CSC869 Term Project Report- **Movie Recommendation System**

Problem

Over-The-Top app faces a challenge of providing the best content to its users in a way each user gets the most based on its interest, also it has to help its content providers to reach the right users. Hence, a system is required that can be helpful for both the users and the content providers. This machine learning project aims to create a movie recommendation system that gives viewers personalized movie recommendations based on their interests and watch history. To provide accurate and relevant movie suggestions, the recommendation engine will make use of machine learning techniques to examine user behavior and movie characteristics. The goal is to create a powerful movie recommendation system. which will suggest movies to users based on collaborative filtering approaches. Furthermore, once it has sufficient information about the user's choice, it will provide related movies based on similarity index. It uses a collaborative filtering system for movies that enables it to filter content based on the reactions of users who are similar to the person.

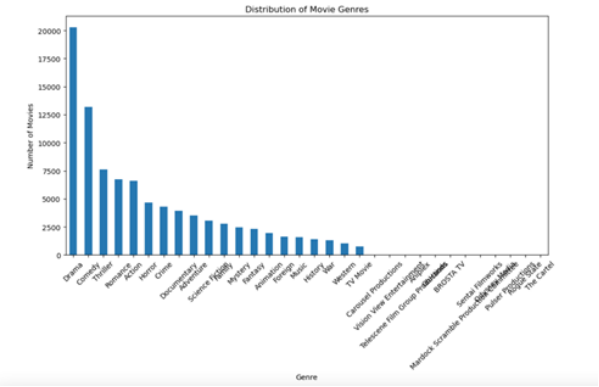
Data Set

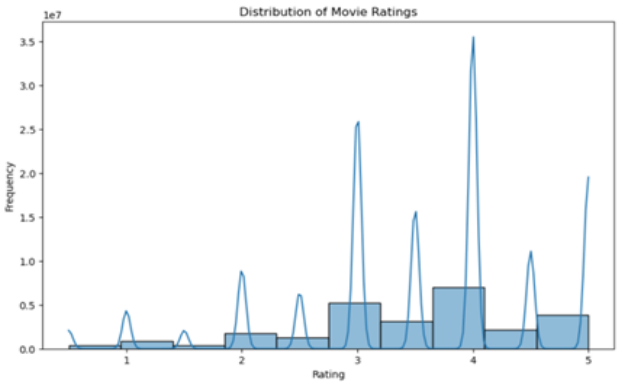
The Movies Dataset is used for this project which is available on Kaggle [1]. This dataset contains metadata about 45,000 movies which were released on or before July 2017. Additionally, files with 26 million user ratings for all 45,000 films from 270,000 users are included in this dataset. Ratings are taken from the official GroupLens website and are on a scale of 1 to 5. The dataset includes movies_metadata, credits, keywords, links, and ratings.

These datasets contains the following files in csv format:

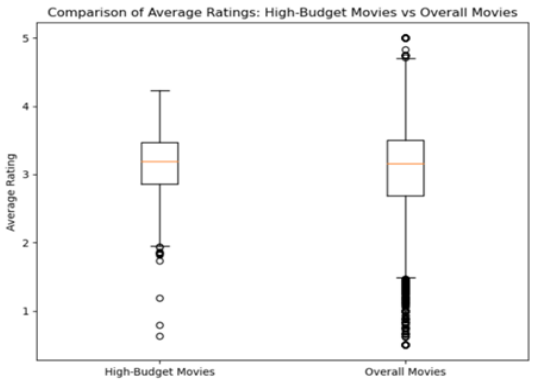
- movies_metadata.csv: Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
- keywords.csv: Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.
- credits.csv: Consists of Cast and 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.
- ratings.csv: 26 million User ratings for movies. Crucial for collaborative filtering.

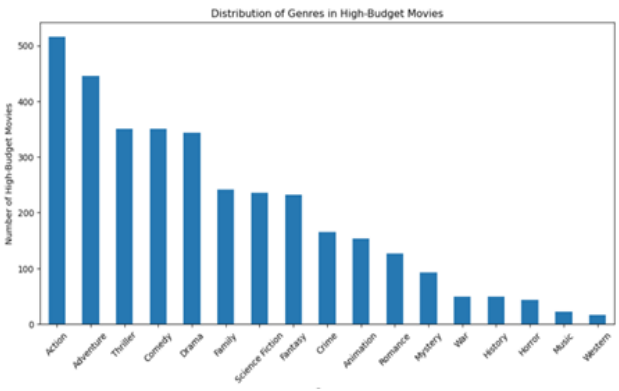
Important Data Visualizations:





The right-skewed bar graph indicates that majority of the movies in the dataset belong to genres such as drama, comedy, thriller, romance, action, and horror. The higher concentration of movies in these top genres suggests a strong prevalence of these categories. Also, the rating graph shows most of the movies are of rating ranging from 3 to 5.





Movies with high budgets predominantly fall into genres such as action, adventure, thriller, comedy, and drama. Box plot comparing average ratings for high-budget and overall movies, with both medians near 3. High-budget movies have a narrower rating distribution, while overall movies display a wider range with more outliers.

Note: Links dataset doesn't have up-to-date data. TMDB ID's could have been updated on their websites.

Methods and Strategy

In this section the data preparation and model fitting approach is explained. For data preparation we have performed the data preprocessing and feature extraction. These steps are explained below:

Data Preprocessing and Integration

ID Standardization and Conversion: Transforming all movie and user IDs across different datasets into a consistent format for seamless integration and processing. It is helpful step as these datasets will be merged using IDs, hence consistent format is indeed required.

Dataset Merging: Combining data from movies_metadata, credits, keywords, links, and ratings_small to create a comprehensive dataset. This step is crucial for a holistic view of each movie, encompassing all aspects from cast and crew to user ratings. These are performed using the common keys that joins a pair of datasets together, we have performed the left join on these dataset to bring all required features into the single data frame.

Parsing and Cleaning: Converting JSON strings in columns like genres, keywords, cast, and crew into structured lists. This step is essential for the extraction of meaningful features from textual data. As Movie metadata dataset contains crucial meta-data information in the form of JSON, hence these are converted into a data frame for analysis.

Feature Extraction

Director and Cast Analysis: We have extracted the director's name and main actors from the movie's crew and cast. This helps in identifying movies with similar artistic direction and acting styles. As some users have a personal inclination towards a specific director as they may have specific preferences for directors and casts e.g. about their directing style, favorite actor/actress, consistent quality makers, genre expertise and other factors.

Genre and Keyword Processing: Utilizing genres and keywords to capture the thematic and stylistic elements of movies. These features are key to understanding the content and context of a movie. This information may be useful as the system can have tailoring recommendations and content suggestions that aligns with a user's specific genre preferences.

Textual Data Handling: Applying natural language processing techniques to preprocess movie overviews. This includes tokenization, stop word removal, lemmatization, and the creation of a normalized textual representation of each movie's synopsis. Tokenization divides textual content into discrete "tokens", which is essential for content-based filtering and similarity matrix computing. The process of tokenization makes it easier to create meaningful representations, like word embeddings or TF-IDF vectors, which are used to calculate similarity scores between items in content-based recommendation systems.

Aggregation of Features: Creating a combined feature string for each movie, integrating genres, keywords, main actors, director's name, and processed overviews. This comprehensive feature set serves as the basis for assessing movie similarities.

Content-Based Filtering Implementation

In an era when new content is published over the OTT platform so frequently. There is no historical interaction data about those, hence, it is a challenge when working with novel or specialized movies with which systems have little to no past interaction data. Recommendations based on movie qualities can still be made by content-based filtering, which does not primarily rely on user-item interactions.

Feature Aggregation for Similarity Assessment: Creating a combined_features attribute by concatenating genres, keywords, main actors, director's name, and the preprocessed overview. This aggregated feature set is used to represent each movie's unique content profile.

TF-IDF Vectorization: Utilizing TF-IDF Vectorizer to transform the combined features into a TF-IDF matrix, enabling a numerical representation of the textual data. TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is critical for Content-Based Filtering. TF-IDF allows the generation of feature vectors that indicate the significance of words or keywords in item descriptions in the context of content-based filtering. These vectors are used to compute similarity scores, which allow the system to offer items that match a user's tastes based on the prominence of shared phrases in the item descriptions.

Cosine Similarity Calculation: Implementing cosine similarity on the TF-IDF vectors to measure the content similarity between movies. This method identifies movies with similar themes, styles, and content. It is useful because it calculates the cosine of the angle between two feature vectors and produces a normalized measure of similarity that is independent of the vectors' magnitudes, cosine similarity is helpful in content-based filtering with TF-IDF vectorization. A reliable comparison of items is made possible by the cosine similarity, which accurately represents the similarity in terms of the relevance of common phrases in the context of TF-IDF vectors representing item descriptions. This metric helps identify movies that have similar feature characteristics, which helps in making suggestions based on movie textual aspects more accurate. It is especially useful for content-based filtering.

Comparatively speaking to collaborative filtering techniques, content-based filtering is less computationally demanding because it frequently involves calculating similarities based on item properties.

User-Based Collaborative Filtering

User-Based Collaborative Filtering identifies users with similar likes, it offers personalized movie recommendations, improving the whole movie-watching experience. This methodology successfully tackles the issue of sparse data in movie recommendation datasets, enabling it to withstand scenarios in which users have not extensively reviewed or engaged with the movie. Furthermore, User-Based Collaborative Filtering helps new users by recommending movies based on the interests of other users who share the same likes, thus solving the issue of cold start. This collaborative filtering approach makes for a more interesting and varied movie suggestion experience by promoting exploration and utilizing the social component of movie choices. For collaborative filtering, we used the SVD (Singular Value Decomposition) algorithm from the Surprise Library. The numeric rating data is used for this model. The model is trained by dividing the data into training and testing sets. Then SVD is applied on the training set and ratings are predicted for the test set. We have also implemented a function to get the top-N movie recommendations for a specific user.

Item-Based Collaborative Filtering

Item-Based Collaborative Filtering is a good fit for movie recommendation systems. It enables the detection of related movies according to user preferences, improving the accuracy and customisation of suggestions. When handling the sparsity of user-item interaction data, which is frequently present in movie recommendation datasets, this method works especially well. Additionally, Item-Based Collaborative Filtering makes it easier to find related or comparable movies, which increases user engagement by making suggestions for movies that suit their preferences. It is also compatible with recommendation algorithms that scale with an increasing number of movies, which makes it a useful option for movie databases that have a large collection. We have implemented this first by constructing a pivot table from the ratings data, mapping user IDs to

movie IDs. Then converted the pivot table to a sparse matrix for efficiency. Similarity Computation is computed using cosine similarity measures, that is used to build the Similarity Matrix. It is a similarity matrix representing movie-to-movie similarities.

This matrix can be used for recommendation, just by picking a movie ID for which we wish to find similar movies. We have also implemented a function that can identify top 10 movies most similar to the selected movie based on similarity scores.

Evaluation:

In the context of recommendation systems, evaluation measures like Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are frequently employed. These metrics measure the discrepancies between expected and actual values in order to estimate the accuracy of predicted evaluations. The average magnitude of these differences is measured by RMSE, which highlights bigger errors resulting from the squared terms. The average absolute differences, on the other hand, are determined by MAE, providing a simpler indicator of prediction accuracy. Lower values for both metrics indicate better accuracy, therefore they are useful for evaluating how well recommendation systems function. It is imperative to give careful thought to these measures in order to optimize recommendation system precision and fine-tune models. We have used both RMSE and MAE values to evaluate the model's performance.

Picked a few movies from your dataset that are well-known and generated content-based recommendations for each of these movies using your system and manually checked if the recommended movies are similar in genre, themes, directorial style, or cast in popular IMDb/ TMDb websites

Results:

The above mentioned three approaches were implemented and tested for different scenarios e.g. the content-based filtering suggested top 10 movies successfully if for example a new movie is just released, so other recommendations are made which have similar properties as the given movie. Similarly for User-based collaborative filtering the system has successfully made recommendations for a particular user, as this approach considers the past interest of the user and on the basis of it the implemented system produces recommendations. Finally, for item-based collaborative filtering it also gives the recommendations successfully. For User-based collaborative filtering approach we also evaluated the model using the evaluation approach mentioned in the previous section. The calculated Root Mean Squared and Mean Absolute errors are **0.874134** and **0.6786** respectively. The results for movie recommendations are summarized below:

Method	**Recommendation For**	**Top 10 recommendations**
Content Based Filtering	Movie: The Matrix	The Matrix Revolutions, The Matrix Reloaded, The Thirteenth Floor, 23, Tron, War Games, Underground: The Julian Assange Story, Red Planet, The Invisible Boy, Speed Racer
User-based collaborative Filtering	User ID 1	The Million Dollar Hotel, Sleepless in Seattle, The Thomas Crown Affair, Galaxy Quest, Scarface, The Thomas Crown Affair, Straw Dogs, While You Were Sleeping, Once Were Warriors, The Sixth Sense

Item-based collaborative Filtering	Movie ID 549 Title: Basquiat	Basquiat, Life Is a Long Quiet River, Love and Other Disasters, The Mummy Returns, The Soft Skin, Edward Scissorhands, Ninotchka, Night and Fog, The Kite Runner, The Wild Geese

Run-Time Analysis: We've face challenges (memory errors while running algorithm during implementation of projects due to lack of computing resources as the project demands computational power. Using 16GB RAM of Windows OS, below are detailed run time analysis of main aspects of project.

Data Loading from csv to data frame: ~25 Seconds

Data Visualization: ~6 Minutes

Data Pre-Processing (preprocessing columns, merging data files, text processing & extracting features): ~20 Seconds.

Content Based Recommendation Algorithm: ~5 Seconds.

User Based Recommendation Algorithm: ~5 Minutes 50 Seconds

Movie based Recommendation Algorithm: ~19 Minutes 45 Seconds.

Evaluation of User Based Recommendation algorithm using SVD Model: ~ 8 Minutes

If you encounter memory error while executing User/Movie based recommendations algorithm, please re-start kernel and try to run particular algorithm alone, as it takes input of 26 million records and vectorizes them it need lot of memory, else you can redirect rating to ratings_small file which has subset of rows.

Pros and Cons:

Collaborative filtering is efficient at identifying complicated associations and patterns in user's preferences. It can handle sparse data effectively. Also, it doesn't require details about the new movie for which it has little or no past interaction data. However, it can be vulnerable to the cold start problem for new users, as it doesn't have any historical data about them. Also, this method may pose challenges if data size grows. On the other hand, User-based approach handles cold-start problems effectively well among all three approaches that we implemented. It provides customized recommendations based on user choices that share similar interests. Though, this approach is better for cold-start problems, but its performance suffers if data is sparse. The item-based filtering focuses more on scalability and can effectively recommend new movies. However, this approach is still affected by the cold-start problem and heavily relies on the content information to calculate similarity matrices.

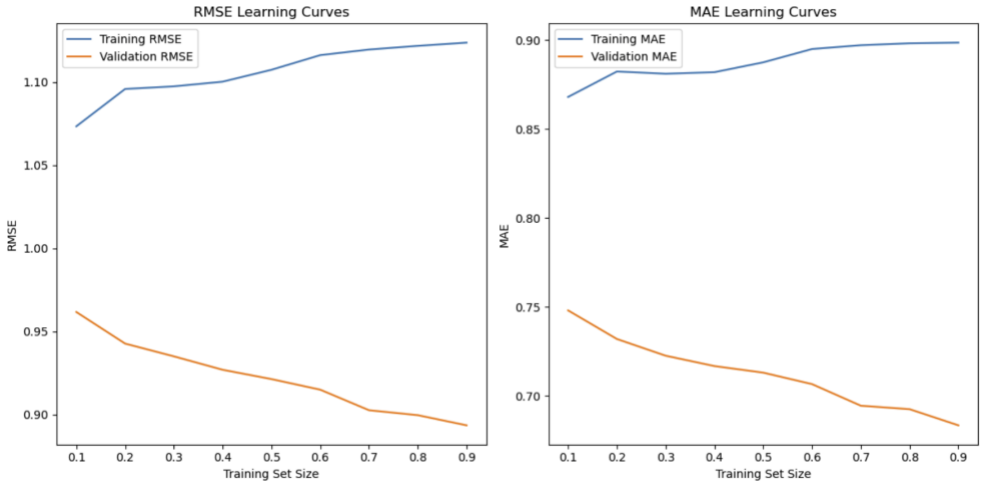
In general, all three approaches can have potential biases towards popular movies or genres. Handling sparse data, especially in user-movie interactions, poses a challenge in accurately predicting preferences, especially in collaborative filtering. The cold-start problem, where recommending movies for new users or new movies with little data is difficult. Overfitting of training data in models like SVD in collaborative filtering might reduce the model's ability to generalize to new data.

Performance Comparison:

Our recommendation system slightly outperforms the benchmark with an RMSE of 0.898 compared to 0.9, and an MAE of 0.692 versus the benchmark of 0.7, indicating reliable predictive accuracy.

Learning curves to Diagnose Algorithms:

The learning curves indicate that the model is learning but does not significantly improve with more data. Overfitting and underfitting are not evident. Further performance gains may require model improvements beyond adding data.



Error Analysis: In error analysis, we have explored false positives where our system predicted higher ratings than users gave. For instance, we saw predictions like 3.688 for a user-rated 3.0 movie and 3.865 for a 2.0 rated movie, highlighting cases of overestimation. On the other side, true positives show where the model accurately predicted high ratings, revealing its strengths without the issues seen in false positives. Analysing FN & TN helps us locate areas where the model underestimates user satisfaction or correctly predicts low ratings. This comprehensive analysis helps point areas for improvement, like incorporating nuanced feedback or adjusting sensitivity to specific features in our recommendation system.

Conclusion and Future Directions

The project successfully implemented three strategies by integrating content-based, collaborative, and item-based filtering techniques. The system achieved a balance between content similarity and user preferences, providing accurate recommendations by leveraging movie features such as genres, keywords, and thematic elements. Collaborative filtering personalized suggestions based on individual user tastes using the SVD algorithm for nuanced rating predictions. Evaluation metrics, specifically RMSE, indicated high accuracy in predicting user ratings, with lower values showcasing effective preference prediction. The comprehensive feature analysis considered various aspects like directors, cast, and detailed movie overviews. Data visualizations, including genre and rating distributions, informed recommendation logic. The system generated top-N recommendations through a custom function, enhancing the user experience with relevant and concise movie options. Future plans could involve exploring hybrid models, incorporating dynamic real-time data, and employing advanced machine learning techniques such as neural networks. Additionally, there is a focus on building a web-based UI for broader accessibility.

Team Members Contribution:

Presentation Slides, Report	Everyone worked on it.
Data Visualization, Content Based Recommendation System Learning Curves, Error Analysis	Lokesh (primary), Thanoj
Collaborative and User Based Filtering	Thanoj(primary), Lokesh
Evaluation Strategies	Aiden