In [14]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt,seaborn as sns
```

In [15]:
```
train_df = pd.read_csv(r"C:\Users\sudheer\Downloads\Mobile_Price_Classification_train (1).csv")
train_df
```

Out[15]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483

2000 rows × 21 columns

In [16]:
```
test_df = pd.read_csv(r"C:\Users\sudheer\Downloads\Mobile_Price_Classification_test (1).csv")
test_df
```

Out[16]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	644
996	997	609	0	1.8	1	0	0	13	0.9	186	...	2	1152
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	477
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	38
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	457

1000 rows × 21 columns

In [17]:

```
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 [18]:

```
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 [19]:
```
x=train_df.drop('wifi',axis=1)
y=train_df['wifi']
```

In [20]:
```
x=test_df.drop('wifi',axis=1)
y=test_df['wifi']
```

In [21]:
```
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[21]:
```
((700, 20), (300, 20))
```

In [22]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[22]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [23]:
```
rf = RandomForestClassifier()
```

In [24]:
```
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 [25]:
```
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[25]:
```
▸           GridSearchCV
▸ estimator: RandomForestClassifier
      ▸ RandomForestClassifier
```

In [26]:
```
grid_search.best_score_
```

Out[26]:
```
0.5514285714285714
```

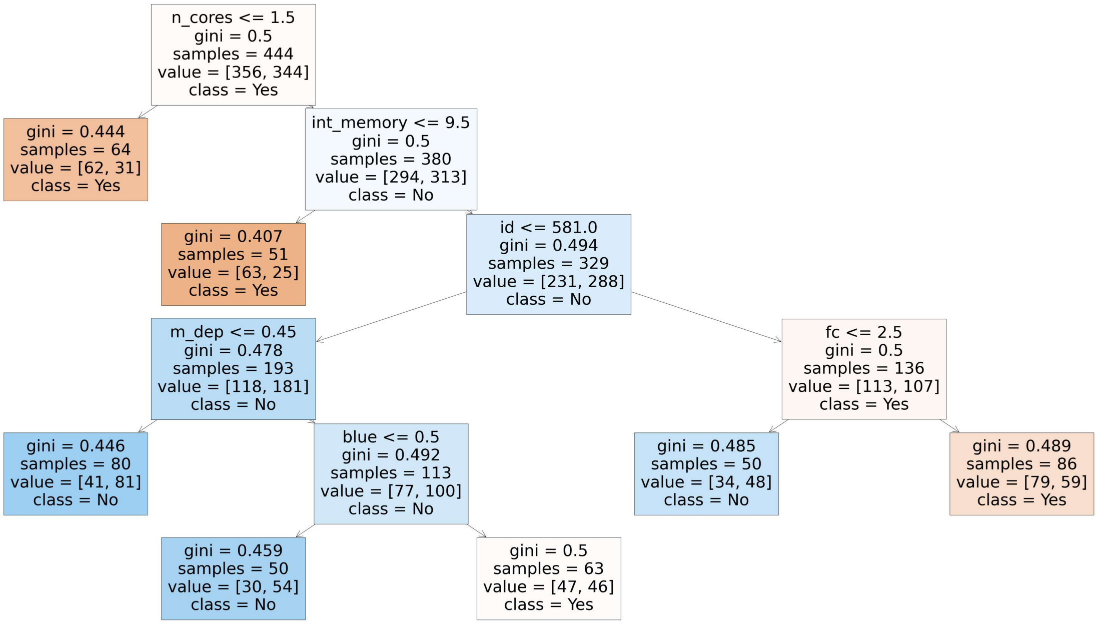
In [27]:
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[27]:
```
▾                     RandomForestClassifier

RandomForestClassifier(max_depth=20, min_samples_leaf=50)
```

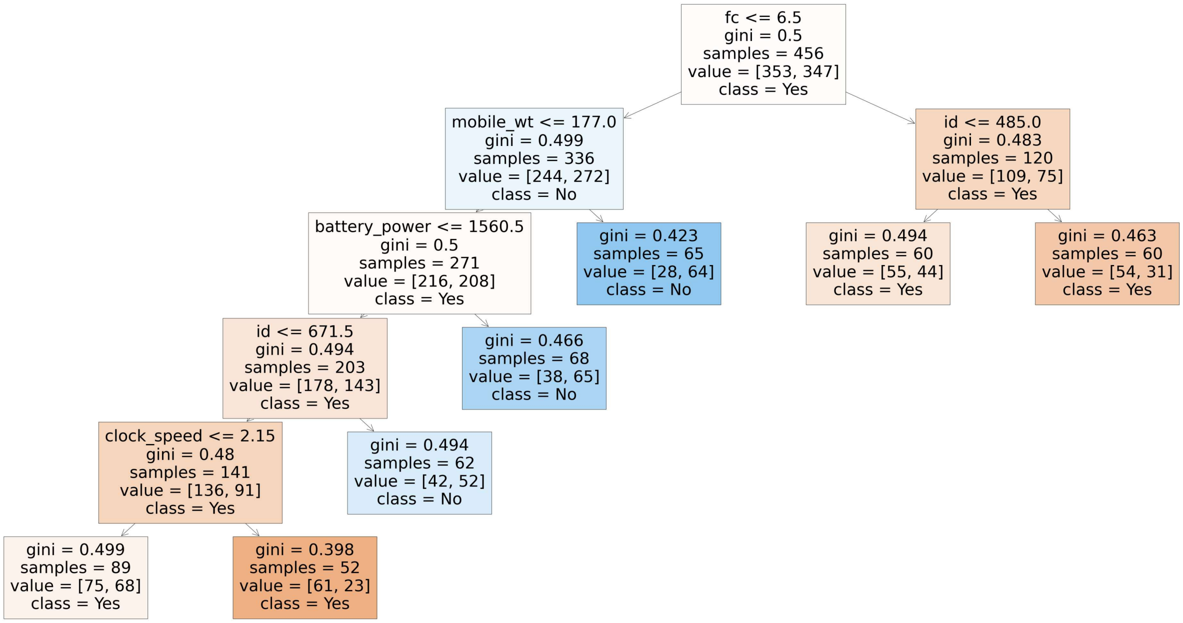
```
In [28]: from sklearn.tree import plot_tree
         plt.figure(figsize=(80,40))
         plot_tree(rf_best.estimators_[5], feature_names = x.columns,class_names=['Yes',"No"],filled=True
```

```
Out[28]: [Text(0.25, 0.9166666666666666, 'n_cores <= 1.5\ngini = 0.5\nsamples = 444\nvalue = [356, 344]
         \nclass = Yes'),
          Text(0.125, 0.75, 'gini = 0.444\nsamples = 64\nvalue = [62, 31]\nclass = Yes'),
          Text(0.375, 0.75, 'int_memory <= 9.5\ngini = 0.5\nsamples = 380\nvalue = [294, 313]\nclass = N
         o'),
          Text(0.25, 0.5833333333333334, 'gini = 0.407\nsamples = 51\nvalue = [63, 25]\nclass = Yes'),
          Text(0.5, 0.5833333333333334, 'id <= 581.0\ngini = 0.494\nsamples = 329\nvalue = [231, 288]\nc
         lass = No'),
          Text(0.25, 0.4166666666666667, 'm_dep <= 0.45\ngini = 0.478\nsamples = 193\nvalue = [118, 181]
         \nclass = No'),
          Text(0.125, 0.25, 'gini = 0.446\nsamples = 80\nvalue = [41, 81]\nclass = No'),
          Text(0.375, 0.25, 'blue <= 0.5\ngini = 0.492\nsamples = 113\nvalue = [77, 100]\nclass = No'),
          Text(0.25, 0.08333333333333333, 'gini = 0.459\nsamples = 50\nvalue = [30, 54]\nclass = No'),
          Text(0.5, 0.08333333333333333, 'gini = 0.5\nsamples = 63\nvalue = [47, 46]\nclass = Yes'),
          Text(0.75, 0.4166666666666667, 'fc <= 2.5\ngini = 0.5\nsamples = 136\nvalue = [113, 107]\nclas
         s = Yes'),
          Text(0.625, 0.25, 'gini = 0.485\nsamples = 50\nvalue = [34, 48]\nclass = No'),
          Text(0.875, 0.25, 'gini = 0.489\nsamples = 86\nvalue = [79, 59]\nclass = Yes')]
```



In [29]:
```python
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[29]:
```
[Text(0.6363636363636364, 0.9166666666666666, 'fc <= 6.5\ngini = 0.5\nsamples = 456\nvalue = [3
53, 347]\nclass = Yes'),
 Text(0.45454545454545453, 0.75, 'mobile_wt <= 177.0\ngini = 0.499\nsamples = 336\nvalue = [24
4, 272]\nclass = No'),
 Text(0.36363636363636365, 0.5833333333333334, 'battery_power <= 1560.5\ngini = 0.5\nsamples =
271\nvalue = [216, 208]\nclass = Yes'),
 Text(0.2727272727272727, 0.4166666666666667, 'id <= 671.5\ngini = 0.494\nsamples = 203\nvalue
= [178, 143]\nclass = Yes'),
 Text(0.18181818181818182, 0.25, 'clock_speed <= 2.15\ngini = 0.48\nsamples = 141\nvalue = [13
6, 91]\nclass = Yes'),
 Text(0.09090909090909091, 0.08333333333333333, 'gini = 0.499\nsamples = 89\nvalue = [75, 68]\n
class = Yes'),
 Text(0.2727272727272727, 0.08333333333333333, 'gini = 0.398\nsamples = 52\nvalue = [61, 23]\nc
lass = Yes'),
 Text(0.36363636363636365, 0.25, 'gini = 0.494\nsamples = 62\nvalue = [42, 52]\nclass = No'),
 Text(0.45454545454545453, 0.4166666666666667, 'gini = 0.466\nsamples = 68\nvalue = [38, 65]\nc
lass = No'),
 Text(0.5454545454545454, 0.5833333333333334, 'gini = 0.423\nsamples = 65\nvalue = [28, 64]\ncl
ass = No'),
 Text(0.8181818181818182, 0.75, 'id <= 485.0\ngini = 0.483\nsamples = 120\nvalue = [109, 75]\nc
lass = Yes'),
 Text(0.7272727272727273, 0.5833333333333334, 'gini = 0.494\nsamples = 60\nvalue = [55, 44]\ncl
ass = Yes'),
 Text(0.9090909090909091, 0.5833333333333334, 'gini = 0.463\nsamples = 60\nvalue = [54, 31]\ncl
ass = Yes')]
```



In [30]:
```python
rf_best.feature_importances_
```

Out[30]:
```
array([0.06031287, 0.07588404, 0.01249582, 0.08418056, 0.01220114,
       0.05887866, 0.02418764, 0.07420338, 0.06991117, 0.12339559,
       0.01764515, 0.04971102, 0.06579852, 0.10809968, 0.06200155,
       0.02275696, 0.02744152, 0.04592882, 0.00219524, 0.00277066])
```

In [31]:
```
imp_df=pd.DataFrame({'Varname':x_train.columns,'Imp':rf_best.feature_importances_})
imp_df.sort_values(by='Imp',ascending=False)
```

Out[31]:
```
         Varname       Imp
 9     mobile_wt  0.123396
13      px_width  0.108100
 3   clock_speed  0.084181
 1 battery_power  0.075884
 7    int_memory  0.074203
 8         m_dep  0.069911
12     px_height  0.065799
14           ram  0.062002
 0            id  0.060313
 5            fc  0.058879
11            pc  0.049711
17     talk_time  0.045929
16          sc_w  0.027442
 6        four_g  0.024188
15          sc_h  0.022757
10       n_cores  0.017645
 2          blue  0.012496
 4      dual_sim  0.012201
19  touch_screen  0.002771
18       three_g  0.002195
```

In [ ]: