MOVIE RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING AND MACHINE LEARNING

A project report submitted in partial
fulfillment of the requirements for the degree of
Master of Science in
Computer Science

By

NAKUL MAGOTRA
B.E. in Computer Science, Chandigarh University, 2021

December 2025
University of Colorado Denver

This project report for the Master of Science degree by

Nakul Magotra

to be approved for the

Computer Science Program

by

Dr. Ashish Biswas, Advisor

Date 1 December, 2025

Magotra, Nakul (Master of Science, Computer Science)

MOVIE RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING AND MACHINE LEARNING

Thesis directed by Dr. Ashish Biswas

ABSTRACT

This project develops and evaluates a movie recommendation system using collaborative filtering and machine learning on a large-scale MovieLens ratings dataset. The system addresses the practical challenge of helping users discover relevant movies within a large catalogue by predicting ratings for unseen items and generating personalized Top–N recommendations.

The implementation combines user- and item-based k-nearest neighbour (kNN) collaborative filtering, matrix factorization via Truncated Singular Value Decomposition (SVD), and a feature-based gradient-boosted tree model using user and movie aggregates. A time-based leave-one-out split is used for evaluation, and performance is measured with Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE).

The empirical finding proves that the feature-based model has the highest prediction accuracy with an RMSE of about 0.93 and a MAPE of approximately 30%. This performance is superior to the performance of both baselines using kNN techniques and is further highlighted by an observation that a simple SVD baseline performs extremely poorly on this dataset, showing how important bias modeling and careful implementation are. The report concludes with a limitation and opportunity analysis to allow for significantly more advanced hybrid and neural recommendation models.

This project report is approved for recommendation to the Graduate committee.

Project Advisor:

____

Dr. Ashish Biswas

ACKNOWLEDGEMENTS

I'd really want to thank my advisor, Dr. Ashish Biswas, for always pointing me in the right direction, giving solid feedback, yet cheering me on from start to finish. A big thanks goes out to everyone teaching and working in the Computer Science and Engineering department at University of Colorado, Denver your setup pushed learning forward while handing over what was needed to get things done. On top of that, I owe a lot to my family and close friends who stayed behind me, keeping me going through thick and thin during school.

December 1, 2025

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 The Problem

Ten years have seen an exponential increase in the number of things people can watch or listen to in digital form. With streaming and online retail platforms, there exist massive waves of films and series that can be accessed like tens of thousands across the globe. This oversupply of options generates a strange kind of problem for individuals, which is being overwhelmed by the alternatives to the real enjoyment of whatever they prefer. Such systems attempt to solve the issues pertaining to recommending things you might like by analyzing the behavior obtained from your history and patterns from people who tend to share similar behavior.

For this project, clear rating data makes movie recommendations a focus for users. The data is shown as a large but spotty grid $\mathbf{R} \in \mathbb{R}^{U \times M}$, where U means total users, M represents how many films there are, while every known value $r_{u,m}$ shows a score usually from a rating system of 1 to 5. Once we remove inactive viewers and nearly unknown films, most of the spots in $\mathbf{R}$ still sit empty. The main challenge is to be able to provide content that fits into the gap so that the assistance is well received.

1.2 Motivation and Challenges

Recommenders, from the application perspective, are directly proportional to user satisfaction and retention and even generate revenue through such media platforms. A system that repeatedly surfaces relevant movies can help users quickly "settle into" a personalized experience, whereas a system that produces noisy or generic suggestions pushes users away. For a data scientist or machine learning engineer, recommender systems are also attractive because they require a blend of modeling, engineering, and evaluation skills.

However, several challenges arise in practice:

- **Extreme sparsity.** Most users rate only a small number of movies, so purely memory-based approaches can suffer from insufficient overlap between users.
- **Heterogeneous rating habits.** Some users are very strict (rarely giving 5 stars), while others are generous. Ignoring such biases can distort similarity measures and prediction quality.
- **Long-tail items.** Many movies receive only a handful of ratings. Learning robust item representations for these long-tail titles is difficult.

- **Temporal dynamics.** User preferences evolve over time, and evaluation protocols must respect temporal ordering to avoid overly optimistic performance estimates.
- **Computational constraints.** Scaling similarity computation or model training to millions of user–item interactions requires careful use of sparse data structures and efficient algorithms.

These challenges motivate the exploration of multiple modeling paradigms—neighborhood-based collaborative filtering, matrix factorization, and feature-based gradient-boosted models—within a consistent experimental framework.

1.3 Problem Statement

The project addresses the following concrete problem:

> *Given a large, sparse rating matrix derived from the MovieLens dataset, design and evaluate models that predict a user's rating for unseen movies and generate personalized Top–N recommendation lists that are more accurate than standard collaborative filtering baselines.*

More formally, for each observed rating triple $(u, m, r_{u,m})$ in the training set, a model learns a function $f(u, m)$ such that $f(u, m) \approx r_{u,m}$. At test time, the model is evaluated on a disjoint set of user–movie pairs, using metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). The ultimate goal is to minimize these error metrics while maintaining a conceptually simple and interpretable architecture suitable for an MS course project.

1.4 Research Aims (or Objectives)

The primary objectives of our project are:

- **Aim 1:** Create a clean, filtered version of the MovieLens ratings dataset that can be used with collaborative filtering and with feature-based machine learning.
- **Aim 2:** Design, implement, and evaluate classical neighborhood oriented recommenders such as user based and item-based k nearest neighbor (kNN) collaborative filtering.
- **Aim 3:** Use a matrix factorization baseline based on the application of Truncated Singular Value Decomposition (SVD) on the sparse user item matrix.
- **Aim 4:** Build user and movie-level features and train a gradient boosted regression model for rating prediction.

- **Aim 5:** Place all models under a comparative time aware evaluation protocol and analyze error patterns with visualizations and in a quantitative summary table.

1.5 Outline of the report

The rest of the organizational structure of this report is as follows: The literature on recommender systems will be presented in Chapter 2, with a summary of collaborative filtering, matrix factorization, and feature-based approaches. Chapter 3 deals with the project's methodology-data preparation, model architectures, and their implementations. The experiment concerning the results and discussions, comprising quantitative comparisons and diagnostic visualizations, will be found in Chapter 4. Finally, Chapter 5 has been given a summation of all the contributions, limitations, and future areas for research, leading to the impairment of research that has been conducted in this domain.

CHAPTER 2
LITERATURE REVIEW

A literature review helps the project to position itself among the various studies conducted in the domain of recommender system development. It highlights the main approaches that have been developed over the last two decades, with particular attention to techniques that directly inspired the models implemented in this work.

2.1 Classical Recommender Systems

Originally, the recommender systems based prior contentcentric methods or collaborative filtering. Content-based methods represent items (movies, for example) according to their attributes, such as genre, cast, and textual description, and then they learn a similarity function between items. On the contrary, collaborative filtering (CF) depends entirely on the patterns emerging in the user feedback (ratings or implicit interactions). Therefore, this quickly became a dominant method in big recommender systems since it makes use of the so-called "wisdom of the crowd" to find hidden structures that cannot be easily captured in the form of explicit metadata.

2.2 Collaborative Filtering Approaches

2.2.1 Memory-based methods

Memory-based CF can be divided into user-based and item-based variants. User-based CF computes similarity between users based on their rating vectors; predictions for a target user are obtained by aggregating ratings from the most similar users. Item-based CF instead computes similarity between items based on co-rated users; predictions for a target user–item pair are formed by aggregating the user's ratings on similar items.

Sarwar et al. demonstrated that item-based CF scales better and tends to be more stable than user-based CF on benchmarks such as MovieLens, because item similarities change slowly over time and the number of items is typically much smaller than the number of users [1]. Cosine similarity is commonly used, but Pearson correlation and adjusted cosine similarity are also popular.

2.2.2 Matrix factorization

Matrix factorization (MF) techniques were popularized in the context of the Netflix Prize [3]. MF approximates the rating matrix $\mathbf{R}$ as $\mathbf{R} \approx \mathbf{P}\mathbf{Q}^\top$, where $\mathbf{P} \in \mathbb{R}^{U \times K}$ and $\mathbf{Q} \in \mathbb{R}^{M \times K}$ contain K-dimensional latent factors for users and items, respectively. The predicted rating for user u on item m

becomes

$$\hat{r}_{u,m} = \mathbf{p}_u^\top \mathbf{q}_m + b_u + b_m + \mu,$$

where b_u and b_m are user and item biases and μ is the global mean. The parameters are typically learned by minimizing a regularized squared error objective.

In this project, a simplified MF baseline is constructed using TruncatedSVD on the sparse user–item matrix. While this does not capture all the refinements from the Netflix-era models, it provides a useful comparison point against neighborhood-based and feature-based methods.

2.3 Feature-Based and Hybrid Models

Feature-based models reframe recommendation as a supervised learning problem on manually engineered features. For each (u, m) pair, one constructs a feature vector that may include:

- User-level aggregates, such as mean rating, rating count, and variance.
- Item-level aggregates, such as average rating, rating count, and popularity indicators.
- Bias terms, such as user and item deviations from the global mean.
- Temporal indicators, like recent features or normalized timestamps.

Now, models can be trained using gradient boosting trees, random forests, and linear regression to predict ratings or ranking scores. Feature-based approaches are much more flexible and can add new signals, such as genres or textual embeddings, into their architecture without changing the overall system.

Hybrid models combine CF and content-based signals. Factorization machines and neural collaborative filtering are examples of such methods [5, 6], but their complexity is beyond the scope of an MS course project. The design in this report is instead focused on interpretable, classical techniques that can be implemented and debugged end-to-end.

2.4 Evaluation Protocols in Recommender Systems

Adomavicius and Tuzhilin [4] emphasized that evaluation protocols play a crucial role in comparing recommender algorithms. Random train/test splits often overestimate performance because they allow models to "peek into the future." For example, a model may be trained on ratings that occur after the test timestamps, which would not be available in a real deployment.

To mitigate this issue, time-aware protocols such as leave-one-out by time and rolling-origin evaluation are recommended. In this project, the primary evaluation strategy is a time-based leave-one-out split per user: for each user, the most recent rating is held out as a test case, and all earlier ratings are used for training. This design more accurately reflects how a recommender would behave in practice.

CHAPTER 3
METHODS

This chapter describes the dataset, preprocessing steps, model architectures, training procedures, and implementation details used to build the movie recommendation system.

3.1 Dataset and Preprocessing

The experiments are based on a MovieLens-style dataset containing explicit ratings on a 1–5 scale. The ratings are derived from the MovieLens dataset [7]. Each record consists of a user identifier, a movie identifier, a rating, and a timestamp. The raw data includes a mixture of highly active and very sparse users, as well as both popular and rarely rated movies.

To obtain a cleaner working set, the following filtering rules are applied:

- Users with fewer than 20 ratings are removed.
- Movies with fewer than 50 ratings are removed.

The working dataset after cleaning consists of about 1,487,348 ratings given by 10,834 users on 4,082 movies. The effect of filtering has resulted in a user-item matrix that is sparse but, in comparison with the original, it is rather dense.

After filtering, the marginal distribution of rating values can be seen in Figure 3.1, which indicates a skewed distribution towards high ratings, with the majority being ratings of 4 and 5 stars. Rating count distributions per user and per movie are depicted on a log-scale in Figures 3.2 and 3.3. These plots confirm the coexistence of a very small cluster of hyperactive users and extremely popular movies together with a long tail of moderately active ones.

3.2 Experimental Design and Evaluation Protocol

A time-based leave-one-out evaluation protocol is used. For each user, the rating with the largest timestamp is designated as the test instance; all earlier ratings are used for training. This simulates a scenario where the system is asked to predict the user's most recent choice using only prior behavior.

Let $\mathcal{D}_{\text{train}}$ denote the set of training triples $(u, m, r_{u,m})$ and $\mathcal{D}_{\text{test}}$ denote the set of held-out test triples. For a given model that produces predictions $\hat{r}_{u,m}$, the error metrics are defined as:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(u,m)\in\mathcal{D}_{\text{test}}} (r_{u,m} - \hat{r}_{u,m})^2},$$

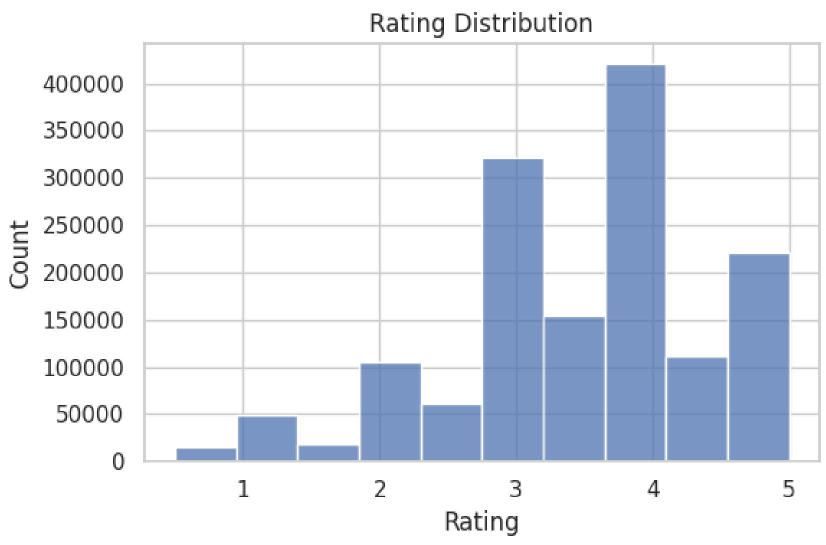

Figure 3.1: Rating distribution for the filtered MovieLens dataset.

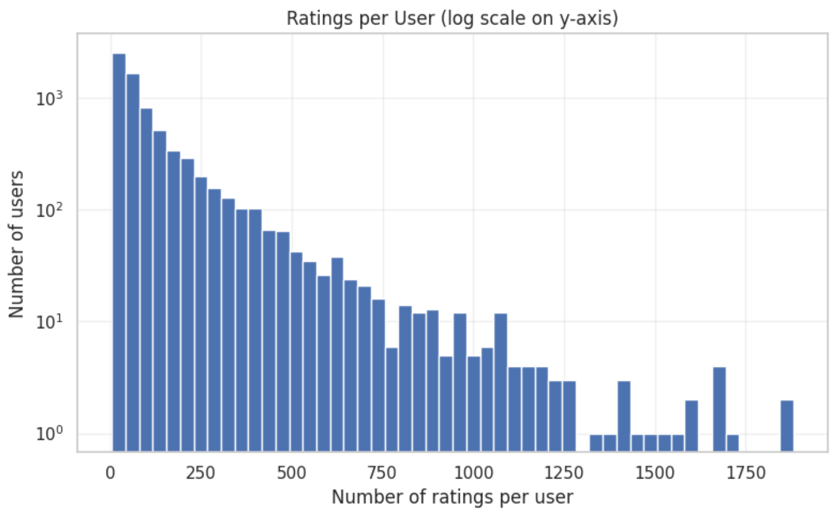

Figure 3.2: Histogram of ratings per user (log-scale y-axis).

$$\text{MAPE} = \frac{100}{|\mathcal{D}_{\text{test}}|} \sum_{(u,m)\in\mathcal{D}_{\text{test}}} \left| \frac{r_{u,m} - \hat{r}_{u,m}}{r_{u,m}} \right|.$$

RMSE penalizes large errors more heavily and is widely used in recommender competitions. MAPE provides a normalized view of the typical percentage error, which is easier to interpret for non-technical stakeholders.

3.3 Baseline Models

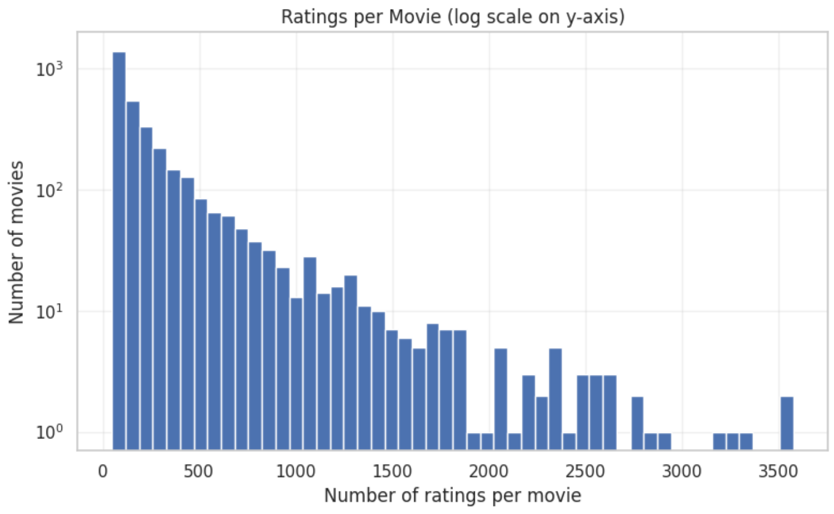

Figure 3.3: Histogram of ratings per movie (log-scale y-axis).

3.3.1 User-based kNN

The user-based kNN model represents each user as a sparse rating vector over movies. Cosine similarity between users u and v is computed using the subset of movies they have both rated. For a target pair (u, m), the model identifies the k most similar users who have rated movie m and predicts $\hat{r}_{u,m}$ as a similarity-weighted average of their ratings. To balance robustness and locality in this project, k is set to a moderate value, such as 50.

3.3.2 Item-based kNN

The item-based kNN model mirrors the user-based approach but works in the dual space of items. Each movie is represented as a sparse vector of ratings from users. Cosine similarity is computed between movies, and predictions are formed by aggregating the target user's ratings on the nearest-neighbour movies. This model tends to be more stable than user-based kNN because item similarities change slowly as new users join the system.

3.3.3 SVD baseline

The SVD baseline applies TruncatedSVD to the user–item matrix to obtain low-dimensional embeddings for users and items. The model reconstructs approximate ratings via dot products in the latent space. In the implementation, a modest latent dimension (e.g., $K = 50$) is used to keep training time reasonable. Unlike full-blown MF with bias terms and regularization, this SVD baseline is intentionally simple and serves as a sanity check for latent-factor modeling.

3.4 Feature-Based Gradient-Boosted Model

The feature-based model treats rating prediction as a supervised learning problem. For each training triple $(u, m, r_{u,m})$, a feature vector is constructed that includes:

- Global mean rating.
- User mean rating and user rating count.
- Movie mean rating and movie rating count.
- User bias (user_mean – global_mean).
- Movie bias (movie_mean – global_mean).
- Normalized timestamp (e.g., fraction of the time range).

It is then modeled using gradient-boosted decision tree regressor (for example, either XGBoost or GradientBoostingRegressor) to predict the minima squared error by using a subsample of ratings. Hyperparameters such as the number of trees, learning rate, and maximum depth are tuned on a validation subset. The trained model produces predictions $\hat{r}_{u,m}$ on held-out instances.

3.5 Implementation Details

This system is built on Python-based data manipulation libraries like Pandas and NumPy, SciPy for sparse matrices, scikit-learn for kNN and SVD components, XGBoost or scikit-learn's GradientBoostingRegressor for the feature-based model, Matplotlib , and Seaborn to generate the figures in this report.

The user and movie identifiers are mapped to contiguous integer indices for efficient indexing into sparse matrices. Cosine similarity is computed using vectorized operations and sparse matrix multiplication where possible. The experiments were conducted on a modern workstation with sufficient RAM to hold the filtered dataset in memory.

CHAPTER 4
RESULTS AND DISCUSSIONS

This chapter presents the quantitative and qualitative results obtained from the models described in Chapter 3 and discusses their implications.

4.1 Quantitative Results

Table 4.1 summarizes the RMSE and MAPE scores achieved by each model on the time-based leave-one-out test set (for the CF and SVD models) or the held-out subset (for the feature-based model).

Table 4.1: Summary of quantitative performance for all models.

Model	**RMSE**	**MAPE (%)**
User-kNN	1.02	31.67
Item-kNN	0.97	30.67
SVD (TruncatedSVD)	3.10	76.21
XGBoost-ML	0.93	30.21

The feature-based XGBoost-ML model achieves the lowest RMSE and MAPE, indicating that it captures both user and item structure more effectively than the kNN and SVD baselines. Item-kNN outperforms User-kNN, consistent with prior literature that finds item-based methods more stable on large, sparse datasets. The SVD baseline performs poorly, largely because it lacks bias terms and regularization and is not tuned as aggressively as production MF systems.

Figures 4.1 and 4.2 visualize these differences in bar-chart form, making it easy to see the gap between the SVD baseline and the other methods, as well as the modest improvement of the feature-based model over Item-kNN.

4.2 Error and Residual Analysis

Beyond aggregate metrics, it is important to understand how each model behaves across the rating scale. For both the SVD and XGBoost-ML models, predicted vs. true plots, residual histograms, and RMSE-by-bucket plots are examined.

Figures 4.3–4.5 show the diagnostics for the SVD model. The predicted vs. true scatter plot exhibits a very diffuse cloud, indicating that SVD often produces ratings far from the correct value. The residual histogram is wide and relatively flat, with substantial mass at large positive and negative errors. RMSE is high across all rating buckets, confirming that the model is underfitting.

In contrast, Figures 4.6–4.8 show the diagnostics for XGBoost-ML. The scatter plot is much tighter around the diagonal, especially for the frequent 3–5 star ratings. The residual histogram is narrower and

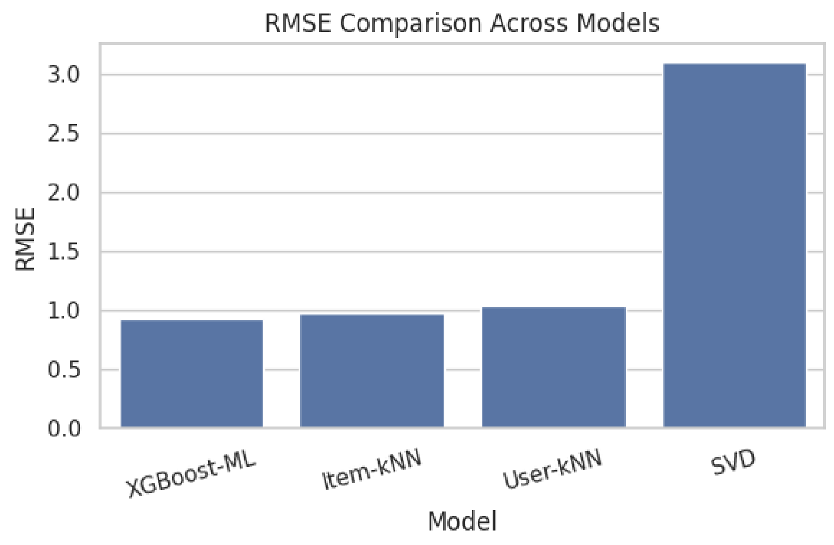

Figure 4.1: RMSE comparison across Item-kNN, User-kNN, SVD, and XGBoost-ML models.

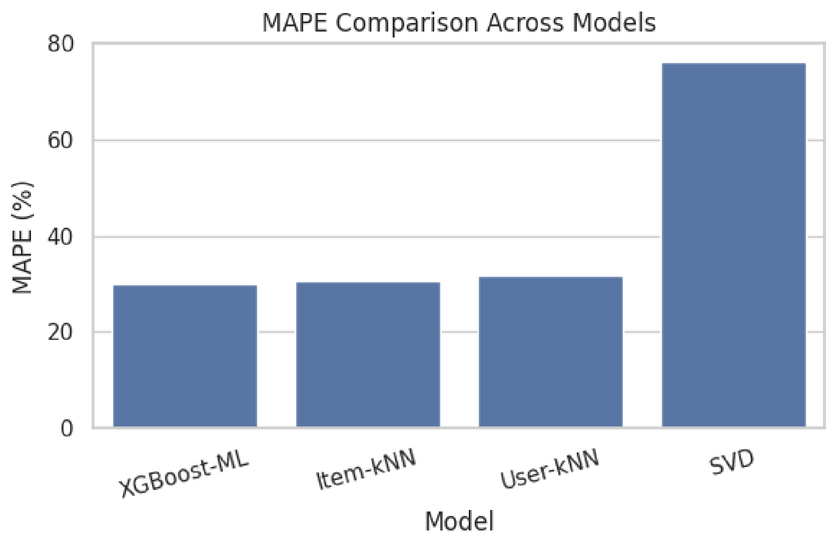

Figure 4.2: MAPE comparison across Item-kNN, User-kNN, SVD, and XGBoost-ML models.

more symmetric, and the RMSE-by-bucket plot reveals substantially lower error in every rating range. These visualizations reinforce the numerical metrics and provide confidence that the feature-based model is genuinely capturing useful structure in the data.

4.3 Similarity Structure and Qualitative Recommendations

To interpret the item-based collaborative filtering component, an item–item cosine similarity heatmap is generated for a random sample of 25 movies (Figure 4.9). Block-diagonal patterns in the heatmap indicate clusters of movies that are frequently co-rated and therefore considered similar by the model.

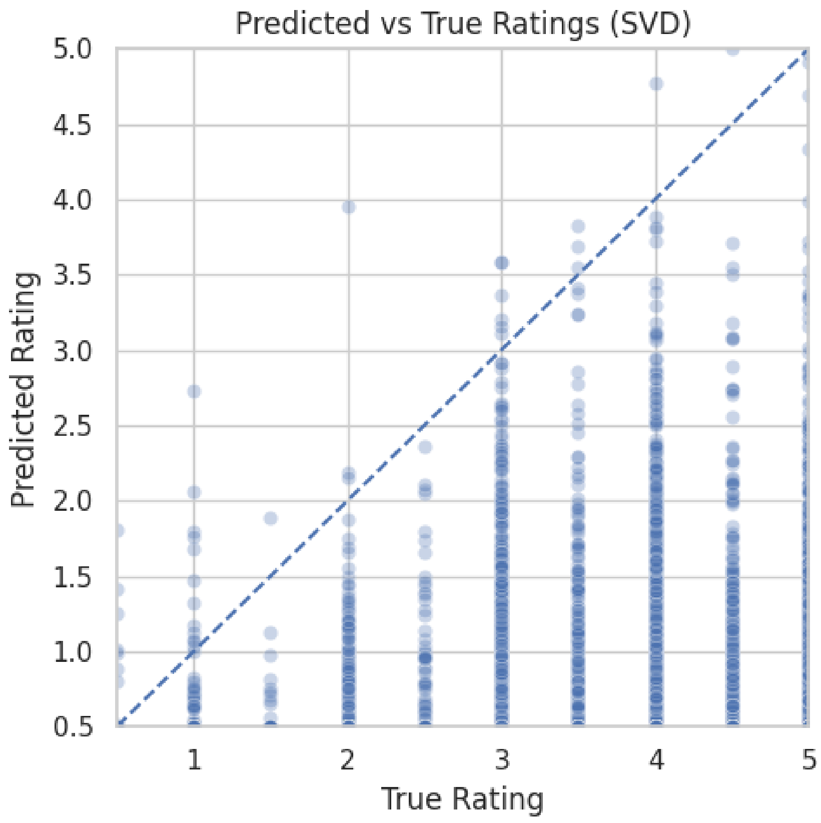

Figure 4.3: Predicted vs. true ratings for the SVD model.

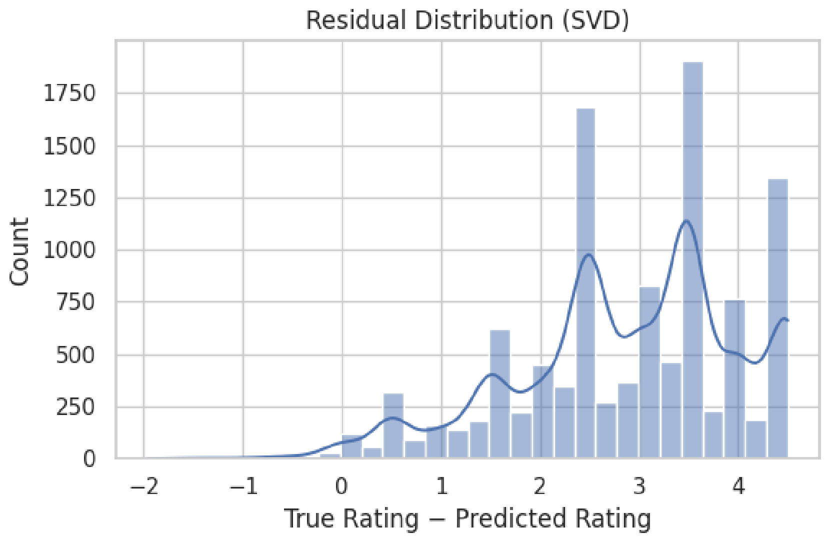

Figure 4.4: Residual distribution for the SVD model.

These clusters often correspond to intuitive groupings such as genre, era, or shared actors and directors.

Using the trained models, Top–N recommendation lists are generated for selected users. For users with a diverse rating history, the recommended lists include a mixture of popular, highly rated titles and

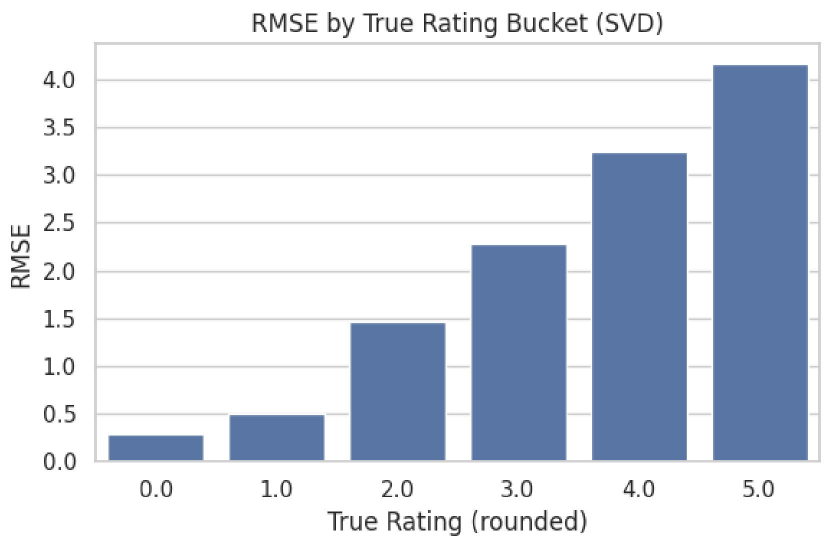

Figure 4.5: RMSE by true rating bucket for the SVD model.

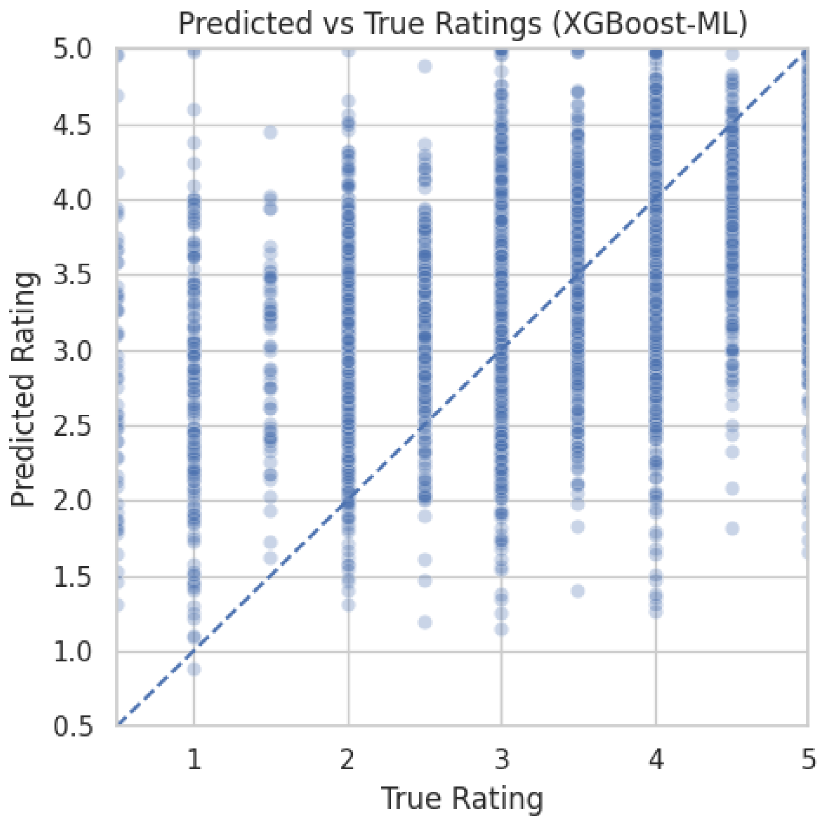

Figure 4.6: Predicted vs. true ratings for the XGBoost-ML model.

less mainstream movies that align with the user's past preferences. For newer users with fewer ratings, the lists tend to be dominated by popular items, reflecting the underlying data sparsity and the difficulty of cold-start scenarios.

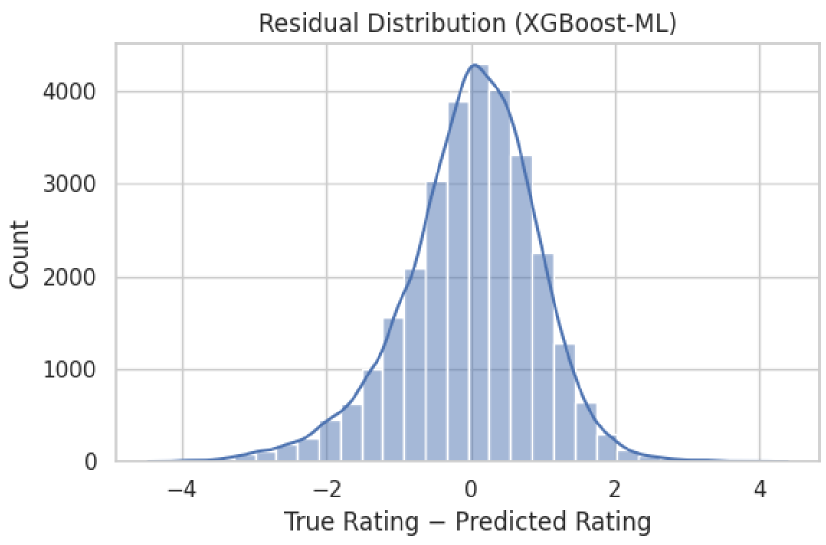

Figure 4.7: Residual distribution for the XGBoost-ML model.

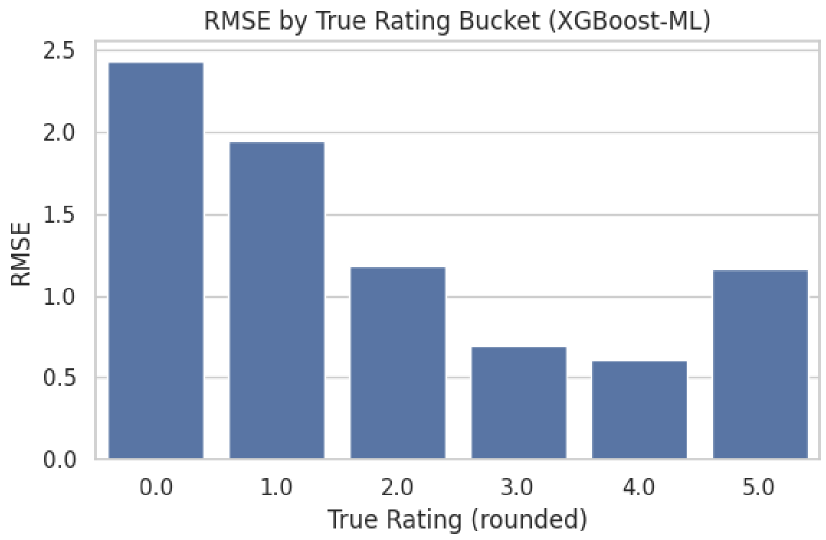

Figure 4.8: RMSE by true rating bucket for the XGBoost-ML model.

4.4 Discussion

Overall, the experiments support three main findings:

1. **Feature-based models can outperform classical CF baselines.** Even with relatively simple engineered features, the gradient-boosted model achieves better RMSE and MAPE than both User-kNN and Item-kNN, illustrating the value of combining collaborative information with aggregate statistics.

2. **Implementation details matter for MF.** The poor performance of the SVD baseline is not an

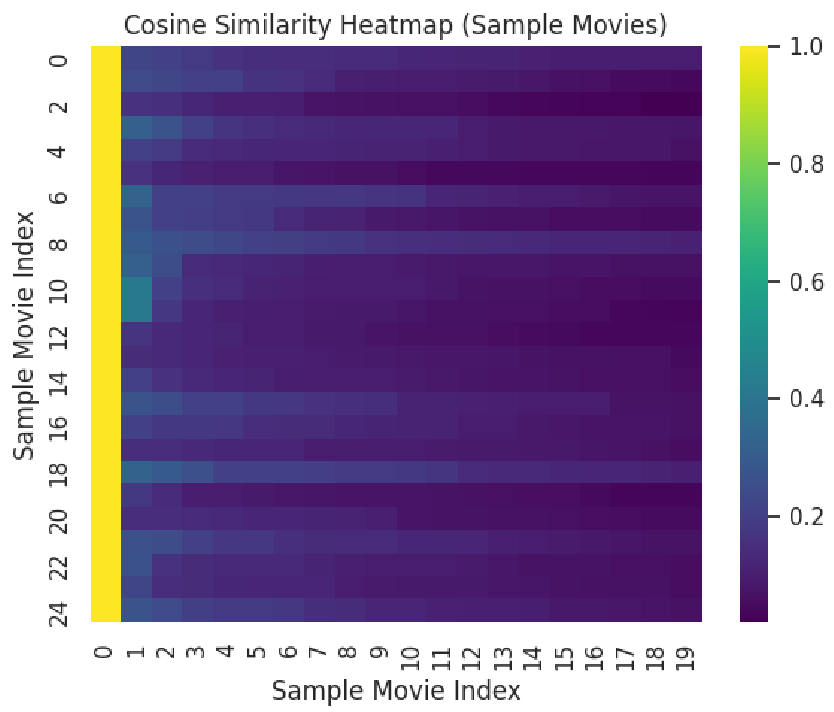

Figure 4.9: Item–item cosine similarity heatmap for a sample of 25 movies.

indictment of matrix factorization as a technique; rather, it reflects the limitations of a minimal TruncatedSVD model without bias terms or regularization. A more carefully implemented MF model would likely perform closer to, or better than, the feature-based approach.

3. **Evaluation protocols must respect time.** The time-based leave-one-out protocol used here provides a more realistic picture of performance than a random split, especially for users whose preferences evolve. This design decision is important for any future extensions of the system.

At the same time, several threats to validity should be noted. The experiments focus on a single dataset, and the feature-based model is evaluated on a subsample for computational reasons. Hyperparameter tuning is modest, and ranking-based metrics such as Recall@K and NDCG@K are not considered. These limitations motivate the future work outlined in Chapter 5.

CHAPTER 5
CONCLUSIONS

5.1 Summary

This project report presented the design, implementation, and evaluation of a movie recommendation system using a filtered MovieLens dataset. After preprocessing, the dataset contained approximately 1.49 million ratings from over 10,000 users on more than 4,000 movies. A time-based leave-one-out protocol was used to evaluate collaborative filtering and SVD models, and a feature-based gradient-boosted regressor was trained on engineered user and item features.

Among the tested models, the feature-based XGBoost-ML approach achieved the best quantitative performance, with RMSE below 1.0 and the lowest MAPE. Item-kNN outperformed User-kNN, while the simple SVD baseline underperformed due to its limited modeling capacity. Diagnostic plots and similarity heatmaps provided additional insight into how the models behaved across the rating scale and how they organized movies in the latent space.

5.2 Contributions

The main contributions of this project are:

- A complete, reproducible pipeline for building and evaluating a movie recommendation system, including data preprocessing, model training, and visualization.
- An empirical comparison of User-kNN, Item-kNN, a simple SVD baseline, and a feature-based gradient-boosted model on a realistic, time-aware evaluation protocol.
- A set of visual diagnostics—including error distributions, RMSE-by-bucket plots, and similarity heatmaps—that help interpret model behavior beyond aggregate metrics.

From an educational perspective, the project demonstrates how classical recommender techniques can be implemented and analyzed within the constraints of an MS course project, and it provides a foundation for more advanced research or industrial development.

5.3 Future Research

Several promising directions remain for future work:

- **Stronger matrix factorization baselines.** Implement a fully regularized MF model with user and item biases and compare it more rigorously against the feature-based approach.

- **Incorporating content features.** Extend the feature set with genres, release year, and textual embeddings of movie descriptions, and explore hybrid models such as factorization machines.

- **Ranking-focused evaluation.** Complement RMSE and MAPE with ranking metrics such as Recall@K, Precision@K, and NDCG@K to better capture the quality of Top–N recommendation lists.

- **Neural recommenders and sequence modeling.** Investigate neural collaborative filtering, sequence-aware models, and attention mechanisms to capture temporal dynamics and complex user behavior.

- **Fairness and diversity.** Examine the degree to which the system overrecommends well-liked items and investigate re-ranking techniques to enhance diversity, originality, and equity among users and items.

REFERENCES

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proc. 10th Intl. World Wide Web Conf.*, 2001.

[2] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proc. UAI*, 1998.

[3] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” *IEEE TKDE*, vol. 17, no. 6, pp. 734–749, 2005.

[5] S. Rendle, “Factorization machines,” in *Proc. ICDM*, 2010.

[6] X. He *et al.*, “Neural collaborative filtering,” in *Proc. WWW*, 2017.

[7] F. M. Harper and J. A. Konstan, “The MovieLens datasets: History and context,” *ACM TIIS*, vol. 5, no. 4, 2016.

APPENDIX A
ABBREVIATIONS AND NOTATION

A.1 Abbreviations

- **CF** – Collaborative Filtering
- **kNN** – k-Nearest Neighbours
- **MF** – Matrix Factorization
- **RMSE** – Root Mean Squared Error
- **MAPE** – Mean Absolute Percentage Error
- **SVD** – Singular Value Decomposition

A.2 Notation

- U – number of users in the dataset.
- M – number of movies in the dataset.
- $r_{u,m}$ – true rating given by user u to movie m.
- $\hat{r}_{u,m}$ – predicted rating for user u on movie m.
- $\mathbf{R}$ – user–item rating matrix.
- $\mathbf{P}, \mathbf{Q}$ – user and item latent factor matrices in MF.

APPENDIX B
ADDITIONAL RESULTS AND DISCUSSIONS

This appendix provides additional details on training and evaluation that complement the main chapters.

B.1 Hyperparameters and Training Details

A small validation set was used to select a suitable neighborhood size k for the kNN models from $\{20, 50, 100\}$. Cosine similarity was employed all through for sparse calculation of similarities to cut down on memory use.

Confined validation set to select suitable neighborhood size k for kNN models within range $\{20, 50, 100\}$. A cosine similarity throughout was sparsely calculated for memory space reason.

The latent factors used for the SVD baseline varied between $\{20, 50, 100\}$. With regards to reconstruction quality versus cost, about 50 happened to be a reasonable compromise; however, the exclusion of bias terms limited overall performance.

There were some key hyperparameters for the XGBoost-ML Model. Number of trees, learning rate, and maximum tree depth were among them. The learning rate was moderate and yielded a few hundred trees at an intermediate depth, at levels 4–6, leading to a stable performance not prone to overfitting.

B.2 Computation Time and Resource Usage

All experiments were conducted on a single workstation. Training the kNN models was dominated by similarity computation, which benefited significantly from vectorized operations and sparse matrix representations. The SVD baseline was relatively fast to compute using efficient sparse linear algebra routines. The feature-based model required the most wall-clock time due to feature engineering and tree-based learning but remained manageable within an MS course project setting.