Bachelor Thesis
Econometrics and Operations Research

Predicting MovieLens preferences using softImpute-ALS

Phung Dang Bao Ngoc i6290653

Supervisor: Dr. Ines Wilms

School of Business and Economics
Maastricht University
June 2024

Contents

List of Figures

List of Tables

Abstract

In the ever-expanding landscape of consumer-focused platforms, recommender systems play an important role in delivering personalized content to users. This thesis investigates the efficacy of the softImpute-ALS algorithm, a matrix completion technique, in predicting movie ratings for recommender systems. By leveraging low-rank approximations, softImpute-ALS addresses the inherent sparsity in user-item interaction matrices, thus improving the reliability of recommendations.

This study focuses on the MovieLens 25M dataset, evaluating the theoretical foundations and practical implementation in movie recommender system of the softImpute-ALS algorithm. We employ a subsample of the dataset to assess the algorithm's performance, tuning parameters using grid search and cross-validation to achieve optimal results. The evaluation metrics, primarily Root Mean Square Error (RMSE), was used to explore how the model performed in different data sections, with detailed analysis revealing biases and tendencies in prediction accuracy across various user and movie segments.

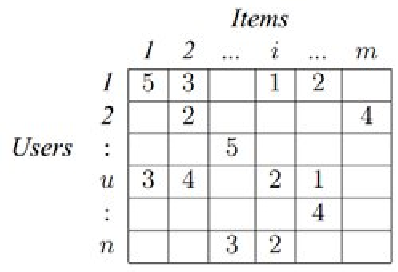
The findings demonstrate that softImpute-ALS effectively handles data sparsity and predicts movie ratings quite efficiently. However, the analysis also highlights areas for refinement, particularly in addressing prediction biases for extreme ratings and improving accuracy for certain genres. This thesis contributes valuable insights into the application of softImpute-ALS in movie recommender systems, providing a robust foundation for future research and development in this field.

Keywords: matrix completion, rating prediction, recommender system, softImpute, MovieLens, low rank assumption

1 Introduction

Recommender systems have become increasingly vital in the adoption of consumer-oriented platforms such as Netflix, Amazon, and Goodreads. As these platforms offer extensive catalogs of items to their numerous clients, recommender systems help provide tailored suggestions to individual customers based on various variables, such as user ratings and item descriptions. This personalization is essential for retaining user engagement and satisfaction, as it helps users discover relevant content from vast libraries of products, movies, books, and more.

One of the key strategies employed by recommender systems is collaborative filtering (CF). CF systems leverage rating data to recommend items to users without considering any additional information about the users or items (LeBlanc et al., 2024). Ratings represent a user's explicit feedback, visually indicating how much a user likes a particular item (Chen and Wang, 2022). For instance, a user rating a movie 4.5 out of 5 stars signifies a high level of satisfaction with the rated item, while a user rating a movie 1 star illustrating the opposite. Intuitively, users who express similar ratings on multiple items tend to display comparable interest for a new item. By analyzing user-item interaction matrices generated from these ratings, CF identifies correlations between different items or users, allowing for data-driven and accurate recommendations.



		Items					
		1	*2*	...	*i*	...	*m*
	1	5	3		1	2	
	2		2				4
Users	:			5			
	u	3	4		2	1	
	:					4	
	n			3	2		

Figure 1: A User-Item Interaction Matrix Example (Karatzoglou et al., 2014)

However, generating accurate recommendations is fundamentally challenging due to the extreme sparsity of the data. In large datasets, most users interact with only a small fraction of available items, leading to many missing entries in the matrix 9 (LeBlanc et al., 2024). For instance, the MovieLens dataset (Harper and Konstan, 2015) contains 25 million ratings across 62,423 movies from 162,541 users, but approximately 99.75% of the user-item rating pairs are missing. This high level of sparsity indicates that the majority of potential ratings are not available, making it difficult to draw reliable underlying movie preference patterns of users to make recommendations.

To address such obstacles, CF systems often employ matrix completion techniques to recover the missing values of user-item interaction matrices (Chen and Wang, 2022). Matrix completion aims to infer these missing entries by learning the underlying behavior of the observed data, enhancing the accuracy and reliability of recommendations. There are two main approaches for collaborative filtering with matrix completion: neighborhood-based models (NBMs) and latent factor models (LFMs) (Chen and Wang, 2022).

Neighborhood-based models (NBM) are memory-based, meaning they rely on the entire user-item interaction matrix to make recommendations by calculating similarities between every pair of users and items. This approach is based on the idea that users with similar tastes or items with similar features are likely correlated. For example, a user might get recommendations for an item that other similar users have liked, or for items that are similar to those items that the user has previously enjoyed.

While NBMs can be effective and are relatively simple to implement, they face significant challenges. One major issue is high data sparsity, which makes it difficult to accurately calculate similarity scores between all users and items. Additionally, NBMs struggle with high dimensionality (LeBlanc et al., 2024). As the number of users and items grows, the computational burden increases exponentially. This results in scalability problems, making NBMs computationally expensive and less practical for large-scale systems. The need to compute and store similarity measures for potentially millions of users and items further exacerbates these issues, highlighting the limitations of NBMs in handling extensive, sparse datasets (LeBlanc et al., 2024).

In contrast to NBMs, latent factor models (LFMs) offer a model-based approach that addresses many of the limitations faced by NBMs such as scalabilty and sparsity. LFMs work by projecting both user and item features into a lower-dimensional space defined by latent factors (LeBlanc et al., 2024). These latent factors are essentially vectors that capture the underlying preferences of users and the implicit characteristics of items. The key assumption is that the preferences and behaviors of users, as well as the attributes of items, can be effectively represented by only a relatively small number of latent factors (Chen and Wang, 2022). It means that despite the vast diversity of items and user preferences, there are fundamental patterns or factors that can explain the majority of the interactions. Hence, the dimensions of the rating matrix can be effectively reduced to those that represents the most information of the data.

This dimensionality reduction in LFMs significantly reduces the complexity of the data. By transforming high-dimensional data into latent factors, LFMs can uncover underlying patterns that are not explicit in the original data while being computationally efficient. For instance, in a movie recommender system, latent factors might represent genres, specific themes, or even abstract preferences like a preference for fast-paced action or deep, emotional narratives. While being able to capture the sophisticated dynamics of users' preferences through latent factors, LFMs also have an edge over NBMs in terms of handling sparsity and computational complexity due to the reduction of dimensionality.

One popular matrix completion technique that shares many common principles with LFMs is softImpute-ALS, proposed by Hastie et al. (2015). While not developed specifically for collaborative filtering, softImpute solves the matrix completion task that can be applied in recommender systems. This method iteratively fills in the missing entries by leveraging low-rank approximations, thus enhancing the overall accuracy of recommendations. A low-rank approximation involves representing the original high-dimensional matrix with a smaller number of ranks, which capture the most significant patterns in the data. This approach reduces the complexity of the matrix while retaining its essential features, making it easier to identify and predict user preferences based on fewer, more meaningful components. This is particularly advantageous in collaborative filtering, where data is often sparse and high-dimensional. Among various matrix completion techniques, SoftImpute-ALS stands out for its ability to handle large, sparse datasets efficiently, making it particularly suitable for applications in recommender systems, especially for highly sparse datasets like MovieLens.

In this thesis, we evaluate the softImpute algorithm by delving into its theoretical foundations and performance evaluation using a real-world dataset. By focusing on a subsample of the mentioned MovieLens 25M dataset, we aim to explore the efficacy of softImpute in mitigating the sparsity problem and improving the quality of recommendations in movie rating platforms.

2 Literature Review

2.1 Matrix Completion - Existing Research

In the context of matrix completion research, there are many different techniques. These methods generally fall into two categories: matrix factorization and neural networks. Within matrix factorization category, one of the foundational techniques in matrix factorization is the Singular Value Decomposition (SVD) method. SVD is a mathematical method that decomposes a given matrix into smaller matrices. This decomposition allows for the approximation of the original matrix with only a fraction of all data, effectively capturing the most significant features of the data while minimizing noise and redundancy. We will delve into the concept of SVD in later part of this thesis (Section 3.1), as it is one of the core principles behind our algorithm softImpute.

Buiding on this foundation, Koren (2008) enhanced the traditional SVD model by incorporating implicit feedback into the SVD++ model. Implicit feedback in SVD++ includes user interactions such as clicks, views, and purchase history, which are not explicit ratings but also indirectly indicate user preferences. This model optimizes its learnable variables by minimizing the squared error function using a descent-based method. In matrix factorization category, there is also the LLORMA (Local Low-Rank Matrix Approximation) model, proposed by Lee et al. (2013). In this model, the observed matrix is assumed to be low-rank in only the vicinity of certain row-column combinations instead of the whole matrix. Therefore, this model creates multiple local low-rank models by targeting different "regions" of the data matrix, with each local model optimized independently.

In some other state-of-the-art methods, instead of matrix factorization, neural networks are also used to derive latent factors for matrix completion (He et al., 2017). Neural networks are computational models inspired by the human brain, consisting of interconnected layers of nodes (neurons) that process input data to recognize patterns and make predictions. In matrix completion, neural networks can effectively learn complex, non-linear relationships within the data, capturing intricate interactions that traditional matrix factorization methods might miss. For instance, Cheng et al. (2016) introduced the wide and deep learning approach, which jointly trains wide linear models and deep neural networks to enhance recommender systems. The wide component remembers how different features occur together, while the deep component learns more complex patterns from the data. By combining these two approaches, the model benefits from both simple and complex insights, improving recommendation accuracy.

This thesis, however, focuses on the matrix factorization techniques since they are rather simple to implement and relatively computationally inexpensive. Among these techniques, a particularly notable approach is softImpute-ALS.

2.2 The Context of SoftImpute

The softImpute-ALS algorithm, proposed by Hastie et al. (2015), draws inspiration from nuclear-norm-regularized matrix approximation (also known as softImpute-SVD) (Candès and Tao, 2009); (Mazumder et al., 2010) and maximum-margin matrix factorization (ALS)

(Srebro et al., 2005). In specific, softImpute-ALS incorporates concepts from softImpute-SVD, which addresses missing data by using thresholding to reduce the dimension of the rating matrix, thus helping in noise reduction and achieving a low-rank approximation. It also employs the ALS optimization technique that resolves matrix factors by iteratively minimizing the loss function for one factor while keeping the other fixed, leading to a more efficient optimization process. Notably, softImpute-ALS has been demonstrated by Hastie et al. (2015) to outperform both of these foundational algorithms softImpute-SVD and ALS in terms of dealing with large sparse matrices.

Given the positive theoretical performance of softImpute as a matrix completion method, its performance in real life application aspect is indeed of high interest for researchers. In practice, softImpute has been implemented in various domains, such as recommender systems, healthcare prediction, and image reconstruction. For example, Kidzinski and Kidzinski (2017) evaluated softImpute in matrix completion tasks using different hyperparameters for a sweets and candies ranking platform. Additionally, Aydin and Ozturk (2021) predicted the length of stay in intensive care units and found that softImpute's performance in the missing data imputation task was superior to other applied methods . In the field of hydrometeorology, Sun et al. (2020) used softImpute as part of their method to reconstruct total electron content (TEC) video maps, ensuring spatial smoothness and temporal consistency while preserving crucial structures of these maps.

We aim to extend the research on the application and evaluation of softImpute[1] to the domain of movie recommender systems by applying it to the MovieLens 25M dataset. In essence, we use softImpute to predict unobserved ratings in the rating matrix and assess whether the algorithm is beneficial to make rating predictions for movie recommendations. This analysis provides valuable insights into how effectively softImpute can handle the unique challenges inherent in movie recommendation systems. To this end, in the later part of this thesis, we analyze its performance using evaluation metrics and analyze the behavior of the algorithm in this setting.

3 SoftImpute-ALS Algorithm

3.1 Low Rank Assumption

One of the main principles in matrix completion is singular value decomposition (SVD). Singular value decomposition of an $m \times n$ matrix X is a factorization of the form:

$$X = U \Sigma V^T$$

where U is an $m \times m$ orthogonal matrix, Σ is an $m \times n$ diagonal matrix with nonnegative entries, and V is an $n \times n$ orthogonal matrix.

In matrix Σ, the diagonal entries $\sigma_i = \Sigma_{ii} \geq 0$ and are known as the singular values of X. The number of non-zero singular values is called the rank of matrix X. These singular values provide crucial information about the properties of a matrix. In the context of softImpute-ALS, they are manipulated to fill in missing entries of the matrix by imposing a low-rank constraint.

[1]We use the terms 'softImpute' and 'softImpute-ALS' interchangably to refer to the 'Rank-Restricted Efficient Maximum-Margin Matrix Factorization algorithm: softImpute-ALS' proposed by Hastie et al. (2015). Note that there is a fundamental difference between the algorithms softImpute-ALS and softImpute-SVD.

Similar to many other matrix factorization methods, softImpute-ALS relies on the low rank assumption. This assumption implies that the observed matrix should have a low rank, meaning that it has only a few non-zero singular values. Since most rating records in the user-item interaction matrix are similar and correlated across rows or columns, most information within the rating matrix tends to be redundant. As a result, many singular values in the observed matrix would be zero or close to zero. By making this assumption, underlying patterns in the rating matrix can be extracted by decomposing the matrix into low-rank components.

3.2 Objective Function

Let $X \in \mathbb{R}^{m \times n}$ be a matrix with elements x_{ij}. Let Ω be the set of indices corresponding to the observed entries in X.

$$\Omega = \{(i, j) \mid x_{ij} \in X \text{ is observed} \quad \forall\, i \leq m \text{ and } j \leq n\}$$

Following Candès and Tao (2009), we define the projection $P_\Omega(X)$ to be an $m \times n$ matrix with the observed elements of X preserved, and the missing entries replaced with 0.

$$P_\Omega(X)_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{if } (i,j) \notin \Omega \end{cases}$$

Likewise, the $P_{\Omega^\perp}(X)$ matrix projects onto the complement of the set Ω.

3.2.1 Nuclear-Norm-Regularized Matrix Approximation

To impute the missing values in matrix X, we aim to find a completed matrix M that minimizes the reconstruction error and the rank of the solution matrix. We achieve this by the following steps:

Firstly, by minimizing the Frobenius norm of the difference between X and M at the observed indices $\|P_\Omega(X - M)\|_F^2$, we thus minimize the reconstruction error, essentially maximizing how well the imputed values match the original observed data.

Secondly, we wish to minimize for the rank function $rank(M)$ of estimated matrix M. However, this rank function is non-convex, hence solving optimization problems for this function will be NP-hard. Notably, we have the nuclear norm of matrix M $\|M\|_*$, which is defined as the sum of all singular values in M, is a convex relaxation of the rank function. By minimizing this nuclear norm, softImpute encourages the solution to have smaller singular values, effectively promoting a low-rank structure.

Mathematically, such objective can be defined as the following:

$$\min_M \frac{1}{2}\|P_\Omega(X - M)\|_F^2 + \lambda \|M\|_*. \tag{1}$$

where matrix M is the approximate for observed incomplete matrix X, $\|\cdot\|_F^2$ is the Frobenius norm to measure data fidelity, $\|\cdot\|_*$ is the nuclear norm that acts as a convex relaxation of the rank function, and λ is the regularization term.

In specific, λ is a critical parameter that balances between two objectives: ensuring the completed matrix M is close to the observed entries matrix X and encouraging M to be low-rank. By choosing a small value for λ, the emphasis is placed more on minimizing the data fidelity term $\|P_\Omega(X - M)\|_F^2$ to encourage the matrix M to fit closely to X at the observed indices, potentially leading to a higher rank solution. Meanwhile, when λ is large, the algorithm puts more weight on minimizing the nuclear norm $\|M\|_*$, encouraging M to be low-rank. By making matrix M having lower rank, although the simpler matrix M might not fit closely to observed data, it however generalizes better to unseen data.

Choosing suitable λ requires considerable considerations, since a too large value of λ oversimplifies our model, making it underfitting and failing to capture the underlying pattern of observed data, while a too small value makes our model overfitting to training data and cannot generalize to future unseen data. We can vary λ to provide a range of solutions from high-rank to low-rank, allowing for choosing a suitable λ that is able to achieve the desired balance between fitting the data and maintaining a low-rank structure.

If solution matrix $\hat{M}$ solves the convex-cone problem defined in (1), it satisfies the following stationarity condition:

$$\hat{M} = S_\lambda(Z) \tag{2}$$

where $Z = P_\Omega(X) + P_{\Omega^\perp}(\hat{M})$. To clarify, Z is the "filled in" version of X with estimated entries from $\hat{M}$. In specific, the operator $S_\lambda(Z)$ applied to matrix Z in (2) does the following. First, it computes the SVD of $Z = U\Sigma V^T$, and let σ_i be the singular values in Σ. Secondly, it applies soft-thresholding operator to the singular values: $\sigma_i^* = (\sigma_i - \lambda)_+$. It then reconstructs the new SVD: $S_\lambda(Z) = UD^*V^T$. In short, this operator iteratively uses the current estimate for M to create matrix Z and update M by the soft-thresholded SVD of Z.

When the stationarity condition is satisfied, it means that the iterative process has stabilized. The imputed matrix $\hat{M}$ no longer changes significantly between iterations, indicating that softImpute has reached a solution, where $\hat{M}$ is consistent with the observed entries in the original matrix and has a low-rank structure enforced by the soft-thresholding operator.

However, for large matrices, iteratively calculating SVD for the *full-rank* matrix Z is computationally intensive. Hence, Mazumder et al. (2010) uses two important tricks to avoid the computational challenges. Firstly, they anticipate that the solution $\hat{M}$ will have low rank. Therefore, they only compute a *low-rank* SVD of Z using alternating subspace methods instead of the full rank SVD. The low-rank subspace methods (ALS) will be discussed in more detail in the later part of this section. Secondly, they re-express matrix Z at each iteration that separates the observed and estimated parts of the matrix as:

$$Z = P_\Omega(X) - P_\Omega(\hat{M}) + \hat{M} \tag{3}$$

Leveraging the fact that only parts of the matrix change significantly between iteratons, this re-expression separates the contributions of the observed entries and estimated missing entries, making it possible to update the matrix efficiently without having to recompute the SVD from scratch for the whole matrix. This is called the "Sparse + Low Rank" representation of matrix Z, which is inexpensive to store and compute. Additionally, this special structure also allows for efficient computations of left and right multiplications by skinny matrices.

3.2.2 Maximum-Margin Matrix Factorization

SoftImpute-ALS imposes a rank constraint that limits the rank of the solution. Since we assume low-rank structure, this means that if the algorithm returns a solution to the rank-restricted problem, namely returning a solution with rank lower than the specified rank, the algorithm also solves for the unrestricted problem (Hastie et al., 2015). With the rank constraint, we consider the following alternative criterion (Rennie and Srebro, 2005):

$$\min_{A,B} \left\| P_\Omega(X - AB^T) \right\|_F^2 + \lambda \left(\|A\|_F^2 + \|B\|_F^2 \right). \tag{4}$$

where r is the imposed rank constraint, A is an $m \times r$ matrix and B is an $n \times r$ matrix, where m and n are number of rows and columns in the matrix X, with $r \leq \min(m, n)$.

In the expression (1), the nuclear norm of matrix M serves as a convex relaxation of the rank function; however, in function (4), this maximum-margin matrix factorization (MMMF) criterion is not convex in A and B, hence we cannot minimize the rank function for A and B simultaneously. Nevertheless, function (4) is in fact bi-convex. This means that for fixed B, the criterion is convex in A, while for fixed A, the criterion is convex in B. We can therefore minimize the rank of A and B by alternately minimize for A and B separately by fixing one matrix and updating the other. Remarkably, although function (4) is not convex, it has a solution that satisfies $\hat{A\hat{B}^T} = \hat{M}$, where $\hat{M}$ is the solution in (1) (Hastie et al., 2015).

Similar to expression (2) and (3), we characterize the stationarity conditions and "Sparse + Low Rank representation", in terms of $\hat{A}$ and $\hat{B}$ for this criterion:

$$Z = P_\Omega(X) + P_{\Omega^\perp}(\hat{A}\hat{B}^T) \tag{5}$$

$$Z = P_\Omega(X) - P_\Omega(\hat{A}\hat{B}^T) + \hat{A}\hat{B}^T \tag{6}$$

With softImpute-ALS, we calculate the alternating subspace SVD using alternating minimization algorithm (ALS). Without loss of generality, we consider A fixed, and wish to minimize for matrix B. This is equivalent to solving the following ridge regression of columns of Z on the fixed (latest updated) A:

$$\min_{\tilde{B}} \left\| Z - A\tilde{B} \right\|_F^2 + \lambda \left\| \tilde{B} \right\|_F^2. \tag{7}$$

We then obtain $\tilde{B^T} = (A^T A + \lambda I)^{-1} A^T Z$. After each alternating regression, we update the components of $\tilde{A}$ and $\tilde{B}$, and then fill in matrix Z with updated estimates. This alternating process is repeated until a convergence criterion is met. This convergence criterion is user-defined, involving the selection of a maximum number of iterations and a convergence threshold value to determine when the algorithm has sufficiently minimized the reconstruction error. In the R Package published by Hastie et al. (2015), by default, this threshold is set to 1e-05 and the maximum number of iterations is automatically restricted at 100.

3.3 Technical Considerations and R Implementation

3.3.1 Data Centering & Scaling

As suggested by Hastie et al. (2015), it is often beneficial to remove row and/or column means from the matrix before performing a SVD or running our matrix completion algorithms. This preprocessing step, known as centering, helps to eliminate biases in the data, ensuring that the algorithm's focus is on the underlying structure rather than on shifts in the data's baseline levels. Similarly, we may wish to standardize the rows and/or columns so that they have unit variance, a process known as scaling. Standardizing ensures that each feature contributes equally to the analysis by adjusting for differences in scale, thereby improving the performance and stability of the algorithms.

Although highly helpful in the matrix completion context, centering and scaling for large sparse matrices is not a simple task. Since sparse matrices contain many missing entries, the algorithms used for scaling and centering needed to be efficient in handling these missing entries without converting the matrix to a dense format, which can be computationally expensive and memory-intensive. Therefore, Hastie et al. (2015) presented an important algorithm to preprocess our data before implementing softImpute-ALS.

We have a two-dimensional array $X = \{x_{ij}\} \in \mathbb{R}^{m \times n}$ with $(i, j) \in \Omega$ observed, and the rest is missing. To standardize the rows and columns of X to mean zero and variance one simultaneously, Hastie et al. (2015) consider the mean/variance model:

$$x_{ij} \sim (\mu_{ij}, \sigma_{ij}^2)$$

with

$$\begin{aligned} \mu_{ij} &= \alpha_i + \beta_j; \\ \sigma_{ij} &= \tau_i \gamma_j \end{aligned}$$

where α_i represents the mean offset for the i-th row of matrix X, and β_j for j-th column of matrix X; τ_i is the scaling factor for the i-th row of matrix X, adjusting the variance within the row and normalizing it to unit variance, and γ_j solves the similar task as τ_i but to the j-th column of matrix X.

Given the parameters of the model, they standardize each observation via:

$$\begin{aligned} \tilde{x_{ij}} &= \frac{x_{ij} - \mu_{ij}}{\sigma_{ij}} \\ &= \frac{x_{ij} - \alpha_i - \beta_j}{\tau_i \gamma_j} \end{aligned} \tag{8}$$

This dual centering and scaling algorithm differs from similar algorithms in several significant ways. Firstly, it accommodates missing data, a feature that many other algorithms lack. Additionally, it learns the parameters in equation (8) for centering and scaling dynamically rather than simply applying predetermined values. This approach is particularly suitable for matrix completion tasks not only because of the extreme sparsity of the data but also due to the necessity of reversing the centering and scaling on our predictions. Consequently, this method ensures more accurate and reliable imputation and prediction results in sparse data contexts.

We find that, on the MovieLens dataset, implementing softImpute in R requires this preprocessing step to achieve meaningful results. Without scaling, softImpute-ALS performs significantly worse, with high errors of around 4 for all predictions, given that the movie rating scale ranges from 0.5 to 5. To apply the dual centering and scaling algorithm described above in R, one can utilize the $biScale()$ function included in the softImpute R package published by Hastie et al. (2015). This function should be applied to the matrix before using softImpute. Notably, with $biScale()$, the output is automatically adjusted back to the appropriate rating scale after implementing softImpute, ensuring accurate and interpretable results.

3.3.2 Runtime Complexity

As shown by Hastie et al. (2015), the total cost of an iteration in softImpute-ALS is $\mathcal{O}(2r|\Omega| + mr^2 + 3nr^2 + r^3)$ where r is the rank constraint, $|\Omega|$ is the size of observed entries set, and our rating matrix is defined as $X \in \mathbb{R}^{m \times n}$.

SoftImpute demonstrates efficient runtime complexity in the context of extremely sparse high-dimensional matrix completion. This efficiency makes it particularly well-suited for applications such as the MovieLens dataset, where the data's sparsity and high dimensionality present significant computational challenges. The algorithm's ability to handle large-scale, incomplete matrices with great speed and lower computational cost underscores its practical utility and effectiveness in real-world scenarios.

4 Data

4.1 User Feedback

In recommender systems, feedback is defined as the information provided by users that reflects their preferences, interests, and behaviors (Jannach et al., 2012). Recommender systems use this feedback input to make predictions about the preferences of a user and make appropriate recommendations. There are two primary types of feedback: explicit and implicit feedback (LeBlanc et al., 2024). Explicit feedback refers to direct user input, such as numerical ratings or reviews for movies, where users rate films on a scale from one to five stars. This type of feedback provides clear and quantifiable preferences that can be directly used to predict future user preferences. On the other hand, implicit feedback is derived indirectly from user behavior, such as viewing history, search queries, or the time spent watching a movie.

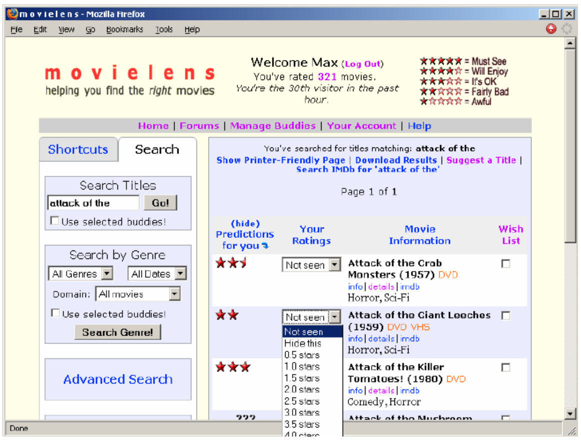
There is a trade-off between utilizing explicit and implicit feedback. Explicit feedback may be more informative, yet implicit feedback requires fewer resources to collect (Nilashi et al., 2013). Explicit feedback is easy to interpret as it directly reflects a user's preference for an item, but obtaining this feedback can be challenging and costly since many users do not respond to surveys and rating systems consistently. On the other hand, although implicit feedback is relatively easier to acquire, decoding its underlying message is not straightforward. In this thesis, since the MovieLens dataset provides explicit feedback from users in the form of movie ratings, we evaluate using only this input.

4.2 Sampling

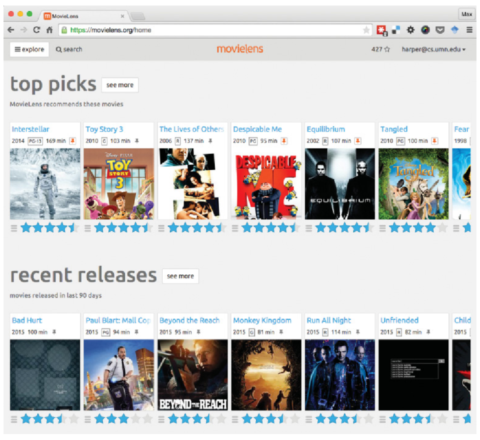
For our evaluation of the softImpute algorithm, we use a subsample of the MovieLens 25M dataset. The original dataset captures 5-star rating activities from the MovieLens movie recommendation service, encompassing 25 million ratings for 62,424 movies by 162,541 users between January 9, 1995, and November 21, 2019. MovieLens has consistently used a "star" rating system, a common user interface for expressing preferences. On February 18, 2003, MovieLens introduced a significant update to its rating interface, changing from "whole star" ratings to "half star" ratings with the release of version 3 (v3). The 25M dataset includes randomly selected users throughout all 4 version releases, each of which had rated a minimum of 20 movies.

For our subsample, we first filter for ratings occurring after February 17, 2003, to maintain consistency with the v3 release and the transition from "whole star" ratings (1 to 5) to "half star" ratings (0.5 to 5). We then randomly sample for 3% of these ratings to maintain the computational feasibility given limited time and computing power.

It is important to note that besides sparsity, recommender systems also face cold start problems, characterized by insufficient data about a user or an item (LeBlanc et al., 2024). In specific, when a new user joins the platform or when a new item is added to the plat-



(a) v3 Interface



(b) v4 Interface

Figure 2: MovieLens Interface Versions after 17/02/2003

form, there are minimal to no user interactions or ratings associated with it. While the original dataset avoids this issue by construction as it includes only users who have rated at least 20 movies, our sampling process may inadvertently introduce this cold start problem. Therefore, to mitigate this challenge, we filter for users who had rated at least 20 movies and movies that had been rated at least 15 times in our final subsample.

In the end, our subsample includes 77,641 ratings across 2,246 movies from 2,836 users, with v3 and v4 interfaces. This results in a user-item interaction matrix of dimension 2,246 by 2,836, containing 6,292,015 missing values, which corresponds to approximately 98.78% sparsity. This smaller sample requires less computing time and memory usage compared to the original 25M dataset, which is suitable for limited available resources, while still enabling us to explore the implementation of softImpute on a extremely sparse high-dimensional matrix and evaluate the algorithm in the context of movie rating prediction.

4.3 Sample Properties

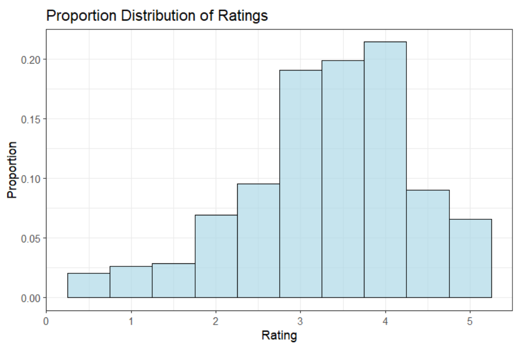


Figure 3: Probability Distribution of Movie Ratings

Statistics	Value
Mean	3.32
Median	3.37
Standard Dev.	1.01
Min	0.5
Max	5
Q1	3
Q3	4
Skewness	-0.62
Kurtosis	3.23

The descriptive statistics and distribution histogram in Figure 3 suggest that the ratings in our sample do not perfectly follow a normal distribution. The distribution of ratings is left-skewed, indicating that there are more ratings above the mean of 3.32 than below it. This observation aligns with the common belief regarding user behavior, which suggests a reluctance to assign low ratings to movies. The observed ratings vary from 0.5 to 5, showing that users utilized the full range of the rating scale. The kurtosis value of 3.23 is close to 3, indicates that the likelihood of extreme ratings, both high and low, is similar to what would be expected in a normal distribution. Therefore, while the distribution of movie ratings is left-skewed, it generally resembles a normal distribution but with a slight bias towards higher ratings and a tendency for more extreme values.

Table 1: Number of Movies in Each Genre

Genres	Number of Movies	Genres	Number of Movies
Drama	9437	Fantasy	1232
Comedy	6622	Children	1165
Thriller	3587	Mystery	1157
Romance	3037	Animation	1149
Action	2972	War	803
Crime	2212	Musical	627
Horror	2205	(no genres listed)	437
Adventure	1916	Western	393
Sci-Fi	1565	Film-Noir	199
Documentary	1394	IMAX	175

Note: There are movies that fall into many genres. We count the movies in all genres they belong to.

Our sample features 19 genres, namely: Drama, Comedy, Action, Thriller, Adventure, Sci-Fi, Crime, Romance, Fantasy, Mystery, Children, Horror, Animation, IMAX, War, Musical, Western, Documentary, and Film-Noir. According to MovieLens dataset, there are some movies that do not fall into any genre, represented as "no genres listed". Table 1 indicates that Drama and Comedy genres have the highest representation in the sample, with 9437 and 6622 movies respectively. Mid-tier genres such as Crime and Horror also have a substantial number of movies, reflecting their significant presence in the dataset. Genres such as Musical, Western, Film-Noir and IMAX are however relatively under-represented in the sample.

We investigate the popularity patterns of movie genres within our sample. First, we define "popularity" of movie genres in two ways: by the highest volume of ratings and by the highest average rating.

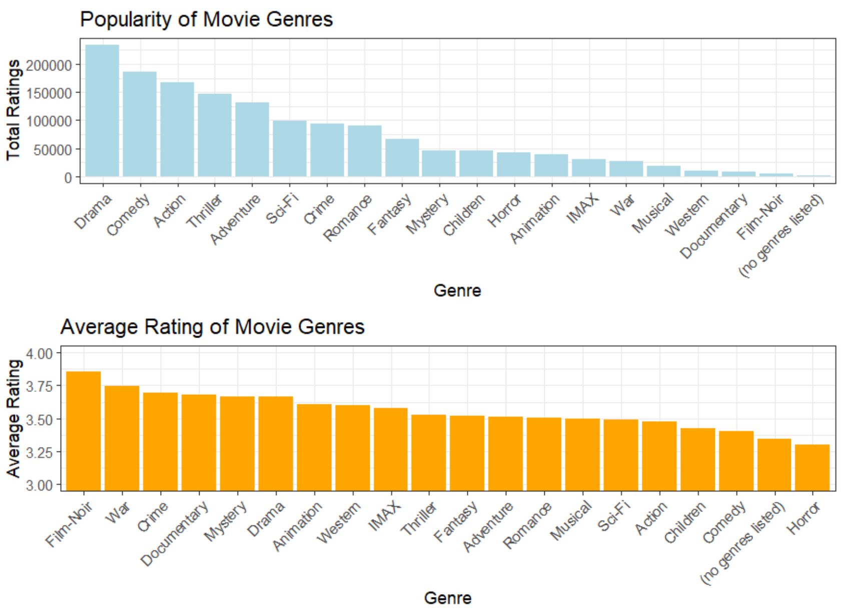


Figure 4: Movie Popularity by Genres

As illustrated in Figure 4, in terms of the number of ratings, Drama, Comedy, and Action are the most frequently rated genres, while Western, Documentary, and Film-Noir receive the fewest ratings. Nevertheless, the number of ratings a genre receives is influenced by the quantity of products produced in each genre, meaning that the number of ratings may not fully represent user rating preferences as there may be more movies that fall into certain genres than others. For instance, Film-Noir, despite having the fewest ratings, boasts the highest average rating of approximately 3.8 stars. In contrast, genres such as Comedy and Horror, while having a high volume of ratings, feature among the lowest average ratings across all genres. However, it is important to note that the difference in average ratings among genres is not substantial, as the average ratings of all genres fall within the 3-4 star range.

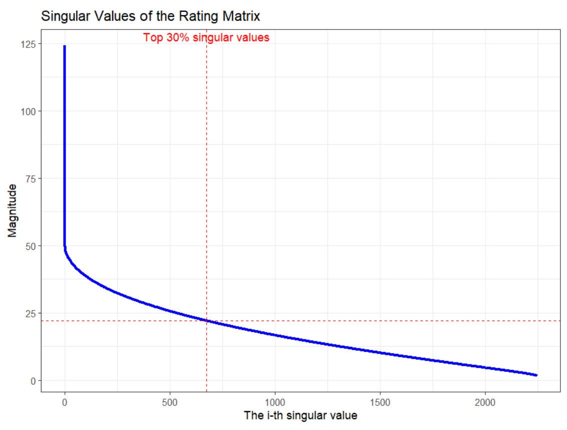


Figure 5: Singular Values of Rating Matrix

Before analyzing softImpute on MovieLens subsample, we consider the low rank assumption we previously mentioned in Section 3.1 on the chosen subsample. The low rank assumption is crucial for the softImpute algorithm, suggesting that most information in the rating matrix is redundant. Figure 5 illustrates that within our sample, the first few singular values of the user-item iteraction matrix have the highest magnitudes, indicating that a few factors dominate the structure of the rating matrix, supporting the low rank assumption. Notably, the first 700 singular values in the diagonal matrix Σ, which are the top 30% singular values with highest magnitudes, account for 54.61% of the total sum of singular values, meaning that they explain the majority of the information within the matrix. This confirms that the low rank assumption is reasonable for our subsample of the MovieLens dataset. It suggests that even with many ratings and potential differences, the main patterns and user preferences can be captured by a smaller number of key factors, hence validating the use of the softImpute algorithm.

5 Research Design

In this section, we explain the methodology used for splitting the data and the approach taken to tune the model parameters. The data splitting and model validation processes are performed to ensure that our model generalizes well to unseen data and does not overfit to training data.

We randomly split our sample into two datasets: a training dataset containing 80% of observed ratings, a test set containing the remaining 20%. We then use the train set to tune our parameters of the models, namely the regularization parameter λ and the rank constraint r, by finding the best values of these parameters that yields the lowest errors.

5.1 Evaluation Metrics

To assess the performance of the softImpute algorithm, we employ Root Mean Square Error (RMSE) as our primary metric. RMSE is popular in recommender systems because it heavily penalizes large errors, making it a more sensitive measure of prediction accuracy (Ricci et al., 2011). This metric is particularly relevant in movie recommendations, where extreme ratings hold significant importance. For example, suggesting a movie a user might strongly dislike can result in an undesirable negative user experience. By minimizing significant deviations between predicted and actual ratings, RMSE helps maintain user satisfaction with the recommender system (Koren et al., 2009).

Taking inspirations from Dax (2014), we define our metric as following: Previously in Section 3.2, we define the index set of observed entries to be Ω. In our application, Ω is split into two distinct sets: a training set $\tilde{\Omega}$, and a test set $\hat{\Omega}$ such that:

$$\Omega = \tilde{\Omega} \cup \hat{\Omega} \quad \text{and} \quad \tilde{\Omega} \cap \hat{\Omega} = \emptyset$$

Then, the algorithm will achieve imputing using $\tilde{\Omega}$, generating predicted values p_{ij} for all entries in $(i, j) \notin \tilde{\Omega}$. Therefore, since $\forall (i, j) \in \hat{\Omega}$, both p_{ij} (predicted ratings) and x_{ij} (actual ratings) are available, the prediction error is estimated from this set. Mathematically, we formulate RMSE by:

$$RMSE = \sqrt{\frac{1}{\hat{v}} \sum_{(i,j)\in\hat{\Omega}} (p_{ij} - x_{ij})^2}$$

where $\hat{\Omega}$ is the index set of observed entries in the test set, $\hat{v} = |\hat{\Omega}|$ denotes the number of observed values in the test set, p_{ij} are predicted ratings, and x_{ij} are actual ratings.

5.2 Parameters Tuning using K-Fold Cross Validation

To fine-tune the parameters of our softImpute model, we employ grid search in conjunction with 5-fold cross-validation. Grid search allows us to systematically explore a specified range of hyperparameter values for the regularization parameter λ and the rank constraint r. By iterating over a defined grid of values, we can identify the combination of λ and r that minimizes the RMSE on the validation sets.

First, we choose the range for the regularization term λ. Using a built-in function of the softImpute R Package, we identify the smallest value of λ that shrinks the solution matrix to rank zero, defined as λ_0. This value λ_0, which is computed to be 11.71, is the maximum value that λ can take since any value larger or equal to λ_0 will shrink the rank of solution matrix to 0 (Hastie et al., 2015).

$$\begin{aligned}\lambda_0 &= \min\{\lambda \mid \operatorname{rank}(X^\lambda) = 0\} \\ &= \max\{\sigma_1, \sqrt{\sum_{i=1}^{n} \sigma_i^2}\}\end{aligned}$$

where X^λ is the matrix obtained after applying soft-thresholding operator with parameter λ, σ are singular values where σ_1 is the largest singular value in the matrix. For the rank of the matrix to become zero, λ must be at least as large as the largest singular value σ_1. Additionally, considering the Frobenius norm provides a bound for when the combined effect of all singular values is nullified. Hence, λ_0 is the maximum of these two quantities, ensuring all singular values are zeroed out, making the rank of the matrix zero.

Thus, for our grid search, we consider five values of $\lambda \leq \lambda_0$, specifically ranging from 0.5 to λ_0 in equal increments. This involves iteratively computing the errors for $\lambda = 0.5, 3.30, 6.11, 8, 91, 11.71$, while adhering to the rank constraints that we determine.

Secondly, the rank constraints are selected for our grid search. Our selection method is rather simplistic, by setting $r = 20, 40, 60, 80, 100$. We note that for our problem, there is a condition stated in Section 3.2 that the rank constraint $r \leq min(m, n) = min(2246, 2836) = 2246$. However, since using higher rank constraints increase running time of the algorithm significantly, we restrict the range of r to the mentioned values. With the defined regions of λ and r, we combine grid search approach with 5-fold cross validation to tune parameters for softImpute.

In 5-fold cross-validation, the training set is divided into five equal parts, or "folds". The training set of observed entries $\hat{\Omega}$ is randomly split into k disjoint subsets of similar size: $\hat{\Omega}_p$, where $p = 1, 2, ..., 5$. The model is trained on four folds and validated on the remaining fold. This process is repeated five times, with each fold serving as the validation set once. In specific, at the pth round, Ω_p is used as a test set, and the union of the other 4 subsets serve as the training set (Dax, 2014). The average RMSE across all five folds is computed to ensure more robust parameter tuning:

$$
\begin{aligned}
\text{Average RMSE} &= \frac{1}{5}\sum_{p=1}^{5}\text{RMSE}_p \\
&= \frac{1}{5}\sum_{p=1}^{5}\sqrt{\frac{1}{|\Omega_p|}\sum_{(i,j)\in\Omega_p}(x_{ij}-p_{ij})^2}
\end{aligned}
$$

where Ω_p is the set of observed indices in test fold p, $|\Omega_p|$ is the number of entries in test fold p, x_{ij} is the actual rating value, and p_{ij} is the predicted rating.

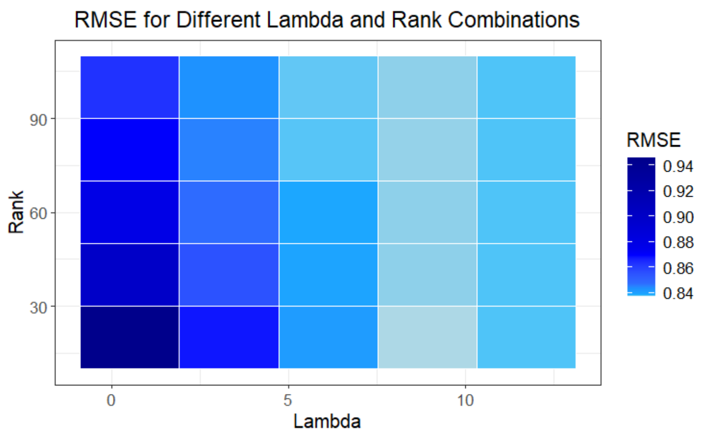


Figure 6: Parameters Tuning using Grid Search & 5-Fold Cross Validation

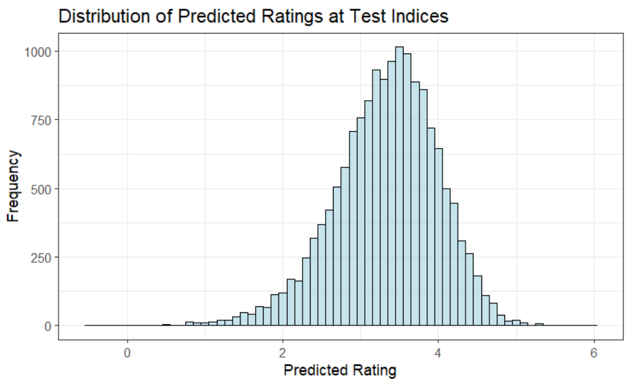
The tuning process identifies $\lambda = 8.91$ and $r = 20$ as the optimal combination, resulting in the lowest RMSE of 0.837 among all tested combinations[2]. For the next section of analysis, we fit the model on entries in test set with this combination of parameters: $\lambda = 8.91$ and $r = 20$.

6 Empirical Results

In this section, we report the results as we apply the softImpute-ALS algorithm to a subsample of the MovieLens dataset using tuned parameters. The output of the softImpute-ALS algorithm is a completed matrix containing real number ratings, which are continuous rather than adhering to the discrete, half-integer rating scale ranging from 0.5 to 5. This deviation from the original rating scale means that ratings can take on fractional values, which is not ideal as it does not align with the intended scale of MovieLens rating system.

[2]For any combination with $\lambda \geq \lambda_0$, RMSE for those models are fixed at 0.839 since the rank of the solution matrix is shrunk to 0 for all rank constraints with such values of λ.

6.1 Distribution of Rating Predictions



Statistics	Value
Mean	3.33
Median	3.37
Standard Dev.	0.66
Min	-0.07
Max	6.22
Skewness	-0.41
Kurtosis	3.61

Figure 7: Distribution of Predicted Ratings

Notably, as illustrated in Figure 7, at the indices of test set, the mean of the predicted ratings is very close to the mean of the actual ratings, although the median of the predicted ratings is slightly lower than that of the actual ratings. Despite this, both predicted and actual ratings exhibit a negative skewness, indicating a left-skewed distribution showing a tendency for higher ratings. The kurtosis of the predicted ratings is slightly higher than that of the actual ratings but remains close to the kurtosis of a normal distribution, suggesting similar tail behavior. In general, softImpute-ALS predictions at the test indices reflect its ability to somewhat capture the underlying distribution characteristics of the ratings to make meaningful predictions.

However, an issue arises with the minimum predicted rating being -0.07 and the maximum being 6.22, both of which fall outside the valid movie rating scale of 0.5 to 5. Specifically, a rating below 0.5 appears 5 times and a rating above 5 occurs 36 times, which represents approximately 0.03% and 0.23% of the 15,528 total predictions, respectively. While these occurrences account for a relatively small proportion of the predictions, they highlight potential areas for refinement in the algorithm to ensure all predicted ratings fall within the acceptable range, thereby maintaining the integrity of the analysis.

To address these out-of-scale predictions and the continuous nature of the predicted ratings, we implement a post-processing method. In this method, any predicted value less than 0.5 is rounded up to 0.5, and any value above 5 is rounded down to 5. For the remaining values, we round them to the nearest half-integer, such as a prediction of 3.6 will be rounded to 3.5. This approach ensures that all predictions adhere to the valid movie rating scale and align more closely with the discrete nature of user ratings. By applying this post-processing step, we aim to enhance the practical applicability of the model's predictions and further examine their characteristics to put them in perspective of real-world movie recommender systems.

6.2 RMSE and Segmented RMSE

As Table 2 suggests, our softImpute-ALS model achieves a RMSE of 0.830 with the unclipped results. This indicates that, for instance, if a movie has a true rating of 4.0, the predictions made by the model would typically fall within the range of approximately 3.2 to 4.8. We find that the post-processing process introduces some additional errors, bring-

Table 2: RMSE Performance Metrics

Model	RMSE
Naive Median-Based Model	0.938
SoftImpute Overall	0.830
Post-Processed SoftImpute Overall	0.841
High Rating Users	0.737
Low Rating Users	0.876
Users with More Ratings	0.819
Users with Fewer Ratings	0.841
Movies with More Ratings	0.823
Movies with Fewer Ratings	0.836
High Average Rating Movies	0.812
Low Average Rating Movies	0.840

ing the model's RMSE to 0.841. Although clipping the predictions at the extreme ends 0.5 and 5 reduces some prediction errors, this reduction is relatively small since there are only a few observations that fall out of the rating scale. However, the rounding process in our post-procssing method to convert softImpute's predictions to the movie rating scale inadvertently introduces additional errors that increase the model's RMSE.

To benchmark our model's performance, we compare it to a naive model that predicts test ratings using the median ratings of each movie. This naive median-based model yields a significantly higher RMSE of 0.938. Thus, the post-processed softImpute-ALS model outperforms the naive model by approximately 10.40%. This underscores the effectiveness of the softImpute-ALS model in providing more accurate predictions than naive simple heuristics.

To further analyze the model's performance, we segment the RMSE results based on user and movie characteristics. We examine the segmented performance by segmenting users based on their average ratings and their rating frequency, and also segmenting movies based on their average ratings and the number of ratings they have received. Users and movies are divided into "more" and "fewer" ratings groups based on the 75th and 25th percentiles of their rating counts, and are divided into "high rating" and "low rating" based on the similar percentiles of average ratings.

The RMSE for high average rating users is around 0.737, while for low average rating users, it is approximately 0.876. This significant discrepancy suggests that the model may be biased towards users who generally rate items more positively. This could mean that the model has learned patterns more effectively from higher ratings, since the rating distribution is shown to be left-skewed, such ratings are more frequent. Furthermore, for users that rate more frequently than others, the RMSE is 0.819 compared to 0.841 for users with fewer ratings, suggesting slightly improved performance with more data points per user. This suggests a potential bias where the model performs better for well-represented users, since it is easier for model to learn their preferences accurately.

Movies with more ratings have an RMSE of approximately 0.823, whereas movies with fewer ratings have an RMSE of 0.836, highlighting the model's slightly better performance with more frequently rated movies. Similar to user ratings, movies with more ratings also provide a richer dataset for the model to learn from, leading to better performance, although this improvement is not too significant in this case. Additionally, high average rating movies have an RMSE of 0.812, while low average rating movies have an RMSE of around 0.840. This suggests that the model might be biased towards predicting ratings more accurately for popular or well-liked movies. This bias can be problematic as it may result in less accurate predictions for niche or less popular movies.

These segmented results suggest that the model tends to perform better for users and items with more data and higher average ratings. While the predictions generally perform well for those well-represented segments, this performance can be problematic since it may fail to adequately capture the preferences and ratings of less represented segments. In a movie recommender system, this bias could lead to a lack of diversity in recommendations, disadvantaging lesser-known movies and users with unconventional tastes.

6.3 Overall Prediction Bias

Prediction bias is defined as the difference between the post-processed predicted ratings and the actual ratings. It is a crucial metric in our analysis because it helps us understand whether our model systematically overestimates or underestimates user preferences. For overestimation, this prediction bias will be positive, while for underestimation, the bias will be negative. By examining the prediction bias, we can identify any consistent tendencies in the matrix completion softImpute-ALS model's predictions that might affect the overall accuracy of a recommender system..

6.3.1 Descriptive Statistics of Prediction Bias

Table 3: Descriptive Statistics of Unprocessed Prediction Bias

	Mean	Median	Minimum	Maximum	Standard Dev.
Bias	0.0008	-0.05	-4.19	4.96	0.83

Table 4: Descriptive Statistics of Post-Processed Prediction Bias

	Mean	Median	Minimum	Maximum	Standard Dev.
Bias	0.0009	0	-4	4.5	0.84

Table 3 and 4 report the prediction bias statistics for our softImpute model. The mean prediction bias of the unprocessed predictions is 0.0008, while the post-processed predictions show a mean bias of 0.0009. Although the mean bias is close to zero in both cases, indicating that the average of the prediction errors is balanced, it is important to note that this does not necessarily mean the predictions are very close to the true ratings. Instead, the small mean bias suggests that on average, the magnitudes of overestimations and underestimations tend to cancel each other out.

A notable improvement is observed in the median bias, which shifts from -0.05 in the unprocessed predictions to 0 in the post-processed predictions. This shift indicates that rounding helps in centering the bias distribution around zero. Remarkably, the minimum prediction bias improves from -4.19 to -4 through post-processing. Similarly, the maximum bias reduces from 4.96 to 4.5. These reductions in the extremities of bias demonstrate that rounding helps mitigate the most severe prediction errors. Furthermore, the standard deviation of bias remains relatively stable, with a slight increase from 0.83 in the unprocessed predictions to 0.84 in the post-processed predictions. This consistency suggests that the overall variability in prediction errors does not change significantly with rounding.

However, the presence of significant outliers, with minimum bias improving from -4.19 to -4 and maximum bias reducing from 4.96 to 4.5, highlights that while rounding reduces the extremity of prediction errors, some large deviations still occur. For instance, a minimum bias of -4 could mean that if a user would truly dislike a movie and rate it a 0.5, the model might incorrectly predict a rating of 4.5. Conversely, a maximum bias of 4.5 could indicate that while a user would rate a movie 5, the model might erroneously predict a rating of 0.5. This is particularly problematic since for movie recommendations, user experience can be negatively affected by the wrong recommendations of a movie that the user would greatly dislike, while movies that maybe suitable will not be recommended.

6.3.2 Misprediction Tendency

The graph presented in Figure 8 offers a detailed depiction of prediction biases for given true ratings. Utilizing violin plots, each plot illustrates the distribution of prediction biases associated with specific true ratings, emphasizing the density and variability of these biases.

The analysis reveals a pronounced overestimation bias for movies with true ratings of 0.5 and 1. For these lower ratings, softImpute frequently assigns higher ratings than true ratings. Specifically, the majority of prediction biases for true ratings of 0.5 are concentrated around 1.5 to 2, while for true ratings of 1, biases cluster around 1. There is also a considerable spread in prediction biases for these ratings, where the prediction biases range widely from 0 to around 4, indicates a high degree of variability and inconsistency in the model's predictions for these lower ratings.

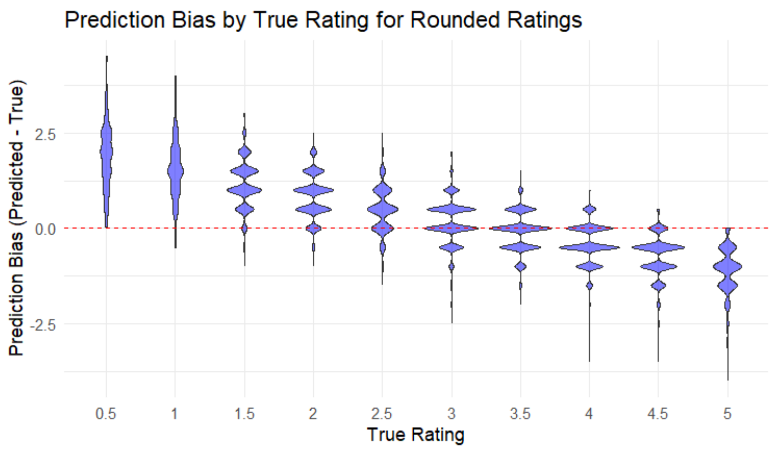
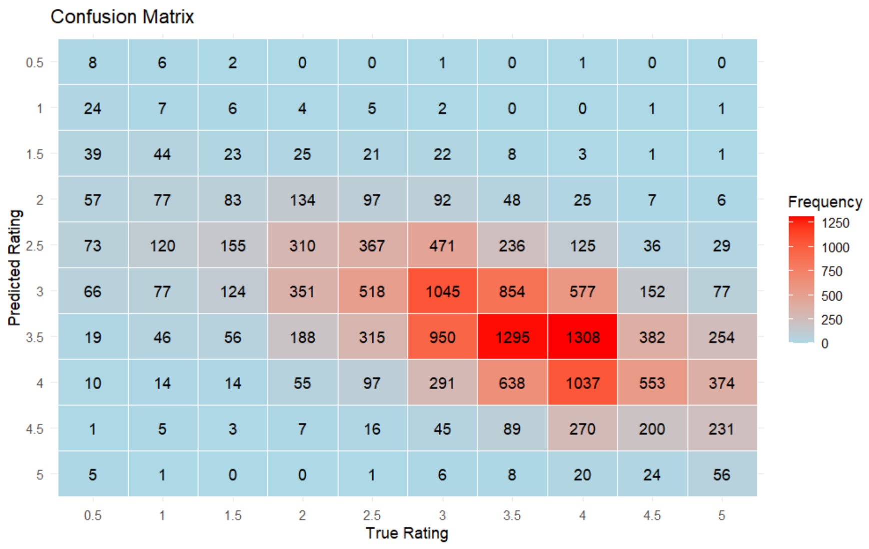


Figure 8: Prediction Bias by True Ratings

As true ratings vary from 1.5 to 5, the bias distributions show a trend towards centralization around the zero line, signifying improved prediction accuracy. Particularly, the relatively balanced and symmetrical bias distributions for mid-range ratings (2 to 3.5) suggest that the model's predictions are relatively accurate for these ratings, exhibiting less variability compared to lower ratings. Despite this trend, instances of both overestimation and underestimation are still observed.

However, the analysis identifies significant variability for higher ratings (4 to 5). The density of the violin plots for these ratings indicates that most prediction biases are also centered slightly under zero, suggesting that there is a tendency to underpredict by a small margin compared to the true ratings. However, we observe that there remain some instances of extreme biases. This highlights the occasional significant underestimations for movies at the high end of the rating scale.



Predicted Rating \ True Rating	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
0.5	8	6	2	0	0	1	0	1	0	0
1	24	7	6	4	5	2	0	0	1	1
1.5	39	44	23	25	21	22	8	3	1	1
2	57	77	83	134	97	92	48	25	7	6
2.5	73	120	155	310	367	471	236	125	36	29
3	66	77	124	351	518	1045	854	577	152	77
3.5	19	46	56	188	315	950	1295	1308	382	254
4	10	14	14	55	97	291	638	1037	553	374
4.5	1	5	3	7	16	45	89	270	200	231
5	5	1	0	0	1	6	8	20	24	56

Figure 9: Confusion Matrix

In addition to the violin plots in Figure 8, the confusion matrix in Figure 9 provides a comprehensive analysis of the model's prediction performance. Each cell in the matrix quantifies the frequency with which a particular true rating is predicted as another rating, with the diagonal entries are the accurate rating predictions.In particular, the diagonal cells, especially for true middle ratings of 3, 3.5, and 4, exhibit the highest frequencies. This indicates that the model most accurately predicts these mid-range ratings, corroborating the findings from the violin plots, where the bias distributions for these ratings are relatively centered and symmetrical.

The confusion matrix also reveals that true ratings of 0.5 and 1 are often significantly overestimated. For instance, the model predicted a true rating of 1 as 2 in 77 cases and as 2.5 in 120 cases. Similarly, true ratings of 0.5 are rarely predicted accurately, with predictions scattered across higher ratings with most predictions lie around 2 to 3. This aligns with the significant overestimation bias observed in the violin plots for these low ratings. The scattered predictions for extreme ratings (0.5 and 5) indicate that the model struggles to accurately predict these extreme values, often estimating them closer to the mid-range values. This bias towards central ratings is evident in the off-diagonal cells, where true low and high ratings are predicted as more moderate values.

Furthermore, the confusion matrix highlights the variability in predictions for mid-range ratings. While the model generally performs well for ratings around 3 to 4, there are notable frequencies in the off-diagonal cells, indicating that the model sometimes predicts neighboring ratings instead of the exact true rating. For instance, a true rating of 3 was predicted as 2.5 in 471 instances and as 3.5 in 950 instances. The practical implications of these findings nevertheless depend on user tolerance and the specific requirements of the recommender system. In some cases, "close" predictions can still be valuable. For example, a predicted rating of 4.5 may indicate sufficient user interest for an item that the user would rate 5 out of 5. However, the direction and magnitude of these prediction errors must be carefully evaluated to determine how "close" these mispredictions can be tolerated so that their impacts on user satisfaction and recommendation quality are not detrimental to the system.

From an application perspective, it is generally more preferable to have slight underpredictions in movie ratings rather than overpredictions (Srinivasan, 2020). Underpredictions might result in potentially enjoyable movies being overlooked, leading to missed opportunities for user engagement. However, significant over-predictions, such as predicting a true rating of 0.5 as a 5, can lead to user dissatisfaction and distrust in the system. Users may be recommended movies that they greatly dislike, leading to a negative viewing experience. Therefore, while the model should aim to minimize both types of errors, intuitively, underpredictions should be more tolerable than overpredictions in movie recommendations.

Overall, the confusion matrix complements the analysis from the violin plots by providing a granular view of prediction frequencies, reinforcing the observed biases and variability. Statistically, the model's overall accuracy is 26.78%, with a Kappa value of 0.116. While these metrics indicate that the model's predictions are slightly better than random guessing, there is significant room for improvement. It is important to note that this accuracy metric is based on exact matches between predicted and true ratings without considering "close" predictions. Thus, the low accuracy percentage does not necessarily imply that SoftImpute-ALS is unsuitable for movie recommender systems.

6.4 Prediction Bias By Genres

In this section, we analyze the prediction bias segmented by genres to determine if any particular patterns emerge. By examining the biases specific to each genre, we can identify trends and tendencies in the model's performance that may be inherent within one genre.

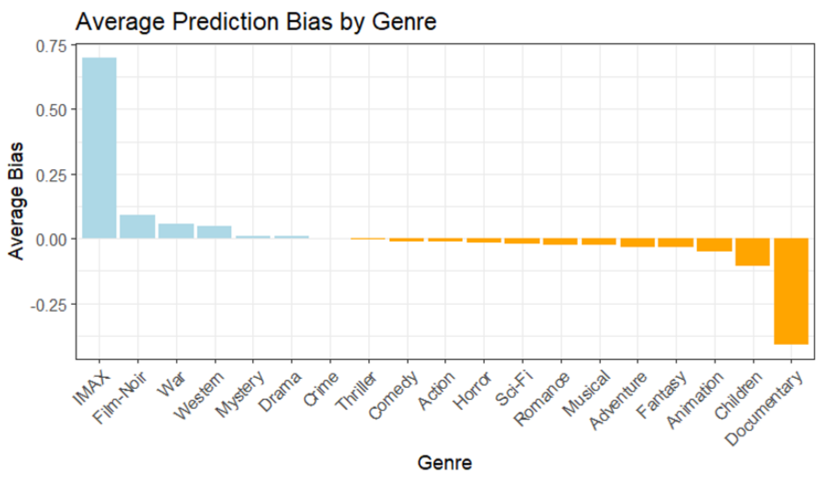


Figure 10: Average Prediction Bias by Genre

Figure 10 illustrates the average prediction bias for movie ratings across different genres. We observe that, for most genres, the average bias is approximately zero, with notable exceptions for IMAX and Documentary, which exhibit biases of 0.7 and -0.4 respectively. However, the near-zero mean prediction biases for most genres does not imply that the predicted ratings are close to the true ratings. Instead, it indicates that, on average, the extent and frequency of underestimations and overestimations within these genres offset each other in the mean. This balancing effect results in an overall average bias near zero, despite potential inaccuracies in individual predictions.

The 0.7 and -0.4 prediction biases for IMAX and Documentary indicate that on average, IMAX movie ratings tend to be overpredicted more than underpredicted, whereas Documentary ratings are typically underpredicted. We notice that IMAX movies are quite under-represented in our sample with only 175 movies present. Similar to the patterns observed in low-rating users and movies in Section 6.2, one could argue that the mispredictions are typically attributed to the lack of training data for a specific genre. However, this explanation does not hold for all genres. For instance, despite showing a relatively high average underprediction bias, Documentary movies are actually well-represented in the dataset, with 1,394 movies included. This suggests that factors other than sample size, such as the inherent characteristics of the genres, may contribute to the observed prediction biases.

To analyze the prediction behaviors within genres even further, we examine patterns in Figure 11 and Table 5 that report the prediction bias tendency and magnitude of mispredictions for different genres.

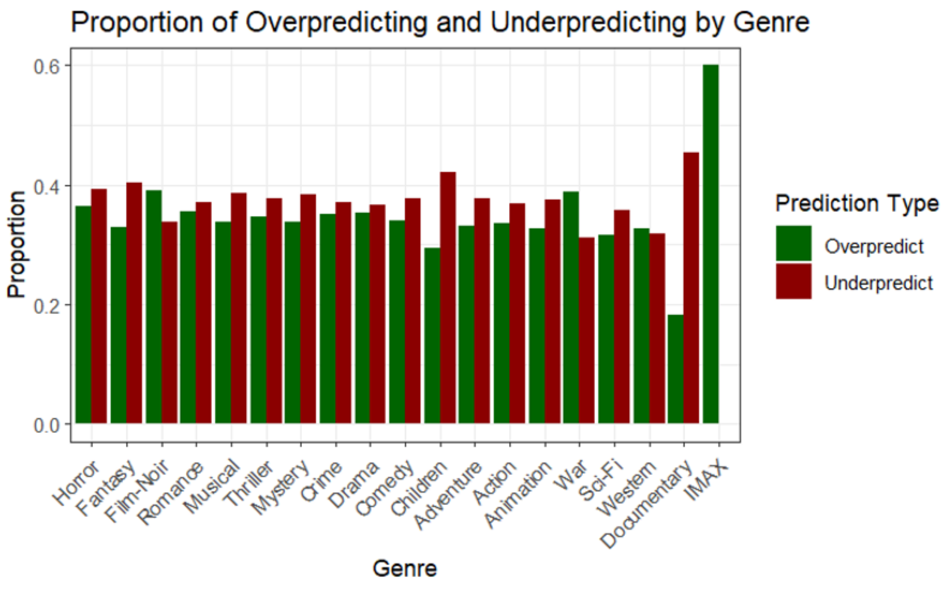


Figure 11: Proportion of Overpredictions and Underpredictions by Genre

For most genres, the tendency for both underestimations and overestimations is fairly balanced, with each type of misprediction accounting for approximately 30-40% of all predictions, and around 30% of the predictions being accurate. Specifically, IMAX, Film-Noir, and War movies are the top three genres that most frequently exhibit overpredictions. IMAX overpredicts about 60% of the time, with an average overestimation of 1.17, while Film-Noir and War overpredict approximately 40% of the time. In contrast, although Documentary movies also show a high average overprediction of about 1, this only accounts for 18% of Documentary rating predictions.

A significant portion of predictions for Documentary movies are underpredictions, constituting 45% of the predictions with an average magnitude of -1.3, which is notably larger than most genres in our sample. Fantasy and Children movies also have a high proportion of underpredictions, around 40% of total predictions. However, the magnitude of the underpredictions for these genres is relatively moderate, which is why this trend is not prominently reflected in the average prediction bias. Remarkably, among the few IMAX predictions, there are no underpredictions. Out of five predictions for IMAX movies, three are overpredictions and two are accurate. However, this finding is not particularly significant given the extremely small sample size for IMAX predictions.

The general trends in Table 5 suggests that many genres like Drama, Crime, Thriller, Comedy, and Action show balanced proportions of over- and underpredictions around 33-38%, indicating that both types of errors occur with similar frequency. The average magnitudes of prediction errors in a genre generally hovers around $\pm$ 0.9, except for extreme cases like IMAX and Documentary. Most genre have correct prediction proportions ranging around 28-38%, indicating a moderal level of accuracy.

		Overpred.		Underpred.		Correct Pred.
Genre	**Total Pred.**	**Proportion**	**Magnitude**	**Proportion**	**Magnitude**	**Proportion**
IMAX	5	0.60	1.17	0.00	-	0.40
Film-Noir	77	0.39	0.93	0.34	-0.81	0.27
War	237	0.39	0.84	0.31	-0.86	0.30
Horror	348	0.36	0.88	0.39	-0.86	0.24
Romance	877	0.35	0.90	0.37	-0.92	0.27
Drama	2041	0.35	0.94	0.37	-0.88	0.28
Crime	668	0.35	0.95	0.37	-0.89	0.28
Thriller	1280	0.35	0.95	0.38	-0.88	0.28
Comedy	1597	0.34	0.94	0.38	-0.88	0.28
Musical	251	0.34	0.95	0.39	-0.89	0.27
Mystery	381	0.34	1.00	0.38	-0.86	0.28
Action	1158	0.34	0.93	0.37	-0.88	0.30
Adventure	865	0.33	0.88	0.38	-0.86	0.29
Fantasy	484	0.33	0.89	0.40	-0.81	0.27
Western	110	0.33	0.97	0.32	-0.84	0.35
Animation	306	0.33	0.90	0.38	-0.92	0.30
Sci-Fi	675	0.32	0.93	0.36	-0.88	0.33
Children	562	0.29	0.89	0.42	-0.88	0.28
Documentary	11	0.18	1.00	0.45	-1.30	0.36

Table 5: Bias Tendency, Proportion and Magnitude by Genre

Notes: The proportions of over- and under-predictions for each genre are calculated based on the total number of predictions for that genre. The magnitudes of each type of misprediction represent the prediction bias, averaged across all instances of that type of misprediction within a genre. Note that some movies belong to multiple genres, and these results do not account for the overlap between genres.

7 Discussion

7.1 Result Summary

This study set out to evaluate how well the SoftImpute-ALS algorithm can predict movie ratings in the context of a movie recommender system. By using a subsample of the MovieLens 25M dataset, we focused on how effectively the algorithm handles data sparsity and maintains recommendation accuracy.

SoftImpute-ALS generated predictions of continuous scale with a few predictions fall out of the rating scale, as opposed to the true discrete rating scale of MovieLens. Therefore, we required post-processing steps to clip the results that exceed the extreme values of the scale and convert the ratings to the true scale from 0.5 to 5 stars.

The overall performance of the SoftImpute-ALS algorithm, as indicated by the Root Mean Square Error (RMSE), showed a significant improvement over a simple median-based model. The SoftImpute-ALS model had an RMSE of 0.830, compared to 0.938 for the naive model. The segmented RMSE analysis offered deeper insights into the algorithm's performance across different user and movie characteristics. The model performed better for users who tend to give higher ratings and for those who rated more frequently, suggesting that the algorithm benefits from having more comprehensive user data. Similarly, movies with higher average ratings and more frequent ratings also showed lower RMSE values. This indicates that the SoftImpute-ALS algorithm is particularly effective when there is ample data available for a user or an item, illustrated by the fact the characteristic groups mentioned above are relatively more well-represented in our sample.

Our analysis of prediction bias revealed some systematic tendencies. For example, the algorithm tended to overestimate ratings for movies that received lower ratings and slightly underestimate ratings for higher-rated movies. This bias can be partly explained by the inherent challenge of accurately predicting extreme values in sparse datasets. Regardless, the prediction bias is particularly problematic in the context of movie recommender systems as overestimating low ratings could result in users being recommended movies they are less likely to enjoy, while underestimating high ratings might cause users to miss out on highly-rated content that fits their preferences.

When looking at genre-specific performance, the SoftImpute-ALS algorithm showed a balanced prediction bias for most genres, with notable exceptions like IMAX and Documentary films, which had significant overestimation and underestimation biases, respectively. This genre-specific analysis highlights the importance of considering the unique characteristics of different movie genres when developing and evaluating recommender systems. The variability in prediction accuracy across genres suggests that additional refinement and genre-specific adjustments might be necessary to improve the overall effectiveness of the recommendation algorithm.

Overall, these findings demonstrate that while the SoftImpute-ALS algorithm performs decently in sparse data conditions, there are still many areas for improvement, especially in handling extreme rating values and specific genres.

7.2 Implications

The findings of this study contribute to the theoretical understanding of matrix completion and collaborative filtering in recommender systems. By demonstrating the effectiveness of the SoftImpute-ALS algorithm in handling data sparsity and predicting ratings with a subsample of the MovieLens dataset, this research supports the viability of softImpute-ALS as a matrix completion technique in real-world applications, specifically in predicting movie preferences. The observed performance and addressed biases provide valuable insights into the strengths and limitations of the technique, highlighting how the algorithm can be assessed to be implemented in a movie recommender system.

The SoftImpute-ALS algorithm demonstrates strong potential for movie recommender systems. Its ability to handle data sparsity effectively and predict accurately makes it a valuable tool for platforms that deal with high-dimensional user-item interaction data. This is particularly relevant for systems like Netflix or Amazon, where data sparsity is a common issue due to the vast number of items available compared to the number of ratings provided by users.

Regardless, there are weaknesses of softImpute-ALS in movie ratings predictions that need to be addressed for it to be reliable in recommender systems. For instance, there needs to be a focus on reducing the prediction biases at extreme ratings. This could involve refining the algorithm to better handle extreme rating values and ensuring a more balanced prediction across the rating scale. Furthermore, it may also be beneficial to address some genre-specific adjustments to help mitigate the observed biases and improve recommendation accuracy for all types of movies.

7.3 Limitations and Future Research

This thesis has several important limitations that merit discussion and offer avenues for future research. Primarily, the study relies solely on explicit feedback, particularly movie ratings, to make predictions. Although explicit feedback provides clear and measurable user preferences, it accounts for only a small portion of the available interaction data. Including implicit feedback, such as viewing history, search queries, and time spent on content, can give a more comprehensive picture of user preferences. Recent research has demonstrated that incorporating implicit feedback can greatly enhance the accuracy of recommendations and user satisfaction (See Koren et al. (2009); Liu et al. (2023)).

Secondly, this study only examines a subsample of the MovieLens dataset. Real-life data is typically much more high-dimensional and complex, involving a broader array of user interactions and preferences. The subsample used may not fully capture the diversity and complexity of the entire dataset, potentially limiting the generalizability of the findings. Future research should aim to validate the algorithm's performance on larger, more diverse datasets to ensure robustness and scalability in real-world applications.

Moreover, other weaknesses of the study include the observed prediction biases and the limited handling of extreme rating values. The SoftImpute-ALS algorithm tends to overestimate low ratings and underestimate high ratings, which can negatively impact user satisfaction. Addressing these biases is crucial for improving recommendation accuracy. Future research could explore methods such as modifying the existing version of softImpute-ALS into a weighted version where it places more weights on minimizing the loss function for extreme ratings, or examining hybrid models that combine collaborative filtering with content-based filtering or deep learning techniques to enhance the prediction of extreme values (See for example Zhao et al. (2020)). More sophisticated regularization methods could also be beneficial in refining the predictions for extreme values and reducing bias.

Another meaningful extension to this thesis could be the investigation of the algorithm's robustness to shilling attacks. Shilling attacks, also known as profile injection attacks, involve malicious users inserting fake profiles into a recommender system to manipulate the recommendation outcomes for specific items (LeBlanc et al., 2024). These attacks can significantly distort the quality of recommendations and undermine the system's credibility. To simulate such a study, it would be necessary to create synthetic user profiles with strategically biased ratings and inject them into the dataset. The effectiveness of the softImpute-ALS algorithm in handling these attacks would then be evaluated by analyzing changes in recommendation accuracy and bias. This would involve comparing the performance of the algorithm with and without the presence of shilling attacks, utilizing metrics such as Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) to quantify the extent of the impact.

8 Conclusion

In conclusion, this thesis has demonstrated that the SoftImpute-ALS algorithm efficiently predicts movie ratings by effectively handling data sparsity. However, the observed biases and limitations indicate that there is room for further refinement and optimization. Future research should focus on addressing these issues, exploring hybrid models, and tailoring algorithms to specific genres to improve the overall effectiveness of movie recommender systems. The insights gained from this study provide an insight into softImpute's capability to deal with characteristics inherent in movie rating context, with the ultimate goal of delivering more accurate and personalized recommendations that enhance user satisfaction and engagements.

References

Aydin, Z. E. and Ozturk, Z. K. (2021). Prediction length of stay in intensive care unit in the presence of missing data. *Artificial Intelligence Theory and Applications*, 1(2):1–10.

Candès, E. J. and Tao, T. (2009). The power of convex relaxation: Near-optimal matrix completion. *CoRR*, abs/0903.1476.

Chen, Z. and Wang, S. (2022). A review on matrix completion for recommender systems. *Knowledge and Information Systems.*

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM.

Dax, A. (2014). Imputing missing entries of a data matrix: A review. *Advances in Linear Algebra & Matrix Theory*, 4(3):98–122.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19.

Hastie, T., Mazumder, R., Lee, J. D., and Zadeh, R. (2015). Matrix completion and low-rank svd via fast alternating least squares. *Journal of Machine Learning Research*, 16(1):3367–3402.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering.

Jannach, D., Karakaya, Z., and Felfernig, A. (2012). Recommender systems: From algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–104.

Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. (2014). Recommender systems in light of big data.

Kidzinski, and Kidzinski, D. (2017). Sweetrs: Dataset for a recommender systems of sweets. Accessed: 2024-06-27.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 426–434. ACM.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

LeBlanc, P. M., Banks, D., Fu, L., Li, M., Tang, Z., and Wu, Q. (2024). Recommender systems: A review. *Journal of the American Statistical Association*, 119(545):773–785.

Lee, J., Kim, S.-I., Lebanon, G., and Singer, Y. (2013). Local low-rank matrix approximation. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 82–90. ICML.

Liu, Y., Wu, H., Rezaee, K., Khosravi, M. R., Khalaf, O. I., Khan, A. A., Ramesh, D., and Qi, L. (2023). Interaction-enhanced and time-aware graph convolutional network for successive point-of-interest recommendation in traveling enterprises. *IEEE Transactions on Industrial Informatics*, 19(1):635–643.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322.

Nilashi, M., Bagherifard, K., Ibrahim, O., Alizadeh, H., Nojeem, L. A., and Roozegar, N. (2013). Collaborative filtering recommender systems. *Research Journal of Applied Sciences, Engineering and Technology*, 5(16):4168–4182.

Rennie, J. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 713–719. ACM.

Ricci, F., Rokach, L., and Shapira, B. (2011). *Recommender Systems Handbook*. Springer.

Srebro, N., Rennie, J., and Jaakkola, T. (2005). Maximum margin matrix factorization. In *Advances in Neural Information Processing Systems*, volume 17, pages 1329–1336.

Srinivasan, P. (2020). Machine learning for the analysis and prediction of film performance (filmchain). Technical report, Imperial College London.

Sun, Y., Kürüm, E., Demir, Ï., and Ng, M. K. (2020). Matrix completion methods for the total electron content video reconstruction. *arXiv preprint arXiv:2012.01618*.

Zhao, X., Zhu, Z., Zhang, Y., and Caverlee, J. (2020). Improving the estimation of tail ratings in recommender system with multi-latent representations. In *WSDM '20: Proceedings of the 13th International Conference on Web Search and Data Mining*, New York, NY, USA. ACM.

9 Appendix

A R Implementation

The code utilized for this thesis is publicly available at: `https://github.com/pdbn/Predicting-MovieLens-with-softImpute`.

B Extra Analysis - Sample Ratings

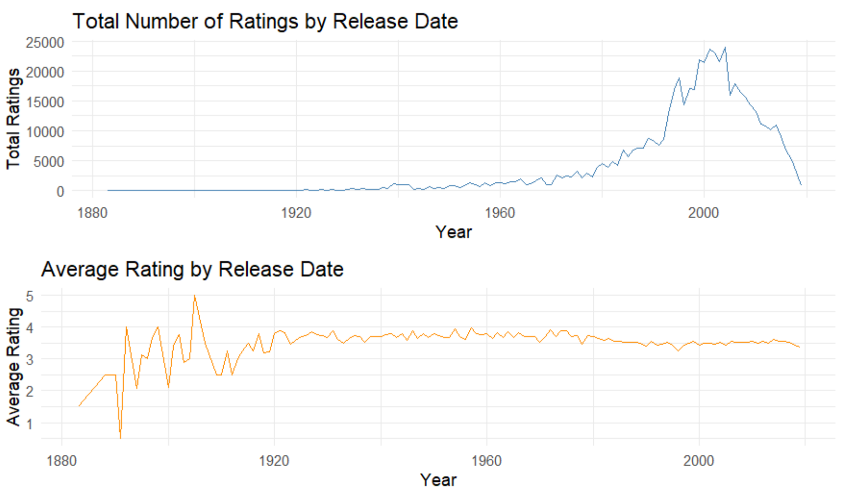


Figure 12: Popularity of Movies by Release Date

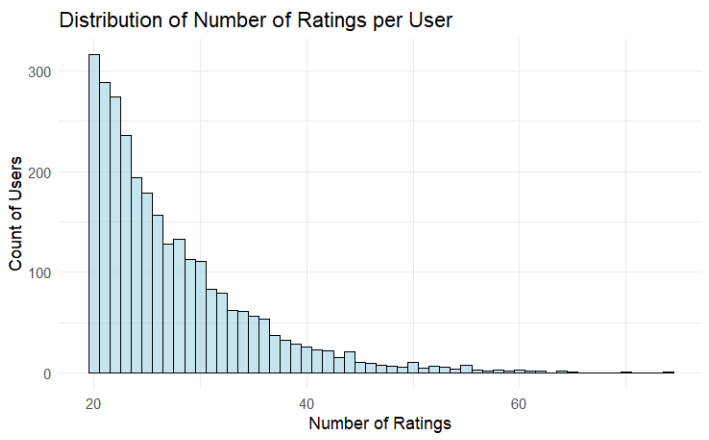


Figure 13: Distribution of Number of Ratings per User

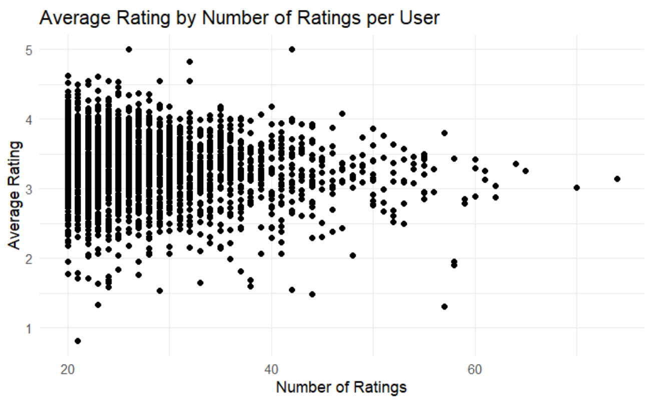


Figure 14: Average Rating by Number of Ratings per User

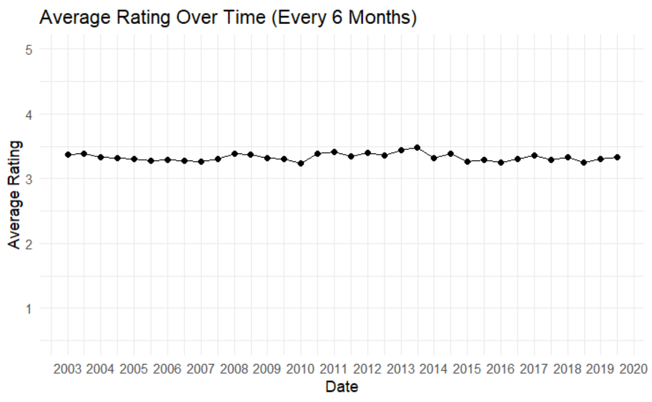


Figure 15: Average Movie Rating Over Time

C Official Statement of Originality

By signing this statement, I hereby acknowledge the submitted thesis, entitled:

Predicting MovieLens preferences using softImpute-ALS

to be produced independently by me, without external help. Wherever I paraphrase or cite literally, a reference to the original source (journal, book, report, internet, etc.) is given. By signing this statement, I explicitly declare that I am aware of the fraud sanctions as stated in the Education and Examination Regulations (EERs) of the SBE.

Place: Maastricht, the Netherlands
Date: 28/08/2024
Signature:

[illegible]