Web Scraping Project Description

Participants:

Jan Frąckowiak, **405866**
Anirban Das, **454449**
Rahila Mammadova-Huseynova, **454075**

Project topic & scraped website:

Rottentomatoes is an online movie-rating platform which displays lists of rated movies in numerous genres and determines its custom movie quality measure called “tomatometer”.

In our project we have scraped tomatometer values for top **n** (for the purpose of demonstration **n** = 100) movies listed in the category “best-netflix-movies-to-watch-right-now”. The score was scraped together with movie’s title,, year of release and link to Rottentomatoes webpage stroring all information about respective production.

Here is the link to the ranking webpage which served as a starting point for all our scrapers https://editorial.rottentomatoes.com/guide/best-netflix-movies-to-watch-right-now/.

Descriptions of scrapers (in brackets: name of the sole responsible participant):

1. BeautifulSoup Scraper (Rahila Mammadova-Huseynova):

Web scraping was performed using BeautifulSoup and other libraries to gather information about the top 100 movies on Netflix from Rotten Tomatoes' webpage. The script sends a GET request to the webpage URL and retrieves its content. BeautifulSoup with the lxml parser is used to parse the HTML content and extract movie links. The links are obtained from <a> tags with the class "article_movie_poster". An empty DataFrame is created to store movie information, including links, title, rating percentage, and year. Iterating through each movie link, the script sends another GET request to retrieve the movie page content. BeautifulSoup is used again to extract the movie title, rating percentage, and release year from the HTML. The movie information is appended to the DataFrame. Finally, the DataFrame is saved as a CSV file for further analysis or usage.
Beautiful Soup Running time: (102,34 s.)

2. Selenium Scraper (Jan Frąckowiak):

In the first step program is initialising a firefox webdriver with a gecko path and option headless set to **True** for the sake of performance. Than it gets to the ranking website, handles cookies button and (if the parameter **“Scrape_100”** is set to **True**) gathers 100 links to movie pages from elements classified as **“article_movie_poster”**. To extract the links attribute href and a relevant regex expression is utilised.

After gathering the links the program runs a for loop which iterates over the links, finds the elements containing movie title, year of release and tomatometer score and appends this information in a form of dictionary into a list. The list is later transformed to a DataFrame and saved in CSV format.

Scraper Running time: (212 s. - 280 s.)

Extra Detail:

I have also built a YahooFinance twitter profile scraper with somewhat similar mechanics utilising the ability of Selenium to scroll through the website. It was our first abandoned approach to webscraping project, which unfortunatelywould not allow for making comparisons between scrapers. (More details in the recorded presentation & Github! **:)**)

3. Scrapy Scraper (Anirban Das):

The above mentioned link for rottentomatoes is scraped.
The main mechanism is that the first page consists of the names of the top 100 movies and links to sub-page with more details. In this part of the task, *scrapy* is used to scrape the website. Coming to the program, under the *NetflixSpider*, the set spider name is *“**t1_spider**”* which is very crucial to get the required output. The domain "*rottentomatoes.com*" is set and the url is strictly put forward.

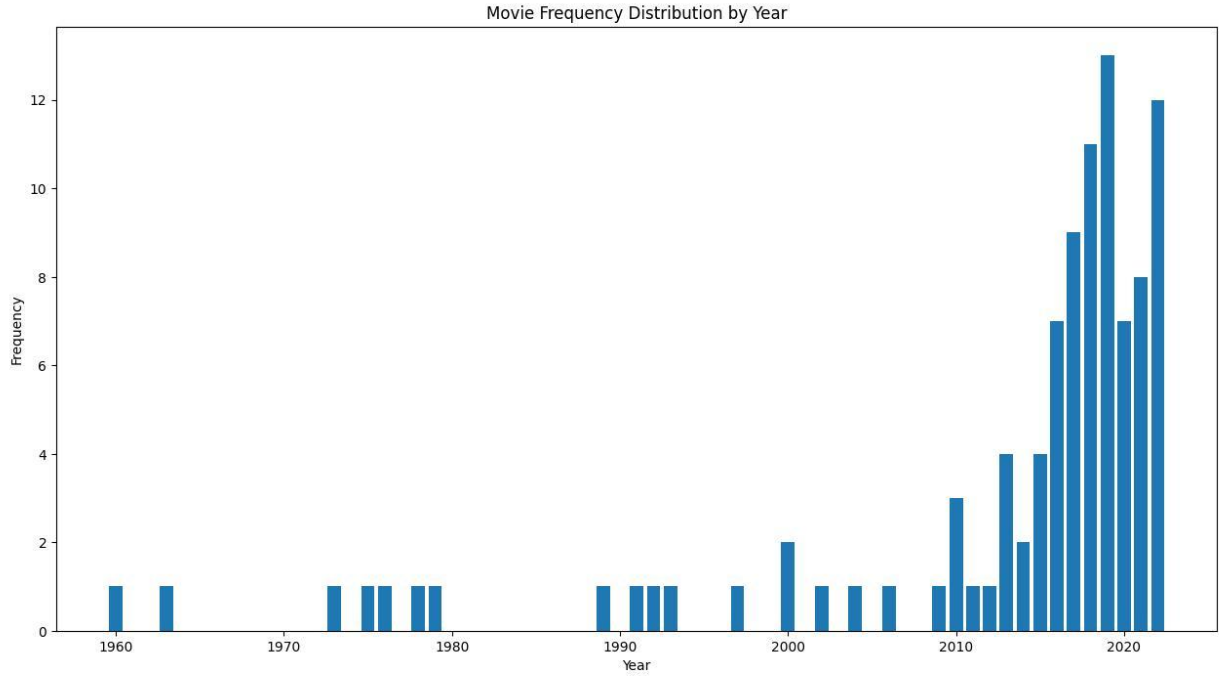
Using the ***parse*** method, the Xpath of the response from *start_url* is accessed. Here an empty list is created and then it is appended with urls of the movies by running a for loop which iterates 100+ times over the XPath to get 100+ urls. This url is set with a limit of 100, which is further requested and responses are returned to the ***parse_movie*** method by *yield* statement.

At ***parse_movie*** the responses are handled to extract the name, year, and ratings of the movies from each of the links and then they are extracted by the yield statement.
In the mean time, the XPath of ‘year’ variable was providing a string with release year separated by ‘,’ and it is splitted to target only the year of release in the output.
To get the output as csv format, please run the command: ***scrapy crawl t1_spider -O outputname.csv***

Scrapy Runtime: (7-11 seconds)

Output & Analysis :

After extracting the output a simple Exploratory data analysis was performed.
The main goal was to check which decade provided the most popular movies and their frequency distribution.
After running the code, it was found that movies released around the late 2010s and early 2020s had the highest frequency of popularity.



There can be many possible reasons for the popularity of movies released this period:

- **Movie Availability**: Netflix alway do not have all the movies available in their platform, so viewer's choice is limited.
- **Viewer Preference**: Movies released during this period has the viewership mostly from Generation Z, preferring different aspects of movies and also mostly binge watching the next best movie on Netflix.
- **Viewer Experience**: Due to huge changes in movie development and productions (IMAX Cameras, better sound quality), viewers might prefer the better experience apart from the story line.
- **Directors**: With future analysis, Directors can also be scrapped to verify the popularity of the specified individuals on their films.

Concluding the analysis, it provides a very overview of the 100 movies and hypotheses developed from their popularity.