

```
In [35]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt,seaborn as sns
```

```
In [36]: df=pd.read_csv(r"C:\Users\user\Downloads\Mobile_Price_Classification_train (1).csv")
df
```

Out[36]:

	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 × 21 columns

```
In [37]: test_df=pd.read_csv(r"C:\Users\user\Downloads\Mobile_Price_Classification_test.csv")
test_df
```

Out[37]:

	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 [38]: 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 [39]: x=df.drop('wifi',axis=1)
         y=['wifi']
```

```
In [40]: df['dual_sim'].value_counts()
```

```
Out[40]: dual_sim
1      1019
0       981
Name: count, dtype: int64
```

```
In [41]: H0={"four_g":{"Yes":1,"No":0}}
df=df.replace(H0)
print(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 [42]:

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 [43]:

test_df.describe()

Out[43]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	1248.510000	0.516000	1.540900	0.517000	4.593000	0.487000	33.652000	0.517500	139.500000
std	288.819436	432.458227	0.499994	0.829268	0.499961	4.463325	0.500081	18.128694	0.280861	34.800000
min	1.000000	500.000000	0.000000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000
25%	250.750000	895.000000	0.000000	0.700000	0.000000	1.000000	0.000000	18.000000	0.300000	109.750000
50%	500.500000	1246.500000	1.000000	1.500000	1.000000	3.000000	0.000000	34.500000	0.500000	139.000000
75%	750.250000	1629.250000	1.000000	2.300000	1.000000	7.000000	1.000000	49.000000	0.800000	170.000000
max	1000.000000	1999.000000	1.000000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000

8 rows × 11 columns

In [44]:

test_df['blue'].value_counts()

Out[44]:

```
blue
1     516
0     484
Name: count, dtype: int64
```

```
In [45]: test_df['fc'].value_counts()
```

```
Out[45]: fc
0      210
1      124
2       97
4       80
5       74
3       70
6       59
7       50
9       41
8       38
10      37
11      29
13      21
12      17
14      16
15      12
16      11
18      10
17       2
19       2
Name: count, dtype: int64
```

```
In [46]: x=df.drop('price_range',axis=1)
y=df['price_range']
```

```
In [47]: 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[47]: ((1400, 20), (600, 20))
```

```
In [48]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
Out[48]: ▾ RandomForestClassifier
RandomForestClassifier()
```

```
In [49]: rf=RandomForestClassifier()
```

```
In [50]: param={'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 [51]: from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=param,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

```
Out[51]: ▸ GridSearchCV
▸ estimator: RandomForestClassifier
▸ RandomForestClassifier
```

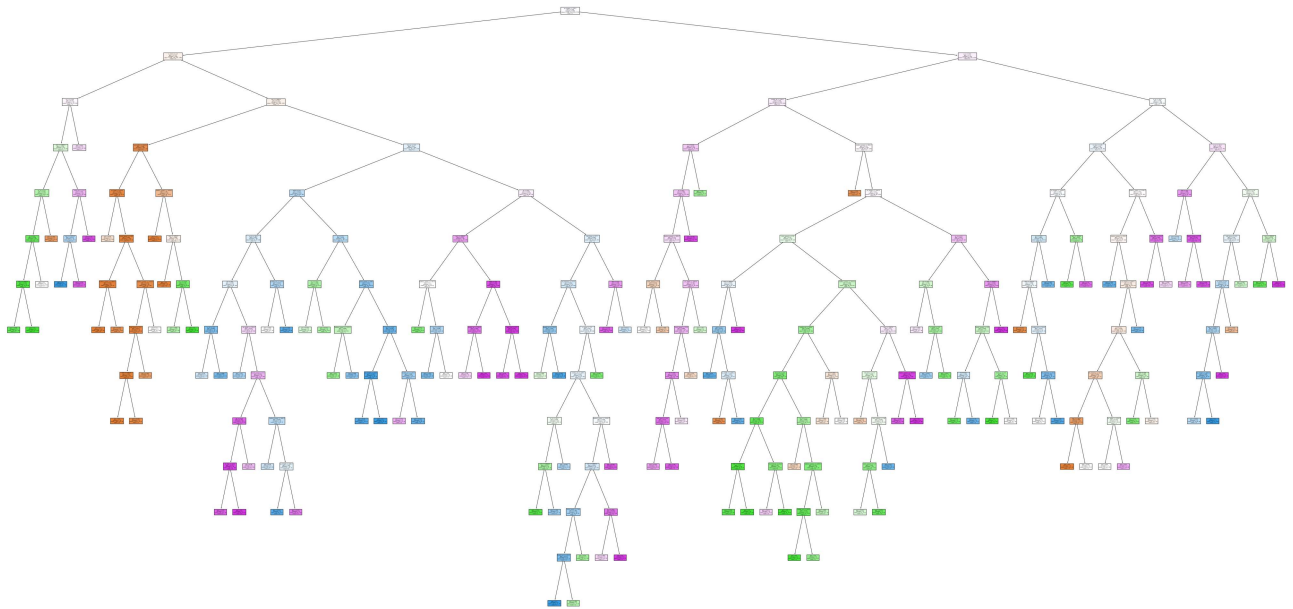
```
In [52]: grid_search.best_score_
```

```
Out[52]: 0.835
```

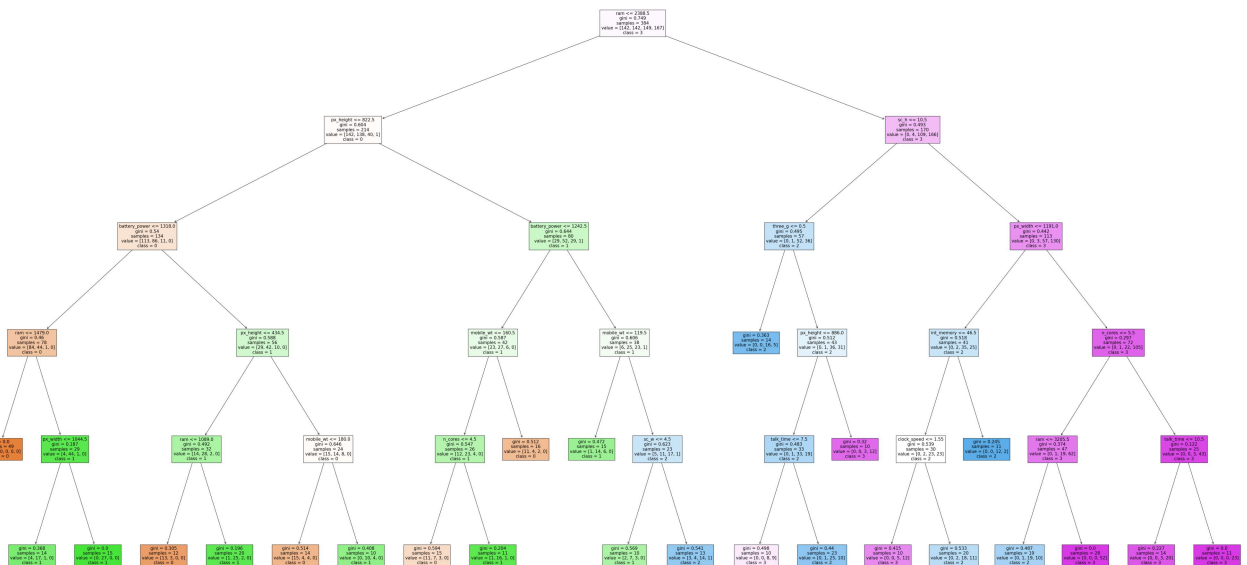
```
In [53]: rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=20, min_samples_leaf=5)
```

```
In [54]: from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['0','1','2','3'],filled=True);
```



```
In [73]: from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=['0','1','2','3'],filled=True);
```



```
In [56]: rf_best.feature_importances_
```

```
Out[56]: array([0.06726606, 0.00398477, 0.02021607, 0.00500412, 0.01554225,
0.00533026, 0.02995187, 0.01812967, 0.02912186, 0.01404124,
0.02321364, 0.05042497, 0.04602417, 0.59827838, 0.02027905,
0.01955554, 0.02034547, 0.00343017, 0.00451574, 0.00534467])
```

```
In [57]: imp_df=pd.DataFrame({"Varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[57]:

	Varname	Imp
13	ram	0.598278
0	battery_power	0.067266
11	px_height	0.050425
12	px_width	0.046024
6	int_memory	0.029952
8	mobile_wt	0.029122
10	pc	0.023214
16	talk_time	0.020345
14	sc_h	0.020279
2	clock_speed	0.020216
15	sc_w	0.019556
7	m_dep	0.018130
4	fc	0.015542
9	n_cores	0.014041
19	wifi	0.005345
5	four_g	0.005330
3	dual_sim	0.005004
18	touch_screen	0.004516
1	blue	0.003985
17	three_g	0.003430

```
In [58]: imp_df=pd.DataFrame({"Varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[58]:

	Varname	Imp
13	ram	0.598278
0	battery_power	0.067266
11	px_height	0.050425
12	px_width	0.046024
6	int_memory	0.029952
8	mobile_wt	0.029122
10	pc	0.023214
16	talk_time	0.020345
14	sc_h	0.020279
2	clock_speed	0.020216
15	sc_w	0.019556
7	m_dep	0.018130
4	fc	0.015542
9	n_cores	0.014041
19	wifi	0.005345
5	four_g	0.005330
3	dual_sim	0.005004
18	touch_screen	0.004516
1	blue	0.003985
17	three_g	0.003430

```
In [59]: X=test_df.drop('dual_sim',axis=1)
Y=test_df['dual_sim']
```

```
In [60]: 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[60]: ((1400, 20), (600, 20))
```

```
In [61]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_test,y_test)
```

```
Out[61]: ▾ RandomForestClassifier
RandomForestClassifier()
```

```
In [63]: rf=RandomForestClassifier()
```

```
In [64]: param={'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 [66]: from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=param,cv=2,scoring="accuracy")
grid_search.fit(x_test,y_test)
```

```
Out[66]: ▸ GridSearchCV
▸ estimator: RandomForestClassifier
▸ RandomForestClassifier
```

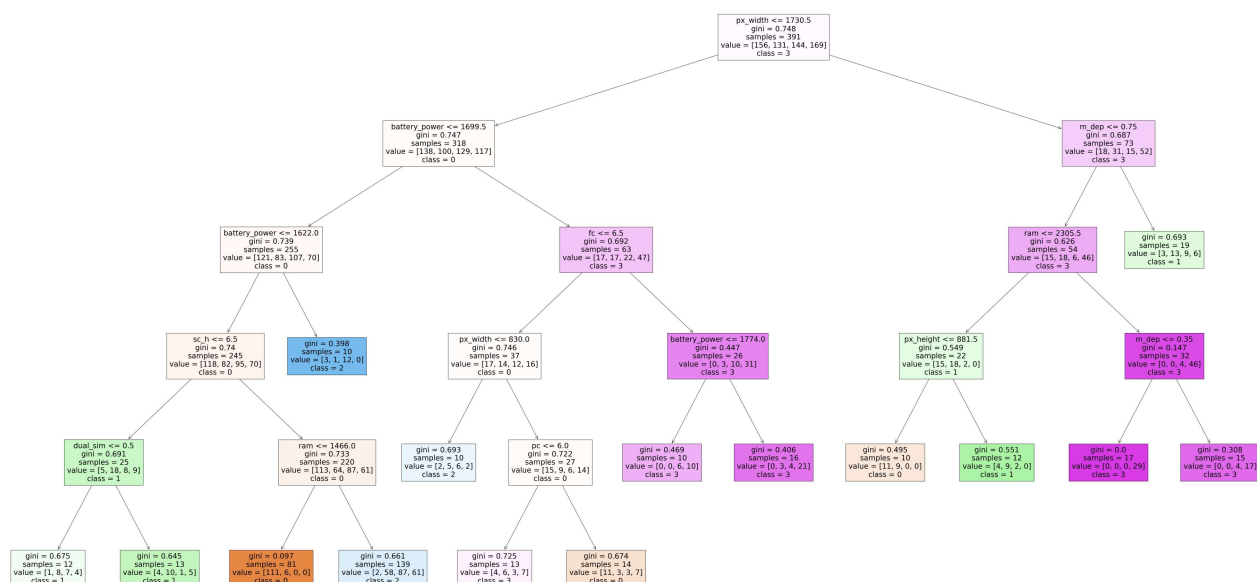
```
In [67]: grid_search.best_score_
```

```
Out[67]: 0.8166666666666667
```

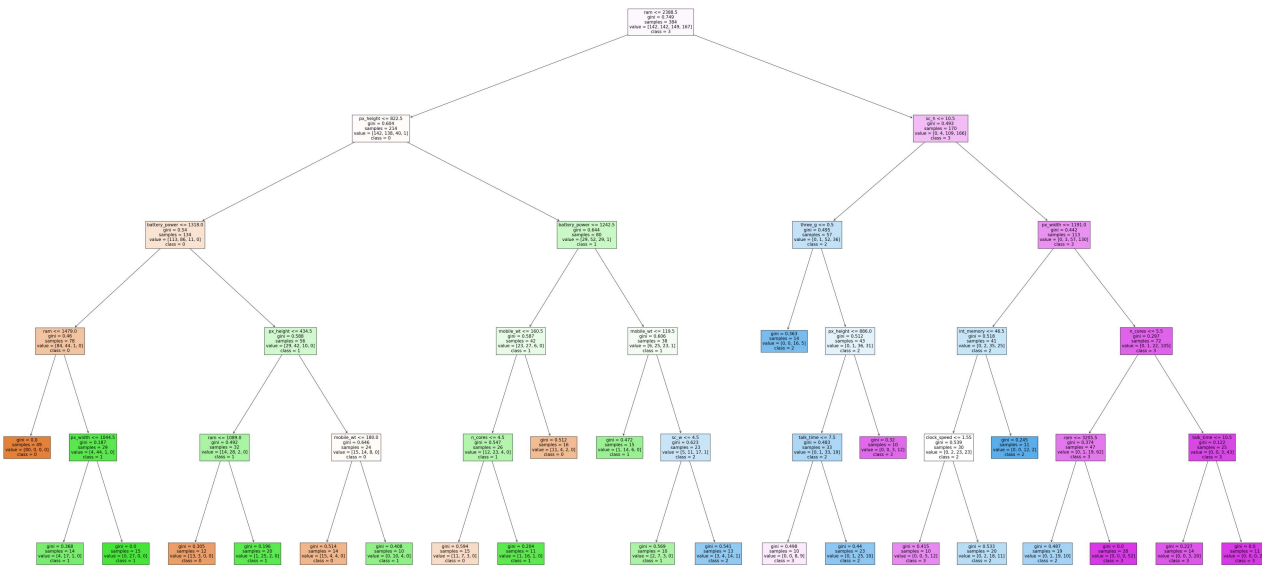
```
In [68]: rf_best=grid_search.best_estimator_
print(rf_best)
```

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

```
In [71]: from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['0','1','2','3'],filled=True);
```




```
In [70]: from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=['0','1','2','3'],filled=True);
```



```
In [74]: rf_best.feature_importances_
```

```
Out[74]: array([0.07271496, 0.00606811, 0.02160265, 0.00715252, 0.01269713,
0.00455432, 0.03302288, 0.01138988, 0.03179086, 0.02156046,
0.02872272, 0.05255705, 0.05551974, 0.56294726, 0.01892744,
0.01927508, 0.0222952 , 0.00118176, 0.00812567, 0.00789429])
```

```
In [75]: imp_df=pd.DataFrame({"Varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[75]:

	Varname	Imp
13	ram	0.562947
0	battery_power	0.072715
12	px_width	0.055520
11	px_height	0.052557
6	int_memory	0.033023
8	mobile_wt	0.031791
10	pc	0.028723
16	talk_time	0.022295
2	clock_speed	0.021603
9	n_cores	0.021560
15	sc_w	0.019275
14	sc_h	0.018927
4	fc	0.012697
7	m_dep	0.011390
18	touch_screen	0.008126
19	wifi	0.007894
3	dual_sim	0.007153
1	blue	0.006068
5	four_g	0.004554
17	three_g	0.001182

In []: