Random Forest

In [24]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [25]:
```
test_df=pd.read_csv(r"C:\Users\sneha\Downloads\Mobile_Price_Classification_test.csv")
test_df
```

Out[25]:

	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 [26]:
```
train_df=pd.read_csv(r"C:\Users\sneha\Downloads\Mobile_Price_Classification_train.csv")
train_df
```

Out[26]:

	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 [27]:
```
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 [28]:
```
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 [29]:

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

In [30]:

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

In [31]:

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

Out[31]:

```
dual_sim
1    1019
0     981
Name: count, dtype: int64
```

In [32]:

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

Out[32]:

```
blue
1    516
0    484
Name: count, dtype: int64
```

In [33]:

```
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[33]:

```
((700, 20), (300, 20))
```

In [34]:

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

Out[34]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [35]:

```
rf = RandomForestClassifier()
```

In [36]:

```
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 [37]:

```
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[37]:

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

In [38]:

```
grid_search.best_score_
```

Out[38]:

```
0.5557142857142857
```

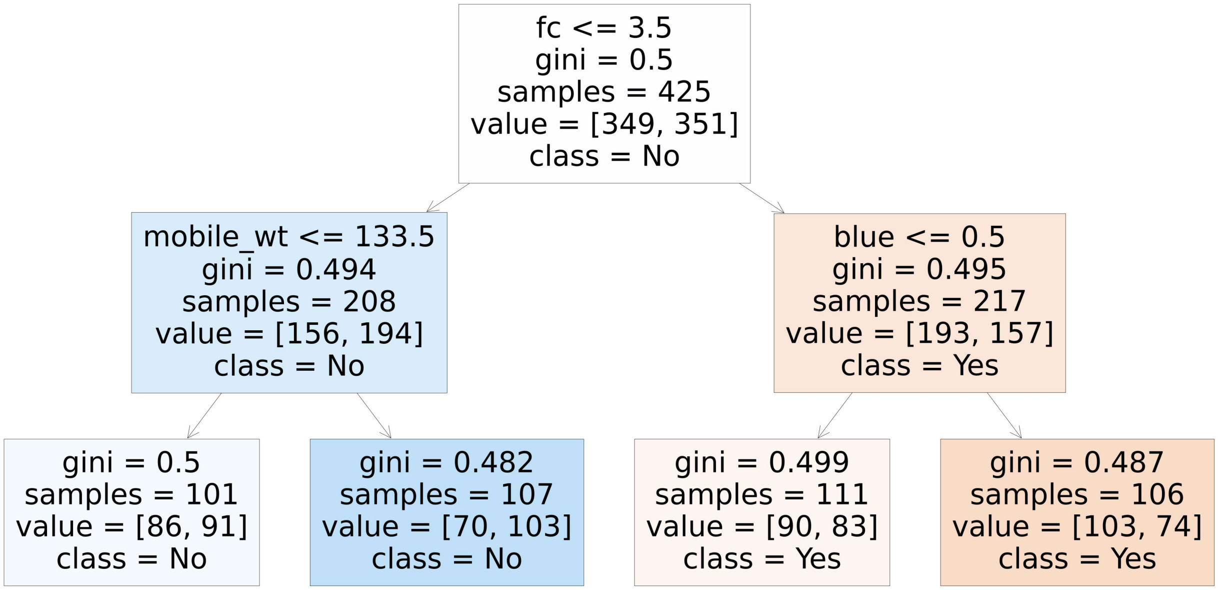
In [39]:
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[39]:
```
▾                     RandomForestClassifier

RandomForestClassifier(max_depth=20, min_samples_leaf=100, n_estimators=30)
```

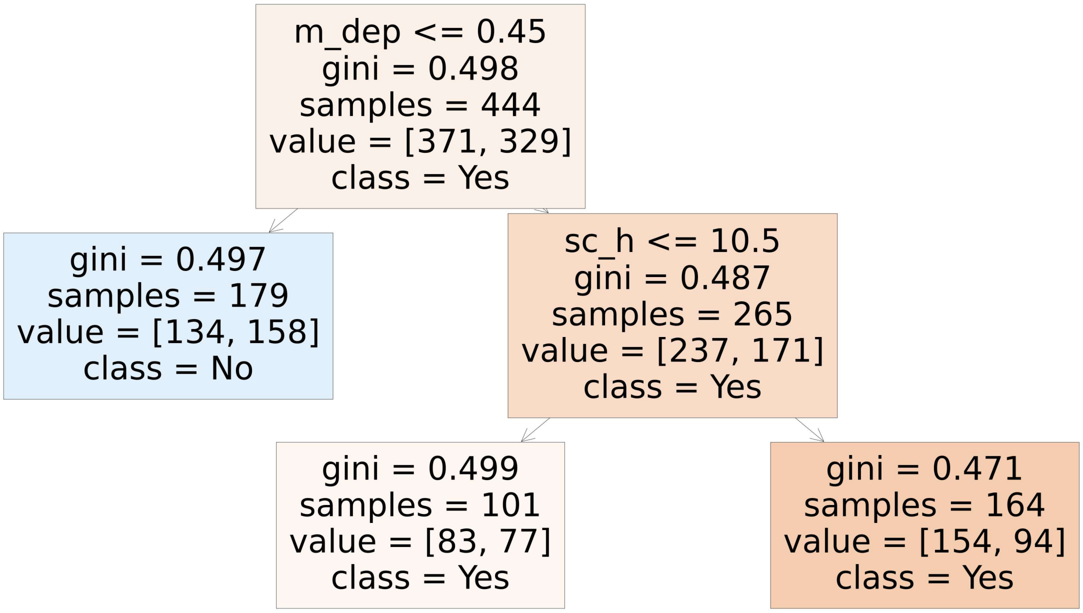
In [40]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5], feature_names = x.columns,class_names=['Yes',"No"],filled=True
```

Out[40]:
```
[Text(0.5, 0.8333333333333334, 'fc <= 3.5\ngini = 0.5\nsamples = 425\nvalue = [349, 351]\nclass
= No'),
 Text(0.25, 0.5, 'mobile_wt <= 133.5\ngini = 0.494\nsamples = 208\nvalue = [156, 194]\nclass =
No'),
 Text(0.125, 0.16666666666666666, 'gini = 0.5\nsamples = 101\nvalue = [86, 91]\nclass = No'),
 Text(0.375, 0.16666666666666666, 'gini = 0.482\nsamples = 107\nvalue = [70, 103]\nclass = N
o'),
 Text(0.75, 0.5, 'blue <= 0.5\ngini = 0.495\nsamples = 217\nvalue = [193, 157]\nclass = Yes'),
 Text(0.625, 0.16666666666666666, 'gini = 0.499\nsamples = 111\nvalue = [90, 83]\nclass = Ye
s'),
 Text(0.875, 0.16666666666666666, 'gini = 0.487\nsamples = 106\nvalue = [103, 74]\nclass = Ye
s')]
```



In [41]:
```
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[41]:
```
[Text(0.4, 0.8333333333333334, 'm_dep <= 0.45\ngini = 0.498\nsamples = 444\nvalue = [371, 329]
\nclass = Yes'),
 Text(0.2, 0.5, 'gini = 0.497\nsamples = 179\nvalue = [134, 158]\nclass = No'),
 Text(0.6, 0.5, 'sc_h <= 10.5\ngini = 0.487\nsamples = 265\nvalue = [237, 171]\nclass = Yes'),
 Text(0.4, 0.16666666666666666, 'gini = 0.499\nsamples = 101\nvalue = [83, 77]\nclass = Yes'),
 Text(0.8, 0.16666666666666666, 'gini = 0.471\nsamples = 164\nvalue = [154, 94]\nclass = Yes')]
```



In [42]:
```
rf_best.feature_importances_
```

Out[42]:
```
array([0.06286367, 0.03353052, 0.01478924, 0.07447493, 0.01598226,
       0.08606662, 0.03150052, 0.10674868, 0.02865005, 0.11403191,
       0.        , 0.02335593, 0.06032871, 0.1677183 , 0.08599477,
       0.03850082, 0.01934232, 0.03612077, 0.        , 0.        ])
```

In [43]:
```
imp_df=pd.DataFrame({'Varname':x_train.columns,'Imp':rf_best.feature_importances_})
imp_df.sort_values(by='Imp',ascending=False)
```

Out[43]:

	Varname	Imp
13	px_width	0.167718
9	mobile_wt	0.114032
7	int_memory	0.106749
5	fc	0.086067
14	ram	0.085995
3	clock_speed	0.074475
0	id	0.062864
12	px_height	0.060329
15	sc_h	0.038501
17	talk_time	0.036121
1	battery_power	0.033531
6	four_g	0.031501
8	m_dep	0.028650
11	pc	0.023356
16	sc_w	0.019342
4	dual_sim	0.015982
2	blue	0.014789
18	three_g	0.000000
10	n_cores	0.000000
19	touch_screen	0.000000

In [ ]:

```
```