MOBILE PRICE TRAIN DATASET

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

In [2]:
```
df1=pd.read_csv(r"C:\Users\manis\OneDrive\Pictures\Documents\Mobile_Price_Classi
df1
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	m
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 [3]:
```
df1.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 [4]:
```
df1['price_range'].value_counts()
```

Out[4]:
```
price_range
1    500
2    500
3    500
0    500
Name: count, dtype: int64
```

In [5]:
```
x=df1.drop('blue',axis=1)
y=df1['blue']
```

In [6]:
```
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=4
x_train.shape,x_test.shape
```

Out[6]:
```
((1400, 20), (600, 20))
```

In [7]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

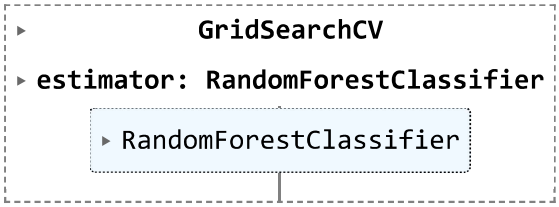
Out[7]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [8]:
```
rf=RandomForestClassifier()
params={'max_depth':[2,3,4,5,6],'min_samples_leaf':[5,10,15,20,50,100],'n_estima
```

In [9]:
```
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[9]:



In [10]:
```
grid_search.best_score_
```

Out[10]:
```
0.5335714285714286
```

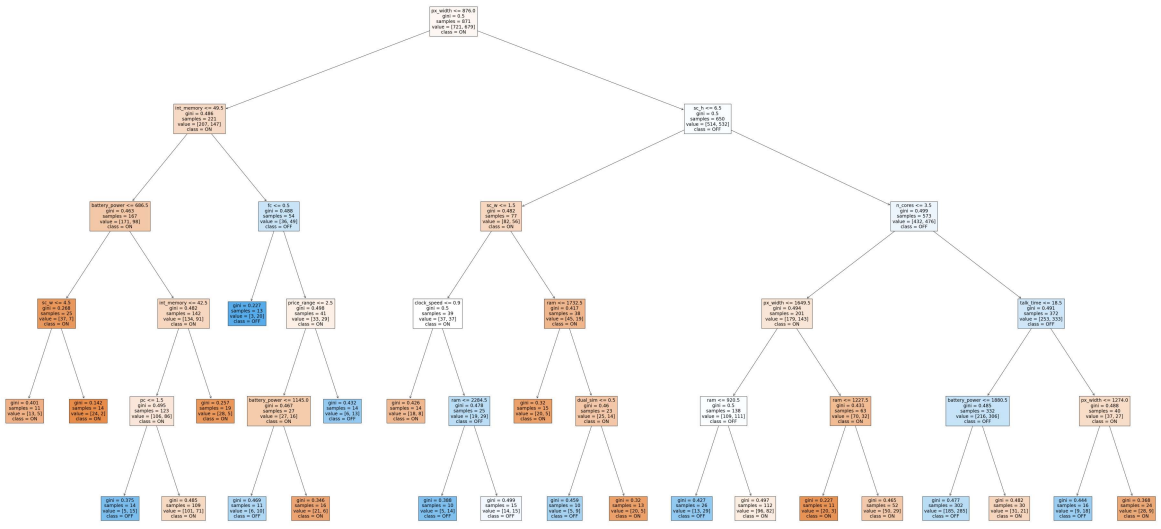
In [11]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

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

In [12]:
```
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=['ON','OFF'
```

Out[12]: [Text(0.3918918918918919, 0.9166666666666666, 'px_width <= 876.0\ngini = 0.5\nsamples = 871\nvalue = [721, 679]\nclass = ON'),
 Text(0.17567567567567569, 0.75, 'int_memory <= 49.5\ngini = 0.486\nsamples = 221\nvalue = [207, 147]\nclass = ON'),
 Text(0.10810810810810811, 0.5833333333333334, 'battery_power <= 686.5\ngini = 0.463\nsamples = 167\nvalue = [171, 98]\nclass = ON'),
 Text(0.05405405405405406, 0.4166666666666667, 'sc_w <= 4.5\ngini = 0.268\nsamples = 25\nvalue = [37, 7]\nclass = ON'),
 Text(0.02702702702702703, 0.25, 'gini = 0.401\nsamples = 11\nvalue = [13, 5]\nclass = ON'),
 Text(0.08108108108108109, 0.25, 'gini = 0.142\nsamples = 14\nvalue = [24, 2]\nclass = ON'),
 Text(0.16216216216216217, 0.4166666666666667, 'int_memory <= 42.5\ngini = 0.482\nsamples = 142\nvalue = [134, 91]\nclass = ON'),
 Text(0.13513513513513514, 0.25, 'pc <= 1.5\ngini = 0.495\nsamples = 123\nvalue = [106, 86]\nclass = ON'),
 Text(0.10810810810810811, 0.08333333333333333, 'gini = 0.375\nsamples = 14\nvalue = [5, 15]\nclass = OFF'),
 Text(0.16216216216216217, 0.08333333333333333, 'gini = 0.485\nsamples = 109\nvalue = [101, 71]\nclass = ON'),
 Text(0.1891891891891892, 0.25, 'gini = 0.257\nsamples = 19\nvalue = [28, 5]\nclass = ON'),
 Text(0.24324324324324326, 0.5833333333333334, 'fc <= 0.5\ngini = 0.488\nsamples = 54\nvalue = [36, 49]\nclass = OFF'),
 Text(0.21621621621621623, 0.4166666666666667, 'gini = 0.227\nsamples = 13\nvalue = [3, 20]\nclass = OFF'),
 Text(0.2702702702702703, 0.4166666666666667, 'price_range <= 2.5\ngini = 0.498\nsamples = 41\nvalue = [33, 29]\nclass = ON'),
 Text(0.24324324324324326, 0.25, 'battery_power <= 1145.0\ngini = 0.467\nsamples = 27\nvalue = [27, 16]\nclass = ON'),
 Text(0.21621621621621623, 0.08333333333333333, 'gini = 0.469\nsamples = 11\nvalue = [6, 10]\nclass = OFF'),
 Text(0.2702702702702703, 0.08333333333333333, 'gini = 0.346\nsamples = 16\nvalue = [21, 6]\nclass = ON'),
 Text(0.2972972972972973, 0.25, 'gini = 0.432\nsamples = 14\nvalue = [6, 13]\nclass = OFF'),
 Text(0.6081081081081081, 0.75, 'sc_h <= 6.5\ngini = 0.5\nsamples = 650\nvalue = [514, 532]\nclass = OFF'),
 Text(0.43243243243243246, 0.5833333333333334, 'sc_w <= 1.5\ngini = 0.482\nsamples = 77\nvalue = [82, 56]\nclass = ON'),
 Text(0.3783783783783784, 0.4166666666666667, 'clock_speed <= 0.9\ngini = 0.5\nsamples = 39\nvalue = [37, 37]\nclass = ON'),
 Text(0.35135135135135137, 0.25, 'gini = 0.426\nsamples = 14\nvalue = [18, 8]\nclass = ON'),
 Text(0.40540540540540543, 0.25, 'ram <= 2284.5\ngini = 0.478\nsamples = 25\nvalue = [19, 29]\nclass = OFF'),
 Text(0.3783783783783784, 0.08333333333333333, 'gini = 0.388\nsamples = 10\nvalue = [5, 14]\nclass = OFF'),
 Text(0.43243243243243246, 0.08333333333333333, 'gini = 0.499\nsamples = 15\nvalue = [14, 15]\nclass = OFF'),
 Text(0.4864864864864865, 0.4166666666666667, 'ram <= 1732.5\ngini = 0.417\nsamples = 38\nvalue = [45, 19]\nclass = ON'),
 Text(0.4594594594594595, 0.25, 'gini = 0.32\nsamples = 15\nvalue = [20, 5]\nclass = ON'),
 Text(0.5135135135135135, 0.25, 'dual_sim <= 0.5\ngini = 0.46\nsamples = 23\nvalue = [25, 14]\nclass = ON'),
 Text(0.4864864864864865, 0.08333333333333333, 'gini = 0.459\nsamples = 10\nvalue = [5, 9]\nclass = OFF'),
 Text(0.5405405405405406, 0.08333333333333333, 'gini = 0.32\nsamples = 13\nvalue = [20, 5]\nclass = ON'),

```
 Text(0.7837837837837838, 0.5833333333333334, 'n_cores <= 3.5\ngini = 0.499\nsa
mples = 573\nvalue = [432, 476]\nclass = OFF'),
 Text(0.6756756756756757, 0.4166666666666667, 'px_width <= 1649.5\ngini = 0.494
\nsamples = 201\nvalue = [179, 143]\nclass = ON'),
 Text(0.6216216216216216, 0.25, 'ram <= 920.5\ngini = 0.5\nsamples = 138\nvalue
= [109, 111]\nclass = OFF'),
 Text(0.5945945945945946, 0.08333333333333333, 'gini = 0.427\nsamples = 26\nval
ue = [13, 29]\nclass = OFF'),
 Text(0.6486486486486487, 0.08333333333333333, 'gini = 0.497\nsamples = 112\nva
lue = [96, 82]\nclass = ON'),
 Text(0.7297297297297297, 0.25, 'ram <= 1227.5\ngini = 0.431\nsamples = 63\nval
ue = [70, 32]\nclass = ON'),
 Text(0.7027027027027027, 0.08333333333333333, 'gini = 0.227\nsamples = 11\nval
ue = [20, 3]\nclass = ON'),
 Text(0.7567567567567568, 0.08333333333333333, 'gini = 0.465\nsamples = 52\nval
ue = [50, 29]\nclass = ON'),
 Text(0.8918918918918919, 0.4166666666666667, 'talk_time <= 18.5\ngini = 0.491
\nsamples = 372\nvalue = [253, 333]\nclass = OFF'),
 Text(0.8378378378378378, 0.25, 'battery_power <= 1880.5\ngini = 0.485\nsamples
= 332\nvalue = [216, 306]\nclass = OFF'),
 Text(0.8108108108108109, 0.08333333333333333, 'gini = 0.477\nsamples = 302\nva
lue = [185, 285]\nclass = OFF'),
 Text(0.8648648648648649, 0.08333333333333333, 'gini = 0.482\nsamples = 30\nval
ue = [31, 21]\nclass = ON'),
 Text(0.9459459459459459, 0.25, 'px_width <= 1274.0\ngini = 0.488\nsamples = 40
\nvalue = [37, 27]\nclass = ON'),
 Text(0.918918918918919, 0.08333333333333333, 'gini = 0.444\nsamples = 16\nvalu
e = [9, 18]\nclass = OFF'),
 Text(0.972972972972973, 0.08333333333333333, 'gini = 0.368\nsamples = 24\nvalu
e = [28, 9]\nclass = ON')]
```



In [13]:
```
rf_best.feature_importances_
imp_df=pd.DataFrame({"Varname":x_train.columns,"Imp":rf_best.feature_importances
imp_df.sort_values(by="Imp",ascending=False)
```

Out[13]:

	Varname	Imp
0	battery_power	0.114392
11	px_width	0.099486
5	int_memory	0.093574
10	px_height	0.087005
12	ram	0.085032
7	mobile_wt	0.076003
14	sc_w	0.067118
1	clock_speed	0.057733
15	talk_time	0.048170
9	pc	0.047710
13	sc_h	0.042458
6	m_dep	0.042250
3	fc	0.039663
8	n_cores	0.038240
19	price_range	0.018187
2	dual_sim	0.015327
16	three_g	0.010524
18	wifi	0.007785
17	touch_screen	0.005590
4	four_g	0.003755

MOBILE PRICE TEST DATASET

In [14]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt,seaborn as sns
```

In [15]:
```
df=pd.read_csv(r"C:\Users\manis\OneDrive\Pictures\Documents\Mobile_Price_Classif
df
```

Out[15]:

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

1000 rows × 21 columns

In [16]:
```
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 [21]:
```
df['m_dep'].value_counts()
```

Out[21]:
```
m_dep
0.1    139
0.5    122
0.9    107
0.8    105
0.7    101
0.6     98
0.4     96
0.2     95
0.3     88
1.0     49
Name: count, dtype: int64
```

In [22]:
```
x=df.drop('wifi',axis=1)
y=df['wifi']
```

In [25]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.7,random_state=42
x_train.shape,x_test.shape
```

Out[25]:
```
((300, 20), (700, 20))
```

In [28]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_test,y_test)
```

Out[28]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [35]:
```
rf=RandomForestClassifier()
params={'max_depth':[1,3,4,7,9],'min_samples_leaf':[5,10,15,25,55,105],'n_estima
```

In [36]:
```
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[36]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [37]:
```
grid_search.best_score_
```

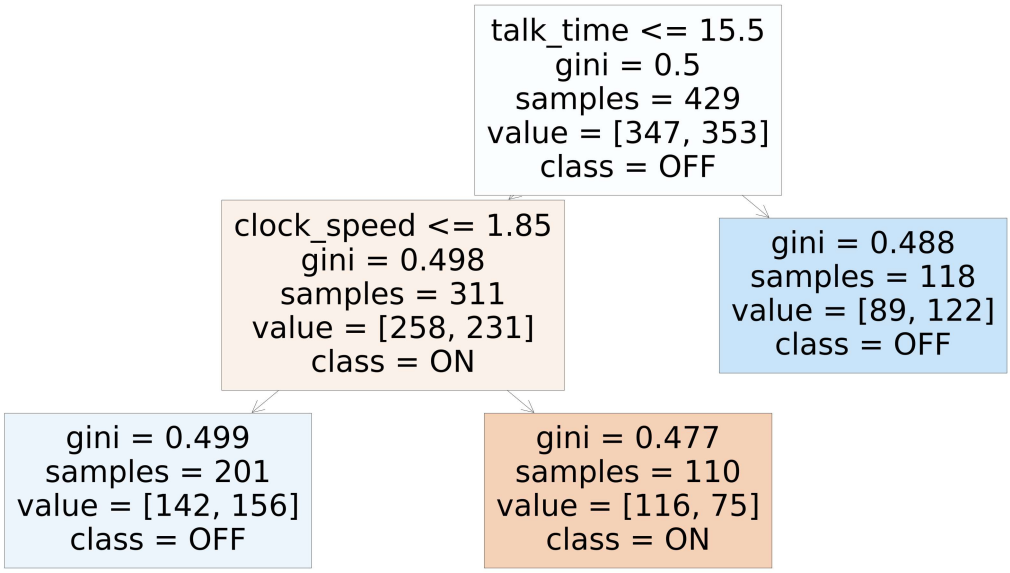
Out[37]:
```
0.5485714285714286
```

In [38]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=4, min_samples_leaf=105, n_estimators=10)
```

In [39]:
```
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=['ON','OFF'
```

Out[39]:
```
[Text(0.6, 0.8333333333333334, 'talk_time <= 15.5\ngini = 0.5\nsamples = 429\nv
alue = [347, 353]\nclass = OFF'),
 Text(0.4, 0.5, 'clock_speed <= 1.85\ngini = 0.498\nsamples = 311\nvalue = [25
8, 231]\nclass = ON'),
 Text(0.2, 0.16666666666666666, 'gini = 0.499\nsamples = 201\nvalue = [142, 15
6]\nclass = OFF'),
 Text(0.6, 0.16666666666666666, 'gini = 0.477\nsamples = 110\nvalue = [116, 75]
\nclass = ON'),
 Text(0.8, 0.5, 'gini = 0.488\nsamples = 118\nvalue = [89, 122]\nclass = OFF')]
```



In [40]:
```
rf_best.feature_importances_
imp_df=pd.DataFrame({"Varname":x_test.columns,"Imp":rf_best.feature_importances_
imp_df.sort_values(by="Imp",ascending=False)
```

Out[40]:

	Varname	Imp
13	px_width	0.321374
15	sc_h	0.167529
9	mobile_wt	0.094973
12	px_height	0.082146
14	ram	0.071698
3	clock_speed	0.054693
17	talk_time	0.054383
0	id	0.050554
10	n_cores	0.049446
16	sc_w	0.032471
8	m_dep	0.020733
1	battery_power	0.000000
11	pc	0.000000
7	int_memory	0.000000
6	four_g	0.000000
5	fc	0.000000
4	dual_sim	0.000000
2	blue	0.000000
18	three_g	0.000000
19	touch_screen	0.000000

In [ ]: