🎬 *Movie Lens Recommender System*

Movie Recommendations with Weighted Matrix Factorization

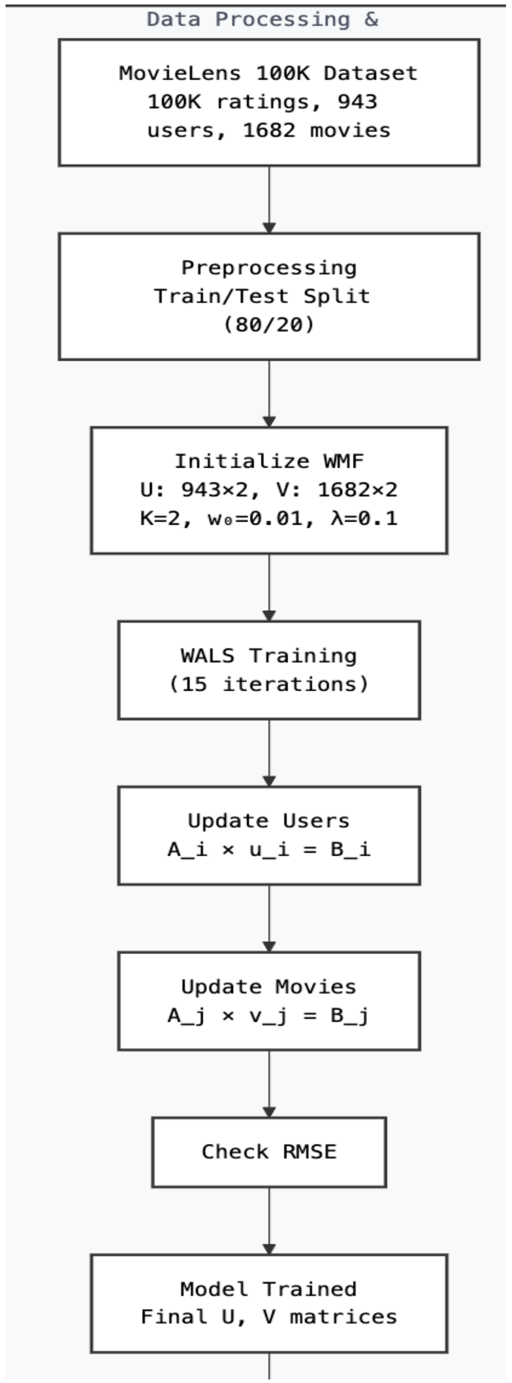
Roshanak Behrouz

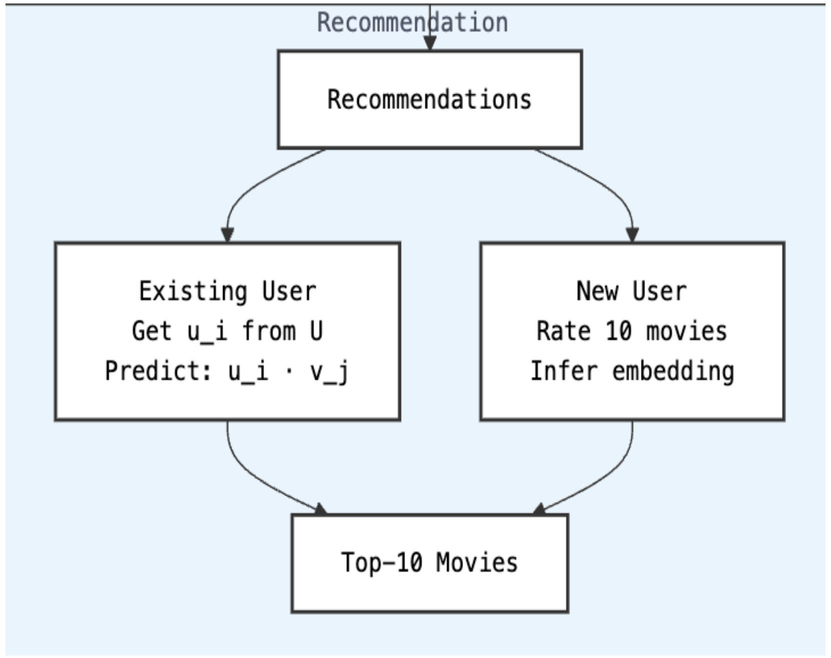
Problem Statement

- **Sparse Data**: Users typically rate <1% of available movies
- **Cold Start**: How to recommend to new users?
- **Scalability**: Need efficient algorithms for large datasets
- **Standard MF** treats missing ratings as zeros

How it works?

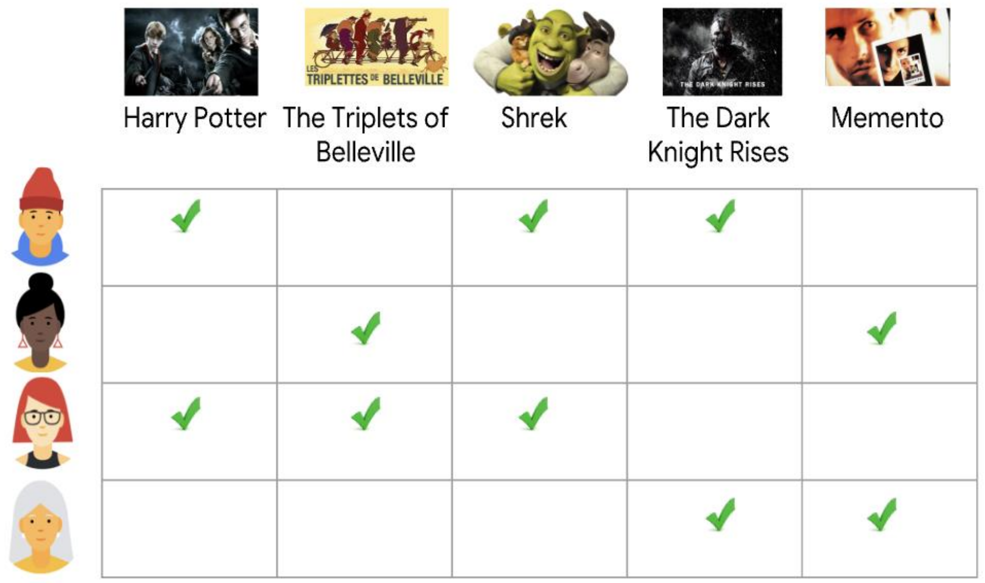
1. Data Loading
2. Model Training
3. User Interaction (Two Scenarios)
    a. For Existing Users
    b. For New Users (Cold Start)
4. Prediction Logic


Data Processing &
MovieLens 100K Dataset 100K ratings, 943 users, 1682 movies
Preprocessing Train/Test Split (80/20)
Initialize WMF U: 943×2, V: 1682×2 K=2, w₀=0.01, λ=0.1
WALS Training (15 iterations)
Update Users A_i × u_i = B_i
Update Movies A_j × v_j = B_j
Check RMSE
Model Trained Final U, V matrices


Recommendation
Recommendations
Existing User Get u_i from U Predict: u_i · v_j
New User Rate 10 movies Infer embedding
Top-10 Movies

Data Preparation

- **MovieLens 100k Dataset:** The system starts by downloading and processing the MovieLens 100k dataset.
- **Source:** GroupLens Research (http://files.grouplens.org/datasets/movielens/ml-100k.zip)
- **Content:**
  - u.data: 100,000 ratings from 943 users on 1,682 movies
  - u.item: Movie titles and genre information.
- **Feedback Matrix:**
- Each row -> user
- Each column -> movie



The Model: Weighted Matrix Factorization (WMF)

- **Goal:** Decompose the sparse user-item rating matrix into two lower-dimensional matrices:
  - **User Embeddings (U):** num_users x latent_features
  - **Item Embeddings (V):** num_movies x latent_features
- **Prediction:** The predicted rating for a movie j by user i is typically the dot product of their respective embeddings
- **User matrix (U):** latent preferences
- **Item matrix (V):** latent attributes

Prediction:

$$\hat{C}_{ij} = u_i^\top v_j$$

Optimization: Weighted Alternating Least Squares (WALS)

The loss function being implemented is:

$$\underbrace{\sum_{(i,j)\in \text{Obs}} w_{ij}(C_{ij} - u_i^T v_j)^2}_{\text{observed loss}} + \underbrace{w_0 \sum_{(i,j)\in \text{Nobs}} (u_i^T v_j)^2}_{\text{unobserved loss}} + \underbrace{\lambda(\|u\|^2 + \|v\|^2)}_{\text{regularization}}$$

Where:

- C_{ij}: known rating of user i for item j
- u_i: user latent vector (embedding)
- v_j: item latent vector
- $w_{ij} = 1$ for known (observed) ratings
- w_0: small constant weight for unknown (unobserved) ratings
- λ: regularization parameter

The main idea of the algorithm

- Start with U and V randomly generated,

$$C'_{i,j} = \mathbf{u}_i^T \mathbf{v}_j$$

- Fix V and find, by solving a linear system, the best U.
- Fix U and find, by solving a linear system, the best V.

- Repeat as needed.
- The algorithm is guaranteed to converge and can be parallelized.

Cold Start Problem for New Users

- **Inferring New User Preferences:** For a completely new user who hasn't rated any movies yet, the system cannot look up their past ratings. To address this, my system asks the new user to rate a small number of **popular movies.**
- **Dynamic Embedding:** Based on these initial ratings, the system can then infer a latent factor vector specifically for this new user, even though they weren't part of the original training data. This inferred embedding allows the model to generate personalized recommendations.

Recommendation Generation

Once the model is trained or a new user's embedding is inferred:

- **Prediction for Unrated Movies:** For a given user (existing or new), the system predicts ratings for all movies they haven't seen yet.
- **Top-N Recommendations:** It then sorts these predicted ratings and presents the TOP_N_RECOMMENDATIONS with the highest predicted ratings as the final recommendations.

Model Evaluation During Training

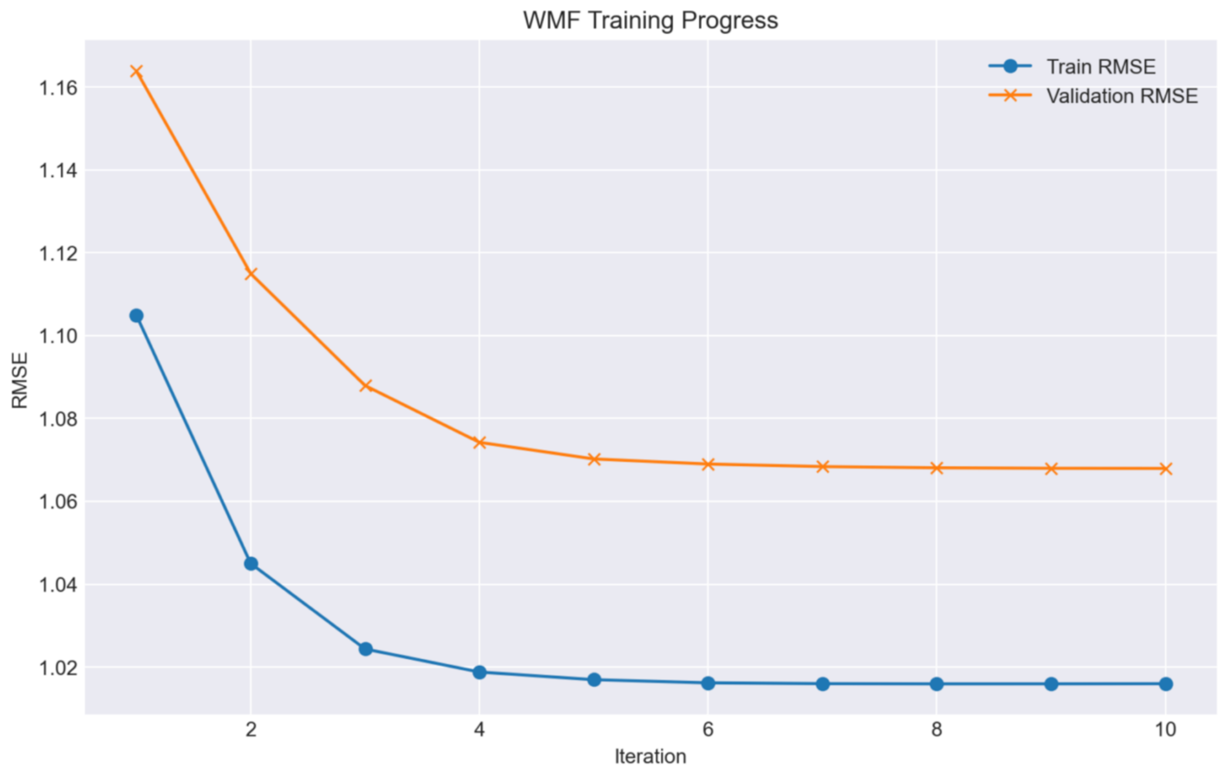
In this movie recommender system, Root Mean Squared Error is a metric used to evaluate the performance of Weighted Matrix Factorization (WMF) model.

It quantifies the average magnitude of the errors between predicted ratings and actual ratings.

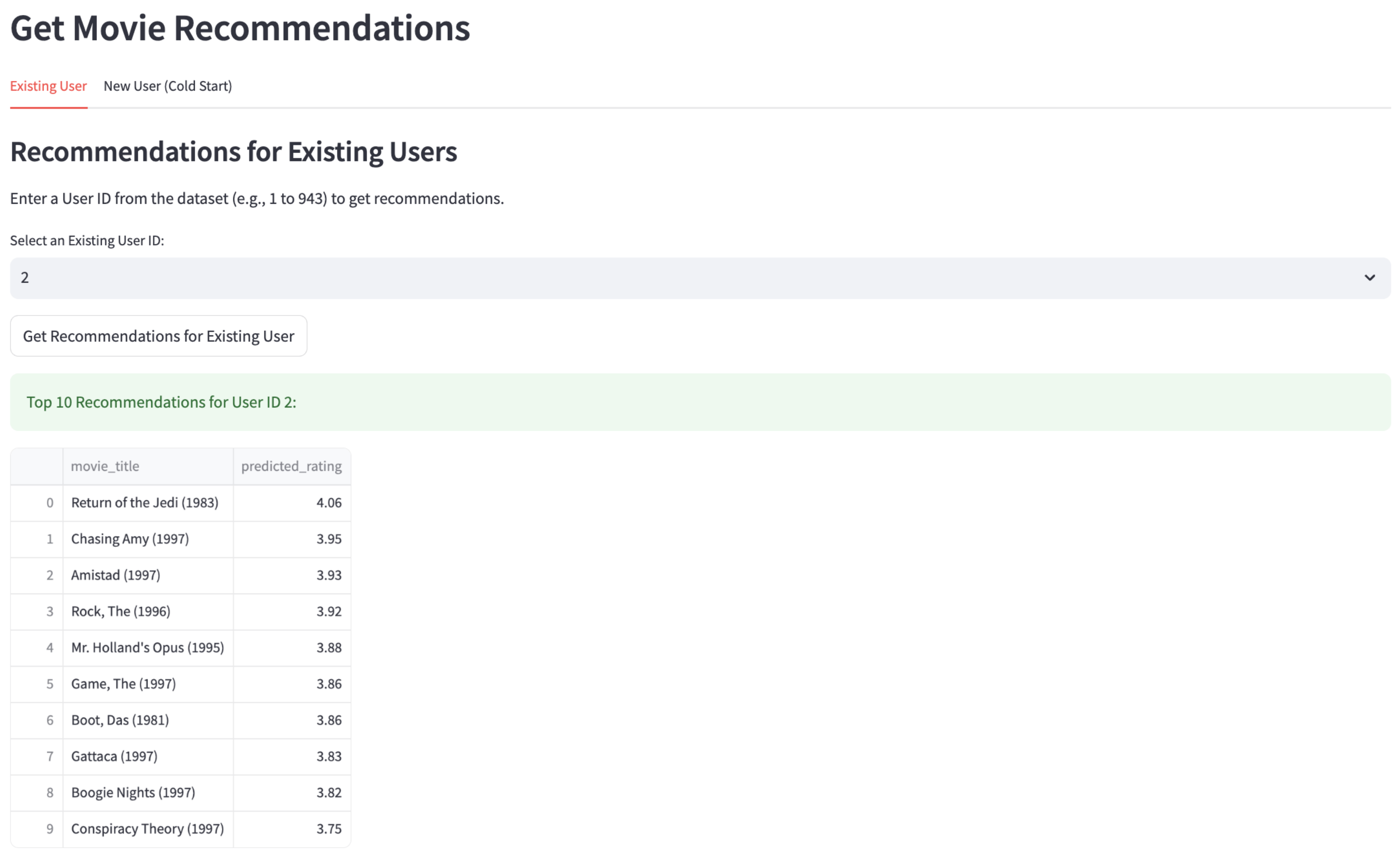
During the training process of the WMF model using Weighted Alternating Least Squares RMSE is calculated at the end of each iteration for both the training dataset and a separate validation dataset.

Learning Curve

- **Plot:** Shows how Training RMSE and Validation RMSE change with each iteration.
- **Interpretation:**
  - Decreasing RMSE indicates the model is learning.
  - 5 iterations seems enough.



The APP



Get Movie Recommendations

Recommendations for Existing Users

Enter a User ID from the dataset (e.g., 1 to 943) to get recommendations.

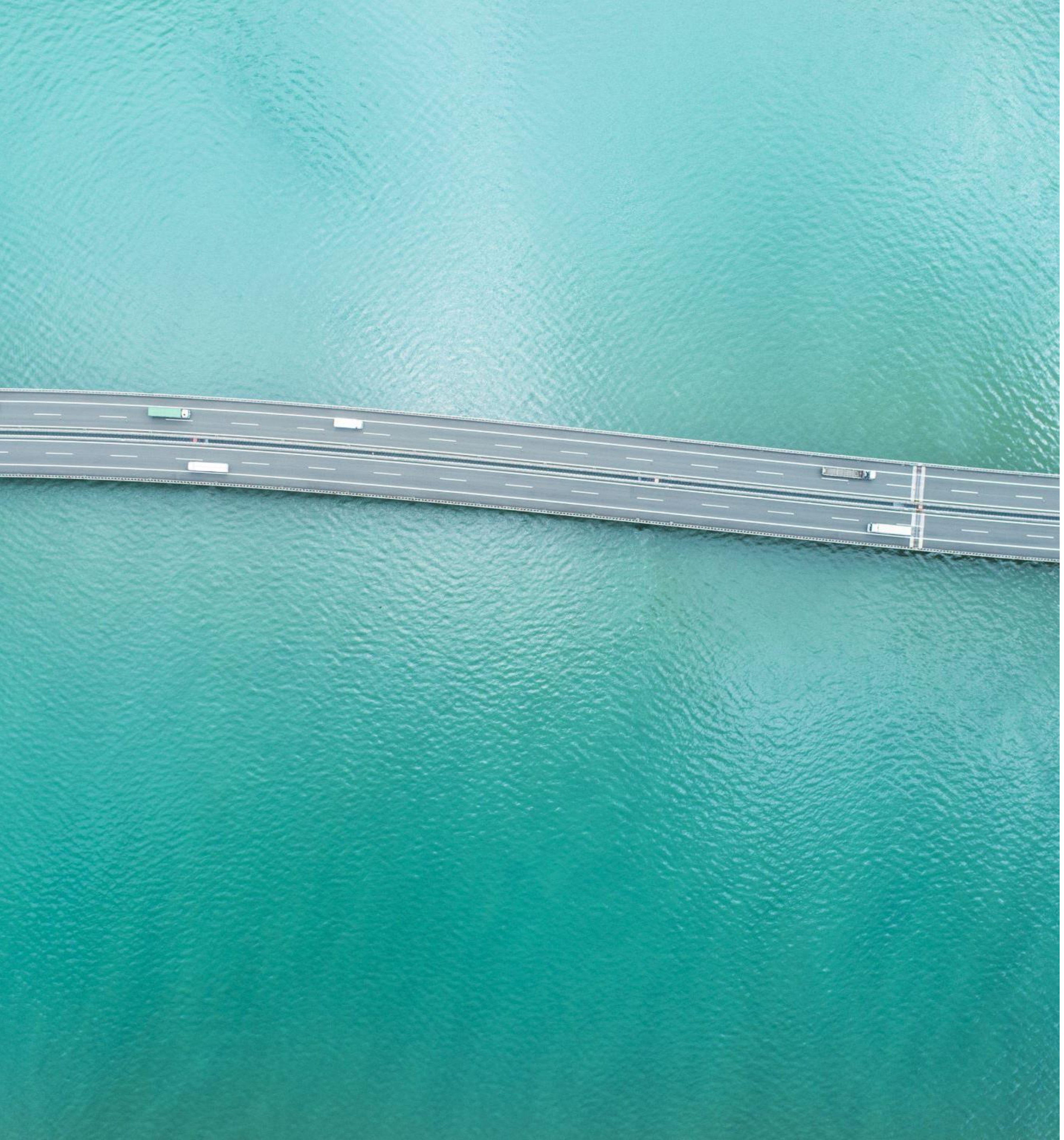
Select an Existing User ID: 2

Top 10 Recommendations for User ID 2:

	movie_title	predicted_rating
0	Return of the Jedi (1983)	4.06
1	Chasing Amy (1997)	3.95
2	Amistad (1997)	3.93
3	Rock, The (1996)	3.92
4	Mr. Holland's Opus (1995)	3.88
5	Game, The (1997)	3.86
6	Boot, Das (1981)	3.86
7	Gattaca (1997)	3.83
8	Boogie Nights (1997)	3.82
9	Conspiracy Theory (1997)	3.75

Future Work

- **Potential Improvements:**
  - **Hyperparameter Tuning:** Systematically find optimal latent_features, w0, reg_lambda, and iterations.
  - **Scalability:** For larger datasets, consider distributed WALS implementations



Thank you