In [1]:

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

train

In [2]:

```
train_df=pd.read_csv(r"C:\Users\shaha\OneDrive\Desktop\Excel\Mobile_Price_Classification_
train_df
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_v
0	842	0	2.2	0	1	0	7	0.6	18
1	1021	1	0.5	1	0	1	53	0.7	13
2	563	1	0.5	1	2	1	41	0.9	14
3	615	1	2.5	0	0	0	10	0.8	13
4	1821	1	1.2	0	13	1	44	0.6	14
...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	10
1996	1965	1	2.6	1	0	0	39	0.2	18
1997	1911	0	0.9	1	1	1	36	0.7	10
1998	1512	0	0.9	0	4	1	46	0.1	14
1999	510	1	2.0	1	5	1	45	0.9	16

2000 rows × 21 columns

In [3]:

```
train_df.head()
```

Out[3]:

	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

5 rows × 21 columns

In [4]:

```
train_df.tail()
```

Out[4]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_v
1995	794	1	0.5	1	0	1	2	0.8	10
1996	1965	1	2.6	1	0	0	39	0.2	18
1997	1911	0	0.9	1	1	1	36	0.7	10
1998	1512	0	0.9	0	4	1	46	0.1	14
1999	510	1	2.0	1	5	1	45	0.9	16

5 rows × 21 columns

In [5]:

```
train_df.shape
```

Out[5]:

```
(2000, 21)
```

In [6]:

```
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 [7]:

```
train_df.describe()
```

Out[7]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_m
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.(
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.(
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.1
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.(
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.(
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.(
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.(
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.(

8 rows × 21 columns

In [8]:

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

In [9]:

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

```
0.6    74
1.4    70
1.3    68
1.5    67
2.0    67
1.9    65
0.7    64
2.9    62
1.8    62
1.0    61
1.7    60
2.2    59
0.9    58
2.4    58
0.8    58
1.2    56
2.6    55
2.7    55
1.1    51
3.0    28
```

In [10]:

```
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=25)
x_train.shape
```

Out[10]:

```
(1400, 20)
```

In [11]:

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

Out[11]:

```
RandomForestClassifier
RandomForestClassifier()
```

In [12]:

```
rf=RandomForestClassifier()
```

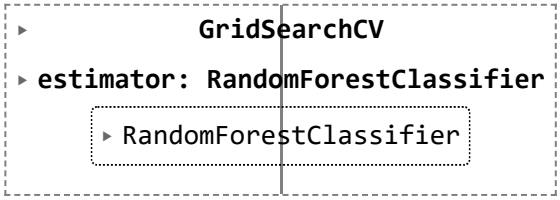
In [13]:

```
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 [14]:

```
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[14]:



In [15]:

```
grid_search.best_score_
```

Out[15]:

```
0.5178571428571428
```

In [16]:

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

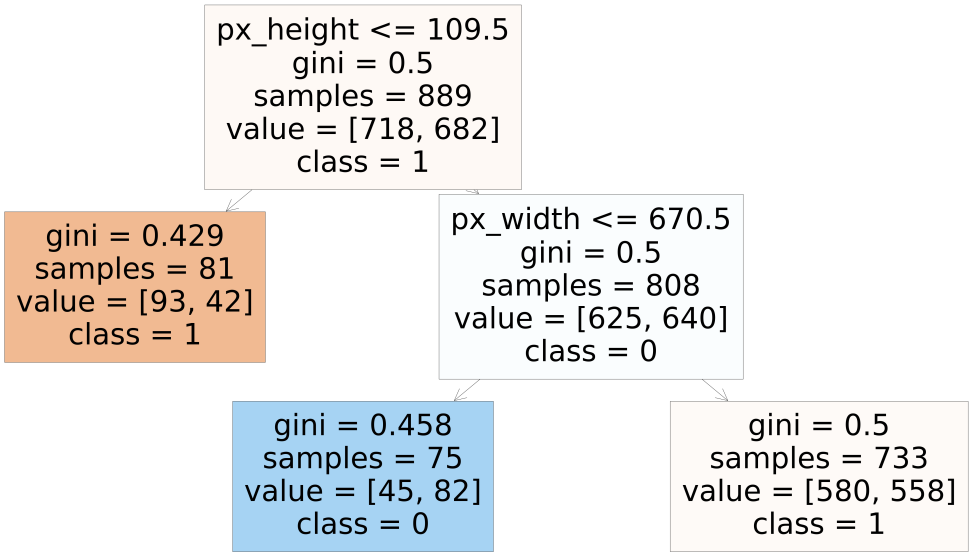
```
RandomForestClassifier(max_depth=2, min_samples_leaf=50, n_estimators=25)
```

In [17]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=['1','0'],filled=Tr
```

Out[17]:

```
[Text(0.4, 0.8333333333333334, 'px_height <= 109.5\ngini = 0.5\nsamples =
889\nvalue = [718, 682]\nclass = 1'),
 Text(0.2, 0.5, 'gini = 0.429\nsamples = 81\nvalue = [93, 42]\nclass =
1'),
 Text(0.6, 0.5, 'px_width <= 670.5\ngini = 0.5\nsamples = 808\nvalue = [62
5, 640]\nclass = 0'),
 Text(0.4, 0.16666666666666666, 'gini = 0.458\nsamples = 75\nvalue = [45,
82]\nclass = 0'),
 Text(0.8, 0.16666666666666666, 'gini = 0.5\nsamples = 733\nvalue = [580,
558]\nclass = 1')]
```



In [18]:

```
rf_best.feature_importances_
```

Out[18]:

```
array([0.0346861 , 0.01871166, 0.01708762, 0.        , 0.12656838,
       0.        , 0.11360976, 0.04030123, 0.05479728, 0.02367189,
       0.03870892, 0.14946964, 0.1034957 , 0.06271748, 0.03405665,
       0.01071626, 0.12214479, 0.        , 0.02576145, 0.0234952 ])
```

In [19]:

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

Out[19]:

	Varname	Imp
11	px_height	0.149470
4	fc	0.126568
16	talk_time	0.122145
6	int_memory	0.113610
12	px_width	0.103496
13	ram	0.062717
8	mobile_wt	0.054797
7	m_dep	0.040301
10	pc	0.038709
0	battery_power	0.034686
14	sc_h	0.034057
18	touch_screen	0.025761
9	n_cores	0.023672
19	price_range	0.023495
1	blue	0.018712
2	clock_speed	0.017088
15	sc_w	0.010716
5	four_g	0.000000
3	dual_sim	0.000000
17	three_g	0.000000

test

In [20]:

```
test_df=pd.read_csv(r"C:\Users\shaha\OneDrive\Desktop\Excel\Mobile_Price_Classification_
test_df
```

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	p
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	1
1	2	841	1	0.5	1	4	1	61	0.8	191	...	1
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	2
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	1
...	...	...	...	...	...	...	...	...	...	...	...	
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	1
996	997	609	0	1.8	1	0	0	13	0.9	186	...	
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	1
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	1
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	1

In [21]:

```
test_df.head()
```

Out[21]:

	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
1	2	841	1	0.5	1	4	1	61	0.8	1
2	3	1807	1	2.8	0	1	0	27	0.9	1
3	4	1546	0	0.5	1	18	1	25	0.5	
4	5	1434	0	1.4	0	11	1	49	0.5	1

5 rows × 21 columns

In [22]:

```
test_df.tail()
```

Out[22]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mo
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	

5 rows × 21 columns

In [23]:

```
test_df.shape
```

Out[23]:

```
(1000, 21)
```

In [24]:

```
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 [25]:

```
test_df.describe()
```

Out[25]:

	id	battery_power	blue	clock_speed	dual_sim	fc	
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	100
mean	500.500000	1248.510000	0.516000	1.540900	0.517000	4.593000	
std	288.819436	432.458227	0.499994	0.829268	0.499961	4.463325	
min	1.000000	500.000000	0.000000	0.500000	0.000000	0.000000	
25%	250.750000	895.000000	0.000000	0.700000	0.000000	1.000000	
50%	500.500000	1246.500000	1.000000	1.500000	1.000000	3.000000	
75%	750.250000	1629.250000	1.000000	2.300000	1.000000	7.000000	
max	1000.000000	1999.000000	1.000000	3.000000	1.000000	19.000000	

8 rows × 21 columns

In [26]:

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

In [27]:

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

Out[27]:

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

In [28]:

```
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_test.shape
```

Out[28]:

```
(300, 20)
```

In [29]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_test,y_test)
```

Out[29]:

```
RandomForestClassifier
RandomForestClassifier()
```

In [30]:

```
rf=RandomForestClassifier()
```

In [31]:

```
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 [32]:

```
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[32]:

```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [33]:

```
grid_search.best_score_
```

Out[33]:

```
0.5866666666666667
```

In [34]:

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

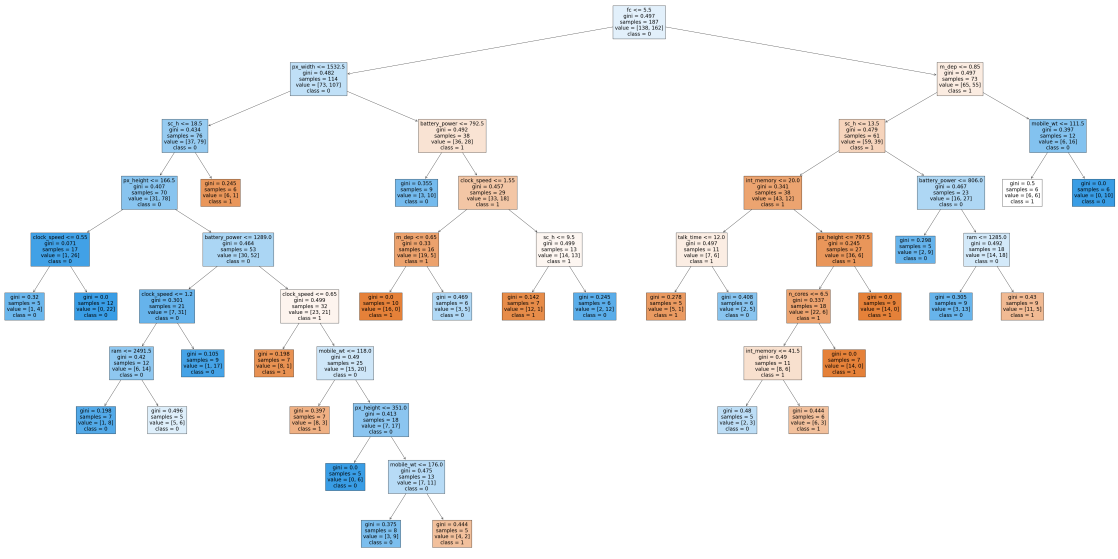
```
RandomForestClassifier(max_depth=10, min_samples_leaf=5, n_estimators=50)
```

In [35]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=['1','0'],filled=Tr
```

Out[35]:

```
[Text(0.5703125, 0.95, 'fc <= 5.5\ngini = 0.497\nsamples = 187\nvalue = [1
38, 162]\nclass = 0'),
 Text(0.2890625, 0.85, 'px_width <= 1532.5\ngini = 0.482\nsamples = 114\nv
alue = [73, 107]\nclass = 0'),
 Text(0.171875, 0.75, 'sc_h <= 18.5\ngini = 0.434\nsamples = 76\nvalue =
[37, 79]\nclass = 0'),
 Text(0.140625, 0.65, 'px_height <= 166.5\ngini = 0.407\nsamples = 70\nval
ue = [31, 78]\nclass = 0'),
 Text(0.0625, 0.55, 'clock_speed <= 0.55\ngini = 0.071\nsamples = 17\nvalu
e = [1, 26]\nclass = 0'),
 Text(0.03125, 0.45, 'gini = 0.32\nsamples = 5\nvalue = [1, 4]\nclass =
0'),
 Text(0.09375, 0.45, 'gini = 0.0\nsamples = 12\nvalue = [0, 22]\nclass =
0'),
 Text(0.21875, 0.55, 'battery_power <= 1289.0\ngini = 0.464\nsamples = 53
\nvalue = [30, 52]\nclass = 0'),
 Text(0.15625, 0.45, 'clock_speed <= 1.2\ngini = 0.301\nsamples = 21\nvalu
e = [7, 31]\nclass = 0'),
 Text(0.125, 0.35, 'ram <= 2491.5\ngini = 0.42\nsamples = 12\nvalue = [6,
14]\nclass = 0'),
 Text(0.09375, 0.25, 'gini = 0.198\nsamples = 7\nvalue = [1, 8]\nclass =
0'),
 Text(0.15625, 0.25, 'gini = 0.496\nsamples = 5\nvalue = [5, 6]\nclass =
0'),
 Text(0.1875, 0.35, 'gini = 0.105\nsamples = 9\nvalue = [1, 17]\nclass =
0'),
 Text(0.28125, 0.45, 'clock_speed <= 0.65\ngini = 0.499\nsamples = 32\nval
ue = [23, 21]\nclass = 1'),
 Text(0.25, 0.35, 'gini = 0.198\nsamples = 7\nvalue = [8, 1]\nclass = 1'),
 Text(0.3125, 0.35, 'mobile_wt <= 118.0\ngini = 0.49\nsamples = 25\nvalue
= [15, 20]\nclass = 0'),
 Text(0.28125, 0.25, 'gini = 0.397\nsamples = 7\nvalue = [8, 3]\nclass =
1'),
 Text(0.34375, 0.25, 'px_height <= 351.0\ngini = 0.413\nsamples = 18\nvalu
e = [7, 17]\nclass = 0'),
 Text(0.3125, 0.15, 'gini = 0.0\nsamples = 5\nvalue = [0, 6]\nclass = 0'),
 Text(0.375, 0.15, 'mobile_wt <= 176.0\ngini = 0.475\nsamples = 13\nvalue
= [7, 11]\nclass = 0'),
 Text(0.34375, 0.05, 'gini = 0.375\nsamples = 8\nvalue = [3, 9]\nclass =
0'),
 Text(0.40625, 0.05, 'gini = 0.444\nsamples = 5\nvalue = [4, 2]\nclass =
1'),
 Text(0.203125, 0.65, 'gini = 0.245\nsamples = 6\nvalue = [6, 1]\nclass =
1'),
```



In [36]:
```
rf_best.feature_importances_
```

Out[36]:
```
array([0.0553984 , 0.07385643, 0.04627033, 0.01135697, 0.05588569,
       0.01024495, 0.05718877, 0.06860488, 0.06803284, 0.02793173,
       0.05475433, 0.11880176, 0.06892912, 0.06977885, 0.06849571,
       0.05028252, 0.04854735, 0.01098116, 0.02127546, 0.01338275])
```

In [37]:
```
imp_df=pd.DataFrame({"Varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[37]:

	Varname	Imp
11	px_height	0.118802
1	battery_power	0.073856
13	ram	0.069779
12	px_width	0.068929
7	m_dep	0.068605
14	sc_h	0.068496
8	mobile_wt	0.068033
6	int_memory	0.057189
4	fc	0.055886
0	id	0.055398
10	pc	0.054754
15	sc_w	0.050283
16	talk_time	0.048547
2	clock_speed	0.046270
9	n_cores	0.027932
18	touch_screen	0.021275
19	wifi	0.013383
3	dual_sim	0.011357
17	three_g	0.010981
5	four_g	0.010245