Recommender System Analysis Report: MovieLens Data

1. Exploratory Data Analysis (EDA)

The dataset used is the MovieLens "Small" dataset, comprising 100,836 ratings applied to 9,724 movies by 610 users.

1.1 Data Overview and Sparsity

The initial inspection revealed that the dataset is clean with no missing values in the core columns (userId, movieId, rating). A critical characteristic of recommender system datasets is **sparsity**. The calculated sparsity is **98.30%**. This indicates that out of all possible user-movie combinations, only 1.7% actually contain a rating. This high sparsity is typical in recommender systems.

1.2 Distribution of Ratings - see *Figure 1*

The distribution of ratings (ranging from 0.5 to 5.0) shows a distinct **negative skew (left-skewed)**. We can see that the users are generally positive; ratings of 3.0, 4.0, and 5.0 are much more frequent than lower ratings. And this suggests a "positivity bias," where users tend to rate movies they like or choose to watch movies they expect to enjoy.

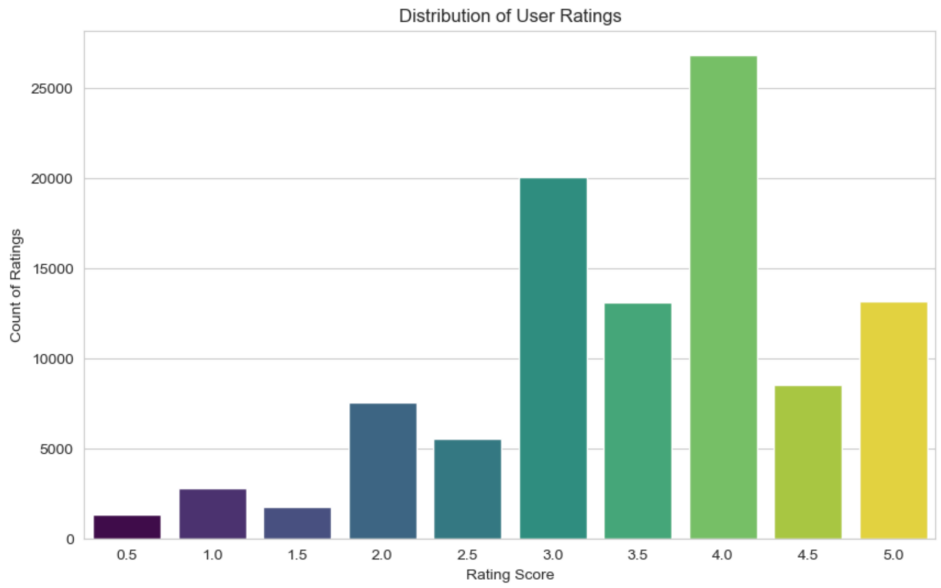

Figure 1 – Distribution of User Ratings

1.3 The Long Tail (Item Popularity)

The analysis of ratings per movie (see *Figure 2*) confirms what we expected, the **"Long Tail" phenomenon**, we can see a very small percentage of movies receive the vast majority of

ratings. And the long tail of the distribution, containing thousands of movies with only 1 or 2 ratings. This makes recommending items from the "tail" difficult due to a lack of data points.

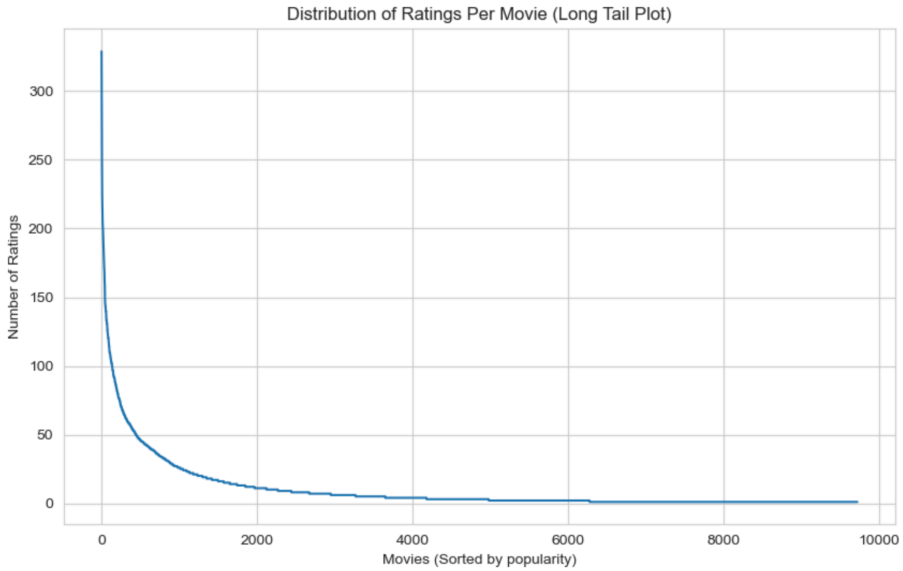

Figure 2 - Distribution of Ratings Per Movie

1.4 User Activity

As we can see in Figure 3, the user engagement varies significantly, average user provided approximately **165 ratings**, the distribution is right-skewed; while there are "power users" who rate hundreds of movies, the majority of users rate fewer.

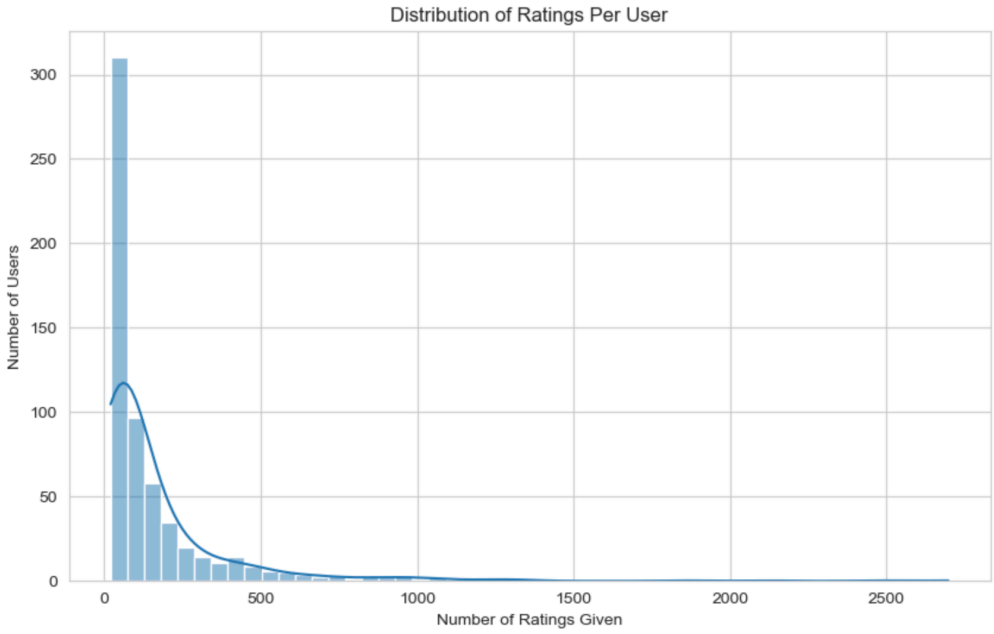

Figure 3- Distribution of Ratings Per User

1.5 Temporal Trends - Ratings Over Time

As shown in the lectures, how temporal analysis affected the understanding of Netflex data, we performed temporal analysis from different aspects.

In Figure 4 we can see that the number of ratings collected per year fluctuates significantly rather than showing steady growth. Notable peaks occurred around **2000** and in **2017**. We also analyzed user activity by month (see Figure 5) in order to inspect seasonality. We can see two peacs in **May** and **November**. These periods may correlate with major holiday seasons (start of summer, pre-Christmas) or academic breaks when users have more leisure time to watch and rate movies.

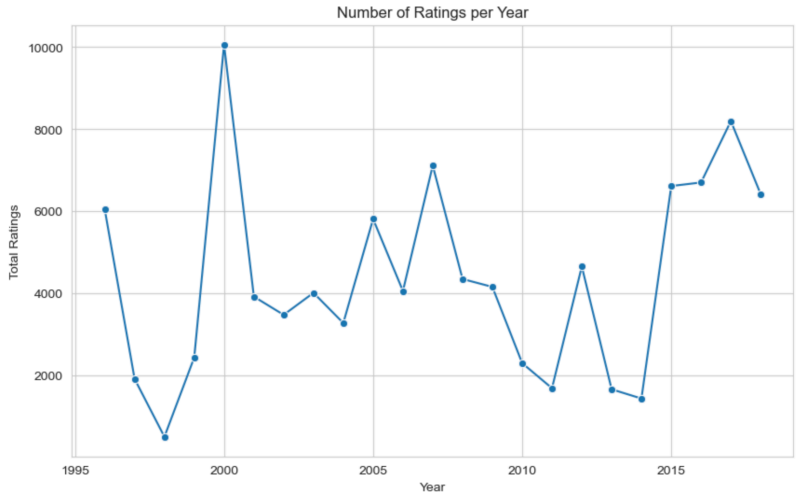

Figure 4 - Number of Ratings per Year

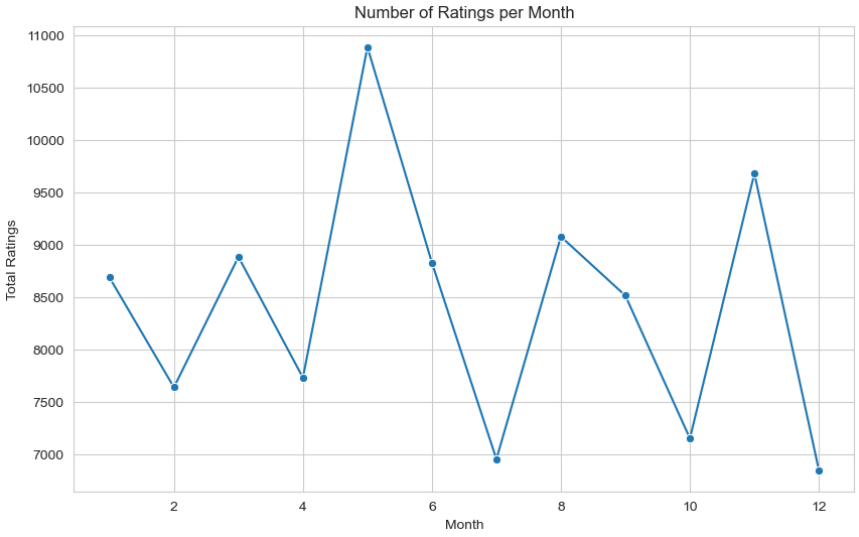

Figure 5 - Number of Ratings per Month

We also were interested to see how many unique users we have per year (see Figure 6), we can see the drop of users from the platform whitch suggests the early dropping problem.

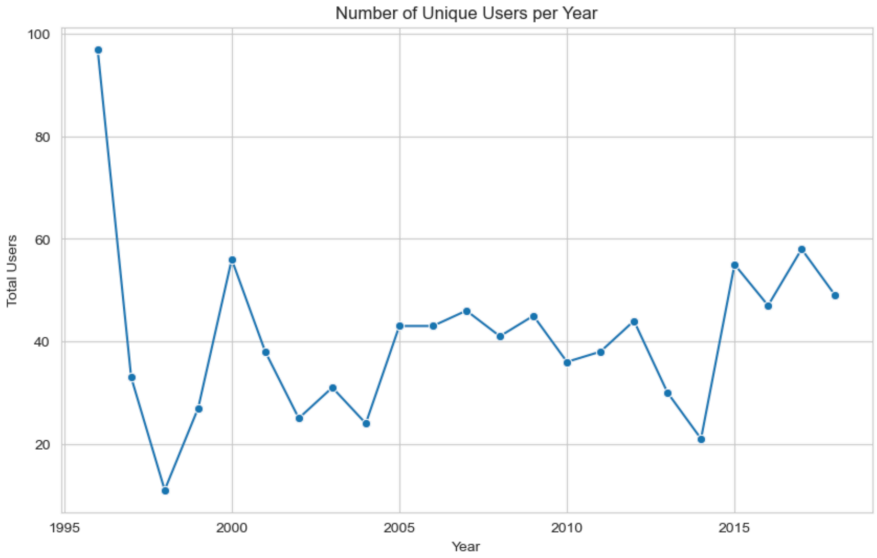

Figure 6 - Number of Unique Users per Year

In Figure 7, we investigated the relationship between a movie's popularity (number of ratings) and its perceived quality (average rating) to check for bias. We can see that popular movies generally have higher ratings. Movies with a high volume of ratings (the far right of the x-axis) consistently maintain an average rating above **3.0**, with many clustering between 4.0 and 5.0.

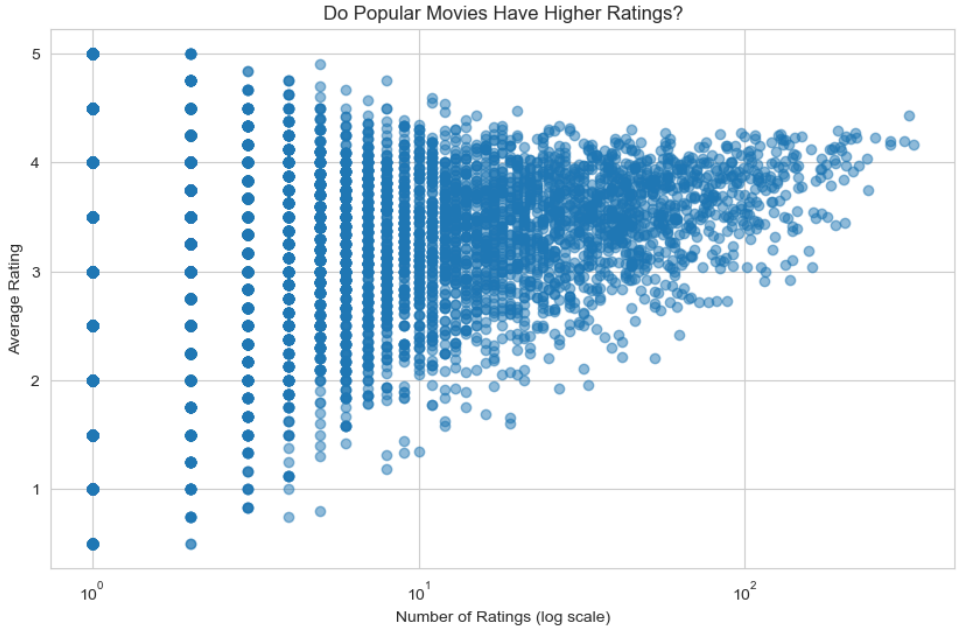

Figure 7 - Average Rating per Movie as a function of Number of Ratings

2. Memory-Based Collaborative Filtering

This section evaluated K-Nearest Neighbors (KNN) algorithms using **Cosine Similarity**. We compared **User-Based** (User-User) vs. **Item-Based** (Item-Item) filtering across different neighborhood sizes (k=[3, 5, 10]).

2.1 Methodology

- **Metric:** Mean Squared Error (MSE) and Mean Absolute Error (MAE).
- **Validation:** 5-Fold Cross-Validation.

2.2 Results Summary

K-Neighbors (k)	Approach	MSE (Error)	Result Interpretation
3	Item-Based	1.2589	Highest Error
5	Item-Based	1.1413	Improved compared with k=3
10	Item-Based	1.0426	Best for Item-Based
3	User-Based	1.1247	
5	User-Based	1.0392	
10	**User-Based**	**0.9758**	**Lowest Error (Best for both approaches)**

2.3 Key Observations

We were expecting the item based approach to perform better than the user based approach, as we saw in the lectures, yet the figures (Figure 8 and Figure 9) show that both evaluations (MSE and MAE) have better results with the user based model.

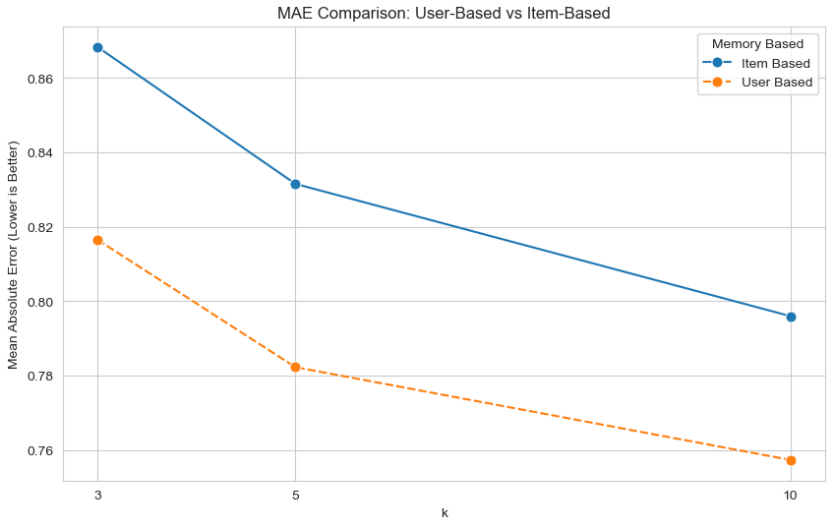

Figure 8- MAE Comparison

This result is likely due to the specific characteristics of the MovieLens Small dataset:

The dataset contains only 610 users compared to roughly 9,724 movies. This means the 'User Space' is significantly smaller and denser than the 'Item Space.' It is statistically easier for the algorithm to find reliable neighbors among a small group of users than to find similar items within a vast, sparse catalog of movies. Also, from the "Distribution of Ratings per Movies" graph in the previous section, we saw that low number of movies were rated, obviously the popular ones, this will positivly affect the model of user based, because most users rated the same popular movies, this makes it easier to ensure high overlap for the Cosine Similarity calculation.

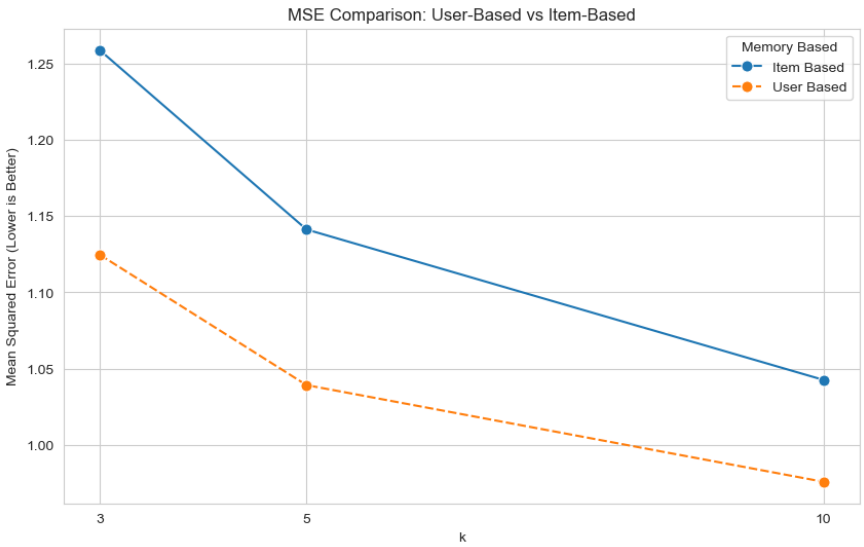

Figure 9 - MSE Comparison

For the same reason of rating popular movies, the Item-Based model likely suffered from the 'Long Tail' problem, where thousands of movies have very few ratings, making their similarity scores unreliable.

We also observed that increasing the neighborhood size (K) from 3 to 10 reduced error rates. This suggests that relying on a very small number of neighbors (K=3) makes the model sensitive to outliers, whereas (K=10) provides a more robust average. Yet, as we saw in the lectures increasing this number too much will not really help and eventualy we will reach a plato.

3. Model-Based Collaborative Filtering

This section evaluated Latent Factor Models using Matrix Factorization. We compared **SVD**, **SVD++**, and **NMF**.

3.1 Experimental Phases

1. **Default Parameters:** Running models with library defaults.
2. **Standardized Dimension:** Running all models with a latent dimension (n_factors) of 5 to ensure a fair structural comparison.

3.2 Results Summary

Default Parameters

Model	MSE	MAE	Rank
SVD++	**0.7393**	**0.6590**	**1st (Best)**
SVD	0.7626	0.6714	2nd
NMF	0.8496	0.7060	3rd

Standardized (k=5)

Model	MSE	MAE	Rank
SVD++	**0.7412**	**0.6598**	**1st (Best)**
SVD	0.7561	0.6684	2nd

Model	MSE	MAE	Rank
NMF	0.7625	0.6689	3rd

3.3 Experiaments and Algorithm Analysis

In this section, we evaluated three Matrix Factorization algorithms: SVD, SVD++, and NMF. We conducted the experiment in two phases: first using the library's default hyperparameters, and second by standardizing the latent dimension (k) to ensure a fair comparison.

1. Hyperparameter Tuning

Default Dimensions: The default n_factors (k) in the Surprise library vary significantly between models (SVD: k=100, SVD++: k=20, NMF: k=15).

Standardized Dimensions: To compare the algorithms fairly, we tested setting n_factors to both 5 and 50 for all models. We observed that k=5 yielded better results (likely due to the small size of the dataset avoiding overfitting), so we used k=5 for the final comparison.

Biased NMF: In the second phase, we explicitly set biased=True for the NMF model (default is False). This allowed NMF to learn user and item biases, ensuring that the primary structural difference remaining between it and SVD was the non-negative constraint.

Regularization: We attempted to standardize the regularization for NMF to 0.02 (to match SVD/SVD++), but this degraded performance. We decided to keep NMF at its default regularization of 0.06.

2. Results & Interpretation

The final results show that In both experaments, SVD++ performed best, followed by SVD, with NMF performing the worst. SVD++ achieved the lowest error because it incorporates implicit feedback. It accounts not just for the explicit rating value, but also for the fact that a user chose to rate an item at all. This additional layer of information is crucial for sparse datasets like ours. The SVD performed well because it uses standard Matrix Factorization, allowing latent factors to be negative. This flexibility allows the model to capture complex user preferences. Additionally, SVD includes bias terms (user and item biases), which helps separate user leniency from actual preference. The NMF had the highest error. This is expected for two reasons: first the Non-Negative Constraint, by forcing all factors to be positive (>= 0), the model loses the ability to represent "dislike" or negative correlations effectively. Second, the lack of bias. The default NMF implementation is unbiased, meaning it does not account for the fact that some users inherently rate higher or some movies are inherently more popular.

4. Final Conclusion

For the MovieLens Small dataset, **SVD++** is the recommended algorithm.
While Memory-Based approaches provided insight into the dataset's density (favoring User-Based over Item-Based), the Model-Based approaches significantly reduced the error rates (User-Based KNN MSE ~0.97 vs. SVD++ MSE ~0.74). The ability of SVD++ to leverage implicit feedback makes it the most robust choice for this specific data distribution.