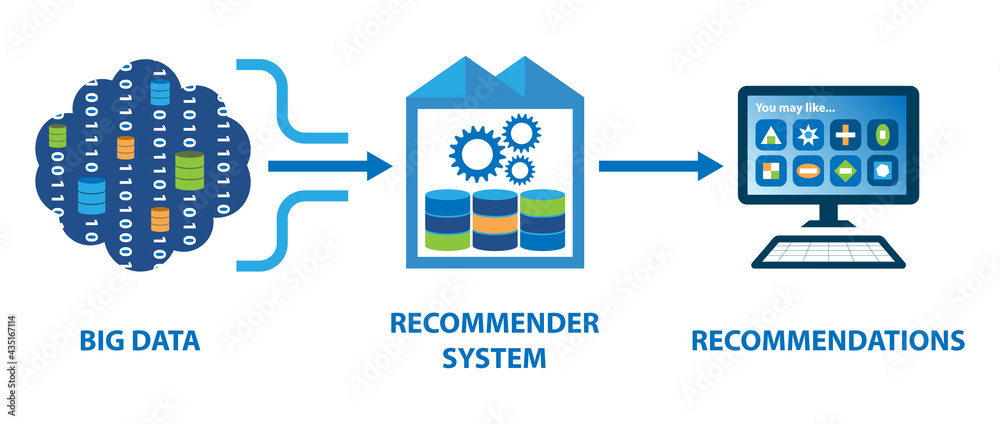
Capstone Project

Data Recommender System Recommendations

# Movie Recommendation System

By

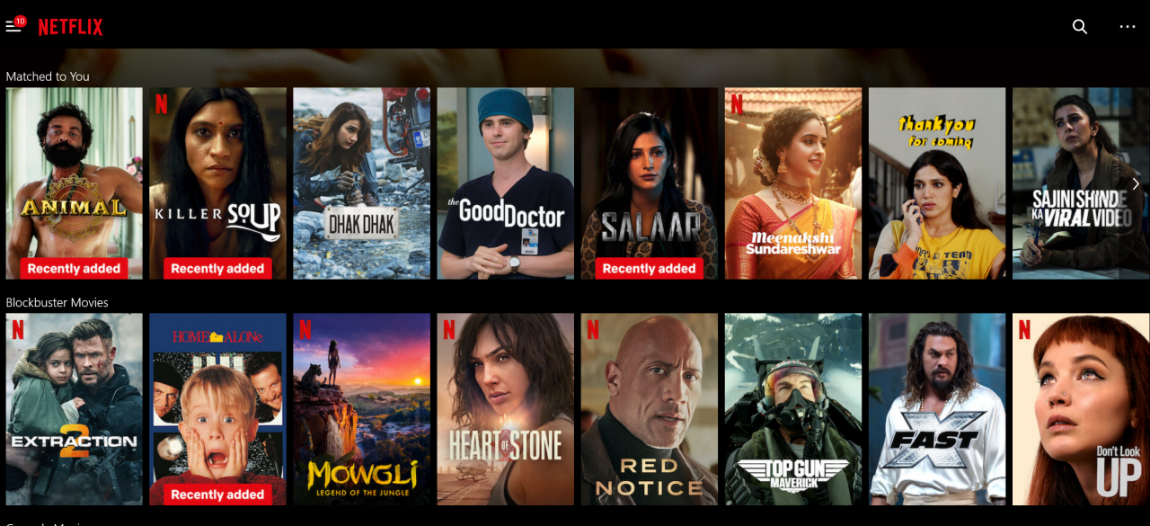
Rahul Pandey

Date : 10-Feb-2024

Table of Contents

1. **Problem Statement .................................................................................................... 3**
2. **Industry/Domain ........................................................................................................ 4**
3. **Stakeholders ......................................................................................................…….4**
4. **Business Question ..................................................................................................... 4**
5. **Data Question .....................................................................................................…….4**
6. **Data ......................................................................................................………………. 5**
7. **Data Analysis ............................................................................................……........... 5**
8. **Modelling ......................................................................................................………..8**
9. **Outcomes ......................................................................................................……… 14**
10. **Implementation .....................................................................................................…. 14**
11. **Data Answer ........................................................................................................ …..14**
12. **Business Answer ...................................................................................................... 14**
13. **Response to Stakeholders ......................................................................................... 15**
14. **End-to-End Solution ..........................................................……................................ 15**
15. **References .....................................................................................................……….. 15**

Problem statement:



In the current landscape of numerous streaming platforms, individuals often face challenges when deciding which movie to watch due to the overwhelming array of choices available. Existing recommendation systems employed by these platforms often fall short in providing accurate and personalized suggestions.

The issue lies in the simplicity of the algorithms used in these recommendation systems. They primarily rely on past viewing history to make suggestions, resulting in recommendations that may not align with individual preferences. Additionally, these systems tend to offer limited diversity in their suggestions, often presenting users with repetitive or irrelevant content.

Furthermore, there is a concern regarding the creation of 'filter bubbles,' wherein users are exposed only to content similar to their past choices. This limits exposure to new and diverse content, hindering the discovery of lesser-known titles.

To address these challenges, there is a need for a more sophisticated movie recommender system that utilizes advanced algorithms and user-centric approaches. Such a system would aim to deliver tailored recommendations based on individual preferences, while also promoting diversity and discovery in movie selection. This would enhance the overall user experience and satisfaction in navigating the vast landscape of movie options available on streaming platforms.

Industry/Domain:

Operating in the entertainment and media sector, our organization focuses on providing on-demand movie content through digital platforms. With an extensive library of films and TV shows, we prioritize delivering personalized recommendations to enhance user satisfaction. By analyzing user preferences and viewing habits, we aim to optimize content discovery while ensuring operational efficiency.

Our approach aligns with industries utilizing automated processes for production and delivery. Whether it's refining inventory management or enhancing customer engagement, our movie recommender system can be adapted to drive value across various sectors, emphasizing operational efficiency and superior customer experiences.

Stakeholders:

Our main stakeholders are the Chief Content Officer and the Head of User Experience. They aim to enhance user engagement by delivering personalized movie recommendations. Additionally, they seek to improve operational efficiency within the organization. Overall, they expect to see improvements in user satisfaction and platform performance.

Business Question:

The central business inquiry revolves around assessing the value proposition of implementing a movie recommender system. Specifically, the organization seeks to quantify the benefits in terms of user engagement and operational efficiency resulting from personalized movie recommendations. The question is: "What is the estimated value added by deploying a movie recommender system?"

The organization anticipates improvements in user satisfaction and platform performance through enhanced content discovery. The objective is to determine the potential impact of the recommender system on user engagement metrics, such as increased viewing time and higher user retention rates. Ultimately, the aim is to evaluate the return on investment in terms of improved user experience and platform usage.

Data Question:

The data question for the movie recommender system project is: "Which data variables are essential for predicting user movie preferences?"

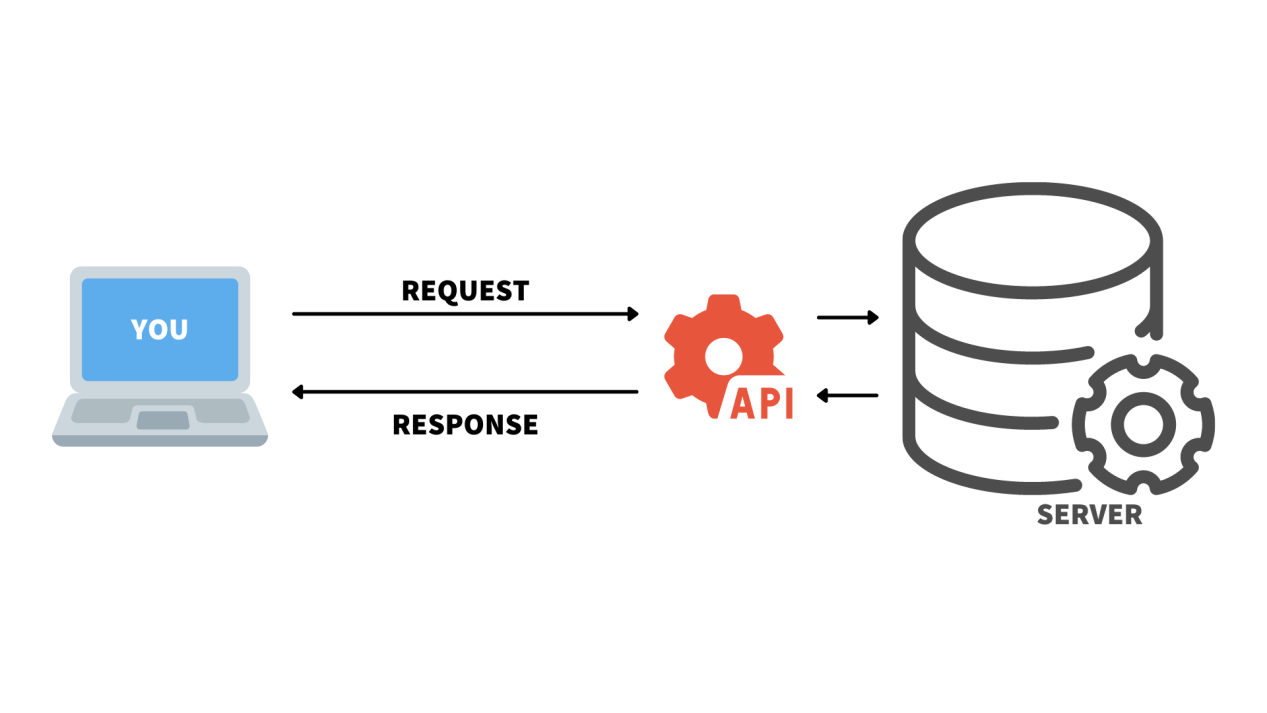
Key variables include user interactions (viewing history, ratings), movie metadata (genre, cast), user demographics (age, location), and user feedback (ratings, reviews). These data points enable personalized recommendations, enhancing user satisfaction and engagement.

Data Description, Source, and Quality:

To construct an effective movie recommendation system, we rely on comprehensive datasets sourced from reputable platforms and APIs. Our primary data sources include:

The Movies Dataset: This dataset, sourced from the TMDB Open API, is available on Kaggle. It encompasses movie details, credits, and keywords essential for our recommendation engine. Key features include movie\_id, cast (including lead and supporting actors), and crew (including director, editor, composer, writer, etc.).

TMDB 5000 Movie Dataset: Also available on Kaggle, this dataset is sourced from The Movie Database API. It provides extensive movie metadata crucial for our recommendation engine, with 19 columns containing essential information for further processing.



Both datasets have been vetted for quality and reliability, meeting our requirements for accurate analysis and model training. By leveraging these datasets, our recommendation system can effectively learn patterns in user viewing history and generate personalized movie recommendations, enhancing user satisfaction and engagement on our platform.

Data Analysis:

The dataset consists of 23 columns, with varying data types:

Integer: budget, id, revenue, vote\_count, movie\_id

Float: popularity, runtime, vote\_average

Object: genres, homepage, keywords, original\_language, original\_title, overview, production\_companies, production\_countries, release\_date, spoken\_languages, status, tagline, title, cast, crew

Key observations from the data include:

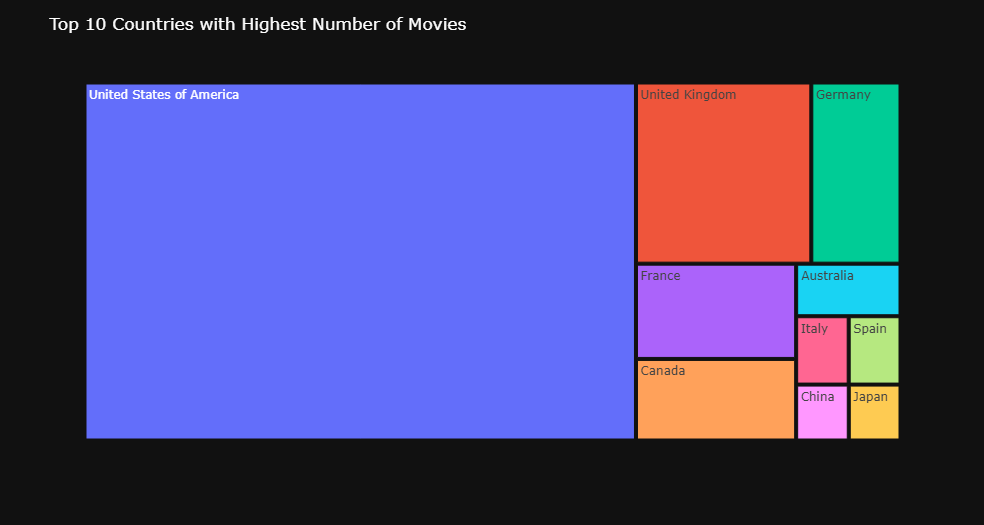
A significant portion of homepage and tagline columns contain null values.

The release\_date column has one null value.

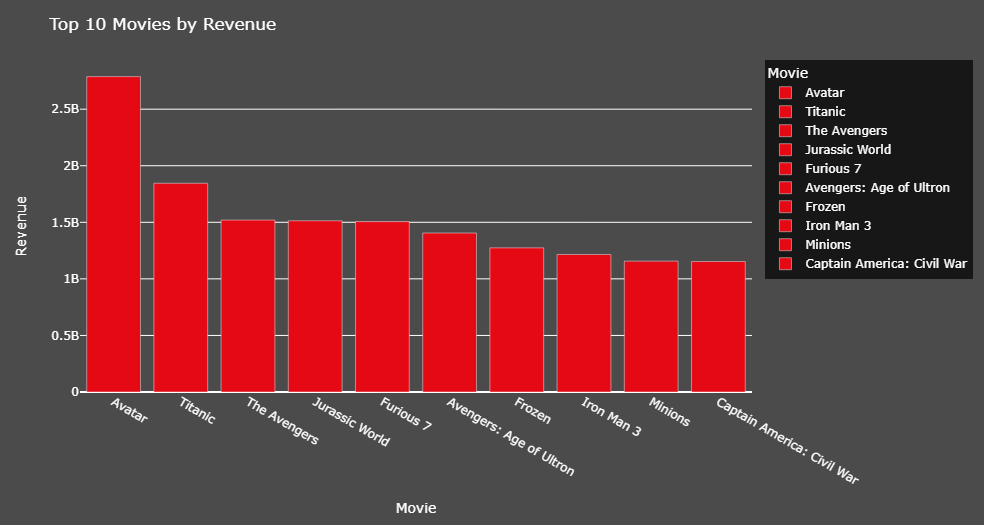
The runtime column has two null values.

The dataset includes essential movie attributes such as budget, genres, popularity, revenue, runtime, and vote counts.

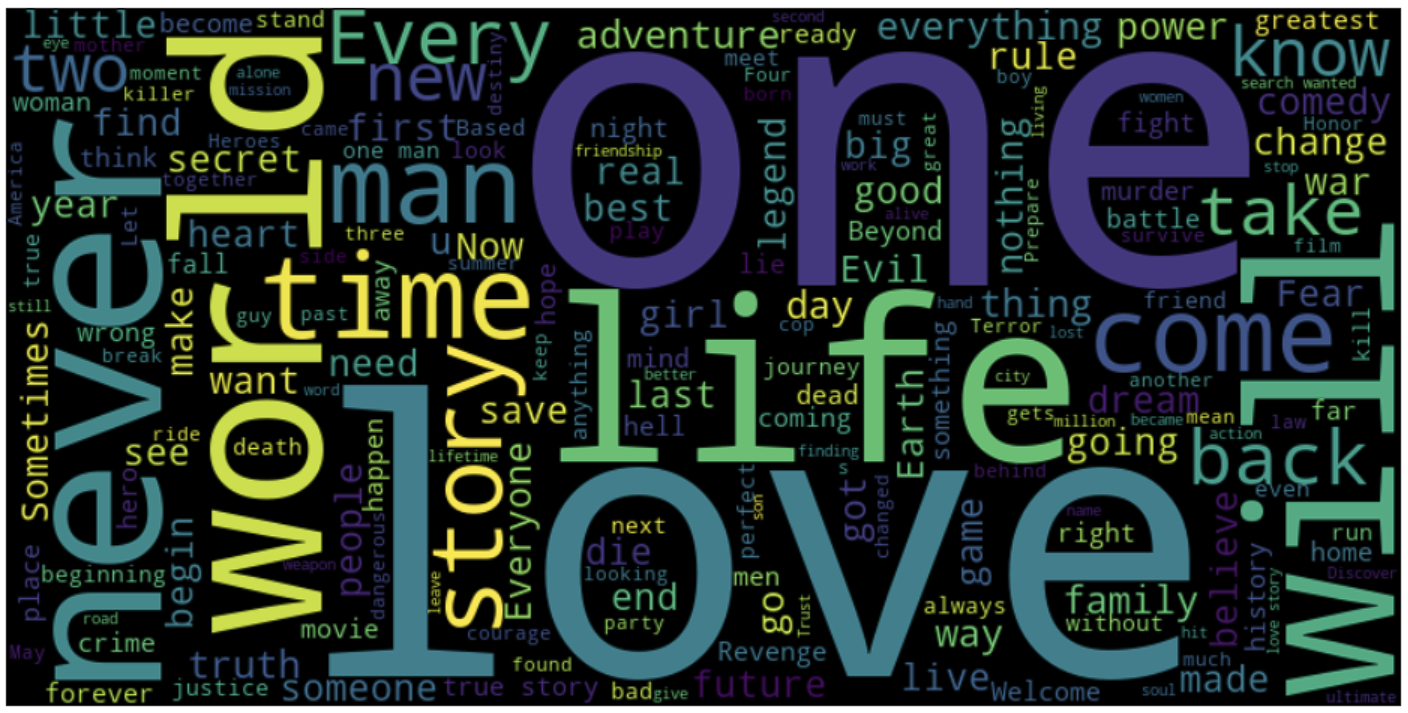
Further analysis will involve exploring relationships between variables, identifying trends in movie attributes, and extracting insights to inform the development of the movie recommender system.



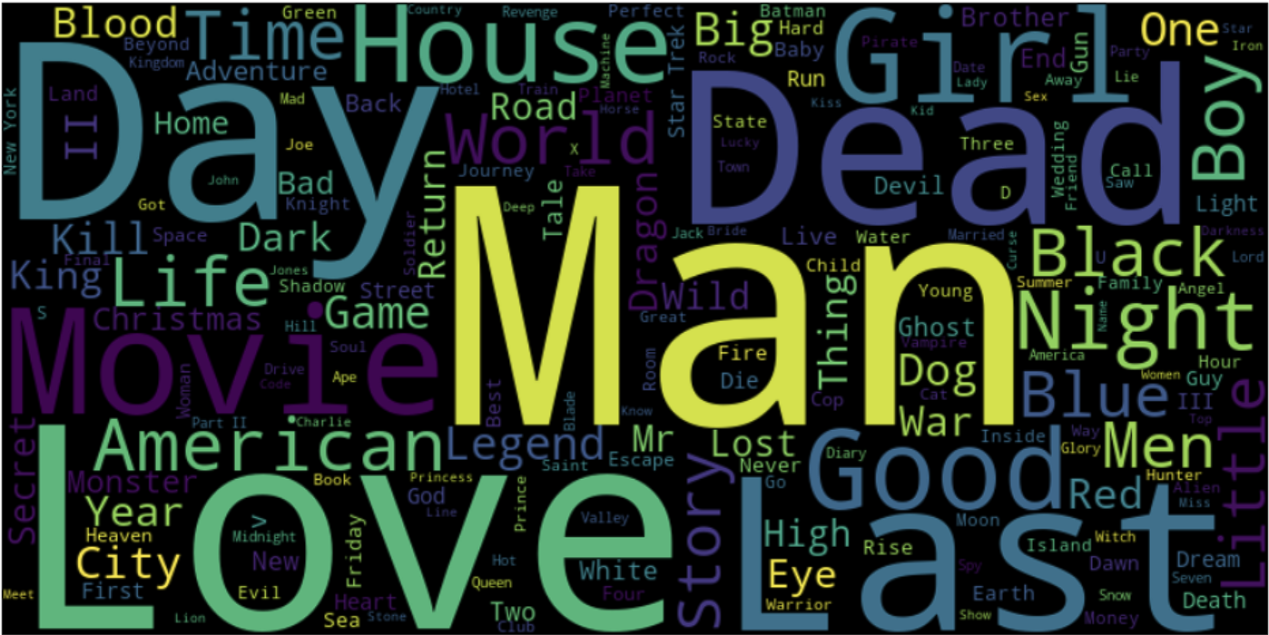
## **Top Countries with Highest Number of Movies**



## **Top 10 Movies by Revenue**

****

## **Most Common Words in Taglines**



## **Most Common Words in Titles**

# **Modeling:**

# **1)** **Demographic Filtering**-

# Before getting started with this -

# we need a metric to score or rate a movie

# Calculate the score for every movie

# Sort the scores and recommend the best rated movie to the users.

# We can use the average ratings of the movie as the score but using this won't be fair enough since a movie with 8.9 average rating and only 3 votes cannot be considered better than the movie with 7.8 as as average rating but 40 votes. So, I'll be using IMDB's weighted rating (wr) which is given as :-

# IMG_256

# where,

# v is the number of votes for the movie;

# m is the minimum votes required to be listed in the chart;

# R is the average rating of the movie; And

# C is the mean vote across the whole report

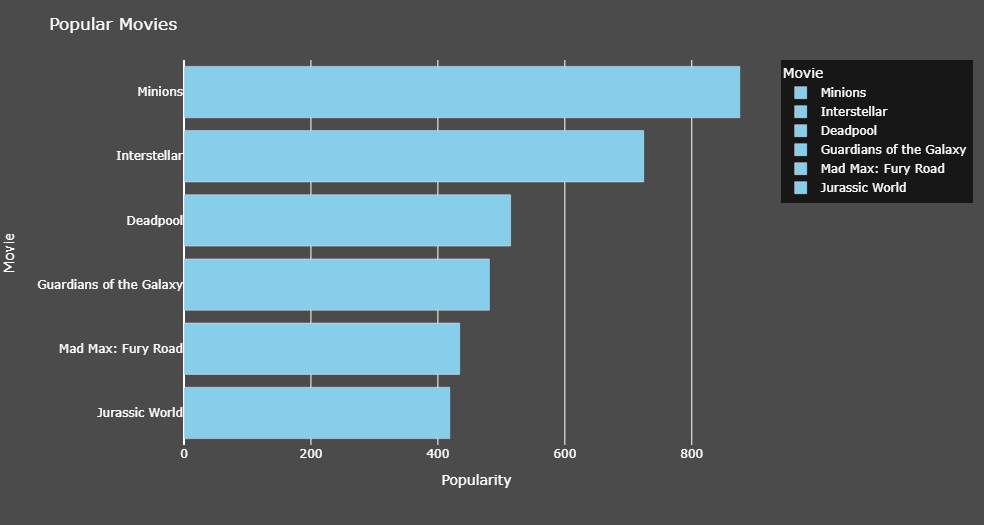
# We already have v(vote\_count) and R (vote\_average) and C can be calculated as

# C= movies['vote\_average'].mean()

C = 6.092514036182159

1. **Average Rating Calculation**: First, we calculate the average rating for all movies, which turns out to be around 6 out of 10.
2. **Determining the Minimum Votes Required**: Next, we need to establish a threshold value, represented by "m," which indicates the minimum number of votes a movie must receive to be included in the charts. To set this threshold, we use the 90th percentile. This means that for a movie to appear in the charts, it must receive more votes than at least 90% of the movies in the list.
3. **Qualifying Movies**: With the threshold set, we find that there are 481 movies that qualify to be in this list.
4. **Calculating the Weighted Rating**: Now, we define a function called weighted\_rating(), which we'll use to calculate a new feature called "score" for each qualified movie. This score reflects the movie's overall rating and vote count.
5. **Sorting the DataFrame:** Finally, we sort the DataFrame based on the score feature and extract the top 10 movies' titles, vote counts, vote averages, and weighted ratings or scores.

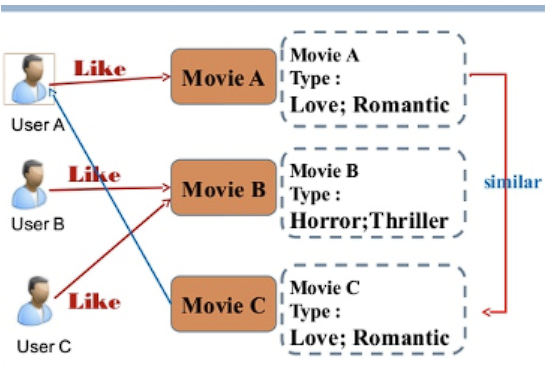
By following this process, we've created our first (albeit basic) recommender system.In the Trending Now section of these systems, we feature highly popular movies. These are easily identifiable by sorting the dataset based on the popularity column.

****

It's important to note that demographic recommenders offer a broad list of recommended movies to all users. They do not consider the specific interests and preferences of individual users.

This is where we transition to a more sophisticated system - Content-Based Filtering.

# **Content Based Filtering-**

In this type of recommender system, we look at the content of each movie, such as its plot summary, actors, directors, keywords, and taglines. Then, we compare this information to find similarities between different movies. Based on these similarities, we recommend movies that are most likely to be similar to ones the user has liked before. Essentially, we suggest movies that share common characteristics with those the user has enjoyed in the past.

## ****Plot description based Recommender-**** We have computed pairwise similarity scores for all movies based on their plot descriptions and recommended movies based on that similarity score. The plot description is given in the **overview** feature of our dataset. Let's take a look at the data.

**movies['overview'].head(5)**

**0 In the 22nd century, a paraplegic Marine is di...**

**1 Captain Barbossa, long believed to be dead, ha...**

**2 A cryptic message from Bond’s past sends him o...**

**3 Following the death of District Attorney Harve...**

**4 John Carter is a war-weary, former military ca...**

**Name: overview, dtype: object**

We've already converted the word vector of each movie overview. We used a method called Term Frequency-Inverse Document Frequency (TF-IDF). This method calculates how often words appear in each movie's overview and how common they are across all overviews.

TF-IDF helps us understand which words are important in describing a movie's plot without giving too much weight to common words that don't tell us much. We used a tool called TfIdfVectorizer in scikit-learn to do this quickly and easily.

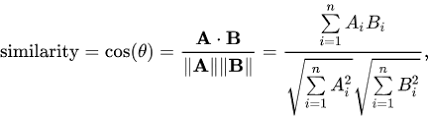
Now, we have a matrix where each row represents a movie, and each column represents a word from all the overviews combined. This process helps us find similarities between movies based on their plot descriptions.

We've noticed that our dataset contains 4,800 movies, with descriptions using more than 20,000 different words.

Now, with this matrix ready, we move on to calculate a similarity score. There are several metrics for this, like Euclidean, Pearson, and cosine similarity scores. Each metric has its strengths, so it's good to try different ones.

In our approach, we'll use cosine similarity to measure how similar two movies are. We chose cosine similarity because it's easy to calculate and doesn't depend on the magnitude of the vectors. Mathematically, it look

like this:



Since we've used the TF-IDF vectorizer, we can find the cosine similarity score directly by calculating the dot product. That's why we'll use sklearn's linear\_kernel() instead of cosine\_similarities() because it's faster.

Later we've implemented a function that takes a movie title as input and provides a list of the 10 most similar movies as output. However, before we could do this, we needed a way to map movie titles to their corresponding indices in our dataset.

In other words, we needed a mechanism that allowed us to identify the index of a movie in our metadata DataFrame based on its title. This mapping enables us to efficiently retrieve information about a movie from our dataset when needed.

**We are now ready to define our recommendation function. The following steps outline our approach:**

1. Retrieve the index of the movie based on its title.
2. Obtain the list of cosine similarity scores for that particular movie with all other movies. Convert it into a list of tuples, where the first element represents its position and the second element denotes the similarity score.
3. Sort the list of tuples based on the similarity scores, i.e., the second element.
4. Extract the top 10 elements from this sorted list, ignoring the first element as it refers to the movie itself (the most similar movie to a particular movie is the movie itself).
5. Return the titles corresponding to the indices of the top elements.

**While our system has been able to identify movies with similar plot descriptions to some degree, the quality of recommendations isn't quite up to par. For example, a search for "The Dark Knight Rises" might yield all Batman movies, but it's more likely that people who enjoyed that film would also appreciate other works by Christopher Nolan. Unfortunately, our current system isn't able to capture this nuance.**

**B)Credits, Genres and Keywords Based Recommender :**In order to improve the quality of our recommender system, we recognize the importance of using more advanced metadata. This is the focus of our next step. We plan to create a recommender system based on specific metadata: the top 3 actors, the director, related genres, and plot keywords for each movie.

To accomplish this, we'll need to extract the three main actors, the director, and the keywords associated with each movie from the 'cast,' 'crew,' and 'keywords' features. Currently, our data is stored as "stringified" lists, which need to be converted into a structured and usable format.

By incorporating more metadata into our recommender system, we've seen promising results in providing better recommendations. For example, fans of Marvel or DC comics are likely to enjoy movies from the same production house. Therefore, we've decided to enhance our features by including the production company information. Additionally, we've increased the importance of the director by adding this feature multiple times to our data analysis process.

# **Collaborative Filtering-**

Our movie recommendation system has some significant limitations. It can only suggest movies similar to a specific one, which means it can't cater to different tastes or recommend across different genres.

Furthermore, our system doesn't take personal preferences into account. It provides the same recommendations to everyone, regardless of their individual preferences.

To overcome these challenges, we're turning to a method called Collaborative Filtering. There are two main types:

1. **User-based filtering -** These systems recommend movies based on what similar users have liked. We determine similarity between users using methods like Pearson correlation or cosine similarity. This is like using a table where rows represent users, columns represent movies (except the last column, which shows how similar they are to our target user), and each cell represents a user's rating for a movie.
2. **Item-Based Collaborative Filtering** - Instead of comparing users' preferences, this method suggests items based on how similar they are to items the target user has rated. Similarity can be calculated using methods like Pearson Correlation or Cosine Similarity. The main difference is that item-based CF fills in the missing values vertically, unlike user-based CF which fills them horizontally.

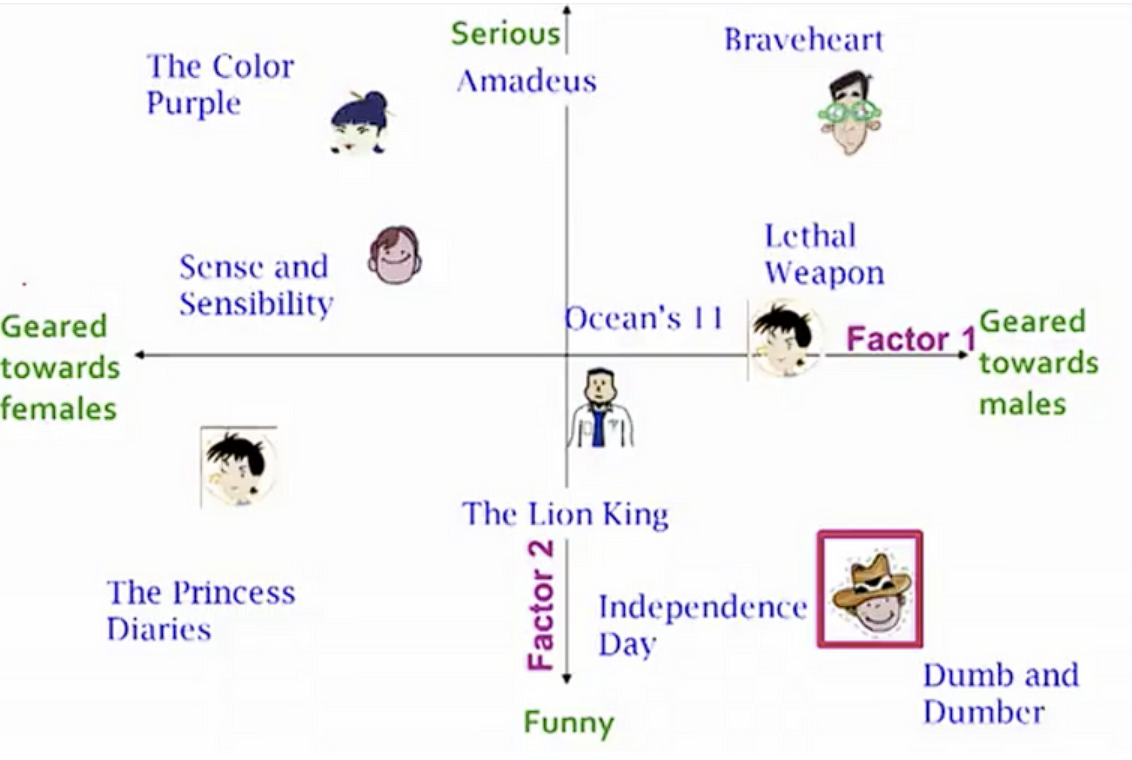
This method effectively addresses the issue of changing user preferences since item-based CF tends to be more stable. However, it still encounters challenges, particularly in terms of **scalability**. The computational workload increases as both the number of users and items grow. In the worst-case scenario, the complexity can reach O(mn) with m users and n items.

Another challenge is **sparsity.** For instance, even if only one user rated both "Matrix" and "Titanic," their similarity is calculated as 1. In extreme cases with millions of users, two vastly different movies could be deemed highly similar simply because they received similar ratings from the sole user who rated them both.

# **Single Value Decomposition (SVD) -**

To tackle the challenges of scalability and sparsity in collaborative filtering (CF), one method is to use Single Value Decomposition (SVD). This approach utilizes **a latent factor model** to capture similarities between users and items. **Essentially, it transforms the recommendation task into an optimization problem**, aiming to predict item ratings for users accurately. A common metric used to assess the performance of such models is Root Mean Square Error (RMSE), where lower values indicate better performance.

Now, let's discuss what latent factors are. They represent underlying properties or concepts associated with users or items. For instance, in music, a latent factor could represent the genre of the music. **SVD reduces the dimensionality of the utility matrix by extracting these latent factors.** This process essentially maps each user and item to a latent space with a dimensionality of r. As a result, it enables a more direct comparison between users and items, enhancing our understanding of their relationships.



We have used **'ratings\_small.csv'** dataset for SVD model.

We achieved a mean **Root Mean Square Error of approximately 0.89**, which is satisfactory for our purposes. Now, let's proceed to train on our dataset and generate predictions.

For the movie labeled as ID 302, the system guesses that people might rate it around **2.6** What's interesting is that this system doesn't really care about what the movie is about. It just looks at its ID number and guesses how good people might think it is based on how others have rated it.

### To compare with SVD we have also used SVDPP model, and For the movie we predicted earlier using SVD and got a result of **2.6**, the SVDpp model gave a prediction of **2.73,** which is very close.

**From these results, we can see that SVDpp generally performs better in terms of RMSE and MAE, with lower mean values for both metrics compared to SVD. However, SVDpp requires significantly more time for fitting and testing compared to SVD.**

**Therefore, if computational resources are not a constraint and better accuracy is desired, SVDpp would be preferred. However, if faster computation is more important and slightly lower accuracy is acceptable, SVD could be a better choice.**

**Outcomes:**

Following extensive exploration and feature engineering, we successfully identified correlated features within the movie dataset. Notably, our recommender system exhibited remarkable performance, particularly with the implementation of collaborative filtering methods. Through rigorous testing, we found that the collaborative filtering approach outperformed other techniques, highlighting its effectiveness in providing accurate and personalized movie recommendations. These outcomes signify the capability of our system to deliver high-quality recommendations tailored to individual user preferences, thus enhancing the overall user experience on the platform.

**Implementation**

To implement our movie recommender system, we'll focus on:

1. Setting up data acquisition, storage, and program execution systems.
2. Collaborating with stakeholders to determine the forecasting period.
3. Developing a user-friendly interface for data visualization. By addressing these aspect, we aim to create a seamless and user-centric movie recommendation experience.

**Data answer**

The movie recommender system employs three approaches: Simple Recommender generates top charts based on TMDB votes; Content-Based Recommender analyzes overviews, taglines, and metadata like cast and crew; Collaborative Filtering, using Surprise Library, achieves low RMSE with SVD and SVDpp, offering estimated ratings.

**Business Answer:**

Our movie recommender system employs three strategies: Simple Recommender uses TMDB votes, Content-Based analyzes overviews and metadata, while Collaborative Filtering achieves low RMSE with SVD and SVDpp. This strategic initiative aims to enhance user engagement, satisfaction, and retention while optimizing operational efficiency.

**Response to Stakeholders:**

The movie recommender model demonstrates significant promise and is anticipated to meet its objectives in enhancing user engagement and satisfaction. Based on the outcomes and implementation plan, it is recommended to proceed with the deployment of the movie recommender system.

**End-to-End Solution:**

To implement the end-to-end solution for the movie recommender system, the organization should establish infrastructure for data acquisition and storage, along with systems capable of running the recommendation model efficiently. Additionally, interfaces should be developed to display personalized movie recommendations to users, ensuring seamless integration and optimal user experience.

**References:**

1. **[Movie Recommendation System]**(https://www.kaggle.com/code/ashfakyeafi/movie- recommendation-system)
2. **[Getting Started with a Movie Recommendation System]** (https://www.kaggle.com/code/ibtesama/getting-started-with-a-movie-recommendation-system)
3. **[Introduction to Recommender System Part 1: Collaborative Filtering & Singular Value Decomposition]**

(https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75)

4. **[Movie Recommender Systems]**(https://www.kaggle.com/rounakbanik/movie-recommender-systems)