



Web-Scraping in Python with BeautifulSoup

Web Scraping

Web Scraping is a technique to fetch data and information from websites.



Everything you see on a webpage can be scraped.



Can be done in most programming languages, we'll use Python.

Is Web Scraping illegal ?

- ▶ Violation of the Computer Fraud and Abuse Act (CFAA)
- ▶ Violation of the Digital Millennium Copyright Act (DMCA)
- ▶ Trespass to Chattel
- ▶ Copy right infringement
- ▶ Breach of contract

Web crawling

-
- ▶ A web crawler (also known as a web spider or web robot).
 - ▶ It is a program or automated script which browses the World Wide Web in a methodical, automated manner.
 - ▶ Crawlers can also be used for automating maintenance tasks on a Web site.
 - ▶ Such as checking links or validating HTML code.

Uses

- ▲ Web Crawlers
- ▲ E-Commerce price comparer
- ▲ Preparing dataset for ML model
- ▲ Scraping Social Media Profiles
- ▲ Weather Data etc.

Purpose of project

Purpose of the project is to scrap data from flipcart mobile price list and store data in csv.

The data would be used for sentiment analysis.



Libraries, Workflow and Explanation

The project is basically divided into two parts :

(a) Scraping data from website.

(b) With the help of data sentiment analysis based on rating.

Scraping Data from Flipcart

- ▶ Library used:
- ▶ BeautifulSoup
- ▶ requests
- ▶ Csv
- ▶ Pandas
- ▶ matplotlib.pyplot
- ▶ Seaborn
- ▶ numpy

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import csv
from bs4 import BeautifulSoup
import requests
import seaborn as sp
import numpy as np
```


Csv lib:

- ▶ CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases.

In this project we used csv library for create spreadsheet which having fields or columns : name, price, rating, count_rating

So that their values get saved its respected field

```
file=open('flipkart2.csv','w',encoding='utf-8')  
fieldnames=['name','price','rating','count_rating']
```

Create flipkart.csv file with written mode and provides fields

Requests lib:

- ▶ Requests will allow you to send HTTP/1.1 requests using Python
- ▶ In this project we request from flipkart link

Link or url is

https://www.flipkart.com/mobiles/pr?sid=tyy%2C4io&otracker=clp_metro_expandable_5_3.metroExpandable.METRO_EXPANDABLE_Shop%2BNow_mobile-phones-store_Q0QIS4SPJNLH_wp3&fm=neo%2Fmerchandising&iid=M_c59a68bf-abb0-414d-a5a1-c7cc6ccdb202_3.Q0QIS4SPJNLH&ppt=browse&ppn=browse&ssid=b3fap623lc0000001638369228009&p%5B%5D=facets.brand%255B%255D%3DPOCO

Requesting:

```
page=requests.get(url)
```

BeautifulSoup:

Beautiful Soup is a Python package used for parsing HTML.

In this project : we parsing the code by html.parser(it will extract all html code of webpage)

```
soup=BeautifulSoup(page.content, 'html.parser')
print(soup)
```

Code is :

```
<!DOCTYPE doctype html>
<html lang="en"><head><link href="https://rukminim1.flixcart.com" rel="preconnect"/><link href="//static-assets-web.flixcart.com/www/linchpin/fk-cp-zion/css/app_modules.chunk.94b5e7.css" rel="stylesheet"/><link href="//static-assets-web.flixcart.com/www/linchpin/fk-cp-zion/css/app.chunk.dd97f3.css" rel="stylesheet"/><meta content="text/html; charset=utf-8" http-equiv="Content-Type"/><meta content="IE=Edge" http-equiv="X-UA-Compatible"/><meta content="102988293558" property="fb:page_id"/><meta content="658873552,624500995,100000233612389" property="fb:admins"/><meta content="noodp" name="robots"/><link href="https://static-assets-web.flixcart.com/www/promos/new/20150528-140547-favicon-retina.ico" rel="shortcut icon"/><link href="/osdd.xml?v=2" rel="search" type="application/opensearchdescription+xml" /><meta content="website" property="og:type"/><meta content="Flipkart.com" name="og:site_name" property="og:site_name"/><link href="//apple-touch-icon-57x57.png" rel="apple-touch-icon" sizes="57x57"/><link href="//apple-touch-icon-72x72.png" rel="apple-touch-icon" sizes="72x72"/><link href="//apple-touch-icon-114x114.png" rel="apple-touch-icon" sizes="114x114"/><link href="//apple-touch-icon-144x144.png" rel="apple-touch-icon" sizes="144x144"/><link href="//apple-touch-icon-57x57.png" rel="apple-touch-icon" /><meta content="app" name="twitter:card"/><meta content="@flipkart" name="twitter:site"/><meta content="@flipkart" name="twitter:creator"/><meta content="Mobile Price List | Compare Mobiles on Buy Online @ Flipkart" name="twitter:title"/><meta content="Shop for electronics, apparels & more using our Flipkart app Free Shipping & COD." name="twitter:description"/><meta content="in" name="twitter:app:country"/><meta content="Flipkart" name="al:ios:app_name"/><meta content="742044692" name="al:ios:app_store_id"/><meta content="Flipkart" name="twitter:app:name:iphone"/><meta content="742044692" name="twitter:app:id:iphone"/><meta content="http://dl.flipkart.com/dl/home?twitter:app:url:iphone" /><meta content="Flipkart" name="twitter:app:url:home?" name="twitter:app:url:ipad"/><meta content="742044692" name="twitter:app:id:ipad"/><meta content="http://dl.flipkart.com/dl/home?" name="twitter:app:url:ipad"/><meta content="Flipkart" name="
```

code

- ▶ Numbers of pages you want to scraping in this we used 4 pages which contain 72 products.

There are 8 empty lists are used for store their values

likes:

product list store mobile name or product name.

P_price list store prices of products

rating , sound ,hd, reviews lists store rating

, battery details, hd quality, reviews respectively

```
number_of_pages=4
product=[]
p_price=[]
rating=[]
apps= []
os= []
hd= []
sound= []
reviews=[]
```

For each page we used for loop

```
for i in range(1,number_of_pages):  
    j=str(i)
```

Then we work on products division which contain all details of products we call them by using their class name **_3ply-c row**

```
content=soup.find_all('div',class_='_3ply-c row')  
for item in content:
```

For each class of product we have to scraping to find details

```
for item in content:  
    item_name=item.find('div',class_='4rR01T')  
    item_price=item.find('div',class='_30jeq3_1_WHN1')  
    item_rating=item.find('div', attrs={'class': '_3LWZ1K'})  
    specification= item.find('div', attrs={'class': 'fMghEO'})
```

For each product we scrap their name ,price , rating ,specification by using of their class name a each division of respected class we have same details for all products

For item_name for div class name =“_4rR01T” contains names of products as shown

```
<div class="_4rR01T">POCO C31 (Royal Blue, 64 GB)</div>
<div class="_4rR01T">POCO C31 (Royal Blue, 32 GB)</div>
<div class="_4rR01T">POCO C31 (Shadow Gray, 64 GB)</div>
<div class="_4rR01T">POCO X3 Pro (Steel Blue, 128 GB)</div>
<div class="_4rR01T">POCO M2 Pro (Two Shades of Black, 64 GB)</div>
<div class="_4rR01T">POCO C31 (Shadow Gray, 32 GB)</div>
<div class="_4rR01T">POCO M2 Pro (Out of the Blue, 64 GB)</div>
<div class="_4rR01T">POCO C3 (Matte Black, 32 GB)</div>
<div class="_4rR01T">POCO M3 Pro 5G (Yellow, 64 GB)</div>
<div class="_4rR01T">POCO M3 Pro 5G (Yellow, 128 GB)</div>
<div class="_4rR01T">POCO C3 (Arctic Blue, 64 GB)</div>
<div class="_4rR01T">POCO M2 Pro (Green and Greener, 64 GB)</div>
<div class="_4rR01T">POCO M3 Pro 5G (Cool Blue, 64 GB)</div>
<div class="_4rR01T">POCO C3 (Matte Black, 64 GB)</div>
<div class="_4rR01T">POCO C3 (Arctic Blue, 32 GB)</div>
<div class="_4rR01T">POCO C3 (Lime Green, 32 GB)</div>
<div class="_4rR01T">POCO M3 Pro 5G (Power Black, 64 GB)</div>
<div class="_4rR01T">POCO F3 GT (Predator Black, 128 GB)</div>
<div class="_4rR01T">POCO F3 GT (Predator Black, 256 GB)</div>
<div class="_4rR01T">POCO F3 GT (Gunmetal Silver, 128 GB)</div>
<div class="_4rR01T">POCO F3 GT (Gunmetal Silver, 128 GB)</div>
<div class="_4rR01T">POCO F3 GT (Predator Black, 128 GB)</div>
<div class="_4rR01T">POCO F1 (Graphite Black, 256 GB)</div>
<div class="_4rR01T">POCO F1 (Steel Blue, 64 GB)</div>
<div class="_4rR01T">POCO C31 (Royal Blue, 64 GB)</div>
<div class="_4rR01T">POCO C31 (Royal Blue, 32 GB)</div>
<div class="_4rR01T">POCO C31 (Shadow Gray, 64 GB)</div>
<div class="_4rR01T">POCO X3 Pro (Steel Blue, 128 GB)</div>
<div class="_4rR01T">POCO M2 Pro (Two Shades of Black, 64 GB)</div>
<div class="_4rR01T">POCO C31 (Shadow Gray, 32 GB)</div>
<div class="_4rR01T">POCO M2 Pro (Out of the Blue, 64 GB)</div>
<div class="_4rR01T">POCO C3 (Matte Black, 32 GB)</div>
<div class="_4rR01T">POCO M3 Pro 5G (Yellow, 64 GB)</div>
<div class="_4rR01T">POCO M3 Pro 5G (Yellow, 128 GB)</div>
<div class="_4rR01T">POCO C3 (Arctic Blue, 64 GB)</div>
```

For item_price for div class name =“_30jeq3” contains names of products as shown

```
<div class="_30jeq3" _1_WHN1">₹9,499</div>
<div class="_30jeq3" _1_WHN1">₹8,499</div>
<div class="_30jeq3" _1_WHN1">₹9,499</div>
<div class="_30jeq3" _1_WHN1">₹20,999</div>
<div class="_30jeq3" _1_WHN1">₹14,999</div>
<div class="_30jeq3" _1_WHN1">₹8,499</div>
<div class="_30jeq3" _1_WHN1">₹13,999</div>
<div class="_30jeq3" _1_WHN1">₹8,499</div>
<div class="_30jeq3" _1_WHN1">₹14,499</div>
<div class="_30jeq3" _1_WHN1">₹16,499</div>
<div class="_30jeq3" _1_WHN1">₹9,499</div>
<div class="_30jeq3" _1_WHN1">₹13,999</div>
<div class="_30jeq3" _1_WHN1">₹14,499</div>
<div class="_30jeq3" _1_WHN1">₹9,499</div>
<div class="_30jeq3" _1_WHN1">₹8,499</div>
<div class="_30jeq3" _1_WHN1">₹8,499</div>
<div class="_30jeq3" _1_WHN1">₹14,499</div>
<div class="_30jeq3" _1_WHN1">₹28,999</div>
<div class="_30jeq3" _1_WHN1">₹30,999</div>
<div class="_30jeq3" _1_WHN1">₹28,999</div>
<div class="_30jeq3" _1_WHN1">₹28,999</div>
<div class="_30jeq3" _1_WHN1">₹28,999</div>
<div class="_30jeq3" _1_WHN1">₹28,999</div>
<div class="_30jeq3" _1_WHN1">₹28,999</div>
<div class="_30jeq3" _1_WHN1">₹21,999</div>
<div class="_30jeq3" _1_WHN1">₹9,499</div>
<div class="_30jeq3" _1_WHN1">₹8,499</div>
<div class="_30jeq3" _1_WHN1">₹9,499</div>
<div class="_30jeq3" _1_WHN1">₹20,999</div>
<div class="_30jeq3" _1_WHN1">₹14,999</div>
<div class="_30jeq3" _1_WHN1">₹8,499</div>
<div class="_30jeq3" _1_WHN1">₹13,999</div>
<div class="_30jeq3" _1_WHN1">₹8,499</div>
<div class="_30jeq3" _1_WHN1">₹14,499</div>
<div class="_30jeq3" _1_WHN1">₹16,499</div>
```


For specifications for div class name = "_1xgFaf" contains all details related to products as shown

```
<div class="fMghEO"><ul class="_1xgFaf"><li class="rgWa7D">4 GB RAM | 64 GB ROM | Expandable Upto 512 GB</li><li class="rgWa7D">16.59 cm (6.53 inch) HD+ Display</li><li class="rgWa7D">13MP + 2MP | 5MP Front Camera</li><li class="rgWa7D">5000 mAh Lithium-ion Polymer Battery</li><li class="rgWa7D">MediaTek Helio G35 Processor</li><li class="rgWa7D">1 Year Warranty for Handset, 6 Months for Accessories</li></ul></div>
<div class="fMghEO"><ul class="_1xgFaf"><li class="rgWa7D">3 GB RAM | 32 GB ROM | Expandable Upto 512 GB</li><li class="rgWa7D">16.59 cm (6.53 inch) HD+ Display</li><li class="rgWa7D">13MP + 2MP | 5MP Front Camera</li><li class="rgWa7D">5000 mAh Lithium-ion Polymer Battery</li><li class="rgWa7D">MediaTek Helio G35 Processor</li><li class="rgWa7D">1 Year Warranty for Handset, 6 Months for Accessories</li></ul></div>
<div class="fMghEO"><ul class="_1xgFaf"><li class="rgWa7D">4 GB RAM | 64 GB ROM | Expandable Upto 512 GB</li><li class="rgWa7D">16.59 cm (6.53 inch) HD+ Display</li><li class="rgWa7D">13MP + 2MP | 5MP Front Camera</li><li class="rgWa7D">5000 mAh Lithium-ion Polymer Battery</li><li class="rgWa7D">MediaTek Helio G35 Processor</li><li class="rgWa7D">1 Year Warranty for Handset, 6 Months for Accessories</li></ul></div>
<div class="fMghEO"><ul class="_1xgFaf"><li class="rgWa7D">8 GB RAM | 128 GB ROM | Expandable Upto 1 TB</li><li class="rgWa7D">16.94 cm (6.67 inch) Full HD+ Display</li><li class="rgWa7D">48MP + 8MP + 2MP | 20MP Front Camera</li><li class="rgWa7D">5160 mAh Lithium-ion Polymer Battery</li><li class="rgWa7D">Qualcomm Snapdragon 860 Processor</li><li class="rgWa7D">Multiple Hands-free Voice Assistants</li><li class="rgWa7D">One Year Warranty for Handset, 6 Months for Accessories</li></ul></div>
<div class="fMghEO"><ul class="_1xgFaf"><li class="rgWa7D">6 GB RAM | 64 GB ROM | Expandable Upto 512 GB</li><li class="rgWa7D">16.94 cm (6.67 inch) Full HD+ Display</li><li class="rgWa7D">48MP + 8MP + 5MP + 2MP | 16MP Front Camera</li><li class="rgWa7D">
```


Extract details from specification
division

Like :extract its sound details , its
hd details , its os detail so one ...

```
for each in specification:
    col=each.find_all('li', attrs={'class':'rgWa7D'})
    app=col[0].text
    os = col[1].text
    hd_ = col[2].text
    sound_ = col[3].text
    #if item name is not None.
```

```
df=pd.DataFrame({'Product_Name':product, 'Supported_apps':apps, 'sound_system':sound, 'Os':os, "Resolution":hd, 'Price':p_price, 'Rat
```

We made a dataframe that will store all values or details columns wise

Output :

	Product_Name	Supported_apps	sound_system	OS	Resolution	Price	Rating
0	POCO C31 (Royal Blue, 64 GB)	4 GB RAM 64 GB ROM Expandable Upto 512 GB	5000 mAh Lithium-ion Polymer Battery	16.59 cm (6.53 inch) HD+ Display	13MP + 2MP + 2MP 5MP Front Camera	₹9,499	4.4
1	POCO C31 (Royal Blue, 32 GB)	3 GB RAM 32 GB ROM Expandable Upto 512 GB	5000 mAh Lithium-ion Polymer Battery	16.59 cm (6.53 inch) HD+ Display	13MP + 2MP + 2MP 5MP Front Camera	₹8,499	4.3
2	POCO C31 (Shadow Gray, 64 GB)	4 GB RAM 64 GB ROM Expandable Upto 512 GB	5000 mAh Lithium-ion Polymer Battery	16.59 cm (6.53 inch) HD+ Display	13MP + 2MP + 2MP 5MP Front Camera	₹9,499	4.4
3	POCO X3 Pro (Steel Blue, 128 GB)	8 GB RAM 128 GB ROM Expandable Upto 1 TB	5160 mAh Lithium-ion Polymer Battery	16.94 cm (6.67 inch) Full HD+ Display	48MP + 8MP + 2MP + 2MP 20MP Front Camera	₹20,999	4.4
4	POCO M2 Pro (Two Shades of Black, 64 GB)	6 GB RAM 64 GB ROM Expandable Upto 512 GB	5000 mAh Lithium-ion Polymer Battery	16.94 cm (6.67 inch) Full HD+ Display	48MP + 8MP + 5MP + 2MP 16MP Front Camera	₹14,999	4.4
5	POCO C31 (Shadow Gray, 32 GB)	3 GB RAM 32 GB ROM Expandable Upto 512 GB	5000 mAh Lithium-ion Polymer Battery	16.59 cm (6.53 inch) HD+ Display	13MP + 2MP + 2MP 5MP Front Camera	₹8,499	4.3
6	POCO M2 Pro (Out of the Blue, 64 GB)	4 GB RAM 64 GB ROM Expandable Upto 512 GB	5000 mAh Lithium-ion Polymer Battery	16.94 cm (6.67 inch) Full HD+ Display	48MP + 8MP + 5MP + 2MP 16MP Front Camera	₹13,999	4.3
7	POCO C3 (Matte Black, 32 GB)	3 GB RAM 32 GB ROM	5000 mAh Li-ion Polymer	16.59 cm (6.53 inch)	13MP + 2MP + 2MP 5MP	₹8,499	4.3

- Totals Products in 4 pages are 72 and 7 columns

Now we got data and arrange them in csv file which contains all details related product

71	SAMSUNG Galaxy A03 Core (Blue, 32 GB)	2 GB Ex
72 rows x 7 columns		

Sentiment Analysis



In [13]:

```
from textblob import TextBlob
from wordcloud import WordCloud
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import import RegexpTokenizer
from nltk.stem.porter import import PorterStemmer
```

Library used:

- TextBlob
- WordCloud
 - re
 - nltk
- Stopwords
- RegexpTokenizer
- PorterStemmer
 - matplotlib

TextBlob and WordCloud

- ▶ TextBlob is a Python library for processing textual data
- ▶ Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.

For plot word cloud we use code:

```
allWords = ''.join([twts for twts in df['Product_Name']])
wordCloud = WordCloud(width = 500,height = 300,random_state = 21,max_font_size =119).generate(allWords)
plt.imshow(wordCloud,interpolation = "bilinear")
plt.axis('off')
plt.show()
```

In this word get cloud for samsung mobile data

- ▶ In this we a function which tokenize the word with respect to word +
- ▶ And make pattern for not take stop Words
- And clean the text
- Return that all text

```
: def getCleanedText(text):  
    text=text.lower()  
    tokenizer=RegexTokenizer(r'\w+')  
    ps=PorterStemmer()  
    en_stop=set(stopwords.words('english'))  
    tokens=tokenizer.tokenize(text)  
    new_tokens=[token for token in tokens if token not in en_stop]  
    stemmed_tokens=[ps.stem(tokens) for tokens in new_tokens]  
    clean_text=" ".join(stemmed_tokens)  
    return clean_text
```


Apply clean text function in column

```
df['Price']=df['Price'].apply(getCleanedText)
```

df

	Product_Name	Supported_apps	sound_system	OS	Resolution	Price	Rating
0	SAMSUNG Galaxy F22 (Denim Blue, 128 GB)	6 GB RAM 128 GB ROM Expandable Upto 1 TB	6000 mAh Lithium-ion Battery	16.26 cm (6.4 inch) HD+ Display	48MP + 8MP + 2MP + 2MP 13MP Front Camera	14 999	4.3
1	SAMSUNG Galaxy F22 (Denim Black, 128 GB)	6 GB RAM 128 GB ROM Expandable Upto 1 TB	6000 mAh Lithium-ion Battery	16.26 cm (6.4 inch) HD+ Display	48MP + 8MP + 2MP + 2MP 13MP Front Camera	14 999	4.3
2	SAMSUNG Galaxy F12 (Sky Blue, 64 GB)	4 GB RAM 64 GB ROM Expandable Upto 512 GB	6000 mAh Lithium-ion Battery	16.55 cm (6.515 inch) HD+ Display	48MP + 5MP + 2MP + 2MP 8MP Front Camera	11 499	4.2
3	SAMSUNG Galaxy F12 (Celestial Black, 64 GB)	4 GB RAM 64 GB ROM Expandable Upto 512 GB	6000 mAh Lithium-ion Battery	16.55 cm (6.515 inch) HD+ Display	48MP + 5MP + 2MP + 2MP 8MP Front Camera	11 499	4.2
4	SAMSUNG Galaxy F12 (Sea Green, 64 GB)	4 GB RAM 64 GB ROM Expandable Upto 512 GB	6000 mAh Lithium-ion Battery	16.55 cm (6.515 inch) HD+ Display	48MP + 5MP + 2MP + 2MP 8MP Front Camera	11 499	4.2
...
67	SAMSUNG Galaxy A03 Core (Black, 32 GB)	2 GB RAM 32 GB ROM Expandable Upto 1 TB	5000 mAh Li-ion Battery	16.51 cm (6.5 inch) HD+ Display	8MP Rear Camera 5MP Front Camera	7 999	4.5
68	SAMSUNG Galaxy A70s (Prism Crush White, 128 GB)	8 GB RAM 128 GB ROM Expandable Upto 512 GB	4500 mAh Lithium Ion Battery	17.02 cm (6.7 inch) Full HD+ Display	64MP + 5MP + 8MP 32MP Front Camera	19 999	4.3
69	SAMSUNG Galaxy M32 (Black, 128 GB)	6 GB RAM 128 GB ROM Expandable Upto 1 TB	6000 mAh Battery	16.26 cm (6.4 inch) Full HD+ Display	64MP + 8MP + 2MP + 2MP 20MP Front Camera	17 730	4.3

Then we normalized the rating value as true

- ▶ As it show value for 4.2 rating we get normalize value of 0.541667
- ▶ And similar to others

```
new_df['Rating'].value_counts(normalize=True)

4.2    0.541667
4.3    0.375000
4.5    0.083333
Name: Rating, dtype: float64
```

Now it turn of accuracy

(for accuracy we design a model)

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(ngram_range=(1,2))
X_cv=cv.fit_transform(df['Product_Name']).toarray()
X=X_cv
y=df['Rating']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
RF=RandomForestClassifier()
RF_model=RF.fit(X_train,y_train)
print("Train Accuracy",RF_model.score(X_train,y_train))
print("Test Accuracy",RF_model.score(X_test,y_test))
```

Train Accuracy 1.0
Test Accuracy 1.0

Thank you

