	1
75%	4.420000	1.270000	4.580000	10.640000	0.250000	86.540001	2
max	54.560001	11.890000	77.260002	150.600006	2.880000	292.700012	19

In [7]:
```
df.columns
```

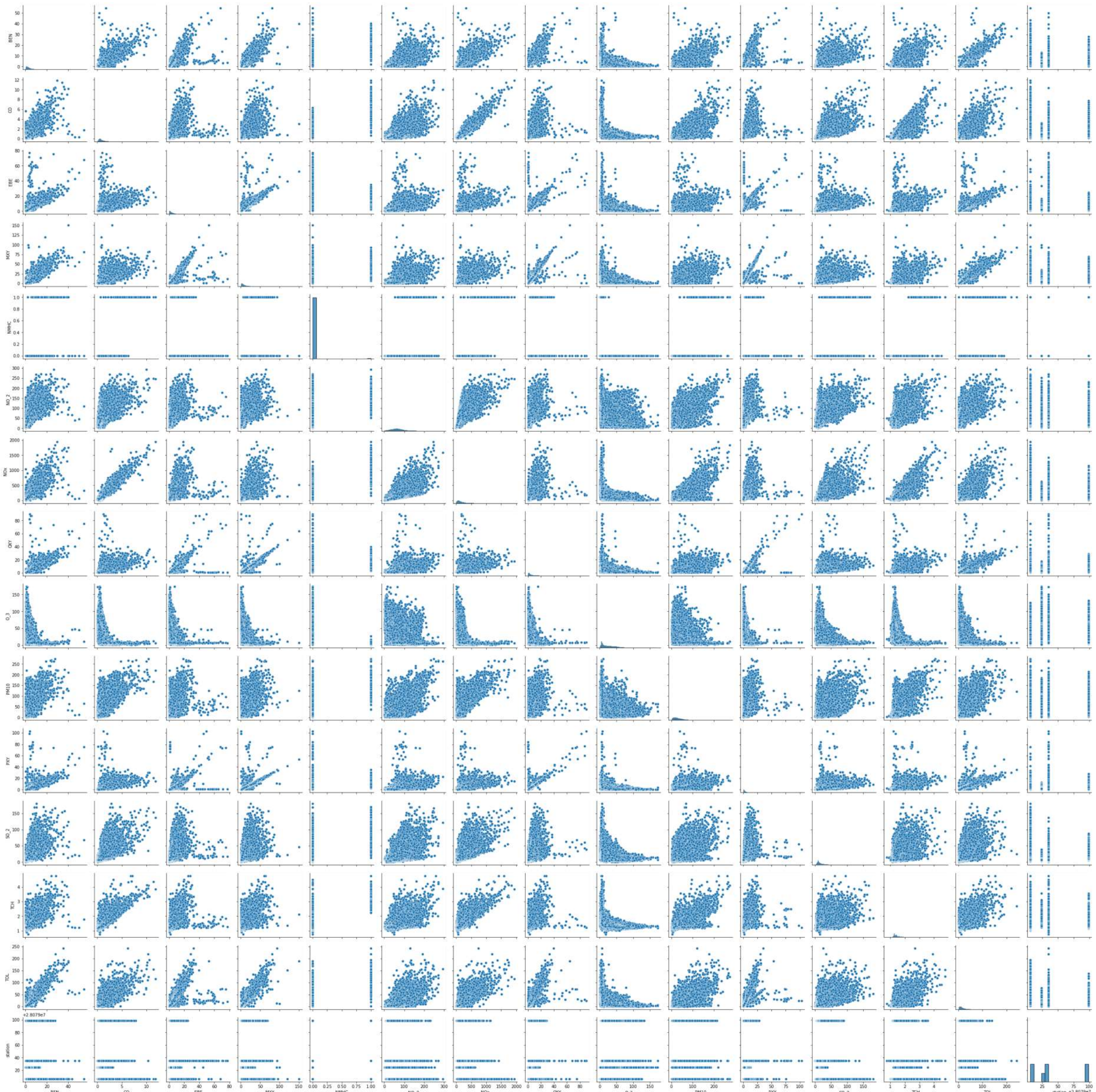
Out[7]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [8]:
```
df.loc[df['NMHC']<1,'NMHC']=0
df.loc[df['NMHC']>1,'NMHC']=1
df['NMHC']=df['NMHC'].astype(int)
```

```
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-8-c5145d14383f>: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['NMHC']=df['NMHC'].astype(int)
```

In [9]:
```
sns.pairplot(df)
```

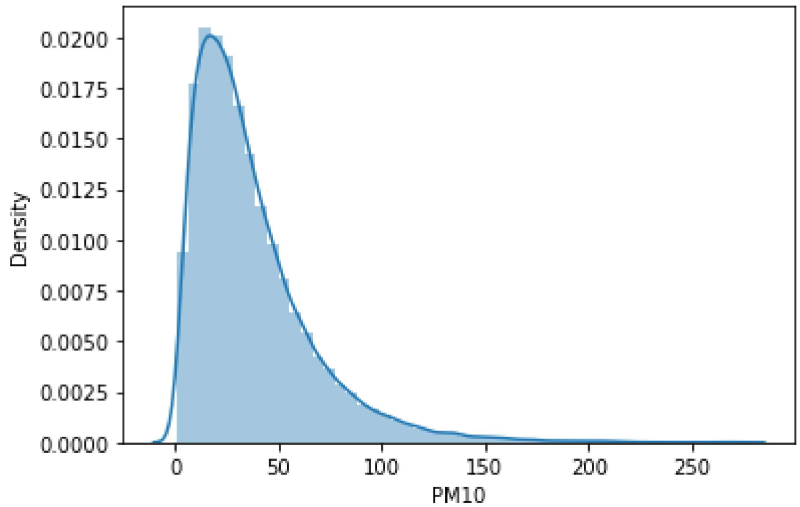
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1a131c8dfd0>
```



In [10]:
```
sns.distplot(df['PM10'])
```

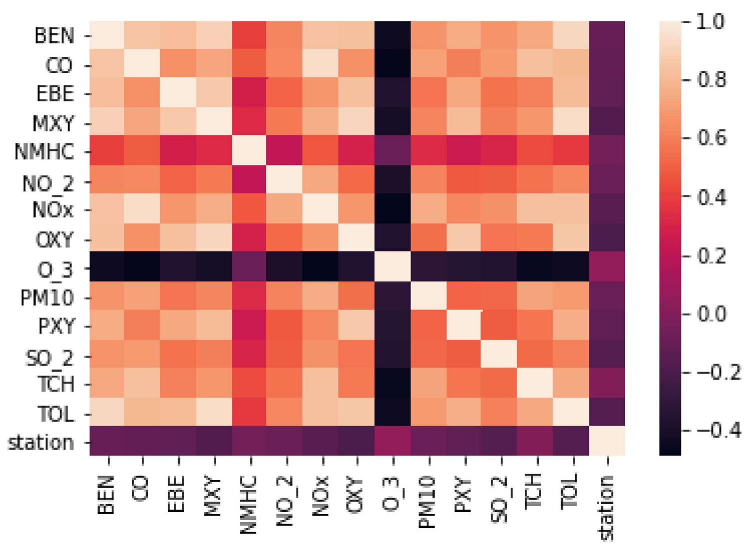
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[10]: <AxesSubplot:xlabel='PM10', ylabel='Density'>



In [11]:
```
sns.heatmap(df.corr())
```

Out[11]: <AxesSubplot:>



LinearRegression

In [56]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NO_2', 'NOx', 'OXY', 'O_3',
       'station', 'PXY', 'SO_2', 'TCH','TOL']]
y=df['PM10']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [57]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[57]:
```
LinearRegression()
```

In [58]:
```
print(lr.intercept_)
```

```
180436.05527078675
```

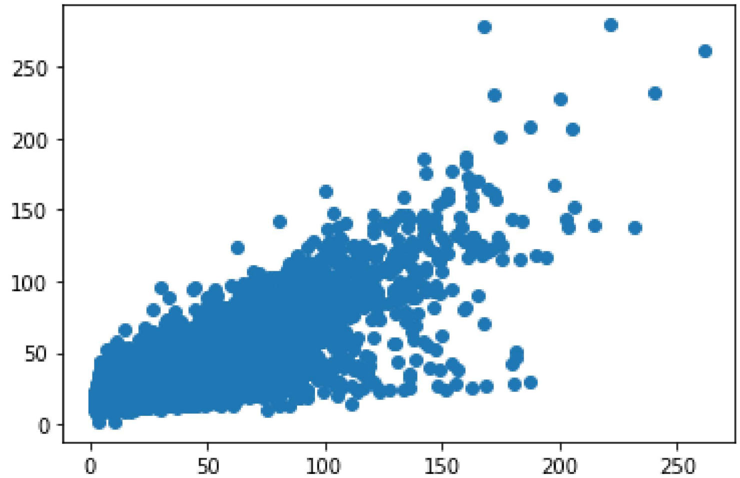
In [59]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[59]:

	Co-efficient
BEN	-0.760819
CO	-0.757628
EBE	0.562237
MXY	-0.593835
NO_2	0.143722
NOx	0.050776
OXY	-0.202122
O_3	0.120388
station	-0.006427
PXY	0.014732
SO_2	0.115195
TCH	35.300122
TOL	0.549980

In [60]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[60]: <matplotlib.collections.PathCollection at 0x1a14a3a7ee0>



In [61]:
```
print(lr.score(x_test,y_test))
```

0.628018474497537

Ridge,Lasso

In [62]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[62]: Ridge(alpha=10)

In [63]:
```
rr.score(x_test,y_test)
```

Out[63]: 0.6279356506816863

In [64]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[64]: Lasso(alpha=10)

In [65]:
```
la.score(x_test,y_test)
```

Out[65]: 0.5917301056999964

ElasticNet

In [66]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[66]:
```
ElasticNet()
```

In [67]:
```
print(en.coef_)
```

```
[-0.10220228  0.          0.32042828 -0.44626543  0.0994667   0.10047579
 -0.35859599  0.09424934  0.0240884   0.          0.06791359  0.53809407
  0.58111191]
```

In [68]:
```
print(en.intercept_)
```

```
-676376.177026843
```

In [69]:
```
print(en.predict(x_train))
```

```
[33.2024578  29.26592359 12.38848736 ... 54.60105985 36.04478949
 29.73010763]
```

In [70]:
```
print(en.score(x_train,y_train))
```

```
0.6055408001396526
```

In [71]:
```
from sklearn import metrics
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 12.577694478154767
```

In [72]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 316.8277572354703
```

In [73]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 3.5465045436534783
```

LogisticRegression

In [30]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH','TOL', 'station']]
y=df['NMHC']
from sklearn.linear_model import LogisticRegression
lgr=LogisticRegression()
lgr.fit(x_train,y_train)
```

```
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[30]:
```
LogisticRegression()
```

In [31]:
```
lgr.predict(x_test)
```

Out[31]:
```
array([28079099, 28079035, 28079035, ..., 28079035, 28079024, 28079006],
      dtype=int64)
```

In [32]:
```
lgr.score(x_test,y_test)
```

Out[32]:
```
0.6304909560723514
```

In [33]:
```
from sklearn.preprocessing import StandardScaler
fs=StandardScaler().fit_transform(x)
logr=LogisticRegression()
logr.fit(fs,y)
```

Out[33]:
```
LogisticRegression()
```

In [34]:
```
o=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
prediction=logr.predict(o)
print(prediction)
```

```
[0]
```

In [35]:
```
logr.classes_
```

Out[35]:
```
array([0, 1])
```

In [36]:
```
logr.predict_proba(o)[0][0]
```

Out[36]:
```
0.9999926193823634
```

In [37]:
```
logr.predict_proba(o)[0][1]
```
Out[37]:
```
7.3806176366125295e-06
```

RandomForest

In [48]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```
Out[48]:
```
RandomForestClassifier()
```

In [49]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [50]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```
Out[50]:
```
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 [51]:
```
grid_search.best_score_
```
Out[51]:
```
0.7233724961479199
```

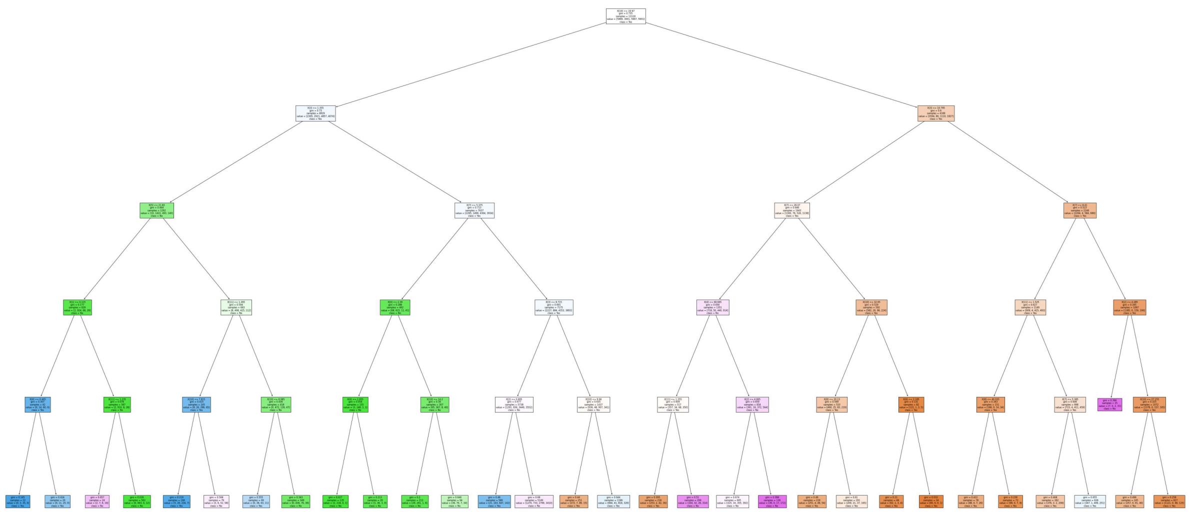
In [52]:
```
rfc_best=grid_search.best_estimator_
```

In [55]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],class_names=['Yes','No','Yes','No'],filled=Tr
```

Out[55]:
```
[Text(2352.9, 1993.2, 'X[10] <= 19.67\ngini = 0.735\nsamples = 13118\nvalue =
[5899, 3001, 5967, 5901]\nclass = Yes'),
 Text(1190.4, 1630.8000000000002, 'X[3] <= 1.355\ngini = 0.73\nsamples = 8929
\nvalue = [2305, 2921, 4857, 4074]\nclass = Yes'),
 Text(595.2, 1268.4, 'X[5] <= 21.83\ngini = 0.464\nsamples = 1292\nvalue = [1
0, 1422, 493, 140]\nclass = No'),
 Text(297.6, 906.0, 'X[1] <= 0.115\ngini = 0.177\nsamples = 629\nvalue = [2,
924, 68, 28]\nclass = No'),
 Text(148.8, 543.5999999999999, 'X[6] <= 0.465\ngini = 0.307\nsamples = 42\nv
alue = [0, 14, 60, 0]\nclass = Yes'),
 Text(74.4, 181.19999999999982, 'gini = 0.145\nsamples = 22\nvalue = [0, 3, 3
5, 0]\nclass = Yes'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.424\nsamples = 20\nva
lue = [0, 11, 25, 0]\nclass = Yes'),
 Text(446.40000000000003, 543.5999999999999, 'X[11] <= 1.235\ngini = 0.078\ns
amples = 587\nvalue = [2, 910, 8, 28]\nclass = No'),
 Text(372.0, 181.19999999999982, 'gini = 0.657\nsamples = 24\nvalue = [2, 7,
8, 16]\nclass = No'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.026\nsamples = 563\nva
lue = [0, 903, 0, 12]\nclass = No'),
 Text(892.8000000000001, 906.0, 'X[11] <= 1.265\ngini = 0.594\nsamples = 663
\nvalue = [8, 498, 425, 112]\nclass = No'),
 Text(744.0, 543.5999999999999, 'X[10] <= 7.815\ngini = 0.415\nsamples = 245
\nvalue = [8, 26, 286, 65]\nclass = Yes'),
 Text(669.6, 181.19999999999982, 'gini = 0.219\nsamples = 166\nvalue = [5, 2
0, 234, 7]\nclass = Yes'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.568\nsamples = 79\nval
ue = [3, 6, 52, 58]\nclass = No'),
 Text(1041.6000000000001, 543.5999999999999, 'X[10] <= 8.985\ngini = 0.436\ns
amples = 418\nvalue = [0, 472, 139, 47]\nclass = No'),
 Text(967.2, 181.19999999999982, 'gini = 0.555\nsamples = 69\nvalue = [0, 36,
63, 11]\nclass = Yes'),
 Text(1116.0, 181.19999999999982, 'gini = 0.343\nsamples = 349\nvalue = [0, 4
36, 76, 36]\nclass = No'),
 Text(1785.6000000000001, 1268.4, 'X[7] <= 5.375\ngini = 0.713\nsamples = 763
7\nvalue = [2295, 1499, 4364, 3934]\nclass = Yes'),
 Text(1488.0, 906.0, 'X[0] <= 2.39\ngini = 0.288\nsamples = 462\nvalue = [68,
615, 11, 41]\nclass = No'),
 Text(1339.2, 543.5999999999999, 'X[6] <= 2.655\ngini = 0.054\nsamples = 155
\nvalue = [3, 248, 3, 1]\nclass = No'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.027\nsamples = 135\nv
alue = [2, 218, 0, 1]\nclass = No'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.213\nsamples = 20\nva
lue = [1, 30, 3, 0]\nclass = No'),
 Text(1636.8000000000002, 543.5999999999999, 'X[10] <= 14.2\ngini = 0.39\nsam
ples = 307\nvalue = [65, 367, 8, 40]\nclass = No'),
 Text(1562.4, 181.19999999999982, 'gini = 0.2\nsamples = 213\nvalue = [29, 29
1, 1, 6]\nclass = No'),
 Text(1711.2, 181.19999999999982, 'gini = 0.646\nsamples = 94\nvalue = [36, 7
6, 7, 34]\nclass = No'),
 Text(2083.2000000000003, 906.0, 'X[3] <= 8.715\ngini = 0.691\nsamples = 7175
\nvalue = [2227, 884, 4353, 3893]\nclass = Yes'),
 Text(1934.4, 543.5999999999999, 'X[2] <= 0.895\ngini = 0.677\nsamples = 5738
\nvalue = [1291, 836, 3446, 3552]\nclass = No'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.46\nsamples = 589\nva
lue = [21, 103, 647, 142]\nclass = Yes'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.68\nsamples = 5149\nv
```

```
alue = [1270, 733, 2799, 3410]\nclass = No'),
 Text(2232.0, 543.5999999999999, 'X[10] <= 9.66\ngini = 0.635\nsamples = 1437
\nvalue = [936, 48, 907, 341]\nclass = Yes'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.44\nsamples = 251\nva
lue = [272, 7, 89, 15]\nclass = Yes'),
 Text(2306.4, 181.19999999999982, 'gini = 0.644\nsamples = 1186\nvalue = [66
4, 41, 818, 326]\nclass = Yes'),
 Text(3515.4, 1630.8000000000002, 'X[3] <= 10.795\ngini = 0.6\nsamples = 4189
\nvalue = [3594, 80, 1110, 1827]\nclass = Yes'),
 Text(2976.0, 1268.4, 'X[7] <= 26.07\ngini = 0.646\nsamples = 1943\nvalue =
[1300, 76, 526, 1138]\nclass = Yes'),
 Text(2678.4, 906.0, 'X[4] <= 68.665\ngini = 0.656\nsamples = 1351\nvalue =
[718, 50, 440, 914]\nclass = No'),
 Text(2529.6000000000004, 543.5999999999999, 'X[11] <= 1.355\ngini = 0.609\ns
amples = 517\nvalue = [357, 34, 68, 350]\nclass = Yes'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.395\nsamples = 218\nv
alue = [253, 2, 42, 36]\nclass = Yes'),
 Text(2604.0, 181.19999999999982, 'gini = 0.51\nsamples = 299\nvalue = [104,
32, 26, 314]\nclass = No'),
 Text(2827.2000000000003, 543.5999999999999, 'X[2] <= 4.665\ngini = 0.659\nsa
mples = 834\nvalue = [361, 16, 372, 564]\nclass = No'),
 Text(2752.8, 181.19999999999982, 'gini = 0.674\nsamples = 695\nvalue = [325,
16, 355, 392]\nclass = No'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.384\nsamples = 139\nv
alue = [36, 0, 17, 172]\nclass = No'),
 Text(3273.6000000000004, 906.0, 'X[10] <= 32.65\ngini = 0.529\nsamples = 592
\nvalue = [582, 26, 86, 224]\nclass = Yes'),
 Text(3124.8, 543.5999999999999, 'X[8] <= 20.13\ngini = 0.569\nsamples = 510
\nvalue = [460, 25, 83, 219]\nclass = Yes'),
 Text(3050.4, 181.19999999999982, 'gini = 0.46\nsamples = 219\nvalue = [251,
4, 46, 54]\nclass = Yes'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.61\nsamples = 291\nva
lue = [209, 21, 37, 165]\nclass = Yes'),
 Text(3422.4, 543.5999999999999, 'X[6] <= 3.185\ngini = 0.131\nsamples = 82\n
value = [122, 1, 3, 5]\nclass = Yes'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.21\nsamples = 48\nval
ue = [62, 1, 3, 4]\nclass = Yes'),
 Text(3496.8, 181.19999999999982, 'gini = 0.032\nsamples = 34\nvalue = [60,
0, 0, 1]\nclass = Yes'),
 Text(4054.8, 1268.4, 'X[7] <= 8.41\ngini = 0.523\nsamples = 2246\nvalue = [2
294, 4, 584, 689]\nclass = Yes'),
 Text(3868.8, 906.0, 'X[11] <= 1.525\ngini = 0.627\nsamples = 1149\nvalue =
[909, 4, 425, 493]\nclass = Yes'),
 Text(3720.0000000000005, 543.5999999999999, 'X[8] <= 40.035\ngini = 0.343\ns
amples = 151\nvalue = [186, 0, 14, 34]\nclass = Yes'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.422\nsamples = 78\nva
lue = [88, 0, 7, 26]\nclass = Yes'),
 Text(3794.4, 181.19999999999982, 'gini = 0.239\nsamples = 73\nvalue = [98,
0, 7, 8]\nclass = Yes'),
 Text(4017.6000000000004, 543.5999999999999, 'X[7] <= 5.385\ngini = 0.646\nsa
mples = 998\nvalue = [723, 4, 411, 459]\nclass = Yes'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.468\nsamples = 362\nv
alue = [376, 3, 2, 208]\nclass = Yes'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.655\nsamples = 636\nv
alue = [347, 1, 409, 251]\nclass = Yes'),
 Text(4240.8, 906.0, 'X[2] <= 4.285\ngini = 0.345\nsamples = 1097\nvalue = [1
385, 0, 159, 196]\nclass = Yes'),
```

```
 Text(4166.400000000001, 543.5999999999999, 'gini = 0.366\nsamples = 25\nvalu
e = [7, 0, 2, 31]\nclass = No'),
 Text(4315.200000000001, 543.5999999999999, 'X[10] <= 27.155\ngini = 0.325\ns
amples = 1072\nvalue = [1378, 0, 157, 165]\nclass = Yes'),
 Text(4240.8, 181.19999999999982, 'gini = 0.496\nsamples = 245\nvalue = [257,
0, 91, 40]\nclass = Yes'),
 Text(4389.6, 181.19999999999982, 'gini = 0.258\nsamples = 827\nvalue = [112
1, 0, 66, 125]\nclass = Yes')]
```



In [ ]:
```
Best model:LogisticRegression
```