Predicting Film Ratings

1. **Introduction**
    a. Let's first go back to the 1930s, when an average of 25 films came out annually in the U.S., and 282 films were released that decade [13], making deciding what to watch much more limiting. But even with those limited releases, 85 million Americans saw those films weekly [12]. Now, just in this past year, 449 movies have been released [11], almost doubling the number released in the entire decade of the 1930s, and that's only a fraction of the over 500,000 films produced just in the U.S. today. And now, even more, people are watching, with most films being easily accessible through streaming services that 1.1 billion people have subscriptions to, like *Netflix* and *Hulu*, and the ability to rent movies online, the number of moviegoers is higher than ever [14]. But with the abundant number of options, deciding what to watch can sometimes become daunting for these viewers. Even after choosing, who knows how someone might feel coming out of the viewing. Maybe they loved the film or hated it, or perhaps they will come out of it thinking, "Why did I waste my time watching this?" That's why we wanted to develop a recommendation system that implements large datasets of movies, their ratings, and individual user ratings for those films, and based on individual users reviews and the overall ratings for the movies, recommend them a list of films that they would most likely enjoy, and predict the possible rating they might give it. Also, if a user wants to know if a specific movie would be a good fit for what they enjoy, they can look up the film, and our system will predict what they would most likely rate. With the billions of people looking for the next thing to watch, our system will significantly help streamline that process. We originally wanted to implement the correlation between the users film ratings and the films genre, leading actors, and its runtime, but it proved difficult to implement those relationships in a timely manner with a group of two. We also wanted to implement users' written reviews as well, but it proved challenging to find a substantial amount of accessible data, and pulling individual words from those reviews to implement for correlation analysis was more problematic than we had expected.
2. **Literary Surve**
    a. a. The use of item-to-item recommendation systems for recommending movies is well established. Not only do streaming services use them to suggest similar movies to the ones a user has already watched on their service, but *YouTube* uses them not only to show similar videos to the ones a user is already watching but also to recommend similar content creators to the ones a user already follows [1]. However, these service recommendation systems are limited by the movies they host on their services. They also base their recommendation on what other users

who have watched the same film have also watched, so there is no way of knowing if what they recommend is something the initial user would be interested in seeing [5]. But our system attempts to predict a list of the most based on how similar they are to films the user has rated highly and movies that other users with similar ratings have also enjoyed in order of most to most minor similarities. With access to any film that has an *IMDB* page [8], the range of films that can be recommended increases immensely. Also, if a user has a particular movie they are wondering if they should watch, they can directly plug in the title with the year it was released, and our system generates a predicted score the user might have given the film if they had watched it. We got the idea to create this system because we had similar experiences with going on to one of the many streaming services and proceeding to flip through their catalog of thousands of movies, wondering which one to watch. We wanted to create a more specific recommendation system to predict what we would most likely enjoy based on our other film ratings and expect what we might give those films if we had watched them.

3. Methodology

a. User-Item-Based Collaborative Filtering:

i. Item-based and user-based collaborative filtering are standard methods used for recommendation algorithms. Item-based methods recommend what an individual user might enjoy based on their prior preference. It relies on the idea that users like and dislike similar products and will give similar ratings to the same products [15]. As well, user-based methods predict a user's recommended items based on other users with similar taste ratings. It works off the assumption that similar users will have similar tastes in items and recommends what those similar users have also rated well [15]. We combined these two models to implement a user-item-based collaborative filtering model [10]. The model starts by calculating the similarity scores of items based on all user ratings for that item. We created a user-item matrix where the items are the rows, the users are the columns, and their values are given by the user's ratings of those items. We normalize that matrix by subtracting the average rating of each item to calculate the mean-centered cosine similarity; we do so to normalize the ratings across all users; this is also known as the Pearson correlation coefficient. Then, we calculate the pairwise correlation between the items and the users in the matrix, producing the item's similarity scores. It then finds the top number of items that are the most similar to the item of interest, and we can decide how many of the top items are produced. After, it calculates the weighted average score for the most similar items

by the user, ranks those items based on that score, and picks the top number of items to recommend based on their previous ratings.

b. Data:

i. We retrieved the movie titles and overall ratings from an *IMDB* database [8] containing data for every movie, short, and TV episode with a rating page on their website, but limited it down to only contain movies. We also gathered data on hundreds of thousands of individual user ratings from the movie dataset site *GroupLens* [7], which organizes its data by individual user IDs and their ratings for individual movies, so we had data to predict movie recommendations for individual users. We merged those two data sets into one big file to use as our primary dataset. We created a table to hold a list of all the movies we had gathered from the two datasets. We made unique IDs to recognize the individual film and all their attributes like genres and runtime, and a table to hold all the user and overall ratings of the movies we had gathered, giving each user a unique ID to separate their reviews from others. We then used parallel processing functions to merge the two tables based on the movie IDs we had given them so that our entries for the rating had all the film's information. And once the merge was completed, we used this combined dataset as our primary sampling data.

4. Implementation

In the initial stages of crafting the Python movie-based classifier, I integrated data from two distinct datasets: one containing user movie reviews and another comprising details on a vast array of 9 million movies. Leveraging parallel processing techniques, I efficiently sorted and merged this diverse dataset, orchestrating a seamless integration of user reviews with the extensive movie database. I cleaned the combined dataset, ensuring data integrity and coherence. The refined dataset was subsequently written to a CSV file, laying the groundwork for its utilization in the movie-based classifier.

Moving on to the preprocessing stage, I created a function to aggregate ratings by movie title. This involved calculating the mean rating and the number of ratings for each movie. I filtered out movies with less than 100 ratings, resulting in a refined DataFrame called agg_ratings_GT100. I made sure to print the head of this DataFrame to check the modifications.

To further refine the aggregated ratings, I utilized another function called update_agg_ratings. This function grouped the data by movie title, calculated additional statistics, and updated the DataFrame columns accordingly. Once again, I printed the head of the updated DataFrame to observe the changes.

With the preprocessed and refined data in hand, I proceeded to create a test set. This involved randomly selecting one row for each unique user, considering only those

users who had provided a rating. The same random sample was ensured for consistency using the random_state=1 argument until ready to test.

Simultaneously, I created a training set by excluding the test set rows from the agg_ratings_GT100 DataFrame. The head of both the training and test sets was printed to verify the composition of the datasets.

Next, I transformed the training set into a user-item matrix, representing the ratings users gave to movies. This matrix was normalized by subtracting the mean rating of each movie. Subsequently, I calculated the similarity between movies based on this normalized matrix, resulting in an item similarity matrix.

To assess the performance of the classifier, I predicted ratings for users and movies in the test set using an item-based collaborative filtering approach. The predicted ratings were then compared with the actual ratings, and the results were logged in a text file for future analysis.

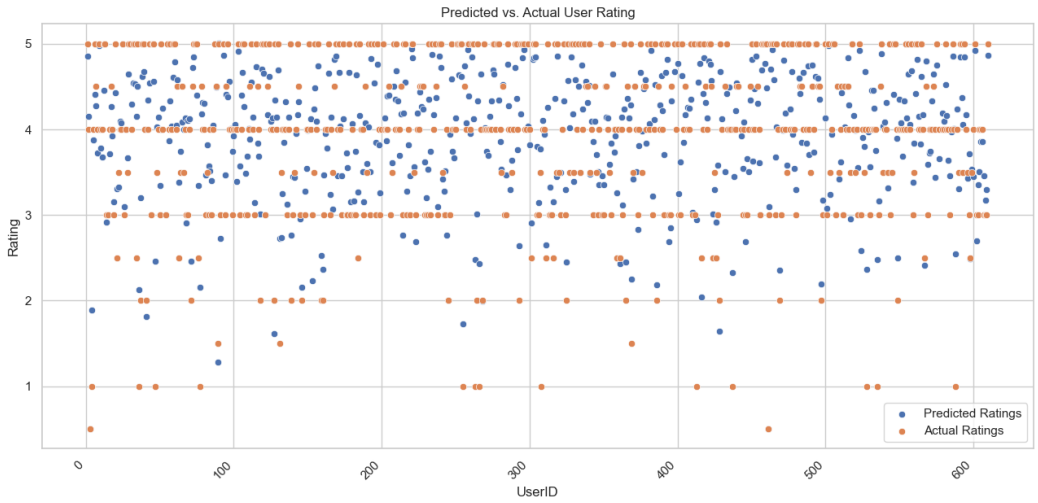
Additionally, I employed the item-based recommendation function to suggest movies for users in the test set. This involved identifying the top and bottom predictions, as well as determining the user's favorite and least favorite movies. These recommendations were also documented in a text file for further examination.

In conclusion, the script undertook a comprehensive process of data loading, preprocessing, refinement, transformation, and evaluation to create a movie-based classifier using collaborative filtering techniques.

5. Experimental Results

a.

```
Head of matrix_norm DataFrame:
user_id                          1    ...  610
title                                 ...
2001: A Space Odyssey (1968)     NaN  ...  4.5
Head of item_similarity DataFrame:
title                            2001: A Space Odyssey (1968)  ...  X-Men (2000)
title                                                          ...
2001: A Space Odyssey (1968)                         1.000000  ...     -0.123862
Ace Ventura: Pet Detective (1994)                   -0.036319  ...      0.045676
Aladdin (1992)                                       0.017446  ...      0.285480
Alien (1979)                                         0.318523  ...      0.030257
Aliens (1986)                                        0.317386  ...      0.225923
```

b. 

c. We aggregated 19,192 user review samples from 100,837 films from movie database references for our experiment. We had 600 test samples of user ratings to test the rating prediction ability of our algorithm after implementing 19,192 training samples. Our experiment consisted of item and user-based collaborative filtering, similarity matrices, and Pearson Correlation coefficient. In our first data set, we compared the similarities of the films *2001: A Space Odyssey* and *X-Men* to *Ace Ventura: Pet Detective, Aladdin, Alien, Aliens, and 2001: A Space Odyssey*. We compared *2001* to itself to show that it produces the correct output to show the similarity scale when compared to something identical. Our similarity bases itself on how similar the overall average of the user ratings of the two films are. The closer the measurement is to one, the more similar the two films are to each other, and the closer it is to negative one, the more unlike they are. *2001* has an average user rating of 3.89, and *Alien* has 3.96. Our similarity score was impacted by the most minor differences in the overall user ratings, so the similarity between the two was only 0.3185. And with *X-men* having an average score of 3.69, the similarity will be even lower.

d. We created a scatter plot to show the accuracy of our predicted user rating of a movie against their actual rating. We used 600 individual user ratings of one movie, where they had a rating scale of one to five but could only choose the exact number or that number and a half, with the median of the scale being 2.5. But, since we based our predicted ratings on our similarity scores, we predicted scores that were in between that scale. With those being predicted and actual measurements, our predicted ratings follow very closely with most of the actual ratings, and when there is distance between those points, our predicted rating normally has less than a 0.5 difference from a user's actual rating. For example, the third user in our dataset gave the film *Schindler's List* a rating of 0.5, and our predictions algorithm was able to predict it exactly. We were not able to get our rating prediction's algorithm completely accurate though. For example, the 34th

user in our dataset had given the movie *X-men* a perfect rating, but our algorithm predicted it to be 4.35 instead. Most of our predicted ratings fell into the ladder example category where it was off by less than one of the actual user ratings of a film. Which meant that our prediction model that it is functioning as we had intended it to.

6. **Conclusion and Future Research**
    a. In conclusion, the movie recommendation algorithm we created uses a dataset of user-written reviews to recommend specific media to a given user better. Using a user-item-based collaborative filtering model, we created an algorithm that predicted user ratings. We compared them to user preferences to recommend media best suited to users' tastes. In comparing thousands of user reviews with the expected user ratings, our algorithm demonstrated that it was capable of recommending movies a user may like with reliability and accuracy. We only completed this project after hitting a few bumps in the road. The biggest challenges we faced were working in such a small team, which made the workload heavier on the two of us, as well as having to work with such an extensive database, which made compiling very difficult as well as many of the compilers we used not did now allow for the usage of such large data sets. In the future, it may be more beneficial to break up the data set into multiple smaller pieces or use a compiler that allows for the mass understanding of 100,837 data. We recognize the importance of efficiently handling large datasets in future projects for quicker turnaround time. This project was also one of our first exposures to Python, so there was a learning curve, but a new skill has also been developed for future use. If we continue to develop this algorithm, we would want to increase the variables we used to impact our similarity score and predicted ratings, such as the film genre, runtime, and actors involved in the films a user enjoys or dislikes. All in all, this was a challenging but fulfilling project where we gained valuable information on the uses of collaborative filtering models and their applications.

7. **References**

    a. [1] Contal, E. (2022, September 8). *What are the top recommendation engine algorithms used nowadays?*. Medium. https://itnext.io/what-are-the-top-recommendation-engine-algorithms-used-nowadays-646f588ce639
    b. [2] Chen, D. (2020, August 5). *Recommender System — Singular value decomposition (SVD) & truncated SVD*. Medium. Retrieved October 13, 2023, from https://medium.com/@m_n_malaeb/singular-value-decomposition-svd-in-recommender-systems-for-non-math-statistics-programming-4a622de653e9
    c. [3] Zhang, A., Li, M., Smola, A. J., & Lipton, Z. (2023). *Dive Into Deep Learning*. Cambridge University Press.
    d. [4] Vidiyala, Ramya. “How to Build a Movie Recommendation System?” Medium, Towards Data Science, 21 Oct. 2020, towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109.

e. [5] Jayalakshmi S, Ganesh N, Čep R, Senthil Murugan J. Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. Sensors (Basel). 2022 Jun 29;22(13):4904. doi: 10.3390/s22134904. PMID: 35808398; PMCID: PMC9269752.
f. [6] Kniazieva, Yuliia. "Guide to Movie Recommendation Systems Using Machine Learning." High Quality Data Annotation for Machine Learning, Label Your Data, 14 Apr. 2022, labelyourdata.com/articles/movie-recommendation-with-machine-learning.
g. [7] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872, from, https://www.citethisforme.com/cite/sources/websiteautociteconfirm
h. [8] IMDb (n.d.). *IMDb Non-Commercial Datasets*. IMDb Developer. Retrieved October 13, 2023, from https://developer.imdb.com/non-commercial-datasets/
i. [9] Vidiyala, R. (2020) *How to build a movie recommendation system?*, *Medium*. From,https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109 (Accessed: 08 December 2023).
j. [10] Grab N Go Info (2022) *Google colaboratory*, *Google Colab*. Available at: https://colab.research.google.com/drive/1GYSJNXK6lRl8kb2FvtMPLqW8EuZLcJjN?usp=sharing#scrollTo=_NLnHDM34R2B (Accessed: 08 December 2023).
k. [11] Published by Statista Research Department, S.R.D. (2023) *U.S. & Canada: Movie releases per year 2022*, *Statista*. Available at: https://www.statista.com/statistics/187122/movie-releases-in-north-america-since-2001/ (Accessed: 08 December 2023).
l. [12] "1930s: Film and Theater." *bowling, beatniks, and Bell-bottoms: Pop culture of 20th-century America. . encyclopedia.com. 15 Nov. 2023* . (2023) *Encyclopedia.com*. Available at: https://www.encyclopedia.com/history/culture-magazines/1930s-film-and-theater#:~:text=Eighty%2Dfive%20million%20people%20a,movies%20from%20which%20to%20choose (Accessed: 08 December 2023).
m. [13] Nash Information Services, LLC (no date) *Top united states movies of each year*, *The Numbers*. Available at: https://www.the-numbers.com/United-States/movies#tab=year (Accessed: 08 December 2023).
n. [14] Duarte, F. (2023) *Video Streaming Services Stats (2023)*, *Exploding Topics*. Available at: https://explodingtopics.com/blog/video-streaming-stats (Accessed: 08 December 2023).
o. [15] Pinela, C. (2017) *Recommender Systems - user-based and item-based collaborative filtering*, *Medium*. Available at: https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f (Accessed: 08 December 2023).