

```
In [2]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [3]: train_df=pd.read_csv(r"C:\Users\ubin\Downloads\Mobile_Price_Classification_train_data.csv")
train_df
```

```
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
...
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 [4]: test_df=pd.read_csv(r"C:\Users\jangidi veena\OneDrive\Documents\jupyter\Mobile  
test_df
```

```
Out[4]:
```

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

1000 rows × 21 columns



```
In [5]: 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: 328.3 KB
```

In [6]: 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 [7]: x=train_df.drop('wifi',axis=1)
y=train_df['wifi']

In [8]: x=test_df.drop('wifi',axis=1)
y=test_df['wifi']

In [9]: 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
x_train.shape,x_test.shape

Out[9]: ((700, 20), (300, 20))

In [10]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)

Out[10]: RandomForestClassifier()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [11]: rf=RandomForestClassifier()
```

```
In [12]: 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 [13]: 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[13]: GridSearchCV(cv=2, estimator=RandomForestClassifier(),
                    param_grid={'max_depth': [2, 3, 5, 10, 20],
                                'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                                'n_estimators': [10, 25, 30, 50, 100, 200]},
                    scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

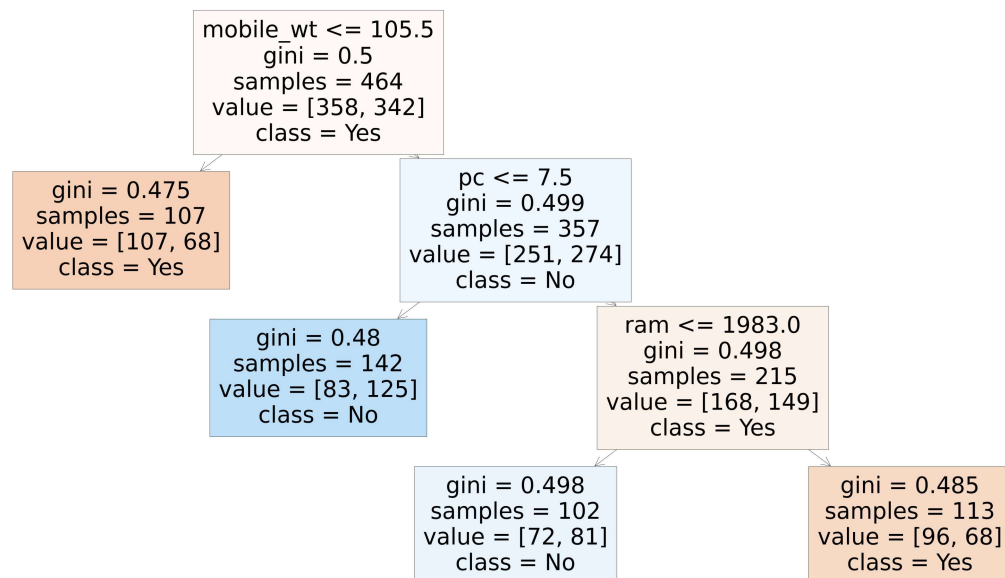
```
In [14]: grid_search.best_score_
```

```
Out[14]: 0.5614285714285714
```

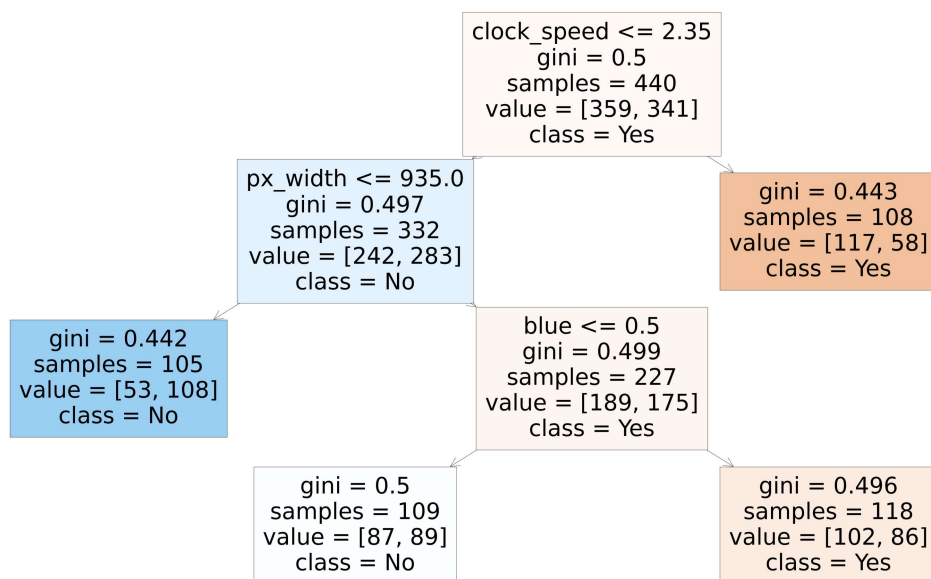
```
In [15]: rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=3, min_samples_leaf=100, n_estimators=200)
```

```
In [16]: from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'])
```



```
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=['Yes','No'])
```



```
In [18]: rf_best.feature_importances_
```

```
Out[18]: array([0.04582584, 0.07071127, 0.02287273, 0.11097868, 0.00939248,
0.06616402, 0.02471583, 0.07045867, 0.06064414, 0.10247415,
0.00616537, 0.04054562, 0.04776337, 0.13870076, 0.05979664,
0.02660986, 0.04916749, 0.03959261, 0.00742048])
```

```
In [19]: imp_df=pd.DataFrame({'Varname':x_train.columns,"Imp":rf_best.feature_importance_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[19]:

	Varname	Imp
13	px_width	0.138701
3	clock_speed	0.110979
9	mobile_wt	0.102474
1	battery_power	0.070711
7	int_memory	0.070459
5	fc	0.066164
8	m_dep	0.060644
14	ram	0.059797
16	sc_w	0.049167
12	px_height	0.047763
0	id	0.045826
11	pc	0.040546
17	talk_time	0.039593
15	sc_h	0.026610
6	four_g	0.024716
2	blue	0.022873
4	dual_sim	0.009392
19	touch_screen	0.007420
10	n_cores	0.006165
18	three_g	0.000000

In []: