<Neural Network Based Movie Ratings Predictions

<Hamidur Rahman >

<20009146>

UXCFXK-30-3

Digital Systems Project



# Abstract

This project uses a deep learning neural network model to power an advanced movie recommendation system that predicts user ratings for a personalised user experience. In the framework of this project, my system makes use of a wealth of data regarding user interactions to forecast precise user preferences by combining collaborative filtering with content-based filtering techniques. The robust Flask web framework and MySQL database architecture have more effectively integrated the fundamental features of managing user accounts and accessing comprehensive movie data. One of the system's significant characteristics is its ability to visualise data and analytics insights from a user's perspective, making the representation simple for users to understand and analyse in order to understand the reasoning behind suggestions. This degree of transparency in the data display would improve the user experience and boost confidence in the system's predictions. We have met the majority of the project's primary goals, but prioritisation issues have delayed several extra features like social integration and real-time data processing until a later stage of development. This gave us a great deal of insight into the difficulties of combining sophisticated machine learning models with user-friendly interfaces to enable flexible and scalable model training. Future developments will focus on enhancing the system's interactive capabilities and prediction accuracy in order to further increase the responsiveness and accuracy of the system.

# Acknowledgements

**Thanks to Elias Pimenidis and Marco Perez Hernandez for their feedback and involvement in this project.**

# Table of Contents

[Abstract 1](#_Toc164842117)

[Acknowledgements 2](#_Toc164842118)

[Table of Contents 3](#_Toc164842119)

[Table of Figures 6](#_Toc164842120)

[Introduction 7](#_Toc164842121)

[Problem Statement and Importance 7](#_Toc164842122)

[Aims and Objectives 7](#_Toc164842123)

[Literature Review Insights 7](#_Toc164842124)

[Approach to the Problem 7](#_Toc164842125)

[Project Outcome 8](#_Toc164842126)

[Report Outline 8](#_Toc164842127)

[Literature Review 9](#_Toc164842128)

[Introduction 9](#_Toc164842129)

[Requirements 15](#_Toc164842130)

[Introduction 15](#_Toc164842131)

[Functional Requirements 15](#_Toc164842132)

[Must Have 15](#_Toc164842133)

[Should Have 15](#_Toc164842134)

[Could Have 16](#_Toc164842135)

[Won't Have 16](#_Toc164842136)

[Non-Functional Requirements 16](#_Toc164842137)

[Must Have 16](#_Toc164842138)

[Should Have 17](#_Toc164842139)

[Could Have 17](#_Toc164842140)

[Won't Have 17](#_Toc164842141)

[Methodology 17](#_Toc164842142)

[Use Case Diagram 18](#_Toc164842143)

[Sequence Diagram 19](#_Toc164842144)

[Entity Relationship Diagram 20](#_Toc164842145)

[Data Flow Diagram 20](#_Toc164842146)

[Test Design 21](#_Toc164842147)

[Implementation 22](#_Toc164842148)

[System Architecture 22](#_Toc164842149)

[Overview 22](#_Toc164842150)

[Data Model Description 23](#_Toc164842151)

[Data Handling 23](#_Toc164842152)

[Data Collection 23](#_Toc164842153)

[Data Pre-processing 24](#_Toc164842154)

[Model Implementation 25](#_Toc164842155)

[Architecture 25](#_Toc164842156)

[Importance of Weights in Model Training 26](#_Toc164842157)

[Calculation of Weights 26](#_Toc164842158)

[Implementation in Training and Validation Steps 27](#_Toc164842159)

[Impact on Model Performance Metrics 27](#_Toc164842160)

[Overview of Hyperparameter Tuning 27](#_Toc164842161)

[Definition of Hyperparameter Space 27](#_Toc164842162)

[Objective Function for Optimization 28](#_Toc164842163)

[Model Performance 28](#_Toc164842164)

[Weighted vs. Non-Weighted Model Outcomes 28](#_Toc164842165)

[Reflection on Test Results 29](#_Toc164842166)

[Insights and Future Directions 29](#_Toc164842167)

[Model Integration 30](#_Toc164842168)

[Prediction For Single Movie 31](#_Toc164842169)

[Implementation Details 32](#_Toc164842170)

[Software Testing 33](#_Toc164842171)

[Tools and Frameworks Used 33](#_Toc164842172)

[Example 33](#_Toc164842173)

[Rationale Behind Testing Choices 34](#_Toc164842174)

[Project Evaluation 34](#_Toc164842175)

[Evaluation on Research 34](#_Toc164842176)

[Evaluation on Requirements 35](#_Toc164842177)

[Challenges 35](#_Toc164842178)

[Feedback Integration 35](#_Toc164842179)

[Improvement Opportunities 36](#_Toc164842180)

[Further Work and Conclusions 37](#_Toc164842181)

[Summary of Project Aims and Achievements 37](#_Toc164842182)

[Discrepancies Between Planned and Actual Outcomes 37](#_Toc164842183)

[Growth in Ideas 37](#_Toc164842184)

[Further Research 37](#_Toc164842185)

[Conclusion 38](#_Toc164842186)

[References / Bibliography 39](#_Toc164842187)

# Table of Figures

[Figure 1: Use Case Diagram 18](#_Toc164842197)

[Figure 2: Sequence Diagram 19](#_Toc164842198)

[Figure 3: Entity Relationship Diagram 20](#_Toc164842199)

[Figure 4: Data Flow Diagram 21](#_Toc164842200)

[Figure 5: Test Plan 22](#_Toc164842201)

[Figure 6: Number of Movies per Tag for Top Ten Tags 25](#_Toc164842202)

[Figure 7: Bar Chart Showing Number of Ratings per Rating Value 26](#_Toc164842203)

[Figure 9: R2 Comparison Chart 29](#_Toc164842204)

[Figure 10: Predicted Movie Details Page 32](#_Toc164842205)

# Introduction

In today's digital age, personalized content delivery is paramount, especially in the entertainment industry where understanding user preferences can significantly enhance user engagement and satisfaction. The project developed an advanced movie recommendation system designed to address this need by predicting user ratings and personalizing content through a deep learning neural network model.

### Problem Statement and Importance

The primary real-world problem addressed by this project is the challenge of effectively recommending movies to users based on their unique preferences and past interactions. This issue is significant as it directly impacts user satisfaction and retention in highly competitive digital entertainment platforms. By providing accurate and personalized movie recommendations, platforms can ensure a higher degree of user engagement and increased usage.

### Aims and Objectives

The aim of this project was to create a robust movie recommendation system that utilizes advanced machine learning techniques to predict user preferences and suggest personalized content. The objectives were to:

* Implement a deep learning model that accurately predicts movie ratings.
* Integrate collaborative and content-based filtering techniques for effective recommendation.
* Develop an intuitive interface that displays data visualizations, helping users understand why certain movies are recommended.

### Literature Review Insights

The literature review revealed that integrating machine learning with recommendation systems has shown promising results in enhancing the accuracy of predictions and the personalization of content. Studies highlighted the effectiveness of neural networks in capturing complex patterns in data, which are crucial for predicting user preferences more accurately than traditional methods.

### Approach to the Problem

The approach involved designing a system architecture that supports sophisticated data processing and machine learning algorithms. This included setting up a Flask web framework and a MySQL database, along with the implementation of a neural network using PyTorch. The system was designed to not only provide recommendations but also to present data visualizations that offer users insights into the recommendation logic.

### Project Outcome

The project successfully developed a fully functional movie recommendation system that met its primary aims. Although some features were deferred for future updates, the system effectively demonstrates the capability of deep learning models in a practical, user-focused application.

### Report Outline

* **Chapter 2: Literature Review** - This chapter provides a critical review of existing works and methodologies relevant to recommendation systems and deep learning applications.
* **Chapter 3: Requirements** - Details the comprehensive requirements for the project, including functional and non-functional specifications, structured using the MoSCoW method.
* **Chapter 4: Design** - Describes the system design.
* **Chapter 5: Implementation** - Discusses the implementation process, the architecture of the deep learning model, and presents the results derived from the model.
* **Chapter 6: Evaluation of Project** - Analysis of the project’s effectiveness, covering the performance of the recommendation system and the integration of feedback.
* **Chapter 7: Further Work and Conclusions** - Summary of the project outcomes, challenges faced, lessons learned, and detailed suggestions for future enhancements and research directions.

# Literature Review

# Introduction

The development of digital technology has completely changed how we interact with media, particularly about the way we choose and consume films. An ever-growing selection of films are available to us, therefore effective and personalised recommendation systems are becoming increasingly crucial. To satisfy this need, machine learning-based movie recommendation systems are setting the standard by offering users customised recommendations based on a variety of factors, such as social trends and personal preferences. The goal of this study of the literature is to perform a comprehensive review on the creation, use, and effectiveness of machine learning-based movie recommendation systems. From the initial stages of content-based and collaborative filtering to the sophisticated combination of deep learning and cross-content techniques, this review examines a variety of methodologies. This review aims to provide comprehensive insights into the challenges, advancements, and prospects of these systems. The goal is to understand how these technological advancements have transformed the cinematic experience by establishing a link between vast movie databases and the unique preferences of individual spectators. This study presents the latest developments in recommendation systems technology and paves the way for a deeper comprehension of what is needed to drive future innovation in this quickly developing field.

Evolution and Effectiveness of Recommender Systems

The significant development of machine learning applications in the entertainment sector is best illustrated by the creation of movie recommendation systems. These systems were initially simple, using basic algorithms to suggest films according to popular genres or trends. But as the digital era progressed, an enormous volume of data and a diverse range of user preferences emerged, underscoring the need for increasingly sophisticated recommendation algorithms.

Two different eras may be identified in the history of recommender systems: the collaborative filtering period and the content-based filtering emerging. By placing a strong emphasis on user behaviour, collaborative filtering—which rose to prominence in the late 1990s and early 2000s—significantly altered the methodology. It works on the premise that people with similar viewing histories are going to have similar tastes in films. While this approach proved effective, it was not without limitations, particularly when it came to handling new users who had no prior experience—a problem known as the "cold start" issue.

On the other hand, content-based filtering makes movie recommendations based on the movie's attributes, including its genre, director, or cast of actors. While this approach does a good job of addressing the problem of beginning from scratch, it often ignores the haphazard nature of finding films, leading to recommendations that are narrow.

The effectiveness of these systems was significantly increased with the addition of machine learning algorithms. According to Mohammad and Urolagin (2022), the incorporation of computational intelligence and data clustering into recommender systems has resulted in enhanced personalisation and precision of recommendations, thereby improving the user experience on digital platforms (Mohammad & Urolagin, 2022). Furthermore, PireciSejdiu, N., Ristevski, B., & Jolevski, I. (2022) pay a lot of attention to the impact on the efficacy of movie recommendation systems in their study. However, despite establishing the crucial role that algorithmic selection plays in driving recommendation accuracy, the research did not provide a thorough understanding of the contextual and user-centric factors influencing recommendation results. However, a more in-depth inquiry into the adaptability of these algorithms to dynamic user preferences and emerging movie trends would enable a more nuanced understanding of how they are adaptable to real-life situations. This gap therefore opens an opportunity to integrate machine learning advancements with the evolving patterns of entertainment consumption for the refinement of personalisation of the recommendation systems as a focus of future research.

These advancements have enormous implications for the user experience. Modern recommender systems offer a more nuanced understanding of individual preferences, making the process of finding films more engaging and rewarding. In addition to films that align with their stated likes, users are also offered with options that could positively surprise them, making their entire movie-watching experience even more enjoyable.

Overall, advancements in artificial intelligence and machine learning are reflected in the creation of movie recommendation systems. These systems, which range from simple algorithms to sophisticated, customised recommendation engines, have significantly improved how audiences discover and enjoy films, making the vast world of cinema more approachable and tailored to specific tastes.

Novel Approaches in Recommendation Systems

Movie recommendation systems have recently shifted towards more inventive and multidimensional methods, particularly by utilising transfer learning. These developments not only improve the customisation of recommendations but also expand the range of discoveries for users.

Valliyammai and Ephina Thendral's clustering-based transfer learning for cross-domain recommendations (2019) offer a fresh view to personalisation in sparse domains. This, however, brings us to the question of whether the same may be applied in movie recommendation systems as genres and preferences are dynamic. The methodology's biggest challenge is adapting to evolving tastes over time, which might not be possible in a quickly changing environment. The study offers valuable insights into cold start problems and improved personalisation, while the author suggests further refinement. It implies the need for tailoring these techniques to better represent and adapt to changing movie tastes and user interactions.

The study carried out by Lavanya, Gogia, and Rai (2021) stands as one of the notable additions to the pool of works aimed at movie recommender systems with auto encoders in collaborative filtering. This unique approach using the MovieLens dataset demonstrates the potential for deep learning to improve personalisation in movie recommendations. Their approach therefore holds great potential, especially with its increased ability to predict user ratings, and would be tremendously important towards making recommendations that cater to individual user preferences. This study is pioneering because it uses auto-encoders, but it can open new opportunities for further research, most importantly in assessing how various machine learning models can be combined to capture multi-faceted movie preferences. This represents a valuable contribution to the development of more responsive and user-centric movie recommender systems, pointing towards a future where recommendations could be even better aligned to individual tastes and viewing habits.

These innovative methods are fundamentally transforming the framework of movie recommendation systems. By harnessing the powers of machine learning and broadening the availability of information, these systems not only adapt to users' unique preferences but also expose them to a wider array of diverse experiences. The use of multifaceted strategies and cutting-edge algorithms demonstrates a substantial advancement in the development of recommendation platforms, resulting in more dynamic, interactive, and immersive user experiences with films and related information.

Exploration into Deep Learning

The integration of deep learning into movie recommender systems represents an important leap over conventional machine learning techniques, providing a more refined methodology for understanding user preferences and improving the quality of recommendations. Deep learning enhances recommendation algorithms by leveraging its capacity to analyse and acquire knowledge from large datasets, surpassing the capabilities of previous methods.

Yao (2023) provides a detailed review of the deep learning implementation of movie recommender systems, drawing a large stride towards attaining the personalisation of the user experience. This study accurately outlines how deep learning surpasses traditional limits like the cold start problem and data sparsity by intelligently using complex data patterns to show tailor-made movie selections. Yao (2023) evaluates two deep learning approaches, focusing on their effectiveness in improving the accuracy of recommendations, an important feature for recommender systems that aspire to match users with suitable content satisfying their preferences. While advocating these deep learning advancements for movie recommendations in the paper, it subtly suggests the necessity of ongoing research to refine these systems further. Especially, integrating real-time user feedback could dynamically fine-tune recommendations while alluding to the evolving relationship between user preferences and system outputs. Yao's (2023) work gives the reader a firm picture of how deep learning is perfectly linked to movie recommendation systems in the aggregate, hence helping in outlining further advances characterised by highly intuitive and very responsive user immunity.

Lin and Chi (2019) take up the challenge to innovate in the sphere of movie recommendation systems by fusing collaborative filtering with neural networks, with an aim towards refining accuracy in terms of user-specific movie suggestions. This hybrid approach, combining the best features of Scikit-learn and TensorFlow, is a major step forward in breaking the traditional system barriers, particularly when it comes to predictable accuracy. This model hybridises what could be done with a neural network that could offer up more subtle suggestions. While the research focuses on primarily technical advancements, the practical implications of the study pave the way for a promising journey to make more personalised and accurate movie recommendations, which invites an exploration of how this could be applied in various recommendation system environments.

To summarise, the use of deep learning in movie recommendation systems indicates a significant progression beyond conventional approaches. The user experience is enhanced by its capacity to analyse intricate data, derive insights from implicit patterns, and offer highly customised recommendations. Deep learning is a fundamental technology that is driving the future of personalised entertainment as these systems continue to advance.

Comprehensive Analysis of Movie Recommendation Systems

The field of movie recommendation systems is a complex combination of algorithms, performance measurements, and inherent difficulties. The design and effectiveness of these systems are influenced by multiple aspects, as emphasised in an in-depth review of existing literature.

The algorithms are crucial components of these systems. Originally, collaborative filtering (CF) and content-based filtering (CBF) were the main approaches to this subject. However, it has since progressed and developed. Collaborative filtering algorithms, however successful in utilising user ratings, encounter challenges in terms of scalability and the cold start problem. Concentrating on movie attributes, such as CBF, frequently results in a narrow selection of recommendations. Current trends involve the integration of collaborative filtering (CF) and content-based filtering (CBF) in hybrid models, as well as the utilisation of advanced machine learning methods, specifically deep learning.

Jayalakshmi et al. (2022) undertake an intricate study on movie recommender systems and discuss the applications of complex algorithms such as K-means clustering and principal component analysis. This systematic review observes the mass leap in computerised intelligence to personalise movie recommendations. While the study adeptly outlines present methodologies and challenges, it paves the way further towards integrating metaheuristic-based systems to further accurate recommendation. The paper forms a baseline for further research aiming at surpassing the implementation challenges for a better experience and, therefore, emphasising the need to enhance machine learning techniques in today's highly changeable domain of movie recommendations.

Despite technological advancements, difficulties such as the cold start problem, data sparsity, and scalability continue to exist. An essential element involves preserving diversity and unpredictability in recommendations, thereby avoiding viewers being confined within a "filter bubble" and encouraging exposure to a broader selection of movies.

To summarise, the field of movie recommendation systems is characterised by sophisticated algorithms and ongoing challenges. This analysis emphasises the importance of constant innovation to accommodate the ever-changing preferences of customers and the growing landscape of the movie industry.

Implementation and Evaluation of Recommendation Systems

The successful adoption and acceptance of movie recommendation systems heavily rely on the effective implementation and subsequent evaluation of their practical functionality. These systems, commonly integrated into broader digital platforms, utilise diverse machine learning algorithms to offer customised movie recommendations. Case studies and real-world applications offer significant insights into the effectiveness and difficulties of various implementations.

The study conducted by Ahuja, Solanki, and Nayyar (2019) highlights the potential of integrating K-means clustering and K-nearest neighbour algorithms in recommender systems. This combination of machine learning techniques can lead to more precise and improved recommendations.

Assessment metrics are crucial in evaluating the effectiveness of movie recommendation systems. Typical measures are accuracy, precision, recall, F1 score, MAE, and RMSE. These metrics aid in quantifying the system's capacity to accurately forecast user preferences and the margin of error in these predictions. Precision quantifies the ratio of pertinent recommendations out of all recommendations provided, whereas recall evaluates the ratio of pertinent films that were successfully recommended.

The evaluation procedure frequently entails the comparison of the performance of various algorithms or models within the same system. For instance, one can compare the efficacy of collaborative filtering with content-based filtering or evaluate hybrid methods in comparison to independent machine learning models. The objective is to determine the algorithm that achieves the highest level of customer satisfaction and system efficiency.

To summarise, the execution and assessment of movie recommender systems are complex procedures that necessitate meticulous examination of diverse algorithms and performance measures. Case studies and realistic deployments offer essential insights into the functionality of the systems, while thorough evaluation ensures their effectiveness and ongoing enhancement. These techniques are essential for the creation of recommendation systems that prioritise the needs of users and are technologically robust.

Novel Techniques and Future Directions

The movie recommendation systems have been impacted by new technologies and techniques, which are offering novel approaches that aim to improve the user experience and increase the accuracy of recommendations.

A survey by Abbas, Zhang, and Khan (2015) provides elaborate coverage of context-aware recommender systems, which explicitly focuses on the use of computational intelligence to improve the accuracy of these systems and the issues of sparsity and cold starts. Its detailed classification of computational intelligence techniques and application space across domains is very insightful. In the context of movie recommendations, the application of these techniques would ever-more enhance the probable methods of every viewer to locate movies by state-of-the-art contextual aspects like time of viewing or social environment, with significantly more valuable suggestions for every individual viewer. However, such considerations of privacy and the dynamic nature of user preferences must be very carefully considered in real movie recommendation systems. Overall, this is an excellent contribution towards enhancing the sophistication of movie recommendation systems through computational intelligence.

Zhang et al. (2021) propose a deep learning-based architecture of sentiment analysis of movie reviews that enhances the achieved accuracy to 83.13%. This approach is considered by the authors as a promise that deep learning demonstrates in analysing intricate aspects of users' sentiments and augurs well for future work on personalised movie suggestions. However, a focus on technical achievement in the work without an extensive comparison to state-of-the-art models or an exploration of how such models fit into broader recommendation systems limits the applicability of this work. While promising, the research leaves room for further exploration to its efficacy across varied genres and user demos, essential in tailoring recommendations in an entertainment landscape that is diverse.

Moreover, the integration of AI (Artificial Intelligence) and machine learning with privacy and ethical considerations will certainly be a prominent trend. As recommendation systems progress, it will be crucial to prioritise user data protection and uphold ethical standards in algorithm usage. Li et al. (2020) emphasised this aspect by examining the concept of differential privacy in movie recommendation systems. They specifically focused on the importance of ensuring privacy security in cloud-based recommendation environments (Li, Zeng, Guo, & Guo, 2020).

Movie recommendation systems are on the verge of substantial progress through the implementation of new technologies and approaches. These advancements hold the potential to create personalised, interactive, and privacy-aware suggestions, paving the way for an exciting future in this industry.

Conclusion

The present literature review provides a comprehensive overview of the current state and future possibilities of movie recommendation systems that rely on machine learning. The key findings suggest that these systems have undergone substantial advancements, transitioning from simple collaborative and content-based filtering to more complex hybrid models that use advanced machine learning approaches. Deep learning has become a powerful force that can provide detailed and highly customised suggestions by analysing intricate user behaviour patterns and preferences.

The incorporation of innovative technology such as voice assistants, big data analytics, and privacy-conscious algorithms has enhanced the functionalities of these platforms. These innovations improve the user experience by providing more interactive and personalised recommendations. Additionally, they tackle important challenges like data sparsity, scalability, and the cold start problem.

The movie recommendation systems field is poised to experience substantial growth and innovation in the future. Future research will focus on developing algorithms for real-time flexibility, exploring ethical and privacy problems in greater depth, and integrating future technology to handle the ever-increasing volume of data. As these systems advance in complexity and prioritise the needs of users, they will continue to have a crucial impact on defining the experience of watching films. They will help connect the divide between extensive movie collections and the specific preferences of individual viewers.

# Requirements

## Introduction

This chapter outlines the comprehensive requirements for the development of the movie site. To ensure clarity, the requirements are divided into functional and non-functional categories and prioritized using the MoSCoW method.

## Functional Requirements

Functional requirements specify the essential operations and behaviours of the system. These requirements ensure the system fulfils its objective of recommending movies effectively.

### Must Have

1. **User Account Management (FR1):** Qin and Zhang (2021) introduced an attention mechanism to their model to enhance recommendation performance by tracking user preferences over time, so user accounts are crucial for documenting the changing preferences of users over time.
2. **Movie Ratings (FR2):** Allow users to add their own ratings to movies. Chen, Yeh and Ma (2021) utilises ratings to enhance their recommendation model by using the users’ ratings to build positive and negative profiles where the recommendation algorithm will look for movies similar to their positive profile and most different to their negative profile.
3. **Personalised Recommendations (FR3):** At the centre of the transition from simple algorithms to advanced, deep learning-driven systems. This requirement supports the transition to more sophisticated models capable of addressing longstanding issues such as the cold start problem and data sparsity. The model will use user ratings and movie metadata to generate personalised recommendations to the user.
4. **Access to Detailed Movie Information (FR4):** It will enable the use of both collaborative and content-based filtering techniques on extensive datasets for data analysis. This requirement is crucial for gaining a deeper understanding of the movies. It provides a means to enhance the precision of recommendations by employing content-based methods and utilising a more comprehensive dataset in machine learning models.
5. **Administrator Statistics (FR5):** One of the users' needs further advancement in the recommendation algorithms of future research. Elaborated usage statistics make it possible to establish tendencies in user behaviour and the effectiveness of various recommendation strategies that may guide improvements or adaptations in the future.

### Should Have

1. **Movie Search Functionality (FR6):** The technology facilitates user engagement, enabling immediate access to a vast movie database and enhancing user friendliness. This feature enables a wide range of content search options and helps users directly engage with diverse user experiences while avoiding the limitations of filter bubbles.
2. **New and Trending Recommendations (FR7):** The shifting landscape is characterised by ever-changing user choices and rapid fluctuations in movie popularity. Consequently, it is necessary to include the latest and well-received films to update and enhance the system, making it appealing to keep up with evolving preferences and growing patterns.

### Could Have

1. **Social Features (FR9):** Inspired by novel approaches in recommendation systems that leverage social networks and user interactions. Integration with social features can offer an opportunity for the next level of personalisation in recommendations by analysing social behaviour and preference data.
2. **Watchlist Feature (FR10):** This reveals knowledge on user behaviour, hence underscoring the need for capturing dynamic preferences. For example, by allowing users to save movies for later, it would add more data about intentions and user preferences, leading to better recommendations and thus mitigating the challenge of evolving tastes as an adaptive challenge.

### Won't Have

1. **External Streaming Service Integration (FR11):** The decision to prioritise core functionalities and not to make this complex feature, but rather to focus on improving the recommendation engine itself while, at the same time, ensuring the best seamless user experience. It will not be implemented with the decision to first solve the basic problems and opportunities in recommendation accuracy and personalisation instead of expanding the scope of the system.

## Non-Functional Requirements

Non-functional requirements outline the system's quality attributes, performance metrics, and operational constraints.

### Must Have

1. **NFR1 (Performance):** Response times of less than 2 seconds are maintained to assure satisfaction and active engagement. Within a recommendation system, users consistently want quick and precise responses. This requirement is of utmost importance to prevent the user from becoming frustrated with the programme and abandoning its use.
2. **NFR2 (Security and Compliance):** Strict following of GDPR and Data Protection Laws. User data is protected by law and important for keeping the trust of its users while at the same time avoiding breaches.
3. **NFR3 (Availability):** The service must be universally available since it is designed so that users from around the world can access it. To allow users access to the system at any time and provide reliable service, the system must be available 99.9% (outside scheduled maintenance) of the time.
4. **NFR4 (Data Integrity):** The accuracy of user data and recommendations is critical for the credibility of the recommendation system. Ensuring data integrity through validation and error-handling mechanisms is paramount to providing reliable and valuable recommendations to users.

### Should Have

1. **NFR5 (Scalability):** This feature becomes crucial due to its ability to accommodate the increasing trend of growth in both the number of users and movie databases. Though not critical at the current stage, planning for growth will prevent performance erosion over time.
2. **NFR6 (Usability):** This is an especially important design criteria for giving users can easily understandable interface that will help them move around the application with ease while using its features without frustration, which influences the time length and frequency of using the application.
3. **NFR7 (Maintainability):** The design should be such that it facilitates easy maintenance and updating. Failure to do so will lead to the system's obsolescence. It shall allow the system to change and improve over time, according to users and technological development.

### Could Have

1. **NFR8 (Multilingual Support):** Multilingual support will bring an experience that will significantly enhance the app's engagement and expand its use. It is important to serve a larger audience; this need not be taken up immediately for launch and can be built upon from user feedback and business demand.
2. **NFR9 (Accessibility):** This is an important aspect of inclusivity, and the application could be accessible to differently abled users. This ideally must be of higher priority than many others, which shall come under the "could have" category due to resource and time constraints. Still, efforts will be made to follow the accessibility standards as closely as possible.

### Won't Have

1. **NFR10 (Backup and Recovery):** Even though this is critically important for system reliability and data integrity, the comprehensive plan of backup and recovery might fall into the "Won't Have" category simply because of resource and time constraints. However, this should be worked out as soon as possible post-launch to be able to minimize the risks that are associated with data loss and system failure.

# Methodology

The Agile methodology, particularly Scrum, was chosen for its adaptability and iterative nature, which is well-suited for projects that involve complex software development. This methodology facilitated a structured yet flexible approach to the project's execution, allowing for continuous refinement through iterative cycles of planning, development, and testing. Adopting Scrum enabled regular evaluation against the project’s objectives and effective incorporation of new insights and adjustments based on ongoing analysis and feedback. This approach was instrumental in maintaining a focus on achieving the research goals while adapting to evolving requirements and challenges.Design

## Use Case Diagram

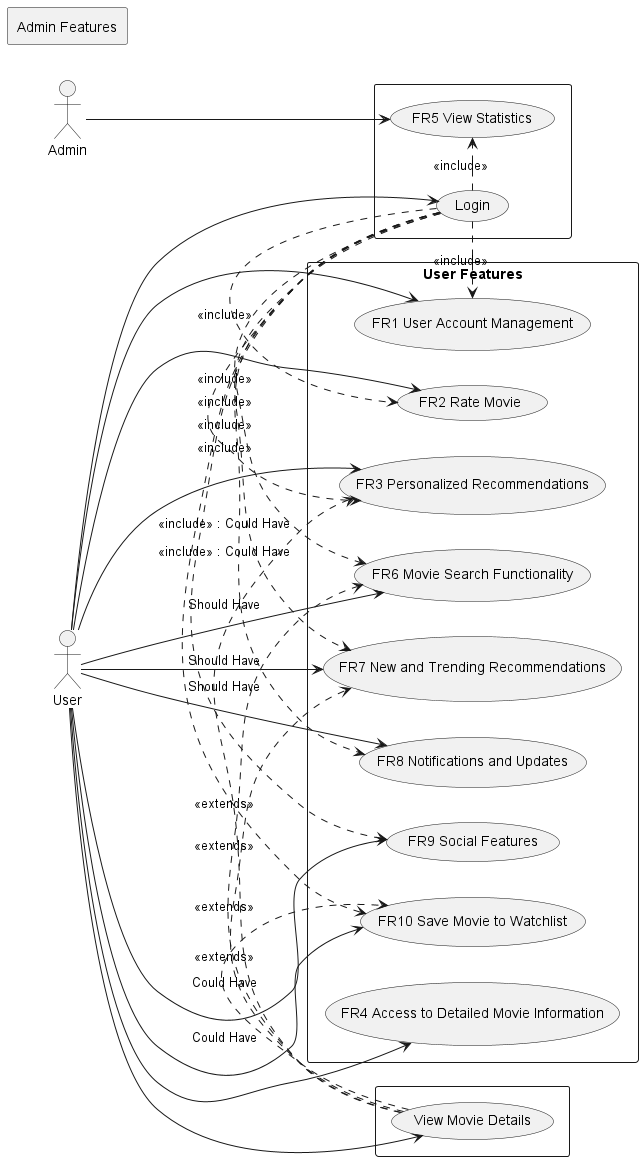


Figure 1: Use Case Diagram

The use case diagram illustrates the system's comprehensive functionality and user interaction pathways. The use case diagram features two distinct actors: the user and the administrator, each with unique functionalities for interacting with the system. The user's interaction places an emphasis on the user-centric design of the system. For functionalities at the core of the diagram, login is required to show security and personalisation; to view movie details, this applies to various functionalities of the diagram, all in an attempt to bolster user activity and system interactivity. Defining explicitly the lines of operation between user and administrative functionalities with a view to focused and efficient administrative oversight. The administrative role, which grants access to the administrator's statistics, must be clearly defined. Such clarity of organisation at the diagrammatic level adds to both the consolidation of robustness and flexibility of the system and represents a focus on the user experience that has to be intuitive and at the same time very rich.

## Sequence Diagram

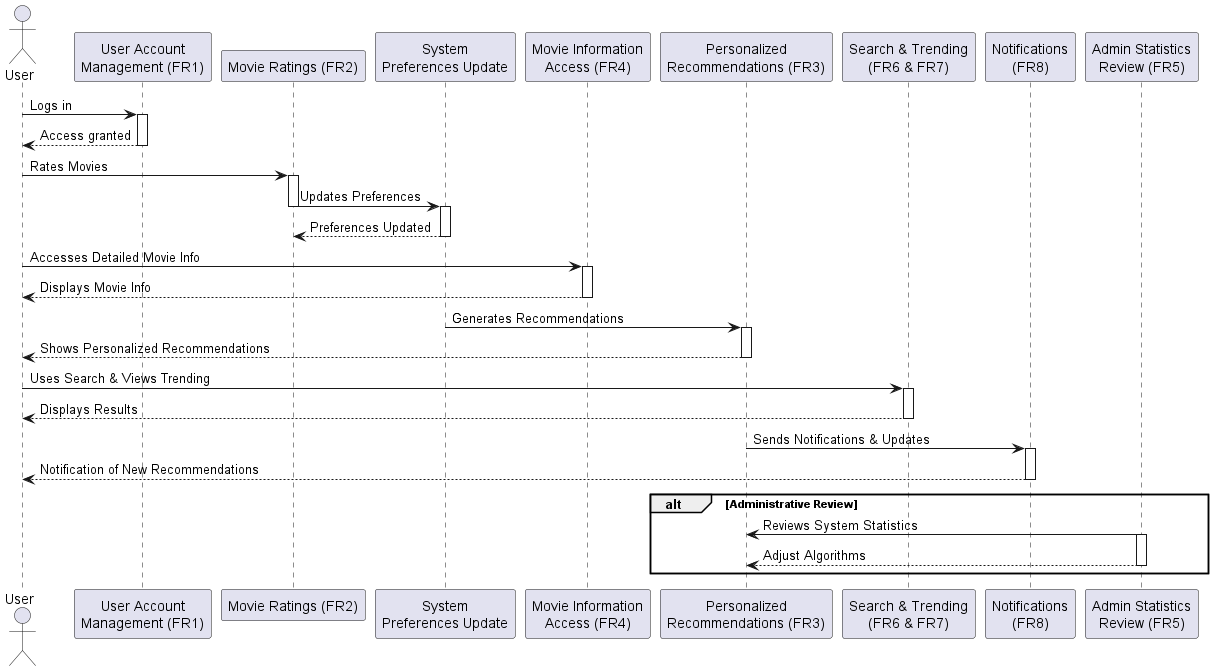


Figure 2: Sequence Diagram

This sequence diagram elaborates on the operation of the system flow, through which the user interacts with the different components of the system. Key system interactions will involve user login, movie ratings, an update in user preferences, access to movie information to aid them in making choices, receiving recommendations made by the system, using a search, viewing trending movies, and an administrative review of the system statistics for adjusting the algorithm.

We use this diagram because it effectively illustrates the sequence of interactions, highlighting the dependencies and flow of various parts within the system. This helps in understanding how the different features of the system work together, hence making it easier to understand how the users’ interface with the system components and how the said components relate to each other. It also aids in observing the possibility of bottlenecks or inefficiencies in the flow of the interaction with respect to focused improvements to the system design. The tags of the functional requirements seen next to each participant only exemplify the alignment of system interactions with specified requirements and show the design rationale of the system, which primarily focuses on meeting the needs and preferences of users.

## Entity Relationship Diagram

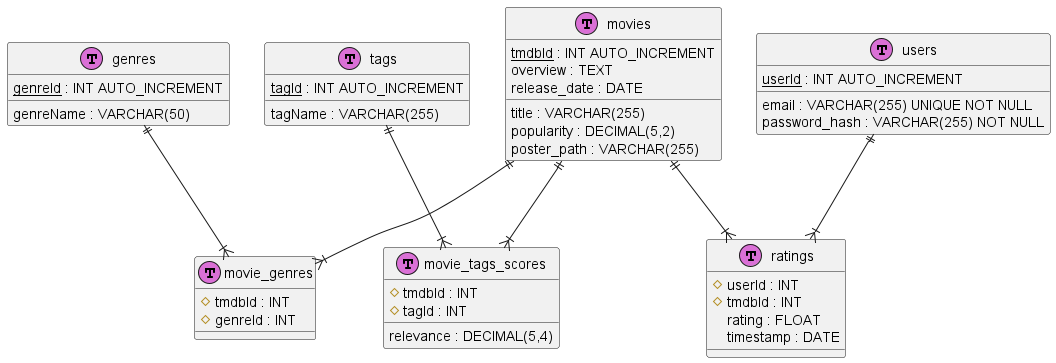


Figure 3: Entity Relationship Diagram

The entity-relationship diagram (ERD) shows a blueprint for the database structure using the MovieLens dataset. This diagram shows, with precision, the relationship between the entities. The ERD maintains data security and smooth data retrieval; therefore, there are foreign keys (‘movieId’) that relate table rows from movies to rows in genres, ratings, and tag relevance scores.

The great usefulness of this chart includes very helpful displays that show, in a visual way, relations of data that are priceless for understanding schema development and the ability to understand and perceive how the data interacts with each other. Therefore, the recommendation algorithm that is solely dependent on this relationship needs to bring out the analysis of user preference and movie metadata. It would thereby be able to tag information to give out personalised movie recommendations.

The project benefits more from the early integration of the ERD into the designing stage of Flask web application development, as it enables a structured approach to database design. This approach structures the identification and addressing of potential data management challenges, optimises data storage, and, at the same time, ensures smooth data flow within the application. The ERD suggests that the development team must produce a scalable and maintainable database that can easily handle complex queries from deep learning components for the recommendation system.

Data Flow DiagramA diagram of a diagram

Description automatically generated

Figure 4: Data Flow Diagram

The data flow diagram clearly indicates the architecture of a system and how information circulates between processes, external entities, and data stores. At the heart of the diagram are the management of user accounts, the management of film ratings, the management of personalised recommendations, and the management of detailed film information, all of which are related to administrator statistics. The elements were strategically placed to emphasize the system's core features and their interaction with databases and users.

User account management is the ability to deal with user authentication and preferences, thus laying the basis for personalised services. This is then fed into the process of personalised recommendations, where, with the use of deep learning algorithms, user data is combined with movie metadata and ratings in order to produce tailored movie recommendations.

The development of this diagram serves multiple purposes: it provides a clear blueprint of the system's operations, aids in understanding data flows and process interactions, and most importantly, aids in identifying potential system improvements or bottlenecks. In this way, such a visual representation would be of invaluable value for the developers and stakeholders in understanding the functionality and structure of the system in a succinct manner.

## Test Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature Category** | Feature ID & Name | Objective | Test Cases | Expected Outcome |
| **Must Have** | FR1: User Account Management | Validate account creation, management, and preference tracking. | Account operations; preference tracking and updates. | Successful account operations; accurate preference tracking. |
| **Must Have** | FR2: Movie Ratings | Ensure user ratings accurately influence recommendations. | Rating submission; impact on recommendations. | Ratings affect recommendations as expected. |
| **Must Have** | FR3: Personalized Recommendations | Verify delivery of highly personalized recommendations. | Accuracy and personalization of recommendations. | Recommendations are personalized and relevant, based on user feedback and accuracy metrics. |
| **Must Have** | FR4: Access to Detailed Movie Info | Confirm detailed movie information is accessible and utilized. | Availability and accuracy of movie details; utilization in recommendations. | Movie information is comprehensive and enhances recommendations. |
| **Must Have** | FR5: Administrator Statistics | Check for accurate and actionable administration statistics. | Generation and utility of usage statistics. | Statistics are accurate and inform improvements. |
| **Should Have** | FR6: Movie Search Functionality | Ensure robust and user-friendly search functionality. | Search accuracy and speed. | Searches yield relevant and quick results. |
| **Should Have** | FR7: New and Trending Recommendations | Validate inclusion and accuracy of new and trending movie recommendations. | Timeliness and relevance of new and trending suggestions. | Recommendations include up-to-date and popular movies. |
| **Could Have** | FR10: Social Features | (If implemented) Assess the impact of social features on recommendation precision. | Integration and impact of social features on recommendations. | Social interactions enhance personalization. |
| **Could Have** | FR11: Watchlist Feature | (If implemented) Evaluate the watchlist's functionality and influence. | Adding, modifying, deleting watchlist items; impact on recommendations. | Watchlist is user-friendly and informs recommendations. |
| **Non-Functional** | NFR1: Performance | Ensure system response times are below 2 seconds. | Load and performance testing. | System maintains <2 second response time under load. |
| **Non-Functional** | NFR2: Security and Compliance | Validate adherence to GDPR and data protection laws. | Security vulnerability assessments; compliance checks. | System is secure and compliant with relevant laws. |
| **Non-Functional** | NFR3: Availability | Guarantee system availability at 99.9% outside scheduled maintenance. | Uptime monitoring; redundancy testing. | System achieves near constant uptime. |
| **Non-Functional** | NFR4: Data Integrity | Ensure accuracy of user data and recommendations. | Data validation and integrity checks. | High integrity and accuracy of data and recommendations. |

Figure 5: Test Plan

# Implementation

## System Architecture

### Overview

The recommended movie system is a three-tier structure architecture with a website, a neural network model, and a data management layer, with each part handling some subset of roles in the system efficiently. The website, built with Flask, allows a friendly user interface for things like the submission of movie ratings and the plotting of visual analyses for the user versus predicted ratings. Flask's lightweight and adaptable framework supports dynamic content delivery, user session management, and secure interactions.

PyTorch Lightning powers the model, facilitating operations such as data processing, model training, validation, and prediction execution. This setup provides a clean separation of the machine learning logic from the application logic, aiding maintainability and scalability.

A MySQL database stores data, providing detailed information about movies, genres, users, and various interactions, such as rating and tag relevance. The relational database model guarantees data integrity and supports concurrency in data operations, thereby enhancing the system's responsiveness.

### Data Model Description

The MySQL schema includes several tables designed to store structured data efficiently:

* movies: Stores metadata about each movie, including a unique movie ID, title, release date, popularity, and average rating. It also keeps track of the number of ratings each movie has received, which helps in generating recommendations based on popularity and relevance.
* genres **and** movie\_genres: The genres table lists all possible movie genres. The movie\_genres table links movies to their respective genres, facilitating genre-based recommendations and searches.
* users: Contains user information, including a hashed password for security purposes, supporting system access control and personalized user experiences.
* ratings: Records the ratings that users assign to movies, which are crucial for training the recommendation model and personalizing user recommendations.
* tags **and** tag\_scores: The **tags** table lists descriptive tags for movies. The **tag\_scores** table quantifies the relevance of each tag to each movie, supporting content-based filtering that enhances recommendation precision.

This data model not only supports the functional requirements of the movie recommendation system but also enhances the system's ability to provide insightful analytics and personalized user experiences. The use of foreign key constraints ensures data consistency and integrity across operations, crucial for maintaining a reliable and robust system.

## Data Handling

### Data Collection

The MovieLens dataset (ml-25m) forms the core of our data collection strategy for the movie recommendation system. This extensive dataset is publicly available through the GroupLens website and comprises comprehensive information about movies rated by users over a period spanning from January 1995 to November 2019. The dataset includes 25,000,095 ratings and 1,093,360 tag applications across 62,423 movies, provided by 162,541 users who each rated at least 20 movies. This rich dataset does not include demographic information, maintaining user anonymity using unique user IDs.

#### Dataset Composition:

* **Ratings**: The ratings.csv file contains each user's rating for movies on a 5-star scale with half-star increments.
* **Movies**: The movies.csv file lists movies with their corresponding movie IDs, titles, and genres. Movie titles sometimes include the release year and are typically sourced from The Movie Database (TMDb).
* **Links**: The links.csv file connects MovieLens movie IDs with identifiers from other popular databases like IMDb and TMDb, facilitating cross-referencing and data enrichment.
* **Tags and Tag Genome**: Two files, genome-scores.csv and genome-tags.csv, provide a detailed tag relevance scoring system. "Tag Genome" is a dense matrix representation of how well tags describe the movies, computed using machine learning algorithms based on user-generated content including ratings and textual reviews.

#### Additional Data Sources:

* **TMDb Daily Updates**: For additional metadata, we incorporate data from the TMDb Daily Update Dataset, which offers information on over 700,000 movies. This dataset includes **tmdbId** and other useful metadata. It is important for enriching the movie information in our system, providing users with detailed descriptions and visual cues that aid in movie selection.

### Data Pre-processing

#### Filtering

Movie-related tags not only needed to be cleaned, but they also needed to be heavily filtered so that only the most pertinent tags remained. As a result, the system uses targeted, highest-impact tags in each suggested movie throughout the top-N selection process, leveraging relevance ratings, increasing the recommendation system's accuracy and efficacy.

Films with fewer than ten ratings were removed from the dataset. This has helped to guarantee that there is enough user interaction data for every movie in the collection to enable the derivation of stable and relevant embeddings. Such sparse data may result in poorly formed embeddings, causing overfitting and erroneous recommendations. To train over more dependable and statistically significant data and improve overall effectiveness and efficiency in suggestions, the model primarily considers films with sufficient ratings.

 Consequently, we eliminated the "original" tag from our dataset. The tag was extremely general and applied to many films without providing context or insight into the topics and content of the films. The recommended set of characteristics will become even more focused on distinct and descriptive tags if this tag is removed.

A graph of a number of movies

Description automatically generated

Figure 6: Number of Movies per Tag for Top Ten Tags

#### Data Integration and Consistency

Maintaining the necessary consistency of the datasets is a fundamental difficulty for any data-driven system that integrates several data sources.

We ensured that the movie IDs were consistent throughout all datasets. To achieve this, we remapped movieId and userId in a continuous range by merging TMDB data with MovieLens connections; this is crucial for embeddings as well as database indexing and speed. Additionally, by remapping the data, the perspective from many data sources was unified, which is critical for analytics and accurate machine learning model training.

## Model Implementation

### Architecture

A machine-learning model is at the heart of our movie recommendation system, which uses PyTorch Lightning and the PyTorch framework. When predicting user ratings for films, this model considers input parameters such as movie genres and tags and uses data from both the user and the films.

The model's multi-layered architecture includes a type of attention mechanism that captures and attends to many of the smaller features seen in film quality. We use representations for users, movies, tags, and categories in a shared latent space. We use the attention technique on tags and genres to dynamically weight their relevance based on user preferences and movie metadata.

Fundamentally, the model works by running the embeddings through several layers of a neural network. User and movie embeddings essentially assign each user and movie to a vector in the latent space, as is intuitive. However, attention mechanisms assess the relevance of a tag or genre before propagating it throughout the network.

This is critical because it allows the model to focus on the most important aspects of a film and provide suggestions based on the user's preferences. It concatenates processed embeddings and runs them through a fully connected network to predict a user's movie rating. It minimises the prediction error by comparing it to real ratings during training and adjusting its parameters accordingly.

### Importance of Weights in Model Training

A weighted loss function has been included to the movie recommendation system to address the issue of rating imbalance. All kinds of datasets created from user interactions include rating imbalance, which is the situation where there is a comparatively higher number of certain ratings than other ratings. A rating of 4.0 and 5.0, for example, would be more typical than a rating of 0.5 and 1.0. This discrepancy can lead the model to prioritize accuracy in predicting more frequent ratings at the expense of less frequent, yet equally important, ratings.

A graph of blue bars

Description automatically generated

Figure 7: Bar Chart Showing Number of Ratings per Rating Value

### Calculation of Weights

We then address this imbalance by calculating weights that are inversely proportionate to the frequency of each rating.

# Calculate weights based on the inverse of rating frequencies

rating\_counts = {0.5: 3768, 1.0: 7717, 1.5: 3909, 2.0: 16194, 2.5: 12563, 3.0: 48348, 3.5: 31313, 4.0: 66219, 4.5: 22103, 5.0: 35665}

total\_ratings = sum(rating\_counts.values())

weights = {rating: total\_ratings / count for rating, count in rating\_counts.items()}

Weights are determined by counting each individual rating. Since the total would be divided by the count, ratings with lower occurrence rates would likewise have been assigned higher weights. In actuality, a model of this kind is more susceptible to mistakes when predicting the less common scores.

### Implementation in Training and Validation Steps

The weighted loss is implemented in both training and validation steps as follows:

# Apply weights to the loss

loss = torch.mean(weight \* (predicted\_rating - rating) \*\* 2)

### Impact on Model Performance Metrics

The use of weighted loss not only influences the training process but also the evaluation of model performance. We calculate weighted Mean Squared Error (MSE) and Mean Absolute Error (MAE) as our primary performance metrics:

# Calculate weighted MSE and MAE

weighted\_mse = torch.mean(weight \* (predicted\_rating - rating) \*\* 2)

weighted\_mae = torch.mean(weight \* torch.abs(predicted\_rating - rating))

These criteria will ensure that the model is fair and successful in its forecast for a wide range of user ratings, as well as more accurately depict the model's performance across all rating levels. This approach is crucial to the creation of a strong and dependable recommender system that both accommodates the wide range of user preferences and recognises that every user comment is equally significant for improving the model.

### Overview of Hyperparameter Tuning

Hyperparameter tuning is an essential aspect of machine learning that involves optimizing the model parameters to enhance performance. In our movie recommender system, we employed Bayesian optimization for hyperparameter tuning, which is a probabilistic model-based approach. This method is particularly useful for optimizing complex models like ours, as it efficiently navigates the parameter space to find the best settings based on the model's validation loss.

### Definition of Hyperparameter Space

The space for tuning includes several key parameters that directly influence the model's learning dynamics and performance:

space = [Integer(32, 256, name='embedding\_dim'),

Real(10\*\*-5, 10\*\*0, "log-uniform", name='lr'),

Integer(1, 10, name='step\_size'),

Real(0.1, 0.9, name='gamma')]

### Objective Function for Optimization

The objective function is defined to measure the effectiveness of each parameter configuration by training the model and calculating the validation loss:

@use\_named\_args(space)

def objective(\*\*params):

...

return trainer\_tuning.callback\_metrics['val\_loss'].item()

res\_gp = gp\_minimize(objective, space, n\_calls=10, random\_state=0)

Within this function, we train the model using the specified parameters, returning the validation loss. This loss is then used by the Bayesian optimization algorithm to decide the next set of parameters to evaluate.

This process uses a Gaussian process to model the parameter space and selects the next parameters to evaluate by estimating which might result in the lowest validation loss. This method is particularly efficient for high-dimensional spaces with complex relationships between parameters.

### Model Performance

### Weighted vs. Non-Weighted Model Outcomes

The results indicate significant differences between the weighted and non-weighted models, especially in handling users with different rating behaviours:

Weighted Model: When compared to its non-weighted counterpart, the model's overall performance is mediocre, but it dramatically outperforms for the lowest-averaging users. Therefore, these results may be read as supporting the idea that, in cases where ratings are typically lower and possibly more variable, the weighting approach—which allows less frequent ratings to carry more weight—really helps to improve overall prediction accuracy.

Non-Weighted Model: While this model performs slightly better for users with the most ratings, it significantly underperforms for users with lower average ratings, indicating a possible overfitting to the more frequent higher ratings.

A graph of a number of different colored squares

Description automatically generated with medium confidence

Figure 9: R2 Comparison Chart

### Reflection on Test Results

These findings unmistakably show that the model is capable of independently modifying predictions in light of the biases present in the data.

Both models performed quite similarly for the top users with the highest ratings, despite the fact that the weighted average did not increase as theoretically predicted. Most likely, for these users, a large amount of data dilutes the weight of less frequent ratings.

The non-weighted model's sharp underperformance for users with the lowest ratings is an obvious indicator of the danger of failing to address the imbalance in the data. Furthermore, the R2 takes a negative value, indicating that the predictions are in fact less accurate than a mean-based prediction alone. This emphasises the need for weighting in skewed datasets.

Given that the user with the highest average ratings may not have varied their ratings sufficiently to forecast variations from the mean, the neutral R2 values for this person would suggest that none of the models had a significant enough impact.

### Insights and Future Directions

These findings highlight the necessity of specialised methods for various data sets.

Weighting Scheme Refinement: By adding non-linear scaling variables, it is possible to improve the existing weighting scheme's ability to balance the effects of frequent and infrequent ratings.

Segment-Specific Models: Using either segment-specific models or hyper-parameters tailored to the identified user groups would increase overall accuracy and, consequently, user satisfaction.

Future iterations of this model may include more sophisticated algorithms, such as ensemble techniques or neural collaborative filtering, which are better able to capture subtler patterns in user-item interactions. In retrospect, all of these emphasise how crucial it is to have systematic testing and assessment processes in addition to efficient ones for creating recommendation systems. If we can figure out when and why one strategy could work or not, the better equipped we will be to navigate this maze of complications.

### Model Integration

#### Initial Approach with TorchServe

The original plan was to integrate the machine learning model into the Flask application using TorchServe, which is a more efficient way to serve PyTorch models in a production system. TorchServe facilitates several features, such as model scalability, model versioning, and very efficient concurrent request serving. We configured the system in the following way:

* **Archiving the Model:** Consolidate the PyTorch model into the necessary configuration files, which include variables such as API port settings and unique handlers for preprocessing, inference, and post-processing.
* The Docker container provided TorchServe with a sort of isolation environment, a reduction in deployment complexity, and a guarantee of consistency in certain installations.

Notwithstanding these benefits, there were also notable drawbacks:

* **Complicated Setup:** Because TorchServe provides so many customisation options, configuring it has proven to be a challenging undertaking. Despite its flexibility, there is a steep learning curve and takes a long time to set up.
* **Batch inputs:** After implementing batch predictions, TorchServe encountered difficulties with the amount of data received. Optimising batch processing or increasing resource availability may mitigate the need for additional tweaking and experimentation.

#### Decision to Load the Model Directly in Flask

Due to time constraints, the model was loaded straight into the Flask application, as there was little time to spare on loading it in a more thorough, well-articulated manner. Within the parameters of the project, this gives some immediate benefits but has less scalability than TorchServe:

**Simplicity:** Eliminates the need for additional infrastructure, lowers overall complexity, and eliminates additional possible points of failure.

**Ease of Integration:** Directly integrating the model into the Flask application makes the codebase cleaner and easier to debug. The Flask application should handle all actions, such as loading, inferring, and managing failures, during the model interaction.

**Control and insight:** During the development and early deployment phases of model management, there is clear insight into the performance and behaviour thanks to direct control over the loading and usage of the model.

Considering the project's limitations, simply integrating the model into a Flask application makes sense. This also suggests a lower overhead and simpler implementation for overseeing an independent model-serving infrastructure. Although it is not as scalable as server-side solutions like TorchServe, it nevertheless provides enough functionality to meet the project's present demands and allows for the possibility of revisiting more scalable choices, like TorchServe, as the application's requirements change in the future.

### Prediction For Single Movie

The Flask application's prediction route is an essential interface that provides users with information about how the model makes predictions about movie ratings in this particular scenario. A feature like this might help concentrate on the model's clarity and interpretability, which would greatly boost user participation and trust—two factors that are critical for a successful business operation. Users will be better equipped to relate to or align those insights with their preferences as a result of knowing which elements the model values most highly. This is true not only for the predicted rating, but also for the top-influencing tags and genres.

A screenshot of a video game

Description automatically generated

Figure 10: Predicted Movie Details Page

### Implementation Details

**Attention Mechanism Visualisation:** Using the model's attention mechanism, the predictive function highlights the top three tags and genres that have affected the prediction. This is a crucial step because it gives the model's output an interpretative layer that makes it easier for the user to understand not just what the model predicted but also why. Bar charts provide a clear and effective method for visualising this data, transforming the abstract concept of "attention" in the model into a human-readable form.

# Get top tag and genre indices

top\_tag\_indices = torch.topk(tag\_attn\_weights, k=3, dim=1).indices + 1

top\_genre\_indices = torch.topk(genre\_attn\_weights, k=3, dim=1).indices + 1

#### Rationale Behind the Design Choices

We made the decision to forego additional navigation or an add-on step and instead insert the model predictions directly into the movie information page, which will further encourage user involvement. Integrating predictive insights directly into familiar elements of the application (like movie details) ensures that the functionality is both accessible and practically useful.  
  
The "black box" aspect of prediction models is a frequently raised issue with AI and machine learning. The algorithm attempts to address this by incorporating visualisation for tags and genres. As a result, the user feels more in control and has the opportunity to use the model's predictions to make an even better decision.   
  
This approach demonstrates the effective incorporation of the newest machine-learning technologies into web apps, enhancing not only their functionality but also aspects such as user trust and transparency. Given that the technology is human-centric—that is, it incorporates user engagement and information into the technology itself—it does demonstrate a strong and wide commitment to the predictive use of machine learning.

## Software Testing

The most crucial phase of software development is the testing phase. It ensures the programme functions as needed and serves as a buffer to try to prevent regressions when software is changed. I created a testing plan that covered several aspects of the application's functionality. This covers user interaction with the Flask application, including signing up, logging in, and rating movies.

### Tools and Frameworks Used

unittest: A built-in Python library that supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

Flask-Testing: An extension for Flask applications that adds support for testing. It provides a test client that simulates requests to the application and checks the responses.

### Example

class TestRegisterRoute(BaseTestCase):

def test\_register\_new\_user(self):

with self.client:

# Register a new user

response = self.client.post('/register', data={

'email': 'test@test.com',

'password': 'password123'

})

self.assertStatus(response, 302)

user = User.query.filter\_by(email='test@test.com').first()

self.assertIsNotNone(user)

### Rationale Behind Testing Choices

The chosen testing tools and frameworks enable a comprehensive approach to testing Flask applications. Using unittest and Flask-Testing together provides a robust environment for simulating requests and responses, allowing tests to mimic user behaviour closely.

# Project Evaluation

### Evaluation on Research

The method used to create our movie recommendation system was directly inspired by the advances discussed in the literature review, particularly with regard to the way machine learning may improve the user experience by enabling the display of personalised content. Our system architecture follows the design from simple algorithms to more complex machine learning approaches, as outlined by Mohammad et al. (2022) and Pireci-Sejdiu et al. (2022). It employs both collaborative-based and content-based filtering methods. As a result, these sources have focused on integrating computational intelligence to increase recommendation precision—a concept that influenced the creation of the neural network model that PyTorch Lightning is using in this instance.   
  
Because of this, our hybrid model architecture is best suited for "cold start" and data sparseness problems. According to Yao (2023), they use a large-scale linear model to blend dynamically weighted genre and tag embeddings with user and movie embeddings, summarizing the most recent advances in deep learning for recommender systems. Therefore, this technique adapts the features of content attributes to new ones, enabling the generation of predictions not only on historical data but also on new or sparse data, thereby enhancing the prediction quality for less-rated films or new users.  
  
  
We simulate the integration of attention processes into our model, which is now going through a renaissance in order to concentrate deep learning methods on the most important aspects. Lin and Chi (2019) have noted that we closely monitor this aspect to tailor each recommendation to the user's unique preferences, all while maintaining a broad scope of discovery. This is the deliberate incorporation of literary concepts into our system's architecture, which is a direct application of theoretical development to real-world, practical issues with the movie recommendation system.

### Evaluation on Requirements

The project has successfully implemented some of the main functional requirements of the original plan, including user account management, a system for movie ratings, and a personalised recommendation system. The project incorporated the literature's findings, which suggest that considering user-rated movies in movie recommendations enhances the tailored experience for users through content personalization. In the case of continuous tracking of user ratings until model retraining, the base configuration adheres to the fundamentals of recording user preferences to inform future suggestions.

Personalised predictions are the main component of this system; they generate customised predictions dynamically by utilising movie metadata and user ratings. The project's conceptual stages explored major problems like data sparseness and the cold start problem, aligning with the trend towards sophisticated, deep learning-based systems. By concentrating on these core features, the system adheres closely to the 'Must Have' criteria of making extensive movie information accessible and leveraging it to improve recommendation precision. It also makes good use of the rich dataset to produce suggestions that are more pertinent and accurate.

The MoSCoW model categorises features like watchlist and administrator statistics as "should have" or "could have" needs, which this iteration of the project did not implement. We've taken this action to concentrate efforts on improving the core recommendation engine so that it better serves user demands and meets system performance goals. The system maintains the robustness and consistency of recommendation outputs rather than providing real-time updates or admin functions. The goal of this concentrated streamlining is to make sure that the essential components of the recommendation system are sufficiently established and efficient to provide a solid basis for future improvements.

### Challenges

Since this was my first experience with PyTorch, the process of porting our neural network model provided me with a wealth of learning possibilities. At first, understanding PyTorch's dynamic computation network and how it handles tensors was critical in order to use the tool's capacity to effectively model deep learning systems. It was beneficial to switch to PyTorch Lightning later on, since it made it possible to simplify many basic coding jobs and concentrate more of my efforts on refining the model rather than handling a ton of boilerplate code.

We were able to minimise resource restrictions and waiting periods for model training by using only 10% of the data during the early training stages. This was a critical success element in keeping the project moving forward.

These experiences not only enhanced the project's efficiency but also deepened my understanding of managing and scaling deep learning models in practice.

### Feedback Integration

My supervisor made a point of stressing the importance of incorporating a readily tested recommendation system into our creation of the movie recommendation system. As a result, I implemented a ratings prediction model that is testable with several metrics, including Mean Absolute Error (MAE) and Mean Squared Error (MSE). This allowed for a direct comparison of the expected and actual user ratings, ensuring the accuracy and efficacy of the model.

In an attempt to address criticism on how I could set my system apart from usual recommendation algorithms, which, in reality, seldom present the rating prediction directly to the user, I developed features that would enable this. By highlighting the most significant tags and genres for every suggested film, we were able to provide greater context for the model's predictions. In the user interface, we also incorporated a scatter graph that compares actual user ratings to expected ratings. This not only makes the recommendation process extremely visible, but it also encourages user participation by outlining the rationale behind each proposal. By giving users a greater understanding of suggestion creation, these enhancements not only made the system exceptional but also directly addressed customer input and helped to increase user trust and happiness.

### Improvement Opportunities

In the future, the following methods could be used to further enhance the functionality and sophistication of the movie recommendation system:

One of the most crucial areas for development would be to incorporate real-time data processing. This would allow the system to adjust suggestions dynamically in response to real-time user interactions, thereby enhancing user personalisation and effectively addressing the cold start issue.

More research into more advanced machine learning techniques could greatly enhance the specificity and accuracy of the suggestions. This is because methods that are based on complex patterns in user behaviour and the attributes of movies may make the system more adaptable to a wider range of user profiles.

Additionally, it presents a chance to improve the system's educational value and interaction by giving thorough justifications for the movie recommendations made based on user-specific information. This may have to do with enhancing the visuals to provide more context and a more accurate depiction of the visualisation, which may aid users in discovering their preferences and the factors that influence recommendations.

Additionally, improving the weighted loss function would significantly raise the lower ratings' prediction accuracy. Examining different weighting schemes or, at the very least, adjusting the model to dynamically adjust weights based on real-time feedback—sensitive to the less frequently provided ratings and, consequently, less sensitive to their rating.

# Further Work and Conclusions

### Summary of Project Aims and Achievements

The project's goal was to create an intelligent movie recommendation system that could provide users with tailored suggestions for films based on their interests and input from system interactions by generating predicted ratings. The system uses both content-based and collaborative filtering methods, along with a strong data model and machine learning architecture that are based on a set of functional and non-functional criteria.

Key achievements include:

* **User Account Management:** Enabled dynamic tracking of user ratings, essential for personalizing recommendations.
* **Movie Ratings and Personalized Recommendations:** Integrated user ratings to enhance recommendation accuracy and address challenges such as the cold start problem.
* **Detailed Movie Information:** Facilitated deeper analysis and precision in recommendations through extensive data utilization.

### Discrepancies Between Planned and Actual Outcomes

The project achieved the majority of its main goals; however, owing to time constraints and the need to prioritise essential capabilities, items that had first been included under the "Should Have" and "Could Have" categories had to be postponed until later iterations.

### Growth in Ideas

Many valuable insights regarding the challenges of fusing extremely complex machine learning models with intuitive web interfaces have come from the development process. Practical implementation issues led to a greater understanding of system scalability, data integrity, and the significance of model training that can account for user variability in ratings.

### Further Research

Enhancement of Current Features  
1. **Improved Real-Time Data Processing:** You may greatly increase the responsiveness and personalisation of the system by implementing streaming data capabilities to modify suggestions based on in-the-moment interactions.

2. **Social feature integration:** Create features that employ social interactions to make recommendations, which might improve user interaction and system accuracy.

3. **Advanced Tag and Genre Analysis:** Improve the tag and genre weighting systems to better represent user preferences, thereby deepening the content-based filtering.

#### Development of New Features

1. **Real-time user input Implementation:** Incorporate direct user input methods to enhance model accuracy and dynamically modify suggestions.

2. **Creation of Models Specific to Segments:** Examine the viability of developing recommendation models tailored to individual user segments in order to serve different user groups and improve system satisfaction and performance as a whole.

#### Technical Advancements

1. **Optimising the Machine Learning Model for Scalability:** Concentrate on fine-tuning the model for scalability and efficiency, maybe employing more complex algorithms such as neural collaborative filtering or ensemble approaches.

2. **Improved Visualization Tools:** Offer more user-friendly and educational visualisation tools to assist users in understanding the process of making suggestions, thereby enhancing system transparency and confidence.

### Conclusion

The project successfully established a foundational movie recommendation system with substantial potential for future enhancements. The journey highlighted the critical role of adaptive learning in recommendation systems and the importance of balancing user-centric features with technical robustness. Future developments should aim to not only refine the system's capabilities but also to explore innovative features that continue to push the boundaries of personalized content delivery. ￼

# References / Bibliography

1. Abbas, A., Zhang, L. and Khan, S.U. (2015) A Survey on context-aware Recommender Systems Based on Computational Intelligence Techniques. *Computing* [online]. 97 (7), pp. 667–690. [Accessed 24 August 2021].
2. Afoudi, Y., Lazaar, M. and Al Achhab, M. (2021) Hybrid Recommendation System Combined content-based Filtering and Collaborative Prediction Using Artificial Neural Network. *Simulation Modelling Practice and Theory* [online]. 113, p. 102375.
3. Ahuja, R., Solanki, A. and Nayyar, A. (2019) Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor. *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* [online].
4. Aramuthakannan, S., Ramya Devi, M., Lokesh, S. and Manimegalai, R. (2023) Movie recommendation system via fuzzy decision making based dual deep neural networks. *Journal of Intelligent & Fuzzy Systems* [online]. pp. 1–14. [Accessed 23 January 2023].
5. Chen, J., Zhou, X. and Jin, Q. (2012) Recommendation of optimized information seeking process based on the similarity of user access behavior patterns. *Personal and Ubiquitous Computing* [online]. 17 (8), pp. 1671–1681. [Accessed 30 April 2020].
6. Chen, W., Cai, F., Chen, H. and Rijke, M.D. (2019) Joint Neural Collaborative Filtering for Recommender Systems. *ACM Transactions on Information Systems* [online]. 37 (4), pp. 1–30.
7. Chen, Y.-L., Yeh, Y.-H. and Ma, M.-R. (2021) A Movie Recommendation Method Based on users’ Positive and Negative Profiles. *Information Processing & Management* [online]. 58 (3), p. 102531.
8. Codina, V., Ricci, F. and Ceccaroni, L. (2015) Distributional semantic pre-filtering in context-aware recommender systems. *User Modeling and User-Adapted Interaction* [online]. 26 (1), pp. 1–32. [Accessed 25 May 2019].
9. Eyad Kannout, Marek Grzegorowski, Grodzki, M. and Hung Son Nguyen (2024) Clustering-based Frequent Pattern Mining Framework for Solving Cold-Start Problem in Recommender Systems. *IEEE Access* [online]. pp. 1–1. [Accessed 3 February 2024].
10. Jayalakshmi, S., Ganesh, N., Čep, R. and Senthil Murugan, J. (2022) Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. *Sensors* [online]. 22 (13), p. 4904.
11. Karabila, I., Darraz, N., El-Ansari, A., Alami, N. and El Mallahi, M. (2023) Enhancing Collaborative Filtering-Based Recommender System Using Sentiment Analysis. *Future Internet* [online]. 15 (7), p. 235. Available from: https://www.mdpi.com/1999-5903/15/7/235.
12. Kondepudi Yasaswi, Sai, L., Ch. Vyshnavi, Safia Begum M and Jonnalagadda Surya Kiran (2022) Movie Recommendation System based on user’s search history using incremental clustering. [online]. [Accessed 3 February 2024].
13. Lavanya, R., Gogia, E. and Rai, N. (2021) Comparison Study on Improved Movie Recommender Systems. *Webology* [online]. 18 (Special Issue 04), pp. 1470–1478. [Accessed 30 May 2022].
14. Li, J., Li, C., Liu, J., Zhang, J., Zhuo, L. and Wang, M. (2019) Personalized Mobile Video Recommendation Based on User Preference Modeling by Deep Features and Social Tags. [online]. 9 (18), pp. 3858–3858. [Accessed 24 May 2023].
15. Li, M., Zeng, Y., Guo, Y. and Guo, Y. (2020) The Movie Recommendation System Based on Differential Privacy. *Communications in Computer and Information Science* [online]. 1298, pp. 318–328. [Accessed 3 February 2024].
16. Lin, C.-H. and Chi, H. (2019) A Novel Movie Recommendation System Based on Collaborative Filtering and Neural Networks. *Advanced Information Networking and Applications* [online]. 926, pp. 895–903.
17. Mohammad, J.F. and Urolagin, S. (2022) Movie Recommender System Using Content-based and Collaborative Filtering. *IEEE Xplore* [online]. pp. 963–968. Available from: https://ieeexplore.ieee.org/document/9872515 [Accessed 27 April 2023].
18. Mu, Y. and Wu, Y. (2023) Multimodal Movie Recommendation System Using Deep Learning. *Mathematics* [online]. 11 (4), p. 895. Available from: https://www.mdpi.com/2227-7390/11/4/895?type=check\_update&version=1.
19. Park, H., Yong, S., You, Y., Lee, S. and Moon, I.-Y. (2022) Automatic Movie Tag Generation System for Improving the Recommendation System. *Applied Sciences* [online]. 12 (21), p. 10777. Available from: https://www.mdpi.com/2076-3417/12/21/10777 [Accessed 25 April 2023].
20. PireciSejdiu, N., Blagoj Ristevski and Ilija Jolevski (2022) Performance Comparison of Machine Learning Algorithms in Movie Recommender Systems. *2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)* [online].
21. Qin, Z. and Zhang, M. (2021) Towards a Personalized Movie Recommendation System: a Deep Learning Approach. *2021 2nd International Conference on Artificial Intelligence and Information Systems* [online].
22. Srikanth, P., E. Ushitaasree, Sai and G. PaavaiAnand (2021) Movie Recommendation System Using Deep Autoencoder. *2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA)* [online]. [Accessed 3 February 2024].
23. Tahmasebi, H., Ravanmehr, R. and Mohamadrezaei, R. (2020) Social movie recommender system based on deep autoencoder network using Twitter data. *Neural Computing and Applications* [online].
24. Valliyammai, C. and Ephina Thendral, S. (2019) Ontology Matched Cross Domain Personalized Recommendation of Tourist Attractions. *Wireless Personal Communications* [online]. [Accessed 20 May 2019].
25. Widiyaningtyas, T., Hidayah, I. and Adji, T.B. (2021) User profile correlation-based similarity (UPCSim) algorithm in movie recommendation system. *Journal of Big Data* [online]. 8 (1).
26. Yao, Z. (2023) Review of Movie Recommender Systems Based on Deep Learning. Cheo-Chun, S. and Belém Nunes, A.M., eds. *SHS Web of Conferences* [online]. 159, p. 02010. [Accessed 28 February 2023].
27. Yassine, A., Mohamed, L. and Al Achhab, M. (2021) Intelligent recommender system based on unsupervised machine learning and demographic attributes. *Simulation Modelling Practice and Theory* [online]. 107, p. 102198. [Accessed 13 May 2021].
28. Zhang, F., Zeng, Q., Lu, L. and Li, Y. (2021) Sentiment Analysis of Movie Reviews Based on Deep Learning. *Journal of Physics: Conference Series* [online]. 1754 (1), p. 012234. [Accessed 16 November 2021].