Your Next Binge

Snigdha Kanuparthy

Michael Mokiao

Udisha Bhattacharyya

William David Atwood

Miles Gregory Hewitson

BACKGROUND AND SIGNIFICANCE

Escapism through binge-watching is a common practice amongst many adults and teens [1], but the satisfaction we may get from consuming a new show can come crashing down when our favorite show gets cancelled. *Bloodline, Santa Clarita Diet,* and many other canceled streaming shows highlight a problem we would like to address: how to find your next binge.

Along with the issues that come with having one's favorite show or movie ending, there's not enough done to create a seamless transition between streaming services to continue enjoying similar content. Often with little transparency for the recommendation algorithms used on each platform, there is concern of potentially biased recommendations that may steer customers away from non-approved provider content [2], resulting in increased user dissatisfaction [3].

Ultimately, the project opens the door to potential cross-collaboration and promotion between streaming service providers. Currently solutions are only limited to in-platform recommendations and index on proprietary algorithms that only take single platform factors and features into account. By creating an ensemble model with multiple datasets factored into the features, we will be able to create a more holistic representation of content and provide a better consumer experience.

OBJECTIVES

Problem Statement

Our goal is to create a recommendation system for TV shows and movies that creates a similarity score to help viewers discover and watch similar content on their streaming services to the one they just finished, based on the identification of common patterns and characteristics among these shows.

Hypothesis

H_0: Recommendation system leads to no increase in user satisfaction.

H_a: Recommendation system leads to an increase in user satisfaction.

METHODS AND RESULTS

Data Source

The data used in our analysis was taken from multiple sources. A range of streaming service movie and show information was obtained from Kaggle. Supplemental data was obtained from TMDB using an API [6] to fetch additional information. This dataset was much larger and required significant time and code to compile. The API call requires querying each title individually from the database. Fortunately, there is a data dump of all available titles we could reference. This data though is rather robust and includes many unique variables we did not have in the Kaggle data. Our primary concern was reducing the data to a workable size and filtering to the most relevant variables for our project.

Netflix: 5,850 unique titles, 1981 KB size	Hulu: 2,398 unique titles, 997 KB size	Disney+: 1,535 unique titles, 611 KB size
Amazon Prime Video: 9,871 unique titles, 3,795 KB size	HBO Max: 3,294 unique titles, 1,287 KB size	Paramount+: 2,825 unique titles, 1,092 KB size
Total: 23,553 unique titles		

Table 1. Links to Kaggle datasets of streaming services with information on number of observations and file size.

Data Cleaning

The datasets, as shown in Table 1, were combined into one large dataset and a new column was added to label which streaming service each observation belonged to. This dataset will eventually be cleaned and used in our final creation of the recommendation system, once testing and refining of our algorithm has been completed on smaller datasets.

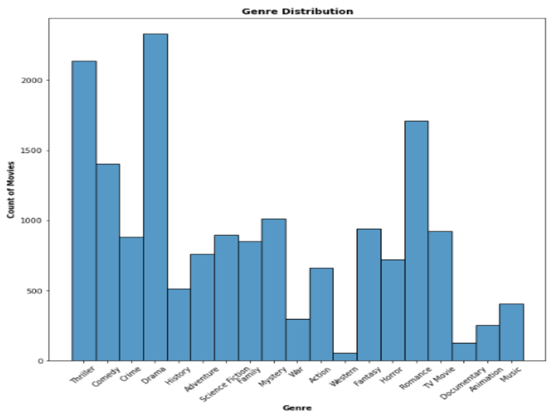
The TMDB data cleaning was unique from the Kaggle data due to its size and storage method. The original API call took several days, so we implemented parallel processing from the Joblib library to bring the call into a reasonable timeline. Additionally, the data was stored as a JSON with nested data in several of the columns based on the original structure from TMDB. We needed to convert these variables into formats that allowed our modeling efforts to intake them, so we choose unique approaches depending on the variable to best capture the desired effect on the model.

To prepare our dataset to be modelled, we used one-hot encoding to create separate binary variables for our categorical variables genre and streaming service and took out datapoints with null vote averages. We then joined the merged streaming service dataset with the TMDB data using a source key that overlapped the two and required additional API calls.

Exploratory Data Analysis

Baseline characteristics were observed through exploratory analysis. Plots of the relationship between variables were preformed to better understand the data collected such as: the count of movies by release year, distribution of release year by streaming service, etc. An interesting relationship established through exploratory data analysis was genre distribution, Figure 1 which shows movie genres that are overrepresented. This may lead to potential bias and may need to be addressed in future iterations of the model.

Figure 1. Movie Count by Genre



Next, to compare the IMDB scores across streaming service, we made Figure 2. We notice that there are a couple of streaming services such as Amazon Prime Video and Paramount+ with higher variable between the first and third quantiles of their IMDB score distribution while HBO Max has the lowest variability and highest third quantile along with Hulu. This suggests that there may be more viewership now or in the future on the streaming services we notice have higher IMDB score averages and this recommender should work well for the streaming services in higher demand. This visual also shows a couple of low outliers that need to get future investigated or removed.

Figure 2. Distribution of IMDB Scores by Streaming Service

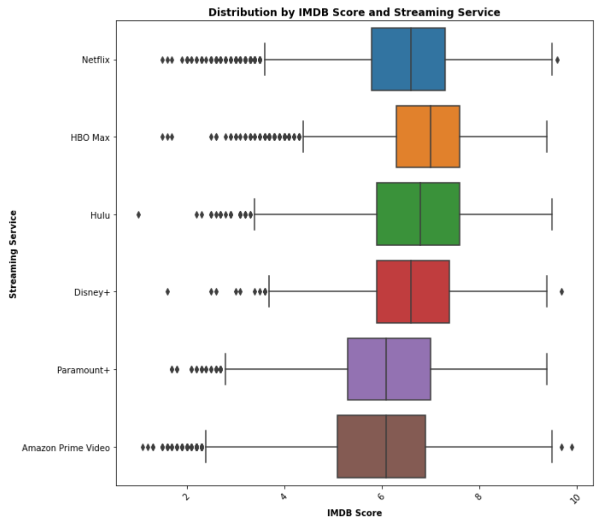


Figure 3. Movie and Show Count by Year

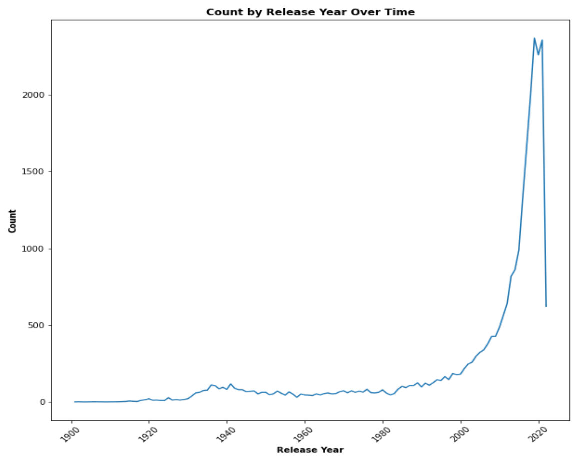


Figure 3 shows the frequency of TV Shows and Movies over time. We notice that 77% of the data is for release years after 2000. This shows that in more recent years, there is a lower barrier to entry to create content, and an outpaced rate of content creation which further validates the need for more advanced filtering and search recommendations.

Feature Engineering Considerations

Prior to model development, weighted average was calculated to renormalize and standardize weights on a 15-point scale; it is difficult to claim that a piece of media with five votes and an average score of eight is a more beloved movie or show than something with thousands of votes and an average score of six. As per IMDB internal documentation, when comparing movies and shows, a simple weighted average is performed to renormalize the rating such that the content can be more accurately compared. The following equation was used to calculate the weighted average:

$$\frac{imdb.votes_i \cdot imdb.score_i}{imdb.votes_{90th\ quantile} + imdb.votes_i} + \frac{imdb.votes_i \cdot imdb.score_{mean}}{imdb.votes_{90th\ quantile} + imdb.votes_i}$$

We took a couple of steps to prepare the data

For the linear regression model, we used one-hot encoding to create binary columns for streaming service and genre factors. We used natural language processing (NLP) on TV shows and movie descriptions, genre, keywords, actors to address and incorporate a vital component of identifying similar movies and shows into our model. We standardized genre tagging using loose regex and to identify and create macro-groups within the data that corresponded to genres and assign each data point to a macro group based on the match score.

Key Variable Identification

Mallow's cp was used for initial variable identification by generating regression models for each combination of independent variables and identifying the model with the lowest Mallow's cp value. Interestingly, if we take the variables chosen using Mallow's cp, the model's R-squared value lowers to 0.20 and mean squared error increases to 6.82 [4].

Further variable selection techniques were used to identify key variables such as: forward stepwise regression, backward stepwise regression, LASSO regression, and elastic net regularization. Akaike information criterion (AIC) was used to determine model performance for backward stepwise and forward stepwise regression. Interestingly, backward stepwise regression failed to eliminate variables because the full model had the lowest AIC score, indicating the best model performance.

Mean-squared prediction error (MSPE) was calculated and used to determine the model with the best fit; a lower MSPE indicates a better fit. As shown in Table 2, the full model and backward stepwise regression model, with all variables included, had the lowest MSPE. This suggests that a model with all variables included should be used for model development.

Model	MSPE
Full	**5.719271**
Forward	5.827960
Backward	5.719271
Lasso	5.720232
Elastic Net	5.720117

Table 2. Key Variable Identification Method MSPE Values

Model Development and Performance

Linear Regression

Development:

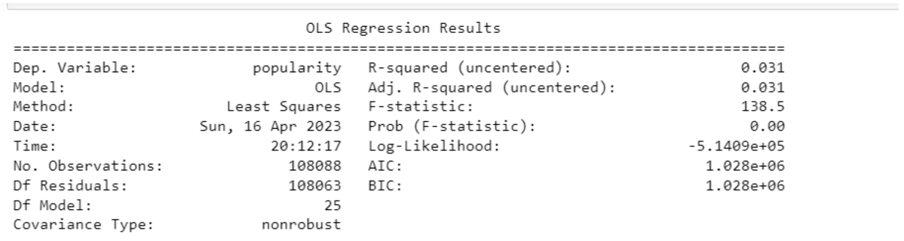
The development of the recommendation system was an iterative process. Two linear regression models were first used to generate recommendations for users using the Kaggle dataset. Popularity and weighted average TMDB scores were used as response variables for each of the two models; 18 binary variables for genre, six for streaming services, runtime, and release year were used as explanatory variables. The linear regression models resulted in an output that reflected a predicted rating; no y-value was a good enough proxy for a recommendation for a "next watchable show."

Results:

- **Scope**: Movies from our TMDB dataset
- **Data Cleaning**: Extracted the top genre and production company and removed null rows for the two responses we are considering: vote_average and popularity. Calculated weighted average

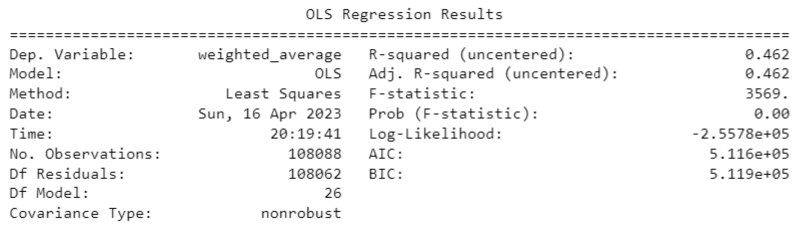
using vote average and vote count. Row count went down from 287,160 to 108,088 movies after cleaning.

- **Factors**: 18 binary variables for genre and 6 for streaming service, runtime, and release year
- **Response Variable**: **popularity** → Results indicated it had poor model performance with R-squared value being the highest taking either runtime and release year or streaming service at 0.031 and mean squared error of 800. Distribution plots and full regression results are shown in Appendix.
- **Response Variable**: **weighted average** → Results indicated it was a better fit. Our initial R-Squared shows that the model has a low predictive strength at an R-Squared value of 0.47 and mean squared error of 6.58. We see predictive abilities with this response taking all our factors into consideration but realize due to the poor performance that other methods may be more effective modeling our factors. We look to developing a K-Nearest Neighbor model next.



```
                            OLS Regression Results
=================================================================================
Dep. Variable:             popularity   R-squared (uncentered):             0.031
Model:                            OLS   Adj. R-squared (uncentered):        0.031
Method:                 Least Squares   F-statistic:                        138.5
Date:                Sun, 16 Apr 2023   Prob (F-statistic):                  0.00
Time:                        20:12:17   Log-Likelihood:               -5.1409e+05
No. Observations:              108088   AIC:                            1.028e+06
Df Residuals:                  108063   BIC:                            1.028e+06
Df Model:                          25
Covariance Type:            nonrobust
```

Table 3. Summary of Linear Regression Model - Popularity



```
                            OLS Regression Results
=================================================================================
Dep. Variable:       weighted_average   R-squared (uncentered):             0.462
Model:                            OLS   Adj. R-squared (uncentered):        0.462
Method:                 Least Squares   F-statistic:                        3569.
Date:                Sun, 16 Apr 2023   Prob (F-statistic):                  0.00
Time:                        20:19:41   Log-Likelihood:               -2.5578e+05
No. Observations:              108088   AIC:                            5.116e+05
Df Residuals:                  108062   BIC:                            5.119e+05
Df Model:                          26
Covariance Type:            nonrobust
```

Table 4. Summary of Linear Regression Model – Weighted Average

K-Nearest Neighbor (KNN)

Development:

To address the limitations of the linear regression model, we opted to use the K-Nearest Neighbors (KNN) algorithm in the development of our recommendation system using a collaborative approach to recommendation system creation [1]. The KNN model created was able to cluster based on audience votes and audience size - here the thinking was that if a connoisseur and aficionado of film decides to watch a B-movie like *Sharknado*, she will not be interested in something with a heavier overtone with more critical acclaim such as *Phantom of the Opera*. The assumptions made with KNN were that movie tone and tenor overcame the genre preference.

Because the KNN model only informed us about the relative popularity of a film and sorted it into the appropriate tier, if we assigned recommendations at this stage, it would be for nearest neighbors based on the relative popularity, and not a more holistic interpretation of the film itself. This led us to pursue building an ensemble model.

The next stage was to add the descriptions, genres and actors back to the dataset needing modelling. The logic was that if a viewer liked a show such as *Haunting on Hill House*, which has a description that references ghosts and drama, he might also like *The Conjuring:* a movie that shares a genre with and has a similar plot as the show that he enjoyed. Similarly, if the viewer is a huge fan of Nicholas Cage, maybe she really liked *The Wicker Man* because of Nicholas Cage and would be interested in seeing more Nicolas Cage B-movies.

Results:

The KNN model clusters were validated through categorical groupings by checking for a balanced distribution of clusters, comparing variable similarity, and validating recommendation implications. Based on the performance of the model, cluster density performed at a higher granularity than needed for our final model, so we used the KNN model to supplement our results.

Linear Kernel

Development:

The next stage of model development was to use a linear kernel function in a Support Vector Machine (SVM) model to create recommendations for the user built on top of the KNN model created. One of the most important components of a movie or show is the plot, which is encapsulated in the description. By vectorizing the description, it is possible to turn the text into a feature. Based on the earlier text exploration and PCA, we can say that the data is linearly separable enough to use a kernel method and a line of separation to find the nearest text match and rank shows and movies based on their similarity to the specified content. The actors list and the genre list of the movies were also vectorized and we performed a linear kernel similarity analysis on each of the vectors with the original "chosen" movie vector. This was a lightweight way to test a similarity score between two strings with similar concepts but not identical wording. The output of each linear kernel was then added together as a flat average, and the top ten recommendations were taken by choosing the top 10 highest averages. Lastly, we added a function to filter by streaming service so that the user can only get recommendations for the streaming services they have access to. The relationship between a vector created from text vectorization and the potential inputs to the functions was not strongly validated, because we largely used existing libraries to preform our standardization of the text which limits our insights into the decisions.

Results:

To assess the linear kernel SVM model performance, three main scenarios were tested using a range of input criteria. Group members then evaluated whether the recommendations from our model were reasonable based on personal sentiment as well as comparison to other streaming service recommendations.

In the first scenario, one streaming service dataset was used with *Monty Python* and other shows and movies with a large library of things with the same cast, similar concepts, and similar genres as our input. The top five recommendations from the model were reasonable and suggested other popular movies from the other five in that set.

In the second scenario, one streaming service with shows and movies that may not have as many sequels, remakes or similar casts were used as input. *Sailor Moon Crystal*, an all-ages children's anime, was one example of the input into our recommendation system. Though some of the recommendations made by our model were like *Sailor Moon Crystal*, the mature-themed thriller *Prison Break* was also recommended, which doesn't match the tone and theme very well. It is worth noting that *Little Witch Academia* is on the same streaming service (Netflix) and would have been a better fit, as it is also about adolescent girls with magic powers and is also in the anime genre.

In the third scenario, all streaming services were used with a variety of shows and movies as input. There were mixed results. Using *Monty Python* as an input gave inconclusive results as the recommendations were not as relevant as using one streaming service's dataset as in the case of the first scenario. An issue may have been that the descriptions for these shows, and the genre tagging were not always 1:1 on other streaming services, so the quality of the expert label and alignment between services really did not matter.

Cosine Similarity

Development:

The final recommendation system that yielded highest model performance was when we swapped a linear kernel function for cosine similarity. Cosine similarity is a similar natural language processing technique that fundamentally scores based on term frequency and inverse data frequency. The unique element is that cosine similarity creates two vectors and measures the angle between them to determine closeness. Applying this to each of our text variables and summarizing into the composite score prove the most performant.

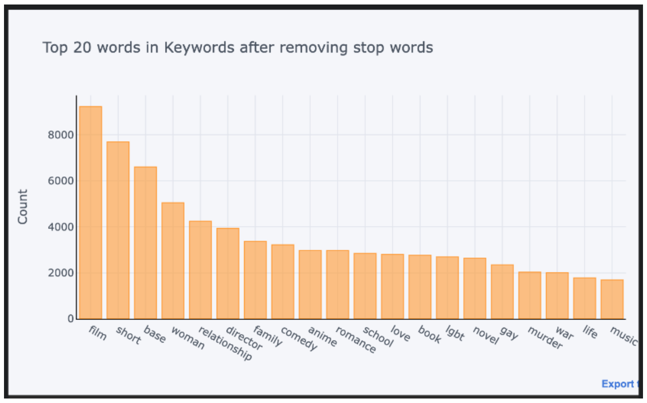
In the journey of testing out this new model, we expanded several aspects of our project. While the clustering and linear kernel heavily relied on the Kaggle dataset and its limitations, this new model drew heavily from the TMDB data. We even expanded the variables we scraped from the API to enhance the recommendation quality of the model. Adding new descriptions, extended genres, keywords, cast, and crew variables. Parsing the data was time-consuming due to the size and transfer rate of the API. Unoptimized scripts would run for days, fortunately, we were able to parallelize many of the API pulls and early data-cleaning actions. In addition to the regular data cleaning the cosine similarity library required, we benefited from heavy preprocessing of the text data to standardize and clean up character oddities which improved the similarity scoring. The libraries we utilized for this NLP were massive unlocks and time saver. After we had fully processed our data, we passed the text variables into the cosine similarity function and then ranked it in our recommender function to produce the results.

This iterative process provided insights into many additional opportunities for enhancements that we will review in the limitations and future work sections below. From alternative text processors to entirely new datasets we have a wide array of opportunities to take this even further.

Results:

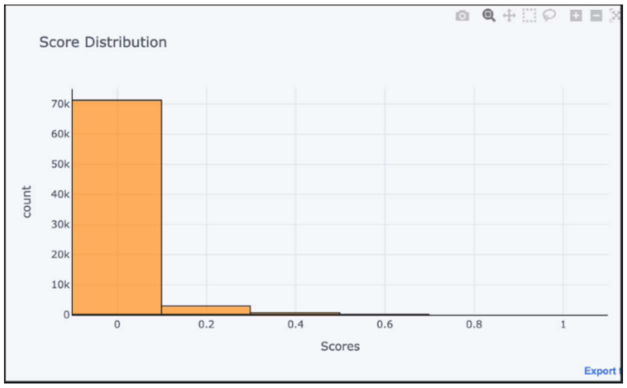
Due to the nature of our recommender systems, our performance checks are largely comparative, logical, and anecdotal, rather than statistical. Because this was our final model, we were able to compare the results with all our previous models. The recommendations aligned more closely with expectations across an array of inputs. One of our first efforts was to validate assumptions about how the model was executing the similarity scoring. Frequently used words should have insignificant impact, and rare words, found in two unique variables, should drive most of the score. Our visuals confirm this behavior. First, words with high frequency across all titles as shown in Figure 4, would not differentiate titles.

Figure 4. Top 20 Keywords Removed from Cosine Similarity Model



Also, we expect a well-performing model to produce far fewer closely similar titles and most titles to have little to no similarity. This result was confirmed when we examined them as seen in Figure 5. In this example, we have slightly over 74k titles, and most fall in the lowest bins of scores, indicating little to no similarity. This distribution shape fits industry accepted best practices for content-based filtering, using cosine similarity. This threshold of similarity can be tuned to improve accuracy or quantity of recommendations. Across our limited tests we found much beyond the .5 score threshold for this model was a noticeable drop in quality recommendations. This often produced 5 or fewer results.

Figure 5. Cosine Similarity Score Distribution



Statistical Methods

All data were processed and analyzed using R and some Python   Our code is on a publicly available repository through GitHub at https://github.gatech.edu/MGT-6203-Spring-2023-Canvas/Team-50.

DISCUSSION

Literature Survey

According to the literature evidence base, users are eager to find new “resonant experiences” in media but are limited in this pursuit by the scope of available recommender systems. A cross-platform, knowledge-based system would be an ideal methodology for creating robust recommendations that resonate with users.

Vorderer and Klimmt (2021) explore the differences between “entertainment experiences,” or responses that serve the users’ hedonic motivations, needs or interests, and “resonant experiences,” or responses that leave individuals altered or touched and therefore lead to “ongoing and dynamic transformation and diversification.” [1] Resonant experiences in media may trigger a development in a person’s life and motivate them to do as much as possible to guarantee similar situations in the future [1]. Though keeping an open mind and finding new resonant experiences with media can prove elusive, particularly in modern day and when limited only to in-platform recommendations, users are eager to find similar experiences.

An ideal solution to this problem would be a robust, platform-agnostic recommendation system. According to Ricci et al, a knowledge-based system will, “recommend items based on specific domain knowledge about how certain item features meet users’ needs and preferences and, ultimately, how the item is useful for the user.” [5] With access to cross-platform data, a knowledge-based system would be a strong recommender system technique for creating robust recommendations that resonate with users. A risk identified with this approach is, while initially they may perform well, without learning components in the system, the system can be surpassed.

The reviewed literature suggests users are eager to find new and exciting “resonant experiences” in media and a knowledge-based recommender system would be an ideal methodology to help find new experiences. We conclude, based on the current evidence, there is a limitation in what users will be recommended, which then limits their likelihood of finding new, resonant experiences in media.

Final Results

Recommendation System

Through the iterative process of model development and performance evaluation, we created a cross-platform streaming service recommendation system using cosine similarity for evaluating similarity between vectorized text fields backed by a KNN model. This model used a combination of key words, description, genre, and other relevant factors to create a tool that may provide more robust recommendations based on the streaming services a user is subscribed to.

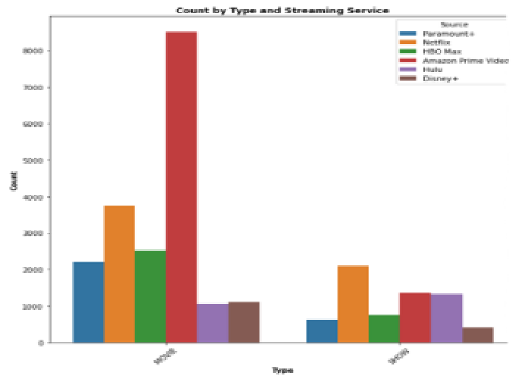
We created a unique recommendation system that sources data from multiple streaming services while avoiding the potential biases streaming services may have to promote their own content. Our model's cross-platform capability allows users to mix and match streaming services while having a recommender that can be used across all their services. An example usage of this is shown on Tables 5 and 6 in the Appendix.

Potential future uses of our recommendation system are an adjunct service such as a Chrome extension that allows users to bypass streaming service's recommendation algorithms or as a web browser where users can input a movie or show title and receive recommendations based on the streaming services they currently pay for.

Limitations

- Size of the data: The dataset used required significant time and optimization to be usable, but the recommender is constrained by the number of TV shows and movies we were able to account for. Future iterations will require more work to increase the data size.
- Time: We iteratively tested new variables and modeling approaches and the limitation of time constrained us from completing all of our experiments. Future iterations will require more time to improve model quality to ensure it is robust.
- Data quality: missing data, biased data, and poorly structured data. Exploratory data analysis illustrated some bias in certain streaming services, like Amazon Prime Video having disproportionately more movies than other streaming services, as shown in Figure 6 below. Future work would need to improve the quality of the data to get rid of all bias and improve the quality of the data used for the model.

Figure 6. Count of Movies and TV Shows by Streaming Service



Future Work

1. Improve data and model quality, as stated in the Limitations section above.
2. Add weights to genre, actor, keyword, and description similarity scores during overall similarity score calculation based off learnings on what weight values should be.
3. Create a front-end user interface for the recommender for a seamless user experience.
4. Conduct a random survey of users for the recommender to assess whether the recommendations were satisfactory to them.
5. Incorporate collaborative-based filtering in the future to learn what to recommend based off data collected by the system from user activity instead of a fully content-based filtering by working with streaming services to integrate this recommender and publish it.

CONCLUSION

The idea set forth at the beginning of this project was to create a recommendation system that increased user satisfaction by providing recommendations that were not restricted to a particular streaming service and not limited to the biases a streaming service may incorporate into its recommendation algorithm. The journey to creating a robust recommendation system has led us to a functioning model that recommends similar content across multiple streaming services. Though there is more work to be done increasing model performance as well as future work measuring customer satisfaction, the foundation of a usable model is there.

References:

1. Vorderer, Peter, and Christoph Klimmt. *The Oxford Handbook of Entertainment Theory*. *Google Books*, Oxford University Press, 12 Feb. 2021.
2. Bourreau, M., & Gaudin, G. (2022). Streaming platform and strategic recommendation bias. *J Econ Manage Strat*. 31, 25– 47. https://doi.org/10.1111/jems.12452
3. "Netflix Has Finally Hit a Wall. Where Does It Go from Here?" *Bloomberg.com*, 20 Apr. 2022, www.bloomberg.com/news/newsletters/2022-04-20/why-is-netflix-losing-subscribers-and-where-can-it-go-from-here.
4. Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* **9**, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5
5. Ricci, F., Rokach, L., Shapira, B. (2022). **Recommender Systems: Techniques, Applications, and Challenges**. In: Ricci, F., Rokach, L., Shapira, B. (eds) Recommender Systems Handbook. Springer, New York, NY. https://doi.org/10.1007/978-1-0716-2197-4_1
6. TMDB, The Movie Database. API documentation https://developers.themoviedb.org/

Appendix: Distribution Plots of Movie Popularity and Vote Average

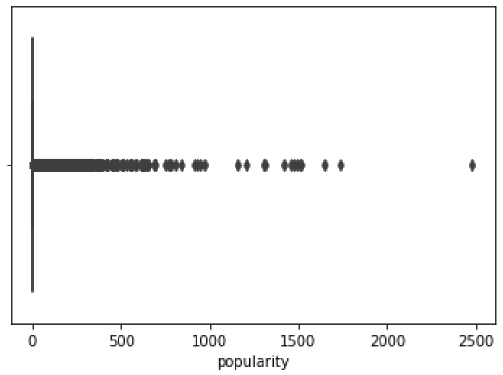


Figure 7. Distribution of Popularity Count

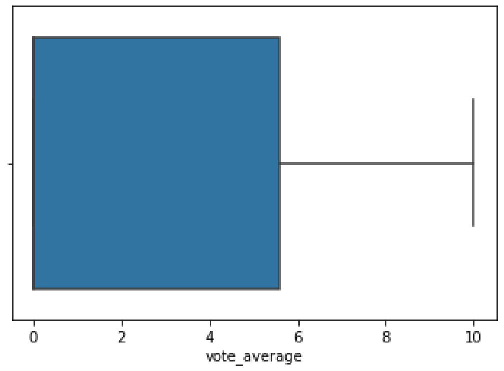
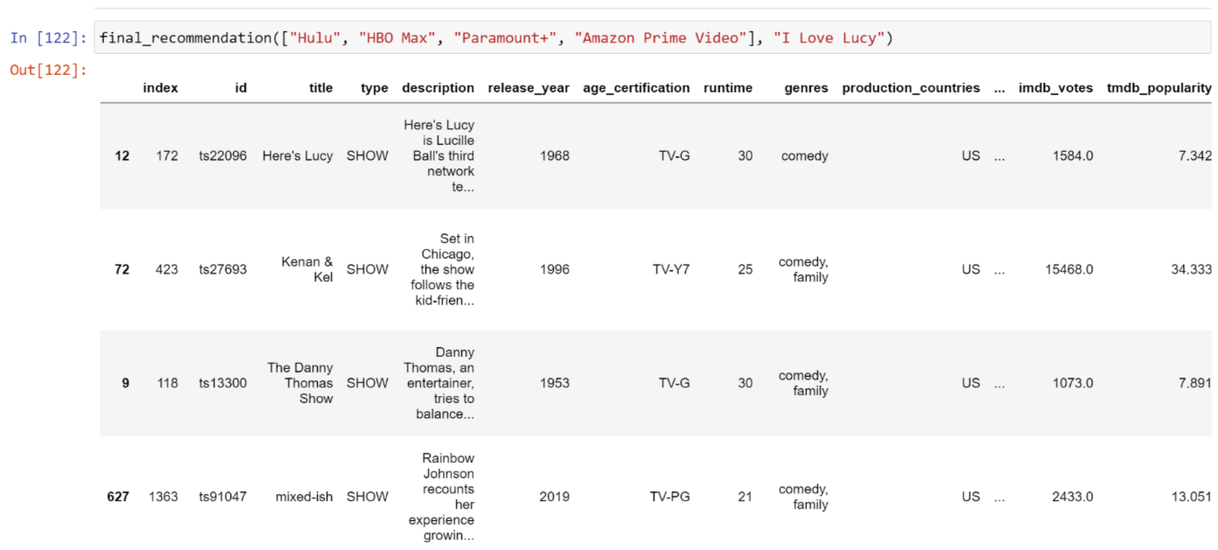


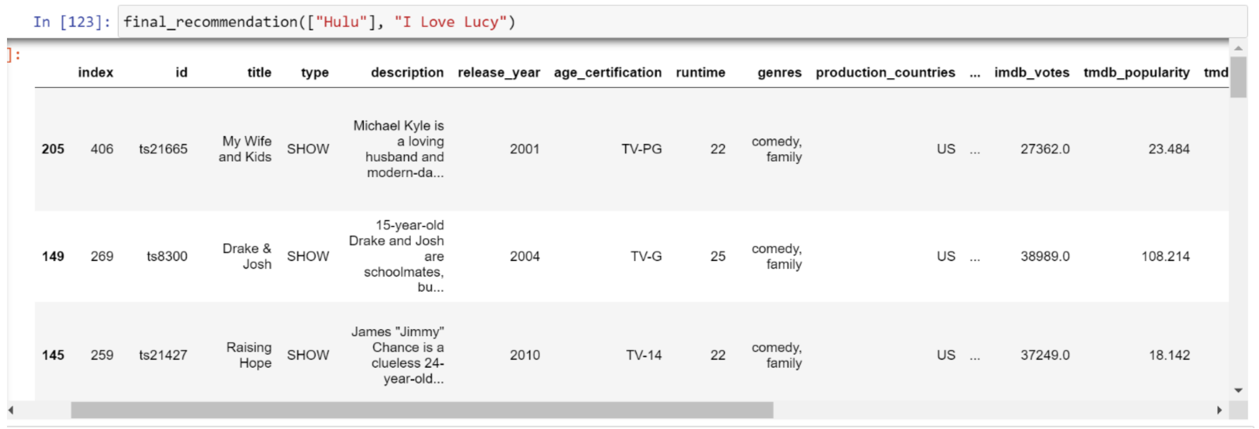
Figure 8. Distribution of Average Votes



```
In [122]: final_recommendation(["Hulu", "HBO Max", "Paramount+", "Amazon Prime Video"], "I Love Lucy")
Out[122]:
```

	index	id	title	type	description	release_year	age_certification	runtime	genres	production_countries	...	imdb_votes	tmdb_popularity
12	172	ts22096	Here's Lucy	SHOW	Here's Lucy is Lucille Ball's third network te...	1968	TV-G	30	comedy	US	...	1584.0	7.342
72	423	ts27693	Kenan & Kel	SHOW	Set in Chicago, the show follows the kid-frien...	1996	TV-Y7	25	comedy, family	US	...	15468.0	34.333
9	118	ts13300	The Danny Thomas Show	SHOW	Danny Thomas, an entertainer, tries to balance...	1953	TV-G	30	comedy, family	US	...	1073.0	7.891
627	1363	ts91047	mixed-ish	SHOW	Rainbow Johnson recounts her experience growin...	2019	TV-PG	21	comedy, family	US	...	2433.0	13.051

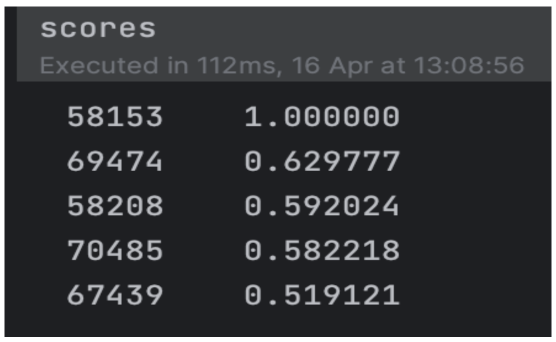
Table 5. Output of Recommendations Amongst Four Streaming Services



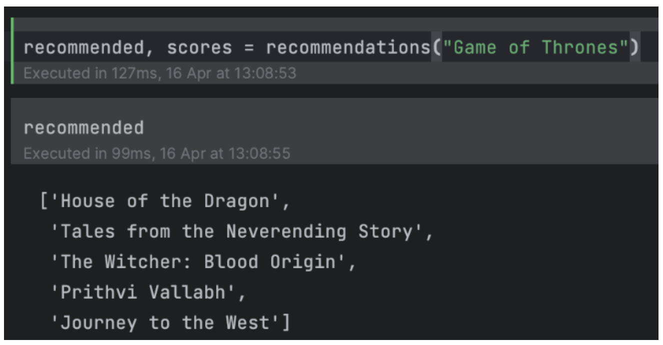
```
In [123]: final_recommendation(["Hulu"], "I Love Lucy")
]:
```

	index	id	title	type	description	release_year	age_certification	runtime	genres	production_countries	...	imdb_votes	tmdb_popularity	tmd
205	406	ts21665	My Wife and Kids	SHOW	Michael Kyle is a loving husband and modern-da...	2001	TV-PG	22	comedy, family	US	...	27362.0	23.484	
149	269	ts8300	Drake & Josh	SHOW	15-year-old Drake and Josh are schoolmates, bu...	2004	TV-G	25	comedy, family	US	...	38989.0	108.214	
145	259	ts21427	Raising Hope	SHOW	James "Jimmy" Chance is a clueless 24-year-old...	2010	TV-14	22	comedy, family	US	...	37249.0	18.142	

Table 6. Output of Recommendations from Hulu

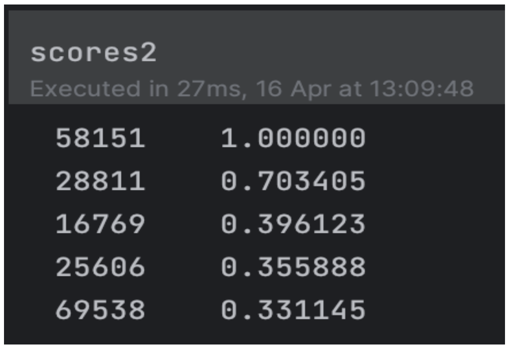


```
scores
Executed in 112ms, 16 Apr at 13:08:56

58153    1.000000
69474    0.629777
58208    0.592024
70485    0.582218
67439    0.519121
```

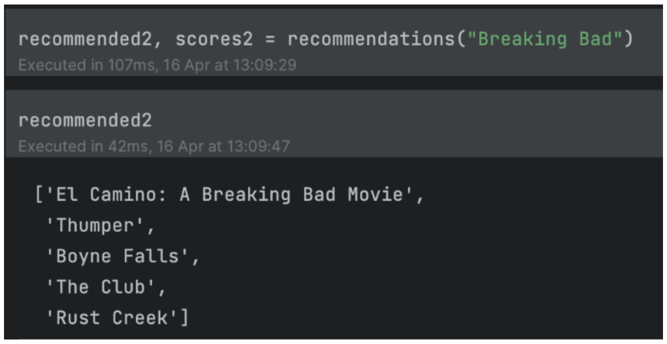


```
recommended, scores = recommendations("Game of Thrones")
Executed in 127ms, 16 Apr at 13:08:53

recommended
Executed in 99ms, 16 Apr at 13:08:55

['House of the Dragon',
 'Tales from the Neverending Story',
 'The Witcher: Blood Origin',
 'Prithvi Vallabh',
 'Journey to the West']
```

Figure 9. Cosine Similarity Score and Recommendations for *Game of Thrones*

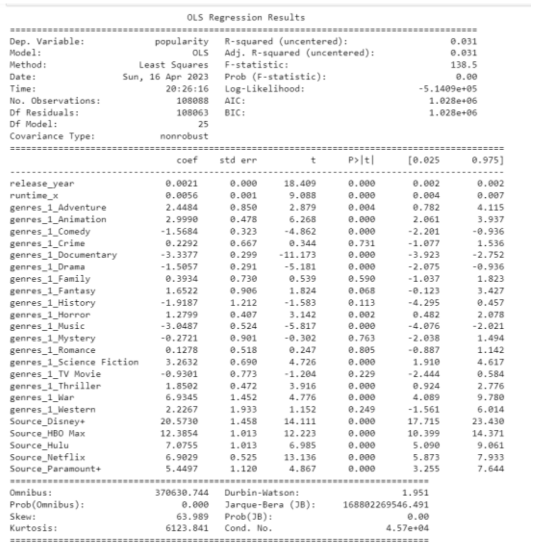


```
scores2
Executed in 27ms, 16 Apr at 13:09:48

58151    1.000000
28811    0.703405
16769    0.396123
25606    0.355888
69538    0.331145
```

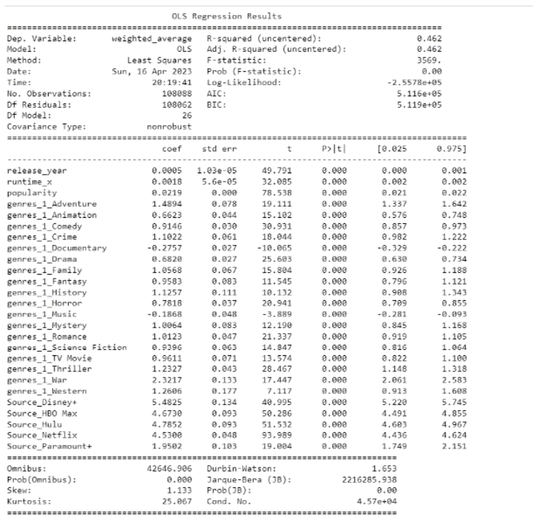


```
recommended2, scores2 = recommendations("Breaking Bad")
Executed in 107ms, 16 Apr at 13:09:29

recommended2
Executed in 42ms, 16 Apr at 13:09:47

['El Camino: A Breaking Bad Movie',
 'Thumper',
 'Boyne Falls',
 'The Club',
 'Rust Creek']
```

Figure 10. Cosine Similarity Score and Recommendations for *Breaking Bad*



```
                              OLS Regression Results
=======================================================================================
Dep. Variable:             popularity   R-squared (uncentered):                   0.031
Model:                            OLS   Adj. R-squared (uncentered):              0.031
Method:                 Least Squares   F-statistic:                              138.5
Date:                Sun, 16 Apr 2023   Prob (F-statistic):                        0.00
Time:                        20:26:16   Log-Likelihood:                     -5.1409e+05
No. Observations:              108088   AIC:                                  1.028e+06
Df Residuals:                  108063   BIC:                                  1.028e+06
Df Model:                          25
Covariance Type:            nonrobust
============================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
release_year                0.0021      0.000     18.409      0.000       0.002       0.002
runtime_x                   0.0056      0.001      9.088      0.000       0.004       0.007
genres_1_Adventure          2.4484      0.850      2.879      0.004       0.782       4.115
genres_1_Animation          2.9990      0.478      6.268      0.000       2.061       3.937
genres_1_Comedy            -1.5684      0.323     -4.862      0.000      -2.201      -0.936
genres_1_Crime              0.2292      0.667      0.344      0.731      -1.077       1.536
genres_1_Documentary       -3.3377      0.299    -11.173      0.000      -3.923      -2.752
genres_1_Drama             -1.5057      0.291     -5.181      0.000      -2.075      -0.936
genres_1_Family             0.3934      0.730      0.539      0.590      -1.037       1.823
genres_1_Fantasy            1.6522      0.906      1.824      0.068      -0.123       3.427
genres_1_History           -1.9187      1.212     -1.583      0.113      -4.295       0.457
genres_1_Horror             1.2799      0.407      3.142      0.002       0.482       2.078
genres_1_Music             -3.0487      0.524     -5.817      0.000      -4.076      -2.021
genres_1_Mystery           -0.2721      0.901     -0.302      0.763      -2.038       1.494
genres_1_Romance            0.1278      0.518      0.247      0.805      -0.887       1.142
genres_1_Science Fiction    3.2632      0.690      4.726      0.000       1.910       4.617
genres_1_TV Movie          -0.9301      0.773     -1.204      0.229      -2.444       0.584
genres_1_Thriller           1.8502      0.472      3.916      0.000       0.924       2.776
genres_1_War                6.9345      1.452      4.776      0.000       4.089       9.780
genres_1_Western            2.2267      1.933      1.152      0.249      -1.561       6.014
Source_Disney+             20.5730      1.458     14.111      0.000      17.715      23.430
Source_HBO Max             12.3854      1.013     12.223      0.000      10.399      14.371
Source_Hulu                 7.0755      1.013      6.985      0.000       5.090       9.061
Source_Netflix              6.9029      0.525     13.136      0.000       5.873       7.933
Source_Paramount+           5.4497      1.120      4.867      0.000       3.255       7.644
==============================================================================
Omnibus:                   370630.744   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):   168802269546.491
Skew:                          63.989   Prob(JB):                         0.00
Kurtosis:                    6123.841   Cond. No.                     4.57e+04
==============================================================================
```



```
                              OLS Regression Results
=======================================================================================
Dep. Variable:       weighted_average   R-squared (uncentered):                   0.462
Model:                            OLS   Adj. R-squared (uncentered):              0.462
Method:                 Least Squares   F-statistic:                              3569.
Date:                Sun, 16 Apr 2023   Prob (F-statistic):                        0.00
Time:                        20:19:41   Log-Likelihood:                     -2.5578e+05
No. Observations:              108088   AIC:                                  5.116e+05
Df Residuals:                  108062   BIC:                                  5.119e+05
Df Model:                          26
Covariance Type:            nonrobust
============================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
release_year                0.0005   1.03e-05     49.791      0.000       0.000       0.001
runtime_x                   0.0018    5.6e-05     32.085      0.000       0.002       0.002
popularity                  0.0219      0.000     78.538      0.000       0.021       0.022
genres_1_Adventure          1.4894      0.078     19.111      0.000       1.337       1.642
genres_1_Animation          0.6623      0.044     15.102      0.000       0.576       0.748
genres_1_Comedy             0.9146      0.030     30.931      0.000       0.857       0.973
genres_1_Crime              1.1022      0.061     18.044      0.000       0.982       1.222
genres_1_Documentary       -0.2757      0.027    -10.065      0.000      -0.329      -0.222
genres_1_Drama              0.6820      0.027     25.603      0.000       0.630       0.734
genres_1_Family             1.0568      0.067     15.804      0.000       0.926       1.188
genres_1_Fantasy            0.9583      0.083     11.545      0.000       0.796       1.121
genres_1_History            1.1257      0.111     10.132      0.000       0.908       1.343
genres_1_Horror             0.7818      0.037     20.941      0.000       0.709       0.855
genres_1_Music             -0.1868      0.048     -3.889      0.000      -0.281      -0.093
genres_1_Mystery            1.0064      0.083     12.190      0.000       0.845       1.168
genres_1_Romance            1.0123      0.047     21.337      0.000       0.919       1.105
genres_1_Science Fiction    0.9396      0.063     14.847      0.000       0.816       1.064
genres_1_TV Movie           0.9611      0.071     13.574      0.000       0.822       1.100
genres_1_Thriller           1.2327      0.043     28.467      0.000       1.148       1.318
genres_1_War                2.3217      0.133     17.447      0.000       2.061       2.583
genres_1_Western            1.2606      0.177      7.117      0.000       0.913       1.608
Source_Disney+              5.4825      0.134     40.995      0.000       5.220       5.745
Source_HBO Max              4.6730      0.093     50.286      0.000       4.491       4.855
Source_Hulu                 4.7852      0.093     51.532      0.000       4.603       4.967
Source_Netflix              4.5300      0.048     93.989      0.000       4.436       4.624
Source_Paramount+           1.9502      0.103     19.004      0.000       1.749       2.151
==============================================================================
Omnibus:                    42646.906   Durbin-Watson:                   1.653
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          2216285.938
Skew:                           1.133   Prob(JB):                         0.00
Kurtosis:                      25.067   Cond. No.                     4.57e+04
==============================================================================
```

Table 7. Coefficients of Linear Regression Model – Weighted Average