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

In [9]:
```
test_df=pd.read_csv(r"C:\Users\pavan\Downloads\Mobile_Price_Classification_test1.csv")
test_df
```

Out[9]:

blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch
1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	7	2	0	
1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	0	7	1	
1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	10	10	0	
0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	0	7	1	
0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	8	7	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121	14	8	15	1	
0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933	8	1	19	0	
0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223	5	0	14	1	
1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509	15	11	6	0	
1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828	9	2	3	1	

In [8]:
```
train_df=pd.read_csv(r"C:\Users\pavan\Downloads\Mobile_Price_Classification_train1.csv")
train_df
```

Out[8]:

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

In [10]:
```
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 [11]:
```
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
```

In [12]:
```
x=train_df.drop('ram',axis=1)
y=train_df['ram']
```

In [13]:
```
x=test_df.drop('ram',axis=1)
y=test_df['ram']
```

In [14]:
```
train_df['dual_sim'].value_counts()
```

Out[14]:
```
dual_sim
1    1019
0     981
Name: count, dtype: int64
```

In [15]:
```
test_df['dual_sim'].value_counts()
```

Out[15]:
```
dual_sim
1    517
0    483
Name: count, dtype: int64
```

In [16]:
```
T={"three_g":{'Yes':1,'No':0}}
train_df=train_df.replace(T)
train_df
```

Out[16]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668	13	
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032	11	
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057	9	
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869	18	
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919	19	

2000 rows × 21 columns

In [17]:
```
T={"three_g":{'Yes':1,'No':0}}
test_df=test_df.replace(T)
test_df
```

Out[17]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121	14	
996	997	609	0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933	8	
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223	5	
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509	15	
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828	9	

1000 rows × 21 columns

In [18]:
```
x=train_df.drop('touch_screen',axis=1)
y=train_df['touch_screen']
train_df
```

Out[18]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668	13	
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032	11	
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057	9	
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869	18	
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919	19	

2000 rows × 21 columns

In [19]:
```
x=test_df.drop('touch_screen',axis=1)
y=test_df['touch_screen']
```

In [20]:
```
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,x_test.shape
```

Out[20]:
```
((700, 20), (300, 20))
```

In [21]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[21]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [22]:
```
rf=RandomForestClassifier()
```

In [23]:
```
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 [26]:
```
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[26]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [27]:
```
grid_search.best_score_
```

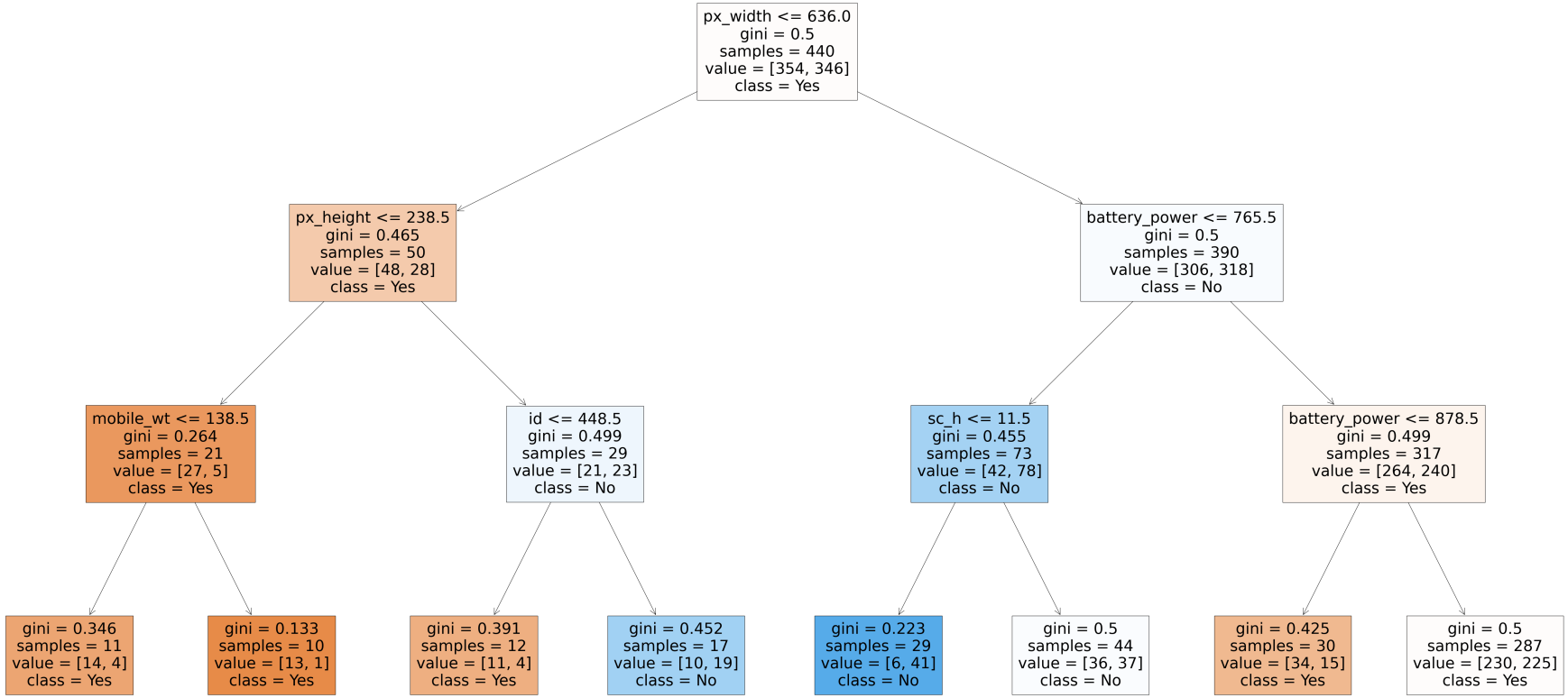
Out[27]:
```
0.52
```

In [28]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=3, min_samples_leaf=10, n_estimators=30)
```

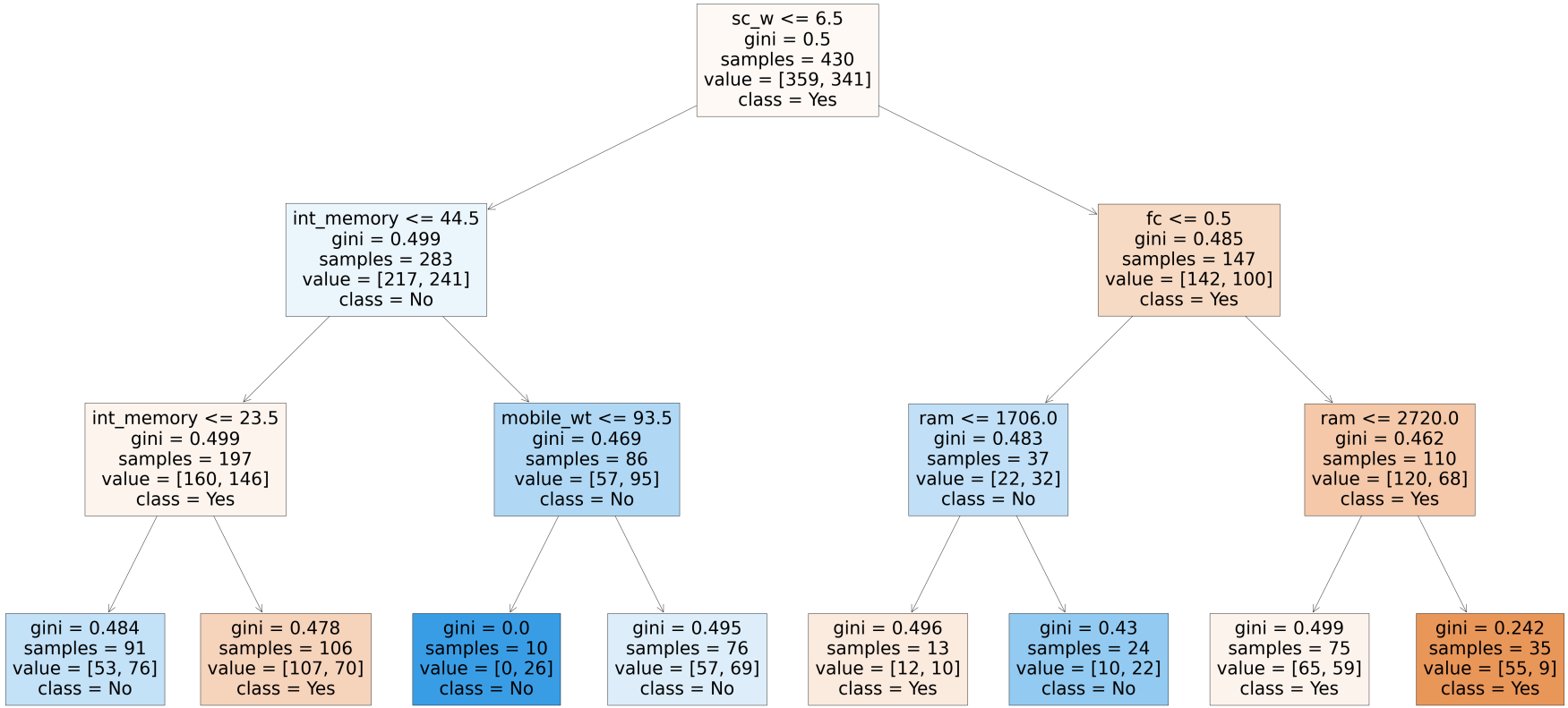
In [29]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True)
```

Out[29]:
```
[Text(0.5, 0.875, 'px_width <= 636.0\ngini = 0.5\nsamples = 440\nvalue = [354, 346]\nclass = Yes'),
 Text(0.25, 0.625, 'px_height <= 238.5\ngini = 0.465\nsamples = 50\nvalue = [48, 28]\nclass = Yes'),
 Text(0.125, 0.375, 'mobile_wt <= 138.5\ngini = 0.264\nsamples = 21\nvalue = [27, 5]\nclass = Yes'),
 Text(0.0625, 0.125, 'gini = 0.346\nsamples = 11\nvalue = [14, 4]\nclass = Yes'),
 Text(0.1875, 0.125, 'gini = 0.133\nsamples = 10\nvalue = [13, 1]\nclass = Yes'),
 Text(0.375, 0.375, 'id <= 448.5\ngini = 0.499\nsamples = 29\nvalue = [21, 23]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.391\nsamples = 12\nvalue = [11, 4]\nclass = Yes'),
 Text(0.4375, 0.125, 'gini = 0.452\nsamples = 17\nvalue = [10, 19]\nclass = No'),
 Text(0.75, 0.625, 'battery_power <= 765.5\ngini = 0.5\nsamples = 390\nvalue = [306, 318]\nclass = No'),
 Text(0.625, 0.375, 'sc_h <= 11.5\ngini = 0.455\nsamples = 73\nvalue = [42, 78]\nclass = No'),
 Text(0.5625, 0.125, 'gini = 0.223\nsamples = 29\nvalue = [6, 41]\nclass = No'),
 Text(0.6875, 0.125, 'gini = 0.5\nsamples = 44\nvalue = [36, 37]\nclass = No'),
 Text(0.875, 0.375, 'battery_power <= 878.5\ngini = 0.499\nsamples = 317\nvalue = [264, 240]\nclass = Yes'),
 Text(0.8125, 0.125, 'gini = 0.425\nsamples = 30\nvalue = [34, 15]\nclass = Yes'),
 Text(0.9375, 0.125, 'gini = 0.5\nsamples = 287\nvalue = [230, 225]\nclass = Yes')]
```


px_width <= 636.0
gini = 0.5
samples = 440
value = [354, 346]
class = Yes
px_height <= 238.5
gini = 0.465
samples = 50
value = [48, 28]
class = Yes
battery_power <= 765.5
gini = 0.5
samples = 390
value = [306, 318]
class = No
mobile_wt <= 138.5
gini = 0.264
samples = 21
value = [27, 5]
class = Yes
id <= 448.5
gini = 0.499
samples = 29
value = [21, 23]
class = No
sc_h <= 11.5
gini = 0.455
samples = 73
value = [42, 78]
class = No
battery_power <= 878.5
gini = 0.499
samples = 317
value = [264, 240]
class = Yes
gini = 0.346
samples = 11
value = [14, 4]
class = Yes
gini = 0.133
samples = 10
value = [13, 1]
class = Yes
gini = 0.391
samples = 12
value = [11, 4]
class = Yes
gini = 0.452
samples = 17
value = [10, 19]
class = No
gini = 0.223
samples = 29
value = [6, 41]
class = No
gini = 0.5
samples = 44
value = [36, 37]
class = No
gini = 0.425
samples = 30
value = [34, 15]
class = Yes
gini = 0.5
samples = 287
value = [230, 225]
class = Yes

In [30]:
```
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'],filled=True)
```

Out[30]:
```
[Text(0.5, 0.875, 'sc_w <= 6.5\ngini = 0.5\nsamples = 430\nvalue = [359, 341]\nclass = Yes'),
 Text(0.25, 0.625, 'int_memory <= 44.5\ngini = 0.499\nsamples = 283\nvalue = [217, 241]\nclass = No'),
 Text(0.125, 0.375, 'int_memory <= 23.5\ngini = 0.499\nsamples = 197\nvalue = [160, 146]\nclass = Yes'),
 Text(0.0625, 0.125, 'gini = 0.484\nsamples = 91\nvalue = [53, 76]\nclass = No'),
 Text(0.1875, 0.125, 'gini = 0.478\nsamples = 106\nvalue = [107, 70]\nclass = Yes'),
 Text(0.375, 0.375, 'mobile_wt <= 93.5\ngini = 0.469\nsamples = 86\nvalue = [57, 95]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.0\nsamples = 10\nvalue = [0, 26]\nclass = No'),
 Text(0.4375, 0.125, 'gini = 0.495\nsamples = 76\nvalue = [57, 69]\nclass = No'),
 Text(0.75, 0.625, 'fc <= 0.5\ngini = 0.485\nsamples = 147\nvalue = [142, 100]\nclass = Yes'),
 Text(0.625, 0.375, 'ram <= 1706.0\ngini = 0.483\nsamples = 37\nvalue = [22, 32]\nclass = No'),
 Text(0.5625, 0.125, 'gini = 0.496\nsamples = 13\nvalue = [12, 10]\nclass = Yes'),
 Text(0.6875, 0.125, 'gini = 0.43\nsamples = 24\nvalue = [10, 22]\nclass = No'),
 Text(0.875, 0.375, 'ram <= 2720.0\ngini = 0.462\nsamples = 110\nvalue = [120, 68]\nclass = Yes'),
 Text(0.8125, 0.125, 'gini = 0.499\nsamples = 75\nvalue = [65, 59]\nclass = Yes'),
 Text(0.9375, 0.125, 'gini = 0.242\nsamples = 35\nvalue = [55, 9]\nclass = Yes')]
```



In [31]:
```
rf_best.feature_importances_
```

Out[31]:
```
array([0.12946625, 0.06528793, 0.01360827, 0.04551509, 0.        ,
       0.06585844, 0.00388945, 0.05232273, 0.02150574, 0.08150169,
       0.01656414, 0.03169023, 0.09414652, 0.05282527, 0.12388779,
       0.09210783, 0.07765048, 0.0258927 , 0.00627947, 0.        ])
```

In [32]:
```
imp_df=pd.DataFrame({'Varname':x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[32]:

	Varname	Imp
0	id	0.129466
14	ram	0.123888
12	px_height	0.094147
15	sc_h	0.092108
9	mobile_wt	0.081502
16	sc_w	0.077650
5	fc	0.065858
1	battery_power	0.065288
13	px_width	0.052825
7	int_memory	0.052323
3	clock_speed	0.045515
11	pc	0.031690
17	talk_time	0.025893
8	m_dep	0.021506
10	n_cores	0.016564
2	blue	0.013608
18	three_g	0.006279
6	four_g	0.003889
4	dual_sim	0.000000
19	wifi	0.000000

In [ ]: