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\my pc\downloads\Mobile_Price_Classification_train.csv")
test_df
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_hei
0	842	0	2.2	0	1	0	7	0.6	188	2	...	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	9
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1
...	...	...	...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	9
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	8
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	3
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	4

2000 rows × 21 columns

In [3]:
```
test_df.head()
```

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height
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 [4]:
```
test_df.tail()
```

Out[4]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_heig
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	12
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	9
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	8
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	3
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	4

5 rows × 21 columns

In [5]:
```
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: 328.2 KB
```

In [6]:
```
test_df.shape
```

Out[6]: (2000, 21)

In [7]:
```
test_df.isna().any()
```

Out[7]:
```
battery_power    False
blue             False
clock_speed      False
dual_sim         False
fc               False
four_g           False
int_memory       False
m_dep            False
mobile_wt        False
n_cores          False
pc               False
px_height        False
px_width         False
ram              False
sc_h             False
sc_w             False
talk_time        False
three_g          False
touch_screen     False
wifi             False
price_range      False
dtype: bool
```

In [8]:
```
x=test_df.drop("price_range",axis=1)
y=test_df["price_range"]
```

In [9]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
x_train.shape,x_test.shape
```

Out[9]:
```
((1500, 20), (500, 20))
```

In [10]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[10]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [11]:
```
rf=RandomForestClassifier()
```

In [12]:
```
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 [13]:
```
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[13]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [14]:
```
grid_search.best_score_
```

Out[14]:
```
0.8466666666666667
```

In [15]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=200)
```

In [16]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,filled=True)
```

```
 Text(0.5398230088495575, 0.4230769230769231, 'gini = 0.0\nsamples = 5\nvalue = [0, 8, 0,
0]'),
 Text(0.5575221238938053, 0.4230769230769231, 'gini = 0.18\nsamples = 5\nvalue = [0, 9, 1,
0]'),
 Text(0.5663716814159292, 0.5, 'gini = 0.406\nsamples = 5\nvalue = [0, 1, 6, 1]'),
 Text(0.584070796460177, 0.6538461538461539, 'px_height <= 1604.0\ngini = 0.227\nsamples =
13\nvalue = [0, 3, 20, 0]'),
 Text(0.5752212389380531, 0.5769230769230769, 'gini = 0.105\nsamples = 8\nvalue = [0, 1, 1
7, 0]'),
 Text(0.5929203539823009, 0.5769230769230769, 'gini = 0.48\nsamples = 5\nvalue = [0, 2, 3,
0]'),
 Text(0.78125, 0.8846153846153846, 'int_memory <= 15.5\ngini = 0.541\nsamples = 429\nvalue
= [0, 33, 284, 350]'),
 Text(0.665929203539823, 0.8076923076923077, 'battery_power <= 1844.5\ngini = 0.506\nsampl
es = 105\nvalue = [0, 5, 100, 64]'),
 Text(0.6570796460176991, 0.7307692307692307, 'px_height <= 827.0\ngini = 0.473\nsamples =
97\nvalue = [0, 5, 100, 48]'),
 Text(0.6327433628318584, 0.6538461538461539, 'sc_w <= 9.5\ngini = 0.413\nsamples = 69\nva
lue = [0, 5, 79, 24]'),
```

In [17]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[20],feature_names=x.columns,filled=True)
```

```
0, 0]'),
 Text(0.5, 0.43333333333333335, 'gini = 0.469\nsamples = 5\nvalue = [0, 3, 5, 0]'),
 Text(0.5084745762711864, 0.5, 'gini = 0.397\nsamples = 6\nvalue = [0, 8, 3, 0]'),
 Text(0.4745762711864407, 0.6333333333333333, 'gini = 0.355\nsamples = 9\nvalue = [3, 10,
0, 0]'),
 Text(0.5338983050847458, 0.7, 'mobile_wt <= 158.0\ngini = 0.643\nsamples = 28\nvalue = [1
7, 10, 10, 0]'),
 Text(0.5254237288135594, 0.6333333333333333, 'battery_power <= 898.5\ngini = 0.559\nsampl
es = 19\nvalue = [13, 9, 2, 0]'),
 Text(0.5169491525423728, 0.5666666666666667, 'gini = 0.0\nsamples = 6\nvalue = [6, 0, 0,
0]'),
 Text(0.5338983050847458, 0.5666666666666667, 'blue <= 0.5\ngini = 0.586\nsamples = 13\nva
lue = [7, 9, 2, 0]'),
 Text(0.5254237288135594, 0.5, 'gini = 0.531\nsamples = 6\nvalue = [5, 2, 1, 0]'),
 Text(0.5423728813559322, 0.5, 'gini = 0.46\nsamples = 7\nvalue = [2, 7, 1, 0]'),
 Text(0.5423728813559322, 0.6333333333333333, 'gini = 0.521\nsamples = 9\nvalue = [4, 1,
8, 0]'),
 Text(0.5169491525423728, 0.7666666666666667, 'gini = 0.18\nsamples = 7\nvalue = [0, 9, 1,
0]'),
 Text(0.8919491525423728, 0.9, 'px_height <= 1296.0\ngini = 0.562\nsamples = 438\nvalue =
```

In [18]:
```
rf_best.feature_importances_
```

Out[18]:
```
array([0.07253516, 0.00541411, 0.01994608, 0.0045408 , 0.01753728,
       0.00482876, 0.02629184, 0.01643191, 0.02885589, 0.0164592 ,
       0.01997988, 0.04915981, 0.05385844, 0.59241764, 0.01813642,
       0.02091135, 0.02102381, 0.00244283, 0.00487873, 0.00435005])
```

In [19]:
```
imp_df=pd.DataFrame({"varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[19]:

	varname	Imp
13	ram	0.592418
0	battery_power	0.072535
12	px_width	0.053858
11	px_height	0.049160
8	mobile_wt	0.028856
6	int_memory	0.026292
16	talk_time	0.021024
15	sc_w	0.020911
10	pc	0.019980
2	clock_speed	0.019946
14	sc_h	0.018136
4	fc	0.017537
9	n_cores	0.016459
7	m_dep	0.016432
1	blue	0.005414
18	touch_screen	0.004879
5	four_g	0.004829
3	dual_sim	0.004541
19	wifi	0.004350
17	three_g	0.002443

In [22]:
```
test_df=pd.read_csv(r"C:\Users\my pc\downloads\Mobile_Price_Classification_test.csv")
test_df
```

Out[22]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_hei
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	2
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	7
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	12
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	2
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	7
...	...	...	...	...	...	...	...	...	...	...	...	...	
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	6
996	997	609	0	1.8	1	0	0	13	0.9	186	...	2	11
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	4
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	4

1000 rows × 21 columns

In [23]:
```
test_df.head()
```

Out[23]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749

5 rows × 21 columns

In [24]:
```
test_df.tail()
```

Out[24]:

blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	s
1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121	14	
0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933	8	
0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223	5	
1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509	15	
1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828	9	

In [25]:
```
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 [26]:
```
test_df.isna().any()
```

Out[26]:
```
id               False
battery_power    False
blue             False
clock_speed      False
dual_sim         False
fc               False
four_g           False
int_memory       False
m_dep            False
mobile_wt        False
n_cores          False
pc               False
px_height        False
px_width         False
ram              False
sc_h             False
sc_w             False
talk_time        False
three_g          False
touch_screen     False
wifi             False
dtype: bool
```

In [27]:
```
test_df.shape
```

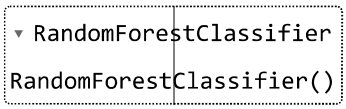
Out[27]:
```
(1000, 21)
```

In [29]:
```
x=test_df.drop("three_g",axis=1)
y=test_df["three_g"]
```

In [30]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
x_train.shape,x_test.shape
```

Out[30]: ((750, 20), (250, 20))

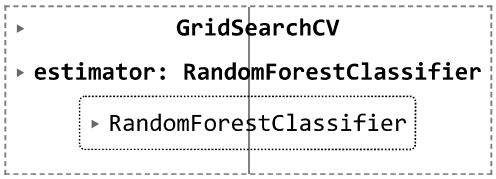
In [31]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[31]:


In [32]:
```
rf=RandomForestClassifier()
```

In [33]:
```
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 [34]:
```
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[34]:


In [35]:
```
grid_search.best_score_
```

Out[35]: 0.756

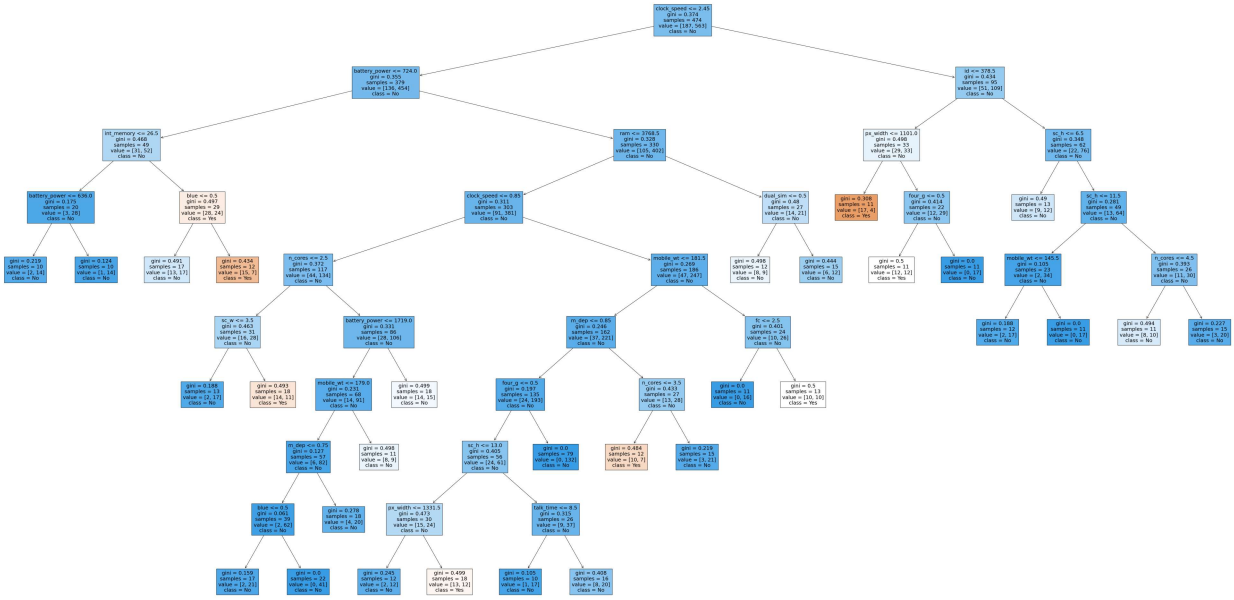
In [36]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=10, min_samples_leaf=10, n_estimators=50)
```

In [37]:
```
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[37]:
```
[Text(0.5519366197183099, 0.95, 'clock_speed <= 2.45\ngini = 0.374\nsamples = 474\nvalue =
[187, 563]\nclass = No'),
 Text(0.31514084507042256, 0.85, 'battery_power <= 724.0\ngini = 0.355\nsamples = 379\nvalue
= [136, 454]\nclass = No'),
 Text(0.11267605633802817, 0.75, 'int_memory <= 26.5\ngini = 0.468\nsamples = 49\nvalue = [3
1, 52]\nclass = No'),
 Text(0.056338028169014086, 0.65, 'battery_power <= 636.0\ngini = 0.175\nsamples = 20\nvalue
= [3, 28]\nclass = No'),
 Text(0.028169014084507043, 0.55, 'gini = 0.219\nsamples = 10\nvalue = [2, 14]\nclass = N
o'),
 Text(0.08450704225352113, 0.55, 'gini = 0.124\nsamples = 10\nvalue = [1, 14]\nclass = No'),
 Text(0.16901408450704225, 0.65, 'blue <= 0.5\ngini = 0.497\nsamples = 29\nvalue = [28, 24]
\nclass = Yes'),
 Text(0.14084507042253522, 0.55, 'gini = 0.491\nsamples = 17\nvalue = [13, 17]\nclass = N
o'),
 Text(0.19718309859154928, 0.55, 'gini = 0.434\nsamples = 12\nvalue = [15, 7]\nclass = Ye
s'),
 Text(0.5176056338028169, 0.75, 'ram <= 3768.5\ngini = 0.328\nsamples = 330\nvalue = [105, 4
02]\nclass = No'),
 Text(0.4014084507042254, 0.65, 'clock_speed <= 0.85\ngini = 0.311\nsamples = 303\nvalue =
[91, 381]\nclass = No'),
 Text(0.2535211267605634, 0.55, 'n_cores <= 2.5\ngini = 0.372\nsamples = 117\nvalue = [44, 1
34]\nclass = No'),
 Text(0.19718309859154928, 0.45, 'sc_w <= 3.5\ngini = 0.463\nsamples = 31\nvalue = [16, 28]
\nclass = No'),
 Text(0.16901408450704225, 0.35, 'gini = 0.188\nsamples = 13\nvalue = [2, 17]\nclass = No'),
 Text(0.22535211267605634, 0.35, 'gini = 0.493\nsamples = 18\nvalue = [14, 11]\nclass = Ye
s'),
 Text(0.30985915492957744, 0.45, 'battery_power <= 1719.0\ngini = 0.331\nsamples = 86\nvalue
= [28, 106]\nclass = No'),
 Text(0.28169014084507044, 0.35, 'mobile_wt <= 179.0\ngini = 0.231\nsamples = 68\nvalue = [1
4, 91]\nclass = No'),
 Text(0.2535211267605634, 0.25, 'm_dep <= 0.75\ngini = 0.127\nsamples = 57\nvalue = [6, 82]
\nclass = No'),
 Text(0.22535211267605634, 0.15, 'blue <= 0.5\ngini = 0.061\nsamples = 39\nvalue = [2, 62]\n
class = No'),
 Text(0.19718309859154928, 0.05, 'gini = 0.159\nsamples = 17\nvalue = [2, 21]\nclass = No'),
 Text(0.2535211267605634, 0.05, 'gini = 0.0\nsamples = 22\nvalue = [0, 41]\nclass = No'),
 Text(0.28169014084507044, 0.15, 'gini = 0.278\nsamples = 18\nvalue = [4, 20]\nclass = No'),
 Text(0.30985915492957744, 0.25, 'gini = 0.498\nsamples = 11\nvalue = [8, 9]\nclass = No'),
 Text(0.3380281690140845, 0.35, 'gini = 0.499\nsamples = 18\nvalue = [14, 15]\nclass = No'),
 Text(0.5492957746478874, 0.55, 'mobile_wt <= 181.5\ngini = 0.269\nsamples = 186\nvalue = [4
7, 247]\nclass = No'),
 Text(0.4788732394366197, 0.45, 'm_dep <= 0.85\ngini = 0.246\nsamples = 162\nvalue = [37, 22
1]\nclass = No'),
 Text(0.4225352112676056, 0.35, 'four_g <= 0.5\ngini = 0.197\nsamples = 135\nvalue = [24, 19
3]\nclass = No'),
 Text(0.39436619718309857, 0.25, 'sc_h <= 13.0\ngini = 0.405\nsamples = 56\nvalue = [24, 61]
\nclass = No'),
 Text(0.3380281690140845, 0.15, 'px_width <= 1331.5\ngini = 0.473\nsamples = 30\nvalue = [1
5, 24]\nclass = No'),
 Text(0.30985915492957744, 0.05, 'gini = 0.245\nsamples = 12\nvalue = [2, 12]\nclass = No'),
 Text(0.36619718309859156, 0.05, 'gini = 0.499\nsamples = 18\nvalue = [13, 12]\nclass = Ye
s'),
 Text(0.4507042253521127, 0.15, 'talk_time <= 8.5\ngini = 0.315\nsamples = 26\nvalue = [9, 3
7]\nclass = No'),
 Text(0.4225352112676056, 0.05, 'gini = 0.105\nsamples = 10\nvalue = [1, 17]\nclass = No'),
 Text(0.4788732394366197, 0.05, 'gini = 0.408\nsamples = 16\nvalue = [8, 20]\nclass = No'),
 Text(0.4507042253521127, 0.25, 'gini = 0.0\nsamples = 79\nvalue = [0, 132]\nclass = No'),
 Text(0.5352112676056338, 0.35, 'n_cores <= 3.5\ngini = 0.433\nsamples = 27\nvalue = [13, 2
8]\nclass = No'),
 Text(0.5070422535211268, 0.25, 'gini = 0.484\nsamples = 12\nvalue = [10, 7]\nclass = Yes'),
 Text(0.5633802816901409, 0.25, 'gini = 0.219\nsamples = 15\nvalue = [3, 21]\nclass = No'),
 Text(0.6197183098591549, 0.45, 'fc <= 2.5\ngini = 0.401\nsamples = 24\nvalue = [10, 26]\ncl
```

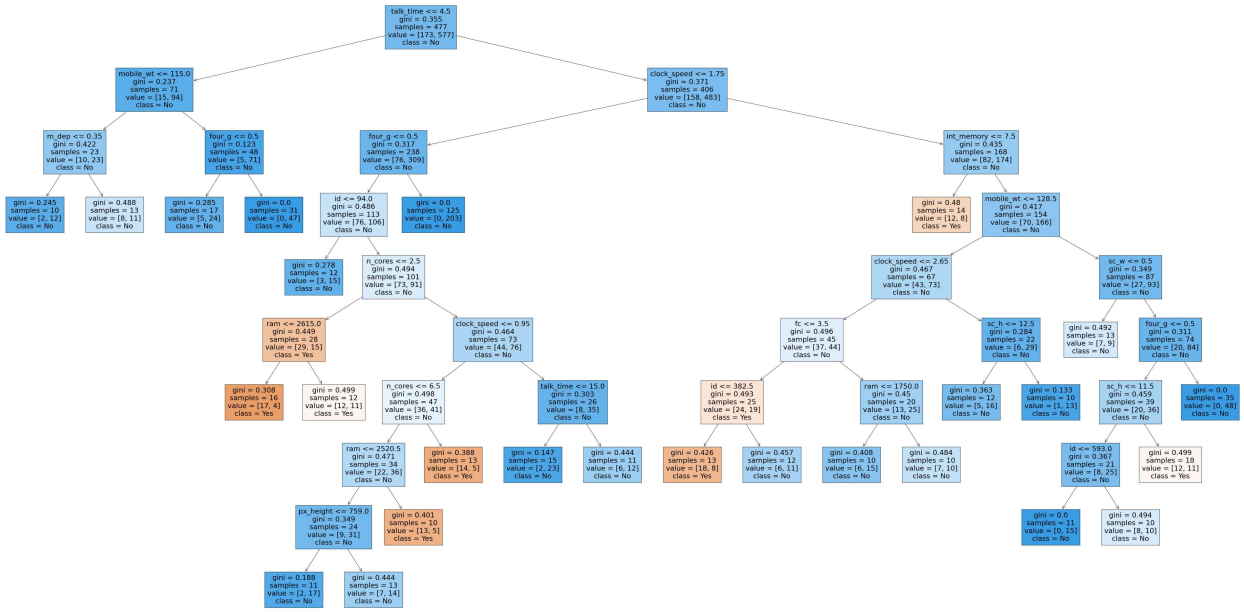
```
ass = No'),
 Text(0.5915492957746479, 0.35, 'gini = 0.0\nsamples = 11\nvalue = [0, 16]\nclass = No'),
 Text(0.647887323943662, 0.35, 'gini = 0.5\nsamples = 13\nvalue = [10, 10]\nclass = Yes'),
 Text(0.6338028169014085, 0.65, 'dual_sim <= 0.5\ngini = 0.48\nsamples = 27\nvalue = [14, 2
1]\nclass = No'),
 Text(0.6056338028169014, 0.55, 'gini = 0.498\nsamples = 12\nvalue = [8, 9]\nclass = No'),
 Text(0.6619718309859155, 0.55, 'gini = 0.444\nsamples = 15\nvalue = [6, 12]\nclass = No'),
 Text(0.7887323943661971, 0.85, 'id <= 378.5\ngini = 0.434\nsamples = 95\nvalue = [51, 109]
\nclass = No'),
 Text(0.7183098591549296, 0.75, 'px_width <= 1101.0\ngini = 0.498\nsamples = 33\nvalue = [2
9, 33]\nclass = No'),
 Text(0.6901408450704225, 0.65, 'gini = 0.308\nsamples = 11\nvalue = [17, 4]\nclass = Yes'),
 Text(0.7464788732394366, 0.65, 'four_g <= 0.5\ngini = 0.414\nsamples = 22\nvalue = [12, 29]
\nclass = No'),
 Text(0.7183098591549296, 0.55, 'gini = 0.5\nsamples = 11\nvalue = [12, 12]\nclass = Yes'),
 Text(0.7746478873239436, 0.55, 'gini = 0.0\nsamples = 11\nvalue = [0, 17]\nclass = No'),
 Text(0.8591549295774648, 0.75, 'sc_h <= 6.5\ngini = 0.348\nsamples = 62\nvalue = [22, 76]\n
class = No'),
 Text(0.8309859154929577, 0.65, 'gini = 0.49\nsamples = 13\nvalue = [9, 12]\nclass = No'),
 Text(0.8873239436619719, 0.65, 'sc_h <= 11.5\ngini = 0.281\nsamples = 49\nvalue = [13, 64]
\nclass = No'),
 Text(0.8309859154929577, 0.55, 'mobile_wt <= 145.5\ngini = 0.105\nsamples = 23\nvalue = [2,
34]\nclass = No'),
 Text(0.8028169014084507, 0.45, 'gini = 0.188\nsamples = 12\nvalue = [2, 17]\nclass = No'),
 Text(0.8591549295774648, 0.45, 'gini = 0.0\nsamples = 11\nvalue = [0, 17]\nclass = No'),
 Text(0.9436619718309859, 0.55, 'n_cores <= 4.5\ngini = 0.393\nsamples = 26\nvalue = [11, 3
0]\nclass = No'),
 Text(0.9154929577464789, 0.45, 'gini = 0.494\nsamples = 11\nvalue = [8, 10]\nclass = No'),
 Text(0.971830985915493, 0.45, 'gini = 0.227\nsamples = 15\nvalue = [3, 20]\nclass = No')]
```



In [38]:
```
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[38]:
```
[Text(0.3392857142857143, 0.95, 'talk_time <= 4.5\ngini = 0.355\nsamples = 477\nvalue = [173, 577]\nclass = No'),
 Text(0.12698412698412698, 0.85, 'mobile_wt <= 115.0\ngini = 0.237\nsamples = 71\nvalue = [15, 94]\nclass = No'),
 Text(0.06349206349206349, 0.75, 'm_dep <= 0.35\ngini = 0.422\nsamples = 23\nvalue = [10, 23]\nclass = No'),
 Text(0.031746031746031744, 0.65, 'gini = 0.245\nsamples = 10\nvalue = [2, 12]\nclass = No'),
 Text(0.09523809523809523, 0.65, 'gini = 0.488\nsamples = 13\nvalue = [8, 11]\nclass = No'),
 Text(0.19047619047619047, 0.75, 'four_g <= 0.5\ngini = 0.123\nsamples = 48\nvalue = [5, 71]\nclass = No'),
 Text(0.15873015873015872, 0.65, 'gini = 0.285\nsamples = 17\nvalue = [5, 24]\nclass = No'),
 Text(0.2222222222222222, 0.65, 'gini = 0.0\nsamples = 31\nvalue = [0, 47]\nclass = No'),
 Text(0.5515873015873016, 0.85, 'clock_speed <= 1.75\ngini = 0.371\nsamples = 406\nvalue = [158, 483]\nclass = No'),
 Text(0.31746031746031744, 0.75, 'four_g <= 0.5\ngini = 0.317\nsamples = 238\nvalue = [76, 309]\nclass = No'),
 Text(0.2857142857142857, 0.65, 'id <= 94.0\ngini = 0.486\nsamples = 113\nvalue = [76, 106]\nclass = No'),
 Text(0.25396825396825395, 0.55, 'gini = 0.278\nsamples = 12\nvalue = [3, 15]\nclass = No'),
 Text(0.31746031746031744, 0.55, 'n_cores <= 2.5\ngini = 0.494\nsamples = 101\nvalue = [73, 91]\nclass = No'),
 Text(0.23809523809523808, 0.45, 'ram <= 2615.0\ngini = 0.449\nsamples = 28\nvalue = [29, 15]\nclass = Yes'),
 Text(0.20634920634920634, 0.35, 'gini = 0.308\nsamples = 16\nvalue = [17, 4]\nclass = Yes'),
 Text(0.2698412698412698, 0.35, 'gini = 0.499\nsamples = 12\nvalue = [12, 11]\nclass = Yes'),
 Text(0.3968253968253968, 0.45, 'clock_speed <= 0.95\ngini = 0.464\nsamples = 73\nvalue = [44, 76]\nclass = No'),
 Text(0.3333333333333333, 0.35, 'n_cores <= 6.5\ngini = 0.498\nsamples = 47\nvalue = [36, 41]\nclass = No'),
 Text(0.30158730158730157, 0.25, 'ram <= 2520.5\ngini = 0.471\nsamples = 34\nvalue = [22, 36]\nclass = No'),
 Text(0.2698412698412698, 0.15, 'px_height <= 759.0\ngini = 0.349\nsamples = 24\nvalue = [9, 31]\nclass = No'),
 Text(0.23809523809523808, 0.05, 'gini = 0.188\nsamples = 11\nvalue = [2, 17]\nclass = No'),
 Text(0.30158730158730157, 0.05, 'gini = 0.444\nsamples = 13\nvalue = [7, 14]\nclass = No'),
 Text(0.3333333333333333, 0.15, 'gini = 0.401\nsamples = 10\nvalue = [13, 5]\nclass = Yes'),
 Text(0.36507936507936506, 0.25, 'gini = 0.388\nsamples = 13\nvalue = [14, 5]\nclass = Yes'),
 Text(0.4603174603174603, 0.35, 'talk_time <= 15.0\ngini = 0.303\nsamples = 26\nvalue = [8, 35]\nclass = No'),
 Text(0.42857142857142855, 0.25, 'gini = 0.147\nsamples = 15\nvalue = [2, 23]\nclass = No'),
 Text(0.49206349206349204, 0.25, 'gini = 0.444\nsamples = 11\nvalue = [6, 12]\nclass = No'),
 Text(0.3492063492063492, 0.65, 'gini = 0.0\nsamples = 125\nvalue = [0, 203]\nclass = No'),
 Text(0.7857142857142857, 0.75, 'int_memory <= 7.5\ngini = 0.435\nsamples = 168\nvalue = [82, 174]\nclass = No'),
 Text(0.753968253968254, 0.65, 'gini = 0.48\nsamples = 14\nvalue = [12, 8]\nclass = Yes'),
 Text(0.8174603174603174, 0.65, 'mobile_wt <= 128.5\ngini = 0.417\nsamples = 154\nvalue = [70, 166]\nclass = No'),
 Text(0.7301587301587301, 0.55, 'clock_speed <= 2.65\ngini = 0.467\nsamples = 67\nvalue = [43, 73]\nclass = No'),
 Text(0.6507936507936508, 0.45, 'fc <= 3.5\ngini = 0.496\nsamples = 45\nvalue = [37, 44]\nclass = No'),
 Text(0.5873015873015873, 0.35, 'id <= 382.5\ngini = 0.493\nsamples = 25\nvalue = [24, 19]\nclass = Yes'),
 Text(0.5555555555555556, 0.25, 'gini = 0.426\nsamples = 13\nvalue = [18, 8]\nclass = Yes'),
 Text(0.6190476190476191, 0.25, 'gini = 0.457\nsamples = 12\nvalue = [6, 11]\nclass = No'),
 Text(0.7142857142857143, 0.35, 'ram <= 1750.0\ngini = 0.45\nsamples = 20\nvalue = [13, 25]\nclass = No'),
 Text(0.6825396825396826, 0.25, 'gini = 0.408\nsamples = 10\nvalue = [6, 15]\nclass = No'),
 Text(0.746031746031746, 0.25, 'gini = 0.484\nsamples = 10\nvalue = [7, 10]\nclass = No'),
 Text(0.8095238095238095, 0.45, 'sc_h <= 12.5\ngini = 0.284\nsamples = 22\nvalue = [6, 29]\n
```

```
class = No'),
 Text(0.7777777777777778, 0.35, 'gini = 0.363\nsamples = 12\nvalue = [5, 16]\nclass = No'),
 Text(0.8412698412698413, 0.35, 'gini = 0.133\nsamples = 10\nvalue = [1, 13]\nclass = No'),
 Text(0.9047619047619048, 0.55, 'sc_w <= 0.5\ngini = 0.349\nsamples = 87\nvalue = [27, 93]\n
class = No'),
 Text(0.873015873015873, 0.45, 'gini = 0.492\nsamples = 13\nvalue = [7, 9]\nclass = No'),
 Text(0.9365079365079365, 0.45, 'four_g <= 0.5\ngini = 0.311\nsamples = 74\nvalue = [20, 84]
\nclass = No'),
 Text(0.9047619047619048, 0.35, 'sc_h <= 11.5\ngini = 0.459\nsamples = 39\nvalue = [20, 36]
\nclass = No'),
 Text(0.873015873015873, 0.25, 'id <= 593.0\ngini = 0.367\nsamples = 21\nvalue = [8, 25]\ncl
ass = No'),
 Text(0.8412698412698413, 0.15, 'gini = 0.0\nsamples = 11\nvalue = [0, 15]\nclass = No'),
 Text(0.9047619047619048, 0.15, 'gini = 0.494\nsamples = 10\nvalue = [8, 10]\nclass = No'),
 Text(0.9365079365079365, 0.25, 'gini = 0.499\nsamples = 18\nvalue = [12, 11]\nclass = Ye
s'),
 Text(0.9682539682539683, 0.35, 'gini = 0.0\nsamples = 35\nvalue = [0, 48]\nclass = No')]
```



In [39]:
```
rf_best.feature_importances_
```

Out[39]:
```
array([0.05057771, 0.05215466, 0.00371754, 0.03918893, 0.00539518,
       0.02279506, 0.4286357 , 0.03025779, 0.02295929, 0.05776009,
       0.02614634, 0.02236943, 0.04664862, 0.04563953, 0.04545766,
       0.04185337, 0.02128077, 0.02849118, 0.00341511, 0.00525604])
```

In [40]:
```
imp_df=pd.DataFrame({"varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[40]:

	varname	Imp
6	four_g	0.428636
9	mobile_wt	0.057760
1	battery_power	0.052155
0	id	0.050578
12	px_height	0.046649
13	px_width	0.045640
14	ram	0.045458
15	sc_h	0.041853
3	clock_speed	0.039189
7	int_memory	0.030258
17	talk_time	0.028491
10	n_cores	0.026146
8	m_dep	0.022959
5	fc	0.022795
11	pc	0.022369
16	sc_w	0.021281
4	dual_sim	0.005395
19	wifi	0.005256
2	blue	0.003718
18	touch_screen	0.003415