In [1]:

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

In [2]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Desktop\Mobile_Price_Classification_train.csv")
df
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_c
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]:

```
df.describe()
```

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000

8 rows × 21 columns

In [4]:

```
1 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: 328.3 KB
```

In [5]:

```
1 df.isna().sum()
```

Out[5]:

```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [6]:

```
1 df.isna().any()
```

Out[6]:

```
battery_power    False
blue             False
clock_speed      False
dual_sim         False
fc               False
four_g           False
int_memory       False
m_dep            False
mobile_wt        False
n_cores          False
pc               False
px_height        False
px_width         False
ram              False
sc_h             False
sc_w             False
talk_time        False
three_g          False
touch_screen     False
wifi             False
price_range      False
dtype: bool
```

In [7]:

```
1 df.head()
```

Out[7]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_core
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 [8]:

```
1 df.shape
```

Out[8]:

```
(2000, 21)
```

In [9]:

```
x=df.drop("wifi",axis=1)
y=df["wifi"]
```

In [10]:

```
from sklearn.model_selection import train_test_split
```

In [11]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=30)
x_train.shape,x_test.shape
```

Out[11]:

```
((1400, 20), (600, 20))
```

In [12]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

In [13]:

```
rfc.fit(x_train,y_train)
```

Out[13]:

RandomForestClassifier

RandomForestClassifier()

In [14]:

```
params={"max_depth":[2,4,6,8,9],"min_samples_leaf":[5,10,15,20,25],"n_estimators":[4,8,12
```

In [15]:

```
from sklearn.model_selection import GridSearchCV
```

In [16]:

```
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [17]:

```
grid_search.fit(x_train,y_train)
grid_search.best_score_
```

Out[17]:

```
0.5314285714285714
```

In [18]:

```
1 rfc_best=grid_search.best_estimator_
2 print(rfc_best)
```

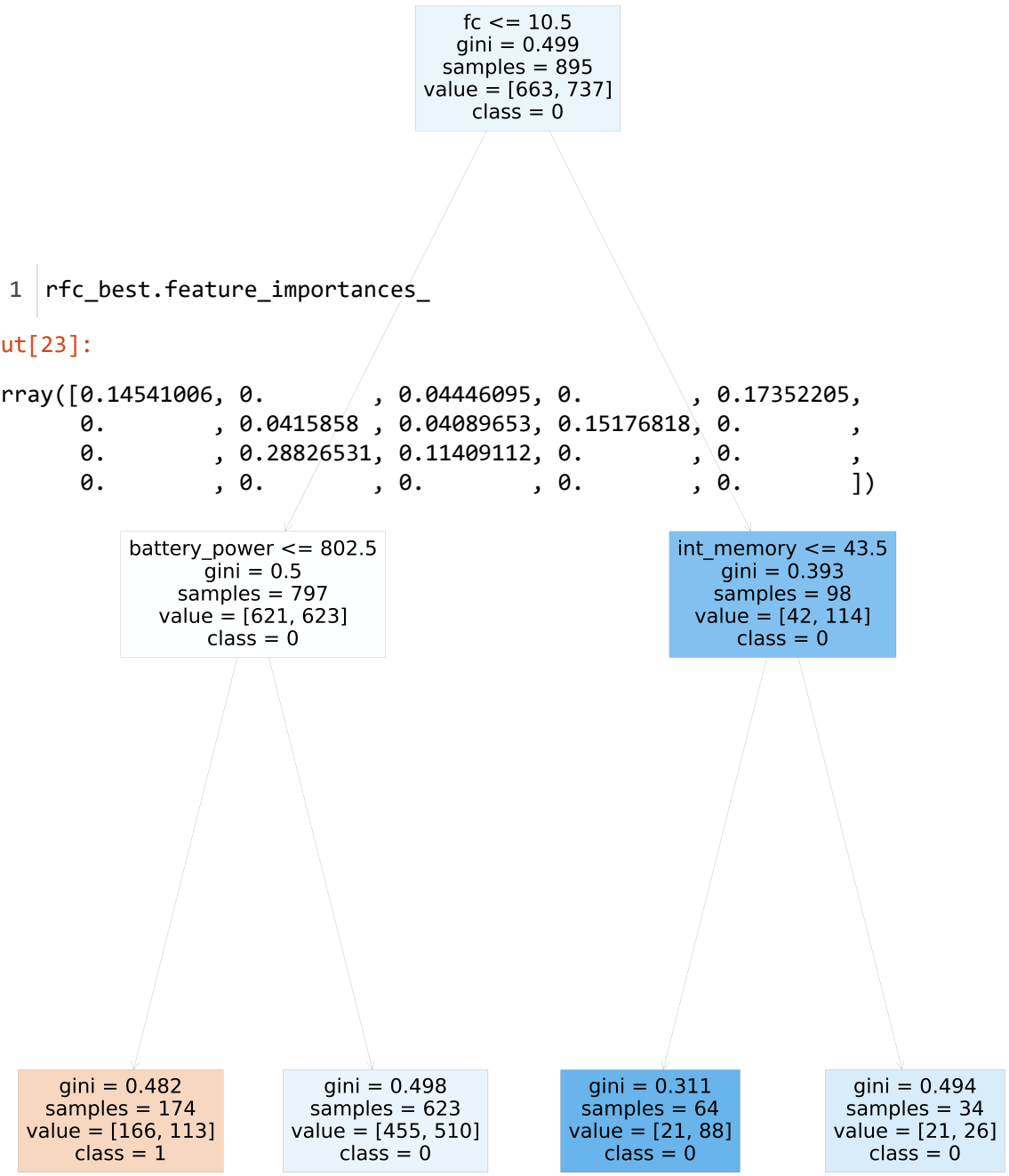
```
RandomForestClassifier(max_depth=2, min_samples_leaf=15, n_estimators=4)
```

In [19]:

```
1 from sklearn.tree import plot_tree
2 plt.figure(figsize=(200,300))
3 plot_tree(rfc_best.estimators_[3],feature_names=x.columns,class_names=["1","0"],filled=Tru
```

Out[19]:

```
[Text(0.5, 0.8333333333333334, 'fc <= 10.5\ngini = 0.499\nsamples = 895\nvalue
= [663, 737]\nclass = 0'),
 Text(0.25, 0.5, 'battery_power <= 802.5\ngini = 0.5\nsamples = 797\nvalue = [6
21, 623]\nclass = 0'),
 Text(0.125, 0.16666666666666666, 'gini = 0.482\nsamples = 174\nvalue = [166, 1
13]\nclass = 1'),
 Text(0.375, 0.16666666666666666, 'gini = 0.498\nsamples = 623\nvalue = [455, 5
10]\nclass = 0'),
 Text(0.75, 0.5, 'int_memory <= 43.5\ngini = 0.393\nsamples = 98\nvalue = [42,
114]\nclass = 0'),
 Text(0.625, 0.16666666666666666, 'gini = 0.311\nsamples = 64\nvalue = [21, 88]
\nclass = 0'),
 Text(0.875, 0.16666666666666666, 'gini = 0.494\nsamples = 34\nvalue = [21, 26]
\nclass = 0')]
```




```
1 rfc_best.feature_importances_
```

Out[23]:

```
array([0.14541006, 0.        , 0.04446095, 0.        , 0.17352205,
       0.        , 0.0415858 , 0.04089653, 0.15176818, 0.        ,
       0.        , 0.28826531, 0.11409112, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ])
```

In [21]:

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

Out[21]:

	varname	Imp
9	n_cores	0.000000
17	three_g	0.000000
16	talk_time	0.000000
15	sc_w	0.000000
14	sc_h	0.000000
13	ram	0.000000
10	pc	0.000000
18	touch_screen	0.000000
19	price_range	0.000000
5	four_g	0.000000
3	dual_sim	0.000000
1	blue	0.000000
7	m_dep	0.040897
6	int_memory	0.041586
2	clock_speed	0.044461
12	px_width	0.114091
0	battery_power	0.145410
8	mobile_wt	0.151768
4	fc	0.173522
11	px_height	0.288265

In [22]:

```
pd.read_csv(r"C:\Users\MY HOME\Desktop\Mobile_Price_Classification_test.csv")
df
```

Out[22]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_c
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 [24]:

```
df.describe()
```

Out[24]:

_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen
.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
.108000	1251.515500	2124.213000	12.306500	5.767000	11.011000	0.761500	0.503000
.780811	432.199447	1084.732044	4.213245	4.356398	5.463955	0.426273	0.500116
.000000	500.000000	256.000000	5.000000	0.000000	2.000000	0.000000	0.000000
.750000	874.750000	1207.500000	9.000000	2.000000	6.000000	1.000000	0.000000
.000000	1247.000000	2146.500000	12.000000	5.000000	11.000000	1.000000	1.000000
.250000	1633.000000	3064.500000	16.000000	9.000000	16.000000	1.000000	1.000000
.000000	1998.000000	3998.000000	19.000000	18.000000	20.000000	1.000000	1.000000

In [25]:

```
1 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: 328.3 KB
```

In [26]:

```
1 df.shape
```

Out[26]:

```
(2000, 21)
```

In [37]:

```
1 x=df.drop("wifi",axis=1)
2 y=df['wifi']
```

In [47]:

```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5,random_state=19)
3 x_train.shape,x_test.shape
4
```

Out[47]:

```
((1000, 20), (1000, 20))
```

In [48]:

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

Out[48]:

RandomForestClassifier

RandomForestClassifier()

In [50]:

```
params={"max_depth":[2,3,4,5,6],"min_samples_leaf":[10,20,30,40,50],"n_estimators":[5,10,
```

In [54]:

```
from sklearn.model_selection import GridSearchCV
```

In [57]:

```
srav=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [63]:

```
srav.fit(x_train,y_train)
srav.best_score_
```

Out[63]:

0.531

In [64]:

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

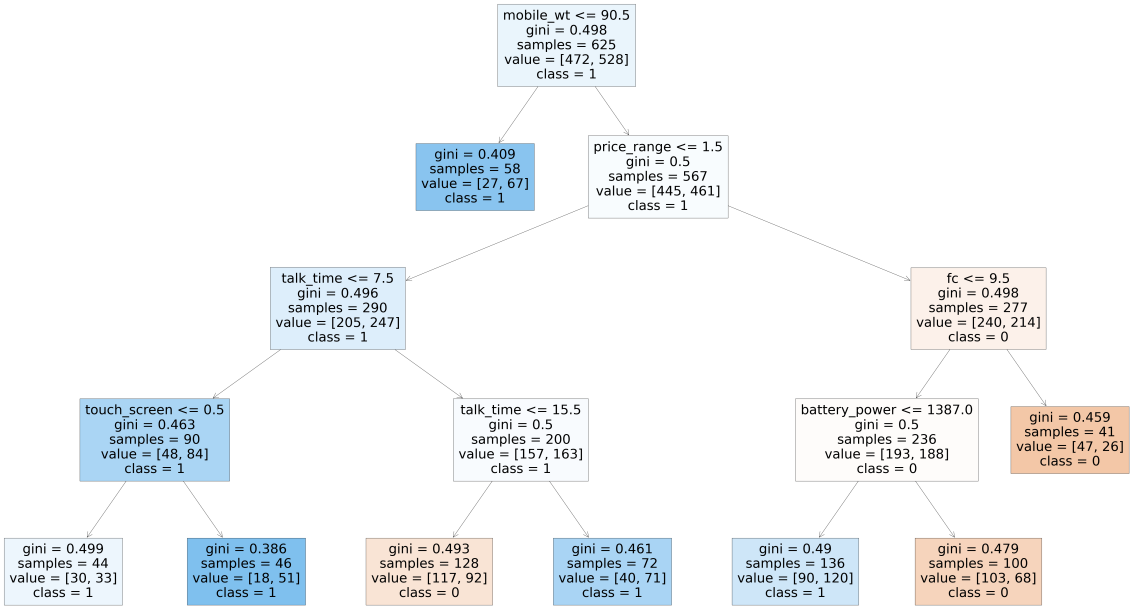
```
RandomForestClassifier(max_depth=4, min_samples_leaf=40, n_estimators=20)
```

In [65]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(90,50))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=["0","1"],filled=Tru
```

Out[65]:

```
[Text(0.5, 0.9, 'mobile_wt <= 90.5\ngini = 0.498\nsamples = 625\nvalue = [472,
528]\nclass = 1'),
 Text(0.4230769230769231, 0.7, 'gini = 0.409\nsamples = 58\nvalue = [27, 67]\nc
lass = 1'),
 Text(0.5769230769230769, 0.7, 'price_range <= 1.5\ngini = 0.5\nsamples = 567\n
value = [445, 461]\nclass = 1'),
 Text(0.3076923076923077, 0.5, 'talk_time <= 7.5\ngini = 0.496\nsamples = 290\n
value = [205, 247]\nclass = 1'),
 Text(0.15384615384615385, 0.3, 'touch_screen <= 0.5\ngini = 0.463\nsamples = 9
0\nvalue = [48, 84]\nclass = 1'),
 Text(0.07692307692307693, 0.1, 'gini = 0.499\nsamples = 44\nvalue = [30, 33]\n
class = 1'),
 Text(0.23076923076923078, 0.1, 'gini = 0.386\nsamples = 46\nvalue = [18, 51]\n
class = 1'),
 Text(0.46153846153846156, 0.3, 'talk_time <= 15.5\ngini = 0.5\nsamples = 200\n
value = [157, 163]\nclass = 1'),
 Text(0.38461538461538464, 0.1, 'gini = 0.493\nsamples = 128\nvalue = [117, 92]
\nclass = 0'),
 Text(0.5384615384615384, 0.1, 'gini = 0.461\nsamples = 72\nvalue = [40, 71]\nc
lass = 1'),
 Text(0.8461538461538461, 0.5, 'fc <= 9.5\ngini = 0.498\nsamples = 277\nvalue =
[240, 214]\nclass = 0'),
 Text(0.7692307692307693, 0.3, 'battery_power <= 1387.0\ngini = 0.5\nsamples =
236\nvalue = [193, 188]\nclass = 0'),
 Text(0.6923076923076923, 0.1, 'gini = 0.49\nsamples = 136\nvalue = [90, 120]\n
class = 1'),
 Text(0.8461538461538461, 0.1, 'gini = 0.479\nsamples = 100\nvalue = [103, 68]
\nclass = 0'),
 Text(0.9230769230769231, 0.3, 'gini = 0.459\nsamples = 41\nvalue = [47, 26]\nc
lass = 0')]
```



In [66]:

```
1 rf_best.feature_importances_
```

Out[66]:

```
array([0.10496438, 0.02410719, 0.01859386, 0.        , 0.04176763,
       0.01393418, 0.06483421, 0.02741537, 0.09176893, 0.05335214,
       0.06062782, 0.19001857, 0.07691346, 0.06012377, 0.00904769,
       0.04244615, 0.089963  , 0.        , 0.00612226, 0.02399943])
```

In [68]:

```
1 imp_df=pd.DataFrame({"varname":x_train.columns,"Imp":rf_best.feature_importances_})
2
```

In [69]:

```
1 imp_df.sort_values(by="Imp",ascending=True)
```

Out[69]:

	varname	Imp
3	dual_sim	0.000000
17	three_g	0.000000
18	touch_screen	0.006122
14	sc_h	0.009048
5	four_g	0.013934
2	clock_speed	0.018594
19	price_range	0.023999
1	blue	0.024107
7	m_dep	0.027415
4	fc	0.041768
15	sc_w	0.042446
9	n_cores	0.053352
13	ram	0.060124
10	pc	0.060628
6	int_memory	0.064834
12	px_width	0.076913
16	talk_time	0.089963
8	mobile_wt	0.091769
0	battery_power	0.104964
11	px_height	0.190019