Data Scraping and Analysis using Python

Competitive Pricing using Data Scraping.

Web Scraping is the process of collecting and parsing raw data from the Web, and the Python community has come up with some pretty powerful web scraping tools.

This technique is highly useful in **competitive pricing.** To check what our product's optimal price should be we can compare the similar products that are already in the market. These prices can vary a lot. So, in this blog, I'm going to show how we can scrap data regarding a particular product.

The Internet hosts perhaps the greatest source of information—and misinformation—on the planet. Many disciplines, such as data science, business intelligence, and investigative reporting, can benefit enormously from collecting and analysing data from websites.

Scrape and Parse Text From Websites

Collecting data from websites using an automated process is known as web scraping. Some websites explicitly forbid users from scraping their data with automated tools like the ones you'll create in this tutorial. Websites do this for two possible reasons:

1. The site has a good reason to protect its data. For instance, Google Maps doesn't let you request too many results too quickly.
2. Making many repeated requests to a website's server may use up bandwidth, slowing down the website for other users and potentially overloading the server such that the website stops responding entirely.

The most common technique for Data Scraping is using **BeautifulSoup** library in Python. It extracts the html for the page and stores it as an unstructured data. We'll have to convert that into structured format.

Let's import all the necessary libraries are:

```
import requests

from fake_useragent import UserAgent
```

```
from bs4 import BeautifulSoup
import pandas as pd
from urllib.parse import urljoin
import bs4
```

Data Extract is unstructured data and stored in an empty lists in a structured form.

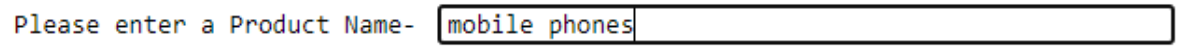
```
lstproductname=[]# List to store name of the product
lstprice =[] # List store pirice of the product
lstrating =[] # List to store ratings of the product
lstspecification = [] #List sto store specifications of the product
lstprocessor=[]
lstram=[]
lststorage=[]
lstos=[]
lstdisplay=[]
lstcamera=[]
lstbattery=[]
lstwarranty=[]
lstsimstype=[]
#lsthybridsim=[]
base_url="https://www.flipkart.com" #to read the sim type
```

Creating a user agent. Refer to this link https://pypi.org/project/fake-useragent/

```
user_agent = UserAgent()
```

Provide an input as a product name. The extracted data will be related to that product.

```
product_name = input("Please enter a Product Name- ")
```

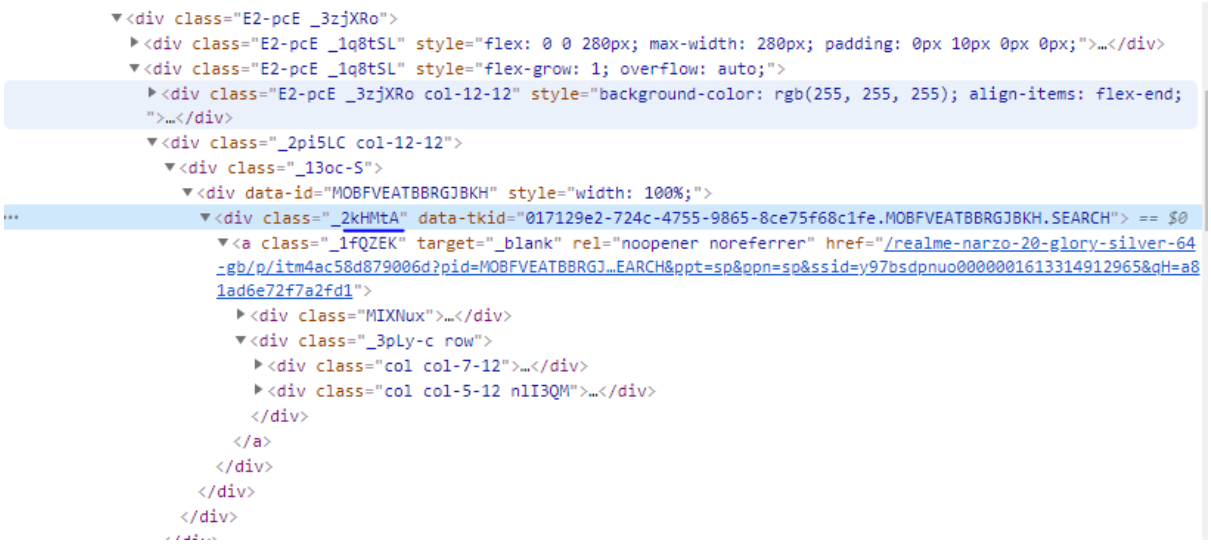


To extract data from multiple pages of the product listing we are going to use a for loop. The range will specify the number of pages to be extracted.

```
for i in range(1,12):
    url="https://www.flipkart.com/search?q={0}&page={1}"

    url=url.format(product_name,i)

    ##gettinh the response from the page using get method of requests module
    page = requests.get(url,headers={"user-agent":user_agent.chrome})
     ##storing the content of the page in a variable
    html = page.content

    ##creating beautifulsoup object
    page_soup =BeautifulSoup(html,"html.parser")

    for containers in page_soup.findAll('div',{'class':'_2kHMtA'}):

        productname=containers.find('div',attrs={'class':'_4rR01T'}) #_2rpwqI

        product_link=containers.find('a',attrs={'class':'_1fQZEK'})
        price=containers.find('div',attrs={'class':'_30jeq3'}) #_2rpwqI
        rating = containers.find('div',attrs={'class':'_3LWZlK'})

        intIndex=0
        for feature in containers.find('ul',attrs={'class':'_1xgFaf'}):

            if intIndex==0:
                storage =feature.text
            elif intIndex ==1:
                display = feature.text
            elif intIndex ==2:
                if feature.text:
                    camera = feature.text
                else:
                    camera ='NaN'
            elif intIndex ==3:
                battery = feature.text
            elif intIndex ==4:
                processor = feature.text
            elif intIndex ==5:
                warranty = feature.text

            intIndex +=1
        #Pass the child url to get the details of the product
        child_url=urljoin(base_url,product_link['href'])

        resp = requests.get(child_url)
        page_child = BeautifulSoup(resp.text, 'html.parser')

        for table in page_child.findAll('table',attrs={'class':'_14cfVK'}): #test:
            ths = table.find_all('td')#,attrs={'class':'_1hKmbr col col-3-12'})
            tds = [th.text for th in ths]
            for td in tds:
                if td == 'SIM Type':
                    sim_type=tds[tds.index(td)+1]
               # elif td == 'Hybrid Sim Slot':
                  #  hybridsim_slot = tds[tds.index(td)+1]

            else:
                break
            break
```

```
        #Loading the all the product sepecifications into the lists
        lstproductname.append(productname.text)
        lstprice.append(price.text)
        lststorage.append(storage)

        if type(rating) == bs4.element.Tag:
            lstrating.append(rating.text)
        else:
            lstrating.append('NaN')

        lstdisplay.append(display)
        lstbattery.append(battery)
        lstprocessor.append(processor)
        lstwarranty.append(warranty)
        lstcamera.append(camera)
        lstsimstype.append(sim_type)
        #Lsthybridsim.append(hybridsim_slot)
        #'Hybrid Sim Slot':lsthybridsim,
    products={'Proudct Name':lstproductname,'Processor':lstprocessor,'Storage':lststorage,
              'Display':lstdisplay,'Camera':lstcamera,'Battery':lstbattery,'Sim Type':lstsimstype,
              'Warranty':lstwarranty,'Price':lstprice,'Ratings':lstrating}
```

For extracting data from soup form you need to specify the html tags you want retrive the data it. You could use inspect element on the webpage..



The above code will store the data in a structured format. And when you print the dfProd you'll get:

```
dataset= pd.DataFrame(data=products)
```

```
dataset.head()
```

	Proudct Name	Processor	Storage	Display	Camera	Battery	Sim Type	Warranty	Price	Ratings
0	Realme Narzo 20 (Glory Silver, 64 GB)	MediaTek Helio G85 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	₹10,499	4.3
1	Realme Narzo 20 (Victory Blue, 128 GB)	MediaTek Helio G85 Processor	4 GB RAM | 128 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	₹11,499	4.3
2	Realme Narzo 20A (Victory Blue, 64 GB)	Qualcomm Snapdragon 665 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.51 cm (6.5 inch) HD+ Display	12MP + 2MP + 2MP | 8MP Front Camera	5000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	₹9,499	4.4
3	Realme Narzo 20 (Glory Silver, 128 GB)	MediaTek Helio G85 Processor	4 GB RAM | 128 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	₹11,499	4.3
4	Realme Narzo 20A (Glory Silver, 64 GB)	Qualcomm Snapdragon 665 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.51 cm (6.5 inch) HD+ Display	12MP + 2MP + 2MP | 8MP Front Camera	5000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	₹9,499	4.4

```
dfProd = dataset # changed from dataset to dfProd
```

Cleaning up Data

Remove the symbols from Price and clean the unnecessary special characters as well.

```
dfProd['Price'] = dfProd['Price'].str.lstrip('₹')
dfProd['Price'] = dfProd['Price'].replace({',':''}, regex=True)
dfProd.head()
```

	Product Name	Processor	Storage	Display	Camera	Battery	Sim Type	Warranty	Price	Ratings
0	Realme Narzo 20 (Glory Silver, 64 GB)	MediaTek Helio G85 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	10499	4.3
1	Realme Narzo 20A (Victory Blue, 64 GB)	Qualcomm Snapdragon 665 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.51 cm (6.5 inch) HD+ Display	12MP + 2MP + 2MP | 8MP Front Camera	5000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	9499	4.4
2	Realme Narzo 20 (Glory Silver, 128 GB)	MediaTek Helio G85 Processor	4 GB RAM | 128 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	11499	4.3
3	Realme Narzo 20 (Victory Blue, 128 GB)	MediaTek Helio G85 Processor	4 GB RAM | 128 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	11499	4.3
4	Realme Narzo 20A (Glory Silver, 64 GB)	Qualcomm Snapdragon 665 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.51 cm (6.5 inch) HD+ Display	12MP + 2MP + 2MP | 8MP Front Camera	5000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	9499	4.4

Split the product name by comma (,) assign the product name from first arrays of string and second arrays into color of the product.

```
#Split the Product name column and move the color into column and name to product name column
dfProd['Product Name']=dfProd['Product Name'].str.split(',', 1).str[0]
dfProd['Product Name'], dfProd['Color']=dfProd['Product Name'].str.split('(', 1).str
dfProd.head()
```

	Product Name	Processor	Storage	Display	Camera	Battery	Sim Type	Warranty	Price	Ratings	Color
0	Realme Narzo 20	MediaTek Helio G85 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	10499	4.3	Glory Silver
1	Realme Narzo 20A	Qualcomm Snapdragon 665 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.51 cm (6.5 inch) HD+ Display	12MP + 2MP + 2MP | 8MP Front Camera	5000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	9499	4.4	Victory Blue
2	Realme Narzo 20	MediaTek Helio G85 Processor	4 GB RAM | 128 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	11499	4.3	Glory Silver
3	Realme Narzo 20	MediaTek Helio G85 Processor	4 GB RAM | 128 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	11499	4.3	Victory Blue
4	Realme Narzo 20A	Qualcomm Snapdragon 665 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.51 cm (6.5 inch) HD+ Display	12MP + 2MP + 2MP | 8MP Front Camera	5000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	9499	4.4	Glory Silver

To check the data types for price and ratings

```
#Convert Price and ratings to float type
import numpy as np
dfProd['Price'] = dfProd['Price'].astype(np.float)
dfProd['Ratings']= dfProd['Ratings'].astype(np.float)
```

```
dfProd.dtypes
```

```
Product Name      object
Processor         object
Storage           object
Display           object
Camera            object
Battery           object
Sim Type          object
Warranty          object
Price            float64
Ratings          float64
Color             object
dtype: object
```

Create a product **company** name by splitting the product name column.

```
#Get a company name from product name its beginning of the first word.
dfProd['Company']=dfProd['Product Name'].str.split(' ', 1).str[0]
```

```
dfProd.head()
```

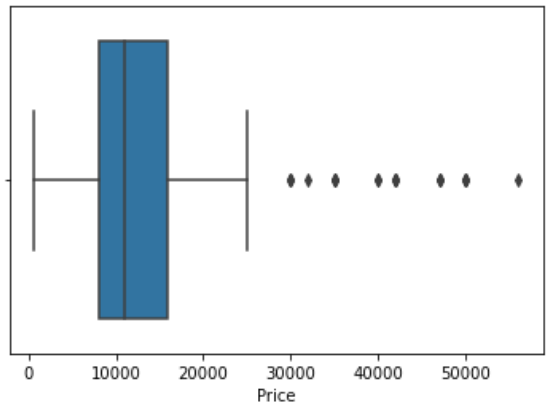
	Product Name	Processor	Storage	Display	Camera	Battery	Sim Type	Warranty	Price	Ratings	Color	Company
0	Realme Narzo 20	MediaTek Helio G85 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	10499.0	4.3	Glory Silver	Realme
1	Realme Narzo 20A	Qualcomm Snapdragon 665 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.51 cm (6.5 inch) HD+ Display	12MP + 2MP + 2MP | 8MP Front Camera	5000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	9499.0	4.4	Victory Blue	Realme
2	Realme Narzo 20	MediaTek Helio G85 Processor	4 GB RAM | 128 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	11499.0	4.3	Glory Silver	Realme
3	Realme Narzo 20	MediaTek Helio G85 Processor	4 GB RAM | 128 GB ROM | Expandable Upto 256 GB	16.56 cm (6.52 inch) HD+ Display	48MP + 8MP + 2MP | 8MP Front Camera	6000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	11499.0	4.3	Victory Blue	Realme
4	Realme Narzo 20A	Qualcomm Snapdragon 665 Processor	4 GB RAM | 64 GB ROM | Expandable Upto 256 GB	16.51 cm (6.5 inch) HD+ Display	12MP + 2MP + 2MP | 8MP Front Camera	5000 mAh Lithium-ion Battery	Dual Sim	Brand Warranty of 1 Year Available for Mobile ...	9499.0	4.4	Glory Silver	Realme

Fundamental Analysis of the mobile phone data.

Plotting Boxplot

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x=dfProd['Price']) # To  check if any outlier where the price range is very high
```

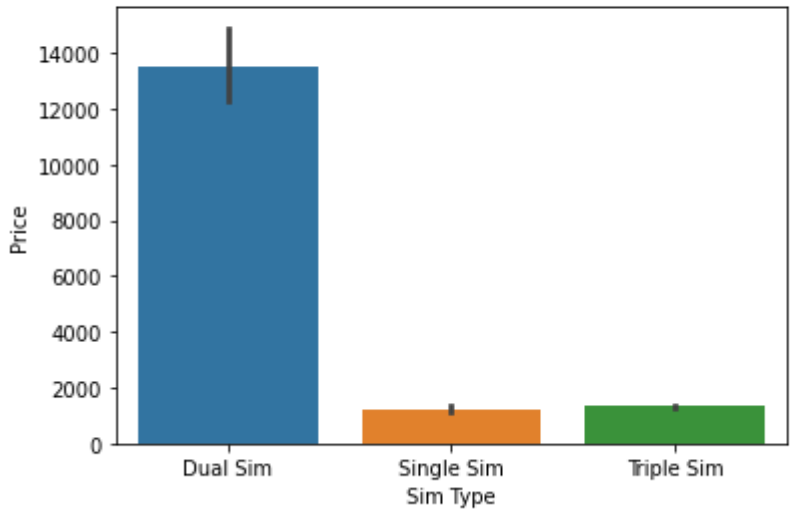
```
<matplotlib.axes._subplots.AxesSubplot at 0x17f9042d9d0>
```



Barplot for Sim Type vs Price

```
sns.barplot(x=dfProd['Sim Type'],y=dfProd['Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x17f90c28ee0>
```

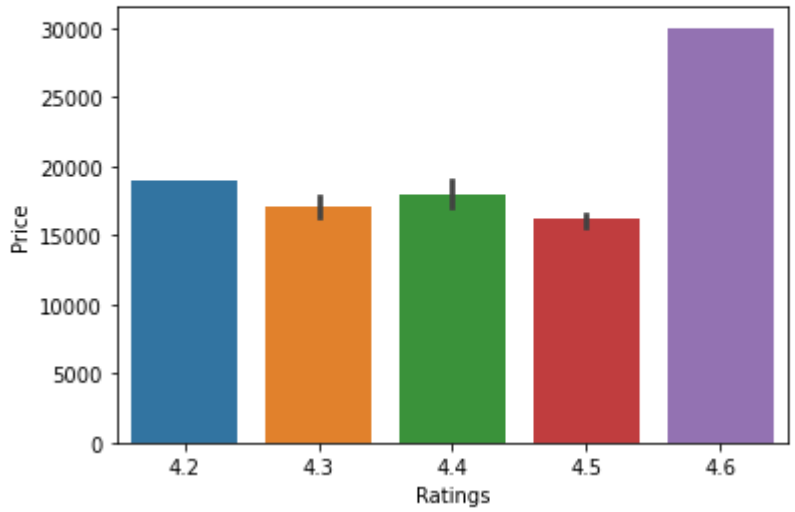


Will choose the budget range between Rs 14000 to 30000.

```
#Since our budget is 25k select price between 14000 to 30000 ruppes
dfProd1=dfProd[((dfProd['Price']>14000) & (dfProd['Price']<=30000))]
```

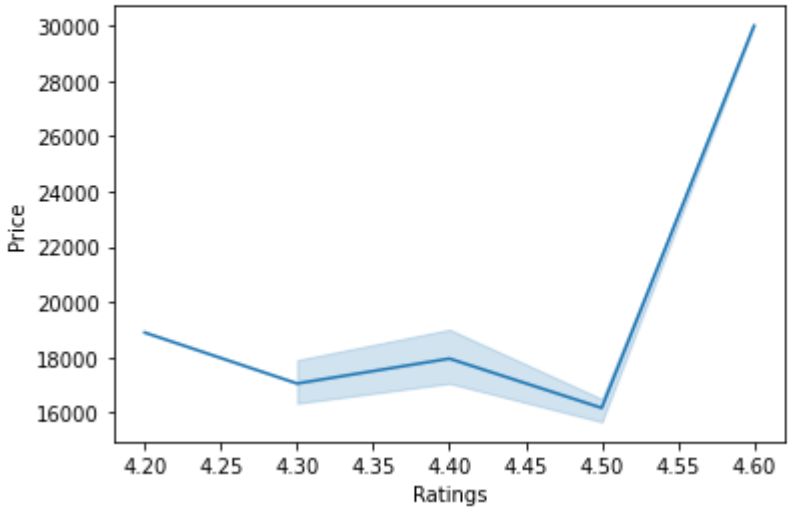
```
sns.barplot(y=dfProd1['Price'],x=dfProd1['Ratings'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x17f9047f460>
```



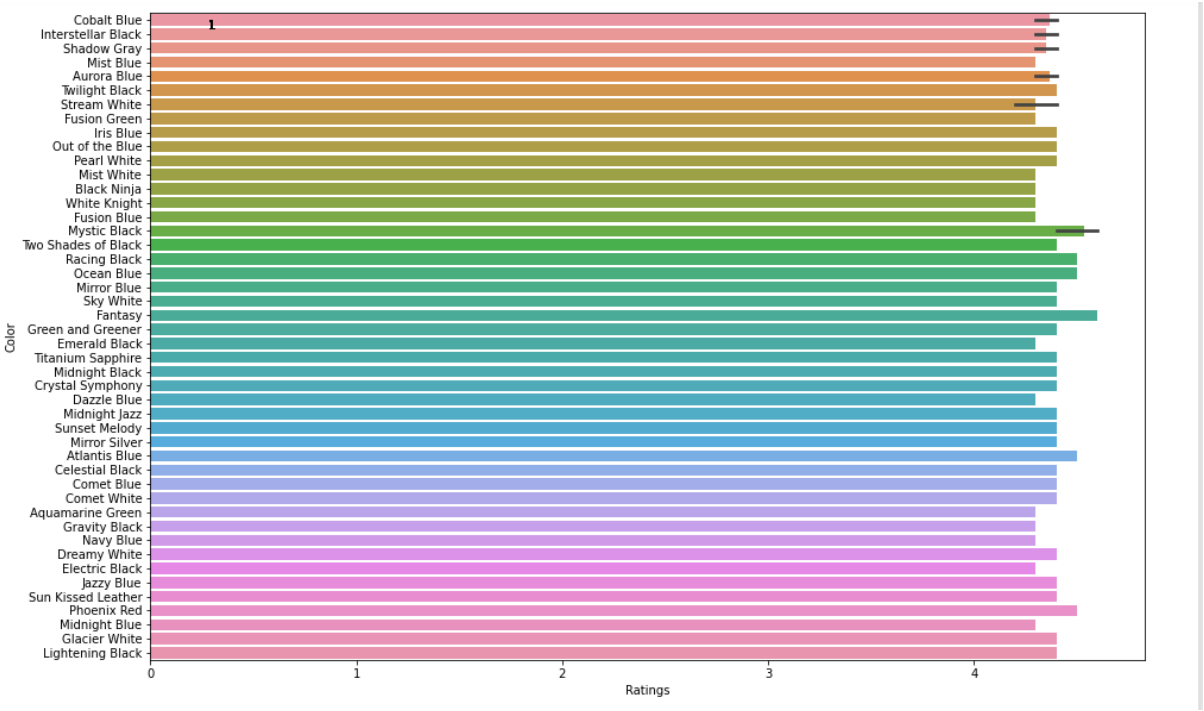

```
sns.lineplot(x=dfProd1['Ratings'],y=dfProd1['Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x17f90494e50>
```



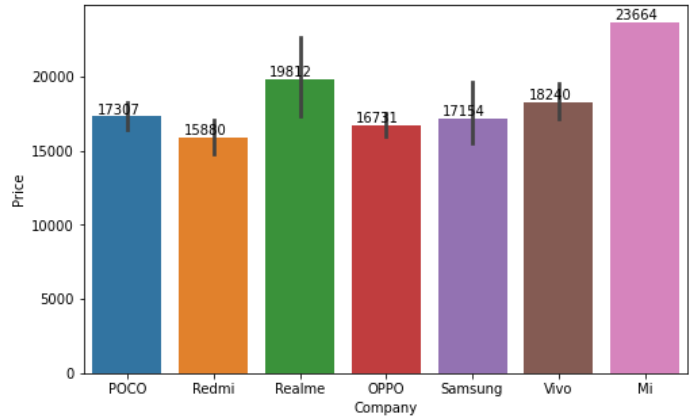
We can conclude from here that products with lower price have a higher ratings to some extent.

```
plt.figure(figsize=(15,10))
graph = sns.barplot(x=dfProd1['Ratings'], y=dfProd1['Color'], data = dfProd1)
for p in graph.patches:
        graph.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.3, p.get_height()),
                    ha='center', va='bottom',
                    color= 'black')

plt.show()
```



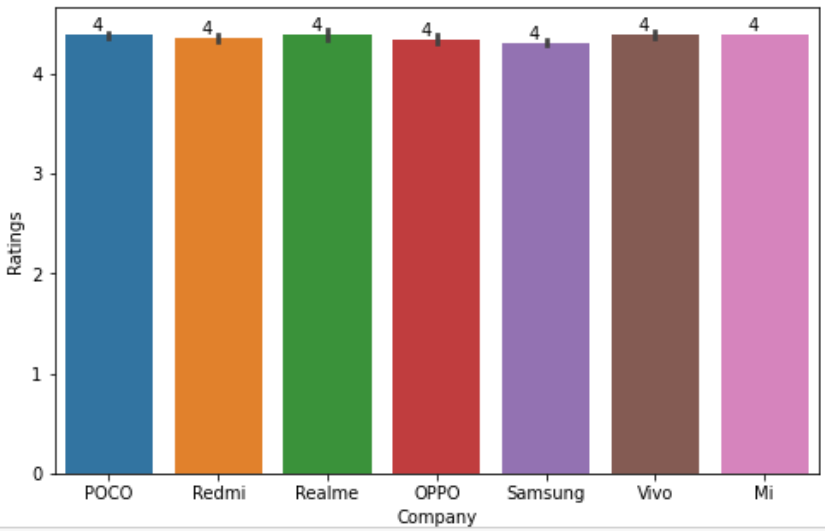
We can also observe that the colour has almost no effect on the ratings of the product.

```
plt.figure(figsize=(8,5))
graph = sns.barplot(x=dfProd1['Company'], y=dfProd1['Price'], data = dfProd1 )
for p in graph.patches:
        graph.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.3, p.get_height()),
                    ha='center', va='bottom',
                    color= 'black')
plt.show()
```



We can also observe that how the prices have effected on the company of the product

```
plt.figure(figsize=(8,5))
graph = sns.barplot(x=dfProd1['Company'], y=dfProd1['Ratings'], data = dfProd1 )
for p in graph.patches:
        graph.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.3, p.get_height()),
                    ha='center', va='bottom',
                    color= 'black')
plt.show()
```



Most of the company products are have ratings above 4

In conclusion the best option (with in the Rs 25000 budget) it would be more preferable to buy the Mi branded phone based on the ratings and the number of customers that have bought the product, means that the product will be more reliable.