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

Abstract

Netflix, IMDB, and TMDB data was analyzed and combined to try and provide more insight into what type of trends are present in the movie industry and to try and see if movie recommendations can be improved with added information. One of the major questions we sought to answer was if we can make a recommendation system based on prior customer ratings. From the data, we were able to achieve this using an SVD based recommendation model. Our error for this model was found to have a RMSE of 0.825 which was equivalent to other studies. We then tried to improve on this through integration of IMDB genre data and ensemble recommendation modeling. With the added data, we were able to achieve an improved RMSE of 0.749. Following this, we wanted to observe additional insights into our data in order to see different trends in the movie industry. We were able to clean and observe IMDB data and found genre trends related to release date with rating. We were also able to observe strong correlations between movie budget and revenue when compared to popularity while also observing how this correlation does not hold for rating. This gave extra insights on how budget and revenue of media may affect success of a piece of media differently.

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 Introduction

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.

2 Related Work

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 higher 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 which may have resulted in their worse score, 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 Methodology

Work for this project is broken up 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 involves 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 is kept to a minimum in order to retain the dataset's integrity.

2.3 Data Integration

In order to get more useful information from the Netflix dataset, datasets from the Netflix, IMDB, and TMDB datasets are integrated 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 pretrained bert model which vectorizes each title as a sentence. This is beneficial as it takes into consideration word ordering as well as what the word is. Another option is to use an algorithm to match title close-ness. This can be implemented through the difflib python library. Each method will be tried and compared to get the best results.

2.4 Design

After integration, we implemented a recommendation model through a singular value decomposition (SVD) factorization-based model. For this, we leveraged multiple modeling methodologies to provide the most stable and consistent model results. We first created a base model by only using Netflix data. We then applied other data taken from outside sources and integration in order to try and improve on the model's score. Due to time and computation constraints, we only looked at two different models – movie id and genre. These were ensembled in a very simplistic manner by averaging the predicted ratings.

3 Dataset

The 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.

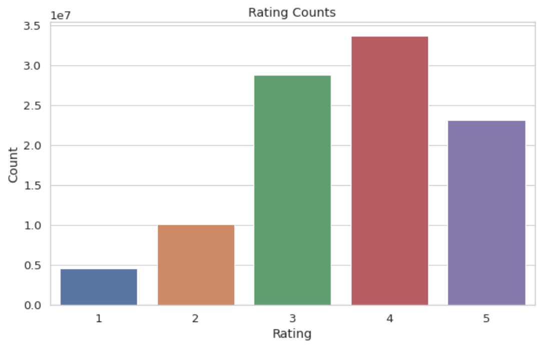

Figure 1: Distribution plot of Ratings for Netflix data.

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.

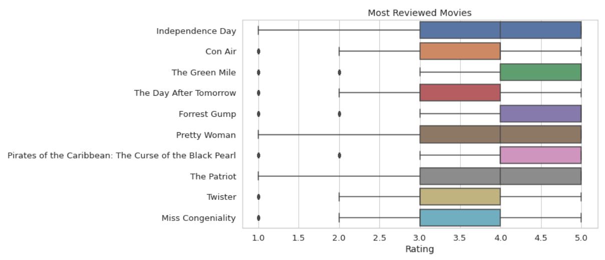

Figure 2: Box plot of the top 10 most reviewed ratings.

Figure 2 shows the top 10 most reviewed ratings. As shown, the 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.

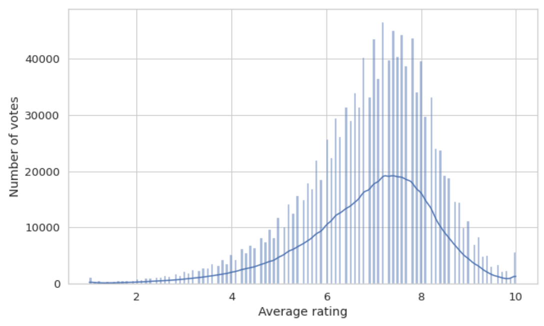

Figure 3: Distribution plot of Ratings for IMDB data.

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.

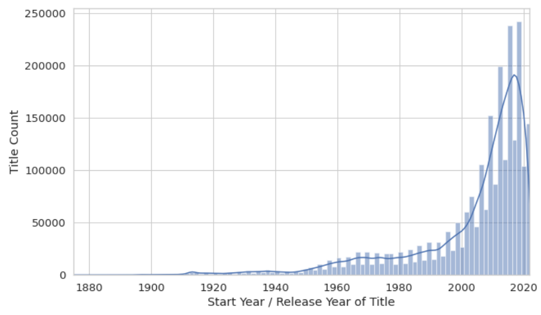

Figure 4: Distribution plot of titles per year for IMDB data.

As shown in Figure 4, the 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. The TMDB dataset is particularly useful because it contains data not found in the Netflix or IMDB datasets, such as movie budget and movie revenue.

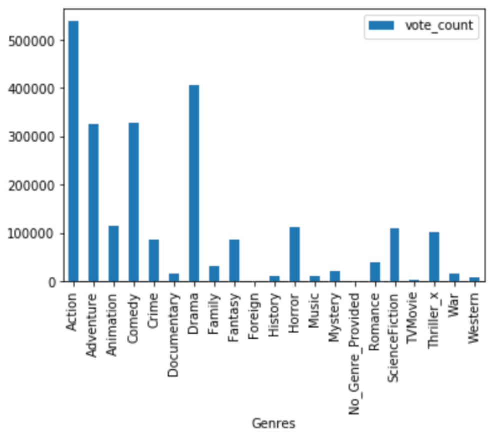

Figure 5: Plot of genres vs vote count for TMDB data.

Figure 5, above, shows that the four genres with the most titles rated are action, drama comedy, and adventure.

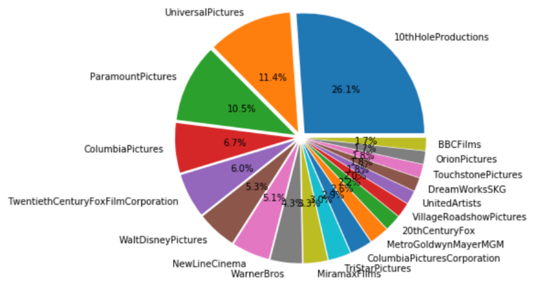

Figure 6: Pie chart showing share of movies by production company in the TMDB dataset.

Figure 6 above shows that 10th Hole Productions created the most titles in the TMDB dataset. 10th Hole Productions is an independent movie studio that tends to put out large numbers of lesser-known movies. More well-known companies like Universal, Walt Disney, MGM, etc. tend to put out fewer movies that reach a broader audience.

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 datasets, we look at three 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 100 resulting title pairs and visually confirm if they are correct. The third is a time-based metric. This is how much time it takes to compute each method. We will use the integration method with the highest qualitative accuracy and the lowest quantitative data loss and consider how much time each method takes to complete.

4.2 Data Model

The goal of our recommendation model is to maximize both accuracy and generalization. Our models will predict the rating of each movie based on the user id and other information given to it (either the movie id or the genre of a movie). For this, we will try to minimize error in our prediction. To do this, we will look at root mean square error (RMSE) and mean absolute error (MAE). RMSE is the standard deviation of the residuals in a model. This is how far from the regression line prediction the true data points are and will show how well our estimate fits the data. This is measured by:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}$$

where $\hat{y}_t$ is the predicted value. Likewise, MAE is the measurement of error between observations expressing the same phenomena and is calculated to be sum of absolute errors over the size of the sample.[15][16] The equation to this is as follows:

$$MSE = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n}$$

where y_i is the prediction value and x_i is the actual value.

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 is utilized as a data pooling method. AWS S3 is a simple storage service which allow us to culminate datasets. A public bucket (moviereview.data) has been made to access all files. Boto3, a python library, interface with AWS to retrieve data.

5.2 Data Analysis

Pandas and Numpy are used for major data analysis. These are python libraries which help us manipulate csv data and perform calculations. The python library Seaborn is 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.

5.3 Machine Learning Methods

Integration requires 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 an absolute year difference > 4 being highly penalized. This threshold is created to avoid matching rebooted movies that may have the same or similar title.

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 without having to compute many computed ratings. 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]

6 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 in this project took a high number of computational resources and long computation times so it was imperative to reduce this data as much as possible while minimizing loss of information.

6.1.1 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. The reasoning for this was that 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. The reasoning of this was 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. We used 1.5*IQR to represent the threshold for an outlier. 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.

6.1.2 IMDB Data. IMDB data was much easier to clean as they are 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. For some analysis and for the recommendation model, we had to separate genres into their respective attributes for analysis. Genres in the IMDB dataset came in a list for each movie and needed to be parsed out so that there was one object for each genre in each movie. This exponentially increased our data and so we looked at a simple version of apriori algorithm to remove infrequent genres. For this, we looked at frequency of each genre and set a confidence value of 10% of the data. This removed genres for 'News', 'Game-Show', 'Talk-Show', 'Adult', and 'Film-Noir'.

7 Results

7.1 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 chosen as 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. In terms of time, vectorization took the longest to compute at 3.5 days. This wasn't a huge issue, however, because movie title vectors were able to be saved and reused for later. Results of method comparisons is shown in Table 1 above.

7.2 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.

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

Fold	RMSE	MAE	Fit Time (s)	Test Time (s)
1	0.825	0.703	1045.862	432.187
2	0.824	0.701	1072.961	308.564
3	0.824	0.701	958.641	203.421
4	0.824	0.701	897.596	198.323
5	0.824	0.701	932.118	221.168
Avg.	0.824	0.701	981.436	272.730

Table 2: Table showing recommendation model results of movie titles.

The error in the model was slightly higher than literature values with an RMSE of 0.824. This may be attributed to the reduction in data during cleaning. Since we reduced the data by 30%, we may have removed important information.

7.2.2 Genre Based Model. A model was then created using genre types of the movie. After data integration, genres for each movie were encoded into the Netflix dataset. Like the previous model, a 60/40 split was created, and the model was cross-validated with 5 folds. The test results to this are as follows:

Fold	RMSE	MAE	Fit Time (s)	Test Time (s)
1	0.888	0.719	600.917	202.503
2	0.888	0.718	603.431	137.481
3	0.888	0.718	602.818	128.296
4	0.888	0.718	603.871	203.589
5	0.888	0.717	599.386	120.589
Avg.	0.888	0.718	602.085	158.491

Table 3: Table showing recommendation model results of genre type.

As shown, the error in the model is about the same as the previous model with an RMSE of 0.888 and a MSE of about 0.718.

7.2.3 Ensembled Model. The models were then ensembled together. Due to time and computation constraints, this was a simple combination of model results in which we averaged the predicted ratings together.

	RMSE	MAE
Movie ID	0.824	0.701
Genre	0.888	0.718
Ensembled	0.749	0.603

Table 4: Table showing recommendation model comparisons between movie id, genre, and ensembled recommendation.

From Table 4 above, we can see that by adding more information and ensembling models together, we were able to achieve a better model with a lower error.

7.2.4 Collaborative Filtering. 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. An example of this in use can be found on the github Netflix Recommendation notebook.

7.3 Data Exploration

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

7.3.1 IMDB Dataset. From the IMDB dataset we were able to look at trending movie data for 8 million movies, tv shows, and other media.

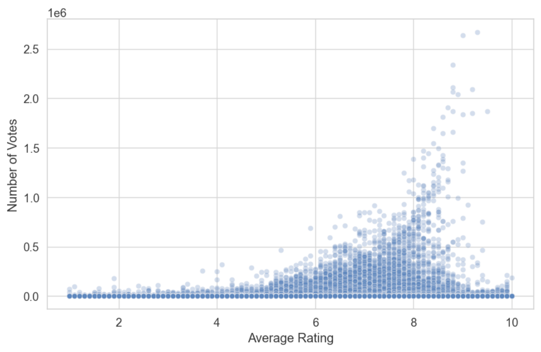

Figure 7: Figure showing scatter plot of average movie rating with respect to the number of votes.

From the scatter plot in Figure 7, we can see the average rating of a particular piece of media a scale of 1-10 with respect to the number of votes a movie received. With this figure, we can visually see a positive trend between number of votes and average rating. This makes sense as more popular movies will get more votes and will therefore have a higher media rating. While most of the data is below 500,000 votes, all votes that are larger than this value are considered at least 6/10 rated 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).

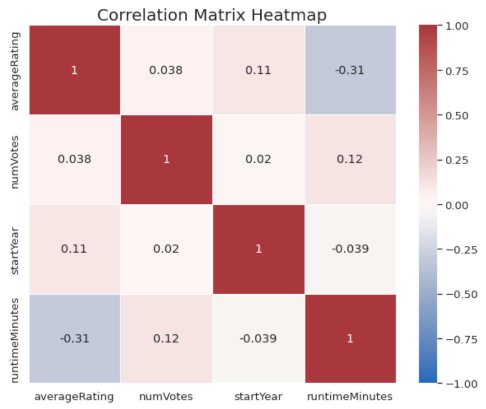

Figure 8: Figure showing Pearson correlation of movie runtime, movie release year, number of votes, and average rating

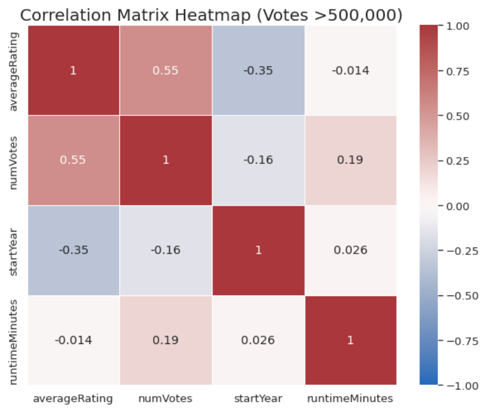

Figure 9: Figure showing Pearson correlation with movies that have more than 500,000 votes

Pearson correlation of the continuous variables was taken from the IMDB data. The results of these can be seen on the correlation matrix in Figure 8. From the correlation matrix, we can see fairly low correlations throughout. While a positive trend can be seen in Figure 7, the correlation between the two is only about 0.038. This is mostly due to so many movies having little to no votes and are generally very unpopular which brings the correlation down. Again, the positively trend that can be seen in Figure 7 is mostly only for movies that get greater than 500,000 reviews. Moving forward, we decided to isolate and observe only media that are above 500,000 votes in order to remove unpopular or unknown media and to achieve better trends. One attribute in the heatmap on Figure 8 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.

As stated previously, we decided to isolate and look at only media that have a higher vote count that is greater than 500,000 votes. Interestingly, when looking at the Pearson correlation of this isolated data as shown in Figure 7, a few correlations have changed. Runtime is no longer slightly negatively correlated with average rating as it was when observing the entire dataset. 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.

Due to the slight trend in movie year, we decided to look at different attributes with respect to release year to get a better idea of the data.

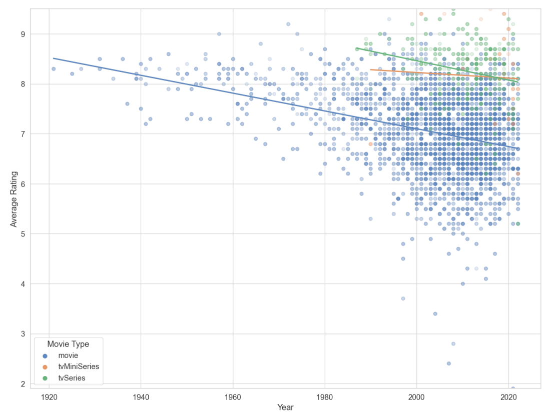

Figure 10: Figure showing average rating vs. year released and color encoded with the type of media.

Figure 10 above shows media that have greater than 500,000 votes. These are color encoded with the media type and contain a regression line. From this data, we can see the slightly negative correlation of rating with respect to year that was shown in Figure 9. In addition to this we can also see how variation of movie rating tends to increase with newer media. We can also see that movies tend to be consistently rated lower than television series.

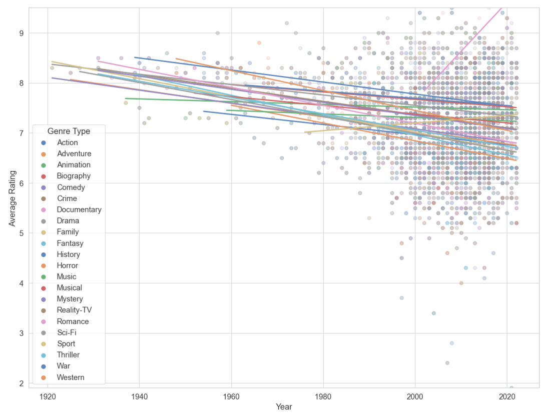

Figure 11: Figure showing average rating vs. year released and color encoded with genre.

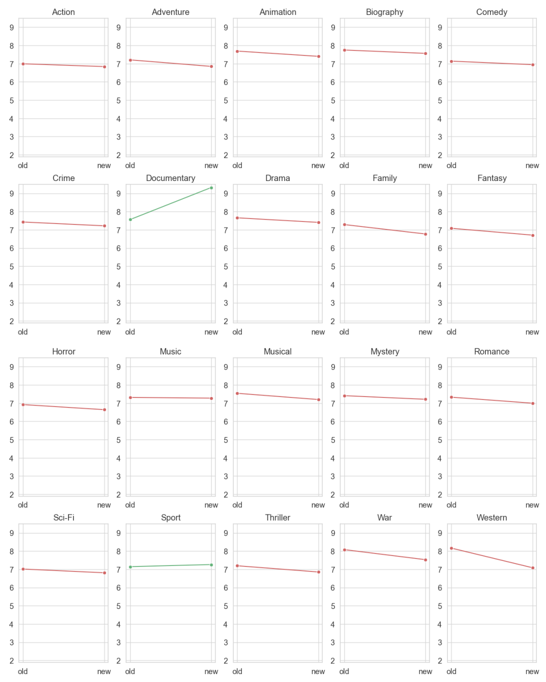

Figure 12: Figure showing average ratings of old (before 2005) and new (after 2005) media with different genres and color encoded based on positive or negative slope.

Continuing with the previous insight, we also encoded for genre as shown in Figure 11 above. Majority of genres follow a similar trend with respect to year except for two genres: Documentary and Sports. These trends can be better compared in Figure 12 in which data are categorized into ‘old’ and ‘new’ where the threshold between the two is 2005. These trends are also color encoded based on the trend slope. As shown by the figure, majority of genres have a slightly negative slope except for Documentary and Sport where Documentaries have increased their rating by almost 2 in recent years. Another interesting thing to note is with Westerns. Westerns do markedly worse than any of the other genres which coincides with the idea that Westerns have generally performed worse over the course of the past century.

7.3.2 TMDB Dataset. The TMDB dataset is very similar to the IMDB dataset but contains budget and revenue information. Therefore, we performed a quick correlation analysis using Pearson correlation.

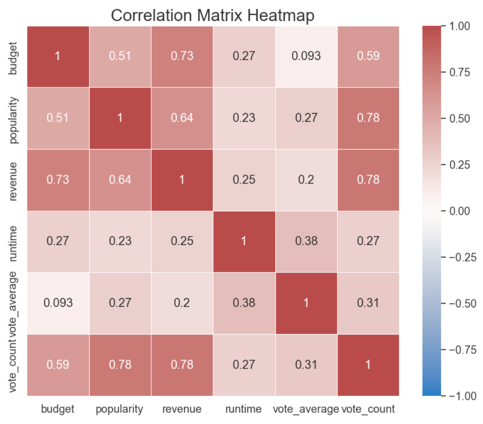

Figure 13: Figure showing heatmap matrix of Pearson correlation of TMDB attributes.

Figure 13 shows the Pearson correlation matrix of TMDB continuous attributes. As shown, revenue and budget are included in this dataset. We can see a few highly correlated values. One such value is the correlation between revenue and budget confirming that movies and media tend to get more money out of larger investments. Another interesting correlation is with revenue and budget with vote count. This may show that larger movies and shows tend to bring in a larger audience. One thing to note is that while vote count is strongly correlated vote average is not, which is interesting as having a larger budget and a larger revenue amount does not seem to result in much higher review scores.

8 Discussion and Conclusion

8.1 Knowledge Gained

We found that we could improve the prediction of a target user's ratings by combining models. This suggests exploring the combination of additional models in further study.

We also found that newer movies had a wider variance and lower average than older movies. This may be a comment on the deterioration of movie quality, the loss of movie theater revenue for making quality movies, or some manner in which the industry is spreading itself too thin. It might not be a comment on the industry at all but rather an artifact of selection bias or survivorship bias. Movies that are still discussed have stood the test of time. TV serials likewise enjoy high ratings, perhaps for the same reason: a fragmentation of the market. Those who like a niche market will return to it.

Documentaries are bucking the negative trend. This may result from the ubiquity of CGI, an increased use in educational settings, or a preference for the most up-to-date works. Building off the earlier observation that popular movies have more ratings, documentaries may be finding a broader audience, either as YouTube helps to make viewers knowledgeable enough to appreciate what is interesting or a targeting of the subjects or celebrity narrators at a broader audience.

Also bucking the trend were sports-related shows and movies.

8.2 Application

The above knowledge could be used to help movie industry executives decide where to direct production investments for improved returns.

The approach of aggregating models into a hybrid prediction could be helpful to a movie rental company such as Netflix. The use of correlation coefficients for credibility weighting of user ratings could be refined. Breaking out movies by genre to calculate genre-specific credibility weights was not found to improve recommendations significantly. One might expect that if user A is aligned with a target user's taste on romances but user B is better aligned for horror movies, then weighting user A's romance recommendations more than user B's would seem to improve predictions. Why this was not the case did not seem to be due to sparsity of data, since users were selected for overlap with the target user's ratings. Rather, the issue was that each movie is tagged with several genres. The approach could be salvaged by narrowing down the genre label for each movie to only one genre.

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
[15] https://en.wikipedia.org/wiki/Mean_absolute_error
[16] https://en.wikipedia.org/wiki/Root-mean-square_deviation