Mobile price classification using machine learning

Task 3 By Rajat Bairagi Data Science Intern at Coderscave

In [2]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:

```
df=pd.read_csv("C:\\Users\\astha\\Downloads\\train.csv")
##df
df.head()
```

Out[4]:

	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

5 rows × 21 columns

In [5]:

```
df.shape
```

Out[5]:

```
(2000, 21)
```

In [6]:

```
df.describe()
```

Out[6]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_m
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.(
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.(
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.1
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.(
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.(
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.(
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.(
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.(

8 rows × 21 columns

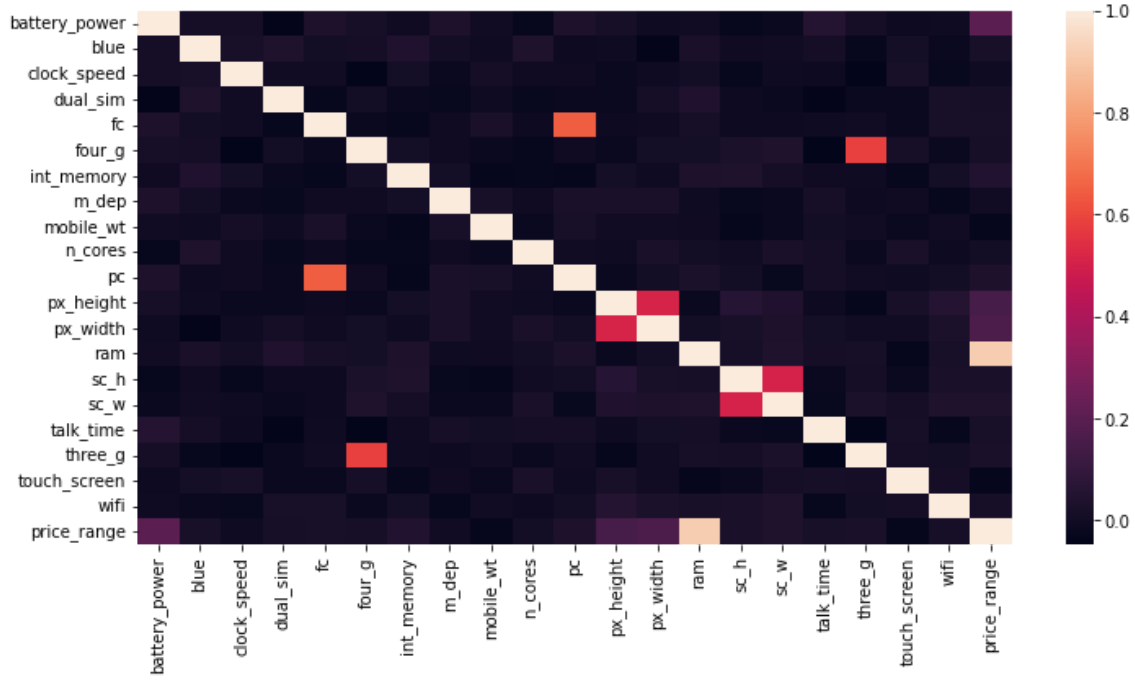
In [7]:

```
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 [8]:

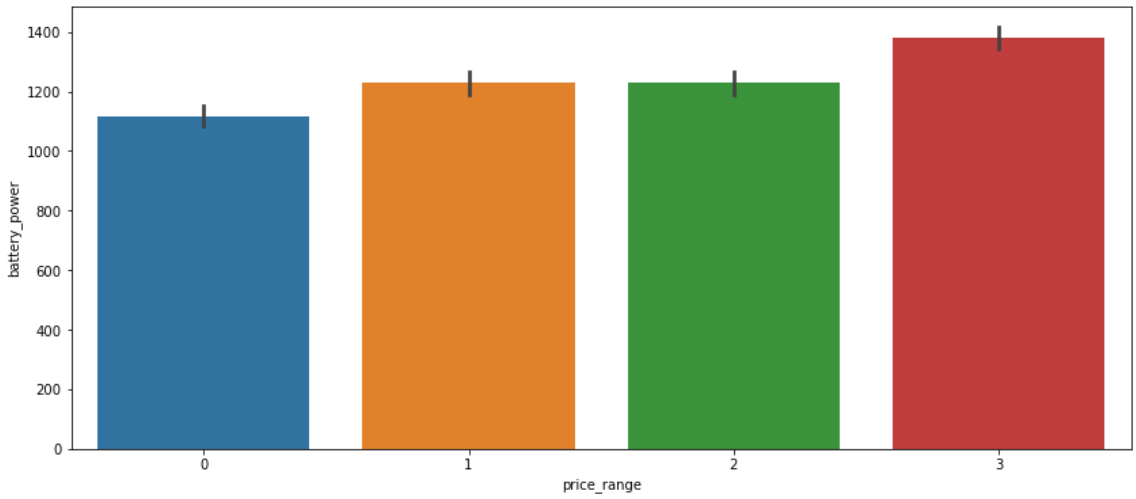
```
## HEAT MAP
plt.figure(figsize=(12,6))
sns.heatmap(df.corr())
plt.show()
```



plotting Relation between price_range and Battery power

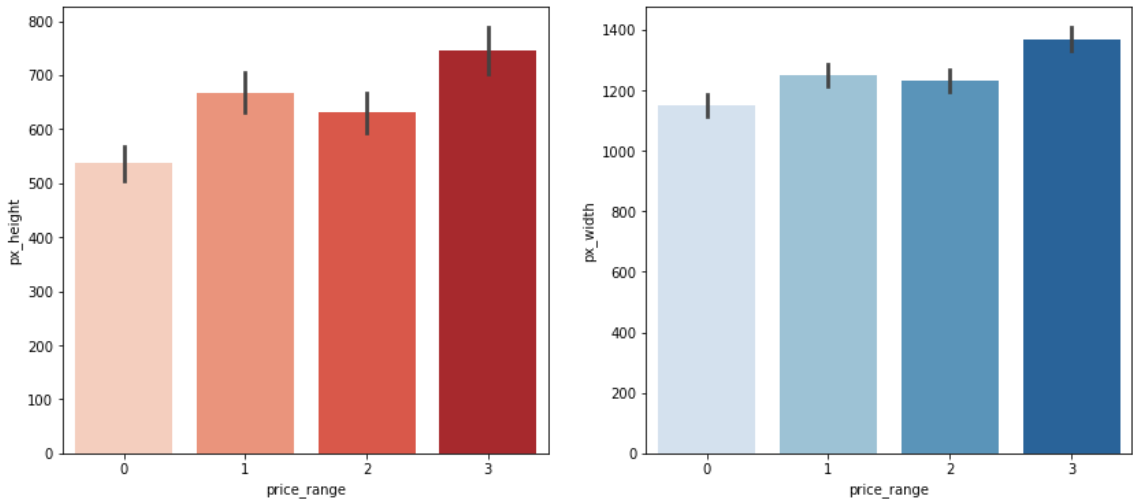
In [9]:

```
plt.figure(figsize=(14,6))
sns.barplot(x ='price_range',y ='battery_power',data=df)
plt.show()
```



In [10]:

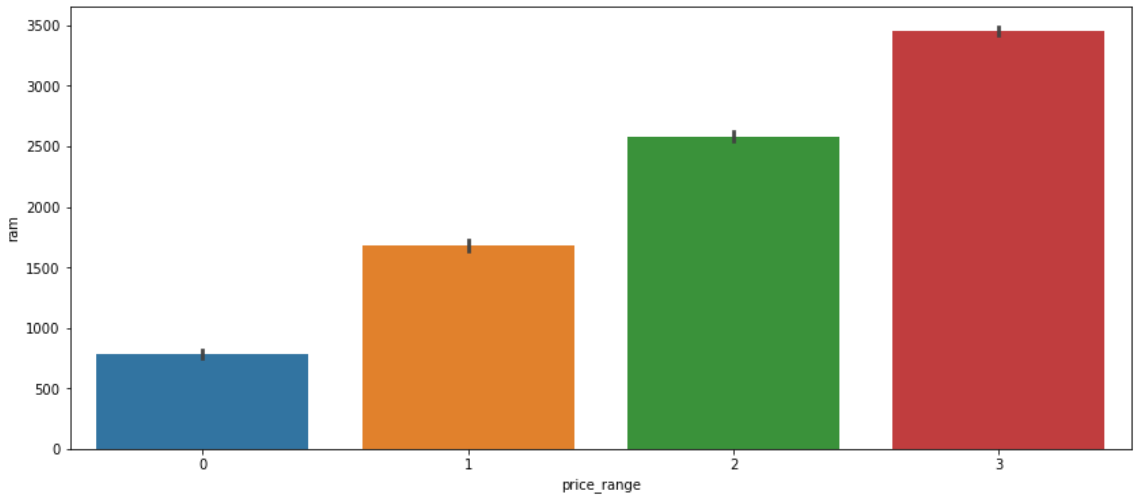
```
plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
sns.barplot(x ='price_range',y ='px_height',data=df,palette='Reds')
plt.subplot(1,2,2)
sns.barplot(x ='price_range',y ='px_width',data=df,palette='Blues')
plt.show()
```



relation between price_range and ram

In [11]:

```
plt.figure(figsize=(14,6))
sns.barplot(x ='price_range',y ='ram',data=df)
plt.show()
```



Data preprocessing

In [20]:

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

In [21]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

knn

In [22]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=10)
knn.fit(x_train,y_train)
```

Out[22]:

```
KNeighborsClassifier(n_neighbors=10)
```

In [23]:

```
knn.score(x_train,y_train)
```

Out[23]:

```
0.9457142857142857
```

In [24]:

```
predictions=knn.predict(x_test)
```

In [25]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,predictions)
```

Out[25]:

```
0.935
```

predcting values for test csv

In [31]:

```
test_df=pd.read_csv("C:\\Users\\astha\\Downloads\\test (2).csv")
test_df.head()
```

Out[31]:

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

5 rows × 21 columns

In [32]:

```
test_df.shape
```

Out[32]:

```
(1000, 21)
```

In [33]:

```
test_df=test_df.drop(['id'],axis=1)
test_df.shape
```

Out[33]:

```
(1000, 20)
```

In [34]:

```
test_pred=knn.predict(test_df)
```

In [35]:

```
test_df['predicted_price']=test_pred
```

In [36]:

```
test_df.head()
```

Out[36]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
0	1043	1	1.8	1	14	0	5	0.1	193
1	841	1	0.5	1	4	1	61	0.8	191
2	1807	1	2.8	0	1	0	27	0.9	186
3	1546	0	0.5	1	18	1	25	0.5	96
4	1434	0	1.4	0	11	1	49	0.5	108

5 rows × 21 columns

In [ ]:

```

```