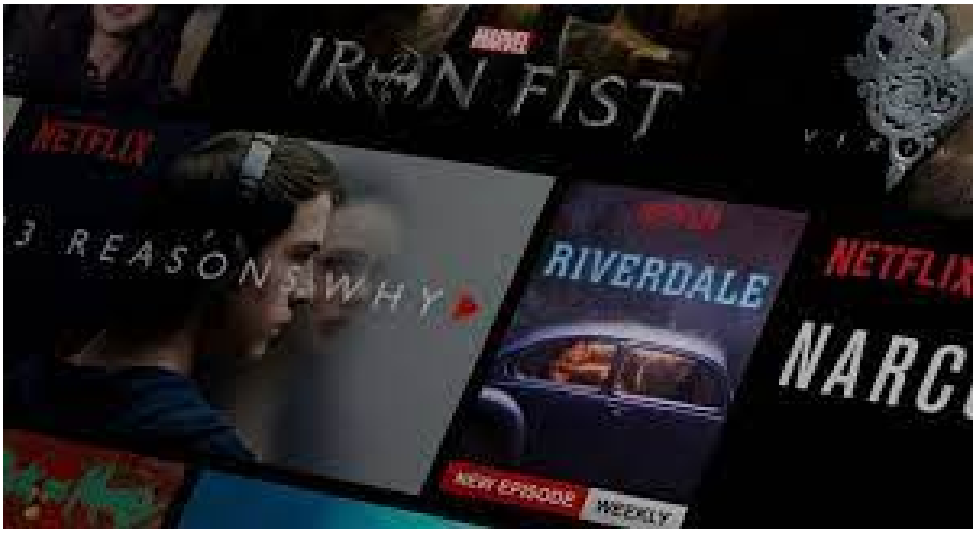
PROJECT

ON

Movie Recommendation System



SUBMITTED TO: Dr. Bharti Rana

SUBMITTED BY: Pooja Sahu(33)

CLASS : MCA Sem-5 (2021-2024)

Department of Computer Science

University Of Delhi

Background and Problem Definition

In today's interconnected world, where the internet plays a crucial role in people's lives, users often grapple with the overwhelming abundance of choices. Whether searching for accommodation or seeking investment opportunities, the sheer volume of information can be daunting. To assist users in navigating this information overload, companies have implemented recommendation systems to provide tailored guidance. Research in this field has spanned decades, fueled by the myriad practical applications and challenges within the domain.Various online platforms, such as Amazon.com for books, MovieLens.org for movies, and CDNow.com (an Amazon subsidiary) for CDs, have successfully integrated recommendation systems. These systems contribute significantly to the economic success of e-commerce giants like Amazon.com and streaming services like Netflix. The profitability of these websites underscores the pivotal role recommendation systems play in enhancing user experiences.

Recommender systems generate suggestions, which users can accept based on their preferences. User actions and feedback, whether implicit or explicit, are stored in recommender databases. This data becomes a valuable resource for generating future recommendations, creating a dynamic feedback loop between users and systems.

The economic potential of recommender systems has prompted major e-commerce websites, such as Amazon.com and snapdeal.com, as well as online movie rental platforms like Netflix, to integrate these systems prominently into their interfaces. The provision of high-quality, personalised recommendations not only enhances user satisfaction but also adds a new dimension to the overall user experience. Recently, web personalised recommendation systems have expanded to provide diverse types of customised information to cater to the unique needs of individual users

In general the recommendation system has two basic flavours such as Collaborative filtering and Content based recommendation system. The most common technique used for recommender systems is Collaborative filtering. The Collaborative Filtering algorithm makes predictions based on neighbours or relevant users. This system recommends a useful list of movies which are fil-tered based on the similarity of other users rating. The crucial step of collaborative filtering is finding the similar preference of the user with respect to other users. But it suffers from limitations such as **sparsity, cold start** and **scalability**.

Content-based recommendation method uses information from user’s profile or item’s profile for providing recommendation. The profile of target user will be analysed and items which are matching to user’s profile will be selected for suggestions.The combination of these two systems or other combinational systems leads to the hybrid recommender.

Already existing movie recommendation systems make use of content based filtering to provide recommendations which has been working quite well so far. But the changing era and expectation of instant and accurate results are putting an urge to improve the efficiency of the existing one and a hybrid approach is the only way possible to do so. The hybrid approach has the ability to incorporate both content-based and collaborative filtering techniques improving the efficiency and accuracy of the system by providing optimal results and thus its need.

Issues/problems in the existing solution(s)

Nowadays, the problems of recommendation systems mainly include **data sparsity**, **cold start**, **big data processing**, incremental computing, user behaviour mining, and user image . This paper mainly focuses on the optimization of data sparsity and cold start. Sparsity is a term used to describe the percentage of unfilled and filled cells in a database table.The sum of sparsity and density should equal 100%. A table with a density of 10% has 10% cells filled with non-zero values. For example, 90 percent sparse means 90 percent of its cells are either not filled with data or zero. Generally speaking, the larger the index data scale is, the lighter the user-item matrix is, and the fewer data that can be calculated by the recommendation system, the lower the recommendation accuracy rate will be. For example, on a movie website, if the users only watch a small number of movies, this interactive data will not be enough for the recommendation system to make accurate recommendations to users. The traditional approach uses the nearest-neighbour model fusion algorithm to solve the problem. Still, the time complexity is high in searching for the nearest neighbour to calculate the similarity between users (items) directly. To solve this problem, this research applied a mixed-weighted prediction filling algorithm. Based on the characteristics of the resources accessed by users and the popularity of the resources accessed in the whole user group, the data visited by users but not evaluated are predicted and filled. This way can reduce the sparsity of the similarity matrix caused by the lack of user evaluation data and improve the recommendation accuracy. Experimental results on the MovieLens dataset showed that the proposed algorithm could significantly improve the recommendation accuracy. A cold start is when a user interacts with a few items. The recommendation system will not be available to recommend the personalised information immediately because the user similarity table is calculated offline at regular intervals. Cold start is the problem of designing a personalised recommendation system without a large amount of users' interactive data and making users satisfied with the recommendation results. For the cold start problem, the traditional approach recommends popular items to the users. Such a recommendation does not have personalised significance, and sometimes it will even backfire and cause users' aversion to the system. Starting from the actual application scenario of the recommendation system based on the movie website, this project mainly uses **Content Based Algorithm** and **Singular Value Decomposition (SVD)** combined with the user rating data of MovieLens. Based on the data sparsity and cold start problems of the collaborative filtering algorithm and the limitations of Item-CF and User-CF algorithms, an improved algorithm was used, and the SVD algorithm was applied to the MovieLens ML-100K dataset.

Objective is to enhance user experience with personalised suggestions for movies based on their preferences and behaviour.

Related work

A content-based recommendation system, which mainly relies on cognitive filtering, is "a system that displays item recommendations by comparing alternative items with those associated with user profiles". In the paper **Movie Recommendation Systems Using Actor-Based Matrix Computations in South Korea** [paper1] describes the data collection and preprocessing steps for the study on a content-based movie recommendation system that considers genres and actors. The data, obtained from the Korean Film Council, includes movie details, film companies, actors, and directors. Duplicate movie titles (3429) were identified and removed, focusing on feature films. Additional data from the Naver movie site and the Korea Box Office Information System were incorporated, encompassing various movie-related information. The dataset was further refined by excluding certain genres and using only movies rated by more than 30 people. Data plots were extracted from the Naver series for the content-based recommendation system. Preprocessing involved removing unnecessary characters and employing Korean natural language processing. The final dataset combined columns for actors, directors, genres, and plots, supplemented with viewers' ratings, sales, attendance, and sales rate. The study also detailed the computation of rank correlation between movies and genres, as well as the correlation between actors and genres using Pearson's correlation coefficient. The results indicated marked differences in correlation between movies and genres compared to actors and movie genres. A user survey was conducted to evaluate the recommendation system, with participants expressing preferences based on movie genres or actors. The study concludes by highlighting academic and practical implications, emphasising the potential for developing content-based recommendation systems and contributing to hybrid recommendation systems in the South Korean film industry.

The actor-based recommendation system proposed in the study presents notable advantages, primarily focusing on personalization by considering individual user preferences related to actors, thereby potentially enhancing overall user satisfaction. The study also contributes by addressing limitations observed in existing recommendation systems, particularly those faced by platforms with insufficient user data. However, certain disadvantages exist within the study's framework. Firstly, there is a limited generalisation of results due to the focus on South Korean movies from a specific timeframe, potentially restricting the applicability of findings to a broader context. Additionally, while the emphasis on actors is valuable, the study acknowledges the importance of other features in movies, suggesting that future research may need to incorporate additional elements for a more comprehensive analysis. Lastly, the study lacks a comprehensive comparison with other movie recommendation techniques, leaving room for further exploration and evaluation of different approaches in the field.

Collaborative filtering (CF) is one of the mainstream recommendation system algorithms, among which ItemCF and User-CF are widely used. User-CF refers to analysing the user's behaviour towards goods (e.g., browsing, bookmarking, adding to the shopping cart, purchasing...) Figure out which users have similar interests and then recommend products that similar users care about to each other.In second paper **Movie Recommendation System Based on SVD Collaborative Filtering** [paper2] discusses the challenges of information overload on

the Internet and the need for personalised recommendation systems. It emphasises the role of recommendation systems in addressing the issue by guiding users to relevant information based on their interests. The focus is on movie recommendation systems, and the data used for analysis is obtained from the MovieLens dataset. The document explores various recommendation algorithms, including Item-based Collaborative Filtering (Item-CF), User-based Collaborative Filtering (User-CF), and Singular Value Decomposition (SVD). The analysis compares their performance using evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The results show that SVD outperforms Item-CF and User-CF, especially in scenarios with sparse data.

The recommendation system offers several advantages, including enhanced personalization through algorithms like Singular Value Decomposition (SVD), which effectively predicts user preferences and provides tailored suggestions. The system demonstrates efficiency in handling vast amounts of data, thereby reducing the time users spend manually searching for movies. SVD, in particular, stands out for its superior accuracy compared to other algorithms, contributing to more reliable movie recommendations. Additionally, the system exhibits flexibility, as discussed in the document, allowing for the adjustment of parameters such as the proportion of the training set and the value of k in SVD. However, certain disadvantages accompany these benefits. The complexity of SVD introduces challenges in interpreting hidden vectors, making the recommendation process less transparent. Data sparsity poses a hurdle for collaborative filtering algorithms like Item-CF and User-CF, leading to reduced recommendation accuracy in scenarios with sparse data. The cold start problem is highlighted, indicating the system's struggle when users have limited interaction data, impacting immediate personalization. Furthermore, each recommendation algorithm has its specific limitations; for instance, Item-CF is less effective with many users, while User-CF faces difficulties in maintaining user similarity matrices.

Method

Data Collection and Preprocessing

We took the **The Movies Dataset** [dataset] from KAGGLE.com which contains metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

This dataset consists of the following files:

movies_metadata.csv: The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.

keywords.csv: Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.

credits.csv: Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.

links.csv: The file that contains the TMDB and IMDB IDs of all the movies featured in the Full MovieLens dataset.

links_small.csv: Contains the TMDB and IMDB IDs of a small subset of 9,000 movies of the Full Dataset.

ratings_small.csv: The subset of 100,000 ratings from 700 users on 9,000 movies.

The Full MovieLens Dataset consisting of 26 million ratings and 750,000 tag applications from 270,000 users on all the 45,000 movies in this dataset can be accessed here

To build our standard metadata based content recommender, we will need to **merge our current dataset with the crew and the keyword datasets**

Pre-processing :

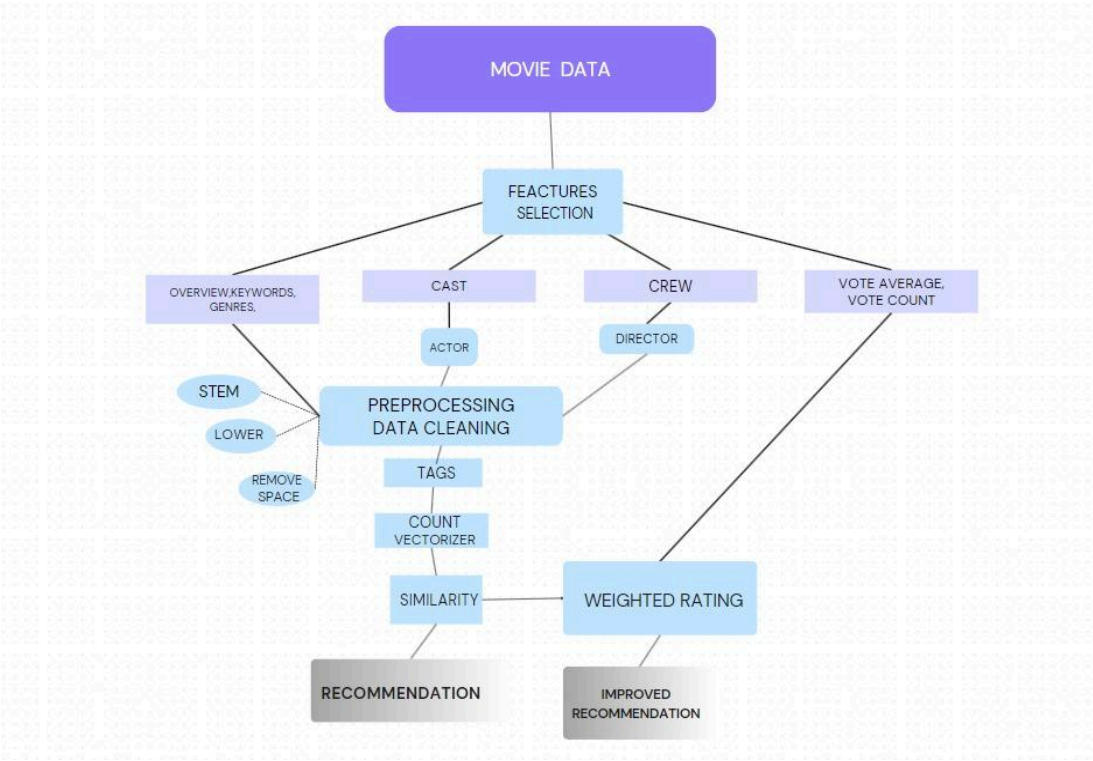
From our credit dataframe we took cast, crew ,from keywords dataframe keywords, from links_small imdbId ,tmdbId from ratings_small rating which has rating according to a user.

After wrangling of data ,feature selection takes place which includes overview, keywords, genres, cast, crew, voting average, votecount. From the cast we fetch top 3 actors , from the crew fetch director.

Now, merging all the data except voting average, votecount in one dataframe ,then performing operations like stemming (shortens words to their base form by removing suffixes) then removing whitespaces and converting them into lowercase so that there is no duplicacy.

Creating a tag for each movie, use a Countvectorizer to create our count matrix, calculate the **cosine similarities** and return movies that are most similar and then recommend movies.

This is a **content based movie recommendation system**. Now we have **improvised the movie recommendation system** using vote count and vote average .



CF based recommendation system

This is also called collaborating filtering. A Collaborative Filtering (CF) based recommendation system recommends items by leveraging user-item interactions. It operates in two types:

- User-Based CF: Recommends items based on the preferences of users with similar tastes.
- Item-Based CF: Recommends items similar to those the user has liked or interacted with.

Example: If User A likes movies similar to User B, the system recommends movies liked by User B but not yet seen by User A. CF systems are widely used in personalised content recommendations.

SVD (Singular Value Decomposition) is a popular matrix factorization technique commonly used in collaborative filtering-based movie recommendation systems. In a movie recommendation system, SVD is applied to decompose the user-item interaction matrix into three matrices:

- User Matrix (U): Represents users and their latent factors.
- Item Matrix (V): Represents items (movies) and their latent factors.
- Singular Value Matrix (Σ): Represents the singular values.

The user-item interaction matrix M is decomposed as M is nearly equal to $U.\Sigma.V^T$

The SVD algorithm is shown below in Figure

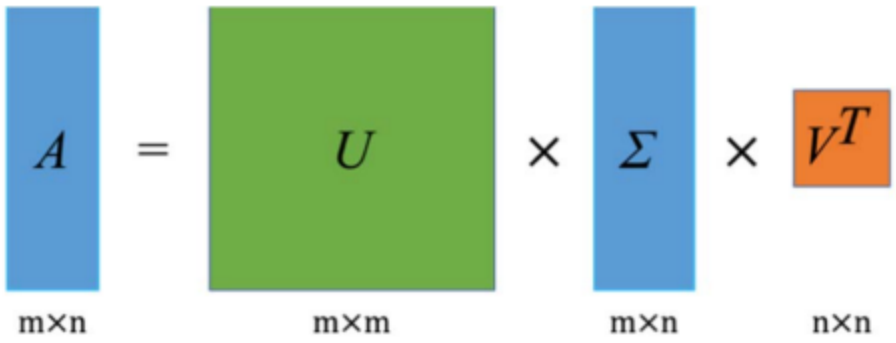


Fig SVD Algorithm

Here's how SVD is typically applied in a movie recommendation system:

Create the User-Item Matrix:

Build a matrix where rows represent users, columns represent items (movies), and the entries are the ratings given by users to movies.

Fill Missing Values:

Fill missing values (unrated movies) with zeros or another suitable value.

Apply SVD:

Apply SVD to decompose the user-item matrix into three matrices.

Select Latent Factors:

Select a certain number of latent factors (dimensions) to represent users and items.

Predict Ratings:

Predict the missing ratings for unrated movies using the decomposed matrices.

Recommendation:

Recommend movies with the highest predicted ratings to users.

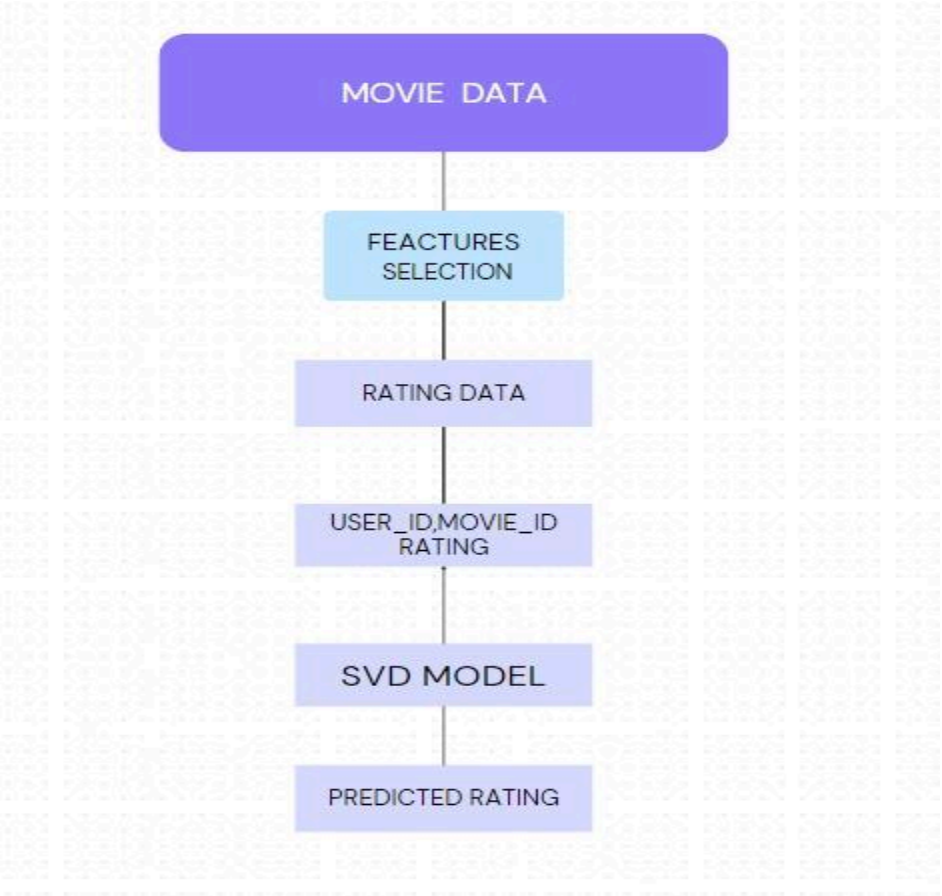
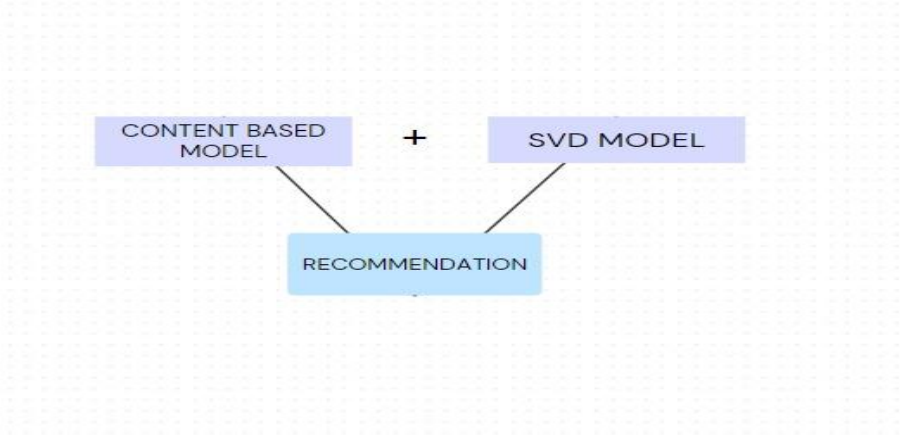


Fig. Recommendation flow of collaborative model

Hybrid Movie Recommendation System

A hybrid movie recommendation system combines multiple recommendation approaches to leverage their strengths and provide more accurate and diverse recommendations. Typically, hybrid systems combine collaborative filtering and content-based filtering techniques.



Computings involved:

RMSE : Root-mean-square-error

RMSE is a quadratic scoring rule that also measures the average magnitude of the error.

It's the square root of the average of squared differences between prediction and actual observation.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

MAE : Mean Absolute Error

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction.

It is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$MAE = \frac{1}{n}\sum_{j=1}^{n} | y_j - \hat{y}_j |$$

Mathematically, it is represented as follows:

WeightedRating(*WR*) = (v/(v+m) * R) + (m/(m+v) * C)

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

C is the mean vote across the whole report

And recommend according to content based + WeightedRating.

Experimental Results and Discussion

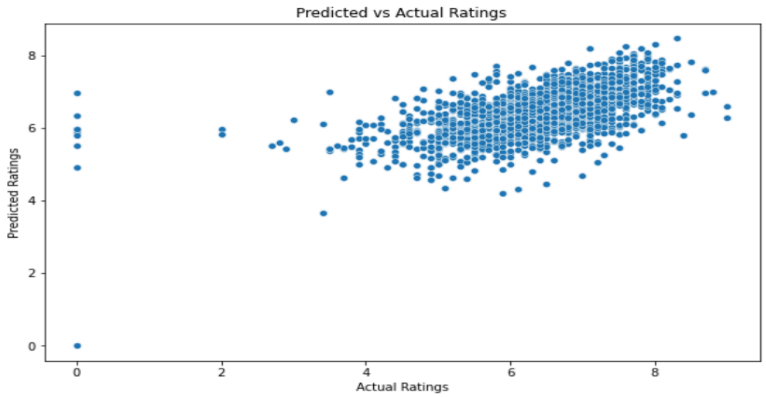


Fig A: Relation between predicted and actual ratings of content based model

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to extract features from movie descriptions, genres, keywords, crew, and cast. It then applies a linear regression model to predict movie ratings, evaluating performance with the Root Mean Squared Error (RMSE) on the test set of the content based model RMSE: 0.8458514722271445

.

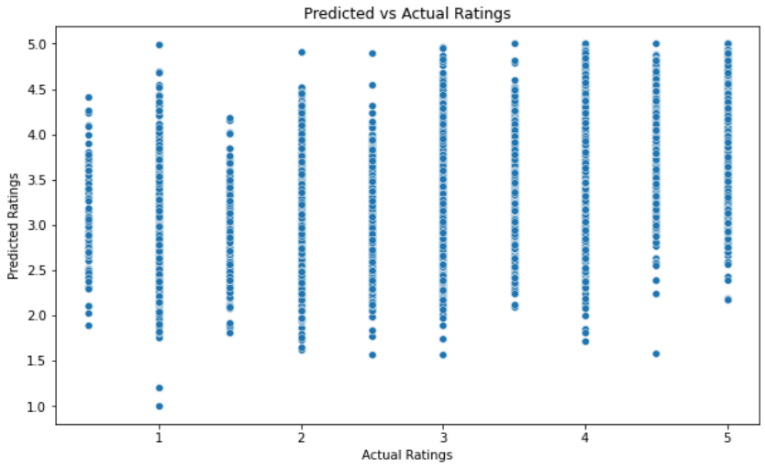


Fig B: Relation between predicted and actual ratings of collaborative based model

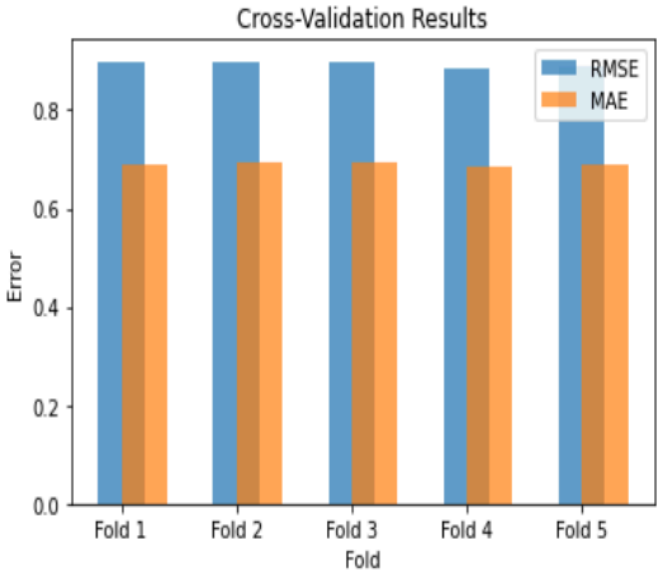


Fig C: RMSE and MAE values for each fold in a cross-validation experiment, comparing different models of SVD

In fig c We get a mean Root Mean Square Error of 0.89 approx which is more than good enough for our case. Let us now train on our dataset and arrive at predictions.

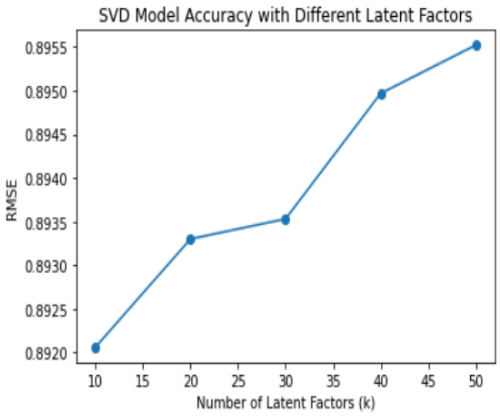


Fig D: RMSE of SVD models trained with varying numbers of latent factors (k)

Fig D demonstrates collaborative filtering using Singular Value Decomposition (SVD) with varying numbers of latent factors (k). It trains the SVD model for different k values and plots the Root Mean Squared Error (RMSE) for each k, helping to identify the optimal number of latent factors for the collaborative filtering recommendation system. The displayed RMSE values indicate the model's accuracy for each k.

Conclusions and Future Work

We see that for our hybrid recommender, we get different recommendations for different users although the movie is the same. Hence, our recommendations are more personalised and tailored towards particular users.

In this Project, we have built 4 different recommendation engines based on different ideas and algorithms. They are as follows:

Content Based Recommender: We built two content based engines; one that took movie overview and taglines as input and the other which took metadata such as cast, crew, genre and keywords to come up with predictions. We also devised a simple filter to give greater preference to movies with more votes and higher ratings.

Improved Content Based Recommender: This system used overall TMDB Vote Count and Vote Averages to build Top Movies Charts, in general and for a specific genre. The IMDB Weighted Rating System was used to calculate ratings on which the sorting was finally performed.

Collaborative Filtering: We used the powerful Surprise Library to build a collaborative filter based on single value decomposition. The RMSE obtained was less than 1 and the engine gave estimated ratings for a given user and movie.

Hybrid Engine: We brought together ideas from content and collaborative filtering to build an engine that gave movie suggestions to a particular user based on the estimated ratings that it had internally calculated for that user.

Some potential areas for future work and improvement:

- Feature Engineering and Selection:
    - Experiment with additional features for content-based recommendation, exploring different textual representations and feature combinations.
    - Consider sentiment analysis for textual features to capture user preferences more effectively.
- Content-Based Recommender:
    - Enhance content-based recommenders by incorporating more advanced natural language processing techniques for better understanding of movie overviews and taglines.
    - Experiment with deep learning models to capture complex relationships between content features.
- Improved Content-Based Recommender:
    - Further refine the TMDB Vote Count and Vote Averages system by considering temporal trends or user-specific preferences.
    - Explore alternative weighted rating systems to assess their impact on recommendation accuracy.
- Collaborative Filtering:
    - Explore advanced collaborative filtering techniques, such as matrix factorization variants or deep learning-based models, to potentially improve accuracy.
    - Investigate methods to handle cold-start problems for new users or items in collaborative filtering.
- Hybrid Engine:
    - Experiment with different combinations of content-based and collaborative filtering components in the hybrid engine to optimise the recommendation strategy.
    - Integrate real-time user feedback to dynamically adjust the hybrid engine's recommendations.
- Evaluation Metrics:
    - Explore additional evaluation metrics beyond RMSE to provide a more comprehensive assessment of recommendation system performance.
    - Consider incorporating user-centric metrics, such as precision, recall, or F1 score, to evaluate the practical usefulness of recommendations.
- Scalability and Efficiency:
    - Evaluate and enhance the scalability and efficiency of the recommendation engines, especially as the dataset grows.
    - Explore distributed computing or parallel processing for handling larger datasets.

References

- https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
- https://grouplens.org/datasets/movielens/latest/
- https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75
- https://press.aboutamazon.com/2002/12/cdnow-and-amazon-com-announce-relaunch-of-cdnow-com-site
- https://www.sciencedirect.com/science/article/pii/S0024379599001342/pdf?md5=1c88fbcde70f1ae747d85397d6284e01&pid=1-s2.0-S0024379599001342-main.pdf
- https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c
- https://thepoints.medium.com/feature-extraction-from-text-using-countvectorizer-tfidfvectorizer-9f74f38f86cc
- https://mitpress.mit.edu/9780262012119/introduction-to-machine-learning/