UNIVERSITY OF LONDON

BSc Computer Science (Machine Learning and Artificial Intelligence)

CM3070 PROJECT
FINAL PROJECT REPORT

PERSONALIZED MOVIE RECOMMENDATION SYSTEM

Table of Contents

Chapter 1: Introduction

Project Template: Deep Learning on a Public Dataset (CM3015 Machine Learning and Neural Networks)

Project Title: Personalized Movie Recommendation System

Project concept

Recommender systems are intelligent algorithms that analyze user preferences and behavior. The purpose of these systems is to assist users in discovering relevant items or content, through which the system enhances user experience and boosts customer satisfaction. The aim of this project is to develop a personalized movie recommender system that takes advantage of these intelligent algorithms to enhance the movie selection process. By utilizing existing techniques and frameworks in recommendation systems, the recommendation system will provide tailored movie recommendations based on user preferences and behaviors. While it may be ambitious to revolutionize the entire movie selection process due to the limitations of time and available resources, this project serves as a vital stepping stone towards exploring the potential for improvement and refinement in this domain. The primary focus will be on implementing essential features and functionalities while ensuring a robust and functional movie recommender system that adds value to the recommendation experience. Additionally, the project will lay the groundwork for future exploration and enhancements beyond the project timeline, offering potential avenues for further research and development.

Supporting Motivation

The motivation behind this project arises from personal experiences and a broader recognition of the challenges users confront in the contemporary digital landscape. As a movie enthusiast, the arduous task of finding a suitable film has often surpassed the actual pleasure of watching it. Existing OTT platforms have frequently failed to provide recommendations that align with my personal preferences. This led to a frustrating and time-consuming search process. This personal struggle sparked an interest in exploring ways to enhance recommendation systems by offering users a more personalized and gratifying experience.

The issues described above are not unique to me but are also faced by many users worldwide. A journal titled "Personalized Recommendation Algorithm for Movie Data Combining Rating Matrix and User Subjective Preference" revealed that users frequently spend considerable time searching for movies of interest, resulting in tedious and frustrating experiences. [1] This study also demonstrated that personalized recommendation systems increased users' satisfaction from 20% to 50% and decreased dissatisfaction from 15% to 8%. [1]

The article above also emphasized the problem of information overload. This is one of the main reasons that leads to the considerable increase in the time to find a movie to watch, as users often find themselves overwhelmed with the vast number of options available. The discovery of diverse movies is also significantly impacted due to the problem of information overload. Due to the overwhelming number of options, users often stick to familiar genres, directors, or actors, causing them to miss out on a wide array of diverse and potentially

enriching content. This phenomenon is known as the "filter bubble" effect, where users are often exposed to a limited perspective based on their previous choices and preferences. [2] This not only restricts the exposure to diverse content but also hinders the growth of lesser-known or indie films that might align with the user's taste but are not presented due to the lack of visibility in the recommendation system.

Another study titled "Improving User Experience with Recommender Systems by Informing the Design of Recommendation Messages" identified that the level of user engagement also influences the effectiveness of movie recommendation systems in providing feedback. [3]
This study revealed that increasing the specificity of problem and solution-related information within recommendations enhances user intention and acceptance of the recommendations while reducing decision-making time. Additionally, presenting recommendations in a problem-to-solution sequence contributes to shorter decision-making times. The study also confirms the connections between user beliefs, attitudes, and behavioral intentions, emphasizing the importance of information sufficiency and transparency. To address the problem mentioned above, this project will incorporate nuanced user feedback on specific movie genres, such as humor in an action film, to recommend similar films within those genres.

Key Research Questions

1) **Balancing personalization and filtering:** How to achieve the optimal level of constraint in content filtering to provide tailored movie recommendations while still allowing for exploration and discovery?

2) **Effectiveness of standard practices:** To what extent do movie metadata, genre tags, and user preferences analysis contribute to accurately understanding user preferences and improving recommendation quality?

3) **Impact of user feedback:** What role do user reactions and feedback, such as ratings and reviews, play in predicting outcomes and influencing subsequent recommendations? How can user feedback be leveraged to enhance the recommender system's user-centric approach?

Project Deliverables

The project deliverables include a baseline model that will serve as a benchmark for more complex models in the personalized movie recommender system. These complex models will incorporate item-based filtering, considering genres to provide more accurate recommendations. The models will be evaluated using metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), and the best-performing model will be selected for further fine-tuning. Additionally, the fine-tuned model will undergo evaluation with real users, including myself and selected friends and family, to assess its effectiveness in recommending movies. These deliverables aim to provide a comprehensive evaluation and improvement process for the movie recommender system while ensuring its accuracy and suitability for users' preferences.

Chapter 2: Literature Review

(CHAPTER PAGE COUNT EXCLUDING IMAGES – 5)

Literature 1: How Streaming Services Use Algorithms [4]

Objective: Comparative Analysis of Recommendation Models Used by Existing Streaming Platforms

Over-The-Top(OTT) streaming platforms like Netflix, Disney+, Hulu, and HBO Max have entirely changed how we enjoy media. They provide a wide range of content at our fingertips, offering convenience and the freedom to watch what we want when we want. However, with so much content available, finding something that truly matches the user's preferences and interests can be challenging. According to consumer research conducted by Netflix in 2015, Netflix estimated that a subscriber loses interest after 60 to 90 seconds of browsing before they choose something or abandon the streaming platform. This is where recommendation systems come into play. [4]

The article "How Streaming Services Use Algorithms" provides an in-depth analysis of how various OTT platforms utilize these algorithms for content recommendation. It discusses the strategies used by Netflix, Hulu, HBO Max, and other platforms and highlights the potential downsides of relying solely on algorithms for content suggestions.

The article explains that most recommendation systems can be categorized into three types: Content-Based, Collaborative Filtering, and Knowledge-Based. Content-based algorithms use the attributes of an item (like metadata, tags, or text) to make recommendations similar to items the user has previously interacted with. Collaborative filtering uses the interests and behaviors of other users with similar tastes to make recommendations. Knowledge-based methods use an item's attributes and correlate them with a user's preferences to make recommendations based on similarities. This method is beneficial for discovering new content as it is not based solely on past behaviors.

Netflix, a prominent OTT platform, utilizes a combination of sophisticated algorithms to enhance user experience and satisfaction. These algorithms are employed in various platform aspects, such as page construction, genre rows, trending videos, order of videos, and even icons. By leveraging algorithms like the Personalized Video Ranker (PVR), Top-N Video Ranker, Trending Now, Continue Watching, Video-Video Similarity, and the Page Generation algorithm, Netflix personalizes the content displayed to each member. The effectiveness of these algorithms is demonstrated by their impact on decreasing churn rate and saving billions of dollars annually. An image depicting how these algorithms work together is given below to visualize the interplay of Netflix's algorithms. [4]

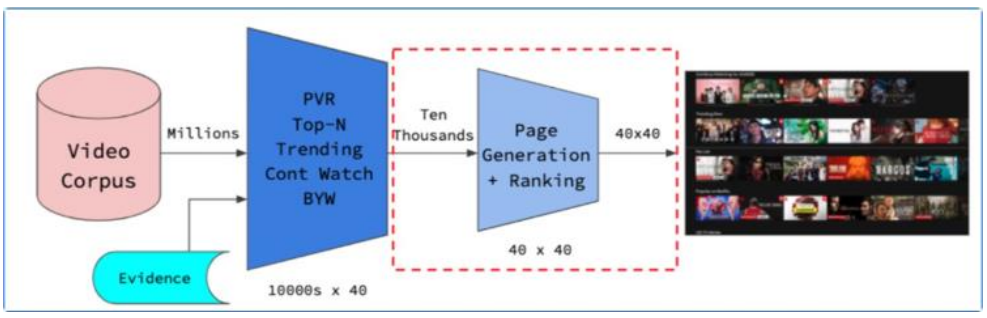


Figure 1 – Netflix algorithm

A potential downside of solely relying on algorithms discussed in the article is the presence of bias in algorithmic recommendations. Despite efforts to mitigate bias, algorithms built on human data can inadvertently perpetuate prejudices or assumptions. Netflix faced criticism for racially biased image recommendations targeted at certain users, highlighting the risk of biased inferences. Additionally, algorithms lack creativity and decision-making autonomy, relying on flawed data inputs and struggling to interpret complex human preferences accurately. An image depicting the problems with content recommendations generated via AI on streaming services, based on consumer feedback in the United States in October 2018, is shown below. [4]

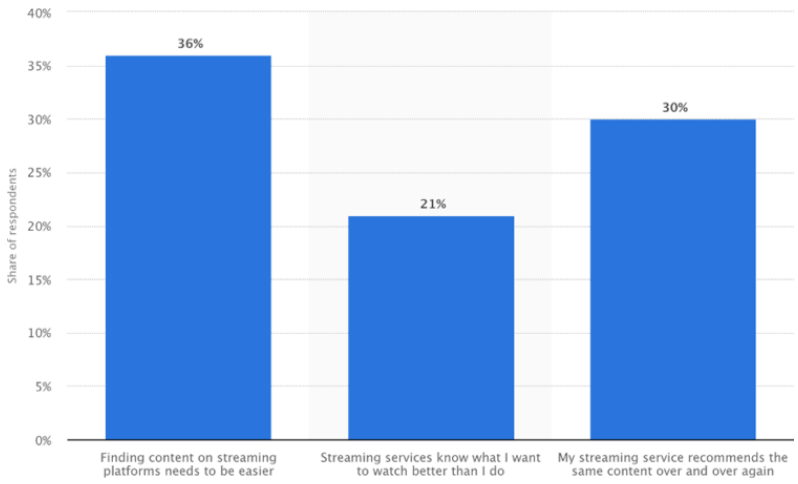


Figure 2 – Consumer feedback on problems with AI content recommendation

Alternative approaches to content curation have emerged, such as human curation models adopted by HBO Max, Hulu, and Criterion. These models aim to break feedback loops by incorporating peer suggestions and celebrity endorsements, offering users a more comprehensive range of recommendations. Cross-platform algorithms or recommendation systems that effectively help users navigate content across multiple services should be explored. However, the effectiveness of this hybrid approach in improving user satisfaction and content discovery is yet to be fully understood and requires further research.

In conclusion, algorithms are instrumental for content recommendation on OTT platforms, but they have limitations, such as creating echo chambers and biases. Platforms like Netflix and Hulu employ a combination of algorithms and human curation to enhance user experience. The ongoing efforts in this field highlight the need for further exploration and development of more sophisticated and unbiased algorithms. This literature review provides insights for my project, and the limitations will be addressed by striking a balance between mitigating bias and exploring alternative approaches. The article is published by researchers from Carnegie Mellon University's Master of Arts Management program. The articles and reports published on this site are thoroughly reviewed before they are posted. Hence, giving credibility to the information presented.

Literature 2: Recommendation System Using Autoencoders [5]

(Diana Ferreira, Sofia Silva, António Abelha, and José Machado, 2020)

Objective: Exploring the Technique of Autoencoders

Collaborative filtering is a popular technique used in recommendation systems, where users' preferences and behaviors are analyzed to make recommendations. It operates under the assumption that users who agreed in the past will agree in the future and will like similar items as they liked in the past. [6] While this technique has proven effective, deep learning has introduced new methods of building more robust and accurate recommendation systems.

In the paper titled "Recommendation System Using Autoencoders," the researchers proposed one such approach called autoencoders to build a product recommendation system. Autoencoders are a type of artificial neural network designed to learn efficient representations of input data. Autoencoders consist of three layers: an input, a hidden, and an output layer. The input and hidden layers form the encoder, compressing input information into a different latent space. The hidden and output layers form the decoder, reconstructing the original information from the latent space. The autoencoder aims to minimize the error measure between the original data and the reconstructed data. [7] A diagram representing a typical structure of an autoencoder is shown below. [5]

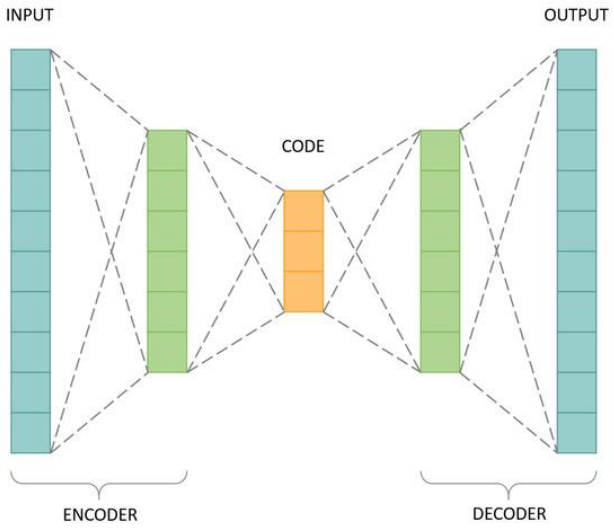


Figure 3 – Typical Structure of an autoencoder

Given the researchers' discussion on parameter selection for the autoencoder, they highlight the challenges of choosing appropriate settings for the loss function, optimizer, and activation function. Through trial and error, they identified the optimal parameters for their model, which involved utilizing the Mean Square Error (MSE) as the loss function, ADAM as the optimizer, Tanh for the encoder layer, and Linear for the output layer. Considering the researchers' successful results and high scores achieved using these metrics, I will adopt the same evaluation metrics and parameters to assess the performance of my model except for using ReLU for my encoder layer. The results they achieved for their Singular Value Decomposition (SVD) model, their comparison model, and their results for their autoencoder model are given below. [5]

	RMSE	
	ML-1M	**ML-10M**
SVD	0.184	0.176
Autoencoder	0.029	0.010

Figure 4 – Results of SVD and autoencoder models

However, the researchers also acknowledge that collaborative filtering, which their model is based on, has limitations. It does not work well without a reasonable amount of data and struggles with users who have very different preferences from others. It also relies heavily on explicit user feedback, such as product reviews. Hence, the researchers suggest exploring hybrid filtering and testing the model with new datasets for future work.

In conclusion, the researchers suggest that using autoencoders for recommendation systems shows excellent potential. Despite some limitations, they found that their autoencoder model outperformed the SVD model in terms of RMSE values. Based on the advantages discussed in the paper, using stacked autoencoders as the primary model to build a movie recommendation system seems like a promising approach. Autoencoders can effectively learn the non-linear user-item relationship and encode complex abstractions into data representations, improving the recommendations' accuracy. They also perform well with large datasets, which is beneficial given the vast amount of movie data available. However, the potential limitations and challenges highlighted in the paper will be considered to ensure the robustness and scalability of the developed system.

Literature 3: Personalized Movie Recommendation Method Based on Deep Learning [8] (Jingdong Liu, Won-Ho Choi, and Jun Liu, 2021)

Objective: Exploring a similar project to gain valuable techniques and results.

In the realm of recommendation systems, deep learning techniques have been gaining traction due to their ability to handle complex data and generate more accurate recommendations. One such technique is using Seq2Seq (Sequence to Sequence) models, typically used in machine translation tasks but have also found their application in recommendation systems. [9]

In the "Personalized Movie Recommendation Method Based on Deep Learning" paper, the researchers proposed a personalized movie recommendation system based on a Seq2Seq model. The Seq2Seq model is a type of recurrent neural network (RNN) that uses an encoder-decoder structure. The encoder processes the input sequence and compresses the information into a context vector. The decoder then uses this context vector to generate the output sequence. The researchers used the Long Short-Term Memory (LSTM) variant of RNNs for both the encoder and decoder due to its ability to handle long sequences and mitigate the vanishing gradient problem common in traditional RNNs. A diagram representing the structure of a Seq2Seq model is shown below. [8]

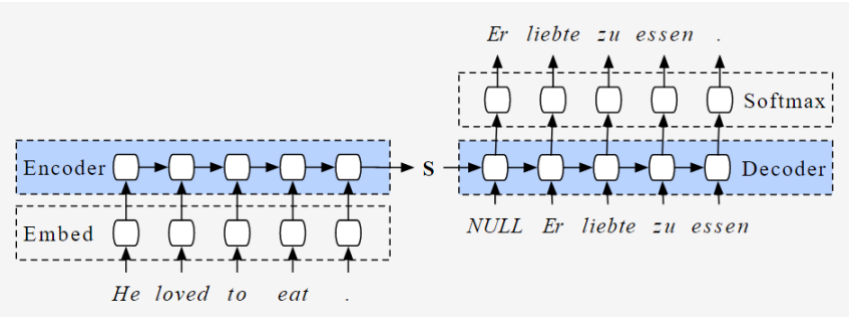


Figure 5 – Structure of a Seq-2-Seq model

The researchers used movie titles and their associated metadata to train their model. They constructed short texts from movie titles of the same director and input these into the Seq2Seq model. The output at the last time step was taken as the interest feature vector of the current audience. This vector was then used to calculate similarity with other movie titles, and the results were integrated into the Lucene sorting algorithm to generate personalized search results. The researchers also incorporated an attention mechanism into their model, allowing the decoder to use different intermediate vectors at each time step, thereby improving its ability to capture relevant information.

The researchers conducted experiments to compare the performance of their Seq2Seq model with traditional collaborative filtering algorithms. They found that their model achieved high accuracy on the training set (96.27%) and the test set (95.89%). This indicates that the Seq2Seq model can effectively provide personalized movie recommendations. The results of their experiments are given below. [8]

Table 7: Loss function and accuracy rate changes.

Epoch	Loss training	Loss test	Accuracy training (%)	Accuracy test (%)
0~3	0.0152	0.0146	71.24	64.14
4~6	0.0131	0.0137	79.56	75.63
7~9	0.0127	0.0121	82.31	80.41
10~12	0.0114	0.0117	85.47	82.67
13~15	0.0097	0.0106	90.26	89.17
16~18	0.0086	0.0095	92.51	93.67
19~21	0.0082	0.0089	93.38	94.59
22~24	0.0078	0.0074	95.46	95.21
25~27	0.0073	0.0069	96.27	95.89

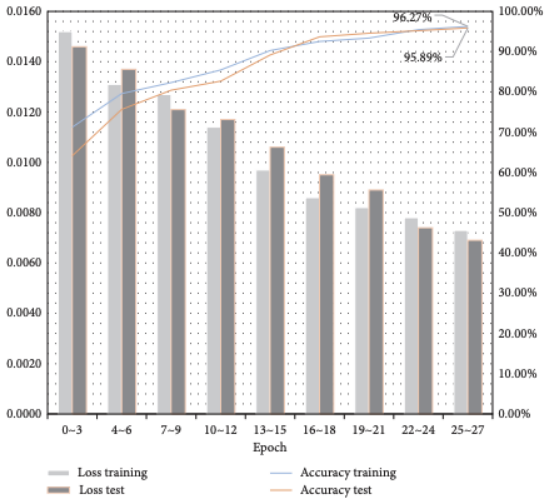


Figure 6: Loss function and accuracy rate changes.

Figure 6 – Experimentation results

Despite achieving excellent scores with their Seq2Seq model, they acknowledged that their model has limitations. The first limitation highlighted was that their model requires a large amount of data to train effectively, and the quality of the recommendations depends on the quality of the input data. Another limitation highlighted was that the model's performance could be affected by the choice of parameters, such as the learning rate and the number of LSTM neurons.

In conclusion, the research paper suggests that using Seq2Seq models for personalized movie recommendations holds great promise, showcasing superior accuracy compared to traditional collaborative filtering algorithms. While this approach shows potential, I have decided not to implement the Seq2Seq model in my project due to its complexity and resource requirements.

However, the findings of this paper provide valuable insights for future studies, where the feasibility and scalability of integrating Seq2Seq models into recommendation systems can be further explored. Additionally, despite not implementing the Seq2Seq model in my project, reviewing this paper has provided valuable information on various aspects, such as techniques, preprocessing methods, and insights into the challenges and potential solutions in personalized movie recommendations. By studying this paper, I have gained a deeper understanding of the current state-of-the-art approaches in the field, allowing me to make informed decisions and potentially leverage certain aspects or insights in my project. This research paper has enriched my knowledge and provided a valuable foundation for exploring alternative techniques and methodologies in developing my personalized movie recommendation system.

Literature 4: Forecasting Movie Rating Using K-Nearest Neighbor-Based Collaborative Filtering [10]

(Prakash Pandharinath Rokade, PVRD Prasad Rao, Aruna Kumari Devarakonda, 2022)

Objective: Exploring the Use of Hybrid Algorithms in Movie Recommendation Systems

In the paper titled "Forecasting Movie Rating Using K-Nearest Neighbor Based Collaborative Filtering," the researcher proposes a movie rating system using a k-nearest neighbor (KNN-based) collaborative filtering (CF) approach. He discusses the importance of recommendation systems in today's digital age, where the demand for personalized content is increasing. Movie recommendation systems, in particular, have become an exciting area of research, with numerous machine-learning algorithms being used to design these systems. However, the researcher acknowledges that several challenges, such as sparsity problems, cold start problem, scalability, and grey sheep problem, still exist and need to be addressed using hybrid algorithms. [10]

The researcher's proposed system compares users' ratings for different movies to get top K users. Then, the top K set is used to find missing ratings by a user for a movie using CF. This proposed system shows promising results for movie recommendations compared to existing systems. A diagram representing the architecture of the proposed method is shown below. [10]

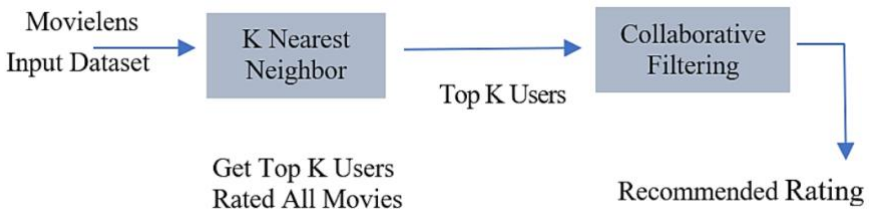


Figure 7 – Architecture of proposed method

The researcher also discusses several research challenges and objectives in designing a better recommender system. These include addressing sparsity problems, where the quality of a recommendation depends on data sparsity, and the cold start problem, which occurs when new users or items enter the system. The researcher also highlights the issue of scalability, where the accuracy of results reduces as the dataset increases, and the grey sheep problem,

where a user's inconsistent rating of related items can degrade the performance of the recommendation system. [10]

In conclusion, the researcher suggests that using a KNN-based CF approach for movie recommendation systems holds great promise. Despite the challenges discussed, the author found that their proposed system effectively predicted ratings for unrated items by considering the top K user set with their ratings. The researcher also emphasizes that selecting the proper value of K improved the prediction result and resolved both the issue of underfitting and overfitting. [10]

This paper provides valuable insights into using hybrid algorithms in movie recommendation systems. As discussed in the paper, the advantages of using a hybrid model have caught my attention. Therefore, I will consider exploring the possibility of incorporating a hybrid model or evaluating a different model that is more appropriate for comparison against my primary model, stacked autoencoders. This will allow me to thoroughly assess the performance and effectiveness of different recommendation approaches and make informed decisions in developing my personalized movie recommendation system. By leveraging the knowledge gained from this research paper, I can further enhance the robustness and suitability of my project, ensuring it aligns with the latest advancements in the field.

Chapter 3: Project Design

(CHAPTER PAGE COUNT EXCLUDING IMAGES – 4)

Domain and Users

The focus of this project lies in the domain of movie recommendation systems, specifically within the realm of Over-The-Top (OTT) streaming platforms. The goal is to enhance the movie selection experience by offering personalized recommendations tailored to individual user preferences and behaviors. With a vast collection of movie titles spanning various genres, this domain presents a complex and diverse landscape to navigate. To tackle this challenge, the project will leverage the power of machine learning and deep learning techniques to provide precise and customized movie suggestions.

The users of this project encompass movie enthusiasts who regularly utilize OTT platforms for their entertainment needs. From occasional viewers to passionate movie buffs, these users seek a solution to the overwhelming task of finding movies that align with their preferences. Additionally, the project aims to cater to users who desire to explore new content beyond their usual choices but face difficulties due to the limitations of existing recommendation systems.

This project also holds relevance for smaller OTT platform providers or emerging players in the market. These providers can benefit from enhancing their user experience through more accurate and personalized movie recommendations. By improving the recommendation process, these providers have the potential to boost user engagement, reduce customer attrition, and ultimately expand their subscriber base.

Design Choices and Overall Structure

The design decisions made for this project are rooted in a deep understanding of the movie recommendation system domain and the users' needs. In movie recommendations, dealing with vast data, including user preferences, movie metadata, and user-item interactions, is essential. The users, who are movie enthusiasts seeking personalized recommendations, are at the core of the design choices. The project's structure is crafted to provide a systematic and efficient approach to building a personalized movie recommendation system.

The project's initial phase focuses on data preprocessing and exploratory data analysis (EDA). This phase plays a crucial role in comprehending the dataset, identifying patterns, and making informed decisions for subsequent steps. It involves tasks such as understanding different dataset features, checking for duplicates and gaining insights into feature distributions. This meticulous phase ensures that the data is clean and ready for modeling, enhancing the subsequent models' accuracy and reliability. Libraries such as Python and pandas will be used in this phase for creating dataframes and using methods like .nunique() to get unique values. Python's built-in functions will be utilized to find and remove duplicates. Seaborn and matplotlib libraries will be employed for data visualization.

The second phase of the project is the model-building phase. This phase starts with building a baseline model. In this case, the Singular Value Decomposition (SVD) model is chosen as the baseline model. The choice of SVD as the baseline model is justified by its balance between simplicity and effectiveness, serving as a reasonable foundation and a benchmark for performance evaluation. The Surprise library will be used for building the SVD model. Building

upon this baseline, the primary model for the project is a stacked autoencoder. This model is selected for its capability to capture non-linear user-item relationships and encode intricate abstractions into data representations. Its effectiveness with large datasets aligns perfectly with the domain of the project. PyTorch, a powerful library for deep learning models, will be used to build the stacked autoencoder. In addition to the stacked autoencoder, the project will incorporate an SVD++ model, a Restricted Boltzmann Machine (RBM), and a hybrid algorithm to enhance the model-building process further.

The final phase focuses on model evaluation and fine-tuning. The best model will be selected based on a comparison of evaluation metric scores like MAE and RMSE, and it will undergo fine-tuning for optimal performance. Additionally, the best model undergoes evaluation by incorporating real reviews from myself and selected friends and family. This approach ensures that the model is statistically robust and validated by genuine user feedback.

After the final phase, future work will be discussed, including areas of improvement and exploration for further development. This discussion is crucial as it provides a roadmap for potential enhancements and expansions of the project, ensuring its continuous evolution and adaptation to user needs.

In conclusion, the design choices for this project are thoughtfully made to enhance the user experience and deliver accurate, personalized movie recommendations. The systematic approach, encompassing data preprocessing, model building, and evaluation, ensures a robust and practical design that caters to the users' needs and the requirements of the movie recommendation domain. Using libraries such as Python, pandas, seaborn, matplotlib, Surprise, and PyTorch will further enhance the project's efficiency and effectiveness. A high-level workflow chart is given below to help visualize the project's overall structure.

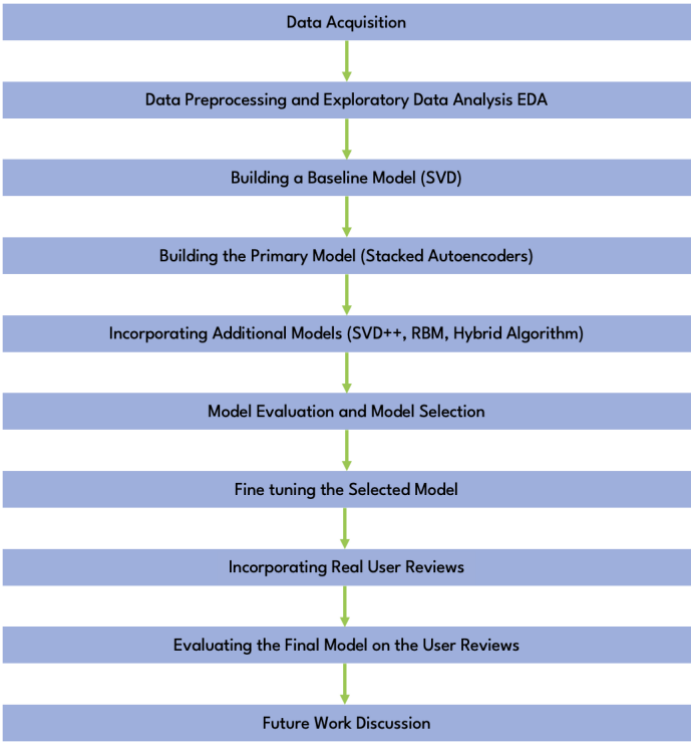


Figure 8 – High-level project workflow chart

Workflow

A Gantt chart explaining the project's workflow is given below in Appendix A1 and Appendix A2. The project was completed on the 5th of September.

Technologies and Methods

Python

Python serves as the primary programming language for this project, offering a diverse set of libraries and tools that streamline data preprocessing, modeling, evaluation, and visualization tasks. Its extensive ecosystem empowers seamless implementation of various functionalities, enhancing the overall project workflow. Python's versatility and extensive ecosystem make it ideal for building a personalized movie recommendation system. Throughout the project, various Python libraries will be employed to streamline the development process.

Singular Value Decomposition (SVD) and SVD++

The Surprise library, a Python library designed to build and evaluate recommender systems, will be used to implement the SVD and SVD++ models. The decision to use the Surprise library instead of manual implementation is based on several factors. Firstly, the library provides optimized implementations of collaborative filtering algorithms, including SVD and SVD++, which significantly improves performance. Secondly, the Surprise library simplifies the model creation, evaluation, and parameter tuning process, allowing for a more streamlined development workflow. Lastly, the library integrates seamlessly with other data preprocessing and evaluation tools, providing a comprehensive solution for building the baseline model. SVD is chosen as the baseline model due to its simplicity and effectiveness in capturing latent factors and underlying patterns in user-item interactions. The SVD model will be used to predict the ratings, and a manual function will be written to recommend movie titles to users based on the decreasing order of the predicted ratings. SVD++ is an extension of the SVD model that incorporates implicit feedback, such as user-item interactions, and the explicit ratings used in SVD. By incorporating implicit feedback, SVD++ can potentially improve the accuracy of the recommendation system by capturing more nuanced user-item interactions. The performance of all models will be evaluated using metrics such as MAE and RMSE.

Stacked Autoencoders

The PyTorch library, a popular deep-learning framework, will be used to implement the stacked autoencoder model. The process of implementing the stacked autoencoder involves several steps. Firstly, the dataframe will be converted into a sparse matrix representation suitable for efficiently processing large datasets. Missing values in the sparse matrix will be filled with zeros to ensure compatibility with the autoencoder architecture. Next, the sparse matrix will be converted into a tensor, which is the fundamental data structure used in PyTorch for deep learning models. The architecture of the stacked autoencoder will be designed, consisting of multiple hidden layers and an output layer. The model will be trained using backpropagation and optimization algorithms to minimize the reconstruction error. After training, the stacked autoencoder will be tested and evaluated using appropriate metrics to assess its performance in generating personalized movie recommendations. Like the baseline model, the stacked autoencoder model will also be used to predict ratings and recommend movies to the user. Its performance will be evaluated using the MAE and RMSE metrics, and the model's output will be compared against the baseline model.

Restricted Boltzmann Machines (RBM)
RBMs are generative models that have shown effectiveness in collaborative filtering tasks. They capture complex user-item relationships by learning hidden representations. They will use the same data preprocessing and modeling steps the Stacked Autoencoders used. Like the Stacked Autoencoders, the RBM will be implemented using PyTorch, which offers flexible and efficient tools for constructing deep learning models.

Hybrid Model
Alternatively, a hybrid model that combines the strengths of different models will be implemented. Hybrid models often integrate collaborative filtering techniques with content-based approaches or utilize ensemble methods to leverage the advantages of multiple models. Implementing a hybrid model can involve integrating the outputs or features from different models and training a meta-model that combines their predictions.

Evaluation Strategy
The evaluation strategy for the movie recommendation system will involve comparing the performance of different models using evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). These metrics provide insights into the predictive accuracy of the models and help determine which model performs best in minimizing the difference between predicted and actual ratings.

MAE measures the average absolute difference between predicted and actual ratings. It provides a straightforward interpretation of the average prediction error, indicating how close the model's predictions are to the ground truth ratings. RMSE, on the other hand, measures the square root of the average squared difference between predicted and actual ratings. It amplifies more significant prediction errors and is more sensitive to outliers. [12]

Both MAE and RMSE are suitable evaluation metrics for this movie recommendation system as they capture the prediction accuracy and quantify the model's ability to estimate user ratings for movies. The most accurate and effective model that minimizes the difference between predicted and actual ratings can be determined by comparing the performance of different models using these metrics. This enables the selection of the best model that delivers precise predictions, enhancing the overall quality of the recommendation system.

In addition to the quantitative evaluation, real user feedback will be incorporated to assess the effectiveness of the recommendation system. This evaluation involves selecting a group of users who will provide movie ratings for movies that they are familiar with.

For each user, their ratings for the selected movies will be collected. The final tuned model will then be utilized to predict the ratings for these movies based on the user's historical data and preferences. The recommended movies will be generated based on these predicted ratings. A questionnaire or survey will be created using a form to gain insights into the user's perception and assess if the model recommendations align with their preferences.

Users will be requested to provide feedback on the recommendations, expressing their likes, dislikes, and suggestions for improvement. Valuable insights into how the system performs, its understanding of user preferences, and areas that can be further improved can be assessed

from this qualitative feedback. This feedback will be instrumental in refining and fine-tuning the recommendation system to better meet the users' needs and expectations. Keeping ethical considerations in mind, all user information will be anonymized when presenting the results to protect the participant's identity.

CHAPTER 4: Implementation

(CHAPTER PAGE COUNT EXCLUDING IMAGES – 3)

Data Enrichment and Refinement

Utilizing the movielens-100k dataset, I focused on the "movies.csv" and "ratings.csv" files. "movies.csv" offers movie attributes like titles and genres, while "ratings.csv" captures user interactions via IDs, ratings, and timestamps. Both datasets are linked to each other through the movie IDs.

Further exploration after the preliminary report submission exposed genre gaps, with some movies labeled "(No genres listed)." Missing genres were meticulously enriched using reliable sources such as IMDB. This elevated user experience in filtering movies.

movieId	title	genres
114335	La cravate (1957)	(no genres listed)
122888	Ben-hur (2016)	(no genres listed)
122896	Pirates of the Caribbean: Dead Men Tell No Tales (2017)	(no genres listed)
129250	Superfast! (2015)	(no genres listed)
132084	Let It Be Me (1995)	(no genres listed)
134861	Trevor Noah: African American (2013)	(no genres listed)
141131	Guardians (2016)	(no genres listed)
141866	Green Room (2015)	(no genres listed)
142456	The Brand New Testament (2015)	(no genres listed)
143410	Hyena Road	(no genres listed)
147250	The Adventures of Sherlock Holmes and Doctor Watson	(no genres listed)
149330	A Cosmic Christmas (1977)	(no genres listed)
152037	Grease Live (2016)	(no genres listed)
155589	Noin 7 veljestv§ (1968)	(no genres listed)
156605	Paterson	(no genres listed)
159161	Ali Wong: Baby Cobra (2016)	(no genres listed)
159779	A Midsummer Night's Dream (2016)	(no genres listed)
161008	The Forbidden Dance (1990)	(no genres listed)
165489	Ethel & Ernest (2016)	(no genres listed)
166024	Whiplash (2013)	(no genres listed)
167570	The OA	(no genres listed)
169034	Lemonade (2016)	(no genres listed)
171495	Cosmos	(no genres listed)
171631	Maria Bamford: Old Baby	(no genres listed)
171749	Death Note: Desu nv¥to (2006,Äi2007)	(no genres listed)
171891	Generation Iron 2	(no genres listed)
172497	T2 3-D: Battle Across Time (1996)	(no genres listed)
172591	The Godfather Trilogy: 1972-1990 (1992)	(no genres listed)
173535	The Adventures of Sherlock Holmes and Doctor Watson: The Hunt for the Tiger (1980)	(no genres listed)
174403	The Putin Interviews (2017)	(no genres listed)
176601	Black Mirror	(no genres listed)
181413	Too Funny to Fail: The Life and Death of The Dana Carvey Show (2017)	(no genres listed)
181719	Serving in Silence: The Margarethe Cammermeyer Story (1995)	(no genres listed)
182727	A Christmas Story Live! (2017)	(no genres listed)

Figure 9 – Dataset with missing genres

movieId	title	genres
114335	La cravate (1957)	Thriller
122888	Ben-hur (2016)	Drama
122896	Pirates of the Caribbean: Dead Men Tell No Tales (2017)	Adventure
129250	Superfast! (2015)	Comedy
132084	Let It Be Me (1995)	Romance
134861	Trevor Noah: African American (2013)	Comedy
141131	Guardians (2016)	Action
141866	Green Room (2015)	Horror
142456	The Brand New Testament (2015)	Comedy
143410	Hyena Road	Drama
147250	The Adventures of Sherlock Holmes and Doctor Watson	Mystery|Sci-Fi|Thriller
149330	A Cosmic Christmas (1977)	Sci-Fi
152037	Grease Live (2016)	Musical
155589	Noin 7 veljestv§ (1968)	Comedy
156605	Paterson	Romance
159161	Ali Wong: Baby Cobra (2016)	Comedy
159779	A Midsummer Night's Dream (2016)	Fantasy
161008	The Forbidden Dance (1990)	Musical
165489	Ethel & Ernest (2016)	Drama
166024	Whiplash (2013)	Drama
167570	The OA	Mystery
169034	Lemonade (2016)	Musical
171495	Cosmos	Sci-Fi
171631	Maria Bamford: Old Baby	Comedy
171749	Death Note: Desu nv¥to (2006,Äi2007)	Thriller
171891	Generation Iron 2	Documentary
172497	T2 3-D: Battle Across Time (1996)	Sci-Fi
172591	The Godfather Trilogy: 1972-1990 (1992)	Crime
173535	The Adventures of Sherlock Holmes and Doctor Watson: The Hunt for the Tiger (1980)	Mystery
174403	The Putin Interviews (2017)	Documentary
176601	Black Mirror	Sci-Fi
181413	Too Funny to Fail: The Life and Death of The Dana Carvey Show (2017)	Documentary
181719	Serving in Silence: The Margarethe Cammermeyer Story (1995)	Drama
182727	A Christmas Story Live! (2017)	Musical

Figure 10 – Dataset after manual cleaning

Simultaneously, the "ratings.csv" dataset incorporated real user feedback, which is essential for system testing. Each user received a distinct user ID, aligning with the data format while ensuring privacy. User ratings are from user ID 11111 to 11110, and a **snippet** of these added ratings is shown below.

1	userId	movieId	rating	timestamp
100838	11111	1186	5	0
100839	11111	1933	1	0
100840	11111	1947	5	0
100888	11112	2	0.5	0
100889	11112	20	1	0
100890	11112	36	4.5	0
100974	11113	50872	5	0
100975	11113	51077	3	0
100976	11113	52722	4	0

Figure 11 – Addition of user ratings to the dataset

Data Cleaning and Optimization

Upon reviewing the "movies.csv" dataset, duplicate movie titles were identified. Closer inspection confirmed that these duplicates referred to the same movies. Notably, duplicates with fewer genres consistently share genres with their counterparts having more genres. For instance, "Emma" with movieId 26958 has the "Romance" genre, while its duplicate with movieId 838 has "Comedy," "Drama," and "Romance." This pattern prompted retaining duplicates with more genres and omitting those with fewer genres for enriched content representation and improved genre-based filtering for recommendations.

```
# Getting the duplicate titles from the movies dataset
duplicate_titles = movies_df[movies_df.duplicated(['title'],
                                                  keep = False)]
duplicate_titles
```

	movieId	title	genres	
650	838	Emma (1996)	Comedy|Drama|Romance	Duplicate with more genre
2141	2851	Saturn 3 (1980)	Adventure|Sci-Fi|Thriller	
4169	6003	Confessions of a Dangerous Mind (2002)	Comedy|Crime|Drama|Thriller	Duplicate with less genre
5601	26958	Emma (1996)	Romance	
5854	32600	Eros (2004)	Drama	
5931	34048	War of the Worlds (2005)	Action|Adventure|Sci-Fi|Thriller	
6932	64997	War of the Worlds (2005)	Action|Sci-Fi	
9106	144606	Confessions of a Dangerous Mind (2002)	Comedy|Crime|Drama|Romance|Thriller	
9135	147002	Eros (2004)	Drama|Romance	
9468	168358	Saturn 3 (1980)	Sci-Fi|Thriller	

Figure 12 – Duplicate Titles

The omitted duplicate records were also removed from the "ratings.csv" file to keep the data consistent between both files.

```
# Validating that the records have also been deleted from the ratings dataframe

# Filtering the ratings_clean dataframe for the specified movieIds
check_in_ratings_clean = ratings_clean[ratings_clean['movieId'].isin(movie_ids_to_check)].copy()

# Creating a DataFrame to store the results
is_in_ratings_clean = pd.DataFrame(columns = ['movieId',
                                              'Found in ratings_clean'])

# Iterating over each movieId and checking if it is present in the ratings_clean dataset
for movie_id in movie_ids_to_check:
    if movie_id in check_in_ratings_clean['movieId'].values:
        result = pd.DataFrame({'movieId': [movie_id],
                               'Found in ratings_clean': 'Yes'})
    else:
        result = pd.DataFrame({'movieId': [movie_id],
                               'Found in ratings_clean': 'No'})
    is_in_ratings_clean = pd.concat([is_in_ratings_clean, result],
                                    ignore_index = True)

# Displaying the results
print(is_in_ratings_clean.to_string(index=False))
```

```
movieId Found in ratings_clean
    838                    Yes
   2851                    Yes
   6003                     No
  26958                     No
  32600                     No
  34048                    Yes
  64997                     No
 144606                    Yes
 147002                    Yes
 168358                     No
```

Figure 13 – Ensuring data consistency between both datasets

The **timestamp column** was also removed from the "ratings.csv" file as it had no use for this project's scope. This meticulous curation process optimizes the dataset's effectiveness and strengthens the recommendation system with enriched and accurate information.

Building and Evaluating the SVD and SVD++ Models

The baseline choice, SVD, offers a balanced approach, leveraging latent factors to capture user-item interaction patterns. This sets a strong foundation, streamlining model implementation and evaluation without additional preprocessing.

The SVD model was trained and evaluated on the training and test sets. This involved creating a reader object with the rating scale, loading data from the DataFrame, splitting the data, training the model, and assessing its performance using metrics like Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The SVD++ model was also implemented, following a similar process of training and evaluation. These models serve as vital components for comprehensively evaluating the recommendation system's performance, setting the stage for more intricate modeling techniques in subsequent phases. The scores of both models are visualized below in Figures 21, 22, and 23, and summarized in Figure 24.

Building the helper functions and recommending movie titles

I developed crucial functions for optimizing movie recommendations, enhancing model evaluation, and refining user-centric suggestions to create tailored user experiences.

1) "build_user_profile" Function: Constructs genre-based profiles by combining user ratings and movie attributes to compute average genre ratings. This tailors suggestions, crafting a personalized movie experience.

2) Printing Functions: "print_recommended_movies_svd" is tailored for SVD and SVD++ model recommendations, while "print_recommended_movies" is used for Deep Learning models. This distinction arises from the nature of the models and their respective outputs. The SVD model predicts and recommends specific movie titles, aligning well with a single iteration print. In contrast, DL models generate predictions for multiple movies, necessitating an iterative print process to maintain a clear presentation.

3) Recommendation Functions for Models: Functions like "svd_recommend_movies," "autoencoder_recommend_movies," "rbm_recommend_movies," and "hybrid_recommend_movies" predict top-rated movies using distinct model approaches.

4) Evaluating Precision: All functions calculate precision@k scores, assessing the relevance of top-k recommendations and quantifying algorithm effectiveness.

```
# Creating a function to extract the real ratings of the user
def get_actual_ratings(user_id, ratings_final):
    actual_ratings = ratings_final[ratings_final['userId'] == user_id][['movieId', 'rating']].values.tolist()
    return actual_ratings

# Creating a function to extract the predicted ratings of the user
def get_predicted_ratings(user_id, svd_model, movies_data):
    predicted_ratings = []
    for movie_id in movies_data['movieId']:
        predicted_rating = svd_model.predict(user_id,
                                             movie_id).est
        predicted_ratings.append((movie_id,
                                  predicted_rating))
    return predicted_ratings

# Creating a function to build the user profile
def build_user_profile(user_id, ratings_final, movies_clean):
    user_ratings = ratings_final[ratings_final['userId'] == user_id]

    user_data = user_ratings.merge(movies_clean, on='movieId')

    user_data['genres'] = user_data['genres'].str.split('|')

    user_data = user_data.explode('genres')

    genre_profile = user_data.groupby('genres')['rating'].mean().to_dict()

    return genre_profile
```

Figure 14 – Helper Functions Part 1

```python
# Creating a function to calculate precision@k
def precision_at_k(predicted_ratings, actual_ratings, genre_profile, k, threshold=4.5):
    top_k_movie_ids = [pred[0] for pred in predicted_ratings[:k]]

    ground_truth_ratings = actual_ratings

    num_relevant_items = sum(1 for movie_id in top_k_movie_ids \ # using "\" to split the code into a new line.
                             if (ground_truth_ratings.get(movie_id, 0) \
                             > threshold) or any(genre_profile.get(genre, 0) \
                             > threshold for genre in movies_clean.loc[movies_clean['movieId'] == movie_id, 'genres'].values[0].split('|')))

    precision = num_relevant_items / k

    return precision
```

```python
def recall_at_k(predicted_ratings, actual_ratings, genre_profile, k, threshold=4.5):
    top_k_movie_ids = [pred[0] for pred in predicted_ratings[:k]]

    ground_truth_ratings = {movie_id: rating for movie_id,
                            rating in actual_ratings}

    # Counting the number of relevant items among the top-k predictions
    num_relevant_items = sum(1 for movie_id in top_k_movie_ids \
                             if (ground_truth_ratings.get(movie_id, 0) > \
                             threshold) or any(genre_profile.get(genre, 0) > \
                             threshold for genre in movies_clean.loc[movies_clean['movieId'] == movie_id, 'genres'].values[0].split('|')))

    # Counting the total number of relevant items in the dataset
    total_relevant_items = sum(1 for rating in ground_truth_ratings.values() if rating > threshold)

    # Calculating Recall@k
    recall = num_relevant_items / total_relevant_items if total_relevant_items > 0 else 0.0

    return round(recall, 1)
```

```python
def print_recommended_movies_svd(recommended_movies):
    print("Recommended movies for user:", user_id)
    for ids, movie in enumerate(recommended_movies, 1):
        print(ids, movie['title'].values[0], "| Genres:", movie['genres'].values[0])
```

Figure 15 – Helper Functions Part 2

Data Preprocessing for the Deep Learning models (Stacked Autoencoders and RBM)

Essential preprocessing steps were undertaken to facilitate the Stacked Autoencoders and RBM models. A pivotal step was to convert the "ratings_final" dataset into a sparse matrix and fill the NaN values with zeros. This sparse matrix was then converted into a Pytorch tensor to be input into the model.

```python
# Converting the ratings_final dataframe into a user-item matrix so that it can be input into the autoencoder
user_item_matrix = ratings_final.pivot(index='userId', columns='movieId', values='rating')

# Filling the NaN ratings in the matrix with 0
user_item_matrix = user_item_matrix.fillna(0)
user_item_matrix.head(3)
```

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585	193587
userId																				
1	4.0	0.0	4.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

3 rows × 9719 columns

```python
# Converting the user_item_matrix into PyTorch tensors
user_item_tensor = torch.Tensor(user_item_matrix.values)
user_item_tensor
```

```
tensor([[4., 0., 4.,  ..., 0., 0., 0.],
        [0., 0., 0.,  ..., 0., 0., 0.],
        [0., 0., 0.,  ..., 0., 0., 0.],
        ...,
        [0., 0., 0.,  ..., 0., 0., 0.],
        [0., 0., 0.,  ..., 0., 0., 0.],
        [0., 0., 0.,  ..., 0., 0., 0.]])
```

Figure 16 – User item matrix and PyTorch tensor conversion

Another critical step in the data modeling was masking the Pytorch tensors. Masking involves the creation of a binary matrix that selectively obscures a fraction of non-zero ratings in the user-item tensor, effectively simulating missing data. This practice mirrors real-world scenarios where user-item interactions might be incomplete. By implementing masking, the DL models learn to predict ratings for both observed and unobserved interactions, thus enhancing their generalization capability. The creation of the masking matrix was realized

through the "create_mask" function. This function effectively identified the observed ratings, calculated the required number of masked ratings based on a specified mask ratio, and created a binary mask matrix highlighting the masked positions.

Incorporation of Advanced Techniques discussed in chapter 7 of Francois Chollet's Textbook (2018) [11]

The architecture of both DL models (Stacked Autoencoders and RBM) demonstrates the application of advanced techniques, such as the PyTorch implementation of the Keras functional API. Additionally, the integration of TensorBoard facilitates the visualization of loss dynamics and architectural representation, further enhancing the sophistication of the models. A snippet of one of the TensorBoard visualizations is shown below, and the remaining TensorBoard visualizations are given below in Appendix B.

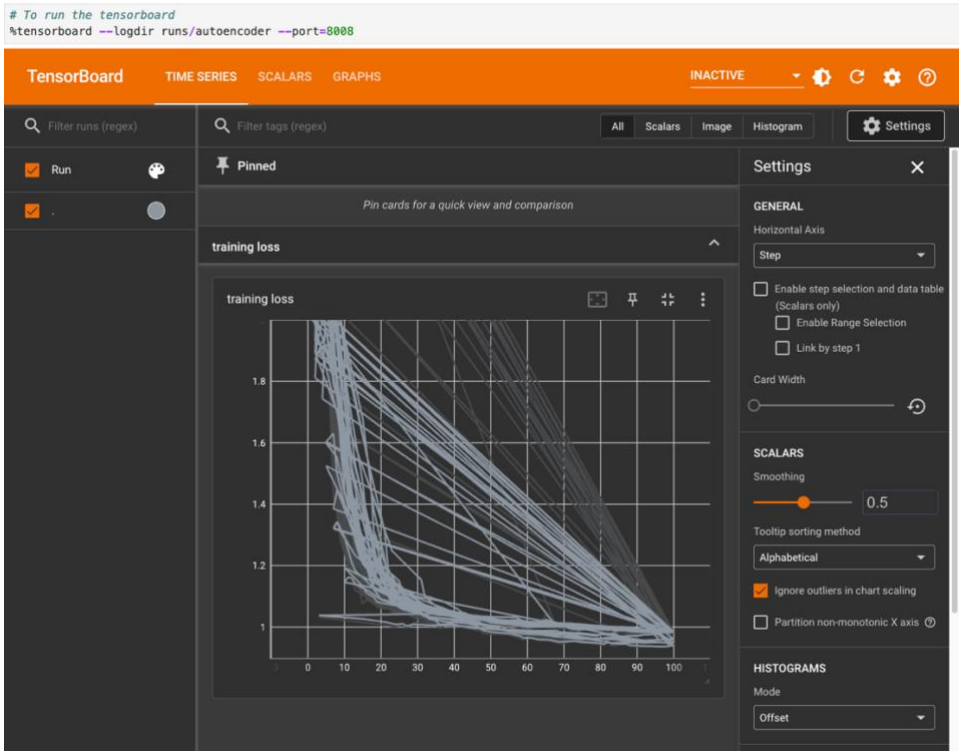


Figure 17 – Loss function TensorBoard Visualization for Stacked Autoencoder training

Implementing the Stacked autoencoder model

In the pursuit of constructing an advanced movie recommendation system, I harnessed the power of Stacked Autoencoders. This neural network architecture was designed to learn and map the intricate patterns underlying user-movie interactions, facilitating personalized recommendations. The architecture was implemented as a "StackedAutoEncoder" class, encapsulating the encoder and decoder layers. The encoder segment comprises three linear layers, progressively reducing the feature space dimensions from the number of movies to 32, fostering abstraction. The decoder counterpart mirrors this structure, increasing dimensions to reconstruct the original input. The rectified linear unit (ReLU) activation function and dropout layers were judiciously integrated to enhance learning and mitigate overfitting.

For training, the Mean Squared Error (MSE) loss function was employed to quantify the difference between the predicted and actual ratings. The Adam optimizer was chosen for its adaptive learning rate, efficiently navigating the optimization landscape. The model was

trained over one hundred epochs, iteratively updating weights through backpropagation to minimize the loss.

Comparing the architecture with the preliminary model revealed a pivotal transformation. In the current model, the architecture became more intricate with three encoder and decoder layers, facilitating deeper feature abstraction. This deepening of the network contributed to improved predictive capabilities, resulting in significantly lower MAE and RMSE scores during training and testing.

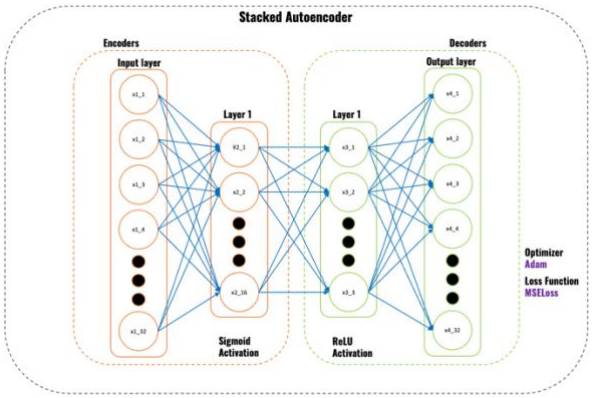


Figure 18 – Old model architecture

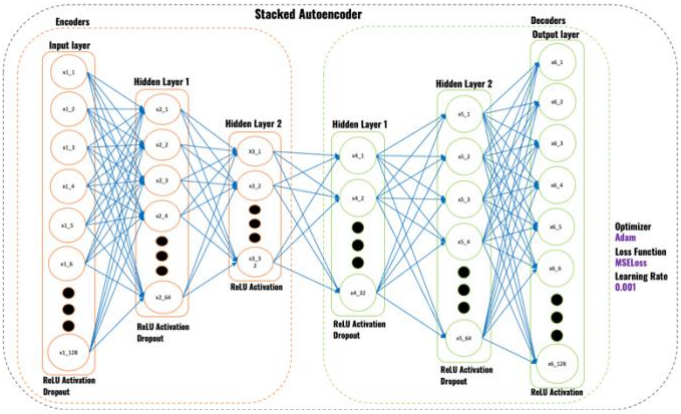


Figure 19 – Current model architecture

Implementing the Gaussian Restricted Boltzmann Machine (RBM) model

Implemented as the "GaussianRBM" class, this model is a powerful tool for uncovering latent patterns within user-movie interactions, contributing to a personalized recommendation system. This model integrates visible and hidden units in a probabilistic manner to learn representations of the input data. The architecture is initialized with weights, biases, and hyperparameters. During training, the model iterates through epochs and employs Gibbs sampling for updating visible and hidden units, optimizing parameters to minimize the MSE loss.

At its core, the Gaussian RBM operates by sampling hidden units given the visible units (encoding) and vice versa (decoding). The weights and biases are adjusted through the training process, iteratively fine-tuning the model's ability to reconstruct input data. Contrastive divergence is employed for training, enhancing the model's convergence and performance. Additionally, early stopping is implemented to prevent overfitting. TensorBoard is leveraged to visualize and analyze the training progress (Appendix B2). This architecture enables the model to capture intricate user-movie interactions, effectively learning patterns that lead to accurate recommendations.

Implementing the Hybrid model of SVD and Stacked Autoencoders

The hybrid movie recommendation model seamlessly integrates the predictive power of both the Stacked Autoencoder and SVD models. This fusion is achieved through a weighted average approach, combining predictions from both models to provide enhanced recommendations. Notably, the SVD predictions hold more weight in this combination.

```
for movie_id in all_movies:
    # Predicting the rating using SVD
    svd_prediction = svd_model.predict(user_id,
                                       movie_id).est

    # Getting the autoencoder prediction
    autoencoder_prediction = autoencoder_predictions[user_item_matrix.columns.get_loc(movie_id)]

    # Combining the predictions (with more preference to SVD predictions)
    combined_prediction = (3 * svd_prediction + autoencoder_prediction) / 4

    predictions.append((movie_id,
                        combined_prediction))
```

Figure 20 – Prediction split for the Hybrid model

This preference towards the SVD model's predictions is strategic, as SVD has proven to excel in capturing latent patterns and user-movie interactions, making it a strong contributor to recommendation accuracy.

CHAPTER 5: Evaluation

(CHAPTER PAGE COUNT EXCLUDING IMAGES – 3)

I employed a multifaceted evaluation approach in this project to comprehensively assess my movie recommendation system's performance. I utilized the MAE and RMSE metrics to gauge the predictive accuracy of individual models' ratings. Additionally, I employed precision@K to evaluate the quality of recommendations produced by each model quantitatively. After meticulously evaluating these metrics, I selected the optimal model for deployment, reflecting a balanced consideration of predictive accuracy and recommendation quality. To further enhance the evaluation, I conducted user testing to solicit valuable feedback on the recommendations generated by the chosen model. This process highlighted the system's successes and limitations and guided its refinement for an enhanced user experience.

Predictive Performance Evaluation and Model Comparison

I comprehensively evaluated the predictive performance of the implemented models using MAE and RMSE. For the SVD and SVD++ models, I leveraged the "accuracy" module from the surprise library to compute their MAE and RMSE scores. The Stacked Autoencoder and RBM models required manual calculation of these metrics. In the Stacked Autoencoder, MAE and RMSE were calculated using the training and test loss, reflecting 0.985 and 0.993 for training and 1.081 and 1.040 for testing, respectively. The RBM model's scores are 1.074 and 1.036 for training and 2.302 and 1.517 for testing. The Hybrid model exhibited MAE and RMSE scores of 0.586 and 0.703, respectively.

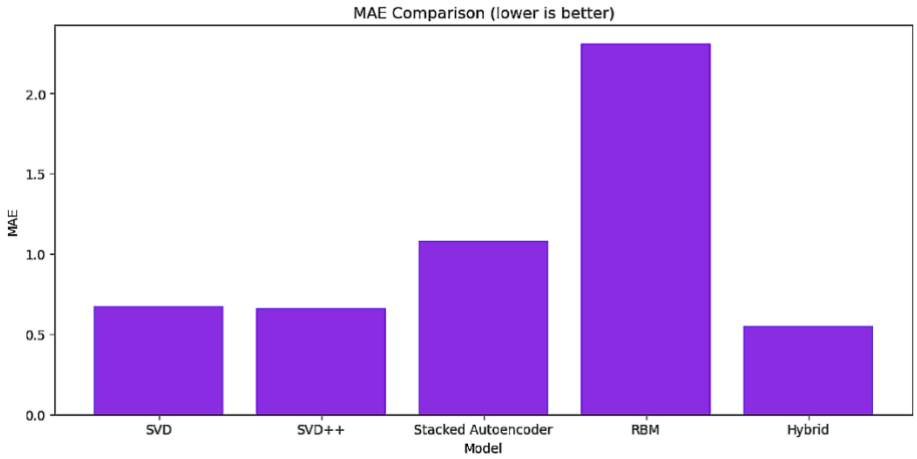


Figure 21 – Summary of MAE scores

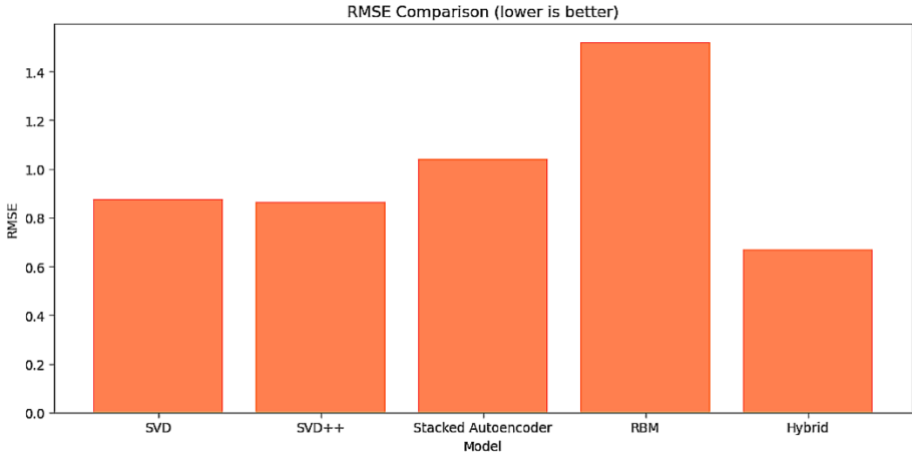


Figure 22 – Summary of RMSE scores

Comparing the models to the baseline SVD model, it becomes evident that the Stacked Autoencoder yielded similar MAE scores for training and testing, indicating comparable predictive accuracy. However, the RBM model displayed higher errors on both training and testing, suggesting limited predictive performance. Intriguingly, the Hybrid model showcased improved performance compared to the baseline SVD model, with a reduced MAE and RMSE on both training and testing sets. This enhancement in predictive accuracy could be attributed to the synergistic blending of SVD and Autoencoder models, leveraging the strengths of each to mitigate the weaknesses of the other. Such model fusion seems to be a promising approach to enhance the system's overall performance.

Evaluation of Title Recommendations and Performance Metrics

To assess the effectiveness of title recommendations, I employed precision@k and recall@k metrics. For the SVD model, both precision and recall were calculated for user 1. The precision@10 score was 0.8, indicating that 80% of the top 10 recommended titles aligned with the user's preferences. Meanwhile, the recall@10 score was 0.1, indicating that only 10% of the user's favored titles were included in the top 10 recommendations. It is important to note that the lower recall value is not necessarily a drawback in this context, as capturing all favored titles within a small set of recommendations is challenging, given the dataset's extensive size. Hence, I decided to calculate only precision@k for the remaining models.

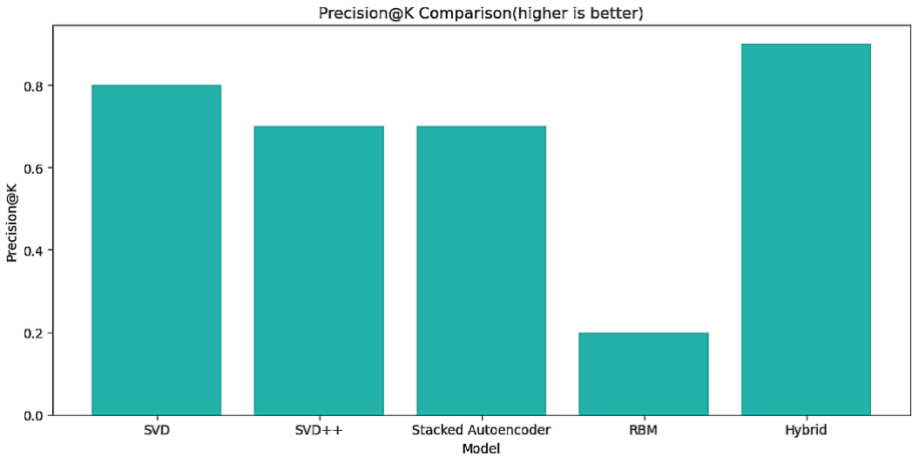


Figure 23 – Summary of precision@k scores

An interesting observation emerged with the stacked autoencoder and hybrid models. Their precision@10 score for user 1 fluctuated on each run. This behavior can be attributed to the nature of these models, which leverage neural networks. These models can produce slightly different results in each run due to the inherent randomness in initialization and training processes. In contrast, the more traditional models like SVD and SVD++ exhibit stable precision@10 scores across evaluations. To fairly compare model performances, I considered their overall performance by analyzing precision@10 scores and their respective MAE and RMSE metrics, considering the balance between recommendation quality and predictive accuracy.

Model	MAE	RMSE	Precision@K
SVD	0.67	0.88	0.8
SVD++	0.66	0.86	0.7
Stacked Autoencoder	1.08	1.04	0.7
RBM	1.52	2.31	0.2
Hybrid	0.55	0.67	0.9

Figure 24 – Model Comparison

User Testing and Evaluation

In the user testing and evaluation phase, my strategy underwent a series of refinements to ensure optimal data collection and meaningful insights. Here, I present an overview of the feedback received from 10 users through a Google Form survey, outlining its implications and how it has shaped the direction of my project.

Refining Data Collection Targets

Initially, the target was set at 20 ratings per user, categorized into 10 highly enjoyed (4* and above), 5 neutral (3* and 3.5*), and 5 disliked (below 3*) ratings based on my own testing and experimentation with the hybrid model. However, this initial threshold proved insufficient in comprehensively discerning user preferences during testing. I conducted further analysis and experimentation to ensure the recommendation system had a more robust basis for accurate recommendations. Through this iterative process, I determined that collecting 50 movie ratings per user would be a more suitable benchmark. Subsequently, I presented this refined approach to the users for testing, as it underscored the importance of gathering a substantial volume of user data to enhance the system's performance.

User-Centric Design Enhancements

I adopted a user-centric design approach to facilitate user testing, incorporating several features to enhance user experience and ensure dependable data collection. As a foundational step, I implemented a tokenization process on the original dataset to extract release years from movie titles. This tokenization was pivotal in creating a year column, enabling users to promptly identify and rate movies from their preferred years. Moreover, I used pivot tables to further categorize movies into specific year ranges, enhancing user convenience. Users commended these features, including filters and slicers based on the year range, allowing them to narrow down movie choices quickly. Feedback also highlighted the value of the genre filter and the simplicity of the Excel table features. These insights emphasize the significance of user-friendly design elements in enhancing data collection and user satisfaction.

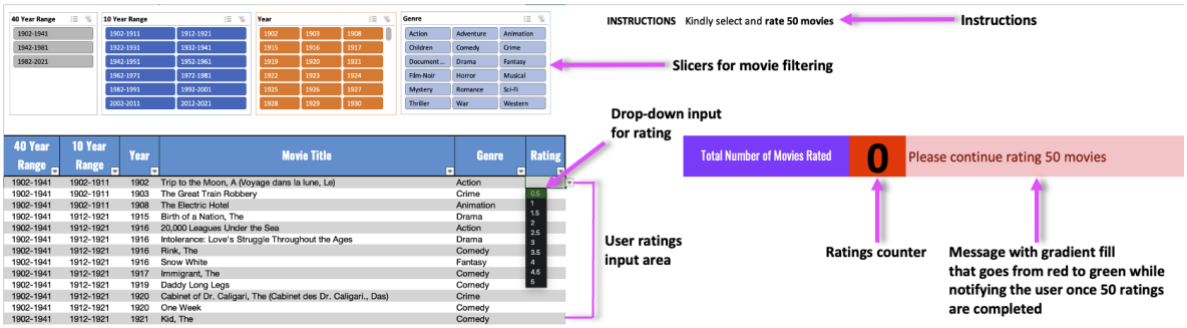


Figure 25 – User Testing Interface

Data Validation and Privacy Measures

Data validation mechanisms ensured data accuracy by restricting acceptable ratings to a specific range of 0.5 to 5 with a step increment of 0.5. Predefined dropdown rating options were also offered, minimizing the likelihood of incorrect or inconsistent inputs. The users also commended the integration of a movie counter and real-time feedback message that informed them of their progress.

Figure 26 – Data Validation

Furthermore, a practical solution was implemented to safeguard data privacy and integrity, where each Excel file was uniquely named after the user. Users' ratings remained confidential, and data cross-contamination was prevented. This feedback emphasizes the importance of stringent data validation and privacy measures to maintain data quality and user trust. Personal user information was also kept anonymous as the ratings of each user in the dataset are defined by a unique user ID assigned to them.

Addressing Formatting and Accessibility Challenges

Challenges arose when distributing separate Excel sheets, including formatting inconsistencies and the inconvenience of users not having Excel installed. In response, I contemplated Google Sheets as a prospective alternative, as it promises cloud-based accessibility and uniform user experiences across various devices. While I have yet to explore this option thoroughly, its potential to mitigate formatting and accessibility issues makes it a viable consideration for future implementations. This feedback highlights the need for flexible data collection methods and consideration of diverse user environments.

Ethical Considerations and Data Integrity

Ethical responsibility and data integrity challenges were noted, as they are common in data collection at scale. User feedback underscored the importance of ethical data collection

practices, as users have a responsibility to provide accurate and genuine feedback. This feedback highlights the need for transparency and ethical awareness in user-centered endeavors, aligning with industry best practices.

Enhancing Recommendation Quality

Post-incorporation of user ratings into the dataset, an evaluation of the hybrid model's recommendations was conducted. Users expressed an intention to watch the recommended movies, indicating promising potential. However, it was consistently observed that recommended movies tended to be older, which users found less appealing. This prompted the decision to exclude movies released before 1960 and to focus on enhancing recommendations from more recent decades in future implementations as they would align better with user preferences. This feedback underscores the significance of refining the recommendation algorithm to better cater to user tastes.

In conclusion, the user testing phase refined the data collection process and yielded pivotal feedback for enhancing the recommendation system's performance. The insights gathered from user feedback have set the stage for future refinements, emphasizing the iterative nature of this project and a commitment to delivering a recommendation system that genuinely resonates with users. The feedback of all users and a summary of the feedback collected are given below in Appendix C.

CHAPTER 6: Conclusion

The primary goal of this project was to construct a Personalized Movie Recommendation System. The endeavor was initiated by the persistent challenge of locating the ideal movie choice in overwhelming options. As a passionate movie enthusiast, the frustration of spending an extended period searching for that ideal cinematic experience resonated profoundly. This project also found inspiration in academic studies that spotlighted user discontent with existing recommendation systems and the pervasive issue of information overload within the digital landscape. The project unfolded under the guiding principle of a user-centric design approach, aiming to strike a harmonious balance between personalization and exploration in movie recommendations.
In pursuing this goal, state-of-the-art models such as SVD, SVD++, Stacked Autoencoders, RBM, and hybrid were employed, which are renowned for building robust and efficient recommendation systems. [12] These models have been recognized and utilized in significant competitions(such as Netflix's 1 million prize competition in 2006)[13] and systems, showcasing their effectiveness and advanced capabilities.

Below are some key research questions posed at the beginning of this project to help me steer this project toward refining the movie recommendation experience.

1) **Balancing personalization and filtering:** Through meticulous experimentation, it became evident that the SVD model excelled in recommending movies aligned with users' preferences and those present on their watchlists. However, this approach had limitations in discovering niche or less popular films, potentially restricting users' exposure to diverse content. In contrast, the Stacked Autoencoder model specialized in recommending these niche and less mainstream movies.

Recognizing the strengths of each model, a hybrid approach was crafted. By implementing a weighted average, this hybrid model primarily drew recommendations from the SVD model, ensuring that users received suggestions closely aligned with their tastes and those they were likely to have heard of. Simultaneously, it incorporated recommendations from the Stacked Autoencoder, introducing users to lesser-known yet potentially captivating films. User testing results underscored the success of this approach, with all users expressing interest in new movies they had not previously encountered. Moreover, 80% of users affirmed their willingness to watch the recommended movies, while the remaining 20% considered the possibility, affirming the effectiveness of this model in enhancing movie discoverability and user satisfaction. This research question's exploration significantly contributed to achieving the project's overarching goal of providing a personalized movie recommendation system that balances personalization and exploration for users.

2) **Effectiveness of standard practices:** The project's investigation into the effectiveness of standard practices in movie recommendation systems yielded valuable insights. Movie metadata, mainly genre tags, was pivotal in enhancing recommendation quality by aligning with users' genre preferences. Additionally, user preferences analysis proved instrumental in tailoring suggestions to individual tastes. However, a persistent issue of recommending older movies remained, which detracted from user satisfaction. In conclusion, while standard practices demonstrated substantial potential, further refinements are essential to ensure

recommendations align better with user preferences, ultimately enhancing overall satisfaction.

3) **Impact of user feedback:** User feedback, encompassing ratings and reviews, emerged as a crucial factor in enhancing the user-centric approach of the recommender system. Ratings provided valuable signals for predicting user preferences and influencing subsequent recommendations. The ability to gauge user satisfaction with recommended movies and tailor suggestions accordingly proved instrumental in improving the overall user experience. The feedback was particularly effective in ensuring a balance between personalization and exploration, as it helped the system identify niche and lesser-known movies that aligned with user preferences. This underlines the importance of user feedback in fine-tuning the recommender system to cater better to individual tastes and enrich the movie discovery process.

Further Work

In shaping the future development of the Personalized Movie Recommendation System, I will be exploring several key areas for further refinement and expansion. Firstly, I plan to remove older movies, specifically those released before 1960, and replace them with more recent movies from this or the previous decade. This step aims to enhance the relevance of recommendations, ensuring they resonate with a broader audience.

Expanding the dataset to include movies in different languages is another pivotal step towards broader inclusivity. I will consider languages based on statistics, focusing on the second-largest language after English movies. Alternatively, I will explore the languages of movies and shows available on the three largest OTT platforms at the time of implementation to align with user preferences.

Additionally, I will be incorporating data on TV shows, recognizing that many users enjoy both movies and TV series. This expansion will allow for a more comprehensive and holistic recommendation experience. To complement this, I will include information about the OTT platforms where recommended shows and movies are available, facilitating convenient access for users.

Beyond content enrichment, future implementations will streamline the process of collecting user feedback at scale. While Google Forms served well for this project's scope, I will explore automated feedback collection or alternative methods for large-scale data gathering. One innovative approach could involve the creation of a browser extension that allows users to rate and provide feedback directly after watching a movie on an OTT platform. This seamless integration will enhance user convenience and provide real-time data on user preferences and satisfaction.

Furthermore, evaluating the system's performance through a more extensive and diverse user base will provide more robust insights. This expansion could potentially involve collaboration with OTT platforms to access a broader user pool, ensuring that the recommendation system resonates with a wider audience and continues evolving into a valuable tool for movie enthusiasts worldwide. These future endeavors hold the promise of elevating the Personalized Movie Recommendation System to new heights.

Appendices

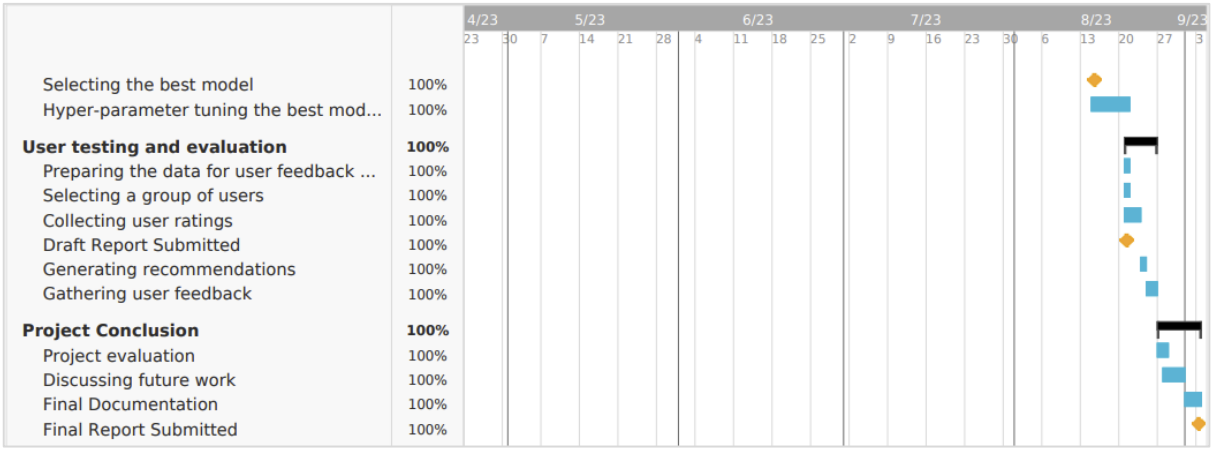
Summary of Page count

Chapter Number	Title	Page info	Page count
1	Introduction	With Images	No images
		Without Images	2 Pages
2	Literature Review	With Images	7 Pages
		Without Images	5 Pages
3	Project Design	With Images	5 Pages
		Without Images	4 Pages
4	Implementation	With Images	7 Pages
		Without Images	3 Pages
5	Evaluation	With Images	5 Pages
		Without Images	3 Pages
6	Conclusion	With Images	No Images
		Without Images	2 Pages

Appendix A

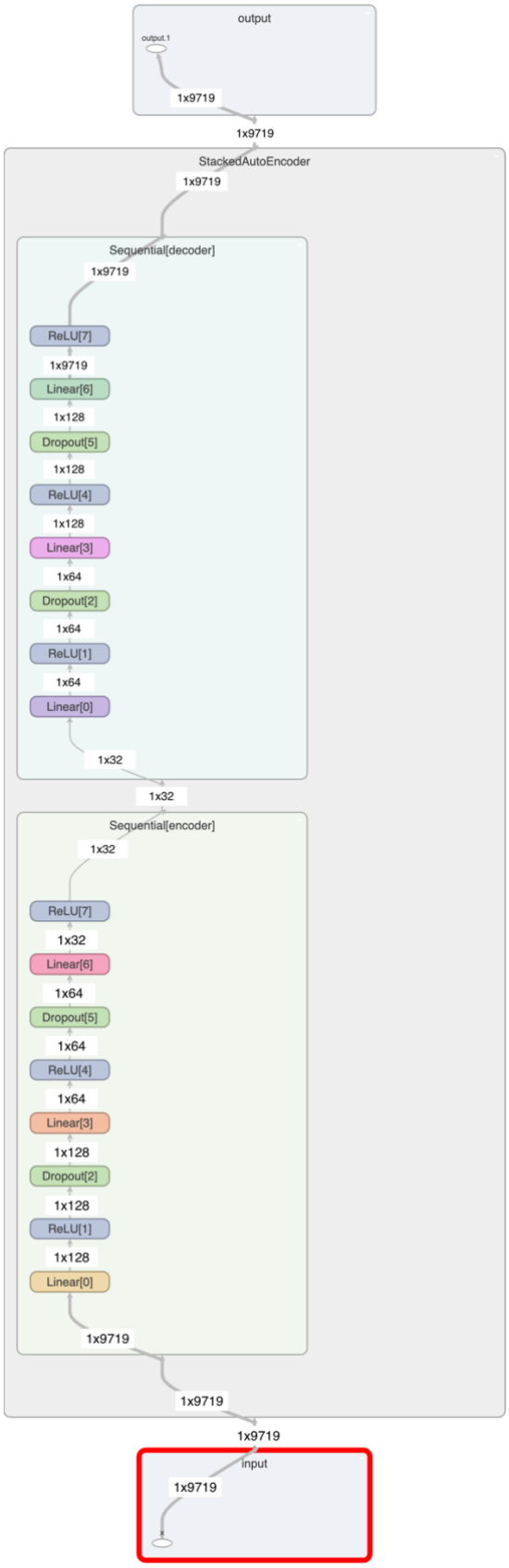
Appendix A1. Gantt Chart Part 1.

Appendix A2. Gantt Chart Part 2.



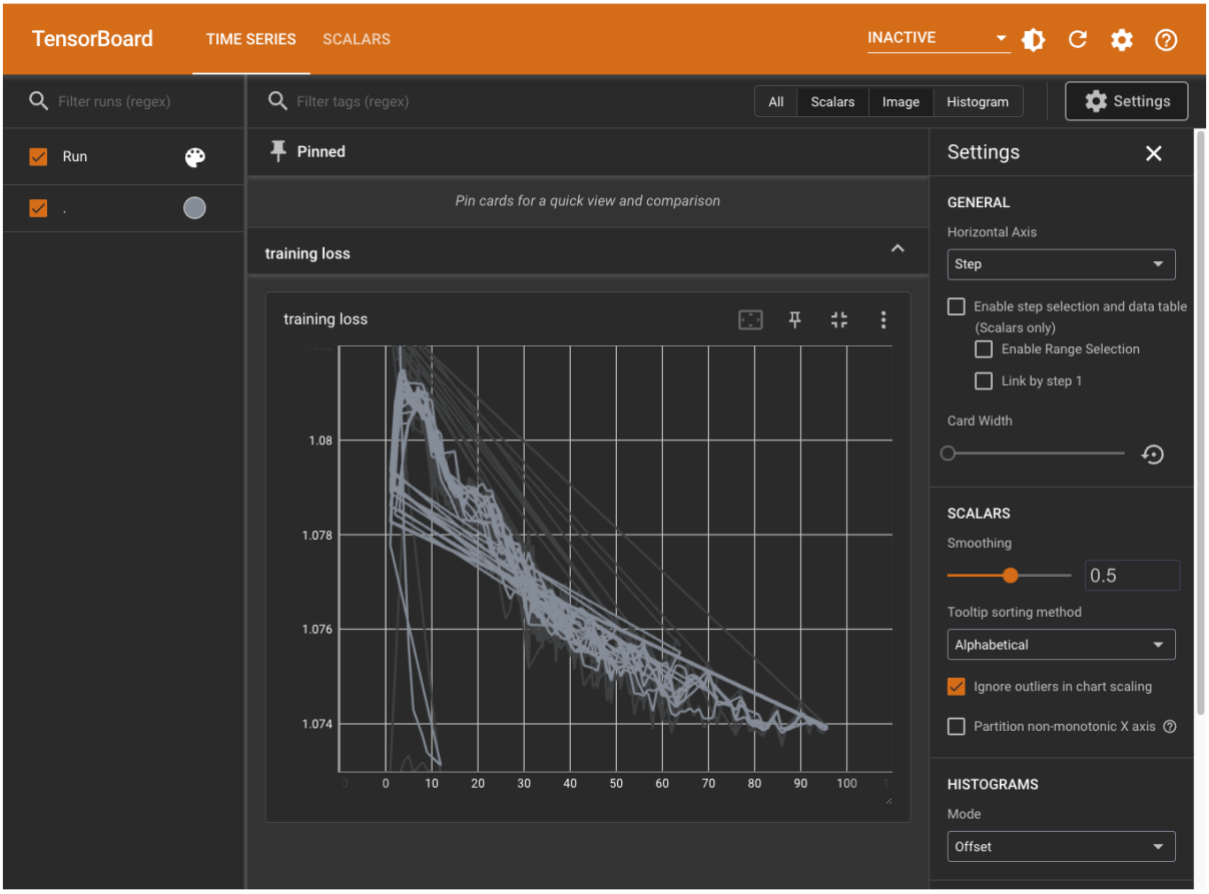
Appendix B

Appendix B1. TensorBoard Graph for Stacked Autoencoders.



Appendix B2. Restricted Boltzmann Machine(RBM) TensorBoard Training Loss.

```
# To run the tensorboard
%tensorboard --logdir runs/rbm --port=8009
```



Appendix C

Appendix C1. User 1 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ○ Slightly relevant
- ○ Moderately relevant
- ◉ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ○ Maybe
- ◉ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ◉ Somewhat accurate order
- ○ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C1. User 1 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- ○ N/A(I haven't seen any of the recommended movies before)
- ○ No
- ○ Yes, somewhat accurately
- ◉ Yes, very accurately

What did you like about the movie recommendations you received? *

They were quite accurate and from different time periods. Many of the recommendations were already on my watch list.

What didn't you like about the movie recommendations? *

They were from mostly the same genre: Drama.

How could the movie recommendations be improved? *

The genres it recommends could be more diverse. Currently I had 7 out of 10 movies recommended already on my watch list. They could be more obscure.

Do you have any other feedback or comments about the recommendation system?

Appendix C2. User 2 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ○ Slightly relevant
- ◉ Moderately relevant
- ○ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ○ Maybe
- ◉ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ◉ Somewhat accurate order
- ○ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C2. User 2 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- ○ N/A(I haven't seen any of the recommended movies before)
- ○ No
- ○ Yes, somewhat accurately
- ◉ Yes, very accurately

What did you like about the movie recommendations you received? *

Intrigued

What didn't you like about the movie recommendations? *

Some repetitions

How could the movie recommendations be improved? *

Considering cast and director as well along with the plot and genre.

Do you have any other feedback or comments about the recommendation system?

Appendix C3. User 3 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ○ Slightly relevant
- ○ Moderately relevant
- ◉ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ○ Maybe
- ◉ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ○ Somewhat accurate order
- ◉ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C3. User 3 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- ○ N/A(I haven't seen any of the recommended movies before)
- ○ No
- ◉ Yes, somewhat accurately
- ○ Yes, very accurately

What did you like about the movie recommendations you received? *

It recommended the movie genres that I like the most

What didn't you like about the movie recommendations? *

The movies are abit old

How could the movie recommendations be improved? *

Recommend movies that are more recent

Do you have any other feedback or comments about the recommendation system?

This system is really good. Comparable to nextflix recommender system.

Appendix C4. User 4 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ◉ Slightly relevant
- ○ Moderately relevant
- ○ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ◉ Maybe
- ○ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ◉ Somewhat accurate order
- ○ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C4. User 4 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- (●) N/A(I haven't seen any of the recommended movies before)
- ( ) No
- ( ) Yes, somewhat accurately
- ( ) Yes, very accurately

What did you like about the movie recommendations you received? *

I liked the difference choices of movies that came up in all genres i choose.

What didn't you like about the movie recommendations? *

All the movies i choose where 20s movies and maybe a few in early 90s, cause i'm not a big fan of old black and white or movies below the 80s. But despite this all my recommendations were old movies.

How could the movie recommendations be improved? *

Time period of the movies being chosen should be considered, a couple being old movie is fine maybe not all. And even tho two movies can be of the same genre, it may be of different story line, movie ratings, difference director and so on. So even tho it has the same genre it can be of different taste of viewer. Maybe can try to incorporate more of similar director, actors, movie ratings for better accuracy.

Do you have any other feedback or comments about the recommendation system?

Already mentioned above

Appendix C5. User 5 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ○ Slightly relevant
- ◉ Moderately relevant
- ○ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ○ Maybe
- ◉ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ◉ Somewhat accurate order
- ○ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C5. User 5 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- (●) N/A(I haven't seen any of the recommended movies before)
- ( ) No
- ( ) Yes, somewhat accurately
- ( ) Yes, very accurately

What did you like about the movie recommendations you received? *

They were movies I have not seen before but am interested

What didn't you like about the movie recommendations? *

There was lack of variety in terms of genre. Being a huge fan of animated movies and having rated a couple I was slightly disappointed to see no animations recommended

How could the movie recommendations be improved? *

More variety perhaps

Do you have any other feedback or comments about the recommendation system?

Appendix C6. User 6 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ◉ Slightly relevant
- ○ Moderately relevant
- ○ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ○ Maybe
- ◉ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ◉ Somewhat accurate order
- ○ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C6. User 6 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- ○ N/A(I haven't seen any of the recommended movies before)
- ○ No
- ◉ Yes, somewhat accurately
- ○ Yes, very accurately

What did you like about the movie recommendations you received? *

It somehow correlates to the genre of movie i am interested in

What didn't you like about the movie recommendations? *

Some movies are more towards the early 2000s hence it might not be something I'd be attracted to watching

How could the movie recommendations be improved? *

Perhaps more in dept questions towards the genre that people like

Do you have any other feedback or comments about the recommendation system?

Nil

Appendix C7. User 7 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ○ Slightly relevant
- ◉ Moderately relevant
- ○ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ○ Maybe
- ◉ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ○ Somewhat accurate order
- ◉ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C7. User 7 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- (●) N/A(I haven't seen any of the recommended movies before)
- ( ) No
- ( ) Yes, somewhat accurately
- ( ) Yes, very accurately

What did you like about the movie recommendations you received? *

Matches up with the kind of movies I selected

What didn't you like about the movie recommendations? *

Not much, it's pretty good

How could the movie recommendations be improved? *

Could include more recent movies as well

Do you have any other feedback or comments about the recommendation system?

Appendix C8. User 8 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ○ Slightly relevant
- ◉ Moderately relevant
- ○ Extremely relevant

Next response

Would you consider watching the movies recommended to you? *

- ○ No
- ◉ Maybe
- ○ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ○ Somewhat accurate order
- ◉ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C8. User 8 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- (●) N/A(I haven't seen any of the recommended movies before)
- ( ) No
- ( ) Yes, somewhat accurately
- ( ) Yes, very accurately

What did you like about the movie recommendations you received? *

The genre is what I like

What didn't you like about the movie recommendations? *

The movie recommendations are old movies.

How could the movie recommendations be improved? *

Recommend some newer movies
Not all movie can be found in the list

Do you have any other feedback or comments about the recommendation system?

N/A

Appendix C9. User 9 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ○ Slightly relevant
- ◉ Moderately relevant
- ○ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ○ Maybe
- ◉ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ◉ Not accurate order
- ○ Somewhat accurate order
- ○ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C9. User 9 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- (●) N/A(I haven't seen any of the recommended movies before)
- ( ) No
- ( ) Yes, somewhat accurately
- ( ) Yes, very accurately

What did you like about the movie recommendations you received? *

I liked the range of the movie genres recommended so its not all drama but also has romcom, crime, thriller, etc.

What didn't you like about the movie recommendations? *

I don't like that it suggested really old films dated before the 90s :,)

How could the movie recommendations be improved? *

Maybe put the movie's year of release into consideration for the movie recommendations? Also this might be complicated but if possible I would also like to know if such movie recommendations are on netflix/amazon prime/disney+ or not eheh

Do you have any other feedback or comments about the recommendation system?

I see potential in this system you're working on. I'll definitely give the movies a watch in my free time! Thanks for letting me try and I wish you all the best of luck in developing this further (if you do) :))

Appendix C10. User 10 Feedback Part 1.

How relevant were the movie recommendations to your personal taste? *

- ○ Not at all relevant
- ○ Slightly relevant
- ◉ Moderately relevant
- ○ Extremely relevant

Would you consider watching the movies recommended to you? *

- ○ No
- ○ Maybe
- ◉ Yes

How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant) *

- ○ Not accurate order
- ○ Somewhat accurate order
- ◉ Mostly accurate order
- ○ Perfect order

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in? *

- ○ No
- ◉ Yes

Appendix C10. User 10 Feedback Part 2.

If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them? *

- (●) N/A(I haven't seen any of the recommended movies before)
- ( ) No
- ( ) Yes, somewhat accurately
- ( ) Yes, very accurately

What did you like about the movie recommendations you received? *

They had some element or the other in the movie that would appeal my taste

What didn't you like about the movie recommendations? *

i believe this is not the systems fault but i dont generally prefer older movies like i am good with movies from 90's and 80' a little but 50's and 60's is a little far back. However i like the stroies that these old movie have so i maybe watch them and have a different view on it after.

How could the movie recommendations be improved? *

Have more filters as to which years do we want suggestions from and which particular genre after the system has understood my taste

Do you have any other feedback or comments about the recommendation system?

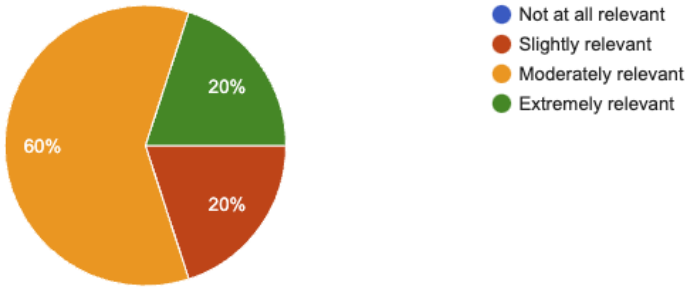
Nothing as of now, but i had a good time being part of this experiment

Appendix C11. User Feedback Summary Part 1.

How relevant were the movie recommendations to your personal taste?

Copy

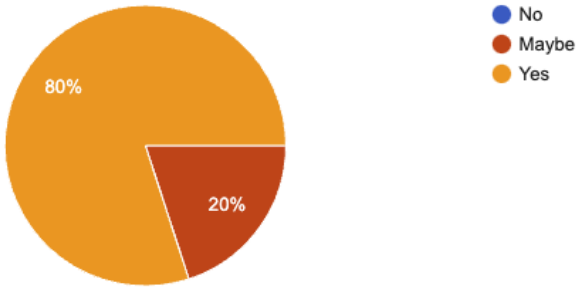
10 responses



Would you consider watching the movies recommended to you?

Copy

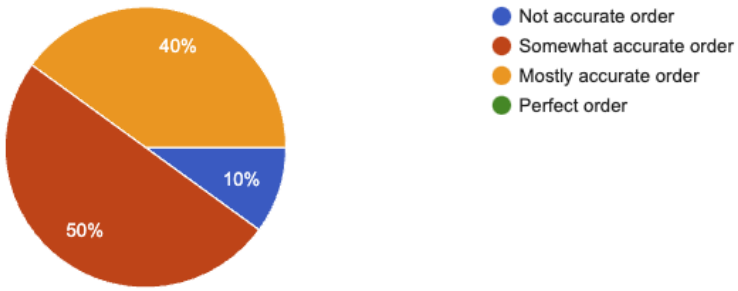
10 responses



How would you rate the order of the recommended movies? (1 being the most relevant and 10 being the least relevant)

Copy

10 responses

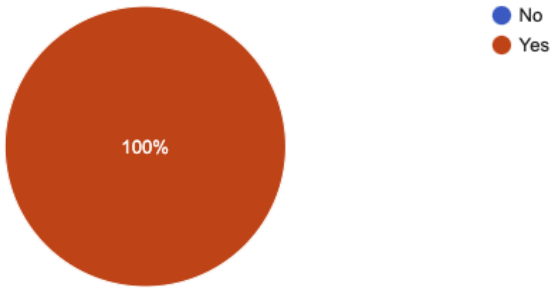


Appendix C11. User Feedback Summary Part 2.

Did the recommendations introduce you to new movies that you had not heard of, but are now interested in?

Copy

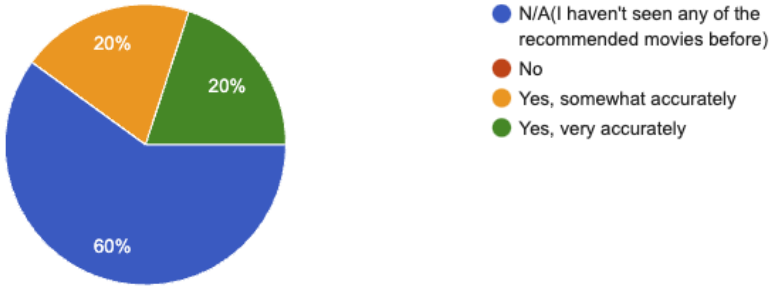
10 responses



If you have seen any of the recommended movies before, did the recommendation system accurately capture how much you liked them?

Copy

10 responses



Code Repository Link

https://github.com/adityavp99/CM3070_FYP

The complete Jupyter notebook has also been added below after the **References** section for quick access.

References

All the references below are according to the ACM reference guide format provided by the UOL online library. The guide can be accessed at:
https://onlinelibrary.london.ac.uk/sites/default/files/files/guides/ACMGuide.pdf

[1] Chang Liu. 2022. Personalized Recommendation Algorithm for Movie Data Combining Rating Matrix and User Subjective Preference. Computational Intelligence and Neuroscience. 2022, 2970514, 1-11.

[2] TechTarget Contributor. 2018. What is filter bubble? | Definition from TechTarget. filter bubble.
https://www.techtarget.com/whatis/definition/social-media

[3] Antoine Falconnet, Constantinos K. Coursaris, Joerg Beringer, Wietske Van Osch, Sylvain Sénécal, and Pierre-Majorique Léger. 2023. Improving User Experience with Recommender Systems by Informing the Design of Recommendation Messages. Applied Sciences. 2023,13, 4, 2706.

[4] Devyn Hinkle. 2021. How Streaming Services Use Algorithms — AMT Lab @ CMU. How Streaming Services Use Algorithms.
https://amt-lab.org/blog/2021/8/algorithms-in-streaming-services

[5] Diana Ferreira, Sofia Silva, António Abelha, and José Machado. 2020. Recommendation System Using Autoencoders. Applied Sciences. 10, 16, 5510

[6] Vihar Kurama. 2022. What Is Collaborative Filtering: A Simple Introduction | Built In. What Is Collaborative Filtering: A Simple Introduction.
https://builtin.com/data-science/collaborative-filtering-recommender-system

[7] Great Learning Team. 2022. Introduction to Autoencoders? What are Autoencoders Applications and Types?. Introduction to Autoencoders? What are Autoencoders Applications and Types?.
https://www.mygreatlearning.com/blog/autoencoder/

[8] Jingdong Liu , Won-Ho Choi, and Jun Liu. 2021. Personalized Movie Recommendation Method Based on Deep Learning. Hindawi. 2021, 1-12.

[9] Guest_blog. 2020. Seq2Seq Model | Understand Seq2Seq Model Architecture. A Simple Introduction to Sequence to Sequence Models.
https://www.analyticsvidhya.com/blog/2020/08/a-simple-introduction-to-sequence-to-sequence-models/

[10] Prakash Pandharinath Rokade, PVRD Prasad Rao, Aruna Kumari Devarakonda. 2020. Forecasting movie rating using k-nearest neighbor based collaborative filtering. International Journal of Electrical and Computer Engineering (IJECE). 12, 6, 6506-6512.

[11] François Chollet. 2018. Deep Learning with Python. Part 2: Deep Learning in Practice. Chapter 7: Advanced Deep Learning Best Practices. 233-268.

[12] Ishwari Singh Rajput, Rakshita Mall, Anand Shanker Tewari, and Arvind Kumar Tiwari. 2021. State-of-the-Art Review of Deep Learning Techniques in Recommendation System. In ICACSE 2021 - International Conference on Advanced Computing and Software Engineering, 2021. 182-188

[13] Mita Chaturvedi. 2021. How Useful Was The Netflix Prize Really?. https://analyticsindiamag.com/how-useful-was-the-netflix-prize-really/.