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

In [2]:
```
train_data=pd.read_csv(r"C:\Users\manasa\Downloads\Mobile_Price_Classification
train_data
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
0	842	0	2.2	0	1	0	7	0.6	188
1	1021	1	0.5	1	0	1	53	0.7	136
2	563	1	0.5	1	2	1	41	0.9	145
3	615	1	2.5	0	0	0	10	0.8	131
4	1821	1	1.2	0	13	1	44	0.6	141
...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	106
1996	1965	1	2.6	1	0	0	39	0.2	187
1997	1911	0	0.9	1	1	1	36	0.7	108
1998	1512	0	0.9	0	4	1	46	0.1	145
1999	510	1	2.0	1	5	1	45	0.9	168

2000 rows × 21 columns

In [3]:
```
test_data=pd.read_csv(r"C:\Users\manasa\Downloads\Mobile_Price_Classification_
test_data
```

Out[3]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile
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 [4]:
```
train_data.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: 328.3 KB
```

In [5]:
```
test_data.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 [6]:
```
x=train_data.drop('wifi',axis=1)
y=train_data['wifi']
```

In [7]:
```
x=test_data.drop('wifi',axis=1)
y=test_data['wifi']
```

In [8]:
```
train_data['dual_sim'].value_counts()
```

Out[8]:
```
dual_sim
1    1019
0     981
Name: count, dtype: int64
```

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

Out[9]:
```
dual_sim
1    517
0    483
Name: count, dtype: int64
```

In [10]:
```
TG={"three_g":{"yes":1,"No":0}}
train_data=train_data.replace(TG)
print(train_data)
```

```
      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 [11]:
```
TG={"three_g":{"yes":1,"No":0}}
test_data=test_data.replace(TG)
print(test_data)
```

```
       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 [12]:
```
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
x_train.shape,x_test.shape
```

Out[12]:
```
((700, 20), (300, 20))
```

In [13]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[13]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [15]:
```
rf=RandomForestClassifier()
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 [21]:
```
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)
```

```
C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\model_selection\_validation.py:378: FitFailedWarning:
60 fits failed out of a total of 300.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\sklearn\model_selection\_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\sklearn\ensemble\_forest.py", line 340, in fit
    self._validate_params()
  File "C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\sklearn\base.py", line 600, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\sklearn\utils\_param_validation.py", line 97, in validate_parameter_cons
traints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'min_samples_leaf'
parameter of RandomForestClassifier must be an int in the range [1, inf) or a
float in the range (0.0, 1.0). Got 20.5 instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\model_selection\_search.py:952: UserWarning: One or more of the test sco
res are non-finite: [0.50714286 0.49428571 0.52       0.51285714 0.49857143
0.48857143
 0.51285714 0.48857143 0.48857143 0.49714286 0.50571429 0.49285714
        nan        nan        nan        nan        nan        nan
 0.48428571 0.50285714 0.49285714 0.51714286 0.5        0.5
 0.5        0.5        0.5        0.5        0.5        0.5
 0.48285714 0.49571429 0.51285714 0.5        0.49428571 0.51
 0.47857143 0.52142857 0.47285714 0.5        0.49428571 0.48714286
        nan        nan        nan        nan        nan        nan
 0.48571429 0.51285714 0.48428571 0.49       0.49857143 0.50142857
 0.5        0.5        0.5        0.5        0.5        0.5
 0.48428571 0.51428571 0.49428571 0.49714286 0.51714286 0.48857143
 0.50714286 0.48428571 0.49571429 0.49857143 0.49857143 0.49857143
        nan        nan        nan        nan        nan        nan
 0.48857143 0.48142857 0.50428571 0.50857143 0.49714286 0.50142857
 0.5        0.5        0.5        0.5        0.5        0.5
 0.50428571 0.47571429 0.51285714 0.48714286 0.46571429 0.51285714
 0.52285714 0.51       0.50857143 0.49142857 0.49571429 0.50428571
        nan        nan        nan        nan        nan        nan
 0.50428571 0.48285714 0.48285714 0.48571429 0.50857143 0.5
 0.5        0.5        0.5        0.5        0.5        0.5
 0.51857143 0.50428571 0.49428571 0.49285714 0.52428571 0.49142857
 0.52       0.50285714 0.50714286 0.50714286 0.48714286 0.50428571
        nan        nan        nan        nan        nan        nan
```

```
 0.45714286 0.50428571 0.48714286 0.48857143 0.49285714 0.49571429
 0.5        0.5        0.5        0.5        0.5        0.5       ]
  warnings.warn(
```

Out[21]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20.5, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [22]:
```
grid_search.best_score_
```

Out[22]:
```
0.5242857142857142
```

In [23]:
```
grid_search.fit(x_train,y_train)
```

```
C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\model_selection\_validation.py:378: FitFailedWarning:
60 fits failed out of a total of 300.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\sklearn\model_selection\_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\sklearn\ensemble\_forest.py", line 340, in fit
    self._validate_params()
  File "C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\sklearn\base.py", line 600, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-pack
ages\sklearn\utils\_param_validation.py", line 97, in validate_parameter_cons
traints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'min_samples_leaf'
parameter of RandomForestClassifier must be an int in the range [1, inf) or a
float in the range (0.0, 1.0). Got 20.5 instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\manasa\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\model_selection\_search.py:952: UserWarning: One or more of the test sco
res are non-finite: [0.49857143 0.47142857 0.51857143 0.51       0.49857143
0.50571429
 0.48857143 0.5        0.51285714 0.50142857 0.49857143 0.49428571
        nan        nan        nan        nan        nan        nan
 0.51428571 0.50714286 0.48428571 0.49       0.50285714 0.5
 0.5        0.5        0.5        0.5        0.5        0.5
 0.49857143 0.53142857 0.47285714 0.49       0.51142857 0.48142857
 0.54       0.49285714 0.51285714 0.50285714 0.47       0.49
        nan        nan        nan        nan        nan        nan
 0.45714286 0.48857143 0.50857143 0.50285714 0.49857143 0.50714286
 0.5        0.5        0.5        0.5        0.5        0.5
 0.50285714 0.5        0.53714286 0.50142857 0.51142857 0.49142857
 0.50428571 0.50285714 0.51       0.49857143 0.47857143 0.49714286
        nan        nan        nan        nan        nan        nan
 0.49571429 0.46714286 0.50285714 0.47857143 0.48285714 0.49571429
 0.5        0.5        0.5        0.5        0.5        0.5
 0.49       0.50571429 0.5        0.50285714 0.52142857 0.50142857
 0.51142857 0.49714286 0.48428571 0.48857143 0.49428571 0.48142857
        nan        nan        nan        nan        nan        nan
 0.46714286 0.46857143 0.5        0.49857143 0.48571429 0.48428571
 0.5        0.5        0.5        0.5        0.5        0.5
 0.49714286 0.50142857 0.5        0.49571429 0.48714286 0.50285714
 0.51285714 0.46       0.49571429 0.48857143 0.49142857 0.50714286
        nan        nan        nan        nan        nan        nan
```

```
 0.51428571 0.47571429 0.48857143 0.49142857 0.48428571 0.48571429
 0.5        0.5        0.5        0.5        0.5        0.5       ]
  warnings.warn(
```

Out[23]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20.5, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:
```
grid_search.best_score_
```

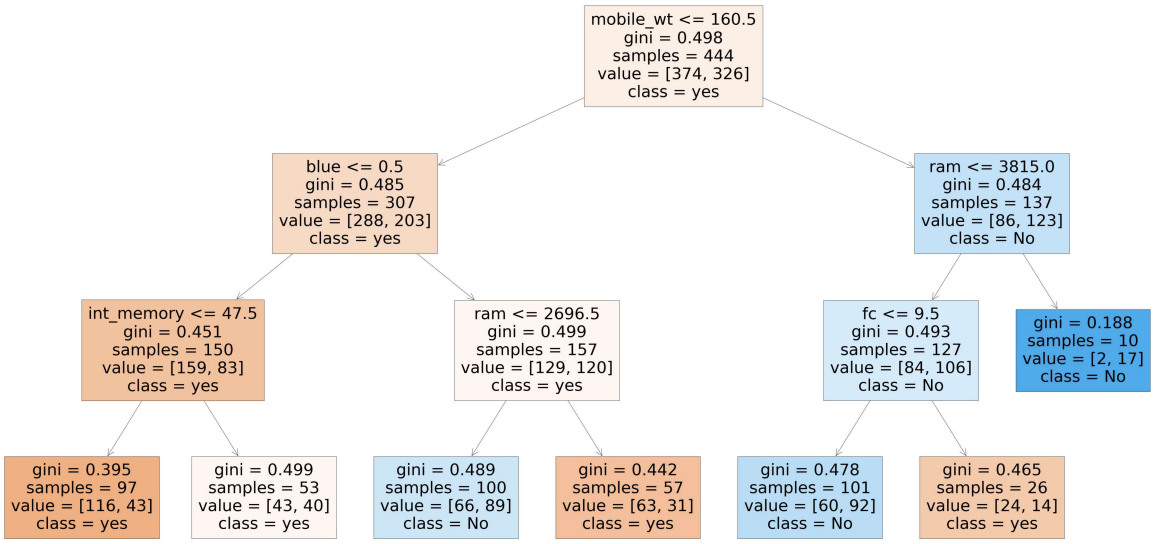
Out[24]:
```
0.54
```

In [25]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=3, min_samples_leaf=10, n_estimators=10)
```

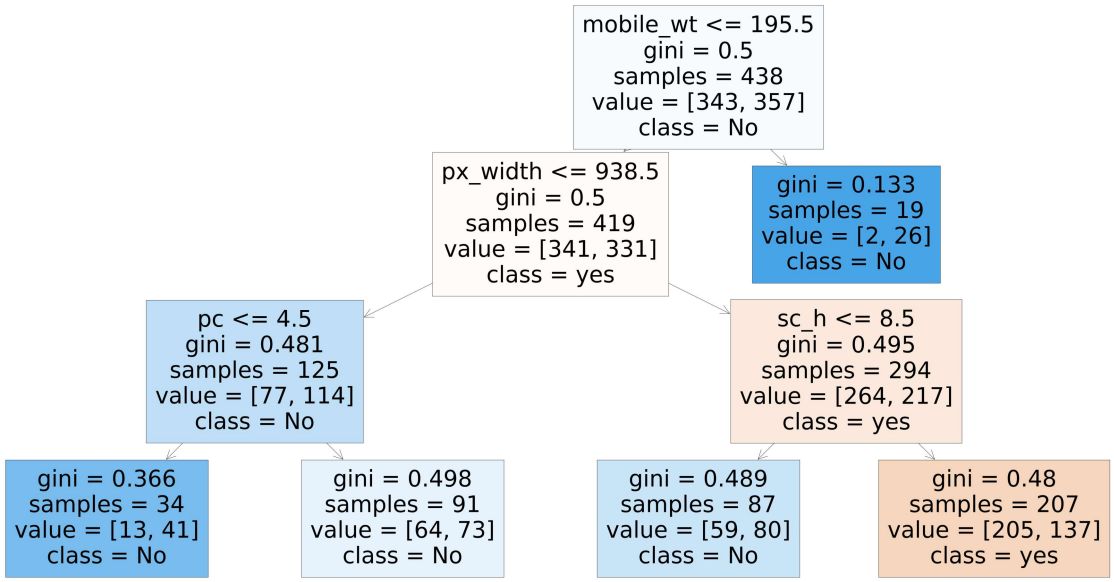
In [27]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=["yes","N
```

Out[27]:
```
[Text(0.5769230769230769, 0.875, 'mobile_wt <= 160.5\ngini = 0.498\nsamples =
444\nvalue = [374, 326]\nclass = yes'),
 Text(0.3076923076923077, 0.625, 'blue <= 0.5\ngini = 0.485\nsamples = 307\nv
alue = [288, 203]\nclass = yes'),
 Text(0.15384615384615385, 0.375, 'int_memory <= 47.5\ngini = 0.451\nsamples
= 150\nvalue = [159, 83]\nclass = yes'),
 Text(0.07692307692307693, 0.125, 'gini = 0.395\nsamples = 97\nvalue = [116,
43]\nclass = yes'),
 Text(0.23076923076923078, 0.125, 'gini = 0.499\nsamples = 53\nvalue = [43, 4
0]\nclass = yes'),
 Text(0.46153846153846156, 0.375, 'ram <= 2696.5\ngini = 0.499\nsamples = 157
\nvalue = [129, 120]\nclass = yes'),
 Text(0.38461538461538464, 0.125, 'gini = 0.489\nsamples = 100\nvalue = [66,
89]\nclass = No'),
 Text(0.5384615384615384, 0.125, 'gini = 0.442\nsamples = 57\nvalue = [63, 3
1]\nclass = yes'),
 Text(0.8461538461538461, 0.625, 'ram <= 3815.0\ngini = 0.484\nsamples = 137
\nvalue = [86, 123]\nclass = No'),
 Text(0.7692307692307693, 0.375, 'fc <= 9.5\ngini = 0.493\nsamples = 127\nval
ue = [84, 106]\nclass = No'),
 Text(0.6923076923076923, 0.125, 'gini = 0.478\nsamples = 101\nvalue = [60, 9
2]\nclass = No'),
 Text(0.8461538461538461, 0.125, 'gini = 0.465\nsamples = 26\nvalue = [24, 1
4]\nclass = yes'),
 Text(0.9230769230769231, 0.375, 'gini = 0.188\nsamples = 10\nvalue = [2, 17]
\nclass = No')]
```



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","N
```

Out[28]:
```
[Text(0.625, 0.875, 'mobile_wt <= 195.5\ngini = 0.5\nsamples = 438\nvalue =
[343, 357]\nclass = No'),
 Text(0.5, 0.625, 'px_width <= 938.5\ngini = 0.5\nsamples = 419\nvalue = [34
1, 331]\nclass = yes'),
 Text(0.25, 0.375, 'pc <= 4.5\ngini = 0.481\nsamples = 125\nvalue = [77, 114]
\nclass = No'),
 Text(0.125, 0.125, 'gini = 0.366\nsamples = 34\nvalue = [13, 41]\nclass = N
o'),
 Text(0.375, 0.125, 'gini = 0.498\nsamples = 91\nvalue = [64, 73]\nclass = N
o'),
 Text(0.75, 0.375, 'sc_h <= 8.5\ngini = 0.495\nsamples = 294\nvalue = [264, 2
17]\nclass = yes'),
 Text(0.625, 0.125, 'gini = 0.489\nsamples = 87\nvalue = [59, 80]\nclass = N
o'),
 Text(0.875, 0.125, 'gini = 0.48\nsamples = 207\nvalue = [205, 137]\nclass =
yes'),
 Text(0.75, 0.625, 'gini = 0.133\nsamples = 19\nvalue = [2, 26]\nclass = N
o')]
```



In [29]:
```
rf_best.feature_importances_
```

Out[29]:
```
array([0.04650189, 0.10299328, 0.01439367, 0.        , 0.        ,
       0.05532626, 0.01964821, 0.13808652, 0.03685331, 0.09960983,
       0.03901548, 0.09029733, 0.04591778, 0.10955797, 0.10997793,
       0.0477564 , 0.        , 0.0169304 , 0.00460595, 0.02252778])
```

In [35]:
```
imp_df=pd.DataFrame({"varname":x_train.columns,"Imp":rf_best.feature_importanc
imp_df.sort_values(by="Imp",ascending=False)
```

Out[35]:

	varname	Imp
7	int_memory	0.138087
14	ram	0.109978
13	px_width	0.109558
1	battery_power	0.102993
9	mobile_wt	0.099610
11	pc	0.090297
5	fc	0.055326
15	sc_h	0.047756
0	id	0.046502
12	px_height	0.045918
10	n_cores	0.039015
8	m_dep	0.036853
19	touch_screen	0.022528
6	four_g	0.019648
17	talk_time	0.016930
2	blue	0.014394
18	three_g	0.004606
4	dual_sim	0.000000
3	clock_speed	0.000000
16	sc_w	0.000000

In [ ]:
```

```