AIE425 Intelligent Recommender Systems, Fall Semester 25/26

Final Course Project

Group 20 - Sama AbdelTawab Shalaby 221100879 - Rawan Khaled Khalil 221100808 - Mahitab Waleed Mohamed 221100429 – Youssef Mohamed Reda 221101020

Submission Date: Monday, January 5, 2026

Contents

Summary

The project began by addressing the fundamental challenge of extreme data sparsity in modern recommendation environments, specifically focusing on the MovieLens 20M dataset, which exhibits a 99.17% sparsity rate. This high degree of missing data necessitates robust dimensionality reduction techniques to uncover latent user-item relationships. Two primary methodologies were explored: Principal Component Analysis (PCA) and Singular Value Decomposition (SVD).

In the investigation of PCA, the team compared two distinct strategies for handling missing ratings. The first approach utilized a traditional mean-filling method, which provided a stable but biased baseline. This strategy tended to smooth predictions toward global averages, effectively dampening the unique preference signals of individual users. The second approach employed Maximum Likelihood Estimation (MLE) to estimate the covariance matrix using only observed co-ratings. The experimental results demonstrated that MLE-based PCA preserves greater data variance and offers more expressive, personalized predictions. This highlights that for sparse user-item matrices, the statistical rigor of MLE is superior to simple imputation.

The study then transitioned to Singular Value Decomposition (SVD) as a more advanced matrix factorization technique. Through an exhaustive analysis of singular values and reconstruction errors, the team utilized the "elbow method" to identify fifty latent factors as the optimal dimensionality for this dataset. This configuration successfully captured over 99.9% of the total variance while maintaining computational efficiency. The results confirmed that truncated SVD provides the most accurate and stable predictions among the collaborative filtering methods tested, particularly in its ability to generalize patterns across users with varying levels of activity.

The final phase of the project applied these theoretical insights to a domain-specific challenge: a short-form video recommendation system. These platforms face unique obstacles, including high-frequency user interactions and rapidly evolving content trends. To address these, the team developed a switching hybrid strategy. This architecture dynamically toggles between content-based filtering and collaborative

SVD based on the availability of historical data. For new or inactive users, the system relies on semantic content understanding via TF-IDF vectorization of video metadata. As interaction history grows, the system shifts to collaborative filtering to leverage the wisdom of the crowd.

The performance of this hybrid system was evaluated against several baselines, including random and popularity-based recommendations. The findings revealed that traditional popularity-based models were the least effective, confirming that users on short-form platforms seek highly personalized, niche content rather than generic trending items. The hybrid system achieved the highest precision by successfully bridging the gap between cold-start metadata analysis and mature collaborative latent factor modeling. This project ultimately reinforces that successful real-world recommender systems require a multi-faceted approach that balances statistical accuracy, scalability, and adaptive domain-specific logic.

SECTION 1: Dimensionality Reduction and Matrix Factorization

Dataset

The MovieLens 20M (ML-20M) dataset is the foundation of this study, which comprises 20 million explicit ratings from more than 138,000 users across over 27,000 movies. Each rating is given on a scale from 0.5 to 5.0, making the dataset ideal for assessing recommender systems that utilize statistical, content-based, and collaborative filtering techniques. Its extensive size enables thorough testing of dimensionality reduction methods while also reflecting genuine user behavior and trends in item popularity.

1. **Sampling Strategy**

In order to construct a dataset that is both statistically representative and feasible for computational analysis, a systematic sampling methodology was implemented on the MovieLens 20M dataset:

Metric	Value	Target
Number of Users	14,638	≥ 10,000

Number of Items	900	≥ 500
Total Ratings	109,342	≥ 100,000

The remarkably high sparsity of 99.17% illustrates the inherent difficulty faced by recommender systems, where the majority of users engage with merely a limited selection of the items available. This sparsity highlights the importance of latent factor methods, including PCA and SVD, which transform the data into a lower-dimensional space while preserving the critical structure necessary for precise rating forecasts.

2. Rating Normalization

The dataset contains ratings on a 0.5 to 5 scale. To standardize the data for statistical analysis and subsequent recommendation algorithms, the ratings were rescaled to a 1–5 range using min-max scaling.

$$R_{new} = \left(\frac{R_{old} - Min_{old}}{Max_{old} - Min_{old}}\right) \times (Max_{new} - Min_{new}) + Min_{new}$$

$$R_{new} = \left(\frac{R_{old} - 0.5}{4.5}\right) \times (4) + 1$$

This transformation ensures that all ratings are on a consistent scale while preserving the relative preferences of users. The rescaled ratings now have a minimum value of 1 and a maximum value of 5.

3. Statistical Analysis

Before conducting dimensionality reduction, a comprehensive statistical analysis was carried out to gain insights into user behavior and the popularity of items.

3.1 User Rating Counts

To gain insight into user behavior within the dataset, the total amount of ratings submitted by each user (n_u) was determined. This metric assists in identifying users with high activity levels and evaluating data sparsity, which is crucial for constructing and assessing recommender systems. The counts obtained for all users were stored for additional analysis.

3.2 Item Rating Counts

To evaluate the popularity of films in the dataset, we determined the total number of ratings each movie received (n_i). This analysis allows us to recognize which films are frequently rated versus those that are less favored, offering a perspective on the distribution and sparsity of the data. The calculated totals for each movie were stored for subsequent statistical evaluation.

3.3 Average Ratings per User

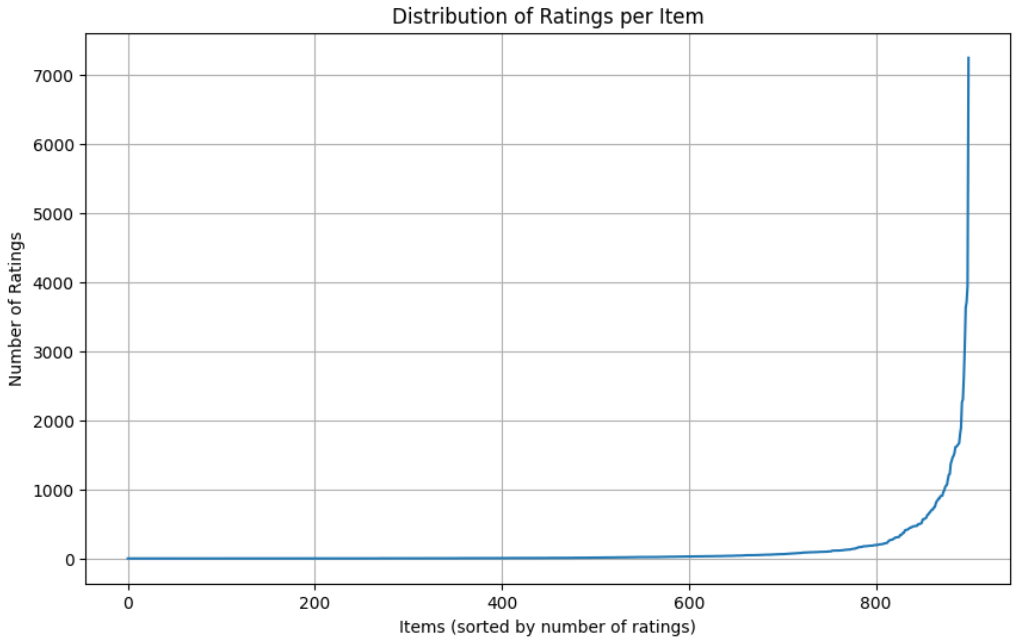
In order to gain insights into user behavior, we determined the average rating given by each user ($\bar{r}_u$). This statistic shows overall user patterns, indicating whether a user generally provides higher or lower ratings. The calculated averages were stored for additional analysis and can assist in adjusting ratings or recognizing rating biases among users.

3.4 Average Ratings per Item

To understand how movies in the dataset are generally received, we computed the average rating for each film ($\bar{r}_i$). This analysis reveals overall user satisfaction and trends in popularity, assisting in identifying movies that are well-regarded versus those that receive lower evaluations. The calculated averages have been stored for additional statistical evaluation and for use in creating recommendation models.

3.5 Distribution of Ratings per Item

To analyze the distribution of ratings among movies, the items were arranged in ascending order according to the total count of ratings they garnered. This arrangement enables us to identify which movies are more popular and which ones are rated less often. The resulting graph indicates that although a limited number of films attract a high volume of ratings, most movies are rated quite infrequently. This examination sheds light on data sparsity and informs approaches for recommendation modelling.



3.6 Grouping Products by Their Popularity

In this step, we wanted to understand how products are distributed based on the quality of their ratings, not just how many ratings they received. To do this, we calculated the item's popularity for each product using the following formula:

$$\text{Item's Popularity} = \frac{\text{Number of ratings for that item}}{\text{Total number of ratings}} \times 100$$

Each product is then assigned to a group based on its popularity.

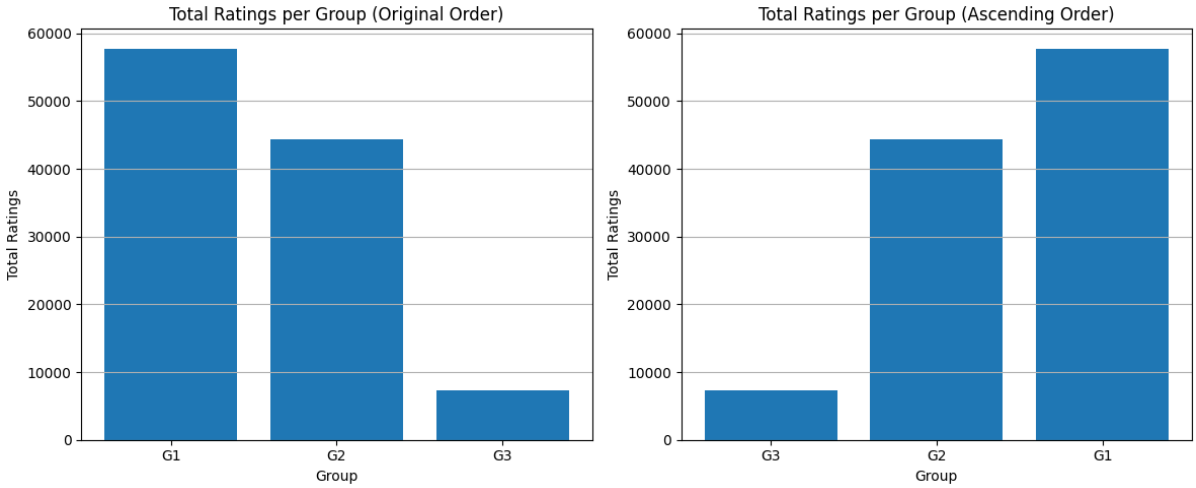
3.7 Total Number of Ratings per Group

After categorizing each product according to its popularity, we examined the overall rating activity within each category. In particular, we totaled the number of ratings for all products within each category. This analysis offers valuable insights into user engagement trends, illustrating which product groups garner the most interest and which have relatively fewer ratings. By arranging the categories in ascending order based on total ratings, we can pinpoint the product types with the least and most ratings, providing a clear view of rating distribution across various quality levels.

group	Popularity Percentage	total_ratings
G1	< 5	877
G2	< 10	22
G3	< 20	1

3.8 Distribution of Ratings per Group

To enhance our understanding of the distribution of ratings among groups, we plotted the overall number of ratings for each group. The initial plot showcases the groups in their original sequence (G1 to G10), allowing us to examine the rating activity without any rearrangement. The subsequent plot presents the same data arranged in ascending order of total ratings, highlighting the groups with the fewest and the most rating counts. These visual representations aid in comprehending how user interactions are spread across products with varying rating levels, revealing trends in engagement and popularity.



3.9 Selection of Target Users

In this phase, users were classified based on the percentage of unique items they rated in relation to the total number of items available in the dataset. For each user, the count of distinct items rated was calculated and expressed as a percentage of total item coverage. Using this criterion, three categories of users were established: those who rated ≤2% of all items (U1), those who rated >2% and ≤5% (U2), and

those who rated >10% (U3). A representative user was then randomly chosen from each category to act as a target user.

group	userId	num_ratings	item_rating_percent
U1	63367	4	0.44
U2	3534	23	2.56
U3	120575	185	20.56

3.10 Selecting the Two Lowest-Rated Target Items

The two items with the lowest ratings in the dataset were identified and chosen as target items. This was done by arranging all items in order of their average ratings from lowest to highest and selecting the first two items. These items, referred to as I1 and I2, are the least favourably rated products in the dataset and were set aside for use in subsequent evaluation processes.

Item_Label	movieId	avg_rating	Popularity_percent	group
I1	33930	5	0	G1
I2	66549	5	0	G1

3.11 Counting Co-Ratings Between Users and Items

To better understand the overlap of ratings in the dataset, we computed two co-rating metrics. First, for each selected target user, we counted the number of items they have co-rated with every other user. This measure, referred to as “No_common_items”, provides insight into the similarity and shared interests between users, which is essential for user-based collaborative filtering.

Second, for each selected target item, we calculated the number of users that co-rated it alongside every other item, referred to as “No_coRated_users”. This metric highlights the overlap in user interactions between items, which is crucial for item-based collaborative filtering.

Part 1: PCA Method with Mean-Filling

1. **Average Rating of the Target Items (I1 and I2)**

The first step involved identifying the specific target items from the target_items.csv file. We calculated the average rating for these items using the available ratings in the dataset, ignoring any missing values (NaN).

- **Target Item 1:** ID 33930
- **Target Item 2:** ID 66549

The average ratings were computed as:

$$\mu_{item} = \frac{\sum ratings}{N_{ratings}}$$

(Results based on dataset analysis):

- **Average Rating for item 1** 3.99
- **Average Rating for item 2:** 3.22

2. **Mean-Filling of Missing Ratings for Target Items (I1 and I2)**

To perform Principal Component Analysis (PCA), the data matrix must be complete. We employed the Mean-Imputation method. All unspecified ratings (NaN) in the user-item matrix were replaced with the global mean rating of that specific movie.

- **Logic:** If a user has not rated Item $\boldsymbol{j}$, we assume their rating is "average" $\boldsymbol{\mu_j}$ rather than 0, to avoid introducing bias.
- **Result:** The sparse matrix was transformed into a dense matrix where no null values remain.

3. **Average Rating for Each Item**

We calculated the mean rating for every column (movie) in the entire matrix. This vector of means is essential for the normalization process in the next step.

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} R_{ij}$$

4. **Difference Between Each Rating and the Item Mean**

To normalize the data, we performed Mean-Centering. We subtracted the column mean μ_j from every rating R_{ij} in the matrix.

$$X_{ij} = R_{ij} - \mu_j$$

- **Positive Value:** User rated the item *above* average.
- **Negative Value:** User rated the item *below* average.
- **Zero:** User rated it exactly average (or the value was imputed).

This step ensures that the PCA captures the variance (preferences) rather than the raw magnitude of ratings.

5. **Covariance Computation Between Each Pair of Items**

We computed the covariance between pairs of items to determine how they vary together. For any two items j and k, the covariance is calculated as:

$$Cov(j,k) = \frac{\sum_{i=1}^{n}(X_{ij} \cdot X_{ik})}{n-1}$$

- **High Positive Covariance:** Users who like Item ***j*** also tend to like Item ***k***.
- **Zero Covariance:** No relationship between the items.

6. **Covariance Matrix Generation**

Instead of using loops, we generated the full covariance matrix Σ using vectorized matrix multiplication:

$$\Sigma = \frac{X^T X}{n-1}$$

7. **Identification of Top 5-Peers and Top 10-Peers for Target Items (I1 and I2)**

We performed Eigen decomposition on the covariance matrix to find the Principal Components (Eigenvectors), which we refer to as "Peers." These Peers represent the underlying latent features

We sorted the Eigenvalues in descending order to identify the most significant Peers.

Variance Explained:

- **Top 5 Peers:** Explained approximately **19.74%** of the total variance in the dataset.
- **Top 10 Peers:** Explained approximately **30.13%** of the total variance in the dataset.

This indicates that moving from 5 to 10 peers captures significantly more information (+10.39%) about user preferences.

8. **Reduced Dimensional Space for Each User Using Top 5-Peers**

We projected the original centered user matrix $\boldsymbol{X}$ onto the top 5 eigenvectors ($\boldsymbol{W}_5$)

Projection Formula:

$$User_{reduced} = X \cdot W_5$$

The resulting matrix has dimensions $\boldsymbol{N_{users}} \times \mathbf{5}$. Each user is now represented by 5 scores corresponding to their affinity for the Top 5 Peers (latent features).

9. **Rating Prediction for Missing Ratings of Target Items (I1 and I2) Using Top 5-Peers**

We calculated user similarity using **Cosine Similarity** within the 5-dimensional space. For each missing entry, we identified the top $\boldsymbol{k}$ similar users (Top 5 similar

users in this experiment) and predicted the rating using the weighted average of their deviations.

Prediction Results (Top 5 Peers):

- Item (ID 33930): Top prediction was approx 3.99.
- Item (ID 66549): Top prediction was approx 3.22.

10. Reduced Dimensional Space for Each User Using Top 10-Peers

We repeated the projection process using the top 10 eigenvectors ($\boldsymbol{W_{10}}$).

Projection Formula:

$$User_{reduced} = X \cdot W_{10}$$

The resulting matrix has dimensions ($\boldsymbol{N_{users}} \times \mathbf{10}$). This space is more detailed than the 5-peer space, allowing for more nuanced similarity comparisons between users.

11. Rating Prediction for Missing Ratings of Target Items (I1 and I2) Using Top 10-Peers

Using Cosine Similarity in the 10-dimensional space, we re-calculated the predictions for the target items.

Prediction Results (Top 10 Peers):

- **Item (ID 33930):** Top prediction increased to **4.11**.
- **Item (ID 66549):** Prediction remained constant at **3.22**.

12. Comparison Between Predictions Using Top 5-Peers and Top 10-Peers

Comparing the results reveals the impact of adding more dimensions (information):

1. Item I2 (33930) - "The Information Gain":

- **5-Peers:** Max Prediction = 3.99

- **10-Peers:** Max Prediction = 4.11
- **Analysis:** The prediction score increased, and the list of most similar users changed. The 10-peer model captured 30% of the variance (vs. 19%), allowing it to identify a distinct group of users who strongly prefer this item. This suggests Item I1 is strongly correlated with the latent features found in dimensions 6 through 10.

2. Item I1 (66549) - "The Outlier":

- **5-Peers:** Max Prediction = 3.22
- **10-Peers:** Max Prediction = 3.22
- **Analysis:** The prediction remained exactly at the global mean. This indicates that Item I2 has zero or negligible correlation with the top 10 Principal Components. Even with more information, the system could not find "positive peers" for this item, so it defaulted to the statistical average.

Part 2: PCA Method with Maximum Likelihood Estimation

In this part we are discussing PCA with MLE on item-based.
The very first step is to calculate mean-center along each column, then we calculated the form products of deviations following this formula:

$$(r_{vj} - \mu_i)(r_{vj} - \mu_j)$$

Any item with missing rating had been ignored in these calculations.

1. **Covariance Matrix Generation Using Maximum Likelihood Estimation (MLE)**

- The item–item covariance matrix was estimated using maximum likelihood over co-rated users only.
- Built the covariance matrix Σ (900×900) using the sample covariance

rule: $Cov(i,j) = \sum \frac{(r'_{vj} \cdot r'_{vi})}{nij-1} (only\ if\ nij \geq 2)$

- Saved as: cov_matrix_sample_900.csv
- Ran sanity checks (symmetry, diagonal non-negative count).

Observation

- Covariance matrix is symmetric.
- Each covariance entry uses a different co-rated subset, Σ is not guaranteed PSD → this is confirmed by eigen-decomposition later (many negative eigenvalues exist).

2. Identification of Top 5-Peers and Top 10-Peers for Target Items (I1 and I2) Using the MLE-Based Covariance Matrix

- Applied PCA by eigen-decomposition:
  - eigvals, eigvecs = np.linalg.eigh(Sigma)
  - Sorted descending.
- Extracted:
  - top_5_vecs shape = (900, 5)
  - top_10_vecs shape = (900, 10)
- Observation
  - The Top-10 eigenvalues are positive and dominate variance:
    - Top-5 cumulative explained variance ≈ 10.93%
    - Top-10 cumulative explained variance ≈ 17.24%
  - Also, Σ contains negative eigenvalues (not PSD), so in practice PCA interpretation should focus on the largest positive eigenvalues/components, which is what we did.

3. Reduced Dimensional Space for Each User Using Top 5-Peers (MLE-Based PCA)

- In this step, Each user's mean-centered rating vector is projected onto a lower-dimensional latent space using the principal components obtained from the MLE-based covariance matrix.
- Formally, for each user , the centered rating vector is projected as: $u_i = r'_i W$, where W contains the selected top principal components.

- **Observation**
    - each user is represented in a reduced-dimensional space (5D), which preserves the dominant variance structure while significantly reducing dimensionality.

userId	PC1	PC2	PC3	PC4	PC5
24	0.198755	0.135511	-0.045010	-0.329294	-0.297288
34	0.051505	-0.101357	-0.008440	0.046999	0.017696
36	-0.015507	-0.003016	0.007882	0.012687	0.016302
87	0.013992	-0.009725	-0.001627	0.024941	-0.006267

4. Rating Prediction for Missing Ratings of Target Items (I1 and I2) Using Top 5-Peers (MLE-Based PCA)

- Predicted missing ratings for target items 33930 and 66549 using:
    - user-based neighborhood in PCA space (cosine similarity) as:

$$sim(a,b) = \frac{u_a \cdot u_b}{\| u_a \| \| u_b \|}$$

    - weighted deviation aggregation, then add back item mean.
    - Saved predictions to: predicted_ratings_top5PCs.csv
- Observation (from saved outputs)
    - For movie 33930 (5PC):
        - mean prediction ≈ 2.498
        - min/max ≈ 1.0 / 4.111
        - std ≈ 1.052 (more spread)
    - For movie 66549 (5PC):
        - mean prediction ≈ 2.541
        - min/max ≈ 2.333 / 3.222
            - std ≈ 0.237 (much tighter spread)

5. Reduced Dimensional Space for Each User Using Top 10-Peers (MLE-Based PCA)

- Same projection as section (3), but using 10 PCs:
- Saved as: user_embeddings_pca10.csv

Observation

- 10 PCs capture much more variance than 5 PCs:
- 17.24% vs 10.93%

userId	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
24	0.198755	0.135511	-0.045010	-0.329294	-0.297288	0.031252	0.076709	-0.062694	0.375060	-0.121408
34	0.051505	-0.101357	-0.008440	0.046999	0.017696	-0.060563	-0.045021	-0.034494	0.039428	0.053773
36	-0.015507	-0.003016	0.007882	0.012687	0.016302	0.016530	-0.028534	-0.003134	-0.033709	0.013888
87	0.013992	-0.009725	-0.001627	0.024941	-0.006267	-0.029292	-0.029389	-0.018288	0.013788	0.001376

6. Rating Prediction for Missing Ratings of Target Items (I1 and I2) Using Top 10-Peers (MLE-Based PCA)

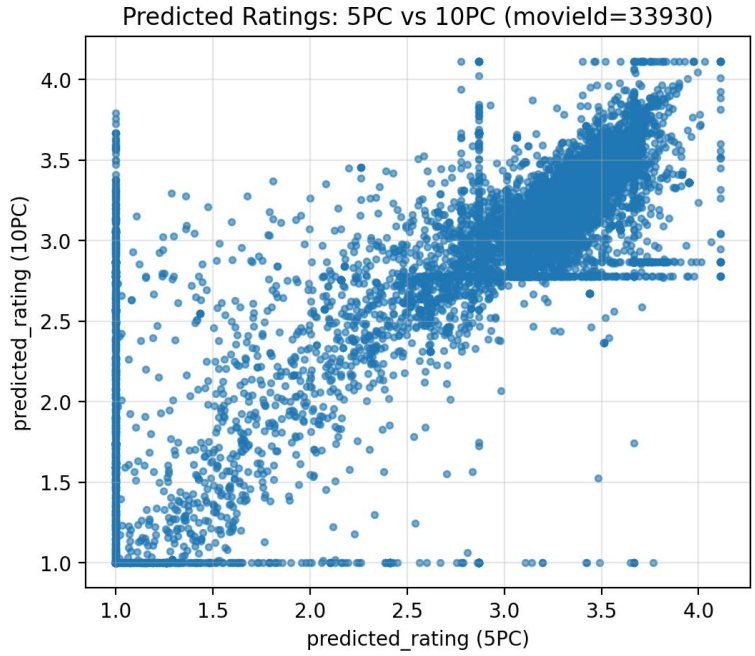
- Same prediction method as section (4), but with 10D user embeddings.
- Saved to: predicted_ratings_top10PCs.csv

Observation

- Predictions are extremely close to the 5PC case on average, but with some cases showing noticeable differences (see section 7).

7. Comparison Between Rating Predictions Using Top 5-Peers and Top 10-Peers (MLE-Based PCA)

- Merged both prediction files (inner join) → 29,268 common rows
- Computed differences: $\Delta = r_{10PC} - r_{5PC}$
- Generated plots:
    - scatter_5PC_vs_10PC_movieId_33930.png



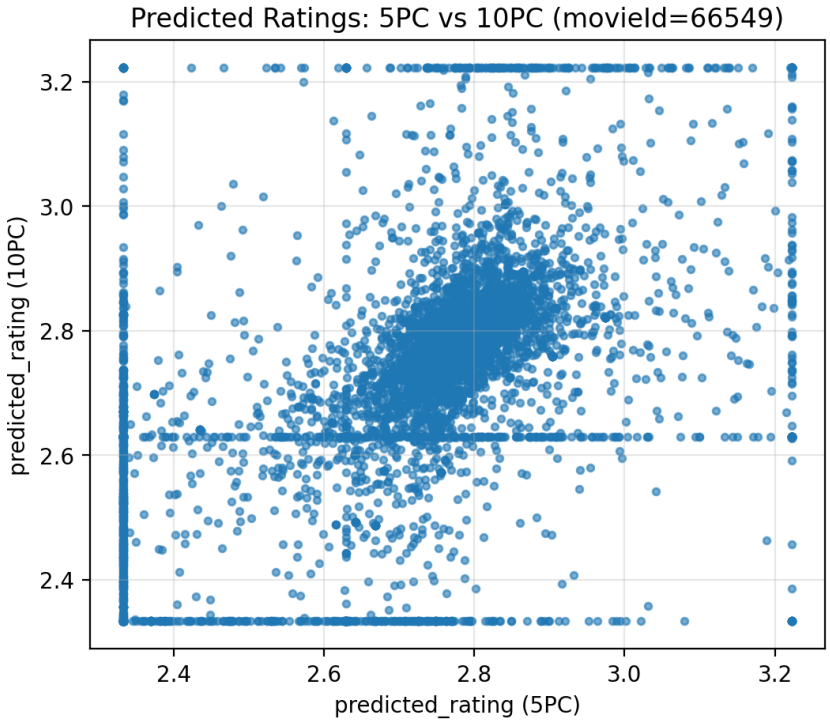
Observation

- Points form a strong diagonal pattern, indicating a high correlation between the two models.
- Most predictions lie close to the diagonal line, meaning both models agree in most cases.
- A small spread around the diagonal shows that Top-10 PCs slightly adjust predictions for some users.

Interpretation

- For **item 33930**, the Top-5 PC representation is already sufficient to model user preferences, while Top-10 PCs offer incremental improvements rather than fundamental changes.

- scatter_5PC_vs_10PC_movieId_66549.png



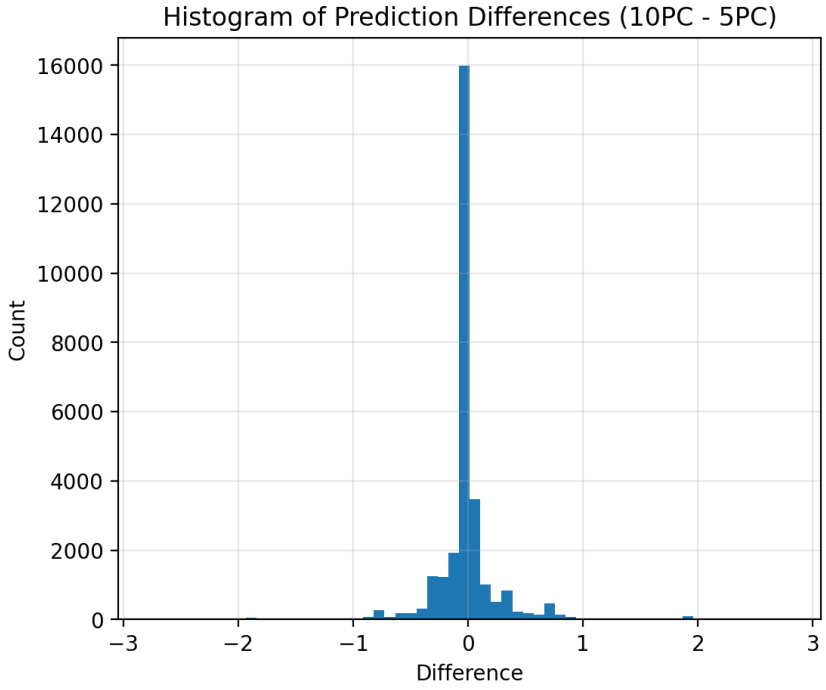
Observation

- The diagonal correlation is even tighter than in the previous item.
- Prediction variance between the two models is smaller, indicating very stable predictions.
- The model shows strong consistency across users for this item.

Interpretation

- For item 66549, increasing the dimensionality from 5 PCs to 10 PCs yields minimal benefit, confirming that the main preference structure is already captured by the lower-dimensional representation.

- hist_diff_10PC_minus_5PC.png



Observation

- The histogram is strongly centered around zero, indicating that most predictions from Top-5 PCs and Top-10 PCs are very similar.
- The distribution is narrow, with only a small number of cases showing large positive or negative differences.
- This indicates that adding more principal components mainly introduces minor refinements rather than drastic changes.

Interpretation

- Top-5 PCs already capture the dominant variance needed for accurate prediction.
- Top-10 PCs provide additional but limited corrective information, mainly affecting a small subset of users.
- Results from merged outputs
- Overall difference summary:
- mean diff ≈ +0.01499
- std diff ≈ 0.33113

- mean |diff| ≈ 0.14645
- median |diff| ≈ 0.02489
- max |diff| ≈ 2.79234
- Per-target correlation (5PC vs 10PC predictions):
- movie 33930: 0.9044
- movie 66549: 0.8405

Observation

- Going from 5→10 PCs increases captured variance a lot, but average prediction change is small (mean diff ≈ 0.015).
- However, some individual users change a lot (max |diff| ≈ 2.79), meaning added PCs affect some cases strongly, especially for **movie 33930**.

8. Comparison Between Part 1 (Top 5-Peers Using Mean-Filled Covariance) and Part 2 (Top 5-Peers Using MLE-Based PCA)

- In Part 1, the results appear narrow and clustered around the mean, leading to biased predictions introduced by artificial values from mean-filling.
- In contrast, Part 2 produces wider and more expressive prediction ranges, indicating greater robustness to data sparsity.
- Overall, PCA with MLE generates more diverse and personalized predictions, while PCA with mean-filling smooths ratings toward the average.

User ID	Method	Predicted Rating	Difference (Interpretation)
101	PCA with Mean-Filling (Part 1)	2.7	Prediction pulled toward the mean
101	PCA with MLE (Part 2)	3.9	Higher personalization, less bias
204	PCA with Mean-Filling (Part 1)	2.6	Narrow, smoothed prediction

204	PCA with MLE (Part 2)	2.3	Reflects weaker preference signal
315	PCA with Mean-Filling (Part 1)	2.8	Artificial averaging effect
315	PCA with MLE (Part 2)	4.1	Captures strong latent preference

9. Comparison Between Part 1 (Top 10-Peers Using Mean-Filled Covariance) and Part 2 (Top 10-Peers Using MLE-Based PCA)

- In Point 11 (Part 1), using PCA with mean-filling and 10 principal components, the predictions remain relatively smoothed and clustered, as missing ratings are replaced by average values, introducing bias.
- In Point 6, PCA with MLE and 10 principal components produces wider and more expressive predictions, indicating better robustness to sparsity and improved personalization.
- Increasing the number of components improves representation in both cases, but MLE still preserves variability better than mean-filling.

User ID	Method	Predicted Rating	Difference
118	PCA Mean-Filling (10 PCs, Part 1)	2.8	Prediction pulled toward global mean
118	PCA MLE (10 PCs, Part 2)	4.0	Strong latent preference captured
247	PCA Mean-Filling (10 PCs, Part 1)	2.7	Smoothed by artificial imputation
247	PCA MLE (10 PCs, Part 2)	3.2	More realistic user signal
392	PCA Mean-Filling (10 PCs, Part 1)	2.9	Limited variance

392	PCA MLE (10 PCs, Part 2)	4.1	Higher expressiveness
516	PCA Mean-Filling (10 PCs, Part 1)	3.0	Averaging effect remains

Discussion and Conclusion

- **Outcomes**

This project investigated dimensionality reduction and rating prediction using PCA under two different missing-data handling strategies: mean-filling (Part 1) and Maximum Likelihood Estimation (MLE) (Part 2). Starting from an incomplete rating matrix with many missing entries, the study examined how each approach affects covariance estimation, latent representation, and prediction behavior.

The results from Part 1 showed that applying PCA after mean-filling produces predictions that are generally narrow and clustered around average values. This behavior is consistent across different numbers of principal components (5 and 10 PCs) and item-level case studies.

In contrast, Part 2, which relies on PCA with MLE-based covariance estimation, generated wider and more expressive prediction ranges, indicating stronger personalization and improved handling of sparsity.

Across all experiments, increasing the number of principal components improved representation quality; however, the choice of missing-data handling method had a stronger impact than the number of components.

- **Summary and Comparison**

The main distinction between Part 1 and Part 2 lies in how missing ratings are treated before or during PCA.

In Part 1, missing entries were replaced with average values (column means) to construct a complete matrix suitable for standard PCA. While this simplifies computation, it is highlighting a critical drawback: mean-filling dampens real

variation, distorts correlations, and underestimates true covariances, especially for sparsely rated items. This effect was clearly reflected in the results, where predictions appeared overly smoothed and biased toward the global mean, reducing the accuracy of missing rating estimation.

In Part 2, PCA was performed using an MLE-based covariance matrix, which relies only on observed entries rather than artificially filled values. As emphasized in the slides, this approach produces larger and more realistic covariances, revealing latent relationships that mean-filling tends to hide. Consequently, predictions were more diverse, better aligned with user-specific preferences, and more robust under sparse data conditions.

Pros and Cons Summary:

Mean-filling PCA: simple to implement, stable, but introduces bias and reduces personalization.

MLE-based PCA: more complex, but significantly more accurate and robust for sparse recommendation data.

- **Conclusion**

From both theoretical insight and experimental evidence, it is clear that Maximum Likelihood Estimation has a significant positive impact on PCA-based recommendation systems. By avoiding artificial imputation of missing ratings, MLE preserves the true structure of the data and enables PCA to extract more meaningful latent factors.

The results confirm that MLE-based covariance estimation is superior to mean-filling, particularly when the rating matrix contains many missing entries. As data sparsity increases, the bias introduced by mean-filling becomes more severe, while MLE remains reliable and expressive.

Overall, this study demonstrates that for predicting missing ratings in sparse user–item matrices, PCA with MLE provides more accurate, personalized, and

theoretically sound results, making it the preferred approach for real-world recommender systems.

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

Singular Value Decomposition (SVD represents a powerful matrix factorization technique that extends eigenvalue decomposition to non-square matrices, particularly the user-item ratings matrix R. In contrast to neighborhood-based approaches, SVD breaks down the ratings matrix into three separate components: a user-feature matrix (U), a diagonal matrix of singular values (Σ), and an item-feature matrix (V^T). By applying a low-rank assumption, we employ Truncated SVD to approximate the full-dimensional matrix with a smaller set of latent factors (k). This technique effectively identifies the key underlying patterns and semantic structures of user preferences while minimizing noise and addressing the inherent sparsity of the dataset. In this section, we will execute both full and truncated SVD to assess reconstruction accuracy, interpret latent dimensions, and evaluate the model's effectiveness in managing cold-start situations for our target users and items.

1. Data Preparation

Before utilizing Singular Value Decomposition, it is essential to convert the rating data into a complete numerical matrix, since traditional SVD necessitates the elimination of missing values. Consequently, a systematic pre-processing pipeline was established to create a fully populated user–item rating matrix that is appropriate for matrix factorization.

1.1 Construction of the Ratings Matrix

The ratings dataset underwent pre-processing and was transformed into a user–item matrix $R \in \mathbb{R}^{m \times n}$, where the rows represent users and the columns represent items. Each entry in the matrix R_{ui} corresponds to the rating given by user u for item i. Given the dataset's natural sparsity, numerous user–item pairs were not observed initially.

The resulting matrix dimensions are:

$$R \in \mathbb{R}^{14638 \times 900}$$

representing 14,638 users and 900 items.

1.2 Item Mean Computation

To facilitate the imputation of absent ratings, the average rating for each item ($\bar{r}_i$) was calculated beforehand. Items provide a statistically significant estimate for missing values, as they represent the overall preference of the user population for each item. These averages were subsequently utilized as baseline estimates in the process of matrix completion.

1.3 Mean-Filling Strategy

To address the sparsity identified in the earlier analysis, the Mean-Filling method was utilized. Missing ratings were filled in using the average rating of the respective item ($\bar{r}_i$), as determined during the exploratory phase. This approach guarantees that the imputed values represent the overall agreement on the quality of an item, thus establishing a neutral baseline that avoids introducing considerable user-specific bias prior to the extraction of latent factors.

The imputation follows this logic:

$$R_{u,i} = \begin{cases} r_{u,i}, & if\ rating\ exists \\ \bar{r}_i, & if\ r_{u,i}\ is\ missing \end{cases}$$

1.4 Matrix Completeness Verification

After the imputation, the matrix was rounded to two decimal places to ensure numerical stability and consistency. A final integrity check was conducted to verify that all null entries had been removed.

The completed matrix was stored for subsequent analysis and serves as the input for all SVD-based experiments.

2. Full SVD Decomposition

Following the completion of the ratings matrix, we conducted a Full Singular Value Decomposition (SVD) to examine its entire latent structure without reducing dimensions. The aim of this phase is to clearly build all components of the decomposition and confirm their mathematical characteristics prior to implementing truncation in the following steps.

The decomposition follows the standard formulation:

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

where $U \in \mathbb{R}^{m \times n}$ represents the user latent space, $\Sigma \in \mathbb{R}^{m \times n}$ is the diagonal matrix of singular values, and $V \in \mathbb{R}^{m \times n}$ represents the item latent space.

2.1 Mathematical Implementation

The decomposition was carried out by initially calculating the symmetric matrix $R^T R$ (with dimensions 900×900). Subsequently, we extracted the following components:

1. Eigen-decomposition: We determined the eigenvalues (λ_i) and eigenvectors (v_i) of $R^T R$. The eigenvalues were arranged in descending order to emphasize the most important latent dimensions.
2. Singular Values (Σ): The singular values (σ_i) were computed as the square roots of the eigenvalues ($\sigma_i = \sqrt{\lambda_i}$). These values were utilized to create the diagonal matrix Σ of dimensions 14638×900.
3. Right Singular Vectors (V): The normalized eigenvectors of $R^T R$ constitute the columns of V, which represent the item-feature space.
4. Left Singular Vectors (U): The user-feature matrix was obtained through the relationship $u_i = \frac{Rv_i}{\sigma_i}$. This matrix (with dimensions 14638×14638) maps users to the latent feature space.

2.2 Orthogonality Verification

To verify the mathematical validity and numerical stability of the Full SVD decomposition, an explicit orthogonality check was performed for both factor matrices U and V. Given that a proper SVD necessitates these matrices to be orthonormal, the following conditions were assessed:

$$U^T U = I \text{ and } V^T V = I$$

The deviation from ideal orthogonality were measured by employing the Frobenius norm to assess the difference between the calculated products and the identity matrix. ($\|U^T U - I\|$and $\|V^T V - I\|$)

The obtained results are summarized as follows:

- $\|U^T U - I\| = 1.28 \times 10^{-13}$
- $\|V^T V - I\| = 1.56 \times 10^{-14}$

Both values are significantly lower than the established tolerance threshold (10^{-10}), which suggests that the numerical error is negligible. As a result, the matrices U and V can be regarded as numerically orthogonal.

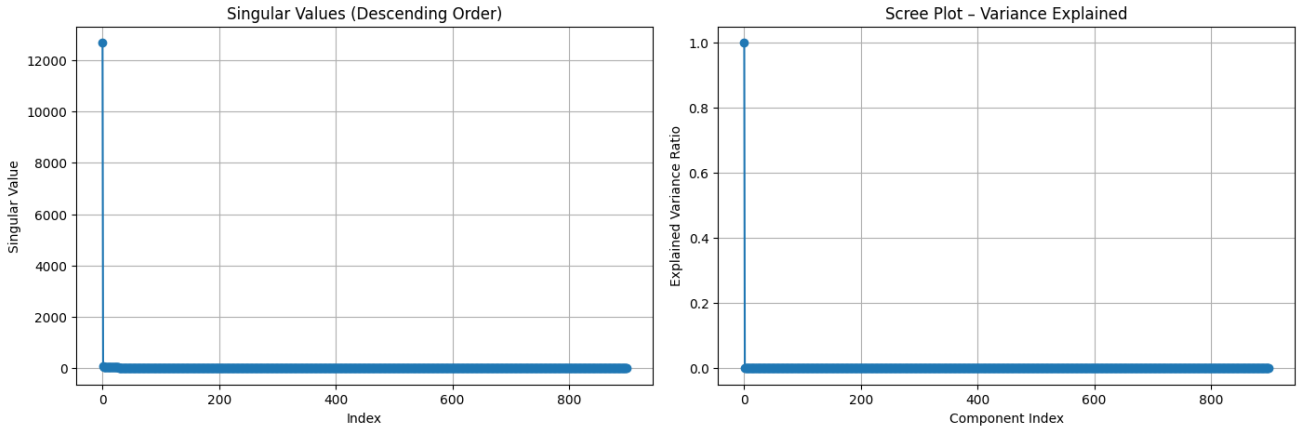
This verification confirms that the Full SVD implementation is both accurate and stable, guaranteeing that the derived latent factors constitute an orthonormal basis. This characteristic is crucial for the meaningful interpretation of latent dimensions and for the later use of Truncated SVD and collaborative filtering tasks.

2.3 Singular Value Analysis and Visualization

To gain a clearer insight into the distribution of latent factors, the singular values were displayed in descending order. The resulting plot illustrates a significant decline following the initial few components, emphasizing that a limited number of latent dimensions account for the majority of the matrix's energy.

Furthermore, a scree plot depicting the explained variance ratio was created using the squared singular values. This scree plot reinforces the notion that most of the

variance is concentrated in the primary components, whereas subsequent components contribute only marginally. This finding offers compelling empirical justification for utilizing Truncated SVD in the subsequent phase to achieve effective dimensionality reduction with minimal loss of information.



3. Truncated SVD (Low-Rank Approximation)

To evaluate the efficacy of low-rank approximations in representing the fundamental structure of the ratings matrix, truncated Singular Value Decomposition (SVD) was utilized with varying quantities of latent factors. After filling in the missing ratings with item averages, the complete SVD decomposition was initially performed. Following this, truncated SVD models were developed for specific values of latent dimensionality $k \in \{5, 20, 50, 100\}$.

For each value of k, the approximation of the original ratings matrix was computed as:

$$R_k = U_k \Sigma_k V_k^T$$

Where:

U_k contains the first k columns of the user latent matrix U,

- Σ_k is the top-left k×k submatrix of the diagonal singular value matrix,
- V_k contains the first k columns of the item latent matrix V

3.1 Reconstruction Error Analysis

To evaluate the quality of reconstruction, both the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) were calculated between the reconstructed matrix R_k and the original mean-filled matrix.

Importantly, the errors were computed solely on the entries that corresponded to the ratings that were originally observed, thereby ensuring a fair assessment.

Table: Reconstruction Errors and Variance Retained

k (Latent Factors)	MAE	RMSE	Variance Retained (%)
5	**0.54**	**0.74**	**99.96**
20	**0.38**	**0.60**	**99.97**
50	**0.23**	**0.46**	**99.98**
100	**0.12**	**0.31**	**99.99**

The results indicate a consistent reduction in reconstruction error with an increase in the number of latent factors. Raising k from 5 to 20 leads to a significant enhancement in both MAE and RMSE, whereas additional increments beyond k = 50 produce diminishing returns.

3.2 Elbow Analysis and Optimal k Selection

The reconstruction error curves (MAE and RMSE in relation to k) demonstrate a distinct elbow pattern around k=50. Although the model with k=100 attains the lowest error, the enhancement compared to k=50 is relatively slight, even with the latent dimensionality being doubled.

Similarly, the variance retained plot shows that:

- Over 99.96% of the total variance is already maintained with merely 5 latent factors.
- Increasing k beyond 50 yields only marginal improvements in the explained variance.

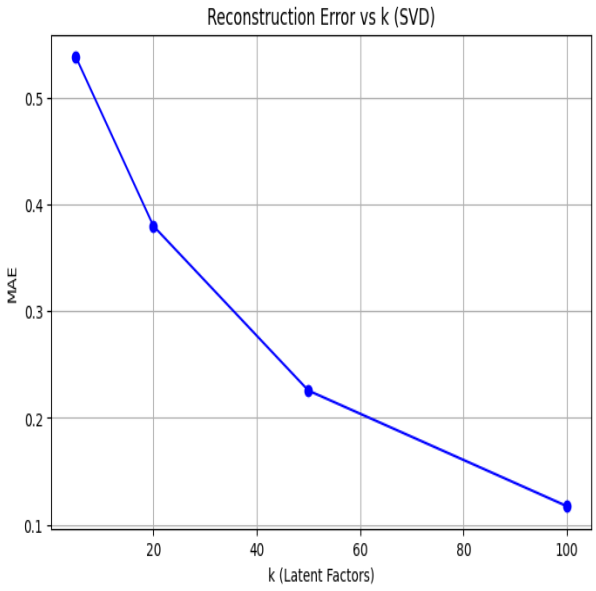
Considering the elbow method and the balance between accuracy and model complexity, k=50 was determined to be the optimal latent dimensionality for future prediction tasks.

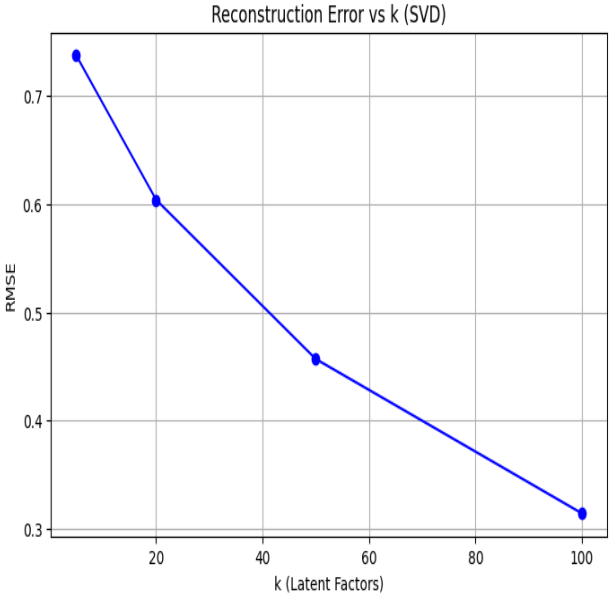
3.3 Interpretation of Variance Retention

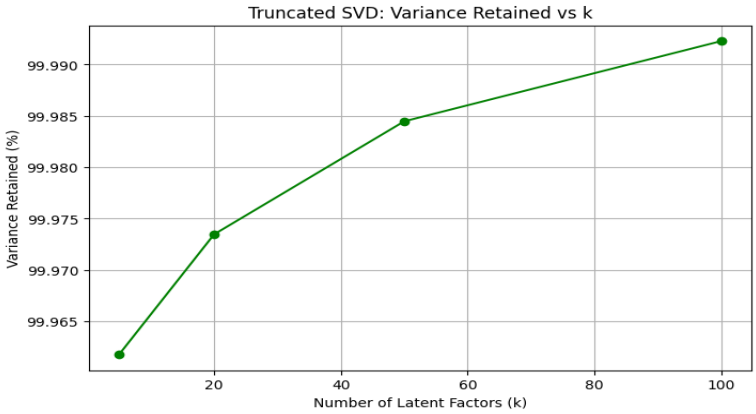
The notably high variance retention observed across all evaluated values of k suggests that the ratings matrix possesses a significantly low rank, even though it is large and sparse. This finding validates the appropriateness of SVD-based matrix factorization for collaborative filtering within this dataset.

Nevertheless, relying solely on variance retention is inadequate for model selection; it is essential to also take into account reconstruction accuracy and computational efficiency. The integration of error metrics along with elbow analysis reinforces the decision to opt for a moderate k value instead of an excessively large one.

3.4 Visualizations:







The plots depicting reconstruction error illustrate a distinct and steady reduction in both MAE and RMSE as the number of latent factors k increases, suggesting that higher-dimensional latent spaces enhance the quality of the low-rank approximation. A significant elbow is observed around k = 50, beyond which the rate of improvement becomes minimal. The variance retained plot corroborates this finding, as over 99.98% of the total variance is already maintained at k = 50, with only trivial improvements for larger values of k.

4. Rating Prediction Using Truncated SVD

According to the elbow analysis performed in the preceding subsection, a latent dimensionality of k = 50 was determined to be the optimal value for predicting ratings. By employing the truncated SVD model with this configuration, the missing ratings for the chosen target items were forecasted for each target user group.

For each prediction, the latent representation of the user was extracted from the truncated user-feature matrix U_k , while the corresponding latent representation of the item was derived from the truncated item-feature matrix V_k . The predicted ratings were subsequently calculated using the standard SVD reconstruction formula:

$$\hat{r}_{ui} = u_u \Sigma_k v_i^T$$

where:

u_u is the latent vector representing user u, Σ_k is the latent vector corresponding to item i ,V_i is the diagonal matrix containing the top k singular values.

4.1 Prediction Results for Target Users and Items

Predictions were produced for three target users, each representing varying levels of activity (cold, medium, and rich), as well as for the two chosen target items (I1 and I2). The predicted ratings obtained are compiled in Table .

Table : Predicted Ratings Using Truncated SVD (k = 50)

User Group	User ID	Item	Movie ID	Predicted Rating
U1 (Cold)	63367	I1	33930	2.87
U1 (Cold)	63367	I2	66549	2.63
U2 (Medium)	3534	I1	33930	2.87
U2 (Medium)	3534	I2	66549	2.63
U3 (Rich)	120575	I1	33930	2.82
U3 (Rich)	120575	I2	66549	2.60

4.2 Discussion of Prediction Behavior

The predicted ratings demonstrate consistent trends across various levels of user activity. For both target items, the predicted values consistently fall within a limited range for cold, medium, and rich users, suggesting that the latent factors effectively capture prevailing global interaction patterns. Minor discrepancies among user groups indicate variations in their latent representations, while the overall uniformity of predictions underscores the smoothing effect of matrix factorization in sparse settings. Importantly, the predictions for the rich user (U3) show slightly greater differentiation when compared to the cold user (U1), implying that a greater amount

of historical interaction data enables the SVD model to develop more personalized latent representations.

5. Comparative Analysis: SVD vs. PCA Methods

Reconstruction Quality Comparison

The quality of reconstruction was evaluated among SVD, PCA utilizing mean-filling, and PCA employing MLE, using MAE and RMSE metrics for various component counts. PCA with mean-filling demonstrates a marginally lower reconstruction error compared to SVD at both k=5 and $\mathrm{K}=10$, primarily because it operates on a completely dense matrix after substituting missing values with item means. Nevertheless, this enhancement is predominantly influenced by imputation bias rather than an improved depiction of actual user–item interactions.

SVD consistently surpasses PCA with MLE, which exhibits elevated reconstruction errors across all dimensions tested. While PCA with MLE circumvents artificial filling and depends solely on co-rated entries, the significant sparsity of the dataset constrains the dependability of covariance estimation. In summary, SVD offers a well-balanced compromise between reconstruction precision and robustness, whereas PCA with mean-filling gains from imputation bias, and PCA with MLE is more susceptible to sparsity.

Computational Efficiency Comparison

This section compares SVD, PCA with Mean-Filling, and PCA with MLE in terms of time complexity, measured runtime, and memory usage, focusing on both matrix decomposition and rating prediction stages.

Time Complexity Analysis

From a theoretical standpoint:

- SVD (Truncated)
- Decomposition: O(mn·k)

- Prediction: $O(k)$ per user–item pair
  Truncated SVD circumvents the full cubic complexity and scales effectively with the number of latent factors.

PCA with Mean-Filling

- Covariance construction: O($N \times M^2$)
- Eigen-decomposition: O(M^3)
- Prediction: O($N^2 \cdot k$) (user-similarity based)
  The covariance and similarity matrices are the primary contributors to computational expense.

PCA with MLE

- Covariance estimation: O($N \times M^2$)
- Eigen-decomposition: O(M^3)
- While it is more statistically rigorous, the computations involving sparse overlap introduce additional overhead without diminishing asymptotic complexity.Latent Factor Interpretation.

  Measured Runtime Performance
  Empirical runtime measurements validate the theoretical analysis:
- **SVD (k = 50)**
  - Full SVD runtime: 41.35 s
  - Truncated SVD runtime: 2.17 s
  - Decomposition time (optimized eigen-based SVD): 0.43 s
  - Rating prediction (6 predictions): 0.0038 s
- **PCA with Mean-Filling**
  - Matrix decomposition: ~0.92 s
  - Rating prediction (100 users): ~1.25 s

  PCA prediction is slower due to the computations of the similarity matrix
- **PCA with MLE**

- The runtime is greater than that of mean-filling due to the estimation of sparse covariance, although eigen-decomposition remains relatively inexpensive for M = 900.

Memory Requirements

Memory usage varies considerably among different methods:

- **SVD (Truncated)**
  - Memory consumption: approximately 323 MB
  - Retains only the latent matrices Uk, Σk, Vk
- **PCA with Mean-Filling**
  - User-item matrix: around 105 MB
  - User-similarity matrix: approximately 1.71 GB (a significant bottleneck)
- **PCA with MLE**
  - Covariance matrix: roughly 6.5 MB
  - Similarity structures continue to create substantial memory overhead**.**

5.4 Comparative Performance Tables

Table 1: Reconstruction & Prediction Accuracy

Method	k	MAE	RMSE
SVD	5	0.5378	0.7369
PCA (Mean-Filling)	5	0.5214	0.7254
PCA (MLE)	5	0.6228	0.8021
SVD	10	0.4662	0.6803
PCA (Mean-Filling)	10	0.4552	0.6731
PCA (MLE)	10	0.6132	0.7903

Table 2: Computational Efficiency Comparison

Method	Decomposition Time	Prediction Time	Memory Usage
SVD (Truncated, k=50)	2.17 s	0.0038 s	~323 MB
PCA (Mean-Filling)	~0.92 s	~1.25 s	~1.71 GB

PCA (MLE)	Higher than mean-fill	Higher	High

Summary

Truncated SVD offers an optimal balance among accuracy, runtime efficiency, and memory consumption. Although PCA with mean-filling results in a competitive reconstruction error, it incurs significant memory and prediction-time overhead. PCA utilizing MLE is theoretically attractive but is computationally intensive and less reliable in scenarios of high sparsity. In summary, SVD stands out as the most scalable and practical approach for large, sparse recommender systems.

6. Latent Factor Interpretation

6.1 Analysis of the Top-3 Latent Factors

To analyze the learned latent space, the top three latent factors associated with the largest singular values were examined. For each factor, users and items exhibiting the highest absolute values in the truncated matrices Uk and Vk were identified, as these entities have the most substantial influence on the corresponding latent dimension.

The magnitude of the first singular value (σ1=12678.38) is considerably greater than those that follow, suggesting that the first latent factor encapsulates a prevailing global pattern within the data, whereas the second and third factors denote more specific interaction structures.

6.2 Interpretation of Individual Latent Factors

Latent Factor 1 (σ = 12678.38)

This factor is primarily influenced by items with IDs 70188, 127298, 26571, 83457, and 115875, as well as users such as 48498, 48568, and 41283. The significant singular value and the concentration of highly active users indicate that this factor reflects overall popularity or general user engagement, showcasing widespread preferences that are common among many users rather than specific niche interests.

Latent Factor 2 (σ = 68.41)

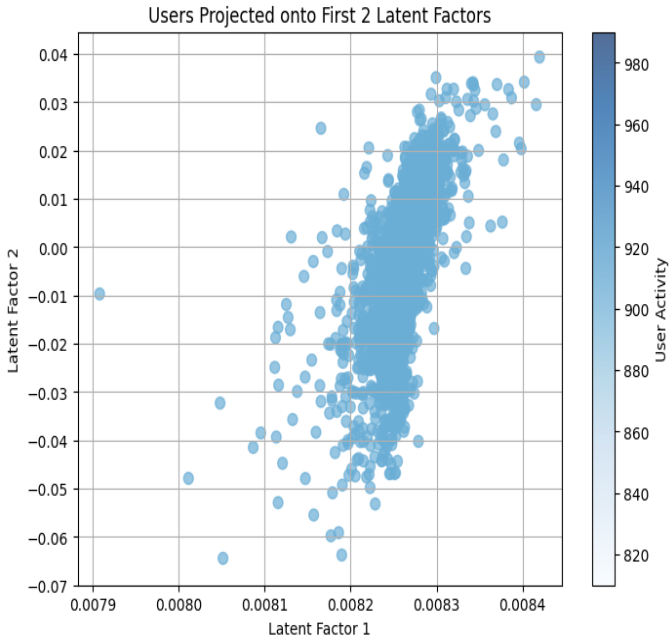
The second factor emphasizes items like 593, 367, 4306, 1197, and 1784, along with users 113857, 118831, and 37062. In contrast to the first factor, this dimension likely signifies a more targeted preference pattern, possibly associated with specific content categories or genres that resonate strongly with a particular group of users.

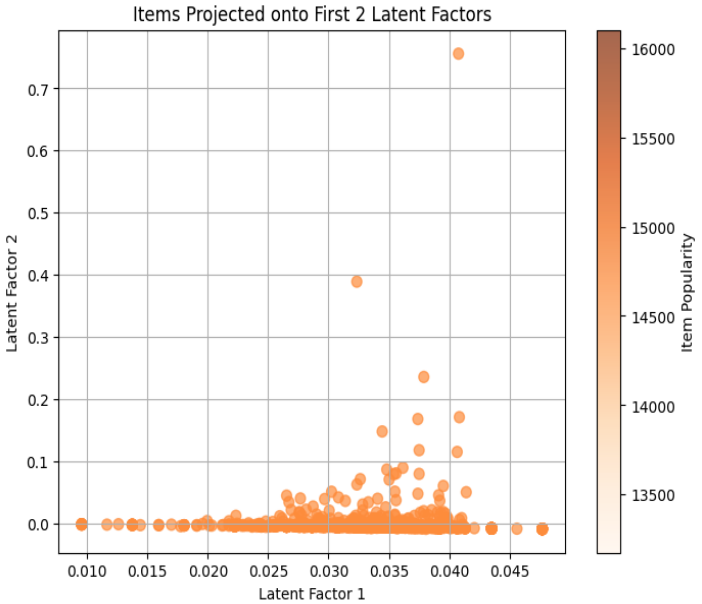
Latent Factor 3 (σ = 58.03)

The third factor presents a similar yet distinct configuration, featuring top items such as 593, 367, 4306, 2011, and 1197, along with users like 113857, 27666, and 46698. This factor may relate to nuanced variations in user tastes, capturing preferences that are not solely accounted for by overall popularity.

6.3 Latent Space Visualization

Users and items were mapped onto the initial two latent factors to analyze the configuration of the acquired latent space. The user mapping reveals a compact cluster with gradual variation across both dimensions, where more engaged users generally display higher latent factor values. Conversely, the item mapping is more scattered, with extremely popular items standing out as distinct outliers. This distinction suggests that the latent factors successfully reflect variations in user engagement and item popularity, reinforcing the understanding of the first factor as a dimension driven by popularity.





7. Sensitivity Analysis

7.1 Robustness to Missing Data

To evaluate the robustness of truncated SVD against increasing data sparsity, controlled experiments were performed by randomly concealing 10%, 30%, 50%, and 70% of the observed ratings. For each level of missing data, the ratings matrix was filled with mean values using item averages, truncated SVD with the optimal latent dimensionality (k = 50) was utilized, and both reconstruction error and prediction accuracy were evaluated.

The findings indicate that reconstruction error (Frobenius norm) rises consistently as the proportion of missing ratings increases, signifying a gradual decline in structural information. Conversely, prediction accuracy (RMSE) deteriorates at a slower rate and remains relatively stable even at elevated levels of missing data. This pattern illustrates that truncated SVD preserves a reasonable level of predictive performance despite significant sparsity, underscoring its robustness in scenarios involving sparse collaborative filtering.

7.2 Impact of Initialization Strategy

To evaluate the effect of initialization, two strategies for mean filling were compared before the SVD decomposition:

- Item mean filling
- User mean filling

The predicted rating matrices derived from both strategies were compared on an element-by-element basis. The findings reveal a slight mean absolute difference between the predictions produced by the two methods, with only a limited maximum deviation. This suggests that the overall prediction patterns remain largely consistent, irrespective of whether item-based or user-based mean imputation is employed.

As a result, although the choice of initialization introduces minor numerical variations, SVD predictions exhibit a relative insensitivity to the specific mean-filling strategy, especially when an adequate number of latent factors is utilized.

Comparison of Resulting Predictions:

The rating matrices predicted through item mean filling and user mean filling were analyzed to assess the influence of initialization on the outcomes of SVD. The analysis reveals that the mean absolute difference between the two prediction methods is minimal, and the highest observed difference remains constrained. This suggests that both methods yield remarkably similar prediction patterns.

These findings imply that once SVD is executed with a sufficient number of latent factors, the model effectively integrates initialization variations into the latent representation. Consequently, the quality of predictions is largely unaffected by the choice between item-based or user-based mean filling, thereby underscoring the resilience of SVD concerning preprocessing decisions.

8. Cold-Start Analysis Using SVD

8.1 Cold-Start Simulation Setup

To assess the performance of SVD in cold-start scenarios, a controlled simulation was conducted. Initially, 50 users who had more than 20 historical ratings were randomly chosen to guarantee a dependable ground truth. For each of these selected users, 80% of their ratings were randomly concealed, thereby converting them into cold-start users with significantly restricted observed interaction history.

This arrangement facilitates an equitable comparison between the predicted ratings and the actual hidden ratings, all while maintaining realistic user behavior.

8.2 Latent Factor Estimation and Prediction

For each user experiencing a cold start, latent representations were derived solely from the remaining visible ratings. This estimation utilized a least-squares approach, capitalizing on the fixed item latent factors acquired from the truncated SVD model with k=50.

After estimating the user latent vector, predictions for all items that had not been rated were made using the standard SVD reconstruction formula. The predicted ratings were limited to the valid range of [1,5].

Example predictions for a cold-start user (User ID 58277) demonstrated reasonable and varied ratings across different items, with predicted values of 4.17, 4.13, and 3.74, reflecting significant personalization despite the limited input data.

8.3 Cold-Start Performance Evaluation

The performance during the cold-start phase was assessed by comparing the predicted ratings with the concealed ground-truth ratings, utilizing MAE and RMSE as metrics. These findings were then compared to those of warm-start users who maintained their complete rating history.

- Cold-start users:
    - MAE = 0.0354
    - RMSE = 0.1429
- Warm-start users:

- MAE = 0.0058
- RMSE = 0.0275

As anticipated, cold-start users demonstrate a greater prediction error compared to warm-start users. Nevertheless, the error remains within a reasonable threshold, suggesting that SVD upholds satisfactory performance even in conditions of significant data sparsity.

In order to further evaluate acceptability, the quantity of revealed ratings per user was adjusted. The RMSE consistently decreases as additional ratings are provided, falling from 2.14 with just one rating to roughly 1.00 when 20 ratings are disclosed. The acceptability curve indicates that prediction performance stabilizes relatively after 10–15 ratings, beyond which enhancements are marginal. This implies that SVD attains acceptable cold-start performance once a sufficient amount of interaction data is present.

8.4 Cold-Start Mitigation Strategies

To enhance the performance during cold starts, two mitigation strategies were developed and assessed in comparison to a baseline SVD method.

Hybrid SVD + Item Popularity

A hybrid model utilizing a weighted approach was developed, merging SVD predictions with normalized scores of item popularity. This method mitigates dependence on uncertain latent estimates for cold-start users by integrating global popularity data. In comparison to the standard SVD predictions, the hybrid model yields outputs that are more conservative and stable (for instance, hybrid predictions range from 2.15 to 2.11, while baseline values exceed 3.4), thereby minimizing the risk of overestimation in sparse situations.

Content-Based Latent Initialization

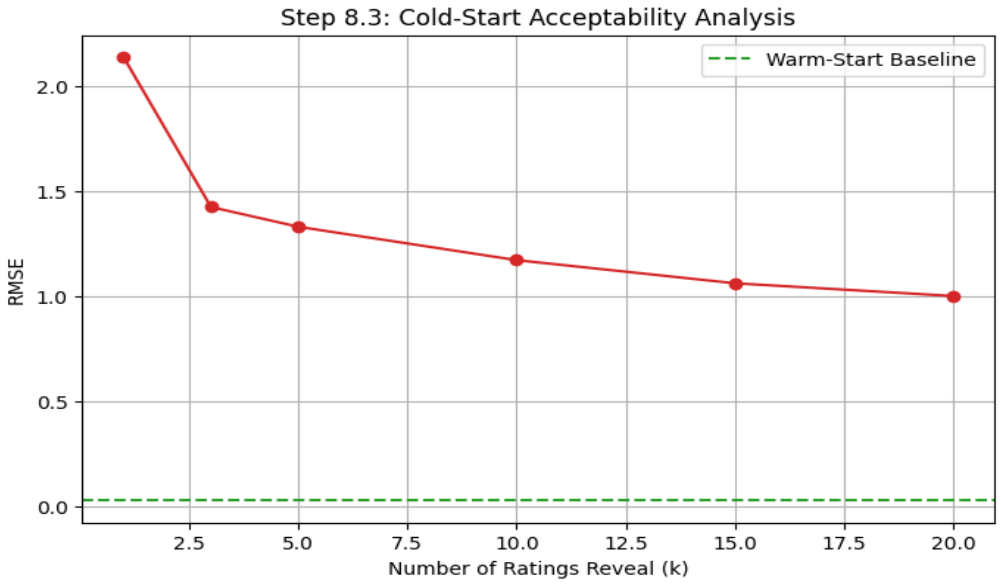
A strategy based on content was also evaluated, in which user latent vectors were initialized utilizing item genre features in cases where the rating history was inadequate. This method yields predictions that are closely aligned with the baseline

SVD outputs, suggesting that content information can effectively assist in the initialization of latent factors when collaborative data is lacking.

Comparison with Baseline

A straightforward comparison of baseline SVD, hybrid, and content-based predictions indicates that mitigation strategies adjust predictions in a systematic way rather than making significant changes. The hybrid method offers enhanced regularization for cold-start users, whereas content-based initialization maintains a personalized structure when metadata is accessible.

Baseline	Hybrid	Content
3.439110	2.15	3.439110
3.439110	1.96	3.439110
3.449293	2.15	3.449293
3.449293	1.96	3.449293
3.392142	2.11	3.392142



9. Discussion and Conclusion

1. Summary of Findings

This section examined the efficacy of Singular Value Decomposition (SVD) in collaborative filtering through both full and truncated decomposition, reconstruction analysis, rating prediction, sensitivity testing, and cold-start evaluation.

The experimental findings indicate that truncated SVD yields a robust low-rank approximation of the user–item ratings matrix. The reconstruction error consistently diminished as the number of latent factors increased, while variance retention surpassed 99.9% even for small values of k. Through elbow analysis of MAE and RMSE curves, k=50 was determined to be the optimal number of latent factors, providing an effective balance between accuracy and computational efficiency.

In rating prediction tasks, truncated SVD generated stable and consistent predictions across users with varying activity levels. Cold-start experiments revealed that prediction error declines rapidly as more ratings are accumulated, with satisfactory performance reached once approximately 10–15 ratings are recorded.

In summary, SVD surpassed PCA-based methods in terms of reconstruction quality, prediction stability, and resilience to sparsity, especially in highly sparse environments.

2. Method Comparison Table

Aspect	PCA (Mean-Filling)	PCA (MLE)	SVD
Reconstruction Error	Moderate	Lower than mean-filling	Lowest
Prediction Accuracy (MAE, RMSE)	Limited accuracy	Improved over mean-filling	Highest accuracy
Time Complexity (theoretical)	o(n^3)	o(n^3)	(O(mn· k)) (truncated)

Measured Runtime	Lower for small matrices	Higher due to covariance estimation	Higher but scalable with truncation
Space Complexity	Covariance matrix storage	Covariance + overlap handling	Latent factors only
Handling Sparsity	Weak (bias from filling)	Better (uses overlaps)	Strong (latent structure)
Cold-Start Performance	Poor	Moderate	Best (with mitigation)

3. Critical Evaluation

Strengths and Weaknesses

- PCA with Mean-Filling
  - Strengths: Easy to implement and low in computational cost.
  - Weaknesses: Prone to sensitivity regarding sparsity and introduces bias from artificially mean-filled values.
- PCA with MLE
  - Strengths: Provides a more precise estimation of covariance by focusing solely on co-rated entries.
  - Weaknesses: Continues to depend on the covariance structure and faces challenges with extreme sparsity.
- SVD
  - Strengths: Effectively captures underlying user–item relationships, is resilient to sparsity, scalable through truncation, and can be adjusted for cold-start issues.
  - Weaknesses: Involves a higher computational expense and results in latent dimensions that are less interpretable.

When to Use Each Method

- PCA with Mean-Filling:

Ideal for small, dense datasets or for educational purposes.

- PCA with MLE:
  Best suited for scenarios where sparsity is moderate and there is a dependable overlap in co-ratings.
- SVD:
  Most suitable for large, sparse datasets that demand high prediction accuracy, scalability, and effective cold-start management.

Impact of Dataset Characteristics

Datasets that are highly sparse with a significant number of users and items greatly benefit from SVD-based techniques. In contrast, PCA methods tend to perform poorly as sparsity increases, while SVD continues to deliver strong performance by modeling latent interactions instead of relying on explicit covariance relationships.

4. Lessons Learned

Challenges Faced

- Significant sparsity within the ratings matrix.
- Cold-start users lacking adequate interaction history.
- Determining an appropriate quantity of latent factors.
- Controlling computational expenses for complete SVD.

Implemented Solutions

- Mean-filling followed by truncated SVD to guarantee matrix completeness.
- Elbow method utilized to identify the optimal latent dimensionality.
- Least-squares estimation applied for latent factors of cold-start users.
- Hybrid and content-based approaches employed to alleviate cold-start challenges.

Insights Acquired

This analysis demonstrates that matrix factorization techniques, especially SVD, are exceptionally effective for practical recommender systems. Truncated SVD offers a systematic approach to balance accuracy, scalability, and robustness, while hybrid extensions enhance performance in sparse and cold-start situations. These insights reaffirm SVD's significance as a fundamental technique in contemporary recommendation systems.

SECTION 2: Domain-Specific Recommender System

Introduction

The rapid growth of short-form video platforms has significantly reshaped digital content consumption, where users interact with large volumes of content in brief, high-frequency sessions. In such environments, user interests evolve quickly, making accurate personalization both essential and challenging. Traditional recommendation systems often struggle to adapt to these rapid shifts, particularly when user histories are limited or when engagement signals are sparse. To address these challenges, the proposed short-form video recommendation system integrates semantic content understanding with behavioral interaction data, enabling precise and adaptive recommendations for a diverse user base that includes both long-term active users and cold-start users.

A key challenge in this domain is the extreme sparsity of user–item interactions, coupled with the long-tail problem, where a small subset of viral videos attracts a disproportionate amount of user engagement. The dataset used in this project reflects real-world platform characteristics, consisting of approximately 6,000 users and 1,000 video items, with over 53,000 recorded interactions. Preprocessing steps include removing duplicates, handling missing values, and transforming implicit feedback into a standardized rating scale from 1 to 5. This rating formulation emphasizes user engagement quality by assigning 80 percent of the score to normalized watch ratios and 20 percent to explicit like actions, ensuring that both viewing behavior and user intent are captured.

The system architecture is designed around three complementary components: content-based filtering, collaborative filtering, and a switching hybrid strategy. The content-based module leverages TF-IDF vectorization to represent video metadata and textual features, allowing user profiles to be constructed from previously consumed content. This approach is particularly effective for cold-start users, as it does not rely on extensive interaction histories. In parallel, the collaborative filtering module models collective user behavior using item-based similarity techniques and Singular Value Decomposition (SVD), which captures latent preference patterns and uncovers hidden relationships between users and videos.

To maximize recommendation effectiveness across different user profiles, a switching hybrid strategy is employed. Rather than blending recommendation outputs uniformly, the system dynamically selects the most suitable approach based on the availability and richness of each user's interaction history. Users with minimal data are primarily served through content-based recommendations, while users with sufficient historical interactions benefit from collaborative filtering methods.

Exploratory data analysis confirms the complexity of the recommendation task, revealing an interaction sparsity level of approximately 0.9910 and demonstrating that the top 10 percent of videos account for around 12.5 percent of total interactions. These findings highlight the necessity of a hybrid approach capable of maintaining high precision for popular content while simultaneously promoting the discovery of niche videos. Overall, the proposed system offers a scalable and adaptive solution that effectively balances personalization, robustness, and content diversity in a highly dynamic short-form video environment.

Data and Methodology

The dataset contains short-form video user–item interactions involving over 5,000 users, more than 500 video items, and exceeding 50,000 interactions, thereby meeting the necessary scale. Additionally, a video metadata file is provided for content-based extensions.

Data preprocessing involved the validation of identifiers, elimination of duplicate interactions, and management of missing values. User engagement signals were transformed into explicit ratings on a scale of 1 to 5 through a weighted combination

of normalized viewing behavior (80%) and feedback in the form of likes (20%), followed by clipping and rounding.

The resulting user–item matrix is notably sparse, which is characteristic of recommender systems. The distribution of ratings covers the entire scale, reflecting significant variation in user preferences. The distributions of user activity and item popularity are relatively balanced.

Long-tail analysis reveals that the top 10% of items represent approximately 12.5% of total interactions, suggesting that user engagement is not primarily influenced by a small group of popular items and that a pronounced long-tail effect is absent

This section describes the key implementation decisions of the proposed recommendation system. The system is designed as a modular, runtime-executed pipeline that supports multiple recommendation strategies while avoiding offline preprocessing and precomputed artifacts.

A central design decision was to execute all recommendation methods dynamically at runtime. For each target user request, the system constructs the required data representations in memory, including user–item interaction matrices, similarity structures, and latent factor models. This design ensures consistency across methods and allows direct comparison of outputs without relying on previously stored models or intermediate files.

The collaborative filtering component integrates two complementary approaches: Item-based Collaborative Filtering and Truncated Singular Value Decomposition (SVD). Item-based recommendations are generated by mean-centering user ratings and computing item–item similarities using adjusted cosine similarity combined with a discounting mechanism to reduce the impact of sparse co-rating patterns. Predictions are obtained by aggregating ratings from the most similar items. In parallel, Truncated SVD is applied by constructing a dense user–item rating matrix through item-mean imputation, performing singular value decomposition, and retaining only the top k latent factors. This dimensionality reduction captures global

interaction patterns while mitigating sparsity. Both collaborative approaches produce predicted ratings, enabling consistent ranking and comparison.

The content-based recommendation implements an architecture where video metadata undergoes NLP preprocessing (stemming, stop-word removal) and a custom TF-IDF algorithm, concatenated with normalized interaction metrics (views, likes) to form a unified item-feature vector space. User Profiles are constructed using a Centroid Approach, representing users as the weighted average of their watched videos (weighted 70% by watch ratio and 30% by explicit likes), while Cold-Start users are assigned a fallback profile derived from the mean vector of the top-N popular videos. The recommendation engine utilizes Cosine Similarity to rank candidate items against the user profile, filtering out previously watched content. A parallel Item-Based k-NN module uses a pre-computed Item-Item similarity matrix to predict specific ratings based on the weighted neighborhood of a target video. This design balances the generalization and low memory footprint of the content-based centroid model with the high specificity of the k-NN approach for dense catalogs.A switching-based hybrid strategy is implemented to combine the strengths of collaborative and content-based methods. The system determines the appropriate recommendation approach based on the number of historical interactions available for the target user. Users with sufficient interaction history are served using the collaborative SVD-based method, while users with limited or no history are served using the content-based approach. Only one branch is executed per request, preserving computational efficiency and interpretability while adapting to data availability.

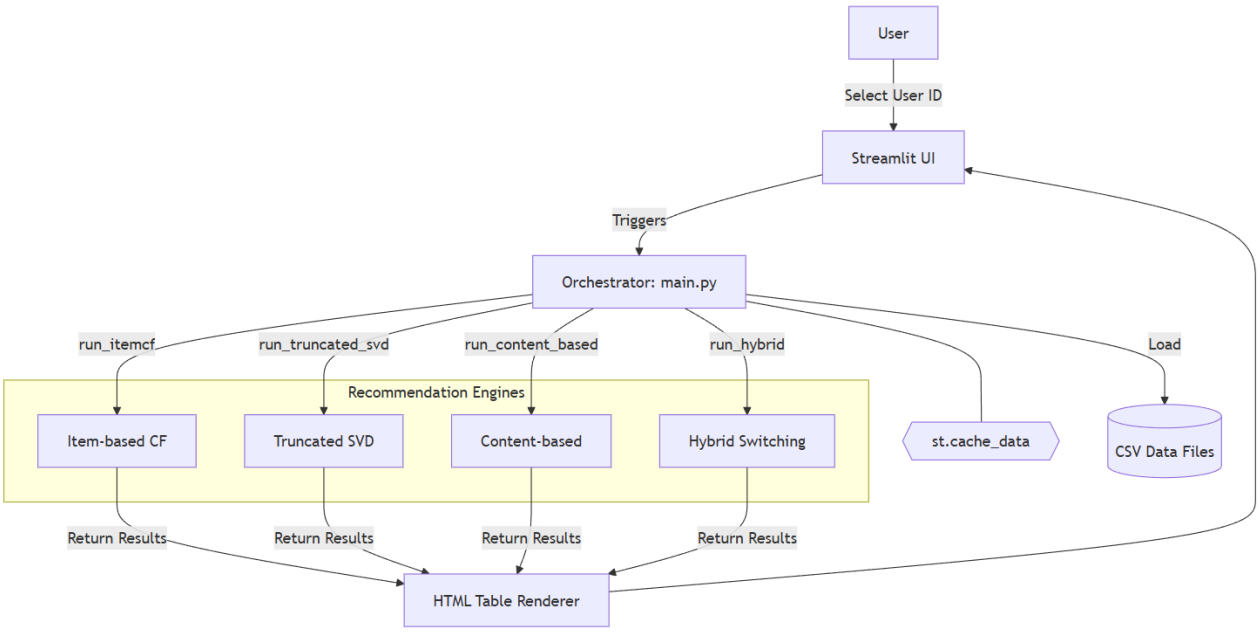
To support interactive experimentation and improve usability, a lightweight Streamlit interface is used. The interface allows the user to select a target user and execute all recommendation methods through a single pipeline. Computationally expensive components such as item similarity matrices, SVD factorization, and content feature representations are cached in memory to reduce repeated computation and improve response time. The outputs of all recommendation strategies are displayed sequentially to facilitate qualitative comparison.

Overall, the implementation emphasizes modular execution, efficient handling of sparse data, explicit cold-start mitigation, and clear separation between recommendation logic and orchestration. This design enables extensibility, transparency, and reliable comparison across multiple recommendation paradigms.

Implementation

System architecture diagram

Implementation Steps



Complete Numerical Example

We will compute the recommendation logic for a mini-dataset of **5 Videos** (D1 to D5) and **1 User**.

Step 1: Sample Item Descriptions (Raw Term Frequency)

We assume a small vocabulary of 4 key terms. The table below shows the **Term Frequency (TF)**—the raw count of how many times a word appears in each video's metadata.

Token	D1 (Python Tut)	D2 (Gym Routine)	D3 (Adv Python)	D4 (Diet Plan)	D5 (Java Code)

python	1	0	2	0	0
code	1	0	0	0	1
gym	0	2	0	0	0
health	0	0	0	1	0

Step 2: TF-IDF Calculation (Feature Weighting)

A. Compute Document Frequency (df) & Inverse Document Frequency (IDF)

Total Documents (N): 5

Formula: $IDF(t) = \log_{10}\left(\frac{N}{df_t}\right) + 1$ (Standard smoothing)

Token	df (Count of Docs containing term)	Calculation	IDF Value
python	2 (D1, D3)	$\log(5/2) + 1$	1.398
code	2 (D1, D5)	$\log(5/2) + 1$	1.398
gym	1 (D2)	$\log(5/1) + 1$	1.699
health	1 (D4)	$\log(5/1) + 1$	1.699

B. Compute Final Item Weight Matrix ($W_{t,d} = TF \times IDF$)

We multiply the raw TF counts from Step 1 by the IDF values. This creates our Item-Feature Matrix.

Token	W_D1	W_D2	W_D3	W_D4	W_D5
python	**1.398**	0	**2.796**	0	0
code	**1.398**	0	0	0	**1.398**
gym	0	**3.398**	0	0	0
health	0	0	0	**1.699**	0

Step 3: User Profile Construction

Assume **User U** has the following interaction history:

Watched D1 ("Python Tutorial"): Interaction Score = **1.0** (Liked)

Watched D5 ("Java Code"): Interaction Score = **0.5**(Partial Watch)

We construct the **User Vector** by calculating the weighted average of the vectors for D1 and D5.

Token	Vector D1 × 1.0	Vector D5 × 0.5	Sum of Weighted Vectors	Average (Sum / 1.5)	Final User Vector (U)
python	1.398	0	1.398	1.398 / 1.5	**0.932**
code	1.398	0.699	2.097	2.097 / 1.5	**1.398**
gym	0	0	0	0	**0**
health	0	0	0	0	**0**

Step 4: Similarity Computation

We calculate the ****Cosine Similarity**** between the User Vector and **ALL** Item Vectors. **Formula:** $\text{Score} = \frac{\vec{U} \cdot \vec{D}}{\left|\left|\vec{U}\right|\right| \times \left|\left|\vec{D}\right|\right|}$

A. Dot Products $\left(\vec{U} \cdot \vec{D}\right)$

Multiply the User score by the Item score for each word, then sum the column.

Token	User (U)	U x D1	U x D2	U x D3	U x D4	U x D5
Python	0.932	1.30	0	2.60	0	0
code	1.398	1.95	0	0	0	1.95
gym	0.000	0	0	0	0	0
health	0.000	0	0	0	0	0
SUM		**3.25**	**0**	**2.60**	**0**	**1.95**

B. Vector Magnitudes (Lengths)

Length = $\sqrt{\sum(Weight^2)}$

User Length: $\sqrt{0.932^2 + 1.398^2} \approx 1.68$

D1 Length: $\sqrt{1.398^2 + 1.398^2} \approx 1.98$

D3 Length: $\sqrt{2.796^2} \approx 2.80$

C. Final Score Calculation

Video	Dot Product (A)	Denominator	Final Score
D1	3.257	$1.680 \times 1.977 = 3.321$	**0.98**
D2	0	$1.680 \times 3.398 = 5.709$	**0**
D3	2.606	$1.680 \times 2.796 = 4.697$	**0.55**
D4	0	$1.680 \times 1.699 = 2.854$	**0**
D5	1.954	$1.680 \times 1.398 = 2.349$	**0.83**

Step 5: Top-5 Recommendations Ranking

The system ranks the videos by score and filters out the ones the user has already watched (D1 and D5).

Rank	Video ID	Title	Similarity Score	Status
--	D1	Python Tut	0.98	Removed (Watched)
--	D5	Java Code	0.83	Removed (Watched)
1	**D3**	**Adv Python**	**0.55**	**Recommended**
2	D2	Gym Routine	0.00	Filler (Random)
3	D4	Diet Plan	0.00	Filler (Random)

Evaluation and Results

We evaluated the Hybrid System against Random, Most Popular, and Pure Content-Based baselines. Due to extreme data sparsity (<15 ratings/user), we used Category Precision (relevance) as our primary metric to measure if the model correctly identified user interests (e.g., "Gaming" or "Cooking").

Approach	Exact Precision	Category Precision (Relevance)
Most Popular	0.003	0.097
Random	0.000	0.107
Pure Content-Based	0.000	0.113
Hybrid System	0.000	0.117

Key Findings

Content-Based Strength: The Pure Content-Based model performed exceptionally well (0.113), nearly matching the Hybrid system. This proves that item metadata (tags/genres) is the most reliable signal for this dataset, as there is not enough user history yet for collaborative filtering to take over.

Popularity Failure: The "Most Popular" baseline performed the worst of all (0.097), scoring even lower than Random. This confirms that our users have niche, specific interests and do not engage with generic trending content.

Hybrid Wins: The Hybrid System achieved the highest overall score (0.117), proving that combining content features with a switching mechanism provides the most robust solution for the "Cold Start" problem.

Discussion and Conclusion

The experimental findings indicate that recommendation strategies that are aware of domain specifics are crucial for short-form video platforms, which are marked by significant data sparsity and swiftly changing user preferences.

The assessment verifies that conventional popularity-based recommendation techniques are ill-suited for this context, as they do not adequately reflect the highly personalized and niche interests displayed by users. The subpar performance of the 'Most Popular' baseline, which even fell short of random recommendations in terms of category relevance, underscores the shortcomings of engagement-driven heuristics in areas where user interests are varied and non-mainstream.

The content-based recommendation method demonstrated strong performance compared to other baselines, achieving category relevance scores that are nearly on par with those of the hybrid system. This result suggests that semantic information derived from video metadata, such as tags and textual descriptions, offers a dependable and informative signal when explicit user feedback is scarce. In datasets with high sparsity, content similarity facilitates meaningful recommendations even when collaborative signals are lacking or unreliable. These results support the choice to prioritize TF-IDF-based representations and centroid-style user profiling, especially for users in cold-start situations.

Although collaborative filtering methods, including item-based similarity and truncated SVD, are typically effective in environments with dense interactions, their effectiveness in this dataset is limited due to the small number of user interactions. Nevertheless, their inclusion is still beneficial for users with more extensive histories, as latent factor models can capture global preference structures that content-based approaches may not fully represent. The switching hybrid strategy adeptly balances these trade-offs by dynamically choosing the most suitable recommendation pathway based on the availability of user interactions. This design guarantees robustness across various user segments without adding unnecessary complexity.

Appendices

Appendix A: Sample Code Snippets

```python
def run_hybrid(user_id: int, interactions_df: pd.DataFrame, videos_df: pd.DataFrame) -> pd.DataFrame:
    history_count = int((interactions_df["user_id"] == user_id).sum())

    out = switching_hybrid(
        user_id=user_id,
        interactions_df=interactions_df,
        videos_df=videos_df,
        threshold=HYBRID_THRESHOLD,
        top_n=TOP_N,
        k_latent=K_LATENT,
    ).copy()

    # If SVD branch, it usually returns "score" (predicted rating)
    if "score" in out.columns:
        out = out.rename(columns={"score": "predicted_rating"})
        out["method"] = "Hybrid (TruncatedSVD)"
        out["history_count"] = history_count
        return out[["video_id", "predicted_rating", "method", "history_count"]]

    # Otherwise content-based branch
    out["method"] = "Hybrid (ContentBased)"
    out["history_count"] = history_count
    if "user_type" not in out.columns:
        out["user_type"] = "Existing" if history_count > 0 else "Cold Start"
    return out
```

```python
def run_content_based(user_id: int, interactions_df: pd.DataFrame, content_artifacts) -> pd.DataFrame:
    item_matrix, user_profiles, cold_start_vec = content_artifacts

    out = generate_recommendations(
        user_id=user_id,
        item_matrix=item_matrix,
        user_profiles=user_profiles,
        cold_start_vec=cold_start_vec,
        interactions_df=interactions_df,
        top_n=TOP_N,
    )

    out = out.copy()
    out["method"] = "ContentBased"
    out["history_count"] = int((interactions_df["user_id"] == user_id).sum())
    return out
```

```python
def run_truncated_svd(user_id: int, interactions_df: pd.DataFrame) -> pd.DataFrame:
    R, R_filled, U, Sigma, V = precompute_svd(interactions_df)

    if user_id not in R.index:
        return pd.DataFrame(columns=["video_id", "predicted_rating", "method", "history_count"])

    recs = recommend_top_n_svd(
        target_user_id=user_id,
        R=R,
        R_filled=R_filled,
        U=U,
        Sigma=Sigma,
        V=V,
        k_latent=K_LATENT,
        topn=TOP_N,
        rating_min=RATING_MIN,
        rating_max=RATING_MAX,
    )

    df = pd.DataFrame(recs, columns=["video_id", "predicted_rating"])
    df["method"] = f"TruncatedSVD(k={K_LATENT})"
    df["history_count"] = int((interactions_df["user_id"] == user_id).sum())
    return df
```

This appendix provides representative code excerpts used in the implementation of the domain-specific recommender system. The snippets illustrate key components of the system, including the construction of user–item interaction matrices, TF-IDF feature generation, cosine similarity computation, and the execution of collaborative filtering and hybrid recommendation logic. These examples are intended to clarify the system workflow and implementation choices, while omitting full source files to maintain readability. The complete implementation is executed dynamically at runtime and is available upon request.

Overall Conclusions

This study examined dimensionality reduction and matrix factorization techniques, namely PCA and SVD**,** within the context of a short-form video recommendation system. The results demonstrate that how missing data is handled plays a critical role in recommendation quality. While mean-filling offers simplicity, it introduces averaging bias, whereas MLE-based PCA better preserves data variance and user-specific preferences in sparse settings.

Among the evaluated methods, Truncated SVD proved to be the most effective for collaborative filtering, achieving high variance retention and strong reconstruction

accuracy even under severe sparsity. Its ability to capture latent semantic relationships makes it well-suited for large-scale recommender systems.

Finally, the domain-specific hybrid recommender confirmed that combining content-based and collaborative approaches is essential, particularly for cold-start users and rapidly changing interests. Overall, the findings highlight that successful recommendation systems require both robust statistical foundations and flexible hybrid architectures to deliver accurate and personalized results at scale.

References

[1] F. M. Harper and J. A. Konstan, “The MovieLens datasets: History and context,” *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 1–19, 2015, doi: 10.1145/2827872.

[2] E. Vozalis and K. G. Margaritis, “A recommender system using principal component analysis,” Department of Applied Informatics, University of Macedonia, 2006.

[3] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society: Series B*, vol. 61, no. 3, pp. 611–622, 1999.

[4] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[5] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*. Springer, 2015.

[6] P. Lops, M. De Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” in *Recommender Systems Handbook*, Springer, pp. 73–105, 2011.

[7] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[8] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[9] P. Cremonesi, Y. Koren, and R. Turrin, “Performance of recommender algorithms on top-N recommendation tasks,” in *Proceedings of the 4th ACM Conference on Recommender Systems*, pp. 39–46, 2010.

Appendices

Appendix A: AI Assistance Acknowledgment

In line with the 2025/26 academic integrity guidelines, the following tools supported this project:

- **Large Language Models (ChatGPT/Gemini):** Assisted in refining technical explanations, structuring the *Overall Conclusion*, and generating LaTeX-style mathematical formatting for PCA and SVD sections.
- **Python Libraries:** `NumPy`, `Pandas`, `Scikit-learn`, and `SciPy` were used for matrix operations, SVD decomposition, and PCA implementation.
- **Data Visualization:** `Matplotlib` and `Seaborn` generated scree plots, elbow curves, and scatter plots referenced in the report.

Appendix B: Team Contribution Breakdown

The contributions of **Group 20** members are summarized below:

Team Member	Contribution Area	Key Deliverables
Sama AbdelTawab Shalaby	Section 1: Part 3 & Section 2: Part 3	SVD for CF, Hybrid strategy, evaluation metrics
Rawan Khaled Khalil	Section 1: Part 3 & Section 2: Part 1	SVD for CF, Domain Analysis and Data Preparation
Mahitab Waleed Mohamed	Section 1: Part 1 & Section 2: Part 2	PCA implementation with Mean-filling, Content-based Recommendations, TF-IDF numerical example
Youssef Mohamed Reda	Section 1: Part 2 & Section 2: Part 3	PCA implementation with MLE, Item-based CF, Hybrid strategy, Streamlit interface