1. Team members

Joanna Banasiak, index: 308502
Aleksandra Biłas, index: 308520

2. Topic and website description

Topic: Scraping glassdoor.com for Amazon reviews

Glassdoor is a webpage where current or former employees may post their opinions about companies, salaries they get and positions they have. Such a place contains very helpful information about:

- Employees who are searching for a new job – so that they may know what to expect and how to negotiate with a potential employer.
- Companies/employers – so that they may see what employees think about them.
- Companies that are partners with Glassdoor – so that they may get information about jobs postings and use it to their own purposes.

The result of our scrapers will be Pros and Cons of working in Amazon, along with Review ID as an identifier.

3. Scrapers mechanism description

3.1. BeautifulSoup

(For the purpose of logging in, a fake email account was created and provided along with a password.)

Since in case of Glassdoor page number is included in its URL, we will be iterating over pages with simple function:

```
for y in range(page_start, page_end):
url="https://www.glassdoor.com/Reviews/Amazon-Reviews-E6036_P"+str(y)+".htm?filter.iso3Language=eng"
```

For each of pages we will be scraping for 3 items (*review_id, pros* and *cons*) so at the beginning we create empty lists for them. Then we scrape the information with use of BS function – *find_all()* and specified attributes. In next steps the information is transformed and appended to empty lists.
The transformation consists of:

- Replacement of specific characters in Pros and Cons (we don't want to have there semicolons because we would like to save it later to csv with semicolon as delimiter, we also replace other characters just to make it prettier and easier to read).
- Using regex pattern to get *review_id* (unfortunately this one was a bit more sophisticated to get).
- Adding word 'null' for those of Pros and Cons that were empty.

Then all lists are combined together to Pandas dataframe with use of defined *Dict*.
The same script is being run for boolean parameter equal to False. Additionally when it reaches the last page and after it the empty *review_id* list is returned, the script stops.
The results are written to CSV outside the loop.

3.2. Scrapy

(For the purpose of logging in, a fake email account was created and provided along with a password.)

There is only one spider prepared for scraping with Scrapy since the Url for next Glassdoor pages is easy to identify. At the beginning of the script there is a boolean parameter for limitation of scraped pages. List of links to scrape is defined at the very beginning of loop, then it is transformed to Pandas data frame and written to CSV file. The CSV file is later opened in *GetReviews* class, starting from the second line (we are omitting the header). Then we create 3 empty lists, define XPaths for each of 3 elements and define 3 extracts with use of XPaths. In next steps the information is transformed and appended to empty lists.
The transformation consists of:

- Replacement of specific characters in Pros and Cons (we don't want to have there semicolons because we would like to save it later to csv with semicolon as delimiter, we also replace other characters just to make it prettier and easier to read).
- For *review_id* we get only characters from 49 to 72 and then extract the *review_id* number from between "".
- Adding word 'null' for those of Pros and Cons that were empty.

Then all lists are combined together to Pandas data frame. The data frame is written to csv in append mode, without creating the header each time the data is appended.
If the boolean parameter is set to *False*, we will be iterating with itertools library and the script will stop after last page.

3.3. Selenium

The part of the project performed using Selenium is meant to simulate user's actions on the web page. After the browser drivers opens, an action of clicking on Accept Cookies button is done, otherwise the bar may appear all the time and interrupt the script (e.g. may cause the "Element is not clickable at point (x,y)" error). Next, the user must sign in, because in other case it is not possible to scrape all pages – a limitation is imposed by the website as an incentive to sign up to get more viewing possibilities. For the purpose of logging in, a fake email account was created and provided along with a password. The credentials serve as an input in a terminal when a program asks to provide them.

One more thing worth attention is a directive for limiting number of pages to be scrapped (*limit_pages* parameter at the beginning of the script). Its default value equals *True* as per project requirements, which limits the pages to 100. If one wants to scrape all pages the parameter needs to be switched to *False*.

When the user is logged in, the main program starts. The first step is checking a value of *limit_pages* in order to then run appropriate part of the script. The difference between those parts is a while loop which iterates over all pages if the parameter is set to *False*, or iterates through 100 (on default) pages otherwise. After this step, the program looks for all review IDs, pros and cons in the page source and saves it to a list which is further iterated in order to retrieve individual reviews, pros and cons. Each new review ID, pro and con is cleaned using regex (removing \n, \r, and changing ";" into nulls due to the usage of ";" as a separator in *to_csv()* function), and appended to a general list of all values which will be later inserted into a data frame.

When all required pages are scraped, a dictionary containing all retrieved lists of values is created, and then inserted into a data frame. The data frame is also saved to .csv file in the same location as the script.

IMPORTANT NOTE: Since reviews on Glassdoor may occur every second, it is possible that when running our scrapers at different time the results will not be exactly the same. For instance, when an employee posts a review between BeautifulSoup scraper and Scrapy scraper executions, then It will be included only in Scrapy scraper. There is also a risk that one review will be pushed to the next page and will occur twice in the result data frame (as the last review from page No9 and first review from page No10). Such a situation also depends on time.sleep() used to slow down our scrapers – the higher the number of seconds in time.sleep() the grater the chances for repeated reviews.

4. Output description

Each scraper produces an output in a form of data frame and csv file, containing 3 columns:

- review_id – a review identifier,
- pros – pros description written by a website user,
- cons – cons description written by a website user.

5. Output data analysis

5.1. Basic statistics

The data collected from Glassdoor may meant to be used in text mining analyses, e.g. for sentiment analysis in general or for different roles (assuming adding extra dimensions to the data frame), gathering the most frequently used words, etc. In this analysis, we have prepared a basic data quality check in order to assess quantitative side of the variables. It is worth noting that the Amazon web page on Glassdoor has over 6800 pages, so the analysis was done on a limited pages number, i.e. 100.

Script “data_analysis.py” (attached at the end of this chapter)

The first step in the analysis is reading the data from csv files and removing duplicated records. The duplicates occur because while the scripts were running, users were placing new reviews on the website. Thus, the results of each script execution are not the same due to that – it is expected and correct.

When the data is deduplicated, we can check the basic statistics which are: variable type, count of null values, percentage of null values, count of non-missing values, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, maximum. The statistics are computed using a custom function *describe_features,* which is executed 3 times, once for each output. The function calculates length of each pro and con, and for them calculates descriptive statistics, because in this analysis the text as such is not that important. Below we have placed an exemplary summary for Selenium scraper output.

variable	types	count	mean	std	min	25%	50%	75%	max	count_null	perc_null
review_id	int64	906	45488250	3799933	3431779	45846232	46190049	46461427	46676847	0	0
pro_len	int64	906	105	229	16	36	53	102	5893	0	0

con_len	int64	906	167	287	9	37	58	154	4496	0	0

Selenium script resulted in obtaining 906 unique reviews, pros and cons, among which there are no missing values (*count_null* and *perc_null*), so we can say that usually an employee puts both pros and cons, without skipping one of them. On the other hand, one can say that at least a half of pros and cons is one-sentence review (assuming that a sentence has 50 signs on average). Also, median is lower than mean which means that the distribution of pros and cons length is right skewed, and there are much more shorter reviews than longer ones.
To sum up, the data collected from Glassdoor may be used for further, more advanced analyses, since the basic statistics show that the data is of good quality.

5.2. Scripts performance

All three scripts were executed using the same laptop without running any other heavy programs, so that the time measures are somehow comparable. The below table contains summary of the executions, where there was a 20-second break between scraping each new page. Only first 100 pages were scraped for reference.

Library	**Process time of scraping 100 pages with 20-second time.sleep() (minutes)**
BeautifulSoup	40 min
Scrapy	34 min
Selenium	35 min

6. Team members engagement

- BeautifulSoup scraper – script: Joanna Banasiak, debugging together
- Scrapy scraper – script: Joanna Banasiak, debugging together
- Selenium scraper – script: Aleksandra Biłas, debugging together
- Output data analysis, scripts performance – Aleksandra Biłas
- Description.pdf – together
- README.md – Aleksandra Biłas
- Project.txt – together