Movie Recommendation System

Jonathon Scroggins
DSC 680 Fall 2021
https://github.com/JDScroggins

Background

There are so many movies out there to watch and there are a lot of venues to watch those movies especially with all the streaming services out there. No one likes to sit there for too long trying to figure out what they want to watch. Switching between streaming services and continually scrolling trying to figure out what sounds good. Some people will even seek out the advice of their one friend that they deem the "movie person" and ask for recommendations to watch. As a service to their subscribers, many streaming services are using recommender systems to aid in the task of finding something interesting to watch. The services will recommend a movie to watch based on the movie that the subscriber just finished watching or based on the subscriber's viewing history. For this project, datasets from the International Movie Database, or IMDb, will be used to develop a movie recommender system. There are three types of movie recommender systems that can be created: Content-based, Collaborative, and Hybrid Systems. A content-based system would use similar features to recommend movies. Features like genre, director, writer, actors, etc. can be used to develop a content-based recommender. A collaborative system would use other users that are similar and how they have rated or navigated to recommend. A hybrid system uses aspects of both systems to recommend to the user. For this project we will use a content-based system and a hybrid system.

Problem Statement

Using the IMDb datasets, can a movie recommendation system be built based on a given movie title?

Methods

The data for this project comes from https://m.imdb.com/interfaces/, and was downloaded on October 27, 2021. The datasets used are the names dataset,

	nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
0	nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0050419,tt0072308,tt0031983,tt0053137
1	nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0117057,tt0038355,tt0037382,tt0071877
2	nm0000003	Brigitte Bardot	1934	\N	actress,soundtrack,music_department	tt0057345,tt0049189,tt0056404,tt0054452
3	nm0000004	John Belushi	1949	1982	actor,soundtrack,writer	tt0077975,tt0080455,tt0072562,tt0078723
4	nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0060827,tt0050976,tt0050986,tt0083922

the title basics dataset,

	tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
0	tt0000001	short	Carmencita	Carmencita	0	1894	\N	1	Documentary,Short
1	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	\N	5	Animation,Short
2	tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	\N	4	Animation,Comedy,Romance
3	tt0000004	short	Un bon bock	Un bon bock	0	1892	\N	12	Animation,Short
4	tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	\N	1	Comedy,Short

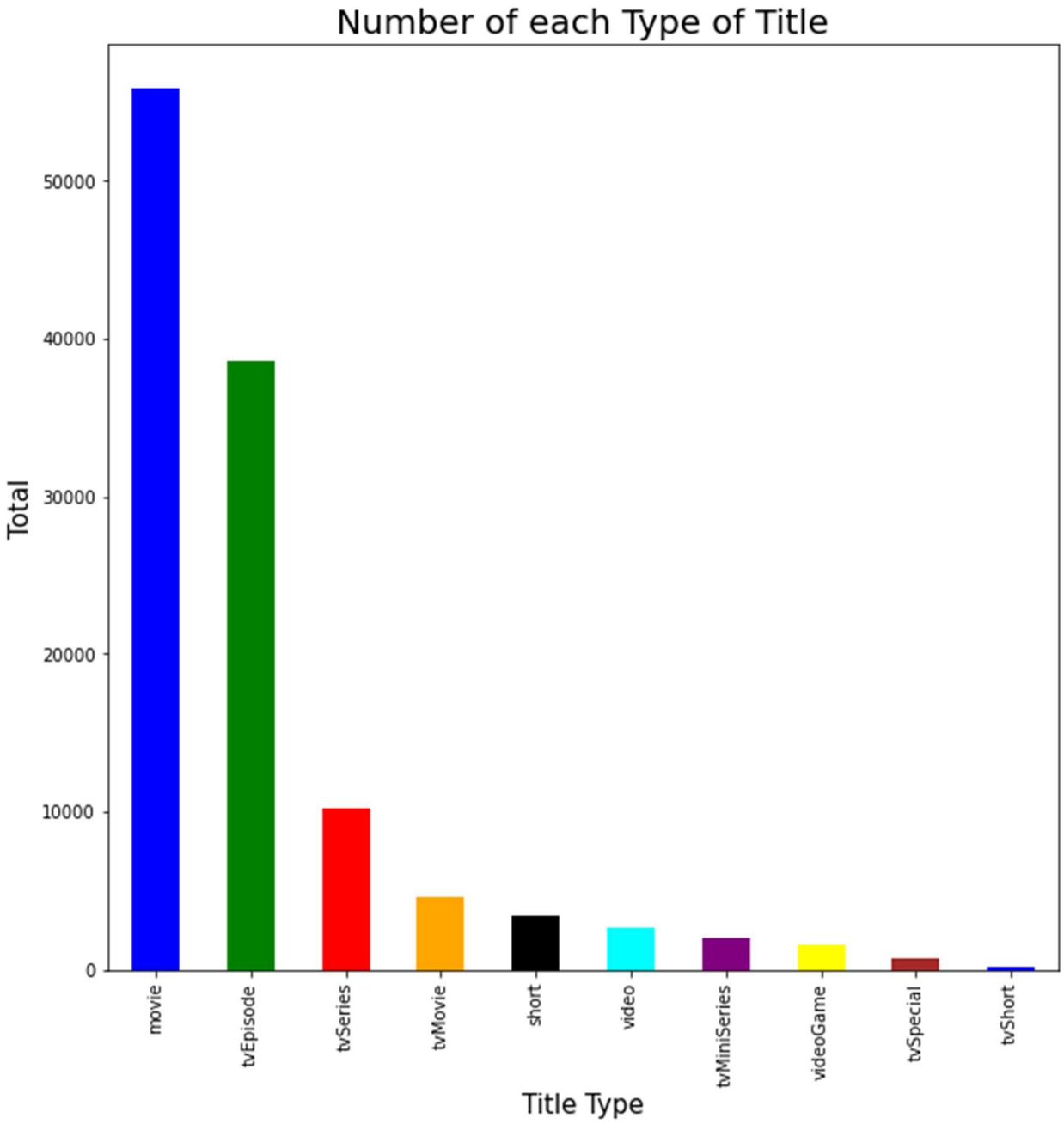
the title crew dataset and the title ratings dataset

	tconst	directors	writers
0	tt0000001	nm0005690	\N
1	tt0000002	nm0721526	\N
2	tt0000003	nm0721526	\N
3	tt0000004	nm0721526	\N
4	tt0000005	nm0005690	\N

	tconst	averageRating	numVotes
0	tt0000001	5.7	1828
1	tt0000002	6.0	236
2	tt0000003	6.5	1589
3	tt0000004	6.0	153
4	tt0000005	6.2	2405

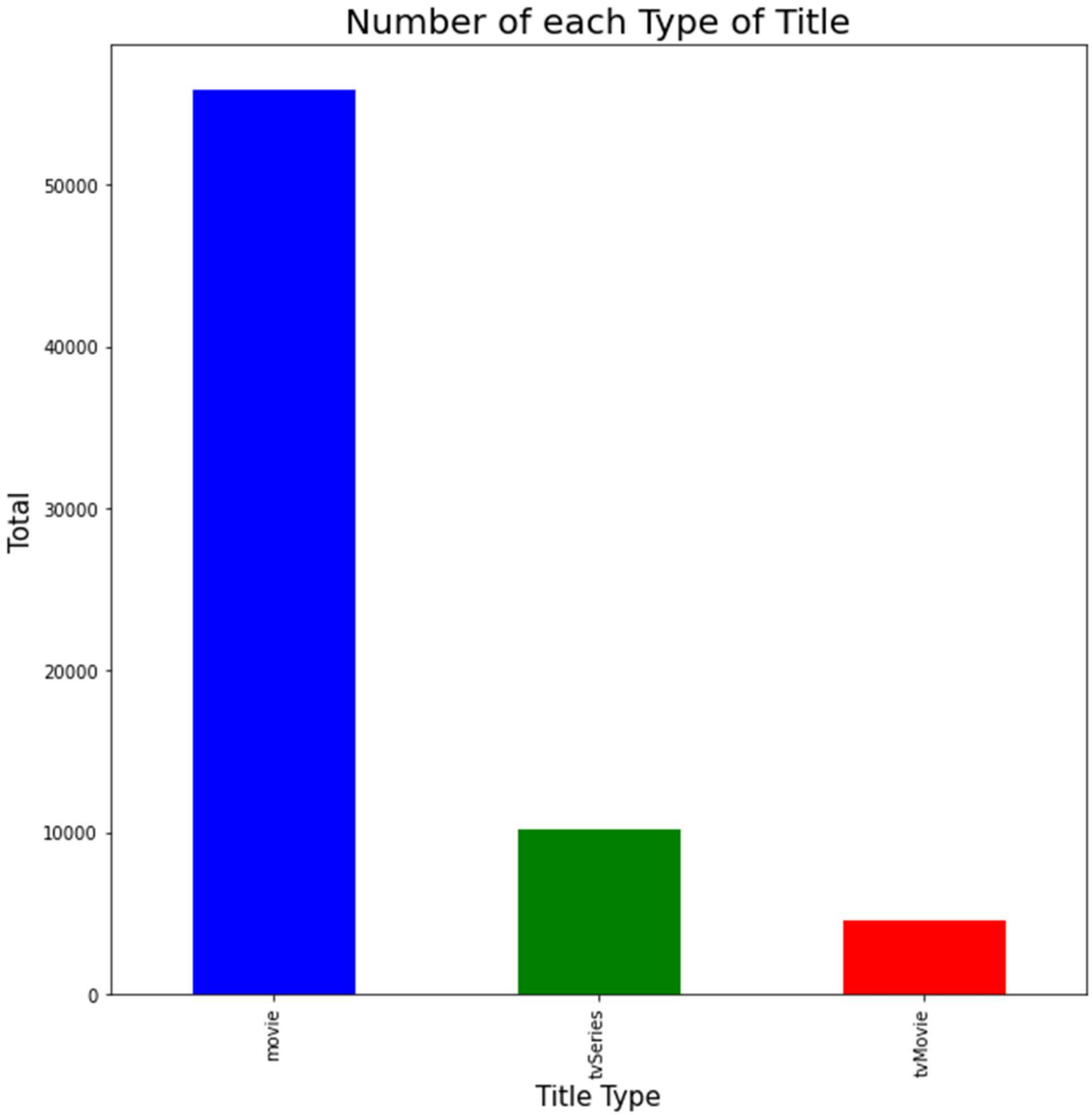
.

Within these datasets, there is information for 119,741 movies, tv shows and more.

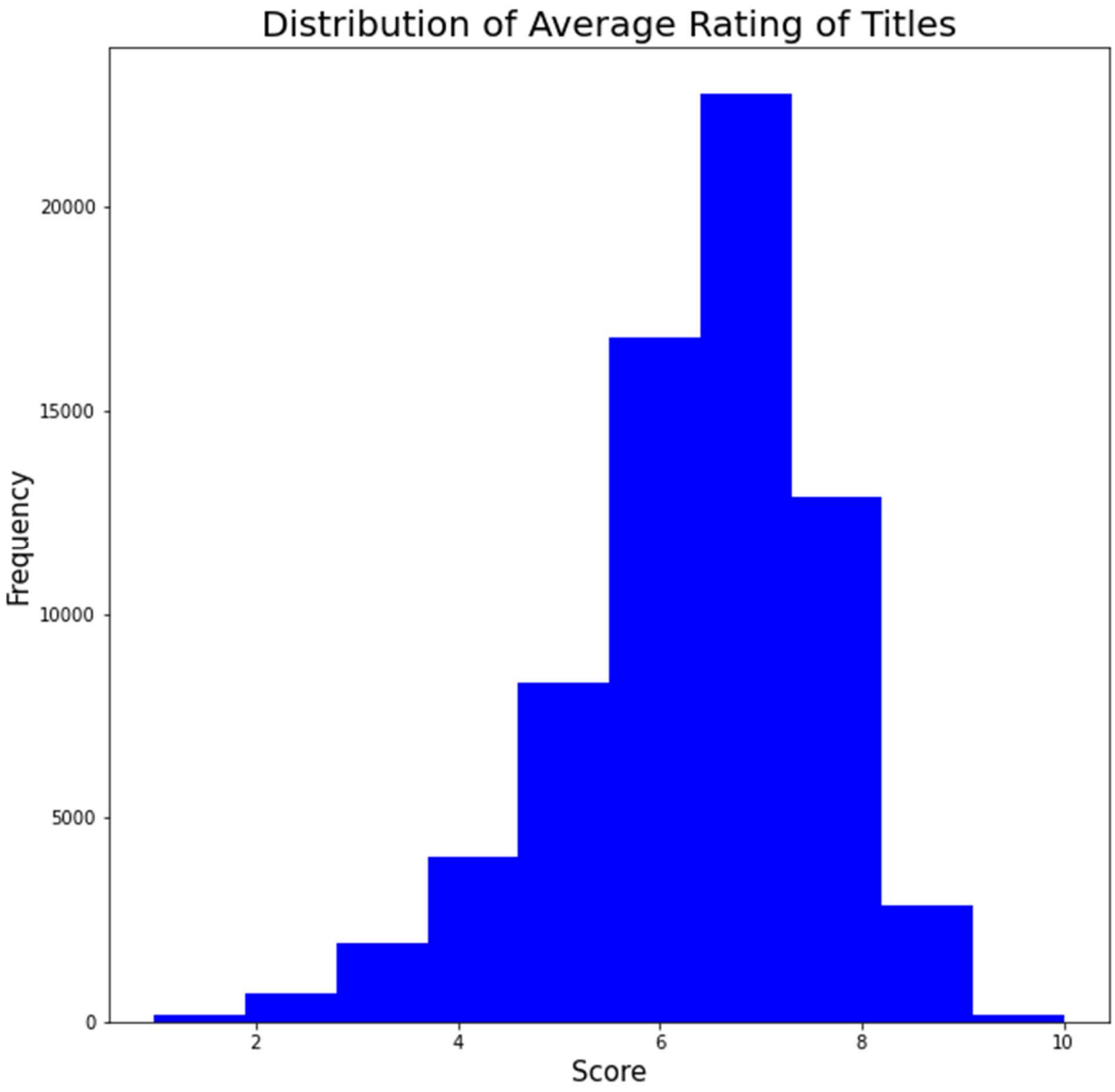


After importing all of the datasets, all of the title datasets were merged together into one dataset using the tconst as the similar column. There were also some null values that had to be dealt with especially in the writers column. Any row with a null value was dropped. The full dataset was also subset into a dataset with just the movie, tvseries and tvmovie column included

leaving us with 70, 633 titles.

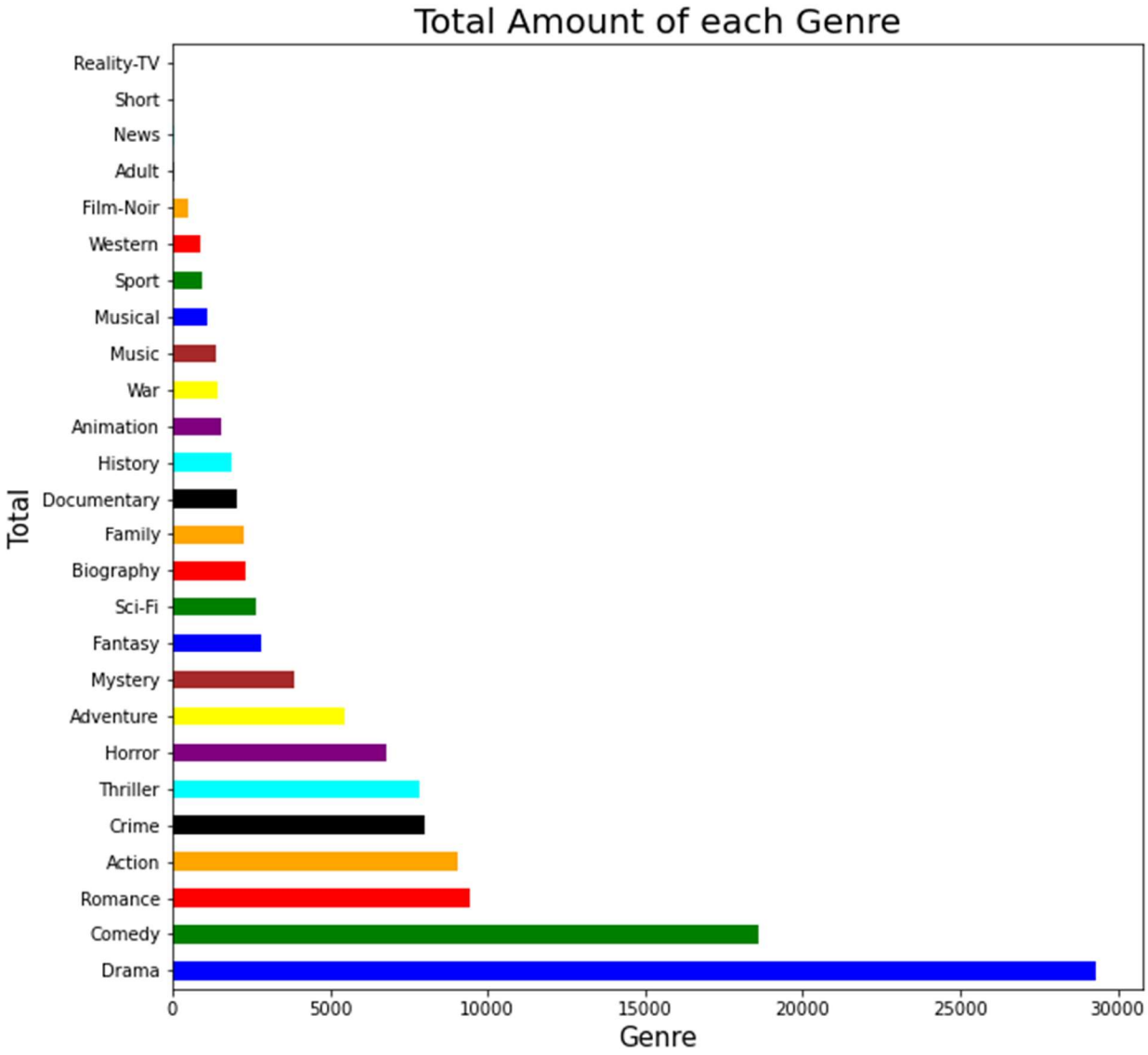


Doing more analysis on the data, we can look at the distribution of the ratings for the titles overall.



This is a close to normal distribution of the average ratings of the titles. There is a slight skewness to the right, but I do not think that it is affected by any outliers that will need to be removed so we will continue with the set as is. The genre column is also an interesting thing. There are many titles that have multiple listings, and we will have to decide how to deal with

that. For most of the recommender systems we can leave them alone and not split them. However, I did split them to get an idea of how the individual genres are represented and found that the drama and comedy genres are much more represented in this dataset.



The first recommender that was made is a content-based recommender based on genre of the title. To do this, we will use the concepts of vectors and their proximity to each other to make recommendations. We must first transform the genres into a vector through TFID-Vectorization and the use the cosine similarity to determine the recommendations. However, as this model was

being built, there was an issue with the size of the dataset, so it was subset further by the year of the movie and the tvshows and tvmovies were dropped and this left 21,053 titles.

The second recommender used both the subset dataset first and then used the full dataset to make recommendations. The second recommender used the concept of nearest neighbor to provide recommendations. A pivot table was created using the title, genre and average rating. That was then put into a matrix and the cosine similarity is used again to find the other titles that are closest to the title.

Results

For the content-based recommendation system, three movie titles were used to get recommendations for are Suicide Squad, The Avengers, and Coco.

```
genre_recommendations('Suicide Squad').head(10)
```

```
51                              Wonder Woman
75     Prince of Persia: The Sands of Time
83                  Atlas Shrugged: Part I
142                            Devil's Knot
145                             The Witches
147                             Tooth Fairy
164                           Maximum Ride
170                          Electric Slide
173                      Hotel Transylvania
248                      A Monster in Paris
```

```
genre_recommendations('The Avengers').head(10)
```

```
23                              John Carter
39                    Alita: Battle Angel
58      Captain America: The First Avenger
117                       Night Catches Us
175             Bhopal: A Prayer for Rain
189                                Bolden
506       Wall Street: Money Never Sleeps
555                    The Warrior's Way
1211            The Stranger by the Beach
1419                  Babysitter Must Die
```

```
genre_recommendations('Coco').head(10)
```

```
22                 Tangled
38              Toy Story 3
49            Puss in Boots
73               The Smurfs
87                  Bandage
111    What's Your Number?
128           Elliot Loves
180                Neowolf
185            Jack & Diane
193                   Dukun
```

These recommendations do a good job of incorporating the genres but are limited to just recommending the genre or genres that the title falls in which is good, but maybe not as in depth or exploratory as expected.

The second recommender based on a nearest neighbor model was also used on the same titles to start and using the subset dataset for just movies made after 2010.

```
get_knn_rec('Suicide Squad')
```

```
Recommendations for Suicide Squad:

1: Fullmetal Alchemist
2: Troy the Odyssey
3: Doctor Strange
4: The 25th Reich
5: Survivor
6: Hellboy
7: Bleach
8: The Mortal Instruments: City of Bones
9: Gods of Egypt
```

```
get_knn_rec('The Avengers')
```

```
Recommendations for The Avengers:

1: Revolt
2: Rampage
3: Moontrap: Target Earth
4: Solis
5: Cosmic Sin
6: Assassin's Creed
7: Power Rangers
8: Transformers: Age of Extinction
9: Oblivion
```

```
get_knn_rec('Coco')
```

```
Recommendations for Coco:

1: Ice Age: Continental Drift
2: Minuscule - Mandibles from Far Away
3: Woody Woodpecker
4: A Monster in Paris
5: Speckles: The Tarbosaurus
6: Angela's Christmas Wish
7: Shaun the Sheep Movie
8: The Wild Life
9: Oggy and the Cockroaches: The Movie
```

This recommender used both the genre and the average rating to recommend movies. A warning was also able to be created for this recommender system. Seeing as it was more subset, there would be a chance of people putting in movies that were not in the subset. There is also the chance that people do not spell it correctly or use proper syntax when typing the title name so the recommender will tell the user that the title was not found for example, when Matrix was searched for it informed the user that Matrix was not found. The other benefit of this

recommender system is that it allowed for the entire dataset to be used and not the subset. This allowed for more things to be in the recommendation list and provided these recommendations for Coco:

```
get_knn_rec2('Coco')
```

```
Recommendations for Coco:

1: The Lion Guard: Return of the Roar
2: Woody Woodpecker
3: The Littl' Bits
4: HarmonQuest
5: The Twisted Tales of Felix the Cat
6: Mune: Guardian of the Moon
7: Muppet Babies
8: Marsupilami
9: Spirit Untamed
```

This also allowed for more movies to be searched for including older movies like The Sound of Music:

```
get_knn_rec2('The Sound of Music')
```

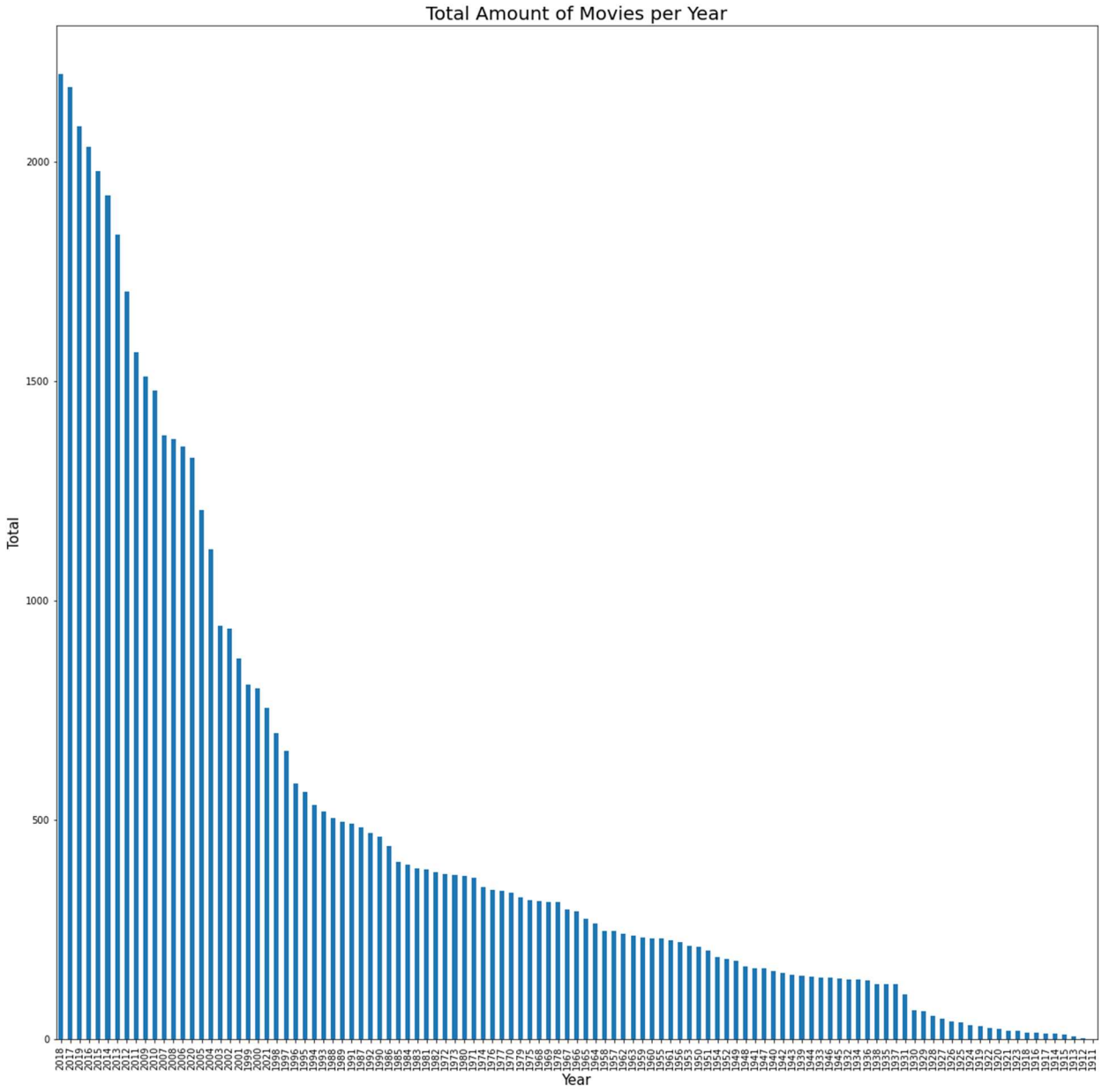
```
Recommendations for The Sound of Music:

1: Right on Track
2: Finding Neverland
3: Beautiful
4: A Street Cat Named Bob
5: The von Trapp Family: A Life of Music
6: A Man Called Peter
7: Thérèse: The Story of Saint Thérèse of Lisieux
8: Oshin
9: Empire of Silver
```

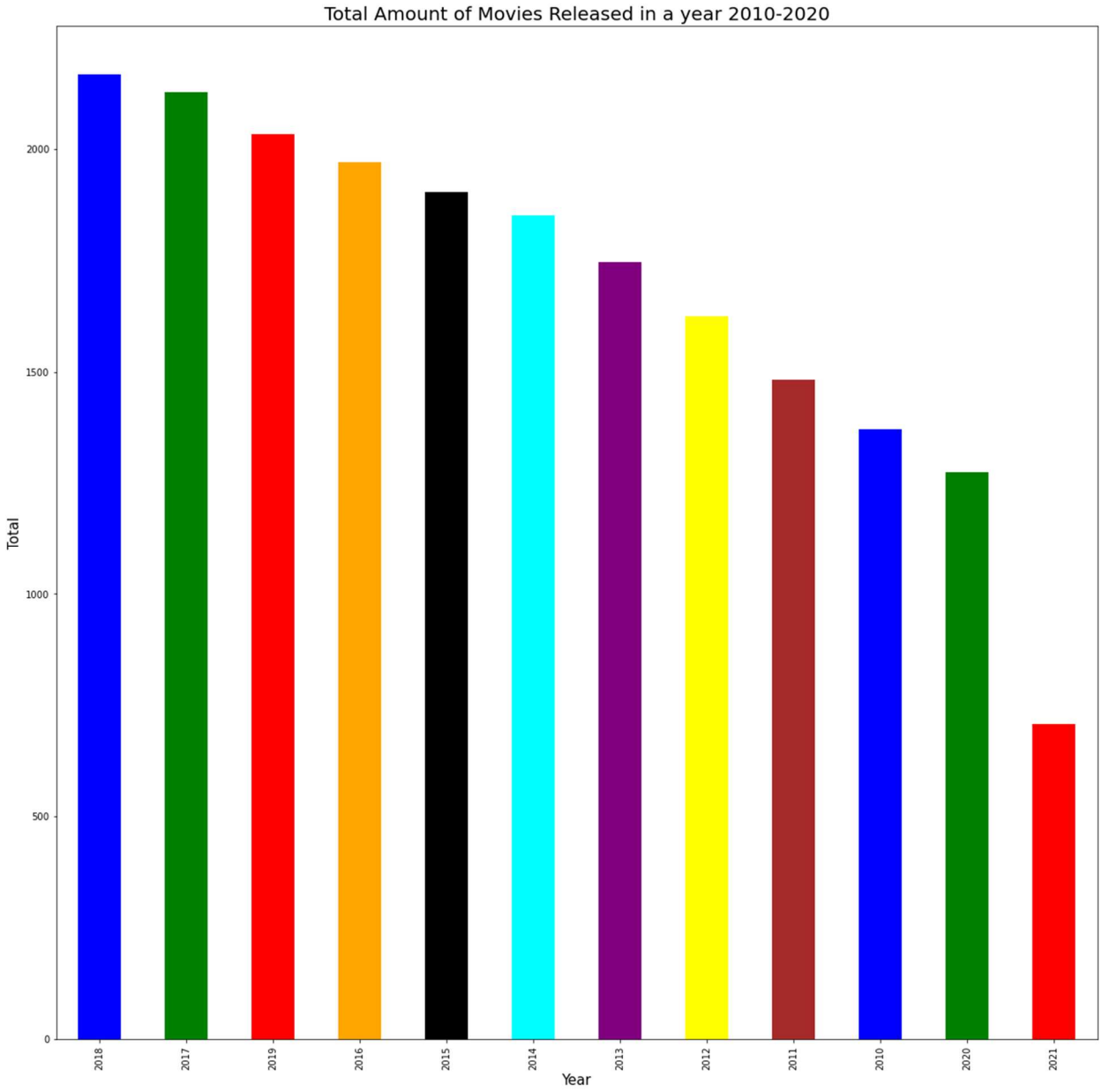
Discussion/Conclusion

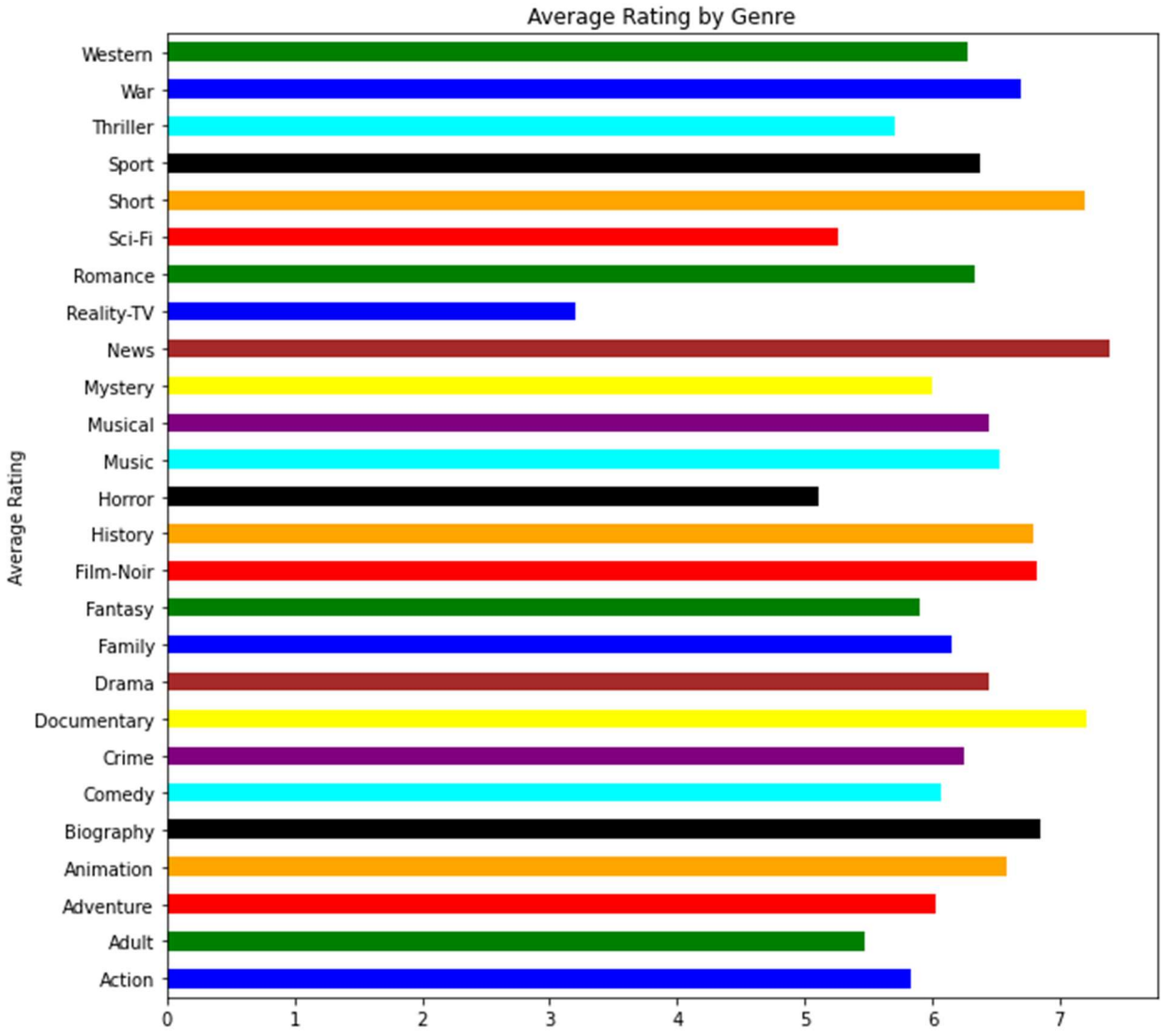
The movie recommender system works by finding similar titles based on certain parameters. For the content-based system that was created the genres were used and based on that the recommendations were created. This worked well, however being able to include more content like descriptions or cast/crew information might help the recommendations improve. For

the hybrid approach that was created being able to use both the genres and the ratings increased the quality of the recommendations and because a larger data set was able to be used also increased the options for what would be recommended and what titles can be searched for. However, again the recommender is limited to just the specific genre or combination of genres and cannot necessarily look beyond that. Looking at the specific titles that were searched for the recommendations are good and when you think about narrowing down the field from over 100,000 options to 10 that is a good system with the possibility of room for improvement and who knows it might even recommend a title that the user never would have thought of and will thoroughly enjoy.

Appendix


Total Amount of Movies per Year
Total
Year
0
500
1000
1500
2000


Total Amount of Movies Released in a year 2010-2020
Total
Year
0
500
1000
1500
2000
2018
2017
2019
2016
2015
2014
2013
2012
2011
2010
2020
2021



Resources:

IMDb.com. (n.d.). IMDb. Retrieved October 30, 2021, from https://m.imdb.com/interfaces/.

Langenderfer, J. (2021, May 3). *Building a movie recommendation system using Python*. Medium. Retrieved October 30, 2021, from https://medium.com/web-mining-is688-spring-2021/building-a-movie-recommendation-system-using-python-c3e352ccc19e.

Le, J. (2018, June 11). *The 4 recommendation engines that can predict your movie tastes*. Medium. Retrieved October 30, 2021, from https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-109dc4e10c52.

Python Recommender Systems: Content Based & collaborative filtering recommendation engines. DataCamp Community. (n.d.). Retrieved October 30, 2021, from https://www.datacamp.com/community/tutorials/recommender-systems-python.

Real Python. (2021, June 5). *Build a recommendation engine with collaborative filtering*. Real Python. Retrieved October 30, 2021, from https://realpython.com/build-recommendation-engine-collaborative-filtering/.

Recommender Systems in Keras: Movie recommendations using Keras. Analytics Vidhya. (2021, May 2). Retrieved October 30, 2021, from https://www.analyticsvidhya.com/blog/2021/05/movie-recommendations-using-keras-recommender-systems/.

Sharma, N. (2021, September 21). *Recommender systems with python - part I: Content-based filtering*. Medium. Retrieved October 30, 2021, from https://heartbeat.comet.ml/recommender-systems-with-python-part-i-content-based-filtering-5df4940bd831.

Shetty, B. (n.d.). *An in-depth guide to how Recommender Systems work*. Built In. Retrieved October 30, 2021, from https://builtin.com/data-science/recommender-systems.

Tan, B. (2020, April 21). *How to build simple recommender systems in python*. Medium. Retrieved October 30, 2021, from https://medium.com/swlh/how-to-build-simple-recommender-systems-in-python-647e5bcd78bd.

Vidiyala, R. (2020, October 21). *How to build a movie recommendation system?* Medium. Retrieved October 30, 2021, from https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109.