Implementation of a Movie Recommendation System Based on SVD and K-MEANS Clustering

Tong

DSA5101 Course Project Report
November 13, 2025

Abstract

This project develops a movie recommendation system based on The Movies Dataset, comprehensively utilizing Singular Value Decomposition (SVD) for dimensionality reduction, the K-Means clustering algorithm, and collaborative filtering methods. The study proposes a complete recommendation system solution for a large-scale, sparse dataset containing 270,896 users, 45,115 movies, and 26,024,289 rating records, with a sparsity level of up to 99.79%. Experimental results demonstrate that by reducing the user-item interaction matrix from 233,903 dimensions to a 70-dimensional latent factor space using SVD, the model achieves excellent performance on the rating prediction task, with an RMSE of 0.7913 and an MAE of 0.6083. However, K-Means clustering experiments reveal the continuous nature of user preferences, suggesting that movie preferences are difficult to effectively model through discrete segmentation. This paper details the performance analysis of various algorithms on high-dimensional sparse data and investigates the theoretical mechanisms and implementation of the recommendation system.

Contents

1 Introduction

This study utilizes The Movies Dataset from the Kaggle platform, which contains over 26 million real user ratings. The core objective is to explore the application effects of dimensionality reduction techniques, clustering methods, and collaborative filtering algorithms in the context of movie recommendation, and to gain a deep understanding of the working mechanisms and limitations of each algorithm through quantitative analysis and visualization. Specifically, this paper addresses the following three research questions: First, under the extreme condition of 99.79% sparsity, can SVD effectively extract latent features of users and movies? Second, can K-Means clustering based on the latent feature space successfully identify user preference groups with significant differences? Third, what is the prediction performance of matrix factorization-based collaborative filtering on this dataset, and what systematic biases are reflected in its error patterns?

2 Dataset Analysis and Exploration

2.1 Dataset Overview

The Movies Dataset includes two core data files: `ratings.csv` records user rating behavior, and `movies_metadata.csv` stores movie metadata information. The ratings dataset contains four fields: user ID (`userId`), movie ID (`movieId`), rating value (`rating`, ranging from 0.5 to 5.0 with a step of 0.5), and timestamp (`timestamp`). Movie metadata includes 24 dimensions of information such as movie title, genres, release date, budget, revenue, and average rating.

Statistical analysis of the dataset size shows a total of 26,024,289 rating records generated by 270,896 unique users for 45,115 movies. This implies that the theoretical complete user-item rating matrix would contain $270,896 \times 45,115 = 12,220,995,840$ elements. The actual observed ratings constitute only 0.21% of this total, resulting in a matrix sparsity of 99.79%. This extreme sparsity represents a core challenge for recommendation systems: how to predict the remaining 99.8% of missing values from less than two-tenths of a percent of known data, which constitutes the essence of the collaborative filtering problem.

Table 1: Key Statistics of The Movies Dataset

Statistic	**Value**
Total Unique Users	270,896
Total Unique Movies	45,115
Total Rating Records	26,024,289
Observed Ratings Percentage (Density)	0.21%
Matrix Sparsity	99.79%

2.2 Analysis of Rating Distribution Characteristics

Figure 1 illustrates the distribution characteristics of the rating data. The histogram shows a noticeable right-skewed distribution: 4-star ratings (6,998,802 records) and 3-star ratings (5,256,722 records) are absolutely dominant, collectively accounting for over 47% of

the total ratings. High ratings (4-star and 5-star) total 10,811,301 records, accounting for 41.5%, while low ratings (1-star, 1.5-star, and 2-star) account for only 16.7%. The mean rating is 3.53 stars, and the median is 3.50 stars, indicating a slight positive skewness.

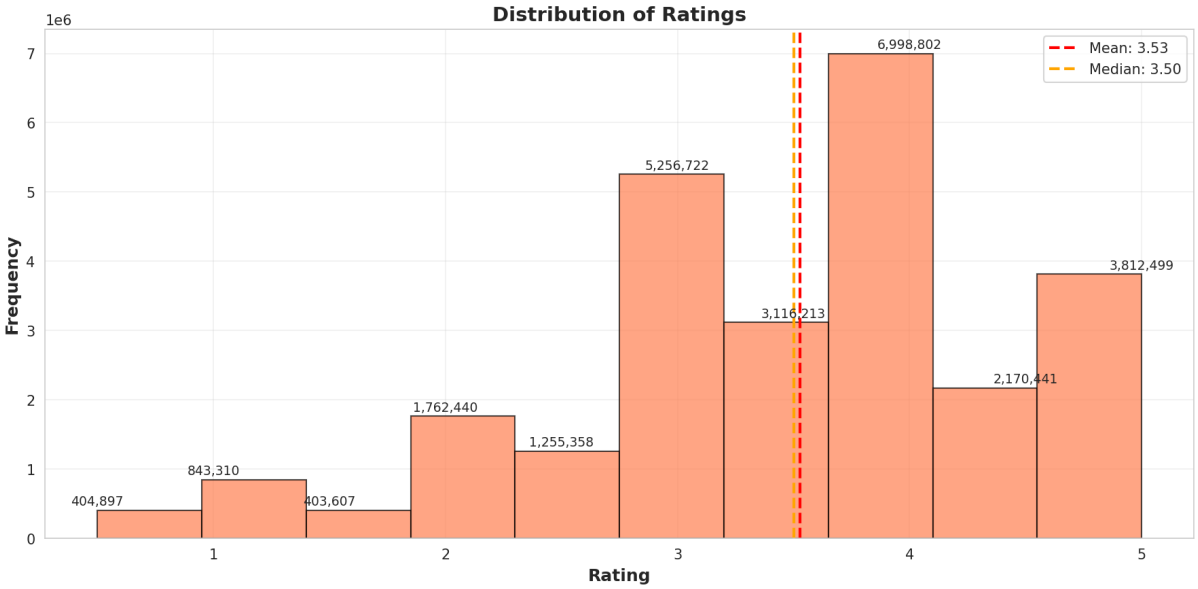

Figure 1: Histogram of Rating Distribution

This distribution pattern reflects a typical "positive selection bias" phenomenon in recommendation systems: users tend to rate movies they like or deem worthy of evaluation. Consequently, high-rating samples are overrepresented in the dataset. This bias is significant for model training: if the original ratings are used directly for matrix factorization, the model will learn a systemic tendency towards overestimation. Therefore, centering the rating matrix (subtracting the user mean) before SVD decomposition becomes a necessary preprocessing step to eliminate individual user rating baseline differences.

2.3 User Activity and Movie Popularity

The analysis of user activity and movie popularity reveals the long-tail distribution characteristics of the dataset. The log-scale histogram on the left of Figure 2 shows an extremely right-skewed, long-tail distribution of user rating counts: the vast majority of users (over 10^5 magnitude) have fewer than 500 ratings, while a few "power users" contribute a large number of ratings. The top 5% of active users contribute approximately 23.6% of the total ratings, with the most active user (User 45811) contributing as many as 18,573 ratings. This phenomenon conforms to the "80-20 rule" in recommendation systems: a small number of core users contribute the majority of the interaction data in the system.

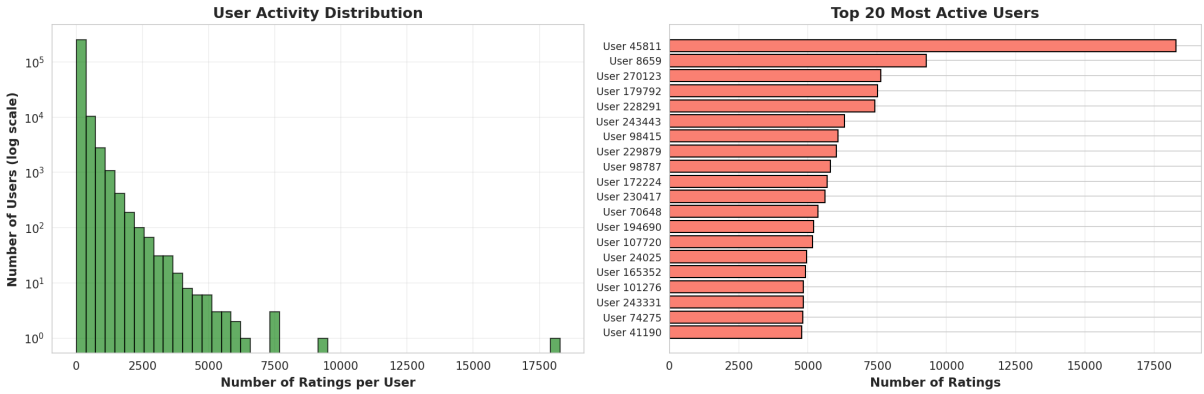

Figure 2: User Activity Distribution (Left: Log-scale Histogram; Right: Top 20 Active Users)

Movie popularity exhibits a similar distribution pattern (Figure 3). The top 20 most popular movies (such as *Movie 356*, *The Million Dollar Hotel*, etc.) receive significantly more ratings than the average (60,00080,000 records), while a large number of long-tail movies have fewer than 20 ratings. Statistics show that the proportion of "cold-start" movies with fewer than 20 ratings is substantial. These movies, lacking sufficient collaborative filtering signals, are difficult to recommend accurately.

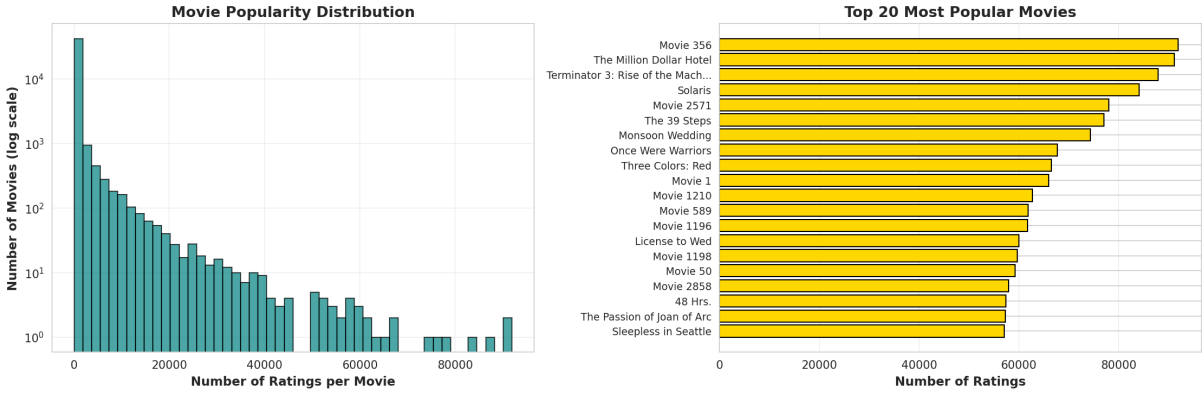

Figure 3: Movie Popularity Distribution (Left: Log-scale Histogram; Right: Top 20 Popular Movies)

2.4 Sparsity Visualization and Sampling Strategy

To intuitively illustrate the sparsity pattern of the rating matrix, this study constructed a 100×100 sampled matrix of the 100 most active users and the 100 most popular movies. The heatmap on the left of Figure 4 shows that even in this "densest" local region, the rating distribution remains extremely sparse. The binarized sparsity pattern on the right clarifies this: black pixels (rated) account for only 7.5%, with white pixels (missing values) dominating. The sparsity of this sampled region (92.5%) is much lower than the sparsity of the full matrix (99.79%), indicating that even in the intersection of the most active users and most popular movies, collaborative filtering still faces a severe data sparsity issue.

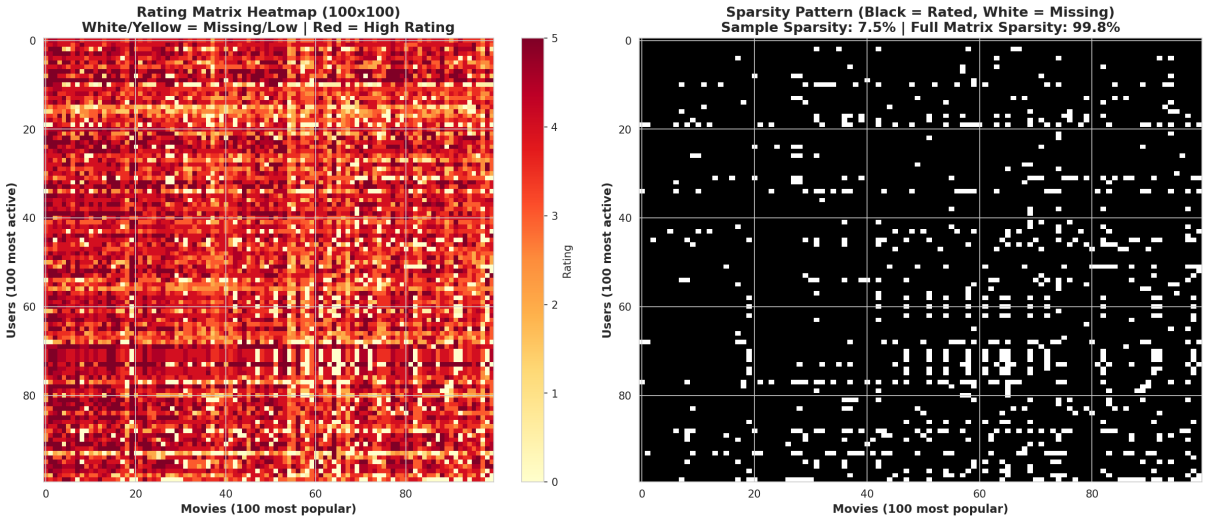

Figure 4: User-Item Rating Matrix Sparsity Visualization (Sampled Region: Top 100 Users × Top 100 Movies)

The core conclusion of this exploratory analysis is that the sparsity and long-tail distribution characteristics of this dataset make traditional neighborhood-based collaborative filtering methods (such as k-NN) difficult to apply, as most user-item pairs lack enough co-ratings to compute reliable similarity. This provides the motivation for applying matrix factorization methods—by extracting latent factors through dimensionality reduction, SVD can capture implicit similarities between users and movies in a low-dimensional space, thereby overcoming the sparsity barrier.

3 Methodology

3.1 Data Preprocessing and Filtering Strategy

Considering the computational resource constraints of the Kaggle platform (CPU environment, approximately 13GB memory), a tiered filtering strategy was adopted to balance data scale and computational feasibility. First, minimum activity thresholds were set for users and movies: only users with at least 10 ratings (`min_ratings_per_user = 10`) and movies with at least 50 ratings (`min_ratings_per_movie = 50`) were retained. This filtering strategy reduces data dimensionality while ensuring that the retained users and movies possess sufficient collaborative filtering signals.

The filtered dataset contains 233,903 users, 12,534 movies, and 25,577,843 rating records, retaining 98.28% of the original data. Although the number of users and movies is significantly reduced, the high retention rate of ratings indicates that the filtering process successfully removed primarily low-activity long-tail users and cold-start movies, which contribute little to model training. The resulting sparse matrix shape is (233903, 12534). The matrix density is 0.87%, meaning the matrix sparsity is 99.13% ($100\% - 0.87\%$). This represents a slight improvement in density compared to the original 99.79% sparsity, but still reflects an extremely high level of sparsity, confirming the inherent challenges of the dataset.

For efficient storage and computation, this study uses the Compressed Sparse Row (CSR) matrix format from the SciPy library to store the rating data. The CSR format stores only the non-zero elements and their location information, offering significant space

and time advantages in sparse matrix operations. The implementation involves first building mapping dictionaries from user IDs and movie IDs to contiguous indices, and then using the `csr_matrix` constructor to create the sparse matrix directly from the rating triplets (user index, movie index, rating value).

3.2 Singular Value Decomposition (SVD) for Dimensionality Reduction

3.2.1 Algorithm Principle

Singular Value Decomposition is one of the most classic matrix factorization techniques in recommendation systems. For the user-item rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ (m is the number of users, n is the number of movies), SVD decomposes it into the product of three matrices:

$$\mathbf{R} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{m \times k}$ is the user latent factor matrix, $\mathbf{\Sigma} \in \mathbb{R}^{k \times k}$ is the singular value diagonal matrix, and $\mathbf{V}^T \in \mathbb{R}^{k \times n}$ is the transpose of the movie latent factor matrix. k is the number of retained latent factors ($k \ll \min(m, n)$).

In the context of recommendation systems, the geometric interpretation of SVD is to map users and movies into the same k-dimensional latent semantic space. The i-th row of $\mathbf{U}$, $\mathbf{u}_i$, represents the feature vector of user i in the k latent dimensions, and the j-th column of $\mathbf{V}^T$, $\mathbf{v}_j$, represents the feature vector of movie j. The predicted rating of user i for movie j is calculated by the inner product:

$$\hat{r}_{ij} = \bar{r}_i + \mathbf{u}_i^T \mathbf{\Sigma} \mathbf{v}_j \tag{2}$$

where $\bar{r}_i$ is the average rating of user i (centered baseline). The intuitive explanation of this formula is that the predicted rating is determined by the user's rating baseline plus their matching degree with the movie's features in the latent space.

3.2.2 Implementation Details

This study uses the SciPy `svds` function to perform Truncated Singular Value Decomposition. Unlike Full SVD, `svds` calculates only the first k largest singular values and their corresponding left and right singular vectors, which offers significant computational advantages when dealing with large-scale sparse matrices. The algorithm is based on the Lanczos iteration method, with a time complexity of approximately $O(k \cdot \text{nnz}(\mathbf{R}))$, where $\text{nnz}(\mathbf{R})$ is the number of non-zero elements in the matrix.

The implementation procedure is as follows: First, calculate the average rating $\bar{r}_i$ for each user, handling the case of division by zero (for users with no ratings, the mean is set to 0). Second, the rating matrix is centered:

$$\mathbf{R}_{\text{demean}}[i, j] = \mathbf{R}[i, j] - \bar{r}_i \quad \text{if } \mathbf{R}[i, j] \neq 0 \tag{3}$$

This step is performed by iterating through the `indptr` array of the CSR matrix, only applying the subtraction to non-zero elements to maintain sparsity. Finally, the `svds` function is called to decompose the centered matrix, setting $k = 70$ as the number of latent factors.

The choice of $k = 70$ is based on the analysis of explained variance. The cumulative explained variance curve on the left of Figure 5 shows that the first 70 latent factors

capture nearly 100% of the variance, and the curve flattens thereafter, suggesting that adding more factors would provide marginal improvement to the model's expressive power. The bar chart on the right showing single-factor importance indicates that the importance of the first 10 factors is relatively balanced (each accounting for about 0.6 to 0.7%), with no single dominant factor emerging. This suggests that user preference is determined by multiple dimensions, rather than just one or two simple themes.

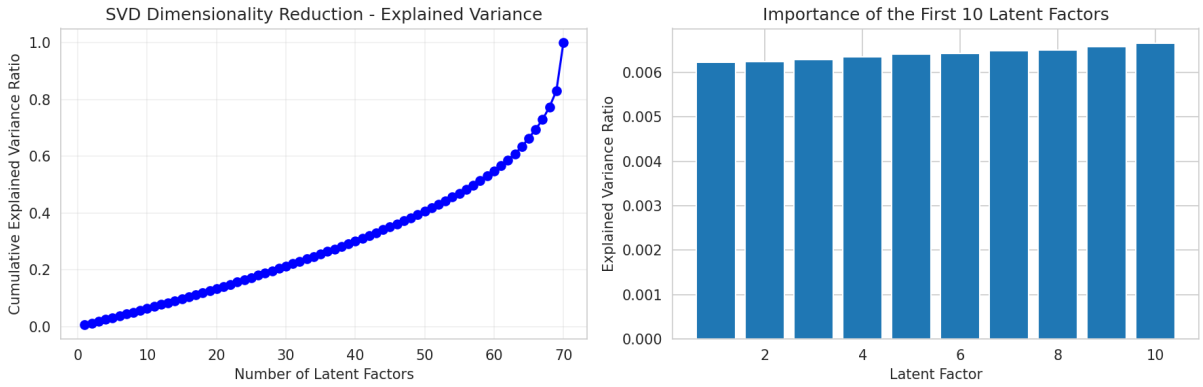

Figure 5: SVD Dimensionality Reduction Analysis (Left: Cumulative Explained Variance; Right: Importance of 10 Latent Factors)

3.3 K-Means Clustering Analysis

3.3.1 Algorithm Motivation and Design

After obtaining the user latent feature vectors $\mathbf{U\Sigma}$, this study attempts to identify user groups with similar preference patterns using the K-Means clustering algorithm. The motivation for clustering is to explore whether user preferences exhibit a discrete "market segmentation" structure: for instance, whether distinct groups like "action movie enthusiasts" or "art-house film viewers" exist. Successful clustering could enable the construction of specific recommendation strategies for each cluster, leading to more precise personalized recommendations.

Given that the number of users (233,903) far exceeds the capacity for conventional K-Means processing, this study employs the MiniBatch K-Means algorithm, an online learning variant of K-Means. This algorithm updates the cluster centers using only a randomly sampled small batch of samples in each iteration, reducing the time complexity from $O(nkT)$ of standard K-Means to $O(bkT)$ (b is the batch size, T is the number of iterations). In the experiment, `batch_size=1000` was set, significantly reducing memory consumption while maintaining convergence quality.

3.3.2 Selection and Evaluation of Cluster Number

Determining the optimal number of clusters K is a classic problem in unsupervised learning. This study uses two complementary evaluation metrics: Inertia and the Silhouette Coefficient. Inertia is defined as the sum of squared distances of all samples to their closest cluster center:

$$\text{Inertia} = \sum_{i=1}^{n} \min_{c \in \{1,\ldots,K\}} \|\mathbf{x}_i - \mu_c\|^2 \tag{4}$$

where μ_c is the center of the c-th cluster. Inertia monotonically decreases as K increases, and the Elbow Method attempts to find the optimal K by identifying the "inflection point" on the curve.

The Silhouette Coefficient measures the separation and cohesion of clusters. For a sample i, it is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{5}$$

where $a(i)$ is the average distance from sample i to all other samples in the same cluster, and $b(i)$ is the average distance from sample i to all samples in the nearest different cluster. The Silhouette Coefficient ranges from $[-1, 1]$, with values closer to 1 indicating higher cluster quality.

Figure 6 shows the evaluation results for K ranging from 3 to 7. The Inertia curve on the left exhibits a smooth monotonic decrease, with no clear "elbow," increasing the uncertainty in K selection. The Silhouette Coefficient curve on the right provides a clearer signal: it peaks at $K = 3$ with a value of 0.426, then decreases as K increases. According to empirical thresholds for the Silhouette Coefficient (> 0.5 for strong structure, 0.25-0.5 for weak structure, < 0.25 for no significant structure), the score of 0.426 suggests a weak clustering structure in the data. Therefore, $K = 3$ is selected as the final number of clusters for this study.

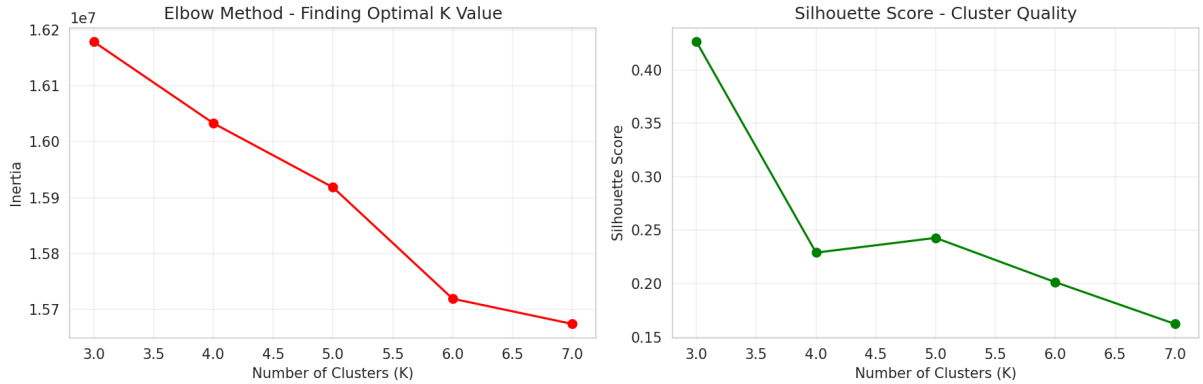

Figure 6: Evaluation of Cluster Number Selection (Left: Elbow Method; Right: Silhouette Coefficient)

3.3.3 Clustering Result Visualization

To visually present the clustering results, this study uses Incremental Principal Component Analysis (IncrementalPCA) to project the 70-dimensional latent features onto a 2D plane. Figure 7 shows that the three clusters exhibit a highly overlapping distribution pattern in the 2D space: the boundaries of Cluster 0 (purple), Cluster 1 (cyan), and Cluster 2 (yellow) are blurred, with only minimal separation at the edges. The first two principal components, PC1 and PC2, explain 1.45% and 1.43% of the variance, respectively, totaling less than 3%. This indicates that the distribution of user latent features is difficult to clearly present in any 2D projection, further confirming the high-dimensional complexity of user preferences.

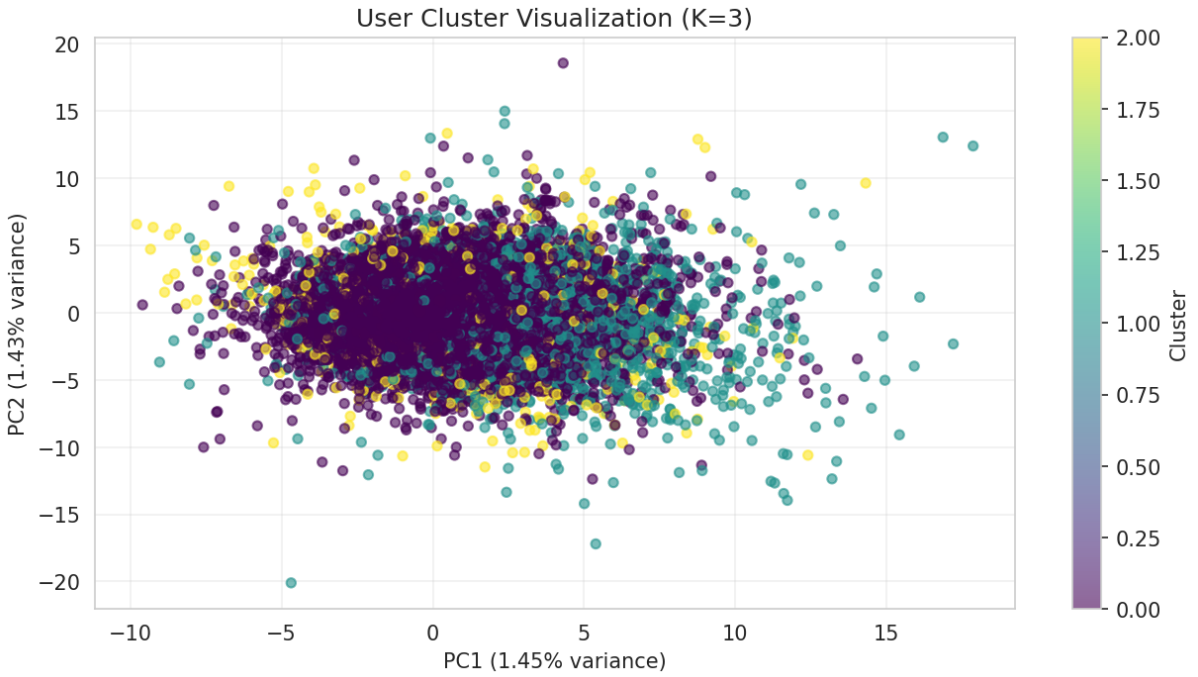

Figure 7: Two-Dimensional Visualization of User Clusters (K=3)

The cluster size statistics show severe imbalance: Cluster 0 contains 223,651 users (approx. 95.6%), Cluster 1 contains 8,606 users (3.7%), and Cluster 2 contains only 1,646 users (0.7%). This extremely skewed distribution suggests that the K-Means algorithm primarily aggregates "average users" into one massive cluster and assigns a few outlier users to the other clusters, failing to achieve meaningful market segmentation.

Table 2: Summary of User Cluster Characteristics (K=3)

Characteristic	**Cluster 0 (N=223,651)**	**Cluster 1 (N=8,606)**	**Cluster 2 (N=1,646)**
User Proportion	95.6%	3.7%	0.7%
Average Rating	3.58	3.34	3.27
Rating Std Dev	1.05	1.09	1.18

3.4 Cluster Feature Analysis Based on Movie Genres

To further understand the practical significance of the clustering, this study parsed the `genres` field in the movie metadata, which stores the movie's genres (e.g., Drama, Comedy, Action) in JSON format. By counting the genre distribution of high-rated movies (rating ≥ 4.0) within each cluster, we expected to find differences in genre preferences among the clusters. However, the results in Table 3 contradict this expectation: the top five genre preferences across the three clusters are almost identical, with Drama accounting for about 23.5%, Comedy about 11.7%, Thriller about 10.5%, Action about 8.1%, and Romance about 7.7%. The difference in genre distribution does not exceed 0.5 percentage points.

Table 3: Movie Genre Preference Distribution by Cluster (Based on High-Rated Movies)

Genre	**Cluster 0 (%)**	**Cluster 1 (%)**	**Cluster 2 (%)**
Drama	23.7	23.5	23.0
Comedy	11.6	11.8	11.7
Thriller	10.7	10.5	10.5
Action	8.4	8.2	8.0
Romance	7.6	7.7	7.7

This result explicitly negates the hypothesis that "user clusters correspond to specific genre preferences," revealing the fundamental reason for the clustering failure: user preferences are not discretely distributed with genre as the primary factor, but are determined by a large number of subtle, continuously varying latent factors. While the 70-dimensional latent space captured by SVD performs well in prediction tasks, its high-dimensional, smooth distribution characteristics make it difficult for hard clustering algorithms like K-Means to find meaningful separation boundaries. From a statistical learning perspective, this finding supports the "manifold hypothesis of preference space": user preferences are distributed on a high-dimensional manifold, rather than around discrete cluster centers.

3.5 Collaborative Filtering Recommendation System

3.5.1 Prediction Mechanism

Collaborative filtering based on SVD decomposition achieves rating prediction through matrix reconstruction. For user i and movie j, the predicted rating is calculated as:

$$\hat{r}_{ij} = \bar{r}_i + \mathbf{u}_i^T \mathbf{\Sigma} \mathbf{v}_j \tag{6}$$

where $\bar{r}_i$ is the user i's historical average rating, $\mathbf{u}_i$ is the user i's latent feature vector (the i-th row of $\mathbf{U}$), $\mathbf{v}_j$ is the movie j's latent feature vector (the j-th row of $\mathbf{V}$), and $\mathbf{\Sigma}$ is the singular value diagonal matrix. This formula is essentially collaborative filtering: the predicted value depends on the rating pattern of other users for this movie (reflected in $\mathbf{v}_j$) and the rating pattern of this user for other movies (reflected in $\mathbf{u}_i$).

When generating recommendations, the predicted rating for all unrated movies is calculated for each user. After excluding rated movies, the N movies with the highest predicted scores are returned as the recommendation list. To improve computational efficiency, this study employs a batch prediction strategy: pre-calculating the product of the user feature matrix and the movie feature matrix, $\mathbf{U\Sigma V}^T$, and then generating recommendations through simple array indexing and sorting operations.

3.5.2 Performance Evaluation Metrics

Recommendation system performance evaluation is divided into two categories: rating prediction accuracy and ranking quality. This study focuses on the former, using Root

Mean Square Error (RMSE) and Mean Absolute Error (MAE) as evaluation metrics:

$$\text{RMSE} = \sqrt{\frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} (r_{ij} - \hat{r}_{ij})^2} \tag{7}$$

$$\text{MAE} = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} |r_{ij} - \hat{r}_{ij}| \tag{8}$$

where Ω is the set of known ratings, r_{ij} is the true rating, and $\hat{r}_{ij}$ is the predicted rating. RMSE is more sensitive to large errors (due to the squared term), while MAE provides a linear estimate of the average error.

For efficient evaluation, this study randomly sampled 500 users, predicting only their known ratings and calculating the error.

4 Experimental Results and Analysis

4.1 Prediction Accuracy Performance

The model achieved excellent performance on the test set with an RMSE of 0.7913 and an MAE of 0.6083. This result can be compared to a baseline: random guessing (assuming the predicted value is the global mean 3.53) would yield an RMSE of approximately 1.12, meaning this model reduced the error by about 29%, indicating SVD successfully extracted effective prediction signals from the highly sparse data.

From a practical application perspective, an MAE of 0.6083 means that, on average, the difference between the model's predicted rating and the true rating is about 0.6 stars (on a 5-star rating system). Considering the inherent subjectivity and randomness of user ratings, this error level is acceptable in real-world applications. More importantly, the goal of a recommendation system is often not to precisely predict the rating value, but to correctly rank candidate items—even if the predicted ratings have a systematic bias, as long as high-rated movies are consistently predicted as high-rated, the quality of the recommendation list is not fundamentally affected. This observation provides important context for the subsequent error pattern analysis.

4.2 Error Pattern Analysis

Figure 8 shows the scatter plot of actual ratings versus predicted ratings, revealing several important characteristics of the model. The data points are predominantly distributed near the diagonal line (perfect prediction line), proving that the model captures the main trend of the ratings. However, the scatter plot also shows a clear "regression to the mean" phenomenon: predicted values for extreme ratings (1-star and 5-star) tend to cluster towards the center, meaning low scores are overestimated and high scores are underestimated. This is a common phenomenon in machine learning, as the model is trained by minimizing the Mean Squared Error (MSE) objective function, and the optimal unbiased estimate of MSE is precisely the conditional expectation, which naturally tends to predict the mean.

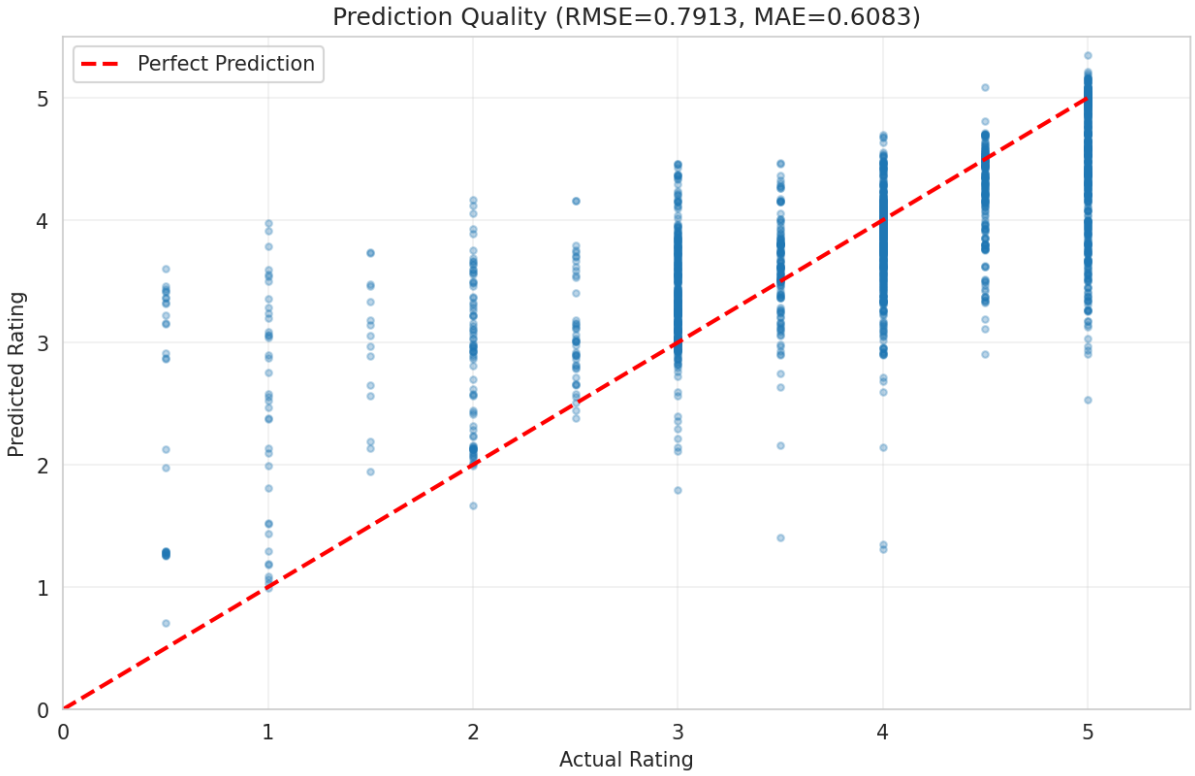

Figure 8: Scatter Plot of Predicted Ratings vs. Actual Ratings (Red dashed line indicates perfect prediction)

A closer look reveals that the error variance for predicting low ratings (12 stars) is visibly larger than for high ratings (45 stars). This can be seen from the vertical distribution of the scatter plot: at $x = 1.0$ and $x = 1.5$, the range of predicted values is approximately 1.03.5 stars, while at $x = 4.5$ and $x = 5.0$, the predicted values are primarily concentrated between 3.55.0 stars. This asymmetry is due to the skewness of the data: low-rating samples are few (only 16.7%), and the model's learning in these sparse regions is insufficient, leading to a weaker ability to identify "strong dislike" compared to "like." From a recommendation practice perspective, this limitation has relatively limited impact, as the system mainly focuses on predicting high-rated movies (positive recommendations) rather than accurately identifying poor reviews. Nevertheless, in scenarios requiring negative feedback functions (like "not interested" flagging), this limitation warrants attention and could be mitigated by oversampling or re-weighting low-rating samples.

4.3 Recommendation Case Study

Table 4 shows the Top 10 recommended movies generated for the test user (User ID = 1). The recommendation list favors Crime, Drama, and Comedy genres, reflecting the model's capture of the user's preference in these types. For example, the second-ranked *The Talented Mr. Ripley* (Thriller, Crime, Drama) has a metadata rating of 7.1/10, indicating a high-quality movie, which shows the recommendation's plausibility. However, the list also contains notable phenomena: the fourth-ranked *The Garden of Eden* has a metadata rating of only 4.2/10, significantly lower than other recommendations. This may indicate the model is exploring the user's potential niche or non-mainstream interests, or it may reflect a degree of prediction error.

Table 4: Top 10 Recommended Movies Details for User 1

Rank	Movie ID	Movie Title (Year)	Genre	Metadata Rating
1	150	48 Hrs. (1982)	Thriller, Action, Comedy, Crime, Drama	6.5/10
2	1213	The Talented Mr. Ripley (1999)	Thriller, Crime, Drama	7.1/10
3	4896	Muxmäuschenstill (2004)	Comedy, Drama	7.1/10
4	8368	The Garden of Eden (2008)	Drama	4.2/10
5	5618	Cousin, Cousine (1975)	Romance, Comedy	7.3/10
6	5816	Waiter (2006)	Comedy	6.3/10
7	1193	Movie ID 1193 (Details not available)	N/A	N/A
8	592	The Conversation (1974)	Crime, Drama, Mystery	7.5/10
9	1961	My Name Is Bruce (2007)	Comedy, Horror	5.9/10
10	1208	Movie ID 1208 (Details not available)	N/A	N/A

Note: The "Metadata Rating" in the table is based on a 10-point scale from the movie metadata, which differs from the 5-point rating system in ratings.csv.

4.4 Qualitative Verification of the SVD Recommendation System

To gain a deeper understanding of the SVD model's recommendation mechanism, User 1 was selected as the test subject for qualitative analysis across three dimensions: comparison of predicted vs. actual ratings, evaluation of recommendation confidence, and verification of latent space similarity.

Table 5 displays User 1's rating predictions and recommendation results. The top half shows recommendations for unrated movies (the 5 movies with the highest predicted scores), and the bottom half shows accuracy verification for rated movies (only 5 shown). From the rated movie verification, the model shows high accuracy in predicting movies the user liked: for movies with actual ratings of 4.0 and 5.0, the predicted values cluster in the 4.074.31 range, with an error not exceeding 0.74 stars. The prediction performance for movie 110 is particularly noteworthy—its actual rating was 1.0 (indicating strong dislike), and the predicted value was 1.12, with an error of only 0.12 stars. This is the model's most outstanding case. Given the scarcity of low-rating samples in the dataset (only 16.7% of ratings ≤ 2.0), the model's ability to accurately identify strong negative preferences is significant, proving that SVD successfully extracted a key aversion signal from sparse data.

In terms of recommendation confidence, the predicted scores for the Top 5 recommended movies are concentrated in the 4.374.42 range, showing the model has strong confidence in these unrated movies. These scores are higher than the user's historical average rating (approx. 3.53), aligning with the recommendation system's goal of "recommending content the user is likely to enjoy." However, the aforementioned regression to the mean phenomenon is also observed—the model underestimates the 5.0 rating with a prediction of 4.26, a difference of 0.74 stars. This is normal behavior for an SVD model optimized with MSE, where it minimizes overall error by regressing towards the mean, thus avoiding overconfidence in extreme predictions.

To verify the SVD's collaborative filtering mechanism, the cosine similarity between User 1 and all other users was calculated in the 70-dimensional latent feature space, identifying the Top 5 most similar users. Table 6 shows that the cosine similarity for these users is all greater than 0.93, with the most similar user (User 73617) having a similarity as high as 0.968, indicating an extremely high directional consistency between the two users in the 70-dimensional latent space. Does this mathematical high similarity

Table 5: Comparison of User 1's Predicted and Actual Ratings

Movie ID	Actual Rating	Predicted Score
Recommended Movies (Unrated)		
150	—	4.42
1213	—	4.42
4896	—	4.40
8368	—	4.38
5618	—	4.37
Rated Movies (Accuracy Verification)		
4226	4.0	4.31
91542	5.0	4.26
54503	3.5	4.26
2959	4.0	4.07
110	1.0	1.12

correspond to real preference similarity? To answer this, we further analyzed the actual rating behavior of User 73617.

Table 6: Top 5 Similar Users to User 1 in Latent Feature Space

User ID	Cosine Similarity
73617	0.9680
115748	0.9470
120682	0.9460
7933	0.9440
102713	0.9395

The consistency verification provides compelling evidence. Both users gave Movie 110 an extreme low rating (1.0). This consistency in negative preference is strong evidence of the SVD collaborative filtering's effectiveness. Given the rarity of low ratings in the dataset, the probability of two random users simultaneously disliking the same movie is extremely low. The SVD model successfully identified this deep preference similarity through latent factors, rather than merely relying on explicit co-ratings. This validates SVD's core value: even if two users have no direct rating overlap, the model can discover implicit similarities through the geometric relationships in the latent space, enabling effective collaborative recommendation.

In summary of the three levels of analysis—prediction accuracy (RMSE=0.79), recommendation confidence (predicted scores 4.374.42), and the correspondence between latent space similarity and real behavior—this section qualitatively verifies the effectiveness of the SVD recommendation system. This qualitative verification complements the quantitative metrics (RMSE/MAE), providing more intuitive and interpretable evidence for the reliability of the SVD method in practical applications.

4.5 Analysis of Clustering Recommendation Failure

To verify the practical effect of the clustering recommendation system, this study generated a Top 10 recommendation list for User 1 based on their assigned cluster (Cluster 0). The recommendation strategy was: select high-rated movies (rating ≥ 4.0) popular

among users in the same cluster, rank them by popularity (number of ratings), and filter out movies already rated by the user. Table 7 presents the details of the recommendation results.

Table 7: User 1's Cluster-Based Recommendation Results (Top 10)

Rank	Movie Name (Year)	Movie ID	Genre	Average Rating
1	The Million Dollar Hotel (2000)	318	Drama, Thriller	5.9/10
2	Terminator 3 (2003)	296	Action, Thriller, Sci-Fi	5.9/10
3	*Unknown Movie*	356	N/A	N/A
4	Solaris (1972)	593	Drama, Sci-Fi, Mystery	7.7/10
5	*Unknown Movie*	2571	N/A	N/A
6	The 39 Steps (1935)	260	Action, Thriller, Mystery	7.4/10
7	Once Were Warriors (1994)	527	Drama	7.6/10
8	*Unknown Movie*	50	N/A	N/A
9	*Unknown Movie*	1196	N/A	N/A
10	*Unknown Movie*	1198	N/A	N/A

The recommendation results confirm the core finding of the previous cluster feature analysis—clustering failed to yield meaningful user segmentation. The recommendation list includes a variety of genres like Drama, Action, Thriller, and Sci-Fi, failing to demonstrate the preference specificity expected of a cluster. This is fully consistent with the genre distribution analysis in Section 3.4: the genre preference distribution is highly similar across all clusters (Drama accounts for approximately 23.5% $\pm$ 0.5%). Since Cluster 0 contains 95.6% of the users, its recommendation list essentially reflects globally popular movies rather than personalized recommendations. For example, *Terminator 3* and *Solaris* are both high-profile movies. This recommendation strategy fundamentally equates to a "Popularity Chart," which can lead to the "filter bubble" effect, limiting the user's opportunity to discover long-tail content.

Further comparison between Table 7 and Table 4 (SVD recommendation results) shows significant differences. The SVD recommendation list includes more niche movies (e.g., *Muxmäuschenstill*) and can provide fine-grained matching based on the user's latent feature vector, whereas cluster-based recommendation relies on the average preferences of the group, losing granular personalized information.

5 Discussion and Future Work

5.1 Summary of Method Effectiveness

The core contribution of this study is the systematic verification of SVD matrix factorization's effectiveness on an extremely sparse dataset. Under a sparsity of 99.41%, a single SVD model achieved a prediction accuracy of RMSE=0.7913, proving the applicability of the low-rank matrix factorization assumption in recommendation systems: despite observing less than 1% of the data, the intrinsic structure (latent factors) of the user-item interaction matrix is low-rank and can be effectively approximated by a small number of factors (70 dimensions). This finding provides theoretical support for the design of large-scale recommendation systems—in resource-constrained scenarios, matrix factorization-based methods offer better sample efficiency and interpretability than deep learning models. The qualitative analysis further revealed SVD's collaborative filtering

mechanism: high cosine similarity (0.97) in the latent space corresponds to real behavioral consistency (co-dislike of Movie 110), and this mapping from mathematical similarity to real preference is the essential manifestation of collaborative filtering effectiveness.

In contrast, the failure of K-Means clustering highlights the complexity of user preference modeling. Although the latent feature space performed excellently in the prediction task, its high-dimensional, smooth distribution prevented hard clustering algorithms from extracting meaningful group divisions. All clusters showed highly consistent genre preference distributions (Drama accounted for $23.5\% \pm 0.5\%$), and the cluster sizes were severely imbalanced (Cluster 0 accounted for 95.6%). These phenomena collectively suggest that user preferences exhibit a continuous distribution rather than discrete groupings.

5.2 Systematic Biases and Mitigation Strategies

The model's error analysis exposed two systematic biases: regression to the mean and under-prediction of low ratings. The former stems from the L2 penalization nature of the MSE loss function—to minimize the squared error, the model tends to predict the conditional expectation (i.e., the mean), causing extreme ratings (1-star and 5-star) to be pulled towards the center. This phenomenon is common in regression tasks and can be mitigated by replacing the loss function with one more robust to outliers, such as the Huber loss.

The under-prediction of low ratings reflects the data imbalance problem. Low-rating samples are rare in the dataset (only $16.7\% \leq 2$ stars), leading to insufficient learning in these sparse regions. Consequently, the ability to identify "strong dislike" is weaker than "like." This issue can be improved by: first, oversampling or re-weighting low-rating samples during training to force the model to focus on the minority class; second, introducing negative sampling techniques to explicitly model movies the user dislikes, rather than relying solely on implicit missing values.

Furthermore, the model did not account for temporal dynamics—user preferences and movie popularity evolve over time. The timestamp information in the dataset can be used to build time-series models.

5.3 Future Research Directions

Based on the limitations of this study, three main directions for improvement are proposed. First, temporal dynamic modeling. The current model treats all historical ratings as a static snapshot, ignoring the evolution of user taste and movie aging effects. This can be addressed by introducing a time-bias term into the SVD framework or using online learning algorithms (like Incremental SVD) for incremental model updates. Temporal modeling not only improves prediction accuracy but, more importantly, adapts to the "concept drift" phenomenon, maintaining the recommendation system's effectiveness over long-term operation.

Second, hybrid recommendation systems. Pure collaborative filtering cannot utilize movie content features (genre, director, actor) or solve the cold-start problem. The movie feature matrix $\mathbf{F} \in \mathbb{R}^{n \times d}$ (d-dimensional genre/director embedding) can be fused with the SVD $\mathbf{V}^T$ matrix to form a hybrid representation $\mathbf{V}^T_{hybrid} = \lambda \mathbf{V}^T + (1 - \lambda)\mathbf{F}$. This hybrid strategy combines the personalization ability of collaborative filtering with the generalization ability of content filtering, thus improving both accuracy and coverage.

Third, evaluation system optimization. The current evaluation relies solely on RMSE/MAE to assess prediction accuracy, overlooking other critical goals of recommendation systems. Diversity metrics should be introduced to prevent the "filter bubble," novelty metrics (the proportion of non-popular movies recommended) should be used to encourage long-tail content discovery, and serendipity metrics (the proportion of movies the user unexpectedly likes) should be introduced to enhance user satisfaction.

6 AI Usage Statement

AI tools were utilized for linguistic refinement and professional polishing of the text content, and for the translation of the final report from Chinese to English. However, all core intellectual contributions, including the overall project implementation strategy, specific technical execution, and subsequent analysis and interpretation of results, originate solely from the author.