Practical Assignment 2

Collaborative Filtering with MovieLens 100K

Students:

Osadici Darius, Rădulescu Adelina

November 18, 2025

Table of Contents

Task 1: Finding Optimal K for User-Based KNN

1.1 Optimal K with 25% Missing Ratings

We tested K values from 5 to 250 to find the optimal number of neighbors for user-based KNN. The results show K=80 achieves the best performance with MAE=0.7452.

The most significant improvements occur with smaller K values - going from K=5 to K=30 reduces MAE by 0.0561 (about 7% improvement). After K=30, we see clear diminishing returns. Each increase brings smaller gains: K=30 to K=50 only improves by 0.0030, and K=50 to K=80 by just 0.0006.

Beyond K=80, MAE actually starts increasing slightly. At K=90 it rises to 0.7453, and continues worsening through K=250 (MAE=0.7462). This suggests that including more than 80 neighbors introduces noise by averaging in dissimilar users, hurting prediction accuracy.

The sweet spot at K=80 balances two factors: it's large enough to reduce noise through averaging, but small enough to exclude dissimilar users who would hurt predictions. This makes K=80 our optimal choice for the 25% sparsity scenario.

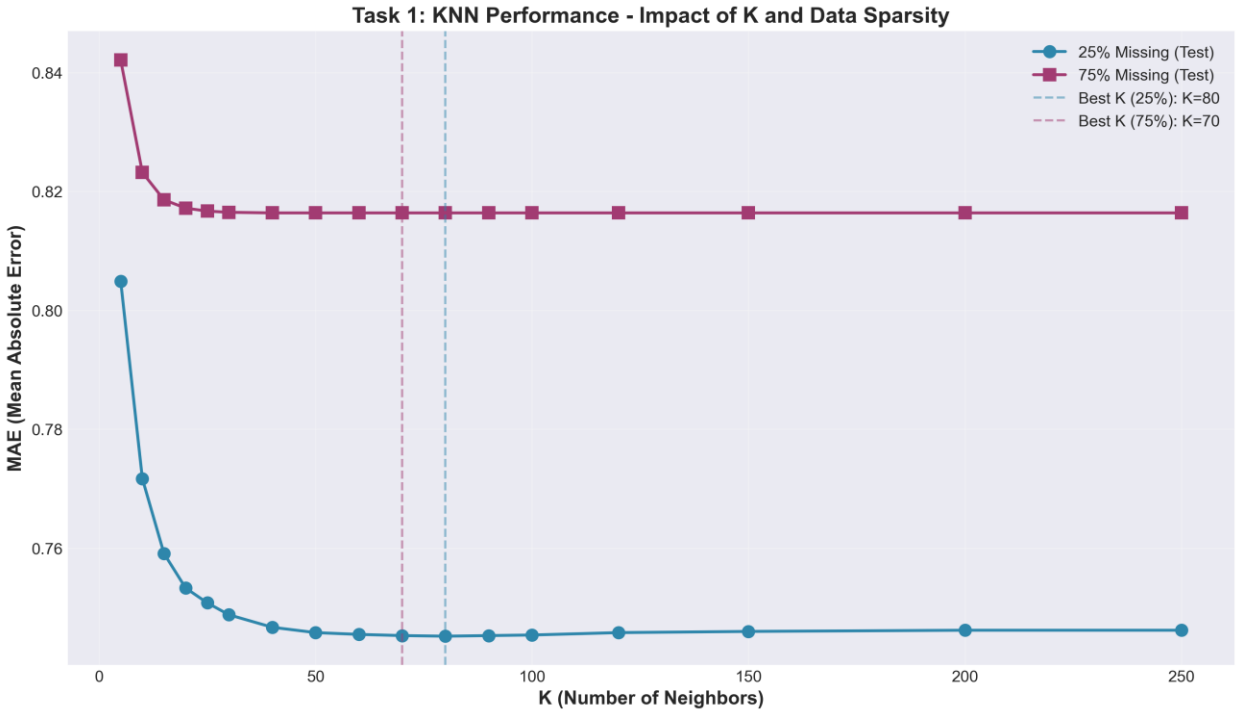

Figure 1: KNN Performance - Impact of K and Data Sparsity

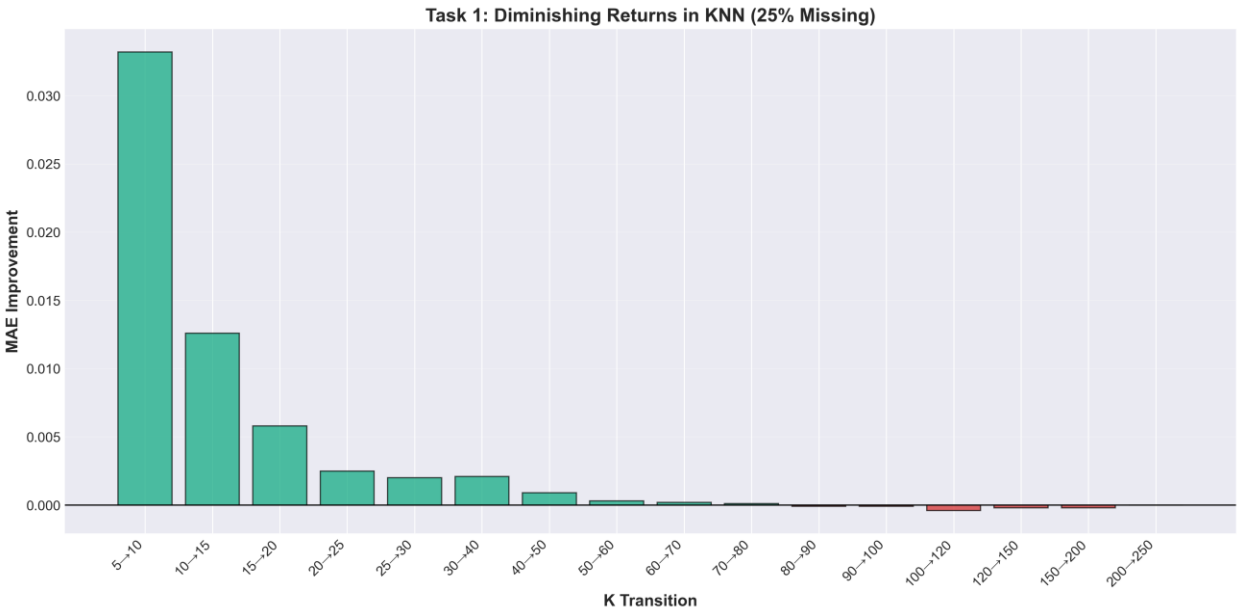

Figure 2: Diminishing Returns Analysis in KNN (25% Missing)

1.2 Optimal K with 75% Missing Ratings (Sparsity Problem)

With 75% of ratings held out for testing, we tested the same K range (5-250). The best performance was K=70 with MAE=0.8164, though K=40 through K=250 all achieved virtually identical results (MAE=0.8164).

The sparsity problem is immediately visible in the higher MAE - 0.8164 compared to 0.7452 with 25% sparsity. That's a 9.5% increase in prediction error. With less training data, the model simply can't learn user preferences as accurately.

Improvements plateau much earlier here than with 25% sparsity. Most of the gains happen from K=5 to K=30 (MAE drops by 0.0256). After K=40, MAE completely flatlines - increasing K further provides zero benefit. This contrasts with 25% sparsity where we saw continued small improvements up to K=80.

This flat plateau reveals the sparsity challenge: with 75% missing ratings, users have fewer rated items in common. Finding more neighbors doesn't help because those neighbors lack sufficient overlap to make better predictions. The algorithm hits a ceiling where additional neighbors contribute no new information. For this scenario, K=70 is technically optimal, but anything from K=40 to K=250 performs identically - the choice doesn't matter once you've included enough neighbors to cover available data.

Task 2: SVD vs KNN - Sparsity Mitigation

We tested SVD (Funk variant) with different numbers of latent factors (5-200) and compared it against the best KNN models from Task 1. SVD outperformed KNN in both scenarios, with dramatically better results under high sparsity.

25% Sparsity: SVD with 10 factors achieved MAE=0.7403 compared to KNN's 0.7452, an improvement of 0.0049 (0.66%). The advantage is modest here because both methods have sufficient training data. Interestingly, increasing factors beyond 10 hurt performance - 200 factors gave MAE=0.7457, worse than KNN. This shows overfitting when the model tries to capture too much detail from limited patterns.

75% Sparsity: This is where SVD truly shines. With just 5 factors, SVD achieved MAE=0.7679 versus KNN's 0.8164 - an improvement of 0.0485 (5.94%). That's nearly 9 times better improvement than with 25% sparsity. More importantly, SVD's MAE of 0.7679 with 75% missing data is actually **better** than KNN's 0.7452 with only 25% missing data. SVD handles sparse data so well it overcomes the information loss**.**

Notice the optimal factor count drops from 10 to 5 as sparsity increases. With less data, fewer latent factors are needed - too many causes overfitting as seen by performance degrading to MAE=0.7822 at 200 factors.

Why SVD mitigates sparsity better: KNN relies on direct user-user similarity, which breaks down when users share few rated items. SVD instead learns latent factors (hidden preferences) that generalize across users, so even sparse data reveals meaningful patterns. This makes SVD the clear winner for sparse recommendation scenarios.

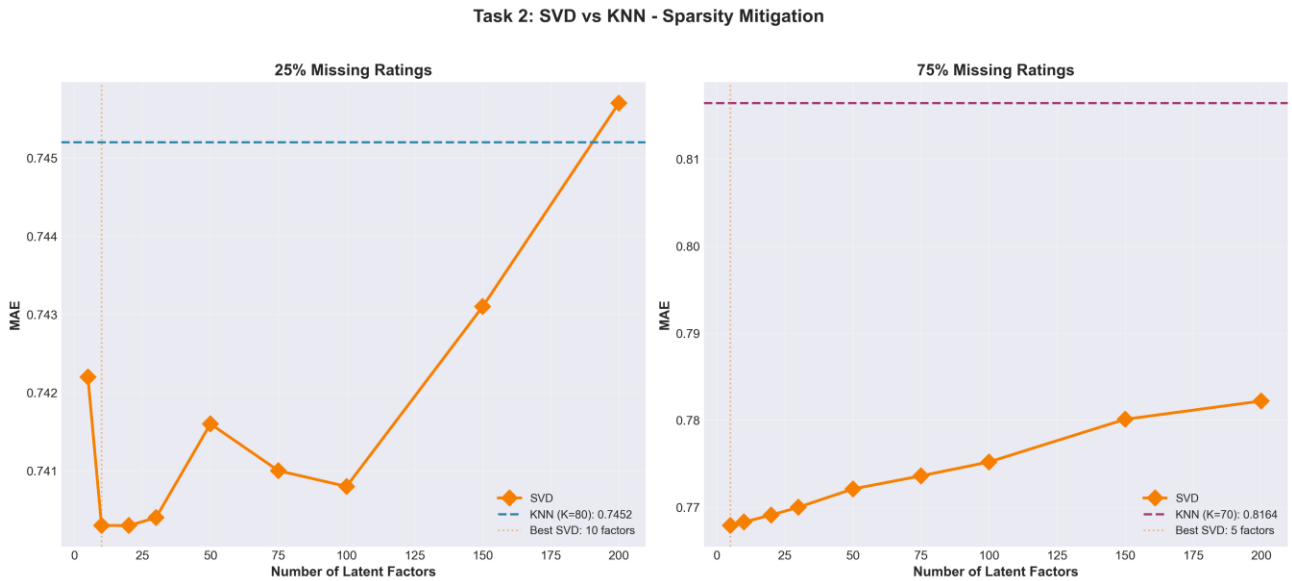

Figure 3: SVD vs KNN - Sparsity Mitigation Comparison

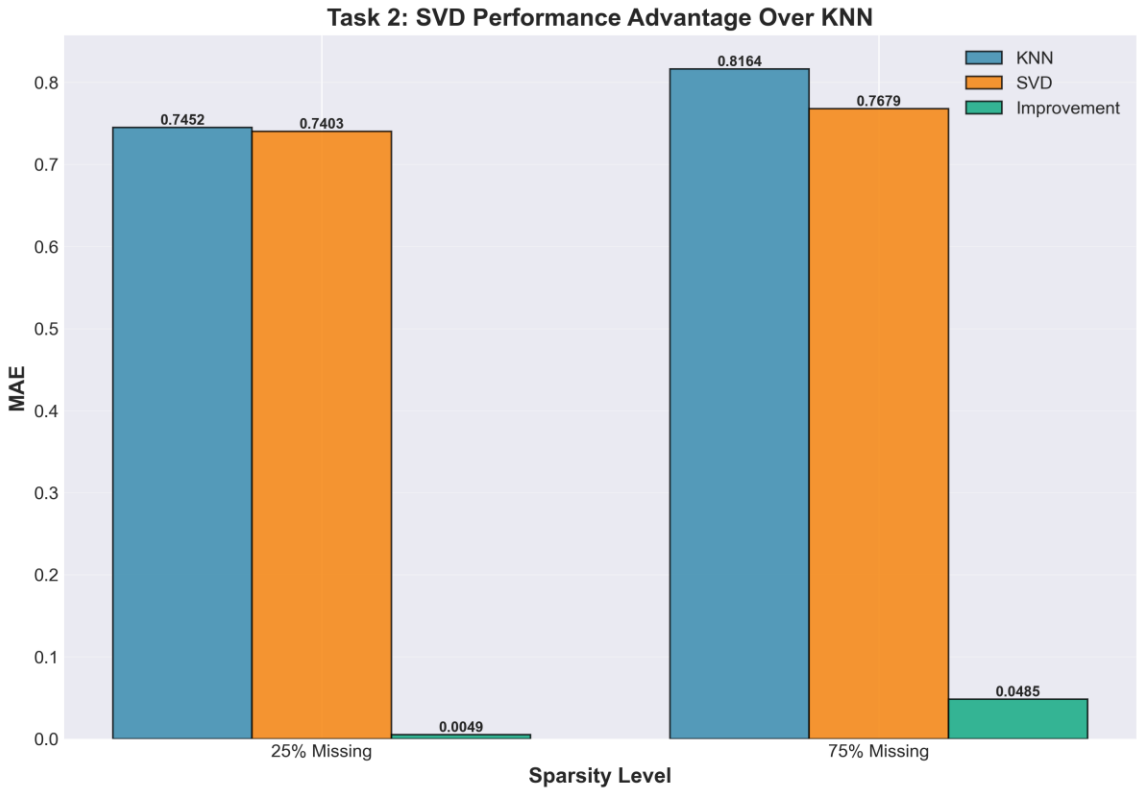

Figure 4: SVD Performance Advantage Over KNN

Task 3: Top-N Recommendations

We evaluated precision, recall, and F1 scores for Top-N recommendations (N=10 to 100) using both KNN and SVD with their optimal parameters. Relevant items were defined as those rated 4 or 5 stars.

****Key Findings:****

- • 25% Sparsity - KNN: Best F1 = 0.6484 at N=10
- • 25% Sparsity - SVD: Best F1 = 0.6454 at N=10
- • 75% Sparsity - KNN: Best F1 = 0.6313 at N=30
- • 75% Sparsity - SVD: Best F1 = 0.6465 at N=30

Precision-Recall Tradeoff: As N increases, recall improves (we capture more relevant items) but precision drops (more non-relevant items are included). This fundamental tradeoff is clearly visible in all scenarios. For example, with KNN at 25% sparsity, precision drops from 0.63 at N=10 to 0.15 at N=100, while recall climbs from 0.67 to 0.99.

Impact of Sparsity on Optimal N: Best F1 scores occur at N=10 for 25% sparsity but shift to N=30 for 75% sparsity. This happens because with sparse data, users have fewer rated items overall, so we need a larger N to achieve good recall. At 25% sparsity, the top 10 items are highly accurate (precision ~0.63), but at 75% sparsity, we need 30 items to balance precision and recall effectively.

SVD vs KNN: SVD consistently matches or slightly beats KNN in F1 scores, especially at 75% sparsity (0.6465 vs 0.6313). This confirms Task 2's findings - SVD handles sparse data better by learning generalizable patterns rather than relying on direct user overlap. The results are remarkably similar at 25% sparsity, showing both methods work well when sufficient data exists.

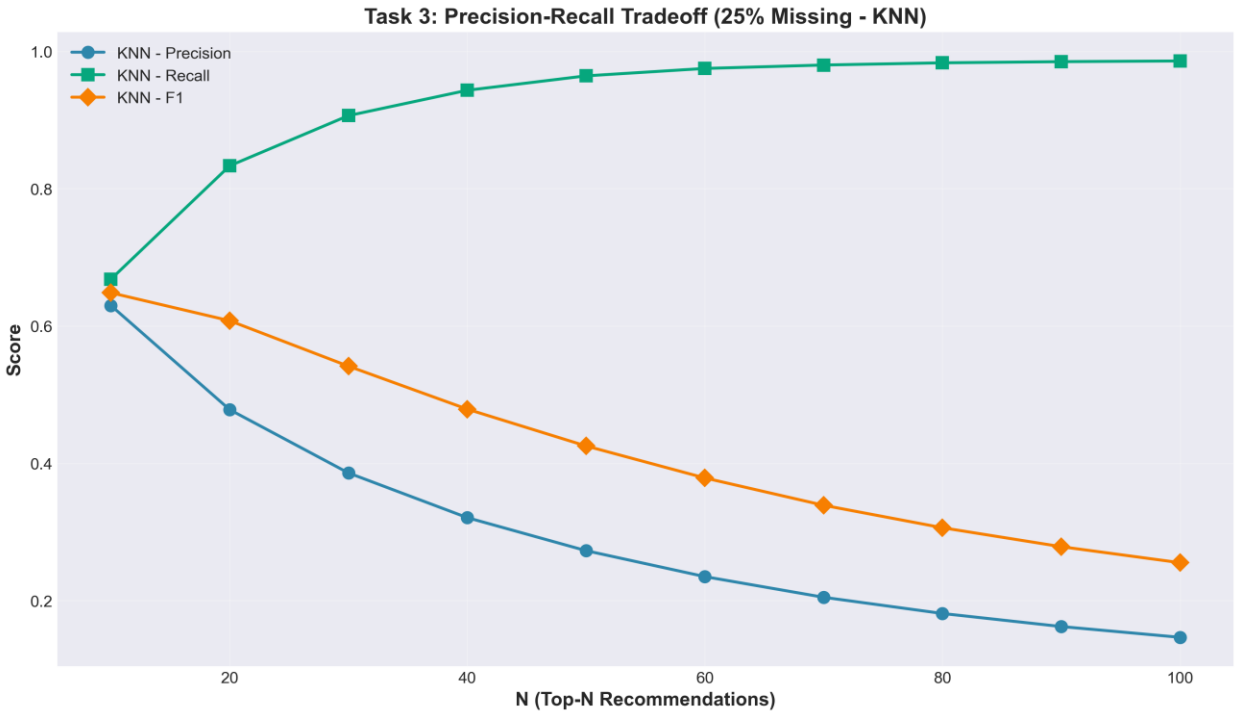

Figure 5: Precision-Recall Tradeoff (25% Missing - KNN)

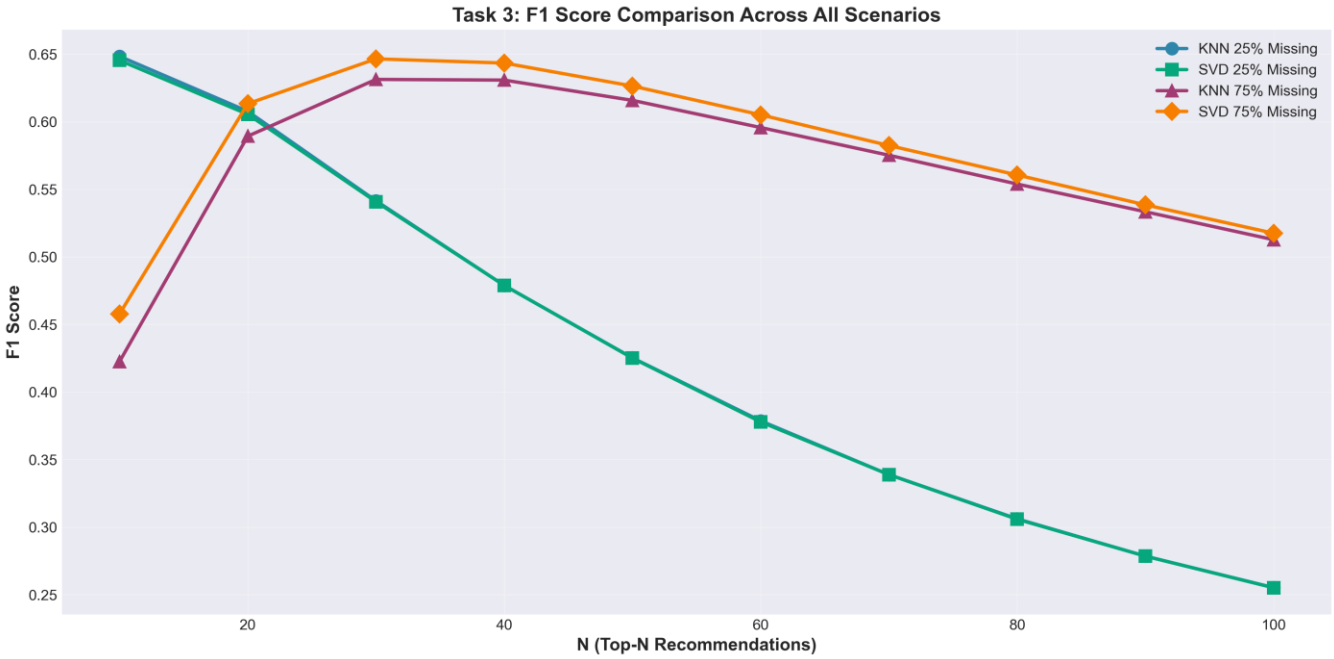

Figure 6: F1 Score Comparison Across All Scenarios

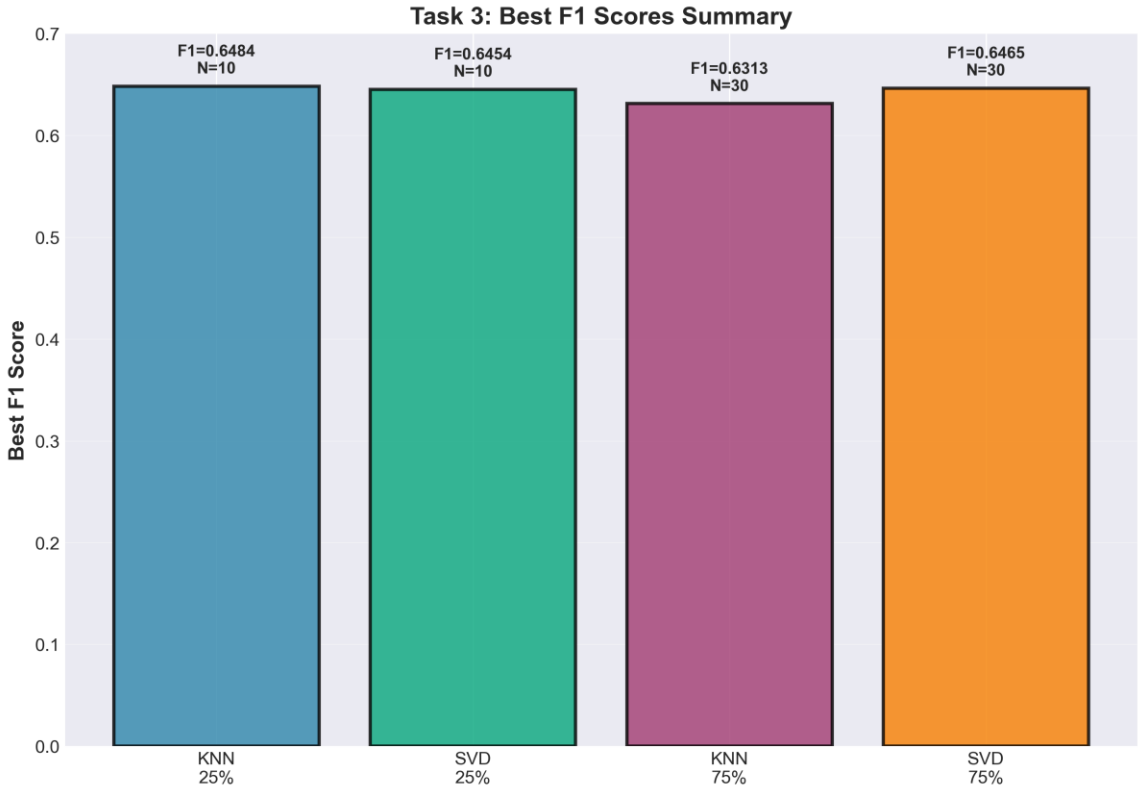

Figure 7: Best F1 Scores Summary

Discussion: Why the Results Make Sense

The results across all three tasks show consistent, logical patterns that align with collaborative filtering theory and the impact of data sparsity.

Task 1: Optimal K Selection

The optimal K values (80 for 25% sparsity, 70 for 75% sparsity) make sense because they balance noise reduction with maintaining similarity quality. Larger K values average more neighbors, reducing random variations. However, we saw diminishing returns beyond K=80 and even slight degradation with very large K values, showing that including too many dissimilar users hurts predictions.

The MAE increase from 0.7452 (25%) to 0.8164 (75%) is expected - with 75% of ratings missing, there's simply less training data to learn user preferences. The 9.5% error increase reflects this information loss.

Task 2: SVD vs KNN

SVD's superior performance under high sparsity (5.94% improvement at 75% vs only 0.66% at 25%) makes perfect sense. KNN relies on finding similar users with overlapping rated items, which becomes difficult when data is sparse. SVD learns latent factors that generalize across users, so it can still extract meaningful patterns even with limited overlap.

The optimal factor count dropping from 10 to 5 as sparsity increases also makes sense - fewer latent dimensions prevent overfitting when training data is scarce.

Task 3: Top-N Recommendations

The precision-recall tradeoff is clearly visible: as N increases, recall improves (we capture more relevant items) but precision drops (more non-relevant items sneak in). This is fundamental to ranking systems.

Best F1 scores occur at N=10 for 25% sparsity but shift to N=30 for 75% sparsity. This happens because with sparse data, users have fewer rated items overall, so we need a larger N to achieve good recall. SVD consistently matching or slightly beating KNN in F1 scores, especially at 75% sparsity (0.6465 vs 0.6313), confirms Task 2's findings.

The results are remarkably similar between KNN and SVD at 25% sparsity, showing both methods work well when sufficient data exists. The real difference emerges under sparsity, where SVD's model-based approach shines.