**Comparative Analysis of Clustering algorithms for Movies, Ratings and Tags dataset**

An exploratory study using EDA, Statistical Analysis, and Machine Learning

**Author:**

Selvi Ganesh Kumar

**Institution:**

CCT College

Dublin 2

Ireland

**Course:**

Higher Diploma in Science in Data Analytics

**Supervisor:**

Mr. Sam Weiss

**Date of Submission:**

18th of May 2025

**Version:**

Final Report

**Contact:**  
Email: [selvi.digitalsolutions@gmail.com]  
GitHub: github.com/research-selvi-datascience

*© 2025 Selvi Ganesh Kumar. All rights reserved.*

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| 1 | Analysis Objectives | 3 |
| 2 | Exploratory Data analysis and Data Pre-processing: | 3 |
| 3 | Clustering Experiments | 3 |
| 3.1 | Experiment 1: Baseline Clustering | 3 |
| 3.2 | Experiment 2: Baseline Models – Scaling and Normalising (L1 Normalisation) | 4 |
| 3.3 | Experiment 3: Movie-Based Segmentation Using Clustering | 4 |
| 3.4 | Experiment 4: User-Based Segmentation | 5 |
| 3.5 | Experiment 5: Temporal Analysis of Ratings and Tags (2013–2015) | 6 |
| 4 | Conclusion - Clustering | 7 |
| 5 | Recommendation Systems: | 8 |
| 5.1 | Experiment 1: Baseline Recommendation System on Full Dataset | 9 |
| 5.2 | Experiment 2: Train-Test Split Across All Models | 9 |
| 5.3 | Experiment 3: Varying User and Item Engagement Levels | 9 |
| 6 | Conclusion – Recommendation systems | 10 |
| 7 | References | 11 |

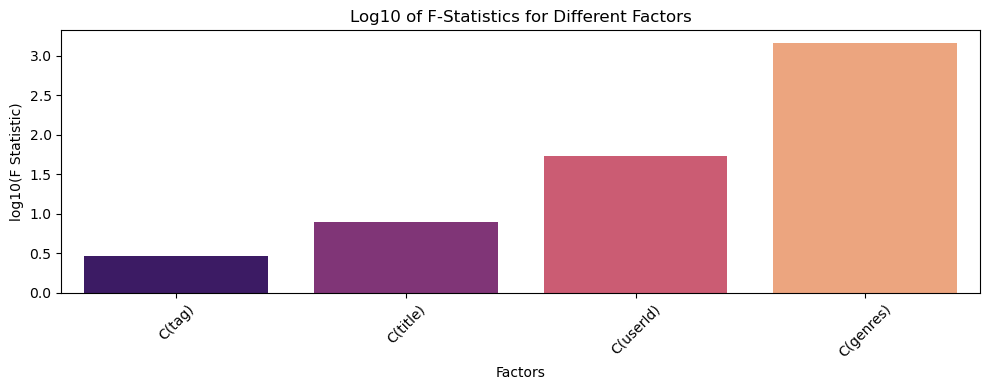
**Analysis Objectives**

The goal of this analysis is to apply and evaluate clustering techniques on a movie dataset to identify natural groupings of movies based on content and metadata such as genres and tags. By uncovering these clusters, the analysis aims to generate actionable insights for business applications like content acquisition, personalized recommendation strategies, and market segmentation.

**Exploratory Data analysis and Data Pre-processing:**

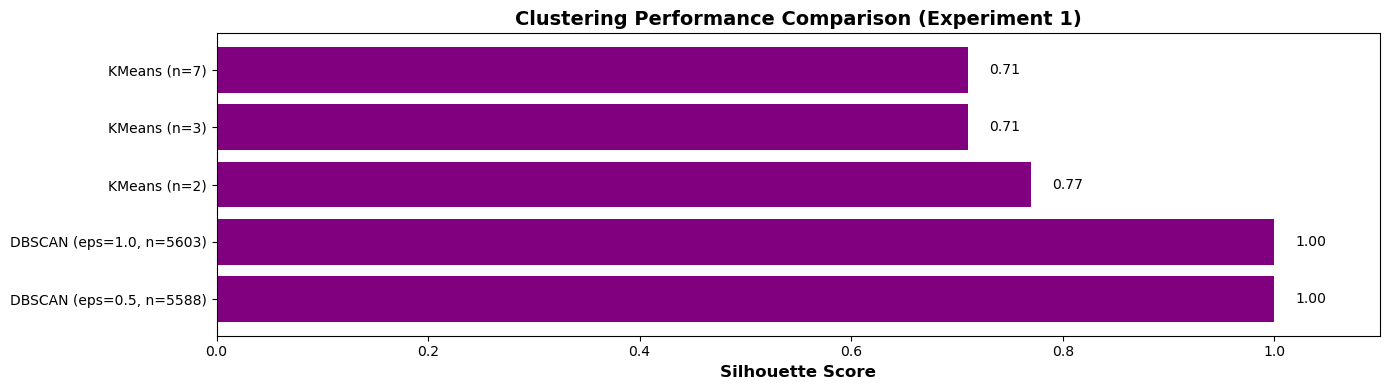
Exploratory analysis showed that a few movies (e.g., Pulp Fiction, Fight Club) dominate the rating counts, suggesting a skew in user attention and potential bias in models. Most ratings fall between 3 and 5, with peaks at 4 and 5, indicating a positivity bias. Ratings distribution is left-skewed (skew = -0.91), with a flatter peak (kurtosis ≈ 0.49), showing fewer extreme values. Anomaly detection via Isolation Forest flagged ratings below 1.5 as outliers, aligning with earlier findings. Shapiro-Wilk tests confirmed the ratings are not normally distributed.

To prepare for clustering, the genres column in the movie dataset was one-hot encoded using MultiLabelBinarizer, converting each genre into a binary feature. Tags were incorporated by combining them with genres into a single text field, then vectorized using TF-IDF, allowing content-based clustering based on user-labeled descriptors.

Together, these preprocessing steps and analyses support robust clustering and provide insight into user preferences, content categorization, and engagement behavior for recommendation, segmentation, and content targeting. ANOVA tests confirmed genre and userId as key drivers of rating variance, supporting their inclusion in clustering.

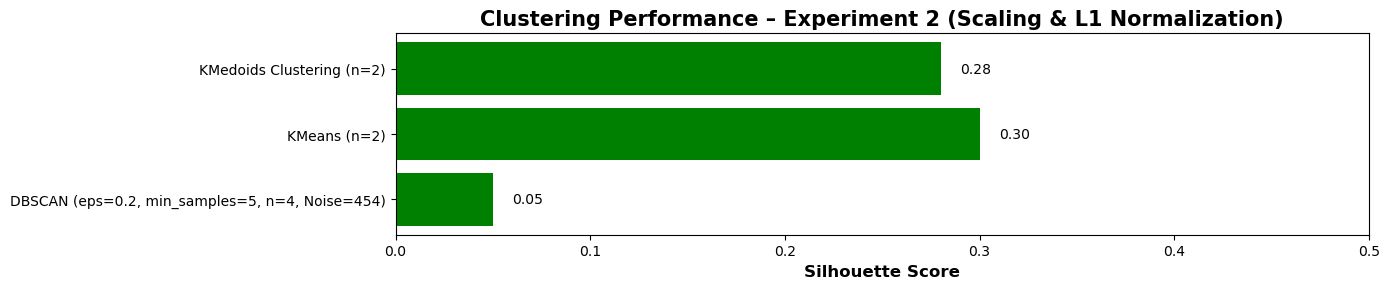
**Experiment 1: Baseline Clustering**

The baseline clustering experiment established a reference point by applying clustering algorithms on raw, unscaled data using numerical features such as genre indicators (one-hot encoded), Box-Cox transformed ratings, and user and movie aggregated statistics (mean and count of ratings). No feature scaling was applied to assess how well algorithms could uncover structure in the unprocessed dataset.



KMeans performed reasonably well, especially with 2, 3, and 7 clusters, achieving silhouette scores of 0.77 and 0.71. This indicated the presence of meaningful patterns even without normalization. DBSCAN misleadingly returned perfect silhouette scores of 1.00, a result of over-fragmentation in high-dimensional, sparse data. Thousands of micro-clusters and noise points rendered the output unusable. KMeans can uncover latent structure in unscaled data. DBSCAN, however, requires careful preprocessing (e.g., scaling and tuning with k-distance plots) to function correctly.

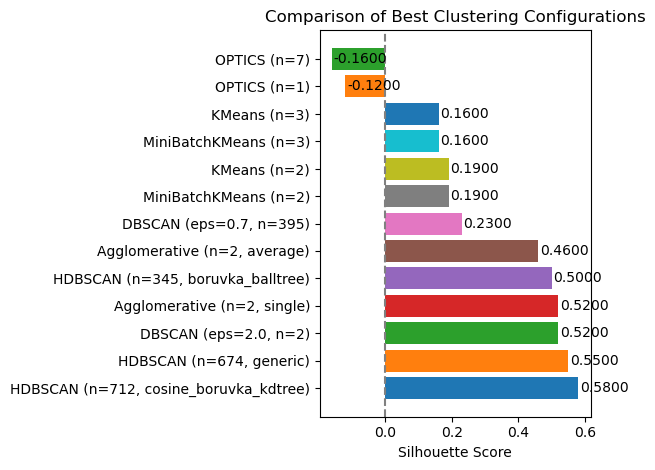
**Experiment 2: Baseline Models – Scaling and Normalising (L1 Normalisation)**This experiment assessed the impact of scaling and L1 normalization, particularly useful for sparse data from one-hot encoded genres and TF-IDF tag features. Due to memory constraints, a representative sample of 5,000 records was analyzed. Sampling validity was verified using KDE plots, PCA, t-tests, KS tests and Independent T Tests.



DBSCAN (after normalization) improved dramatically from prior over-fragmentation, forming 4 clusters with 453 noise points. However, the silhouette score remained low (0.09), suggesting weak clustering despite structural improvements. KMeans (n=2) achieved a silhouette score of 0.30, showing moderate clustering performance post-normalization. KMedoids (k=2) outperformed both with the best score, indicating clearer cluster separation. However, increasing k led to reduced cohesion. Scaling and normalization improve DBSCAN's usability, while KMeans and KMedoids offer solid baseline performance. More tuning and experimentation with distance metrics may further enhance clustering quality.

**Experiment 3: Movie-Based Segmentation Using Clustering**

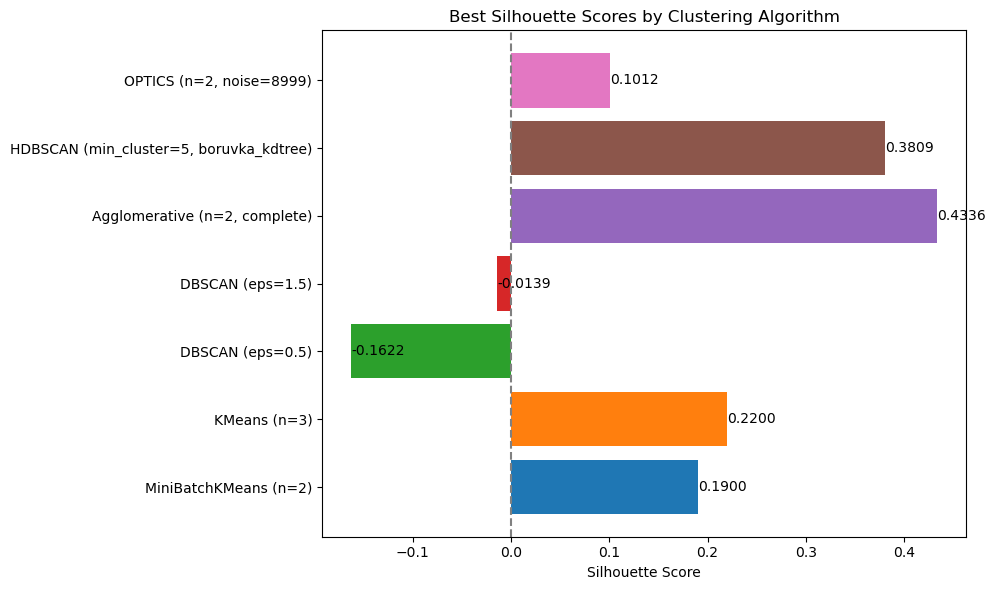
This experiment aimed to cluster movies based on features like One-Hot Encoded genres, mean ratings, rating counts, and Box-Cox transformed ratings. Clustering algorithms used included KMeans, MiniBatchKMeans, Agglomerative Clustering, DBSCAN, HDBSCAN, and OPTICS. PCA and t-SNE were employed for dimensionality reduction and visualization.



While some methods—particularly HDBSCAN and DBSCAN with tuned eps—achieved decent Silhouette Scores, the clustering structures were often suboptimal. Agglomerative Clustering (n=2) with average and single linkage scored 0.46 and 0.52, but formed trivial clusters—mostly one large group with few outliers. Many results suffered from over-segmentation (hundreds of small clusters), trivial partitioning (e.g., one large cluster and one small), or algorithm-data mismatches (e.g., OPTICS performed poorly on sparse data). Overall, most clustering approaches failed to produce interpretable or practically useful groupings.

**Experiment 4: User-Based Segmentation**

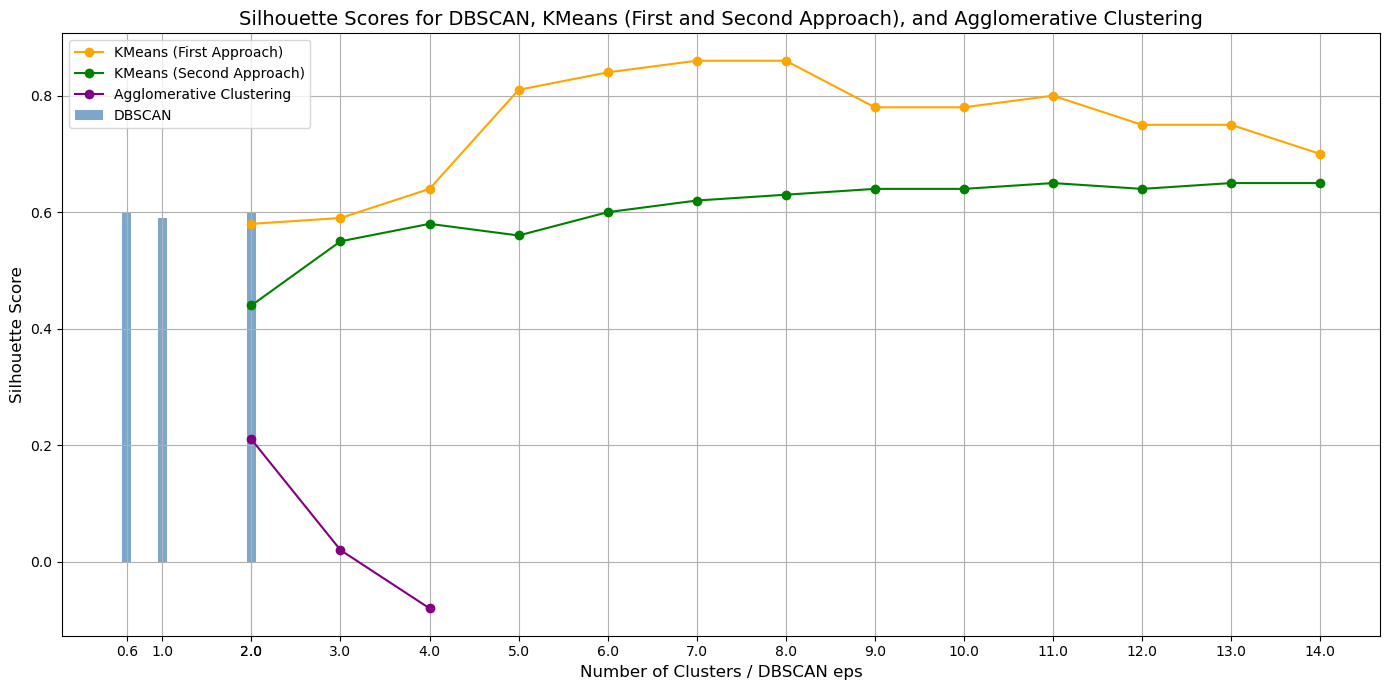
The objective was to segment users based on their rating behavior, using average ratings and count of ratings per genre. This helps understand user preferences and supports personalized recommendations. Due to memory limitations, clustering on the full dataset caused system crashes. A sampling method was adopted to reduce computational load, and validation tests confirmed the sample's statistical representativeness.



Clustering was performed on this sample using several algorithms. Agglomerative Clustering yielded the highest Silhouette Score (0.4336), indicating the best separation. HDBSCAN also performed well (Silhouette Score = 0.3809), offering an effective alternative. KMeans and MiniBatchKMeans had moderate performance. In contrast, DBSCAN and OPTICS underperformed with low or negative scores, making them less suitable for this task.

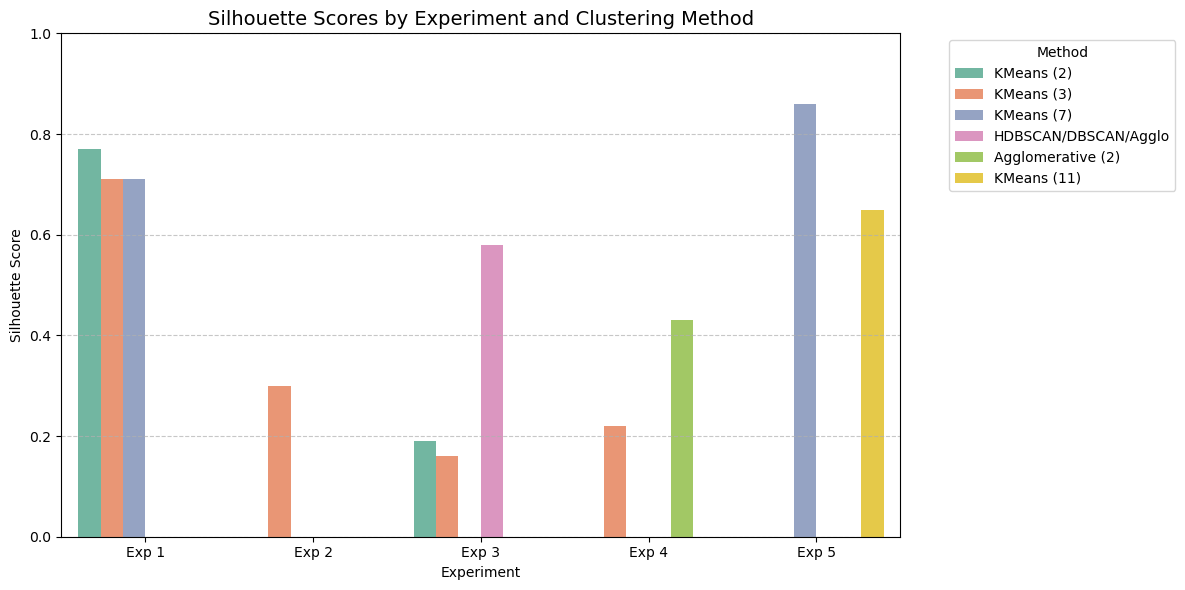
**Experiment 5: Temporal Analysis of Ratings and Tags (2013–2015)**This experiment analyzed user behavior over time using temporal features from rating and tag timestamps. Features included year, month, day, hour, weekend indicator, and recency. Tags were vectorized using TF-IDF (500 features), and two clustering approaches were tested: (1) using only numerical features, and (2) combining numerical features with one-hot encoded genres and TF-IDF tag features. Dimensionality reduction via PCA and t-SNE supported visualization and stability.

KMeans Clustering performed best. The first approach (numerical only) peaked with a Silhouette Score of 0.86 at 7 clusters, ideal for fast, interpretable results. However, the second approach offered better cluster diversity and semantic richness (inclusion of key features), achieving a consistent score of 0.65 with 11 clusters—making it more meaningful for applications like recommendation systems.

In comparison with DBSCAN and Agglomerative, KMeans with numerical, genre, and tag features proved most effective, balancing clustering quality with interpretability and feature richness. It enables more actionable insights despite a slightly lower Silhouette Score compared to simpler models.

**Conclusion**

Across five clustering experiments, KMeans consistently demonstrated reliable performance compared to density-based and hierarchical methods, particularly in sparse data conditions.



KMeans consistently outperformed density-based and hierarchical methods, especially under sparse data conditions. Its top performance came in Experiment 5 with an 86% silhouette score using numerical features (7 clusters), and 65% when incorporating genres and tags (11 clusters). Experiment 1 also showed strong results with scores of 77% (2 clusters) and 71% (3 and 7 clusters). In contrast, DBSCAN, HDBSCAN, and Agglomerative methods struggled with over-segmentation and imbalance due to data sparsity. While KMeans emerged as the most reliable, the experiments highlight the limitations of basic clustering approaches and point to the need for more advanced, scalable methods in future work to enhance cluster quality and interpretability.

**Recommendation Systems:**

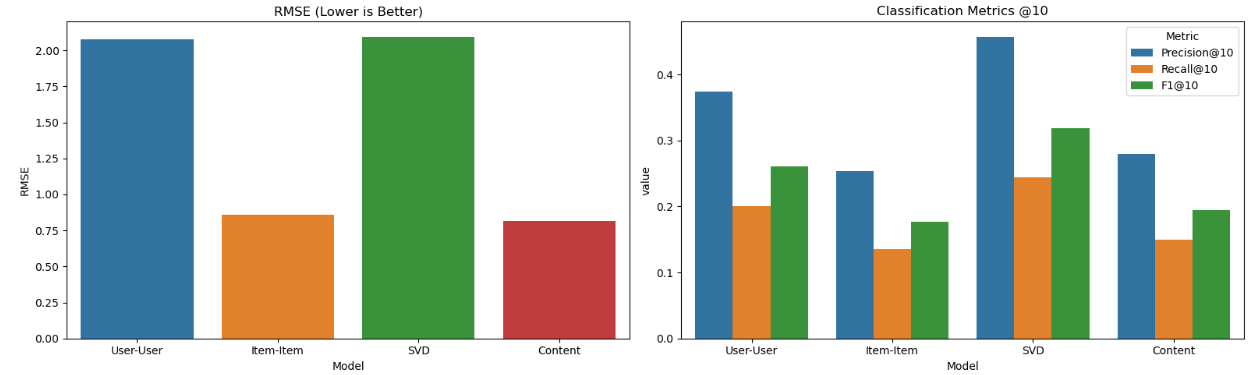
This analysis explores three experiments designed to evaluate and compare multiple recommendation strategies on a movie dataset using collaborative filtering, matrix factorization, and content-based methods.

**Experiment 1: Baseline Recommendation System on Full Dataset**

The first experiment implemented four foundational models: User-User Collaborative Filtering, Item-Item Collaborative Filtering, Singular Value Decomposition (SVD), and Content-Based Filtering. The focus was on understanding basic algorithm behavior without train-test separation. Only highly active users (more than 200 ratings) were included to ensure rich interaction histories. For collaborative models, adjusted ratings (actual rating minus user mean) were used to account for user biases. Cosine similarity was applied for user and item similarities, and SVD was implemented using matrix factorization.

RMSE was used for evaluation: User-User (0.39), Item-Item (0.37), and SVD (0.30), suggesting good accuracy. However, these scores were overly optimistic due to evaluation on the same dataset used for training—leading to data leakage. Content-Based Filtering used TF-IDF on tags and genres, and was evaluated with an 80/20 train-test split using ranking metrics: Precision@10 (0.165), Recall@10 (0.070), and NDCG@10 (0.476). While useful for understanding base performance, this experiment highlighted the need for proper validation protocols to obtain reliable performance metrics.

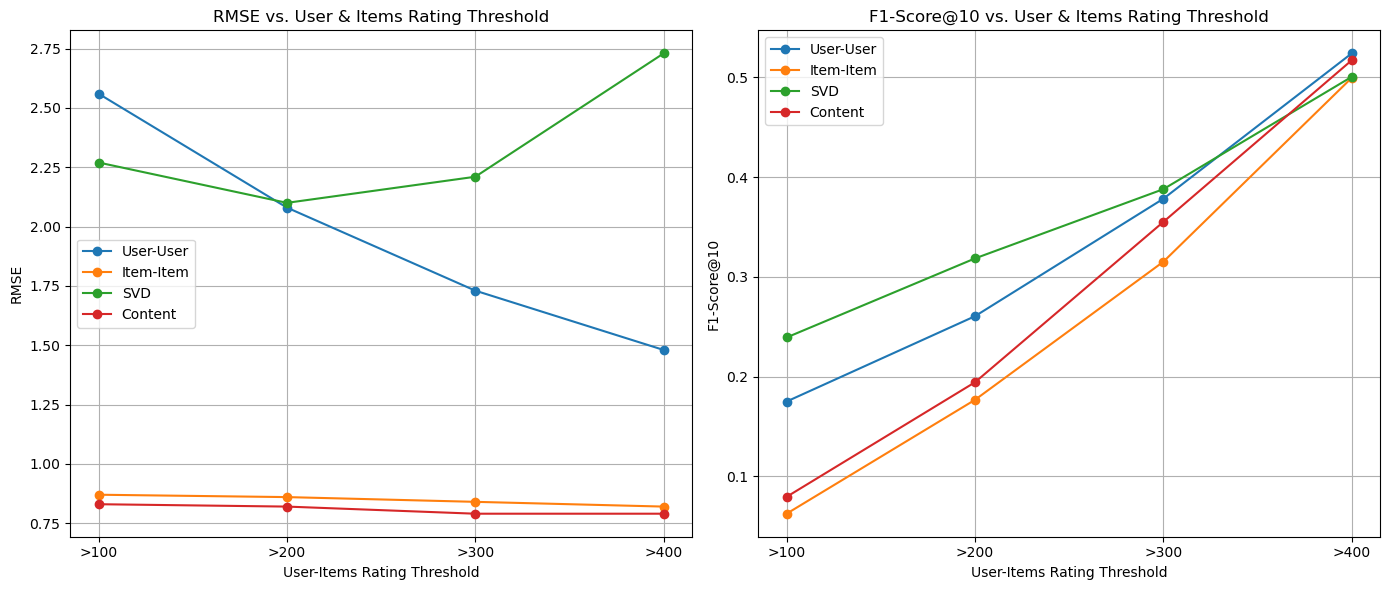
**Experiment 2: Train-Test Split (80/20) Across All Models**



To address the limitations of Experiment 1, Experiment 2 applied an 80/20 train-test split across all four models. Evaluation was based on RMSE for rating prediction, and Precision, Recall, and F1-score at top-10 recommendations to assess ranking performance. This shift allowed for more realistic and generalizable performance measurement.

All models showed a performance drop compared to Experiment 1 due to reduced data access, but the results were more trustworthy. Content-Based Filtering continued to demonstrate strong and stable performance, while SVD and User-User Collaborative Filtering struggled in sparse data scenarios. This experiment reinforced that train-test evaluation is essential and highlighted early signs of where each model type excels or struggles.

**Experiment 3: Varying User and Item Engagement Levels**



In the third and most detailed experiment, the impact of user and item activity levels on model performance was evaluated by filtering the dataset at increasing engagement thresholds (users/items rated more than 100, 200, 300, or 400 times). Each subset underwent a train-test split, and models were assessed using RMSE and ranking metrics (Precision, Recall, F1@10).

Results showed clear trends: as engagement increased, so did model performance across all metrics. SVD showed strong F1-scores at higher thresholds (0.5005 at >400 ratings), although it exhibited instability in RMSE. Content-Based Filtering was consistently reliable, reaching an RMSE of 0.79 and F1 of 0.5171 at the highest threshold. User-User CF significantly improved with denser data, while Item-Item CF remained robust across all sparsity levels.

Overall, Experiment 3 highlighted the importance of interaction density in recommendation quality. It also revealed model-specific strengths, SVD and User-User CF improve with data density, while Content-Based and Item-Item CF maintain stability under sparse conditions.

**Conclusion**

The high RMSEs and moderate ranking scores, at different user activity levels, highlights the need for improved feature engineering, hyperparameter optimization, hybrid models, and potentially deep learning, areas beyond the current project's scope. Nonetheless, this analysis provides a strong foundation for future refinement and experimentation.

**References:**

1. (Text Summarization and Information Extraction, Mr. Weiss)
2. (Recommender System, Mr. Weiss)
3. (Density-based approaches: DBSCAN, OPTICS, Mr. Weiss)
4. (Hierarchical approaches: Agglomerative (AGNES) and Fuzzy C-Means, Mr. Weiss)
5. (Lecture 1 - Introduction to Clustering-1.pdf, Mr. Weiss)
6. Hex.tech. (2025). *Collaborative Filtering Using Python (with examples) | Hex*. [online] Available at: https://hex.tech/templates/data-modeling/collaborative-filtering/ [Accessed 22 May 2025].
7. Jeong, Y. (2021). *Item-Based Collaborative Filtering in Python | Towards Data Science*. [online] Towards Data Science. Available at: https://towardsdatascience.com/item-based-collaborative-filtering-in-python-91f747200fab/ [Accessed 22 May 2025].
8. Urvi Midha (2022). *Recommendation System using Collaborative Filtering in Python*. [online] Medium. Available at: https://medium.com/@urvimidha/recommendation-system-using-collaborative-filtering-in-python-83992251c8f7 [Accessed 22 May 2025].
9. Wilk, J. (2023). *How to Build a Movie Recommendation System Based on Collaborative Filtering*. [online] freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/how-to-build-a-movie-recommendation-system-based-on-collaborative-filtering/.