Collaborative Filtering: Memory-Based and Model-Based Approaches Report

1. Introduction

Basic Statistic examination before moving to the EDA step.

	userId	movieId	rating
count	100836.000000	100836.000000	100836.000000
mean	326.127564	19435.295718	3.501557
std	182.618491	35530.987199	1.042529
min	1.000000	1.000000	0.500000
25%	177.000000	1199.000000	3.000000
50%	325.000000	2991.000000	3.500000
75%	477.000000	8122.000000	4.000000
max	610.000000	193609.000000	5.000000

Interpretation:

The summary statistics show that the dataset contains 100,836 ratings from 610 users and approximately 9,700 movies. Movie IDs span a large non-consecutive range, which explains the high standard deviation in this column. The rating distribution is centered around a mean of 3.5, with quartiles at 3.0, 3.5, and 4.0, confirming that users tend to give positive ratings. This positive bias is common in recommender systems and influences the behavior of collaborative filtering models

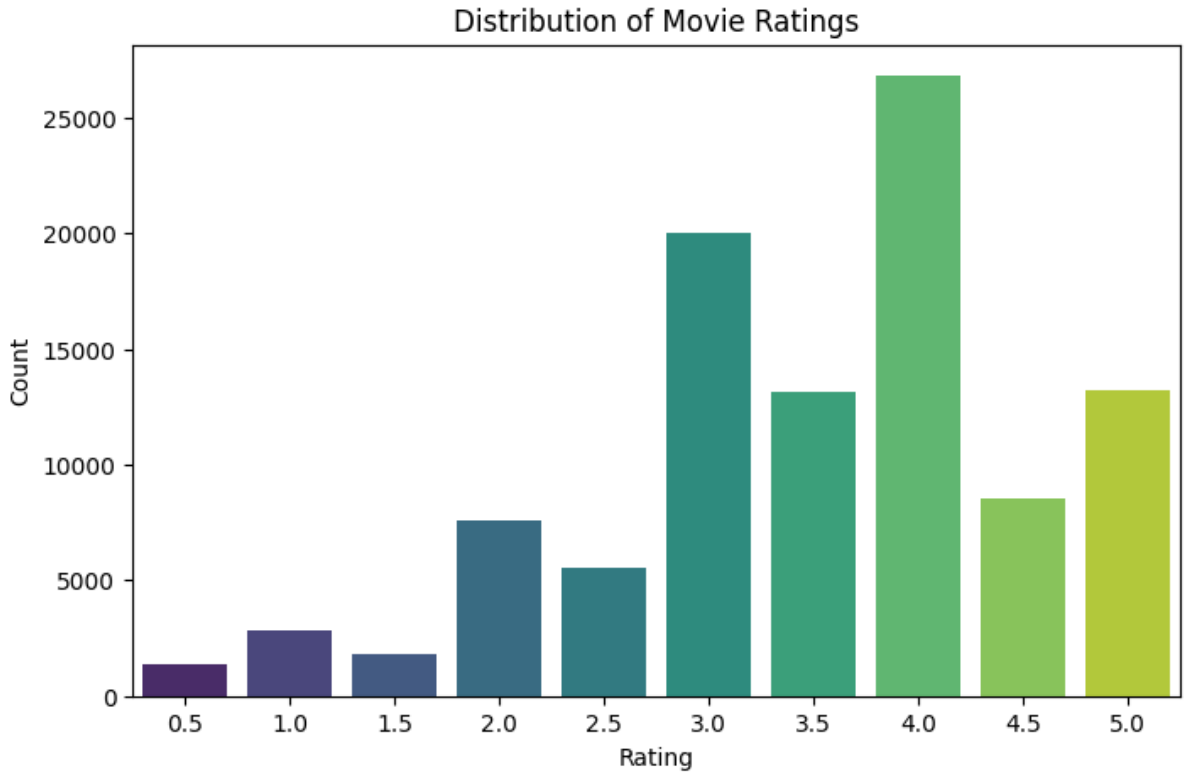
2. Exploratory Data Analysis (EDA)

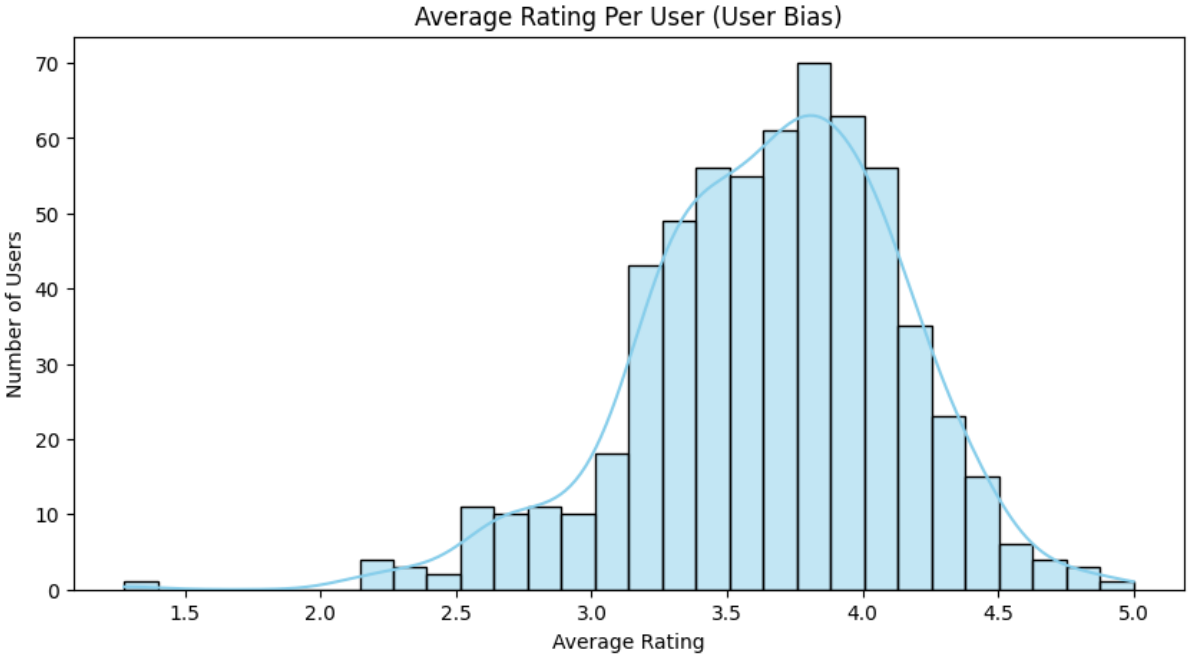
Before training models, we examine the dataset to understand rating patterns, user activity, item popularity, and sparsity.

2.1 Rating Distribution

A histogram of rating values shows that users tend to give positive ratings, mostly between 3.0 and 4.5 while very low ratings are less common. We see a “positive bias, users tend to rate highly if they are rate.

Both the Average Rating Per User plot and the Distribution of Movie Ratings plot highlight an important characteristic of the dataset: users tend to give positive ratings.





Why it's important:
It shows how users generally rate movies and whether the dataset is biased toward positive or negative ratings.

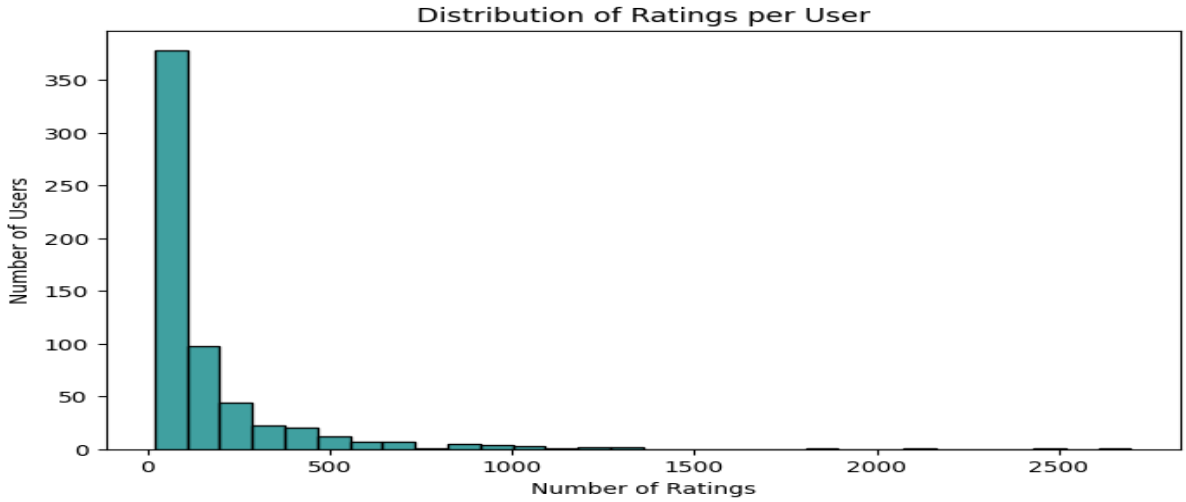
- User bias affects how collaborative filtering models interpret similarities as we seen also in the lecture.
- If most users consistently give higher ratings (3–4 stars), the dataset does not contain a balanced range of preferences.
- Models must account for this bias; otherwise, they may incorrectly assume users have similar tastes simply because they all rate high ratings.

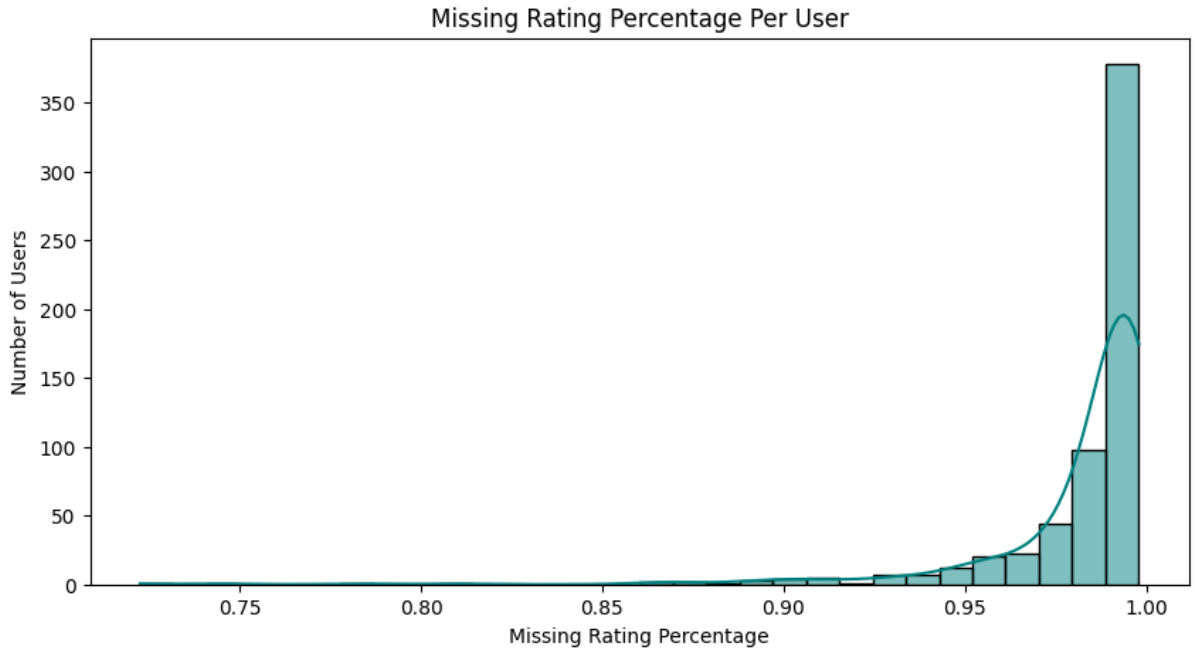
What we learn:
Most ratings are between 3 and 4, meaning users tend to rate movies positively. This affects baseline accuracy and model expectations.

Both the average rating per user plot and the overall rating distribution reveal a strong user bias in the dataset. Users tend to rate movies positively, with most ratings falling between 3.0 and 4.0. This consistent bias is important because collaborative filtering models must distinguish between genuine user preference and general generosity in scoring.

2.2 Number of Ratings per User

The number of ratings each user provides is highly uneven.
Many users rate fewer than 200 movies, while a smaller group of very active users contributes hundreds of ratings.
This makes a long tail pattern that affects user-based CF, as similarity becomes weaker for users with few ratings.



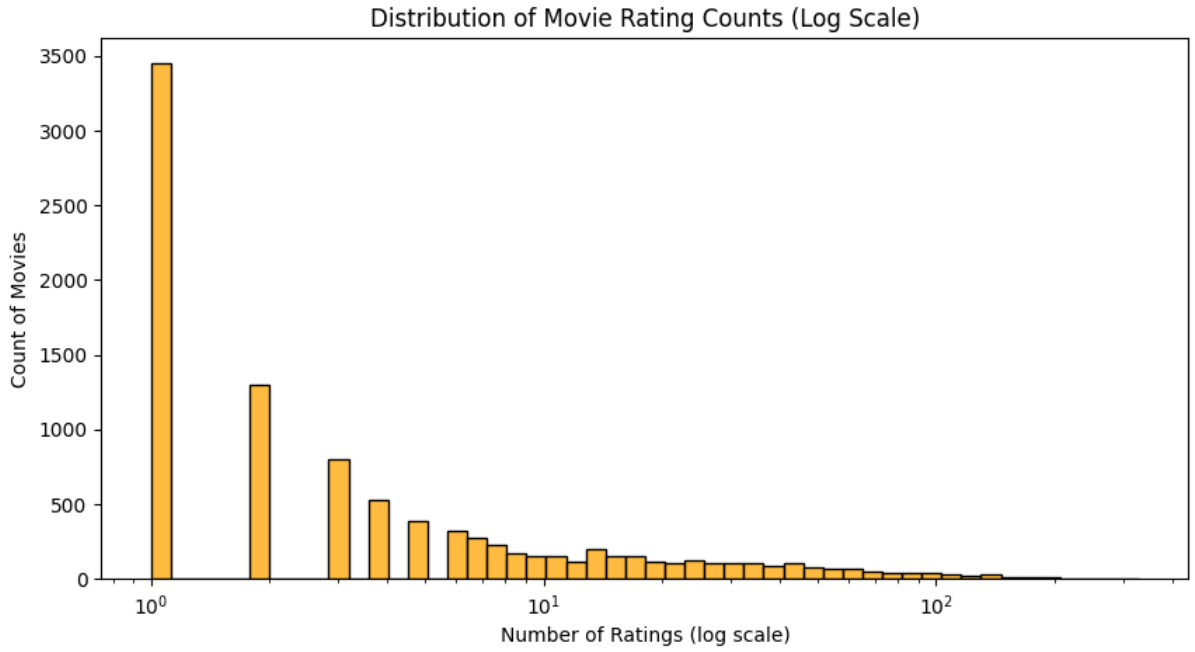


Why it's important:
This reveals how active users are and whether enough data exists to compute meaningful user–user similarity.

What we learn:
Most users rate relatively few movies, while a small group is very active. This imbalance explains why user-based CF works but can be noisy for inactive users.

2.3 Number of Ratings per Movie

Most movies receive only a small number of ratings, while a few very popular movies receive thousands.
This long tail item popularity distribution makes item-based CF more challenging for less rated movies.



Why it's important:
Movie popularity affects the quality of item–item similarity and directly impacts item-based CF performance.

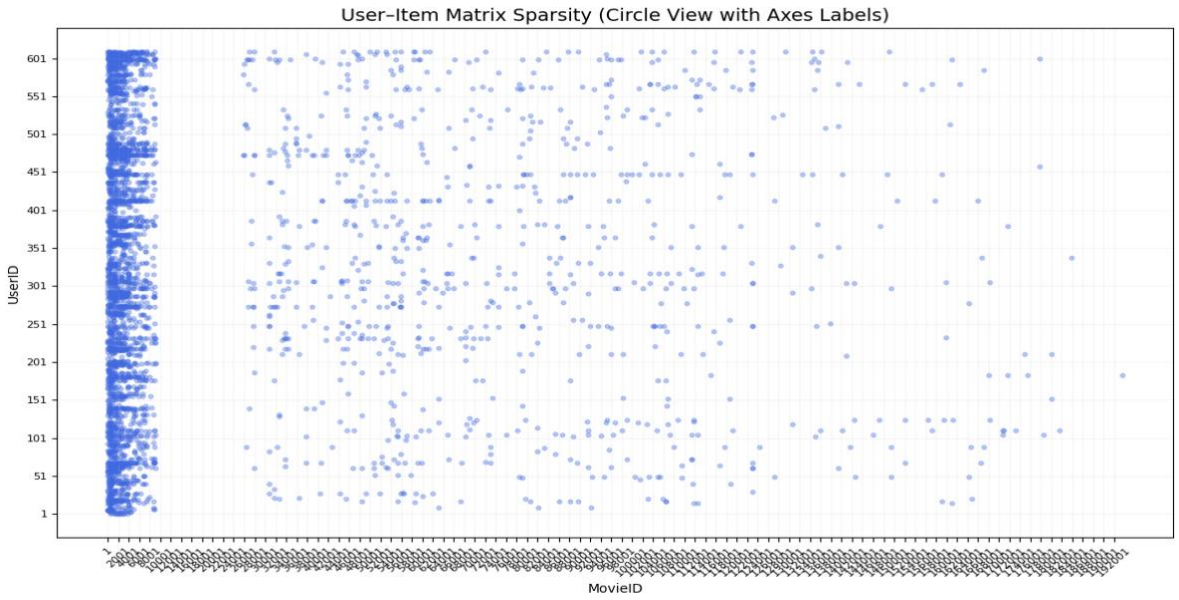
What we learn:
Most movies have very few ratings, which creates sparse item vectors, making item-based CF less reliable for movies with very low number of ratings.

2.4 User–Item Matrix Sparsity

A scatter visualization of the user–item matrix shows that only a tiny fraction of possible user–movie pairs contain ratings.
The dataset is extremely sparse, and this sparsity is a major challenge for memory based algorithms, which rely on overlaps between users or movies.

Why it's important:
It visually demonstrates how many user movie combinations are missing, which is the core difficulty of recommendation systems.

What we learn:
The matrix is extremely sparse only a tiny fraction of all possible ratings exist. This explains why latent factor models outperform memory-based models.



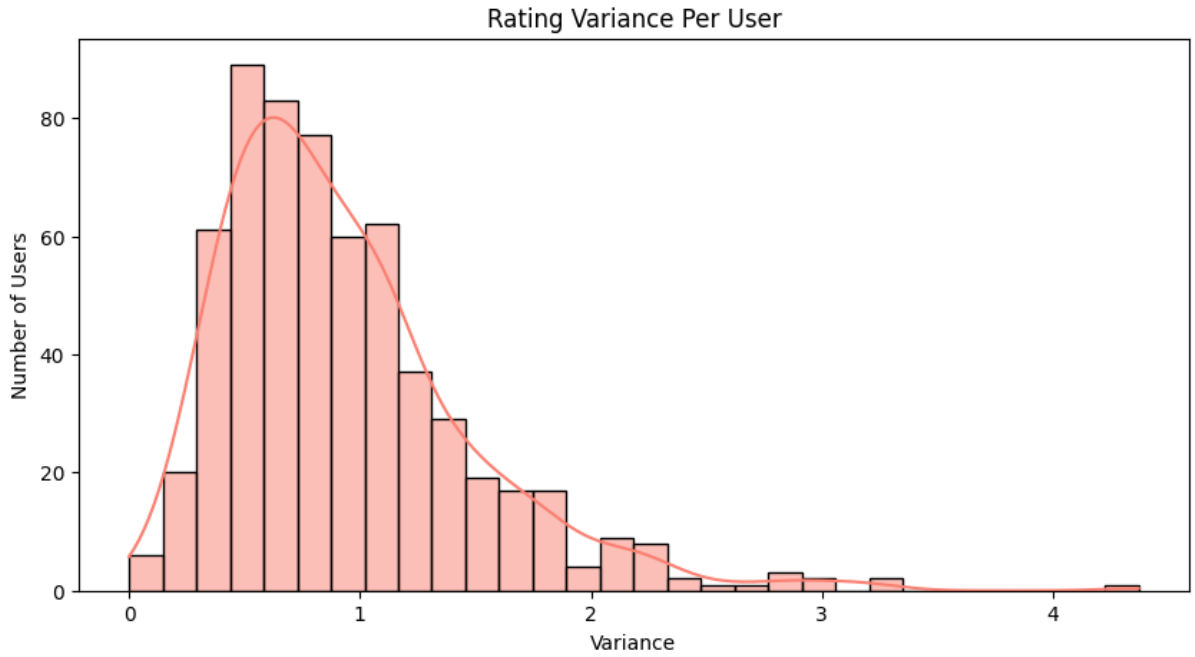
2.5 Rating Variance Per User

The plot shows the distribution of rating variance for each user, measuring how much each user's ratings fluctuate around their individual mean.

Why it's important:

Rating variance reveals how consistent users are in their scoring behavior. Users with low variance rate movies similarly regardless of what they watch, while users with high variance behave more unpredictably. This directly affects collaborative filtering models, because similarity-based methods assume consistent rating patterns.

What we learn:
Most users have variance between 0.5 and 1.5, which means their ratings differ moderately around their average preference. However, a smaller group of users shows very high variance, indicating inconsistent or extreme rating behavior. These high-variance users are harder to model user-based CF may incorrectly match them with others, and prediction errors tend to be higher. This reinforces the need for model-based methods**,** which better capture user specific behavior.

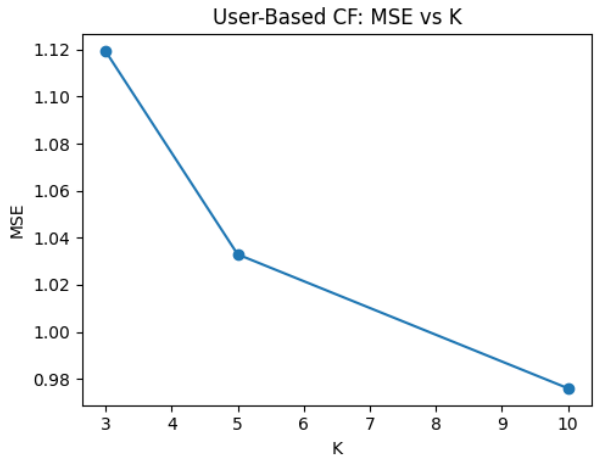


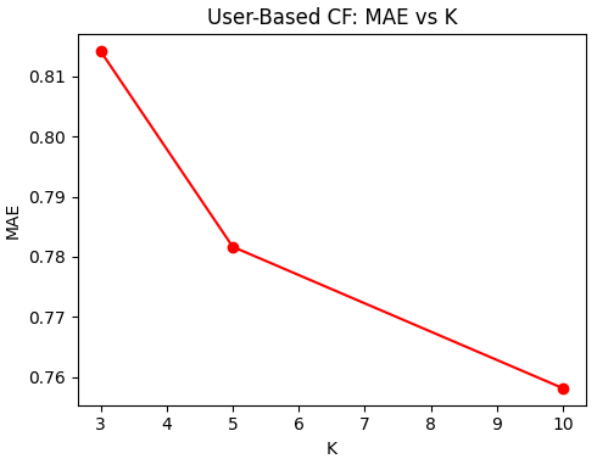
3. Memory-Based Collaborative Filtering

Memory-based CF uses direct similarity between users or items to estimate unknown ratings. We evaluate both User-Based KNN and Item-Based KNN using cosine similarity and neighborhood sizes of K = 3, 5, 10.
Each setting is evaluated using **5-fold cross validation**.

3.1 User-Based Collaborative Filtering

K	***MSE***	***MAE***
3	~1.12	~0.818
5	~1.04	~0.782
10	~0.97	~0.759

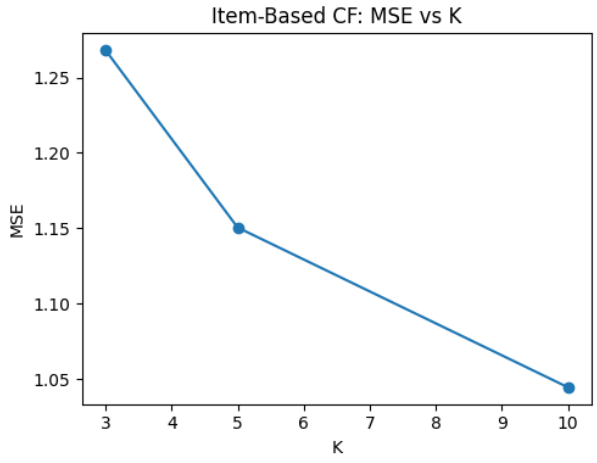


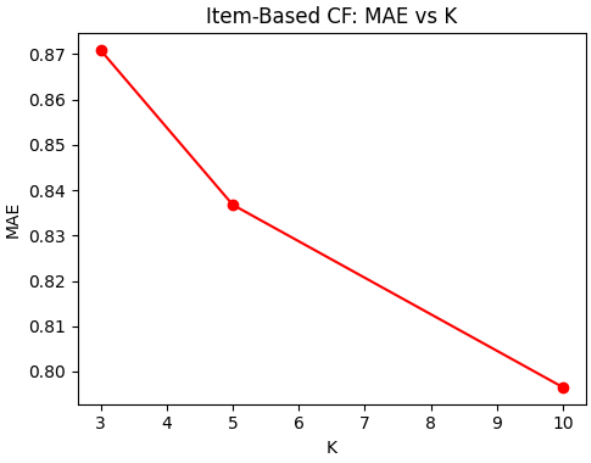


Interpretation:
As K increases, both MSE and MAE decrease, meaning predictions become more stable and less noisy.
User-based CF performs relatively well in this dataset because many users have overlapping ratings.

3.2 Item-Based Collaborative Filtering

K	MSE	MAE
3	~1.256	~0.865
5	~1.157	~0.837
10	~1.042	~0.798





Interpretation:
Error decreases with larger K, but item-based CF performs worse than user-based.
This is expected, since many movies in the dataset have very few ratings, making item-item similarity harder to compute reliably.

3.3 Summary of Memory-Based Methods

- Increasing K improves prediction accuracy.
- User-Based CF consistently outperforms Item-Based CF.
- Data sparsity and long tail item distribution reduce the effectiveness of item-based similarity.

4. Model-Based Collaborative Filtering

Model-based CF uses latent factor models to learn hidden representations of users and items.
We evaluate three widely-used algorithms: SVD, SVD++, and NMF.

4.1 Results

Model	MSE	MAE
SVD	0.7617	0.6703
SVD++	**0.7443**	**0.6614**
NMF	0.8499	0.7061

4.2 Interpretation

- SVD++ performs best overall.
  It incorporates both explicit ratings and implicit feedback (which movies a user interacted with), resulting in more accurate predictions.
- SVD performs slightly worse but still much better than memory-based methods.
- NMF produces the highest error because it is more restricted, lacking user/item bias modeling and using only positive latent components.

5. Final Comparison and Conclusions

5.1 Memory-Based vs Model-Based

Model-based methods clearly outperform memory-based CF.
This is because:

- Memory-based CF suffers from data sparsity.
- Latent factor models can generalize beyond observed ratings.
- SVD-based methods capture underlying patterns that similarity-based methods cannot.

5.2 Best Overall Algorithm

SVD++ is the best model in the entire project, achieving the lowest MSE and MAE.
Its ability to combine explicit and implicit feedback gives it a significant advantage.

5.3 Key Insights

- The MovieLens dataset is extremely sparse, affecting similarity-based methods.
- Latent factor models produce more accurate recommendations.
- Memory-based models are simple but limited, while model-based methods offer more flexible and powerful representations.