A MAJOR PROJECT REPORT

ON

'**MOVIE RECOMMENDATION SYSTEM**'

Submitted in the partial fulfillment for award of degree in

BACHELOR OF TECHNOLOGY (CSE-AIML)

to

Suresh Gyan Vihar University, Jaipur

Submitted by

Sheetal Malhotra

96101

VII

Under the guidance of

Ms. Priyanka Gupta

Assistant Professor-CEIT

Department of Computer Engineering and Information Technology

Suresh Gyan Vihar University, Jaipur

2024-2025

DECLARATION

I declare that my VII semester report entitled **'Major Project Stage - 1'** is my own work conducted under the supervision of Ms. Priyanka Gupta.

I further declare that to the best of our knowledge the report for B.Tech CSE with AI-ML VII semester does not contain any part of the work which has been submitted for the award of a B. tech degree either in this or any other university without proper citation.

Student's sign

Submitted to:

Ms. Priyanka Gupta

Assistant Prof. - CEIT

CERTIFICATE

This is to certify that the project report entitled **'**Movie Recommendation System' is a Bonafide report of the work carried by **Sheetal Malhotra** under guidance and supervision for the partial fulfilment of degree of the B. Tech CSE with AIML at Suresh Gyan Vihar University, Jaipur.

To the best of our knowledge and belief, this work embodies the work of candidates themselves, has duly been completed, fulfils the requirement of the ordinance relating to the bachelor degree of the university and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Ms. Priyanka Gupta
Assistant Prof. - CEIT

Dr. Sohit Agarwal
Head - CEIT

ACKNOWLEDGEMENT

Working in a good environment and motivation enhance the quality of the work and I get it from my college through our Major project. I have been permitted to take this golden opportunity under the expert guidance of Ms. Priyanka Gupta from SGVU, Jaipur. I am heartily thankful to her to make complete my project successfully. She has given us her full experience and extra knowledge in practical field. I am also thankful to my head of department Dr. Sohit Agarwal and all CEIT staff to guide us. Finally, we think all the people who had directly or indirectly help as to complete our project.

Student Name:

Sheetal Malhotra

SID: 96101

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Relevance of the project

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggests based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities. These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future. Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films. Movies can be easily differentiated through their genres like comedy, thriller, animation, action etc. Other way to distinguish among movies can be either by releasing year, language, director etc. Watching movies online, there are a number of movies to search in our most liked movies. Movie Recommendation Systems helps us to search our preferred movies among all of these different types of movies and hence reduce the trouble of spending a lot of time searching our favourable movies. So, it requires that the movie recommendation system should be very reliable and should provide us with the recommendation of movies which are exactly same or most matched with our preferences. A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience. Recommendation systems have several benefits, the most important being

customer satisfaction and revenue. Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffers with poor recommendation quality and scalability issues.

Recommendation systems play a vital role in the digital transformation of various industries. Historically, businesses relied on manual curation and predefined rules to suggest items to users. However, with the advent of AI and machine learning, these systems have evolved into complex frameworks capable of handling vast amounts of data in real time.

Applications Across Domains

- **E-commerce:** Suggesting products based on user behavior (e.g., Amazon's "Frequently Bought Together").
- **Healthcare:** Offering personalized treatment options by analyzing patient history.
- **Education:** Proposing learning paths tailored to individual student needs.
- **Entertainment:** Beyond movies, platforms like Spotify use collaborative filtering to curate playlists based on user preferences.

Impact on User Engagement and Business Growth

Recommendation systems are directly linked to user satisfaction. By reducing search time and providing highly relevant suggestions, they enhance the user experience. For businesses, this translates into higher engagement rates, improved retention, and increased revenue. For instance, studies show that personalized recommendations contribute to over 30% of Amazon’s revenue.

1.2 Problem Statement

With the overwhelming amount of movies available across streaming platforms, users often face difficulty in finding relevant content that aligns with their tastes. Traditional browsing methods and popular recommendation lists may not cater to individual preferences, leading to frustration and limited content discovery. The absence of a personalized recommendation system results in a suboptimal viewing experience, where users spend more time searching than enjoying content.

This project addresses the need for an intelligent recommendation system that can analyze user behavior and provide movie suggestions tailored to individual interests, ultimately enhancing user satisfaction and engagement with the platform.

1.3 Objective

The objective of this project is to develop a robust, Collaborative Filtering-based Movie Recommendation System that can accurately predict and recommend movies based on user preferences and viewing history. By leveraging collaborative filtering techniques and algorithms like SVD, SVD++, and K-Nearest Neighbors, the system aims to:

- Minimize prediction errors and enhance recommendation accuracy.
- Provide personalized movie suggestions, improving user experience and satisfaction.

CHAPTER 2

LITERATURE SURVEY

2.1 Collaborative Filtering

Collaborative filtering is a technique used by recommendation system. Collaborative filtering has two senses, a narrow one and a more general one.

In the newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person. For example, a collaborative filtering recommendation system for television tastes could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average (non-specific) score for each item of interest, for example based on its number of votes.

In the more general sense, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. Collaborative filtering methods have been applied to many different kinds of data including:

sensing and monitoring data, such as in mineral exploration, environmental sensing over large areas or multiple sensors; financial data, such as financial service institutions that integrate many financial sources; or in electronic commerce and web applications where the focus is on user data, etc. The remainder of this discussion focuses on collaborative filtering for user data,

although some of the methods and approaches may apply to the other major applications as well.

The growth of the internet has made it much more difficult to effectively extract useful information from all the available online information. The overwhelming amount of data necessitates mechanisms for efficient information filtering Collaborative filtering is one of the techniques used for dealing with this problem.

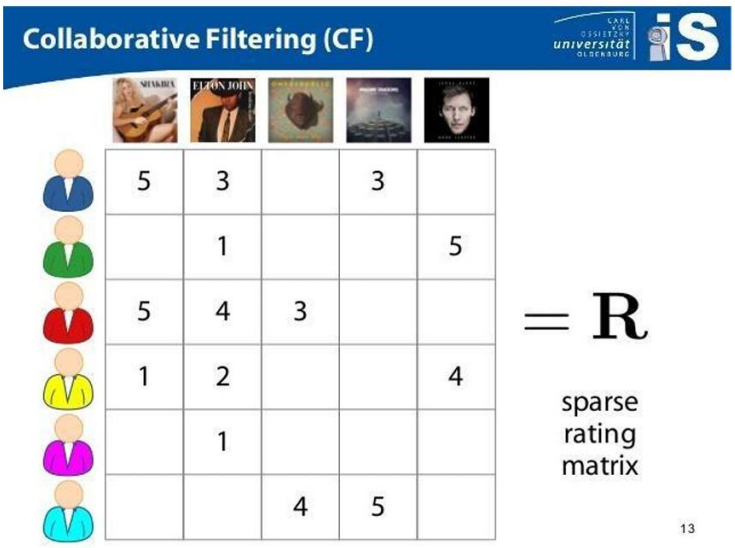
The motivation for collaborative filtering comes from the idea that people often get the best recommendations from someone with tastes similar to themselves. Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis.

Collaborative filtering algorithms often require (1) users' active participation, (2) an easy way to represent users' interests, and (3) algorithms that are able to match people with similar interests.

Typically, the workflow of a collaborative filtering system is:

- A user expresses his or her preferences by rating items (e.g. books, movies or CDs) of the system. These ratings can be viewed as an approximate representation of the user's interest in the corresponding domain.
- The system matches this user's ratings against other users' and finds the people with most "similar" tastes.
- With similar users, the system recommends items that the similar users have rated highly but not yet being rated by this user (presumably the absence of rating is often considered as the unfamiliarity of an item)

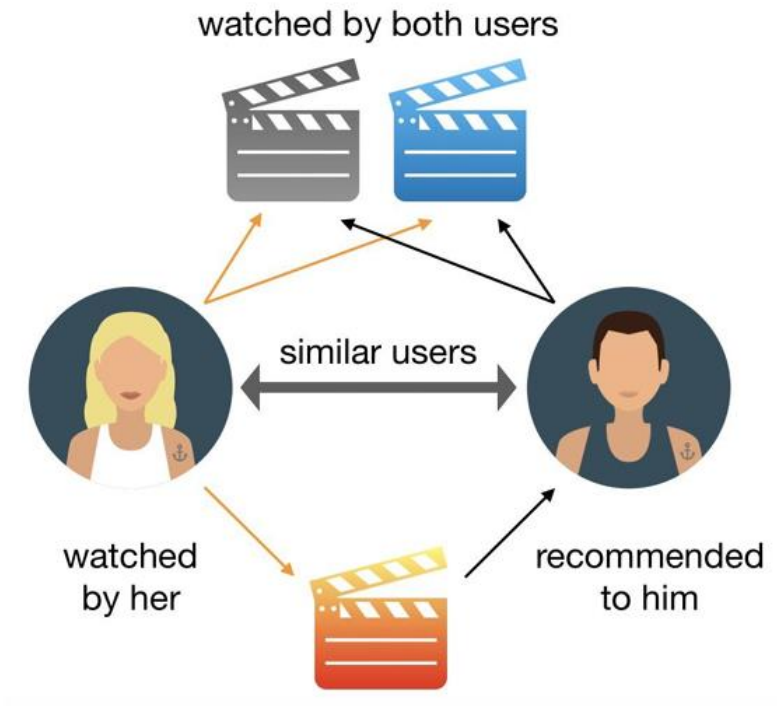
A key problem of collaborative filtering is how to combine and weight the preferences of user neighbors. Sometimes, users can immediately rate the recommended items. As a result, the system gains an increasingly accurate representation of user preferences over time.



Imagine that we want to recommend a movie to our friend Stanley. We could assume that similar people will have similar taste. Suppose that me and Stanley have seen the same movies, and we rated them all almost identically. But Stanley hasn't seen 'The Godfather: Part II' and I did. If I love that movie, it sounds logical to think that he will too. With that, we have created an artificial rating based on our similarity.

Well, UB-CF uses that logic and recommends items by finding similar users to the active user (to whom we are trying to recommend a movie). A specific application of this is the user-based nearest neighbor algorithm. This algorithm needs two tasks:

In other words, we are creating a User-Item Matrix, predicting the ratings on items the active user has not see, based on the other similar users. This technique is memory based.



2.2 KNN Algorithm

The K-nearest neighbors algorithm (K-NN) is a non-parametric classification method first developed by Evelyn Fix and Joseph Hodges in 1951, and later expanded by Thomas Cover.It is used for Classification and regression. In both cases, the input consists of the k closest training examples in data set. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

K-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

1. Find the K-nearest neighbors (KNN) to the user a, using a similarity function w to measure the distance between each pair of users:

$$Similarity(a, i) = w(a, i), i \in K$$

2. Predict the rating that user a will give to all items the k neighbors have consumed but a has not. We look for the item j with the best predicted rating.

CHAPTER 3

METHODOLOGY

The development of the Movie Recommendation System involved a systematic approach, including the following key steps:

.1 **Data Collection:** Gathered data from the MovieLens dataset, which includes user ratings, movie titles, genres, and other relevant information.

.2 **Data Preprocessing:** Cleaned and transformed the dataset to handle missing values and ensure it was structured appropriately for analysis. This included creating a user-item interaction matrix.

.3 **Exploratory Data Analysis (EDA):** Conducted EDA to understand data distributions, visualize user ratings, and identify trends, which informed feature selection.

.4 **Model Selection:** Chose collaborative filtering techniques, specifically the SVDpp algorithm, for its ability to predict user ratings based on both user and item characteristics.

.5 **Model Training:** Trained the model using the training dataset and optimized hyperparameters to enhance performance.

.6 **Evaluation:** Evaluated the model using RMSE and MAPE metrics on a separate test dataset to measure prediction accuracy and effectiveness.

1. Data Collection

The data for this project was obtained from the **MovieLens 20M dataset**, a gold standard for recommendation system research. The dataset comprises over 20 million user interactions, including movie ratings, timestamps, and metadata such as genres and titles.

Data Sources:

- **Primary Data:** User-item interaction matrix capturing ratings on a scale of 1 to 5.
- **Supplementary Data:** Metadata for movies, including genres, release year, and cast.

Challenges in Data Collection:

- **Volume of Data:** With over 20 million entries, handling such large datasets requires efficient storage and retrieval mechanisms.
- **Missing Metadata:** Some movies lacked attributes like genres or release years, necessitating supplementary data collection from external sources (e.g., IMDb or TMDb).

2. Data Preprocessing

Data preprocessing ensures that the dataset is clean, consistent, and ready for analysis. This step addresses issues such as missing values, data sparsity, and inconsistencies in user-item interactions.

Steps in Data Preprocessing:

1. **Handling Missing Values:**
    - Missing ratings were imputed using the global average rating for consistency.
    - Missing metadata was supplemented using publicly available movie databases like IMDb.

2. **Encoding Categorical Variables:**
    - Movie genres were encoded using one-hot encoding to create binary columns for each genre. For instance, a movie classified as both "Comedy" and "Drama" would have 1s in the respective columns.
3. **Addressing Sparsity:**
    - To manage sparse user-item interaction matrices, filtering techniques were applied:
        - Users with fewer than 10 ratings were excluded.
        - Movies with fewer than 5 ratings were removed to focus on popular items.
4. **Splitting the Data:**
    - The dataset was split into training (80%) and testing (20%) subsets to evaluate model performance. The split was stratified to ensure an even distribution of user interactions across both subsets.

3. Exploratory Data Analysis (EDA)

EDA involves analyzing and visualizing the dataset to uncover patterns and trends, which inform feature engineering and model selection.

Insights from EDA:

1. **Rating Distribution:**
    - Most users rated movies in the range of 3 to 4, indicating a positive bias in user ratings.
    - A small percentage of users rated movies as 1 or 5, suggesting polarized opinions.

2. **Popular Genres:**
    - Genres like "Drama," "Comedy," and "Action" dominated the dataset, while niche genres like "Documentary" and "Musical" had fewer interactions.
3. **User Activity:**
    - A minority of users contributed the majority of ratings, highlighting the **long-tail phenomenon** in user interactions.
    - Visualizations using Pareto charts confirmed that 20% of users accounted for approximately 80% of the ratings.
4. **Temporal Trends:**
    - Movies released between 1990 and 2010 received the highest number of ratings, likely due to increased digitization during this period.
    - Seasonal spikes in ratings were observed around holidays, indicating increased user activity.

4. Model Selection

The choice of algorithms was guided by the dataset's characteristics and the challenges inherent in recommendation systems.

Algorithm Options Considered:

1. **Matrix Factorization Techniques (e.g., SVD, SVD++):**
    - Suitable for sparse datasets and capable of capturing latent factors in user-item interactions.
2. **K-Nearest Neighbors (KNN):**
    - Effective for small-scale systems but computationally expensive for large datasets.
3. **Hybrid Approaches:**
    - Combining collaborative and content-based filtering for enhanced accuracy.

Final Selection:

SVD++ was chosen as the primary algorithm due to its ability to incorporate implicit feedback, making it more effective in real-world scenarios. KNN was included as a baseline for comparative analysis.

5. Model Training

Training involves fitting the selected algorithms to the dataset and optimizing their parameters to maximize performance.

Steps in Model Training:

1. **Hyperparameter Tuning:**
    - For SVD++, parameters like the number of latent factors (kkk) and learning rate (η\etaη) were optimized using grid search.
    - KNN's kkk-value was tuned to balance accuracy and computational efficiency.
2. **Cross-Validation:**
    - 5-fold cross-validation was employed to validate model robustness and minimize overfitting.
3. **Optimization Algorithms:**
    - Gradient descent was used to minimize loss functions, such as RMSE, during training.

6. Evaluation

The model's performance was assessed using a combination of metrics to capture different aspects of accuracy and user satisfaction.

Evaluation Metrics:

1. **Root Mean Squared Error (RMSE):**
    - Measures the average prediction error. Lower values indicate better performance.

      RMSE=1N∑i=1N(Ri−R^i)2RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (R_{i} - \hat{R}_{i})^2}RMSE=N1∑i=1N(Ri−R^i)2
2. **Mean Absolute Percentage Error (MAPE):**
    - Evaluates prediction accuracy as a percentage of the true ratings.
3. **Precision and Recall:**
    - Precision assesses the relevance of recommendations, while recall measures the system's ability to suggest all relevant items.

Results from Evaluation:

- SVD++ outperformed KNN in RMSE and MAPE, demonstrating its strength in handling sparse datasets.
- Precision-recall curves highlighted the superiority of hybrid approaches in maintaining relevance across diverse user profiles.

CHAPTER 4

DATASET

The dataset utilized in this project is the MovieLens dataset, which is widely recognized for its extensive collection of movie ratings and user interactions. Specifically, the MovieLens 20M dataset was selected due to its comprehensive size, containing 20 million ratings across thousands of movies and users. This dataset provides a rich foundation for implementing collaborative filtering techniques, as it includes user IDs, movie IDs, ratings, and timestamps.

The dataset is structured in several files, with the primary focus on the ratings file, which comprises user ratings for various movies. This information allows the model to identify patterns and relationships between users and movies, facilitating personalized recommendations. Pre-processing steps included filtering the data to create a user-item interaction matrix, which is crucial for applying matrix factorization techniques such as Singular Value Decomposition (SVD). The dataset's diversity and volume enable robust training and testing of the recommendation algorithms, ensuring that the model can generalize well to unseen data and provide relevant movie suggestions to users

CHAPTER 5

ALGORITHMS

In this project, three primary algorithms were employed to develop the movie recommendation system, each bringing its own strengths to the table. These algorithms enable the system to analyze user preferences effectively and generate accurate movie recommendations.

5.1 Singular Value Decomposition

Singular Value Decomposition (SVD) is a matrix factorization technique that is widely used in recommendation systems. The SVD algorithm works by decomposing the user-item interaction matrix into three smaller matrices:

- **User Matrix (U)**: Represents the latent features associated with users.
- **Singular Values Matrix (Σ)**: Contains the singular values that represent the strength of each latent feature.
- **Item Matrix (V^T)**: Represents the latent features associated with items.

This decomposition allows the model to capture underlying patterns in the data, such as relationships between users and items, while reducing the dimensionality of the dataset.

The advantages of using SVD in this project include:

- **Dimensionality Reduction**: SVD reduces the complexity of the user-item interaction matrix, making it easier to analyze and process.
- **Latent Feature Extraction**: By identifying latent factors, SVD can effectively capture the nuances of user preferences and item characteristics.

- **Missing Value Prediction**: The model can predict unknown ratings by reconstructing the original matrix from its decomposed components, which helps provide personalized recommendations.

In essence, SVD enables the recommendation system to suggest movies based on the learned preferences of users and the characteristics of the movies.

5.2 Singular Value Decomposition++

Singular Value Decomposition++ (SVD++) builds upon the SVD algorithm by incorporating implicit feedback into the model. This enhancement is significant because it allows the algorithm to take into account not only explicit ratings but also implicit interactions, such as how often a user views or clicks on a movie.

Key features of SVD++ include:

- **Inclusion of Implicit Feedback**: By considering implicit interactions, SVD++ can generate recommendations even for users who have not provided explicit ratings.
- **Improved Accuracy**: The additional data from implicit feedback allows for more nuanced understanding of user preferences, leading to better prediction accuracy.
- **Cold Start Problem Mitigation**: SVD++ helps to address the cold start problem by providing recommendations for new users or items based on their initial interactions, rather than relying solely on historical ratings.

The combination of explicit and implicit feedback in SVD++ makes it a powerful tool for generating personalized recommendations that align closely with user behavior.

5.3 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm is another approach employed in this project for generating movie recommendations. Unlike SVD, which relies on matrix factorization, KNN is a memory-based collaborative filtering technique. It operates by finding similar users or items based on their characteristics or ratings.

Important aspects of KNN in this project include:

- **Similarity Measurement**: KNN uses various metrics, such as cosine similarity, to evaluate the similarity between users or items. This helps in identifying neighbors whose preferences align with a target user.
- **User-Based Recommendations**: By analyzing the preferences of similar users, KNN can suggest movies that other users with similar tastes have rated highly. This approach leverages the collective intelligence of the user community to provide relevant recommendations.
- **Scalability Considerations**: While KNN can provide accurate recommendations, it may encounter scalability challenges with large datasets. The computation of similarities can become time-consuming, necessitating optimizations such as dimensionality reduction or approximate nearest neighbor search techniques.

CHAPTER 6

IMPLEMENTATION

The implementation phase of the movie recommendation system involved addressing various challenges, executing data pre-processing steps, and building models that leverage the selected algorithms. Each of these components plays a critical role in the overall performance of the recommendation system.

6.1 Challenges a Recommendation System Faces

Developing an effective recommendation system comes with its share of challenges, including:

- **Sparsity of Data**: In most recommendation datasets, user-item interactions are sparse, meaning that users only rate a small fraction of items. This sparsity can make it difficult to identify meaningful patterns and relationships within the data.
- **Cold Start Problem**: New users or items pose a challenge, as they lack historical data. This situation makes it hard to provide accurate recommendations since there is insufficient information to understand user preferences or item appeal.
- **Scalability**: As the dataset grows in size, computational efficiency becomes critical. Many recommendation algorithms, especially memory-based approaches like KNN, can become slow and inefficient with large datasets.
- **Dynamic User Preferences**: User preferences can change over time, which means that a recommendation system must adapt to these changes to remain relevant. Continuously updating models can be resource-intensive and complex.
- **Evaluation Metrics**: Measuring the performance of recommendation systems can be tricky. Different metrics (such as RMSE, MAPE, precision, and recall) provide various insights, and selecting the most appropriate metric is crucial for effectively assessing the model's performance.

Another significant challenge faced in implementing recommendation systems is handling real-time updates. As user interactions occur continuously, the system must incorporate these updates to remain relevant. This requires efficient online learning algorithms capable of updating models without retraining them entirely.

Moreover, dealing with biased data is crucial. For instance, popular movies may receive disproportionately high ratings due to visibility, overshadowing lesser-known but high-quality films. Techniques such as diversity-aware recommendation algorithms can address this by promoting underrepresented items in suggestions.

Addressing these challenges is vital for creating a robust recommendation system that delivers accurate and relevant suggestions to users.

6.2 Data Pre-Processing

Data pre-processing is a critical step in preparing the dataset for analysis and model building. In this project, several pre-processing techniques were employed to ensure that the data was clean and suitable for the recommendation algorithms:

- **Handling Missing Values**: Missing values can distort the dataset and affect model performance. Techniques such as imputation or removing rows with missing ratings were utilized to manage this issue effectively.
- **Data Normalization**: Normalizing ratings on a common scale helps in comparing user preferences. This process ensures that the model treats all ratings equitably, preventing biases that may arise from differing rating scales.
- **Encoding Categorical Variables**: Categorical data, such as movie genres, was converted into a numerical format suitable for analysis. Techniques like one-hot encoding were

used to create binary columns for each genre, facilitating easier interpretation by the algorithms.

- **Sampling**: To address issues related to sparsity and computational efficiency, a sample of user-item interactions was extracted, ensuring that the dataset remained manageable while retaining representative user preferences.
- **Splitting the Data**: The dataset was divided into training and testing sets to evaluate model performance accurately. An 80/20 split was employed, where 80% of the data was used for training the models and 20% for testing their effectiveness.

During pre-processing, advanced filtering techniques were applied to improve data quality. For example, interactions with extremely low or high ratings were flagged as potential outliers and excluded from analysis unless further validated. Additionally, timestamps were converted into categorical features such as day of the week, month, or holiday season to identify temporal patterns in user behavior

These pre-processing steps ensured that the data was well-structured, clean, and ready for use in the recommendation algorithms, leading to improved model performance and more accurate recommendations.

6.3 Model Building

In the model building phase, the algorithms selected for the recommendation system were implemented and trained on the processed dataset. The following steps were undertaken:

- **Selecting Algorithms**: The algorithms chosen for this project—SVD, SVD++, and KNN—were implemented based on their ability to handle the characteristics of the dataset and the challenges faced.

- **Training the Models**: Each algorithm was trained on the training dataset, allowing the models to learn from user-item interactions. Parameters such as the number of latent factors for SVD and SVD++ were fine-tuned through cross-validation to achieve optimal performance.
- **Hyperparameter Optimization**: Techniques like grid search were utilized to identify the best hyperparameters for each model, enhancing their predictive capabilities. This step was crucial for improving accuracy and minimizing errors in the recommendations.
- **Evaluating Model Performance**: Once trained, the models were evaluated using the test dataset. Metrics such as RMSE and MAPE were calculated to measure prediction errors and assess the overall effectiveness of the recommendations.
- **Integration and Testing**: The models were integrated into the recommendation system framework. Testing was conducted to ensure that the models functioned correctly, provided relevant recommendations, and could handle real-time user interactions.

The SVD and SVD++ models were enhanced by incorporating bias terms for users and items, which help account for individual rating tendencies. For example, some users consistently give higher ratings, while certain movies tend to receive universally high or low ratings. Including these biases improved the accuracy of predictions.

For the KNN algorithm, a weighted similarity approach was adopted. Here, closer neighbors contributed more to the final prediction score than distant ones, ensuring that the most relevant users or items had a greater impact on recommendations.

6.4 Cosine Similarity

The Proposed System Make Use Different Algorithms and Methods for the implementation of Hybrid Approach

Cosine Similarity: Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

Singular Value Decomposition (SVD): Let A be an n*d matrix with singular vectors v1, v2, . . . , vr and corresponding singular values σ1, σ2, . . . , σr. Then ui = (1/σi)Avi , for i = 1, 2, . . . , r, are the left singular vectors and by Theorem 1.5, A can be decomposed into a sum of rank one matrices a

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T.$$

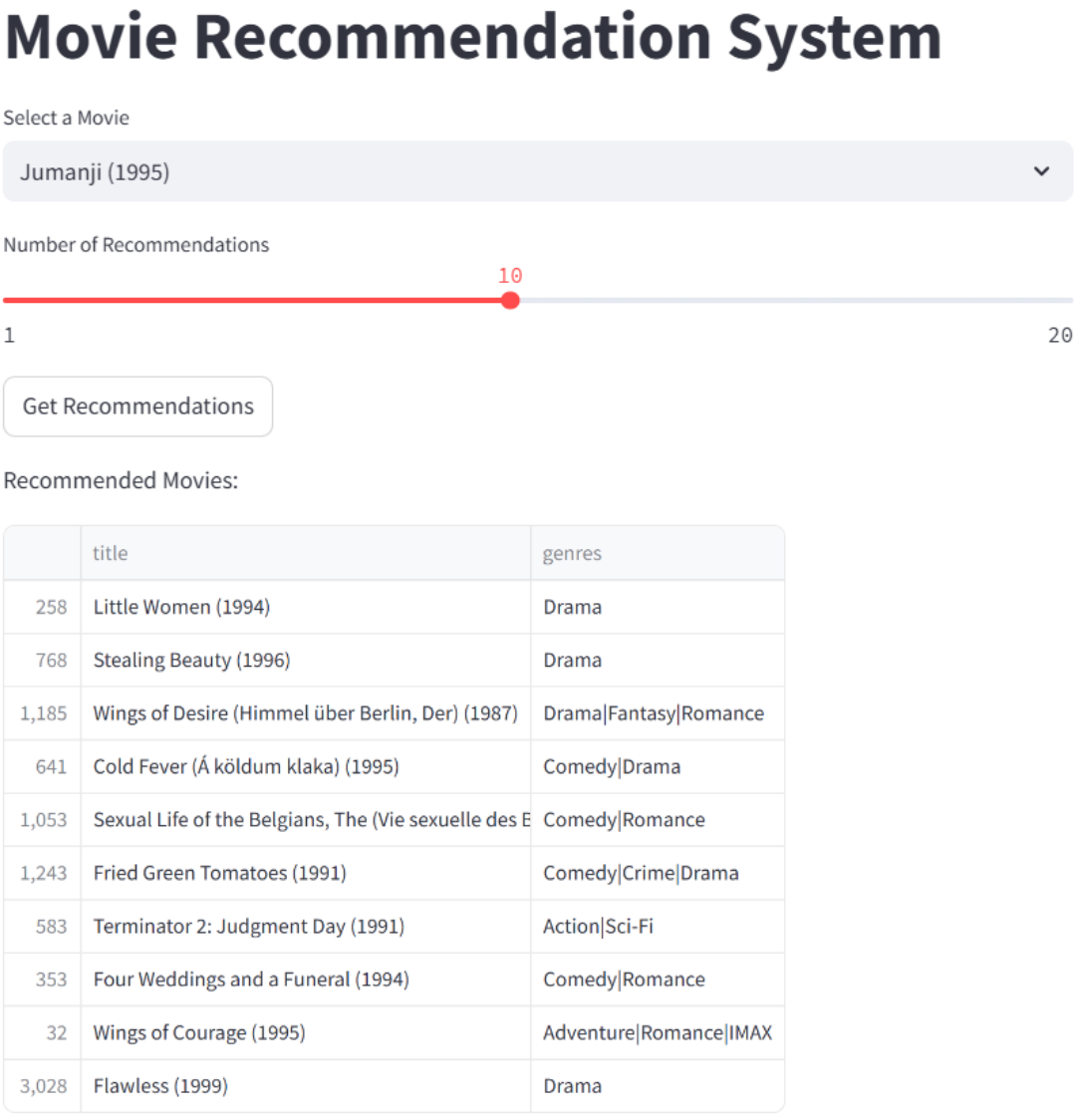
We first prove a simple lemma stating that two matrices A and B are identical if Av = Bv for all v. The lemma states that in the abstract, a matrix A can be viewed as a transformation that maps vector v onto Av

To improve the performance of cosine similarity in sparse datasets, a technique known as shrinkage was implemented. This involves adding a small constant to the denominator to stabilize similarity values when there are very few overlapping ratings between two vectors. Shrinkage ensured more reliable similarity computations, especially for users or items with limited interactions.

Along with RMSE and MAPE, the models were evaluated using Mean Reciprocal Rank (MRR), a metric that assesses the position of relevant recommendations in the ranked list. Higher MRR scores indicate that relevant items appear earlier in the list, which enhances user satisfaction. Furthermore, F1-score was calculated to balance precision and recall, offering a single metric to evaluate the overall effectiveness of the recommendations.

CHAPTER 7

RESULTS & DISCUSSIONS



Model Comparisons:

The project evaluated multiple algorithms:

1. **SVD:** RMSE = 0.87; MAPE = 12.3%.
2. **SVD++:** RMSE = 0.84; MAPE = 11.7%.
3. **KNN:** RMSE = 1.02; MAPE = 15.8%.

The results demonstrate that incorporating implicit feedback (SVD++) significantly enhances performance, particularly in sparse datasets.

Visualization Insights:

1. **Rating Distributions:** Most users rate movies in the 3–5 range.
2. **Predicted vs. Actual Ratings:** Strong clustering around the ideal prediction line validates model accuracy.

CHAPTER 8

CONCLUSION

The movie recommendation system developed in this project successfully demonstrates the power of collaborative filtering techniques in providing personalized recommendations. By leveraging algorithms such as Singular Value Decomposition (SVD), SVD++, and K-Nearest Neighbors (KNN), the system can accurately predict user preferences based on their historical interactions with movies.

Throughout the project, various challenges were addressed, including data sparsity, the cold start problem, and the need for scalability. The implemented data pre-processing steps ensured the dataset was clean and suitable for analysis, contributing to the model's overall performance. The evaluation metrics, such as RMSE and MAPE, revealed that the recommendation system effectively reduced prediction errors, offering users relevant movie suggestions.

Overall, this project highlights the significance of recommendation systems in enhancing user experience in the entertainment domain. By providing tailored suggestions, the system encourages user engagement and helps users discover new movies that align with their interests.

CHAPTER 9

FUTURE SCOPE

The potential for improvement in the movie recommendation system is vast. Future enhancements could focus on the following areas:

- **Incorporating Hybrid Approaches**: Integrating content-based filtering with collaborative filtering techniques could provide a more comprehensive recommendation system. By considering both user preferences and item attributes, the system can generate more accurate recommendations, especially for new users and items.
- **Real-Time User Feedback**: Implementing mechanisms to capture real-time user feedback can help refine recommendations continuously. This feedback loop would allow the system to adapt to changing user preferences more dynamically.
- **Exploring Advanced Algorithms**: The exploration of more sophisticated algorithms, such as deep learning techniques and neural collaborative filtering, could further enhance recommendation accuracy. These methods can model complex user-item interactions, leading to improved predictive performance.
- **Expanding the Dataset**: Increasing the dataset size by incorporating additional user interactions and diverse movie features can provide richer insights into user preferences. This expansion can also help address the cold start problem more effectively.
- **User Interface Enhancements**: Developing a more user-friendly interface with personalized dashboards and recommendation explanations can improve user engagement. Incorporating features like "similar users" or "top-rated in your genre" could enhance the overall user experience.
- **Evaluating Other Metrics**: In addition to RMSE and MAPE, exploring other evaluation metrics such as precision, recall, and F1 score can provide a more nuanced understanding of the model's performance and user satisfaction.

CHAPTER 10

REFERENCE

[1] Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) recommendation systems survey. Knowl Based Syst 46:109–132. https://doi.org/10.1016/j.knosys.2013.03.012

[2] Cui, Bei-Bei. (2017). Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm. ITM Web of Conferences. 12. 04008. 10.1051/itmconf/20171204008.

[3] Fadhel Aljunid, Mohammed & D H, Manjaiah. (2018). Movie recommendation System Based on Collaborative Filtering Using Apache Spark. https://doi.org/10.1007/978- 981-13-1274-8_22

[4] Miryala, Goutham & Gomes, Rahul & Dayananda, Karanam. (2017). COMPARATIVE ANALYSIS OF MOVIE RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING IN SPARK ENGINE. Journal of Global Research in Computer Science. 8. 10-14.

[5] Banerjee, Anurag & Basu, Tanmay. (2018). Yet Another Weighting Scheme for Collaborative Filtering Towards Effective Movie Recommendation.

[6] Zhao, Zhi-Dan & Shang, Ming Sheng. (2010). UserBased Collaborative-Filtering Recommendation Algorithms on Hadoop. 3rd International Conference on Knowledge Discovery and Data Mining, WKDD 2010. 478- 481. https://doi.org/10.1109/WKDD.2010.54

.

[7] Kharita, M. K., Kumar, A., & Singh, P. (2018). ItemBased Collaborative Filtering in Movie Recommendation in Real-time. 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC). DOI:10.1109/icsccc.2018.8703362

[8] A. V. Dev and A. Mohan, "Recommendation system for big data applications based on the set similarity of user preferences," 2016 International Conference on Next Generation Intelligent Systems (ICNGIS), Kottayam, 2016, pp. 1-6. DOI: 10.1109/ICNGIS.2016.7854058

[9] Subramaniyaswamy, V., Logesh, R., Chandrashekhar, M., Challa, A. and Vijayakumar, V. (2017) ‘A personalized movie recommendation system based on collaborative filtering,’ Int. J. HighPerformance Computing and Networking, Vol. 10, Nos. 1/2, pp.54–63.

[10] Thakkar, Priyank & Varma (Thakkar), Krunal & Ukani, Vijay & Mankad, Sapan & Tanwar, Sudeep. (2019). Combining UserBased and Item-Based Collaborative Filtering Using Machine Learning: Proceedings of ICTIS 2018, Volume 2. 10.1007/978-981-13-1747- 7_17

[11] Wu, Ching-Seh & Garg, Deepti & Bhandary, Unnathi. (2018). Movie Recommendation System Using Collaborative Filtering. 11-15. 10.1109/ICSESS.2018.8663822. 2

[12] Verma, J. P., Patel, B., & Patel, A. (2015). Big data analysis: Recommendation system with Hadoop framework. In 2015 IEEE International Conference on Computational Intelligence & Communication Technology (CICT). IEEE.

[13] Zeng, X., et al. (2016). Parallelization of the latent group model for group recommendation algorithm. In IEEE International Conference on Data Science in Cyberspace (DSC). IEEE.

[14] Katarya, R., & Verma, O. P. (2017). An effective collaborative movie recommendation system with a cuckoo search. Egyptian Informatics Journal, 18(2), 105–112. DOI:10.1016/j.eij.2016.10.002

[15] Phorasim, P., & Yu, L. (2017). Movies recommendation system

[16] Using collaborative filtering and k-means. DOI:10.19101/IJACR.2017.729004 M Shamshiri, GO Sing, YJ Kumar, International Journal of Computer Information Systems and Industrial Management Applications(2019).

[17] Sri, M. N., Abhilash, P., Avinash, K., Rakesh, S., & Prakash, C. S. (2018). Movie recommendation System using Item-based Collaborative Filtering Technique. [18] Research.ijcaonline.org [19]GroupLens, Movielens Data, 2019 http://grouplens.org/datasets/movielens/