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\yoshitha lakshmi\OneDrive\Desktop\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 [31]:

```
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[31]:

```
((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 [16]:

```
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[16]:

GridSearchCV

estimator: RandomForestClassifier

RandomForestClassifier

In [20]:

```
grid_search.best_score_
```

Out[20]:

```
0.5833333333333333
```

In [21]:

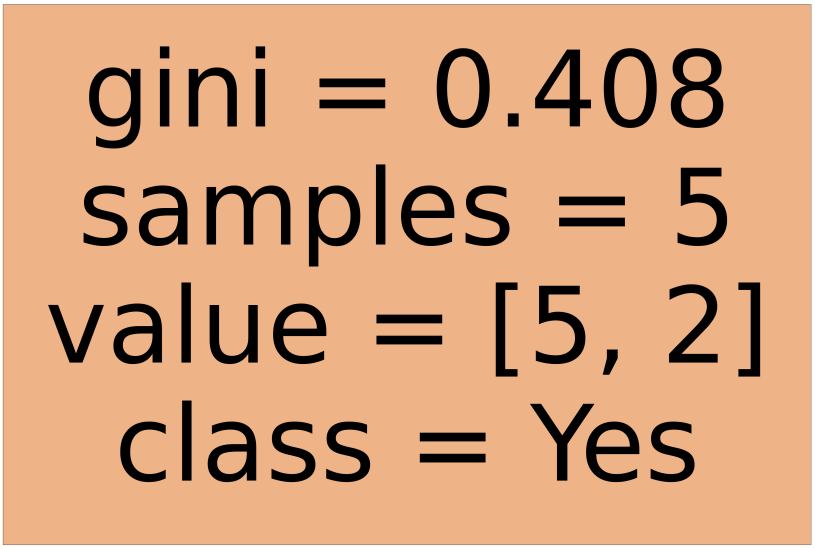
```
rf_best=grid_search.best_estimator_
```

In [30]:

```
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'],f
```

Out[30]:

```
[Text(0.5, 0.5, 'gini = 0.408\nsamples = 5\nvalue = [5, 2]\nclass = Ye
s')]
```



Mobile Price Dataset

Train Dataset

In [80]:

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

In [81]:

```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\Mobile_Price_Cla
df
```

Out[81]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile
0	842	0	2.2	0	1	0	7	0.6	
1	1021	1	0.5	1	0	1	53	0.7	
2	563	1	0.5	1	2	1	41	0.9	
3	615	1	2.5	0	0	0	10	0.8	
4	1821	1	1.2	0	13	1	44	0.6	
...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	
1996	1965	1	2.6	1	0	0	39	0.2	
1997	1911	0	0.9	1	1	1	36	0.7	
1998	1512	0	0.9	0	4	1	46	0.1	
1999	510	1	2.0	1	5	1	45	0.9	

2000 rows × 21 columns

In [82]:

```
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 [83]:

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

In [84]:

```
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[84]:

```
((1200, 20), (800, 20))
```

In [85]:

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

Out[85]:

RandomForestClassifier

RandomForestClassifier()

In [86]:

```
rf=RandomForestClassifier()
```

In [87]:

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

In [88]:

```
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[88]:

GridSearchCV

estimator: RandomForestClassifier

RandomForestClassifier

In [89]:

```
grid_search.best_score_
```

Out[89]:

```
0.7966666666666666
```

In [93]:

```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=40, min_samples_leaf=14, n_estimators=3
4)
```

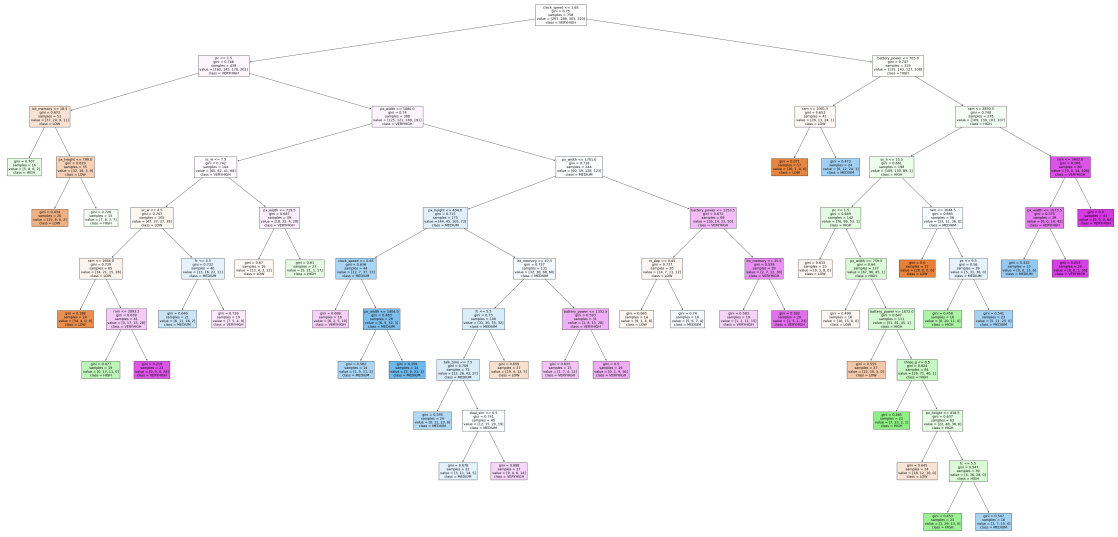
In [105]:

```
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',
```

Out[105]:

```
[Text(0.5007102272727273, 0.9545454545454546, 'clock_speed <= 1.65\ngini
= 0.75\nsamples = 758\nvalue = [297, 288, 305, 310]\nclass = VERYHIGH'),
 Text(0.2002840909090909, 0.8636363636363636, 'pc <= 1.5\ngini = 0.746\ns
amples = 439\nvalue = [162, 145, 178, 202]\nclass = VERYHIGH'),
 Text(0.045454545454545456, 0.7727272727272727, 'int_memory <= 18.5\ngini
= 0.673\nsamples = 51\nvalue = [37, 24, 9, 11]\nclass = LOW'),
 Text(0.022727272727272728, 0.6818181818181818, 'gini = 0.707\nsamples =
16\nvalue = [5, 8, 6, 2]\nclass = HIGH'),
 Text(0.06818181818181818, 0.6818181818181818, 'px_height <= 799.0\ngini
= 0.619\nsamples = 35\nvalue = [32, 16, 3, 9]\nclass = LOW'),
 Text(0.045454545454545456, 0.5909090909090909, 'gini = 0.434\nsamples =
20\nvalue = [25, 8, 0, 2]\nclass = LOW'),
 Text(0.09090909090909091, 0.5909090909090909, 'gini = 0.726\nsamples = 1
5\nvalue = [7, 8, 3, 7]\nclass = HIGH'),
 Text(0.35511363636363635, 0.7727272727272727, 'px_width <= 1080.0\ngini
= 0.74\nsamples = 388\nvalue = [125, 121, 169, 191]\nclass = VERYHIGH'),
 Text(0.19318181818181818, 0.6818181818181818, 'sc_w <= 7.5\ngini = 0.742
\nsamples = 144\nvalue = [65, 62, 41, 68]\nclass = VERYHIGH'),
 Text(0.13636363636363635, 0.5909090909090909, 'sc_w <= 4.5\ngini = 0.747
\nsamples = 105\nvalue = [47, 37, 37, 39]\nclass = LOW'),
 Text(0.09090909090909091, 0.5, 'ram <= 1684.0\ngini = 0.729\nsamples = 6
5\nvalue = [34, 21, 15, 28]\nclass = LOW'),
 Text(0.06818181818181818, 0.4090909090909091, 'gini = 0.188\nsamples = 2
4\nvalue = [34, 4, 0, 0]\nclass = LOW'),
 Text(0.11363636363636363, 0.4090909090909091, 'ram <= 2893.5\ngini = 0.6
39\nsamples = 41\nvalue = [0, 17, 15, 28]\nclass = VERYHIGH'),
 Text(0.09090909090909091, 0.3181818181818182, 'gini = 0.477\nsamples = 1
9\nvalue = [0, 17, 11, 0]\nclass = HIGH'),
 Text(0.13636363636363635, 0.3181818181818182, 'gini = 0.219\nsamples = 2
2\nvalue = [0, 0, 4, 28]\nclass = VERYHIGH'),
 Text(0.18181818181818182, 0.5, 'fc <= 4.5\ngini = 0.732\nsamples = 40\nv
alue = [13, 16, 22, 11]\nclass = MEDIUM'),
 Text(0.1590909090909091, 0.4090909090909091, 'gini = 0.646\nsamples = 21
\nvalue = [6, 11, 18, 2]\nclass = MEDIUM'),
 Text(0.20454545454545456, 0.4090909090909091, 'gini = 0.726\nsamples = 1
9\nvalue = [7, 5, 4, 9]\nclass = VERYHIGH'),
 Text(0.25, 0.5909090909090909, 'px_width <= 719.5\ngini = 0.687\nsamples
= 39\nvalue = [18, 25, 4, 29]\nclass = VERYHIGH'),
 Text(0.22727272727272727, 0.5, 'gini = 0.67\nsamples = 16\nvalue = [13,
4, 3, 12]\nclass = LOW'),
 Text(0.2727272727272727, 0.5, 'gini = 0.61\nsamples = 23\nvalue = [5, 2
1, 1, 17]\nclass = HIGH'),
 Text(0.5170454545454546, 0.6818181818181818, 'px_width <= 1701.0\ngini =
0.718\nsamples = 244\nvalue = [60, 59, 128, 123]\nclass = MEDIUM'),
 Text(0.3977272727272727, 0.5909090909090909, 'px_height <= 404.0\ngini =
0.715\nsamples = 175\nvalue = [44, 45, 105, 73]\nclass = MEDIUM'),
 Text(0.3181818181818182, 0.5, 'clock_speed <= 0.65\ngini = 0.636\nsample
s = 44\nvalue = [12, 7, 37, 13]\nclass = MEDIUM'),
 Text(0.29545454545454547, 0.4090909090909091, 'gini = 0.688\nsamples = 1
6\nvalue = [6, 2, 5, 10]\nclass = VERYHIGH'),
 Text(0.3409090909090909, 0.4090909090909091, 'px_width <= 1404.5\ngini =
0.483\nsamples = 28\nvalue = [6, 5, 32, 3]\nclass = MEDIUM'),
 Text(0.3181818181818182, 0.3181818181818182, 'gini = 0.582\nsamples = 14
\nvalue = [1, 5, 11, 2]\nclass = MEDIUM'),
 Text(0.36363636363636365, 0.3181818181818182, 'gini = 0.359\nsamples = 1
4\nvalue = [5, 0, 21, 1]\nclass = MEDIUM'),
 Text(0.4772727272727273, 0.5, 'int_memory <= 47.5\ngini = 0.727\nsamples
= 131\nvalue = [32, 38, 68, 60]\nclass = MEDIUM'),
 Text(0.4318181818181818, 0.4090909090909091, 'fc <= 5.5\ngini = 0.73\nsa
mples = 100\nvalue = [31, 30, 55, 32]\nclass = MEDIUM'),
 Text(0.4090909090909091, 0.3181818181818182, 'talk_time <= 7.5\ngini =
```

```
0.709\nsamples = 73\nvalue = [12, 26, 43, 27]\nclass = MEDIUM'),
 Text(0.38636363636363635, 0.22727272727272727, 'gini = 0.595\nsamples =
24\nvalue = [0, 11, 23, 8]\nclass = MEDIUM'),
 Text(0.4318181818181818, 0.22727272727272727, 'dual_sim <= 0.5\ngini =
0.741\nsamples = 49\nvalue = [12, 15, 20, 19]\nclass = MEDIUM'),
 Text(0.4090909090909091, 0.13636363636363635, 'gini = 0.678\nsamples = 2
2\nvalue = [3, 11, 14, 5]\nclass = MEDIUM'),
 Text(0.45454545454545453, 0.13636363636363635, 'gini = 0.698\nsamples =
27\nvalue = [9, 4, 6, 14]\nclass = VERYHIGH'),
 Text(0.45454545454545453, 0.3181818181818182, 'gini = 0.659\nsamples = 2
7\nvalue = [19, 4, 12, 5]\nclass = LOW'),
 Text(0.5227272727272727, 0.4090909090909091, 'battery_power <= 1352.5\ng
ini = 0.593\nsamples = 31\nvalue = [1, 8, 13, 28]\nclass = VERYHIGH'),
 Text(0.5, 0.3181818181818182, 'gini = 0.635\nsamples = 15\nvalue = [1,
7, 4, 12]\nclass = VERYHIGH'),
 Text(0.5454545454545454, 0.3181818181818182, 'gini = 0.5\nsamples = 16\n
value = [0, 1, 9, 16]\nclass = VERYHIGH'),
 Text(0.6363636363636364, 0.5909090909090909, 'battery_power <= 1153.5\ng
ini = 0.672\nsamples = 69\nvalue = [16, 14, 23, 50]\nclass = VERYHIGH'),
 Text(0.5909090909090909, 0.5, 'm_dep <= 0.45\ngini = 0.737\nsamples = 30
\nvalue = [14, 7, 11, 12]\nclass = LOW'),
 Text(0.5681818181818182, 0.4090909090909091, 'gini = 0.665\nsamples = 14
\nvalue = [9, 1, 4, 8]\nclass = LOW'),
 Text(0.6136363636363636, 0.4090909090909091, 'gini = 0.74\nsamples = 16
\nvalue = [5, 6, 7, 4]\nclass = MEDIUM'),
 Text(0.6818181818181818, 0.5, 'int_memory <= 35.5\ngini = 0.529\nsamples
= 39\nvalue = [2, 7, 12, 38]\nclass = VERYHIGH'),
 Text(0.6590909090909091, 0.4090909090909091, 'gini = 0.583\nsamples = 19
\nvalue = [1, 2, 11, 15]\nclass = VERYHIGH'),
 Text(0.7045454545454546, 0.4090909090909091, 'gini = 0.382\nsamples = 20
```



```
3, 8, 0]\nclass = LOW'),
 Text(0.7727272727272727, 0.5, 'px_width <= 759.5\ngini = 0.64\nsamples =
127\nvalue = [67, 96, 45, 1]\nclass = HIGH'),
 Text(0.75, 0.4090909090909091, 'gini = 0.499\nsamples = 16\nvalue = [16,
15, 0, 0]\nclass = LOW'),
 Text(0.7954545454545454, 0.4090909090909091, 'battery_power <= 1072.0\ng
ini = 0.647\nsamples = 111\nvalue = [51, 81, 45, 1]\nclass = HIGH'),
 Text(0.7727272727272727, 0.3181818181818182, 'gini = 0.555\nsamples = 27
\nvalue = [22, 10, 3, 0]\nclass = LOW'),
 Text(0.8181818181818182, 0.3181818181818182, 'three_g <= 0.5\ngini = 0.6
24\nsamples = 84\nvalue = [29, 71, 42, 1]\nclass = HIGH'),
 Text(0.7954545454545454, 0.2272727272727273, 'gini = 0.465\nsamples = 2
1\nvalue = [7, 23, 2, 1]\nclass = HIGH'),
 Text(0.8409090909090909, 0.2272727272727273, 'px_height <= 418.5\ngini
= 0.637\nsamples = 63\nvalue = [22, 48, 38, 0]\nclass = HIGH'),
```

In [96]:

```
rf_best.feature_importances_
```

Out[96]:

```
array([0.07367867, 0.00491513, 0.01517821, 0.00444502, 0.01534813,
       0.00316721, 0.02461858, 0.01621786, 0.02049589, 0.00987598,
       0.01784034, 0.03972963, 0.05273638, 0.6462072 , 0.01606024,
       0.01631174, 0.01291987, 0.00156868, 0.00389944, 0.00478582])
```

In [97]:

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

Out[97]:

	varname	Imp
13	ram	0.646207
0	battery_power	0.073679
12	px_width	0.052736
11	px_height	0.039730
6	int_memory	0.024019
8	mobile_wt	0.020406
10	pc	0.017840
15	sc_w	0.016312
7	m_dep	0.016218
14	sc_h	0.016066
4	fc	0.015348
2	clock_speed	0.015178
16	talk_time	0.012920
9	n_cores	0.009876
1	blue	0.004915
19	wifi	0.004786
3	dual_sim	0.004445
18	touch_screen	0.003899
5	four_g	0.003167
17	three_g	0.001569

Mobile Price

Test Dataset

In [98]:

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

In [99]:

```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\Mobile_Price_Cla
df
```

Out[99]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	m
0	1	1043	1	1.8	1	14	0	5	0.1	
1	2	841	1	0.5	1	4	1	61	0.8	
2	3	1807	1	2.8	0	1	0	27	0.9	
3	4	1546	0	0.5	1	18	1	25	0.5	
4	5	1434	0	1.4	0	11	1	49	0.5	
...	...	...	...	...	...	...	...	...	...	
995	996	1700	1	1.9	0	0	1	54	0.5	
996	997	609	0	1.8	1	0	0	13	0.9	
997	998	1185	0	1.4	0	1	1	8	0.5	
998	999	1533	1	0.5	1	0	0	50	0.4	
999	1000	1270	1	0.5	0	4	1	35	0.1	

1000 rows × 21 columns

In [106]:

```
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 [107]:

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

In [109]:

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

Out[109]:

```
dual_sim
1    517
0    483
Name: count, dtype: int64
```

In [119]:

```
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[119]:

```
((600, 20), (400, 20))
```

In [120]:

```
rf=RandomForestClassifier()
```

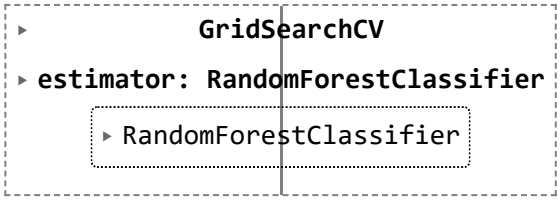
In [121]:

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

In [125]:

```
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[125]:



In [126]:

```
grid_search.best_score_
```

Out[126]:

```
0.6575
```

In [127]:

```
rf_best=grid_search.best_estimator_
print(rf_best)
```

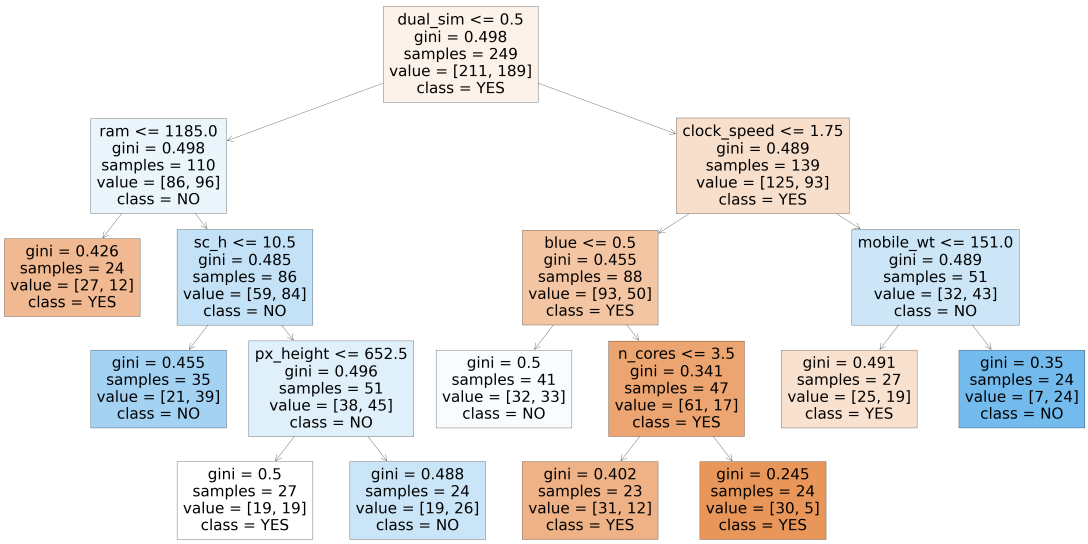
```
RandomForestClassifier(max_depth=34, min_samples_leaf=23, n_estimators=2
3)
```

In [134]:

```
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'],f
```

Out[134]:

```
[Text(0.4230769230769231, 0.9, 'dual_sim <= 0.5\ngini = 0.498\nsamples =
249\nvalue = [211, 189]\nclass = YES'),
 Text(0.15384615384615385, 0.7, 'ram <= 1185.0\ngini = 0.498\nsamples = 1
10\nvalue = [86, 96]\nclass = NO'),
 Text(0.07692307692307693, 0.5, 'gini = 0.426\nsamples = 24\nvalue = [27,
12]\nclass = YES'),
 Text(0.23076923076923078, 0.5, 'sc_h <= 10.5\ngini = 0.485\nsamples = 86
\nvalue = [59, 84]\nclass = NO'),
 Text(0.15384615384615385, 0.3, 'gini = 0.455\nsamples = 35\nvalue = [21,
39]\nclass = NO'),
 Text(0.3076923076923077, 0.3, 'px_height <= 652.5\ngini = 0.496\nsamples
= 51\nvalue = [38, 45]\nclass = NO'),
 Text(0.23076923076923078, 0.1, 'gini = 0.5\nsamples = 27\nvalue = [19, 1
9]\nclass = YES'),
 Text(0.38461538461538464, 0.1, 'gini = 0.488\nsamples = 24\nvalue = [19,
26]\nclass = NO'),
 Text(0.6923076923076923, 0.7, 'clock_speed <= 1.75\ngini = 0.489\nsample
s = 139\nvalue = [125, 93]\nclass = YES'),
 Text(0.5384615384615384, 0.5, 'blue <= 0.5\ngini = 0.455\nsamples = 88\n
value = [93, 50]\nclass = YES'),
 Text(0.46153846153846156, 0.3, 'gini = 0.5\nsamples = 41\nvalue = [32, 3
3]\nclass = NO'),
 Text(0.6153846153846154, 0.3, 'n_cores <= 3.5\ngini = 0.341\nsamples = 4
7\nvalue = [61, 17]\nclass = YES'),
 Text(0.5384615384615384, 0.1, 'gini = 0.402\nsamples = 23\nvalue = [31,
12]\nclass = YES'),
 Text(0.6923076923076923, 0.1, 'gini = 0.245\nsamples = 24\nvalue = [30,
5]\nclass = YES'),
 Text(0.8461538461538461, 0.5, 'mobile_wt <= 151.0\ngini = 0.489\nsamples
= 51\nvalue = [32, 43]\nclass = NO'),
 Text(0.7692307692307693, 0.3, 'gini = 0.491\nsamples = 27\nvalue = [25,
19]\nclass = YES'),
 Text(0.9230769230769231, 0.3, 'gini = 0.35\nsamples = 24\nvalue = [7, 2
4]\nclass = NO')]
```



In [135]:

```
rf_best.feature_importances_
```

Out[135]:

```
array([0.03742735, 0.06805098, 0.01490144, 0.05988351, 0.01763912,
       0.00818625, 0.05355547, 0.03207307, 0.05884514, 0.03886619,
       0.01494106, 0.03856201, 0.02403102, 0.0779887 , 0.04617572,
       0.03178602, 0.02040916, 0.34088762, 0.01357091, 0.00221925])
```

In [136]:

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

Out[136]:

	varname	Imp
17	three_g	0.340888
13	ram	0.077989
1	battery_power	0.068051
3	clock_speed	0.059884
8	mobile_wt	0.058845
6	int_memory	0.053555
14	sc_h	0.046176
9	n_cores	0.038866
11	px_height	0.038562
0	id	0.037427
7	m_dep	0.032073
15	sc_w	0.031786
12	px_width	0.024031
16	talk_time	0.020409
4	dual_sim	0.017639
10	pc	0.014941
2	blue	0.014901
18	touch_screen	0.013571
5	fc	0.008186
19	wifi	0.002219

In [ ]:

```

```