In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
test_df=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\Mobile_Price_Classifi
test_df
train_df=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\Mobile_Price_Classif
train_df

```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_
0	842	0	2.2	0	1	0	7	0.6	1
1	1021	1	0.5	1	0	1	53	0.7	1
2	563	1	0.5	1	2	1	41	0.9	1
3	615	1	2.5	0	0	0	10	0.8	1
4	1821	1	1.2	0	13	1	44	0.6	1
...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	1
1996	1965	1	2.6	1	0	0	39	0.2	1
1997	1911	0	0.9	1	1	1	36	0.7	1
1998	1512	0	0.9	0	4	1	46	0.1	1
1999	510	1	2.0	1	5	1	45	0.9	1

2000 rows × 21 columns

In [3]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.3 KB
```

In [4]:

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.3 KB
```

In [5]:

```
x=train_df.drop('wifi',axis=1)
y=train_df['wifi']
```

In [6]:

```
x=test_df.drop('wifi',axis=1)
y=test_df['wifi']
```

In [7]:

```
test_df['dual_sim'].value_counts()
train_df['blue'].value_counts()
```

Out[7]:

```
blue
0    1010
1     990
Name: count, dtype: int64
```

In [8]:

```
from sklearn.model_selection import train_test_split
```

In [9]:

```
(x_train,x_test,y_train,y_test)=train_test_split(x,y,train_size=0.7,random_state=42
```

In [10]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[10]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [19]:

```
params={'max_depth':[2,3,5,10,20],
       'min_samples_leaf':[5,10,20,50,100,200],
       'n_estimators':[10,25,30,50,100,200]}
```

In [20]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[20]:

```
▸           GridSearchCV
▸ estimator: RandomForestClassifier
     ▸ RandomForestClassifier
```

In [21]:

```
grid_search.best_score_
```

Out[21]:

```
0.5264285714285715
```

In [22]:

```
rf_best=grid_search.best_estimator_
print(rf_best)
```

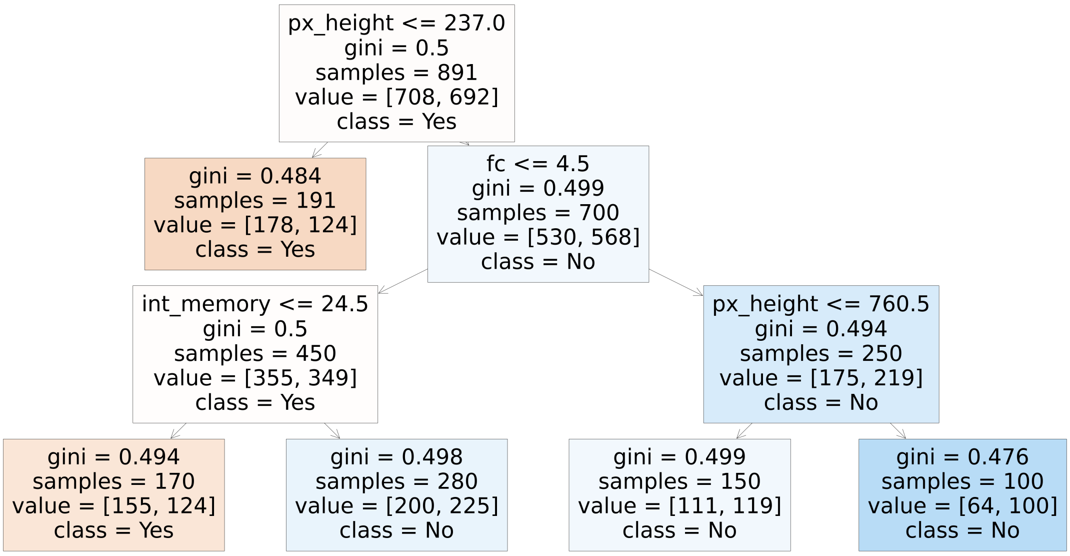
```
RandomForestClassifier(max_depth=3, min_samples_leaf=100, n_estimators=2
5)
```

In [25]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],f
```

Out[25]:

```
[Text(0.375, 0.875, 'px_height <= 237.0\ngini = 0.5\nsamples = 891\nvalue
= [708, 692]\nclass = Yes'),
 Text(0.25, 0.625, 'gini = 0.484\nsamples = 191\nvalue = [178, 124]\nclas
s = Yes'),
 Text(0.5, 0.625, 'fc <= 4.5\ngini = 0.499\nsamples = 700\nvalue = [530,
568]\nclass = No'),
 Text(0.25, 0.375, 'int_memory <= 24.5\ngini = 0.5\nsamples = 450\nvalue
= [355, 349]\nclass = Yes'),
 Text(0.125, 0.125, 'gini = 0.494\nsamples = 170\nvalue = [155, 124]\ncla
ss = Yes'),
 Text(0.375, 0.125, 'gini = 0.498\nsamples = 280\nvalue = [200, 225]\ncla
ss = No'),
 Text(0.75, 0.375, 'px_height <= 760.5\ngini = 0.494\nsamples = 250\nvalu
e = [175, 219]\nclass = No'),
 Text(0.625, 0.125, 'gini = 0.499\nsamples = 150\nvalue = [111, 119]\ncla
ss = No'),
 Text(0.875, 0.125, 'gini = 0.476\nsamples = 100\nvalue = [64, 100]\nclas
s = No')]
```

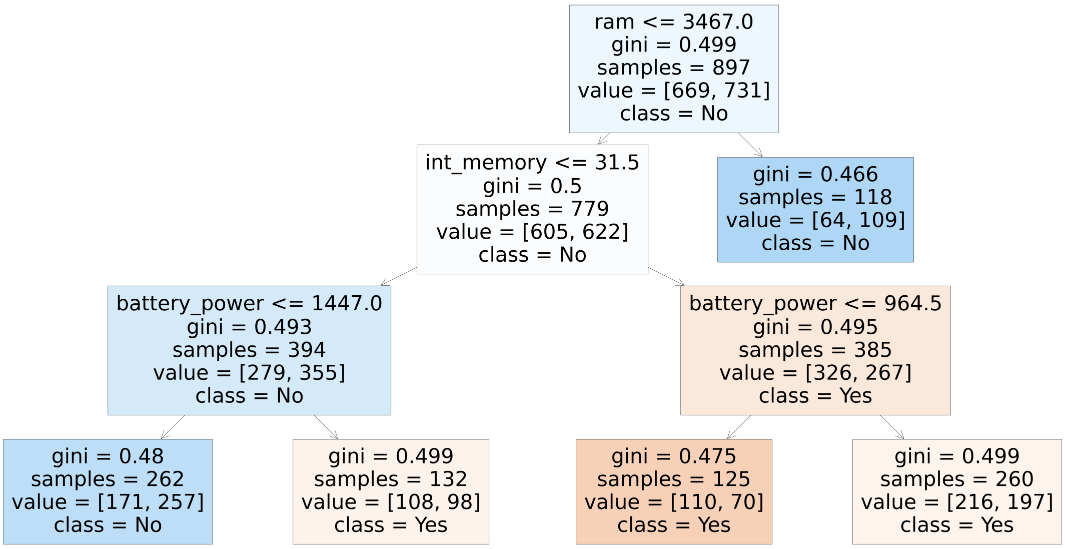


In [26]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=['Yes','No'],f
```

Out[26]:

```
[Text(0.625, 0.875, 'ram <= 3467.0\ngini = 0.499\nsamples = 897\nvalue =
[669, 731]\nclass = No'),
 Text(0.5, 0.625, 'int_memory <= 31.5\ngini = 0.5\nsamples = 779\nvalue =
[605, 622]\nclass = No'),
 Text(0.25, 0.375, 'battery_power <= 1447.0\ngini = 0.493\nsamples = 394
\nvalue = [279, 355]\nclass = No'),
 Text(0.125, 0.125, 'gini = 0.48\nsamples = 262\nvalue = [171, 257]\nclas
s = No'),
 Text(0.375, 0.125, 'gini = 0.499\nsamples = 132\nvalue = [108, 98]\nclas
s = Yes'),
 Text(0.75, 0.375, 'battery_power <= 964.5\ngini = 0.495\nsamples = 385\n
value = [326, 267]\nclass = Yes'),
 Text(0.625, 0.125, 'gini = 0.475\nsamples = 125\nvalue = [110, 70]\nclas
s = Yes'),
 Text(0.875, 0.125, 'gini = 0.499\nsamples = 260\nvalue = [216, 197]\ncla
ss = Yes'),
 Text(0.75, 0.625, 'gini = 0.466\nsamples = 118\nvalue = [64, 109]\nclass
= No')]
```



In [27]:

```
rf_best.feature_importances_
```

Out[27]:

```
array([0.10448548, 0.01349521, 0.0248207 , 0.00451811, 0.04312199,
       0.01809116, 0.07748647, 0.02597417, 0.04594737, 0.04286254,
       0.0331091 , 0.25902077, 0.1241097 , 0.04365679, 0.05300503,
       0.03260587, 0.01534112, 0.        , 0.00918685, 0.02916155])
```

In [28]:

```
imp_df=pd.DataFrame({'Varname':x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[28]:

	Varname	Imp
11	px_height	0.259021
12	px_width	0.124110
0	battery_power	0.104485
6	int_memory	0.077486
14	sc_h	0.053005
8	mobile_wt	0.045947
13	ram	0.043657
4	fc	0.043122
9	n_cores	0.042863
10	pc	0.033109
15	sc_w	0.032606
19	price_range	0.029162
7	m_dep	0.025974
2	clock_speed	0.024821
5	four_g	0.018091
16	talk_time	0.015341
1	blue	0.013495
18	touch_screen	0.009187
3	dual_sim	0.004518
17	three_g	0.000000

In [ ]:

```

```