PERSONALIZED MOVIE RECOMMENDATION USING SVD AND LOW-RANK MATRIX APPROXIMATION

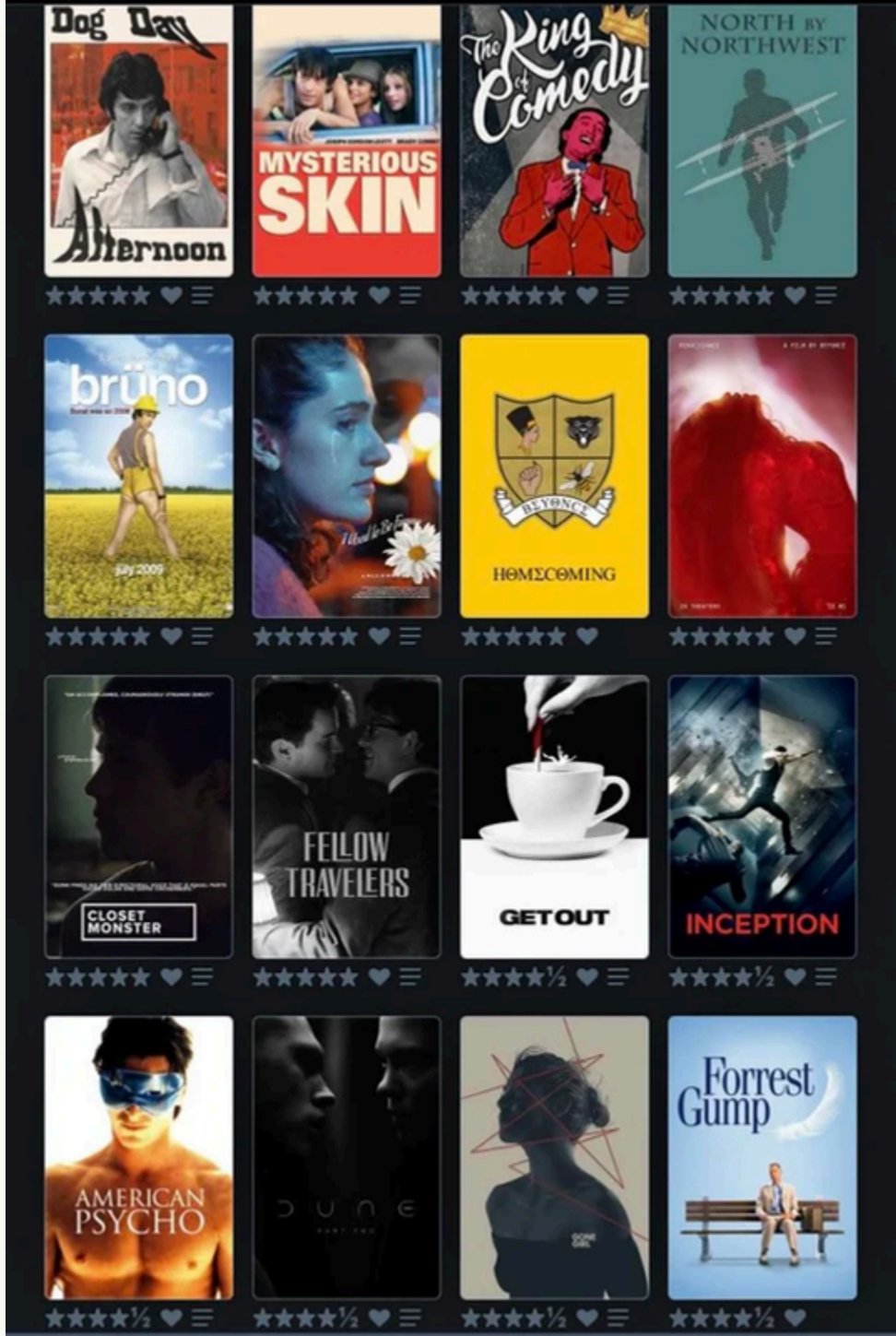
Prepared by: Beibars Anuarov

CONTENT

TOPICS COVERED

PROBLEM

Modern content platforms suffer from choice paralysis: with tens of thousands of items but sparse user interactions, simple recommendation methods cannot extract personalized preferences, leading to poor user experience and engagement loss.



WHY IS THAT?

Choice paralysis—too many options leading to decision fatigue

---

Low discovery—relevant content remains unseen despite being available

---

Poor engagement—users cannot find content matching their preferences, resulting in abandonment

USED DATA

Ready-to-use “The Movie Dataset” from Kaggle

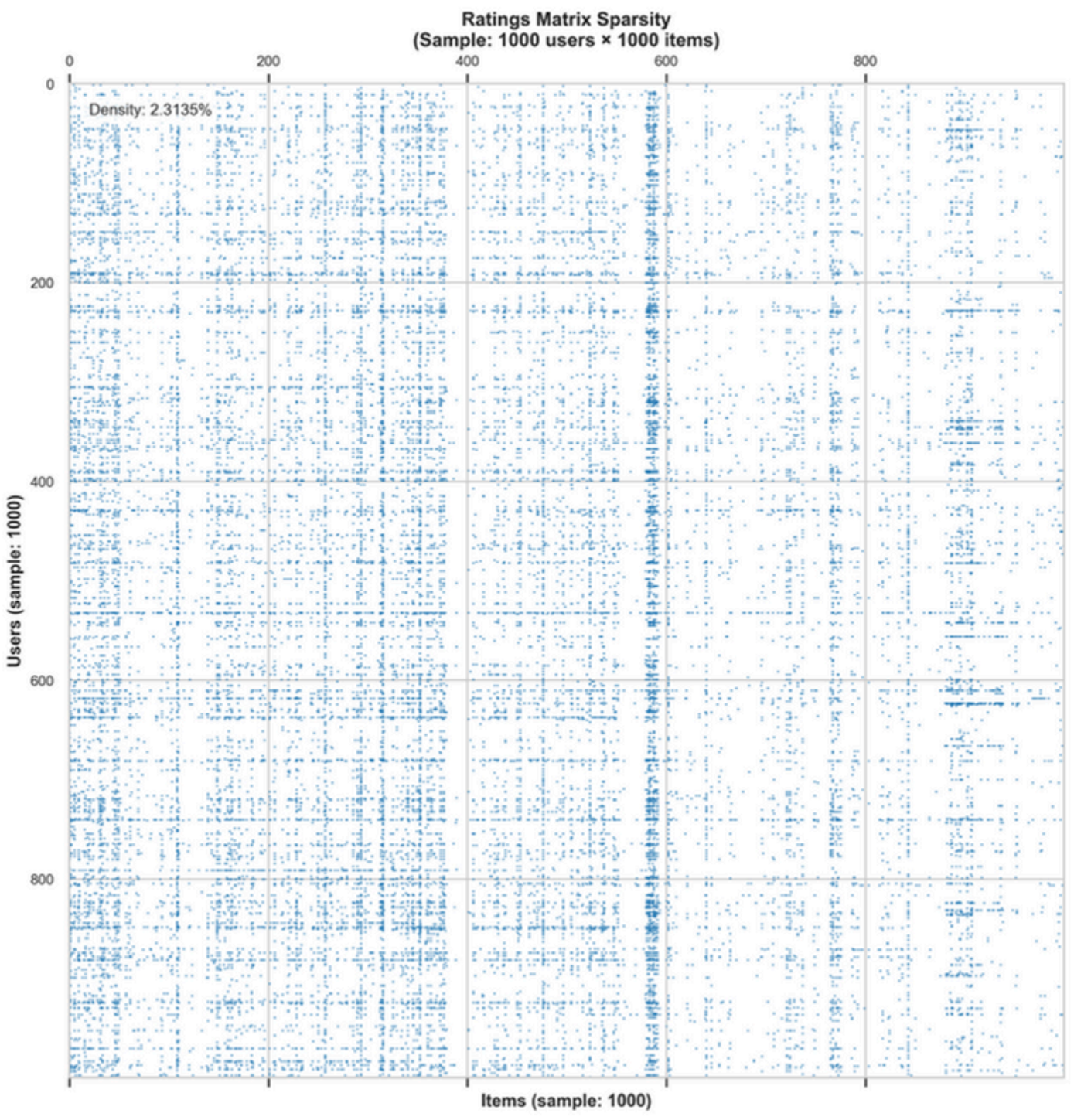
The dataset consists of movies released on or before July 2017 and has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website. has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

Personally, made by myself.

The dataset which was made by me, contains 300 movies that I’ve watched and gave my personal raiting to them. List of movies has been taken from previous dataset. So that, all the movies from this dataset mathches to previous one.

DATA EXPLANATION

Visualization of rating matrix sparsity. Each black dot represents an observed user-movie rating. The overwhelming white space illustrates that only 0.21% of possible interactions are known



METHODOLOGY

Why SVD?

Singular Value Decomposition (SVD) solves this by automatically discovering hidden 'taste dimensions' that explain why users prefer certain movies. These dimensions aren't explicitly labeled (like 'action' or 'comedy'), but the algorithm finds them by analyzing rating patterns across all users and movies.

THE INTUITION BEHIND SVD

Traditional approaches would struggle here. If we simply recommend the most popular movies to everyone, we ignore individual preferences. If we only look at what similar users rated, we miss subtle patterns that connect different types of movies.

Think of SVD as a way to compress and understand a complex rating matrix. Instead of storing all 12.3 billion possible user-movie combinations, we represent users and movies as points in a smaller space of 'taste dimensions'.

Each user gets a profile—a list of 250 numbers describing their position along these 250 hidden taste dimensions.

The 'low-rank approximation' aspect means we're using far fewer dimensions (250) than the full space would require (up to 45,379).

ITEM-MEAN CENTERING

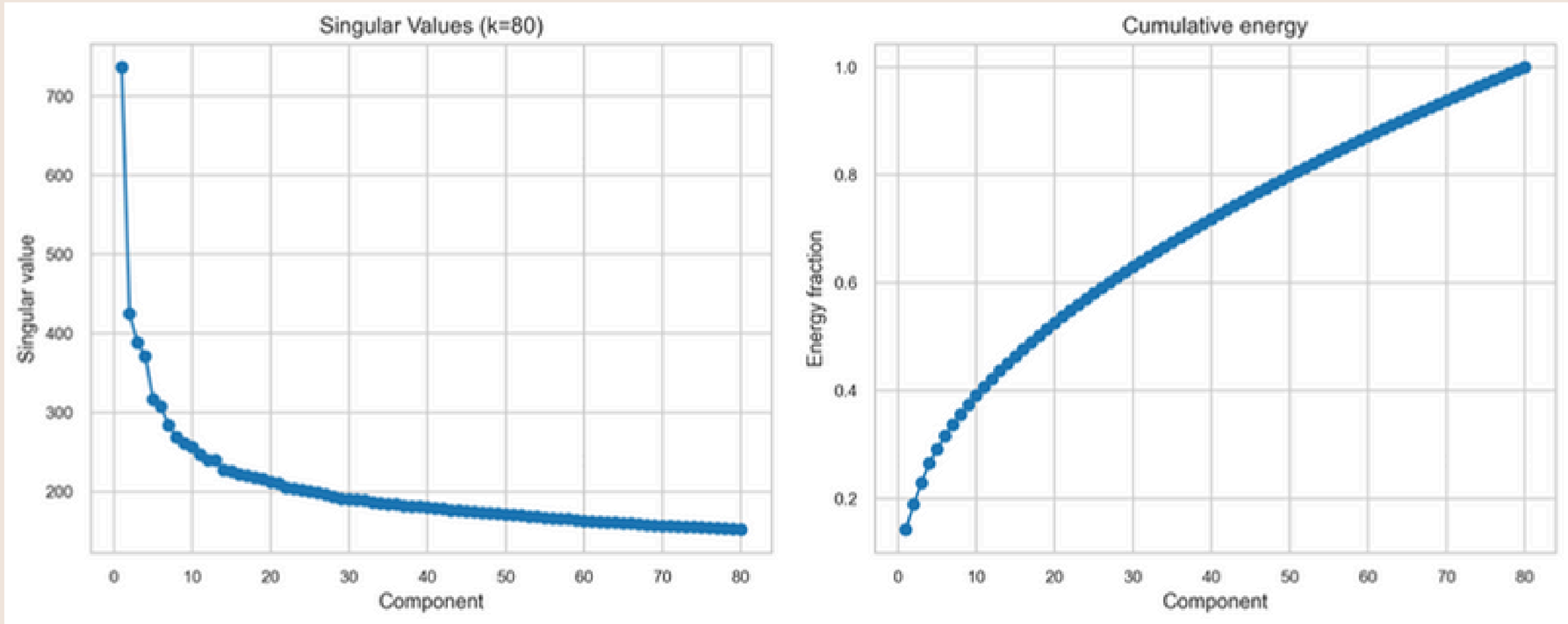
We use item-mean centering: for each movie, we calculate the average rating it received from all users (using only training data), then subtract this average from every rating of that movie. For example, if 'The Dark Knight' has an average rating of 4.5, and a user rated it 5.0, the centered value becomes +0.5. If another user rated it 3.0, the centered value becomes -1.5.

With item-mean centering, the model better captures how user preferences vary across different types of movies, leading to richer and more personalized recommendations.

SVD DECOMPOSITION

The left panel shows that the first few dimensions capture much more information than later ones—this is typical for recommendation data. The right panel shows cumulative energy: using 250 dimensions captures most of the signal in the data, with diminishing returns for additional dimensions.

Scree plot showing singular values. The left plot displays the magnitude of each singular value (dimension importance), while the right plot shows cumulative energy (how much information is captured by using k dimensions).



RECONSTRUCTING RATINGS AND MAKING PREDICTIONS

Once we have U, s, and Vt, we can reconstruct predicted ratings. The reconstruction process involves multiplying these three components together: first multiply U by the singular values s (element-wise, scaling each user's taste profile by dimension importance), then multiply the result by Vt (the movie profiles). This gives us predicted centered ratings.

For efficiency, we don't reconstruct the entire dense matrix (which would require 12.3 billion values). Instead, we compute predictions only for specific user-movie pairs on demand. This allows the model to scale to very large catalogs while maintaining fast prediction times.

THE COMPLETE TRAINING PROCESS

The training process follows a structured pipeline designed to maximize prediction accuracy while ensuring the model generalizes to unseen data.

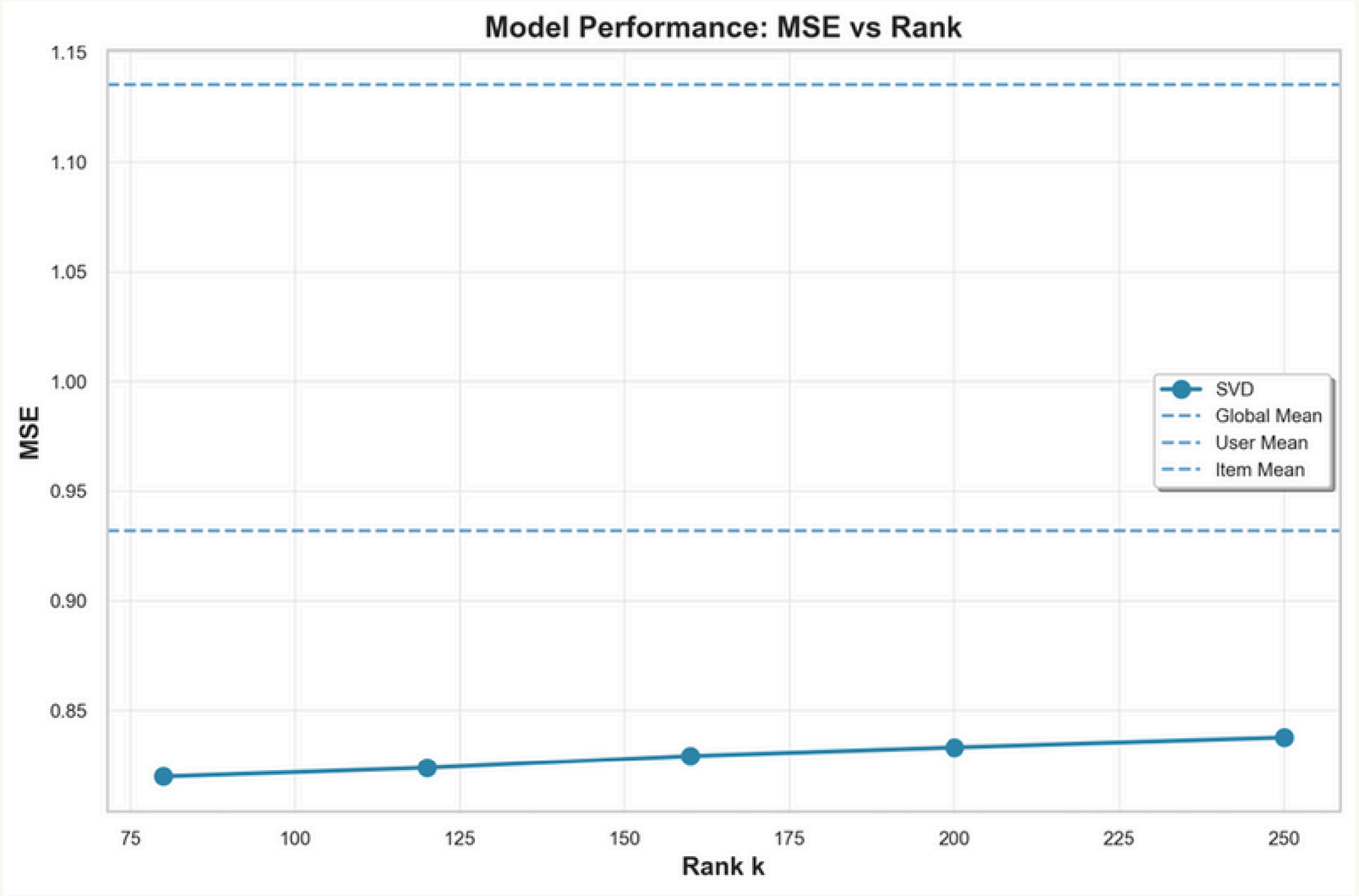
Data Loading and Matrix Construction. We load ratings from CSV files (combining general user ratings with personalized ratings from a target user). Each rating contains a user ID, movie ID, and rating value (1.0 to 5.0). We map these raw IDs to contiguous indices (0 to m-1 for users, 0 to n-1 for movies) for efficient matrix operations. The ratings are stored in Compressed Sparse Row (CSR) format, which only stores non-zero entries, dramatically reducing memory usage.

Train-Test Split. We split ratings on a per-user basis: for each user, we randomly select 20% of their ratings for testing, ensuring at least one rating remains for training. This split preserves realism—it mimics the scenario where we know some of a user's preferences and want to predict others. The split uses a fixed random seed (42) for reproducibility.

Item Mean Calculation and Centering. We compute average ratings for each movie using only training data, then subtract these means from the corresponding ratings. This creates the centered sparse matrix R_c that will be input to SVD.

Hyperparameter Sweep. We experiment with different values of k (the number of hidden dimensions): 80, 120, 160, 200, and 250. For each k, we run SVD with multiple random seeds (42, 123, 2024) to test stability. We evaluate each configuration by computing predictions on the test set and calculating Mean Squared Error (MSE).

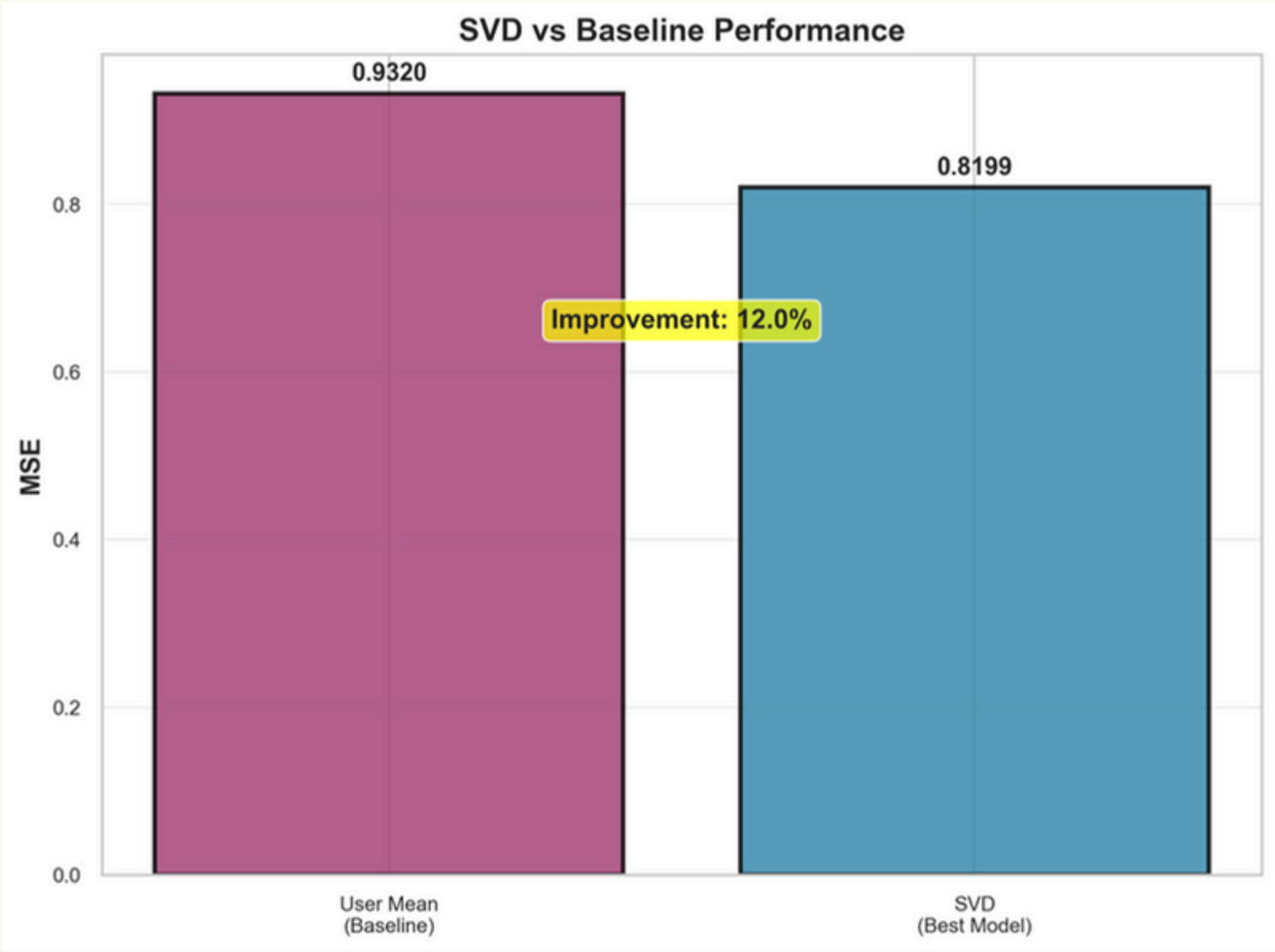
Model Selection. We select the configuration with the lowest MSE, with a preference for slightly higher ranks (within 5% MSE margin) because they produce more diverse predictions—crucial for recommendation quality. More diverse predictions mean the model can distinguish between movies with different appeal, rather than clustering all predictions near the mean.



This plot shows how prediction error changes with the number of hidden dimensions (k). Lower MSE is better. The horizontal dashed lines represent baseline methods (global mean, user mean). SVD consistently outperforms baselines.

RESULTS

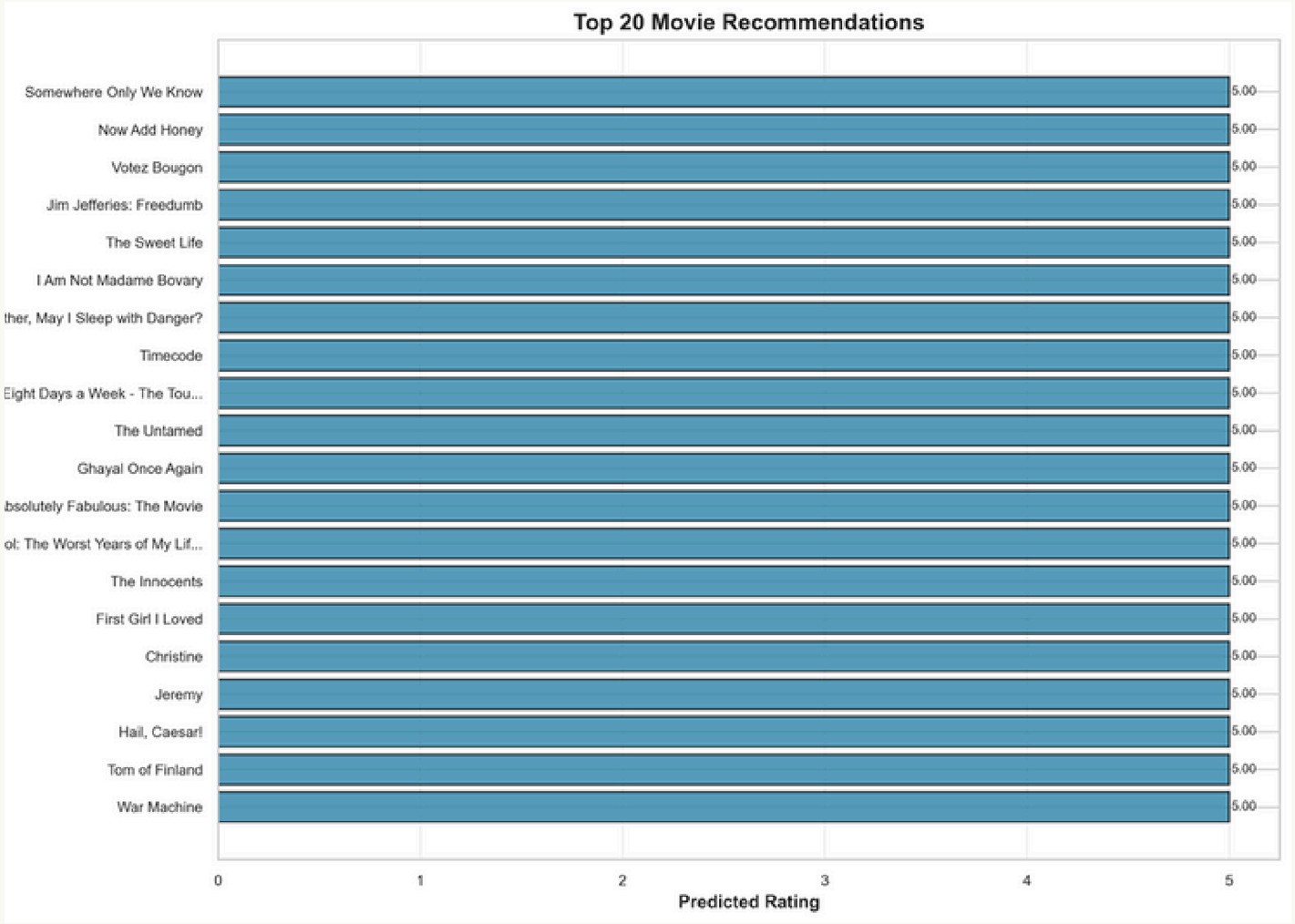
SVD outperforms all baselines (best SVD MSE: 0.820 vs best baseline MSE: 0.932, representing a 12% improvement). This demonstrates that the latent structure SVD discovers—hidden relationships between users and movies—contains information that simple averages cannot capture. SVD learns that 'users who like movies A and B also tend to like movie C,' even when this relationship isn't explicit in the data.



Direct comparison showing the improvement achieved by SVD over the best baseline (user mean). The improvement percentage demonstrates the value of capturing latent structure in collaborative filtering.

USAGE

The process is straightforward: (1) Retrieve the user's taste profile from U (250 numbers). (2) For each movie in the catalog, compute the dot product between the user's profile and the movie's profile, weighted by singular values. This gives a predicted centered rating. (3) Add back the movie's mean rating to get the final prediction. (4) Sort all movies by predicted rating. (5) Exclude movies the user has already rated. (6) Return the top N movies.



THANK YOU FOR YOUR ATTENTION