Project_2_Mobile_price_prediction

February 27, 2025

[1]:
```
#loading required library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
#load the dataset
df = pd.read_csv('dataset.csv')
df.head()
```

[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	\
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	

	mobile_wt	n_cores	…	px_height	px_width	ram	sc_h	sc_w	talk_time	\
0	188	2	…	20	756	2549	9	7	19	
1	136	3	…	905	1988	2631	17	3	7	
2	145	5	…	1263	1716	2603	11	2	9	
3	131	6	…	1216	1786	2769	16	8	11	
4	141	2	…	1208	1212	1411	8	2	15	

	three_g	touch_screen	wifi	price_range
0	0	0	1	1
1	1	1	0	2
2	1	1	0	2
3	1	0	0	2
4	1	1	0	1

[5 rows x 21 columns]

[3]:
```
#check size
df.shape
```

[3]: (2000, 21)

[4]:
```
#checking info
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
```

[5]:
```
#check statistical parameter
df.describe()
```

[5]:

	battery_power	blue	clock_speed	dual_sim	fc	\
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	
std	439.418206	0.5001	0.816004	0.500035	4.341444	
min	501.000000	0.0000	0.500000	0.000000	0.000000	
25%	851.750000	0.0000	0.700000	0.000000	1.000000	
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	
max	1998.000000	1.0000	3.000000	1.000000	19.000000	

	four_g	int_memory	m_dep	mobile_wt	n_cores	…	\

```
       battery_power         blue  clock_speed     dual_sim           fc  \
count    2000.000000  2000.000000  2000.000000  2000.000000  2000.000000   
mean      521.500000    32.046500     0.501750   140.249000     4.520500   
std       499.662000    18.145715     0.288416    35.399655     2.287837   
min         0.000000     2.000000     0.100000    80.000000     1.000000   
25%         0.000000    16.000000     0.200000   109.000000     3.000000   
50%         1.000000    32.000000     0.500000   141.000000     4.000000   
75%         1.000000    48.000000     0.800000   170.000000     7.000000   
max         1.000000    64.000000     1.000000   200.000000     8.000000   

            four_g  int_memory        m_dep    mobile_wt     n_cores  ...  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  ...   
mean      0.521500    32.046500     0.501750   140.249000     4.520500  ...   
std       0.499662    18.145715     0.288416    35.399655     2.287837  ...   
min       0.000000     2.000000     0.100000    80.000000     1.000000  ...   
25%       0.000000    16.000000     0.200000   109.000000     3.000000  ...   
50%       1.000000    32.000000     0.500000   141.000000     4.000000  ...   
75%       1.000000    48.000000     0.800000   170.000000     7.000000  ...   
max       1.000000    64.000000     1.000000   200.000000     8.000000  ...   

       px_height     px_width          ram         sc_h         sc_w  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000   
mean    645.108000  1251.515500  2124.213000    12.306500     5.767000   
std     443.780811   432.199447  1084.732044     4.213245     4.356398   
min       0.000000   500.000000   256.000000     5.000000     0.000000   
25%     282.750000   874.750000  1207.500000     9.000000     2.000000   
50%     564.000000  1247.000000  2146.500000    12.000000     5.000000   
75%     947.250000  1633.000000  3064.500000    16.000000     9.000000   
max    1960.000000  1998.000000  3998.000000    19.000000    18.000000   

         talk_time      three_g  touch_screen         wifi  price_range  
count  2000.000000  2000.000000   2000.000000  2000.000000  2000.000000  
mean     11.011000     0.761500      0.503000     0.507000     1.500000  
std       5.463955     0.426273      0.500116     0.500076     1.118314  
min       2.000000     0.000000      0.000000     0.000000     0.000000  
25%       6.000000     1.000000      0.000000     0.000000     0.750000  
50%      11.000000     1.000000      1.000000     1.000000     1.500000  
75%      16.000000     1.000000      1.000000     1.000000     2.250000  
max      20.000000     1.000000      1.000000     1.000000     3.000000  

[8 rows x 21 columns]
```

[6]:
```
#checking the missing value
df.isnull().sum(axis=0)
```

[6]:
```
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
price_range      0
dtype: int64
```

[7]:
```
#find correlation b/w all numerical columns
df.corr()
```

[7]:

	battery_power	blue	clock_speed	dual_sim	fc	\
battery_power	1.000000	0.011252	0.011482	-0.041847	0.033334	
blue	0.011252	1.000000	0.021419	0.035198	0.003593	
clock_speed	0.011482	0.021419	1.000000	-0.001315	-0.000434	
dual_sim	-0.041847	0.035198	-0.001315	1.000000	-0.029123	
fc	0.033334	0.003593	-0.000434	-0.029123	1.000000	
four_g	0.015665	0.013443	-0.043073	0.003187	-0.016560	
int_memory	-0.004004	0.041177	0.006545	-0.015679	-0.029133	
m_dep	0.034085	0.004049	-0.014364	-0.022142	-0.001791	
mobile_wt	0.001844	-0.008605	0.012350	-0.008979	0.023618	
n_cores	-0.029727	0.036161	-0.005724	-0.024658	-0.013356	
pc	0.031441	-0.009952	-0.005245	-0.017143	0.644595	
px_height	0.014901	-0.006872	-0.014523	-0.020875	-0.009990	
px_width	-0.008402	-0.041533	-0.009476	0.014291	-0.005176	
ram	-0.000653	0.026351	0.003443	0.041072	0.015099	
sc_h	-0.029959	-0.002952	-0.029078	-0.011949	-0.011014	
sc_w	-0.021421	0.000613	-0.007378	-0.016666	-0.012373	
talk_time	0.052510	0.013934	-0.011432	-0.039404	-0.006829	
three_g	0.011522	-0.030236	-0.046433	-0.014008	0.001793	
touch_screen	-0.010516	0.010061	0.019756	-0.017117	-0.014828	
wifi	-0.008343	-0.021863	-0.024471	0.022740	0.020085	
price_range	0.200723	0.020573	-0.006606	0.017444	0.021998	

	four_g	int_memory	m_dep	mobile_wt	n_cores	…	\
battery_power	0.015665	-0.004004	0.034085	0.001844	-0.029727	…	
blue	0.013443	0.041177	0.004049	-0.008605	0.036161	…	
clock_speed	-0.043073	0.006545	-0.014364	0.012350	-0.005724	…	
dual_sim	0.003187	-0.015679	-0.022142	-0.008979	-0.024658	…	
fc	-0.016560	-0.029133	-0.001791	0.023618	-0.013356	…	
four_g	1.000000	0.008690	-0.001823	-0.016537	-0.029706	…	
int_memory	0.008690	1.000000	0.006886	-0.034214	-0.028310	…	
m_dep	-0.001823	0.006886	1.000000	0.021756	-0.003504	…	
mobile_wt	-0.016537	-0.034214	0.021756	1.000000	-0.018989	…	
n_cores	-0.029706	-0.028310	-0.003504	-0.018989	1.000000	…	

```
pc            -0.005598   -0.033273  0.026282   0.018844 -0.001193  …
px_height     -0.019236    0.010441  0.025263   0.000939 -0.006872  …
px_width       0.007448   -0.008335  0.023566   0.000090  0.024480  …
ram            0.007313    0.032813 -0.009434  -0.002581  0.004868  …
sc_h           0.027166    0.037771 -0.025348  -0.033855 -0.000315  …
sc_w           0.037005    0.011731 -0.018388  -0.020761  0.025826  …
talk_time     -0.046628   -0.002790  0.017003   0.006209  0.013148  …
three_g        0.584246   -0.009366 -0.012065   0.001551 -0.014733  …
touch_screen   0.016758   -0.026999 -0.002638  -0.014368  0.023774  …
wifi          -0.017620    0.006993 -0.028353  -0.000409 -0.009964  …
price_range    0.014772    0.044435  0.000853  -0.030302  0.004399  …

               px_height  px_width       ram      sc_h      sc_w  talk_time  \
battery_power   0.014901 -0.008402 -0.000653 -0.029959 -0.021421   0.052510
blue           -0.006872 -0.041533  0.026351 -0.002952  0.000613   0.013934
clock_speed    -0.014523 -0.009476  0.003443 -0.029078 -0.007378  -0.011432
dual_sim       -0.020875  0.014291  0.041072 -0.011949 -0.016666  -0.039404
fc             -0.009990 -0.005176  0.015099 -0.011014 -0.012373  -0.006829
four_g         -0.019236  0.007448  0.007313  0.027166  0.037005  -0.046628
int_memory      0.010441 -0.008335  0.032813  0.037771  0.011731  -0.002790
m_dep           0.025263  0.023566 -0.009434 -0.025348 -0.018388   0.017003
mobile_wt       0.000939  0.000090 -0.002581 -0.033855 -0.020761   0.006209
n_cores        -0.006872  0.024480  0.004868 -0.000315  0.025826   0.013148
pc             -0.018465  0.004196  0.028984  0.004938 -0.023819   0.014657
px_height       1.000000  0.510664 -0.020352  0.059615  0.043038  -0.010645
px_width        0.510664  1.000000  0.004105  0.021599  0.034699   0.006720
ram            -0.020352  0.004105  1.000000  0.015996  0.035576   0.010820
sc_h            0.059615  0.021599  0.015996  1.000000  0.506144  -0.017335
sc_w            0.043038  0.034699  0.035576  0.506144  1.000000  -0.022821
talk_time      -0.010645  0.006720  0.010820 -0.017335 -0.022821   1.000000
three_g        -0.031174  0.000350  0.015795  0.012033  0.030941  -0.042688
touch_screen    0.021891 -0.001628 -0.030455 -0.020023  0.012720   0.017196
wifi            0.051824  0.030319  0.022669  0.025929  0.035423  -0.029504
price_range     0.148858  0.165818  0.917046  0.022986  0.038711   0.021859

                three_g  touch_screen      wifi  price_range
battery_power  0.011522     -0.010516 -0.008343     0.200723
blue          -0.030236      0.010061 -0.021863     0.020573
clock_speed   -0.046433      0.019756 -0.024471    -0.006606
dual_sim      -0.014008     -0.017117  0.022740     0.017444
fc             0.001793     -0.014828  0.020085     0.021998
four_g         0.584246      0.016758 -0.017620     0.014772
int_memory    -0.009366     -0.026999  0.006993     0.044435
m_dep         -0.012065     -0.002638 -0.028353     0.000853
mobile_wt      0.001551     -0.014368 -0.000409    -0.030302
n_cores       -0.014733      0.023774 -0.009964     0.004399
pc            -0.001322     -0.008742  0.005389     0.033599
```

```
px_height     -0.031174      0.021891  0.051824     0.148858
px_width       0.000350     -0.001628  0.030319     0.165818
ram            0.015795     -0.030455  0.022669     0.917046
sc_h           0.012033     -0.020023  0.025929     0.022986
sc_w           0.030941      0.012720  0.035423     0.038711
talk_time     -0.042688      0.017196 -0.029504     0.021859
three_g        1.000000      0.013917  0.004316     0.023611
touch_screen   0.013917      1.000000  0.011917    -0.030411
wifi           0.004316      0.011917  1.000000     0.018785
price_range    0.023611     -0.030411  0.018785     1.000000

[21 rows x 21 columns]
```

[8]:
```
#finding top 10 correcation input feature compared to target variables
df.corr().nlargest(10, ['price_range'])['price_range']
```

[8]:
```
price_range      1.000000
ram              0.917046
battery_power    0.200723
px_width         0.165818
px_height        0.148858
int_memory       0.044435
sc_w             0.038711
pc               0.033599
three_g          0.023611
sc_h             0.022986
Name: price_range, dtype: float64
```

[9]:
```
#find  index of correlation of 10 largest
df.corr().nlargest(10, ['price_range'])['price_range'].index
```

[9]:
```
Index(['price_range', 'ram', 'battery_power', 'px_width', 'px_height',
       'int_memory', 'sc_w', 'pc', 'three_g', 'sc_h'],
      dtype='object')
```

[10]:
```
df_10_feature = df[['price_range', 'ram', 'battery_power', 'px_width',␣
 ↪'px_height', 'int_memory', 'sc_w', 'pc', 'three_g', 'sc_h']]
df_10_feature
```

[10]:
```
      price_range   ram  battery_power  px_width  px_height  int_memory  sc_w  \
0               1  2549            842       756         20           7     7
1               2  2631           1021      1988        905          53     3
2               2  2603            563      1716       1263          41     2
3               2  2769            615      1786       1216          10     8
4               1  1411           1821      1212       1208          44     2
…             …    …                …         …          …           …     …
1995            0   668            794      1890       1222           2     4
```

```
1996            2  2032          1965       1965        915          39     10
1997            3  3057          1911       1632        868          36      1
1998            0   869          1512        670        336          46     10
1999            3  3919           510        754        483          45      4

      pc  three_g  sc_h
0      2        0     9
1      6        1    17
2      6        1    11
3      9        1    16
4     14        1     8
…    ..      …    …
1995  14        1    13
1996   3        1    11
1997   3        1     9
1998   5        1    18
1999  16        1    19

[2000 rows x 10 columns]
```

[11]:
```
#deciding the input feature and target variable
X = df_10_feature.drop(columns=['price_range'])
y= df_10_feature[['price_range']]
X.shape, y.shape
```

[11]:
```
((2000, 9), (2000, 1))
```

[12]:
```
#split input and target variable in to train and test with 80% and 20%
from sklearn.model_selection import train_test_split

x_trian, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
x_trian.shape, x_test.shape, y_train.shape, y_test.shape
```

[12]:
```
((1600, 9), (400, 9), (1600, 1), (400, 1))
```

[13]:
```
#loading model library
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100, random_state=42)
```

[14]:
```
#check random forest classifier model
rfc
```

[14]:
```
RandomForestClassifier(random_state=42)
```

[15]:
```
#fit x_train, y_train to model
rfc.fit(x_trian, y_train)
```

[15]:
```
RandomForestClassifier(random_state=42)
```

[16]:
```
#check prediction of X-test
prediction = rfc.predict(x_test)
```

[17]:
```
#load metrix
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
```

[18]:
```
#check confusion metrix
confusion_matrix(y_test, prediction)
```

[18]:
```
array([[100,   5,   0,   0],
       [  4,  84,   3,   0],
       [  0,   5,  79,   8],
       [  0,   0,  10, 102]])
```

[19]:
```
#check classification report
print(classification_report(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.96      0.95      0.96       105
           1       0.89      0.92      0.91        91
           2       0.86      0.86      0.86        92
           3       0.93      0.91      0.92       112

    accuracy                           0.91       400
   macro avg       0.91      0.91      0.91       400
weighted avg       0.91      0.91      0.91       400
```

based on classification report our model accuracy is 91% which is good.

[20]:
```
#check model feature values
rfc.feature_importances_
```

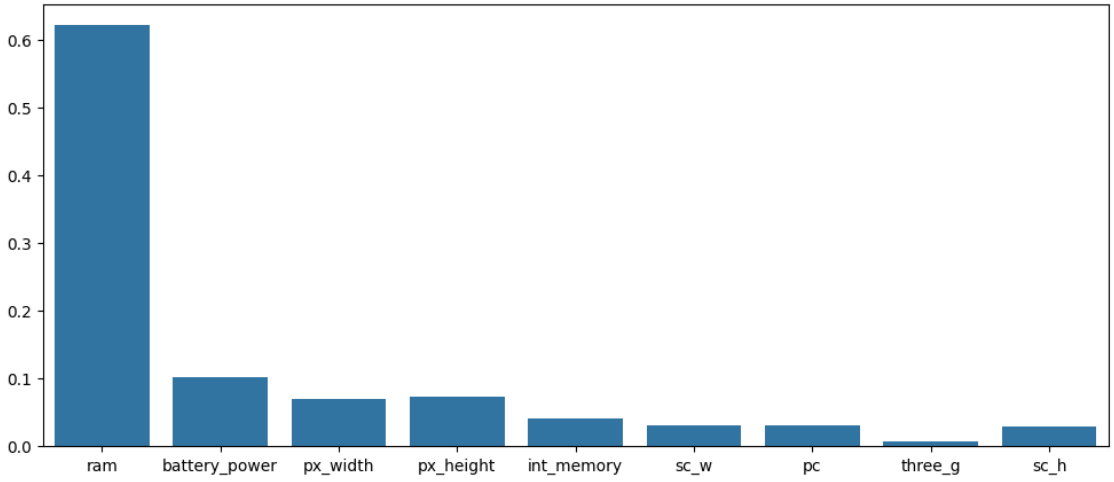
[20]:
```
array([0.62068397, 0.10099558, 0.06982819, 0.07334553, 0.04023982,
       0.03017789, 0.03032757, 0.00621063, 0.02819082])
```

[21]:
```
#check model columns name
rfc.feature_names_in_
```

[21]:
```
array(['ram', 'battery_power', 'px_width', 'px_height', 'int_memory',
       'sc_w', 'pc', 'three_g', 'sc_h'], dtype=object)
```

[22]:
```
#checking low feature importance
plt.figure(figsize=(12,5))
sns.barplot(x=rfc.feature_names_in_, y=rfc.feature_importances_)
```

[22]: <Axes: >



[23]:
```
# above bar graph is less important feature for our model. but we kept for␣
 ↪selecting the 3G phone.
#if we will remove this features then model accuracy will increaes more than 91%
```

[24]:
```
#save the model
import joblib
joblib.dump(rfc, 'rfc_model.pkl')
```

[24]: ['rfc_model.pkl']

0.0.1 Project Completed by Deepak Kumar.

0.0.2 Thanks You !