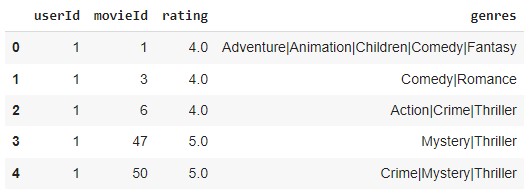
For most people we do not realize how ingrained recommender systems are in our daily lives. There are the usual suspects: Netflix, YouTube, Amazon, Spotify, and other streaming platforms. Recommender systems are also frequently seen on shopping websites. Often while surfing a website a field or pop-up will say something along the lines of “other shoppers who purchased this item also purchased these items” followed by a list of compatible items. Quality recommendations can be a make-or-break point for how often a customer or user may come back to the site. Return visits and increased length of visits are beneficial for a myriad of reasons, mostly hoping the visitor will spend money.

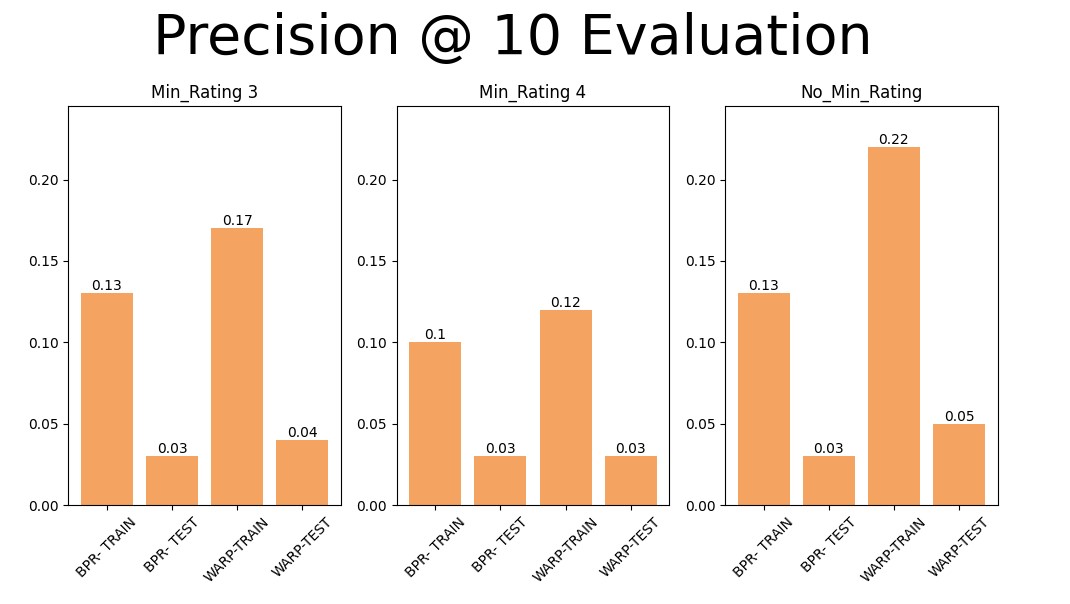
The dataset utilized for this project is the “small” MovieLens 100k dataset obtained via download from [https://grouplens.org/datasets/movielens/.](https://grouplens.org/datasets/movielens/) This download included four comma separated value files (csv); movies, ratings, tags, & links. The links csv is a primary key list of each movieId and their corresponding identification for both the Internet Movie Database and The Movie Database. The tags csv is a list of words/phrases that reviewers associated with movies they were rating along with the movieId. The movies csv includes each movie with its movieId on a separate row along with the full title of the movie and the applicable genres it belongs in. The ratings csv contains each review score on a scale of 0:5 along with the userId and movieId. For my purposes in this project, I wanted to join the movies csv and the ratings csv only keeping the userId, movieId, rating, and genre for each review.

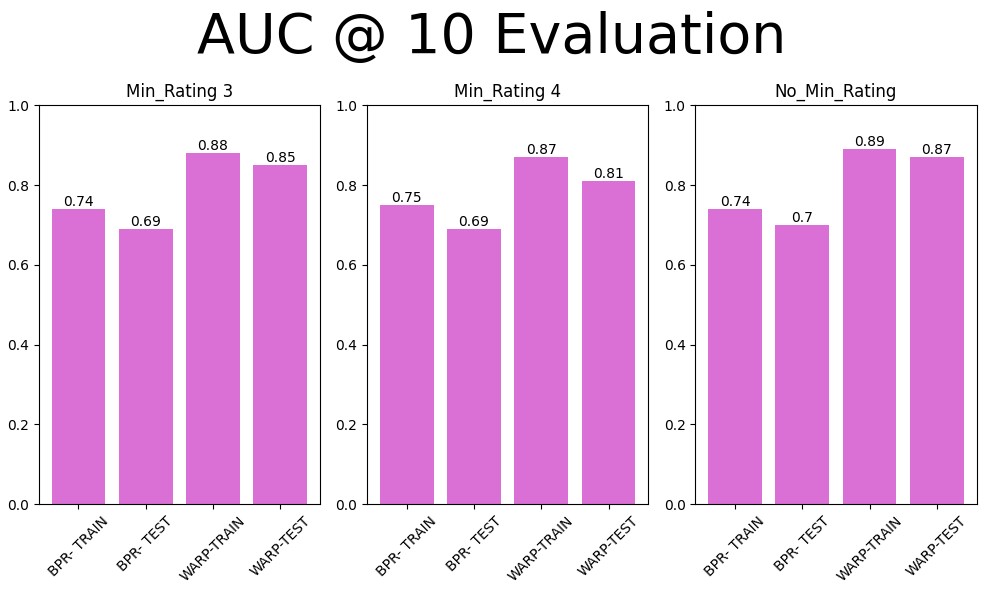


After this I split the genres column by each delimiter creating a list of all genres. As this is a list of all ratings per movie, there were no null or N/A values to contend with thankfully. The recommender model I chose to utilize was the LightFM model which is well-known for combining both content-based filtering as well as collaborative-based filtering. Using LightFM, the model is able to make suggestions for new movies based on the ratings of that specific user, the content of the movies the user has rated, and the ratings of other similar users. LightFM has a built-in dataset class allowing me to take the dataframe above and convert it to the matrices that the model requires.

I tried two different loss functions for the model on three different subsets of the data. Since the goal of the recommender system is to recommend movies a user would like, it may be best to only include ratings that were highly rated. I tried three different subsets: one with only ratings above 3 (3.5, 4.0, 4.5, & 5.0 only), another with only ratings above 4 (4.5 & 5.0 only), and one with all ratings regardless of score. Then each of these three datasets were fit first with a loss function of Bayesian Personalized Ranking (BPR) and second with Weighted Approximate-

Rank Pairwise (WARP). Below are the results:





Using a 20/80 testing/training split on each dataset, the model’s precision on the training data increased from each run of the BPR compared to WARP. The testing precision was fairly even regardless of loss function or data subset. The AUC for BPR on both testing and training data remained fairly similar regardless of data subset. The AUC utilizing WARP loss function increased just as we saw with precision as well. For both precision and AUC the entire dataset (no ratings removed) resulted in the highest scores.

After calling the model using a WARP loss function and fitting it on the entire dataset, I put in a random group of userIds. Here is what the program resulted in:

A screen shot of a computer

Description automatically generated

For UserId 3, the program is recommending Mr. Mom, Walk the Line, and Raiders of the Lost Ark. Next, I used an API to request additional movie information from The Movie Database (https://www.themoviedb.org). The API requires an API key which needs to be obtained from The Movie Database directly. It is free as long as it is for private and non-commercial use. I then sent each movieId from the list above for userId 3. There is a great deal of information available for each film, so I filtered the results to include only the title, a description, and the movie’s homepage if applicable. The resulting JSON file was modified to a list and then entered into a dataframe for easy viewing.

Lastly, I repeated this API process with a different URL to The Movie Database to obtain streaming providers for each film. The resulting JSON file was modified to only report the United States version of the JustWatch website. This was done as I anticipate most users to be based in the US and it is familiar. What the API returned is a web address for each requested film. At that web address there is typically a movie poster image, a list of cast members, and of course providers where a user can stream/rent/purchase the film. It is important to note here that The Movie Database and JustWatch have an agreement to share information via API as long as the source of the data is attributed to JustWatch. Here is an example for Mr. Mom:

A screenshot of a computer

Description automatically generated

<https://www.themoviedb.org/movie/13105-mr-mom/watch?locale=US>

The final step was to take this website generated for the US region and append it to the existing dataframe. Here is what the final product looks like that the user would see:

A screenshot of a phone

Description automatically generated

There are certainly limitations and challenges to be aware of regarding this movie recommender system. As the dataset is a static database, it would be a manual process to update or add additional ratings by users or additional movies. A recommendation to avoid this may be to utilize an updated dataset from MovieLens. There are a number of updated datasets available including one that includes a staggering 25 million ratings. Another limitation is referred to as the ”cold start”. This occurs when attempting to generate recommendations for a new user who does not yet have a robust ratings history from which to mathematically infer correlations. A recommendation for this issue would be to gather additional demographic information on the user and/or request the user rate a number of movies before a recommendation can be generated.

I do not foresee any ethical issues with this project or this data. The userIds are anonymized so that no personal data is being shared and no other identifying information is available. There are other datasets that include additional user information to further enhance user features such as occupation, zip code, and sex. If a database with those additional features is used there may need to be another review for ethical implications. Also, this project is being done as an educational endeavor and therefore does not violate the terms of service with The Movie Database API. Care is to be taken to ensure that all proper credit is attributed to the party responsible.

# Q & A

1. How many different movies were reviewed in the data?
   1. 9,742
2. How many users were included in the data?
   1. 610
3. What are the different genres in the movie data?
   1. There are twenty different genres including “no genre listed”. There are over 100,000 different combinations of genres in which films may be considered in more than one.
4. What was the most popular movie
   1. 296 movies have an average rating of a perfect 5.0. Seven of those films were reviewed twice while the rest were only reviewed once. Those movies are:

|  |
| --- |
| Enter the Void (2009) |
| Come and See (Idi i smotri) (1985) |
| Belle Ã©poque (1992) |
| Jonah Who Will Be 25 in the Year 2000 (Jonas qui aura 25 ans en l'an 2000) (1976) |
| Lesson Faust (1994) |
| Lamerica (1994) |
| Heidi Fleiss: Hollywood Madam (1995) |

1. Least popular movie?
   1. 96 movies are tied for an average rating of 0.50. Only two of those movies were rated more than once: Cyborg (1989) and Pokemon 4 Ever (2002).
2. How could new reviews be added in?
   1. The ratings csv would need to be appended to for each new user. They would need to be assigned a userId and then additional reviews could be included.
3. Is this scalable?
   1. Yes, depending on the use. This would not be scalable or reasonable to create an application for widespread use. Also that already exists via JustWatch. It is able to be amended through adding new movies to the movie csv and ratings to the ratings csv.
4. Can recommendations only be generated for one user at a time?
   1. Recommendations can be generated for one user at a time, a list of users by userId, as well as for the list of users overall.
5. Any dependencies?
   1. The public and free API for The Movie Database is something that will be needed to be applied for if this program is to be utilized by anyone else. The LightFM library also needs to be installed on their machine.
6. Additional user descriptors?
   1. Hopefully more demographic information such as gender, household income, zip code, etc. could be added in the future.

# Works Cited

*Guide to Movie Recommendation Systems Using Machine Learning*. (2022, April 14). Labelyourdata.com. https://labelyourdata.com/articles/movie-recommendation-withmachine-learning#:~:text=The%20primary%20goal%20of%20movie

Gusarova, M. (2023, February 5). *Understanding AUC — ROC and Precision-Recall Curves*.

Medium. https://medium.com/@data.science.enthusiast/auc-roc-curve-ae9180eaf4f7

Harper, F. M., & Konstan, J. A. (2015). The MovieLens Datasets. *ACM Transactions on*

*Interactive Intelligent Systems*, *5*(4), 1–19. https://doi.org/10.1145/2827872

Hope, J. (n.d.). *recommender/recommender.py at master · jamesdhope/recommender*. GitHub. Retrieved July 16, 2023, from https://github.com/jamesdhope/recommender/blob/master/recommender.py *MovieLens*. (2019, April 26). GroupLens. https://grouplens.org/datasets/movielens/

*Welcome to LightFM’s documentation! — LightFM 1.16 documentation*. (n.d.). Making.lyst.com. Retrieved July 16, 2023, from https://making.lyst.com/lightfm/docs/home.html#