**Final\_Project Check point**

Zewen Cui username: zewencui

1. **Project code**

Valid GitHub repo link: https://github.com/zewencui/Final\_Project

contains data access code: none

1. **Data sources** 
   1. **Origin and documentation for each data source is provided accurately:**

https://www.amazon.com/s?k={0}

SEARCH\_QUERY = 'clothing'.replace(' ', '+')

BASE\_URL = 'https://www.amazon.com/s?k={0}'.format(SEARCH\_QUERY)

* 1. **Access techniques are clearly described**

The data is accessed using web scraping techniques, where the program requests data from the Amazon website and extracts the required information using BeautifulSoup.

The relevant data fields such as product name, rating, rating count, price, and product URL are then extracted and stored in a pandas dataframe. The data is filtered by the specified rating count range using a function, and then saved to a CSV file.

The data can be accessed and further manipulated using pandas dataframe methods, such as filtering or sorting based on specific criteria

* 1. **Caching is used where appropriate and evidence is provided :**

import requests\_cache

SEARCH\_QUERY = 'clothing'.replace(' ', '+')

BASE\_URL = 'https://www.amazon.com/s?k={0}'.format(SEARCH\_QUERY)

MAX\_PAGE\_NUM = 50 #

RATING\_COUNT\_RANGE = (0, 1000000)

HEADERS = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0',

'Accept-Language': 'en-US, en;q=0.5'

}

requests\_cache.install\_cache('amazon\_cache', expire\_after=3600)

response = requests.get(BASE\_URL, headers=HEADERS)

* 1. **Data summary is provided and relevant data fields are described**

The data consists of information on different clothing products that are available on Amazon.

The relevant data fields in this dataset are:

\*\*product\*\*:

the name of the clothing product.

\*\*rating\*\*:

the rating of the product as given by Amazon users, in the form of stars out of 5.

\*\*rating count\*\*:

the number of ratings given for the product.

\*\*price\*\*:

the price of the product in USD.

\*\*product url\*\*:

the URL of the product page on Amazon.

1. **Data Structure**
   1. **Summary and is provided and makes sense given the description of data**

the summary of the dataset includes five columns - product, rating, rating count, price, and product URL.

The dataset contains information about various clothing products available on Amazon, including their ratings, rating count, and prices. The product URLs can be used to access the product pages on the Amazon website. The dataset has been filtered based on a specified rating count range to include only the products with a sufficient number of ratings.

Overall, the summary provides a useful overview of the data and makes sense given the description of the dataset.

* 1. **Screenshots demonstrating progress or planning for organizing data into data structures**

The main function main() serves as the entry point for the entire program. It first checks if the data file exists, and if not, it runs the crawl\_data() function to retrieve data and saves it to the clothing.csv file. Then it reads the data through the open\_file() function.

Next, the program enters a loop, asking the user if they want to start using the program. If the user answers "yes", the program calls a series of functions to get the user's search criteria and filter the data, and ultimately recommends a product from all products that meet the criteria.

If there are no products that meet the criteria, the program prompts the user to enter new search criteria; if there is only one product that meets the criteria, the program directly outputs the product; if there are multiple products that meet the criteria, the program sorts them based on the user-specified priority and outputs the final recommended product. Finally, the program asks the user again if they want to continue using the program, and if the user answers "no", the program ends.

Specifically, it includes the following features:

Crawling Amazon website for search results on "clothing", and scraping data from the first 50 pages.

Filtering out clothing products with rating counts outside of the range of 0 to 1000000.

Saving the filtered clothing data into a file named "clothing.csv".

Reading the file "clothing.csv" and extracting the name of each clothing product.

Filtering clothing products that match the specified product type and returning the product names and quantities that match.

Filtering clothing products with rating counts within the specified range and returning the product names and quantities that match.

Filtering clothing products with prices within the specified range and returning the product names that match.

Building a rating tree to group clothing products with the same rating, and then filtering out clothing products that meet the specified minimum and maximum ratings to return the product names that match.

Returning the product names and quantities that match the user's specified product type, price, rating, and rating count from a list containing all clothing data.

Extracting the price, rating, and rating count information of clothing products that match the target product name from the list containing all clothing data, and returning a list of this information.

Choosing the optimal clothing product based on user's priority, which can be determined by the maximum or minimum price, rating count, and rating score.

* 1. **Description of the graphs or trees you plan to organize your data into**

**rest\_tree**: This is a list of dictionaries, where each dictionary contains information about a product. It is created by a list comprehension on the **rows** data and is used in the **get\_filtered\_rows\_by\_rating\_count1** function to filter rows by rating count.

**filtered\_data**: This is a filtered version of **rest\_tree** that contains only the products that meet the specified rating count criteria. It is created in the **get\_products\_within\_rating\_count\_range** function and is used in the **get\_filtered\_rows\_by\_rating\_count** function to filter rows by rating count.

**price\_info, rate\_info, count\_info**: These are dictionaries that contain product information grouped by price, rating, and rating count, respectively. They are created in the **extract\_product\_info** function and are used in the **priority\_choice** function to prioritize products based on the user's input.

**target:** This is a list of products that meet all of the user's specified criteria (i.e., clothing type, price range, rating, and rating count). It is created in the **get\_intersection** function and is used in the **extract\_product\_info** function to group products by price, rating, and rating count.

* 1. **Description of how you will plan to assemble data into thos graphs or trees**

When searching and filtering products, we use two different data structures to help organize and process data. Firstly, we store each product as a dictionary, where each key represents a different product attribute, such as "product name", "price", "rating", "rating count", and so on. These dictionaries are organized into a list, where each dictionary represents a product. Secondly, we use a tree-based data structure to store and organize all products by category, gender, and subcategory. We use the treelib library to construct and store this tree structure, which allows us to quickly find products that match user inputs.

When determining user preferences, we use another data structure: tuples. We store the price, rating, and rating count information for each product as a tuple and organize these tuples into a list so we can sort and compare products to determine the best match for the user's preferences. The use of these data structures helps us better organize and process data, allowing our program to quickly find products that meet user requirements.

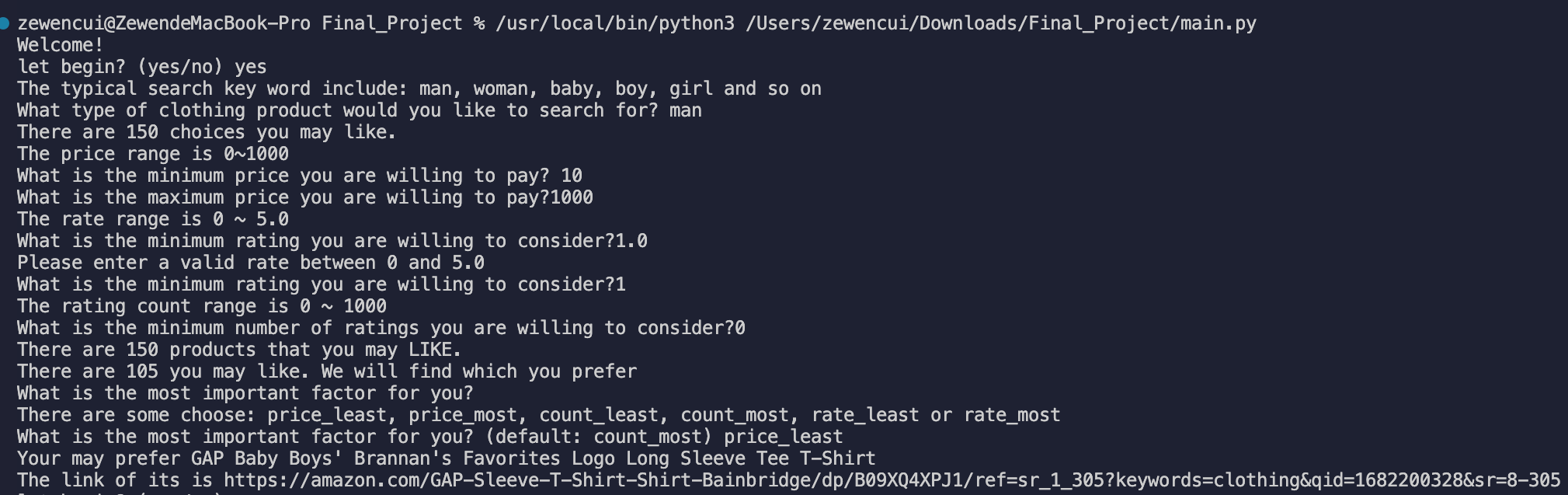
1. **Interaction/ Presentation.**

Plans for application capabilities and interactive/presentation technologies are described clearly and make sense

**command line prompts:**

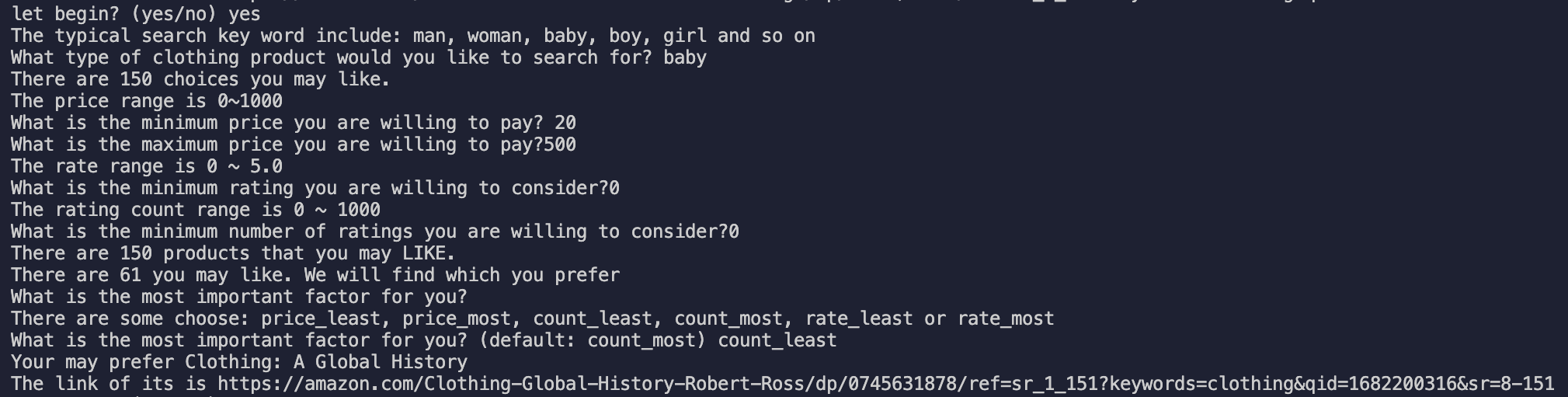
First-try:

We plan help a man find a suitable clothing, who prefer least price.



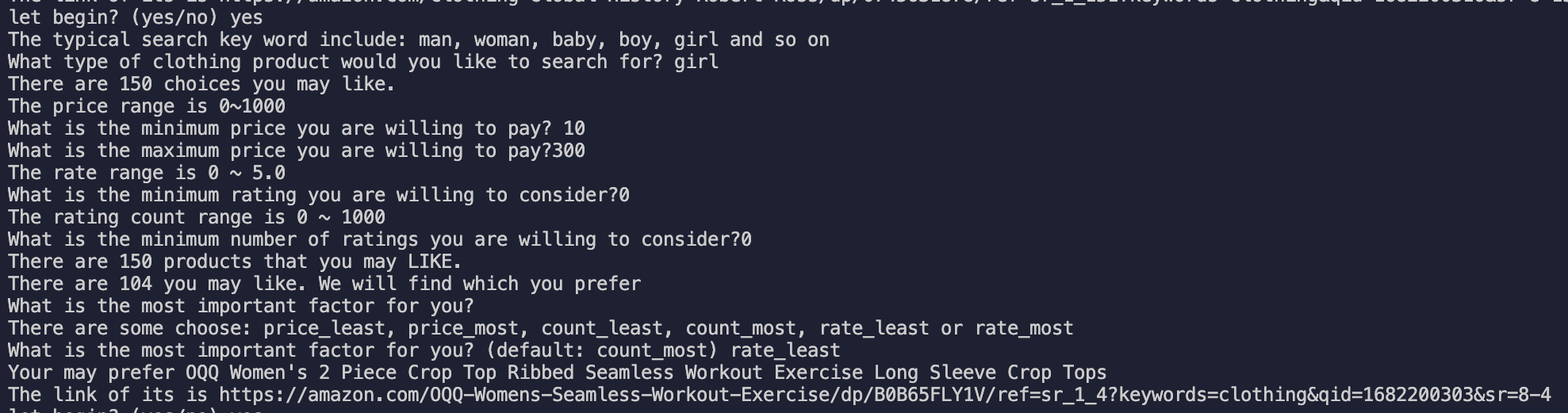
Second try:

We plan help a baby find a suitable clothing, who prefer least count.



3rd try:

We plan help a girl find a suitable clothing, who prefer least rate.



4th try:

We plan help a boy find a suitable clothing, who prefer least price.

