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\91756\Documents\python\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]:

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

In [5]:

```
df['Marital Status'].value_counts()
```

Out[5]:

```
Marital Status
Single      4
Married     4
Divorced    2
Name: count, dtype: int64
```

In [6]:

```
HO={"Home Owner":{"Yes":1,"No":0}}
df=df.replace(HO)
print(df)
```

```
   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 [7]:

```
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 [8]:

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

In [9]:

```
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[9]:

```
((7, 3), (3, 3))
```

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]:

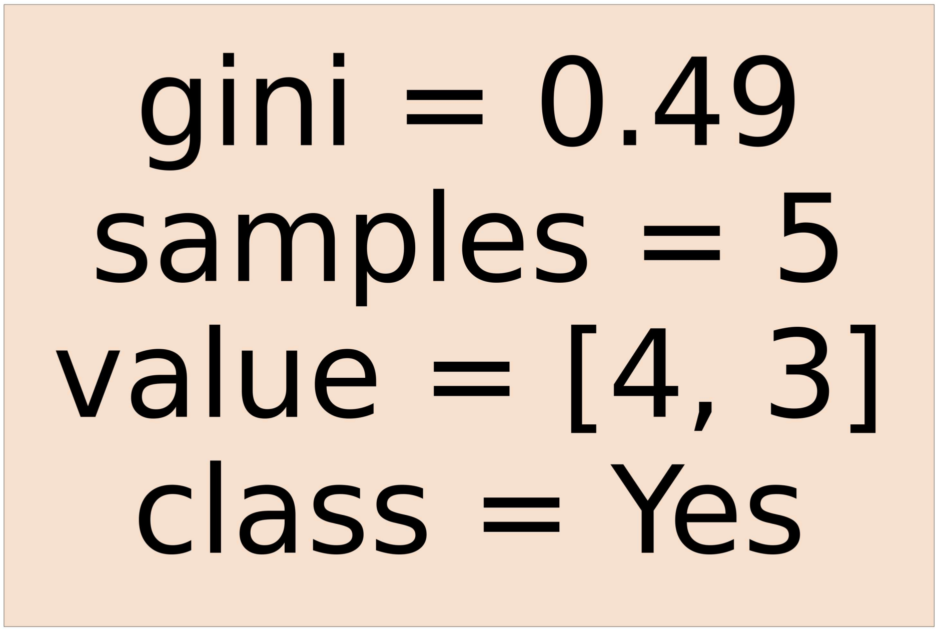
```
0.5833333333333333
```

In [15]:

```
rf_best=grid_search.best_estimator_
```

In [16]:
```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True)
```

Out[16]:
```
[Text(0.5, 0.5, 'gini = 0.49\nsamples = 5\nvalue = [4, 3]\nclass = Yes')]
```



Mobile Price Dataset

Train Dataset

In [17]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt,seaborn as sns
```

In [19]:
```
df=pd.read_csv(r"C:\Users\91756\Documents\python\Mobile_Price_Classification_train.csv")
df
```

Out[19]:

	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
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 [20]:
```
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 [21]:
```
x=df.drop('price_range',axis=1)
y=df['price_range']
```

In [22]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_state=42)
x_train.shape,x_test.shape
```

Out[22]:
```
((1200, 20), (800, 20))
```

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[23]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [24]:
```
rf=RandomForestClassifier()
```

In [25]:
```
params={'max_depth':[10,20,30,40],
 'min_samples_leaf':[30,23,45,14],
 'n_estimators':[45,34,32,12]}
```

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_
```

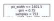
Out[27]:
```
0.8
```

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

```
RandomForestClassifier(max_depth=10, min_samples_leaf=14, n_estimators=12)
```

In [29]:
```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[8],feature_names=x.columns,class_names=['LOW','HIGH','MEDIUM','VERYHIGH'],filled=Tru
```

```
class = HIGH'),
 Text(0.8727272727272727, 0.4375, 'ram <= 2398.0\ngini = 0.696\nsamples = 92\nvalue = [16, 49, 26, 60]\nclass
= VERYHIGH'),
 Text(0.8363636363636363, 0.3125, 'blue <= 0.5\ngini = 0.603\nsamples = 54\nvalue = [16, 49, 22, 2]\nclass = H
IGH'),
 Text(0.8181818181818182, 0.1875, 'ram <= 1254.5\ngini = 0.515\nsamples = 29\nvalue = [8, 28, 7, 0]\nclass = H
IGH'),
 Text(0.8, 0.0625, 'gini = 0.488\nsamples = 14\nvalue = [8, 11, 0, 0]\nclass = HIGH'),
 Text(0.8363636363636363, 0.0625, 'gini = 0.413\nsamples = 15\nvalue = [0, 17, 7, 0]\nclass = HIGH'),
 Text(0.8545454545454545, 0.1875, 'gini = 0.653\nsamples = 25\nvalue = [8, 21, 15, 2]\nclass = HIGH'),
 Text(0.9090909090909091, 0.3125, 'mobile_wt <= 105.0\ngini = 0.121\nsamples = 38\nvalue = [0, 0, 4, 58]\nclas
s = VERYHIGH'),
 Text(0.8909090909090909, 0.1875, 'gini = 0.278\nsamples = 14\nvalue = [0, 0, 4, 20]\nclass = VERYHIGH'),
 Text(0.9272727272727272, 0.1875, 'gini = 0.0\nsamples = 24\nvalue = [0, 0, 0, 38]\nclass = VERYHIGH'),
 Text(0.9636363636363636, 0.4375, 'px_height <= 685.5\ngini = 0.695\nsamples = 38\nvalue = [14, 19, 18, 3]\ncl
ass = HIGH'),
 Text(0.9454545454545454, 0.3125, 'gini = 0.703\nsamples = 15\nvalue = [8, 5, 8, 2]\nclass = LOW'),
 Text(0.9818181818181818, 0.3125, 'gini = 0.653\nsamples = 23\nvalue = [6, 14, 10, 1]\nclass = HIGH')]
```



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
13	ram	0.707798
0	battery_power	0.074359
12	px_width	0.060707
11	px_height	0.029782
8	mobile_wt	0.018743
6	int_memory	0.016066
16	talk_time	0.013612
2	clock_speed	0.012709
7	m_dep	0.011171
4	fc	0.011156
10	pc	0.010022
14	sc_h	0.008385
15	sc_w	0.008191
9	n_cores	0.008148
3	dual_sim	0.002303
1	blue	0.001899
18	touch_screen	0.001677
5	four_g	0.001606
19	wifi	0.000925
17	three_g	0.000741

Mobile Price

Test Dataset

In [32]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt,seaborn as sns
```

In [33]:

```
df=pd.read_csv(r"C:\Users\91756\Documents\python\Mobile_Price_Classification_test.csv")
df
```

Out[33]:

	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
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 [34]:

```
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 [35]:

```
x=df.drop('four_g',axis=1)
y=df['four_g']
```

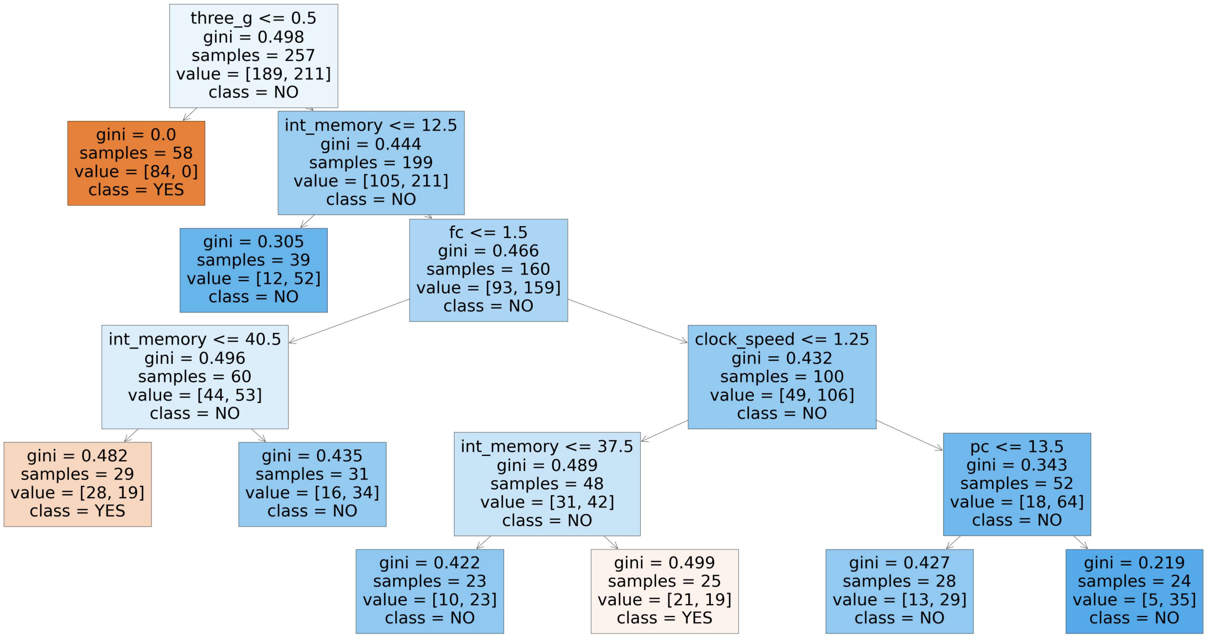
```
In [36]: df['dual_sim'].value_counts()

Out[36]: dual_sim
         1    517
         0    483
         Name: count, dtype: int64

In [37]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_state=42)
         x_train.shape,x_test.shape

Out[37]: ((600, 20), (400, 20))

In [38]: rf=RandomForestClassifier()

In [39]: params={'max_depth':[2,30,45,34,45],
          'min_samples_leaf':[23,34,45,56],
          'n_estimators':[10,23,34,78,89]}

In [40]: from sklearn.model_selection import GridSearchCV
         grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
         grid_search.fit(x_test,y_test)

Out[40]: ▸            GridSearchCV
         ▸ estimator: RandomForestClassifier
                 ▸ RandomForestClassifier

In [41]: grid_search.best_score_

Out[41]: 0.655

In [42]: rf_best=grid_search.best_estimator_
         print(rf_best)

         RandomForestClassifier(max_depth=30, min_samples_leaf=23, n_estimators=78)
```

In [43]:

```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[6],feature_names=x.columns,class_names=['YES','NO'],filled=True)
```

Out[43]:

```
[Text(0.22727272727272727, 0.9166666666666666, 'three_g <= 0.5\ngini = 0.498\nsamples = 257\nvalue = [189, 211]\n
class = NO'),
 Text(0.13636363636363635, 0.75, 'gini = 0.0\nsamples = 58\nvalue = [84, 0]\nclass = YES'),
 Text(0.3181818181818182, 0.75, 'int_memory <= 12.5\ngini = 0.444\nsamples = 199\nvalue = [105, 211]\nclass = N
O'),
 Text(0.22727272727272727, 0.5833333333333334, 'gini = 0.305\nsamples = 39\nvalue = [12, 52]\nclass = NO'),
 Text(0.4090909090909091, 0.5833333333333334, 'fc <= 1.5\ngini = 0.466\nsamples = 160\nvalue = [93, 159]\nclass =
NO'),
 Text(0.18181818181818182, 0.4166666666666667, 'int_memory <= 40.5\ngini = 0.496\nsamples = 60\nvalue = [44, 53]
\nclass = NO'),
 Text(0.09090909090909091, 0.25, 'gini = 0.482\nsamples = 29\nvalue = [28, 19]\nclass = YES'),
 Text(0.2727272727272727, 0.25, 'gini = 0.435\nsamples = 31\nvalue = [16, 34]\nclass = NO'),
 Text(0.6363636363636364, 0.4166666666666667, 'clock_speed <= 1.25\ngini = 0.432\nsamples = 100\nvalue = [49, 10
6]\nclass = NO'),
 Text(0.45454545454545453, 0.25, 'int_memory <= 37.5\ngini = 0.489\nsamples = 48\nvalue = [31, 42]\nclass = NO'),
 Text(0.36363636363636365, 0.08333333333333333, 'gini = 0.422\nsamples = 23\nvalue = [10, 23]\nclass = NO'),
 Text(0.5454545454545454, 0.08333333333333333, 'gini = 0.499\nsamples = 25\nvalue = [21, 19]\nclass = YES'),
 Text(0.8181818181818182, 0.25, 'pc <= 13.5\ngini = 0.343\nsamples = 52\nvalue = [18, 64]\nclass = NO'),
 Text(0.7272727272727273, 0.08333333333333333, 'gini = 0.427\nsamples = 28\nvalue = [13, 29]\nclass = NO'),
 Text(0.9090909090909091, 0.08333333333333333, 'gini = 0.219\nsamples = 24\nvalue = [5, 35]\nclass = NO')]
```



In [44]:

```
rf_best.feature_importances_
```

Out[44]:

```
array([0.08007951, 0.04861858, 0.00215825, 0.03492141, 0.00484123,
       0.01816655, 0.04442906, 0.02061731, 0.06007986, 0.0449101 ,
       0.02074206, 0.03977116, 0.03887831, 0.08717415, 0.02919748,
       0.01727266, 0.02218767, 0.37199406, 0.00876095, 0.00519963])
```

In [45]:

```
imp_df=pd.DataFrame({"varname":x_test.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[45]:

```
         varname       Imp
17       three_g  0.371994
13           ram  0.087174
 0            id  0.080080
 8     mobile_wt  0.060080
 1 battery_power  0.048619
 9       n_cores  0.044910
 6    int_memory  0.044429
11     px_height  0.039771
12      px_width  0.038878
 3   clock_speed  0.034921
14          sc_h  0.029197
16     talk_time  0.022188
10            pc  0.020742
 7         m_dep  0.020617
 5            fc  0.018167
15          sc_w  0.017273
18  touch_screen  0.008761
19          wifi  0.005200
 4      dual_sim  0.004841
 2          blue  0.002158
```

In [ ]: