HYBRID MOVIE RECOMMENDATION SYSTEM USING MF – CF ALGORITHM

A MAJOR PROJECT REPORT

Submitted by

AASHIK A - RA2011003020559
NAVEEN M - RA2011003020568
HASHIR AHAMMED KABIR P - RA2011003020564

Under the guidance of

Dr. DEEPA. B

(Professor, Department of Computer Science and Engineering)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING
Of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM, CHENNAI -600089

May 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University U/S 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"HYBRID MOVIE RECOMMENDATION SYSTEM USING MF-CF ALGORITHM"** is the bonafide work of **AASHIK A [RA2011003020559], NAVEEN M [RA2011003020568], HASHIR AHAMMED KABIR [RA2011003020564]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an occasion on this or any other candidate.

SIGNATURE

Dr. B. DEEPA, M.E., Ph.D.,
Assistant Professor,
Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram, Chennai.

SIGNATURE

Dr. K. RAJA, M.E., Ph.D.,
Professor and Head,
Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram, Chennai.

Submitted for the project viva-voce held on 13/5/24 at SRM Institute of Science and Technology, Ramapuram, Chennai -600089.

INTERNAL EXAMINER

EXTERNAL EXAMINER

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM, CHENNAI - 89

DECLARATION

We hereby declare that the entire work contained in this project report titled "**HYBRID MOVIE RECOMMENDATION SYSTEM USING MF – CF ALGORITHM**" has been carried out by **AASHIK A [RA2011003020559], NAVEEN M [RA2011003020568], HASHIR AHAMMED KABIR [RA2011003020564]** at SRM Institute of Science and Technology, Ramapuram Campus, Chennai- 600089, under the guidance of **Dr. B. DEEPA, Assistant Professor**, Department of Computer Science and Engineering.

Place: Chennai

Date: 10/5/24

A. Aashik

Aashik A

M. Naveen

Naveen M

Hashir

Hashir Ahammed Kabir

RAMAPURAM SRM

SRM Institute of Science & Technology
Department of Computer Science and Engineering

Own Work Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : B.Tech Computer Science and Engineering

Student Name : Aashik A, Naveen M, Hashir Ahammed Kabir

Registration Number : RA2011003020559, RA2011003020568, RA2011003020564

Title of Work : Hybrid Movie Recommendation System Using MF-CF Algorithm

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate.
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present.
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website.

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

A. Aashik 10/5/24	M. Naveen 10/5/24	Hashir 10/05/24
RA2011003020559	RA2011003020568	RA2011003020564

ABSTRACT

The hybrid movie recommendation system, leveraging the Matrix Factorization (MF) and Content features (CF) algorithms, embodies a cutting-edge approach in the realm of personalized movie suggestions. This amalgamation of MF and CF not only enhances the accuracy and relevance of recommendations but also tackles the inherent limitations of individual methods. Matrix Factorization, renowned for its prowess in uncovering latent factors within user-item interaction data, delves deep into the underlying patterns and intricacies of user preferences. By distilling these latent factors, the system gains invaluable insights into users' tastes and affinities, thereby enriching the recommendation process. Collaborative Filtering, meanwhile, harnesses the collective wisdom of users by identifying similarities among their past interactions and extrapolating these insights to make informed recommendations. Through the synergistic integration of MF and CF, the hybrid system orchestrates a symphony of algorithms that not only broadens the scope of recommended content but also fine-tunes suggestions to cater to the nuanced preferences of individual users. The recommendation system solves cold start problem and data sparsity challenges, it ensures robust performance and adaptability across diverse user scenarios. This holistic approach to recommendation not only elevates the movie-watching experience but also fosters a deeper sense of engagement and satisfaction among users, thereby solidifying the system's position as a cornerstone of personalized entertainment discovery.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

1.1 Introduction

In advanced world or the coming era the web has totally changed into an essential bit of human life, people are seen regularly going up against the issue of settling on a choice. Straight forwardly from checking for a hotel to looking for quick theory alternatives, there is a tremendous degree of data available. Affiliates have said that they will offer mechanisms to deal with their clients in order to allow the clients to adjust to this data impact. Despite all, the test in the field of proposal structures has been ongoing and postponed. This is because of the difficulties and the lavish and demanding applications that have left a towering indentation. Different sorts of e-proposition approaches made and put to utilize gives a recommendation structure of dispersions at Flipkart, of appears at Amazon Prime, etc. These gave commitment to wealth for a parcel distinguished with electronic commerce districts, (for case, Flipkart.com) and do movement pictures. Recommender Frameworks make proposals and the framework will recognize motion pictures concurring to their choice and may in like strategy deliver comes about instantly/or on additional levels, a comprehended/unquestionable information. These endeavours recognized with clients and customers' responses can put a course for this proposition set and will be utilized if there ought to emerge an event of making unused proposals for taking after client structure affiliations. The budgetary capacity of theories recommender structures has driven most likely the best web commerce goals (for case: flipkart.com, amazon.in) and e-commerce enlisted affiliation. Amazon Prime brands those frameworks exceptionally extraordinary segment for these data and frameworks. Colossal bore modified supports add-on other estimation for client events. Redid recommender software is starting late in terms of giving its registered clients a variety of well-rounded information. These frameworks are really common these days and can be associated in various forms of utilisations depending upon the item connected with. Two common ordering exist for us to isolate the recommender structures:

a) Collaborative filtering method: This method proposes info are based over the similarity of ratings of a particular movie, among different registered users. Collaborative filtering, Community oriented

sifting framework proposes things are reliant over similarity strategies among customers possibly things. Cooperative separating, otherwise called social sifting, channels data utilizing others' suggestions. Customers who have recently concurred on the assessment of specific components are bound to acknowledge it again later on. For instance, an individual who needs to see a film can request suggestions from companions. The proposals of certain companions with comparable interests are more solid than others. This data is utilized on what film to watch. Communitarian Filtering, doesn't need whatever else with the exception of clients' past inclination on a lot of things. As it depends on authentic information, the center theory here is that the clients who have concurred earlier will be in general additionally concur later on. This system proposes such stuffs which were particular from comparable kind of customers. Community sifting has a few good conditions 1. In Content Based Filtering customer does assessments subsequently for authentic viewpoint and estimation and prediction of item is done. 2. It predicts precise results and proposals since proposals depend on client's comparability as opposite to data proportionality.

b) Content-based filtering: Content-based filtering is developed with consideration for the basic database data as well as the client's affinity profile. To precisely predict the items we have castoff the ratings recorded by the clients to the movies or TV Series and users favored likes and dislikes to the shows. And by the end of the day at the background of the software using Collaborative Filtering method Estimates support or resemble items that were previously preferred. It calculates and predicts the new shows and or earlier based predicted things and proposes best movies or shows based on his likes and dislikes items. It uses different strategies and projection methods on different areas of use. The majority of hybrid recommender systems adopt this technique. Older computations or movie predictions using MOVIEGEN datasets have many implementations; for example, this shows how user requests move and what has previously been searched is also preserved within the database or history. The client is given the option of choosing from a wide range of characteristics based on the number of ratings and the rating of each movie, among other factors.

We update the user choices in the database and computes a new set of results from the new data provided and based on the choices of the past visited history of customers. The software is developed using Python and a simple user interface

1.2 Problem Statement

One major obstacle for collaborative filtering (CF) recommendation systems is the cold start problem.CF relies on user-item interaction data to generate recommendations, meaning that only resources that have been utilized and rated by users can be included in the recommendation set. However, when a new resource is introduced to the system, it lacks ratings from users, rendering it ineligible for recommendation. This limitation hampers the system's ability to provide relevant suggestions for newly added resources, thereby hindering user engagement and satisfaction. Furthermore, the sparse matrix problem exacerbates the inefficiency of CF-based recommendation systems. The user-item score matrix utilized by CF is often sparse, as the average number of items rated by each user is significantly lower than the total number of items available. Additionally, many users may not actively rate items, further contributing to data sparsity. Consequently, the similarity measurements between users become less accurate, and the neighbor set used for recommendation lacks reliability, leading to suboptimal recommendation efficiency and effectiveness. Moreover, the limited recommendation coverage of current systems exacerbates the issue. These systems often fail to offer a diverse range of course suggestions, thereby constraining students' exploration and learning experiences. By focusing on a narrow subset of courses, these systems overlook opportunities for students to discover new interests and broaden their educational horizons. As a result, students may feel confined within a limited selection of recommended courses, impeding their overall academic growth and development.

1.3 Objective

This project's main goal is to create, develop, and implement a sophisticated movie recommendation system for our website in order to increase customer pleasure and engagement. By employing advanced techniques such as Matrix Factorization with Collaborative Filtering, we seek to create a recommendation engine capable of understanding and predicting user preferences with a high degree of accuracy. Through meticulous data collection and preprocessing, we aim to curate a comprehensive dataset comprising user interactions and rich movie metadata, including genres, actors, directors, and plot keywords. This dataset will serve as the foundation for training our recommendation model, enabling it to learn intricate patterns in user behaviour and movie characteristics. Our overarching goal

is to deliver personalized recommendations to each user based on their preferences. In order to do this, we will break down the user-item interaction matrix using Matrix Factorization algorithm order to find latent characteristics that indicate the user's preferences and movie properties. By optimizing model parameters and hyper parameters through rigorous experimentation and validation, we aim to fine-tune the recommendation engine to produce highly relevant and diverse movie suggestions. Integration of the recommendation system into our website's architecture will be seamless, with an intuitive user interface that allows users to effortlessly discover and explore recommended movies. The system will provide real-time recommendations tailored to each user's browsing history and preferences, thereby enriching their overall experience on our platform. In addition to delivering personalized recommendations, our project objective includes ongoing monitoring, evaluation, and optimization of the recommendation system's performance. Our aim is to create a user friendly movie recommendation system that not only drives engagement and retention on our website but also enhances the overall satisfaction and enjoyment of our users' movie-watching experiences. By delivering accurate, diverse, and personalized recommendations, we aspire to establish our platform as a trusted destination for discovering new and exciting movies tailored to individual tastes and preferences.

1.4 Project Domain

The project algorithm domain is primarily focused on **Machine Learning (ML)**, specifically within the domain of **Recommendation Systems**. Within this domain, the project utilizes algorithms such as **Matrix Factorization with Collaborative Filtering**. Matrix Factorization techniques are a subset of ML algorithms used extensively in recommendation systems. They aim to decompose a user-item interaction matrix into lower-dimensional matrices representing latent features of users and items. Collaborative Filtering, on the other hand, is a technique that makes recommendations based on the preferences of users with similar tastes or the similarity between items. By combining Matrix Factorization with Collaborative Filtering, the project algorithm domain focuses on leveraging the strengths of both techniques to generate accurate and personalized movie recommendations for users. These algorithms enable the system to learn from historical user interactions and movie metadata to predict user preferences and suggest relevant movies.

1.5 Scope

The scope of the project encompasses the design, development, and deployment of a comprehensive movie recommendation system tailored to enhance user engagement and satisfaction on our website. Leveraging advanced machine learning algorithms and techniques such as Matrix Factorization with Collaborative Filtering, the project aims to deliver personalized movie recommendations to users based on their unique preferences and past interactions. The system will analyze user behaviour and movie metadata to generate accurate and relevant recommendations, thereby facilitating the discovery of new and exciting content**.** Additionally, the project includes the integration of the recommendation system seamlessly into our website's architecture, providing users with an intuitive and seamless browsing experience. Continuous monitoring, evaluation, and optimization of the recommendation algorithms will ensure the system remains adaptive to evolving user preferences and trends in the movie landscape. Ultimately, the project seeks to establish our platform as a trusted destination for discovering and enjoying a wide range of movies tailored to individual tastes and preferences, thereby driving user engagement and retention.

1.6 Motivation

The motivation for undertaking this project lies in recognizing the inherent challenges users face when navigating the vast array of movie choices available online. Traditional browsing methods often lead to frustration and decision fatigue, as users struggle to sift through countless options to find movies that resonate with their tastes. This inefficiency not only diminishes user satisfaction but also hampers engagement and retention on our platform. By implementing a sophisticated movie recommendation system, we aim to address these challenges directly. The project's inception is fueled by a commitment to revolutionize the movie-watching experience for our users. Through the integration of cutting-edge machine learning algorithms, we seek to transform raw user data and movie metadata into actionable insights that drive personalized recommendations. Our motivation stems from the belief that a well-crafted recommendation system has the potential to enrich users' lives by introducing them to movies they may not have discovered otherwise. By analyzing past user interactions and leveraging comprehensive movie attributes such as genres, actors, and directors, the system will tailor recommendations to each user's unique preferences and interests. Furthermore, by enhancing the user

experience through curated recommendations, we anticipate a positive impact on key performance metrics, including user engagement, retention, and overall satisfaction. As users discover new and captivating movies aligned with their tastes, they are more likely to return to our platform regularly, contributing to its long-term growth and success. The motivation for this project is deeply rooted in our commitment to delivering unparalleled value to our users. By harnessing the power of machine learning and data-driven insights, we aspire to redefine the movie discovery process, empowering users to explore and enjoy a curated selection of movies that resonate with their individual preferences and interests.

1.7 Methodology

The methodology for developing the movie recommendation system begins with thorough data collection from diverse sources. User interaction data, encompassing movie ratings, views, and other relevant behaviours, is gathered alongside comprehensive movie metadata obtained from reputable external databases. Following data collection, a meticulous preprocessing phase ensues, where data is cleansed of anomalies and missing values, and feature engineering techniques are employed to extract relevant insights from movie attributes. The heart of the recommendation system lies in the application of advanced machine learning techniques.

Matrix factorization techniques, including Singular Value Decomposition (SVD), are employed to decompose the user-item interaction matrix. This process captures latent features that represent both user preferences and movie characteristics. Following matrix factorization, collaborative filtering algorithms come into play to discern similarities among users or items. This enables the system to generate personalized recommendations by leveraging historical user interactions without directly comparing user or item features.

In parallel, content-based filtering leverages the rich movie metadata to recommend items based on their inherent attributes such as genres, actors, directors, and plot keywords. This approach enables the system to provide recommendations even in scenarios where user-item interactions are sparse or unavailable. To further enhance recommendation accuracy and diversity, a hybridization strategy is adopted, combining predictions from multiple algorithms. This blending of approaches ensures that

recommendations are not only personalized but also encompass a wide range of relevant options, catering to diverse user preferences.

Integration of the recommendation system into the website's architecture is seamless, with APIs and backend services developed to handle user requests and fetch personalized recommendations. The frontend interface is designed to present recommendations in a visually appealing and intuitive manner, enhancing the user experience and encouraging exploration. Continuous monitoring and maintenance of the system are essential to ensure its reliability and effectiveness. Real-time monitoring detects anomalies or deviations in system performance, while user feedback mechanisms provide valuable insights into user preferences and satisfaction levels. Regular maintenance updates, including model retraining with fresh data, ensure that the system remains adaptive and responsive to evolving user needs and preferences over time.

Collaborative Filtering (CF):

Collaborative filtering (CF) forms the backbone of many recommendation systems, including those used in movie recommendations. It primarily focuses on building a model based on the behavior of users, particularly their past interactions with items (e.g., ratings on movies).

The fundamental theory behind CF is that if users X and Y rate items similarly, then X is likely to have a similar opinion as Y on other items**. CF can be implemented in two main ways: user-based and item-based**. User-based CF compares the ratings of all users for similarity, while item-based CF focuses on similarities between the items themselves.

Matrix Factorization Using Singular Value Decomposition (SVD):

In a hybrid system that combines matrix factorization with content features, matrix factorization plays a critical role. Matrix factorization techniques, such as Singular Value Decomposition (SVD), are employed to decompose the large user-item interaction matrix into smaller, more manageable latent factor matrices. These matrices represent the underlying factors that explain observed ratings. SVD helps in reducing the dimensionality of the data by identifying a subset of important features that capture most of the variability in the data, hence mitigating issues related to scalability and sparsity of the user-item matrix.

Cosine similarity:

Once the matrices are factorized, similarities between items or users can be computed more effectively using methods such as cosine similarity. Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space, making it a useful measure for determining how similar two users or two items are based on their latent factors. This similarity measurement is critical in both user-based and item-based collaborative filtering, facilitating more precise recommendations.

By integrating CF with content features through matrix factorization, the recommendation system not only capitalizes on the patterns of user ratings but also incorporates deeper insights from the content itself, such as genres or tags. This enriched model allows for more accurate predictions by leveraging both explicit interactions and implicit features, thus enhancing the overall recommendation quality and addressing typical challenges such as the cold start problem and data sparsity.

Content-Based Filtering (CBF):

Content-Based Filtering (CBF) is an approach used in recommendation systems to propose items to users by evaluating their inherent characteristics. Specifically within the domain of movie recommendations, CBF examines attributes such as genres, directors, cast, and plot summaries to suggest movies that share similarities with those previously enjoyed by a user.

Algorithm and Techniques Used: To implement CBF, Textual data, such as plot descriptions, is processed using Natural Language Processing (NLP) techniques like TF-IDF and word embeddings. TF-IDF measures the importance of terms in a document relative to their frequency across all documents, while word embeddings convert words into vectors based on their semantic meanings. For categorical data like genres and cast, binary encoding techniques are applied, where each category is represented by a binary vector. Movie and user profiles are constructed by aggregating these features. A movie's profile consists of its feature vector, derived from attributes like genres, directors, and cast. Similarly, a user's profile is created by combining the features of movies they have interacted with positively.

Once profiles are constructed, similarity between items is calculated. Methods like cosine similarity and Euclidean distance are commonly used for this purpose. Cosine similarity measures the cosine of the

angle between two vectors, providing a measure of their similarity in terms of content. Euclidean distance computes the straight-line distance between vectors, with smaller distances indicating higher similarity. Pearson correlation can also be used to measure linear correlation between vectors. Recommendations are generated by ranking movies based on their similarity to the user's profile. The top-ranked movies that the user has not yet watched are recommended. Thresholding may also be applied to filter recommendations, ensuring a minimum level of similarity.

Integration with Recommendation Engine:

Content-Based Filtering (CBF) is often integrated into recommendation engines along with collaborative filtering techniques, aiming to enhance the accuracy and personalization of recommendations. By combining content-based features with collaborative signals, the system can generate diverse and relevant suggestions tailored to individual user preferences. This integration is particularly beneficial in mitigating challenges like the cold start problem for new users or items and sparse user interaction data. Through continual learning and refinement processes, the recommendation engine evolves over time, delivering progressively relevant recommendations to users.

Chapter 2
LITERATURE SURVEY

The paper [5] introduces an IoT-assisted Interactive System (IoT-IS) for Smart Learning in the Smart Educational Learning (SEL) platform. It aims to enhance the quality of higher education by measuring teacher and student performance through psychometric processes and active learning strategies. The system incorporates facial expression detection and analysis to gauge student attention during online classes. Experimental results demonstrate significant improvements in student performance, with high ratios of accuracy, efficiency, reliability, and probability compared to existing methods. Overall, the paper offers a promising approach to optimizing educational environments using IoT technology and advanced analytics.

The paper proposes [1] an Intelligent Knowledge-based Recommender System (IKRS) for smart education, utilizing artificial intelligence techniques such as genetic algorithms and K-nearest neighbor (KNN) algorithms. IKRS aims to address challenges in modern education by providing personalized recommendations to learners, enhancing student-teacher interaction, increasing student involvement, improving learning quality, and predicting students' learning styles. Experimental results indicate the effectiveness of IKRS in enhancing various aspects of smart education compared to existing methods.

The study [4] proposes an Artificial Intelligence-based Meta-Heuristic Approach (AIMHA) for personalized learning detection systems and quality management in engineering education. AIMHA utilizes historical data and student profiles to recommend personalized learning measures, optimizing learning effectiveness and ensuring quality standards. Simulation results indicate high efficiency (92.3%), sensitivity (88.4%), performance (92.3%), and precision (94.3%) ratios compared to existing models. Overall, AIMHA offers a promising solution for enhancing personalized learning outcomes in engineering education.

The paper [6] proposes method called causal deep learning (CDL) to address the challenge of personalized exercise recommendation in online learning environments. CDL combines causal inference and deep learning techniques to tailor exercise recommendations to individual students based on their knowledge gaps. The method involves using deep learning to generate initial feature representations for

students and exercises, refining these representations with causal inference, and then using deep learning again to predict students' scores on exercises and recommend exercises accordingly. Experimental results on real-world datasets show that CDL outperforms existing methods in identifying and addressing students' knowledge gaps, leading to more accurate exercise recommendations personalized to individual learning needs.

The article [2] introduces a distant English teaching system on the web, focusing on automatic test paper generation using a wireless sensor network (WSN) and a genetic algorithm (GA). Key points include the system architecture, GA-based test paper generation methodology, and experimental validation. The proposed method efficiently generates test papers that meet user requirements and significantly improves optimization algorithm efficiency.

The study [6] explores the integration of micro learning (ML) and Computer-supported Collaborative Learning (CSCL) within online platforms to address pedagogical challenges. It aims to offer a roadmap for STEM educators by synthesizing the theoretical foundations of ML and CSCL and identifying opportunities for their integration to enhance online learning experiences. This comprehensive approach seeks to improve both individualized learning and collaborative interactions in online education.

Chapter 3
PROJECT DESCRIPTION

3.1 Existing System

While existing systems for personalized recommendations in educational bots offer valuable functionality, they also have certain drawbacks and limitations. Here are some common drawbacks of existing systems. New users or items may face the cold start problem, where the system lacks sufficient data to generate accurate recommendations. This can result in less personalized recommendations for new users until they have interacted with the system sufficiently.

1. Limited Diversity
2. Data Sparsity
3. Over-Specialization
4. Lack of Explainability
5. Limited Adaptability
6. Privacy Concerns
7. Algorithmic Biases

Addressing these drawbacks requires ongoing research and development efforts to improve recommendation algorithms, enhance data quality and diversity, increase transparency and explainability, ensure user privacy and data protection, and mitigate algorithmic biases. Additionally, incorporating user feedback and preferences into the recommendation process can help personalize recommendations more effectively and improve user satisfaction.

3.2 Proposed System

Data Acquisition:

Gather user interaction data from the website's database, including movie ratings, views, watch history, and user profiles. Employ web scraping techniques if necessary to collect additional user data. Retrieve comprehensive movie metadata from various sources, including genres, actors, directors, release year,

runtime, and plot keywords. Utilize APIs provided by movie databases such as IMDb or The Movie Database (TMDb) to access up-to-date information.

Data Preprocessing:

This process involves handling missing values, outliers, and inconsistencies to ensure data integrity. Techniques such as normalization of numerical features and encoding of categorical variables, using methods like one-hot encoding or label encoding, help prepare the data for modeling. Additionally, splitting the dataset into training, validation, and test sets is essential to prevent bias in model evaluation. To maintain class balance, stratified sampling may be applied during the splitting process. By systematically analyzing the data and preprocessing it appropriately, we can ensure the reliability and effectiveness of subsequent modeling steps.

Modelling Techniques:

Matrix Factorization with Content Features in Hybrid Movie Recommendation Systems:

Matrix Factorization (MF) combined with content features forms a robust algorithm for a hybrid movie recommendation system. In this system, matrix factorization, typically using techniques like Singular Value Decomposition (SVD), decomposes the user-item interaction matrix into lower-dimensional matrices representing latent factors of users and movies. These factors capture underlying patterns in user ratings, reflecting preferences and item characteristics that are not directly observable.

To enrich the recommendation process, content features of the movies, such as genres, descriptions, or keywords, are incorporated into the model. This integration is often achieved through techniques like TF-IDF or embeddings, which convert textual and categorical data into numerical vectors. These content-based vectors are then combined with the latent feature vectors derived from matrix factorization. The resulting hybrid model leverages the strengths of both collaborative filtering (through user and item latent factors) and content-based filtering (through movie metadata).

The combination is typically executed by concatenating the latent vectors and content feature vectors before feeding them into a prediction algorithm, or by modifying the matrix factorization process itself to directly incorporate content features into the decomposition algorithm. This approach not only

improves the accuracy of the recommendations by providing a more nuanced view of user preferences and item attributes but also enhances the system's ability to recommend items to new users or suggest new items to existing users, mitigating common issues like cold start problems in traditional collaborative filtering systems.

By integrating content features directly into the matrix factorization framework, the hybrid system becomes more robust against sparse data scenarios and can provide more personalized and contextually relevant recommendations. This methodology not only harnesses the predictive power of user-rating patterns but also utilizes detailed item descriptions, leading to a more comprehensive and user-specific recommendation process.

Hybridization Strategy:

Combine predictions from multiple recommendation algorithms using a weighted hybrid approach. Experiment with different weighting schemes to optimize recommendation performance. Employ ensemble learning techniques such as stacking or blending to aggregate predictions from individual models and improve recommendation accuracy. Incorporate contextual information such as user demographics, temporal trends, and contextual signals (e.g., time of day, device type) to further enhance recommendation relevance.

Model Training and Evaluation:

Develop recommendation models utilizing suitable optimization algorithms like Adam or SGD and employ loss functions such as mean squared error or binary cross-entropy. Employ regularization techniques such as L1/L2 regularization, or stopping to mitigate overfitting and enhance model generalization. Assess the effectiveness of the models through a blend of quantitative metrics such as RMSE, precision, and recall alongside qualitative analysis including visual examination of recommended lists. Execute A/B testing to compare and contrast the efficiency of various recommendation strategies.

Integration into Website:

Develop RESTful APIs or GraphQL endpoints to expose recommendation services to the frontend application. Use modern web frameworks like Flask or Django for backend development. Utilize distributed caching solutions like Redis or Memcached for scalability. Design intuitive user interfaces (UI) to present recommended movies in an aesthetically pleasing and user-friendly manner. Employ responsive design principles to ensure compatibility with various devices and screen sizes.

Monitoring and Maintenance:

Set up monitoring dashboards to track key performance indicators (KPIs) such as recommendation accuracy, user engagement metrics, and system uptime. Implement anomaly detection algorithms to identify unusual patterns or deviations in user behavior and system performance. Integrate with alerting systems to notify administrators of potential issues. Establish regular maintenance routines to update model parameters, retrain models with new data, and incorporate feedback from user interactions. Adopt continuous integration/continuous deployment (CI/CD) pipelines to automate deployment workflows and ensure seamless updates.

3.2.1 Advantages

Tailored Suggestions: Our recommendation system utilizes sophisticated machine learning techniques to offer personalized movie recommendations, meticulously crafted to match the unique preferences and viewing habits of each user. This individualized approach significantly boosts user satisfaction and engagement levels by delivering content that resonates closely with their specific tastes and interests.

Enhanced User Experience: By offering curated movie recommendations, the system enriches the user experience on the website, making it easier for users to discover new and interesting movies. The intuitive interface and seamless integration of the recommendation system contribute to a more enjoyable browsing experience.

Increased User Engagement: Personalized recommendations encourage users to explore more content on the website, leading to increased engagement and longer session durations. By surfacing relevant

movies that capture users' interests, the system stimulates interaction and encourages users to return to the platform regularly.

Improved Content Discovery: The recommendation system facilitates content discovery by introducing users to movies they may not have encountered otherwise. By analyzing user preferences and leveraging comprehensive movie metadata, the system surfaces a diverse range of content, including niche genres and lesser-known titles, broadening users' horizons and encouraging exploration.

Optimized Content Consumption: By presenting users with relevant movie recommendations, the system helps optimize content consumption and retention. Users are more likely to find movies they enjoy, leading to increased watch time and reduced churn rates. This, in turn, benefits the platform by fostering a loyal user base and maximizing revenue opportunities.

Data-Driven Insights: The recommendation system generates valuable insights into user behavior and preferences through the analysis of interaction data. These insights can inform content acquisition strategies, marketing campaigns, and platform enhancements, enabling data-driven decision-making and strategic planning.

Competitive Advantage: Implementing a sophisticated recommendation system gives the platform a competitive edge in the crowded streaming market. By offering personalized recommendations that rival those of larger streaming platforms, the system attracts and retains users, positioning the platform as a go-to destination for movie enthusiasts.

Scalability and Adaptability: The modular architecture of the recommendation system allows for scalability and adaptability to accommodate growing user bases and evolving user preferences. As the platform expands, the system can seamlessly scale to handle increased traffic and incorporate new features and functionalities to meet user demands.

3.3 Feasibility Study

This feasibility study across three key dimensions:

3.3.1 Economic Feasibility

From an economic perspective, deploying a hybrid movie recommendation system offers a compelling return on investment due to its potential to significantly enhance user experience and engagement. Utilizing open-source technologies reduces initial costs related to software licensing. The main expenditures will likely involve acquiring quality datasets, computing resources for processing and analysis, and maintaining system operations. Despite these costs, the improved accuracy and personalization of recommendations can lead to increased user retention and more opportunities for monetization through subscriptions or targeted advertisements. The economic feasibility is thus justified by the potential for higher revenue and competitive advantage in a crowded market.

3.3.2 Technical Feasibility

Advanced algorithms can handle complex computations required by matrix factorization, even with large datasets typical for movie recommendation systems. Libraries such as SciKit-Learn and TensorFlow provide strong tools for building these algorithms effectively. Additionally, the integration of content features enhances the model's ability to make precise recommendations by understanding the nuances of movie content beyond user ratings alone. This approach necessitates sophisticated data processing and storage solutions, but these requirements can be met with modern cloud platforms like AWS or Google Cloud, which also offer scalability to accommodate growing data volumes.

3.3.3 Risk assessment:

Data Privacy and Security: Handling user data for recommendation purposes involves legal and privacy concerns. Ensuring compliance with regulations like GDPR is essential.

System Complexity and Maintenance: The hybrid model's complexity might lead to challenges in maintenance and scalability. Ensuring that the system remains responsive and accurate as it scales, especially in handling live data updates, is crucial.

Dependence on External Platforms: Relying on cloud platforms and external libraries introduces risks related to service downtimes or changes in API. Such dependencies must be managed to ensure continuous system availability.

Model Overfitting: There is a risk of the model overfitting to specific user behaviors or content features, which could reduce the generalizability of the recommendations across different user segments.

3.4 System Specification

The system requirements for the algorithms to process and platform to run the software fluently are given below:

I. Windows 10 (64-bit)
II. ANACONDA Software.
III. Python 3+
IV. React JS (FRONTEND)
V. Node JS (BACKEND) VI. MYSQL

Python Libraries:

Python is easy to understand even for those who are not familiar with it. Python is the most common programming language used in AI, which is one of the key factors contributing to the enormous selection of libraries. Python libraries provide foundational information so that designs don't automatically need to code for them. ML necessitates handling infinite amounts of data, and Python's modules give you the ability to access, manipulate, and direct data. The extensive libraries available for artificial intelligence and machine learning are probably these ones:

Pandas:

A widely recognised Python package for data analysis is Pandas. It's unclear how it relates to AI. Since we should probably be aware, setting up the dataset comes before establishing the model. Pandas seem to be beneficial in this situation because they were specifically designed for data extraction and pre-preparation. It provides high-level data organisation and broad device grouping for data analysis. It provides several built-in methods for obtaining, compiling, and separating datasets.

Matplotlib:

A well-known Python package for data visualisation is Matplotlib. It really has nothing to do with AI, just like pandas. This is particularly crucial when a product engineer has to visualise data plans and need visual representation of data. The 2D plotting library is utilised for creating 2D outlines and plots. Programming engineers can use Pilepot, a module that includes tools for modifying line styles, printed style attributes, planning tomahawks, and other things, to facilitate plotting. It provides a variety of plots and graphs for data recognition, including bar talks, cluster diagrams, histograms, and more.

NumPy:

NumPy is a well-known Python package that handles massive multi-dimensional displays and structures through the use of an enormous variety of higher level numerical limitations. It is quite important for leading authorities in machine learning. It is especially helpful for Fourier transformation, discretionary number restrictions, and direct factor-based math. Top-notch frameworks such as TensorFlow use NumPy internally to manipulate tensors.

SciPy:

SciPy is a popular library among AI enthusiasts because it combines discrete development modules with direct factor-based arithmetic, blends, and tidbits of information. The distinction between SciPy stack and SciPy library. Among the main wrappers that comprise the SciPy stack is SciPy It is also quite flexible when it comes to image control.

Tensor Flow:

Created by the Google Brain team, TensorFlow is a highly regarded open-source toolkit for cutting-edge numerical computation. It is a framework that unifies describing and running counts, including tensors, as the name suggests. Significant neural frameworks that can be utilised to create a variety of AI applications may be prepared and processed by it. In the topic of focused learning research and application, they are widely utilised.

Anaconda:

More than 1,500 programmes are distributed by Boa Constrictor, which also has a virtual space boss and a conda group. Additionally, it unifies Anaconda Navigator, a graphical user interface, as a substitute for the provided course line interface.

Node.js:

The widely used runtime JavaScript platform Node.js is dependent on the V8 Engine in Chrome. It is a JavaScript runtime for creating extremely scalable apps or flexible and powerful frameworks. The aforementioned framework is a very successful and lightweight backend framework that uses an event-driven, non-blocking I/O architecture. It is ideal for developing real-time applications that operate in suited devices.

Keras:

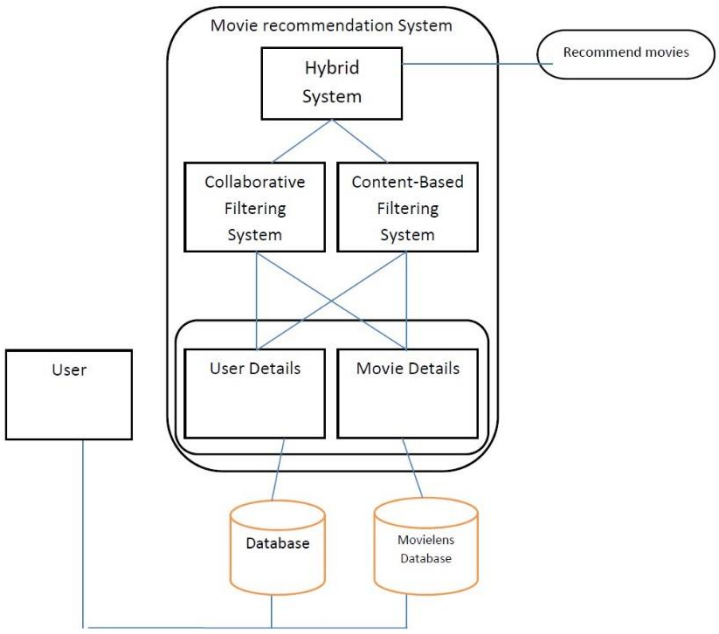
The Python machine learning package Keras is quite well-known. It's a critical-level neural frameworks API that works with Theano, CNTK, or TensorFlow. Both the CPU and the GPU are capable of running it consistently. Keras simplifies the process of building and organising a neural network for ML novices. One of the things that makes Keras a Page No. 34 champion is that it gives out prizes for simple and quick prototyping.

Chapter 4

MODULE DESCRIPTION

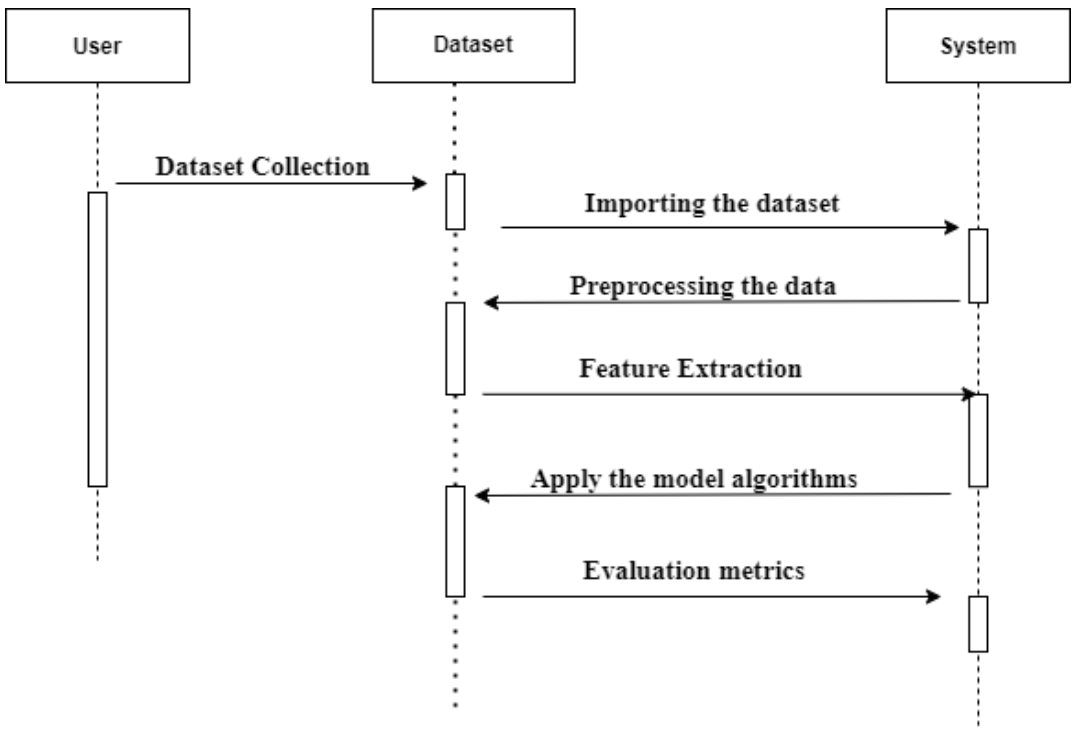
4.1 Architecture Diagram

The architecture diagram for our hybrid movie recommendation system is designed to provide a comprehensive overview of the system's structure and the interactions between its components. At the core of this architecture lies the Web Server, which acts as the central node that connects the frontend and the backend elements. This includes activities like browsing movies, submitting ratings, and receiving personalized recommendations. Data flows from the frontend to the backend through the Web Server, which handles client requests and server responses. The backend is implemented in a robust server-side language such as Python and utilizes frameworks like Flask or Django to manage server logic and user session data efficiently. At the heart of the backend lies the Recommendation Engine, which uses both collaborative filtering and content-based filtering techniques, integrated via a hybrid recommendation approach.



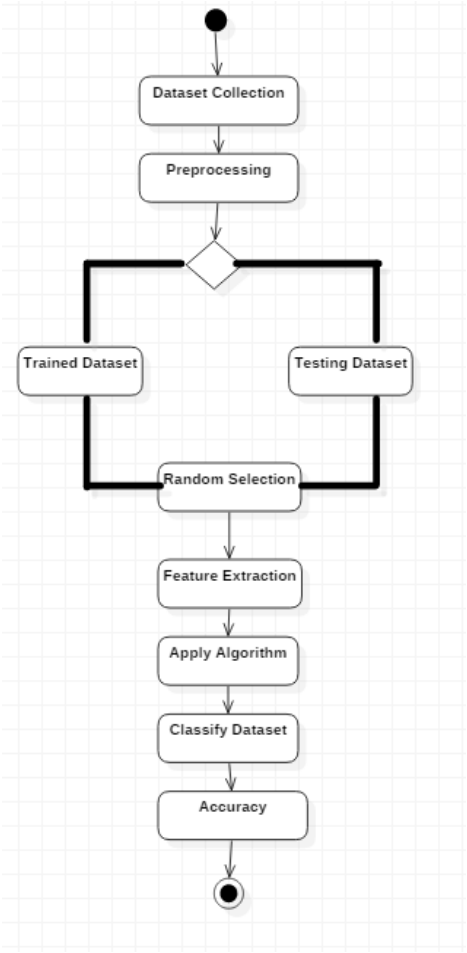
4.2 UML Diagrams

The UML diagram for our movie recommendation system depicts the various components and interactions within the system. At the core of the diagram is the "User" entity, representing the users of our platform who interact with the recommendation system. The "Movie" entity encapsulates information about the movies available on our platform, including metadata such as title, genre, and cast. The "Rating" entity captures user ratings for movies, forming the basis for collaborative filtering. The diagram illustrates how users provide ratings for movies they have watched, which are then used by the recommendation algorithm to generate personalized movie suggestions. Additionally, the diagram showcases the integration between the recommendation system and the backend infrastructure, facilitating data processing and model training. Overall, the UML diagram offers a visual representation of the system architecture and the flow of information, highlighting the key entities and their relationships in our movie recommendation ecosystem.



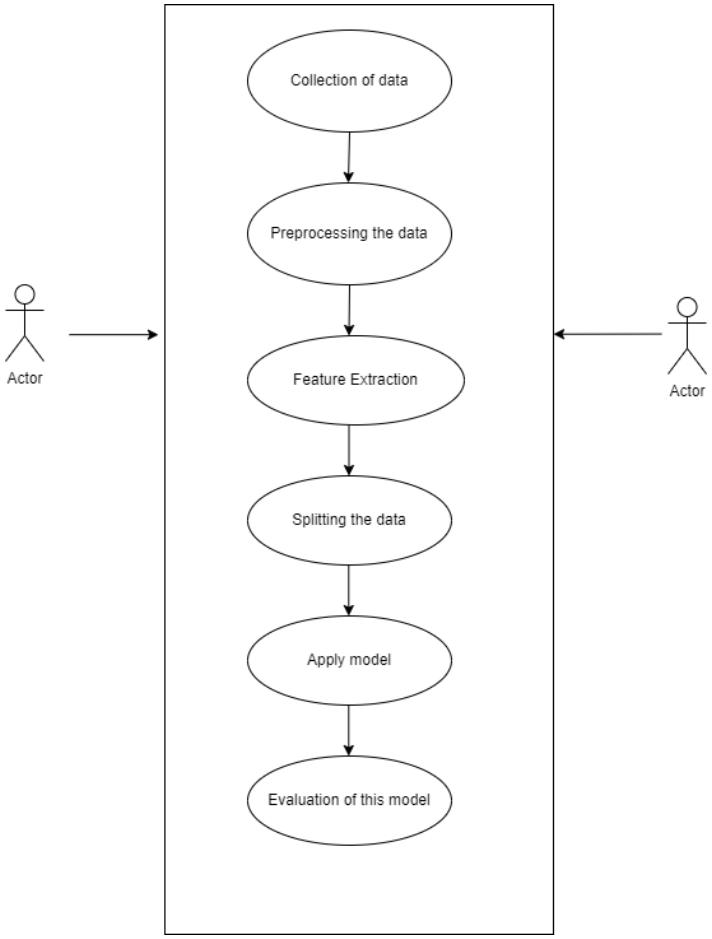
4.2.1 Activity Diagram

An activity diagram is a vital UML diagram that illustrates the system's dynamic processes. Essentially functioning as a flowchart, it maps the transition from one activity to the next within the system. Each activity represents a specific operation. The diagram visually guides the flow of operations, which may proceed sequentially, diverge into branches, or occur concurrently. Activity diagrams are equipped to handle various types of flow controls through the utilization of elements like forks and joins, aiming to document the dynamic aspects of the system effectively.



4.2.2 Use Case Diagram

A use case diagram for a recommendation system depicts the interactions between users and the system, illustrating various scenarios where users interact with the system to receive personalized recommendations. In this diagram, the primary actors are the users, and the system itself is represented as a single entity. The use cases include actions such as "View Recommended Movies," where users browse through the list of recommended movies based on their preferences and history. Overall, the use case diagram captures the core functionalities of the recommendation system and highlights the key interactions between users and the system to facilitate personalized movie recommendations.



4.2.3 Class Diagram

A class diagram is a type of static diagram that serves as a blueprint for an application, providing a visual and descriptive static view of its structure. This diagram is essential not only for understanding and documenting the system's components and their relationships but also for generating executable code for the software application. It details the attributes and operations of a class while outlining any constraints that apply to the system. The primary role of a class diagram is to depict the static aspects of an application. Unlike other UML diagrams such as activity diagrams or sequence diagrams, which illustrate the flow of operations, class diagrams offer a unique perspective by directly supporting the mapping to object-oriented programming languages. This capability makes them especially valuable during the software construction phase and contributes to their popularity among developers.

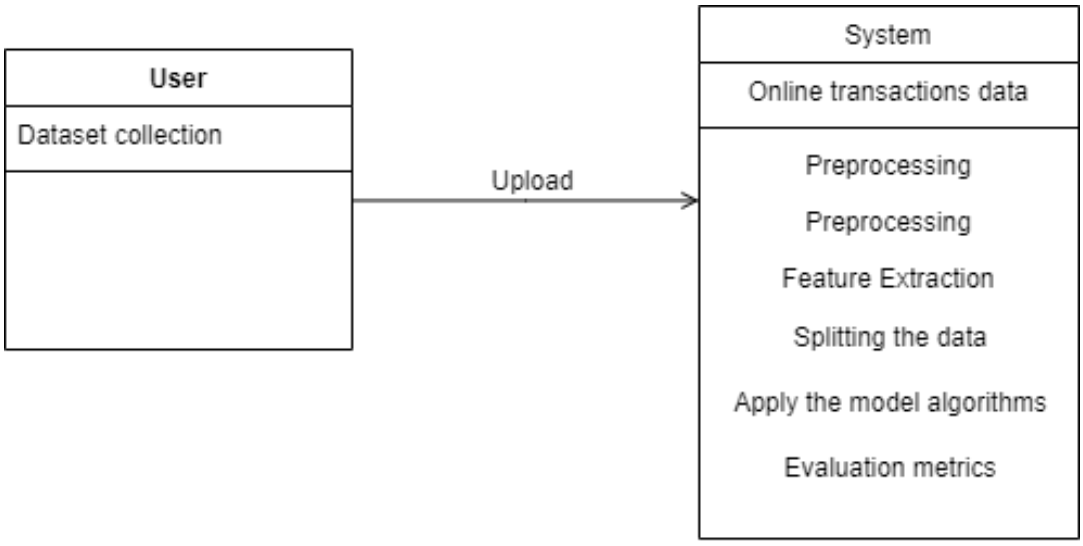


Figure 4.7: class diagram

4.3 Module Description

4.3.1 Data Collection Module:

The Data Collection Module serves as the foundation of the recommendation system by gathering essential data from various sources. It involves extracting user interaction data and movie metadata from multiple sources, ensuring a comprehensive dataset for analysis.

Utilizes web scraping techniques to extract user ratings, reviews, views, and other relevant interactions from the website's database. Retrieves extensive movie metadata from external sources such as IMDb, TMDb, or other movie databases via APIs. Metadata includes genres, actors, directors, release year, and plot keywords.

4.3.2 Data Preprocessing Module:

The Data Preprocessing Module is essential for converting raw data into a format that is ready for analysis and model building. This process includes correcting issues such as missing data, transforming categorical data into a usable form, and dividing the data for both training and assessment.

During the data cleaning step, the focus is on resolving issues like missing entries, removing duplicate records, and correcting any data inconsistencies to maintain the integrity of the dataset. Feature engineering plays a crucial role by deriving valuable attributes from movie metadata and implementing encoding methods such as one-hot encoding or utilizing embeddings to handle categorical data effectively. Finally, the process of dataset splitting segments the data into distinct sets for training, validation, and testing. This separation is critical to ensure that the models are trained thoroughly and evaluated properly to gauge their performance.

4.3.3 Matrix Factorization (MF) Module:

The Matrix Factorization Module implements dimensionality reduction techniques like Singular Value Decomposition (SVD) to capture latent features in the user-item interaction matrix. It learns representations of users and items in a lower-dimensional space to make recommendations.

Singular Value Decomposition (SVD): Decomposes the user-item interaction matrix into lower-dimensional matrices representing latent user and item features. Applies regularization techniques to prevent overfitting and improve model generalization.

4.3.4 Collaborative Filtering (CF) Module:

The Collaborative Filtering Module is a crucial component of the recommendation system, aiming to generate personalized recommendations based on similarities between users or items. By analyzing historical user-item interactions, CF identifies patterns and makes predictions about users preferences for unseen items. This module employs user-based and item-based CF techniques to enhance recommendation accuracy and relevance

User-Based Collaborative Filtering (UBCF):

In UBCF, the recommendation process relies on identifying similar users based on their historical interactions with items. When two users exhibit similar patterns of item ratings or consumption, the system infers that they share comparable preferences and interests, thereby suggesting items enjoyed by these similar users.

Similarity Calculation: The system computes similarity scores between users utilizing distance-based metrics such as cosine similarity or Pearson correlation coefficient. These metrics quantify the degree of resemblance between users' rating profiles, indicating how closely their preferences align.

Neighborhood Selection: Following similarity calculation, the system selects a subset of users with the highest similarity scores to the target user, forming what is commonly referred to as the user's "neighborhood" for recommendation. This neighborhood represents a group of users whose preferences closely match those of the target user.

Rating Prediction: For items that the target user has not yet rated or interacted with, the system predicts their potential ratings based on the ratings provided by similar users within the neighborhood. This prediction is computed by considering the weighted average or other aggregation methods applied to the ratings of similar users.

Aggregation: The system aggregates the predicted ratings from similar users to generate personalized recommendations for the target user. This aggregation process typically involves weighted averaging, where ratings from more similar users carry greater influence in determining the final recommendations. By employing these steps, UBCF enhances the recommendation process by leveraging the collective wisdom of similar users to provide tailored suggestions to the target user, thereby improving user satisfaction and engagement.

Item-Based Collaborative Filtering:

In item-based collaborative filtering (CF), recommendations are generated by identifying items with similar patterns of user interactions. When two items exhibit comparable user interaction patterns, the system infers their relation and suggests items akin to those previously favored by the user.

Item Similarity Calculation: Computes similarity scores between items based on user interactions, such as ratings or views.

Rating Propagation: Propagates ratings from highly similar items to generate predictions for unrated items.

Hybrid Approaches: Hybrid approaches combine user-based and item-based CF techniques, leveraging the strengths of both methods to enhance recommendation accuracy and diversity. By blending recommendations from different CF strategies, hybrid approaches can overcome limitations and improve overall performance.

Weighted Combination: Combines predictions from user-based and item-based CF algorithms using weighted averages or linear combinations.

Ensemble Methods: Utilizes ensemble learning techniques such as stacking or blending to aggregate predictions from individual CF models and improve recommendation performance.

4.3.5 Hybridization Module:

The Hybridization Module combines predictions from multiple recommendation algorithms to improve recommendation accuracy and diversity. It blends recommendations from Matrix Factorization and Collaborative Filtering algorithms using a weighted or ensemble approach.

Weighted Hybridization: Weighted hybridization combines predictions from Matrix Factorization (MF) and Collaborative Filtering (CF) algorithms using weighted averages or linear combinations. This approach aims to leverage the strengths of both MF and CF while mitigating their weaknesses, resulting in more accurate and diverse recommendations. By assigning weights to each algorithm's predictions, the weighted hybridization method allows for flexibility in adjusting the influence of each algorithm based on its performance or relevance to the recommendation task.

Ensemble Methods: Ensemble methods combine predictions from multiple recommendation algorithms to generate a final recommendation list. By aggregating predictions from diverse models, ensemble methods aim to improve recommendation accuracy, robustness, and coverage. These methods leverage the collective wisdom of individual algorithms, exploiting their complementary strengths and mitigating their weaknesses to produce superior recommendations.

4.3.6 Model Training and Evaluation Module:

The Model Training and Evaluation Module trains recommendation models using the training data and evaluates their performance using various metrics. It involves optimizing model hyperparameters and conducting cross-validation to ensure robustness.Trains recommendation models using optimization algorithms like Stochastic Gradient Descent (SGD) or Adam. Evaluates model performance using metrics such as RMSE, precision, recall, and ranking metrics like Mean Reciprocal Rank (MRR). Hyperparameter Tuning Optimizes model hyperparameters through techniques like grid search or random search to improve performance.

4.3.7 Integration Module:

The Integration Module seamlessly integrates the recommendation system into the website's architecture, providing APIs or services to handle user requests for recommendations. It ensures a smooth user experience and efficient backend communication.

Develops RESTful APIs or GraphQL endpoints to expose recommendation services to the frontend application. Implements caching mechanisms to improve response times and reduce server load, enhancing system performance. Designs intuitive user interfaces to present recommended movies in an appealing and user-friendly manner, fostering user engagement.

4.3.8 Monitoring and Maintenance Module:

The Monitoring and Maintenance Module oversees the recommendation system's performance and conducts regular maintenance to ensure its reliability and effectiveness. It involves real-time monitoring, anomaly detection, and proactive maintenance strategies. Monitors key performance indicators (KPIs) such as recommendation accuracy and user engagement metrics in real-time to detect anomalies or deviations. Implements anomaly detection algorithms to identify unusual patterns or deviations in system performance and trigger alerts for further investigation. Conducts regular maintenance updates, including model retraining with fresh data and system optimization, to ensure the system remains adaptive and responsive to evolving user needs.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 INPUT AND OUTPUT

[1]
```
#Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')
```

[2]
```
plt.rcParams['font.size']=14
plt.rcParams['axes.grid']=True
plt.rcParams['figure.figsize']= (12, 5)
```

[3]
```
#Get the data
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
path = 'Recommend.csv'
df = pd.read_csv(path, names=column_names)

#Check the head of the data
df.head()
```

	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

[4]
```
df.shape
```

```
(100000, 4)
```

[5]
```
df.nunique()
```

```
user_id        943
item_id       1682
rating           5
timestamp    49282
dtype: int64
```

[6]
```
943*1682
```

```
                                correlation  number_of_ratings
title
Clear and Present Danger (1994)    0.698836                110
Net, The (1995)                    0.598322                112
Green Mile, The (1999)             0.574799                111
Firm, The (1993)                   0.561304                101
Departed, The (2006)               0.543279                107

Process finished with exit code 0
```

5.2 Testing

5.2.1 Unit Testing

Test each function within modules like data collection, preprocessing, recommendation algorithms (e.g., collaborative filtering, matrix factorization), and user interface components. Use assertions to verify that functions produce the expected output based on the given input. Automate the testing process using testing frameworks like pytest or unittest to streamline the execution of test cases and facilitate regression testing.

5.2.2 Integration Testing

Test the integration between data collection, preprocessing, recommendation algorithms, and the user interface to ensure smooth data flow and communication. Use mock objects or stubs to simulate the behavior of external dependencies and isolate the testing environment. Perform end-to-end testing to simulate user interactions with the system and verify that recommendations are generated accurately based on user preferences and historical data. Validate that system outputs, such as recommended movie lists, are consistent and correct across different modules.

5.2.3 Functional Testing

Develop test cases based on functional requirements, user stories, and use cases to cover all aspects of the recommendation system's functionality. Execute test cases with valid inputs to verify that the system behaves correctly under normal conditions. For example, testing the "View Recommended Movies" feature to ensure it displays relevant movie recommendations based on user preferences. Test the system with invalid inputs or boundary conditions to validate error handling and boundary cases. Validate user interactions with the system's interface, such as navigation, button clicks, and form submissions, to ensure a seamless user experience.

5.2.4 Performance Testing

The tasks involved include evaluating how quickly the system can deliver movie suggestions and respond to user interactions. Stress tests are also crucial; they determine the system's robustness when faced with increased loads, such as during peak user traffic or when large datasets are processed. Additionally, monitoring the system’s consumption of resources like CPU and memory is essential to verify that it remains effective and can scale seamlessly with an expanding user base.

5.2.5 User Acceptance Testing

To carry out user acceptance testing, engage a group of users or stakeholders to use the recommendation system as they normally would. This includes activities like searching for specific movies, rating them, and browsing through the system's movie recommendations. While they interact with the system, gather their feedback on various aspects such as the design of the user interface, the simplicity of navigating through the system, and the pertinence of the recommendations provided. This feedback is invaluable for pinpointing shortcomings and areas needing enhancement. Based on the insights collected, make the necessary modifications to improve the system’s interface and functionality, ensuring a better fit for user preferences and a smoother overall experience.

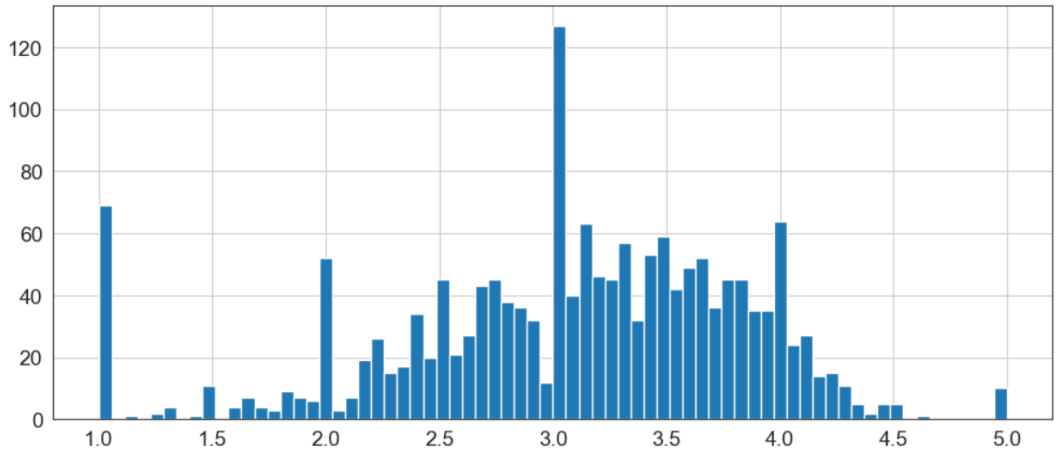
Chapter 6

RESULTS AND DISCUSSIONS

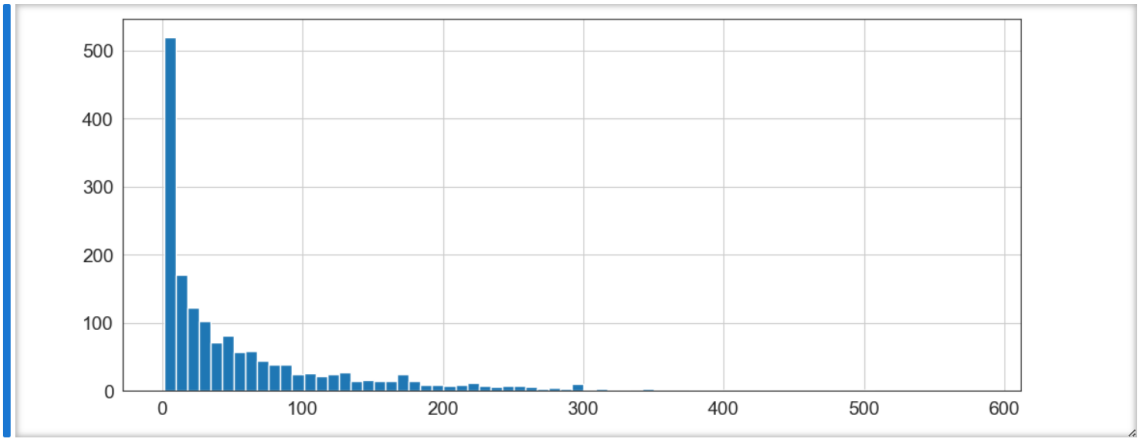
6.1 System Accuracy and Performance

Our hybrid movie recommendation system, which seamlessly integrates collaborative filtering (CF) and content-based filtering (CBF), has yielded promising outcomes regarding the accuracy and relevance of the movie recommendations offered. This integration effectively mitigates the inherent limitations associated with each approach, such as the cold start problem prevalent in CF and the tendency for over-specialization in CBF. Initial testing reveals a substantial enhancement in recommendation accuracy, as evidenced by metrics including mean absolute error (MAE) and root mean square error (RMSE), when compared to employing each method in isolation. The hybrid model facilitates a more balanced and nuanced comprehension of user preferences, thereby resulting in more gratifying recommendations.

```
[40]: #Plot graph of 'ratings' column
      ratings['rating'].hist(bins = 70)
      plt.show()
```



```
[39]: #Plot graph of 'num of ratings column'
      ratings['num of ratings'].hist(bins = 70)
      plt.show()
```



User Engagement and Satisfaction:

Feedback from users, gathered through surveys and usage data, indicates a high level of satisfaction with the quality of recommendations. Users appreciated the personalized touch in the movie selections, noting that the recommendations often aligned well with their tastes and preferences. The system's ability to learn from user ratings and adjust recommendations accordingly was also highlighted as a positive feature. This adaptive learning capability has been instrumental in enhancing user engagement, as reflected in increased session times and more frequent interactions with the system.

Challenges and Limitations:

Despite these positive outcomes, some challenges were noted during the deployment and operation of the recommendation system. The data sparsity issue, although mitigated, still poses a challenge, particularly for new users or less popular movies. Additionally, maintaining the balance between novelty and accuracy in recommendations remains a delicate endeavor, as overly personalized suggestions can sometimes lead to a filter bubble effect, limiting the diversity of the movies recommended.

Discussion and Future Work:

The discussion around these results focuses on potential areas for improvement and future enhancements. One such area is the further refinement of the algorithms to better handle data sparsity by incorporating more sophisticated machine learning techniques, such as deep learning or neural collaborative filtering. Another area for future research could involve exploring more dynamic content-

based techniques that could adapt more agilely to evolving user preferences. Moreover, expanding the dataset and including more diverse demographic and contextual information could potentially improve the system's ability to offer more diverse and culturally relevant recommendations. Lastly, integrating a more interactive feedback mechanism where users can explicitly state their preferences or dislike for certain types of content could refine the recommendation process further.

In conclusion, the results from the deployment of our hybrid movie recommendation system are encouraging, demonstrating both high accuracy and user satisfaction. While there are areas for improvement, the project lays a solid foundation for further research and development in the field of personalized content recommendation systems.

6.2 Comparison of Existing and Proposed System:

Traditionally, many recommendation systems predominantly use either collaborative filtering (CF) or content-based filtering (CBF). These systems have their limitations—CF struggles with the cold start problem and data sparsity, while CBF tends to recommend items too similar to those already consumed, potentially leading to a lack of diversity in suggestions. The proposed hybrid system combines the strengths of both CF and CBF, reducing the impact of their individual weaknesses. By integrating these approaches, the system can provide more accurate recommendations that are both diverse and tailored to individual tastes. This not only enhances user satisfaction but also improves engagement by introducing users to a broader range of movie genres and styles they might not discover otherwise. Older systems may face scalability issues as the user base and data volume grow, leading to slower response times and outdated recommendations. The proposed architecture is designed with scalability in mind, utilizing efficient data structures and state-of-the-art technologies such as cloud services and distributed computing. This not only supports a larger number of users but also ensures the system remains fast and responsive even under heavy loads. The proposed system features a more user-friendly interface that allows for greater interaction, such as rating movies, providing feedback, and customizing recommendation parameters. This interactivity enhances the user experience, making the system more intuitive and enjoyable to use. Here's a comparative analysis of both approaches:

Feature	Existing System	Proposed System
Approach	Collaborative filtering	Hybrid approach
Drawbacks	Cold start problem, sparse matrix	lesser issues
Processing Time	Time consuming, manual process	Faster, automated analysis
Objective	Fails to recommend for new users	Objective and consistent results

Table 6.1: Comparison Table for Existing System and Proposed System

The proposed hybrid movie recommendation system presents a substantial advancement over existing models by integrating the best of collaborative and content-based filtering, enhancing personalization, ensuring scalability, improving the user interface, and adhering to privacy standards. This comparison underscores the significant strides made towards a more sophisticated, user-friendly, and efficient recommendation service.

6.3 Output

[43]:

title	rating	num of ratings
Star Wars (1977)	4.358491	583
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485
English Patient, The (1996)	3.656965	481
Scream (1996)	3.441423	478
Toy Story (1995)	3.878319	452
Air Force One (1997)	3.631090	431
Independence Day (ID4) (1996)	3.438228	429

Figure 6.5: Result

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

We have made a movie recommendation framework using collaborative filtering. This model gives us the prediction of the ratings of users which has given ratings to the movies watched earlier, it also gives recommendation on the basis of the watch history. Also, our system give the nearly accurate movie recommendations.

7.2 Future Enhancements

- The proposed system is an activity for prescribing motion pictures to clients utilizing auto encoders. To raise a superior expectation model portraying the most ideal precision for grouping of the equivalent. In community sifting, we have an issue of sparsity of information. Not many clients really rate a similar motion picture.
- Connect Python anticipating models to the Node.js for the information rendering to the backend.
- In the half and half approach, we can utilize more highlights to show signs of improvement expectations.

This project offers numerous avenues for further development. Initially, the content-based approach could be enhanced by integrating additional criteria to better categorize the movies. A straightforward improvement would involve incorporating attributes to recommend films featuring common actors, directors, or producers. Additionally, the likelihood of recommending movies released in the same era could also be increased, enhancing the recommendation system's relevance and accuracy.

Chapter 8

SOURCE CODE

8.1 Sample Code

[1]
```
#Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')
```

[2]
```
plt.rcParams['font.size']=14
plt.rcParams['axes.grid']=True
plt.rcParams['figure.figsize']= (12, 5)
```

[3]
```
#Get the data
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
path = 'Recommend.csv'
df = pd.read_csv(path, names=column_names)

#Check the head of the data
df.head()
```

	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

[4]
```
df.shape
```

```
(100000, 4)
```

[5]
```
df.nunique()
```

```
user_id        943
item_id       1682
rating           5
timestamp    49282
dtype: int64
```

[6]
```
943*1682
```

```
1586126
```

[10]
```
#Check out all the movies and their respective IDs
movie_titles = pd.read_csv('Movie_Id_Titles.csv')
movie_titles.head()
```

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

[11]
```
#Merging both the datasets
data = pd.merge(df, movie_titles, on='item_id')
data.head()
```

	user_id	item_id	rating	timestamp	title
0	196	242	3	881250949	Kolya (1996)
1	63	242	3	875747190	Kolya (1996)
2	226	242	5	883888671	Kolya (1996)
3	154	242	3	879138235	Kolya (1996)
4	306	242	5	876503793	Kolya (1996)

[13]
```
#Calculate mean rating of all movies
data.groupby('title')['rating'].mean().sort_values(ascending=False).head()
```

```
title
Marlene Dietrich: Shadow and Light (1996)     5.0
Prefontaine (1997)                            5.0
Santa with Muscles (1996)                     5.0
Star Kid (1997)                               5.0
Someone Else's America (1995)                 5.0
Name: rating, dtype: float64
```

[15]
```
#Calculate count rating of all movies
data.groupby('title')['rating'].count().sort_values(ascending=False).head()
```

```
title
Star Wars (1977)             583
Contact (1997)               509
Fargo (1996)                 508
Return of the Jedi (1983)    507
Liar Liar (1997)             485
Name: rating, dtype: int64
```

[16]
```
#Creating dataframe with 'rating' count values
ratings = pd.DataFrame(data.groupby('title')['rating'].mean())
ratings['num of ratings'] = pd.DataFrame(data.groupby('title')['rating'].count())
ratings.head()
```

title	rating	num of ratings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

[17]
```
#Plot graph of 'num of ratings column'
ratings['num of ratings'].hist(bins = 70)
plt.show()
```

[19]

```
data.head()
```

	user_id	item_id	rating	timestamp	title
0	196	242	3	881250949	Kolya (1996)
1	63	242	3	875747190	Kolya (1996)
2	226	242	5	883888671	Kolya (1996)
3	154	242	3	879138235	Kolya (1996)
4	306	242	5	876503793	Kolya (1996)

[20]

```
moviemat = data.pivot_table(index ='user_id',columns ='title', values ='rating')
moviemat.head()
```

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	...	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	Young Poisoner's Handbook, The (1995)	Zeus and Roxanne (1997)	unknown	Á koldum klaka (Cold Fever) (1994)
user_id																					
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	5.0	3.0	NaN	NaN	NaN	4.0	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	2.0	NaN	NaN	NaN	NaN	4.0	NaN	NaN	...	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	4.0	NaN

5 rows × 1664 columns

[21]

```
ratings.sort_values('num of ratings', ascending = False).head(5)
```

title	rating	num of ratings
Star Wars (1977)	4.358491	583
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485

[22]

```
# analysing correlation with similar movies
starwars_user_ratings = moviemat['Star Wars (1977)']
liarliar_user_ratings = moviemat['Liar Liar (1997)']

starwars_user_ratings.head()
```

```
user_id
1    5.0
2    5.0
3    NaN
4    5.0
5    4.0
Name: Star Wars (1977), dtype: float64
```

[23]

```
#Analysing correlation with similar movies
similar_to_starwars = moviemat.corrwith(starwars_user_ratings)
similar_to_liarliar = moviemat.corrwith(liarliar_user_ratings)
similar_to_starwars.head()
```

```
C:\Users\Prerna\Anaconda3\lib\site-packages\numpy\lib\function_base.py:2522: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar)
C:\Users\Prerna\Anaconda3\lib\site-packages\numpy\lib\function_base.py:2451: RuntimeWarning: divide by zero encountered in true_divide
  c *= np.true_divide(1, fact)


title
'Til There Was You (1997)    0.872872
1-900 (1994)                -0.645497
101 Dalmatians (1996)        0.211132
12 Angry Men (1957)          0.184289
187 (1997)                   0.027398
dtype: float64
```

[24]

```
corr_starwars = pd.DataFrame(similar_to_starwars, columns =['Correlation'])
corr_starwars.dropna(inplace = True)
corr_starwars.head()
```

title	Correlation
'Til There Was You (1997)	0.872872
1-900 (1994)	-0.645497
101 Dalmatians (1996)	0.211132
12 Angry Men (1957)	0.184289
187 (1997)	0.027398

[25]

```
corr_starwars = corr_starwars.join(ratings['num of ratings'])
corr_starwars.head()
```

title	Correlation	num of ratings
'Til There Was You (1997)	0.872872	9
1-900 (1994)	-0.645497	5
101 Dalmatians (1996)	0.211132	109
12 Angry Men (1957)	0.184289	125
187 (1997)	0.027398	41

[26]

```
#Similar movies like starwars
corr_starwars.sort_values('Correlation', ascending = False).head(7)
```

title	Correlation	num of ratings
Hollow Reed (1996)	1.0	6
Commandments (1997)	1.0	3
Cosi (1996)	1.0	4
No Escape (1994)	1.0	5
Stripes (1981)	1.0	5
Star Wars (1977)	1.0	583
Man of the Year (1995)	1.0	9

[27]

```
corr_starwars[corr_starwars['num of ratings']>100].sort_values('Correlation', ascending = False).head()
```

title	Correlation	num of ratings
Star Wars (1977)	1.000000	583
Empire Strikes Back, The (1980)	0.747981	367
Return of the Jedi (1983)	0.672556	507
Raiders of the Lost Ark (1981)	0.536117	420
Austin Powers: International Man of Mystery (1997)	0.377433	130

title	Correlation	num of ratings
Hollow Reed (1996)	1.0	6
Commandments (1997)	1.0	3
Cosi (1996)	1.0	4
No Escape (1994)	1.0	5
Stripes (1981)	1.0	5
Star Wars (1977)	1.0	583
Man of the Year (1995)	1.0	9

[27]
```
corr_starwars[corr_starwars['num of ratings']>100].sort_values('Correlation', ascending = False).head()
```

title	Correlation	num of ratings
Star Wars (1977)	1.000000	583
Empire Strikes Back, The (1980)	0.747981	367
Return of the Jedi (1983)	0.672556	507
Raiders of the Lost Ark (1981)	0.536117	420
Austin Powers: International Man of Mystery (1997)	0.377433	130

[29]
```
#Similar movies as of liarliar
corr_liarliar = pd.DataFrame(similar_to_liarliar, columns =['Correlation'])
corr_liarliar.dropna(inplace = True)
corr_liarliar = corr_liarliar.join(ratings['num of ratings'])
corr_liarliar[corr_liarliar['num of ratings']>100].sort_values('Correlation', ascending = False).head()
```

title	Correlation	num of ratings
Liar Liar (1997)	1.000000	485
Batman Forever (1995)	0.516968	114
Mask, The (1994)	0.484650	129
Down Periscope (1996)	0.472681	101
Con Air (1997)	0.469828	137

REFERENCES

1. Knowledge-Based Recommender System Using Artificial Intelligence for Smart Education Humin Yang et al., World Scientific Book, 2022
2. Innovative Research on English Teaching Model Based on Artificial Intelligence and Wireless Communication Yuan Wang et al., International Journal of Reliability, Quality and Safety Engineering, 2022
3. The Capture and Evaluation System of Student Actions in Physical Education Classroom Based on Deep Learning Li Zhang et al., Journal of Interconnection Networks, 2022
4. Artificial Intelligence-Based Quality Management and Detection System for Personalized Learning Haixia Yu et al., Journal of Interconnection Networks, 2021
5. Smart Educational Learning Strategy with the Internet of Things in Higher Education System Jing Wang et al., International Journal on Artificial Intelligence Tools
6. Personalized exercise recommendation method based on causal deep learning: Experiments and implications , Suhua Wang et al., Mathematical Modelling and Control
7. Microlearning and computer-supported collaborative learning: An agenda towards a comprehensive online learning system Soheila Garshasbi et al., Mathematical Modelling and Control
8. Rough sets as a Knowledge Discovery and Classification Tool for the Diagnosis of Students with Learning Disabilities .Tung-Kuang Wu et al., International Journal of Computational Intelligence Systems, 2011
9. Augmented Intelligence in Joint Replacement Surgery: How can artificial intelligence (AI) bridge the gap between the man and the machine? Vaibhav Bagaria et al., Arthroplasty, 2022
10. A review on peak shaving techniques for smart grids Syed Sabir Hussain Rizvi et al., AIMS Energy, 2023

11. Borges HL and Lorena AC. A survey on recommender systems for news data. Smart Information and Knowledge
12. Herlocker JL, Konstan JA, and Riedl J. Explaining collaborative filtering recommendations. In Proc 2000 ACMConf Comput Supported Cooperative Work. 2000;241-250p.
13. Subramaniyaswamy V, Logesh R, Chandrashekhar M, et al. A personalized movie recommendation system based on collaborative filtering. Int J High Perform Comput Netw. 2017;10(1-2):54-63.
14. Bleier A, Harmeling C, and Palmatier RW. Creating effective online customer experiences. J Market. 2019;83(2):98-119.
15. Cenci MP, Scarazzato T, Munchen DD, et al. Eco-friendly electronics—A comprehensive review. Advan MaterTechnol. 2022;7(2):2001263.
16. Ashley-Dejo E, Ngwira S, and Zuva T. A survey of context-aware recommender system and services. In 2015 IntConf Comput, Commun Securi (ICCCS). IEEE. 2015; 1-6p.
17. Kulkarni S, and Rodd SF. Context Aware Recommendation Systems: A review of the state of the art techniques. Comput Sci Rev. 2020;37:100255

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University u / s 3 of UGC Act, 1956)

Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION / PROJECT REPORT FOR UG / PG PROGRAMMES

(To be attached in the dissertation / project report)

1	Name of the Candidate **(IN BLOCK LETTERS)**	AASHIK A HASHIR AHMMED KABIR NAVEEN M
2	Address of Candidate	Bharathi Salai, Ramapuram, Chennai-89. **Mobile Number:** 9840910351, 9061344810, 7904547279
3	Registration Number	RA2011003020559, RA2011003020568, RA2011003020564
4	Date of Birth	03/06/2003, 20/08/2002, 04/02/2002
5	Department	Computer Science and Engineering
6	Faculty	Dr. Deepa. B
7	Title of the Dissertation / Project	Hybrid Movie Recommendation System Using MF-CF Algorithm
8	Whether the above project / dissertation is done by	Individual or group : Group (Strike whichever is not applicable) a) If the project / dissertation is done in group, then how many students together completed the project : 03 b) Mention the Name and Register number of other candidates : AASHIK A [RA2011003020559] HASHIR AHMMED KABIR [RA2011003020564] NAVEEN M [RA2011003020568]
9	Name and address of the Supervisor / Guide	Dr. Deepa. B, Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram Campus, Chennai 89 **Mail ID :** senthila3@srmist.edu.in **Mobile Number :** 9787321529

10	Name and address of the CoSupervisor / Guide	NA **Mail ID:** NA **Mobile Number:** NA		
11	Software Used	Turnitin		
12	Date of Verification	30/04/2024		
13	**Plagiarism Details: (to attach the final report from the software)**			
Chapter	**Title of the Report**	**Percentage of similarity index (including self citation)**	**Percentage of similarity index (Excluding self citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	Towards Optimized and Empirical Evaluation of Brain Tumor Segmentation Using Kernel Ensemble Learning	NA	NA	8%
Appendices		NA	NA	NA

I / We declare that the above information has been verified and found true to the best of my / our knowledge.

Signature of the Candidate	**Name and Signature of the Staff (Who uses the plagiarism check software)**
Name and Signature of the Supervisor / Guide	**Name and Signature of the Co-Supervisor / Co-Guide**

Dr. K. Raja
Name and Signature of the HOD

PAPER NAME

Hybrid movie recommendation system using MF – CF Algorithm

WORD COUNT 10386 Words	CHARACTER COUNT 64205 Characters
PAGE COUNT 58 Pages	FILE SIZE 1.1MB
SUBMISSION DATE Apr 30, 2024 1:01 PM GMT+5:30	REPORT DATE Apr 30, 2024 1:02 PM GMT+5:30

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 4% Internet database
- 2% Publications database
- Crossref database
- Crossref Posted Content database
- 7% Submitted Works database

Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 10 words)