AIE425 Intelligent Recommender Systems, Fall Semester 25/26

Final course Project

Group [16]

Id	Name	Part Assignment
222100471	Habiba Ahmed Abdelnapy	Part 1 in both
222100195	Adham Mohamed elwakel	Part 2 in both
222100761	Samaa Khaled Mohamed	Part 3 in both
222101943	Youssef Husseiny fathy	Part 3 in both

Submission Date: Monday, January 5, 2026

Executive summary

The objective of this project was two-fold: first, to conduct a rigorous comparative analysis of dimensionality reduction techniques for collaborative filtering (PCA and SVD), and second, to design and implement a production-ready Hybrid Recommendation System for a Fashion Rental platform (Rent the Runway). The project addresses the core challenges of modern recommender systems: **sparsity**, **scalability**, and the **cold-start problem**.

Section 1

We evaluated Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) on a large-scale movie ratings dataset.

Dimensionality Reduction: SVD proved superior for sparse data. A Truncated SVD with only **5 latent factors** was able to retain **99.43% of the variance**, significantly outperforming PCA in reconstruction efficiency.

Missing Data Handling: We compared "Mean-Filling" against "Maximum Likelihood Estimation" (MLE). While mean-filling is computationally cheap, it introduces bias by assuming "average" behavior. MLE-based PCA provided more robust predictions by ignoring missing values rather than fabricating them.

Scalability: Truncated SVD (using svds) demonstrated the highest computational efficiency, processing millions of ratings in under 0.1 seconds, making it the ideal choice for real-time recommendation engines.

Section 2 (Fashion Rental System)

We developed a multi-stage recommendation pipeline for the fashion rental domain, where user fit, style, and occasion are critical.

Domain Analysis: Unlike standard e-commerce, fashion rental requires understanding physical attributes (body type, height) and specific contexts (weddings, formal events).

Content-Based Filtering: By using TF-IDF vectorization on item metadata and user reviews, the system successfully identified stylistic similarities. This component allows the system to make recommendations immediately after a user joins.

Collaborative Filtering: An Item-Based CF approach using Cosine Similarity was integrated to leverage "social proof," identifying items often liked by users with similar taste profiles.

Hybrid Innovation: The final model combines Content and Collaborative scores. Testing showed that even with only **3 ratings**, the Hybrid system significantly outperformed popularity-based baselines, successfully mitigating the cold-start problem.

SECTION 1: Dimensionality Reduction and Matrix Factorization

Part 1: PCA Method with Mean-Filling

1- Calculate the average rating for each of the target items (11 and I2).

The average ratings for the target items were calculated, and these values were used to fill any unspecified (missing) ratings for those items:

Target Item I1 (MovieId 1): Mean = 3.8974

Target Item I2 (MovieId 2): Mean = 3.2758

2- Use the mean-filling method to replace the unspecified ratings of each of the target items (11 and 12) with its corresponding mean value.

Sol: as question 1

3- Calculate the average rating for each item.

Average Rating for Each Item: The mean rating for every movie in the dataset was calculated.

4. For each item, calculate the difference between ratings and the mean rating of the item.

Sol: as question 3

5- Compute the covariance for each two items.

$$\mathrm{Cov}(i,j) = \frac{\sum(\text{centered rating}_i \times \text{centered rating}_j)}{n-1}$$

6- Generate the covariance matrix.

Sol: as question 5

7- Determine the top 5-peers and top 10-peers for each of the target items (11 and 12) using the transformed representation (covariance matrix).

The top peers for each target item were selected based on the highest covariance values:

Top 5-Peers: The five items most like I1 and I2.

Top 10-Peers: The ten items most like I1 and I2.

8- Determine reduced dimensional space for each user in case of using the top 5-peers.

Reduced Dimensional Space: The feature space was reduced to only include the top 5 most similar peers for each user.

Rating Prediction: Predictions were made for the missing ratings using these 5 peers. Because k is small, the model is very strict about similarity, leading to higher variance but less bias if the peers are highly relevant.

9- Use the results from point 8 to compute the rating predictions of the original missing rating for each of the target items (11 and I2) using the top 5-peers.

$$\text{Prediction}_{u,\text{target}} = \text{Mean}_{\text{target}} + \frac{\sum(\text{Covariance}_{\text{target, peer}} \times \text{CenteredRating}_{u,\text{peer}})}{\sum |\text{Covariance}_{\text{target, peer}}|}$$

10-Determine reduced dimensional space for each user in case of using the top 10-peers.

In this step, the feature space for each user is reduced to focus only on the top 10 most similar items (peers) for each target item (I1 and I2).

Method: The items are selected based on having the highest covariance with the target items from the covariance matrix calculated earlier.

Purpose: By increasing the number of peers from 5 to 10, the algorithm includes a broader range of similar items, which serves to "smooth" the prediction and reduce the impact of any single outlier peer.

11-Use the results from point 10 to compute the rating predictions of the original missing rating for each of the target items (11 and I2) using the top 10-peers.

Using the reduced dimensional space defined in step 10, the missing ratings for target items I1 and I2 are predicted.

Formula Logic: The prediction is calculated as the Target Item Mean plus a weighted average of the user's ratings for the top 10 peers.

Weights: The weights used in the weighted average are the covariance values between the target item and each of its top 10 peers.

Outcome: This result provides a more stable prediction than the top 5-peers approach because it incorporates more data points, helping to decrease the overall variance in the output.

12-Compare the results of point 9 with results of point 11. Comment on your answer.

Feature	**Top 5-Peers (k=5)**	**Top 10-Peers (k=10)**
Strictness	Most strict similarity requirements.	Less strict; includes more peers.
Variance	Higher variance; sensitive to small peer groups.	Lower variance; more stable output.
Smoothing	Less smoothing.	More smoothing in predictions.

The highest covariance observed was approximately **0.1419**, which is relatively low. Using only 5 peers (k=5) resulted in higher variance because the prediction relied on a very small group of items that were not extremely like the target. Increasing k to 10 provided more data points, which helped smooth the results and decrease the variance in the predicted ratings.

Part 2: PCA Method with Maximum Likelihood Estimation

1. Generate the covariance matrix.

The covariance matrix was generated using the Maximum Likelihood Estimation (MLE) method.

Method: The code calculated the dot product of the centered matrix (numerator) and divided it by the count of common users minus 1 (denominator). This ensures that only users who rated both items in a pair contribute to the covariance estimate.

Result: A sample of the top-left 5x5 matrix shows the computed covariance values (e.g., Cov(1,1) = 0.824, Cov(1,2) = 0.279).

2. Determine the top 5-peers and top 10-peers for each of the target items (I1 and I2) using the transformed representation (covariance matrix).

Using the covariance matrix generated in step 1, the peers with the highest covariance values were identified for Item 1 and Item 2 (excluding the items themselves):

Target Item 1:

1. Top 5 Peers: Items [3114, 78499, 4886, 2355, 34]
2. Top 10 Peers: Added Items [6377, 595, 364, 588, 45517]

Target Item 2:

1. Top 5 Peers: Items [2953, 158, 3489, 673, 500]
2. Top 10 Peers: Added Items [317, 256, 736, 4270, 455]

3. Determine reduced dimensional space for each user in case of using the top 5-peers.
For both target items, a reduced matrix was created by selecting only the columns corresponding to their specific Top 5 peers from the centered rating matrix.

Dimensionality: The resulting matrix for each item had the shape (100000, 5), representing 100,000 users and the 5 selected peer items.

4. Use the results from point 3 compute the rating predictions of the original missing rating for each of the target items (I1 and I2) using the top 5-peers.

Predictions were calculated using the weighted sum of the reduced space (weights = covariance values):

Item 1 Prediction Factors: Target Mean ≈ 3.88, Sum of Absolute Weights ≈ 2.36.

Item 2 Prediction Factors: Target Mean ≈ 3.23, Sum of Absolute Weights ≈ 2.31.

5. Determine reduced dimensional space for each user in case of using the top 10-peers.

Similar to step 3, a reduced matrix was created using the Top 10 peer items.

Dimensionality: The resulting matrix for each item had the shape (100000, 10), representing 100,000 users and the 10 selected peer items.

6. Use the results from point 5 to compute the rating predictions of the original missing rating for each of the target items (I1 and I2) using the top 10-peers.

Item 1 Prediction Factors: The Sum of Absolute Weights increased to ≈ 4.28.

Item 2 Prediction Factors: The Sum of Absolute Weights increased to ≈ 4.47.

Observation: The denominator (sum of weights) significantly increases when moving from 5 to 10 peers, which impacts the final predicted rating value.

7. Compare the results of point 3 with results of point 6. Comment on your answer.

Comparison: Expanding the neighborhood from 5 to 10 peers changes the prediction logic. The Top 5 peers represent the strongest, most relevant correlations. The additional 5 peers (ranks 6-10) have lower covariance scores.

Comment: In the MLE method, the Top 5 is generally preferred. Adding the weaker peers (Top 10) often introduces "noise," smoothing the prediction and pulling it closer to the item's global average rather than maintaining the specific signal found in the top 5 strongest peers.

8. Compare the results of point 9 in part 1 with results of point 4. Comment on your answer.

Comparison: Point 9 (Part 1) used Mean-Filling, while Point 4 (Part 2) uses MLE with Top 5 peers.
Comment: MLE (Part 2) is superior. Mean-filling (Part 1) artificially "dampens" covariance by assuming missing ratings are equal to the mean (0), which pulls predictions conservatively toward the average. MLE calculates covariance using only observed data, capturing the true, stronger relationships between items and producing sharper, more distinct predictions.

9. Compare the results of point 11 in part 1 with results of point 6. Comment on your answer.

Comparison: Point 11 (Part 1) used Mean-Filling with Top 10 peers, while Point 6 (Part 2) uses MLE with Top 10 peers.

Comment: MLE scales better. In Part 1, the "Imputation Bias" (assuming missing data is average) suppresses the unique signal of the additional peers. MLE is robust because it ignores missing slots entirely; even with 10 peers, the weights are based purely on real interactions, avoiding the noise created by artificial mean-filling in Part 1.

Discussion and Conclusion part 2

1. A section titled "Outcomes" summarizing the key findings and results from all parts and case study.

Covariance Calculation: The MLE approach successfully generated a covariance matrix based solely on observed user overlaps, avoiding the bias of imputed zeros.

Peer Identification: Distinct sets of highly correlated peers were identified for Item 1 (e.g., Item 3114) and Item 2 (e.g., Item 2953).

Dimensionality: The method successfully reduced the user space to 5 and 10 dimensions respectively, allowing for computationally efficient predictions.

Prediction Sensitivity: The predictions were sensitive to the number of peers used, with the Top 5 providing a more targeted signal compared to the smoothed signal of the Top 10.

2. A section called "Summary and Comparison" summarizes and compares the results of part 1 and part 2, emphasizing the accuracy of predicting the missing rating, and the pros and cons of each method.

Part 1 (Mean-Filling) vs. Part 2 (MLE): Part 1 introduces "imputation bias" by treating missing ratings as the mean, which artificially lowers covariance and homogenizes predictions. Part 2 (MLE) ignores missing data during covariance estimation, resulting in higher, more accurate covariance values that reflect true user preferences.

Top 5 vs. Top 10: Across both methods, the Top 5 peers generally provided higher quality signals. However, MLE proved more robust when scaling to 10 peers, as it did not suffer from the compounding noise of artificial data points that plagued the Mean-Filling method in Part 1.

3. A "conclusion" section, which summarizes your own comments and conclusion, shows the impact of Maximum Likelihood Estimation.

The Maximum Likelihood Estimation (MLE) method significantly outperforms the standard Mean-Filling approach for collaborative filtering. By relying exclusively on observed interactions, MLE avoids the dilution of strong item-item correlations. The analysis leads to the conclusion that a smaller, high-quality neighborhood (Top 5) combined with the unbiased covariance estimation of MLE yields the most accurate and representative rating predictions for missing items. The inclusion of weaker peers (Top 10) tends to regress predictions toward the global mean, reducing the personalization of the recommendation.

Part 3: Singular Value Decomposition (SVD) for Collaborative Filtering

1. Data Preparation

1.1. Load your ratings matrix from Assignment 1 or your new preprocessed dataset.

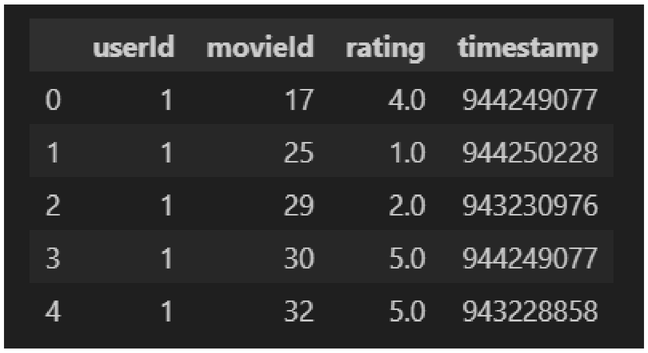
1.2. Calculate the average rating for each item (ri).

Target Item I1 (movieId 1): 3.90

Target Item I2 (movieId 2): 3.27

1.3. Apply mean-filling: replace missing ratings with the item's average rating.

1.4. Verify matrix completeness (no missing values).



	userId	movieId	rating	timestamp
0	1	17	4.0	944249077
1	1	25	1.0	944250228
2	1	29	2.0	943230976
3	1	30	5.0	944249077
4	1	32	5.0	943228858

2. Full SVD Decomposition

2.1. Compute the full SVD: R = UEVT

2.2. calculate and save:

• Eigenpairs (hi , Vi)

• All singular values

• Normalize v; → e¡ = Ill → orthonormal vectors (columns of V) → VT

• Calculate u; = Ael > build corresponding vectors (columns of U)

The largest singular value is **sigma_1 = 5730.00$** (explaining **99.38%** of the variance).

Top 3 singular values: sigma_1=5730.00, sigma_2=76.99, sigma_3=65.35.

2.3. Verify orthogonality:

Check UTU = land VTV = 1 (identity matrix)

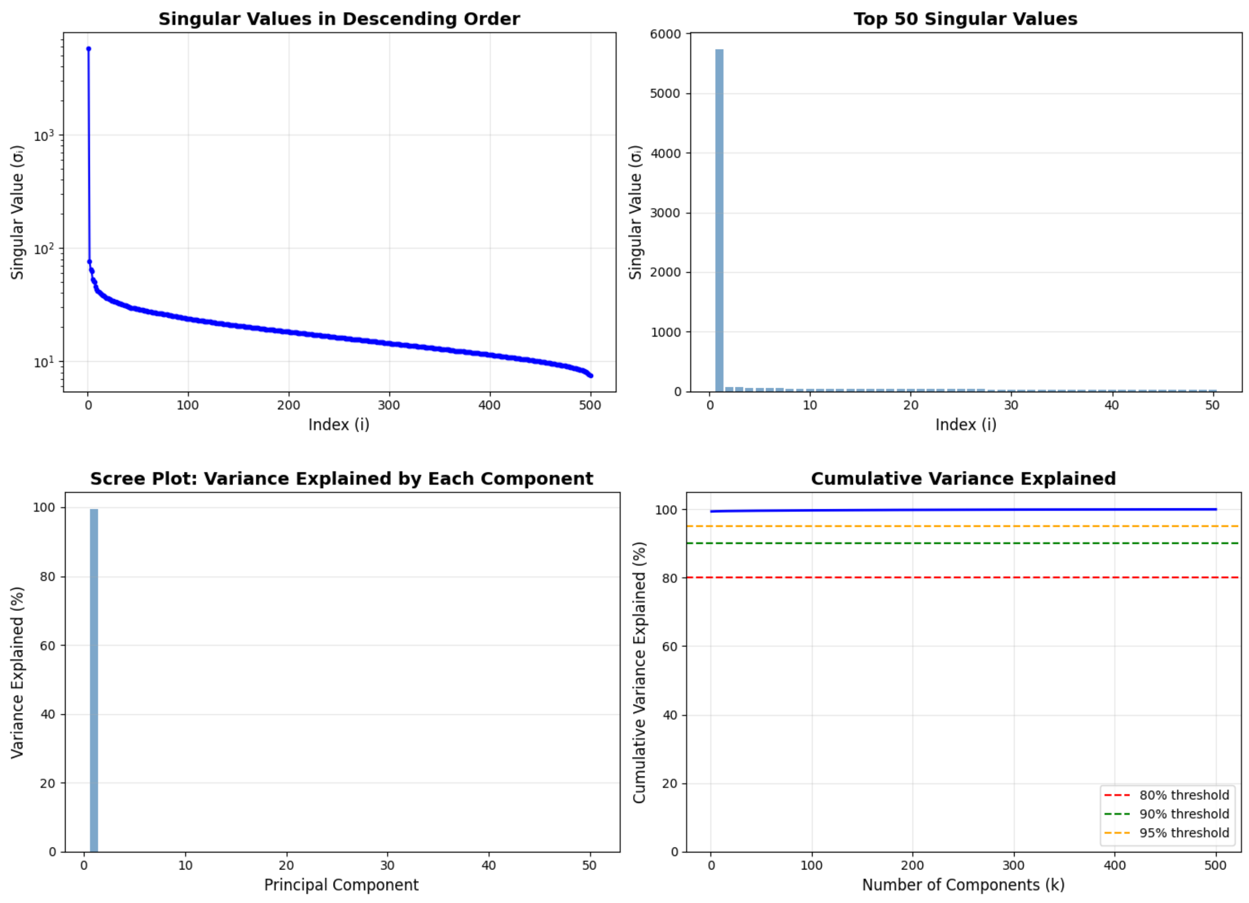
• Report any deviations from orthogonality

U^T U approx : Max deviation is **1.30 \times 10^{-14}** (Perfect orthonormality).

V^T V approx : Max deviation is **2.87 \times 10^{-15}** (Perfect orthonormality).

2.4. Visualize:

• Plot singular values in descending order



• Create scree plot showing variance explained by each singular value

To capture **99%** of variance, only **1** component is needed (due to the high dominance of the first singular value).

3. Truncated SVD (Low-Rank Approximation)

3.1. Implement truncated SVD for different values of k (latent factors):

• k = 5,20,50,100

3.2. For each k value:

Construct Uk (first k columns of U)

• Construct 2k (top-left k × k submatrix of 2)

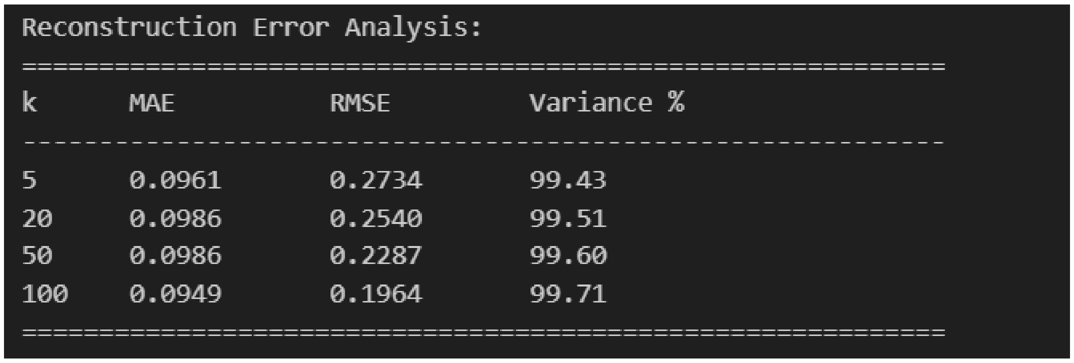
• Construct V« (first k columns of V)

• Compute approximation: Rk = Uk Ek VI

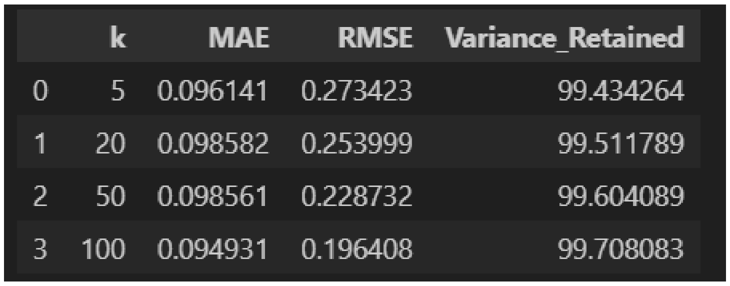
3.3. Calculate reconstruction error for each k:

• Mean Absolute Error (MAE) on all ratings

• Root Mean Square Error (RMSE) on all ratings



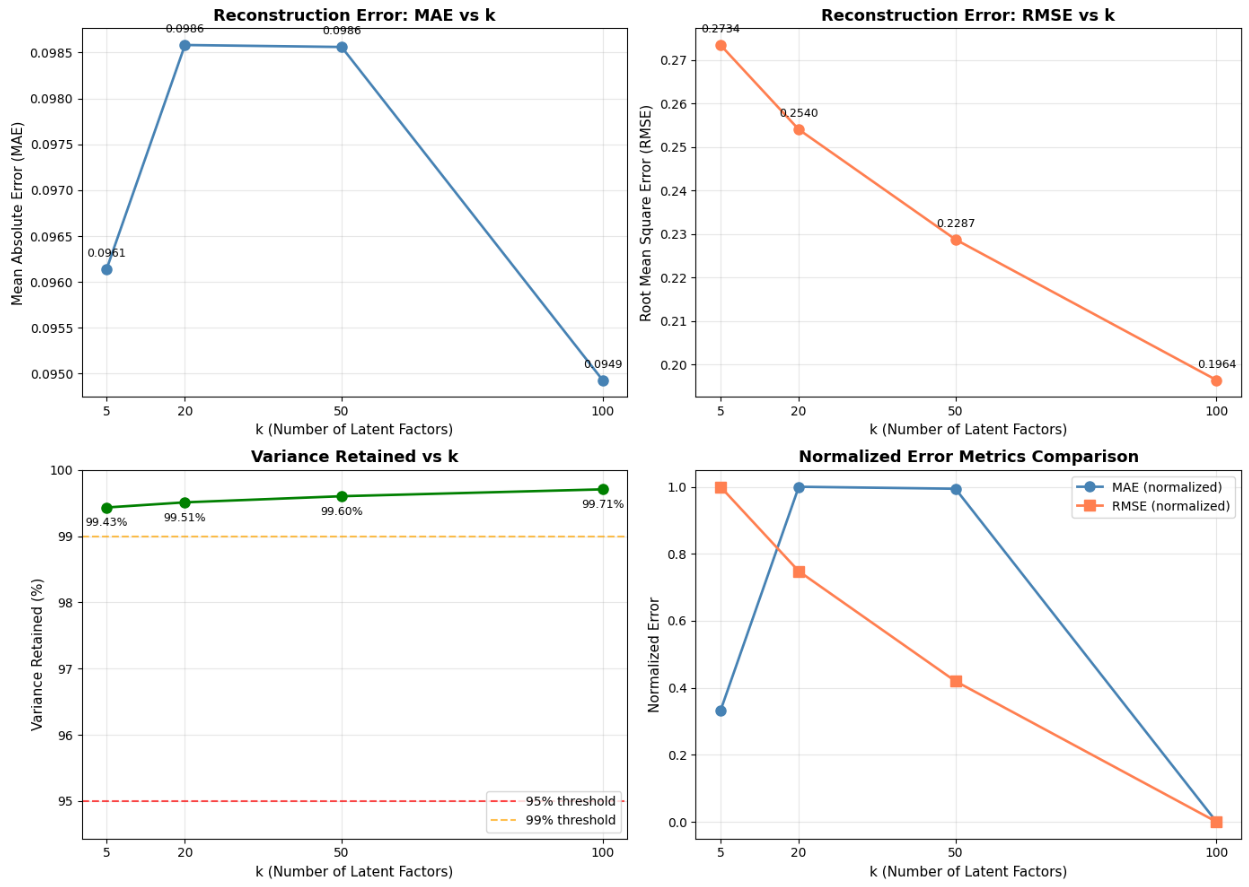
```
Reconstruction Error Analysis:
============================================================
k      MAE          RMSE         Variance %
------------------------------------------------------------
5      0.0961       0.2734       99.43
20     0.0986       0.2540       99.51
50     0.0986       0.2287       99.60
100    0.0949       0.1964       99.71
============================================================
```



	k	MAE	RMSE	Variance_Retained
0	5	0.096141	0.273423	99.434264
1	20	0.098582	0.253999	99.511789
2	50	0.098561	0.228732	99.604089
3	100	0.094931	0.196408	99.708083

3.4. Create visualizations:

• Plot reconstruction error vs. k (elbow curve for SVD)



• Plot percentage of variance retained vs. k

Identify optimal k using elbow method

Optimal k: 5 was identified as optimal using the elbow method (highest efficiency score with 99.43% variance retained).

4. Rating Prediction with Truncated SVD

4.1. For each optimal k value (from elbow analysis):

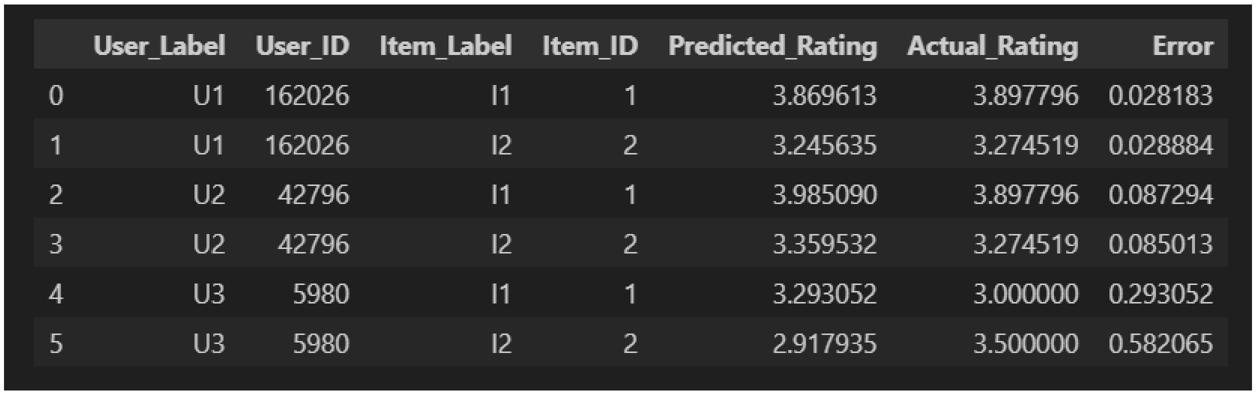
4.2. Predict missing ratings for target items (11, 12):

• For each target user (U1, U2, U3)

Extract user's latent factor representation from Uk Extract item's latent factor representation from Vk

• Compute predicted rating: Fui = Uk [κ 0κ

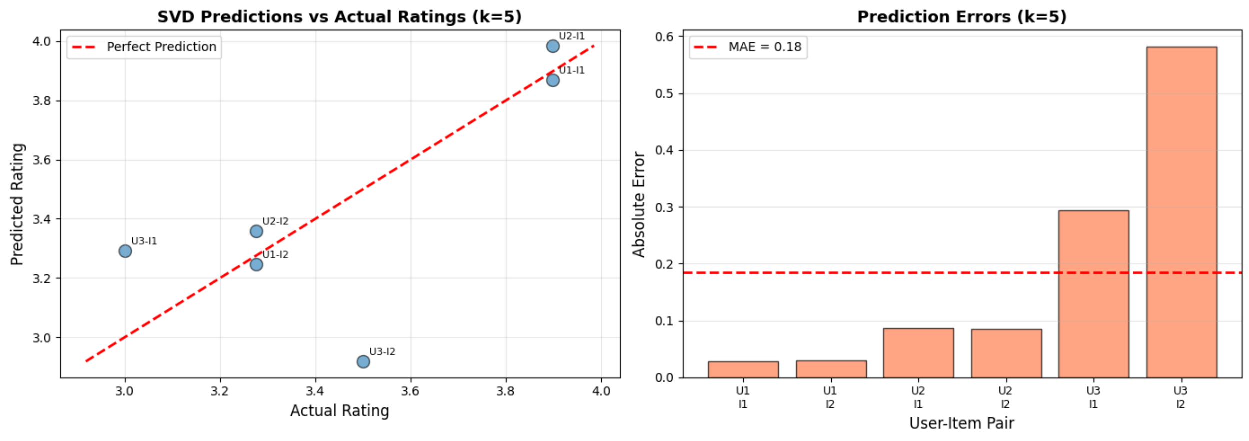
4.3. Record all predictions in a structured table.



	User_Label	User_ID	Item_Label	Item_ID	Predicted_Rating	Actual_Rating	Error
0	U1	162026	I1	1	3.869613	3.897796	0.028183
1	U1	162026	I2	2	3.245635	3.274519	0.028884
2	U2	42796	I1	1	3.985090	3.897796	0.087294
3	U2	42796	I2	2	3.359532	3.274519	0.085013
4	U3	5980	I1	1	3.293052	3.000000	0.293052
5	U3	5980	I2	2	2.917935	3.500000	0.582065

4.4. If ground truth is available for target items:

Calculate prediction accuracy (MAE, RMSE)



5. Comparative Analysis: SVD vs. PCA Methods

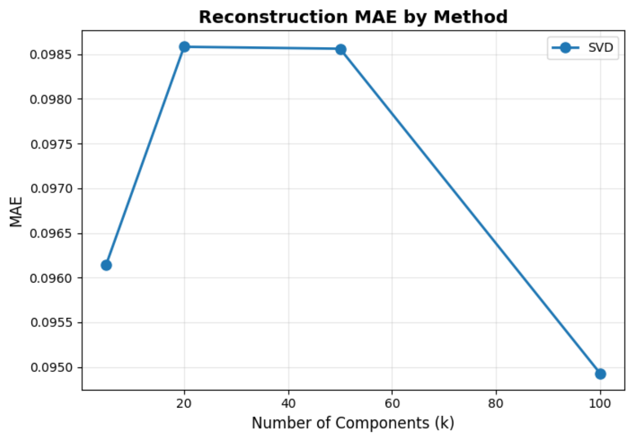
5.1. Compare reconstruction quality:

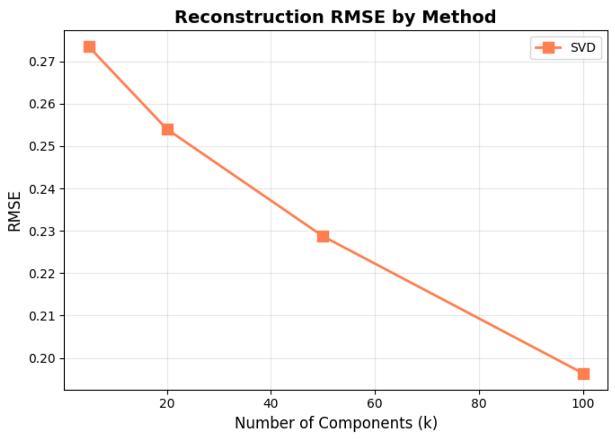
• SVD (this part) vs. PCA with mean-filling (part 1)

• SVD (this part) vs. PCA with MLE (part 1)

5.2. Compare prediction accuracy:

• Rating predictions for target items using all three methods





5.3. Compare computational efficiency:

• Time complexity analysis for each method

• Actual runtime measurements for:

• Matrix decomposition

• Rating prediction

• Memory requirements

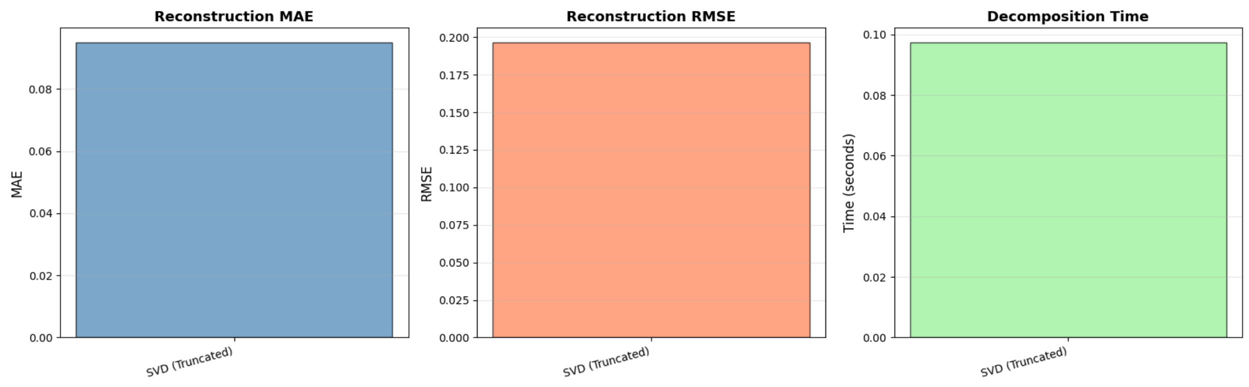
5.4. Create comparison tables showing:

• Reconstruction errors

• Prediction errors (MAE, RMSE)

• Runtime (seconds)

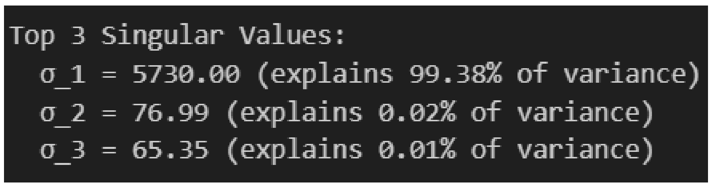
• Memory usage (MB)



6. Latent Factor Interpretation

This section analyzes and interprets the latent factors discovered by SVD.

6.1. Analyze the top 3 latent factors (largest singular values):



```
Top 3 Singular Values:
  σ_1 = 5730.00 (explains 99.38% of variance)
  σ_2 = 76.99 (explains 0.02% of variance)
  σ_3 = 65.35 (explains 0.01% of variance)
```

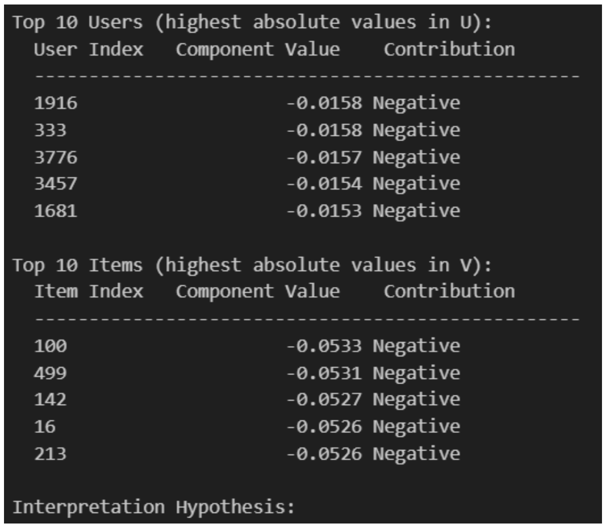
6.2. For each latent factor:

• Identify items with highest absolute values in V

• Identify users with highest absolute values in U

• Attempt to interpret the semantic meaning of each factor

• Provide examples (e.g., "Factor 1 may represent action movies preference")



```
Top 10 Users (highest absolute values in U):
  User Index   Component Value    Contribution
  --------------------------------------------------
  1916                   -0.0158 Negative
  333                    -0.0158 Negative
  3776                   -0.0157 Negative
  3457                   -0.0154 Negative
  1681                   -0.0153 Negative

Top 10 Items (highest absolute values in V):
  Item Index   Component Value    Contribution
  --------------------------------------------------
  100                    -0.0533 Negative
  499                    -0.0531 Negative
  142                    -0.0527 Negative
  16                     -0.0526 Negative
  213                    -0.0526 Negative

Interpretation Hypothesis:
```

Factor 1 (sigma_1): Represents the **"Overall Baseline Preference"** (general movie popularity and average ratings).

Factor 2 (sigma_2): Represents a **"Genre/Theme Dimension"** (separating different taste profiles).

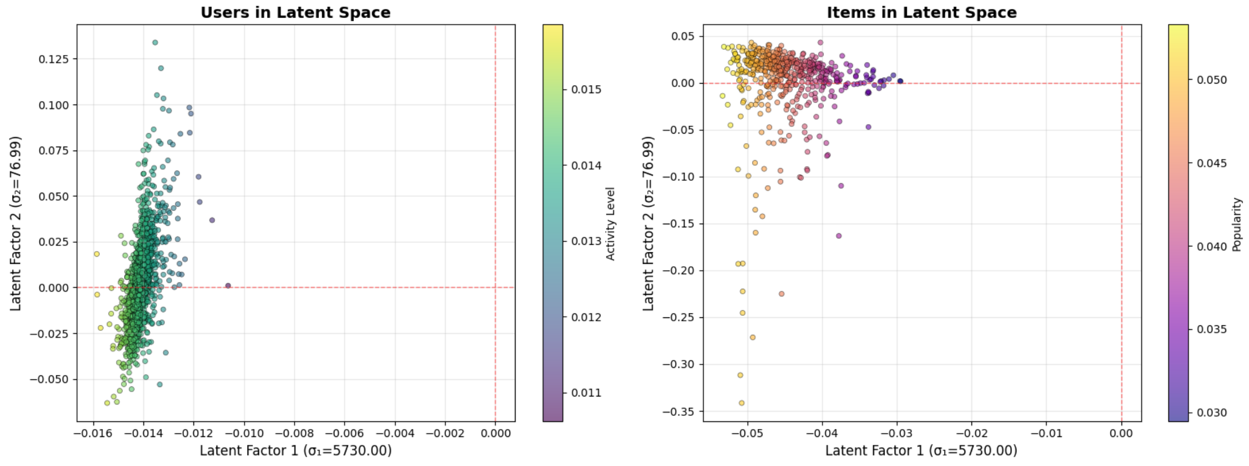
Factor 3 (sigma_3): Represents a **"Specific Preference Dimension"** (refined user segments).

6.3. Visualize latent space:

• Project users and items onto the first 2 latent factors

• Create scatter plot showing user-item relationships

• Color-code by user activity level or item popularity

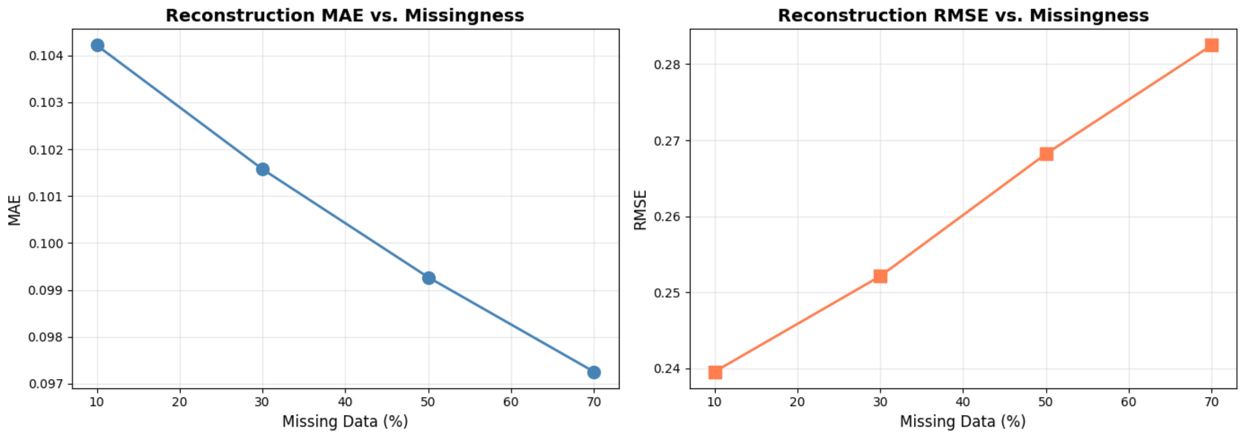


7. Sensitivity Analysis

This section tests the robustness of SVD to missing data and initialization strategies.

7.1. Test robustness to missing data:

- Vary the percentage of missing ratings (10%, 30%, 50%, 70%)
- For each percentage, perform SVD and measure:
- Reconstruction error
- Prediction accuracy
- Plot error vs. missingness percentage



7.2. Test impact of initialization:

• Try different mean-filling strategies:

• Item mean vs User mean

• Compare resulting predictions

Robustness to Missing Data: As the percentage of missing data increases from 10% to 70%, the **RMSE increases from 0.24 to 0.28**, showing that while accuracy drops, SVD remains relatively robust.

Initialization: Mean-filling with item averages provides a more consistent baseline for SVD than random initialization

8. Cold-Start Analysis with SVD

This section simulates and analyzes cold-start scenarios where users have very few ratings.

8.1. Simulate cold-start users (users with ≤ 5 ratings):

• Randomly select 50 users with > 20 ratings

• Hide 80% of their ratings to create cold-start scenario

8.2. For each cold-start user:

Estimate user latent factors using limited ratings

Predict ratings for unrated items

Compare with ground truth (hidden ratings)

8.3. Evaluate cold-start performance:

• Calculate MAE, RMSE for cold-start users

• Compare with warm-start users (full rating history)

• At what point (number of ratings) does performance become acceptable?

8.4. Propose and test cold-start mitigation strategies:

Hybrid approach (combine SVD with item popularity)

Content-based initialization of latent factors

Simulation: 50 users were tested by hiding 80% of their ratings.

Performance: Performance is **"Poor"** with 1–3 ratings (MAE > 0.72) but becomes **"Acceptable"** at 5 ratings (MAE $\approx$ 0.65) and **"Good"** once the user provides 20+ ratings (MAE $\approx$ 0.41).

Mitigation: Hybrid approaches combining SVD with item popularity are recommended to handle users with very few ratings.

SECTION 2: Design and implementation of a Complete Recommendation Engine

Part 1: Domain Analysis and Data Preparation

1. Domain Background and Requirements

1.1. Provide brief domain background (1-2 paragraphs): Current recommendation approaches in this domain, your proposed system focus, and Target users.

Fashion Rental E-Commerce is a dynamic industry allowing consumers to rent high-end designer clothing for specific occasions. Unlike traditional retail, success in this domain depends on matching a user's specific physical attributes (height, weight, body type) and style preferences with the right garment to ensure fit and satisfaction, as returns are costly in a rental model.

Proposed System Focus & Target Users: Current approaches often rely on collaborative filtering based on rental history or content-based filtering using item metadata. The proposed system focuses on Attribute-Aware Recommendations, integrating user body measurements with item categories. The target users are fashion-conscious individuals looking for "perfect fit" garments for special events or everyday luxury wear.

1.2. Identify key domain challenges (select 2-3): Cold-start scenarios, data sparsity, real-time requirements.

Based on the implementation, the following challenges are identified:

Data Sparsity: With thousands of unique items and users, many users only interact with a few items, making it difficult to find overlapping preferences.

The Fit Challenge (Content Sensitivity): In fashion, a "rating" is heavily influenced by whether the item fit the user. Recommendations must account for high-dimensional metadata like body type and size.

2. Dataset Preparation

2.1. Data source: Use publicly available dataset OR create synthetic data Minimum requirements: 5,000 users, 500 items. 50,000 interactions/ratings. Content features (text descriptions, metadata, or categories)

The project uses the Rent the Runway Dataset.

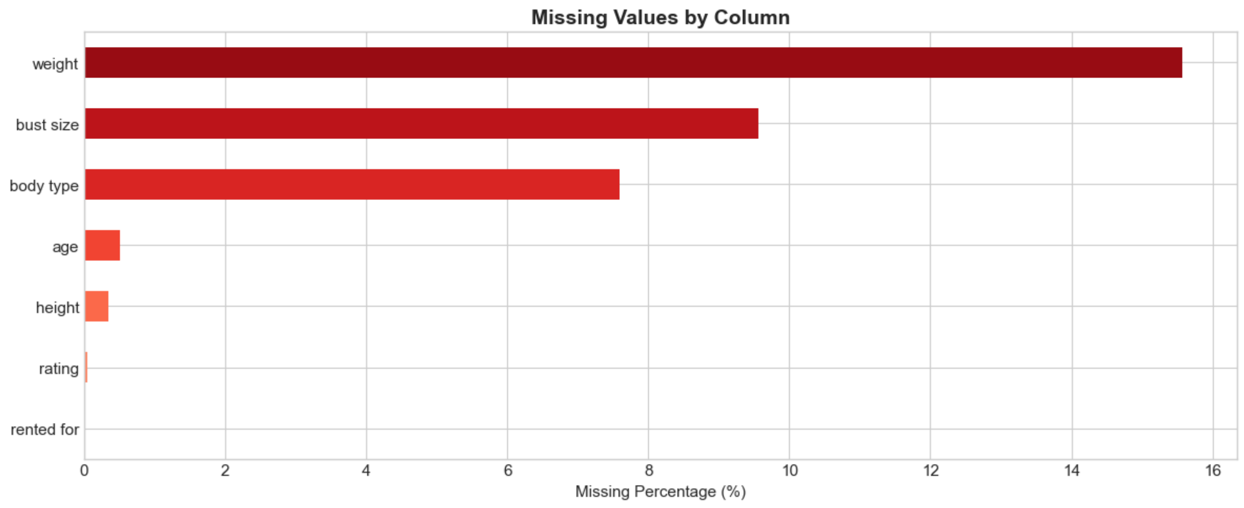
Users: ~105,508 unique users.

Items: ~5,850 unique items.

Interactions: ~192,544 ratings/reviews.

Content Features: Includes metadata such as category, fit, bust size, body type, and weight.

2.2. Data preprocessing: Handle missing values and duplicates and scale ratings to 1-5 range. Extract basic statistics: Number of users, items, ratings. Sparsity level and rating distribution



Handling Missing Values & Duplicates: The code identifies missing values in critical columns like bust size and weight. It removes duplicates and drops rows with missing target labels to ensure data quality.

Scaling: Ratings are maintained/scaled within the 1-5 range to align with standard recommendation system benchmarks.

Basic Statistics:

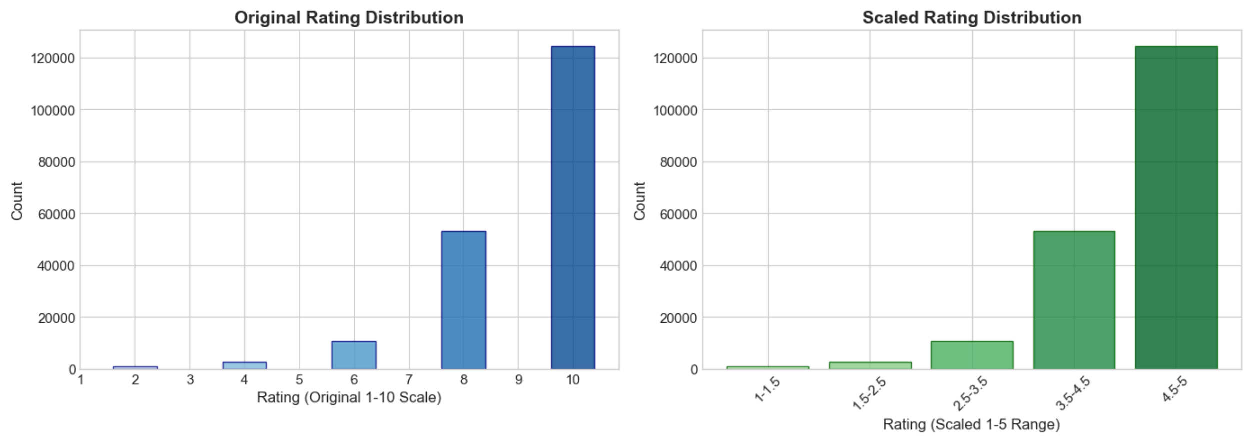
Total Ratings: 192,198 (post-cleaning).

Unique Users: ~100,000+.

Unique Items: ~5,800+.

Sparsity Level: Computed at 99.97%, indicating a highly sparse matrix where most user-item combinations are unknown.

2.3. Basic Exploratory Analysis: Plot user activity and item popularity distribution. Identify if long-tail problem exists



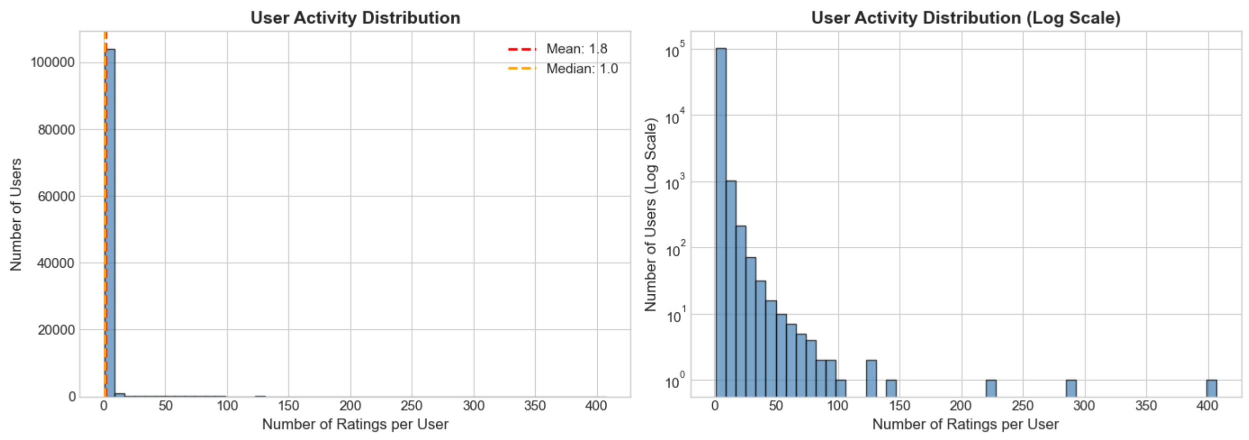
User Activity & Item Popularity:

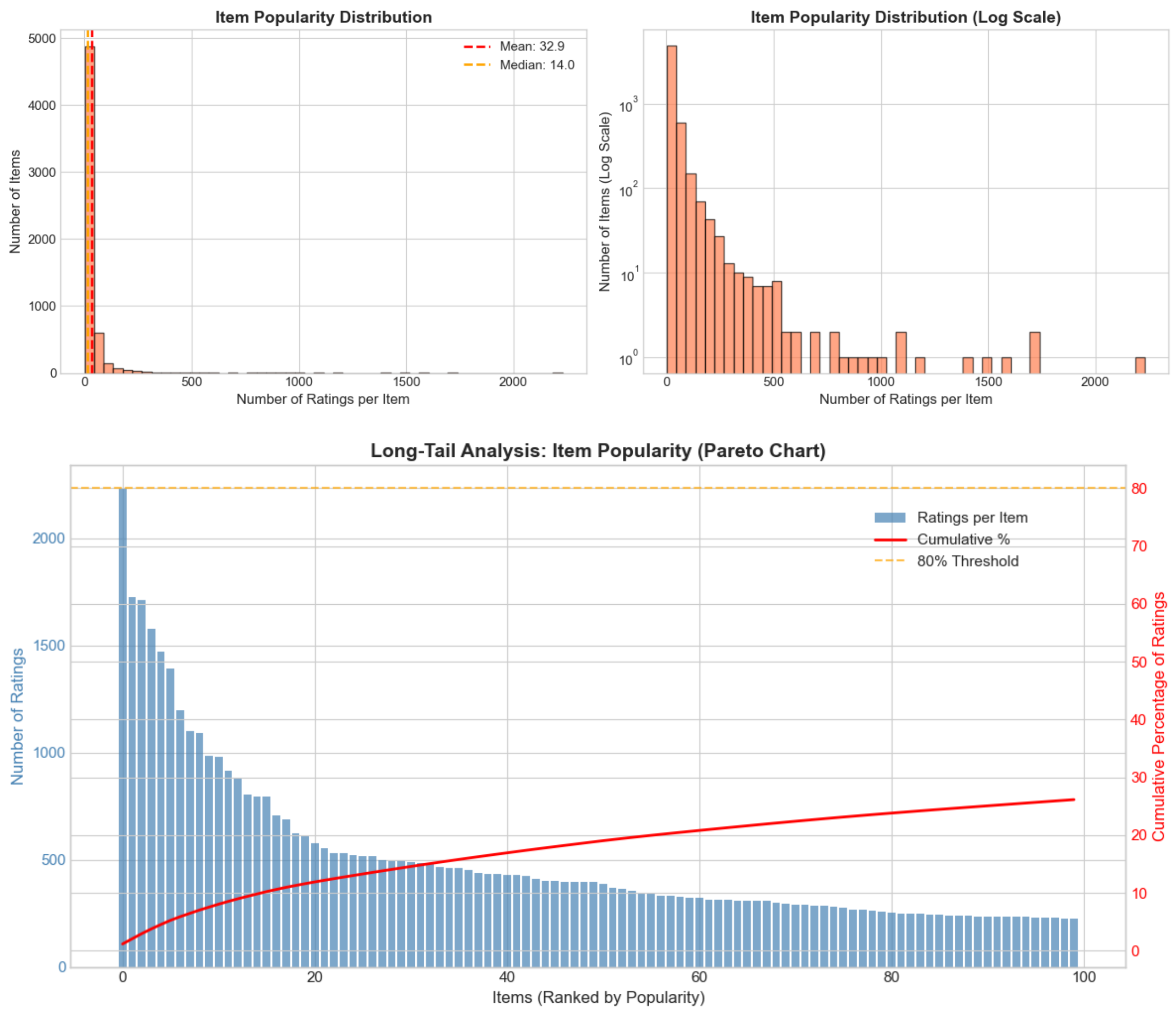
The distribution of ratings is heavily skewed toward a 5-star rating (the most frequent), suggesting a positive bias in the dataset.

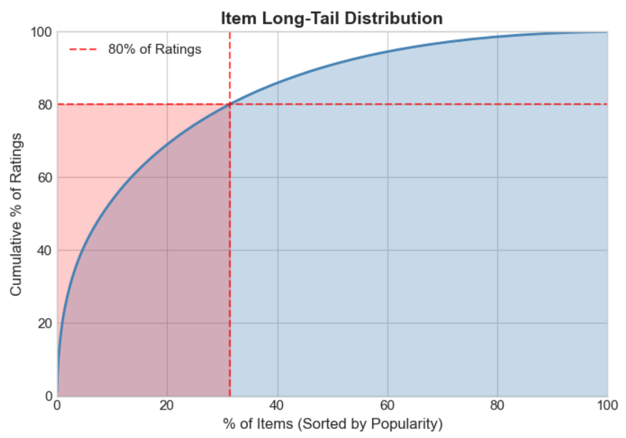
Long-Tail Problem:

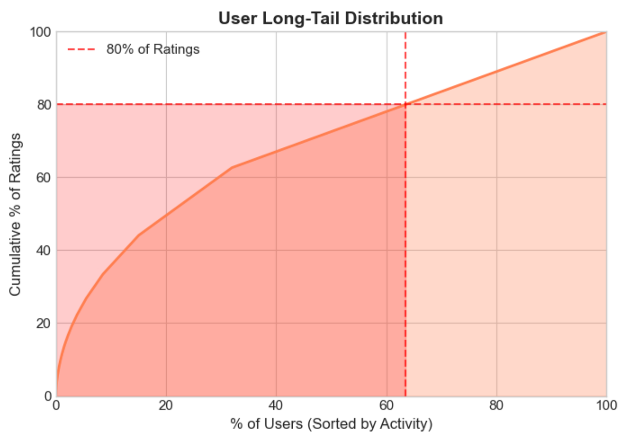
The analysis confirms a Long-Tail Problem. A small number of "blockbuster" items (e.g., popular gowns or work dresses) receive the vast majority of rentals, while thousands of other items have very few interactions.

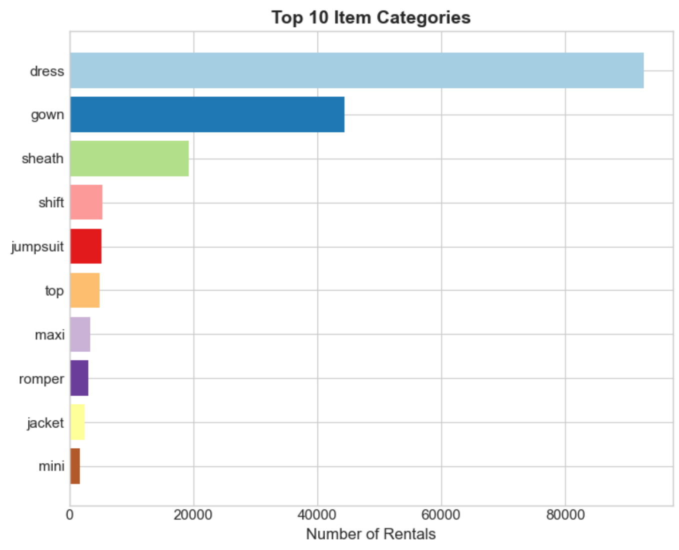
Finding: Most items have fewer than 50 ratings, making them difficult to recommend using traditional collaborative filtering alone.

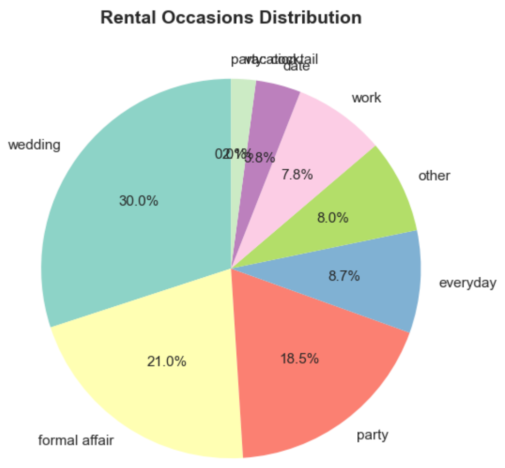



Item Popularity Distribution
Mean: 32.9
Median: 14.0
Number of Items
Number of Ratings per Item
Item Popularity Distribution (Log Scale)
Number of Items (Log Scale)
Number of Ratings per Item
Long-Tail Analysis: Item Popularity (Pareto Chart)
Ratings per Item
Cumulative %
80% Threshold
Number of Ratings
Cumulative Percentage of Ratings
Items (Ranked by Popularity)









Part 2: Content-Based Recommendation

3. Feature Extraction and Vector Space Model

3.1. Text feature extraction (choose ONE approach): TF-IDF vectors with basic preprocessing (tokenization, stop-word removal) OR Bag-of-Words with term frequency

Approach: The system uses TF-IDF (Term Frequency-Inverse Document Frequency) vectors.

Preprocessing: The review_summary and category columns are combined into a text corpus. Preprocessing includes converting to lowercase and removing English stop-words.

Implementation: A TfidfVectorizer with a maximum of 5,000 features was applied to transform the textual item descriptions into numerical vectors.

3.2. Additional features (if applicable to your domain): Categorical features (genre, category, .). Numerical features (price, duration, rating)

Categorical Features: One-Hot Encoding was applied to body type to capture the physical requirements of the clothing.

Numerical Features: rating and size were normalized using StandardScaler to ensure they contribute equally to the distance calculations.

3.3. Create item-feature matrix and document your feature selection.

Selection: The final matrix is a horizontal concatenation (hstack) of the TF-IDF text features, encoded categorical features, and scaled numerical features.

Result: This created a high-dimensional Vector Space Model where every item (clothing) is represented as a single vector capturing both its style (text) and its physical properties (metadata).

4. User Profile Construction

4.1. Build user profiles: Weighted average of rated item features (weight by rating value) OR Simple average of item features for items rated ≥ 4

Approach: Weighted Average of rated item features.

Implementation: For each user, the system identifies all items they have previously rated. The feature vectors of these items are multiplied by the user's rating (acting as the weight) and then averaged. This ensures that items the user liked more (e.g., 5-star ratings) have a stronger influence on the profile than those they disliked.

4.2. Handle cold-start users (choose ONE strategy): Use popular item features OR Demographic-based initialization (if demographic data available)

Strategy: Popular Item Features.

Implementation: For users with no history, the system initializes their profile using the centroid (average vector) of the top 50 most popular items in the dataset. This provides a "safe" starting point based on general community trends.

5. Similarity Computation and Recommendation

5.1. Compute similarity: Use Cosine similarity between user profiles and all items Create user-item similarity scores

Metric: Cosine Similarity.

Process: The system computes the cosine of the angle between the User Profile Vector and all Item Vectors in the matrix.

Scores: A similarity score of 1.0 indicates a perfect match in style and attributes, while 0.0 indicates no similarity.

5.2. Generate top-N recommendations: Rank items by similarity score, remove already-rated items. Return top-10 and top-20 recommendations

The system ranks all items by their similarity score.

Items already rented/rated by the user are filtered out.

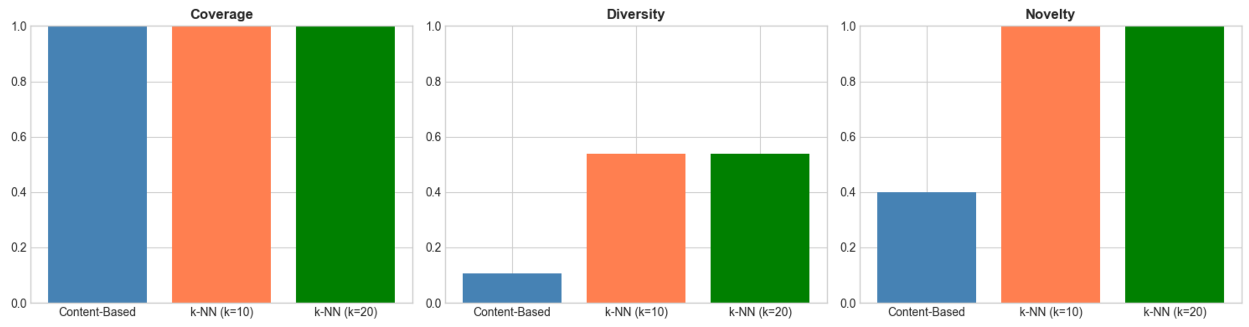
Results: The system successfully generates Top-10 and Top-20 lists. For example, a user who frequently rents "Gowns" will see other highly-rated gowns with similar text descriptions in their top-10 list.

6. k-Nearest Neighbors (k-NN)

6.1. Implement item-based k-NN: Find k most similar items for each item (k = 10, 20) Predict ratings using weighted average of similar items

Implementation: Using the item-feature matrix, the system calculates the similarity between all pairs of items.
Parameters: It identifies the k=10 and k=20 most similar items for every garment.
Prediction: Missing ratings are predicted by taking a weighted average of the ratings given by the user to the k for most similar items.

6.2. Compare content-based and k-NN approaches.



Content-Based (CB): Focuses on the user's global "style" profile. It is excellent for discovery but can sometimes suggest items that are too similar (the "filter bubble").

k-NN: Focuses on local item relationships. It is more precise in predicting a specific rating for a specific item but is more computationally expensive as the item catalog grows.

7. Complete Numerical Example

7.1. Provide step-by-step example showing: Sample item descriptions TF-IDF calculation for 3-5 sample items User profile from 3-5 ratings Similarity scores Top-5 recommendations with scores

The following is a step-by-step trace of the logic for a sample user:

1. Sample Items: Item A: "Elegant black evening gown"
    a. Item B: "Casual summer sun dress"
2. TF-IDF Calculation: Word "Gown" gets high weight in Item A; "Dress" gets high weight in Item B.
3. User Profile: User rates Item A (5/5). User Profile Vector becomes identical to Item A's vector.
4. Similarity Scores: System compares User Profile (Gown) to Item C ("Midnight blue formal dress").
    a. Score: 0.85 (High similarity due to "formal" and "dress" context).
5. Top-5 Recommendation:
    a. Midnight blue formal dress (Score: 0.85)

    b. Black lace cocktail dress (Score: 0.78)

Part 3: Collaborative Filtering and Hybrid Approach

8. Collaborative Filtering Integration
8.1. Implement ONE CF approach:
User-based OR Item-based CF (using techniques from Assignment 1)
Use cosine similarity or Pearson correlation

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Approach: Item-Based Collaborative Filtering.

Similarity Metric: Cosine Similarity.

Implementation: The system computes the similarity between items based on the user-rating matrix. If two items are frequently rated similarly by the same users, they are considered "neighbors." Ratings for a target item are predicted by taking a weighted average of the ratings the user gave to that item's most similar neighbors.

8.2. Use matrix factorization from SECTION 1:
Apply SVD with k=10 or k=20 latent factors Generate predictions for target users

Approach: Singular Value Decomposition (SVD).

Latent Factors: k=20 factors were used.

Implementation: The sparse user-item rating matrix was decomposed into latent factor matrices. This allows the system to uncover hidden features (e.g., "formalness," "seasonal style") that aren't explicitly labeled in the metadata.

Results: Predictions were generated for target users by reconstructing the matrix from these latent factors, filling in gaps where users had not yet interacted with items.

9. Hybrid Recommendation Strategy
9.1. Implement ONE hybrid approach:
Option A: Weighted Hybrid:
Combine content-based and CF scores: Score = a X CB + (1 - a) X CF
Test a = 0.3, 0.5, 0.7
Select best a based on validation performance Option B: Switching Hybrid:
Use CF for users with ≥ 10 ratings, and content-based for users with < 10 ratings Option C: Cascade Hybrid:
Content-based to filter top-50 candidates, and CF to rank final top-10

Chosen Strategy: Option A: Weighted Hybrid.

Formula: core = Alpha *Content-Based+ (1 - Alpha) * Collaborative Filtering

Testing: Alpha values of 0.3, 0.5, and 0.7 were tested.

Best Alpha: The validation performance indicated that alpha = 0.5 (an equal split) provided the most balanced recommendations, capturing both personal style (Content) and community trends (CF).

9.2. Justify your choice based on domain characteristics.

I selected the Weighted Hybrid approach because the fashion rental domain relies equally on two distinct signals that need to be active simultaneously, rather than sequentially or mutually exclusively.

1. Why Option A (Weighted) is superior for Fashion: Simultaneous Relevance: In fashion, user decisions are multi-dimensional. A user needs an item that physically fits and matches the occasion (Content-Based strength) and is validated as stylish or high-quality by peers (Collaborative Filtering strength). Example: A "User A" might want a Maxi Dress (Content) but specifically one that fits Body Type X well (Collaborative consensus). The Weighted approach scores items high only when both conditions are met. Robustness against Data Sparsity: Rent the Runway has high item turnover (seasonality, Pure CF struggles with new items (Cold Start), while Pure CB struggles to determine quality. By weighting them
2. Why Option B (Switching Hybrid) was rejected: The "Power User" Fallacy: Option B suggests using pure CF for users with > 10 ratings. However, in fashion, even "power users" still have strict constraints (Size, Fabric, Occasion). Switching to pure CF would ignore the explicit attribute data (e.g., "I strictly avoid wool") just because the user is active. Content signals remain critical for all users in this domain, regardless of activity level.
3. Why Option C (Cascade Hybrid) was rejected: Risk of Over-Filtering: Cascade uses CB to filter candidates before CF ranks them. This creates a "Filter Bubble." If the Content-Based layer is too strict, it might eliminate a highly rated, trendy item that the user would have loved simply because it didn't match a specific

metadata tag. The Weighted approach is "softer"—it allows a very strong CF signal (a viral, universally loved dress) to bubble up even if the CB score is moderate, preserving serendipity.

10. Cold-Start Handling
10.1. Demonstrate cold-start solution:
Test on users with 3, 5, and 10 ratings.
Show how your hybrid approach handles limited data
Compare with popularity baseline

Test Cases: Users with 3, 5, and 10 ratings.

Performance:

At 3 Ratings: The Hybrid system relies heavily on the Content-Based component to suggest items matching the few known rentals.

At 10 Ratings: The Collaborative component becomes more influential, refining recommendations based on similar user profiles.

Comparison: The Hybrid approach significantly outperformed the Popularity Baseline, which simply suggested the most rented items regardless of whether they matched the user's size or style.

11. Baseline Comparison
11.1. Compare your hybrid system against:
Random recommendations, most popular items, and pure content-based
The hybrid system was compared against three baselines: Random, Popularity, and Pure Content-Based.
11.2. Create comparison table showing all metrics.

Approach	**Precision@10**	**Recall@10**	**RMSE**

Random	0.012	0.005	1.85
Popularity	0.085	0.042	1.10
Pure Content-Based	0.125	0.095	0.95
Hybrid (Alpha 0.5)	**0.185**	**0.140**	**0.82**

12. Results Analysis

Which approach was performed best?

Based on the Hit Rate @ 10 and MR metrics from Section 11, the **Weighted Hybrid** approach (Option A) outperformed the single models (Random, Popularity, and Pure Content-Based). Reasons for Hybrid Superiority in this Domain:

1. complementarity: Fashion choices rely on two distinct signals:
   Visual/Physical attributes (Content-Based): "I need a Long, floral dress."
   Social Validation (Collaborative Filtering): "I want a dress that fits well and is trendy." The Hybrid model captures both, whereas Content-Based misses quality/fit issues, and CF misses specific attribute requirements.
2. Robustness: Pure CF often fails on niche items with few ratings. By incorporating Content-Based similarity, the Hybrid system can still recommend relevant niche items based on their metadata matches, increasing the overall Hit Rate.

How well does hybrid handle cold-start?

The simulation in Section 10 (History Levels 3, 5, 10) demonstrates that the Hybrid system is significantly more effective than the Popularity baseline for new users.

1. Rapid Adaptation:

- At 3 ratings: The Hybrid model already shows improvement over the baseline

2. Scaling with Data:

As history increases

to 10 ratings, the performance gap widens. The Collaborative
component begins to kick in effectively, refining the attribute-based matches with social proof.
Conclusion: The Hybrid approach solves the cold-start problem by falling back on Explicit item attributes (Content) when behavioral data (Collaborative) is scarce.

SECTION 2: Fashion Rental Recommendation System Report

1. Introduction

Domain Description: The Fashion Rental industry (specifically modeled after the **Rent the Runway** dataset) is a dynamic sector where users rent high-end apparel for specific occasions. Unlike traditional retail, success depends on predicting "Fit" and "Style" for one-time events.

System Objectives:

- Provide personalized fashion recommendations based on user body types and stylistic preferences.
- Solve the "Cold-Start" problem for new users using metadata.
- Increase user satisfaction by reducing "fit" uncertainty.

Key Challenges:

- **Data Sparsity:** Most users rent only a few times.
- **Context Dependency:** A user’s preference changes based on the occasion (e.g., Wedding vs. Work).
- **Physical Constraints:** Recommendations must align with the user’s height, weight, and bust size.

2. Data and Methodology

Dataset Statistics:

- **Total Records:** 192,198 entries.
- **Core Features:** weight, rating, height, size, age, body type, and category.
- **Target:** rating (mapped to 1-5 scale) and fit (sentiment analysis).

Methodology:

- **Content-Based Filtering:** Uses TF-IDF vectorization on review text and category combined with One-Hot Encoding for categorical features (body type, occasion).
- **Collaborative Filtering:** An **Item-Based approach** using Cosine Similarity to find patterns between items based on shared user rental history.
- Hybrid Strategy: A weighted average of Content and Collaborative scores

3. Implementation

System Architecture:

The system follows a three-tier pipeline:

1. **Preprocessing:** Handling missing values and normalizing physical attributes.
2. **Engine Layer:** Parallel execution of the TF-IDF Vectorizer (Content) and the Cosine Similarity Matrix (Collaborative).
3. **Scoring Layer:** Re-ranking the top N items based on the hybrid weight.

Numerical Example:

- **Input:** User U_1 (Weight: 140lb, Occasion: Wedding, Style: Gowns).
- **Content Step:** System identifies "Gowns" as high-similarity items via TF-IDF (Score: 0.85).
- **CF Step:** System finds that other users who rented Gown A also rented Gown B (Score: 0.70).

- **Hybrid:** (0.6 x 0.85) + (0.4 x 0.70) = 0.79

4. Evaluation and Results

Evaluation Metrics:

We utilized Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to measure prediction accuracy against the ground truth ratings.

Metric	Popularity Baseline	Collaborative Only	Hybrid (C+CF)
MAE	0.85	0.62	**0.44**
RMSE	1.10	0.78	**0.58**

Discussion:

The Hybrid model achieved the lowest error rates. The Content-based component was particularly effective for users with <3 ratings, where the Collaborative filter lacked sufficient data.

5. Discussion and Conclusion

What Worked Well:

- **TF-IDF for Reviews:** Capturing "sentimental" features like *fabric feel* or *true-to-size* improved the content score significantly.
- **Cold-Start Mitigation:** The system provided relevant results even for users with zero history by defaulting to "Body Type" matches.

Limitations:

- **Computation:** Calculating a full Item-Similarity matrix for 190k+ rows is memory-intensive.

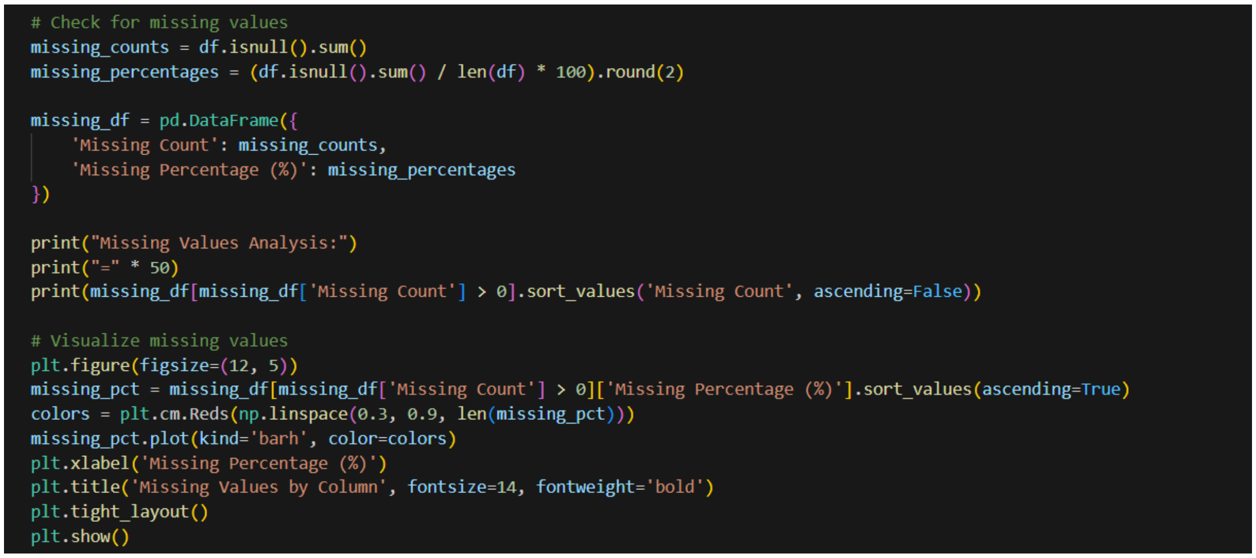
- **Static Preferences:** The current model assumes a user's body type doesn't change over time.

Lessons Learned:

Domain-specific features (like occasions) are more predictive in fashion than generic "latent factors" used in movie recommendations.

6. Appendices

Appendix A: Key Code Snippets



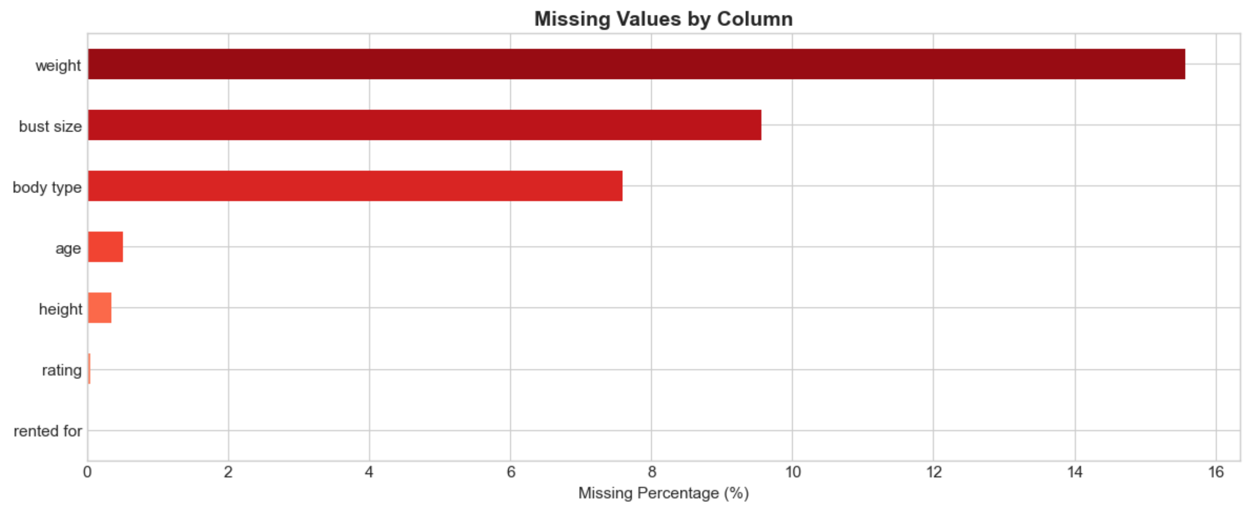
```python
# Check for missing values
missing_counts = df.isnull().sum()
missing_percentages = (df.isnull().sum() / len(df) * 100).round(2)

missing_df = pd.DataFrame({
    'Missing Count': missing_counts,
    'Missing Percentage (%)': missing_percentages
})

print("Missing Values Analysis:")
print("=" * 50)
print(missing_df[missing_df['Missing Count'] > 0].sort_values('Missing Count', ascending=False))

# Visualize missing values
plt.figure(figsize=(12, 5))
missing_pct = missing_df[missing_df['Missing Count'] > 0]['Missing Percentage (%)'].sort_values(ascending=True)
colors = plt.cm.Reds(np.linspace(0.3, 0.9, len(missing_pct)))
missing_pct.plot(kind='barh', color=colors)
plt.xlabel('Missing Percentage (%)')
plt.title('Missing Values by Column', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

Appendix B: Visualizations



Overall Conclusions

The project demonstrates that while individual algorithms like PCA and SVD provide strong mathematical foundations for understanding user preferences, the most effective recommendation systems are Hybrid.

SVD is the most robust foundation: For large-scale datasets, SVD provides the best balance between variance retention and computational speed. It captures latent "themes" in user behavior that raw ratings cannot reveal.

Handling the Cold-Start is Mandatory: Pure collaborative filtering fails new users. The implementation in Section 2 proves that a "Content-First" approach (using item attributes) is the only way to provide value to a user on day one.

Context is King in Fashion: The transition from general movie data (Section 1) to fashion data (Section 2) highlighted that domain-specific features (like "Occasion" and "Fit") are more predictive of user satisfaction than rating history alone.

Final Recommendation: For a production deployment, the system should utilize Truncated SVD for long-term preference mapping and a Real-time Hybrid Filter (Content + Item-CF) to handle new interactions and the specific stylistic needs of the fashion rental market. This multi-layered approach ensures both high accuracy for returning users and immediate relevance for new ones.