**CPS 842 Final Project Report**

By Mayank Kainth

Introduction

This document is the report for the final project in CPS 842. This project was done independently by me, Mayank Kainth. The project option that I have chosen to do was to create a movie recommender system. This report covers all the various details surrounding the project.

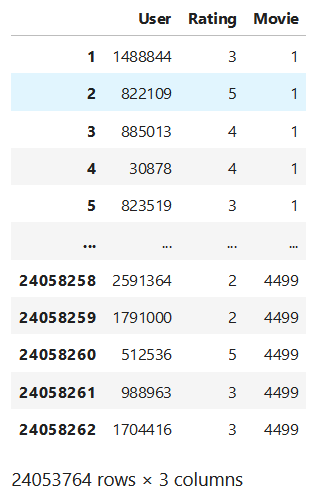
Tools and Data

The programming language used to create this project is Python (version 3.6). The libraries used to create this project are Pandas, NumPy, and Streamlit. Pandas was used extensively, as it is a very powerful library for reading the data, loading them into an efficient data structure (namely the pandas.dataframe for the user-item matrix), and using the provided methods for further computations and processing. NumPy was used for a few provided methods, but mainly because the Pandas library is built upon NumPy. Streamlit is a library I have discovered during the research phase of this project. It was used to create the web application, which is the user interface for interacting with the program. Streamlit is very simple yet powerful for creating webapps in Python. Further tools tools used include PyCharm, which is simply a Python IDE, and Jupyter Notebook, a notebook-style IDE used widely in data science, for the testing and building of my code.

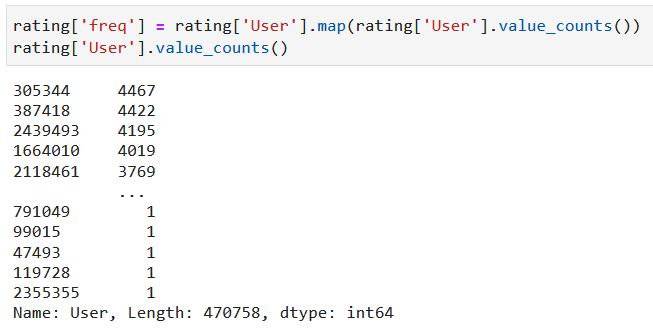
The dataset used for this project was the Netflix prize data, as mentioned in the lectures. The dataset came with many files, but only 2 were used. The files used were *combined\_data\_1.txt* and *movie\_titles.csv*. The movie titles file contains the mapping for the movie ID and titles. The combined data file contains the ratings (from 1-5) that various users have given to every movie. There are a total of 4 combined data files, but only 1 was used as each file is massive and to avoid the increased computational and runtime cost associated with big data. The first file alone contains 24 million ratings by 470 thousand users for 17,770 movies/shows. That is still quite a lot of data, so in my code, I have reduced the user-item matrix to 300,000 of the most active users. In my testing, this reduction comes with a huge gain in performance for a marginal change in results.

Implementation

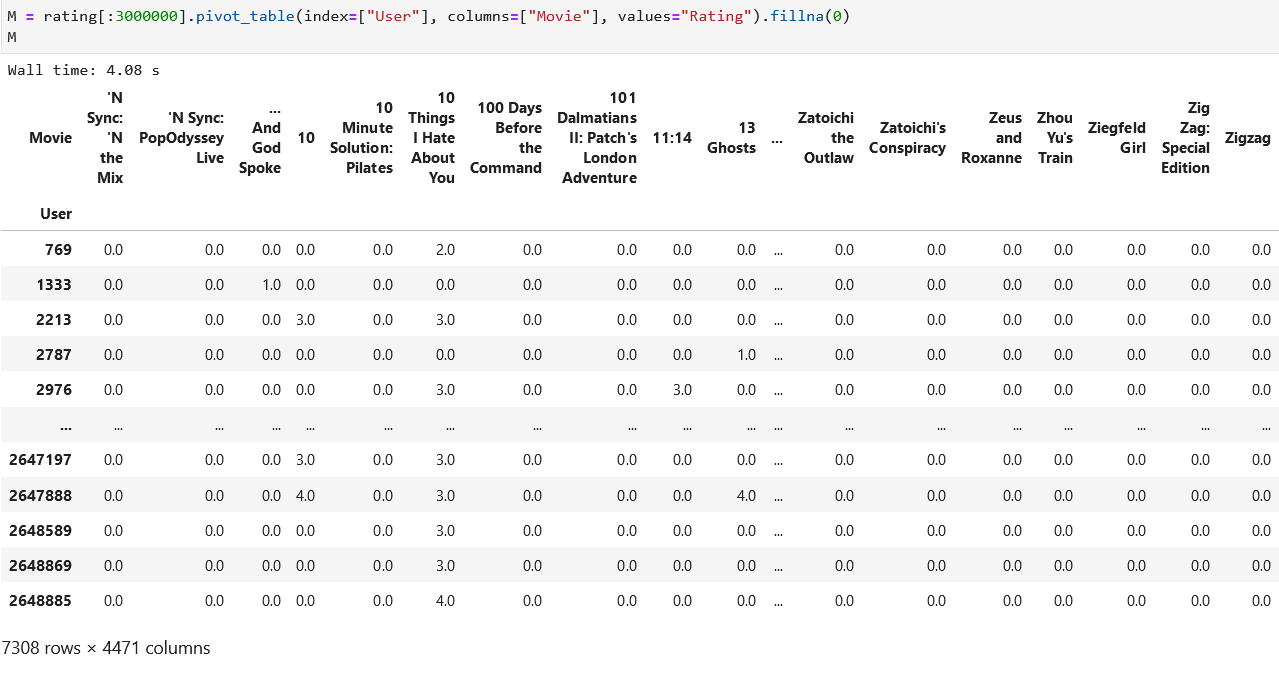
Now that the tools and dataset used have been introduced, I will go over the specifics of the code.

First and foremost, all the libraries used were imported into the program. Then, using the Pandas, I opened *movie\_titles.csv* to load the movie titles and their respective ID’s into a dataframe, and then I converted the dataframe into a hashmap/dictionary. Now that I have all the movie titles and their ID’s in memory, I once again used Pandas to read the *combined\_data\_1.txt*  into a dataframe. This step was messy and required a lot of code because of the unconventional way Netflix provided this data. I have uploaded pictures in this document of how both data frames look.

This is how both data frame look in this step of the process. The size of both tables can be seen from the bottom. Next, I defined a function to change the movieID to movie titles for the second table, and then I sorted this table by the most active users by rating frequency.

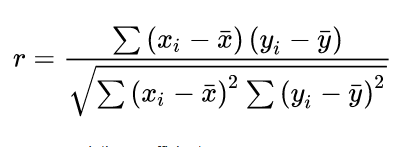
As seen in this picture, the most active user is user 305344 with 4467 ratings.

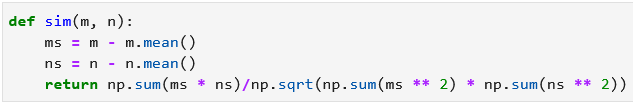
I pivoted the second table to generate what is effectively a user item matrix for all my future processing. Note, the 0 in the table does not mean that the user rated a movie as 0, but that the user has not seen the movie.



After these steps are done, I now have a user-item table to work with. Now I define functions to give recommendations. I defined 2 functions for this. One is item based, and the other is user-based. Both functions utilize Pearson’s R formula for calculating recommendations.

This is the implementation of Pearson’s R formula.





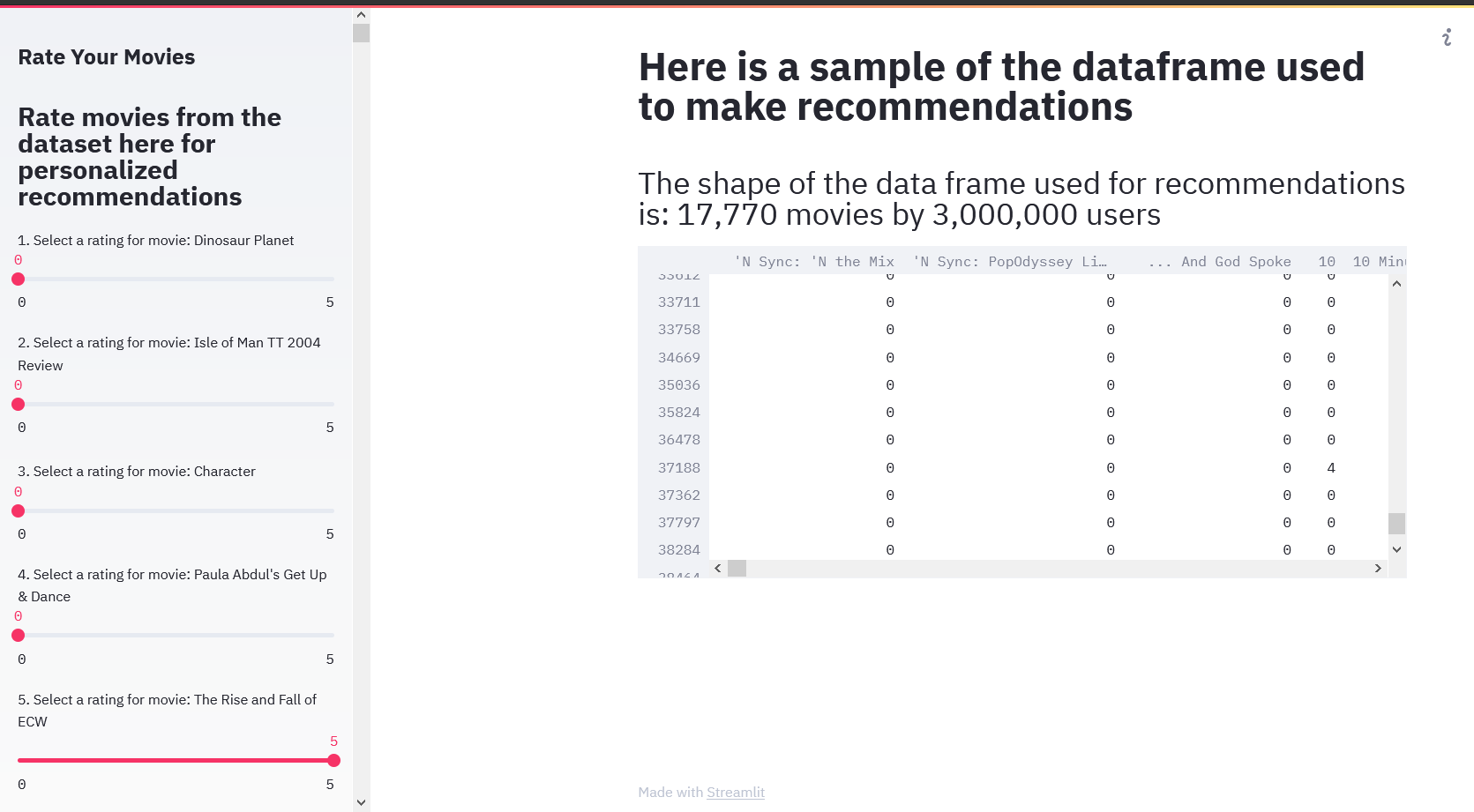
The item-based recommendation compares column of the given movie with all other movie columns, generating a R score for each movie. The function then returns the top 10 movies that are most similar to the given movie.

The user-based recommendation compares the row of the given user with all other user rows, again generating an R value for all users. The difference here is that after all users have been compared, the function adds up the scores given by the top 10 most similar users, and recommends the highest scoring movies. So, this function returns not only the most similar users but also the highest rated movies among these most-similar users for recommendations.

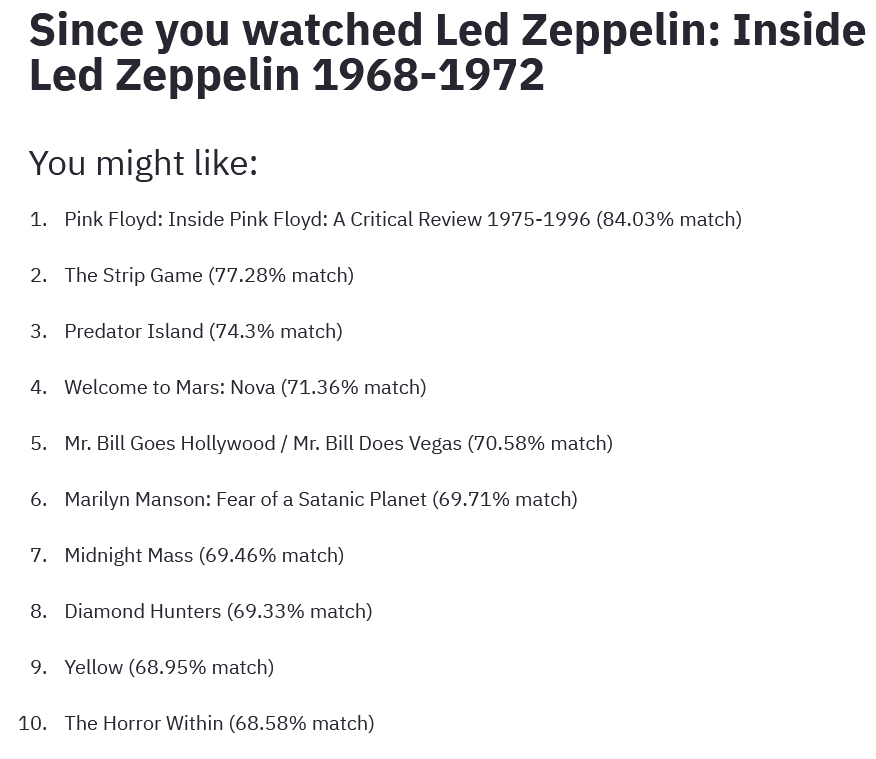
Website

With the data being read and algorithms being implemented, I created the website for user input. The website was built using Streamlit. For user input, I generate a rating slider from 0-5 for every movie (17770 in total) in the dataset that the user can vote on. 0 indicates the movie has not been rated and is the default. After the user has given ratings, the website gives recommendations for the movies that have been rated (item-based), gives most similar users (user-based), and also gives the top movies rated highly by similar users (user-based).

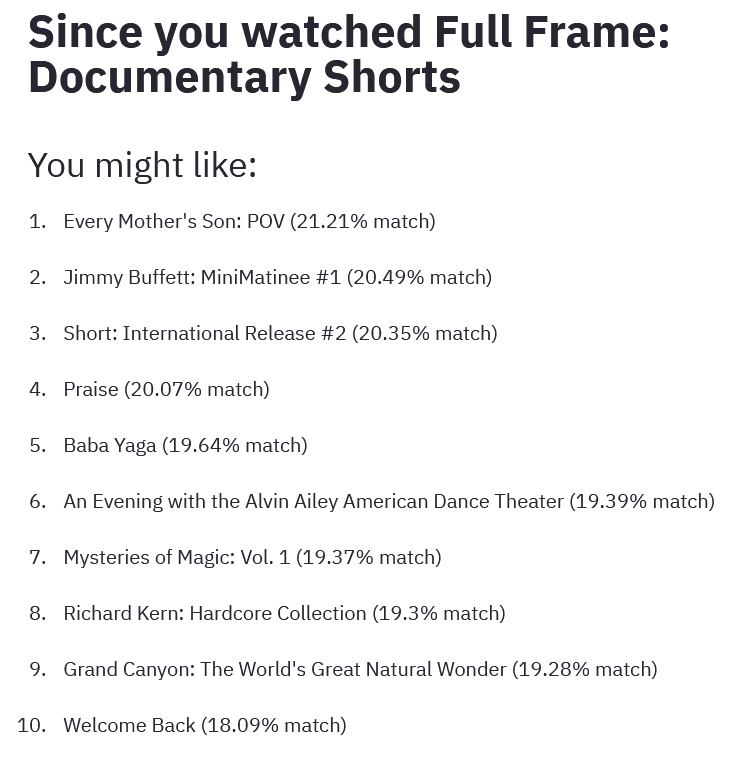
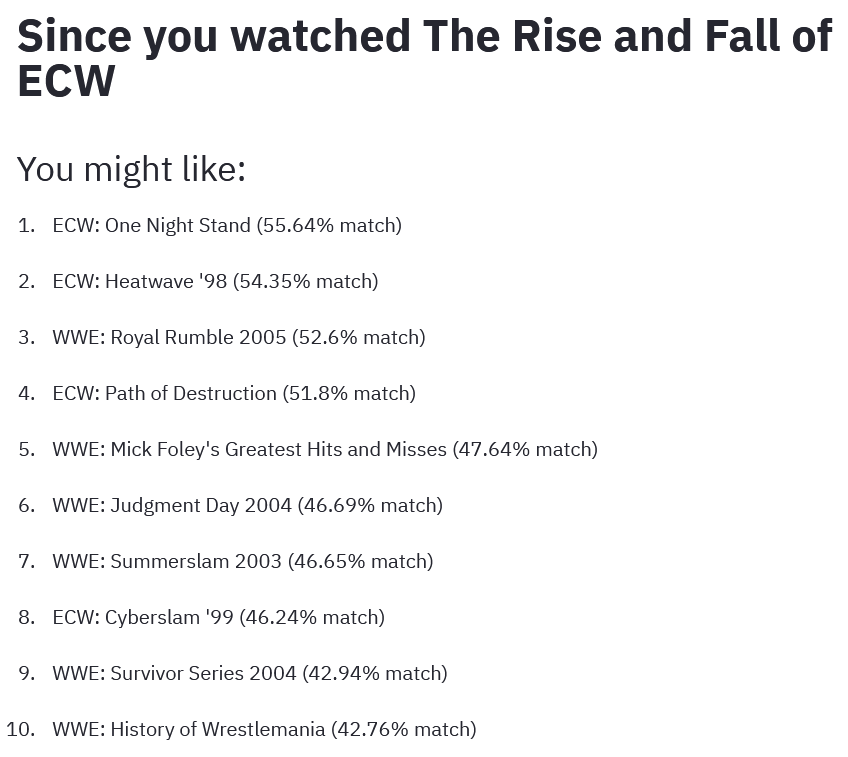
The website looks as such:

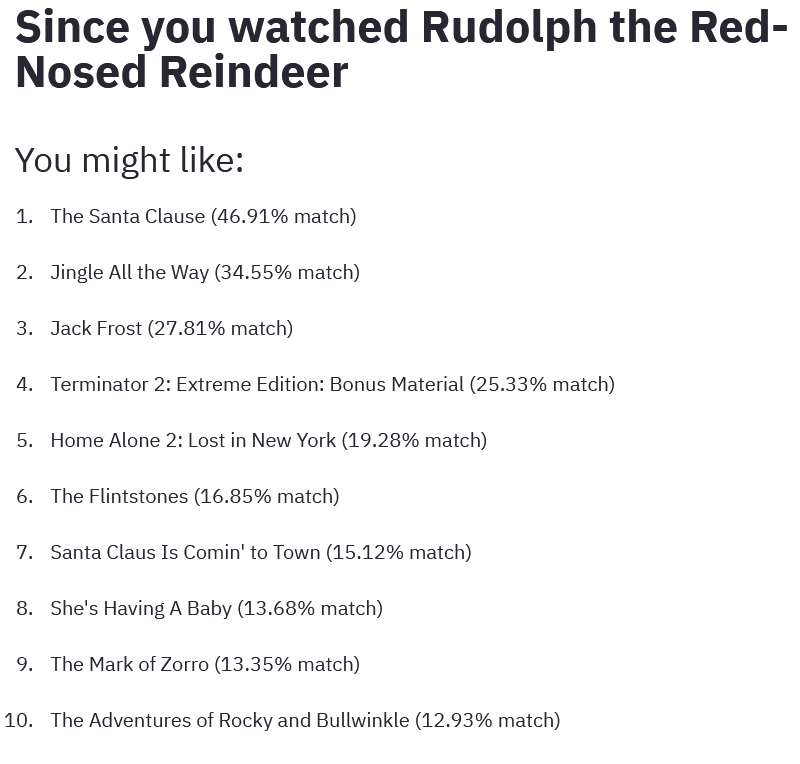
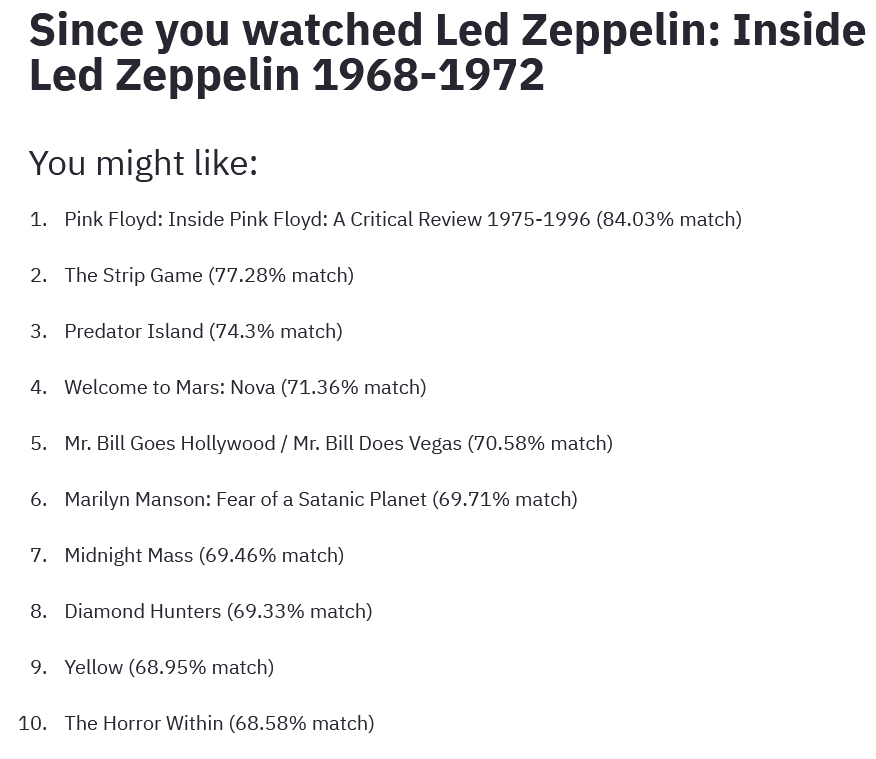


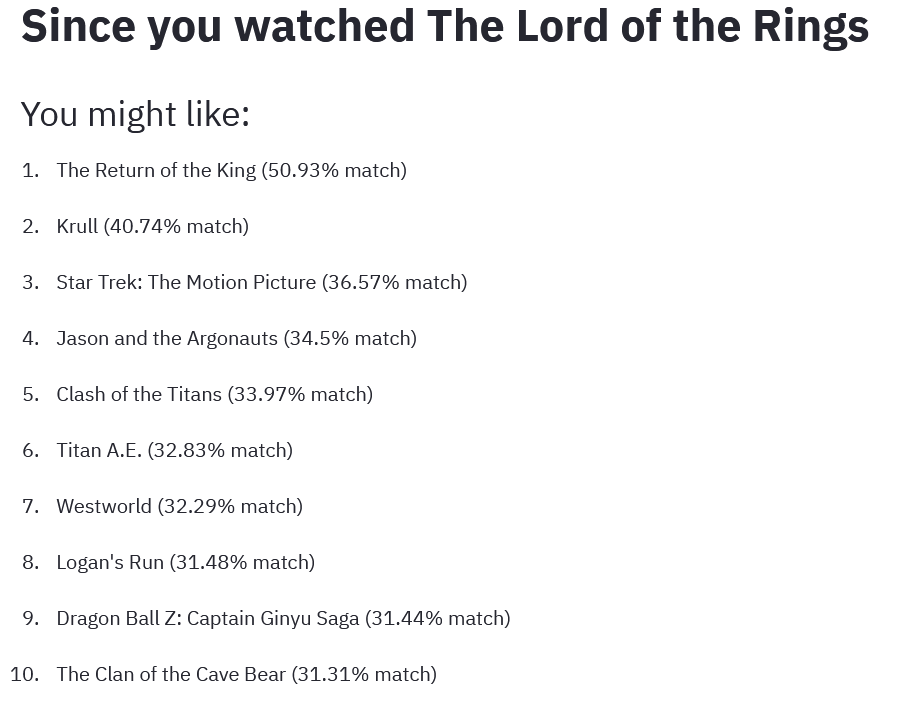
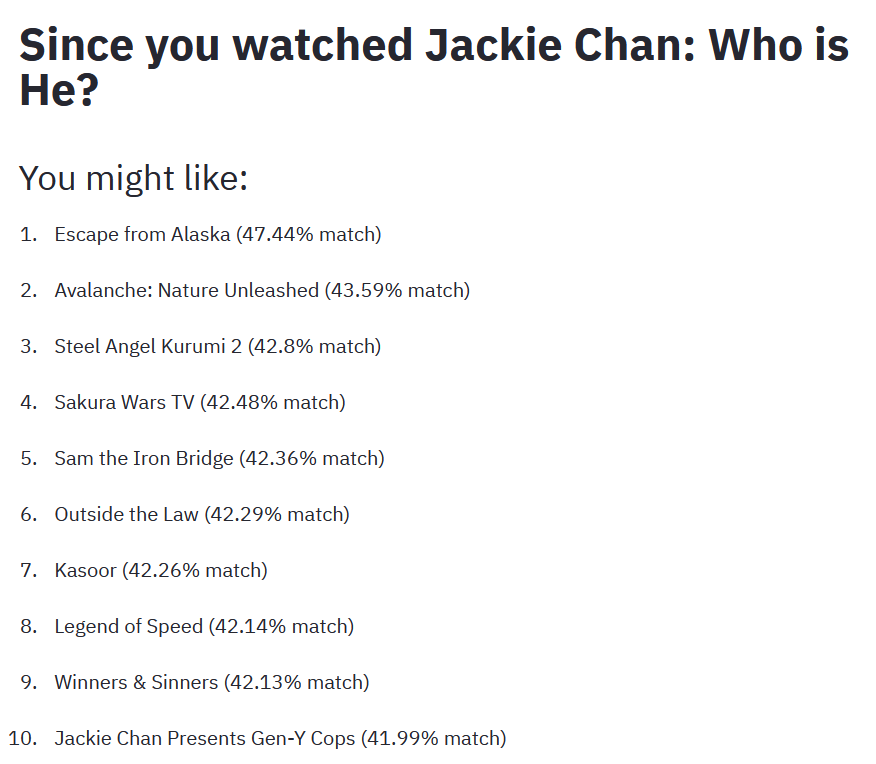
Recommendations

I am very happy with recommendation algorithm. The item-based algorithm works very well, and the user based algorithm is good too. The following are some sample results of the algorithms in action.

Item based

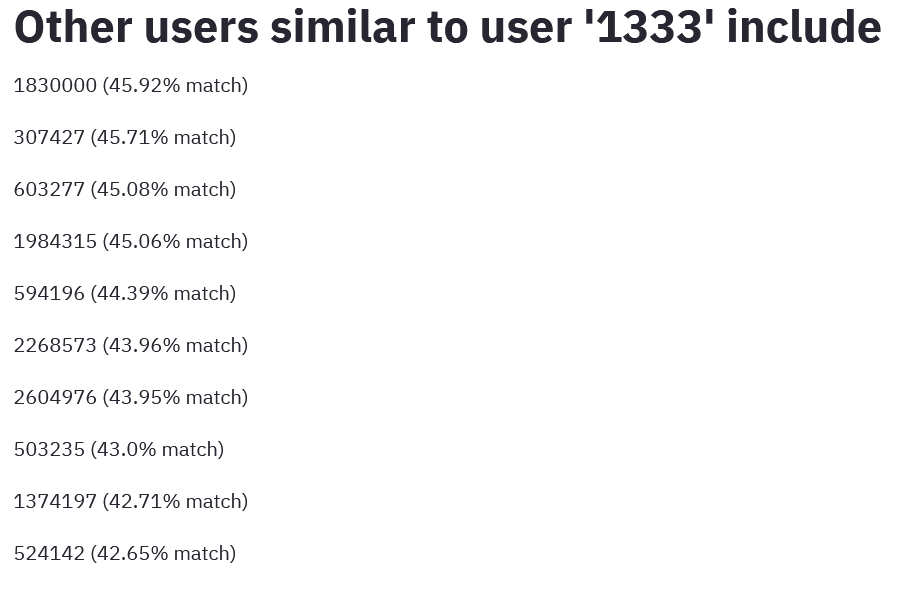
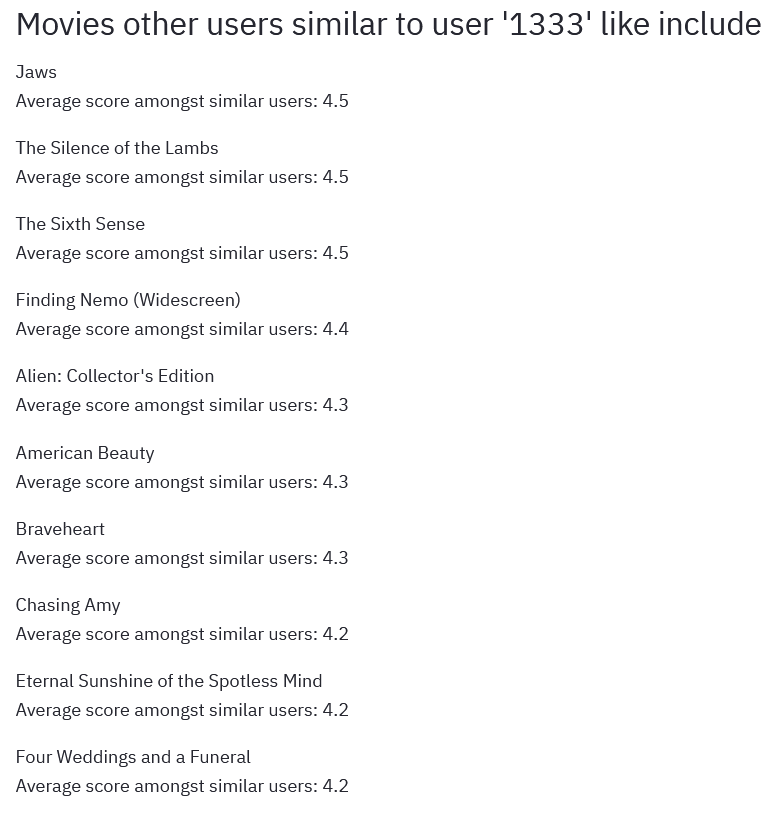


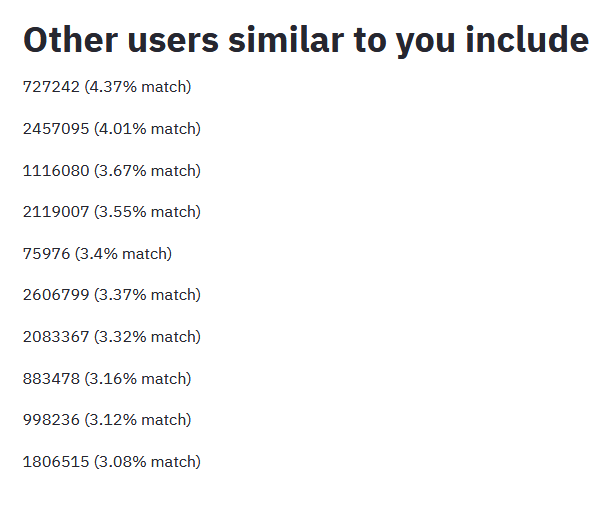
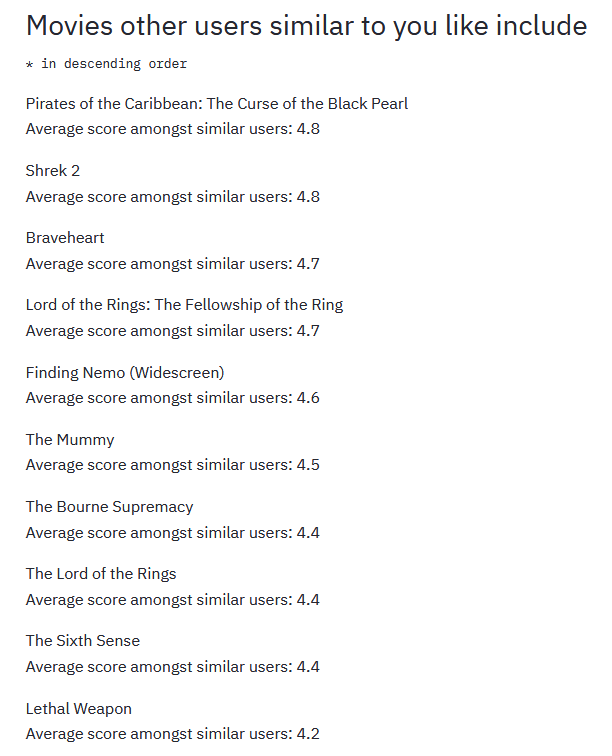
 

User Based

For user in data table

Custom ratings

To Run  
To run code, you must have all files in one directory and all libraries installed using pip. Once this prerequisite is met, go into terminal and run command: streamlit run st\_website.py

This will open the website in the browser. Due to the complications in setting this up, I have included many pictures and samples of my code