

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

```
In [2]: tst=pd.read_csv(r"C:\Users\Sushma sree\Downloads\Mobile_Price_Classification_test.csv")
tst
```

Out[2]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773
...
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121
996	997	609	0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828

1000 rows × 21 columns

```
In [3]: tst.head()
```

Out[3]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15

5 rows × 21 columns

```
In [4]: tst.tail()
```

Out[4]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	:
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121	
996	997	609	0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933	
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223	
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509	
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828	

5 rows × 21 columns

```
In [5]: tst.describe
```

```
Out[5]: <bound method NDFrame.describe of
ry
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 [6]: tst.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]: tst.isnull().sum()
```

```
Out[7]: id                0
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
dtype: int64
```

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

```
In [9]: tst['dual_sim'].value_counts()
```

```
Out[9]: dual_sim
1      517
0      483
Name: count, dtype: int64
```

```
In [11]: m={"three_g":{"yes":1,"no":0}}
tst=tst.replace(m)
print(tst)
```

	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 [12]: x=tst.drop('wifi',axis=1)
y=tst['wifi']
```

```
In [13]: 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[13]: ((700, 20), (300, 20))

```
In [14]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[14]:

RandomForestClassifier

RandomForestClassifier()

```
In [15]: rf=RandomForestClassifier()
```

```
In [17]: 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 [18]: 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[18]:
```

GridSearchCV

▶ estimator: RandomForestClassifier

▶ RandomForestClassifier

```
In [19]: grid_search.best_score_
```

```
Out[19]: 0.5585714285714286
```

```
In [20]: rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=100, n_estimators=50)
```

```
In [21]: rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=100, n_estimators=50)
```

Train Data

```
In [22]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt,seaborn as sns
```

```
In [23]: tr=pd.read_csv(r"C:\Users\Sushma sree\Downloads\Mobile_Price_Classification_train.csv")
tr
```

```
Out[23]:
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411
...
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919

2000 rows × 15 columns



```
In [24]: tr.describe
```

```
Out[24]: <bound method NDFrame.describe of
0      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory
1      842           0      2.2          0  1      0           7
2      1021          1      0.5          1  0      1           53
3      563           1      0.5          1  2      1           41
4      615           1      2.5          0  0      0           10
...      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 [25]: tr.head()
```

```
Out[25]:
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	1
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	1
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	1
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	

5 rows × 21 columns



```
In [26]: tr.tail()
```

```
Out[26]:
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919

5 rows × 21 columns



```
In [27]: tr.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 [29]: x=tr.drop('wifi',axis=1)
         y=tr['wifi']
```

```
In [30]: tr['dual_sim'].value_counts()
```

```
Out[30]: dual_sim
1      1019
0       981
Name: count, dtype: int64
```

```
In [32]: m={"three_g":{"yes":1,"no":0}}
tr=tr.replace(m)
print(tr)
```

	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 [34]: x=tr.drop('wifi',axis=1)
y=tr['wifi']
```

```
In [35]: 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[35]: ((1400, 20), (600, 20))

```
In [36]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[36]:

RandomForestClassifier

RandomForestClassifier()

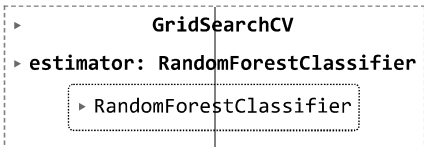
```
In [37]: rf=RandomForestClassifier()
```

```
In [38]: 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 [39]: 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[39]:
```



```
  ▸ GridSearchCV
  ▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

```
In [40]: grid_search.best_score_
```

```
Out[40]: 0.5235714285714286
```

```
In [41]: rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=200, n_estimators=25)
```

```
In [ ]:
```