WebScraping Homework

April 30, 2024

[2]:
```
from bs4 import BeautifulSoup
import requests
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

1 Q1.) Web Crawling Tables

1.0.1 Q1.A.) Create a list of links for all the wikipedia pages for NYSE traded companies A-Z and 0-9

Such a link has the following format: "https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stoc where ("...") contains either "0-9" or a capital letter from A to Z. To create a list of these links, I will use a for loop in where each iteration will ad either "0-9" or a capitall letter from A to Z to the base URL (https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_).
The list of links for all of the Wikipedia pages for NYSE traded companies A-Z and 0-9 will be in a list object called "link_list".

[3]:
```
baseURL = "https://en.wikipedia.org/wiki/
 ↪Companies_listed_on_the_New_York_Stock_Exchange_"
to_add = ['0-9', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',␣
 ↪'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
link_list=[]

for item in to_add:
    link_list.append(f"{baseURL}({item})")

link_list[:10]
```

[3]:
```
['https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(0-9)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(A)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(B)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
```

```
(C)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(D)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(E)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(F)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(G)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(H)',
 'https://en.wikipedia.org/wiki/Companies_listed_on_the_New_York_Stock_Exchange_
(I)']
```

1.0.2 Q1.B.) Crawl through all the URLs and make 1 DF with all the NYSE publically traded companies

Below I write a function that will return all of the tickers of NYSE publicly traded companies from a sepcific URL. The logic is, of course, to then use a for loop to iterate over the list of links from the previous question and create a new list containing the tickers of all publicly traded companies on the NYSE.

[9]:
```
def get_names(URL):
    user_agent_list = ["Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
 ↪537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246"
    ,"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like␣
 ↪Gecko) Chrome/47.0.2526.111 Safari/537.36 "
    ,"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9␣
 ↪(KHTML, like Gecko) Version/9.0.2 Safari/601.3.9 "
    ,"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:15.0) Gecko/20100101 Firefox/
 ↪15.0.1"
    ,"Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/537.36 (KHTML, like␣
 ↪Gecko) Chrome/51.0.2704.64 Safari/537.36"]

    headers = {
        'User-Agent': user_agent_list[2]}

    page = requests.get(URL, headers = headers)

    soup = BeautifulSoup(page.content, "html.parser")

    comps = soup.findAll("tr")

    companies = [comp.find_all('td')[0].get_text(strip=True) for comp in comps␣
 ↪if comp.find('td')]
    tickers = [comp.find_all('td')[1].get_text(strip=True) for comp in comps if␣
 ↪comp.find('td')]
```

```
    return (companies, tickers)
```

I now iterate through each link in the list of links I previously created and retrieve the tickers of all publicly traded companies from that link. I store the names of all NYSE publicly traded companies on a list I call "nyse_tickers".

[13]:
```
nyse_companies=[]
nyse_tickers=[]

for link in link_list:
    companies, tickers = get_names(link)
    nyse_companies.extend(companies)
    nyse_tickers.extend(tickers)

print(nyse_companies[:5])
print(nyse_tickers[:5])
```

```
['10X Capital Venture Acquisition Corp. III', '10X Capital Venture Acquisition
Corp. III', '3D Systems Corporation', '3M Company', 'A. O. Smith Corporation']
['VCXB', 'VCXB.U', 'DDD', 'MMM', 'AOS']
```

[15]:
```
st_df = pd.DataFrame({'Company':nyse_companies, 'Ticker': nyse_tickers})
print(st_df.head())
```

```
                                    Company  Ticker
0  10X Capital Venture Acquisition Corp. III    VCXB
1  10X Capital Venture Acquisition Corp. III  VCXB.U
2                      3D Systems Corporation     DDD
3                                  3M Company     MMM
4                    A. O. Smith Corporation     AOS
```

1.0.3 Q1.C.) What is the percetages of companies that contain 1 letter, 2 letters, 3 letters, 4 letters, 5 letters,... in the ticker (drop punctuation)?

Before computing these percentages, I need to find what longest ticker length so that I know how many percentages to compute. In this case, the longest ticker length is 11, so I will be calculating the percentage of companies containing 1 letter, 2 letters,..., 11 letters.

[40]:
```
max_l=0

for ticker in nyse_tickers:
    if len(ticker) >max_l:
        max_l = len(ticker)

print(f"The longest ticker from our list has {max_l} letters")
```

```
The longest ticker from our list has 11 letters
```

[41]:
```
ticker_lengths = [0] * 11

for ticker in nyse_tickers:
    ticker_lengths[len(ticker)-1] +=1

for i,length in enumerate(ticker_lengths):
    print(f"The percentage of tickers containing {i+1} letters is {round(length/
 ↪len(nyse_tickers)*100,2)}")
```

```
The percentage of tickers containing 1 letters is 0.84
The percentage of tickers containing 2 letters is 6.81
The percentage of tickers containing 3 letters is 63.24
The percentage of tickers containing 4 letters is 15.75
The percentage of tickers containing 5 letters is 0.84
The percentage of tickers containing 6 letters is 1.72
The percentage of tickers containing 7 letters is 9.12
The percentage of tickers containing 8 letters is 1.54
The percentage of tickers containing 9 letters is 0.11
The percentage of tickers containing 10 letters is 0.0
The percentage of tickers containing 11 letters is 0.04
```

2 Q2.) Web Scraping Using Beautiful Soup

2.0.1 Q2.A.) Using Beautiful soup .findAll method you will webscrape the front page of Reddit. Get a list of all of the "timestamps"

[42]:
```
URL = "https://www.reddit.com"
user_agent_list = ["Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
 ↪36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246"
    ,"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like␣
 ↪Gecko) Chrome/47.0.2526.111 Safari/537.36 "
    ,"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9␣
 ↪(KHTML, like Gecko) Version/9.0.2 Safari/601.3.9 "
    ,"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:15.0) Gecko/20100101 Firefox/
 ↪15.0.1"
    ,"Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/537.36 (KHTML, like␣
 ↪Gecko) Chrome/51.0.2704.64 Safari/537.36"]

headers = {'User-Agent': user_agent_list[2]}

page = requests.get(URL, headers = headers)

soup = BeautifulSoup(page.content, "html.parser")

articles = soup.findAll("shreddit-post")
```

```
timestamps = [article.get('created-timestamp') for article in articles if␣
 ↪'created-timestamp' in article.attrs]

print(f"Our timestamp list is {timestamps}")
```

```
Our timestamp list is ['2024-04-29T09:30:53.396000+0000',
'2024-04-29T14:21:07.924000+0000', '2024-04-28T19:42:51.621000+0000']
```

2.0.2 Q2.B.) Using the functions findChild, descendents, etc. locate the post title, text and post time into a dataframe.

Below I write a for loop in which I use the findChild and descendants functions. The findChild function is used to obtain all of the article titles and store these titles in a list called 'titles'. The descdendants function is used to obtain all of the contents of the articles, which are then stored in a list called 'contents'. After completing the 'titles' and 'contents' lists, I add them together wwith the 'timestamps' list to create a dataframe.

[43]:
```
titles=[]
contents=[]

for article in articles:

    title_tag = article.findChild('faceplate-screen-reader-content')

    if title_tag:
        titles.append(title_tag.get_text())
    else:
        titles.append(None)

    content_txt = ''
    for descendant in article.descendants:
        if descendant.name == 'p':
            content_txt += descendant.get_text().strip()+' '

    contents.append(content_txt.strip())

df = pd.DataFrame({'title': titles, 'content':contents, 'timestamp':timestamps})

print(df.head())
```

```
                                               title  \
0                                Patch 01.000.300
1  Decided to fly my NCR flag and my neighbor cho…
2            Some people have zero financial literacy

                                             content  \
0  A subreddit dedicated to HELLDIVERS and HELLDI…
1  The Fallout Network's Subreddit for the Bethes…
2     /r/facepalm - please sir can I have some more?
```

```
                          timestamp
0  2024-04-29T09:30:53.396000+0000
1  2024-04-29T14:21:07.924000+0000
2  2024-04-28T19:42:51.621000+0000
```

[44]:
```
len(timestamps)
```

[44]:
```
3
```

3 Q3.) RegEx

3.0.1 Q3.A.) Using RegEx, get all the urls of ladder faculty profiles for UCLA Economics

[45]:
```
import re

URL = "https://economics.ucla.edu/faculty/ladder"
```

Below I download all of the links from the URL provided and store them in "all_links". After iterating through "all_links", I use RegEx to identify the urls that match the format of ladder faculty profiles ("https://economics.ucla.edu/person/{professor_name}"). Then, I store all of the urls of ladder faculty profiles in a list I call "prof_links".

[46]:
```
link_to_match = "https://economics.ucla.edu/person/"

page = requests.get(URL, headers = headers)

soup = BeautifulSoup(page.content, "html.parser")

all_links = soup.findAll("a", href = True)

prof_links = list(set([link['href'] for link in all_links if re.
 ↪match(link_to_match, link['href'])]))
```

[47]:
```
prof_links[:10]
```

[47]:
```
['https://economics.ucla.edu/person/felipe-goncalves/',
 'https://economics.ucla.edu/person/pablo-fajgelbaum/',
 'https://economics.ucla.edu/person/moritz-meyer-ter-vehn/',
 'https://economics.ucla.edu/person/ariel-burstein/',
 'https://economics.ucla.edu/person/jonathan-vogel/',
 'https://economics.ucla.edu/person/rosa-liliana-matzkin/',
 'https://economics.ucla.edu/person/bernardo-s-silveira/',
 'https://economics.ucla.edu/person/daniel-haanwinckel/',
 'https://economics.ucla.edu/person/alexander-bloedel/',
 'https://economics.ucla.edu/person/jinyong-hahn/']
```

3.0.2 Q3.B.) Webcrawl the links from A and use RegEx to get all the emails and phone numbers of ladder faculty profiles

Below I create a dataframe (which I call df) that will contain all of the available emails and phone numbers of ladder faculty. After completing the dataframe, I print its first ten rows.

[48]:
```
df = pd.DataFrame({'email':[], 'phone':[]})
```

[49]:
```
email_to_match = re.compile(r'[\w\.-]+@econ\.ucla\.edu')
phone_to_match = re.compile(r'\(\d{3}\) \d{3}-\d{4}')
emails=[]
phone_numbers=[]

for no, link  in enumerate(prof_links):

    page = requests.get(link, headers = headers)
    soup = BeautifulSoup(page.content, "html.parser")
    looking_at = soup.findAll("p")

    for content in looking_at:

        content = str(content)

        for match in email_to_match.finditer(content):
            email = match.group()
            if email != 'webmaster@econ.ucla.edu':
                df.at[no,'email'] = email
                break


        for match in phone_to_match.finditer(content):
            phone = match.group()
            df.at[no, 'phone'] = phone
            break
```

[50]:
```
df.head(10)
```

[50]:

	email	phone
0	fgoncalves@econ.ucla.edu	NaN
1	pfajgelbaum@econ.ucla.edu	(310) 794-7241
2	mtv@econ.ucla.edu	NaN
3	arielb@econ.ucla.edu	(310) 206-6732
4	jvogel@econ.ucla.edu	NaN
5	matzkin@econ.ucla.edu	(310) 825-7371
6	silveira@econ.ucla.edu	NaN
7	haanwinckel@econ.ucla.edu	NaN
8	abloedel@econ.ucla.edu	NaN
9	hahn@econ.ucla.edu	NaN

4 Q4.) Selenium

4.0.1 Q4.A.) Pick a website that has useful data to a business or economic question. Put your website you plan to scrape here : https://docs.google.com/spreadsheets/d/1PJ2DOTCVCh51fn0ry1yB7qTyccR33_IXFpkYdc

4.0.2 You must have use website that no other group has. First come first serve

[54]:
```
URL = "https://markets.ft.com/data/indices/tearsheet/historical?s=DJI:DJI"
```

4.0.3 Q4.B.) Use Selenium to scrape valuable information from your website and store in a dataframe.

Below I use Selenium to obtain daily trading data (from Apr 1 to Apr 29) on the Dow Jones Industial Average (DJI).

[95]:
```
import selenium
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
```

[96]:
```
options = webdriver.ChromeOptions()
service = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=service, options=options)
driver.get(URL)
```

I also scroll down the Financial Times Page to obtain all of the data available on the DJI. Then, I create a dataframe containing the Open, High, Low, CLose, and Volume for each trading day from Apr 1st to Apr 29th.

[ ]:
```
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
```

[100]:
```
table = driver.find_element(By.CLASS_NAME,␣
 ↪'mod-ui-table--freeze-pane__container')

table_rows = table.find_elements(By.TAG_NAME, 'tr')[1:]
dates = [row.find_elements(By.TAG_NAME, 'td')[0].text for row in table_rows]
opens = [row.find_elements(By.TAG_NAME, 'td')[1].text for row in table_rows]
highs = [row.find_elements(By.TAG_NAME, 'td')[2].text for row in table_rows]
lows = [row.find_elements(By.TAG_NAME, 'td')[3].text for row in table_rows]
closes = [row.find_elements(By.TAG_NAME, 'td')[4].text for row in table_rows]
volumes = [row.find_elements(By.TAG_NAME, 'td')[5].text for row in table_rows]

dates.reverse()
opens.reverse()
highs.reverse()
```

```
lows.reverse()
closes.reverse()
volumes.reverse()
```

[101]:
```
sel_df = pd.DataFrame({'Date':dates, "Open": opens, 'High': highs, 'Low': lows,␣
 ↪'Close': closes, 'Volume': volumes})

print(sel_df.head())
```

```
                       Date       Open       High        Low      Close  \
0     Monday, April 01, 2024  39,807.93  39,815.00  39,491.22  39,566.85
1    Tuesday, April 02, 2024  39,256.27  39,256.27  39,051.70  39,170.24
2  Wednesday, April 03, 2024  39,139.59  39,305.76  39,017.13  39,127.14
3   Thursday, April 04, 2024  39,343.60  39,421.35  38,559.42  38,596.98
4     Friday, April 05, 2024  38,664.98  39,040.17  38,602.18  38,904.04

        Volume
0  279,113,853
1  325,610,999
2  372,218,331
3  359,292,455
4  322,017,508
```

[ ]:
```
driver.quit()
```

4.0.4 Q4.C.) Write a short paragraph about the businesses or research that would use the data you scraped. Describe it's value and what it can be used for.

The historical trading data for the Dow Jones Industrial Average is a good indicator for investors' confidence in the economy. The data that I scraped may be used by a financial institution to obtain an understanding of how investor sentiment evolved over the month of March. For example, a financial institution could fit a time-series model to my data to make a prediction about the direction of the DJI fot he enxt month. This could help them make investment decisions.