Web Scraping Project Report

Fikrah Elhifzi Harahap 437961,
Nomin Batbayar 442627,
Syed Ahmad Jamal 435056

May 14, 2022

1 Short description of The Webpage

The webpage which we used to scrape is “commonsensemedia.org". This website is used to find reviews and ratings of movies, tv shows, apps, and games for parents before they let their kids watch or use them. These reviews are made by a professional team of the website. We scraped the rating, reviews, titles, year released, platforms, and description of the games in the game review section of the website.

2 Mechanical description of the approaches

In order to scrap the desired output from the webpage, we used Beautiful Soup, Scrapy, and Selenium approaches. In this section, we described what we did with each approach. For all approaches, we gathered the same outputs. Then in next section we made some basic analysis and compared the three approach’s performances by how long each approach takes to scrap the first 100 pages.

2.1 Beautiful Soup

Basically, we are using BeautifulSoup to scrape the information from the website. In this case, we scrape 100 hundred pages. To access the website, it uses request library. In addition to clean and match the pattern in some string, it requires regular expression (RE) library. Technically, we access some information that we need with looking for the class in the inspection. After we find use BeautifulSoup we do some looping to extract the information. In this case it is the title, age, platform, description and rating. After we do this, we clean some information using regular expression. As the function can run properly. We make looping and test for 100 pages and when it stores, the final result will be in CSV format.

2.2 Scrapy

The scraper creates an excel file which has 6 columns. Then it proceeds and runs the spider class where we define the domain and the list of pages to be scraped in the start_urls variable. A “for loop” is set which has a condition of 100 pages if “page_limit ” variable is True and it appends the links of pages to start_urls by adding numbers in the page links defined by the range of loop.

Then we define the function parse which is used to extract relevant data from the website by defining X paths. We extract the game name, game year, and game description by calling it’s

Xpath. Then we extract game ratings and remove the spaces in the text. For extracting the rating score we extract all data from "rating_score" class and then count the number of active stars and append it to the list. For extracting the game platform we first link to 20 instances of the 1st span tag in "review-product-summary" class and then for each instance we go deep into 'a' tag and extract all platforms for each game. In the end we add all the outputs to a dictionary and transform it into a panda data frame. We then append it to the already created csv file.

The scraper first goes to the first link in start_urls and collects all the data defined in the parse function and writes it to csv file then it goes again into the start_urls list and extract data from the second link in the list and so on.

2.3 Selenium

In Selenium, we can automate the web browser. So instead of gathering the page links, we decided to write a loop that automatically clicks the next page link and scrapes the information from each page. Thus, this approach is a lot slower than the previous 2 methods.

First, We import the necessary packages, set the driver path, and define the starting page URL. Then we create an empty list for every information we are going to scrap. We create a loop that goes from the starting page to the 100th page. In the loop, we searched for the elements by XPATH and appended the results into our empty list. We scroll down to get where the next page link locates, and then click on the link. These processes will be repeated until the first 100 pages are scraped.

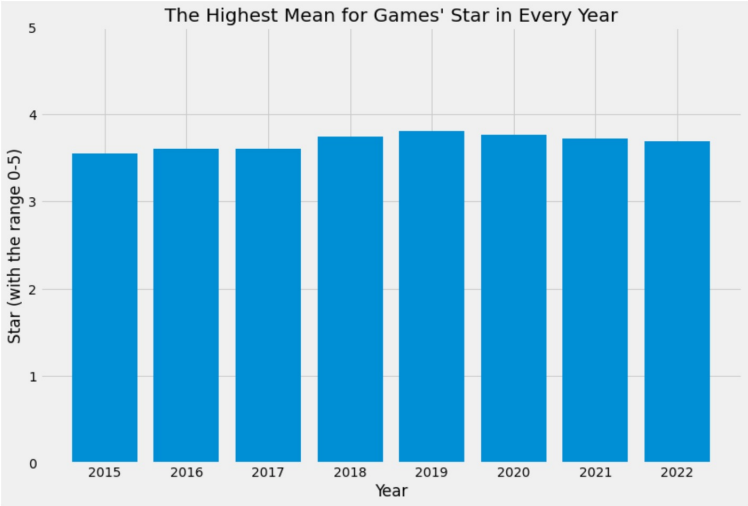
3 Analysis on the output

We append the data to the list every time it runs for the next link. Then save the data in a CSV file. In the table below, you can see the description of the output.

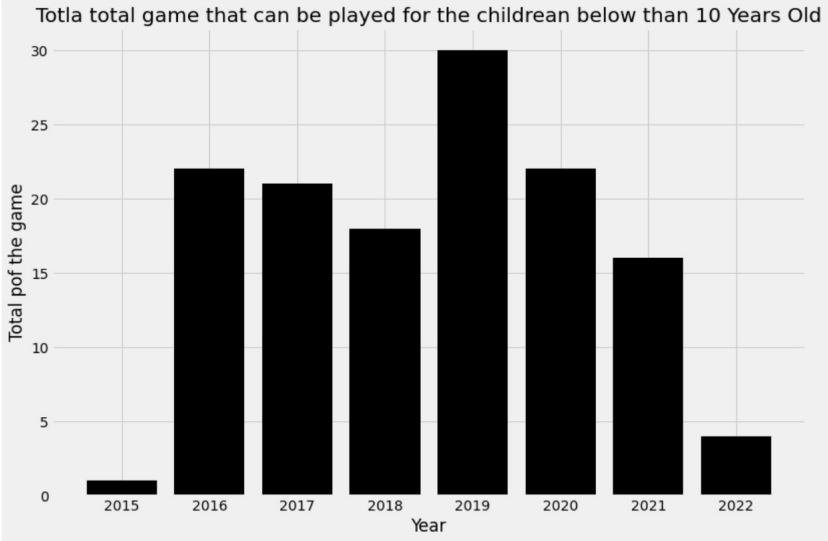
Output	**Description**
Game_name	Name of the game
Age_rating	Advised player age
Rating	Rating score
Description	Short description of the game
Platform	In which platform, the game is compatible with
Year	Released year of the gmae

3.1 Statistical analysis

Here, we can see the best rating games occurred in 2019, which is near to 4 stars rating. Basically, there is no big gap between ratings every year, meaning it tends to be in the range of 3.5 – 4.



In this statistical analysis, we look for information about how many total best games are based on ratings for children below ten years old. As the result, 2019 is the highest number of total games with a full rating for those children, while 2015 has the lowest number of games with the best rating for kids below 10 years old.



3.2 Performance analysis

As we make the script with different methods and library, we conclude the arrange time in different scrapping technique:

- BeautifulSoup : 103 – 105 in second for 100 pages
- Scrapy : 4 – 5 in second for 100 pages
- Selenium : 442 – 445 in second for 100 pages.

We can conclude, in our project, that the fastest script is when we are using Scrapy, following BeautifulSoup and Selenium respectively.