CAPSTONE PROJECTS

DATA ANALYTICS

==

Capstone Project-1

Web Scraping Project:-

Outline:

Scrape any of the following website(s) and scrape details for any of the products like mobile phones, TV's, Laptops, personal health equipment or anything related to your domain.

Choices of websites:

1. Flipkart.com

2. Ajio.com

3. Snapdeal.com

You can choose the website and products from the above.

What you need to do?

For example, for a product called mobile phones, capture the data, clean the data and create

Visualizations which help in understanding the budget phones, and High end phones. Please

Make sure that data has to be visualized as per the features, camera pixels, no of sim slots, and

The data / plots should be displayed in such a way that the user should be able to understand

Which phone to buy if he has a set of features?

Case Study:-

For example: If I have 25000 INR as my budget, and if I need a single sim with some company

Like Samsung in mind, I should be able to select the phones from a group of phones.

How to submit?

1. A clear PPT/PDF explaining the problem statement along with the detailed approach.

2. .ipynb files to validate the code

3. HTML file or .ipynb file with all the visualizations

4. Data file Please note that, once the data is scraped and cleaned please save data as a data frame and send it across.

Evaluation The project will be evaluated based on requirements and instructions provided above.

Sol:-

https://www.Flipkart.com

Data Scraping and Analysis using Python Competitive Pricing using Data Scraping

Data Scraping is a technique to retrieve large amounts of data from the internet. 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. I'm going to show how we can scrap data regarding a particular product.

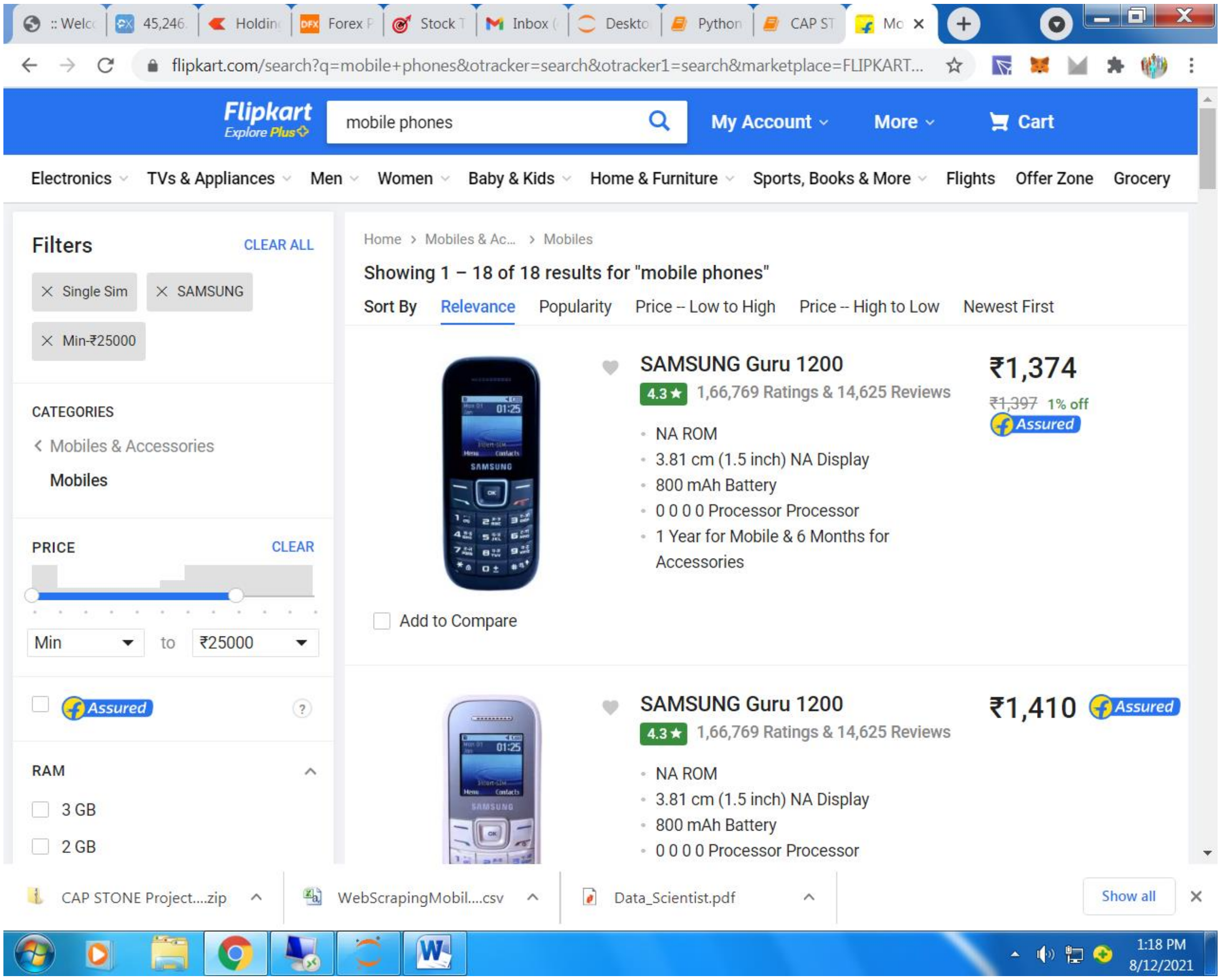
The most common technique for Data Scraping is using **Beautiful Soup**. It extracts the html for the page and stores it as an unstructured data. We'll have to convert that into structured format.

Web scraping-Samsung Mobile Phones According to Brand and price Range and Specifications and Rating.

Step : 1:- # parsing a simple html page

- import bs4
- from bs4 import BeautifulSoup as bs
- import requests

Step : 2:- Filtering the Link as according to Requirement Single sim, Price Range 0-25,000 Rs/- and Brand Name as Samsung.



flipkart.com/search?q=mobile+phones&otracker=search&otracker1=search&marketplace=FLIPKART...

mobile phones

Filters CLEAR ALL

× Single Sim × SAMSUNG × Min-₹25000

CATEGORIES

‹ Mobiles & Accessories

Mobiles

PRICE CLEAR

Min to ₹25000

□ Assured

RAM

□ 3 GB

□ 2 GB

Home › Mobiles & Ac... › Mobiles

Showing 1 – 18 of 18 results for "mobile phones"

Sort By Relevance Popularity Price -- Low to High Price -- High to Low Newest First

SAMSUNG Guru 1200

4.3★ 1,66,769 Ratings & 14,625 Reviews

- NA ROM
- 3.81 cm (1.5 inch) NA Display
- 800 mAh Battery
- 0 0 0 0 Processor Processor
- 1 Year for Mobile & 6 Months for Accessories

₹1,374 ~~₹1,397~~ 1% off Assured

□ Add to Compare

SAMSUNG Guru 1200

4.3★ 1,66,769 Ratings & 14,625 Reviews

- NA ROM
- 3.81 cm (1.5 inch) NA Display
- 800 mAh Battery
- 0 0 0 0 Processor Processor

₹1,410 Assured

CAP STONE Project....zip WebScrapingMobil....csv Data_Scientist.pdf Show all

1:18 PM 8/12/2021

link='https://www.flipkart.com/search?q=mobile+phones&otracker=search&otracker1=search&marketplace=FLIPKART&as-show=on&as=off&as-pos=1&astype=HISTORY&p%5B%5D=facets.sim_type%255B%255D%3DSingle%2BSim&p%5B%5D=facets.price_range.from%3DMin&p%5B%5D=facets.price_range.to%3D25000&p%5B%5D=facets.brand%255B%255D%3DSAMSUNG'

- page = requests.get(link)
- page = requests.get(link)
- page

<Response [200]>

Step : 3:-   Parsing the page content with Beautifulsoup Liberaries

- soup = bs(page.content,'html.parser')

Step : 4:-   Prettify the html code

- print(soup.prettify())

Step : 5:-   Finding all the configurations

# Initialization of the lists to store the extracted data

# The data that we extract is unstructured data. So we'll create empty lists to store them in a structured form,

count=0   # Intialize search row count

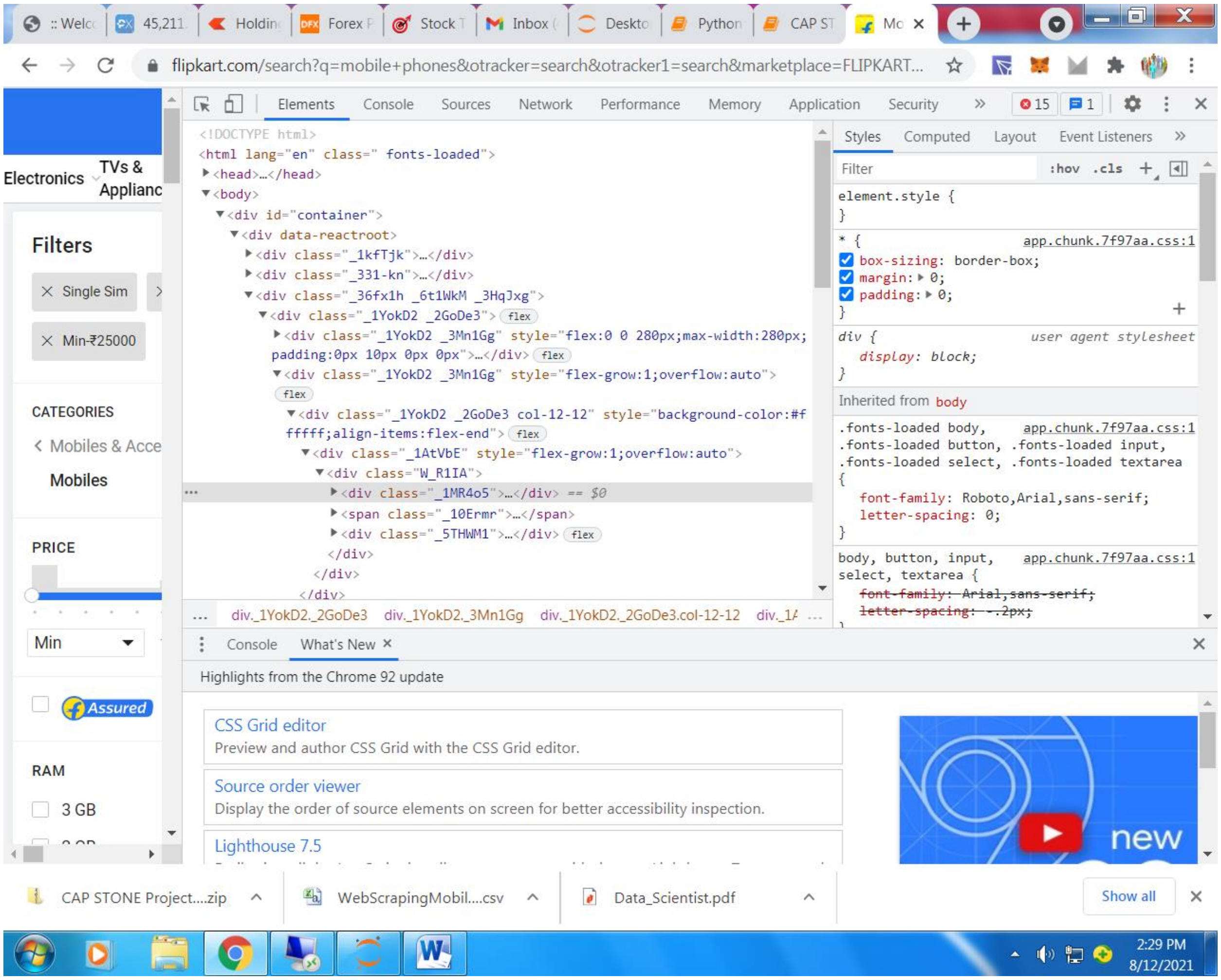
products=[]   #List to store name of the product

prices=[]   #List to store price of the product

```
ratings=[]            #List to store rating of the product

cpu = []              #List to store CPU specifications of the product

ram = []              #List to store RAM specifications of the product

os = []               #List to store OS specifications of the product

hd = []               #List to store HDD specifications of the product

display = []          #List to store Display specifications of the product
```

According to Specification we use options

Step : 6:-   Adding the Class Tags



```
<!DOCTYPE html>
<html lang="en" class=" fonts-loaded">
 ▶<head>…</head>
 ▼<body>
   ▼<div id="container">
     ▼<div data-reactroot>
       ▶<div class="_1kfTjk">…</div>
       ▶<div class="_331-kn">…</div>
       ▼<div class="_36fx1h _6t1WkM _3HqJxg">
         ▼<div class="_1YokD2 _2GoDe3"> flex
           ▶<div class="_1YokD2 _3Mn1Gg" style="flex:0 0 280px;max-width:280px;
           padding:0px 10px 0px 0px">…</div> flex
           ▼<div class="_1YokD2 _3Mn1Gg" style="flex-grow:1;overflow:auto">
           flex
             ▼<div class="_1YokD2 _2GoDe3 col-12-12" style="background-color:#f
             fffff;align-items:flex-end"> flex
               ▼<div class="_1AtVbE" style="flex-grow:1;overflow:auto">
                 ▼<div class="W_R1IA">
…                  ▶<div class="_1MR4o5">…</div> == $0
                   ▶<span class="_10Ermr">…</span>
                   ▶<div class="_5THWM1">…</div> flex
                 </div>
               </div>
             </div>
```

```
element.style {
}
* {                                  app.chunk.7f97aa.css:1
☑ box-sizing: border-box;
☑ margin: ▶ 0;
☑ padding: ▶ 0;
}
div {                               user agent stylesheet
   display: block;
}
Inherited from body
.fonts-loaded body,                  app.chunk.7f97aa.css:1
.fonts-loaded button, .fonts-loaded input,
.fonts-loaded select, .fonts-loaded textarea
{
   font-family: Roboto,Arial,sans-serif;
   letter-spacing: 0;
}
body, button, input,                 app.chunk.7f97aa.css:1
select, textarea {
   ~~font-family: Arial,sans-serif;~~
   ~~letter-spacing: -.2px;~~
```

Highlights from the Chrome 92 update

CSS Grid editor
Preview and author CSS Grid with the CSS Grid editor.

Source order viewer
Display the order of source elements on screen for better accessibility inspection.

Lighthouse 7.5

```
for containers in soup.findAll('a',class_='_1fQZEK'):

       name=containers.find('div', attrs={'class':'_4rR01T'})

        price=containers.find('div', attrs={'class':'_30jeq3 _1_WHN1'})

        rating=containers.find('div', attrs={'class':'_3LWZlK'})

        specification = containers.find('div', attrs={'class':'fMghEO'})
        ## Splitting integrated specification into individual Display specifications

        for col in specification:

             col=col.find_all('li', attrs={'class':'rgWa7D'})

             products.append(name.text) # Add product name to list

             prices.append(price.text) # Add price to list

    #specifications.append(specification.text) if type(specification) == bs4.element.Tag  else
specifications.append('NaN')

       hd.append(hdt) # Add HDD specifications to list

        display.append(displayt) # Add Display specifications to list

        ratings.append(rating.text) if type(rating) == bs4.element.Tag  else ratings.append('NaN') #

        Add Rating to list

         count=count+1 # Increment row count
```

Step : 7:- Print the length of data

- print(len(products))
- print(len(ratings))
- print(len(price))

```
51

51

1
```

Step : 8:- Importing pandas libraries

- import pandas as pd

Step : 9:- Creating a dataframe

- df=pd.DataFrame({'Product Name':products,'Display':display,'Price':prices,'Rating':ratings,})

Step : 10: Display the data frame

- df

```
Product Name Display    Price       Rating

0       SAMSUNG Guru 1200       1 Year for Mobile & 6 Months for Accessories       ₹1,410
        4.3

1       SAMSUNG Guru 1200       1 Year for Mobile & 6 Months for Accessories       ₹1,410
        4.3

2       SAMSUNG Guru 1200       1 Year for Mobile & 6 Months for Accessories       ₹1,100
        4.3

3       SAMSUNG Guru 1200       1 Year for Mobile & 6 Months for Accessories       ₹1,410
        4.3

4       SAMSUNG Guru 1200       1 Year for Mobile & 6 Months for Accessories       ₹1,410
        4.3

5       SAMSUNG Guru 1200       1 Year for Mobile & 6 Months for Accessories       ₹1,100
        4.3

6       SAMSUNG Guru 1200       1 Year for Mobile & 6 Months for Accessories       ₹1,410
        4.3
```

7	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,410	4.3
8	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,100	4.3
9	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,410	4.3
10	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,410	4.3
11	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,100	4.3
12	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,100	4.3
13	SAMSUNG C3520	1 Year for Mobile & 6 Months for Accessories	₹3,738	2.8
14	SAMSUNG E1190	1 Year for Mobile & 6 Months for Accessories	₹3,670	3.2
15	SAMSUNG S3600	1 Year for Mobile & 6 Months for Accessories	₹3,840	2.7
16	SAMSUNG E1270	1 Year for Mobile & 6 Months for Accessories	₹3,899	3.8

17 SAMSUNG Galaxy Alpha (Charcoal Black, 32 GB) 1 Year for Mobile & 6 Months for Accessories ₹22,389 3.6

18 SAMSUNG Galaxy S5 (Shimmery White, 16 GB) 1 Year for Mobile & 6 Months for Accessories ₹5,999 3.9

19 SAMSUNG Galaxy Note 3 (Jet Black, 32 GB) 1 Year for Mobile & 6 Months for Accessories ₹21,999 4.1

20 SAMSUNG Galaxy S5 (Electric Blue, 16 GB) 1 Year for Mobile & 6 Months for Accessories ₹21,999 3.9

21 SAMSUNG Galaxy M20 (Ocean Blue, 32 GB) 1 Year for Mobile & 6 Months for Accessories ₹9,990 4.2

22 SAMSUNG Galaxy S4 (Deep Black, 16 GB) 1 Year for Mobile & 6 Months for Accessories ₹15,999 3.9

23 SAMSUNG Galaxy Note 3 Neo (Black, 16 GB) 1 Year for Mobile & 6 Months for Accessories ₹19,900 4.2

24 SAMSUNG Galaxy S5 (Copper Gold, 16 GB) 1 Year for Mobile & 6 Months for Accessories ₹21,999 3.9

25 SAMSUNG Galaxy S5 (Charcoal Black, 16 GB) 1 Year for Mobile & 6 Months for Accessories ₹21,999 3.9

26	SAMSUNG GT 1200 R/I/M	1 Year for Mobile & 6 Months for Accessories	₹1,099	4.1
27	SAMSUNG Galaxy Alpha (Dazzling White, 32 GB)	1 Year for Mobile & 6 Months for Accessories	₹19,999	3.6
28	SAMSUNG Galaxy S4 Zoom (White, 8 GB)	1 Year for Mobile & 6 Months for Accessories	₹19,999	3.8
29	SAMSUNG Galaxy Note 3 Neo (White, 16 GB)	1 Year for Mobile & 6 Months for Accessories	₹11,250	4.2
30	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,410	4.3
31	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,410	4.3
32	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,100	4.3
33	SAMSUNG Guru 1200	1 Year for Mobile & 6 Months for Accessories	₹1,100	4.3
34	SAMSUNG C3520	1 Year for Mobile & 6 Months for Accessories	₹3,738	2.8
35	SAMSUNG E1190	1 Year for Mobile & 6 Months for Accessories	₹3,670	3.2

36	SAMSUNG S3600	1 Year for Mobile & 6 Months for Accessories	₹3,840	2.7
37	SAMSUNG E1270	1 Year for Mobile & 6 Months for Accessories	₹3,899	3.8
38	SAMSUNG Galaxy Alpha (Charcoal Black, 32 GB)	1 Year for Mobile & 6 Months for Accessories	₹22,389	3.6
39	SAMSUNG Galaxy S5 (Shimmery White, 16 GB)	1 Year for Mobile & 6 Months for Accessories	₹5,999	3.9
40	SAMSUNG Galaxy Note 3 (Jet Black, 32 GB)	1 Year for Mobile & 6 Months for Accessories	₹21,999	4.1
41	SAMSUNG Galaxy S5 (Electric Blue, 16 GB)	1 Year for Mobile & 6 Months for Accessories	₹21,999	3.9
42	SAMSUNG Galaxy M20 (Ocean Blue, 32 GB)	1 Year for Mobile & 6 Months for Accessories	₹9,990	4.2
43	SAMSUNG Galaxy S4 (Deep Black, 16 GB)	1 Year for Mobile & 6 Months for Accessories	₹15,999	3.9
44	SAMSUNG Galaxy Note 3 Neo (Black, 16 GB)	1 Year for Mobile & 6 Months for Accessories	₹19,900	4.2
45	SAMSUNG Galaxy S5 (Copper Gold, 16 GB)	1 Year for Mobile & 6 Months for Accessories	₹21,999	3.9

```
46	SAMSUNG Galaxy S5 (Charcoal Black, 16 GB)	1 Year for Mobile & 6 Months for
Accessories	₹21,999	3.9

47	SAMSUNG GT 1200 R/I/M	1 Year for Mobile & 6 Months for Accessories	₹1,099
	4.1

48	SAMSUNG Galaxy Alpha (Dazzling White, 32 GB)	1 Year for Mobile & 6 Months for
Accessories	₹19,999	3.6

49	SAMSUNG Galaxy S4 Zoom (White, 8 GB)	1 Year for Mobile & 6 Months for
Accessories	₹19,999	3.8

50	SAMSUNG Galaxy Note 3 Neo (White, 16 GB)	1 Year for Mobile & 6 Months for
Accessories	₹11,250	4.2
```

Step : 11:- Dataframe info

- df.info()

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 51 entries, 0 to 50

Data columns (total 4 columns):

 #   Column       Non-Null Count  Dtype
```

```
---  ------        -------------  -----
 0   Product Name  51 non-null     object
 1   Display       51 non-null     object
 2   Price         51 non-null     object
 3   Rating        51 non-null     object
dtypes: object(4)
memory usage: 880.0+ bytes
```

- print(df.shape)

```
(51, 4)
```

Step : 12:-   checking the Null values

- df.isnull()

```
Product Name    Display    Price  Rating
0       False False False False
```

1 False False False False

2 False False False False

3 False False False False

4 False False False False

5 False False False False

6 False False False False

7 False False False False

8 False False False False

9 False False False False

10 False False False False

11 False False False False

12 False False False False

13 False False False False

14 False False False False

15	False	False	False	False
16	False	False	False	False
17	False	False	False	False
18	False	False	False	False
19	False	False	False	False
20	False	False	False	False
21	False	False	False	False
22	False	False	False	False
23	False	False	False	False
24	False	False	False	False
25	False	False	False	False
26	False	False	False	False
27	False	False	False	False
28	False	False	False	False

29	False False False False
30	False False False False
31	False False False False
32	False False False False
33	False False False False
34	False False False False
35	False False False False
36	False False False False
37	False False False False
38	False False False False
39	False False False False
40	False False False False
41	False False False False
42	False False False False

```
43     False False False False

44     False False False False

45     False False False False

46     False False False False

47     False False False False

48     False False False False

49     False False False False

50     False False False False
```

Step : 13:-   Check for null values sum

- df.isnull().sum() # Check for null values

```
Product Name    0

Display         0

Price           0

Rating          0
```

```
dtype: int64
```

Step : 14:- cleaning the data:-   formatting the price values

# Format Price column to remove ₹ and delimiter ',' used for the thousandth place

- df['Price'] = df['Price'].str.lstrip('₹')
- df['Price'] = df['Price'].replace({',' : ''}, regex=True)

Step : 15:-   check if formatting is correct

- df.head() # Check if formatting is correct

```
Product Name     Display       Price  Rating

0       SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
        1410  4.3

1       SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
        1410  4.3

2       SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
        1100  4.3

3       SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
        1410  4.3
```

```
4       SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
        1410  4.3
```

Step : 16:-   Find the data types

- df.dtypes

```
Product Name    object
Display         object
Price           object
Rating          object
dtype: object
```

```
# works as as below code
#df['Price']=df['Price'].astype(float)
#df['Rating']=df['Rating'].astype(float)
df.dtypes
```

```
Product Name    object
```

```
Display       object

Price         object

Rating        object

dtype: object
```

Step : 17:-   # Save cleaned and processed data to a CSV file

- df.to_csv('WebScrapingMobiles.csv', index=False)
- df1=pd.read_csv('WebScrapingMobiles.csv')

Step : 18:-   check the data on df1

- df1.head()

```
Product Name     Display       Price  Rating

0        SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
         1410  4.3

1        SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
         1410  4.3

2        SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
         1100  4.3
```

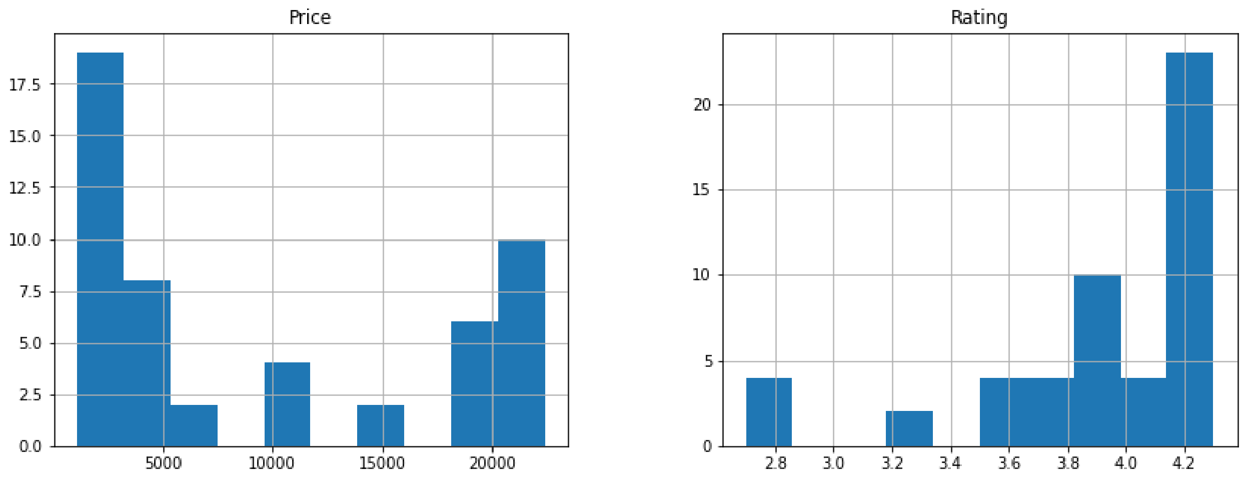
```
3       SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
        1410   4.3

4       SAMSUNG Guru 1200  1 Year for Mobile & 6 Months for Accessories
        1410   4.3
```

Step : 19: - Exploratory Data Anaysis(EDA)

- Univariate Analysis Plot Histograms and BoxPlots

(i)# Plot Histograms of Price and Rating

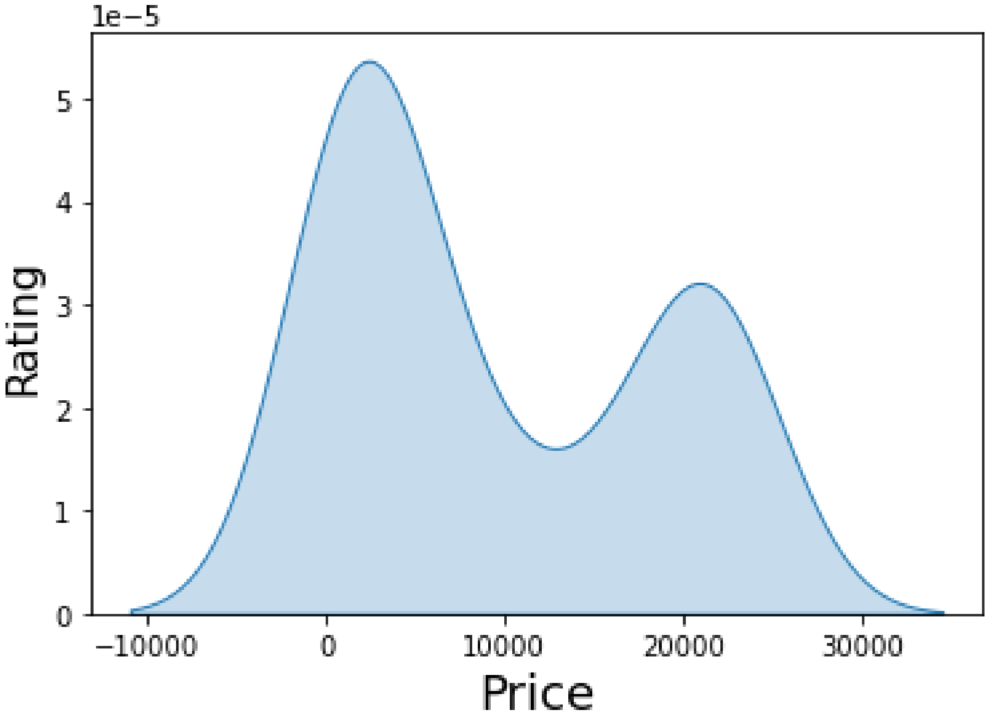
```
import seaborn as sns
import matplotlib.pyplot as plt
df1.hist(figsize=(14,5))
plt.show()
```

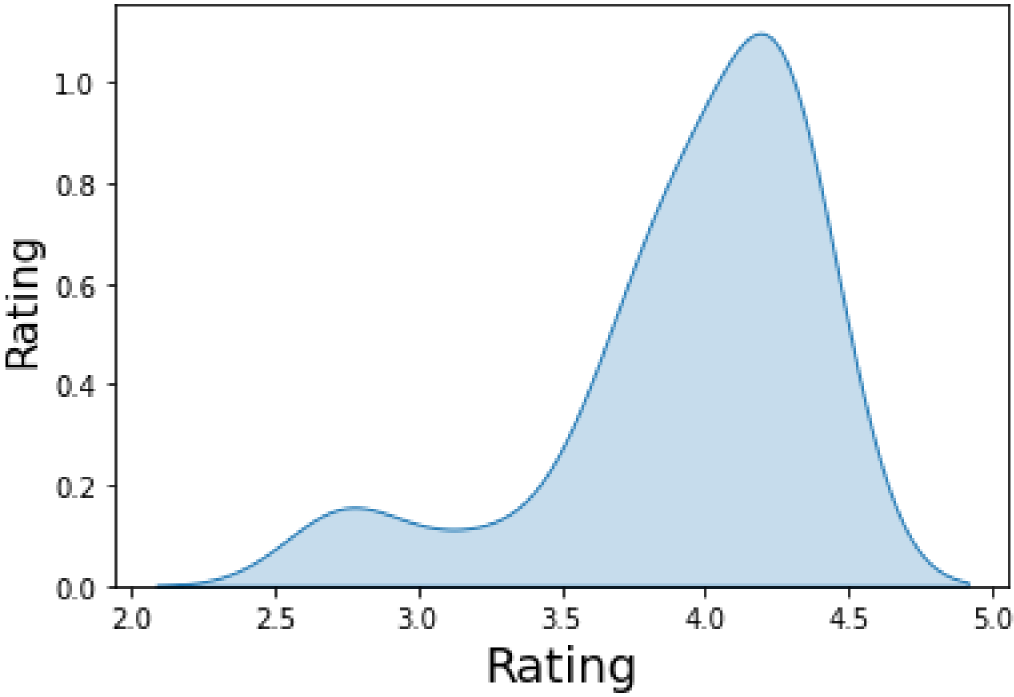


(ii)Rating

# Plot Distibution Plots of Price and Rating

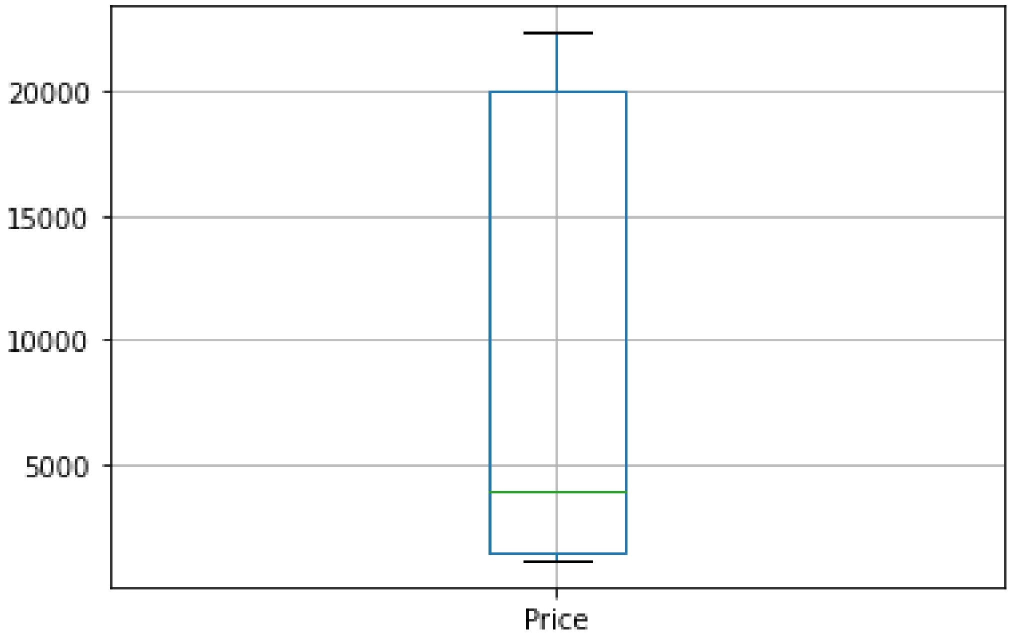
- columns=['Price','Rating']
- for i in columns:
- sns.kdeplot(df1[i],shade=True)
- plt.xlabel(i, fontsize=18)
- plt.ylabel('Rating', fontsize=16)
- plt.show()





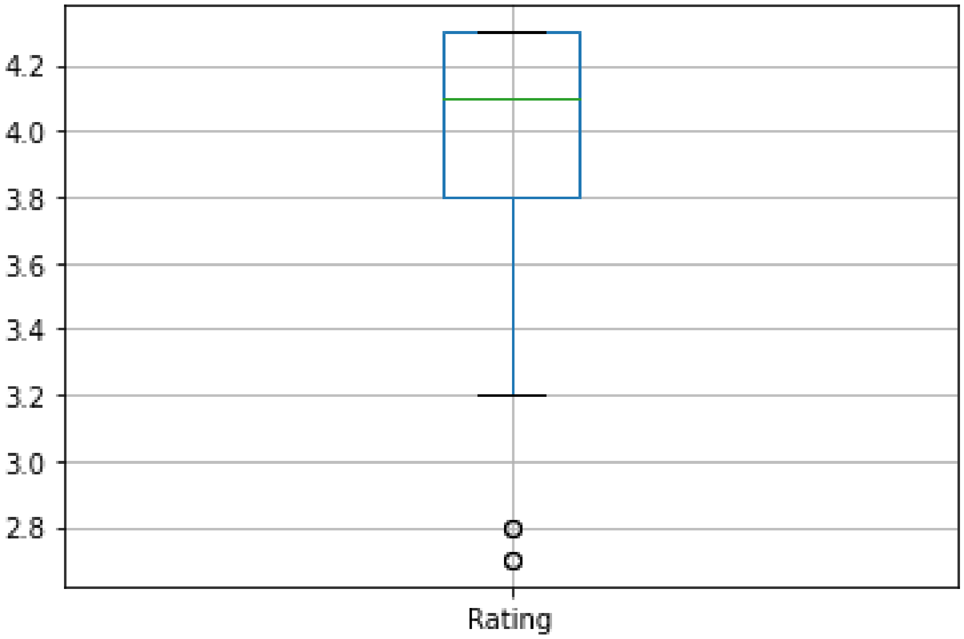
(iii)# Boxplot of Price using Dataframe method

- df1.boxplot(column='Price',grid=True,figsize=(6,4))
- plt.show()



(iv)# Box plot of Rating

- df1.boxplot(column='Rating',grid=True,figsize=(6,4))
- plt.show()

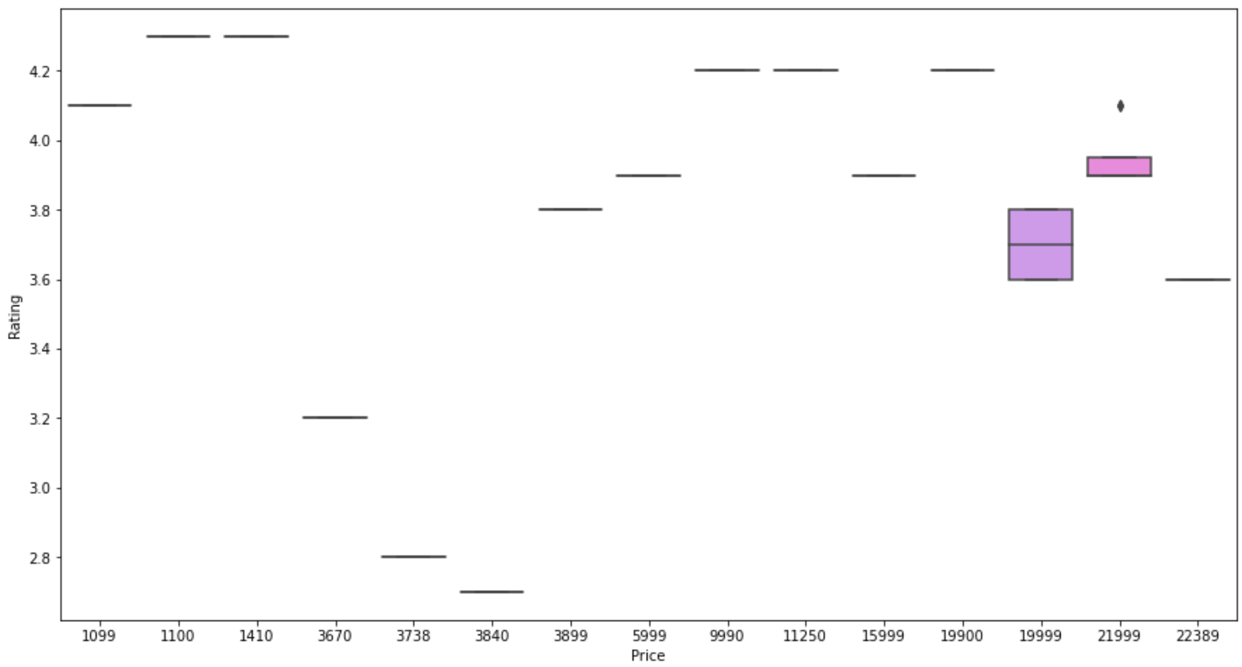


Rating

# Bivariate Analysis

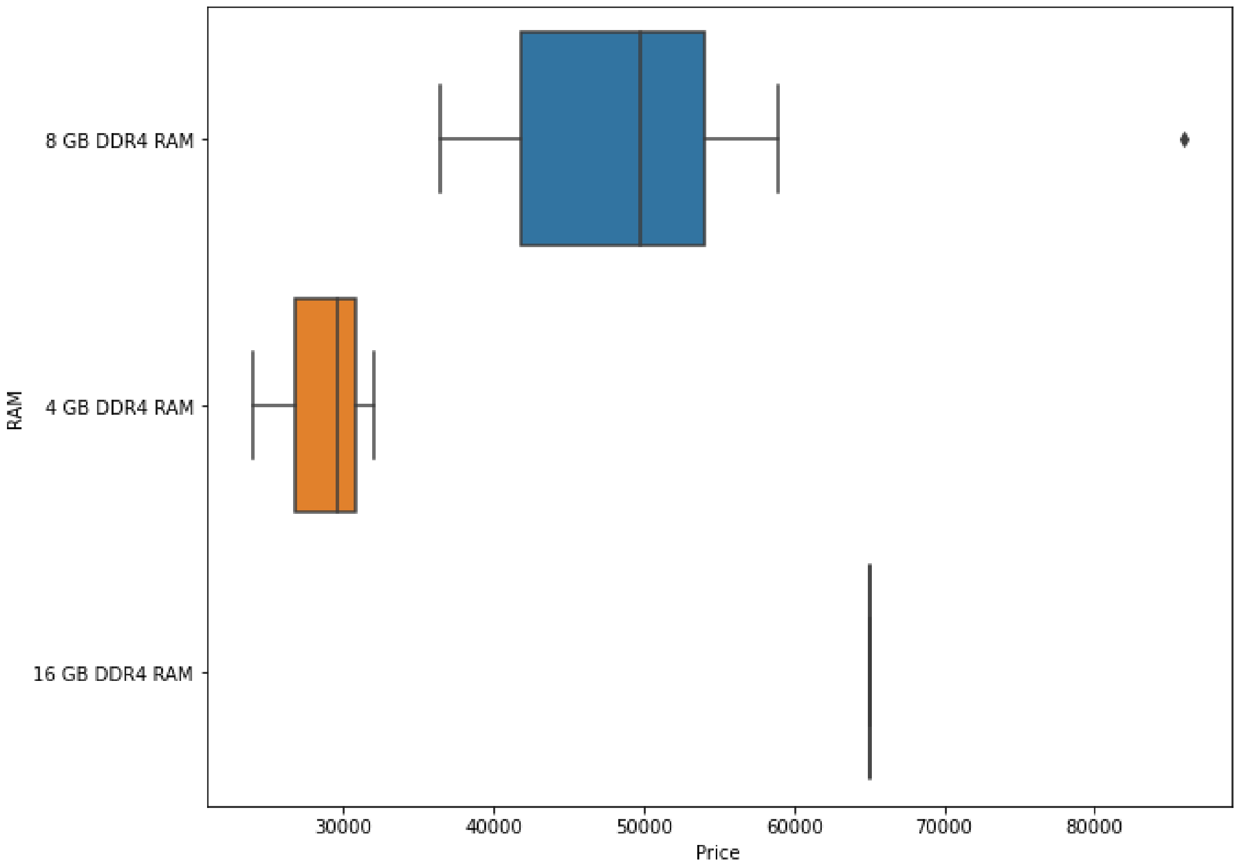
# Box plot of CPU and Price

- plt.figure(figsize=(15,8))
- sns.boxplot(y="Rating",x='Price',data=df1)
- plt.show()



# Box plot of RAM and Price

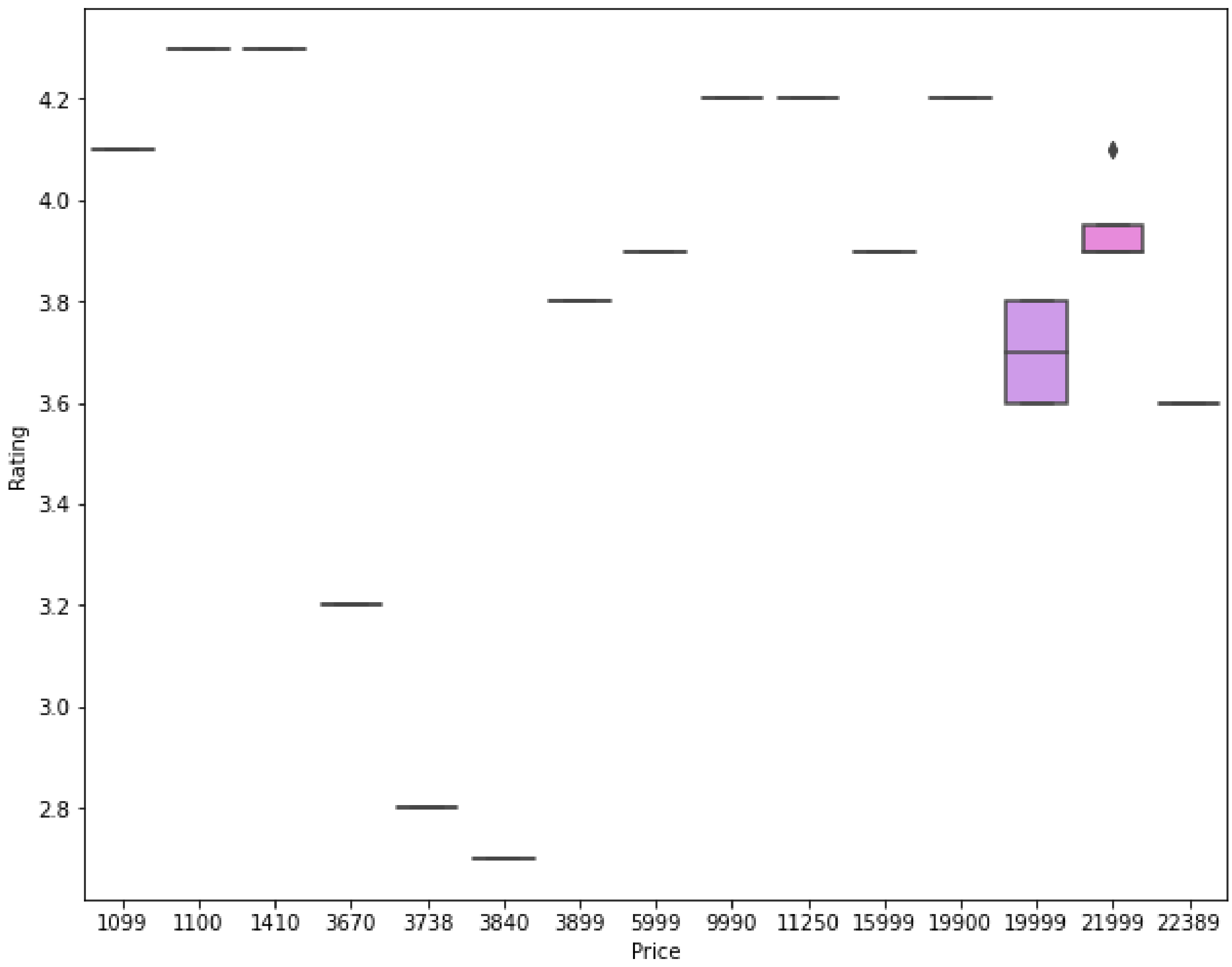
- plt.figure(figsize=(10,8))
- sns.boxplot(y="RAM",x='Price',data=df1)
- plt.show()



Rating

# Box plot of OS and Price

- plt.figure(figsize=(10,8))
- sns.boxplot(y="Rating",x='Price',data=df1)
- plt.show()



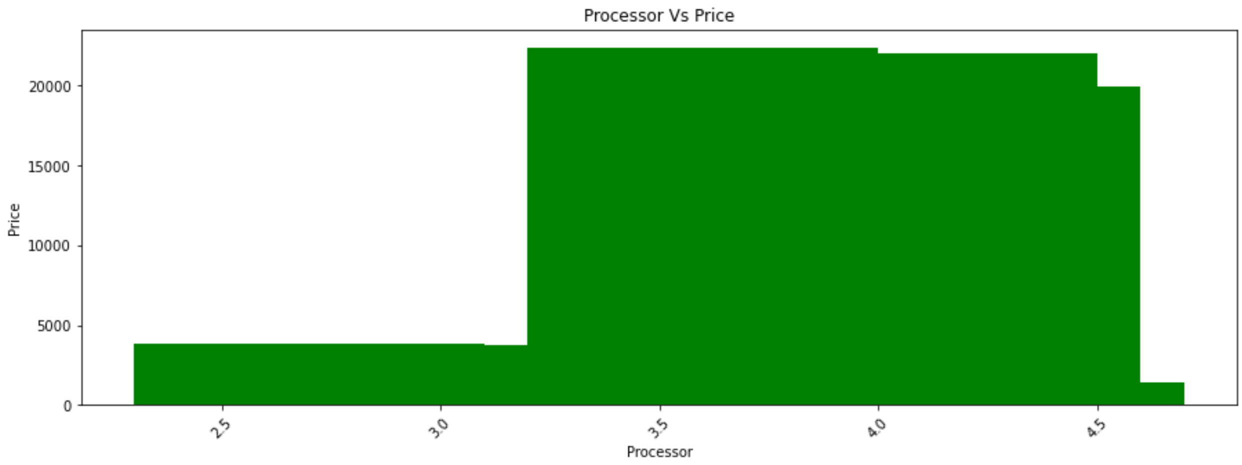
Bar Graphs using Matplotlib

# Bar Graph - Processor Vs Price

# Using plt

- plt.figure(figsize=(15,5))
- plt.bar(df1['Rating'],df1['Price'],color='green')
- plt.xticks(rotation=45)
- plt.xlabel('Processor')

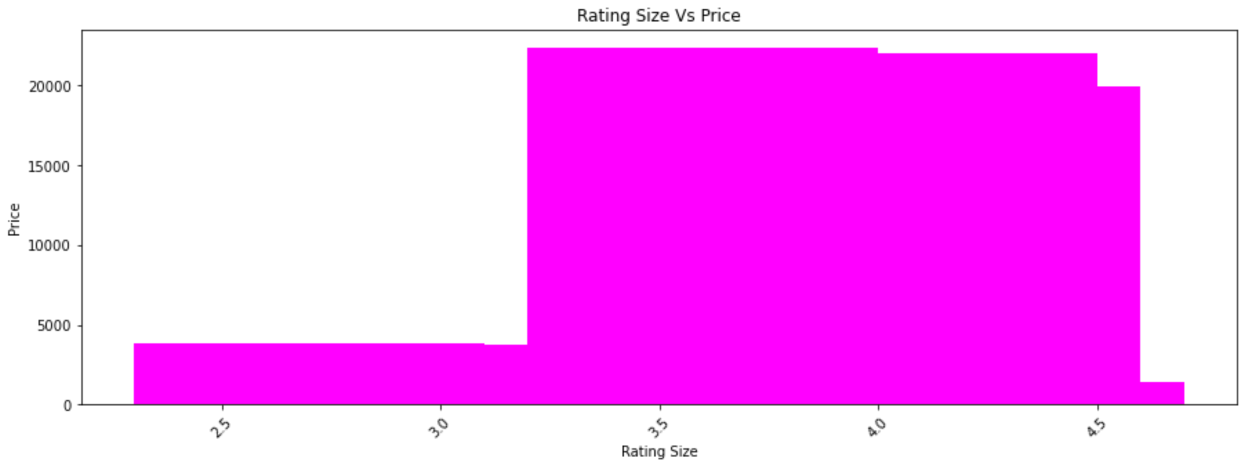
- plt.ylabel('Price')
- plt.title('Processor Vs Price')
- plt.show()



Rating Size

# Bar Graph - RAM Vs Price

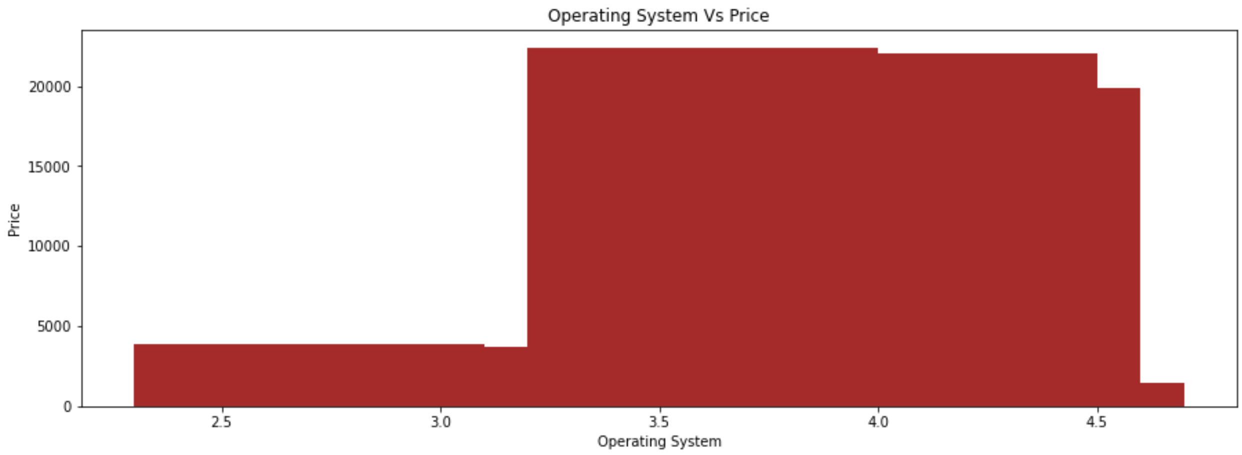
- plt.figure(figsize=(15,5))
- plt.bar(df1['Rating'],df1['Price'],color='fuchsia')
- plt.xticks(rotation=45)
- plt.xlabel('Rating Size')
- plt.ylabel('Price')
- plt.title('Rating Size Vs Price')
- plt.show()



# Findings from fig.

# Bar Graph - OS Vs Price

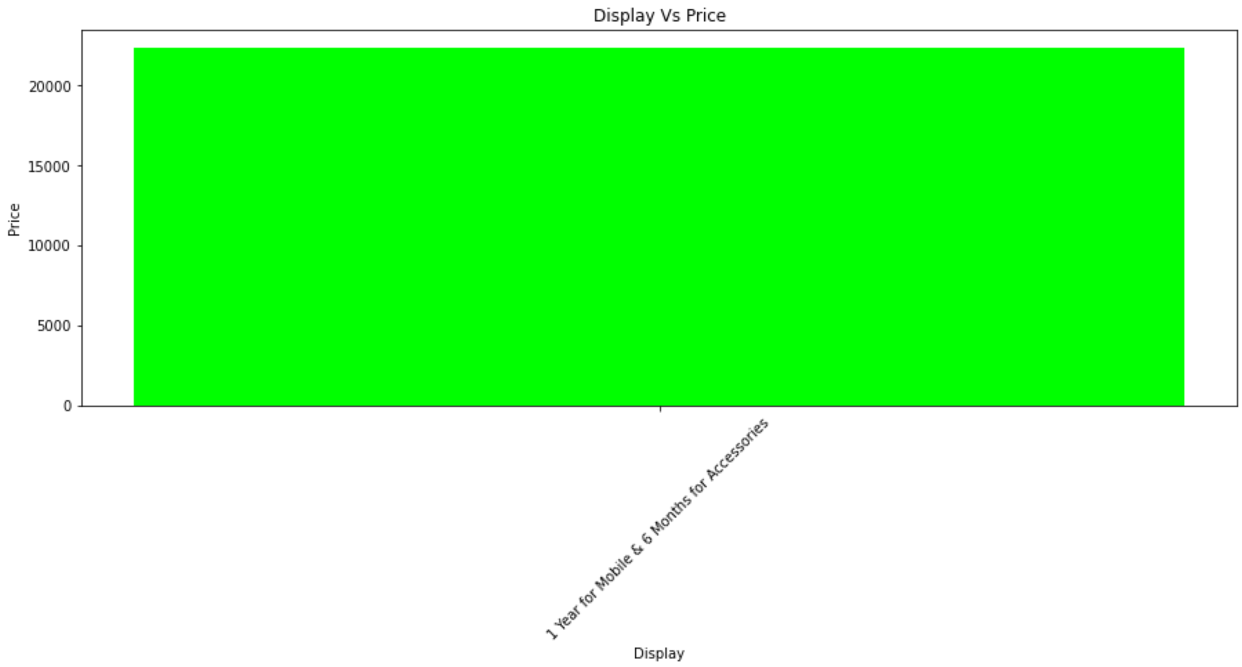
- plt.figure(figsize=(15,5))
- plt.bar(df1['Rating'],df1['Price'],color='brown')
- plt.xticks(rotation=0)
- plt.xlabel('Operating System')
- plt.ylabel('Price')
- plt.title('Operating System Vs Price')
- plt.show()



Display

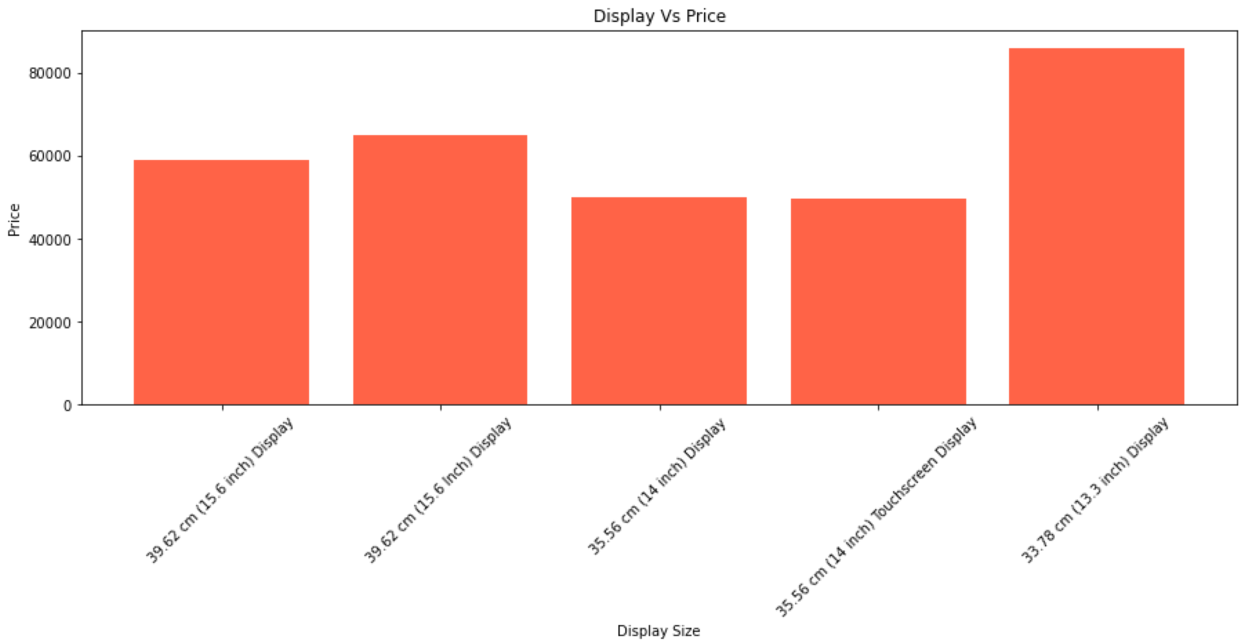
# Bar Graph - HDD Vs Price

- plt.figure(figsize=(15,5))
- plt.bar(df1['Display'],df1['Price'],color='lime')
- plt.xticks(rotation=45)
- plt.xlabel('Display')
- plt.ylabel('Price')
- plt.title('Display Vs Price')
- plt.show()



# Bar Graph - Display Vs Price

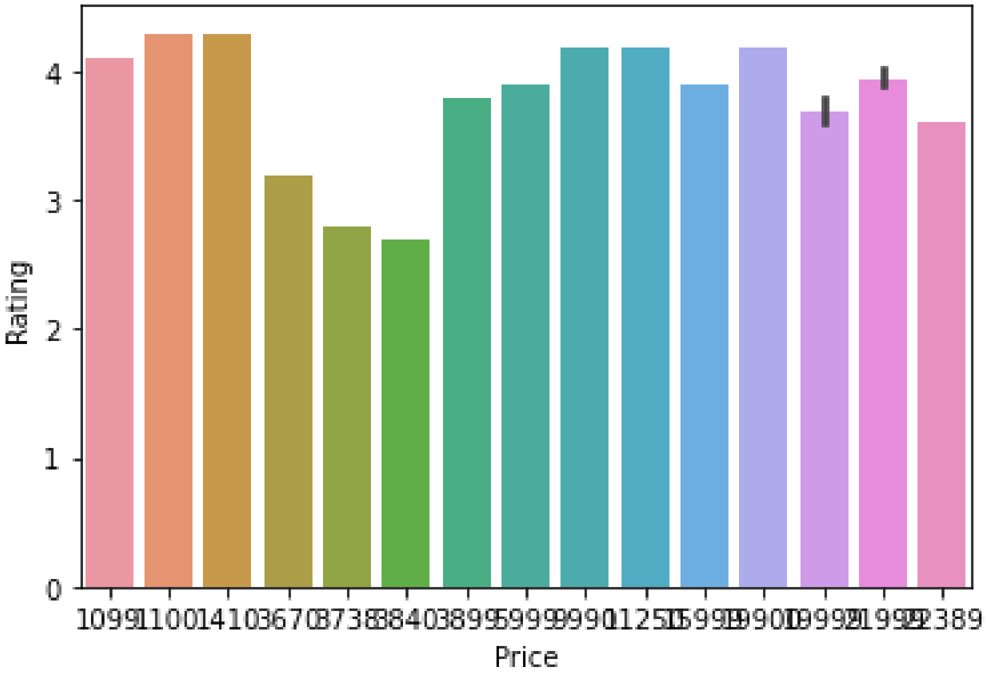
- plt.figure(figsize=(15,5))
- plt.bar(df1['Display'],df1['Price'],color='tomato')
- plt.xticks(rotation=45)
- plt.xlabel('Display Size')
- plt.ylabel('Price')
- plt.title('Display Vs Price')
- plt.show()



BarPlots using Seaborn library

Rating

```
# Bar Plot - Price Vs CPU

#sns.barplot(x=df1.Price, y=df1.Rating)

#sns.barplot(y=df1.Price, x=df1.Rating)

#plt.xticks(rotation=90)

<AxesSubplot:xlabel='Price', ylabel='Rating'>
```

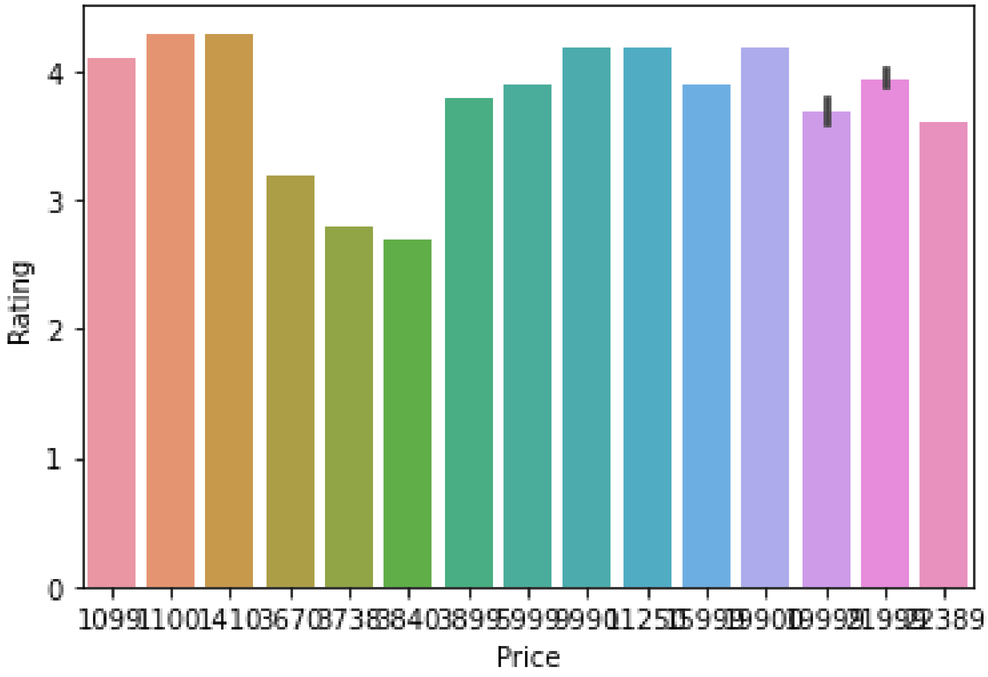


Rating

# Bar Plot - Price Vs RAM

- sns.barplot(x=df1.Price, y=df1.Rating)

<AxesSubplot:xlabel='Price', ylabel='Rating'>

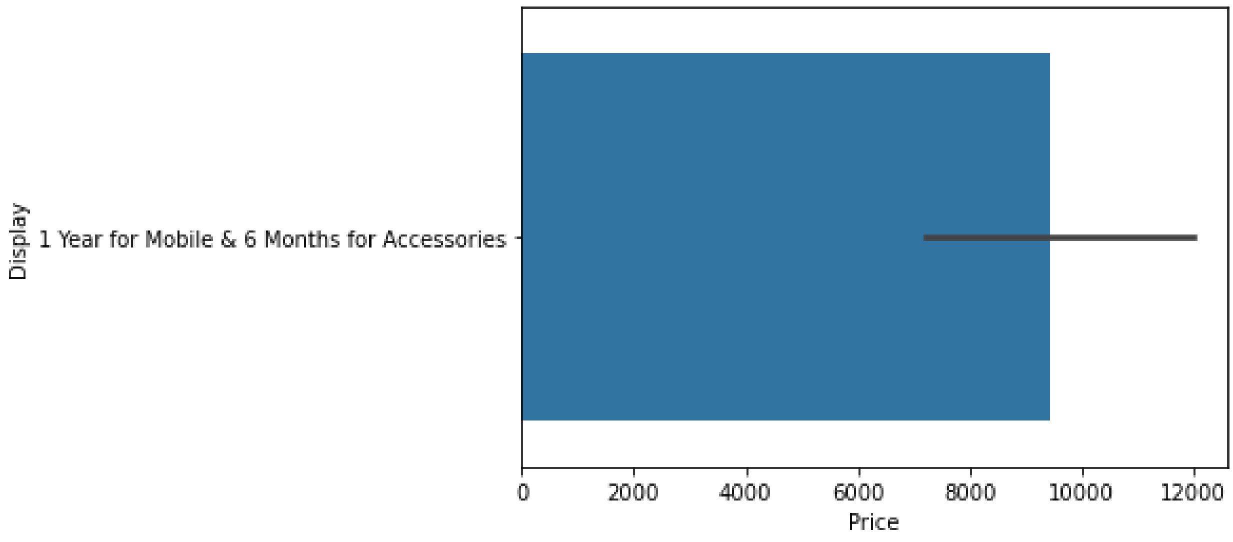


Display

# Bar Plot - Price Vs OS

- sns.barplot(x=df1['Price'], y=df1['Display'])

```
<AxesSubplot:xlabel='Price', ylabel='Display'>
```

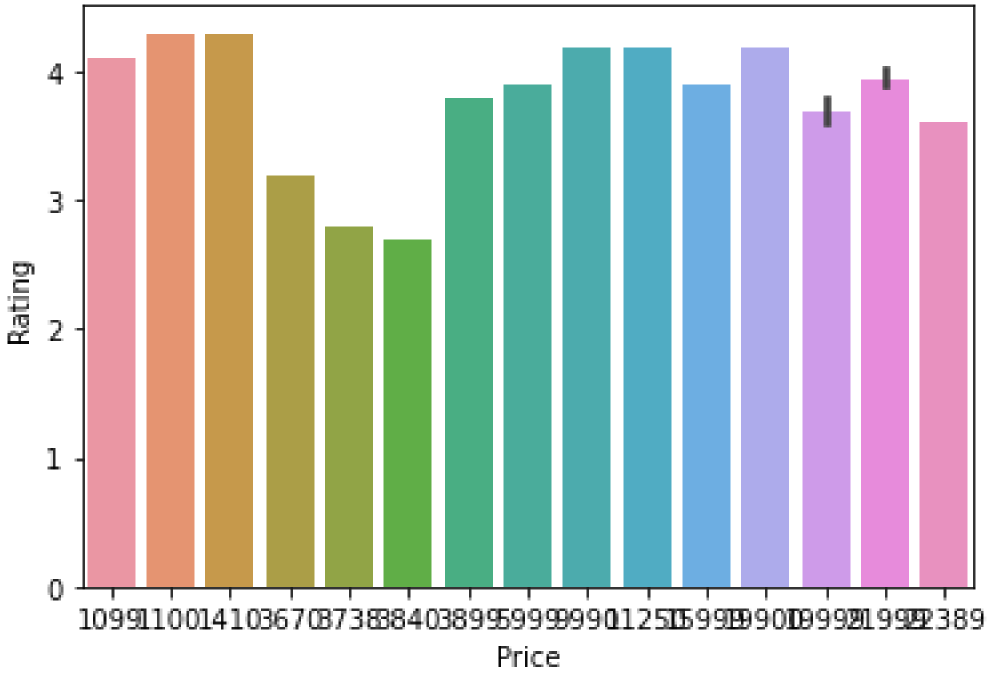


Rating

```
# Bar Plot - Price Vs Rating
```

```
sns.barplot(x=df1['Price'], y=df1['Rating'])
```

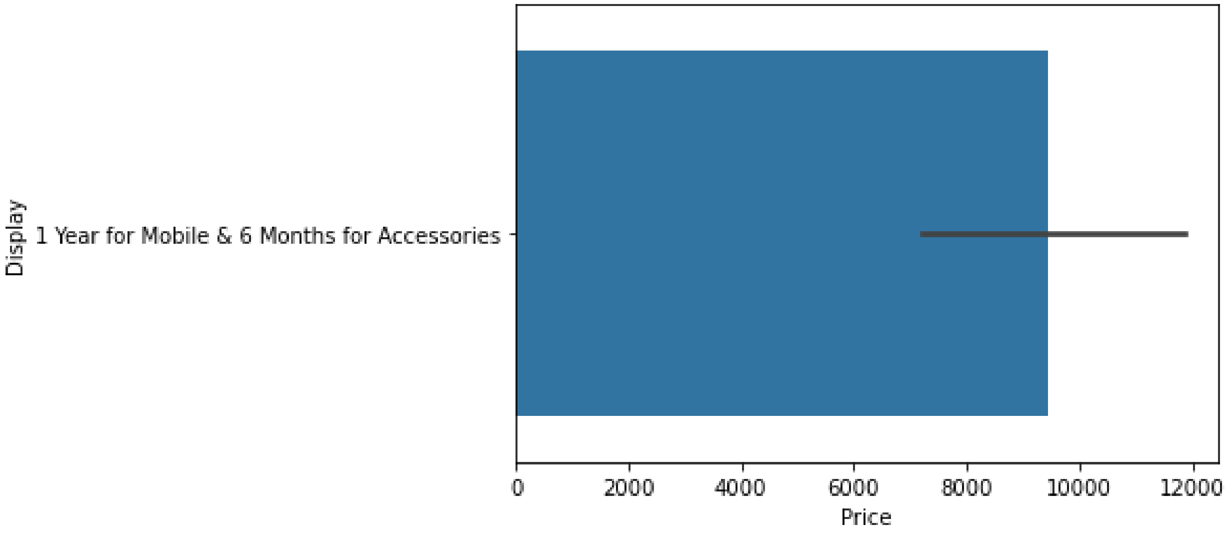
```
<AxesSubplot:xlabel='Price', ylabel='Rating'>
```



# Bar Plot - Price Vs Display

- sns.barplot(x=df1['Price'], y=df1['Display'])

<AxesSubplot:xlabel='Price', ylabel='Display'>



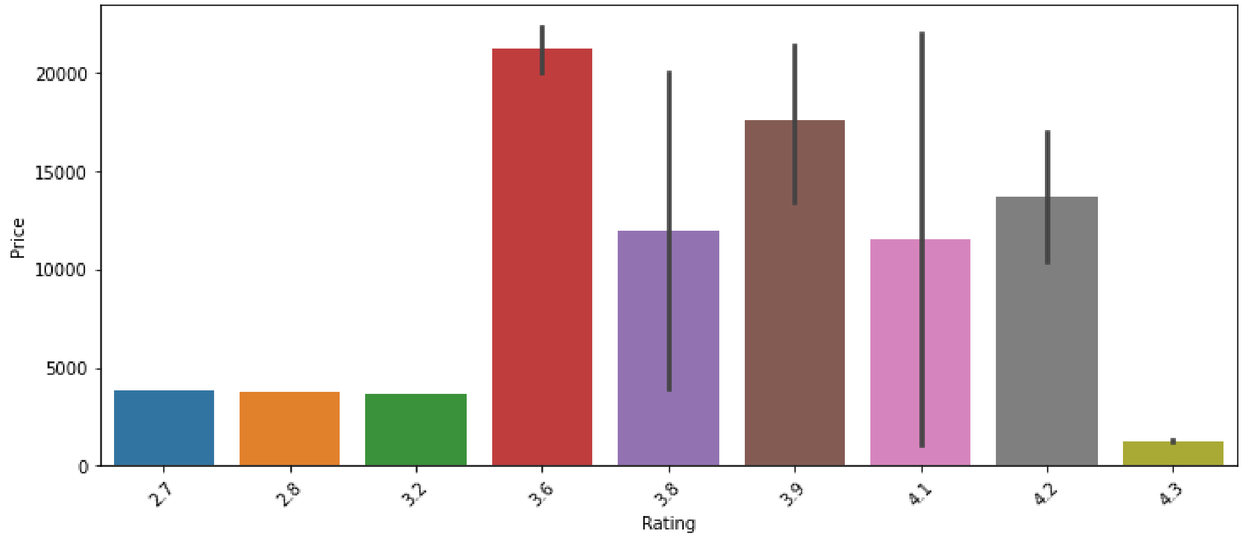
# BarPlots using Seaborn library

# Categorical Variables versus Price

Rating

# Bar Plot - Rating Vs Price

- plt.figure(figsize=(12,5))
- sns.barplot(x=df1['Rating'], y=df1['Price'])
- plt.xticks(rotation=45);

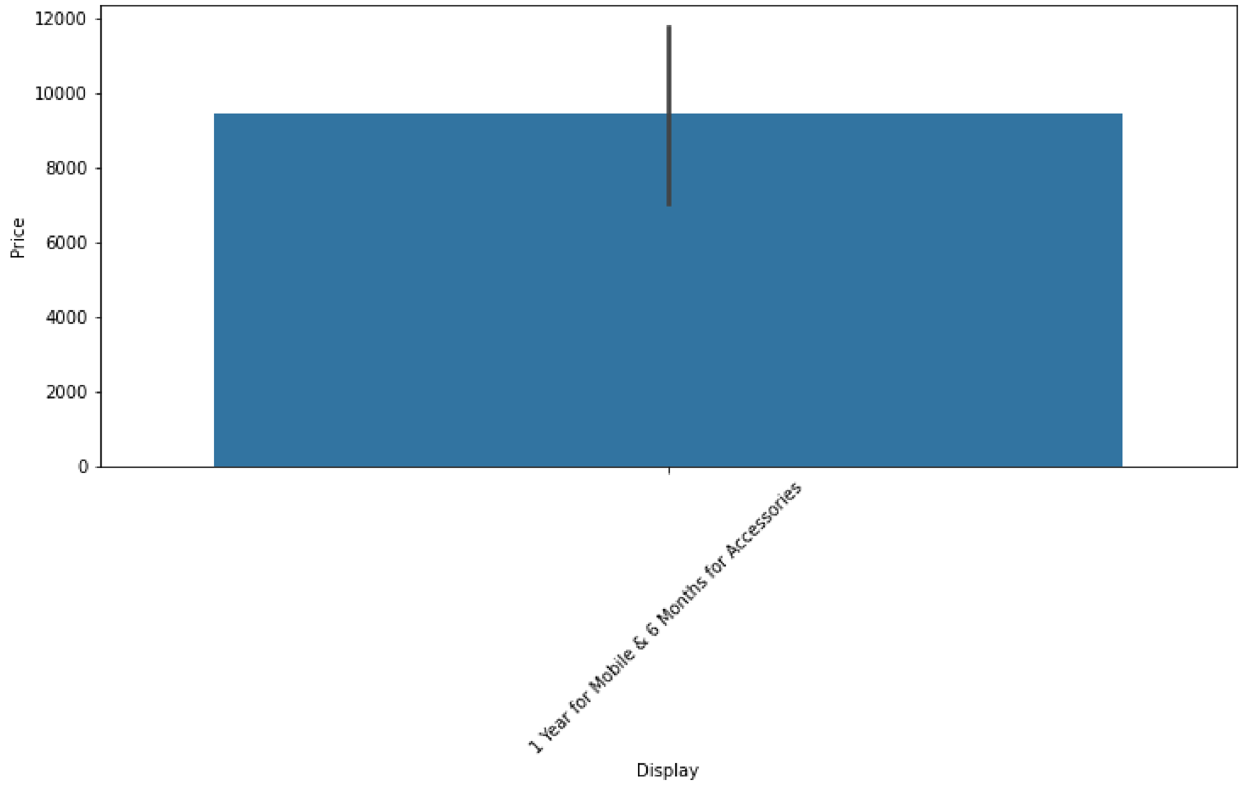


Display

# Bar Plot - Display Vs Price

- plt.figure(figsize=(12,5))

- sns.barplot(x=df1['Display'], y=df1['Price'])
- plt.xticks(rotation=45)
- plt.show();



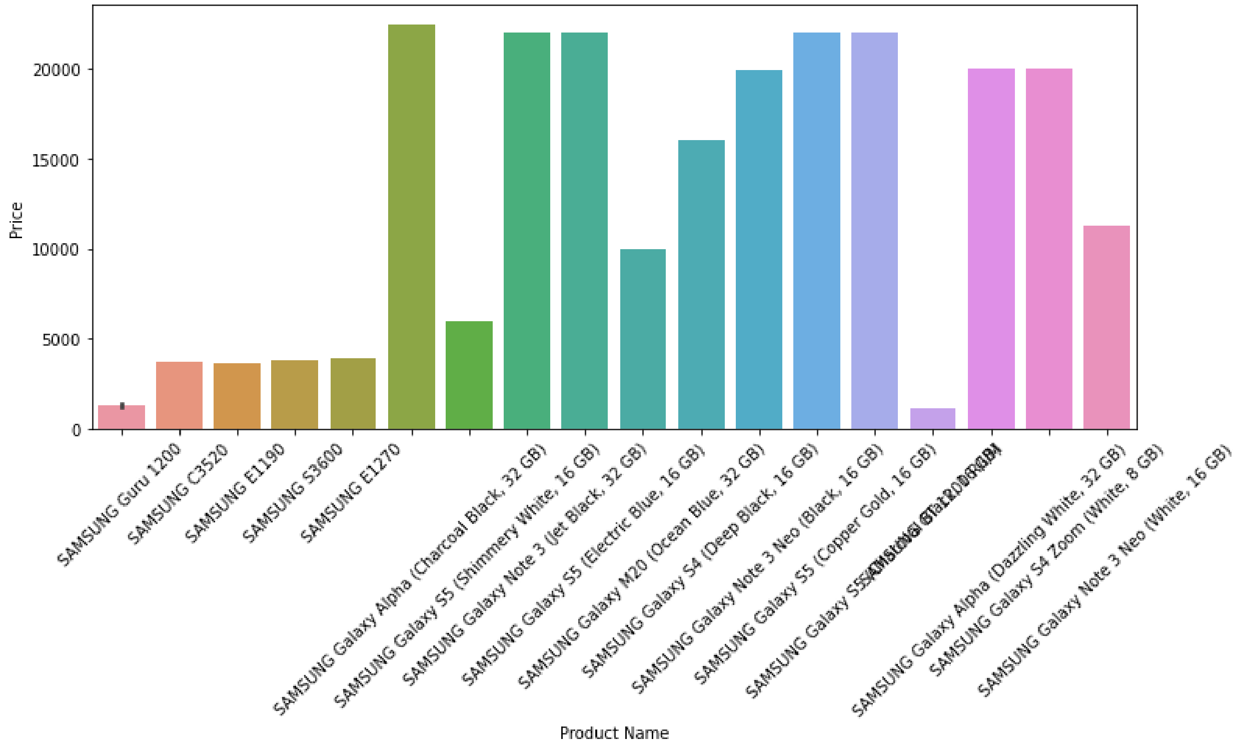
# Bar Plot - OS Vs Price

- plt.figure(figsize=(12,5))
- sns.barplot(x=df1['Rating'], y=df1['Price'])
- plt.xticks(rotation=0);

Product

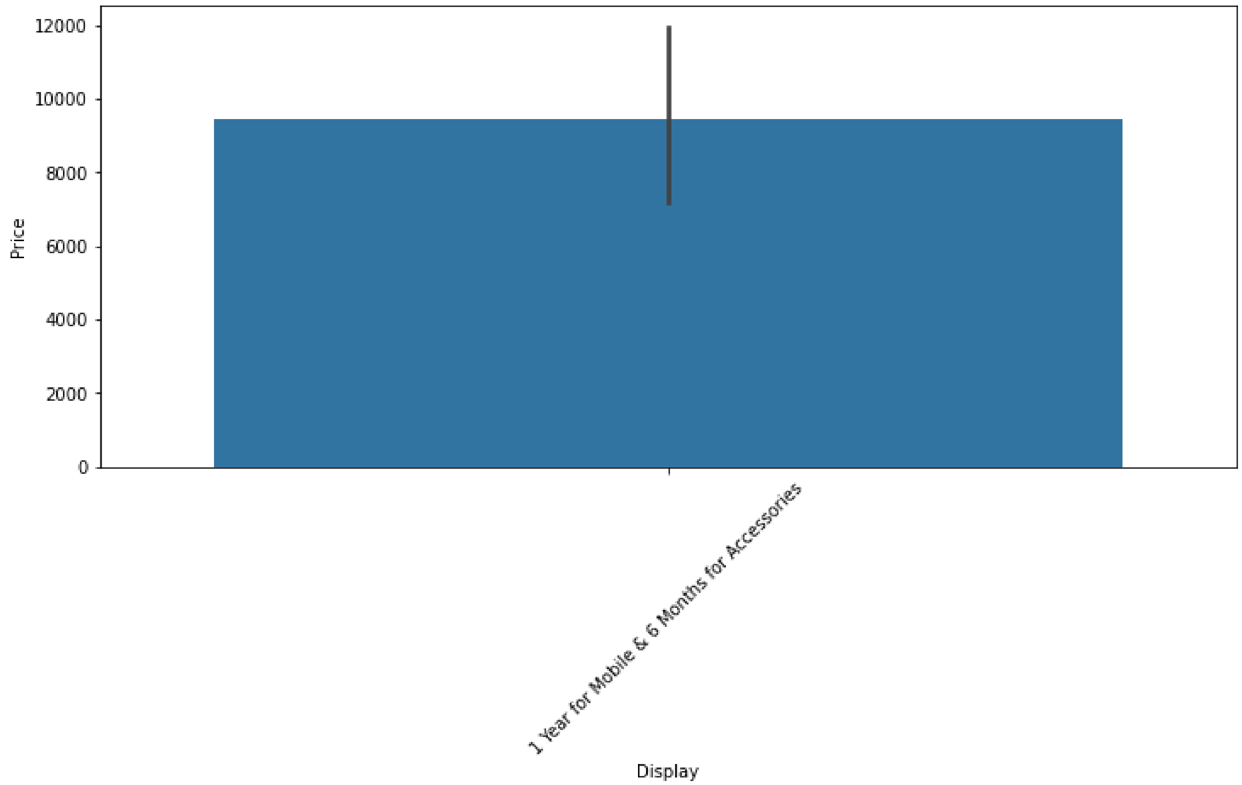
# Bar Plot - Product name Vs Price

- plt.figure(figsize=(12,5))
- sns.barplot(x=df1['Product Name'], y=df1['Price'])
- plt.xticks(rotation=45);



# Bar Plot - Display Vs Price

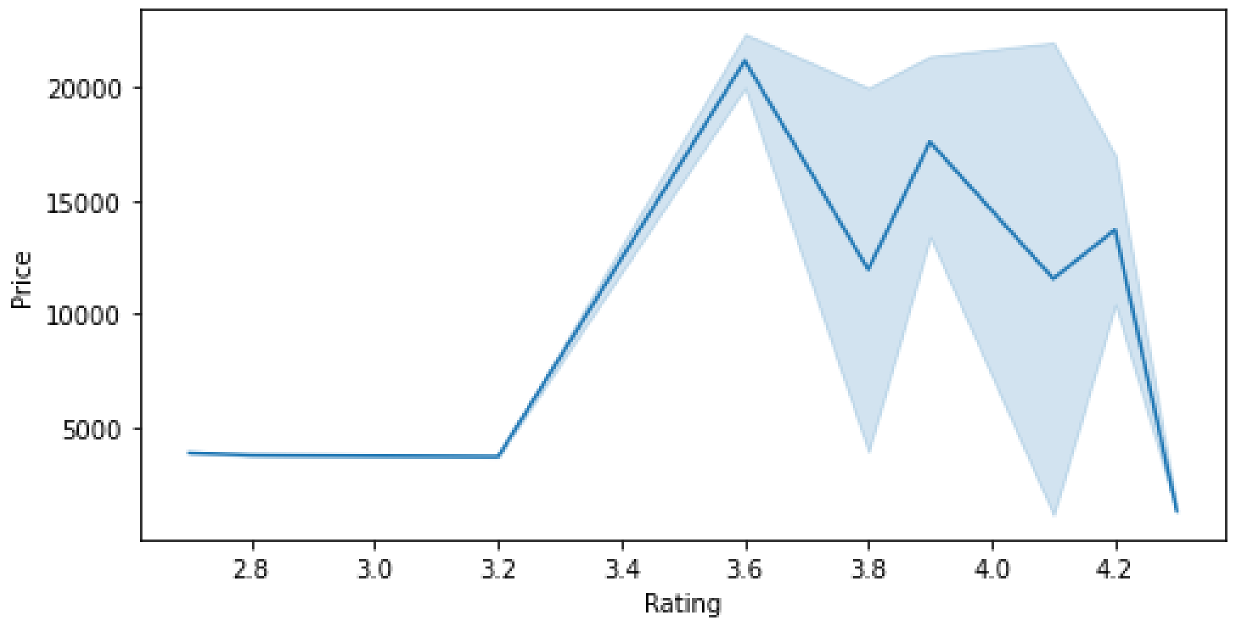
- plt.figure(figsize=(12,5))
- sns.barplot(x=df1['Display'], y=df1['Price'])
- plt.xticks(rotation=45)
- (array([0]), [Text(0, 0, '1 Year for Mobile & 6 Months for Accessories')])



# Line Plot - Rating Vs Price between categorical variables

- plt.figure(figsize=(8,4))
- sns.lineplot(x=df1['Rating'], y=df1['Price'])

```
<AxesSubplot:xlabel='Rating', ylabel='Price'>
```



Step:20: save the File .ipynb and data file in .csv excel file.

- According to final conclusion According to Rating and pricing depending on Rating and budget we can select the best Phone on those brands.

You learned how to:

- Inspect the HTML structure of your target site with your browser's tools
- Gain insight into how to decipher the data encoded in URLs
- Download the page's HTML content using Python's requests library
- Parse the downloaded HTML with Beautiful Soup to extract relevant information.

=====================xxxxx=========================