RANDOM FOREST

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

In [2]:
```
train_df=pd.read_csv(r"C:\Users\Dell\Downloads\Mobile_Price_Classification_train.csv")
train_df
```

Out[2]:

	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 [4]:
```
test_df=pd.read_csv(r"C:\Users\Dell\Downloads\Mobile_Price_Classification_test.csv")
test_df
```

Out[4]:

	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 [5]:
```
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 [6]:
```
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 [7]:
```
x=train_df.drop('wifi',axis=1)
y=train_df['wifi']
```

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

In [9]:
```
train_df['dual_sim'].value_counts()
```

Out[9]:
```
1    1019
0     981
Name: dual_sim, dtype: int64
```

In [10]:
```
test_df['blue'].value_counts()
```

Out[10]:
```
1    516
0    484
Name: blue, dtype: int64
```

In [11]:
```
T={"Home Owner":{"Yes":1,"No":0}}
train_df=train_df.replace(T)
print(train_df)
```

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
0               842     0          2.2         0   1       0           7
1              1021     1          0.5         1   0       1          53
2               563     1          0.5         1   2       1          41
3               615     1          2.5         0   0       0          10
4              1821     1          1.2         0  13       1          44
...             ...   ...          ...       ...  ..     ...         ...
1995            794     1          0.5         1   0       1           2
1996           1965     1          2.6         1   0       0          39
1997           1911     0          0.9         1   1       1          36
1998           1512     0          0.9         0   4       1          46
1999            510     1          2.0         1   5       1          45

      m_dep  mobile_wt  n_cores  ...  px_height  px_width   ram  sc_h  sc_w  \
0       0.6        188        2  ...         20       756  2549     9     7
1       0.7        136        3  ...        905      1988  2631    17     3
2       0.9        145        5  ...       1263      1716  2603    11     2
3       0.8        131        6  ...       1216      1786  2769    16     8
4       0.6        141        2  ...       1208      1212  1411     8     2
...     ...        ...      ...  ...        ...       ...   ...   ...   ...
1995    0.8        106        6  ...       1222      1890   668    13     4
1996    0.2        187        4  ...        915      1965  2032    11    10
1997    0.7        108        8  ...        868      1632  3057     9     1
1998    0.1        145        5  ...        336       670   869    18    10
1999    0.9        168        6  ...        483       754  3919    19     4

      talk_time  three_g  touch_screen  wifi  price_range
0            19        0             0     1            1
1             7        1             1     0            2
2             9        1             1     0            2
3            11        1             0     0            2
4            15        1             1     0            1
...         ...      ...           ...   ...          ...
1995         19        1             1     0            0
1996         16        1             1     1            2
1997          5        1             1     0            3
1998         19        1             1     1            0
1999          2        1             1     1            3

[2000 rows x 21 columns]
```

In [12]:
```
T={"Home Owner":{"Yes":1,"No":0}}
test_df=test_df.replace(T)
print(test_df)
```

```
       id  battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
0       1           1043     1          1.8         1  14       0           5
1       2            841     1          0.5         1   4       1          61
2       3           1807     1          2.8         0   1       0          27
3       4           1546     0          0.5         1  18       1          25
4       5           1434     0          1.4         0  11       1          49
..    ...            ...   ...          ...       ...  ..     ...         ...
995   996           1700     1          1.9         0   0       1          54
996   997            609     0          1.8         1   0       0          13
997   998           1185     0          1.4         0   1       1           8
998   999           1533     1          0.5         1   0       0          50
999  1000           1270     1          0.5         0   4       1          35

     m_dep  mobile_wt  ...  pc  px_height  px_width   ram  sc_h  sc_w  \
0      0.1        193  ...  16        226      1412  3476    12     7
1      0.8        191  ...  12        746       857  3895     6     0
2      0.9        186  ...   4       1270      1366  2396    17    10
3      0.5         96  ...  20        295      1752  3893    10     0
4      0.5        108  ...  18        749       810  1773    15     8
..     ...        ...  ...  ..        ...       ...   ...   ...   ...
995    0.5        170  ...  17        644       913  2121    14     8
996    0.9        186  ...   2       1152      1632  1933     8     1
997    0.5         80  ...  12        477       825  1223     5     0
998    0.4        171  ...  12         38       832  2509    15    11
999    0.1        140  ...  19        457       608  2828     9     2

     talk_time  three_g  touch_screen  wifi
0            2        0             1     0
1            7        1             0     0
2           10        0             1     1
3            7        1             1     0
4            7        1             0     1
..         ...      ...           ...   ...
995         15        1             1     0
996         19        0             1     1
997         14        1             0     0
998          6        0             1     0
999          3        1             0     1

[1000 rows x 21 columns]
```

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

In [14]:
```
x=test_df.drop('wifi',axis=1)
y=test_df['wifi']
```

In [15]:
```
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[15]:
```
((700, 20), (300, 20))
```

In [16]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[16]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [17]:
```
rf = RandomForestClassifier()
```

In [18]:
```
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 [19]:
```
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[19]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [20]:
```
grid_search.best_score_
```

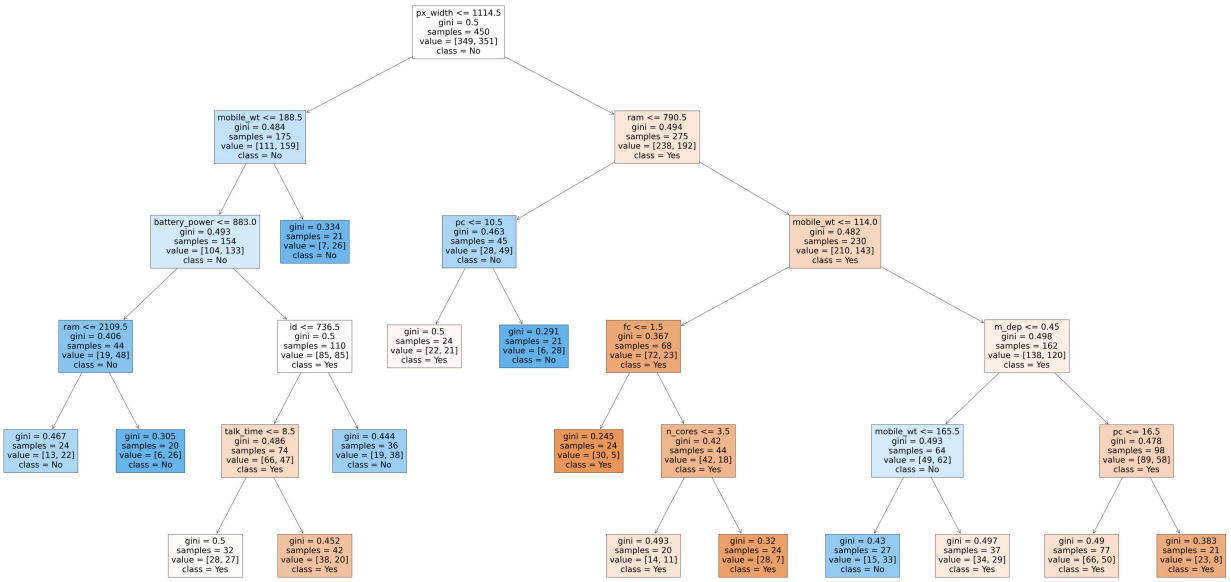
Out[20]:
```
0.5700000000000001
```

In [26]:
```
rf_best = grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=5, min_samples_leaf=20, n_estimators=10)
```

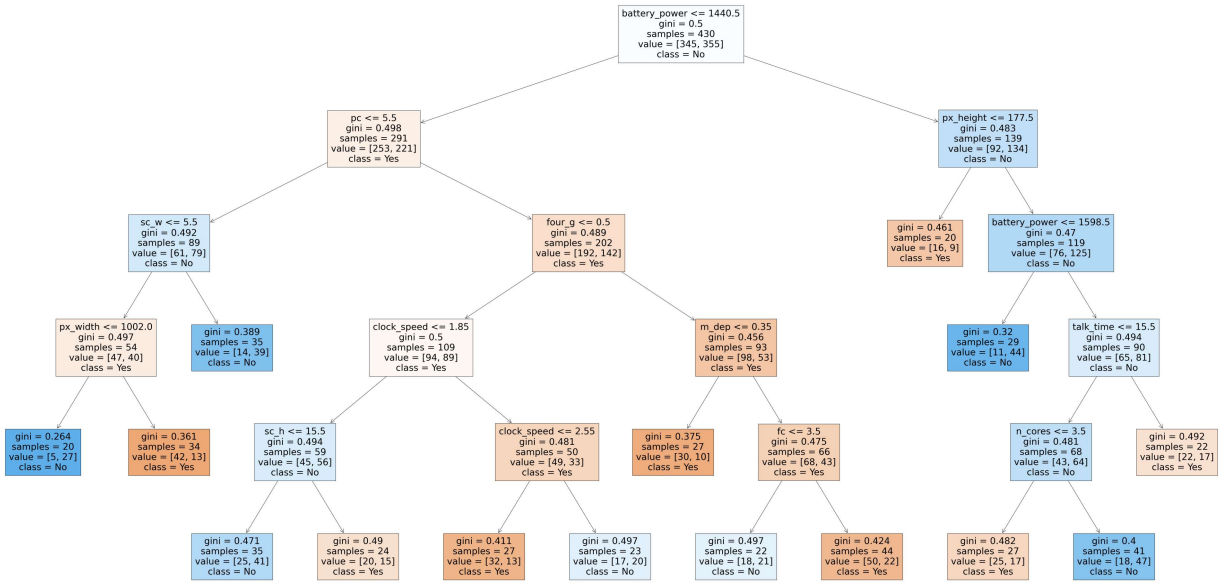
In [27]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names = x.columns,class_names=['Yes',"No"],filled=True
```

Out[27]:
```
[Text(0.375, 0.9166666666666666, 'px_width <= 1114.5\ngini = 0.5\nsamples = 450\nvalue = [349,
351]\nclass = No'),
 Text(0.21739130434782608, 0.75, 'mobile_wt <= 188.5\ngini = 0.484\nsamples = 175\nvalue = [11
1, 159]\nclass = No'),
 Text(0.17391304347826086, 0.5833333333333334, 'battery_power <= 883.0\ngini = 0.493\nsamples
= 154\nvalue = [104, 133]\nclass = No'),
 Text(0.08695652173913043, 0.4166666666666667, 'ram <= 2109.5\ngini = 0.406\nsamples = 44\nval
ue = [19, 48]\nclass = No'),
 Text(0.043478260869565216, 0.25, 'gini = 0.467\nsamples = 24\nvalue = [13, 22]\nclass = No'),
 Text(0.13043478260869565, 0.25, 'gini = 0.305\nsamples = 20\nvalue = [6, 26]\nclass = No'),
 Text(0.2608695652173913, 0.4166666666666667, 'id <= 736.5\ngini = 0.5\nsamples = 110\nvalue =
[85, 85]\nclass = Yes'),
 Text(0.21739130434782608, 0.25, 'talk_time <= 8.5\ngini = 0.486\nsamples = 74\nvalue = [66, 4
7]\nclass = Yes'),
 Text(0.17391304347826086, 0.08333333333333333, 'gini = 0.5\nsamples = 32\nvalue = [28, 27]\nc
lass = Yes'),
 Text(0.2608695652173913, 0.08333333333333333, 'gini = 0.452\nsamples = 42\nvalue = [38, 20]\n
class = Yes'),
 Text(0.30434782608695654, 0.25, 'gini = 0.444\nsamples = 36\nvalue = [19, 38]\nclass = No'),
 Text(0.2608695652173913, 0.5833333333333334, 'gini = 0.334\nsamples = 21\nvalue = [7, 26]\ncl
ass = No'),
 Text(0.532608695652174, 0.75, 'ram <= 790.5\ngini = 0.494\nsamples = 275\nvalue = [238, 192]
\nclass = Yes'),
 Text(0.391304347826087, 0.5833333333333334, 'pc <= 10.5\ngini = 0.463\nsamples = 45\nvalue =
[28, 49]\nclass = No'),
 Text(0.34782608695652173, 0.4166666666666667, 'gini = 0.5\nsamples = 24\nvalue = [22, 21]\ncl
ass = Yes'),
 Text(0.43478260869565216, 0.4166666666666667, 'gini = 0.291\nsamples = 21\nvalue = [6, 28]\nc
lass = No'),
 Text(0.6739130434782609, 0.5833333333333334, 'mobile_wt <= 114.0\ngini = 0.482\nsamples = 230
\nvalue = [210, 143]\nclass = Yes'),
 Text(0.5217391304347826, 0.4166666666666667, 'fc <= 1.5\ngini = 0.367\nsamples = 68\nvalue =
[72, 23]\nclass = Yes'),
 Text(0.4782608695652174, 0.25, 'gini = 0.245\nsamples = 24\nvalue = [30, 5]\nclass = Yes'),
 Text(0.5652173913043478, 0.25, 'n_cores <= 3.5\ngini = 0.42\nsamples = 44\nvalue = [42, 18]\n
class = Yes'),
 Text(0.5217391304347826, 0.08333333333333333, 'gini = 0.493\nsamples = 20\nvalue = [14, 11]\n
class = Yes'),
 Text(0.6086956521739131, 0.08333333333333333, 'gini = 0.32\nsamples = 24\nvalue = [28, 7]\ncl
ass = Yes'),
 Text(0.8260869565217391, 0.4166666666666667, 'm_dep <= 0.45\ngini = 0.498\nsamples = 162\nval
ue = [138, 120]\nclass = Yes'),
 Text(0.7391304347826086, 0.25, 'mobile_wt <= 165.5\ngini = 0.493\nsamples = 64\nvalue = [49,
62]\nclass = No'),
 Text(0.6956521739130435, 0.08333333333333333, 'gini = 0.43\nsamples = 27\nvalue = [15, 33]\nc
lass = No'),
 Text(0.782608695652174, 0.08333333333333333, 'gini = 0.497\nsamples = 37\nvalue = [34, 29]\nc
lass = Yes'),
 Text(0.9130434782608695, 0.25, 'pc <= 16.5\ngini = 0.478\nsamples = 98\nvalue = [89, 58]\ncla
ss = Yes'),
 Text(0.8695652173913043, 0.08333333333333333, 'gini = 0.49\nsamples = 77\nvalue = [66, 50]\nc
lass = Yes'),
 Text(0.9565217391304348, 0.08333333333333333, 'gini = 0.383\nsamples = 21\nvalue = [23, 8]\nc
lass = Yes')]
```


px_width <= 1114.5
gini = 0.5
samples = 450
value = [349, 351]
class = No
mobile_wt <= 188.5
gini = 0.484
samples = 175
value = [111, 159]
class = No
ram <= 790.5
gini = 0.494
samples = 275
value = [238, 192]
class = Yes
battery_power <= 883.0
gini = 0.493
samples = 154
value = [104, 133]
class = No
gini = 0.334
samples = 21
value = [7, 26]
class = No
pc <= 10.5
gini = 0.463
samples = 45
value = [28, 49]
class = No
mobile_wt <= 114.0
gini = 0.482
samples = 230
value = [210, 143]
class = Yes
ram <= 2109.5
gini = 0.406
samples = 44
value = [19, 48]
class = No
id <= 736.5
gini = 0.5
samples = 110
value = [85, 85]
class = Yes
gini = 0.5
samples = 24
value = [22, 21]
class = Yes
gini = 0.291
samples = 21
value = [6, 28]
class = No
fc <= 1.5
gini = 0.367
samples = 68
value = [72, 23]
class = Yes
m_dep <= 0.45
gini = 0.498
samples = 162
value = [138, 120]
class = Yes
gini = 0.467
samples = 24
value = [13, 22]
class = No
gini = 0.305
samples = 20
value = [6, 26]
class = No
talk_time <= 8.5
gini = 0.486
samples = 74
value = [66, 47]
class = Yes
gini = 0.444
samples = 36
value = [19, 38]
class = No
gini = 0.245
samples = 24
value = [30, 5]
class = Yes
n_cores <= 3.5
gini = 0.42
samples = 44
value = [42, 18]
class = Yes
mobile_wt <= 165.5
gini = 0.493
samples = 64
value = [49, 62]
class = No
pc <= 16.5
gini = 0.478
samples = 98
value = [89, 58]
class = Yes
gini = 0.5
samples = 32
value = [28, 27]
class = Yes
gini = 0.452
samples = 42
value = [38, 20]
class = Yes
gini = 0.493
samples = 20
value = [14, 11]
class = Yes
gini = 0.32
samples = 24
value = [28, 7]
class = Yes
gini = 0.43
samples = 27
value = [15, 33]
class = No
gini = 0.497
samples = 37
value = [34, 29]
class = Yes
gini = 0.49
samples = 77
value = [66, 50]
class = Yes
gini = 0.383
samples = 21
value = [23, 8]
class = Yes

In [28]:
```
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[28]:
```
[Text(0.55625, 0.9166666666666666, 'battery_power <= 1440.5\ngini = 0.5\nsamples = 430\nvalue
= [345, 355]\nclass = No'),
 Text(0.3125, 0.75, 'pc <= 5.5\ngini = 0.498\nsamples = 291\nvalue = [253, 221]\nclass = Ye
s'),
 Text(0.15, 0.5833333333333334, 'sc_w <= 5.5\ngini = 0.492\nsamples = 89\nvalue = [61, 79]\ncl
ass = No'),
 Text(0.1, 0.4166666666666667, 'px_width <= 1002.0\ngini = 0.497\nsamples = 54\nvalue = [47, 4
0]\nclass = Yes'),
 Text(0.05, 0.25, 'gini = 0.264\nsamples = 20\nvalue = [5, 27]\nclass = No'),
 Text(0.15, 0.25, 'gini = 0.361\nsamples = 34\nvalue = [42, 13]\nclass = Yes'),
 Text(0.2, 0.4166666666666667, 'gini = 0.389\nsamples = 35\nvalue = [14, 39]\nclass = No'),
 Text(0.475, 0.5833333333333334, 'four_g <= 0.5\ngini = 0.489\nsamples = 202\nvalue = [192, 14
2]\nclass = Yes'),
 Text(0.35, 0.4166666666666667, 'clock_speed <= 1.85\ngini = 0.5\nsamples = 109\nvalue = [94,
89]\nclass = Yes'),
 Text(0.25, 0.25, 'sc_h <= 15.5\ngini = 0.494\nsamples = 59\nvalue = [45, 56]\nclass = No'),
 Text(0.2, 0.08333333333333333, 'gini = 0.471\nsamples = 35\nvalue = [25, 41]\nclass = No'),
 Text(0.3, 0.08333333333333333, 'gini = 0.49\nsamples = 24\nvalue = [20, 15]\nclass = Yes'),
 Text(0.45, 0.25, 'clock_speed <= 2.55\ngini = 0.481\nsamples = 50\nvalue = [49, 33]\nclass =
Yes'),
 Text(0.4, 0.08333333333333333, 'gini = 0.411\nsamples = 27\nvalue = [32, 13]\nclass = Yes'),
 Text(0.5, 0.08333333333333333, 'gini = 0.497\nsamples = 23\nvalue = [17, 20]\nclass = No'),
 Text(0.6, 0.4166666666666667, 'm_dep <= 0.35\ngini = 0.456\nsamples = 93\nvalue = [98, 53]\nc
lass = Yes'),
 Text(0.55, 0.25, 'gini = 0.375\nsamples = 27\nvalue = [30, 10]\nclass = Yes'),
 Text(0.65, 0.25, 'fc <= 3.5\ngini = 0.475\nsamples = 66\nvalue = [68, 43]\nclass = Yes'),
 Text(0.6, 0.08333333333333333, 'gini = 0.497\nsamples = 22\nvalue = [18, 21]\nclass = No'),
 Text(0.7, 0.08333333333333333, 'gini = 0.424\nsamples = 44\nvalue = [50, 22]\nclass = Yes'),
 Text(0.8, 0.75, 'px_height <= 177.5\ngini = 0.483\nsamples = 139\nvalue = [92, 134]\nclass =
No'),
 Text(0.75, 0.5833333333333334, 'gini = 0.461\nsamples = 20\nvalue = [16, 9]\nclass = Yes'),
 Text(0.85, 0.5833333333333334, 'battery_power <= 1598.5\ngini = 0.47\nsamples = 119\nvalue =
[76, 125]\nclass = No'),
 Text(0.8, 0.4166666666666667, 'gini = 0.32\nsamples = 29\nvalue = [11, 44]\nclass = No'),
 Text(0.9, 0.4166666666666667, 'talk_time <= 15.5\ngini = 0.494\nsamples = 90\nvalue = [65, 8
1]\nclass = No'),
 Text(0.85, 0.25, 'n_cores <= 3.5\ngini = 0.481\nsamples = 68\nvalue = [43, 64]\nclass = No'),
 Text(0.8, 0.08333333333333333, 'gini = 0.482\nsamples = 27\nvalue = [25, 17]\nclass = Yes'),
 Text(0.9, 0.08333333333333333, 'gini = 0.4\nsamples = 41\nvalue = [18, 47]\nclass = No'),
 Text(0.95, 0.25, 'gini = 0.492\nsamples = 22\nvalue = [22, 17]\nclass = Yes')]
```



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

Out[30]:
```
array([0.02885635, 0.07416598, 0.00546205, 0.05718657, 0.01801783,
       0.03969807, 0.00536918, 0.07292678, 0.01398008, 0.11247079,
       0.02243256, 0.08726483, 0.09411731, 0.14755223, 0.09522622,
       0.03423163, 0.0088935 , 0.06713687, 0.00466718, 0.010344  ])
```

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

Out[31]:

	Vername	Imp
13	px_width	0.147552
9	mobile_wt	0.112471
14	ram	0.095226
12	px_height	0.094117
11	pc	0.087265
1	battery_power	0.074166
7	int_memory	0.072927
17	talk_time	0.067137
3	clock_speed	0.057187
5	fc	0.039698
15	sc_h	0.034232
0	id	0.028856
10	n_cores	0.022433
4	dual_sim	0.018018
8	m_dep	0.013980
19	touch_screen	0.010344
16	sc_w	0.008894
2	blue	0.005462
6	four_g	0.005369
18	three_g	0.004667

LOAN DATASET

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

In [2]:
```
df=pd.read_csv(r"C:\Users\Dell\Downloads\loan1.csv")
df
```

Out[2]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Home Owner          10 non-null     object
 1   Marital Status      10 non-null     object
 2   Annual Income       10 non-null     int64 
 3   Defaulted Borrower  10 non-null     object
dtypes: int64(1), object(3)
memory usage: 448.0+ bytes
```

In [4]:
```
df['Marital Status'].value_counts()
```

Out[4]:
```
Single      4
Married     4
Divorced    2
Name: Marital Status, dtype: int64
```

In [5]:
```
HO={"Home Owner":{"Yes":1,"No":0}}
df=df.replace(HO)
df
```

Out[5]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	1	Single	125	No
1	0	Married	100	No
2	0	Single	70	No
3	1	Married	120	No
4	0	Divorced	95	Yes
5	0	Married	60	No
6	1	Divorced	220	No
7	0	Single	85	Yes
8	0	Married	75	No
9	0	Single	90	Yes

In [6]:
```
MS={"Marital Status":{"Single":1,"Married":2,"Divorced":3}}
df=df.replace(MS)
print(df)
```

```
   Home Owner  Marital Status  Annual Income Defaulted Borrower
0           1               1            125                 No
1           0               2            100                 No
2           0               1             70                 No
3           1               2            120                 No
4           0               3             95                Yes
5           0               2             60                 No
6           1               3            220                 No
7           0               1             85                Yes
8           0               2             75                 No
9           0               1             90                Yes
```

In [7]:
```
x=df.drop('Defaulted Borrower',axis=1)
y=df['Defaulted Borrower']
```

In [8]:
```
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)
x_train.shape,x_test.shape
```

Out[8]:
```
((7, 3), (3, 3))
```

In [9]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[9]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [10]:
```
rf=RandomForestClassifier()
```

In [11]:
```
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 [12]:
```
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[12]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [13]:
```
grid_search.best_score_
```

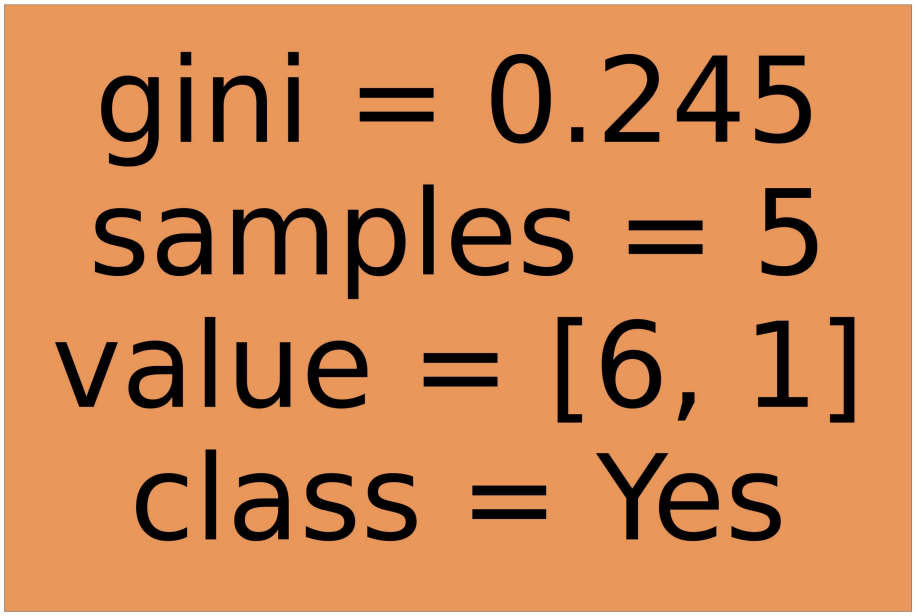
Out[13]:
```
0.7083333333333333
```

In [14]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

In [15]:
```
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[15]:
```
[Text(0.5, 0.5, 'gini = 0.245\nsamples = 5\nvalue = [6, 1]\nclass = Yes')]
```



In [16]:
```
rf_best.feature_importances_
```

Out[16]:
```
array([0., 0., 0.])
```

In [ ]:
```

```