AIE425 Intelligent Recommender Systems
Assignment 1: Neighborhood Collaborative Filters and Clustering in Collaborative Filters
Submission Deadline:

- **Deliverables: Sunday 7 December 2025 before 11:59 PM**
- **Part Implementation (Individual Assessment): During Labs of Week 11**

Using AI assistance (ChatGPT, Copilot, etc.) is allowed for learning but must be acknowledged

1. Assignment Instructions

- This assignment document consists of 18 pages in total. Students are required to review and complete the assignment in order, following the structure provided.
- The assignment is divided into three main sections (with six parts in total).
- Each part contains a clear set of tasks, analytical steps, and implementation requirements, as detailed in Section 3 of the assignment.
- Team Structure and Responsibilities:
  - Students must work in **groups of THREE**.
  - The group will submit one complete assignment on CANVAS and GitHub by the deadline as specified in the deliverable section.
  - **Each member** must be responsible for one specific part of the assignment.
  - The integration, formatting, quality check, and final upload of all components; including the full report, are the shared responsibility of the entire group.
- Team Leader Responsibilities:
  - The designated team leader must send an email to the Lab TA, with CC to the course instructor and all group members, by Thursday 8 December 2025 before 4:00 PM. he email must clearly include:
    - Subject: AIE425 - Assignment 1 Group Registration - [Team Leader ID & Name].
    - Team member names and IDs as on SIS.
    - The assigned part for each member
    - The selected dataset with download link or data source reference.
- Submission Requirements:
  - Deliverables (CANVAS + GitHub)
    - All deliverables must be submitted on Sunday 7 December 2025 before 11:59 PM on CANVAS and GitHub (your group repository).
    - Each submission must include:
      - The complete report (PDF + Word format)
      - The complete source code & the dataset used for the assignment
      - A README file documenting how to run your code
      - All generated outputs, plots, intermediate analysis files, and saved results required in the assignment
  - During the Week 11 Lab, each student will be assessed individually as follows:
    - Each student MUST Complete the implementation for one assigned part from the assignment.
    - Push the code to GitHub within the lab duration
    - Use the dataset already used and uploaded with the deliverables
    - Failure to complete and push your assigned task during the lab will result in a mark of ZERO for the individual implementation component.

- **Dataset Requirements:**
  Groups must select ONE dataset from the following options:
  1. MovieLens Dataset: https://grouplens.org/datasets/movielens/ (100K, 1M, or 10M versions available)
  2. Amazon Product Reviews: https://nijianmo.github.io/amazon/index.html (Books, Electronics, or Movies categories)
  3. Goodbooks-10k: https://github.com/zygmuntz/goodbooks-10k
  4. Yelp Dataset: https://www.yelp.com/dataset

  Other approved sources:
  5. Yongfeng Zhang's Collection: http://yongfeng.me/dataset/
  6. RUCAIBox RecSysDatasets: https://github.com/RUCAIBox/RecSysDatasets
  7. Hugging Face Datasets: https://huggingface.co/datasets?task_categories=recommendation
  8. Book-Crossing: http://www2.informatik.uni-freiburg.de/~cziegler/BX/
  9. Last.fm: https://grouplens.org/datasets/hetrec-2011/
  10. Custom datasets (must be approved by instructor via email at least 2 weeks before deadline)

  Criteria of Dataset selection: Minimum 1,000 users, Minimum 1,000 items, Minimum 10,000 ratings
  Include a **README.md** file in your GitHub repository with dataset name and source; dataset statistics (number of users, items, ratings); Instructions for downloading/accessing the dataset

Note: Some datasets require registration or have usage restrictions. Students should check the terms of use before selecting a dataset.

2. Introduction

This assignment aims to deepen your understanding of Neighborhood-Based Collaborative Filtering, Clustering-Based Collaborative Filtering, and the computational considerations required to scale recommender systems to large datasets. Throughout the assignment, you will work with a real dataset of at least 100,000 users, 1,000 products, and 1 million ratings, applying a complete Recommender Systems pipeline ranging from statistical dataset analysis to clustering and computational complexity evaluation.
This assignment consists of the following THREE Sections.
Section 1: Statistical Analysis
Section 2: Neighborhood CF Filters
- Part 1: user-based CF.
- Part 2: item-based CF.

Section 3: Clustering-based Collaborative Filters
- Part 1: K-means Clustering based on average number of user ratings.
- Part 2: K-means Clustering based on average number of common ratings.
- Part 3: K-means Clustering based on average number of raters.
- Part 4: K-means Clustering for Cold-Start.

3. Assignment Description and requirements

Round your values to 2 decimal places in all calculations.

3.1. Section ONE: Statistical Analysis:

1. Prepare a dataset of at least 100,000 users, ≥ 1000 products, and ≥ 1 million ratings.
2. Preprocess the dataset to adjust the rating on a 1-to-5 scale.
3. Calculate the number of ratings for each user (n_u) and save it
4. Calculate the number of ratings for each item (n_i) and save it.
5. Compute the average ratings per user ($\bar{r}_u$) in your dataset and save it.
6. Compute the average ratings per item ($\bar{r}_i$) in your dataset and save it.
7. Ascendingly order the total number of ratings per item and plot the distribution per item.
8. Compute the number of products based on their average ratings such that G1 is a group of products whose average ratings per product ≤ 1%, 1%<G2≤5%, 5%<G3≤10%, 10%<G4≤20%, 20%<G5≤30%, 30%<G6≤40%, 40%<G7≤50%, 50<G8≤60%, 60%<G9≤70%, 70<G10≤100%.
9. Compute the total number of ratings in each group and order them ascendingly.
10. Plot the distribution of the number of ratings in each group before and after ordering.
11. Select three target users:
    - U1 with ≤ 2% ratings
    - U2 with ratings >2% and ≤5%
    - U3 with ratings >5% and ≤10%
12. Select two target items:
    - Select the two lowest rated items (I1 and I2) as target items
13. Count the number of co-rating users between each target user and other users (No_common_users), and the number of co-rated items between each target item and other items (No_coRated_items).
14. Determine the threshold β: maximum number of users who have co-rated at least 30% of items with each target user.
15. Save all intermediate results for use in later parts.
16. Compare the results from point 13 & 14 and give your insights into the dataset by evaluating and discussing the matrix sparsity, rating bias and long-tail problems. Give your comments in a separate section in your report.

3.2. Section TWO: Neighborhood CF Filters requirements

3.2.1. Part 1: User-Based Collaborative Filtering Requirements and Questions:

Part 1 includes the following THREE case studies.

Case study 1:

1- Apply a user-based CF method using raw Cosine Similarity to calculate the similarity between each target user and other users.

2- Based on the similarity scores, identify the top 20% most similar users to each target user.

3- Use these top similar users to predict unknown ratings for each target user and determine their potential preference for unrated items.
4- Calculate the Discount Factor (DF) and then the Discounted Similarity (DS) for each target user using threshold $\beta \geq 30\%$.
5- Using the DS values, determine a new list of the top 20% most similar users.
6- Again, use this updated similarity to predict missing ratings for each target user.
7- Compare the lists of top users from steps 2 and 5. Discuss any differences or patterns observed.
8- Compare the rating predictions from steps 3 and 6. Provide comments on the impact of using DS.
9- Find users who get a perfect 1.0 cosine similarity however they have different number of rated items. Discuss why this happens?
10- Find users who rated the common items with any of the target users, can we trust them more than who rated more items than the common items?
11- What happens if some users ratings are far below the average, however the Cosine is still high; does that mean they agree with others?
12- Should a high cosine similarity always mean strong agreement?
13- Give your comments in a separate section in your report.

Case study 2:

1- Repeat the process in Case Study 1 but this time, compute mean-centering Cosine similarity.
2- Identify the top 20% most similar users to each target user based on the adjusted similarities.
3- Use these users to predict the unknown ratings.
4- Calculate DF and DS using threshold β.
5- Based on the DS values, identify the top 20% similar users.
6- Predict ratings again using these updated similarities.
7- Compare the top-k user lists from steps 2 and 5 and comment on any changes.
8- Compare predictions from steps 3 and 6 and discuss the differences.
9- Find the users whose mean-centered similarity became -1.0 instead of highly +ve similarity in raw Cosine. Does that make the users who highly +ve raw and men-centered Cosine similarities more reliable?
10- If one some users rate only their favorite items, and other users rate a full list of items, do they appear unfairly close or far?
11- Give your comments in a separate section in your report.

Case study 3:

1- Use Pearson Correlation Coefficient (PCC) to compute similarity between each target user and others.
2- Determine the top 20% closest users for each target user.
3- Predict the missing ratings using these selected users.
4- Is it correct to compute DF and DS considering threshold β?
5- If it is correct, identify the top 20% users using the discounted similarities.
6- Use these for a second round of prediction.
7- Compare similarity lists from steps 2 and 5. Comment.

8- Compare prediction results from steps 3 and 6. Discuss.
9- Find the users whose Pearson correlation became negative even though their cosine was +ve, show why this happens?
10- Does Pearson give meaningful output when two users rate ≤20% of the items?
11- Find users who have very different rating scales; one always generous, one always strict. Discuss what happens in this case?
12- Find cases where Pearson detected opposite patterns compared to the raw Cosine similarity but based on a smaller number of ratings; do we still trust it?
13- Give your comments in a separate section in your report.

- **Final Task for Part 1:**
  Compare and reflect on the outcomes across Case Studies 1, 2, and 3, considering the impact of similarity metrics and bias adjustment.

3.2.2. Part 2: Item-Based Collaborative Filtering Requirements and Questions:

Part 2 includes the following TWO case studies.

Case study 1:

1. Apply item-based collaborative filtering using Cosine similarity with mean-centering.
2. Identify the top 20% of similar items for each target item.
3. Predict missing ratings using these items.
4. Compute DF and DS.
5. Select top 20% items using DS.
6. Use these for updated rating predictions.
7. Compare similarity lists from steps 2 and 5. Provide commentary.
8. Compare predicted ratings from steps 3 and 6. Discuss.
9. Give your comments in a separate section in your report.

Case study 2:

1- Use PCC to compute similarity between target items.
2- Identify the top 20% most similar items.
3- Predict the missing ratings.
4- Compute DF and DS using threshold β.
5- Select the top 20% items based on discounted similarity.
6- Predict ratings again with this new selection.
7- Compare item lists from steps 2 and 5. Provide analysis.
8- Compare predictions from steps 3 and 6. Share insights.
9- Give your comments in a separate section in your report

- **Final Task for Part 2:**
  Compare the outcomes across Case Studies 1, 2, and 3, highlighting differences in prediction performance due to similarity measures and mean-centering.

3.2.3. Discussion and Conclusion:

prepare a report that includes the following

1. A section titled "Outcomes" summarizing the key findings and results from all parts and case studies.
2. A section titled "Summary of the Comparison of Part 1 and Part 2", synthesizing the comparative insights between user-based and item-based filtering, particularly focusing on the impact of applying significance weighting (discounting) in the top-N similarity selection and its effect on prediction accuracy.
3. A conclusion section presenting your reflections and takeaways, emphasizing the observed improvements or limitations introduced by significance weighing and offering your perspective on its practical value.

3.3. Section THREE: Clustering-based Collaborative Filters requirements

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

In this part, you will apply K-means clustering to group users based on their average rating behavior. Users with similar rating patterns will be clustered together, reducing the search space for finding similar users during the collaborative filtering process. This approach can significantly improve computational efficiency while maintaining recommendation quality.

Tasks and questions

1. Use the calculated average rating given by each user ($\bar{r}_u$) from Section ONE.
2. Create a 1-dimensional feature vector for each user where the feature is their average rating value.
3. Calculate the mean of the users' average ratings, $\mu = \frac{\sum_{u=1}^{N} \bar{r}_u}{N}$
4. Compute the Standard deviation of the users' average ratings, $\sigma = \sqrt{\frac{\sum_{u=1}^{N} (\bar{r}_u - \mu)^2}{N}}$
5. Normalize the feature values for each user using standardization (Z-score normalization) to ensure proper clustering, $z_u = \frac{\bar{r}_u - \mu}{\sigma}$
6. Apply K-means clustering with different values of K (e.g., K = 5, 10, 15, 20, 30, 50):
   - 6.1. For each K value, calculate and save the cluster centroids and perform K-means clustering on the user feature vectors.
   - 6.2. Record the cluster assignments for all users.
7. Analyze the clustering results for each K value:
   - 7.1. Calculate the number of users in each cluster.
   - 7.2. Compute the within-cluster sum of squares (WCSS) for each K.
   - 7.3. Plot the elbow curve (WCSS vs. K) to determine the optimal K value.
   - 7.4. Calculate the silhouette score for each K value to assess clustering quality.
8. For the optimal K value (based on elbow method and silhouette score):
   - 8.1. Display the distribution of users across clusters (create a bar chart).
   - 8.2. Show the average rating value for each cluster centroid.
   - 8.3. Identify which clusters contain generous raters (high average) and which contain strict raters (low average).
9. Apply user-based collaborative filtering within each cluster:

9.1. For each target user from Section ONE (U1, U2, U3), identify which cluster they belong to.
9.2. Within the assigned cluster, compute user-user similarity using mean-centered Cosine similarity.
9.3. Select the top 20% most similar users from within the same cluster.
9.4. Predict ratings for the target items (I1 and I2) using only the similar users from the same cluster

10. Compare clustering-based predictions with non-clustering predictions from Section TWO:
    10.1. For each target user, compare the predicted ratings with and without clustering.
    10.2. Calculate the difference in prediction values.
    10.3. Discuss whether clustering improved, maintained, or degraded prediction accuracy.
11. Analyze the computational efficiency gains:
    11.1. Calculate the number of similarity computations needed without clustering (comparing all user pairs).
    11.2. Calculate the number of similarity computations needed with clustering (only within-cluster pairs).
    11.3. Compute the speedup factor: $\left(\frac{computations\ without\ clustering}{omputations\ with\ clustering}\right)$.
    11.4. Express the efficiency gain as a percentage reduction in computations.
12. Evaluate the impact of cluster imbalance:
    12.1. Identify if any clusters are significantly larger or smaller than others.
    12.2. Discuss how cluster imbalance affects computational efficiency.
    12.3. Propose strategies to handle imbalanced clusters.
13. Test the robustness of the clustering approach:
    13.1. Re-run K-means with different random initializations (at least 3 times).
    13.2. Compare the cluster assignments across different runs.
    13.3. Discuss whether the clustering is stable or varies significantly with initialization.
14. Include the results of all the above points in your report and give your insights and comments in a separate section on:
    14.1. The effectiveness of clustering based on average user ratings.
    14.2. The trade-off between prediction accuracy and computational efficiency.
    14.3. Whether this clustering strategy is suitable for your dataset characteristics.
    14.4. How the choice of K affects both accuracy and efficiency.

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

In this part, you will cluster users based on the number of common items they have rated with other users. This approach aims to group users who have overlapping rating histories, which can lead to more reliable similarity computations since the similarities will be based on a sufficient number of co-rated items. This addresses the significance weighting concept discussed in Section TWO.

Tasks and questions

1. Compute the co-rating statistics for each user:
   1.1. For each user, calculate the average number of common items they have with other users.
   1.2. For each user, calculate the maximum number of common items they have with any other user.
   1.3. For each user, calculate the minimum number of common items they have with any other user (excluding zero).
   1.4. Create a feature vector for each user: [average_common, max_common, min_common].
2. Normalize the feature vectors:
   2.1. Apply Z-score standardization to each feature dimension separately:
      2.1.1. For each feature, calculate its mean (μ) and standard deviation (σ) across all users.
      2.1.2. Normalize each feature dimension independently using: $z_{u'f_i} = \frac{x_{u,f_i} - \mu_{f_i}}{\sigma_{f_i}}$
         - x_{u,f_i}= value of feature f_i for user u.
         - μ_{f_i}= mean of feature f_i across all users.
         - σ_{f_i}= standard deviation of feature f_i across all users.

      This normalization ensures all features contribute equally to the clustering by scaling them to comparable ranges (mean = 0, standard deviation = 1), preventing features with larger numerical scales from dominating the distance calculations.
3. Apply K-means clustering with different K values (K = 5, 10, 15, 20, 30, 50):
   3.1. Perform K-means clustering on the normalized feature vectors.
   3.2. Record cluster assignments and centroids for each K.
   3.3. Calculate WCSS and silhouette scores for each K.
4. Determine the optimal K value:
   4.1. Plot the elbow curve (WCSS vs. K).
   4.2. Plot silhouette scores vs. K.
   4.3. Select the optimal K based on both metrics.
5. Analyze the characteristics of each cluster (using optimal K):
   5.1. Calculate the average co-rating statistics for users in each cluster.
   5.2. Identify 'high overlap' clusters (users who share many common ratings with others).
   5.3. Identify 'low overlap' clusters (users with few common ratings with others).
   5.4. Visualize the cluster distribution using a 2D or 3D scatter plot.
6. Apply collaborative filtering within clusters:
   6.1. For each target user (U1, U2, U3), identify their cluster assignment.
   6.2. Within each cluster, compute user-user similarity using mean-centered Cosine similarity.
   6.3. Calculate the Discount Factor (DF) based on the number of common ratings (using threshold β from Section ONE) & Compute Discounted Similarity (DS).
   6.4. Select the top 20% most similar users based on DS within the cluster.

6.5. Predict ratings for target items (I1 and I2) using the selected similar users.

7. Compare this clustering approach with Part 1:
   7.1. Compare the predicted ratings from Part 2 with those from Part 1.
   7.2. Which clustering strategy produces more accurate predictions?
   7.3. Which strategy results in better computational efficiency?
   7.4. Compare the cluster memberships between the two approaches:
      7.4.1. Do the same users end up in the same clusters?
      7.4.2. How do the characteristics of users within each cluster differ between Part 1 and Part 2?
      7.4.3. Which approach produces more cohesive/meaningful user groups?
8. Evaluate the relationship between common ratings and prediction quality:
   8.1. For each target user, calculate the average number of common ratings with their top similar users.
   8.2. Calculate the prediction error for each target user: |actual rating - predicted rating|
   8.3. Is there a correlation between the number of common ratings and prediction accuracy?
9. Analyze the impact on significance weighting:
   9.1. Compare the distribution of DF values within each cluster.
   9.2. Do users in the same cluster tend to have similar DF values?
   9.3. How does clustering affect the effectiveness of significance weighting?
10. Examine extreme cases and challenges:
   10.1. Identify users who have very few common ratings with anyone in their cluster.
   10.2. How should these users be handled?
   10.3. Discuss the cold-start problem within this clustering approach.
11. Include the results of all the above points in your report and give your insights and comments in a separate section on:
   11.1. The effectiveness of clustering based on common rating patterns.
   11.2. How this approach addresses the significance weighting problem from Section TWO.
   11.3. The advantages and disadvantages compared to average rating-based clustering (Part 1).
   11.4. Recommendations for when to use this clustering strategy.

3.3.3. Part 3: K-means Clustering based on average number of raters

In this part, you will cluster items (rather than users) based on how many users have rated each item. This item-based clustering approach addresses the long-tail problem identified in Section ONE, where some items have many ratings while others have very few. By clustering items with similar rating frequencies, you can improve the reliability of item-based collaborative filtering.

Tasks and questions

1. Compute item statistics:
   1.1. For each item, use the total number of raters and the average rating it received from Section ONE.
   1.2. For each item, calculate the standard deviation of its ratings.
   1.3. Create a feature vector for each item: [num_raters, avg_rating, std_rating].
2. Normalize the feature vectors:
   2.1. Apply Z-score standardization independently to each feature dimension:
      2.1.1. For each feature, calculate its mean (μ) and standard deviation (σ) across all items.
      2.1.2. Normalize each feature dimension independently using: $z_{i,f_j} = \frac{x_{i,f_j} - \mu_{f_j}}{\sigma_{f_j}}$

      x_{i,f_j}= value of feature f_j for item i.

      μ_{f_j}= mean of feature f_j across all items.

      σ_{f_j}= standard deviation of feature f_j across all items.

      This normalization ensures all features contribute equally to the clustering by scaling them to comparable ranges (mean = 0, standard deviation = 1), preventing features with larger numerical scales from dominating the distance calculations.
   2.2. Verify that all features are now on the same scale (mean = 0, standard deviation = 1), ensuring equal contribution to the clustering process.
3. Apply K-means clustering to items with different K values (K = 5, 10, 15, 20, 30, 50):
   3.1. Perform K-means clustering on item feature vectors.
   3.2. Record cluster assignments for all items.
   3.3. Calculate WCSS and silhouette scores for each K.
4. Determine the optimal K value:
   4.1. Plot the elbow curve and silhouette scores.
   4.2. Select the optimal K value.
5. Analyze the characteristics of each item cluster (using optimal K):
   5.1. Calculate the average number of raters for items in each cluster.
   5.2. Identify 'popular item' clusters (high number of raters).
   5.3. Identify 'niche item' clusters (low number of raters).
   5.4. Identify 'long-tail item' clusters (very few raters).
   5.5. Visualize the distribution of items across clusters.
6. Analyze the relationship between cluster membership and item popularity:
   6.1. Plot the distribution of number of raters within each cluster.
   6.2. Are items with similar popularity levels grouped together?
   6.3. Analyze how items from different parts of the popularity distribution (head vs. tail from Section ONE) are distributed across the clusters. Are popular and unpopular items separated into different clusters, or mixed within clusters? What does this reveal about the clustering basis?
7. Apply item-based collaborative filtering within clusters:

7.1. For each target item (I1 and I2 from Section ONE), identify their cluster assignment.
7.2. Within each cluster, compute item-item similarity using Adjusted Cosine similarity.
7.3. Select the top 20% most similar items from within the same cluster.
7.4. For each target user (U1, U2, U3), predict their rating for the target items using only similar items from the same cluster.

8. Compare clustering-based item CF with non-clustering item CF from Section TWO:
   8.1. Compare the predicted ratings with and without clustering.
   8.2. Calculate the prediction error for each approach: |actual rating - predicted rating|.
   8.3. Which approach produces more reliable predictions?
9. Evaluate the impact on the long-tail problem:
   9.1. How does clustering affect predictions for items with very few ratings (long-tail items)?
   9.2. Are predictions for long-tail items more or less reliable within their clusters?
   9.3. Compare the number of similar items found for long-tail items with and without clustering.
10. Analyze the computational efficiency:
   10.1. Calculate the reduction in item-item similarity computations due to clustering.
   10.2. Compute the speedup factor compared to non-clustering item-based CF.
   10.3. Is the speedup greater for item-based or user-based clustering?
11. Examine the effect of cluster size on prediction quality:
   11.1. For clusters of different sizes, calculate the average prediction error.
   11.2. Do larger clusters produce better or worse predictions?
   11.3. Is there an optimal cluster size for balancing accuracy and efficiency?
12. Compare user-based clustering (Parts 1 & 2) with item-based clustering (Part 3):
   12.1. Which clustering approach (user or item) is more effective for your dataset?
   12.2. When would you recommend user-based clustering vs. item-based clustering?
   12.3. Can both clustering strategies be combined? Discuss the feasibility and benefits.
13. Include the results of all the above points in your report and give your insights and comments in a separate section on:
   13.1. The effectiveness of item-based clustering for addressing the long-tail problem.
   13.2. The relationship between item popularity and clustering quality.
   13.3. Comparison between user-based and item-based clustering strategies.
   13.4. Recommendations for practical deployment of item-based clustering.

3.3.4. Part 4: K-means Clustering for Cold-Start

In this part, you will develop a clustering-based approach to handle the cold-start problem; the challenge of making recommendations for new users who have provided

only a few ratings, or recommending new items with limited rating history. You will create strategies to assign cold-start users/items to appropriate clusters and generate meaningful recommendations despite limited data.

Tasks and questions

1. Simulate cold-start scenarios:
   - 1.1. From your dataset, randomly select 100 users with more than 50 ratings each.
   - 1.2. For each selected user, hide 80% of their ratings to simulate a cold-start user with only 10-20 ratings.
   - 1.3. Store the hidden ratings as ground truth for evaluation.
   - 1.4. Similarly, select 50 items with many ratings and hide most ratings to simulate cold-start items.
2. Develop a cold-start user assignment strategy:
   - 2.1. For each cold-start user, calculate their limited profile features (e.g., average rating from their few ratings).
   - 2.2. Compute the distance between the cold-start user's feature vector and each cluster centroid from Part 1.
   - 2.3. Assign the cold-start user to the nearest cluster.
   - 2.4. Record the confidence of the assignment (e.g., distance to nearest centroid vs. distance to second-nearest centroid).
3. Generate recommendations for cold-start users:
   - 3.1. Within the assigned cluster, find the most similar users based on the limited rating overlap.
   - 3.2. Use these similar users to predict ratings for items the cold-start user hasn't rated.
   - 3.3. Generate top-10 item recommendations for each cold-start user.
4. Evaluate cold-start user recommendations:
   - 4.1. Compare the predicted ratings with the hidden ground truth ratings.
   - 4.2. Calculate the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).
   - 4.3. Compute precision@10 and recall@10 for the top-10 recommendations.

     $$Precision@10 = \frac{number\ of\ relevant\ items\ in\ top\ 10}{10}$$

     $$Recall@10 = \frac{number\ of\ relevant\ items\ in\ top\ 10}{Total\ number\ of\ relvant\ items}$$
   - 4.4. Compare the accuracy with and without clustering for cold-start users.
5. Develop a cold-start item assignment strategy:
   - 5.1. For each cold-start item, calculate its limited profile (e.g., average rating, number of raters).
   - 5.2. Assign the item to the nearest item cluster from Part 3.
   - 5.3. Calculate the confidence of each cold-start item assignment using the following steps:
     - 5.3.1. Compute the Euclidean distance from the cold-start item to its assigned (nearest) cluster centroid: $d_{nearest}$.
     - 5.3.2. Compute the Euclidean distance from the cold-start item to the second-nearest cluster centroid: d_{second}.
     - 5.3.3. For each cold-start item, calculate the assignment confidence using:

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

Where:

- Values close to 1.0 indicate high confidence (the item clearly belongs to the assigned cluster)
- Values close to 0.0 indicate low confidence (the item is nearly equidistant between clusters)

5.3.4. Present your results in a table showing: cold-start item ID, assigned cluster, $d_{nearest}$, d_{second}, and confidence score.

6. Generate recommendations involving cold-start items:
   - 6.1. For users who might be interested in the cold-start item, predict their ratings.
   - 6.2. Use similar items within the same cluster to make predictions.
   - 6.3. Identify which users would most likely rate the cold-start item highly.
7. Evaluate cold-start item predictions:
   - 7.1. Compare predicted ratings with hidden ground truth.
   - 7.2. Calculate MAE and RMSE for cold-start item predictions.
   - 7.3. Compare accuracy with and without clustering.
8. Analyze the relationship between the number of ratings and prediction accuracy:
   - 8.1. For cold-start users with 5, 10, 15, and 20 ratings, measure prediction accuracy.
   - 8.2. Plot a curve showing how accuracy improves as the number of ratings increases.
   - 8.3. At what point does a user transition from 'cold-start' to having sufficient data?
9. Develop a hybrid cold-start strategy:
   - 9.1. Combine clustering-based CF with content-based features (if available).
   - 9.2. Use demographic information or item attributes to improve cold-start cluster assignment.
   - 9.3. Evaluate whether the hybrid approach improves prediction accuracy.
10. Test the robustness of cold-start handling:
    - 10.1. Vary the amount of information available for cold-start users (e.g., 3, 5, 10, 20 ratings).
    - 10.2. Measure how prediction quality degrades as less information is available.
    - 10.3. Identify the minimum number of ratings needed for acceptable prediction quality.
11. Analyze cluster assignment confidence:
    - 11.1. For cold-start users/items, calculate the confidence of cluster assignments:

      Compute the ratio: Ratio $= \frac{d_{nearest}}{d_{second}}$

      Where $d_{nearest}$ is the distance to the assigned cluster and d_{second} is the distance to the second-nearest cluster

      Lower ratios (<0.5) indicate confident assignments; higher ratios (>0.7) indicate ambiguous assignments.
    - 11.2. Identify cases where cluster assignment is ambiguous (similar distances to multiple clusters).
    - 11.3. Propose strategies for handling ambiguous cases (e.g., multi-cluster membership, weighted recommendations).

12. Compare different cold-start strategies:
    12.1. Strategy 1: Assign to nearest cluster and use cluster-based CF.
    12.2. Strategy 2: Use global (non-clustered) CF with available ratings.
    12.3. Strategy 3: Use popularity-based recommendations.
    12.4. Compare the accuracy and efficiency of each strategy.
13. Analyze the impact of cluster granularity on cold-start performance:
    13.1. Test cold-start performance with different K values (K = 5, 10, 20, 50).
    13.2. Does a smaller K (fewer, larger #clusters) or larger K (more, smaller #clusters) work better for cold-start?
    13.3. Discuss the trade-off between cluster specificity and data availability.
14. Develop a confidence-based recommendation strategy:
    14.1. For each cold-start recommendation, compute a confidence score based on:
        a. Cluster assignment confidence
        b. Number of similar users/items found
        c. Agreement among similar users/items
    14.2. Use confidence scores to filter out low-confidence recommendations.
    14.3. Evaluate whether filtering improves overall recommendation quality.
15. Include the results of all the above points in your report and give your insights and comments in a separate section on:
    15.1. The effectiveness of clustering for handling cold-start problems.
    15.2. Which cold-start strategy performed best and why.
    15.3. The minimum amount of data needed to make reliable recommendations.
    15.4. Practical recommendations for deploying cold-start solutions in real systems.
    15.5. How clustering reduces the cold-start problem compared to non-clustering approaches.

3.4. Discussion and Conclusion for Section THREE:

1. A section titled "Clustering Strategies Comparison" that:
    1.1. Compares all four clustering approaches (Parts 1-4) in terms of accuracy, efficiency, and applicability.
    1.2. Provides a summary table showing key metrics for each approach.
    1.3. Discusses the strengths and weaknesses of each clustering strategy.
2. A section titled "Computational Efficiency Analysis" that:
    2.1. Quantifies the speedup achieved by each clustering approach.
    2.2. Compares memory requirements with and without clustering.
    2.3. Discusses the scalability improvements from clustering.
3. A section titled "Clustering vs. Non-Clustering Tradeoffs" that:
    3.1. Analyzes the accuracy-efficiency trade-off for each clustering strategy.
    3.2. Discusses when clustering is beneficial and when it may not be necessary.
    3.3. Provides recommendations for choosing between clustering approaches based on dataset characteristics.
4. A section titled "Cold-Start Problem Solution" that:
    4.1. Summarizes the effectiveness of clustering for cold-start scenarios.

4.2. Compares cold-start performance across different clustering strategies.
4.3. Provides practical guidelines for handling cold-start users and items.

5. A conclusion section that:
   5.1. Presents your overall findings from all clustering experiments.
   5.2. Discusses which clustering strategy you would recommend for your dataset and why.
   5.3. Reflects on how clustering addresses the challenges identified in Section ONE (sparsity, bias, long-tail).
   5.4. Provides insights on the practical deployment of clustering-based collaborative filtering systems.

4. Coursework Assessment Criteria

- Weighting: This coursework contributes **10%** to the overall course grade and must be completed individually.
- Purpose: The task is designed to evaluate the progression of the student's academic capabilities and to assess the achievement of the course's Intended Learning Outcomes (ILOs).
- Assessment Criteria: Marks will be allocated proportionally based on the following aspects:
  - Depth and accuracy of the technical implementation.
  - Evidence of academic rigor and critical thinking.
  - Clarity and organization of information.
  - Quality and professionalism of the written discussion.
- Penalty for Lack of Thoroughness: If a submission lacks sufficient evidence of thorough analysis and methodology, a **40%** mark reduction will be applied.
- Partial Credit for Near-Miss Attempts: In cases where the student's approach is conceptually sound but contains errors, the maximum score awarded for that section will be capped at **70%**.

5. Written Report (paper)

5.1. Report content

- Cover Page should include:
  - Line 1: AIE425 Intelligent Recommender Systems, Fall Semester 25/26
  - Line 2: Assignment 1: Neighborhood CF & Clustering in CF
  - Line 3: Student ID, Full Name, and Part Assignments.
- Executive summary (1 page maximum)
- Complete solutions to all tasks with mathematical calculations shown step-by-step, tables and visualizations, analysis and interpretations.
- Conclusions and recommendations
- References (if any external sources used)
- File naming: AIE425_Assignment 1_Group[X]_Report

5.2.Report Format:

Your report must follow these formatting standards:

- Must be typed and follow a professional report format and on A4-sized paper.

- Use correct grammar, formal language, appropriate tenses, and standard English spelling.
- The cover page must have double line spacing, be center-aligned, and not numbered.
- All other pages:
  - Use 1.5 line spacing.
  - Use Arial 12 pt font for body text and bold Arial 12 pt for section headings.
  - Page numbers must appear centered at the bottom in the format: Page X of Y.
  - Headings must maintain uniform font, size, and alignment.

6. Plagiarism and Academic Honesty

- Integrity and Collaboration: While discussing concepts with peers is encouraged, all submitted work must be individual. Any collaboration must be clearly acknowledged.
- Copying code from online sources without attribution is strictly prohibited and may result in course failure.
- Submit your own original work. If you use content not created by you, it must be explicitly cited and properly referenced. Code plagiarism from other groups will result in ZERO for all involved parties.
- Avoid using external code libraries unless pre-approved. If necessary proper citations are required for external datasets, code libraries and frameworks, or research papers.
- A Plagiarism Report is mandatory with submission. Submissions exceeding 30% similarity will be rejected.

7. Feedback given to students in response to assessed work.

- The written component will be assessed directly through annotations on the page.
- Feedback for programming will also be placed on the coursework assessment sheet returned with the coursework mark.
- During contact hours, students will receive oral generic comments.
- If students need more input, they are encouraged to speak with the teaching staff.

8. Deliverables and Submission Instructions

The complete assignment, including the report, dataset, and source code, must be submitted via CANVAS and GitHub.

- **CANVAS Submission**, submit a **ZIP file** containing:
  - Written report (PDF)
  - Complete source code
  - Dataset files (or link to dataset in README.md if too large)
  - README.md and requirements.txt
  - File naming convention: AIE425_Assignment_Group[X].zip
- **GitHub Submission**:
  - Create a public repository named: AIE425-Assignment-Group[X]
  - Push all code, documentation, and results to GitHub
  - Submit the GitHub repository link on CANVAS in the submission comments
  - Ensure the repository is accessible (add instructor & TA as collaborators)
  - Source Code (GitHub Repository):
    - Well-documented Python code (.ipynb or .py).

- Ensure all files are accessible and can be opened in Visual Studio Code without modifications.
- Organized folder structure:

```
/dataset
/section1_statistical_analysis
/section2_neighborhood_cf
  /part1_user_based_cf
  /part2_item_based_cf
/section3_clustering_based_cf
  /part1_user_clustering_avg_ratings
  /part2_user_clustering_common_ratings
  /part3_item_clustering_avg_raters
  /part4_cold_start_clustering
/utils (helper functions)
/results (output files, visualizations)
README.md
requirements.txt
```

- Code requirements:
  Clear comments explaining each major step; Function docstrings; Modular design (separate files for each part); Reproducible results; All code must be runnable in Visual Studio Code without modifications; Avoid absolute file paths (use relative paths only); No proprietary or platform-specific code.

- All file paths in your code must be relative to the code file location. Do not use absolute paths (e.g., C:/Users/...) to ensure your code runs correctly when evaluated on a different machine.
- Your full name and student ID must appear at the top of each submitted file, including within the code files.
- Late Submission Policy:
  - Within 24 hours with valid justification and prior arrangement: allowed, but with a **50%** grade deduction.
  - **Beyond 24 hours not accepted**, and the project will receive a grade of **ZERO**.

9. Assessment Breakdown:

1. Group Work – 70%

Component	Weight	Description
Technical Correctness	30%	Accurate implementation of all algorithms, similarity metrics, clustering, predictions, discounting, and computational analysis.
Code Quality & Documentation	15%	Clean, modular, well-structured, and well-commented code stored in a properly organized GitHub repository.
Report Quality & Analysis	15%	Clear writing, correct interpretation of results, insightful analysis, professional formatting, correct integration of tables/figures.
Completeness	10%	All required parts completed; all figures, plots, saved outputs, and intermediate files included; all deliverables submitted on time.

2. Individual Work – 30%

Component	Weight	Description
Week 11 Lab Implementation	30%	Each student must complete one assigned coding task during the Week 11 lab session and push the solution to GitHub. Failure to complete the task results in ZERO for this component.

3. Breakdown by Assignment Sections & Parts

Component / Section / Part	Weight	Notes
Section 1: Statistical Analysis	20%	Dataset cleaning, processing, rating distributions, sparsity, long-tail, bias analysis, and selecting target users/items.
Section 2: Neighborhood CF Filters (Total = 40%)		
Part 1: User-based CF	20%	Cosine/Mean-centered/Pearson similarities, DF/DS, predictions, comparisons.
Part 2: Item-based CF	20%	Item–item similarities, DF/DS, predictions, comparisons.
Section 3: Clustering-based Collaborative Filters (Total = 40%)		
Part 1: K-means on average user ratings	10%	Feature extraction, SSE, silhouette, cluster analysis.
Part 2: K-means on common ratings	10%	Co-rating features, K-means, evaluation, analysis.
Part 3: K-means on number of raters	10%	Item popularity clustering, SSE, silhouette, analysis.
Part 4: K-means for Cold-Start	10%	Multi-feature clustering for cold users/items, evaluation, interpretation.

---------- END ----------