2024

Web Scraping Financial Data and Beautiful Soup: for Investment Decisions

A Project for Data Science

Anesu David Pasipanodya

A PROJECT FOR WEB SCRAPING IN DATA ANALYSIS

ANESU DAVID PASIPANODYA

1. Project Outline

Real Estate Investment Dashboard: Analyzing Market Data for Informed Property Investments

In this project, I took on the role of a Data Scientist/Analyst to Extract business data from the Netflix website.

To accomplish this, I used Python for data extraction and ~~data visualization~~analysis, incorporating web scraping and ~~financial~~ data ~~manipulation~~ libraries. I also leveraged ~~tools such as Skills Network Labs and IBM Watson Studio, which provided a~~ cloud-based environment ~~platforms like Google Colab and AWS~~ to handle data ~~processing~~ and ~~develop the~~ project~~. The key focus was on displaying financial performance indicators clearly and interactively, using~~ development. The dashboard's visualizations are powered by Plotly for rich visualization.~~and Streamlit, providing a user-friendly interface that enables investors to explore financial indicators and trends effectively.~~ The IDE ~~I chose for use was the PyCharm community edition. https://www.jetbrains.com/help/pycharm/quick-start-guide.html~~

~~What is a DataFrame?~~

~~A dataframe is a~~chosen for this project was Visual Studio Code, which offered a versatile environment for both data ~~structure constructed with rows and columns, similar to a database or Excel spreadsheet. It consists of a dictionary of lists in which the list each have their own identifiers or keys, such as "last name" or "food group. https://miamioh.edu/centers-institutes/center-for-analytics-data-science/students/coding-tutorials/python/pandas-dataframes.html#:~:text=A%20dataframe%20is%20a%20data,%E2%80%9D%20or%20%E2%80%9Cfood%20group.%E2%80%9D~~ analysis and dashboard deployment.

Using Webscraping to Extract Stock Data Example

~~Using Webscraping to Extract Stock Data Example~~

I extracted Netflix stock data from here: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/netflix_data_webpage.html.

In this example, we are using yahoo finance website and looking to extract Netflix data.

On the following web page there is a table with columns name (Date, Open, High, Low, close, adj close volume) out of which we must extract following columns

- Date
- Open
- High
- Low
- Close
- Volume

On the following web page we have a table with columns name (Date, Open, High, Low, close, adj close volume) out of which we must extract following columns

- Date
- Open
- High
- Low
- Close
- Volume

Steps for extracting the data

1. Send an HTTP request to the web page using the requests library.
2. Parse the HTML content of the web page using BeautifulSoup.
3. Identify the HTML tags that contain the data you want to extract.
4. Use BeautifulSoup methods to extract the data from the HTML tags.
5. Print the extracted data

Step 1: Send an HTTP request to the web page

Using the attribute info I extracted information about the stock as a Python dictionary.

Code

```
url = https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/netflix_data_webpage.html
```

```
data  = requests.get(url).text
print(data)
```

Result:

```
<!DOCTYPE html><html id="atomic" class="NoJs chrome desktop" lang="en-US"><head prefix="og: http://ogp.me/ns#"><script>window.performance && window.performance.mark && window.performance.mark('PageStart');</script><meta charset="utf-8"/><title>Netflix, Inc. (NFLX) Stock Historical Prices &amp;amp; Data - Yahoo Finance</title><meta name="keywords" content="NFLX, Netflix, Inc., NFLX historical prices, Netflix, Inc. historical prices, historical prices, stocks, quotes, finance"/><meta http-equiv="x-dns-prefetch-control" content="on"/><meta property="twitter:dnt" content="on"/><meta property="fb:app_id" content="458584288257241"/><meta name="theme-color"…..
typeof w.ready === 'function' && w.ready(function () {
        …..   typeof w.on === 'function' && w.on('tab:selected', function (e) {
              try {
                 if (e && e.meta && e.meta.targetElem.id === 'header-notification-menu') {
```

```
                    window.setTimeout(function hideBadge() {w.base.state =
{financeNotification:{hideBadge:'1'}};}, 250);
                    var rapidEvent = w.base.state.financeNotification.i13n.showPanel;
                    window.rapidInstance.beaconEvent(rapidEvent.event, rapidEvent.data,
rapidEvent.outcm);
                }
            } catch (ignore) {}
        });
    }, window);
})();</script>
<script>window.webpackPublicPath='https://s.yimg.com/uc/finance/dd-
site/js/';</script></body></html>
117 Pages of content.
```

What is parsing?

In simple words, parsing refers to the process of analyzing a string of text or a data structure, usually following a set of rules or grammar, to understand its structure and meaning. Parsing involves breaking down a piece of text or data into its individual components or elements, and then analyzing those components to extract the desired information or to understand their relationships and meanings.

Next I took the raw HTML content of a web page or a string of HTML code which needs to be parsed and transformed into a structured, hierarchical format that can be more easily analyzed and manipulated in Python. This can be done using a Python library called Beautiful Soup.

Parsing the data using the BeautifulSoup library
Create a new BeautifulSoup object.

Note: To create a BeautifulSoup object in Python, I passed two arguments to its constructor:

1. The HTML or XML content that you want to parse as a string.
2. The name of the parser that I will to use to parse the HTML or XML content. This argument is optional, and if we don't specify a parser, BeautifulSoup will use the default HTML parser included with the library. here in this lab we are using "html5lib" parser.

Code

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

Step 3: Identify the HTML tags

I scraped the content of the HTML web page and convert the table into a data frame.

I created an empty data frame using the **pd.DataFrame()** function with the following columns:

"Date"

"Open"

"High"

"Low"

"Close"

"Volume"

Code

```
#Step 3: Identify the HTML tags
netflix_data = pd.DataFrame(columns=["Date", "Open", "High", "Low",
"Close", "Volume"])
Print(Netflix_data)
```

Result

```
Empty DataFrame
Columns: [Date, Open, High, Low, Close, Volume]
Index: []
```

Working with HTML table

These are the following tags which were used while creating HTML tables.

<table>: This tag is a root tag used to define the start and end of the table. All the content of the table is enclosed within these tags.

<tr>: This tag is used to define a table row. Each row of the table is defined within this tag.

<td>: This tag is used to define a table cell. Each cell of the table is defined within this tag. You can specify the content of the cell between the opening and closing tags.

<th>: This tag is used to define a header cell in the table. The header cell is used to describe the contents of a column or row. By default, the text inside a tag is bold and centered.

<tbody>: This is the main content of the table, which is defined using the tag. It contains one or more rows of elements.

- I used find() and find_all() methods of the BeautifulSoup object to locate the table body and table row respectively in the HTML. The find() method will return particular tag content. The find_all() method returns a list of all matching tags in the HTML

Code

```
# First we isolate the body of the table which contains all the
information
# Then we loop through each row and find all the column values for each
row
for row in soup.find("tbody").find_all('tr'):
    col = row.find_all("td")
    date = col[0].text
    Open = col[1].text
    high = col[2].text
    low = col[3].text
```

```
    close = col[4].text
    adj_close = col[5].text
    volume = col[6].text

    # Finally I appended the data of each row to the table
    netflix_data = pd.concat([netflix_data, pd.DataFrame(
        {"Date": [date], "Open": [Open], "High": [high], "Low": [low],
"Close": [close], "Adj Close": [adj_close],
         "Volume": [volume]})], ignore_index=True)
```

Result

Step 5: Print the extracted data

We can now print out the data frame using the head() or tail() function.

Code

```
#Step 5: Print the extracted data
print(netflix_data.head())
```

Result

	Date	Open	High	Low	Close	Volume	Adj Close
0	Jun 01, 2021	504.01	536.13	482.14	528.21	78,560,600	528.21
1	May 01, 2021	512.65	518.95	478.54	502.81	66,927,600	502.81
2	Apr 01, 2021	529.93	563.56	499.00	513.47	111,573,300	513.47
3	Mar 01, 2021	545.57	556.99	492.85	521.66	90,183,900	521.66
4	Feb 01, 2021	536.79	566.65	518.28	538.85	61,902,300	538.85

Extracting data using pandas library

We can also use the pandas read_html function from the pandas library and use the URL for extracting data. pd.read_html(url) is a function provided by the pandas library in Python that is used to extract tables from HTML web pages. It takes in a URL as input and returns a list of all the tables found on the web page.

Code

```
read_html_pandas_data = pd.read_html(url)
print(read_html_pandas_data)
netflix_dataframe = read_html_pandas_data[0]
print(netflix_dataframe)
netflix_dataframe.head()
```

Output

[	Date	...	Volume
0	Jun 01, 2021	...	78560600
1	May 01, 2021	...	66927600
2	Apr 01, 2021	...	111573300
3	Mar 01, 2021	...	90183900
4	Feb 01, 2021	...	61902300
..	...	...	...
66	Dec 01, 2015	...	319939200
67	Nov 01, 2015	...	320321800
68	Oct 01, 2015	...	446204400
69	Sep 01, 2015	...	497401200

70 *Close price adjusted for splits.**Adjusted cl... ... *Close price adjusted for splits.**Adjusted cl...
[71 rows x 7 columns]]

	Date	...	Volume
0	Jun 01, 2021	...	78560600
1	May 01, 2021	...	66927600
2	Apr 01, 2021	...	111573300
3	Mar 01, 2021	...	90183900
4	Feb 01, 2021	...	61902300
..	...	...	...
66	Dec 01, 2015	...	319939200
67	Nov 01, 2015	...	320321800
68	Oct 01, 2015	...	446204400
69	Sep 01, 2015	...	497401200

70 *Close price adjusted for splits.**Adjusted cl... ... *Close price adjusted for splits.**Adjusted cl...

[71 rows x 7 columns]