SC1015

ML Disney+ Movie Recommendation System

KAVIPOOJA

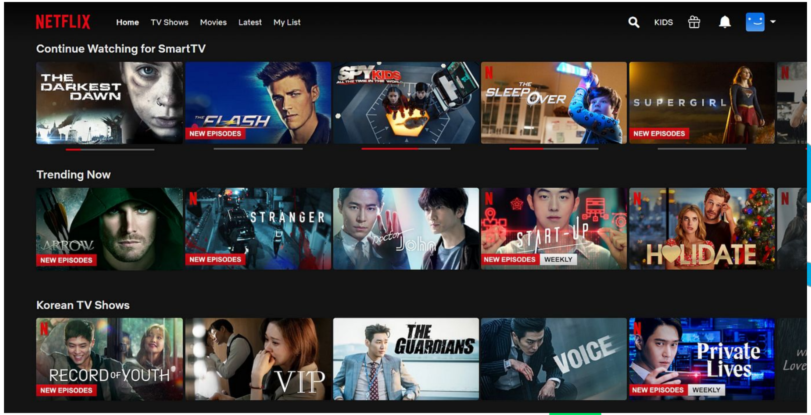
ESVARAN

PRIYADHARSHINY

C126_Team_4

Table of Contents

NETFLIX

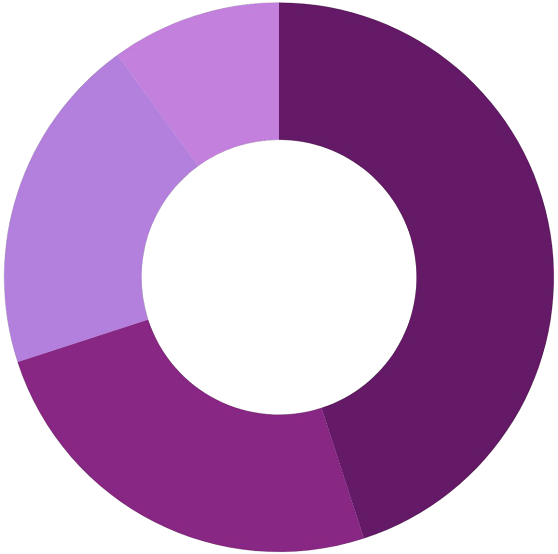


hulu

Problem Statement

How do we build a content-based recommendation system for Disney+ movies that accounts for similarities in genre and description?

DATA PREPARATION AND DATA CLEANING



First, let's clean the data set by removing all the unnecessary columns. This increases the accuracy and quality of the data and reduces the issue of redundancy and memory usage. The characteristics we removed aren't significant to the recommendation system.

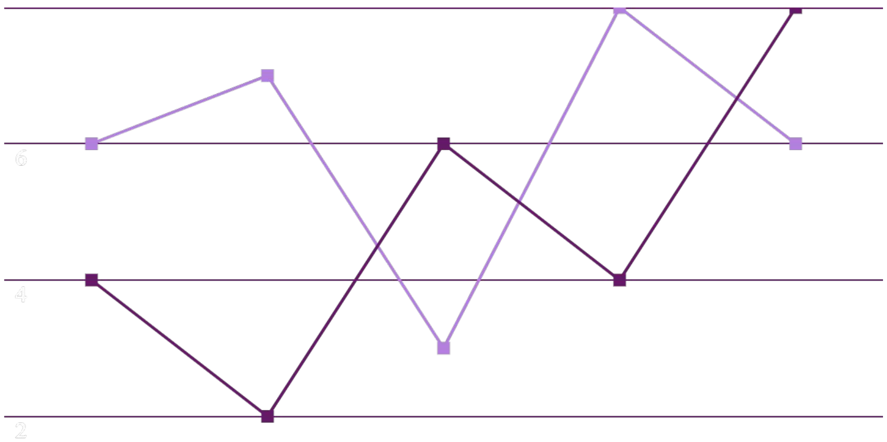
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1535 entries, 0 to 1534
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    1535 non-null   object
 1   title                 1535 non-null   object
 2   type                  1535 non-null   object
 3   description           1529 non-null   object
 4   release_year          1535 non-null   int64
 5   age_certification     1210 non-null   object
 6   runtime               1535 non-null   int64
 7   genres                1535 non-null   object
 8   production_countries  1535 non-null   object
 9   seasons               415 non-null    float64
 10  imdb_id               1133 non-null   object
 11  imdb_score            1108 non-null   float64
 12  imdb_votes            1105 non-null   float64
 13  tmdb_popularity       1524 non-null   float64
 14  tmdb_score            1426 non-null   float64
dtypes: float64(5), int64(2), object(8)
memory usage: 180.0+ KB
```

After dropping the columns: **'release_year', 'seasons', 'production_countries', and 'imdb_id'**

Dropping of some duplicates under some variables and also extraction of required variables for data analysis.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1118 entries, 0 to 1534
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  1118 non-null   object
 1   title               1118 non-null   object
 2   type                1118 non-null   object
 3   description         1118 non-null   object
 4   age_certification   862 non-null    object
 5   runtime             1118 non-null   int64
 6   genres              1118 non-null   object
 7   imdb_score          747 non-null    float64
 8   imdb_votes          744 non-null    float64
 9   tmdb_popularity     1115 non-null   float64
 10  tmdb_score          1075 non-null   float64
dtypes: float64(4), int64(1), object(6)
memory usage: 104.8+ KB
```

EXPLORATORY DATA ANALYSIS AND VISUALIZATION



TOP 5 *imdb_score*

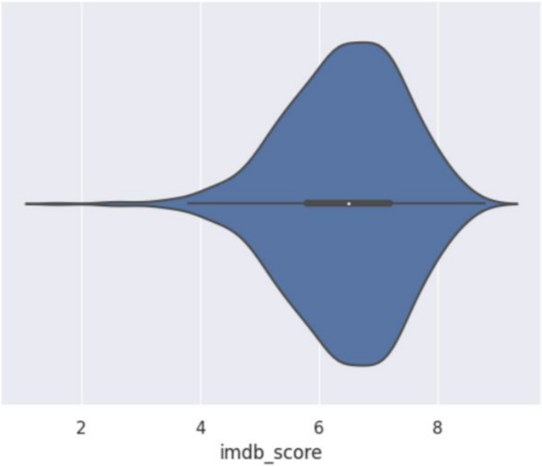
	IMDB Score float64
ASSEMBLED: The Making of Hawkeye	8.8
The Empire Strikes Back	8.7
Star Wars	8.6
The Lion King	8.5
Avengers: Endgame	8.4

Table

From **analysis of the table,**

- The top movie is ASSEMBLED: The Making of Hawkeye with the highest IMDB score of 8.8.
- The top 5 movies range from an IMDB score of 8.4 to 8.8.

Exploring *imdb_score*



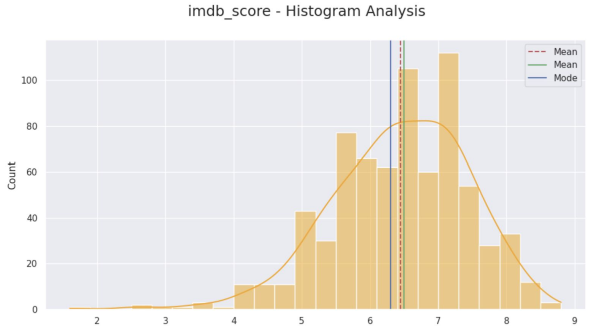
Violin Plot

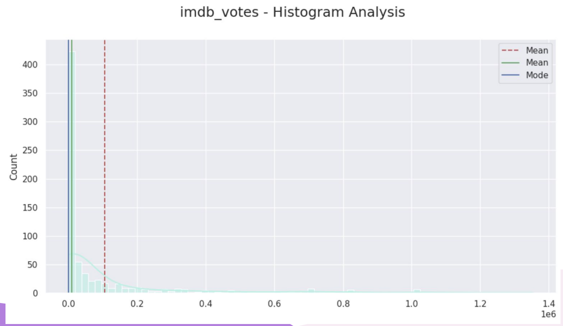
Violin Plot - illustrate the density of data at different values

From the **violin plot,**

- The ratings that have the highest frequency lie between 6 and 8.
- The ratings that have the lowest frequency lie between 4 and 6.

Analysis of *central tendency* across variables





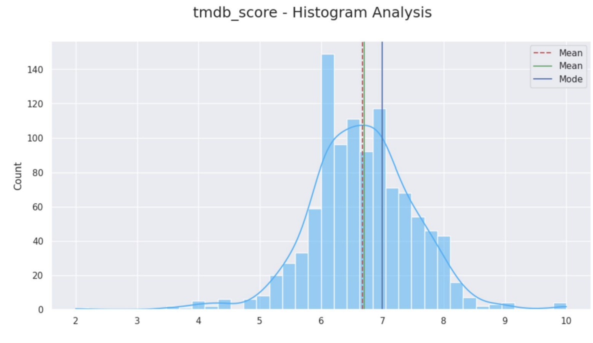
Histogram Plot - provides a visual representation of distribution of data

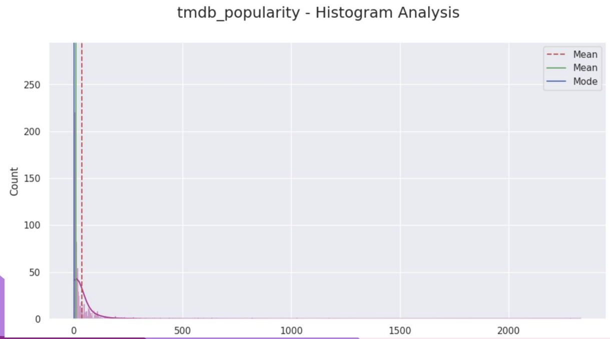
From the **histplot**, showing the central tendency of the data we are able to identify the:

- distribution of data
- the mean, median & mode
- Variables: imdb_score, imdb_votes, tmdb_popularity and tmdb_score.

Histogram

Analysis of *central tendency* across variables





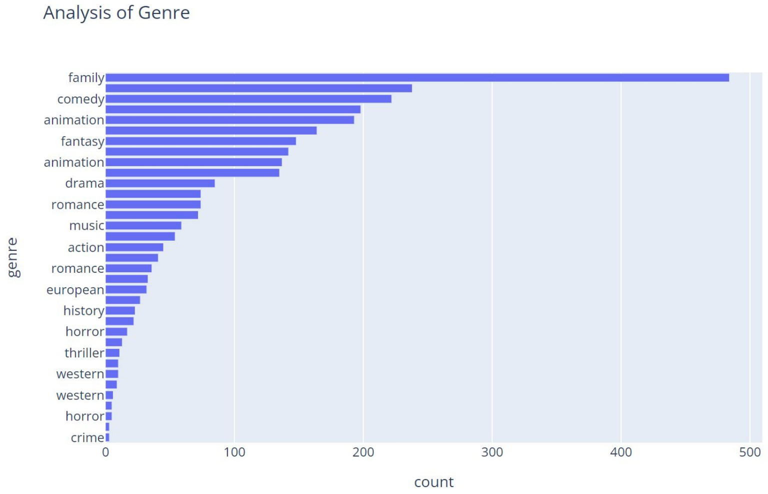
Histogram Plot - provides a visual representation of distribution of data

From the **histplot**, showing the central tendency of the data we are able to identify the:

- distribution of data
- the mean, median & mode
- Variables: imdb_score, imdb_votes, tmdb_popularity and tmdb_score.

Histogram

Exploring *genre*

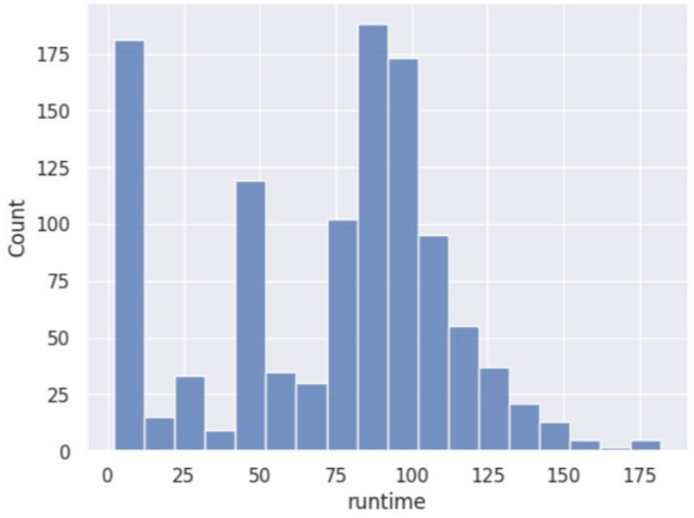


Bar Chart Plot

From the **Barchart plot** (ordered from most to least popular genre, in descending order)

- The most popular genre is "family" with a count of 484.
- "comedy", "animation", "fantasy" fall as the next three genres after "family".
- The least popular genre is "crime" with a count of 3.

Exploring *runtime*



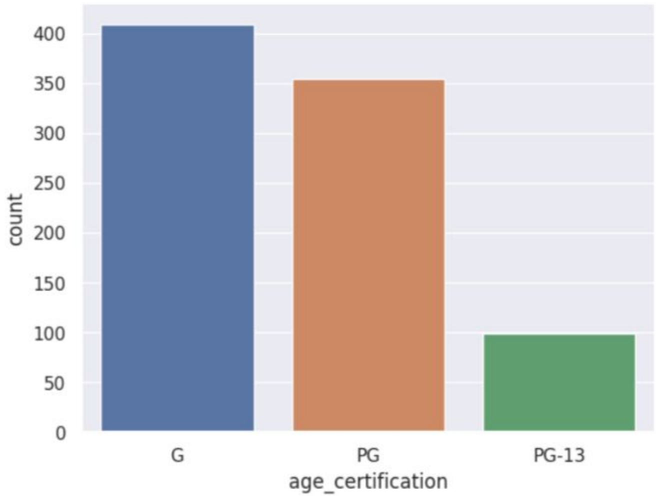
Histogram Plot

Histogram Plot - provides a visual representation of distribution of data

From the **histogram plot**,
(The runtime of the movies are given in terms of minutes.)

- More than 175 movies have a runtime that is above 87 and below 100.
- Next to that, close to 175 movies have a runtime that is between 0 and 12.
- Movies whose runtime is between 150 and 175 are the ones which are lower in frequency.

Exploring *age_certification*



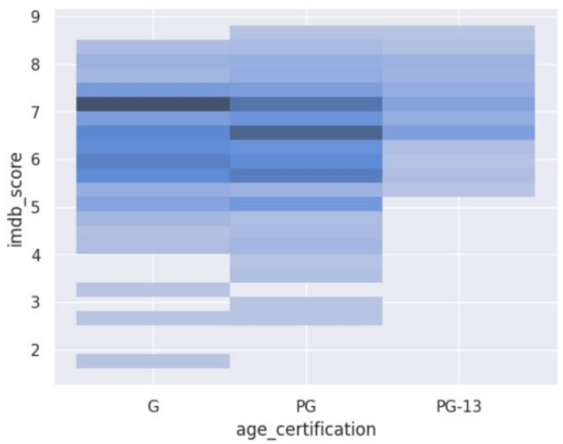
Count Plot

Count Plot - frequency of categorical values can be identified.

From the **count plot**,

- Almost 100 movies are categorized under PG-13.
- Almost 350 movies are categorized under PG.
- Almost 400 movies are categorized under G.

Exploring the relationship between *imdb_score* and *age_certification*

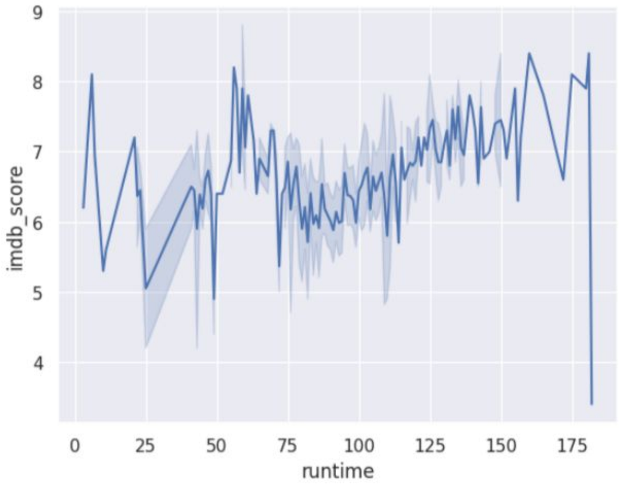


Bivariate Hist Plot with Heatmap

Hist Plot with Heatmap - 2-D space that provides an illustration of the relationship between two variables and shows the most concentrated observations.

- It is evident that movies with an age certification of G have a greater imdb_score compared to the others.
- This trend is observable as there is a larger concentration of color at that section.
- Next to that, a higher imdb_score has been observed for PG that is between 6 and 7 and slightly above 7.

Exploring the relationship between *runtime* and *imdb_score*



Line Plot

Line Plot - show how a variable changes periodically and how one variable is influenced by another.

From the **Line Plot**,

- There is an evident relationship between imdb_score and runtime.
- It's clear that movies with a runtime between 150 and 175 minutes have a higher imdb_score.
- Similarly, movies with a runtime between 0 and 25 minutes and between 50 and 75 minutes have a higher imdb_score.

MACHINE LEARNING TECHNIQUES

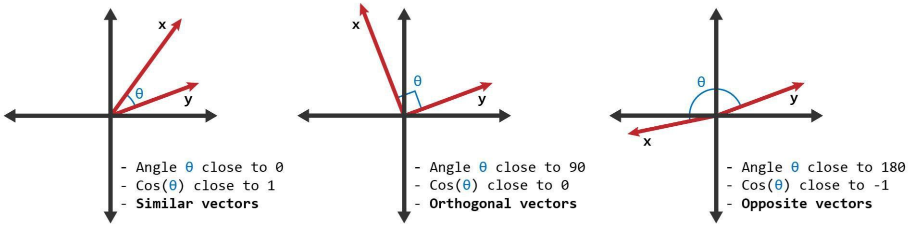
Vectorization Techniques

Textual Analysis

Document Similarity is one of the key metrics involved in textual analysis, and it represents the foundation for our recommendation system.

Understanding Cosine Similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



TF-IDF Vectorizer

Increases the importance of a particular word w if it's **unique** to a movie's description, thereby decreasing the significance of common words, such as 'a', 'the', 'and', etc.

```
[[1.         0.04257928 0.02734134 ... 0.0185537  0.03768828 0.02471722]
 [0.04257928 1.         0.0187942  ... 0.16016232 0.04856555 0.03695879]
 [0.02734134 0.0187942  1.         ... 0.0280489  0.01541093 0.01625423]
 ...
 [0.0185537  0.16016232 0.0280489  ... 1.         0.04682715 0.03997809]
 [0.03768828 0.04856555 0.01541093 ... 0.04682715 1.         0.03124309]
 [0.02471722 0.03695879 0.01625423 ... 0.03997809 0.03124309 1.        ]]
```

Count Vectorizer

Denotes the frequency of each word within the description, providing **equal importance** to all words, including common occurrences.

```
[[1.         0.38334909 0.18731716 ... 0.25       0.38385797 0.28401878]
 [0.38334909 1.         0.16571045 ... 0.42389562 0.41702883 0.28552012]
 [0.18731716 0.16571045 1.         ... 0.14048787 0.15137513 0.14509525]
 ...
 [0.25       0.42389562 0.14048787 ... 1.         0.32829958 0.16137431]
 [0.38385797 0.41702883 0.15137513 ... 0.32829958 1.         0.31298432]
 [0.28401878 0.28552012 0.14509525 ... 0.16137431 0.31298432 1.        ]]
```

Comparison Between Vectorizers

The purpose of constructing different matrices is to illustrate how recommendation systems can **vary vastly** depending on the mechanism we utilize to assign **importance** to words within the string.

Only TF-IDF

Only Count

TF-IDF + Count

Recommendations Based on The Little Mermaid

	title
332	The Little Mermaid II: Return to the Sea
205	Splash
425	The Chronicles of Narnia: Prince Caspian
507	The Little Mermaid: Ariel's Beginning
290	The Brave Little Toaster to the Rescue
512	George of the Jungle 2
5	The Adventures of Ichabod and Mr. Toad
126	The Story of Robin Hood and His Merrie Men
701	The Day the Series Stopped
775	Ice Age: Collision Course

	title
290	The Brave Little Toaster to the Rescue
35	The Grasshopper and the Ants
425	The Chronicles of Narnia: Prince Caspian
215	The Ewok Adventure
5	The Adventures of Ichabod and Mr. Toad
447	Confessions of a Teenage Drama Queen
1087	Russia's Wild Tiger
794	Of Miracles and Men
607	Sacred Planet
799	Drain The Ocean: WWII

	title
425	The Chronicles of Narnia: Prince Caspian
332	The Little Mermaid II: Return to the Sea
290	The Brave Little Toaster to the Rescue
205	Splash
5	The Adventures of Ichabod and Mr. Toad
35	The Grasshopper and the Ants
701	The Day the Series Stopped
762	Avengers: Age of Ultron
126	The Story of Robin Hood and His Merrie Men
775	Ice Age: Collision Course

Recommendation System

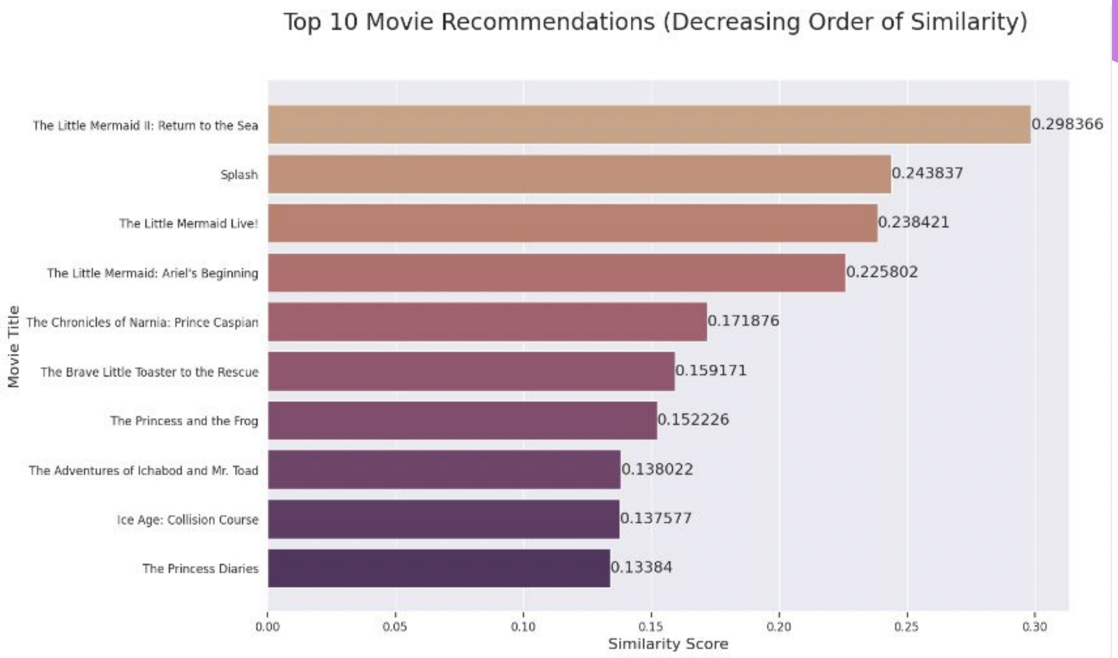
Using the TF-IDF vectorizer, we're going to develop our final recommendation system that'll account for certain characteristics we didn't previously acknowledge.

In addition to analyzing the description, we'll account for the **movie title** (for recommending sequels) and the **genre**. This will permit the system to make **higher-quality** recommendations that consider the user's interests as well.

```
qualities = ['title', 'description', 'genres']

def accumulate_string(frame):
    combine = "";
    for feature in qualities:
        combine += frame[feature] + ' \ ';

    return combine;

disTitles['description'] = disTitles.apply(accumulate_string, axis = 1)
disTitles['description'].head()
```

INSIGHTS AND RECOMMENDATIONS

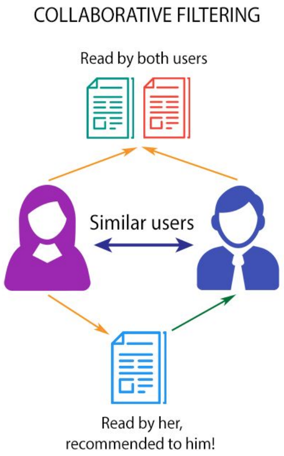

THE LITTLE MERMAID
Walt Disney
THE LITTLE MERMAID II
RETURN TO THE SEA
ICE AGE
COLLISION COURSE
Two days ago, this girl showed up naked at the Statue of Liberty.
For Alan Bauer, it was love at first sight.
Now, everyone is chasing her...
trying to prove she's a mermaid.
From the first laugh you'll be hooked.
Splash
THE PRINCESS AND THE FROG



Similarities of Our Final Recommendations

Collaborative Filtering

Content-based filtering (**our approach**) accounts for the characteristics of the film, including its description, genre, and actors; however, it fails to represent the user's preferences. In order to address these restraints, we introduce a higher-level approach called collaborative filtering, which accounts for **similarities between users** as well.



Accounting for User’s Preferences

Genres User Likes

Favorite Actors

Trending Films

We can filter out films that aren't within the scope of the search space, eliminating choices that the user won't appreciate anyways.

Conclusion

- The movie recommendation system is a convenient and personalized way to discover new films.
- Gives suggestions based on their preferences, prior search histories, the movies they have watched, and various other factors.
- Saves users' time and effort in searching for movies.
- Saves costs for users to watch movies that might interest them.

Thank You!