Movie Recommendations

A Look at Recommendation Systems and the Netflix Dataset

Abdullah Adlouni
Applied Computer Science
University of Colorado
Boulder, CO
abad2907@colorado.edu

Charles Dorman
Applied Computer Science
University of Colorado
Boulder, CO
chdo1939@colorado.edu

Ben Healy
Applied Computer Science
University of Colorado
Boulder, CO
healybg@colorado.edu

David Shackelford
Applied Computer Science
University of Colorado
Boulder, CO
dash9231@colorado.edu

Problem Statement/Motivation

Video streaming services are a large and growing market in the entertainment industry, and the growing competition in this space is more important than ever. The streaming industry is worth an estimated $60.1 Billion in 2021 and is estimated to be worth over $330 Billion by 2030.[1][2] In addition to this, streaming services have a higher amount of traffic than ever before. 86% of households own at least one video streaming service in the US, with streaming services being accessed by over 110 households in 2021.[3] This large amount of growth is partly attributed to the increasing use of big data and integrating information and statistical analysis with the goal to provide a better service to customers. With more competition than ever, it also becomes increasingly important for a company to use this large amount of data to improve their product. With so many different services offered, it is imperative to keep people watching and engaged instead of moving to a better service. This can be done through data as it can highlight hidden insights within a customer base and provide recommendations to the user on what to watch next, or to provide the company with information to know what types of content are better to invest time and money.

The main purpose of this project is to explore the data that large streaming services use in order to gain some insight into what movies people enjoy watching. To accomplish this, we have 3 main goals. The first goal is to create recommendations based on prior knowledge of customer ratings. To accomplish this goal, we will be looking at a dataset provided by Netflix for competitions. Doing this will allow us to look at recommendation systems further in the context of real data and to provide a baseline for our analysis. The second goal we have is to use other attribute data available to create additional inferences and insights. This will entail getting other resources such as data taken from IMDB. The third goal is to observe trends within the movie industry. This involves taking our conclusions from our previous work and adding better explanations of the movie industry through visualization and further analysis.

KEYWORDS

Big Data, Data Mining, Machine Learning, Movie Reviews, NLP, Recommendation

Reference

Abdullah Adlouni, Charles Dorman, Ben Healy, and David Shackelford. 2022. Movie Recommendations: A Look at Recommendation Systems and the Netflix Dataset.

1 Literature Survey

As mentioned, streaming services are a large and growing industry. As such, there have been many research papers on recommendation systems for movies. One of the biggest pieces of work related to this project is known as the Netflix prize,[7] a $1 million competition issued in 2009, which challenges data scientists to implement a collaborative filtering algorithm using real collected data. The top algorithms in this competition are still used today. The winner of this competition was the Bellkor's Pragmatic Chaos Team, which created a recommendation system that was 10% better than Netflix's existing algorithms by using ensembled learning to create a better recommendation system.[8] The Netflix dataset we are using in this project is the same one used for the competition. Using this work as an example will help us navigate through the project.

Another important piece of work is a paper on improving Netflix data based on outside resources.[9] In this, Bhatia *et al* try to use outside resources such as the IMDB dataset in order to improve on the Netflix prediction score. They chose 3 main attributes to improve their results – Genre, Director, and Actor – which when combined with collaborative Netflix movie ratings gives an RMSE for recommendation of 0.915 which is better than the model error when not including added information (0.867). The team mentioned that they were unable to apply added techniques such as NLP to match titles, and that their implementation could greatly improve without the time constraints.

The changes that we will observe in our project will be two-fold. First, unlike the Netflix prize results, we will utilize multiple datasets in order to improve our recommendation model as much as possible. Second, on top of using other data sources, we will examine additional integration methods, such as NLP, and will also try multiple models in order to maximize our results. We will also utilize data visualization to try and get more useful information from our combined datasets.

2 Proposed Work

Work will be split into 5 parts: data collection, preprocessing, integration, design, and evaluation.

2.1 Data Collection

Each dataset is collected online and uploaded to a public AWS S3 bucket (moviereview.data). This will act as our centralized data lake and ensure that everyone has access to the same datasets. Some datasets are very large and would be inefficient to upload and access all datasets through the github repository.

2.2 Preprocessing

Preprocessing will involve removing any blank values, duplicated values, or wrong values. This includes any custom NULL values or values that do not make sense. As part of the integration step, we also need to remove values which do not contain a matching pair for the other datasets. This is found during integration and should be kept to a minimum in order to retain the dataset's integrity.

2.3 Data Integration

Our team then needs to combine the Netflix, IMDB, and TMDB datasets together by using the title name attribute as a key pair. There are a few options to implement this. One is to purely match the titles. This may be the easiest but will produce the worst results. Another option is to use NLP to vectorize the titles and match them based on cosine similarity. This is reliant on a vectorization model such as Google's word2vec model and will require extra preprocessing such as stemming or lemmatization. Another option is to use an algorithm to match title close-ness. Each method will be tried and compared to get the best results.

2.4 Design

After integration, we will implement a recommendation model. For this, we will leverage multiple modeling methodologies to provide the most stable and consistent model results. We will first create a base model by only using Netflix data. We will then apply other data taken from outside sources to try and improve on the model's score. Since there are multiple methods for building a recommendation system, each team member will be responsible for creating a model and the results of all models will be examined after.

2.5 Evaluation

A summary of model performance will be provided in the final writeup that will discuss the reason for final model selection given a specific prompt. The models will be compared based on RMSE.

3 Dataset

Our project makes use of the Netflix Prize dataset, the IMDb datasets, and the TMDB dataset.

3.1 Netflix

The Netflix dataset is taken from Kaggle (https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data) and is created by Netflix for the use of the Netflix Prize competition.[7] The Netflix Prize dataset is made up of three subsets of data, the training dataset, the probe dataset, and a movie file with movie info. The training dataset contains over 100,000,000 movie ratings from 480,189 users on 17,770 different movies.[4] The data is in the format MovieID: user, rating, date of rating. The movie file contains movie information in the format MovieID, YearOfRelease, and Title. The probe dataset with ratings can be used to train models by reducing the RMSE of predicted ratings against actual ratings.

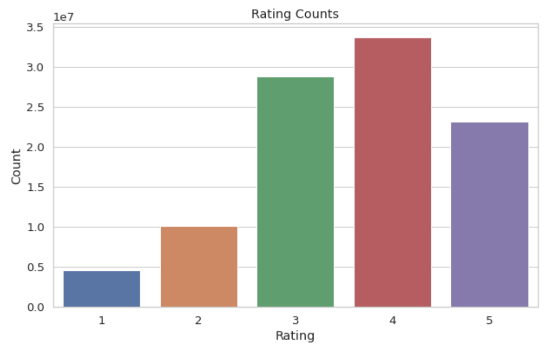

Figure 1: Distribution plot of Ratings for Netflix data.

Figure 1 shows the distribution plot of the ratings for the Netflix data. As shown, ratings are on a scale from 1 to 5 and are skewed slightly left with most ratings around 4.

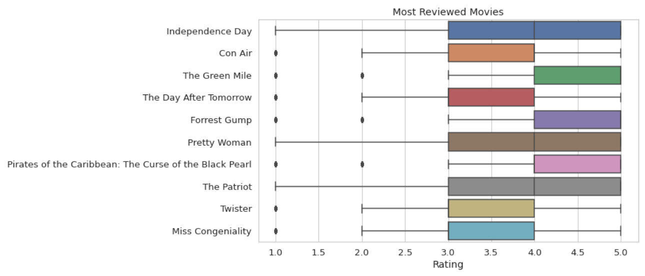

Figure 2: Box plot of the top 10 most reviewed ratings.

Figure 2 shows the top 10 most reviewed ratings. As shown, majority of ratings are high.

3.2 IMDB

The IMDB dataset is taken from IMDB's main website (https://datasets.imdbws.com/) and is a very large set of datasets detailing most movies, tv shows, and specials that have been released throughout history. The IMDB datasets have information on 7,980,307 unique movies, shows, and shorts and 11,906,873 unique cast members.[5] The data sets are broken down into 7 subsets:

1. Title.akas - has regional title name, language, region, and type, with 8 attributes in total.
2. Title.basics – has secondary name, year released, runtime, genre, with 9 attributes total.
3. Title.crew – has info on directors, actors, with 3 attributes total
4. Title.episode - has info on show titles, and episode info with 4 attributes total.
5. Title.principles - has info on the principle cast and crew including ordering and specific role with 6 attributes total.
6. Title.ratings - has IMDB rating and vote information with 3 attributes total.
7. Title.name.basics - has crew information including names, date of birth, date of death, and known-for-titles, with 6 attributes total.

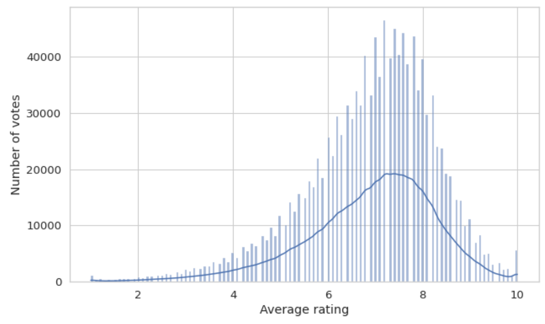

Figure 3: Distribution plot of Ratings for IMDB data.

Figure 3 shows the distribution of ratings for IMDB data. Movie reviews are on a scale from 1 to 10. As shown, the data is skewed slightly left with most reviews around 6-8. This matches closely with the ratings for the IMDB data.

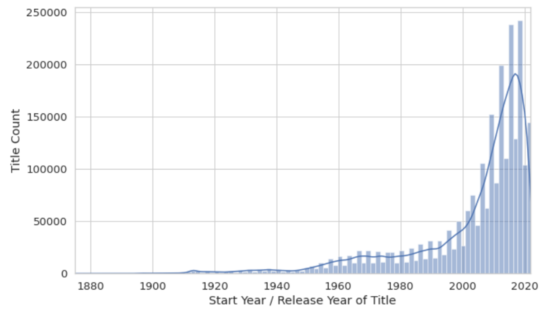

Figure 4: Distribution plot of titles per year for IMDB data.

As shown in Figure 4, majority of titles in the IMDB dataset are within the last 20 years.

3.1 TMDB

The TMDB datasets are a tertiary dataset which consist of added information from the IMDB datasets. This set of data was collected from Kaggle's website (https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata). The TMDB dataset consists of two files with info on 4,800 movies.[6] The credits file contains information in the format columns: movie id, title, cast, crew. The movies file contains information in the format columns: budget, genres, homepage, id, keywords, original language, overview, popularity, production companies, production countries, release date, revenue, runtime, spoken languages, status, tagline, title, vote average, vote count.

4 Evaluation Methods

Evaluation will be used for both data integration and model selection/performance.

4.1 Integration

To test that we were able to successfully integrate both datasets, we will have two metrics. The first metric is a quantitative analysis of the amount of lost data. If we are not able to find a key pair for the datasets, we will drop that object as it will not be useable. Therefore, we want to minimize number of objects that we remove. The second metric is a qualitative analysis of accuracy. Since there is no way to confirm correct matching without going through each title individually, a qualitative analysis will be conducted by taking a random sampling of the resulting title pairs and visually confirm if they are correct. We will use the integration method with the highest qualitative accuracy and the lowest quantitative data loss.

4.2 Data Model

The goal of our data model is to maximize both accuracy and generalization. For this, we will try multiple different models and determine the best choice based on classification metrics. This includes creating a confusion matrix which identifies rates of true-negative (TN), true-positive (TP), false-positive (FP), and false-negative (FN) between actual values and predicted values. Accuracy and RMSE will identify how often our class is correct:

$$accuracy = \frac{TP + TN}{N}$$

Precision will give us an idea of quality in our results:

$$precision = \frac{TP}{TP + FP}$$

Recall will show us our model's ability to find all relevant instances in a class:

$$recall = \frac{TP}{TP + FN}$$

The goal is to maximize the above metrics while also minimizing the RMSE.

5 Tools

Tools are broken up between project management, data analysis, and machine learning.

5.1 Project Management

The group meets in weekly scrum meetings on Thursdays for 30 minutes using Google Meet. Communication is maintained using email and discord. Code and documentation are accumulated using a github repository (github.com/dash2927/ABCD). Each team member works in their own branch with each branch being merged onto a main branch for major milestones and breakthroughs. To maintain consistency and access to data, AWS S3 will be utilized. AWS S3 is a simple storage service which will allow us to culminate datasets. A public bucket (moviereview.data) has been made to access all files. Boto3, a python library, will interface with AWS to retrieve data.

5.2 Data Analysis

Pandas and Numpy are used for major data analysis. These are python libraries which will help us manipulate csv data and perform calculations. The python library Seaborn will be utilized for data visualization. Regex (regular expression) is used to quickly search for text patterns to facilitate the merging of databases, specifically to overcome differences in movie titles between databases. Bash scripting may be used at a later point in the project for pipeline data manipulation, e.g., to assess the model across various metrics.

5.3 Machine Learning

Integration will require NLP methods to match titles from the IMDB dataset to titles from the Netflix dataset. There are multiple ways to perform NLP on the dataset. The main goal is to vectorize each title and then get a similarity score to find which title is the closest. Due to scope of this project, we will use a pretrained model to compute cosine similarities between titles. Bert-as-a-service is a sentence encoding API which allows us to encode a variable-length "sentence" (movie title) to a fixed-length vector. Doing so will allow us to compare the titles of one dataset to the titles of another based on a normalized score. To run this, python 3.6 is needed with tensorflow 1.10. We are using Google's multi-lingual cased bert model to vectorize titles, which is a 12-layer neural network with 768 hidden layers, 12-heads, and 110M parameters.[10][11] Scoring is based on cosine similarity and is weighted based on the difference between the years created of each key-pair, with absolute years > 4 being highly penalized.

There are a few ways we can build a recommendation system. Unfortunately, since the dataset we are using is so large, it is extremely difficult to do anything substantial without a high computation and time cost. For example, a very common step in recommendation systems is to create a sparsity matrix. If we did not make any adjustments to our dataset, we would need a sparsity matrix which contained information for 17,770 unique movies and 480,189 unique customers with 100,480,507 ratings total. This would amount to a sparsity matrix with 8,532,958,530 cells in total - something that would take too much time and too much computational resources.

One way to get past this is through "on-the-fly" collaborative filtering. Collaborative filtering is a technique which uses only information for rating profiles from different users or items.[12] In order to do this, they locate similar users or items that are in the nearest neighborhood and generate recommendations based on the information given.

There are two major types of collaborative filtering - user based filtering and item-based filtering. In user-based filtering, we look for similar users and base the decision making on the choices made by the set of users. In item-based collaboration, we look for similar items based on the items that have already been chosen. A possible path is instead of calculating everything at once, we can calculate recommendations based on who the customer is and compare it with customers that are similar. This greatly cuts down computation by eliminating datapoints which do not matter. The issue with this is that since we are only computing recommendations for single customers in the moment, and because we are only looking at correlation values, it becomes difficult to produce any type of metric.

While the above implementation is very useful, it does not allow us to get any type of metric to evaluate the effectiveness of our recommendation. An alternative implementation would be to use matrix factorization using the SVD algorithm. This algorithm was popularized by Simon Funk during the Netflix Prize competition.[13] This can be accomplished through the surprise library.[14]

We will try to implement both of the described methods and review which implementation might be better.

6 Milestones

Milestones for this project are based on a final submission of Dec. 8th.

6.1 Integration – Nov. 3rd

Find best method to integrate data sources.

6.2 Preprocessing – Nov. 10th

Full EDA should be performed on data. This includes cleaning and preprocessing data. Preliminary data visualizations should be finished in order to help build research story.

6.3 Model Building – Nov. 21st

Create models based on individual team member's subtasks. Generate model suggestions and compare to pick the best model.

6.4 Visualizations – Nov. 25th

Create visualizations that support findings.

6.5 Finalization – Dec. 5th

Piece together all analysis for final presentation and report.

7 Milestones Completed

7.1 Integration

Completed. We were able to go through each method and we determined that NLP using vectorization and cosine similarity was the best choice to match titles. Datasets were then integrated together using the key pairs.

7.2 Preprocessing

Completed. EDAs of all datasets involved were completed. Visualizations and analysis of the data were created to better understand the movie and streaming industry.

7.3 Model Building

Partially completed. Model for the Netflix-only data has been completed with the ability to make recommendations based on collaborative filtering. Model for integrated data still needs to be created but is proving difficult due to the amount of data to process.

7.4 Visualizations

Partially completed. Most visualizations are completed which go into depth and explain trends in the movie industry. This analysis will continue to be ongoing for the next week as we are exploring new areas.

8 Milestones Todo

8.1 Model Building

Still need to create model for integration dataset. This is proving hard due to the size of the data. For example, since each movie can have multiple genres, this will multiply the amount of datapoints of an already large dataset (the Netflix dataset is >100,000,000 objects equalling to about 5Gb of memory and can take some calculations days to complete.). Effort is going in to reduce the data as much as possible and we are currently looking into other solutions such as only using a sample of the data.

8.2 Visualizations

We have already completed most visualizations needed, however this is a flexible milestone in that we are continuously looking for new areas within the data to build on our analysis.

8.3 Finalization

We still need to piece everything together. As we are each working on our own github branch, we need to combine notebooks and analysis into a final report.

9 Results

9.1 Data Cleaning

Data Cleaning was performed on all datasets depending on the goal trying to be achieved. One of the primary purposes for data cleaning was to reduce the size of the datasets in order to be more manageable. Many calculations during the course of this project took a high number of computational resources and time.

9.1.1 Netflix Data. For the Netflix dataset, we started with 100,480,507 total reviews. We first removed movies that were probably not very popular in the data - these are any movies which have a low number of total reviews. Since we are looking to recommend movies, we probably do not really care to recommend a movie with low popularity. We chose to remove any movies that had a fewer number of reviews than the 70th percentile of all reviews. This was about 12,438 movies. We then removed any customers who are not very active in their reviews. If a customer does not review often, then they will most likely be less trustworthy with giving reviews. For this, we used the same threshold as when reducing movies. This was about 335,809 customers. After this, we looked at contextual outliers for each movie. This was any outlier in reviews that were unusual for that movie. In total, there were 2,914,484 reviews which were considered contextual outliers. In total, this allowed us to remove 30,612,243 reviews which was approximately 30% of the original dataset.

9.1.2 IMDB Data. IMDB data was much easier to clean as they smaller sets of data. We first used regex pattern matching to get only interesting media types for analysis. Specifically, these were movie, short, tvSeries, tvMovie, tvSpecial, and tvMiniSeries. After integrating the needed datasets and removing null values, we had to separate genres into their respective attributes for analysis. For this, we looked at frequency of each genre and set a confidence value of 10% of the data. This removed genres for 'News' and 'Film-Noir'.

9.2 Integration

As discussed previously, data integration was especially difficult as not all titles in the IMDB dataset were equivalent to the titles in the Netflix dataset. This results in mismatched and unmatched pairs when trying to combine datasets.

	Mismatched Pairs (/100)	Unmatched Pairs	Time
(1)	0	11,213	15m
(2)	6	6,516	3d 8h
(3)	23	5,731	4h

Table 1: Table showing integration performance of 1:1 matching (1), title vectorization and cosine similarity (2), and difflib title closeness (3).

The methods implemented were to attempt to match each title 1:1 (1), vectorize the titles and compare cosine similarities (2), and using the sequence matcher algorithm from the python library, difflib, to measure title closeness (3). From these options, vectorizing and computing cosine similarity was the best choice. While difflib had more matched pairs, the quality was not as good as the other options with 23/100 randomly sampled key pairs being mismatched. Likewise, while matching 1:1 had 0 mismatched pairs, the method also had the most data lost with 11,213 titles having an unmatched pair. Using vectorization and cosine similarity was the best choice with only 6,516 unmatched pairs and 6/100 randomly sampled pairs being mismatched. A table of these comparisons is shown above.

9.3 Modeling

Models for both Netflix only data and the integrated data were created using the python library, surprise, which included SVD based algorithms for recommendation systems.

9.3.1 Netflix Based Model. The SVD model was created using cross validation and was trained using a 60/40 split in the ratings dataset. The results to this model are as follows:

Fold	RMSE	MAE	Fit Time (s)	Test Time (s)
1	0.744	0.586	1243.673	497.457
2	0.745	0.587	1166.172	245.214
3	0.745	0.587	971.735	193.136
4	0.745	0.587	867.321	214.184
5	0.744	0.587	937.718	210.585
Avg.	0.745	0.587	1037.324	272.115

Table 2: Table showing recommendation model results.

The error in the model was fairly high with a rmse of 0.74. This may be attributed to the reduction in data during cleaning. Since we reduced the data by 30%, we have removed important information.
In addition to the SVD model, due to the large size of the dataset, we also implemented a recommendation system which calculated recommended movie on-the-fly. This model can be found on the github Netflix Recommendation notebook.

9.4 Data Exploration

Looking at our data, we were able to find some trends for both the IMDB and Netflix datasets.

9.4.1 IMDB Dataset. From the IMDB dataset we

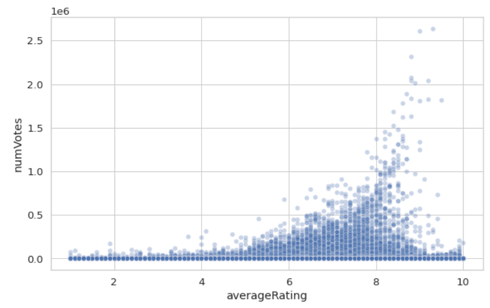

Figure 5: Figure showing scatter plot of average movie rating with respect to the number of votes.

From the scatter plot in Figure 5, we can see that many of the highly voted movies tend to have a higher rating. This makes sense as more popular movies will get more votes and will therefore have a higher movie rating. While most of the data is below 500,000 votes, all votes that larger than this value are considered at least 6/10 movies by the reviewers. This gives credence to the idea that popularity and the spread of awareness for a movie may have some impact on the rating (or vice-versa).

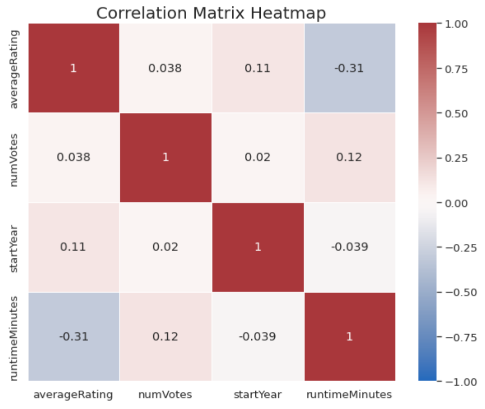

Figure 6: Figure showing Pearson correlation of movie runtime, movie release year, number of votes, and average rating

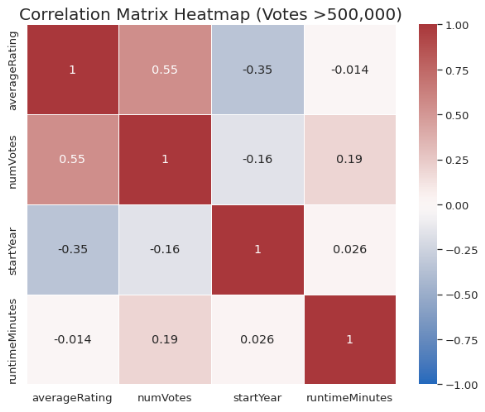

Figure 7: Figure showing Pearson correlation with movies that have more than 500,000 votes

From the correlation matrix on Figure 6, we can see fairly low correlations throughout. Again, the positively correlated trend that can be seen in Figure 5 is mostly only for movies that get greater than 500,000 reviews. One attribute in the heatmap that is interesting is with runtime. There is a slight negative correlation on the rating for a movie with respect to the number of minutes that the movie lasts. Interestingly, when looking at the correlation of movies whose votes exceed 500,000 reviews as shown in Figure 7, this correlation no longer exists. This suggests that movies may not be penalized as severely for a long runtime if they are popular and well known. Alternatively, there is now a slightly negative correlation for startYear. This suggests that older, more popular and voted movies may show better reviews than newer movies.

From the IMDB data, we were also able to look at some trends in the type of media being shown and the media genre.

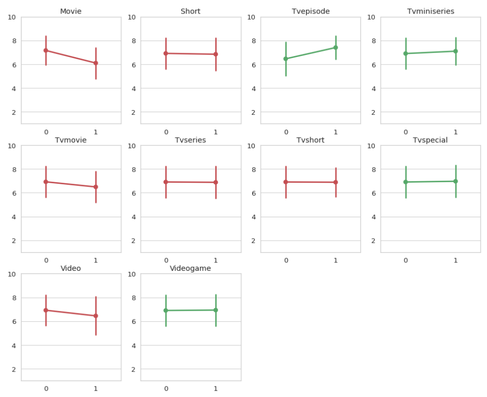

Figure 8: Figure showing ratings of each type of media based on total average rating of data without the media type (0) vs the average rating of the media type (1). Red is reduced rating and green is increased rating.

From Figure 8, we can see that most media types review about the same as the global average movies reviews. Movies and tv shows show a slight trend. Movies tend to review lower than the total average reviews while tv shows tend to review higher. This is about 1/10 of a score higher and lower difference. This may suggest that movies tend to be scored slightly harsher than average while tv shows are scored slightly more gently.

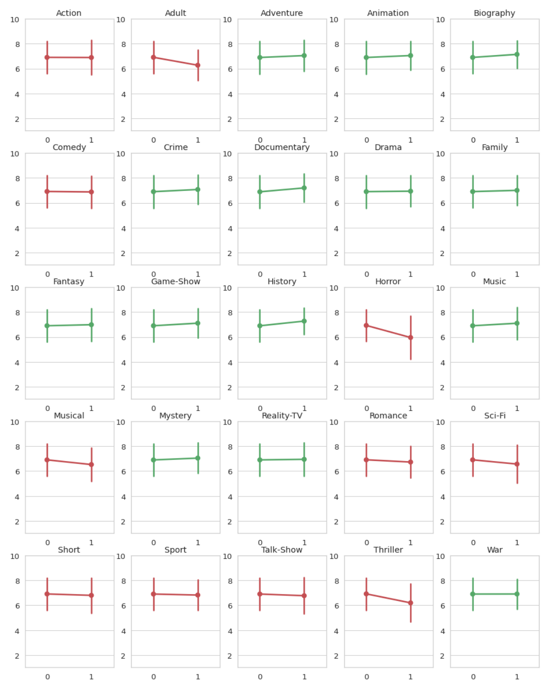

Figure 9: Figure showing ratings of each genre based on total average rating of data without the genre (0) vs the average rating of the genre (1). Red is reduced rating and green is increased rating.

From Figure 9, we can see trends for each type of genre. Some interesting attributes are with Horror, Thriller, and Sci-Fi media which tend to review lower than the other data.

REFERENCES

[1] https://www.grandviewresearch.com/press-release/global-video-streaming-market
[2] https://www.blueweaveconsulting.com/report/video-streaming-market
[3] https://www.kantar.com/north-america/inspiration/technology/us-video-streaming-market-growth-stalls
[4] https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data
[5] https://datasets.imdbws.com/
[6] https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
[7] https://journals.sagepub.com/doi/full/10.1177/1461444814538646#bibr66-1461444814538646
[8] Jahrer, Michael & Töscher, Andreas & Legenstein, Robert. (2010). Combining predictions for accurate recommender systems. 693-702. 10.1145/1835804.1835893.
[9] Bhatia, N., & Patnaik, P. (2008). Netflix Recommendation based on IMDB.
[10] https://github.com/google-research/bert#sentence-and-sentence-pair-classification-tasks
[11] https://bert-as-service.readthedocs.io/en/latest/index.html
[12] https://medium.com/codex/hybrid-recommender-system-netflix-prize-dataset-e9f6b4a875aa
[13] https://sifter.org/~simon/journal/20061211.html
[14] https://surprise.readthedocs.io/en/stable/index.html