

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

```
In [2]: train_df=pd.read_csv(r"C:\Users\91903\Downloads\Mobile_Price_Classification_train.csv")
train_df
```

clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time
2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	1
0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	
0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	
2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	1
1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	1
...
0.5	1	0	1	2	0.8	106	6	...	1222	1890	668	13	4	1
2.6	1	0	0	39	0.2	187	4	...	915	1965	2032	11	10	1
0.9	1	1	1	36	0.7	108	8	...	868	1632	3057	9	1	
0.9	0	4	1	46	0.1	145	5	...	336	670	869	18	10	1
...

```
In [3]: test_df=pd.read_csv(r"C:\Users\91903\Downloads\Mobile_Price_Classification_test.csv")
test_df
```

Out[3]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_heig	
	0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	22
	1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	72
	2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	127
	3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	29
	4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	72

	995	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	62
	996	997	609	0	1.8	1	0	0	13	0.9	186	...	2	115
	997	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	47
	998	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	3
	999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	45

1000 rows × 15 columns

In [4]: 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 [5]: 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 [6]: x=train_df.drop('dual_sim',axis=1)
        y=train_df['dual_sim']
```

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

```
In [9]: train_df['blue'].value_counts()
```

```
Out[9]: blue
        0    1010
        1     990
        Name: count, dtype: int64
```

```
In [10]: test_df['blue'].value_counts()
```

```
Out[10]: blue
         1     516
         0     484
        Name: count, dtype: int64
```

```
In [11]: T={"three_g":{"Yes":1,'No':0}}
train_df=train_df.replace(T)
print(train_df)
```

	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 [12]: T={"three_g":{"Yes":1,'No':0}}
test_df=test_df.replace(T)
print(test_df)
```

	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 [13]: x=train_df.drop('dual_sim',axis=1)
y=train_df['dual_sim']
```

```
In [14]: x=test_df.drop('dual_sim',axis=1)
y=test_df['dual_sim']
```

```
In [15]: 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=42)
x_train.shape,x_test.shape
```

```
Out[15]: ((700, 20), (300, 20))
```

```
Out[16]: ▾ RandomForestClassifier
RandomForestClassifier()
```

```
In [18]: 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]}
```

```
Out[19]: GridSearchCV  
estimator: RandomForestClassifier  
RandomForestClassifier
```

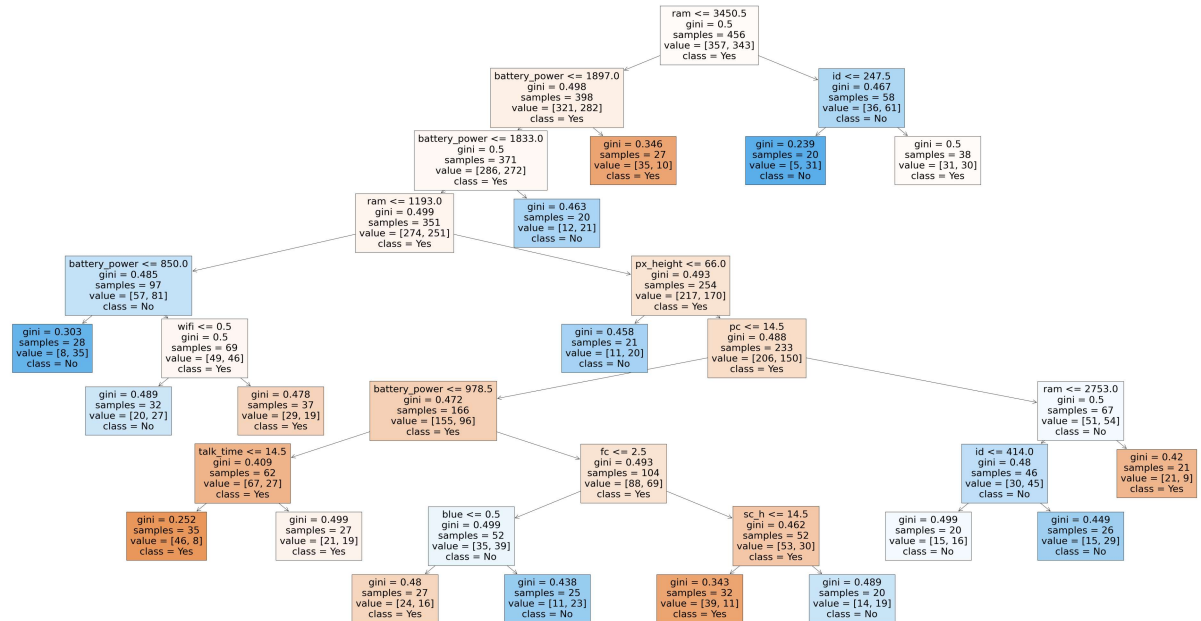
Out[20]: 0.54

```
RandomForestClassifier(max_depth=20, min_samples_leaf=20, n_estimators=25)
```

```

graph TD
    Root["n_cores <= 2.5  
gini = 0.497  
samples = 445  
value = [378, 322]  
class = Yes"]
    Root --> Left["id <= 695.5  
gini = 0.690  
samples = 124  
value = [92, 99]  
class = No"]
    Root --> Right["int_memory <= 34.5  
gini = 0.489  
samples = 321  
value = [286, 223]  
class = Yes"]
    
    Left --> Left_L["fc <= 6.5  
gini = 0.481  
samples = 91  
value = [74, 50]  
class = Yes"]
    Left --> Left_R["n_cores <= 1.5  
gini = 0.393  
samples = 43  
value = [18, 49]  
class = No"]
    
    Left_L --> Left_L_L["int_memory <= 32.5  
gini = 0.447  
samples = 61  
value = [60, 31]  
class = Yes"]
    Left_L --> Left_L_R["gini = 0.482  
samples = 20  
value = [11, 19]  
class = No"]
    
    Left_L_L --> Left_L_L_L["gini = 0.488  
samples = 31  
value = [26, 19]  
class = Yes"]
    Left_L_L --> Left_L_L_R["gini = 0.38  
samples = 30  
value = [35, 12]  
class = Yes"]
    
    Left_R --> Left_R_L["gini = 0.489  
samples = 23  
value = [14, 19]  
class = No"]
    Left_R --> Left_R_R["gini = 0.298  
samples = 20  
value = [4, 30]  
class = No"]
    
    Left_R_R --> Left_R_R_L["battery_power <= 1255.0  
gini = 0.489  
samples = 96  
value = [77, 71]  
class = Yes"]
    Left_R_R --> Left_R_R_R["n_dep <= 0.45  
gini = 0.483  
samples = 55  
value = [35, 51]  
class = No"]
    
    Left_R_R_L --> Left_R_R_L_L["gini = 0.487  
samples = 21  
value = [18, 13]  
class = Yes"]
    Left_R_R_L --> Left_R_R_L_R["gini = 0.437  
samples = 34  
value = [17, 18]  
class = No"]
    
    Left_R_R_R --> Left_R_R_R_L["clock_gated <= 1.45  
gini = 0.444  
samples = 41  
value = [42, 21]  
class = Yes"]
    Left_R_R_R --> Left_R_R_R_R["gini = 0.463  
samples = 20  
value = [21, 12]  
class = Yes"]
    
    Left_R_R_R_L --> Left_R_R_R_L_L["gini = 0.42  
samples = 21  
value = [21, 9]  
class = Yes"]
    Left_R_R_R_L --> Left_R_R_R_L_R["gini = 0.42  
samples = 21  
value = [21, 9]  
class = Yes"]
    
    Right --> Right_L["ram <= 2352.0  
gini = 0.499  
samples = 117  
value = [114, 126]  
class = No"]
    Right --> Right_R["pc <= 5.5  
gini = 0.461  
samples = 164  
value = [172, 97]  
class = Yes"]
    
    Right_L --> Right_L_L["id <= 228.5  
gini = 0.483  
samples = 61  
value = [37, 54]  
class = No"]
    Right_L --> Right_L_R["gini = 0.5  
samples = 39  
value = [29, 28]  
class = Yes"]
    
    Right_L_L --> Right_L_L_L["gini = 0.36  
samples = 12  
value = [8, 26]  
class = No"]
    Right_L_L --> Right_L_L_R["gini = 0.5  
samples = 39  
value = [29, 28]  
class = Yes"]
    
    Right_R --> Right_R_L["mobile_vt <= 131.5  
gini = 0.439  
samples = 51  
value = [66, 24]  
class = Yes"]
    Right_R --> Right_R_R["ram <= 2991.5  
gini = 0.483  
samples = 113  
value = [106, 17]  
class = Yes"]
    
    Right_R_L --> Right_R_L_L["gini = 0.482  
samples = 27  
value = [28, 19]  
class = Yes"]
    Right_R_L --> Right_R_L_R["gini = 0.206  
samples = 24  
value = [138, 3]  
class = Yes"]
    
    Right_R_R --> Right_R_R_L["n_cores <= 5.5  
gini = 0.468  
samples = 83  
value = [84, 50]  
class = Yes"]
    Right_R_R --> Right_R_R_R["gini = 0.5  
samples = 30  
value = [22, 23]  
class = No"]
    
    Right_R_R_L --> Right_R_R_L_L["battery_power <= 1344.5  
gini = 0.363  
samples = 42  
value = [146, 15]  
class = Yes"]
    Right_R_R_L --> Right_R_R_L_R["wifi <= 0.5  
gini = 0.5  
samples = 41  
value = [172, 17]  
class = Yes"]
    
    Right_R_R_L_L --> Right_R_R_L_L_L["gini = 0.245  
samples = 22  
value = [30, 5]  
class = Yes"]
    Right_R_R_L_L --> Right_R_R_L_L_R["gini = 0.492  
samples = 21  
value = [14, 18]  
class = No"]
    
    Right_R_R_L_L_L --> Right_R_R_L_L_L_L["gini = 0.459  
samples = 22  
value = [138, 10]  
class = Yes"]
    Right_R_R_L_L_L --> Right_R_R_L_L_L_R["gini = 0.492  
samples = 21  
value = [14, 18]  
class = No"]
    
    Right_R_R_L_L_R --> Right_R_R_L_R_L["gini = 0.492  
samples = 21  
value = [14, 18]  
class = No"]
    Right_R_R_L_R --> Right_R_R_L_R_R["gini = 0.492  
samples = 21  
value = [14, 18]  
class = No"]
  
```

```
In [23]: from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=['Yes', 'No'],filled=True)
```



```
In [25]: rf_best.feature_importances_
```

```
Out[25]: array([0.0370339 , 0.10107623, 0.01954541, 0.11817536, 0.03618838,
0.11001641, 0.01594087, 0.12155464, 0.01832324, 0.06159801,
0.01315267, 0.02888897, 0.04677771, 0.09954532, 0.05208036,
0.043608 , 0.03525566, 0.03877877, 0. , 0.0024601 ])
```

```
In [26]: imp_df=pd.DataFrame({'Varname':x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[26]:

	Varname	Imp
7	int_memory	0.121555
3	clock_speed	0.118175
5	fc	0.110016
1	battery_power	0.101076
13	px_width	0.099545
9	mobile_wt	0.061598
14	ram	0.052080
12	px_height	0.046778
15	sc_h	0.043608
17	talk_time	0.038779
0	id	0.037034
4	dual_sim	0.036188
16	sc_w	0.035256
11	pc	0.028889
2	blue	0.019545
8	m_dep	0.018323
6	four_g	0.015941
10	n_cores	0.013153
19	touch_screen	0.002460
18	three_g	0.000000

In []: