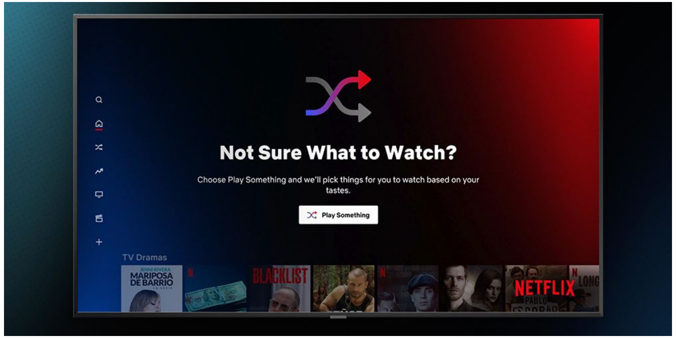
# Movie Recommender Web Application

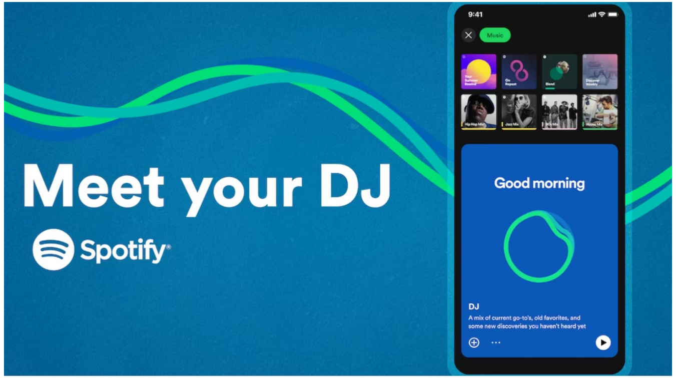
By Marshall Lee
3/21/2023

# Glossary

| Problem Statement | Data Wrangling | Exploratory Data Analysis | Model Building and Evaluation | Conclusion |
| --- | --- | --- | --- | --- |
| What features or techniques make up a quality recommender system? | Clean and standardize our dataset, which contains over 20 million user ratings. | Identify and visualize trends of movies in our dataset using bar graphs, scatter plots, etc. | Create test/training sets and compare the quality of recommendations using three different standardization methods. | Model deployment and final recommendations to stakeholders. |

# Problem Statement




- Recommender systems have become extremely popular in recent years and are what drives the success of companies like Spotify and Netflix

- The purpose of this project is build a movie recommendation website in order to determine which features and techniques are part of a “good” recommender system

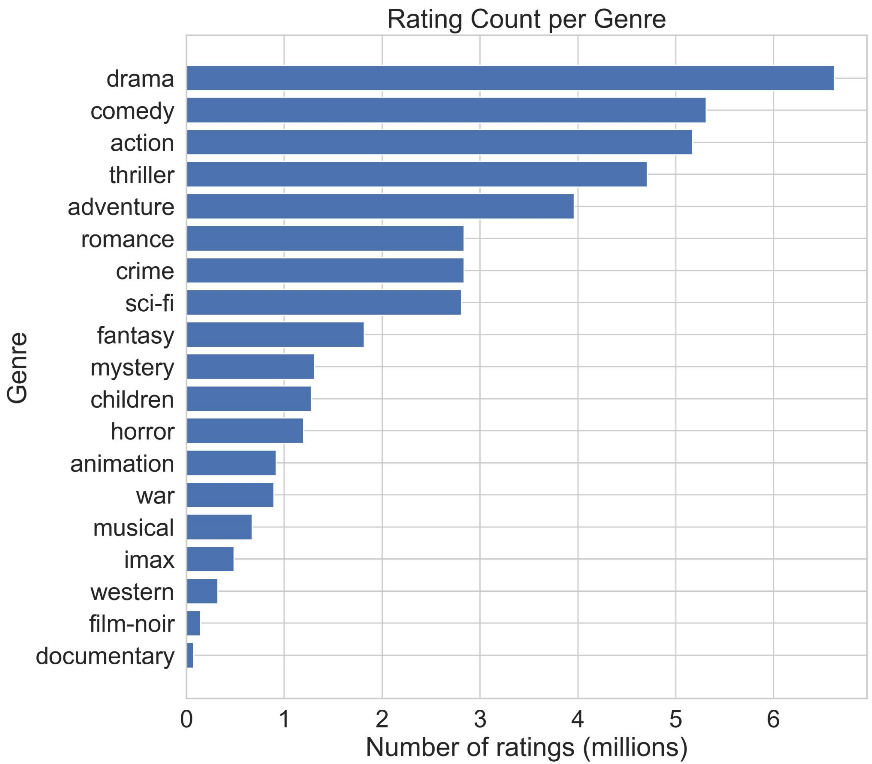
# Step 1: Data Wrangling

# Data Wrangling

- The dataset contains over 20 million user ratings and also includes timestamps, user tags, user Ids, and movie metadata
- Ratings are on a scale of 0.5-5
- Total of around 9000 unique movie titles, 83000 unique users, and 17 million ratings after cleaning
- Source: Movielens 20M dataset

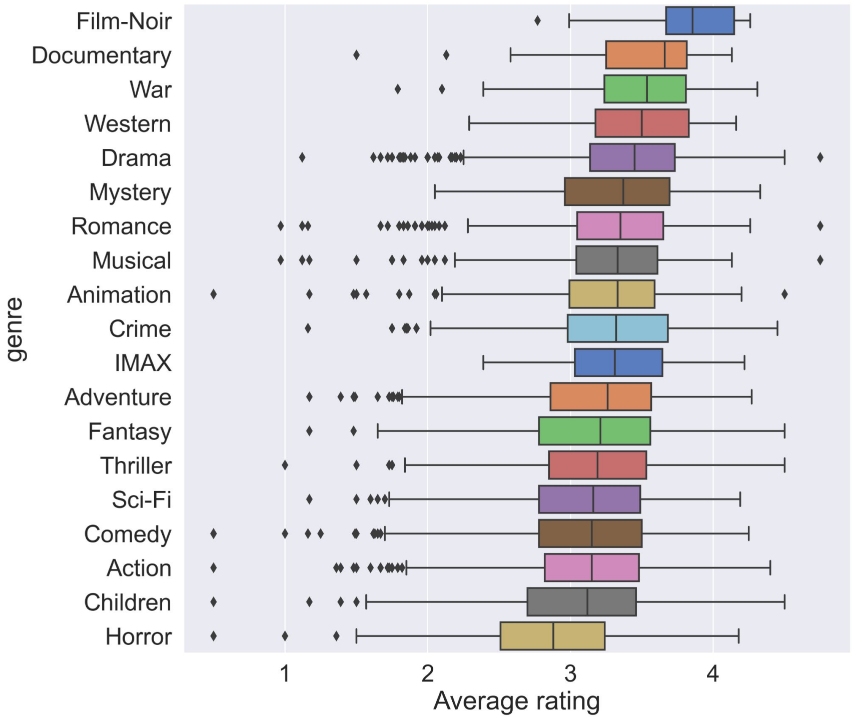
# Step 2: Exploratory Data Analysis

# Popularity by Genre



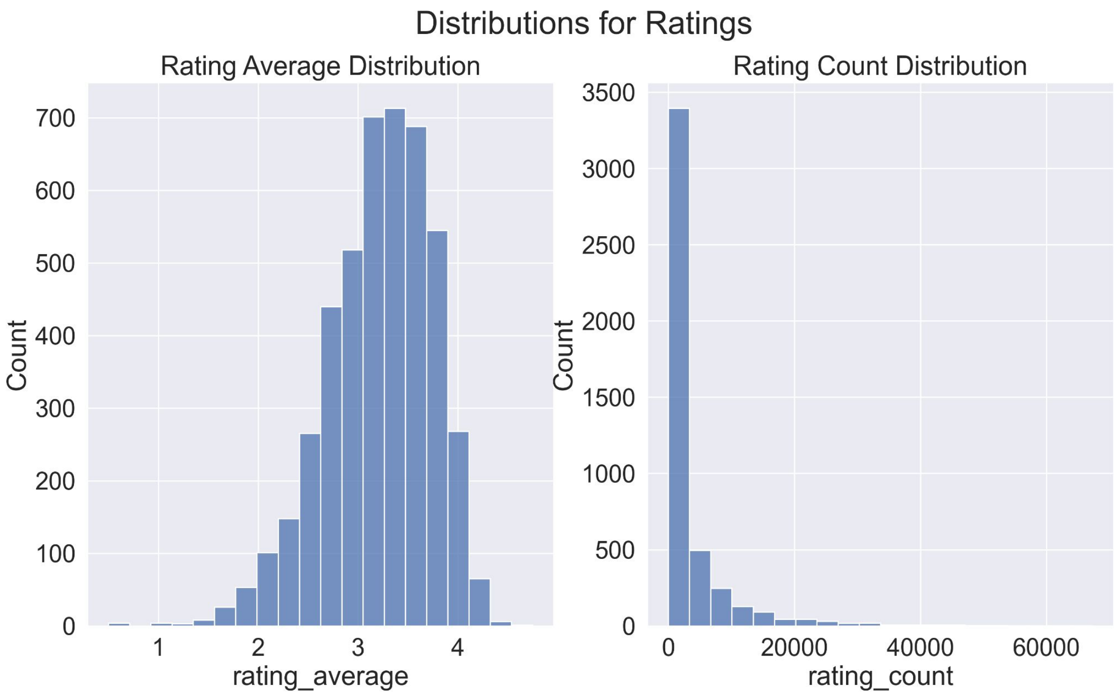

- There are 19 different genres in our dataset
- Each movie is classified as at least one genre (but can be more)
- Drama, comedy, and action are the most popular genres in terms of the number of ratings received

# Average Rating by Genre




- The more popular genres like drama and action actually have a lower average rating than the less-popular genres like Film-noir and Documentaries

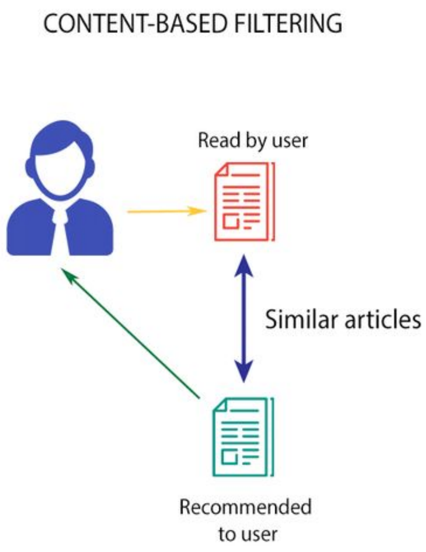
# Distributions for Ratings




- On average, all movies have a rating of 3.5 and have been rated by around 3300 users
- Approximately 0.1% of movies have an average rating of 0.5 or 5
- Only 25% of movies have been rated over 3300 times

# Step 3: Model Building and Evaluation

# Content-based filtering

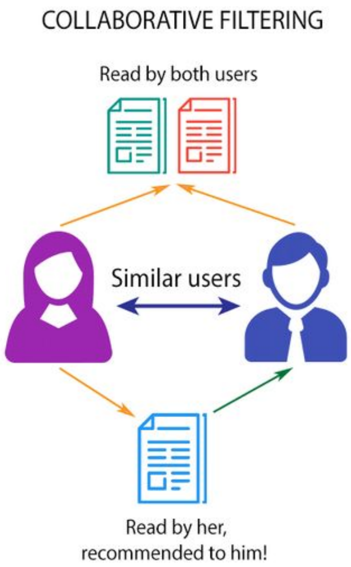



- Recommends items that are similar to other items the user has liked in the past

- **Pros**: Does not require other user ratings to make recommendations

- **Cons**: the cold-start problem (system needs information on user-preferences), limited scope, popularity bias

# Collaborative-based filtering




- Recommends items based on user-item relationships. I.e. two users with similar tastes in music would like the same songs
- Can be broken down into item-based and user-based collaborative filters
- **Pros**: Personalized to user preferences, can handle large datasets
- **Cons**: the cold-start problem, data sparsity, high dimensionality, popularity bias

# Model selection and Evaluation

- I chose to use a collaborative-based filter as my recommender because:
    1. I believed the quality of recommendations to be better than those generated by content-based filtering after experimenting with both methods
    2. Reduces the cold-start problem by giving users a quick questionnaire to ‘learn’ about their preferences
    3. Can take in new user inputs to refine its recommendations
- The model utilizes **Singular Value Decomposition** to deal with the high sparsity (98%) of the dataset and slow computational speeds associated with other methods
- Final model metrics:
    - Latent Factors (k) set to 10
    - RMSE: 1.029
    - Standardized ratings by user with StandardScaler()

# Model Deployment


- How it works:
  - Users are given sets of movies and are asked to choose their favorite
  - “Favorite” movies are given a rating of 5
  - All other movies that were not chosen are given a rating of 0.5
  - Website recommends 10 movies based on user input
- Website link: http://marshalllee.pythonanywhere.com/

# Conclusion

- After experimenting with different methods, collaborative-based filters seemed to be more effective at handling large, sparse datasets while generating quality recommendations
- Suggestions for improvement:
  - Option for users to sign-in with their Netflix or tmdb account to provide their existing preferences
  - Implement a combination of content- and collaborative-based filtering methods to see if the quality of recommendations improves
  - Survey users of the website to give feedback on the quality of recommendations and whether or not they enjoyed the movies recommended
  - Experiment with methods that recommend less-popular movies (movies from the “Long-Tail”)

# References


1. **Dataset -** F. Maxwell Harper and Joseph A. Konstan. *The MovieLens Dataset: 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

2. Thompson, Clive. *If You LIked This, You're Sure to Love That*. November 21, 2008. NY Times

3. Herrada, Oscar Celma. "The Recommendation Problem." *Music Recommendation and Discovery in the Long Tail,* 2008, pp.37

4. Get recommendation function for content filter credited to Ibtesam Ahmed and her post on Kaggle

# Thank you!

Thanks to Ben Bell from Springboard who was my mentor for this project
Full project notebook and report can be found here