Building a Movie Recommender System Using Movie Ratings and the Surprise Scikit Library

Jason Cabrera

Computer Science
College of Natural Sciences
The University of Texas at Austin

ABSTRACT

Movies have always been an escape into many worlds. The way in which we now receive our movie recommendations has changed drastically compared to a few decades ago. Video streaming services like Netflix are now using machine learning-based recommender systems to recommend to us users the movies that best fit our preferences. We briefly explore how recommender systems are used and how they play a critical role in determining whether a user subscribes to one streaming service over another. After said discussion, we move onto implementing our recommender system with the help of the Surprise sci-kit learn library from Python. We benchmark many underlying prediction algorithms provided by Surprise which leads us to use an SVD derivative, SVDpp, as our prediction algorithm for our recommender system. We then create a recommender system using this prediction algorithm and recommend 5 unwatched movies to the user with the highest number of reviews in our dataset. The resulting recommendation are reasonable and fitting for the user's preferences, and thus we have created a successful recommender system. However, there are still areas of improvement if it were to be used in an actual video streaming service.

1. INTRODUCTION

Movies have always been an escape into many worlds and times, fictional and nonfictional. Throughout the digital age, the way in which we consume movies has drastically changed. We've went from physical, dedicated movie stores, to now having access to a potentially unlimited supply of movies through digital streaming services, from the comfort of our home. Naturally, this also changes the way in which we receive recommendations for our next movie. During the era of heading to physical video stores, the average movie watcher would rely more on public opinion to find their next movie for a movie night. With streaming services, we can now receive our recommendations straight from the streaming service itself, such as Netflix or HBO Max. These streaming services use a plethora of machine learning techniques to learn what the user might like and proceeds to recommend it to said user. In this paper, we attempt to build a recommender system that can recommend the best unwatched movies to a user so that they may find a movie to watch for their next movie night. We define the best movie in this scenario as an unwatched movie that a user is most likely to enjoy the most.

Our research question involves the following:

(1) Evaluating a variety of prediction algorithms by Surprise, a Python scikit library to see which ones perform the best on our dataset. We then rank the various prediction algorithms by their RMSE performance metric.

(2) Using the best performing algorithm, we will build a recommender system and recommend 5 movies to the user with the greatest number of ratings. Upon getting the recommendations, we will determine whether the results seem reasonable using data analysis.

2. BACKGROUND

On April 14, 1998, NetFlix.com first launched to consumers as a DVD rental service (Bitran, 2022). It wasn't until 2007, when Netflix started its game changing streaming service (Bitran, 2022). Since that moment and especially during the Covid-19 pandemic, companies have noticed how successful Netflix was and decided to release their own video streaming service as well. Services such as Hulu, HBO Max, and Disney Plus were launched, which resulted in 20+ video streaming services in the United States as of 2022 (Cook, 2022). These companies released their own streaming service in hopes to get their share of users in the huge, still growing market of video streaming consumers. According to CloudWatch in their 2022 streaming service statistics, 85% of U.S. households have at least one video streaming service (Pattison, 2022). Furthermore, CloudWatch estimates that the streaming industry is expected to be worth $330 billion by 2030 which includes video streaming services (Pattison, 2022). Another significant statistic reported from CloudWatch was that in 2021 there was 15 million years' worth of video content streamed in 2021 (Pattison, 2022). These statistics alone show that video streaming is an important and highly competitive space in the world of ecommerce. We now explore how recommendation systems give a competitive edge to one streaming service over another.

To stress the importance of recommendation systems in video streaming, it's important to note what makes a user choose one video streaming service over another. NPR conducted a poll in 2022 which asked users to rank the factors that determine whether they'd subscribe to a service or not (Deggans, 2022). The figure below describes the results of the poll, sorted from most important to least.

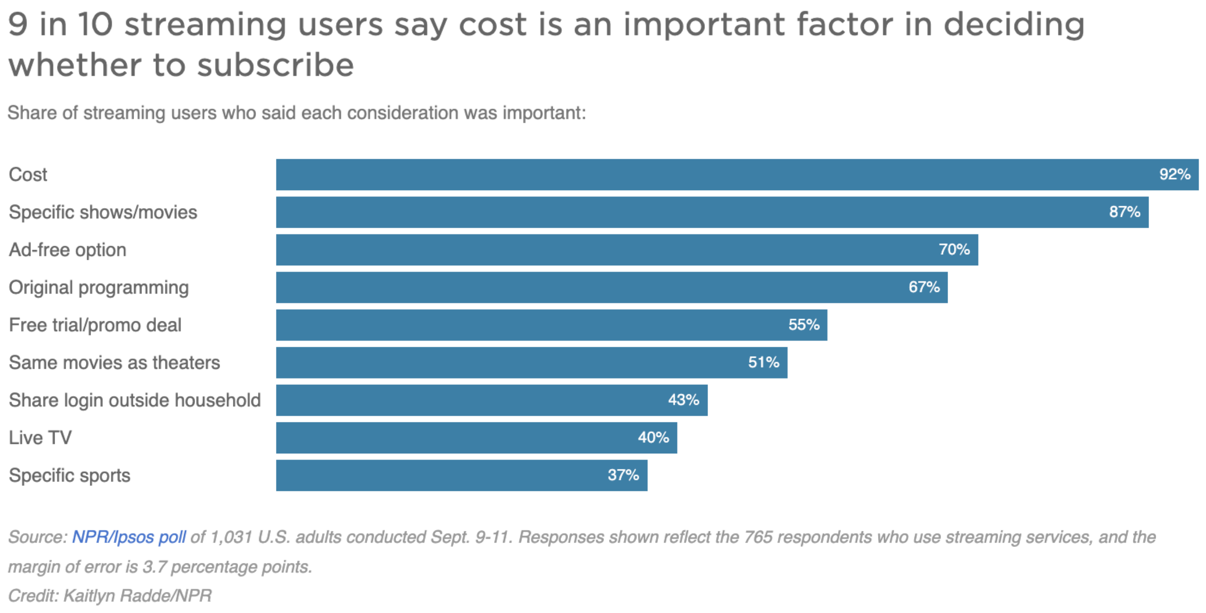


Figure 1. NPR Poll Denoting Most Important Factors When Subscribing to a Streaming Service

It's obvious to note that the most important factor being cost cannot be associated with a recommender system, since prices are fixed and not determined by the user. A recommender system can, however, influence a user's perspective on some of the other factors such as specific movies, original programming, and having the same movies as in theaters. Therefore, it's important that the recommender system is tuned well to the user's preferences so that it can recommend the service's best specific movies, regardless of whether it's original content, as seen in theaters, or neither. By doing so, the streaming service can satisfy users and retain them for longer periods of time, which results in more revenue as well as happy users. We now begin to discuss the data used to build and evaluate the recommender systems, which are being tasked with predicting which unwatched movies a user is most likely to enjoy.

3. MATERIALS / DATA / SOURCES

3.1 Extracting Movie Ratings

We rely on the MovieLens dataset to provide the recommender system with different users' movie ratings, so that it can learn to recommend similar content to a given user after they've watched a movie. We use its latest version that contains a variety of movies from the years 1996 up until 2018 (*Movielens latest datasets* 2021). The dataset consists of around 100k movie ratings from 610 different users (*Movielens latest datasets* 2021). The ratings of the movies are within the ranges of 1 through 5, with 1 representing the least enjoyment rating and 5 representing the most enjoyment (*Movielens latest datasets* 2021). Furthermore, the data is separated into separate csv files: movies.csv, links.csv, ratings.csv, and tags.csv. The movie.csv file contains all the movies represented in the 100k movie ratings, with no duplicate movies being found in the data (*Movielens latest datasets* 2021). The next csv file, links.csv, represents a linking between a given movieID and its IMDB (Internet Movie Database) and TMDB (The Movie Database) identification numbers (*Movielens latest datasets* 2021). This linking allows us to connect to either of the movie content providers API and retrieve additional information about the movie. Ratings.csv will supply us with the ratings from every user while tags.csv denotes one word or a short phrase that the user mentioned in their rating such as "amazing!" or "great movie" (*Movielens latest datasets* 2021). The figure below shows exactly what features each csv file offers.

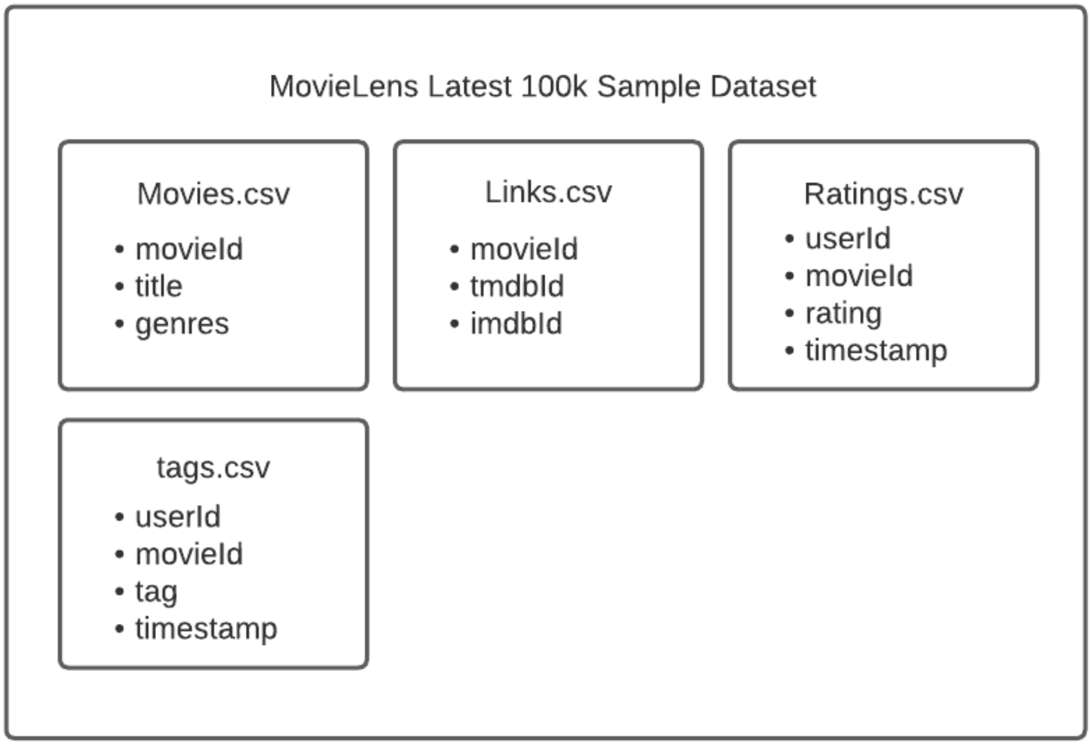
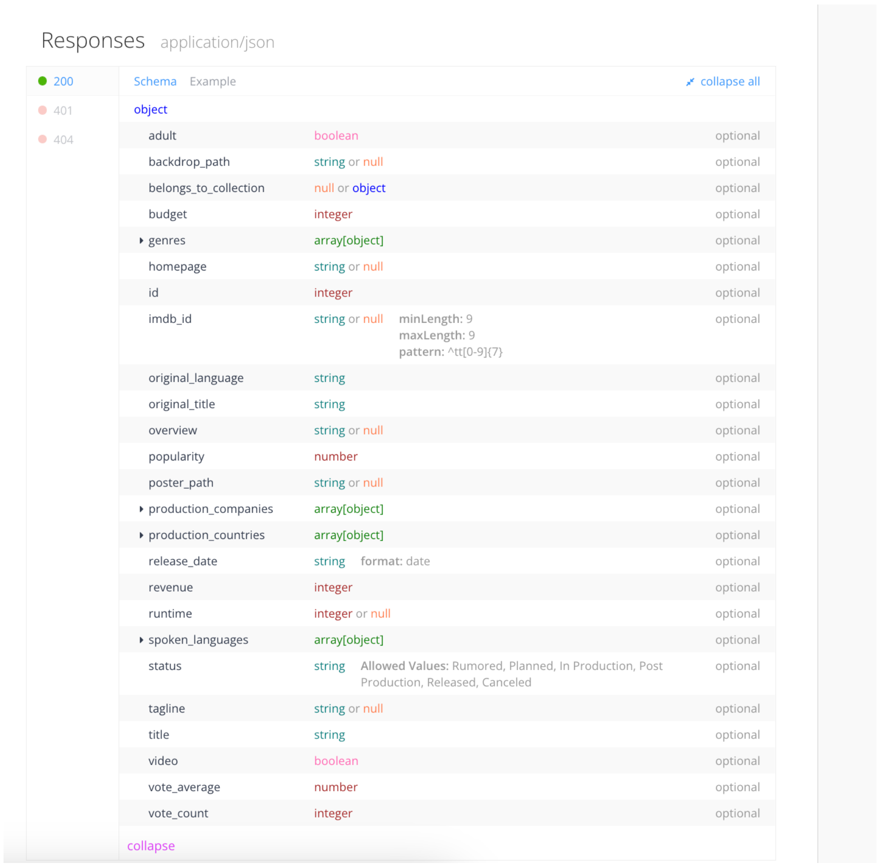


Figure 2. Files and Features Available from MovieLens Dataset

3.2 Extracting Additional Movie Information Using TMDb (The Movie Database) API

The Movie Database (tmdb) consists of a public movie database whose entries have been added/edited since 2008 by the users of their free website (*The Movie Database - About Page*). Using the MovieLens' Links.csv file, we can determine the tmdbId of a given movie and use it to call the tmdb API. The API can provide additional information about a movie that's not available from the MovieLens dataset, such as the movie's production companies, revenue generated, overview (description), etc. We will use this API once we get to the results section to determine the movies which were recommended, and we can use the additional features provided by the API to generate insights on why each movie was recommended in their specific order.

It's important to note that although the movie data is entered by everyday users, it's an extremely credible database given its popularity. According to The Movie Database website, the database is used by millions of users every day and processes over 3 billion requests daily (*The Movie Database - About Page*).



Responses application/json

200 401 404

Schema Example collapse all

object

adult	boolean		optional
backdrop_path	string or null		optional
belongs_to_collection	null or object		optional
budget	integer		optional
genres	array[object]		optional
homepage	string or null		optional
id	integer		optional
imdb_id	string or null	minLength: 9 maxLength: 9 pattern: ^tt[0-9]{7}	optional
original_language	string		optional
original_title	string		optional
overview	string or null		optional
popularity	number		optional
poster_path	string or null		optional
production_companies	array[object]		optional
production_countries	array[object]		optional
release_date	string	format: date	optional
revenue	integer		optional
runtime	integer or null		optional
spoken_languages	array[object]		optional
status	string	Allowed Values: Rumored, Planned, In Production, Post Production, Released, Canceled	optional
tagline	string or null		optional
title	string		optional
video	boolean		optional
vote_average	number		optional
vote_count	integer		optional

collapse

Figure 3. List of Features Provided by The Movie Database API

3.3 Variables

Because the models will be built using the Surprise library, the only variables they will need are the following: userId, movieId, and rating. Using the MovieLens dataset, a pandas DataFrame will be created to store these three variables.

Upon doing analysis of the rating feature, the mean rating score is 3.5 which is a fair average. It shows that most of the ratings were positive but there still are some ratings where the user did not enjoy the movie. However, it's important that the model also has negative ratings supplied to it so that it can fully understand a user's preferences. Using seaborn, we see that the distribution of ratings is mostly normal, with some skewedness to the left because of the number of 4-star reviews. There appears to be no need of normalizing the rating given that the distribution is somewhat normal and that each rating has at least 1,000 reviews.

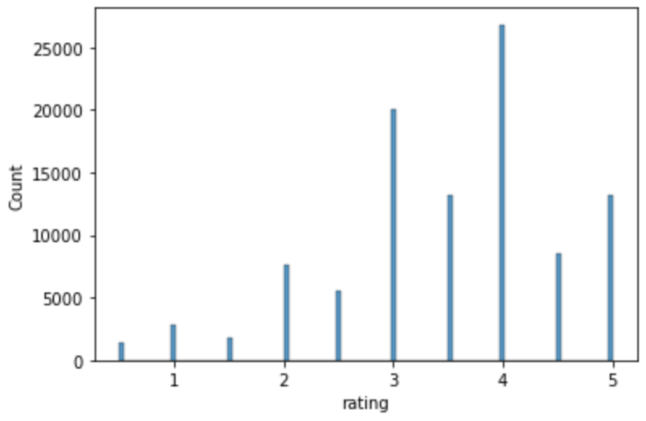


Figure 4. Count of Each Rating

As for the users of the dataset, in total there are 610 users with the minimum number of reviews by a user being 20 and the maximum being 2,698. Using the results from calculating the number of reviews per user, we will predict what the top recommendations are for the user with the most ratings since the models can learn more about their preferences and suggest better movies.

In the MovieLens dataset, there are 9,742 different movies that have been reviewed. The average amount of times a movie has been rated is around 10 times, with "Forrest Gump", a 1994 classic having the highest number of ratings and the "drama" genre having the highest number of ratings compared to the other genres.

4. METHODS

4.1 Predicting Movie Ratings of Unwatched Movies

There are different ways in which the recommender system can filter content for the user: content based and collaborative-based filtering. With content-based filtering, the recommendations supplied to a given user are based on their watching history, not ratings (Jiao, *Ml for Recommendation System: Movie Recommendation and Youtube*). With collaborative based filtering, the content recommended to the user is based on the ratings that similar users gave to those movies that are being recommended (Jiao, *Ml for Recommendation System: Movie Recommendation and Youtube*).

The algorithms that we will be using from the Surprise library will create various collaborative filtering-based prediction algorithms. We will predict what a user will rate an unwatched movie using the following algorithm types provided by Surprise: Basic Algorithms, KNN Algorithms, Matrix Factorization-Based Algorithms, and other algorithms. Within each algorithm type, there are different variations of said algorithm. The table below describes the different algorithm types and their variations.

Table 1. Methods Being Used

Algorithm Type	Variation 1	Variation 2	Variation 3	Variation 4
Basic Algorithms	Normal Predictor	Baseline Only		
k-NN Algorithms	kNN Basic	kNN Means	kNN Z-score	kNN Baseline
Matrix Factorization Based Algorithms	SVD	SVDpp	NMF	
Other Algorithms	Slope One	Co-clustering		

The basic algorithms consist of the normal predictor and the baseline only algorithms. According to Surprise, the normal predictor assigns a random prediction based on the distribution of the training set, which is assumed to be normal (*Using prediction algorithms*). As for the baseline only algorithm, it predicts the baseline estimate for a given user and item (*Using prediction algorithms*).

The k-NN algorithms listed are all derivates of the k-Nearest Neighbors algorithm, in which it looks at the neighbors and counts how many belong to each class and predicts the most common class (Jiao, *Supervised Learning: Classification and Knn*). Each version of the K-NN algorithm takes a different metric or no metric into account. The k-NN with means takes into account the mean ratings of each user when making a prediction (*Using prediction algorithms*). Furthermore, k-NN with Z-score considers the z-score normalization of each user (*Using prediction algorithms*). Lastly, the baseline variation takes into account the baseline rating when making a prediction (*Using prediction algorithms*).

The next class of prediction algorithms are the Matrix Factorization-based algorithms. The first variation is the SVD algorithm, which stands for singular vector decomposition. The next prediction algorithm is SVDpp, which works similarly to SVD but it also considers implicit ratings (*Using prediction algorithms*). The final algorithm of the class is NMF which stands for Non-negative Matrix Factorization.

For the class listed as "Other Algorithms", the algorithms include Slope One and Co-clustering. We did not learn about these algorithms in lecture, but we will still use them in order to have more options for the best prediction algorithm.

4.2 Deciding Which Prediction Algorithm to Use

To determine which prediction algorithm to use for the recommender system, we run cross validation on all the algorithms listed above on our ratings dataset and determine which algorithm reported the lowest RMSE score, given that it's the accuracy metric that we are using since it's the most common metric to use when evaluating a model (Jiao, *Supervised Learning: Regression*).

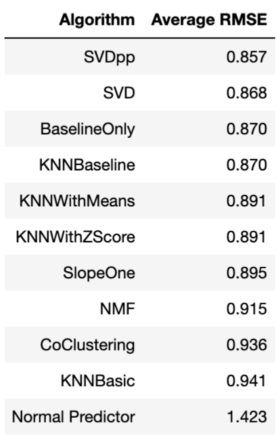
4.3 Building the Recommender System After Making Predictions

Once we've determined which prediction algorithm to use, we will then build a recommender system that relies on that prediction algorithm. The recommender system will start by predicting what ratings a user will give to their unwatched movies using the most accurate prediction algorithm from our tests. After we get our ratings predictions for the unwatched movies of a user, we will then sort them in descending order. The movies at the beginning of the list are the movies we predict that the user will enjoy the most, since the rating predictions are the highest. This approach is inspired by Lab 21 in our course, the movie recommender system lab. (Choi, *Engineering Movie Recommendation System with Filtering. Case Studies of Machine Learning).*

5. RESULTS

5.1 Best Performing Algorithm in terms of RMSE

Using the Surprise library, we were able to benchmark the algorithms listed above and determine the RMSE score of each one when cross validated against our Ratings dataset, with a fold value of 10. The table below ranks the prediction algorithms based on their RMSE scores in ascending order.



Algorithm	Average RMSE
SVDpp	0.857
SVD	0.868
BaselineOnly	0.870
KNNBaseline	0.870
KNNWithMeans	0.891
KNNWithZScore	0.891
SlopeOne	0.895
NMF	0.915
CoClustering	0.936
KNNBasic	0.941
Normal Predictor	1.423

Table 2. Ranked RMSE results for Prediction Algorithms

The prediction algorithm that produced the lowest average RMSE score, after doing cross validation on all 10 folds, was the SVDpp algorithm. Thus, we will build our recommender system by relying on this algorithm.

5.2 Recommending 5 Movies to the User with the Most Ratings

We now recommend 5 movies to the user that has the most ratings in the dataset. Using this user for our evaluation allows the recommender system to better determine which other users are similar, since it can make much more movie rating comparisons, which will result in better recommendations when using collaborative-based filtering. As a reminder, our recommender system works by predicting the ratings a user will give to their unwatched movies and then returning the movies with the highest rating as the recommendations to the user.

The five unwatched movies which had the highest predicted ratings and thus would be the top 5 movie recommendations to the user are listed in the table below, in descending order.

	MovieId	Predicted Rating
0	914.0	5.000
1	1203.0	5.000
2	3030.0	5.000
3	27156.0	4.892
4	7022.0	4.852

Table 3. Top 5 Movies Recommended to the User with the Most Ratings

6. DISCUSSION & CONCLUSION

6.1 Analyzing Results Provided by the Recommender System

The five unwatched movies which had the highest predicted ratings and thus would be the top 5 movie recommendations to the user are listed in the table below, in descending order.

Recommendation #	Movie Title	Translated Movie Title	Genres
1	My Fair Lady	My Fair Lady	Romance, Drama, Music
2	12 Angry Men	12 Angry Men	Drama
3	用心棒	Heart stick	Drama, Thriller
4	新世紀エヴァンゲリオン劇場版 Air / まごころを、君に	Neon Genesis Evangelion the Movie Version Air / Magokoro wo to you	Drama, Animation, Science Fiction, Action
5	バトル・ロワイアル	Battle Royale	Drama, Thriller, Action

Table 4. Titles of Top 5 Movie Recommendations, Including English Translations

When inspecting the top 5 most viewed genres of the user we recommended movies to, we see the results specified in the table below. Looking at the results, each recommended movie had at least one of their most viewed genres, so we can say that the recommendations were reasonable and that it appears the user would be interested in watching these movies.

	Genre	Amount
1	Drama	1309
2	Comedy	1079
3	Action	668
4	Thriller	625
5	Romance	509

Table 5. Users Most Rated Genres

6.2 Conclusion

Overall, the recommender system that we built did a great job recommending relevant movies to the user. The user's most watched genre was "Drama" and each movie that was recommended in their top 5 recommendations had it listed as one of its genres. Furthermore, most of the other genres listed between the movies were also included in the user's top 5 most watched genres. The recommender system was able to successful compare our user to other similar users and determine what to recommend.

However, the performance of the recommender system can be improved by also taking language of the movie into consideration. For example, this user may not have appreciated having three of the five movie recommendations being in another language than English, since they'd have to watch it with subtitles. Therefore, language preferences are an important improvement to make. There are of course many other advanced techniques for building recommender systems that return a much better RMSE than the model we used, which would suggest that they could recommend even better content that fits the user's preferences.

If this recommender system were to be used in the industry for a streaming service, it could also be tweaked so that it can prioritize recommending unwatched movies with the highest predicted rating, that are also originals or movies in theater. In this way we can offer specific content that satisfies some of the areas for what a user looks for in a streaming service, as denoted by the NPR poll in Figure 1.

REFERENCES

1. Bitran, T. (2022, November 2). *Skip intro: Netflix turns 25 Today*. Netflix Tudum. Retrieved 2022, from https://www.netflix.com/tudum/articles/netflix-trivia-25th-anniversary#:~:text=On%20April%2014%2C%201998%2C%20NetFlix,Boss%20Baby%2C%2011%20million%20profiles.

2. Choi, S. J. (n.d.). *Engineering Movie Recommendation System with Filtering*. *Case Studies of Machine Learning*.

3. Cook, S. (2022, February 11). *The complete list of streaming services in 2022 - 200+ services*. Flixed. Retrieved December 3, 2022, from https://flixed.io/complete-list-streaming-services/#:~:text=The%20Complete%20List%20of%20Streaming%20Services%20in%202022%20%E2%80%93%20200%2B%20Services&text=Streaming%20services%20are%20literally%20a,more%20streaming%20services%20than%20that.

4. Deggans, E. (2022, October 28). *What do users want from their TV streaming? A new NPR/Ipsos poll has some answers*. NPR. Retrieved 2022, from https://www.npr.org/2022/10/28/1130669763/what-do-users-want-from-their-tv-streaming-a-new-npr-ipsos-poll-has-some-answers

5. Jiao, J. (n.d.). *Ml for Recommendation System: Movie Recommendation and Youtube*. *Case Studies of Machine Learning*.

6. Jiao, J. (n.d.). *Supervised Learning: Classification and Knn*. *Case Studies of Machine Learning*.

7. Jiao, J. (n.d.). *Supervised Learning: Regression*. *Case Studies of Machine Learning*.

8. *Movielens latest datasets*. GroupLens. (2021, March 2). Retrieved 2022, from https://grouplens.org/datasets/movielens/latest/

9. Pattison, S. (2022, May 29). *35 streaming services statistics you need to know in 2022*. Cloudwards. Retrieved 2022, from https://www.cloudwards.net/streaming-services-statistics/

10. *The Movie Database - About Page*. About TMDB - the movie database (TMDB). (n.d.). Retrieved from https://www.themoviedb.org/about?language=en-US

11. *Using prediction algorithms*. Surprise 1 documentation. (n.d.). Retrieved 2022, from https://surprise.readthedocs.io/en/stable/prediction_algorithms.html