Content-based Disney+ Movie Recommendation System

Introduction:

A content-based recommendation system for Disney+ movies will help users discover movies on a platform that aligns with their interests based on watched movies.
Recommendation systems and data charts for movies can give useful ideas about what audiences like and trends in content. But thoughtfully building these technologies requires solving key problems around information quality, unfair bias, and responsible use. This report looks at a movie data set using Python, balancing technical details with easy-to-understand explanations. By showing release years, reviews, genres, actors, and production countries, we get a view of the data's features. The recommendation system suggests similar movies based on content. While powerful, these techniques risk continuing biases or bubbles. We must be careful to build inclusive, transparent systems that improve human creativity. Overall, this project shows how data science can reveal new patterns while telling interesting stories about our shared culture.

Steps to Build the System:

1. Data Collection & Cleaning:

Collect the data: use open databases like Kaggle to download Disney+ data.
Clean the data: remove duplicates, handle missing values, and convert data into usable formats.

2. Exploratory Data Analysis (EDA):

Analyze the distribution of genres, popularity of directors, frequency of actors, etc., specifically for Disney+ movies.
This will provide insights into the structure of your Disney-specific data.

3. Feature Engineering:

Extract relevant features that encapsulate the content of a Disney+ movie. This could involve: Using NLP techniques to process and vectorize movie descriptions or plot summaries. One-hot encoding categorical data like genres. Aggregating cast, director, and other relevant information.

4. Computing Similarities:

Compute similarity scores between movies using metrics such as: cosine similarity.

5. Recommendation Generation:

Recommend the top 'N' movies with the highest similarity scores for a given Disney+ movie. Alternatively, create a user profile based on Disney+ movies the user liked and generate a list of recommendations by comparing this profile with all available movies.

1. Data collection & Cleaning

Looking through available datasets on Kaggle, the one with the latest update date from Diego Enrique was chosen. There were two files ***titles.csv*** and ***credits.csv.***

The first one contains the following columns:
id: The title ID on JustWatch.
title: The name of the title.
show_type: TV show or movie.
description: A brief description.
release_year: The release year.
age_certification: The age certification.
runtime: The length of the episode (SHOW) or movie.
genres: A list of genres.
production_countries: A list of countries that - produced the title.
seasons: Number of seasons if it's a SHOW.
imdb_id: The title ID on IMDB.
imdb_score: Score on IMDB.
imdb_votes: Votes on IMDB.
tmdb_popularity: Popularity on TMDB.
tmdb_score: Score on TMDB.

While the second one:
person_ID: The person ID on JustWatch.
id: The title ID on JustWatch.
name: The actor or director's name.
character_name: The character name.
role: ACTOR or DIRECTOR.

For cleaning the data firstly missing data from datasets should be identified. From the code included in ***data_visualisation.py*** file. The identified missing data are following:

Credits Data (credits.csv):
character: 1895 missing values.

Titles Data (titles.csv):
description: 9 missing values.
age_certification: 451 missing values.
seasons: 1314 missing values (likely because these are movies and not series).
imdb_id: 478 missing values.

imdb_score: 515 missing values.
imdb_votes: 526 missing values.
tmdb_popularity: 15 missing values.
tmdb_score: 146 missing values.

For the character column in the credits data, we can leave the missing values as they are since they might indicate non-character roles (like directors, producers, etc.). For the description column in the titles data, we can fill missing descriptions with a placeholder string like "No description available". The age_certification can be filled with a placeholder like "Not Specified". The seasons column can be dropped since we're focusing on movies.
For the IMDb and TMDB columns missing values can be left as they are until further investigation.

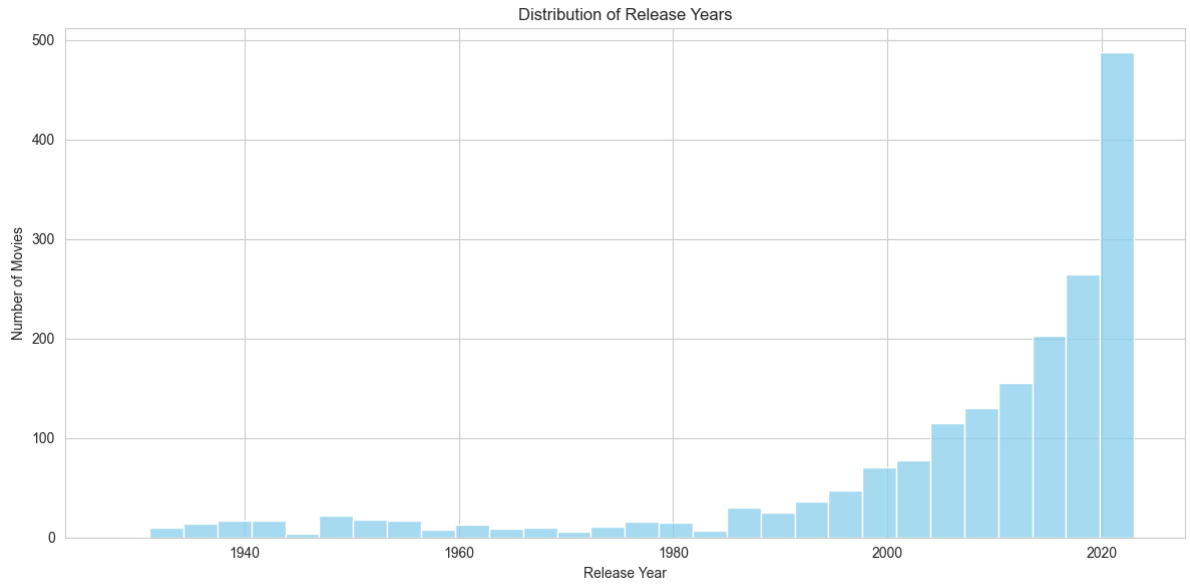
After that tiles and credits data frames were cleaned and merged into ‘merged_df’, the merged data frame has the following information for each movie:

Movie details (title, description, release year, etc.)
Genres and production countries
IMDb and TMDB scores and ratings
Actor details and their roles

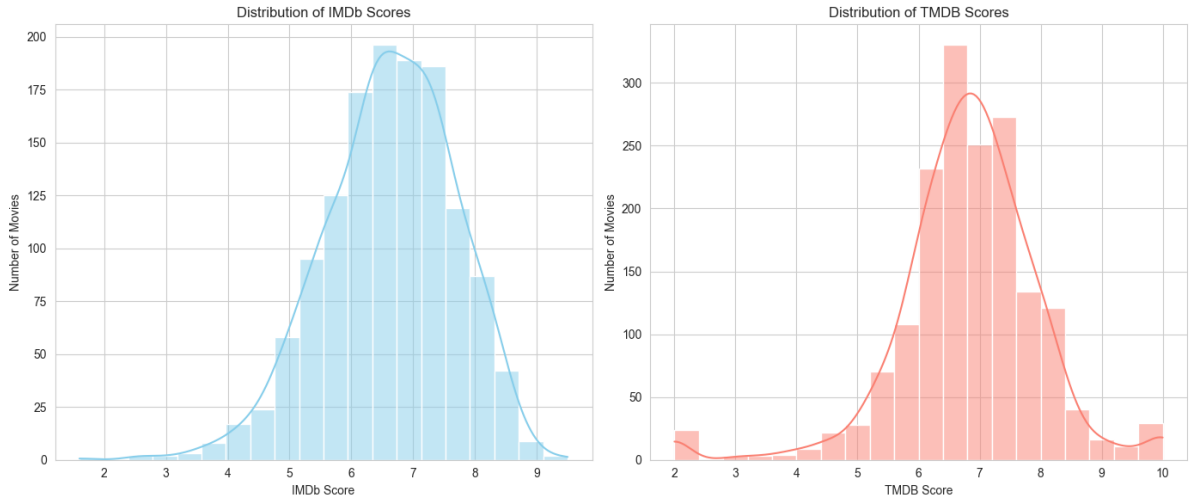
2. Exploratory Data Analysis (EDA)

The EDA will include:
Distribution of Release Years: Understand the timeline of movies in our dataset.
Most Common Genres: Identify which genres are most prevalent.
Distribution of IMDb and TMDB Scores: Analyze the spread of movie ratings.
Most Frequent Actors: Identify the actors who have appeared most frequently.
Most Frequent Production Countries: Check where most of the movies are produced.

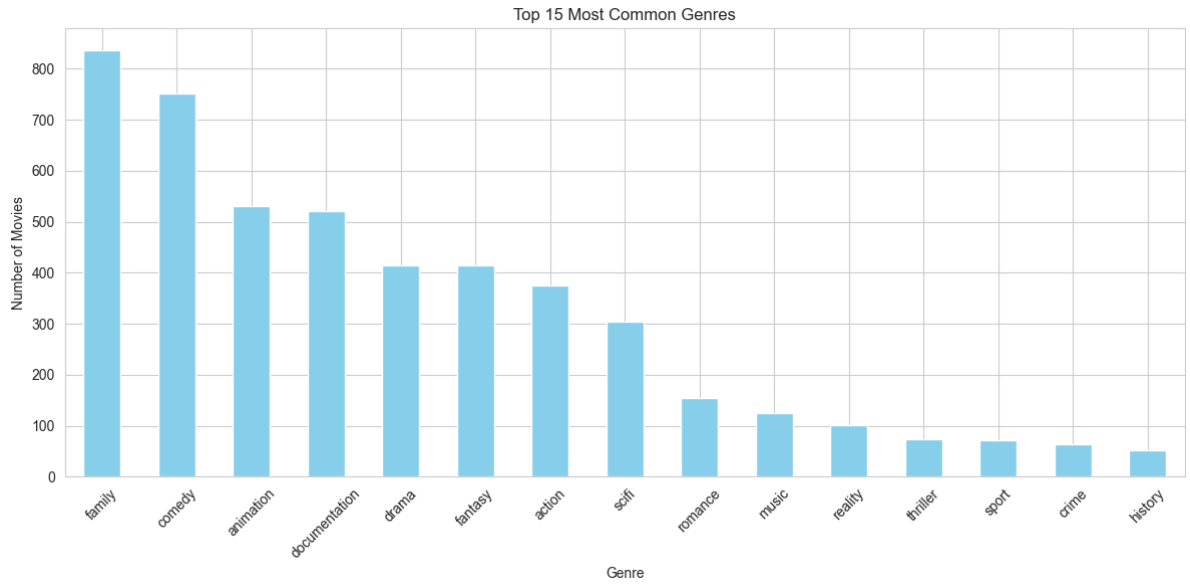
The code for the plots included in file *data_visualisation.py*



The histogram showcases the distribution of movies based on their release years in our dataset. We can observe that:
There's a significant number of movies from the early-mid 1900s.
The number of movies increases around the 2000s and peaks around the 2010s.



From the bar chart, we can see the top 15 most common genres in the dataset:
The most prevalent genres are Family, Animation, and Comedy.
Other popular genres include Action, Adventure, Drama, and Fantasy.



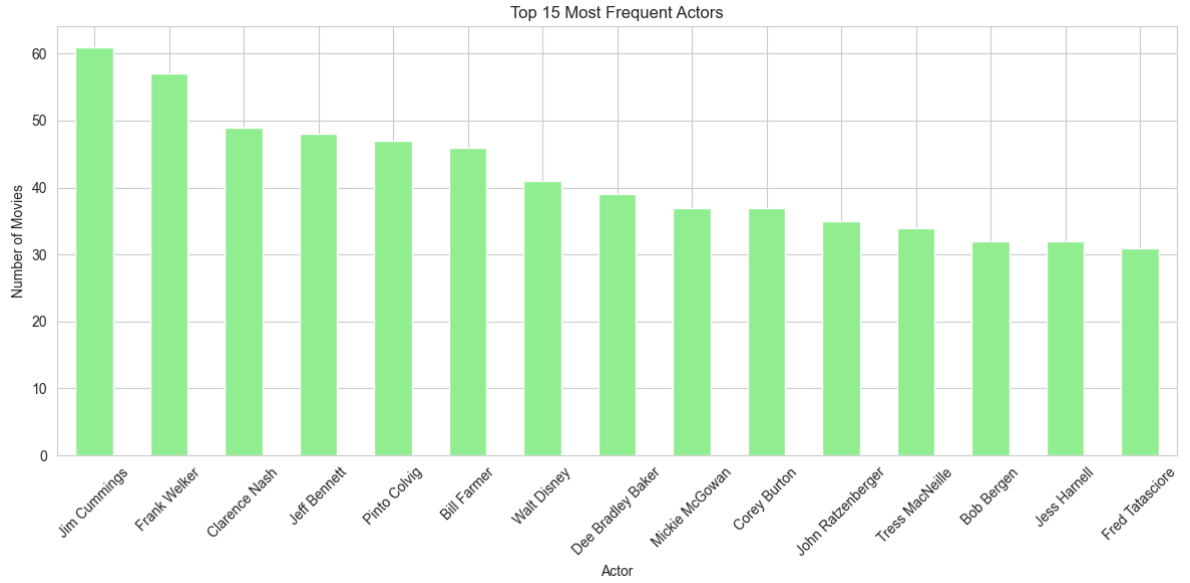
The histograms showcase the distribution of movie ratings from IMDb and TMDB:

IMDb Scores:

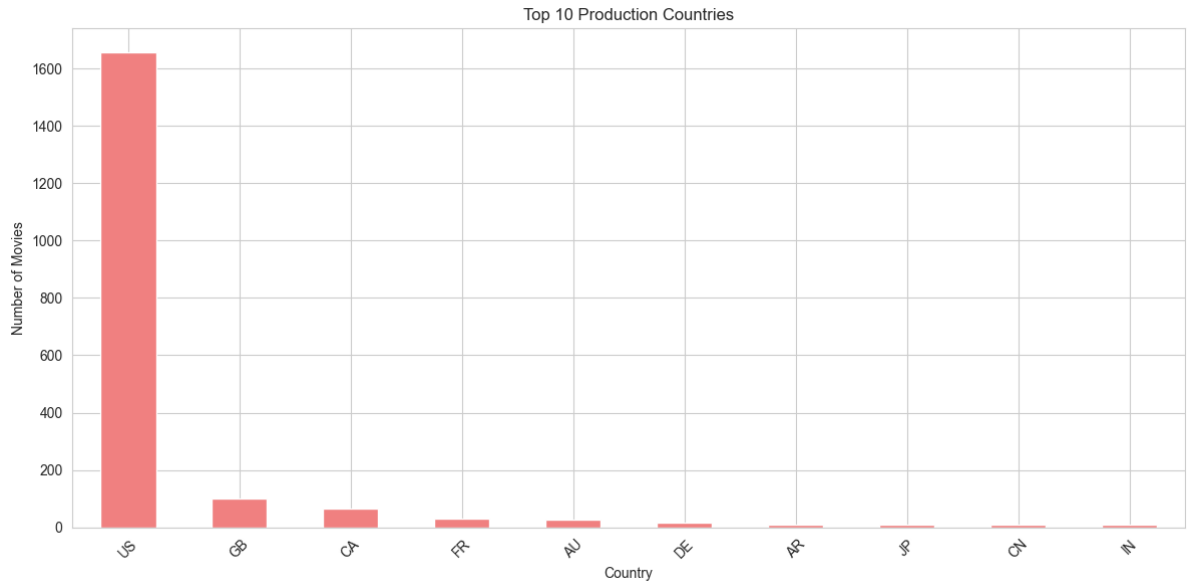
The majority of movies have ratings between 6 and 8.
Few movies have exceptionally high or low ratings.

TMDB Scores:

The distribution is quite similar to IMDb, with most movies having scores between 6 and 8.



The bar chart displays the top 15 most frequent actors in our dataset:
These actors have appeared in multiple movies, making them some of the most recognizable faces in the Disney+ collection.



The bar chart displays the top 10 production countries:

Unsurprisingly, the United States (US) is the dominant production country, producing the vast majority of movies in our dataset. This is expected since Disney is an American company. Other countries also contribute to the production, but their numbers are significantly smaller in comparison to the US.

3. Feature Engineering

Feature engineering involves transforming the data into a format that makes it easier to derive meaningful insights or predictions. For a content-based recommendation system, this usually means extracting features that describe the content. This project will include following features:

Text Vectorization: Convert the movie descriptions into a vector format using techniques such as TF-IDF (Term Frequency-Inverse Document Frequency). This will help us gauge the similarity between movies based on their descriptions.
One-Hot Encoding: Convert categorical data like genres into a one-hot encoded format. This will allow us to find similarities based on genres.
Aggregating Actor and Role Information: Process actor and role information to be used as features.
Normalization: Normalize numerical features like ratings to bring them to a similar scale.

The movie descriptions have been successfully vectorized using TF-IDF, resulting in a matrix where each row represents a movie and each column represents the TF-IDF weight of a specific word in the movie's description.

Next, we performed one-hot encoding for the genres. This will convert the genres into a format that can be used to calculate similarity between movies based on their genres.
Each genre now has its own column, with a "1" indicating the presence of that genre for a movie and "0" indicating its absence.

Next, aggregate the actor and role information. This will help us capture the cast details of each movie in a structured manner. The resulting data frame, cast_df, contains lists of actors and their corresponding roles for each movie.

As a final step in our feature engineering process, let's normalize the numerical features. This includes ratings like imdb_score and tmdb_score. Normalizing these features will ensure that they have a similar scale, which is essential for calculating similarities later.
The numerical features have been successfully normalized to a scale between 0 and 1.

4. Computing Similarities:

This step will include following:
Concatenate Features: Combine all the features we've prepared into a single data frame. This will give us a comprehensive representation of each movie.
Compute Similarity: Use a similarity metric to compute the similarity between movies based on our features. Cosine similarity is a commonly used metric for this purpose.
Generate Recommendations: For a given movie, find the top 'N' movies with the highest similarity scores.

Before presenting recommendation system the NaN values should be dropped, since they cause memory error.