Subject: Web And Media Scraping
Data science and business analytics at the faculty of economic sciences/University of Warsaw

Scraping Yahoo Finance's most active stocks

Group Members

Alparslan Erol / 428681

Anna Sikora / 431354

Tesfahun Tegene Boshe / 432812

Introduction

We have designed mechanisms which scrap the most active stocks of yahoo finance using beautifulsoup, scrapy and selenium separately. The three methods all achieve the same information. The scraped data output is later analysed, enabling us to compare the results.

Our way of working has a perfect collaborative and proper work on GitHub. There is a general project repository and project developed by cloning the forked project in our accounts from the main repository and creating pull requests with descriptive commit messages. Please find the main project repository and track the commit history to observe how this project developed via this link:
https://github.com/AlparslanErol/Stock_Price_Scraping

Please find project requirements for packages inside the **requirements.txt** and for the python version inside the **Readme** file from the main project repository. Dependencies for creating "**virtual environment**" specified into the repository.

Furthermore, there is a **Makefile** created into the repository and thanks to this, processes in below handle automatically:

- Creating virtual environment,
- Installing required packages from requirements.txt file,
- Clean virtual environment.

Scraper Mechanism

1. BeautifulSoup

For the BeautifulSoup part of this project, in addition to other libraries *BeautifulSoup library* was used. Another python library, 'requests' , gets the html of the webpage at url 'https://finance.yahoo.com/most-active', which is later parsed by *BeautifulSoup.*

The webpage has a table with 24 rows and numerous columns. Every row has the symbol attribute, which is later used finding the links to the company's yahoo-finance page.

```
"https://finance.yahoo.com/quote/" + symbol + "?p=" + symbol
```

For every link defined in such a way, first the html copy of the page via the link is requested and then parsed. The variables such as 'Previous Close', 'Open', etc are distributed among two tables: table 1 and table 2. Within their respective tables, every variable is specified by 'data-test' attribute, allowing us to specify and extract a variable of interest one at a time.

The extracted values are appended to already created lists, finally making up a dataframe with the variable names as column headers and the values as a column vector. The dataframe is exported to the csv file ***'stocks_bs.csv***'.

2. Scrapy

For the purpose of scraping Yahoo Finance web page, we also used Scrapy framework. The process could be divided into two parts: first - scraping the links for each of the "Most Active" companies and the second - scraping data for each of the companies.

Links creation happens similarly to the Beautiful Soup and Selenium part of the project. The links are created by adding to a base link the ending specified for each company. To extract them, we used selectors, specifically Xpaths.

The scrapping happens with the use of spiders, the first spider is created for the purpose of extracting the links and the second one to get the data for each company. Spiders are basically a program downloading the content of a given page, they are using a parse function which is called when the URL is crawled and it includes the extraction logic. As not all the information about the companies could be extracted using the same xpath, we used two different xpaths.

The data in our project is stored in csv files. This is specified by the FEED_FORMAT = 'csv'. What is important to note is that, to run the spiders, a **USER_AGENT** = 'Mozilla/5.0 (compatible; Googlebot/2.1; +http://google.com/bot.html)' has to be used.

To run the code provided please follow steps below:

1. In the terminal write: “ scrapy start project Scraping_Stock_Prices (the name can be different)”
2. Paste spiders provided into the spiders folder created by scrapy.
3. In the file “settings.py” change the USER AGENT.
4. In the terminal execute the command: scrapy crawl link_lists -o link_list.csv.
5. Then execute: scrapy crawl stocks -o stocks.csv
6. Take a look at the stocks.csv if the data is scrapped.

3. Selenium

For selenium scraper, a driver for Mozilla Firefox was used. Python script is using command line arguments to get input from the users. There are 3 command lines arguments in the python script.

1. **Flag (--watchlist):** This flag is a boolean variable and not required and its default value is **False**. Aim of this flag is to give opportunity to scrape data from user's own watchlist in yahoo finance website. For this purpose, users should have a favourite list with a "**My Watchlist**" name that stores their favourite companies. Thanks to this list, users are able to monitor stock price information about their favourite companies easily. If users have a list named "My Watchlist" in yahoo finance page, they are able to input this flag as **True**. Thanks to this flag, selenium proceeds the scraping mechanism starting with a login process and going into a favourite list named "My Watchlist". Furthermore, selenium bot open every single company link in this favourite list one by one in a new tab -> scrape data inside the new tab -> save scraped data into a dictionary of lists -> close new tab -> open another company page in a new tab... and this process continue till the last company data scraped and saved into the dictionary. After that, selenium bot terminates itself and scraped data in dictionary converted as pandas dataframe, after that all scraped data is saved into a csv file into the project root directory.
2. **Flag (--email):** This flag is used for watchlist scraping mechanism. --email flag is not required and its default value is **None**. This is another opportunity for users who want to input email addresses before selenium bot start. Thanks to this, users can input their email address safely and get the output faster than input during selenium running.
3. **Flag (--passwd):** This flag is used for watchlist scraping mechanism. --passwd flag is not required and its default value is **None**. This is another opportunity for users who want to input passwords before selenium starts. Thanks to this, users can input their password safely and get the output faster than input during selenium running.

Here is the correct way of flag usage to scrape data from “My Watchlist” list:

*(Please mind, --watchlist, --email and --passed flags are not required to scrape data. If users do not prefer to scrape data from watchlist, the program automatically proceeds the same process to scrape data from **Stocks:Most Active** page under the **Markets** tab)*

```bash

**Regular Scraper for Stocks: Most Active page.**

…. : ~<project_root>$ python3 selenium_scraper.py

**Optional Scraper for favourite list named “My Watchlist”**

…. : ~<project_root>$ python3 selenium_scraper.py --watchlist True --email “abc.def@uw.edu.pl” --passwd “abcdef123.”

```

Definition of Functions (with ordered):

main: Selenium scraper is triggering from the main function by setting up webdriver, gecko path and url.

start_scrape: This function is used to start selenium bot.

cookie_handler: While selenium bot is trying to reach the website, a popup message appears to give consent for cookies in the website. This function used to handle this cookie message by clicking on the “accept all cookies in this page” button.

watchlist_scrape (if --watchlist is True): This function is triggering the watchlist scraper mechanism by:

- Click on the “**sign in**” button,
- Input “**email address**”,
- Input “**password**”,
- Click on the “**My Watchlist**” button.

most_active_scrape (if --watchlist is False): This function is triggering the **Stocks: Most Actives** mechanism to scrape data in this most active page by:

- Click on the "**Markets**" button,
- Click on the "**Stocks: Most Actives**" button.

new_window_scrape: This function is used to scrape data for every single company page by:

- Open company pages in the list into the new tab,
- Scrape data inside the table,
- Save scraped data into dictionary of lists
- Close the new tab,
- Scrape all data in a loop like described above,
- After all data is scraped into the dictionary, convert this to pandas dataframe and save data into a **csv** file (name of csv file created based on the users' prefer, if --watchlist is True, name of the file is **selenium_scrape_watchlist.csv,** else: **selenium_scrape_most_active.csv**).
- Terminate the selenium session.

(Please mind, if users prefer a different name for a favourite company list name, they can only hardcode the name of the list into a python script. Secondly, users are able to optimize the sleep times based on their computer capacity. Because, response times may vary for different computers. So, if you have no output from the Selenium scraper, please update the sleep times inside the python script by hand.)

Technical Description of Output

For all "**Beautiful Soup**", "**Scrapy**" and "**Selenium**" scrapers, scraped data from the yahoo finance page is stored in a **csv** file. Schema for all scraper output is exactly the same in each other. There are 16 columns in the output file. Since the table that we scrape data has at most 25 companies, we skip the statement "*At the beginning of your code you set up a boolean parameter that if it is True, it will limit the number of pages you scrap to 100. Set default as True.*" in the project requirement.

Please find the Table 1 in **Stocks: Most Actives** page and a Table 2 for scraped data from every single company in Table 1 (or **My Watchlist**, if --watchlist is True for selenium) below.

Symbol	Name	Price (Intraday)	Change	% Change	Volume	Avg Vol (3 month)	Market Cap	PE Ratio (TTM)	52 Week Range
OCGN	Ocugen, Inc.	12.67	+0.71	+5.94%	164.401M	80.446M	2.384B	N/A	0.17 18.77
NIO	NIO Inc.	39.84	+0.85	+2.18%	115.673M	94.238M	65.279B	N/A	3.08 66.99
GNW	Genworth Financial, Inc.	4.3200	-0.0600	-1.37%	58.394M	9.449M	2.188B	5.14	1.87 4.76
AAPL	Apple Inc.	131.46	-2.02	-1.51%	109.839M	100.992M	2.207T	29.55	71.58 145.09
F	Ford Motor Company	11.54	+0.28	+2.49%	89.511M	69.517M	46.058B	11.61	4.52 13.62
TWTR	Twitter, Inc.	55.22	-9.87	-15.16%	86.872M	21.477M	44.103B	N/A	27.12 80.75
MVIS	MicroVision, Inc.	15.25	-2.75	-15.28%	74.013M	26.628M	2.409B	N/A	0.55 28.00
NOK	Nokia Corporation	4.6800	+0.0100	+0.21%	66.526M	50.877M	27.13B	N/A	3.21 9.79
GE	General Electric Company	13.14	-0.07	-0.53%	49.9M	76.786M	115.351B	N/A	5.48 14.42
AMD	Advanced Micro Devices, Inc.	81.62	-2.29	-2.73%	45.915M	43.339M	99.17B	34.27	48.42 99.23
TSLA	Tesla, Inc.	709.44	+32.44	+4.79%	40.759M	34.168M	683.425B	710.86	139.60 900.40
BAC	Bank of America Corporation	40.57	-0.54	-1.31%	39.209M	53.512M	347.657B	17.41	20.10 41.13
VALE	Vale S.A.	20.12	-0.94	-4.46%	35.609M	28.445M	103.232B	9.98	7.69 21.12
WFC	Wells Fargo & Company	45.05	-1.19	-2.57%	34.153M	33.03M	186.557B	30.94	20.76 46.39
ITUB	Itaú Unibanco Holding S.A.	5.0000	-0.0600	-1.19%	33.455M	41.964M	47.394B	13.85	3.48 6.37
T	AT&T Inc.	31.41	+0.07	+0.22%	31.307M	41.364M	224.267B	N/A	26.35 33.24
MSFT	Microsoft Corporation	252.18	-0.33	-0.13%	30.945M	28.231M	1.899T	34.37	173.80 263.19
INTC	Intel Corporation	57.53	-0.75	-1.29%	30.732M	30.796M	232.306B	12.91	43.61 68.49
XOM	Exxon Mobil Corporation	57.24	-1.70	-2.88%	30.483M	28.519M	242.327B	N/A	31.11 62.55

Table 1: Yahoo finance stock prices for most active data table that scraped in this project.

Previous Close	39.84	Market Cap	64.787B
Open	40.34	Beta (5Y Monthly)	2.61
Bid	39.40 x 1100	PE Ratio (TTM)	N/A
Ask	39.41 x 800	EPS (TTM)	-1.03
Day's Range	39.31 - 41.45	Earnings Date	Apr 28, 2021
52 Week Range	3.18 - 66.99	Forward Dividend & Yield	N/A (N/A)
Volume	79,665,513	Ex-Dividend Date	N/A
Avg. Volume	94,940,171	1y Target Est	53.50

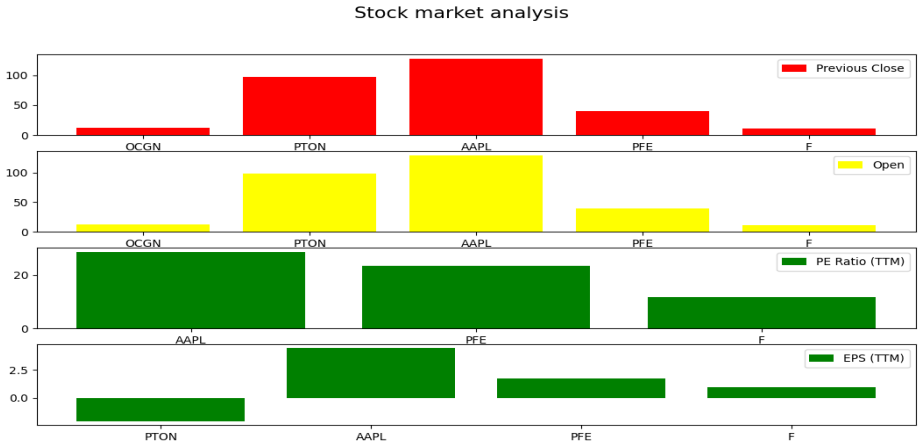
Table 2: Sample table that data is scraped for BS, Scrapy and Selenium.

Company names and the values inside the features in the output file may be different for all scrapers, because output depends on:

- The time when scraper scripts run and generate the output file,
- The domain (may different county domains used),
- Values inside the table that scrapers scrape data (if values have different type or tag/class name inside the html, Scrapy may bypass this company).

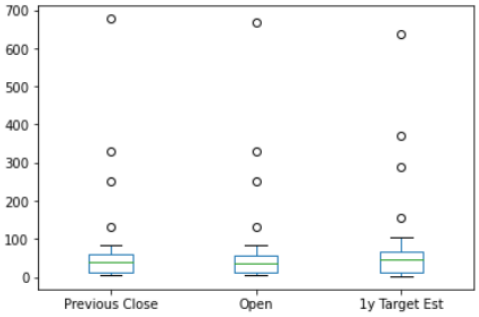
As a result of the points above, it is not guaranteed that every single output file generated from scrapers will be the same because yahoo finance is a dynamic web site. But it is guaranteed that every single output file generated from the scrapers have exactly the same output schema based on the features that mentioned above. There is a control mechanism in Scrapy which allows us to bypass companies that scrape wrong data in table. All scrapers scrape the same information but not the same values in the rows. So this situation is not contradicted by the project goal that “All scrapers should scrape the same information from the domain of your choice”.

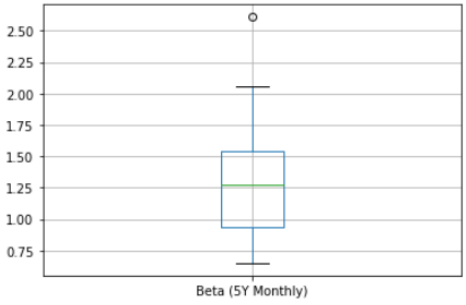
Elementary Data Analysis




```
1 data.info()
```

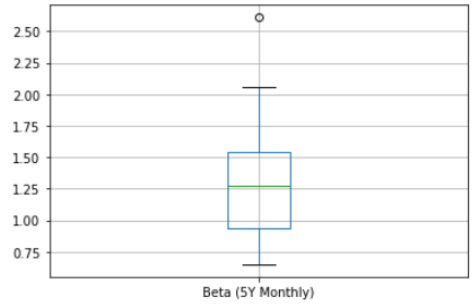
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Symbols                   25 non-null     object
 1   Names                     25 non-null     object
 2   Previous Close            25 non-null     float64
 3   Open                      25 non-null     float64
 4   Bid                       25 non-null     object
 5   Ask                       25 non-null     object
 6   Day's Range               25 non-null     object
 7   52 Week Range             25 non-null     object
 8   Volume                    25 non-null     object
 9   Avg. Volume               25 non-null     object
 10  Market Cap                25 non-null     object
 11  Beta (5Y Monthly)         22 non-null     float64
 12  PE Ratio (TTM)            16 non-null     float64
 13  EPS (TTM)                 24 non-null     float64
 14  Earnings Date             20 non-null     object
 15  Forward Dividend & Yield  25 non-null     object
 16  Ex-Dividend Date          18 non-null     object
 17  1y Target Est             24 non-null     float64
dtypes: float64(6), object(12)
memory usage: 3.6+ KB
```






```
1 data.describe()
```

	Previous Close	Open	Beta (5Y Monthly)	PE Ratio (TTM)	EPS (TTM)	1y Target Est
count	25.000000	25.000000	22.000000	16.000000	24.000000	24.000000
mean	80.968000	79.660800	1.318182	65.453750	0.757625	89.753333
std	146.421309	144.898336	0.507971	172.436334	4.625564	146.887844
min	4.380000	4.310000	0.650000	5.140000	-14.730000	2.500000
25%	11.960000	11.450000	0.937500	13.615000	-0.453250	12.647500
50%	38.600000	37.730000	1.270000	25.240000	0.915000	45.410000
75%	58.940000	57.610000	1.540000	31.772500	2.127500	66.995000
max	677.000000	667.590000	2.610000	710.860000	11.670000	636.350000



Labour Division

Task	**Contributors**
Beautiful Soup	Tesfahun
Scrapy	Anna
Selenium	Alparslan
Github	Tesfahun, Anna, Alparslan
Project Report	Tesfahun, Anna, Alparslan