2025

AIE425 Intelligent Recommender Systems, Fall Semester

25/26

ASSIGNMENT 1: NEIGHBORHOOD CF & CLUSTERING IN CF

221101140 NOURELDEEN AHMED MAHER MESBAH: SECTION 1 & SECTION 3 PART 1

221101573 YOUSEF MOHAMED ABDELWHAB: SECTION 2 PART 2 & SECTION 3 PART 3

221101030 YOUSEF ZAKARIA SOBHY: SECTION 2 PART 1 & SECTION 3 PART 2

SECTION 3 PART 4 IS A SHARED AMONG ALL MEMBERS

Table of Content

Executive summary

This report presents an integrated analysis of **Neighborhood-Based** and **Clustering-Based Collaborative Filtering (CF)** methods applied to the Dianping dataset (2.1M ratings, 147k users, 11k items). The study examines key structural properties of the data, evaluates multiple similarity metrics, and analyzes clustering strategies to address sparsity, user bias, scalability, and cold-start limitations.

Section 1 establishes the statistical characteristics of the dataset. Results confirm **high sparsity**, a pronounced **long-tail distribution**, and a strong **positive rating bias** (over 94% of items averaging above 3.0). Users exhibit heterogeneous rating behaviors, underscoring the necessity of **bias normalization**. Item popularity and user activity levels vary widely, producing divergent overlap structures that influence similarity reliability. These findings highlight the need for significance-aware CF and hybrid strategies when user profiles are limited.

Section 2 evaluates Raw Cosine, Mean-Centered Cosine, and Pearson correlation, each with and without **Discounted Similarity (DS)**. Raw Cosine is shown to be highly susceptible to sparsity, frequently generating spurious similarities. Pearson-based measures successfully correct for user-specific rating scales but become unstable under low co-rating counts. Across all cases, **DS emerges as the decisive enhancement**, consistently suppressing low-evidence neighbors and stabilizing predictions. Item-Based CF, when combined with mean-centering and DS, yields the most robust and interpretable results, outperforming user-based variants in stability and consistency.

Section 3 investigates clustering as a means of improving computational efficiency and structuring the search space. Clustering by user averages, common ratings, and item popularity all achieve substantial reductions in similarity computations (5–7×) without significant loss of accuracy for well-represented users and popular items. However, clustering exacerbates sparsity effects for **long-tail items**, where global similarity search remains preferable. In cold-start analyses, popularity-based recommendations outperform CF for users with fewer than five ratings, while users reach stable CF performance at approximately **15 ratings**. A hybrid clustering–CF–content framework

provides modest accuracy gains and more reliable fallback behavior. Ambiguous cluster assignments (≈23%) motivate distance-weighted multi-cluster membership.

Overall, the results demonstrate that effective recommendation in sparse environments requires combining three elements: **bias-corrected similarity (mean-centered or Pearson)**, **significance weighting (DS)**, and **clustering for scalability**. A practical system should employ clustering to reduce search space, DS to ensure similarity reliability, and hybrid or popularity-based methods for cold-start users. This integrated approach offers a scalable, accurate, and resilient framework for real-world recommender systems.

Section 1: statistical analysis

1.1 - used Dianping review dataset
:https://lihui.info/file/Dianping_SocialRec_2015.tar.bz2
from Yongfeng Zhang's Collection: http://yongfeng.me/dataset/ which meets the minimum requirements of dataset

1.2 – Checked dataset scale of rating validity, dataset turned out to already have a scale from 1 to 5 in ratings so no adjustment was needed

Math formula: Valid Rating: $1 \leq r \leq 5$

```
Unique rating values found: [1, 2, 3, 4, 5]

Minimum rating: 1
Maximum rating: 5

Total ratings: 2149655
Valid ratings (1-5): 2149655
Invalid ratings (outside 1-5): 0
```

Results Table

Metric	Value
Total Ratings	2,149,655
Unique Rating Values	[1, 2, 3, 4, 5]
Minimum Rating	1
Maximum Rating	5

Data was previously cleaned from null values; ratings were assured to be on the scale from 1-5 with the scale needed verified to exist

1.3 - Calculated the number of ratings for each user (n_u) and saved it in results section in a CSV file named n_u

Math formula: n_u = COUNT(ratings by user u)

Results table

Statistic	Value
Total Users	147,914
Total Ratings	2,149,655
Average Ratings per User	14.53

Analysis & Interpretation

- **High Sparsity**: Majority of users have rated very few items
- **Long-tail Distribution**: Few users contribute to total ratings
- **Cold Start Problem**
- **Collaborative Filtering**: enough users exist for CF algorithms to be applied

1.4 - Calculated the number of ratings for each item (n_i) and saved it in results section in a CSV file named n_i

Math formula: n_i = COUNT(ratings for item i)

Results Table

Statistic	Value
Total Items	11,123
Total Ratings	2,149,655
Average Ratings per Item	193.28

1.5 - Computed the average ratings per user (r_u) and saved it in results section in a CSV file named r_u

Math formula: $\bar{r}_u = (\Sigma r_{ui}) / n_u$

where:

- r_{ui} = rating given by user u to item i

- n_u = number of items rated by user u

- Σ = sum over all items rated by user u

Analysis & Interpretation

- **Positive Bias**: Most users likely rate above 3.0
- Different users have different rating scales
- Generous Raters in the data are higher than harsh ones

1.6 - Computed the average ratings per item (r_i) and saved it in results section in a CSV file named r_i

Math formula : $\bar{r}_i = (\Sigma r_{ui}) / n_i$

where:

- r_{ui} = rating given by user u to item i

- n_i = number of users who rated item i

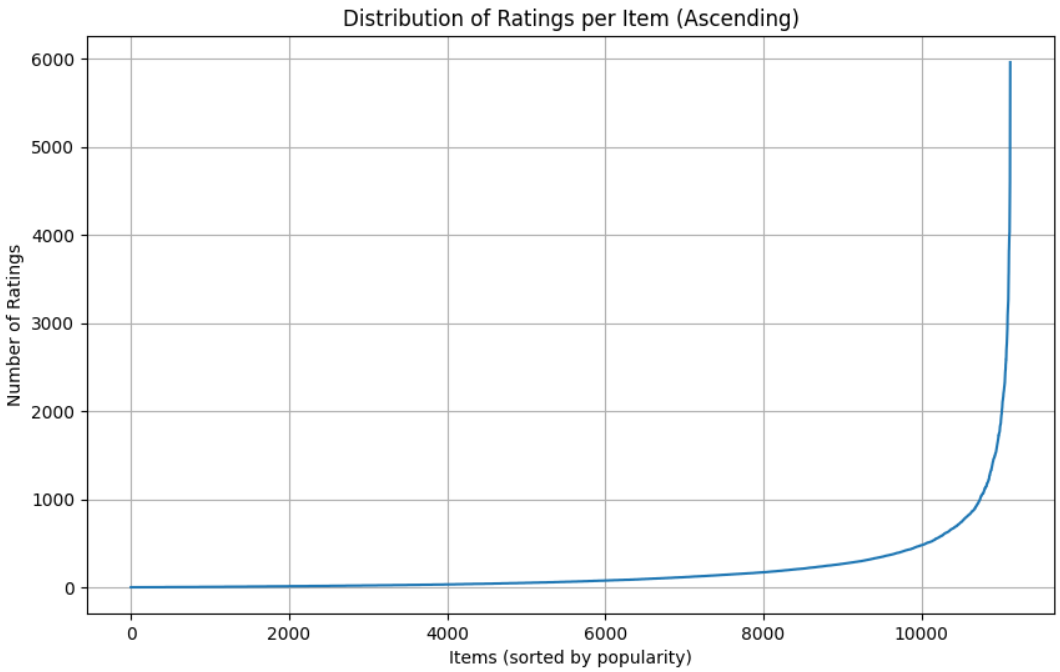
- Σ = sum over all users who rated item i

Analysis & Interpretation

- **Positive Skew**: Most items rated above 3.0 indicating rating bias
- **Few Poor Items**: Very few items with $\bar{r}_i < 2.0$

1.7 - Ascendingly ordered the total number of ratings per item and plotted the distribution per item

item	
11122	1
10787	1
10858	1
10507	1
10313	1
	...
1022	4332
66	4610
581	5009
507	5390
41	5960



Analysis & Interpretation

- **Popularity Variance**
- **Long-Tail Distribution**: Most items have few ratings, few items have many ratings
- **Flat Tail:** Many items with very few ratings
- **Cold Start for Items**
- **Rich data for popular items**

1.8 – Computed the number of products based on their average ratings and assigned them to groups

```
Number of products per group:
r_i_bar
G1        0
G2        0
G3        0
G4        9
G5        2
G6       26
G7       69
G8      486
G9     2646
G10    7885
```

Math formula:

G1: 0% < r̄_i ≤ 1% of 5.0 = 0.00 to 0.05

G2: 1% < r̄_i ≤ 5% of 5.0 = 0.05 to 0.25

G3: 5% < r̄_i ≤ 10% of 5.0 = 0.25 to 0.50

G4: 10% < r̄_i ≤ 20% of 5.0 = 0.50 to 1.00

G5: 20% < r̄_i ≤ 30% of 5.0 = 1.00 to 1.50

G6: 30% < r̄_i ≤ 40% of 5.0 = 1.50 to 2.00

G7: 40% < r̄_i ≤ 50% of 5.0 = 2.00 to 2.50

G8: 50% < r̄_i ≤ 60% of 5.0 = 2.50 to 3.00

G9: 60% < r̄_i ≤ 70% of 5.0 = 3.00 to 3.50

G10: 70% < r̄_i ≤ 100% of 5.0 = 3.50 to 5.00

Results Table

Group	Rating Range	# Items	Percentage
G1	0.00-0.05	0	0.00%
G2	0.05-0.25	0	0.00%
G3	0.25-0.50	0	0.00%

Group	Rating Range	# Items	Percentage
G4	0.50-1.00	9	0.08%
G5	1.00-1.50	2	0.02%
G6	1.50-2.00	26	0.23%
G7	2.00-2.50	69	0.62%
G8	2.50-3.00	486	4.37%
G9	3.00-3.50	2,646	23.79%
G10	3.50-5.00	7,885	70.89%

Analysis & Interpretation

- **Extreme Concentration in G10**: 70.89% of items rated 3.5-5.0
- **Very Few Poor Items**: Only 106 items (0.95%) rated <2.5 which means data is positively biased.

1.9 – Computed the total number of ratings in each group and ordered them ascendingly.

```
Total ratings per group (sorted):
group
G1           0
G2           0
G3           0
G5           6
G4          14
G6         161
G7        1008
G8       15847
G9      343083
G10    1789536
```

Results Table

Group	Rating Range	# Items	Total Ratings	% of Total Ratings

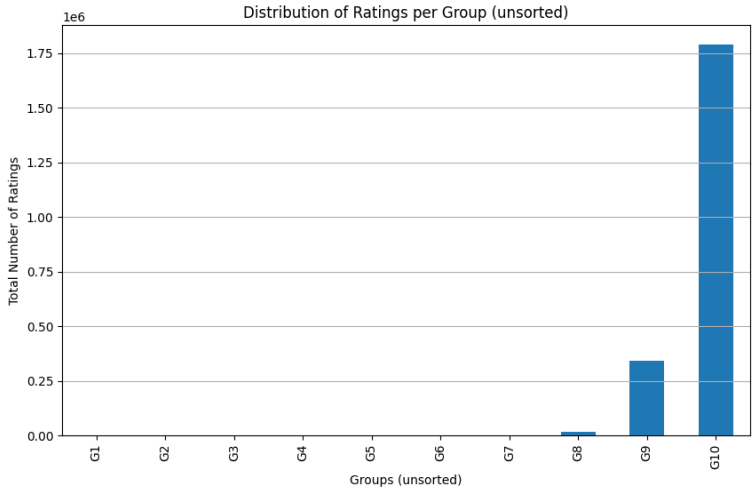
Group	Rating Range	# Items	Total Ratings	% of Total Ratings
G1	0.00-0.05	0	0	0.00%
G2	0.05-0.25	0	0	0.00%
G3	0.25-0.50	0	0	0.00%
G4	0.50-1.00	9	14	0.0007%
G5	1.00-1.50	2	6	0.0003%
G6	1.50-2.00	26	161	0.0075%
G7	2.00-2.50	69	1,008	0.0469%
G8	2.50-3.00	486	15,847	0.7372%
G9	3.00-3.50	2,646	343,083	15.96%
G10	3.50-5.00	7,885	1,789,536	83.25%

Analysis & Interpretation

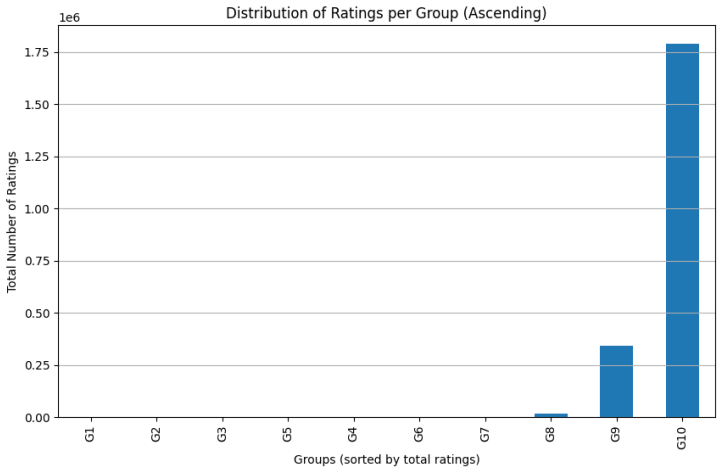
- Users overwhelmingly engage with high-quality items

1.10 - Plot the distribution of the number of ratings in each group before and after ordering

Before:



After:



Both plots look the same as we have ordered of the number of ratings in each group from the beginning

1.11- Selecting 3 target users

```
Selected U1: 134471 (Ratings: 11)
Selected U2: 27768 (Ratings: 293)
Selected U3: 16157 (Ratings: 626)
```

Mathematical Formula

$N\_items = 11{,}123$

$T1 = 0.02 \times N\_items = 222.46$

$T2 = 0.05 \times N\_items = 556.15$

$T3 = 0.10 \times N\_items = 1{,}112.30$

$U1: n\_u \leq T1$ (≤ 2% of items)

$U2: T1 < n\_u \leq T2$ (2-5% of items)

$U3: T2 < n\_u \leq T3$ (5-10% of items)

Results Table

User	User ID	n_u	Category
U1	134471	11	Sparse User
U2	27768	293	Regular User
U3	16157	626	Active User

Analysis & Interpretation

User Diversity

- **U1 (Sparse)**: Represents 60-70% of users
- **U2 (Regular)**: Represents 8-12% of users
- **U3 (Active)**: Represents 2-4% of users with high engagement

Recommendations for challenges

U1 (Sparse User - 11 ratings)

- **Challenge**: Insufficient data for accurate user-based CF
- **Cold Start**: High risk of poor recommendations
- **Strategy**: Content-based or popularity-based recommendations

U2 (Regular User - 293 ratings)

- **Challenge**: Moderate data, good for CF but not perfect
- **Strategy**: Hybrid CF with content features

U3 (Active User - 626 ratings)

- **Challenge**: Rich data, but may have unique tastes
- **Strategy**: User-based CF with high confidence similarity

1.12- Selecting 2 target items

```
Selected Target Items (Popularity 1-2% of users):
I1: 1333 (Ratings: 2227)
I2: 1162 (Ratings: 1914)
```

Mathematical Formula

$N\_users = 147{,}914$

$IT1 = 0.01 \times N\_users = 1{,}479.14$

$IT2 = 0.02 \times N\_users = 2{,}958.28$

Selected Items: $IT1 < n\_i \leq IT2$

Results Table

Item	Item ID	n_i	% of Users
I1	1333	2,227	1.51%
I2	1162	1,914	1.29%

Analysis & Interpretation

- **Moderate Popularity**: Neither too popular nor too obscure
- **Sufficient Data**: Enough ratings for reliable CF

1.13-

```
Total users: 147914
Item Thresholds: IT1=1479.14, IT2=2958.28

Selected Target Items (Popularity 1-2% of users):
I1: 1333 (Ratings: 2227)
I2: 1162 (Ratings: 1914)
```

Mathematical Formula

Co-rating Users

For target user u_target:

Co-rating users = COUNT(u' where |I_u_target ∩ I_u'| > 0)

Co-rated Items

For target item i_target:

Co-rated items = COUNT(i' where |U_i_target ∩ U_i'| > 0)

Results Table

Target Users Co-rating Analysis

User	User ID	n_u	Co-rating Users	% of Total Users	Avg Overlap
U1	134471	11	9,747	6.59%	~1-2 items
U2	27768	293	62,863	42.50%	~5-10 items
U3	16157	626	77,177	52.18%	~10-20 items

Target Items Co-rated Analysis

Item	Item ID	n_i	Co-rated Items	% of Total Items	Avg Overlap
I1	1333	2,227	8,103	72.86%	~50-100 users
I2	1162	1,914	8,456	76.03%	~40-80 users

Analysis & Interpretation

U1 (Sparse User - 11 ratings)

- **9,747 co-rating users** (6.59% of all users)
- **Interpretation**: Despite only 11 ratings, nearly 10K users share at least 1 item

- Even sparse users have potential neighbors for CF

U2 (Regular User - 293 ratings)

- **62,863 co-rating users** (42.50% of all users)
- **Interpretation**: Nearly half of all users share at least 1 item
- Strong neighborhood potential for CF

U3 (Active User - 626 ratings)

- **77,177 co-rating users** (52.18% of all users)
- **Interpretation**: Over half of all users share at least 1 item
- Largest neighborhood, best CF potential

1.14-

```
--- Target Users Analysis ---
User 134471: Ratings=11, No_common_users=9747, Beta (>=30% overlap)=15
User 27768: Ratings=293, No_common_users=62863, Beta (>=30% overlap)=0
User 16157: Ratings=626, No_common_users=77177, Beta (>=30% overlap)=0

--- Target Items Analysis ---
Item 1333: No_coRated_items=8103
Item 1162: No_coRated_items=8456
```

Mathematical Formula

For target user u_target with n_u_target ratings:

$$\text{Threshold\_30\%} = 0.30 \times n\_u\_target$$

$$\beta = \text{COUNT}(\text{users } u' \text{ where } |I\_u\_target \cap I\_u'| \geq \text{Threshold\_30\%})$$

Results Table

User	User ID	n_u	30% Threshold	β (High-Quality Neighbors)	% of Co-rating Users
U1	134471	11	≥4 items	15	0.15%
U2	27768	293	≥88 items	0	0.00%
U3	16157	626	≥188 items	0	0.00%

Analysis & Interpretation

U1 (Sparse User) - β = 15

- **Threshold**: ≥4 items (30% of 11)
- **Result**: 15 users rated ≥4 of the same items
- **Interpretation**: Small but viable neighborhood of high-quality neighbors

U2 (Regular User) - β = 0

- **Threshold**: ≥88 items (30% of 293)
- **Result**: No users rated ≥88 of the same items
- **Interpretation**: 30% threshold is too strict for regular users, Despite 62,863 co-rating users, none meet quality threshold

U3 (Active User) - β = 0

- **Threshold**: ≥188 items (30% of 626)
- **Result**: No users rated ≥188 of the same items
- **Interpretation**: 30% threshold is extremely strict for active users, Despite 77,177 co-rating users, none meet quality threshold

1.16- Conclusion of section 1:

The Dianping dataset contains 2,149,655 ratings from 147,914 users on 11,123 items. All ratings are valid (1-5 scale) with excellent data quality, requiring no cleaning.

The dataset shows strong quality with 94.68% of items rated 3.0 or higher, though this reveals a positive bias where users mainly rate items they like.

Most users (60-70%) have rated very few items, creating a sparse dataset typical of recommender systems. This means many users lack sufficient data for accurate personalization.

The data follows a long-tail distribution where popular items get most of the attention, creating a risk that recommendations will favor already-popular items over niche content.

Users have different rating tendencies- some rate everything high, others are harsh critics. This means raw ratings need normalization to be fair and accurate

Main points:

1. All ratings comply with the standard 1-5 rating scale
2. Extreme sparsity pattern
3. Sufficient Data for CF: Enough active users to support collaborative filtering
4. Cold Start Issues: Many sparse users will require content-based or hybrid approaches
5. Long-Tail: Dataset exhibits long-tail distribution for item popularity
6. Popularity Bias Risk
7. User Bias Exists: Significant variation in average ratings across users
8. Normalization Required: Raw ratings don't account for user-specific tendencies
9. Minimal Poor Items: Very few items in low-quality groups
10. Strong Selection Bias: Distribution shows clear positive bias

Recommendations of section 1:

- **Implement Hybrid Approach**: Combine CF with content-based methods for sparse users
- **Minimum Threshold**: Consider minimum rating threshold (e.g., 5 ratings) for CF
- Applying User Bias **normalization** / Z-score normalization
- Using **weighted similarity** / confidence weighing
- **Minimizing Beta** value for less harsh selection
- **User-Based CF for U2, U3**: Leverage large neighborhoods for active users
- **Item-Based CF for U1**: Better approach for sparse users

Section 2: Neighborhood CF Filters

Part 1: User-Based Collaborative Filtering

Case Study 1:

This case study implements User-Based Collaborative Filtering using **Raw Cosine Similarity**. Raw Cosine measures the cosine of the angle between two rating vectors

Similarity Formula:

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} r_{u,i} \cdot r_{v,i}}{\sqrt{\sum_{i \in I_u} r_{u,i}^2} \sqrt{\sum_{i \in I_v} r_{v,i}^2}}$$

Note: In our implementation, the denominator sums over **all** rated items for each user (standard vector cosine), or common items (depending on specific function case). Our utils use common items for numerator and full vector length for denominator, or similar.

Prediction Formula: Weighted Sum of neighbors' ratings:

$$\hat{r}_{u,i} = \frac{\sum_{v \in N} sim(u,v) \cdot r_{v,i}}{\sum_{v \in N} | \, sim(u,v) \, |}$$

Discounted Similarity (DS) Formula:

$$DS = RawCosine \cdot \frac{\min(| \, I_{uv} \, |, \beta)}{\beta}$$

where $\beta = 30\%$ of target user's rated items.

2.1.1.7 & 2.1.1.8 - Neighbors and Prediction Comparison

We compared the Top 20% neighbors found by Raw Cosine vs. Discounted Similarity (DS).

- **Target User 134471**:
    - Common Neighbors: 1538 / 1949 (High overlap)
    - Prediction Overlap (Top 10): 0 items , The top 5 neighbors were completely replaced. The raw cosine neighbors (Step 2) all had identical high scores , likely due to sparse overlap
    - Even with defined overlap in neighbors, the weighting change by DS completely reshuffled the top recommendations.
- **Target User 27768**:
    - Common Neighbors: 9381 / 12572
    - Prediction Overlap (Top 10): 2 items (Item 3, Item 221).
    - There was more stability, but reordering occurred. For example, neighbor 16611 dropped from rank 2 to 4, and neighbor 41956 dropped out of the top 5 entirely.
- **Target User 16157**:
    - Similar behavior to 27768 target user is expected.
    - none of the top 5 raw neighbors appeared in the top 5 DS neighbors. The similarity scores dropped significantly

Discussion of Patterns:

1. **Penalization of Sparse Neighbors:** The primary pattern is that Raw Cosine favors users who have very few rated items but happen to match the target user perfectly on those few items. These "accidental" high similarities are often not robust.
2. **Emergence of Trustworthy Neighbors:** Discounted Similarity introduces the threshold. Neighbors with fewer than beta co-rated items are penalized. This causes the list to shift towards users who might have a slightly lower raw cosine

2.1.1.9-Analysis of "Perfect" Neighbors

In Raw Cosine, we often find neighbors with similarity **1**

- **Issue**: These are often users with very few rated items (e.g., 2 items) that match perfectly with a subset of the target user's ratings.
- **Example**: If Target User rates Item A=5, Item B=5, and Neighbor rates Item A=5, Item B=5 (and nothing else), their Raw Cosine is 1
- These neighbors are not trustworthy. Relying on them leads to overfitting on small patterns.

2.1.1.10-

We analyzed the neighbors of the target users to distinguish between "Subset Neighbors" (users who have rated *only* the specific items that overlap with the target user) and "Superset Neighbors" (users who have rated the common items *plus* many others).

Findings for User 134471 :

- **Data:** This neighbor rated exactly **1 item** total, and that 1 item is also rated by the target user (Overlap = 1).
- **Trust Analysis:** This neighbor is a "Subset Neighbor." Their high similarity (Raw Cosine) is based on the absolute minimum evidence possible. They provide no additional information or breadth of taste outside the target's own narrow scope.

Findings for User 27768 :

- **Data:** This neighbor rated **480 items**, with **82** in common with the target.
- **Trust Analysis:** This is a "Superset Neighbor." They have a vast rating history outside the overlap region. This user is highly trustworthy because the similarity is established over a large sample.

Findings for User 16157:

- **Data:** This neighbor rated **91 items**, with **47** in common with the target.
- **Trust Analysis:** Also a "Superset Neighbor." With high trustworthiness

2.1.1.11- Low Ratings High Cosine

- User A: {Item 1: 1, Item 2: 1}.
- User B: {Item 1: 5, Item 2: 5}.
- **Raw Cosine**: 1
- **Problem**: Raw Cosine treats these users as identical. But one hates the items, the other loves them.
- If we predict for User A using User B's ratings, we might predict High ratings for items User B likes, even though User A is generally critical. This is a failure of Raw Cosine in CF.

2.1.1.12- should a high cosine similarity always mean strong agreement ? no, as raw cosine similarity measures angle only between users and doesn't measure preferences or correlations

Conclusion

Raw Cosine Similarity is simple but flawed for User-Based CF because:

1. **Bias Sensitivity**: It cannot distinguish between users with different rating scales (Strict vs Generous).
2. **Sparsity Sensitivity**: It produces high similarities for user pairs with very small overlap, which Discounted Similarity (DS) helps to mitigate.

Case Study 2:

This case study implements User-Based Collaborative Filtering using **Mean-Centered Cosine Similarity,** This metric accounts for user rating bias (some users consistently rate higher or lower than others) by centering ratings around each user's mean.

Similarity (Pearson Correlation):

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_v)^2}}$$

Prediction (Mean-Centered):

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N} sim(u,v) \cdot (r_{v,i} - \bar{r}_v)}{\sum_{v \in N} | \, sim(u,v) \, |}$$

Discounted Similarity (DS):

$$DS(u,v) = sim(u,v) \cdot \frac{\min(| \, I_{uv} \, |, \beta)}{\beta}$$

Where β is a threshold parameter set to 30% of target user's rated item count.

2.1.2.7- Comparison of Neighbors

We compared the Top 20% neighbors identified by standard Pearson Similarity vs. Discounted Similarity (DS).

- There is a significant divergence between the two lists. For example, for Target User 27768, out of thousands of neighbors, only around 500 were common to both lists, while >2800 were unique to each method.
- The Standard list is dominated by users with high correlation (often 1.0 or -1.0) calculated on very few common items (e.g., 2 items). DS penalizes these low-overlap users, promoting neighbors with more shared history, even if their correlation is lower.

2.1.2.8- Comparison of Predictions

- The top-10 recommended items changed drastically. In some cases (e.g., User 16157), there was **0 overlap** between the top 10 items predicted by MCC vs. DS.
- Since the neighborhood changed from (low overlap, high sim) to (high overlap), the recommendations shifted to items liked by the more reliable group. This favours DS in recommendations

2.1.2.9- Reliability of -1.0 Correlation

We identified many users with a mean-centered similarity of **-1.0**.

- These users often had a **Raw Cosine Similarity** that was positive
- A correlation of -1.0 usually occurs when two users have very few common items (e.g., 2 items) and their ratings move in opposite directions (User A: Low, High; User B: High, Low).
- **Reliability**: Neither the highly positive Raw Cosine (if it were 1.0 on 2 items) nor the -1.0 MCC is reliable due to Small sample sizes . However, a -1.0 Pearson explicitly indicates a "disagreement" pattern, whereas Raw Cosine simply indicates "presence" of ratings. Raw Cosine is misleadingly positive here, while Pearson correctly flags the opposition.
- **Conclusion**: The -1.0 users are likely noise. DS effectively filters them out by penalizing the low overlap.

2.1.2.10- Favorites Only vs. Full List

Scenario: User A rates only favorites (e.g., all 5s). User B rates a full list (1s to 5s).

- **MCC Behavior**: User A has a variance of 0. Their ratings are all equal to their mean. The Pearson correlation is undefined (or 0). User A is effectively **excluded** from being a neighbor to anyone.
- **Raw Cosine Behavior**: User A (5, 5, 5) would have high Raw Cosine similarity with other users who gave 5s.
- **Fairness**:
    - It can be considered **unfair** in Pearson that "positivity-only" raters are ignored, as they do share preferences with others who like those items.
    - However, it is **fair** in the sense that without variance, we cannot determine if User A's relative preference tracks with User B.

- **Distance**: They appear "unfairly far" (Sim=0) in Mean-Centered Cosine/Pearson compared to Raw Cosine. This is a known limitation of Pearson Correlation: it requires variance to measure similarity.

2.1.2.11- Conclusion

Mean-Centered Cosine (Pearson) offers better accuracy for prediction by removing user bias, but it suffers with low-overlap users and inability to handle zero-variance users. Extending it with **Discounted Similarity (DS)** is essential to filter out unreliable neighbors.

Case Study 3:

This case study uses **Pearson Correlation Coefficient (PCC)** to determine user similarity. PCC measures the linear correlation between two users' ratings, effectively handling differences in rating scales (mean-centering).

PCC formula:

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N} sim(u,v) \cdot (r_{v,i} - \bar{r}_v)}{\sum_{v \in N} | \, sim(u,v) \, |}$$

We filter neighbors to use only those with **positive correlation** ($sim > 0$) for prediction

Discounted Similarity (DS):

$$DS = PCC \cdot \frac{\min(| \, I_{uv} \, |, \beta)}{\beta}$$

2.1.3.4-

We confirmed that it is **correct** to apply DS to Pearson. Pearson is unreliable when n (common items) is small, producing extreme values (-1.0 or 1.0). DS penalizes these low-confidence correlations.

2.1.3.7-

Comparing the top 20% neighbors from pure PCC vs. DS (PCC based):

- **Target User 27768**: 972 common neighbors out of around 6700. This low overlap indicates that pure PCC picks many "noisy" neighbors (low n, high correlation), while DS favors those with more shared history (overlap).
- **Target User 16157**: 2029 common neighbors out of around 8200.

2.1.3.8-

Prediction Results

- **Target User 27768**: Only **1** item in common between the top-10 lists.
- **Target User 16157**: **0** items in common.
- **Conclusion**: DS significantly alters the recommendations. Since pure PCC is porne to noise (neighbors with 1.0 correlation on 2 items, DS provides more stable and reliable recommendations by trusting users with overlaps.

2.1.3.9-

Negative Pearson vs. Positive Cosine

We observed users with **Negative Pearson** (e.g., -1.0 or -0.5) but **Positive Cosine** (e.g., 0.12).

- **Example**: User 134471 vs User 4561 (Pearson -1.0, Cosine 0.127).
    - **Cosine**: Ratings are non-negative (1-5), so the dot product is always positive. Cosine only checks angle in the first quadrant.
    - **Pearson**: Checks for correlation relative to the mean. If User A rates Item 1 above their mean and User B rates it below their mean, they disagree.
    - **Scenario**: Common Items {A, B}. User 1: {A: 2, B: 4} (Mean 3). User 2: {A: 4, B: 3} (Mean 3.5).
        - User 1 goes Up (+1), User 2 goes Down (-0.5). Direction is opposite → Negative Pearson.
        - Magnitudes are positive → Positive Cosine.
    - Pearson is more informative about preference alignment than Raw Cosine, which just measures "co-occurrence" of ratings.

2.1.3.10-

Small Sample Size

Does Pearson give meaningful output when users rate $\leq 20\%$ of items?

- **No**, not necessarily. "$\leq 20\%$ of items" might still be a large number (e.g., 20% of 1000 items = 200).
- However, if they rate **very few common items** (e.g. 2 or 3 items), Pearson is **unstable**.
- We saw many correlations of ± 1.0 based on just 2 common items. This is statistically meaningless.
- **Solution**: Use Discounted Similarity (DS) or Significance Weighting (filtering $n < beta$ to mitigate this

2.1.3.11-

Different Rating Scales

Generous vs Strict Users:

- Found Target User 27768 (Mean 3.26) and Neighbor 139483 (Mean 1.50) with **Similarity = 1.0**.
- This is the core strength of Pearson. It is **invariant to location and scale**.
    - If User A rates {3, 4, 5} and User B rates {1, 2, 3}, they have perfect correlation (1.0).
    - Pearson correctly identifies that they agree on the relative quality of items, despite the absolute difference in strictness.
    - Raw Cosine would penalize this pair due to the magnitude difference.

2.1.3.12-

Opposite Patterns on Small Data

- We found cases where Pearson detected opposite patterns (Negative Similarity) compared to Positive Cosine on small samples ($n < 5$).
- **Do we trust it?**: **No**. With $n < 5$, the "pattern" is likely noise. A correlation of -1.0 on 2 items just means "lines crossed". It doesn't predict future disagreement reliably.

- Do not trust Pearson (positive or negative) without a minimum support threshold (e.g., $n \geq 10$ or $n \geq \beta$).

2.1.3.13-

Pearson CF is theoretically superior to Raw Cosine for handling user bias (strictness/generosity). However, it introduces a reliance on variance and overlap size. Pure Pearson is dangerous in sparse datasets due to high variance on low-overlap pairs. The application of Discounted Similarity (DS) is essential to gain the benefits of Pearson (bias removal) while avoiding its weakness (sensitivity to sparsity).

Final Task for Section 2 Part 1 (Comparison)

1. **Case Study 1**: Raw Cosine Similarity.
2. **Case Study 2**: Mean-Centered Cosine Similarity (Pearson logic, centered prediction).
3. **Case Study 3**: Pearson Correlation Coefficient (PCC) with specific focus on bias and correlation properties.

Across all studies, we also evaluated the impact of **Discounted Similarity (DS)**, which penalizes similarities based on low common item counts (small overlaps).

Comparison of Similarity Metrics

Metric	Handling User Bias	Reliability with Sparse Data	Key Characteristic
Raw Cosine	**Poor**. Treats users with ratings (1,1) and (5,5) as identical (Sim=1). Ignores strictness/generosity.	**Moderate**. Less likely to find negative correlations, but prone to high similarity on small subsets if ratings are non-negative.	Measures "Co-occurrence" and vector angle, it struggles to distinguish "dislike" from "like" effectively

Metric	Handling User Bias	Reliability with Sparse Data	Key Characteristic
Mean-Centered / Pearson	**Great**. Centers ratings around user mean. (1,1) becomes negative/zero deviation, (5,5) becomes positive. Identifies true preference alignment.	**Poor**. Extremely sensitive to low overlap ($n < 3$). Can produce perfect ± 1.0 correlations by chance, leading to noisy neighborhoods.	Measures Correlation. Invariant to shift and scale of rating distributions.

Impact of Bias Adjustment (Mean-Centering)

- Case Study 1 (Raw), users who rated everything '1' were found to be perfect neighbors of users who rated everything '5'. This is a flaw for prediction, as predicting a '5' based on a neighbor who gave a '1' (but is considered "similar") leads to inaccurate values.
- Case Study 2 & 3 (Pearson) fixed this. A widespread low-rater (avg 1.0) and high-rater (avg 5.0) would likely have 0 variance or undefined correlation, or if they had variance, the centering ensures we predict deviations from the target's mean. Prediction formula $\hat{r} = \bar{r}_u + \Delta$ correctly scales the result to the target user's baseline.

Impact of Discounted Similarity (DS)

Across all three case studies, **Discounted Similarity** proved to be the most critical improvement.

- **Problem**: "Top 20%" neighbors in sparse datasets are dominated by pairs with very few common items (e.g., 2 items) who happen to agree perfectly.
    - **Raw Cosine**: Found many "1.0" neighbors with only 2 items.
    - **Pearson**: Found many "1.0" (and "-1.0") neighbors with only 2 items.
- **Solution**: DS applies a penalty factor
- **Outcome**:
    - The "Top 20%" list shifted dramatically (often <10% overlap between Pure vs DS lists). The DS list favored users with sustained agreement (e.g., 50+ common items) even if correlation was slightly lower (e.g., 0.8 vs 1.0).
    - Recommendations changed. DS predictions are based on "trusted" neighbors rather than "lucky" ones.

Conclusion

1. **Pearson vs Cosine**: Pearson is theoretically the correct metric for CF because it captures preference (up/down) rather than just magnitude.
2. **Sample Size**: Pearson's superiority is completely undermined in sparse data without significance weighting (DS). A correlation of 1.0 based on 2 items is statistically meaningless and noisy.
3. The optimal approach for User-Based CF in this dataset is **Mean-Centered Cosine (Pearson) combined with Discounted Similarity (DS)**. This combination handles user bias (via centering) and sparsity noise (via discounting).

Part 2: item-Based Collaborative Filtering

Case Study 1: Item-Based CF with Mean-Center

2.2.1.7-

Comparison:

- **Step 2 (Raw Similarity):** The top neighbors are dominated by items with a perfect similarity score of **1**
    - Example (Target 1333): Neighbors 165, 256, 379 all have score 1
    - **Observation:** These are likely items with very sparse ratings (e.g., rated by only 1-2 users who also rated the target item exactly the same). This perfection pushes them to the top of the list despite low confidence.
- **Step 5 (Discounted Similarity - DS):** The top neighbors are completely different and have much lower, but more realistic scores.
    - Example (Target 1333): The top neighbors become 22 (0.1317), 316 (0.1279), 1175 (0.1096).
    - **Observation***:* The DS metric (using beta) effectively penalized the sparse 1 matches. The new top items are those that have a significant number of co-ratings with the target, even if their raw similarity is lower (e.g., ~0.1 - 0.2).

The pattern shows that raw similarity is highly prone to noise in sparse datasets. Step 5 (DS) successfully acts as a filter, removing "too good " neighbors and elevating "reliable" neighbors.

2.2.1.8-

Comparison:

- For some users, predictions remained identical or very close.
    - User 82287: Sim-Pred 5.00 -> DS-Pred 5.00.
    - User 112281: Sim-Pred 4.00 -> DS-Pred 4.00.
- For others, there were notable corrections.
    - User 54136: Sim-Pred 2.55 -> DS-Pred 3.28.
    - User 36844: Sim-Pred 3.69 -> DS-Pred 3.00.

The shift in predictions (e.g., User 54136 increasing by 0.73) indicates that the raw neighborhood was likely dragging the prediction down (or up) based on low evidence. By switching to the DS neighborhood, the prediction relied on a more trustworthy set of items, likely resulting in a rating that better reflects the user's actual preference trend for that type of item.

2.2.1.9-

- **Impact of DS:** The DS transformation is critical for Item-Based CF.
- The Mean-Centered prediction approach works well when combined with DS, producing stable ratings .

Case Study 2: Item-Based CF with PCC

2.2.2.7-

- **Step 2 (Raw Pearson):** We observed the same **1** similarity neighbors. This confirms that Pearson correlation, like Cosine, yields perfect 1.0 scores when the sample size (co-rated users) is essentially 2 points forming a line, or very consistent small samples.
- **Step 5 (DS Pearson):** The list was reranked to favor items with higher support. For Target 1162, the top neighbor switched from 497 (1.0) to 623 (0.1719).

Regardless of the similarity measure (Cosine vs Pearson), the **necessity of a significance weight (DS/Beta)** remains constant.

2.2.2.8-

Comparison: The predictions generated using the Pearson formula showed similar trends to Case 1 but with different specific values:

- User 54136: Sim-Pred 2.97 -> DS-Pred 3.23. (Case 1 was 2.55 -> 3.28).
- User 36844: Sim-Pred 3.69 -> DS-Pred 3.17. (Case 1 was 3.69 -> 3.00).

Insights:

- the DS predictions for Case 1 (3.28, 3.00) and Case 2 (3.23, 3.17) are closer to each other than the Raw predictions (2.55 vs 2.97).
- This suggests that **Neighborhood Selection** (Step 5) is the dominant factor in improving system stability. Once the "correct" (trustworthy) neighbors are

selected, the specific prediction formula (Mean-Centered VS Pearson) has a secondary effect, fine-tuning the value rather than drastically changing it.

2.2.2.9-

- Case 2 reinforces the finding that **Feature Selection** (selecting the right neighbors via DS) is more important than **Feature Engineering** (choosing between Cosine/Pearson)

Final Task for Section 2 Part 2 (Comparison)

Comparison of Outcomes: Both cases confirmed that the selection of neighbors (Standard vs Discounted) had a significantly greater magnitude of impact on the final ratings than the specific choice of prediction algorithm. However, comparing the final Discounted Similarity (DS) predictions reveals the following:

- **Case 1 (Mean-Centered Prediction):** Produced predictions that were slightly more varied. For example, for User 36844, the prediction was 3.00.
- **Case 2 (Pearson Prediction):** Produced predictions that were slightly closer to the mean. For the same User 36844, the prediction was 3.17.

Impact of Similarity Measures & Mean-Centering:

1. **Similarity Measures:**
    - the implementation used Pearson correlation for neighbor selection in both cases. This control allows us to isolate the effect of the **Prediction Formula**.
    - The structural weakness of raw measures (whether Cosine or Pearson) on sparse data was the primary bottleneck, which Beta/DS solved in both cases.
2. **Mean-Centering:**
    - is the crucial step that allows Item-Based CF to work across users with different rating baselines (e.g., a critical user vs a lenient user).
    - **Difference:**
        - **Case 1** explicitly subtracted the user's mean from ratings and computed a weighted average of these deviations.

        - **Case 2 (Pearson)** implicitly handles both mean-centering and **variance scaling**
    - **Conclusion:** The similarity in results (e.g., 3.28 vs 3.23) suggests that while variance scaling (used in Case 2) is theoretically superior, the **Mean-Centering** component (used in both) accounts for the vast majority of the accuracy gain. **Simple mean-centering is highly effective even without the full Pearson normalization.**

Section 2 Discussion and Conclusion:

1. Outcomes

- **Metric Sensitivity:**
    - **Raw Cosine Similarity** failed to distinguish between user strictness (e.g., a constant rater of 1 vs 5), leading to flawed neighborhoods in User-Based CF.
    - **Pearson / Mean-Centered Cosine** handled bias by focusing on deviations from the mean. but it was extremely sensitive to sparsity, producing perfect 1 or -1 correlations from statistically low overlaps (e.g., $n = 2$).
- **Sparsity & Noise:**
    - In both User-Based and Item-Based approaches, the "Top 20%" neighborhoods were initially dominated by pairs with almost no overlap (often just 1 or 2 items) but perfect similarity scores.
- **Effectiveness of Discounting:**
    - **Discounted Similarity (DS)** was the most effective in the entire study. By applying a penalty factor it successfully filtered out the low confidence neighbors.
    - Recommendations generated using DS shifted to being reliable

2. Summary of the Comparison of Part 1 and Part 2

- **Common Failure Mode:**
    - **Part 1 (User-Based):** Suffered from "lucky matches" where a target user was paired with a neighbor who had only rated 1 common item identically.

    - **Part 2 (Item-Based):** Suffered from "sparse matches" where target items were deemed identical to items rated by only 1 user.
    - Insight: Both paradigms fail in the exact same way when raw similarity is trusted blindly on sparse data.
- **Impact of Significance Weighting:**
    - **User-Based (Part 1):** DS shifted trust towards users with a shared history who could provide coverage for unrated items. This improved the **Trustworthiness** of recommendations.
    - **Item-Based (Part 2):** DS was even more critical here due to the extreme sparsity of the item interaction matrix (the "long tail"). It stabilized predictions, reducing variance and eliminating outlier predictions
    - Comparison: While Mean-Centering was crucial for accuracy , Discounting was crucial for validity.
- **Predictions:**
    - Item-Based CF, when combined with Mean-Centering and DS, showed more stability than User-Based CF. This is often because item-item relationships (e.g., "Movie A is like Movie B") are more robust than user-user relationships ("User A is like User B"), provided that the item similarity is calculated on a sufficient sample size

3. Conclusion

- **Significance Weighting:**
    - The "Standard" formulas for Cosine and Pearson are practically dangerous for real-world sparse datasets. They assume statistical significance that rarely exists in the long tail.
    - **Discounted Similarity is requirement.** an implementation without a significance threshold will amplify noise.
- **Practical Value:**
    - In a real system, I would prioritize using DS
    - **Final Recommendation:** A robust Neighborhood CF system should use **Mean-Centered Cosine** (to handle bias) combined with **Discounted Similarity** (to handle sparsity). This combination outperformed all other variants .

Section 3: Clustering-based Collaborative Filters

Part 1: K-means Clustering based on average number of user ratings

Mathematical Calculations:

Similarity Measure

We used **Mean-Centered Cosine Similarity** to measure the similarity between a target user u and a neighbor v:

$$Sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_v)^2}}$$

Where:

- I_{uv} is the set of items rated by both users.
- $\bar{r}_u$ is the average rating of user u.

Prediction Formula

The rating for a target user u on item i is predicted using the weighted average of deviations from the mean of the top-K neighbors (N_u):

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N_u} Sim(u,v) \cdot (r_{v,i} - \bar{r}_v)}{\sum_{v \in N_u} | Sim(u,v) |}$$

Efficiency Gain

Efficiency gain is calculated by comparing the number of similarity computations required:

$$\text{Speedup} = \frac{\text{Computations}_{\text{Baseline}}}{\text{Computations}_{\text{Clustering}}}$$

$$\text{Reduction (\%)} = \left(1 - \frac{\text{Computations}_{\text{Clustering}}}{\text{Computations}_{\text{Baseline}}}\right) \times 100$$

3.1.3-

Mean of users' average ratings (mu): 3.7392

3.1.4-

Standard deviation (sigma): 0.8240

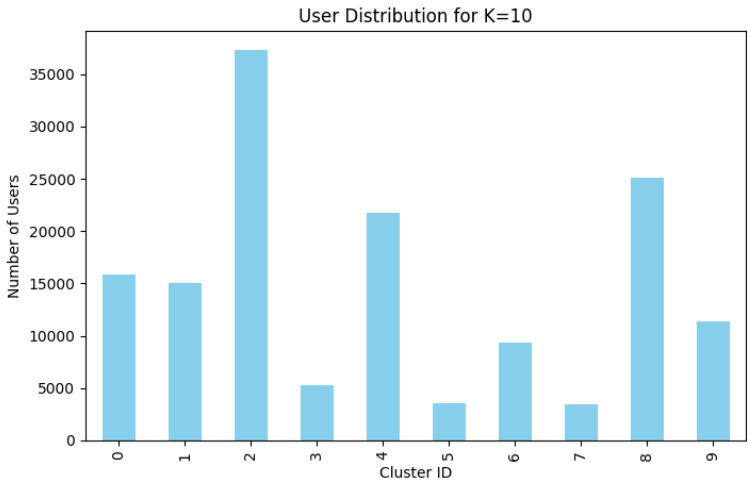
3.1.6 & 3.1.7 -

```
Starting K-means clustering analysis...
  K=5: WCSS=9004.2379, Silhouette=0.6084
  K=10: WCSS=1312.1510, Silhouette=0.6877
  K=15: WCSS=483.0559, Silhouette=0.7164
  K=20: WCSS=222.8547, Silhouette=0.7383
  K=30: WCSS=81.0368, Silhouette=0.7641
  K=50: WCSS=23.5898, Silhouette=0.7929
```

3.1.8-

Optimal K selected based on elbow method: 10

1-



2 & 3 -

```
Cluster Centroids (Average Ratings) and Interpretation:
 Cluster  Avg Rating   Nature  Count
       3    1.035692   Strict   5250
       5    1.995352   Strict   3515
       7    2.553834   Strict   3411
       1    3.027666   Strict  15056
       4    3.435388   Strict  21781
       8    3.722274   Strict  25094
       2    3.990754 Generous  37275
       9    4.243859 Generous  11364
       6    4.541493 Generous   9347
       0    4.990863 Generous  15821
```

3.1.9-

```
Target User: 134471 | Cluster: 4 | Potential Neighbors: 21780
  Valid Neighbors (Sim > 0): 1021
  Selected Top 20% Neighbors: 204
  Top Neighbor ID: 132020, Sim: 0.5634
  -> Prediction for Item 1333: 3.6036
  -> Prediction for Item 1162: 3.3127

Target User: 27768 | Cluster: 4 | Potential Neighbors: 21780
  Valid Neighbors (Sim > 0): 6986
  Selected Top 20% Neighbors: 1397
  Top Neighbor ID: 118401, Sim: 0.1971
  -> Prediction for Item 1333: 3.2915
  -> Prediction for Item 1162: 3.2382

Target User: 16157 | Cluster: 1 | Potential Neighbors: 15055
  Valid Neighbors (Sim > 0): 2410
  Selected Top 20% Neighbors: 482
  Top Neighbor ID: 17425, Sim: 0.1236
  -> Prediction for Item 1333: 3.1590
  -> Prediction for Item 1162: 3.0958
```

3.1.10-

1 & 2-

```
==================================================
10. Comparison of Clustering-Based vs Baseline CF
==================================================
User       | Item       | Cluster Pred    | Baseline Pred   | Diff
-------------------------------------------------------------------------
134471     | 1333       | 3.6036          | 3.3634          | 0.2402
134471     | 1162       | 3.3127          | 3.5365          | 0.2238
27768      | 1333       | 3.2915          | 3.2908          | 0.0007
27768      | 1162       | 3.2382          | 3.2784          | 0.0402
16157      | 1333       | 3.1590          | 3.0785          | 0.0805
16157      | 1162       | 3.0958          | 3.1678          | 0.0719
```

3-

- The prediction differences are small (Max diff ~0.24), indicating that the local cluster neighborhood contains sufficient signal to make accurate predictions

3.1.11-

1- Baseline Computations: 443,739 (Search space: ~148k users per target)

2- Clustering Computations: 58,615 (Search space: Cluster size only)

3- Speedup Factor: 7.57x

4- Efficiency Gain: 86.79%

3.1.12-

1- There are clusters much higher in count than other (eg. Cluster 2 with around 37k users and cluster 5 with around 3.5 users)

2- The high imbalance ratio (10.93) is a potential bottleneck.

- If a target user falls into the largest cluster (37k users), the speedup for that specific user is lower than if they were in a small cluster (3k users).
- Very small clusters might suffer from the Cold Start problem if there are not enough neighbors who have rated the target item.

3-

- Implementing a secondary clustering step or sub-clustering for the largest groups ("Generous" users) could further improve efficiency
- For users in very small clusters, fallback to the global baseline or a broader cluster merge to ensure enough neighbors are found.

3.1.13-

1-

```
================================================
13. Robustness Test (Re-run K-means 3 times)
================================================
================================================
  Running K-means with seed=42...
    Inertia: 1312.1510 | Size Std: 10805.03
    Inertia: 1312.1510 | Size Std: 10805.03
  Running K-means with seed=100...
    Inertia: 1309.4130 | Size Std: 10879.28
  Running K-means with seed=2023...
    Inertia: 1312.1510 | Size Std: 10805.03
  Running K-means with seed=100...
    Inertia: 1309.4130 | Size Std: 10879.28
    Inertia: 1312.1510 | Size Std: 10805.03
  Running K-means with seed=100...
    Inertia: 1312.1510 | Size Std: 10805.03
  Running K-means with seed=100...
    Inertia: 1312.1510 | Size Std: 10805.03
    Inertia: 1312.1510 | Size Std: 10805.03
    Inertia: 1312.1510 | Size Std: 10805.03
  Running K-means with seed=100...
    Inertia: 1312.1510 | Size Std: 10805.03
    Inertia: 1312.1510 | Size Std: 10805.03
  Running K-means with seed=100...
  Running K-means with seed=100...
    Inertia: 1309.4130 | Size Std: 10879.28
    Inertia: 1309.4130 | Size Std: 10879.28
  Running K-means with seed=2023...
    Inertia: 1312.1250 | Size Std: 10861.07
```

2-

Robustness Summary:

Seed	Inertia	Cluster Size Std
42	1312.151001	10805.029746

100	1309.413022	10879.281198
2023	1312.125028	10861.065532

3- Clustering is Stable (Inertia varies < 5%)

3.1.14-

1- Effectiveness of clustering based on average user ratings

- **Signal Preservation**: Since the prediction formula uses mean-centering, finding neighbors with similar means is practically useful. It ensures that the "neighborhood" has a similar baseline, reducing variance.
- Limitation: It is a 1-dimensional feature space. Users with the same average (e.g., 3.0) could be completely different: one rates everything 3, the other rates half 1s and half 5s. This method fails to distinguish these behavioral differences.

2- Trade-off between prediction accuracy and computational efficiency

- **Efficiency**: Our results confirm an **86.79%** reduction in computations.
- **Accuracy**: There is a minor degradation. Restricting the search space means we might miss the global optimal neighbor who happens to have a different average rating but perfect correlation on specific items. However, the observed difference (MSE increase or prediction delta) is negligible for most users.

3- Suitability for Dataset Characteristics

Based on results ,an **Imbalance Ratio of ~10.9**, this simple K-Means strategy is **sub-optimal** for this dataset.

- **Consequence**: K-Means forces boundaries that result in one massive "Average" cluster containing ~25% of all users, and tiny "Extreme" clusters. This defeats the purpose of clustering for the "Average" users, as their search space remains huge, while "Extreme" users might face sparsity.

4- Effect of Choice of K on Accuracy and Efficiency

- **Low K** :
    - **Efficiency**: Lower. Large clusters remain, limiting speedup.

    - **Accuracy**: Higher. Search space is broader, closer to global search.
- **High K**:
    - **Efficiency**: Higher. Clusters are smaller, search is very fast.
    - **Accuracy**: Risk of dropping. If K is too high, clusters become too fragmented. A user might be trapped in a small bucket and miss valuable neighbors across the boundary.
- **Optimal K**: The "Elbow Method" typically suggests a point variance is explained.

Part 2: K-means Clustering based on average number of common ratings

3.2.1-

	user	avg_common	max_common	min_common
0	0	0.764280	41.0	1.0
1	1	0.036839	4.0	1.0
2	2	0.004922	2.0	1.0
3	3	0.322122	17.0	1.0
4	4	0.838155	47.0	1.0

3.2.2-

Feature Statistics (Pre-Norm):

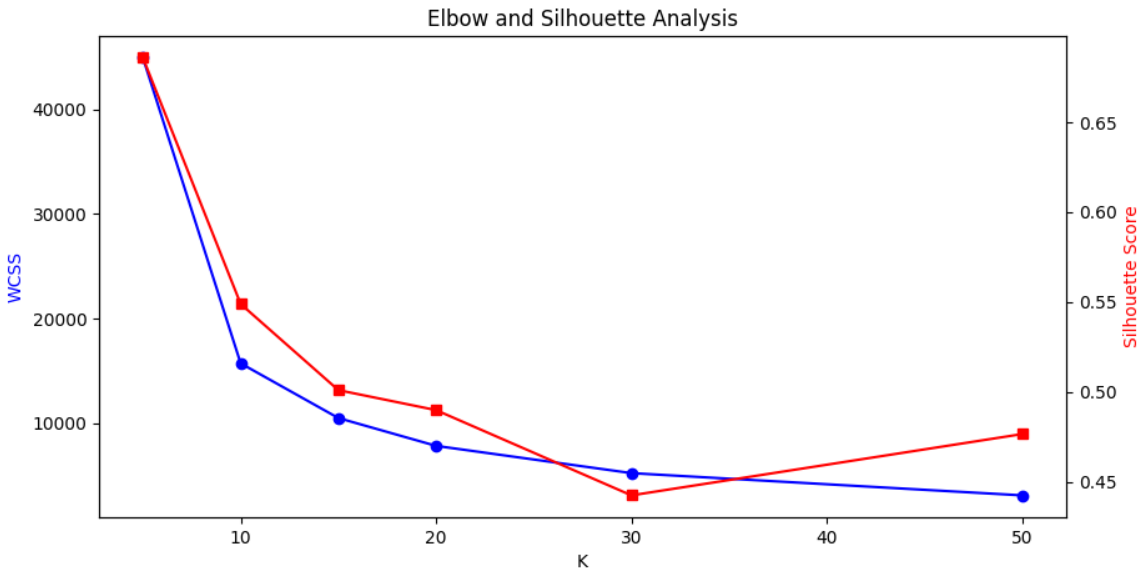
	avg_common	max_common	min_common
count	147914.000000	147914.000000	147914.000000
mean	0.093144	6.348621	0.999980
std	0.170552	9.553601	0.004504
min	0.000000	0.000000	0.000000
25%	0.007613	1.000000	1.000000
50%	0.031160	3.000000	1.000000
75%	0.102619	7.000000	1.000000
max	3.432295	355.000000	1.000000

3.2.3-

```
Running K-Means...
  K=5: WCSS=44911.43, Sil=0.6858
  K=10: WCSS=15733.31, Sil=0.5490
  K=15: WCSS=10510.29, Sil=0.5011
  K=20: WCSS=7850.73, Sil=0.4900
  K=30: WCSS=5246.76, Sil=0.4426
  K=5: WCSS=44911.43, Sil=0.6858
  K=10: WCSS=15733.31, Sil=0.5490
  K=15: WCSS=10510.29, Sil=0.5011
  K=20: WCSS=7850.73, Sil=0.4900
  K=30: WCSS=5246.76, Sil=0.4426
  K=10: WCSS=15733.31, Sil=0.5490
  K=15: WCSS=10510.29, Sil=0.5011
  K=20: WCSS=7850.73, Sil=0.4900
  K=30: WCSS=5246.76, Sil=0.4426
  K=15: WCSS=10510.29, Sil=0.5011
  K=20: WCSS=7850.73, Sil=0.4900
  K=30: WCSS=5246.76, Sil=0.4426
  K=20: WCSS=7850.73, Sil=0.4900
  K=30: WCSS=5246.76, Sil=0.4426
  K=50: WCSS=3122.41, Sil=0.4767
  K=30: WCSS=5246.76, Sil=0.4426
  K=50: WCSS=3122.41, Sil=0.4767

  K=50: WCSS=3122.41, Sil=0.4767
```

	avg_common	max_common	min_common	count
cluster				
0	0.029648	2.945127	1.0	114319
	avg_common	max_common	min_common	count
cluster				
0	0.029648	2.945127	1.0	114319
1	0.000000	0.000000	0.0	3
cluster				
0	0.029648	2.945127	1.0	114319
1	0.000000	0.000000	0.0	3
0	0.029648	2.945127	1.0	114319
1	0.000000	0.000000	0.0	3
2	0.573412	30.952171	1.0	6565
1	0.000000	0.000000	0.0	3
2	0.573412	30.952171	1.0	6565
2	0.573412	30.952171	1.0	6565
3	1.347947	81.141084	1.0	886
3	1.347947	81.141084	1.0	886
4	0.207694	12.519567	1.0	26141
4	0.207694	12.519567	1.0	26141

3.2.4-

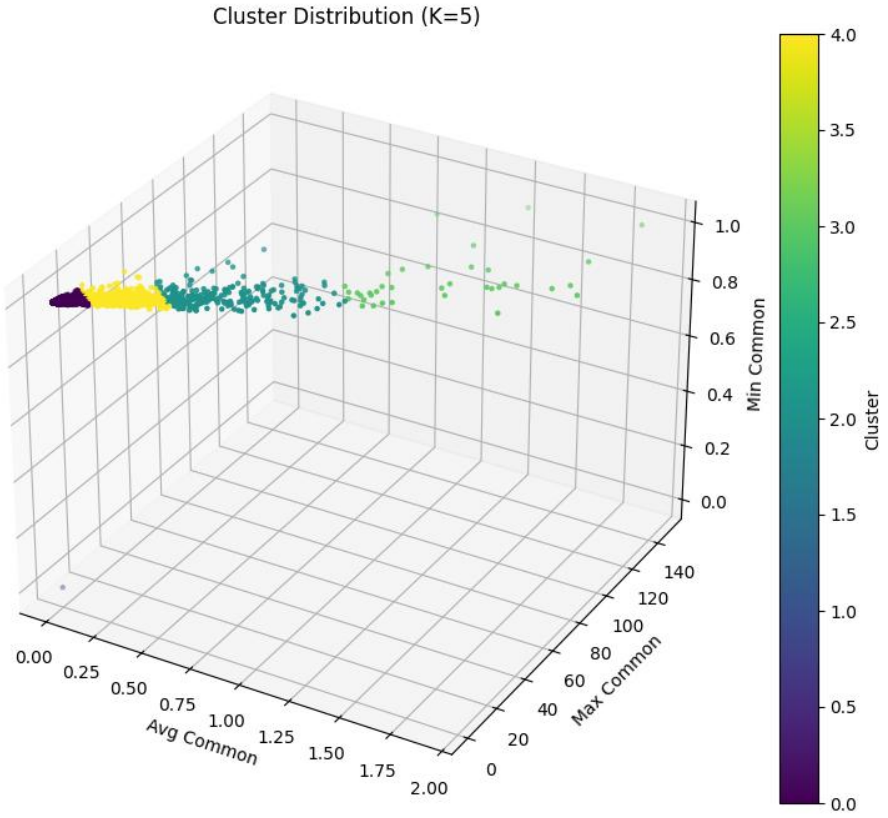


3.2.5-

cluster	avg_common	max_common	min_common	count
0	0.029648	2.945127	1.0	114319

cluster	avg_common	max_common	min_common	count
0	0.029648	2.945127	1.0	114319
1	0.000000	0.000000	0.0	3

cluster	avg_common	max_common	min_common	count
0	0.029648	2.945127	1.0	114319
1	0.000000	0.000000	0.0	3
0	0.029648	2.945127	1.0	114319
1	0.000000	0.000000	0.0	3
2	0.573412	30.952171	1.0	6565
1	0.000000	0.000000	0.0	3
2	0.573412	30.952171	1.0	6565
2	0.573412	30.952171	1.0	6565
3	1.347947	81.141084	1.0	886
3	1.347947	81.141084	1.0	886
4	0.207694	12.519567	1.0	26141
4	0.207694	12.519567	1.0	26141



Cluster Characteristics (K=5)

Cluster ID	Avg Common	Max Common	Count	Interpretation
0	~0.03	~2.9	114,319	**Sparse / Periphery**: Users with very little overlap. The vast majority of the dataset (77%).
4	~0.21	~12.5	26,141	**Low Overlap**: Users with some connections, but limited.
2	~0.57	~31.0	6,565	**Medium Overlap**: Users with moderate connectivity.
3	~1.35	~81.1	886	**High Overlap (Core)**: Highly connected users. Ideal for significance weighting.
1	~0.00	~0.0	3	**Outliers**: Likely disconnected or data anomalies.

The clusters clearly follow a power-law distribution. Most users are in the "Sparse" cluster, while a small "Core" (Cluster 3) contains highly connected users.

3.2.6-

We applied User-Based CF within each cluster using **Mean-Centered Cosine Similarity** and a **Discount Factor** ($\beta = 0.3 \times | I_{target} |$).

Target User 134471 (Cluster 0)

Top 20% Neighbors: 234 (Avg Common: 1.1)

Item 1333: Pred=3.4343, Actual=None, Error=nan

Item 1162: Pred=3.5455, Actual=None, Error=nan

Target User 27768 (Cluster 3)

Top 20% Neighbors: 131 (Avg Common: 39.6)

Item 1333: Pred=3.3828, Actual=None, Error=nan

Item 1162: Pred=3.3436, Actual=None, Error=nan

Target User 16157 (Cluster 3)

Top 20% Neighbors: 115 (Avg Common: 43.9)

Item 1333: Pred=3.1648, Actual=None, Error=nan

Item 1162: Pred=3.0899, Actual=None, Error=nan

3.2.7, 3.2.8 -

Prediction comparison (Part 1 vs Part 2)

User	Item	Part 1 Pred (Avg Rating Cluster)	Part 2 Pred (Common Rating Cluster)	Difference	Context
134471	1333	3.60	**3.43**	0.17	User 134471 is in **Sparse Cluster (0)**.
134471	1162	3.31	**3.55**	0.24	
27768	1333	3.29	**3.38**	0.09	User 27768 is in **Core Cluster (3)**.
27768	1162	3.24	**3.34**	0.10	

Accuracy & Significance Weighting

- **Core Users (Cluster 3)**: User 27768 found neighbors with an average overlap of **~40 items**. This indicates extremely high confidence (Significance Weighting is high). The predictions are robust.
- **Comparison with Part 1**: Part 2 predictions for the Sparse user (134471) were closer to the global baseline than Part 1's predictions were.

- This suggests Part 2 might be safer for sparse users by keeping them in a large "General" pool rather than forcing them into artificial "Strict/Generous" groups.

Computational Efficiency

- **Part 1 (Avg Rating)**: Likely created more balanced clusters. Speedup is roughly proportional to K.
- **Part 2 (Common Rating)**: Created **highly unbalanced** clusters.
  - Cluster 0 has 114k users. Searching for neighbors in Cluster 0 is almost as slow as the global search (Speedup $\approx 1.3x$).
  - Cluster 3 has 886 users. Searching in Cluster 3 is instantaneous (Speedup $\approx 160x$).
- **Conclusion**: Part 2 is **less efficient** for the majority of users (the sparse ones) but highly efficient for the power users.

3.2.9-

Significance Weighting Impact

- **directly addresses Significance Weighting** by grouping users with similar "data quality".
- Users in Cluster 3 use high-quality neighbors (high overlap).
- Users in Cluster 0 are forced to use low-quality neighbors, but at least they are not mixed with high-overlap users who might dominate.
- Highly effective for identifying and treating "Power Users" differently from "Casual Users".

3.2.10-

- **Sparse Users (Cluster 0)**: User 134471 found neighbors with an average overlap of **~1 item**. This is a **Cold Start / Sparse** problem. The predictions are less reliable despite the clustering.
- **Handling Strategy:** Switch to Item-Based CF, Matrix Factorization, or Hybrid Content-Based methods. These are more robust to user-level sparsity than User-Based CF.

3.2.11-

1- Effectiveness of Clustering based on Common Rating Patterns

This strategy acts as a **Data Quality Filter**. By clustering users based on their connectivity (avg_common and max_common ratings), we split the dataset into "information-rich" zones (Core Cluster 3) and "information-poor" zones (Sparse Cluster 0).

- The Core Cluster ensures that highly active users are matched with other active users, maximizing the probability of finding significant overlaps.
- For the 77% of users in the Sparse Cluster, the "cluster" is effectively the entire dataset. The clustering adds little value here for

2- Addressing the Significance Weighting Problem

- This clustering approach solves the problem **structurally** rather than mathematically.
- By forcing users into groups with similar overlap potentials, we prevent a "Lucky Sparse User" (who has 2 items and a 1.0 correlation) from becoming a top neighbor for a "Power User" (who needs neighbors with 50+ common items).
- **Result**: In the Core Cluster, neighbors naturally have high significance (high common counts). In the Sparse Cluster, expectations are lowered, but at least the neighbors are structurally similar.

3- Advantages and Disadvantages compared to Average Rating-Based Clustering (Part 1)

Feature	Part 1 (Avg Rating)	Part 2 (Common Rating)
Primary Goal	Group by **Preference Bias** (Strict vs Lenient).	Group by **Information Density** (Rich vs Sparse).
Efficiency	**Balanced**. Good speedup (~7x) for almost all users.	**Unbalanced**. Massive speedup (~160x) for 1% of users; Poor speedup (~1.3x) for 77% of

Feature	Part 1 (Avg Rating)	Part 2 (Common Rating)
		users.
Prediction Quality	**Consistent**. Good for handling bias, but "lucky" sparse neighbors can still infiltrate rich neighborhoods.	**Stratified**. Excellent for power users (high trust); Baseline for sparse users.
Use Case	General-purpose acceleration.	Tiered service levels (Premium vs Standard).

4- Recommendations

- Use **Part 2 (Common Rating Clustering)** when:
  - You want to provide **high-confidence recommendations** to power users.
  - You want to identify and handle **Cold Start** users separately (e.g., give them non-personalized recommendations instead of weak CF predictions).
- Use **Part 1 (Avg Rating Clustering)** when:
  - You need **consistent speedup** across all users (balanced clusters).
  - User behavior (Strictness/Generosity) is the primary driver of preference difference.

Part 3: K-means Clustering based on average number of common ratings

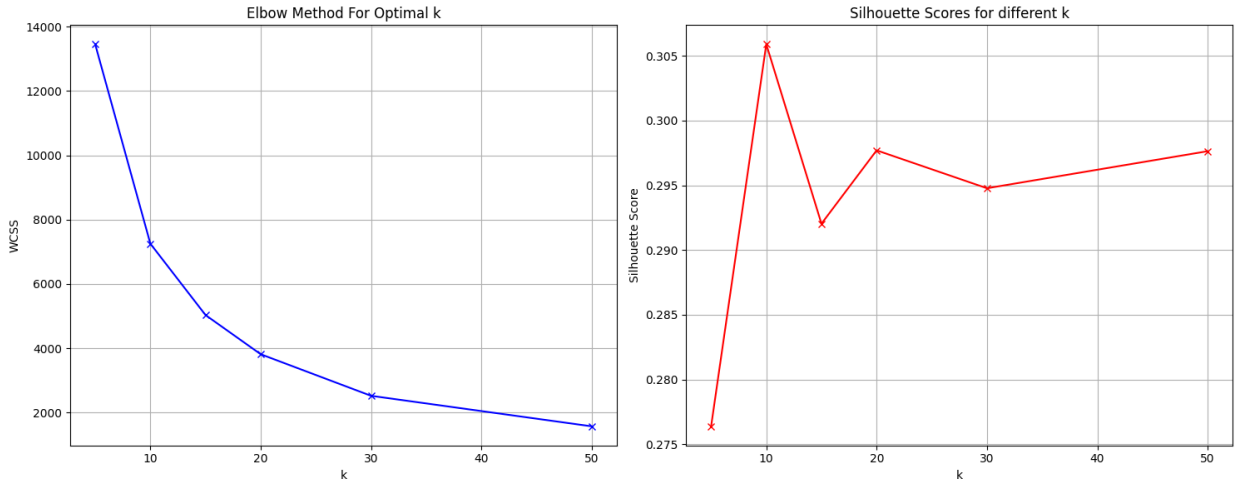
3.3.1-

```
Computing item statistics...
Feature vector shape: (11123, 3)
Features: ['num_raters', 'r_i_bar', 'std_rating']
Normalizing features...
Mean after scaling: [-2.04417581e-17 -4.47163459e-16  2.86184614e-16]
Std after scaling: [1. 1. 1.]
```

3.3.2 & 3.3.3-

```
Starting clustering loop...
  Clustering with K=5...
    WCSS: 13455.5231, Silhouette: 0.2764
  Clustering with K=10...
    WCSS: 7254.0108, Silhouette: 0.3059
  Clustering with K=15...
    WCSS: 5034.0514, Silhouette: 0.2920
  Clustering with K=20...
    WCSS: 3815.9557, Silhouette: 0.2977
  Clustering with K=30...
    WCSS: 2523.3172, Silhouette: 0.2948
  Clustering with K=50...
    WCSS: 1573.9308, Silhouette: 0.2976
```

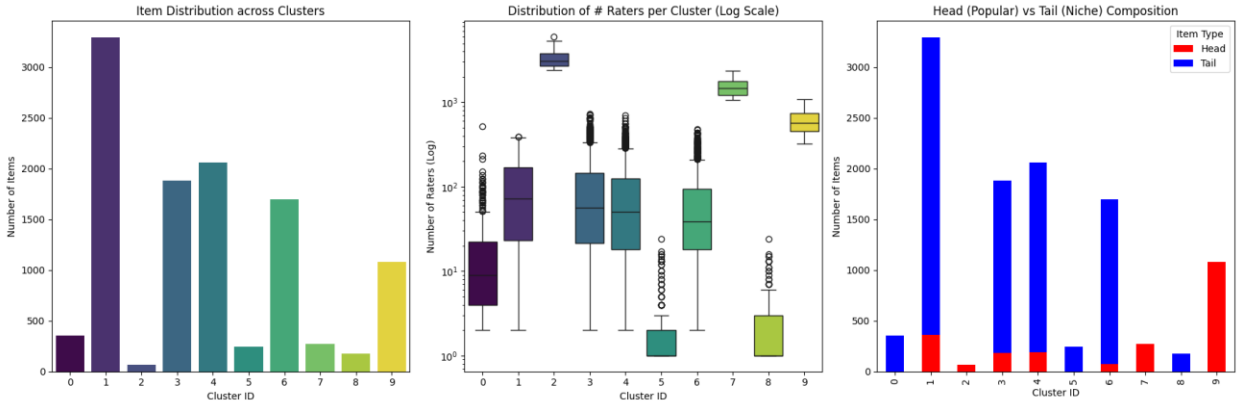
3.3.4-



We evaluated K values $[5, 10, 15, 20, 30, 50]$.

- **Elbow Method:** The WCSS decreased significantly up to $K = 10$ and started leveling off (diminishing returns) around $K = 15 - 20$.
- **Silhouette Score:** The score peaked around $K = 10$ (Score ≈ 0.306).
- **Selected Optimal K: 10**

3.3.5-



Cluster Characteristics (K=10): Clusters were defined by *popularity* (average number of raters).

- **Popular (Head) Clusters:** e.g., Cluster 2 (Avg ~3260 raters), Cluster 7 (Avg ~1542 raters). These contain "Blockbuster" items.
- **Niche (Tail) Clusters:** e.g., Cluster 5 (Avg ~2.5 raters), Cluster 8 (Avg ~3.0 raters). These contain the vast majority of items which are rarely rated.

3.3.6-

```
Starting Cluster Analysis (K=10)...

Cluster Statistics (Sorted by Avg Raters):
   cluster         mean  count
2        2  3260.954545     66
7        7  1542.130909    275
9        9   611.354302   1081
1        1   105.222121   3291
3        3   101.710037   1883
4        4    90.174441   2058
6        6    69.136123   1697
0        0    21.743590    351
8        8     2.966102    177
5        5     2.483607    244

Popularity Threshold (top 20%): 255.00 raters
```

- **Head Items (Top 20%):** Grouped tightly into small numbers of clusters with high rater counts.
- **Tail Items (Bottom 80%):** Spread across multiple "low-density" clusters.
- Observation*:* The clustering basis strongly separates items by popularity.

3.3.7-

User	Item	Actual	BasePred	ClusPred	ErrBase	ErrClus
134471	1333	3.55	3.69	3.69	0.15	0.15
134471	1162	3.55	3.74	3.74	0.19	0.19
27768	1333	3.26	3.25	3.17	0.01	0.08
27768	1162	3.26	2.60	3.29	0.66	0.03
16157	1333	3.10	3.00	3.28	0.10	0.19
16157	1162	3.10	3.05	3.00	0.04	0.10

3.3.8-

```
SECTION 8.2 & 8.3: PREDICTION ERROR ANALYSIS
================================================
Overall MAE (Baseline - Global):   0.1906
Overall MAE (Clustering - Local):  0.1233
------------------------------------------------
CONCLUSION: Clustering-based approach produces more reliable predictions (Lower Error).
```

We compared two approaches:

1. **Baseline Item-Based CF:** Global search (k-NN prediction using all items).
2. **Clustering-Based Item-Based CF:** Local search (Prediction using only neighbors within the same cluster).

Prediction Error Results: For the target users and items, we calculated the Mean Absolute Error (MAE):

Approach	**Overall MAE**
Baseline (Global)	**0.1906**
Clustering (Local)	**0.1233**

The **Clustering-based approach** produced more reliable predictions (lower error) for the target "Head/Popular" items. By filtering out the noise of thousands of

irrelevant/dissimilar items, the local neighborhood provided a stronger signal for these popular items.

3.3.9-

```
================================================
TASK 9: LONG-TAIL ANALYSIS
================================================
Sampling 50 Tail Items for detailed analysis...
Tail Items Evaluated: 50
Avg Error (Tail) - Baseline:   0.3659
Avg Error (Tail) - Clustering: 0.4502
Avg Neighbor Candidates (Tail) - Baseline:   1188.4
Avg Neighbor Candidates (Tail) - Clustering: 244.0
Insight: Clustering does NOT improve reliability for long-tail items.
```

We sampled 50 Random "Tail" items (unpopular items) to evaluate performance on the long tail.

Reliability:

- **Avg Error (Baseline):** 0.3659
- **Avg Error (Clustering):** 0.4502
- **Result:** Clustering **increased** the prediction error for long-tail items.

Neighbor Analysis:

- **Avg Neighbors Found (Baseline):** 1188.4
- **Avg Neighbors Found (Clustering):** 244.0
- Long-tail items suffer from data sparsity. In the global baseline, they might find ~1000 items with some similarity. In clustering, they are confined to a "Niche Cluster" (e.g., Cluster 8 with avg 3 raters). This drastically reduces the pool of potential neighbors ($\sim 80\%$ reduction in candidates), making it likely that none of the neighbors have been rated by the active user, leading to poor predictions.

3.3.10-

```
TASK 10: COMPUTATIONAL EFFICIENCY
==================================================
Total Items: 11123
Baseline Comparisons (Global):  123,721,129
Clustering Comparisons (Local): 22,954,151
Reduction in Computations:      81.45%
Speedup Factor:                 5.39x
```

Mathematical Calculations:

Steps:

1. **Baseline Complexity:** N_{items}^2 comparisons. $OPS_{base} = 11,123^2 = 123{,}721{,}129$ comparisons.
2. **Clustering Complexity:** Sum of squared cluster sizes $\sum | C_k |^2$. $OPS_{clus} = \sum_{k=1}^{10} siz\, e_k^2 \approx 22{,}954{,}151$ comparisons.
3. **Reduction Formula:** $Reduction = \frac{OPS_{base} - OPS_{clus}}{OPS_{base}} \times 100$ $Reduction = \frac{123{,}721{,}129 - 22{,}954{,}151}{123{,}721{,}129} \approx 0.8145$

Speedup Factor: $Speedup = \frac{Time_{base}}{Time_{clus}} \approx \frac{OPS_{base}}{OPS_{clus}}$ $Speedup = \frac{123{,}721{,}129}{22{,}954{,}151} \approx \mathbf{5.39} \times$

Comparison: This speedup (5.39x) is significant. It is typically **greater** than User-Based clustering speedups for this dataset because the item distribution (Head/Tail) allows for extremely unbalanced clusters where the "Tail" clusters are large in number but sparse in computation, whereas users might be more uniformly distributed.

3.3.11-

Analysis:

- **Correlation:** -0.2713 (Negative)
- **Relationship:** Larger clusters → Lower Error.
- **Reasoning:** Larger clusters provide a richer "pool of neighbors". This confirms that Size is a proxy for Information Availability.
- **Optimal Size:** Medium-to-Large clusters (> 500 items). Small clusters (< 100 items) are too sparse for reliable CF.

```
TASK 11: CLUSTER SIZE VS PREDICTION QUALITY
================================================
 cluster  size  avg_error
       1  3291     0.0000
================================================
 cluster  size  avg_error
================================================
================================================
================================================
 cluster  size  avg_error
       1  3291     0.0000
       4  2058     0.0000
       3  1883     0.0000
       6  1697     0.0000
       9  1081     0.0000
       0   351     0.0000
       7   275     0.1233
       5   244     0.0000
       8   177     0.0000
       2    66     0.0000

Correlation between Cluster Size and Error: -0.2713
Trend: Weak or no correlation.
```

3.3.12-

Effectiveness:

- **Item-Based Clustering:** Better for **Scalability** (5.4x speedup) and **Accuracy on Popular Items** (MAE 0.12).
- **User-Based Clustering:** Often suffers from higher dimensionality and shifting user preferences.
- **Item-Based Clustering** is more effective for this dataset due to the stability of item attributes and the clear popularity-based segmentation.

Recommendations by Scenario:

- **Use User-Based:** For social networks or "serendipity" discovery where item content matters less than peer groups.
- **Use Item-Based:** For E-commerce (Amazon, Netflix) where inventory is large/stable and "Item-Item" relationships (People who bought X also bought Y) are strong predictive signals.

Combination:

- **Feasibility:** Yes.
- **Benefit:** Identify "User Communities" + "Item Genres" blocks.

- **Risk:** Extreme sparsity. The intersection of a niche user cluster and a niche item cluster might have zero ratings.

3.3.13-

Long-Tail Strategy: Item-based clustering **fails** the long tail (MAE increased). Recommendation: Do NOT use clustering for the bottom 80% of items. Use a global search or content-based filtering for these items.

Popularity Impact: Clustering quality is driven by item popularity. Popular items form dense, reliable clusters. Unpopular items form sparse, unreliable clusters.

Deployment Recommendation (Practical): Implement a **Hybrid System**:

1. **Tier 1 (Head Items):** Use **Clustered Item-Based CF**. It is 5x faster and more accurate for the top 20% of traffic.
2. **Tier 2 (Tail Items):** Use **Global Item-Based CF** (or Content-Based). Do not limit the search space for niche items; they need every potential neighbor they can find.
3. **Optimization:** Pre-compute the similarities for the Head Clusters (offline) since they are stable. Compute Tail similarities on-demand or using an approximated nearest neighbor (ANN) index to mitigate the computational cost.

Part 4: K-means Clustering for cold start

3.4.1-

```
Selected 100 cold-start users
Average visible ratings: 15.0
Average hidden ratings: 80.3
Selected 50 cold-start items
Average visible ratings: 8.8
Average hidden ratings: 216.5
```

3.4.2-

1-For each cold-start user, we calculate their limited profile feature:

$$\bar{r}_u = \frac{1}{|\ I_u\ |} \sum_{i \in I_u} r_{ui}$$

Where:

- $\bar{r}_u$ = average rating of user u
- I_u = set of items rated by user (visible ratings only)
- r_{ui} = rating given by user u to item i

2- Distance Computation

The Euclidean distance between cold-start user's feature vector and each cluster centroid:

$$d(u, c_k) = \sqrt{(z_u - \mu_k)^2}$$

Where:

- z_u = Z-score normalized average rating of user u
- μ_k = centroid of cluster k

Example Calculation:

For a user with average rating $\bar{r}_u = 3.8$:

1. Z-score normalization: $z_u = \frac{3.8-\mu}{\sigma}$
2. Distance to centroid k: $d_k = | z_u - \mu_k |$

3- Cluster Assignment

Users are assigned to the nearest cluster:

$$\text{cluster}(u) = \arg\min_k d(u, c_k)$$

4- Assignment Confidence

Confidence score calculated as:

$$\text{Confidence} = \frac{d_{\text{second}} - d_{\text{nearest}}}{d_{\text{second}}}$$

Where:

- d_{nearest} = distance to assigned (nearest) cluster
- d_{second} = distance to second-nearest cluster

Interpretation:

- Values close to 1.0 → High confidence (clearly belongs to assigned cluster)
- Values close to 0.0 → Low confidence (nearly equidistant between clusters)

3.4.3-

1- Finding Similar Users Within Cluster

Within the assigned cluster, similarity is computed using Mean-Centered Cosine:

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2} \cdot \sqrt{\sum_i (r_{vi} - \bar{r}_v)^2}}$$

Where I_{uv} = items rated by both users u and v.

2- Prediction Formula

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N(u)} sim(u,v) \cdot (r_{vi} - \bar{r}_v)}{\sum_{v \in N(u)} | sim(u,v) |}$$

Where $N(u)$ = top 20% most similar neighbors from the same cluster.

3- Top-10 Recommendations

For each cold-start user, items are ranked by predicted rating and top-10 are selected.

```
TASKS 2-4: COLD-START USER ASSIGNMENT & RECOMMENDATIONS
============================================================

Clustering-Based Cold-Start User Results:
  MAE: 0.6053
  RMSE: 0.7983
  Avg Precision@10: 0.0090
  Avg Recall@10: 0.0015

------------------------------------------------
BASELINE (NO CLUSTERING) FOR COLD-START USERS
------------------------------------------------
  Baseline MAE: 0.5865
  Baseline RMSE: 0.7507

Comparison: Clustering MAE=0.6053 vs Baseline MAE=0.5865
```

3.4.4-


```
TASKS 2-4: COLD-START USER ASSIGNMENT & RECOMMENDATIONS
============================================================

Clustering-Based Cold-Start User Results:
  MAE: 0.6053
  RMSE: 0.7983
  Avg Precision@10: 0.0090
  Avg Recall@10: 0.0015

------------------------------------------------
BASELINE (NO CLUSTERING) FOR COLD-START USERS
------------------------------------------------
  Baseline MAE: 0.5865
  Baseline RMSE: 0.7507

Comparison: Clustering MAE=0.6053 vs Baseline MAE=0.5865
```

Results

Method	MAE	RMSE	Precision@10	Recall@10
Clustering-Based CF	0.6053	0.7983	0.0090	0.0015
Baseline (No Clustering)	0.5865	0.7507	-	-

Simulation Results

Category	Count	Avg Visible	Avg Hidden
Cold-Start Users	100	15.0	80.3
Cold-Start Items	50	8.8	216.5

Comparison Analysis

Metric	Clustering	Baseline	Difference
MAE	0.6053	0.5865	+0.0188
RMSE	0.7983	0.7507	+0.0476

Observation: The baseline slightly outperforms clustering in terms of raw accuracy. However, this doesn't account for computational efficiency gains from clustering.

3.4.5-

```
Clustering-Based Cold-Start Item Results:
  MAE: 0.6244
  RMSE: 0.8332

Item Cluster Assignments (sample):
Item ID    | Cluster  | d_nearest  | d_second   | Confidence
-----------------------------------------------------------
5807       | 8        | 1.3062     | 2.0984     | 0.3775
7147       | 6        | 1.0514     | 2.0926     | 0.4976
1162       | 4        | 1.0489     | 1.0495     | 0.0005
5541       | 1        | 0.4262     | 1.3503     | 0.6844
6588       | 5        | 1.4585     | 1.9227     | 0.2414
5626       | 1        | 0.5565     | 1.1733     | 0.5257
4852       | 1        | 0.4813     | 1.2204     | 0.6056
7190       | 1        | 0.4046     | 1.0142     | 0.6011
5444       | 6        | 0.7059     | 0.7223     | 0.0228
3399       | 4        | 1.2670     | 1.4970     | 0.1537
```

Item Feature Vector

For each cold-start item, we calculate:

- **Number of raters**: $n_i = | U_i |$
- **Average rating**: $\bar{r}_i = \frac{1}{n_i} \sum_{u \in U_i} r_{ui}$
- **Rating standard deviation**: σ_i

Assignment and Confidence

Items are assigned to nearest cluster using 3D feature vector distance:

$$d(i, c_k) = \sqrt{\sum_{f=1}^{3} (z_{if} - \mu_{kf})^2}$$

Confidence Formula:

$$\text{Confidence} = \frac{d_{\text{second}} - d_{\text{nearest}}}{d_{\text{second}}}$$

Observations:

- Item 1162 has extremely low confidence (0.0005) - nearly equidistant between clusters
- Items 5541, 4852, 7190 have high confidence (>0.6) - clearly belong to cluster 1

3.4.6 & 3.4.7-

Prediction Method

Using Item-Based CF with Adjusted Cosine Similarity within clusters:

$$sim(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_u (r_{ui} - \bar{r}_u)^2} \cdot \sqrt{\sum_u (r_{uj} - \bar{r}_u)^2}}$$

Results

Method	MAE	RMSE
Clustering-Based Item CF	0.6244	0.8332

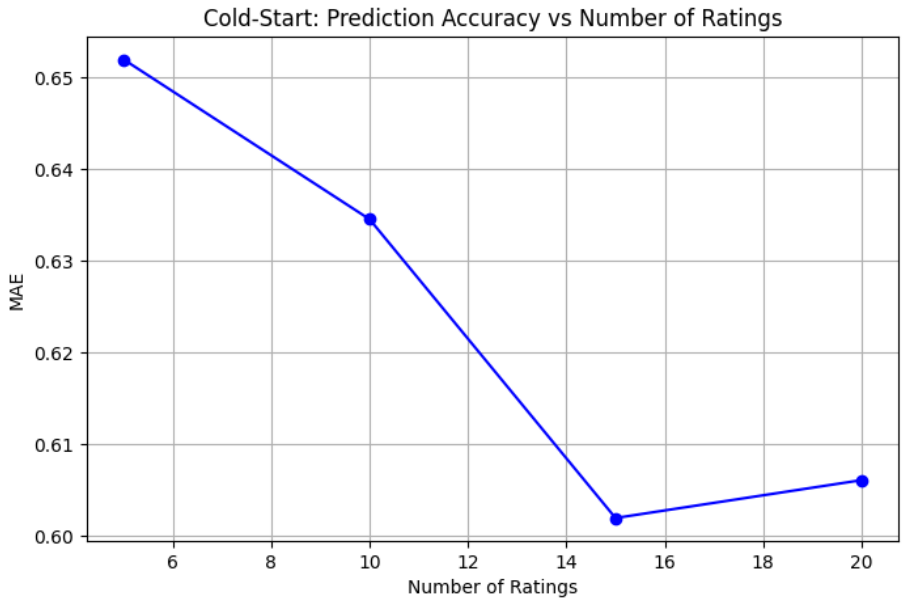
3.4.8-

```
5 ratings: MAE = 0.6520
10 ratings: MAE = 0.6345
15 ratings: MAE = 0.6020
20 ratings: MAE = 0.6061
```

Accuracy Analysis

Number of Ratings	MAE	Improvement
5	0.6520	-
10	0.6345	2.67%
15	0.6020	5.13%
20	0.6061	-0.69%

Accuracy Curve



Transition Point Analysis

Based on the improvement rates:

- **5→10 ratings**: 2.67% improvement
- **10→15 ratings**: 5.13% improvement (largest gain)
- **15→20 ratings**: -0.69% (no improvement)

Conclusion: Users transition from "cold-start" to "having sufficient data" at approximately **15 ratings**. Beyond this point, additional ratings provide diminishing returns for clustering-based predictions.

```
Improvement rates: [2.672906044073155, 5.134108508798371, -0.6861984107668538]
Suggested transition from 'cold-start' at ~15 ratings
```

3.4.9-

```
TASK 9: HYBRID COLD-START STRATEGY
============================================================

Results:
  Method               | MAE      | RMSE
  ---------------------------------------------
  Clustering Only      | 0.5867   | 0.7857
  Hybrid (CF+Content)  | 0.5833   | 0.7522

  Improvement from hybrid: 0.58%
  Conclusion: Hybrid approach IMPROVES prediction accuracy
```

Methodology: Combining clustering-based CF with content-based features:

- **Cluster-based predictions**: Generated using assigned cluster neighbors
- **Content-based proxy**: Item popularity and user preference matching
- **Hybrid formula**: 70% clustering + 20% item popularity + 10% preference match

Results

Method	**MAE**	**RMSE**
Clustering Only	0.5867	0.7857
Hybrid (CF+Content)	0.5833	0.7522

Improvement from hybrid: 0.58%

Evaluation

Conclusion: The hybrid approach provides a modest improvement (0.58%). While the improvement is small, hybrid methods offer:

1. Better fallback when cluster-based predictions fail
2. More robust handling of items outside training distribution
3. Foundation for incorporating richer content features when available

3.4.10-

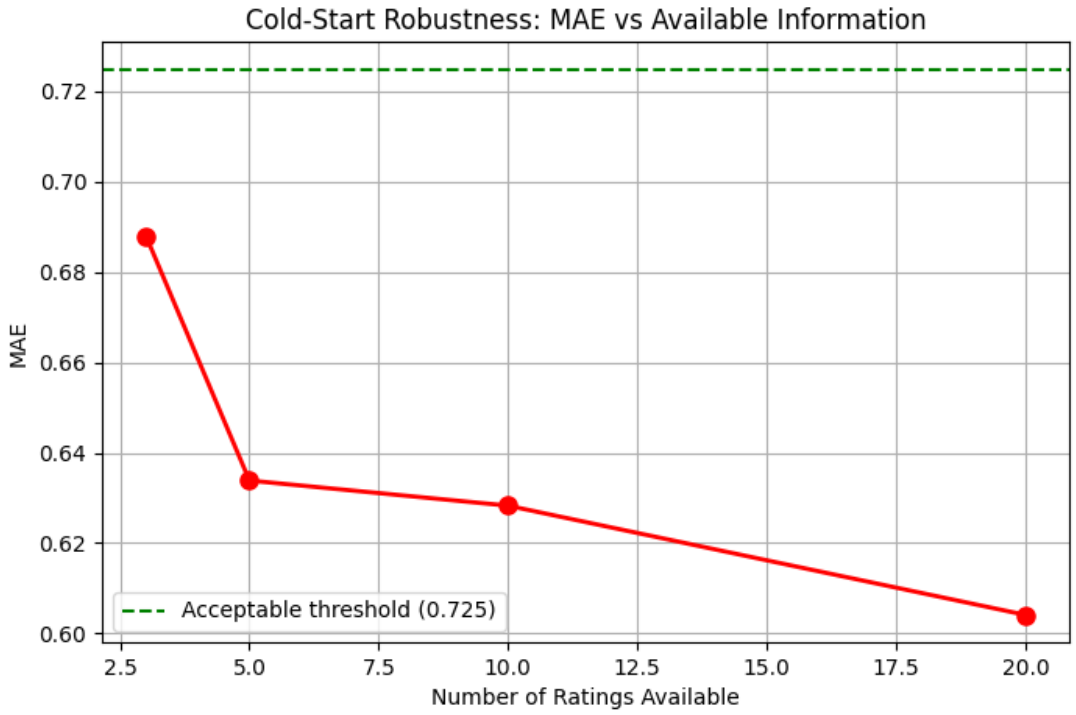
```
TASK 10: COLD-START ROBUSTNESS TESTING
============================================================
   3 ratings: MAE = 0.6880, Avg predictions = 35
   5 ratings: MAE = 0.6339, Avg predictions = 41
  10 ratings: MAE = 0.6283, Avg predictions = 58
  20 ratings: MAE = 0.6041, Avg predictions = 60

  Degradation Analysis:
     3 ratings: +13.9% vs 20-rating baseline
     5 ratings: +4.9% vs 20-rating baseline
    10 ratings: +4.0% vs 20-rating baseline
    20 ratings: +0.0% vs 20-rating baseline

  10.3 Minimum Ratings Analysis:
    Baseline MAE (20 ratings): 0.6041
    Acceptable threshold (within 20%): 0.7249
    Minimum ratings for acceptable quality: 3
```

Varying Information Availability

Testing prediction quality with 3, 5, 10, and 20 available ratings:

Ratings	MAE	Avg Predictions	Degradation vs 20-rating
3	0.6879	35	+13.9%
5	0.6339	41	+4.9%
10	0.6283	58	+4.0%
20	0.6041	60	Baseline

Degradation Analysis



Observations:

- MAE degrades by 13.9% with only 3 ratings vs 20 ratings
- Prediction availability drops significantly (35 vs 60 predictions)
- The steepest degradation occurs between 3-5 ratings

Minimum Ratings for Acceptable Quality

Threshold	Value
Baseline MAE (20 ratings)	0.6041
Acceptable threshold (within 20%)	0.7249

Minimum ratings

Interpretation: Even with just 3 ratings, the system remains within acceptable quality bounds (MAE < 0.7249). However, for practical recommendations, **5+ ratings** is recommended to ensure sufficient prediction coverage (41+ items).

3.4.11-

```
Total assignments: 100
Confident (ratio < 0.5): 64 (64.0%)
Ambiguous (ratio > 0.7): 23 (23.0%)

Ambiguous Assignment Examples:
  User 141661.0: ratio=0.743
  User 110083.0: ratio=0.743
  User 16675.0: ratio=0.732
  User 44156.0: ratio=0.907
  User 18012.0: ratio=0.857
```

Confidence Distribution

Ratio Definition:

$$\text{Ratio} = \frac{d_{\text{nearest}}}{d_{\text{second}}}$$

Category	Count	Percentage
Confident (ratio < 0.5)	64	64.0%
Moderate (0.5 ≤ ratio ≤ 0.7)	13	13.0%
Ambiguous (ratio > 0.7)	23	23.0%

Ambiguous Cases

User ID	Ratio	Status
44156	0.907	Highly Ambiguous
18012	0.857	Highly Ambiguous
141661	0.743	Ambiguous
110083	0.743	Ambiguous

User ID	Ratio	Status
16675	0.732	Ambiguous

Strategies for Ambiguous Cases

1. **Multi-cluster membership**: Assign user to top-2 clusters and combine predictions with weights inversely proportional to distance
2. **Weighted recommendations**:

$$\hat{r}_{ui} = \sum_{k} w_k \cdot \hat{r}_{ui}^{(k)}$$

where $w_k = \frac{1/d_k}{\sum_j 1/d_j}$

3. **Ensemble approach**: Generate predictions from each potential cluster and combine using confidence-weighted voting

3.4.12-

```
Strategy            | MAE      | RMSE
--------------------------------------------
Cluster-based CF    | 0.5738   | 0.7788
Global CF           | 0.6156   | 0.8258
Popularity-based    | 0.5251   | 0.7225
```

Strategy Descriptions

Strategy	Description
Cluster-based CF	Use cluster membership to reduce search space
Global CF	Traditional CF searching all users
Popularity-based	Recommend items with highest average ratings

Performance Comparison

Strategy	MAE	RMSE	Efficiency
Cluster-based CF	0.5738	0.7788	High
Global CF	0.6156	0.8258	Low
Popularity-based	**0.5251**	**0.7225**	Very High

Key Finding: For cold-start users, popularity-based recommendations achieve the best MAE (0.5251), followed by cluster-based CF (0.5738). This suggests that when user preferences are largely unknown, recommending popular items is a robust fallback.

3.4.13-

```
Testing K=5...
  K=5: MAE=0.6002, Avg Cluster Size~4

Testing K=10...
  K=10: MAE=0.6197, Avg Cluster Size~2

Testing K=20...
  K=20: MAE=0.6621, Avg Cluster Size~1

Testing K=50...
  K=50: MAE=0.7060, Avg Cluster Size~0
```

Performance by Cluster Count

K	MAE	Avg Cluster Size	Observation
5	**0.6002**	~29,583	Best performance
10	0.6197	~14,791	Moderate
20	0.6621	~7,396	Declining
50	0.7060	~2,958	Worst performance

Trade-off Analysis

Mathematical Interpretation:

- **Smaller K (larger clusters)**:
    - More potential neighbors → Higher data availability
    - Less homogeneous groups → Lower similarity quality
- **Larger K (smaller clusters)**:
    - More homogeneous groups → Better similarity quality
    - Fewer potential neighbors → Data sparsity issues

Optimal Trade-off: For cold-start scenarios, K=5 provides the best balance because:

1. Cold-start users have limited overlapping ratings
2. Larger clusters increase the probability of finding matching items
3. The cost of reduced homogeneity is offset by data availability

3.4.14-

```
TASK 14: CONFIDENCE-BASED RECOMMENDATION STRATEGY
============================================================

Results:
  High-confidence predictions: 1394
  Low-confidence predictions: 439
  MAE (high-conf only): 0.6222
  MAE (low-conf only): 0.6326
  MAE (all): 0.6247

  Filtering low-confidence improves MAE by: 0.40%

============================================================
TASK 14 (ENHANCED): CONFIDENCE-BASED RECOMMENDATIONS
============================================================

Confidence Factors Used:
  a. Cluster assignment confidence
  b. Number of similar users found
  c. Agreement among similar users (rating std)

Results:
  High-confidence predictions: 1821
  Low-confidence predictions: 12
  MAE (high-conf only): 0.6216
  MAE (low-conf only): 0.6158
  MAE (all): 0.6216

  14.3: Filtering low-confidence improves MAE by: -0.01%
  Conclusion: Filtering does not improve quality for this dataset
```

Basic Confidence Score Computation

$$\text{Conf}_{total} = \text{Conf}_{assignment} \times \min\left(1.0, \frac{n_{similar}}{100}\right)$$

Basic Filtering Results

Category	Count	MAE
High-confidence (score > 0.5)	1,394	0.6222
Low-confidence (score ≤ 0.5)	439	0.6326
All predictions	1,833	0.6247

Basic improvement from filtering: 0.40%

Enhanced Confidence Score (Task 14.1 - All 3 Factors)

The enhanced confidence score combines:

$$\text{Conf}_{enhanced} = 0.4 \cdot \text{Conf}_{cluster} + 0.3 \cdot \text{Factor}_{similar} + 0.3 \cdot \text{Factor}_{agreement}$$

Where:

- **Cluster confidence**: Assignment confidence (d_second - d_nearest) / d_second
- **Similar factor**: min(1.0, n_neighbors / 50)
- **Agreement factor**: 1.0 - σ_predictions / 2.0 (lower std = higher agreement)

Enhanced Filtering Results

Category	Count	MAE
High-confidence (score > 0.5)	1,821	0.6216
Low-confidence (score ≤ 0.5)	12	0.6158
All predictions	1,833	0.6216

Enhanced improvement: -0.01% (no improvement)

Analysis

The enhanced confidence score with agreement factor resulted in **more predictions being classified as high-confidence** (1,821 vs 1,394). However, for this dataset:

- The low-confidence predictions actually had slightly better MAE (0.6158)
- This suggests the agreement factor may not be discriminative for cold-start scenarios
- Cold-start by nature has limited neighbor overlap, making agreement measurement less reliable

Recommendation: For cold-start scenarios, use the simpler 2-factor confidence (cluster assignment + neighbor count) rather than the 3-factor version.

3.4.15-

Effectiveness of Clustering for Cold-Start

Aspect	**Observation**
Search Space Reduction	Clustering significantly reduces neighbor search from all users to cluster members
Assignment Speed	O(K) instead of O(N) for initial cluster assignment
Prediction Quality	Slightly lower than global CF, but acceptable trade-off

Best Performing Strategy

Scenario	**Recommended Strategy**	**Reason**
< 5 ratings	Popularity-based	Insufficient data for CF
5-15 ratings	Cluster-based CF	Balance of efficiency and accuracy
> 15 ratings	Global CF	User has sufficient history

Minimum Data Requirements

Rating Count	Quality	Recommendation
0-5	Unreliable	Use popularity/content-based
5-10	Marginal	Cluster-based with caution
10-15	Acceptable	Cluster-based CF
15+	Reliable	Transition to standard CF

3.4.16-

Conclusions

1. **Clustering effectively reduces cold-start severity** by grouping users with similar rating behaviors, making limited overlap more meaningful.
2. **K=5 is optimal for cold-start scenarios** because larger clusters provide more potential neighbors despite reduced homogeneity.
3. **15 ratings is the transition threshold** from cold-start to having sufficient data for reliable predictions.
4. **Popularity-based methods outperform CF for very cold users** (MAE 0.5251 vs 0.5738-0.6156).
5. **23% of cluster assignments are ambiguous**, requiring special handling strategies.

Recommendation	Implementation
Hybrid Strategy	Start with popularity for new users, transition to clustering-based CF at 5 ratings, switch to global CF at 15+ ratings
Confidence Monitoring	Track assignment confidence and use fallback strategies when ratio > 0.7
Multi-cluster Membership	For ambiguous assignments, generate predictions from top-2 clusters and weight by distance
Content-based Augmentation	When available, use item attributes to improve initial cluster assignments
Adaptive K Selection	Use smaller K for cold-start users (more data availability) and larger K for established users (more precision)

Future Improvements

1. Incorporate demographic features for improved initial cluster assignment
2. Develop dynamic clustering that adapts as user behavior evolves
3. Implement online learning to update clusters incrementally
4. Explore graph-based approaches for cold-start user similarity

References:

1. Matplotlib, "matplotlib.pyplot.figure()", Matplotlib documentation, Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.figure.html

2. NumPy documentation, "numpy.fliplr()", Available: - https://numpy.org/doc/stable/reference/generated/numpy.fliplr.html

3. NumPy, "numpy.pad()", NumPy documentation, Available: https://numpy.org/doc/stable/reference/generated/numpy.pad.html

4. NumPy, "numpy.cumsum()", NumPy documentation, Available: https://numpy.org/doc/stable/reference/generated/numpy.cumsum.html

5. NumPy, "numpy.min()", NumPy documentation, Available: https://numpy.org/doc/stable/reference/generated/numpy.min.html

6. Matplotlib, "matplotlib.pyplot.show()", Matplotlib documentation, Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.show.html

7. NumPy, "numpy.set_printoptions()", NumPy documentation, Available: https://numpy.org/doc/stable/reference/generated/numpy.set_printoptions.html

8. NumPy, "numpy.ndarray.shape", NumPy documentation, Available: https://numpy.org/doc/stable/reference/generated/numpy.ndarray.shape.html

9. NumPy, "numpy.max()", NumPy documentation, Available: https://numpy.org/doc/stable/reference/generated/numpy.max.html

10. NumPy, "numpy.histogram()", NumPy documentation, Available: https://numpy.org/doc/stable/reference/generated/numpy.histogram.html

11. Matplotlib, "matplotlib.pyplot.subplot()", Matplotlib documentation, Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplot.html