In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [4]:

```
train_df=pd.read_csv(r"C:\Users\monim\Downloads\Mobile_Price_Classification_train.csv")
train_df
```

Out[4]:

dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	th
0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	
1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	
1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	
0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	
0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1	0	1	2	0.8	106	6	...	1222	1890	668	13	4	19	
1	0	0	39	0.2	187	4	...	915	1965	2032	11	10	16	
1	1	1	36	0.7	108	8	...	868	1632	3057	9	1	5	
0	4	1	46	0.1	145	5	...	336	670	869	18	10	19	
1	5	1	45	0.9	168	6	...	483	754	3919	19	4	2	

In [5]:

```
test_df=pd.read_csv(r"C:\Users\monim\Downloads\Mobile_Price_Classification_test.csv")
test_df
```

Out[5]:

blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w
1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	7
1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	0
1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	10
0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	0
0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121	14	8
0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933	8	1
0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223	5	0
1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509	15	11
1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828	9	2

In [6]:

```
train_df.head()
```

Out[6]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	

5 rows × 21 columns

In [7]:

```
train_df.shape
```

Out[7]:

```
(2000, 21)
```

In [8]:

```
test_df.head()
```

Out[8]:

blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w
1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	7
1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	0
1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	10
0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	0
0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	8

In [9]:

```
test_df.shape
```

Out[9]:

```
(1000, 21)
```

In [10]:

```
train_df.describe()
```

Out[10]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobi
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.0(
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.2
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.3
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.0(
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.0(
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.0(
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.0(
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.0(

8 rows × 21 columns

In [11]:

```
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: 328.2 KB
```

In [12]:

```
test_df.describe()
```

Out[12]:

m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	
000.000000	1000.00000	...	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1
0.517500	139.51100	...	10.054000	627.121000	1239.774000	2138.998000	11.995000	5.316000	11.085000	
0.280861	34.85155	...	6.095099	432.929699	439.670981	1088.092278	4.320607	4.240062	5.497636	
0.100000	80.00000	...	0.000000	0.000000	501.000000	263.000000	5.000000	0.000000	2.000000	
0.300000	109.75000	...	5.000000	263.750000	831.750000	1237.250000	8.000000	2.000000	6.750000	
0.500000	139.00000	...	10.000000	564.500000	1250.000000	2153.500000	12.000000	5.000000	11.000000	
0.800000	170.00000	...	16.000000	903.000000	1637.750000	3065.500000	16.000000	8.000000	16.000000	
1.000000	200.00000	...	20.000000	1907.000000	1998.000000	3989.000000	19.000000	18.000000	20.000000	

In [13]:

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             1000 non-null   int64  
 1   battery_power  1000 non-null   int64  
 2   blue           1000 non-null   int64  
 3   clock_speed    1000 non-null   float64
 4   dual_sim       1000 non-null   int64  
 5   fc             1000 non-null   int64  
 6   four_g         1000 non-null   int64  
 7   int_memory     1000 non-null   int64  
 8   m_dep          1000 non-null   float64
 9   mobile_wt      1000 non-null   int64  
 10  n_cores        1000 non-null   int64  
 11  pc             1000 non-null   int64  
 12  px_height      1000 non-null   int64  
 13  px_width       1000 non-null   int64  
 14  ram            1000 non-null   int64  
 15  sc_h           1000 non-null   int64  
 16  sc_w           1000 non-null   int64  
 17  talk_time      1000 non-null   int64  
 18  three_g        1000 non-null   int64  
 19  touch_screen   1000 non-null   int64  
 20  wifi           1000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 164.2 KB
```

TO FIND MISSING VALUE

In [14]:

```
train_df.isnull().sum()
```

Out[14]:

```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [15]:

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

In [16]:

```
train_df['touch_screen'].value_counts()
```

Out[16]:

```
touch_screen
1    1006
0     994
Name: count, dtype: int64
```

In [17]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=42)
x_train.shape
```

Out[17]:

```
(1400, 20)
```

In [18]:

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

Out[18]:

```
RandomForestClassifier
RandomForestClassifier()
```

In [19]:

```
rf=RandomForestClassifier()
```

In [20]:

```
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 [21]:

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

Out[21]:

GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier

In [22]:

```
grid_search.best_score_
```

Out[22]:

```
0.5335714285714286
```

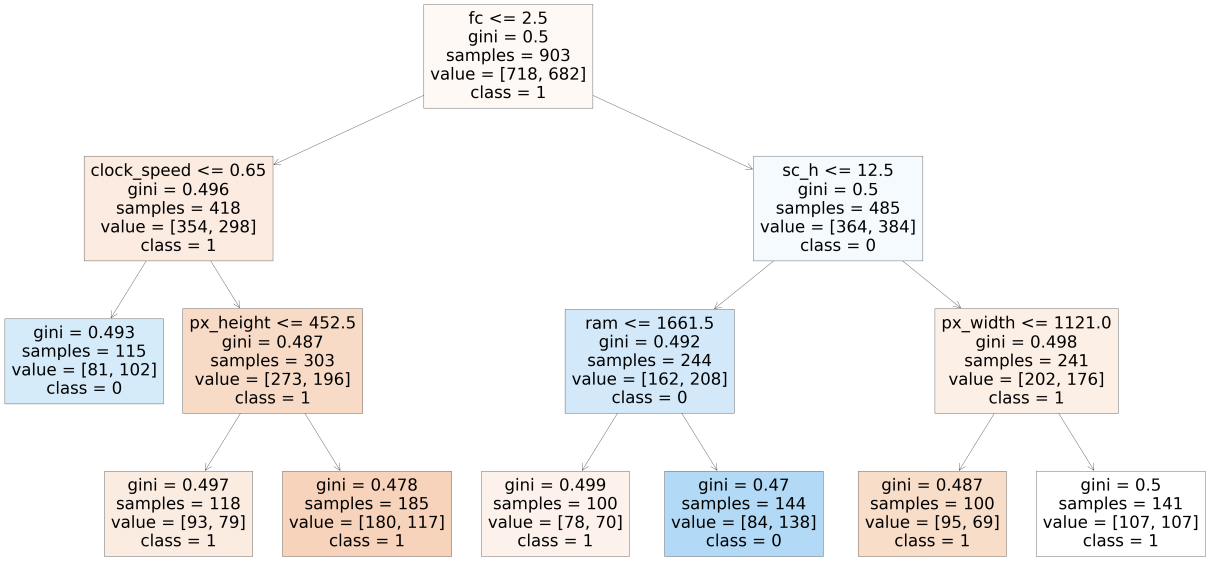
In [23]:

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

```
RandomForestClassifier(max_depth=5, min_samples_leaf=100)
```

In [24]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['1','0'],filled=True);
```



In [26]:

```
imp_df=pd.DataFrame({"Varname":x_train.columns,"Imp":rf_best.feature_importances_})
```

In [28]:

```
imp_df.sort_values(by="Imp",ascending=False)
```

Out[28]:

	Varname	Imp
5	int_memory	0.108074
12	ram	0.105456
11	px_width	0.083195
0	battery_power	0.079388
1	clock_speed	0.076822
10	px_height	0.070453
6	m_dep	0.056494
7	mobile_wt	0.055854
8	n_cores	0.054680
9	pc	0.048977
15	talk_time	0.048186
3	fc	0.048138
13	sc_h	0.044416
14	sc_w	0.043373
2	dual_sim	0.025474
17	touch_screen	0.015536
19	price_range	0.011186
4	four_g	0.010715
18	wifi	0.010058
16	three_g	0.003526

In [56]:

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

In [57]:

```
test_df['touch_screen'].value_counts()
```

Out[57]:

```
touch_screen
1    500
0    500
Name: count, dtype: int64
```

In [58]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=42)
x_test.shape
```

Out[58]:

```
(300, 20)
```

In [59]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_test,y_test)
```

Out[59]:

RandomForestClassifier

RandomForestClassifier()

In [60]:

```
rf=RandomForestClassifier()
```

In [61]:

```
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 [62]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(x_test,y_test)
```

Out[62]:

GridSearchCV

estimator: RandomForestClassifier

RandomForestClassifier

In [63]:

```
grid_search.best_score_
```

Out[63]:

```
0.78
```

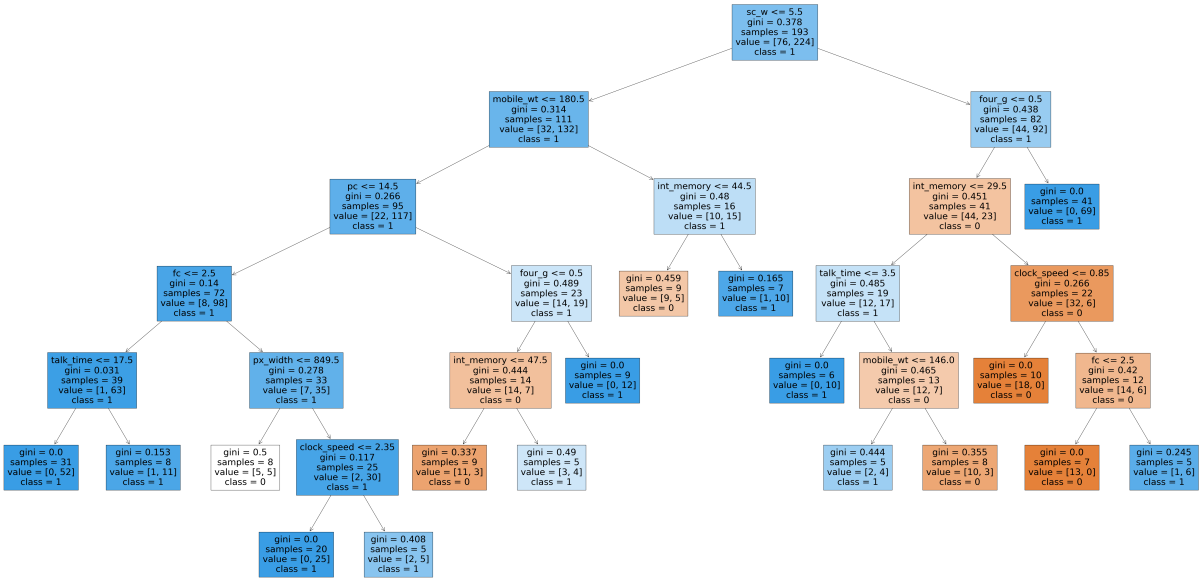
In [64]:

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

```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=25)
```

In [65]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['0','1','2','3'],filled=True);
```



In [ ]: