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['m_dep'].value_counts()
```

Out[4]:
```
m_dep
0.1    320
0.2    213
0.8    208
0.5    205
0.7    200
0.3    199
0.9    195
0.6    186
0.4    168
1.0    106
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]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

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

Out[10]:
```
0.535
```

In [28]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

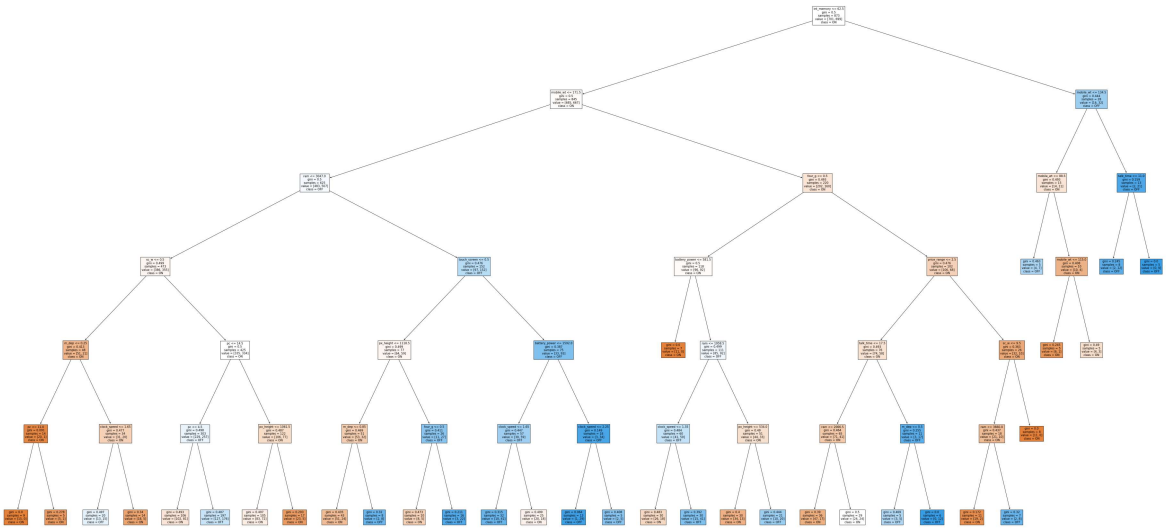
```
RandomForestClassifier(max_depth=6, min_samples_leaf=5, n_estimators=10)
```

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=['ON','OFF'
```

Out[30]:
```
[Text(0.7086864406779662, 0.9285714285714286, 'int_memory <= 62.5\ngini = 0.5\n
samples = 873\nvalue = [701, 699]\nclass = ON'),
 Text(0.4851694915254237, 0.7857142857142857, 'mobile_wt <= 171.5\ngini = 0.5\n
samples = 845\nvalue = [685, 667]\nclass = ON'),
 Text(0.2711864406779661, 0.6428571428571429, 'ram <= 3047.0\ngini = 0.5\nsampl
es = 625\nvalue = [483, 507]\nclass = OFF'),
 Text(0.13559322033898305, 0.5, 'sc_w <= 0.5\ngini = 0.499\nsamples = 473\nvalu
e = [386, 355]\nclass = ON'),
 Text(0.06779661016949153, 0.35714285714285715, 'm_dep <= 0.25\ngini = 0.413\ns
amples = 48\nvalue = [51, 21]\nclass = ON'),
 Text(0.03389830508474576, 0.21428571428571427, 'pc <= 11.0\ngini = 0.091\nsamp
les = 14\nvalue = [20, 1]\nclass = ON'),
 Text(0.01694915254237288, 0.07142857142857142, 'gini = 0.0\nsamples = 9\nvalue
= [15, 0]\nclass = ON'),
 Text(0.05084745762711865, 0.07142857142857142, 'gini = 0.278\nsamples = 5\nval
ue = [5, 1]\nclass = ON'),
 Text(0.1016949152542373, 0.21428571428571427, 'clock_speed <= 1.65\ngini = 0.4
77\nsamples = 34\nvalue = [31, 20]\nclass = ON'),
 Text(0.0847457627118644, 0.07142857142857142, 'gini = 0.497\nsamples = 20\nval
ue = [13, 15]\nclass = OFF'),
 Text(0.11864406779661017, 0.07142857142857142, 'gini = 0.34\nsamples = 14\nval
ue = [18, 5]\nclass = ON'),
 Text(0.2033898305084746, 0.35714285714285715, 'pc <= 14.5\ngini = 0.5\nsamples
= 425\nvalue = [335, 334]\nclass = ON'),
 Text(0.1694915254237288, 0.21428571428571427, 'pc <= 4.5\ngini = 0.498\nsample
s = 303\nvalue = [229, 257]\nclass = OFF'),
 Text(0.15254237288135594, 0.07142857142857142, 'gini = 0.493\nsamples = 106\nv
alue = [102, 81]\nclass = ON'),
 Text(0.1864406779661017, 0.07142857142857142, 'gini = 0.487\nsamples = 197\nva
lue = [127, 176]\nclass = OFF'),
 Text(0.23728813559322035, 0.21428571428571427, 'px_height <= 1061.5\ngini = 0.
487\nsamples = 122\nvalue = [106, 77]\nclass = ON'),
 Text(0.22033898305084745, 0.07142857142857142, 'gini = 0.497\nsamples = 105\nv
alue = [83, 72]\nclass = ON'),
 Text(0.2542372881355932, 0.07142857142857142, 'gini = 0.293\nsamples = 17\nval
ue = [23, 5]\nclass = ON'),
 Text(0.4067796610169492, 0.5, 'touch_screen <= 0.5\ngini = 0.476\nsamples = 15
2\nvalue = [97, 152]\nclass = OFF'),
 Text(0.3389830508474576, 0.35714285714285715, 'px_height <= 1118.5\ngini = 0.4
99\nsamples = 77\nvalue = [64, 59]\nclass = ON'),
 Text(0.3050847457627119, 0.21428571428571427, 'm_dep <= 0.85\ngini = 0.469\nsa
mples = 51\nvalue = [53, 32]\nclass = ON'),
 Text(0.288135593220339, 0.07142857142857142, 'gini = 0.435\nsamples = 43\nvalu
e = [51, 24]\nclass = ON'),
 Text(0.3220338983050847, 0.07142857142857142, 'gini = 0.32\nsamples = 8\nvalue
= [2, 8]\nclass = OFF'),
 Text(0.3728813559322034, 0.21428571428571427, 'four_g <= 0.5\ngini = 0.411\nsa
mples = 26\nvalue = [11, 27]\nclass = OFF'),
 Text(0.3559322033898305, 0.07142857142857142, 'gini = 0.473\nsamples = 10\nval
ue = [8, 5]\nclass = ON'),
 Text(0.3898305084745763, 0.07142857142857142, 'gini = 0.211\nsamples = 16\nval
ue = [3, 22]\nclass = OFF'),
 Text(0.4745762711864407, 0.35714285714285715, 'battery_power <= 1592.0\ngini =
0.387\nsamples = 75\nvalue = [33, 93]\nclass = OFF'),
 Text(0.4406779661016949, 0.21428571428571427, 'clock_speed <= 1.65\ngini = 0.4
47\nsamples = 57\nvalue = [30, 59]\nclass = OFF'),
 Text(0.423728813559322, 0.07142857142857142, 'gini = 0.315\nsamples = 32\nvalu
e = [10, 41]\nclass = OFF'),
 Text(0.4576271186440678, 0.07142857142857142, 'gini = 0.499\nsamples = 25\nval
ue = [20, 18]\nclass = ON'),
```

```
 Text(0.5084745762711864, 0.21428571428571427, 'clock_speed <= 2.25\ngini = 0.1
49\nsamples = 18\nvalue = [3, 34]\nclass = OFF'),
 Text(0.4915254237288136, 0.07142857142857142, 'gini = 0.064\nsamples = 13\nval
ue = [1, 29]\nclass = OFF'),
 Text(0.5254237288135594, 0.07142857142857142, 'gini = 0.408\nsamples = 5\nvalu
e = [2, 5]\nclass = OFF'),
 Text(0.6991525423728814, 0.6428571428571429, 'four_g <= 0.5\ngini = 0.493\nsam
ples = 220\nvalue = [202, 160]\nclass = ON'),
 Text(0.5932203389830508, 0.5, 'battery_power <= 581.5\ngini = 0.5\nsamples = 1
18\nvalue = [96, 92]\nclass = ON'),
 Text(0.576271186440678, 0.35714285714285715, 'gini = 0.0\nsamples = 7\nvalue =
[11, 0]\nclass = ON'),
 Text(0.6101694915254238, 0.35714285714285715, 'ram <= 1958.5\ngini = 0.499\nsa
mples = 111\nvalue = [85, 92]\nclass = OFF'),
 Text(0.576271186440678, 0.21428571428571427, 'clock_speed <= 1.35\ngini = 0.48
4\nsamples = 60\nvalue = [41, 59]\nclass = OFF'),
 Text(0.559322033898305, 0.07142857142857142, 'gini = 0.483\nsamples = 30\nvalu
e = [26, 18]\nclass = ON'),
 Text(0.5932203389830508, 0.07142857142857142, 'gini = 0.392\nsamples = 30\nval
ue = [15, 41]\nclass = OFF'),
 Text(0.6440677966101694, 0.21428571428571427, 'px_height <= 534.0\ngini = 0.49
\nsamples = 51\nvalue = [44, 33]\nclass = ON'),
 Text(0.6271186440677966, 0.07142857142857142, 'gini = 0.4\nsamples = 30\nvalue
= [34, 13]\nclass = ON'),
 Text(0.6610169491525424, 0.07142857142857142, 'gini = 0.444\nsamples = 21\nval
ue = [10, 20]\nclass = OFF'),
 Text(0.8050847457627118, 0.5, 'price_range <= 2.5\ngini = 0.476\nsamples = 102
\nvalue = [106, 68]\nclass = ON'),
 Text(0.7457627118644068, 0.35714285714285715, 'talk_time <= 17.5\ngini = 0.493
\nsamples = 76\nvalue = [74, 58]\nclass = ON'),
 Text(0.711864406779661, 0.21428571428571427, 'ram <= 2066.5\ngini = 0.464\nsam
ples = 65\nvalue = [71, 41]\nclass = ON'),
 Text(0.6949152542372882, 0.07142857142857142, 'gini = 0.39\nsamples = 36\nvalu
e = [47, 17]\nclass = ON'),
 Text(0.7288135593220338, 0.07142857142857142, 'gini = 0.5\nsamples = 29\nvalue
= [24, 24]\nclass = ON'),
 Text(0.7796610169491526, 0.21428571428571427, 'm_dep <= 0.5\ngini = 0.255\nsam
ples = 11\nvalue = [3, 17]\nclass = OFF'),
 Text(0.7627118644067796, 0.07142857142857142, 'gini = 0.469\nsamples = 5\nvalu
e = [3, 5]\nclass = OFF'),
 Text(0.7966101694915254, 0.07142857142857142, 'gini = 0.0\nsamples = 6\nvalue
= [0, 12]\nclass = OFF'),
 Text(0.864406779661017, 0.35714285714285715, 'sc_w <= 9.5\ngini = 0.363\nsampl
es = 26\nvalue = [32, 10]\nclass = ON'),
 Text(0.847457627118644, 0.21428571428571427, 'ram <= 3660.0\ngini = 0.437\nsam
ples = 18\nvalue = [21, 10]\nclass = ON'),
 Text(0.8305084745762712, 0.07142857142857142, 'gini = 0.172\nsamples = 11\nval
ue = [19, 2]\nclass = ON'),
 Text(0.864406779661017, 0.07142857142857142, 'gini = 0.32\nsamples = 7\nvalue
= [2, 8]\nclass = OFF'),
 Text(0.8813559322033898, 0.21428571428571427, 'gini = 0.0\nsamples = 8\nvalue
= [11, 0]\nclass = ON'),
 Text(0.9322033898305084, 0.7857142857142857, 'mobile_wt <= 134.5\ngini = 0.444
\nsamples = 28\nvalue = [16, 32]\nclass = OFF'),
 Text(0.8983050847457628, 0.6428571428571429, 'mobile_wt <= 88.5\ngini = 0.493
\nsamples = 15\nvalue = [14, 11]\nclass = ON'),
 Text(0.8813559322033898, 0.5, 'gini = 0.463\nsamples = 5\nvalue = [4, 7]\nclas
s = OFF'),
 Text(0.9152542372881356, 0.5, 'mobile_wt <= 115.0\ngini = 0.408\nsamples = 10
\nvalue = [10, 4]\nclass = ON'),
```

```
 Text(0.8983050847457628, 0.35714285714285715, 'gini = 0.245\nsamples = 5\nvalu
e = [6, 1]\nclass = ON'),
 Text(0.9322033898305084, 0.35714285714285715, 'gini = 0.49\nsamples = 5\nvalue
= [4, 3]\nclass = ON'),
 Text(0.9661016949152542, 0.6428571428571429, 'talk_time <= 11.0\ngini = 0.159
\nsamples = 13\nvalue = [2, 21]\nclass = OFF'),
 Text(0.9491525423728814, 0.5, 'gini = 0.245\nsamples = 8\nvalue = [2, 12]\ncla
ss = OFF'),
 Text(0.9830508474576272, 0.5, 'gini = 0.0\nsamples = 5\nvalue = [0, 9]\nclass
= OFF')]
```



In [31]:
```
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[31]:

	Varname	Imp
12	ram	0.149274
10	px_height	0.093226
1	clock_speed	0.085026
0	battery_power	0.075030
11	px_width	0.073181
5	int_memory	0.072666
9	pc	0.068831
7	mobile_wt	0.057700
15	talk_time	0.056695
14	sc_w	0.053751
13	sc_h	0.052998
6	m_dep	0.037354
3	fc	0.031398
8	n_cores	0.022649
17	touch_screen	0.018273
18	wifi	0.017269
16	three_g	0.015181
2	dual_sim	0.010107
4	four_g	0.005861
19	price_range	0.003530

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 [17]:
```
df['m_dep'].value_counts()
```

Out[17]:
```
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 [18]:
```
x=df1.drop('blue',axis=1)
y=df1['blue']
```

In [19]:
```
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[19]:
```
((600, 20), (1400, 20))
```

In [20]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_test,y_test)
```

Out[20]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [21]:
```
rf=RandomForestClassifier()
params={'max_depth':[2,3,4,5,6],'min_samples_leaf':[5,10,15,20,50,100],'n_estima
```

In [22]:
```
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[22]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [23]:
```
grid_search.best_score_
```

Out[23]:
```
0.5342857142857143
```

In [24]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

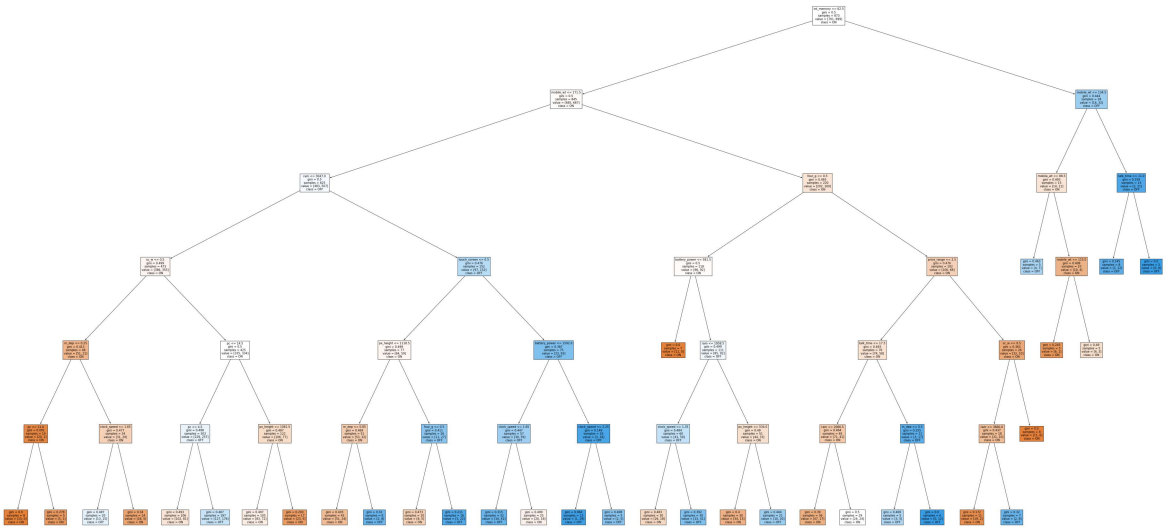
```
RandomForestClassifier(max_depth=6, min_samples_leaf=5, n_estimators=10)
```

In [25]:
```
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[25]:
```
[Text(0.7086864406779662, 0.9285714285714286, 'int_memory <= 62.5\ngini = 0.5\n
samples = 873\nvalue = [701, 699]\nclass = ON'),
 Text(0.4851694915254237, 0.7857142857142857, 'mobile_wt <= 171.5\ngini = 0.5\n
samples = 845\nvalue = [685, 667]\nclass = ON'),
 Text(0.2711864406779661, 0.6428571428571429, 'ram <= 3047.0\ngini = 0.5\nsampl
es = 625\nvalue = [483, 507]\nclass = OFF'),
 Text(0.13559322033898305, 0.5, 'sc_w <= 0.5\ngini = 0.499\nsamples = 473\nvalu
e = [386, 355]\nclass = ON'),
 Text(0.06779661016949153, 0.35714285714285715, 'm_dep <= 0.25\ngini = 0.413\ns
amples = 48\nvalue = [51, 21]\nclass = ON'),
 Text(0.03389830508474576, 0.21428571428571427, 'pc <= 11.0\ngini = 0.091\nsamp
les = 14\nvalue = [20, 1]\nclass = ON'),
 Text(0.01694915254237288, 0.07142857142857142, 'gini = 0.0\nsamples = 9\nvalue
= [15, 0]\nclass = ON'),
 Text(0.05084745762711865, 0.07142857142857142, 'gini = 0.278\nsamples = 5\nval
ue = [5, 1]\nclass = ON'),
 Text(0.1016949152542373, 0.21428571428571427, 'clock_speed <= 1.65\ngini = 0.4
77\nsamples = 34\nvalue = [31, 20]\nclass = ON'),
 Text(0.0847457627118644, 0.07142857142857142, 'gini = 0.497\nsamples = 20\nval
ue = [13, 15]\nclass = OFF'),
 Text(0.11864406779661017, 0.07142857142857142, 'gini = 0.34\nsamples = 14\nval
ue = [18, 5]\nclass = ON'),
 Text(0.2033898305084746, 0.35714285714285715, 'pc <= 14.5\ngini = 0.5\nsamples
= 425\nvalue = [335, 334]\nclass = ON'),
 Text(0.1694915254237288, 0.21428571428571427, 'pc <= 4.5\ngini = 0.498\nsample
s = 303\nvalue = [229, 257]\nclass = OFF'),
 Text(0.15254237288135594, 0.07142857142857142, 'gini = 0.493\nsamples = 106\nv
alue = [102, 81]\nclass = ON'),
 Text(0.1864406779661017, 0.07142857142857142, 'gini = 0.487\nsamples = 197\nva
lue = [127, 176]\nclass = OFF'),
 Text(0.23728813559322035, 0.21428571428571427, 'px_height <= 1061.5\ngini = 0.
487\nsamples = 122\nvalue = [106, 77]\nclass = ON'),
 Text(0.22033898305084745, 0.07142857142857142, 'gini = 0.497\nsamples = 105\nv
alue = [83, 72]\nclass = ON'),
 Text(0.2542372881355932, 0.07142857142857142, 'gini = 0.293\nsamples = 17\nval
ue = [23, 5]\nclass = ON'),
 Text(0.4067796610169492, 0.5, 'touch_screen <= 0.5\ngini = 0.476\nsamples = 15
2\nvalue = [97, 152]\nclass = OFF'),
 Text(0.3389830508474576, 0.35714285714285715, 'px_height <= 1118.5\ngini = 0.4
99\nsamples = 77\nvalue = [64, 59]\nclass = ON'),
 Text(0.3050847457627119, 0.21428571428571427, 'm_dep <= 0.85\ngini = 0.469\nsa
mples = 51\nvalue = [53, 32]\nclass = ON'),
 Text(0.288135593220339, 0.07142857142857142, 'gini = 0.435\nsamples = 43\nvalu
e = [51, 24]\nclass = ON'),
 Text(0.3220338983050847, 0.07142857142857142, 'gini = 0.32\nsamples = 8\nvalue
= [2, 8]\nclass = OFF'),
 Text(0.3728813559322034, 0.21428571428571427, 'four_g <= 0.5\ngini = 0.411\nsa
mples = 26\nvalue = [11, 27]\nclass = OFF'),
 Text(0.3559322033898305, 0.07142857142857142, 'gini = 0.473\nsamples = 10\nval
ue = [8, 5]\nclass = ON'),
 Text(0.3898305084745763, 0.07142857142857142, 'gini = 0.211\nsamples = 16\nval
ue = [3, 22]\nclass = OFF'),
 Text(0.4745762711864407, 0.35714285714285715, 'battery_power <= 1592.0\ngini =
0.387\nsamples = 75\nvalue = [33, 93]\nclass = OFF'),
 Text(0.4406779661016949, 0.21428571428571427, 'clock_speed <= 1.65\ngini = 0.4
47\nsamples = 57\nvalue = [30, 59]\nclass = OFF'),
 Text(0.423728813559322, 0.07142857142857142, 'gini = 0.315\nsamples = 32\nvalu
e = [10, 41]\nclass = OFF'),
 Text(0.4576271186440678, 0.07142857142857142, 'gini = 0.499\nsamples = 25\nval
ue = [20, 18]\nclass = ON'),
```

```
 Text(0.5084745762711864, 0.21428571428571427, 'clock_speed <= 2.25\ngini = 0.1
49\nsamples = 18\nvalue = [3, 34]\nclass = OFF'),
 Text(0.4915254237288136, 0.07142857142857142, 'gini = 0.064\nsamples = 13\nval
ue = [1, 29]\nclass = OFF'),
 Text(0.5254237288135594, 0.07142857142857142, 'gini = 0.408\nsamples = 5\nvalu
e = [2, 5]\nclass = OFF'),
 Text(0.6991525423728814, 0.6428571428571429, 'four_g <= 0.5\ngini = 0.493\nsam
ples = 220\nvalue = [202, 160]\nclass = ON'),
 Text(0.5932203389830508, 0.5, 'battery_power <= 581.5\ngini = 0.5\nsamples = 1
18\nvalue = [96, 92]\nclass = ON'),
 Text(0.576271186440678, 0.35714285714285715, 'gini = 0.0\nsamples = 7\nvalue =
[11, 0]\nclass = ON'),
 Text(0.6101694915254238, 0.35714285714285715, 'ram <= 1958.5\ngini = 0.499\nsa
mples = 111\nvalue = [85, 92]\nclass = OFF'),
 Text(0.576271186440678, 0.21428571428571427, 'clock_speed <= 1.35\ngini = 0.48
4\nsamples = 60\nvalue = [41, 59]\nclass = OFF'),
 Text(0.559322033898305, 0.07142857142857142, 'gini = 0.483\nsamples = 30\nvalu
e = [26, 18]\nclass = ON'),
 Text(0.5932203389830508, 0.07142857142857142, 'gini = 0.392\nsamples = 30\nval
ue = [15, 41]\nclass = OFF'),
 Text(0.6440677966101694, 0.21428571428571427, 'px_height <= 534.0\ngini = 0.49
\nsamples = 51\nvalue = [44, 33]\nclass = ON'),
 Text(0.6271186440677966, 0.07142857142857142, 'gini = 0.4\nsamples = 30\nvalue
= [34, 13]\nclass = ON'),
 Text(0.6610169491525424, 0.07142857142857142, 'gini = 0.444\nsamples = 21\nval
ue = [10, 20]\nclass = OFF'),
 Text(0.8050847457627118, 0.5, 'price_range <= 2.5\ngini = 0.476\nsamples = 102
\nvalue = [106, 68]\nclass = ON'),
 Text(0.7457627118644068, 0.35714285714285715, 'talk_time <= 17.5\ngini = 0.493
\nsamples = 76\nvalue = [74, 58]\nclass = ON'),
 Text(0.711864406779661, 0.21428571428571427, 'ram <= 2066.5\ngini = 0.464\nsam
ples = 65\nvalue = [71, 41]\nclass = ON'),
 Text(0.6949152542372882, 0.07142857142857142, 'gini = 0.39\nsamples = 36\nvalu
e = [47, 17]\nclass = ON'),
 Text(0.7288135593220338, 0.07142857142857142, 'gini = 0.5\nsamples = 29\nvalue
= [24, 24]\nclass = ON'),
 Text(0.7796610169491526, 0.21428571428571427, 'm_dep <= 0.5\ngini = 0.255\nsam
ples = 11\nvalue = [3, 17]\nclass = OFF'),
 Text(0.7627118644067796, 0.07142857142857142, 'gini = 0.469\nsamples = 5\nvalu
e = [3, 5]\nclass = OFF'),
 Text(0.7966101694915254, 0.07142857142857142, 'gini = 0.0\nsamples = 6\nvalue
= [0, 12]\nclass = OFF'),
 Text(0.864406779661017, 0.35714285714285715, 'sc_w <= 9.5\ngini = 0.363\nsampl
es = 26\nvalue = [32, 10]\nclass = ON'),
 Text(0.847457627118644, 0.21428571428571427, 'ram <= 3660.0\ngini = 0.437\nsam
ples = 18\nvalue = [21, 10]\nclass = ON'),
 Text(0.8305084745762712, 0.07142857142857142, 'gini = 0.172\nsamples = 11\nval
ue = [19, 2]\nclass = ON'),
 Text(0.864406779661017, 0.07142857142857142, 'gini = 0.32\nsamples = 7\nvalue
= [2, 8]\nclass = OFF'),
 Text(0.8813559322033898, 0.21428571428571427, 'gini = 0.0\nsamples = 8\nvalue
= [11, 0]\nclass = ON'),
 Text(0.9322033898305084, 0.7857142857142857, 'mobile_wt <= 134.5\ngini = 0.444
\nsamples = 28\nvalue = [16, 32]\nclass = OFF'),
 Text(0.8983050847457628, 0.6428571428571429, 'mobile_wt <= 88.5\ngini = 0.493
\nsamples = 15\nvalue = [14, 11]\nclass = ON'),
 Text(0.8813559322033898, 0.5, 'gini = 0.463\nsamples = 5\nvalue = [4, 7]\nclas
s = OFF'),
 Text(0.9152542372881356, 0.5, 'mobile_wt <= 115.0\ngini = 0.408\nsamples = 10
\nvalue = [10, 4]\nclass = ON'),
```

```
 Text(0.8983050847457628, 0.35714285714285715, 'gini = 0.245\nsamples = 5\nvalu
e = [6, 1]\nclass = ON'),
 Text(0.9322033898305084, 0.35714285714285715, 'gini = 0.49\nsamples = 5\nvalue
= [4, 3]\nclass = ON'),
 Text(0.9661016949152542, 0.6428571428571429, 'talk_time <= 11.0\ngini = 0.159
\nsamples = 13\nvalue = [2, 21]\nclass = OFF'),
 Text(0.9491525423728814, 0.5, 'gini = 0.245\nsamples = 8\nvalue = [2, 12]\ncla
ss = OFF'),
 Text(0.9830508474576272, 0.5, 'gini = 0.0\nsamples = 5\nvalue = [0, 9]\nclass
= OFF')]
```



In [26]:
```
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[26]:

	Varname	Imp
12	ram	0.149274
10	px_height	0.093226
1	clock_speed	0.085026
0	battery_power	0.075030
11	px_width	0.073181
5	int_memory	0.072666
9	pc	0.068831
7	mobile_wt	0.057700
15	talk_time	0.056695
14	sc_w	0.053751
13	sc_h	0.052998
6	m_dep	0.037354
3	fc	0.031398
8	n_cores	0.022649
17	touch_screen	0.018273
18	wifi	0.017269
16	three_g	0.015181
2	dual_sim	0.010107
4	four_g	0.005861
19	price_range	0.003530

In [ ]: