Journal of Autonomous Intelligence (2024) Volume 7 Issue 2
doi: 10.32629/jai.v7i2.1182

ORIGINAL RESEARCH ARTICLE

Comparative analysis of collaborative filtering recommender system algorithms for e-commerce

Kapil Saini[*], Ajmer Singh

Department of Computer Science & Engineering, Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Haryana 131039, India

*** Corresponding author:** Kapil Saini, 19001901905kapil@dcrustm.org

ABSTRACT

Collaborative recommender systems are information filtering systems that seek to predict a user's rating or preference for an item. They play a vital role in various business use cases, such as personalized recommendations, item ranking and sorting, targeted marketing and promotions, content curation and catalog organization, and feedback analysis and quality control. When evaluating these systems, rating prediction metrics are commonly employed. Efficiency, including the prediction time, is another crucial aspect to consider. In this study, the performance of different algorithms was investigated. The study employed a dataset consisting of e-commerce product ratings and assessed the algorithms based on rating prediction metrics and efficiency. The results demonstrated that each algorithm had its own set of strengths and weaknesses. For the metric of Root Mean Squared Error (RMSE), the BaselineOnly algorithm achieved the lowest mean value. Regarding Mean Absolute Error (MAE), the Singular Value Decomposition with Positive Perturbations Singular Value Decomposition with Positive Perturbations (SVDPP) algorithm exhibited the lowest mean value; Mean Squared Error (MSE) also achieved the lowest mean value. Moreover, the BaselineOnly algorithm showcased superior performance with the lowest mean test times when considering efficiency. Researchers and practitioners can use the findings of this study to select the best algorithm for a particular application. Researchers can develop new algorithms that combine the strengths of different algorithms. Practitioners can also use the findings of this study to tune the parameters of existing algorithms.

Keywords: recommender system; collaborative filtering; e-commerce

ARTICLE INFO

Received: 8 September 2023
Accepted: 20 September 2023
Available online: 30 November 2023

1. Introduction

Collaborative recommender systems are tools that provide recommendations based on the preferences and behavior of multiple users[1]. In recent years, these systems have grown in popularity because to their capacity to tailor recommendations for users and enhance the overall user experience. Collaborative recommender systems use data on the preferences and behavior of multiple users to generate recommendations for each individual user[2]. The system identifies similarities between users and their preferences, and then uses these similarities to make recommendations to each user[3]. This differs from content-based recommender systems, which generate recommendations based on information about the objects' qualities. A key challenge in collaborative recommender systems is the sparsity problem, this differs from content-based recommender systems, which generate recommendations based on information about the objects' qualities. To address this problem, several techniques have been developed, including neighborhood-based approaches and matrix factorization methods[4]. Neighborhood-based approaches involve identifying a set of users with similar preferences and using their ratings to make recommendations. Matrix factorization methods involve decomposing the rating matrix into two low-rank matrices, which can be used to predict ratings for new items. Several challenges are associated with evaluating collaborative recommender systems, including the need for diverse evaluation metrics and the difficulty of evaluating the quality of recommendations in real-world settings. Researchers have proposed new evaluation metrics to address these challenges and developed simulation techniques to evaluate recommender systems in realistic scenarios [2]. Overall, collaborative recommender systems are effective instruments for generating individualised recommendations for consumers based on the preferences and actions of several users. While there are several challenges associated with these systems, ongoing research is focused on addressing these challenges and improving the quality of recommendations for users[4]. The problem that this research paper aims to address is the need to evaluate and compare the performance of different collaborative filtering algorithms for a personalized recommendation. Several factors influence the effectiveness of collaborative filtering, such as data sparsity, cold-start, scalability, and interpretability[5]. Additionally, there is a need to investigate how collaborative filtering can be extended to handle evolving data streams, non-stationary recommendation scenarios, and hybrid approaches that combine multiple techniques.

In recommendation systems, collaborative filtering is a popular strategy. There are a number of collaborative filtering algorithm types, including user-based, item-based, model-based, content-based, cluster-based, and hybrid approaches, that combine various similarity measures[6]. Specifically, user-to-user and item-to-item collaborative filtering are two prevalent recommendation system techniques[7]. In addition, there are numerous sorts of improved algorithms for traditional measuring metrics that can produce superior outcomes compared to existing algorithms[8]. In addition, a recent study presented a novel collaborative filtering method based on a restricted random walk, which accounts for the granularity of users' preference drifting and item popularity bias to increase the precision of suggestions[9]. Another study examined two collaborative filtering algorithms, user-based and item-based collaborative filtering algorithms, which use the sparse matrix to evaluate cosine similarity between users, items, and users to items[10]. This study aims to examine and evaluate algorithm performance in a recommender system by analysing fit time, test time, and performance metrics. The study aims to contribute to the understanding of algorithm selection and its implications in recommendation systems while also identifying avenues for future research to enhance the field further.

2. Literature review

The field of recommender systems has witnessed significant advancements in recent years, particularly in the domain of collaborative filtering. Collaborative filtering is a popular approach to generating personalized recommendations by analyzing user behavior, item characteristics, and user-item interactions. In this section, we provide a comprehensive analysis of existing research on collaborative filtering recommender system algorithms, highlighting their strengths, limitations, and advancements. The review aims to contribute to a deeper understanding of the various techniques employed in collaborative filtering and identify potential areas for future research. Several studies have focused on different collaborative filtering techniques, including memory-based, model-based, and hybrid models[11]. Memory-based approaches utilize user-item rating data to identify similarities between users or items and generate recommendations based on nearest neighbors. Model-based methods, on the other hand, employ machine learning algorithms to build predictive models based on user-item interactions and other auxiliary data. Hybrid models combine the strengths of both memory-based and model-based techniques to overcome their individual limitations. The cold start problem, which refers to the difficulty of providing appropriate recommendations for new users or objects with low or no interaction data, is one of the greatest obstacles in collaborative filtering. Various strategies have been proposed to address this challenge, such as content-based filtering, which incorporates item features or attributes to make initial recommendations[12]. Another significant challenge is the sparsity of data, where the available user-item interactions are insufficient to make reliable predictions. To mitigate this issue, researchers have explored techniques like matrix factorization and neighbourhood-based methods to handle sparse data scenarios[13] effectively. The selection of similarity measurements significantly impacts the efficiency of collaborative filtering algorithms. Numerous metrics, including cosine similarity, Pearson correlation, and Jaccard similarity, have been used to quantify the similarities between users or objects. Literature indicates that selecting a suitable similarity measure is contingent on the parameters of the recommendation task and the available data[14]. To ensure the reliability and validity of our literature review, we adopted a systematic approach, considering relevant studies and research papers from interdisciplinary fields such as computer science, information retrieval, and machine learning[15]. The selected references provide a comprehensive understanding of the collaborative filtering techniques, challenges, and advancements in the field of recommender systems. Additionally, they serve as valuable resources for researchers, practitioners, and decision-makers interested in this study area. The findings from this literature review lay the foundation for our comparative analysis of collaborative filtering recommender system algorithms. By examining the strengths and limitations of different approaches, we aim to provide insights into their effectiveness, scalability, and computational complexity. The comparative analysis will assist in identifying the most suitable algorithm for specific recommendation scenarios and potentially highlight areas for further research. Collaborative recommender systems are gaining increasing importance in various fields as they help personalise user recommendations based on their preferences and behaviour. Several studies have focused on the techniques and algorithms used in collaborative filtering-based recommender systems[16]. Collaborative filtering methods utilize the similarities and differences in user preferences to recommend items to users. Recent studies have investigated the use of review texts to enhance the precision of collaborative filtering-based recommender systems. Other research has investigated the use of hybrid models that integrate collaborative filtering with other recommendation methods, such as content-based filtering and matrix factorization[17]. Hybrid models can improve the performance of collaborative recommender systems by utilizing the strengths of multiple techniques. Researchers have also proposed novel approaches to collaborative filtering, such as social-based filtering and group-based filtering. Social-based filtering considers the influence of social connections among users, while group-based filtering utilizes the characteristics of user groups to improve recommendation accuracy[18]. Collaborative recommender systems are a form of a recommender system that recommends things to users based on user ratings or behaviour. There are a number of different algorithms that can be used to build collaborative recommender systems, each with its own advantages and disadvantages are shown below[19–22] (**Table 1**).

Table 1. Advantages & disadvantages of different algorithms for the collaborative recommender system.

Algorithm	Description	Pros	Cons
BaselineOnly	Collaborative filtering algorithm that predicts ratings based on baseline estimates.	-Simplicity and efficiency. -Handles both explicit and implicit ratings. -Robust to sparsity and cold start scenarios.	-May not capture complex user-item interactions. -Limited in handling contextual information.
Singular Value Decomposition (SVD)	Matrix factorization-based algorithm using singular value decomposition.	-Captures latent factors and complex relationships. -Performs well with sparse data. -Effective in handling large datasets.	-Computationally expensive for large matrices. -Requires parameter tuning. -Cold start problem for new users/items.
SVDpp	Enhanced version of SVD algorithm that considers implicit feedback and incorporates additional factors.	-Accounts for implicit feedback. -Better captures user preferences. -Can improve recommendations for new users/items.	-Increased computational complexity. -Requires more data for accurate predictions. -Vulnerable to overfitting.

Table 1. (*Continued*).

Algorithm	Description	Pros	Cons
KNNBaseline	Collaborative filtering algorithm that incorporates baseline estimates and utilizes a similarity-based approach.	-Accounts for baseline estimates. -Handles both explicit and implicit ratings. -Performs well in situations with sparse data.	-Computationally expensive for large datasets. -Sensitive to the choice of similarity metrics. -Cold start problem for new users/items.
KNNWithMeans	Collaborative filtering algorithm that utilizes a similarity-based approach and incorporates mean ratings.	-Simple and easy to implement. -Handles both explicit and implicit ratings. -Robust to sparsity and cold start scenarios.	-Computationally expensive for large datasets. -Sensitive to the choice of similarity metrics. -Cold start problem for new users/items.
KNNWithZScore	Collaborative filtering algorithm that utilizes a similarity-based approach and incorporates mean and standard deviation ratings.	-Handles both explicit and implicit ratings. -Accounts for mean and standard deviation. -Robust to sparsity and cold start scenarios.	-Computationally expensive for large datasets. -Sensitive to the choice of similarity metrics. -Cold start problem for new users/items.
CoClustering	Matrix factorization-based algorithm that simultaneously clusters users and items based on their ratings.	-Captures both user and item clusters -Can handle large datasets. -Effective in handling sparse and high-dimensional data.	-Computational complexity increases with the number of clusters. -May struggle with highly imbalanced data.
SlopeOne	Collaborative filtering algorithm that calculates deviations between item ratings.	-Simple and efficient. -Handles both explicit and implicit ratings. -Robust to sparsity and cold start scenarios.	-Limited in capturing complex user-item interactions. -Cold start problem for new users/items.
KNNBasic	Collaborative filtering algorithm that utilizes a simple similarity-based approach.	-Simple and easy to implement. -Handles both explicit and implicit ratings. -Robust to sparsity and cold start scenarios.	-Computationally expensive for large datasets. -Sensitive to the choice of similarity metrics. -Cold start problem for new users/items.
Non-negative matrix factorization (NMF)	Matrix factorization-based algorithm using non-negative matrix factorization.	-Interpretable latent factors. -Handles sparse data. -Can capture non-linear relationships.	-Requires careful initialization and tuning. -May not perform well with highly sparse or noisy data.
NormalPredictor	Algorithm that predicts ratings based on the distribution of the training set ratings.	-Simple and easy to implement. -Provides a baseline performance for comparison.	-Ignores user-item interactions and preferences. -Limited accuracy compared to more advanced algorithms.

3. Methodology

In this study, we experimentally evaluated various collaborative filtering algorithms to build a recommendation system. The following parameter values of each algorithm were used to ensure consistency and fairness in the comparison.

3.1. Experimental setup

In this study, we evaluated various collaborative filtering algorithms to build a recommendation system with the help of a surprise library[23]. The following parameter values of each algorithm were used to ensure consistency and fairness in the comparison.

1) Experimental design: The dataset was split into training and test sets using a train-test split of 80:20. This allowed us to simulate real-world scenarios by training the algorithms on some of the data and evaluating their performance on unseen ratings.

2) Algorithms: We selected several algorithms, including SVD, SVDpp, SlopeOne, NMF, NormalPredictor, KNNBaseline, KNNBasic, KNNWithMeans, KNNWithZScore, BaselineOnly, and CoClustering.
3) Training: Each algorithm was instantiated and trained on the training set using the default parameter values. The training process involved capturing the underlying patterns in the user-item ratings to build prediction models.
4) Evaluation: The algorithms were evaluated on the test set to measure their performance after training.
5) Parameter values: The following parameters are used in the experiment setup:
   i. n_epochs: This parameter represents the number of iterations or epochs used during the training process. Each epoch consists of one pass through the entire dataset. It is used in SVD, SVDpp, NMF, KNNBaseline, KNNBasic, KNNWithMeans, KNNWithZScore, BaselineOnly, and CoClustering algorithms. The value of n_epochs can be adjusted based on the model’s convergence and the dataset’s characteristics.
   ii. biased: This parameter indicates whether the model should include biased terms or not. Bias terms capture the overall tendencies or biases of users and items in the recommendation system. It is used in SVD, SVDpp, and NMF algorithms.
   iii. init_mean: This parameter controls the mean value of the initial random values assigned to the model’s parameters during initialization. It is used in SVD, SVDpp, and NMF algorithms.
   iv. init_std: This parameter controls the standard deviation of the initial random values assigned to the model’s parameters during initialization. It is used in SVD, SVDpp, and NMF algorithms.
   v. lr_all: This parameter represents the learning rate for the optimization algorithm used to train the model. It determines the step size taken during parameter updates. It is used in SVD and SVDpp algorithms.
   vi. reg_all: This parameter controls the regularization strength applied to all parameters in the model. Regularization helps prevent overfitting by adding a penalty term to the loss function. It is used in SVD and SVDpp algorithms.
   vii. k: This parameter determines the number of nearest neighbours to consider in the k-NN algorithms (KNNBaseline, KNNBasic, KNNWithMeans, KNNWithZScore). It specifies the size of the neighbourhood used for making predictions.
   viii. min_k: This parameter sets the minimum number of neighbours required for a prediction to be made in the k-NN algorithms (KNNBaseline, KNNBasic, KNNWithMeans, KNNWithZScore). The prediction cannot be made if the number of neighbours falls below this threshold.

3.2. Evaluation metrics

RMSE (root mean squared error) is an evaluation metric that is commonly used to measure the accuracy of a recommender system. RMSE measures the difference between predicted and actual ratings on a scale of 0–5 (or another rating scale). It is calculated as the square root of the average of the squared differences between predicted and actual ratings (Equation (1))[24]:

$$\mathrm{RMSE} = \frac{\sqrt[2]{\Sigma(\acute{R} - R)^2}}{N} \tag{1}$$

where N is the total number of ratings in the dataset, $\acute{R}$ is the predicted rating and R represents the actual ratings.

The lower the RMSE value, the more accurate the recommendations made by the system. RMSE is a useful metric because it considers both the magnitude and direction of the prediction errors. A high RMSE value indicates that the system is not very accurate in predicting user ratings, while a low RMSE value indicates that the system is making accurate predictions.

RMSE is particularly useful when the ratings are continuous, and the distribution is normal, as it provides a way to measure the error in terms of standard deviations. However, it may not be as useful when the ratings are binary or ordinal, as it assumes a continuous scale of ratings.

When comparing different collaborative filtering algorithms, the algorithm with the lowest RMSE value is typically considered the best-performing algorithm. However, it's important to note that RMSE is just one of many evaluation metrics that can be used and that different metrics may be more appropriate depending on the specific goals of the recommender system. Additionally, RMSE may not capture other important aspects of recommendation quality, such as diversity or novelty. Therefore, it's important to use multiple evaluation metrics to get a more complete picture of the performance of different algorithms.

The mean absolute error (MAE) is an evaluation metric that measures the average absolute difference between predicted and actual values in a dataset[24]. It is commonly employed in regression and forecasting tasks. MAE calculates the average magnitude of errors without considering their direction, making it a useful metric for assessing the overall model performance. The MAE is computed by taking the absolute difference between each predicted value and its corresponding actual value, summing up these differences, and then dividing by the total number of observations (Equation (2)). The formula for MAE can be represented as:

$$\text{MAE} = \frac{\sum_{i=1}^{N} |y_i - x_i|}{N} \tag{2}$$

where: N is the total number of observations; y_i represents the actual values; x_i represents the predicted values.

The MAE metric provides a straightforward interpretation since it represents the average absolute error in the same units as the predicted and actual values. A lower MAE indicates better accuracy and closer alignment between predictions and actual values.

The mean squared error (MSE) is a widely used evaluation metric in collaborative recommender systems for comparing and assessing the performance of different algorithms[25,26]. It quantifies the average squared difference between the predicted ratings and the true ratings of items in the system, providing a measure of the accuracy and precision of the recommendations (Equation (3)). The MSE can be calculated using the following equation:

$$\text{MSE} = \frac{\sum(\hat{R} - R)^2}{N} \tag{3}$$

where $\hat{R}$ represents the predicted ratings, R represents the true ratings, and N is the total number of ratings. A lower MSE value indicates better accuracy and effectiveness of the algorithms in generating accurate recommendations[27]. MSE enables benchmarking and comparison of different algorithms, making it a valuable metric in the field of recommender systems[25]. However, it is important to consider other evaluation metrics alongside MSE to obtain a comprehensive assessment of algorithm performance. Mean absolute error (MAE) and root mean squared error (RMSE) are commonly used metrics in conjunction with MSE[25]. MAE represents the average absolute difference between the predicted and true ratings, while RMSE captures the square root of the average squared difference. These metrics provide additional insights into the performance of recommender systems.

4. Results and discussion

In this section, we discussed the result generated after executing the experiment. Which is divided into two parts: (i) mean and standard deviation of RMSE, MAE, and MSE for different algorithms, (ii) mean and standard deviation of fit time and test time for different algorithms. The provided **Table 2** presents information about different algorithms in terms of their RMSE (root mean squared error), MAE (mean absolute error), and MSE (mean squared error). The algorithms listed in the table are SVD, SVDPP, SLOPEONE, NMF, NormalPredictor, KNNBaseline, KNNBasic, KNNWithMeans, KNNWithZScore, BaselineOnly, and CoClustering. The table includes mean and standard deviation values for each metric. **Figure 1** represents the metric comparison of mean deviation for different algorithms.

The **Table 2** provides insights into the performance of the algorithms based on their RMSE, MAE, and MSE values. These metrics are commonly used in regression tasks to evaluate the models' accuracy and assess the prediction quality.

i. RMSE: The root mean square error (RMSE) measures the average magnitude of errors between predicted and actual values. A smaller RMSE indicates a more accurate model, as it indicates that the predicted values are closer to the actual values. In the provided table, the RMSE values range from 0.9330 to 1.2814, indicating algorithms with varying degrees of accuracy. RMSE can be used in scenarios where accurate estimation of prediction errors is essential, such as personalized recommendations, item ranking, and sorting, where accurate predictions are crucial for user satisfaction.

Table 2. Mean and standard deviation of RMSE, MAE, and MSE for different algorithms.

ALGORITHMS	**RMSE**		**MAE**		**MSE**	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
SVD	0.9356	0.0049	0.6916	0.0048	0.8794	0.0267
SVDPP	0.9379	0.0095	0.6836	0.0033	0.8726	0.0208
SLOPEONE	1.06	0.0025	0.76	0.0036	1.1182	0.0138
NMF	1.1269	0.0091	0.8571	0.0012	1.264	0.0109
NormalPredictor	1.2814	0.0104	0.9503	0.0026	1.6363	0.0079
KNNBaseline	1.0135	0.0057	0.7080	0.0012	1.0357	0.0178
KNNBasic	1.0736	0.0132	0.7480	0.0084	1.1494	0.0104
KNNWithMeans	1.0153	0.0047	0.7231	0.0054	1.0466	0.0069
KNNWithZScore	1.0248	0.0062	0.7144	0.0001	1.0650	0.0263
BaselineOnly	0.9330	0.0078	1.0650	0.0263	1.0650	0.0263
Clustering	1.0381	0.0099	0.7221	0.0019	1.0807	0.30607

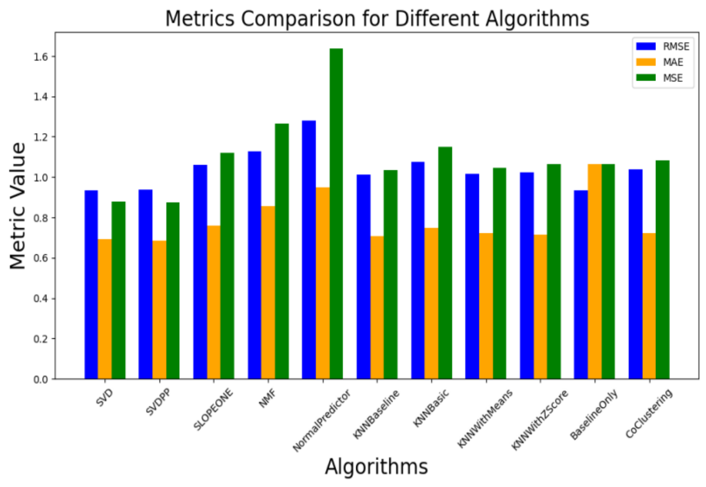

Figure 1. Metrics comparison for different algorithms.

ii. MAE: MAE is the average absolute difference between predicted and observed values. It provides a measure of the average error in prediction. Similar to RMSE, a smaller MAE indicates greater precision. In the table, the MAE values range from 0.6836 to 1.2814, indicating differences in the prediction accuracy of the algorithms. MAE is commonly used in applications where the absolute magnitude of errors is important, such as content curation, catalogue organization, and feedback analysis. It can provide a balanced evaluation of prediction accuracy.

iii. MSE: MSE is the average of the squared deviations between predicted and actual values. It magnifies larger errors than the MAE, making it more sensitive to extreme values. The range of MSE values in the table, from 0.7080 to 1.6363, reflects the variable ability of the algorithms to minimize squared errors. MSE can benefit applications where reducing large prediction errors is critical, such as targeted marketing and promotions. It places more weight on larger errors and can be used to optimize marketing efforts.

Overall, the provided table allows for comparing algorithm performance based on RMSE, MAE, and MSE metrics. However, it is important to consider other factors and conduct further analysis to make informed decisions regarding the selection of an algorithm for a specific task or problem.

The provided **Table 3** presents information related to the performance of different algorithms in terms of fit time and test time. The algorithms listed include SVD, SVDPP, SLOPEONE, NMF, NormalPredictor, KNNBaseline, KNNBasic, KNNWithMeans, KNNWithZScore, BaselineOnly, and CoClustering. The table provides mean and standard deviation values for each algorithm's fit time and test time. The **Figures 2** and **3** represents the mean deviation of fit time and test time for different algorithms.

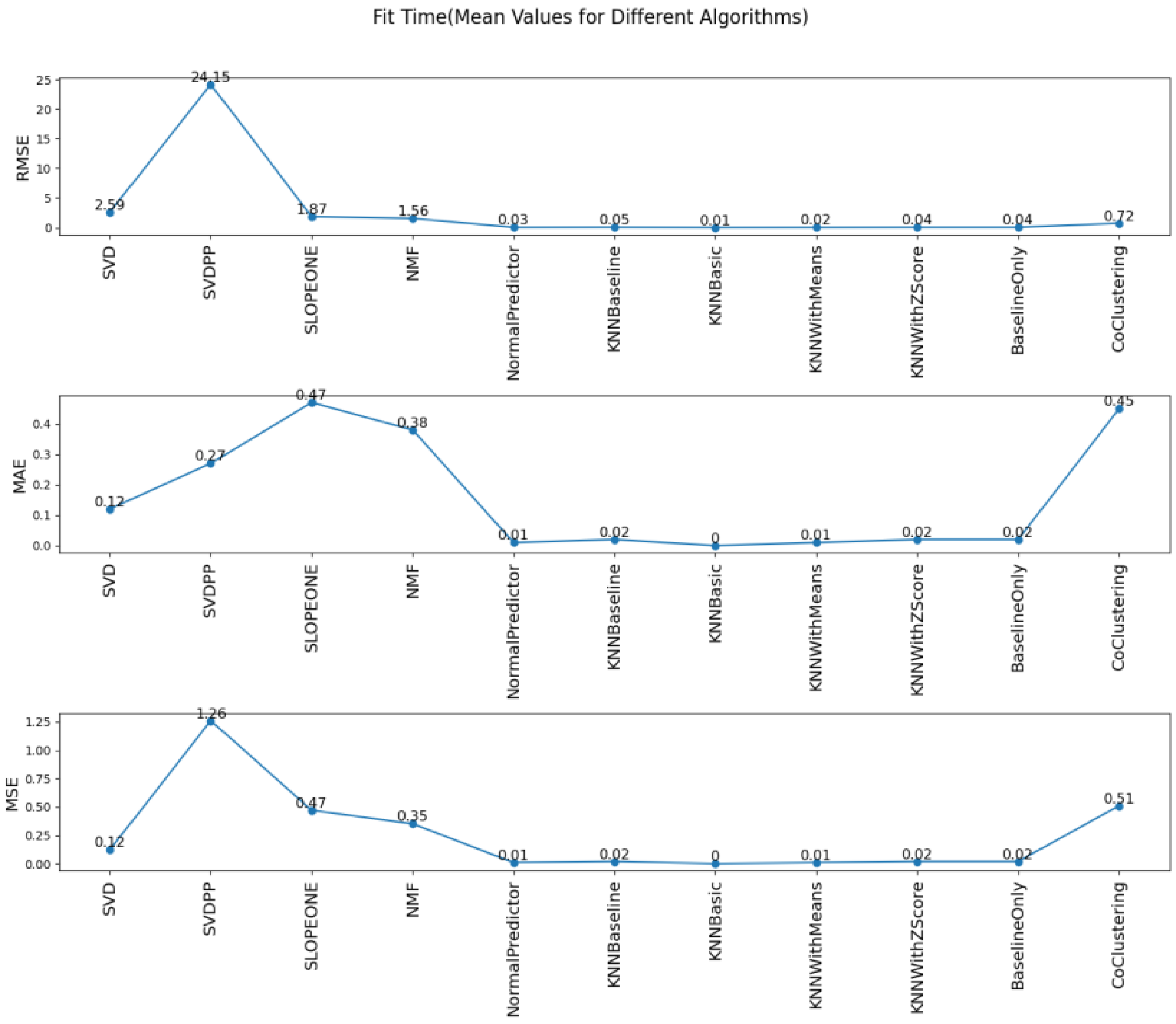

Figure 2. Mean deviation of fit time for different algorithms.

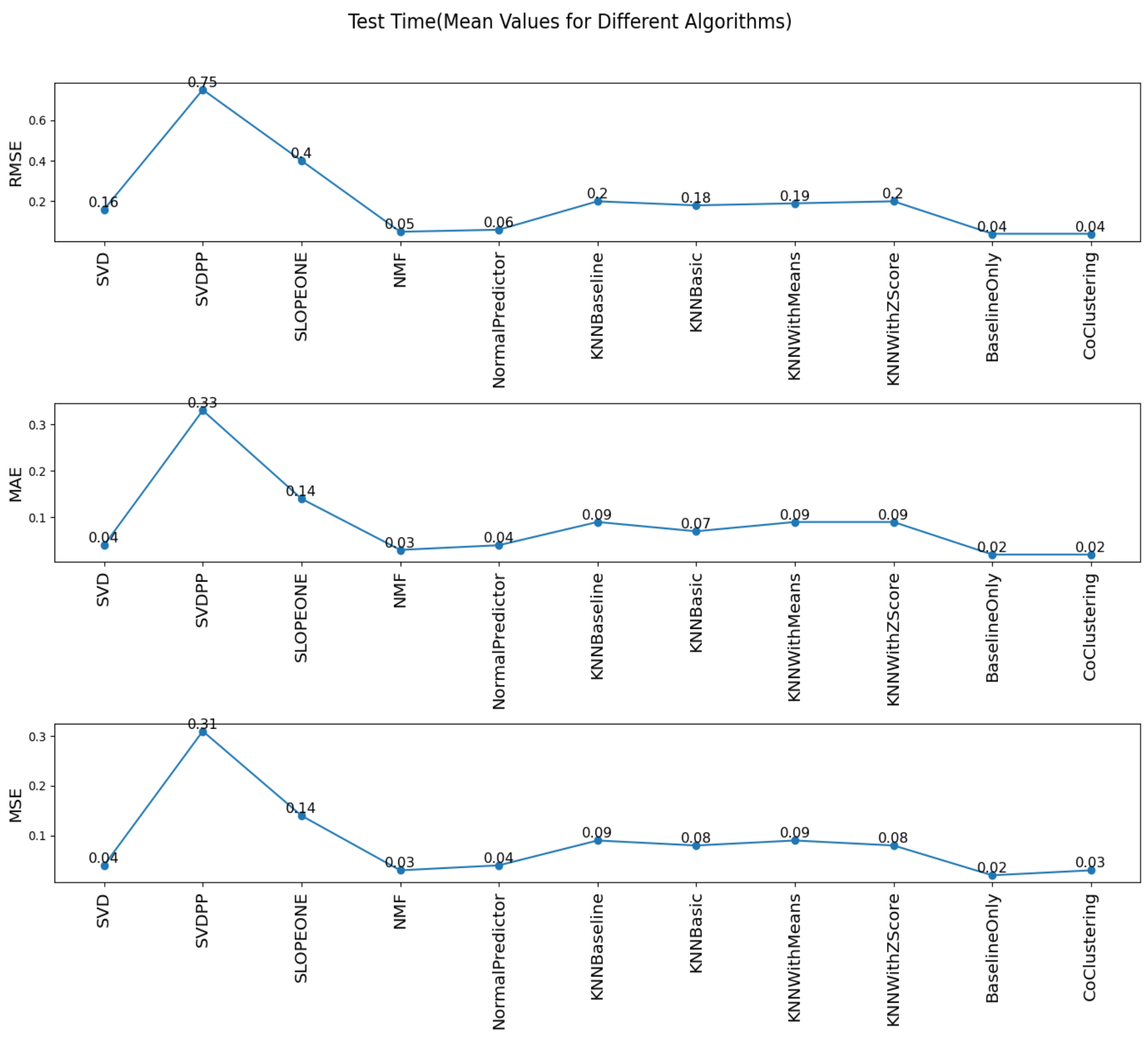

Figure 3. Mean deviation of test time for different algorithms.

Table 3. Mean and standard deviation of fit time and test time for different algorithms.

Algorithms	**Train & inference time**	**RMSE**		**MAE**		**MSE**	
		Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
SVD	Fit time	2.59	0.02	0.12	0	0.12	0
	Test time	0.16	0.02	0.04	0	0.04	0
SVDPP	Fit time	24.15	8.74	0.27	0.03	1.26	0.01
	Test time	0.75	0.31	0.33	0.01	0.31	0
SLOPEONE	Fit time	1.87	0.32	0.47	0.02	0.47	0.02
	Test time	0.4	0.06	0.14	0.02	0.14	0.01
NMF	Fit time	1.56	0.22	0.38	0.02	0.35	0.02
	Test time	0.05	0	0.03	0	0.03	0
NormalPredictor	Fit time	0.03	0	0.01	0	0.01	0
	Test time	0.06	0	0.04	2	0.04	2
KNNBaseline	Fit time	0.05	0	0.02	0	0.02	0
	Test time	0.2	0.02	0.09	0	0.09	0
KNNBasic	Fit time	0.01	0	0	0	0	0
	Test time	0.18	0.03	0.07	0	0.08	0.02

Table 3. (*Continued*).

Algorithms	**Train & inference time**	**RMSE**		**MAE**		**MSE**	
		Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
KNNWithMeans	Fit time	0.02	0	0.01	0	0.01	0
	Test time	0.19	0.03	0.09	0.02	0.09	0.02
KNNWithZScore	Fit time	0.04	0	0.02	0	0.02	0
	Test time	0.2	0.03	0.09	0.02	0.08	0
BaselineOnly	Fit time	0.04	0.01	0.02	0	0.02	0
	Test time	0.04	0	0.02	0	0.02	0
CoClustering	Fit time	0.72	0.02	0.45	0	0.51	0.02
	Test time	0.04	0	0.02	0	0.03	0

When evaluating a recommender system using metrics such as RMSE (root mean square error) and MAE (mean absolute error), etc., calculating the mean and standard deviation (std) values provides valuable insights into the performance and consistency of the system where mean represents the average value of the metric across the three folds in our case (or any number of folds in cross-validation). It provides a central tendency measure and indicates the overall performance of the recommender system. The mean value is particularly useful for comparing different algorithms or variations of the same algorithm to determine which one performs better on average across the folds. And the standard deviation measures the variability or spread of the metric values across the three folds. It quantifies how much the metric values deviate from the mean. A smaller standard deviation indicates more consistent performance across the folds, with less variation in the metric values. A higher standard deviation suggests greater variability in the performance, indicating that the system's performance may be more inconsistent across different subsets of the data. The standard deviation is valuable for understanding the stability and robustness of the recommender system. Lower standard deviation values imply more reliable and consistent predictions.

Fit time refers to the time taken by an algorithm to train or fit a model to the given data. Test time represents the time the algorithm takes to make predictions or perform computations on unseen or test data. For each algorithm, the table includes the mean and standard deviation values for fit time and test time. Regarding algorithm performance analysis, the fit time values range from 0.01 to 24.15, while the test time values range from 0 to 0.75. The standard deviation values provide information about the variability or consistency of each algorithm's fit and test time measurements. When comparing algorithm accuracy, the table does not directly provide accuracy metrics such as accuracy score or precision-recall values. Therefore, it might be necessary to consult additional resources or experiments to gather accuracy-related information for these algorithms. It's important to note that algorithm performance depends on the specific problem, dataset, and context in which the algorithms are being used. The fit time and test time values can provide insights into the computational efficiency of the algorithms, but other factors such as prediction accuracy, scalability, and suitability for the specific task should also be considered when comparing algorithm performance.

To analyze algorithm accuracy, it is recommended to refer to additional resources or studies that evaluate the performance of these algorithms in terms of accuracy metrics specific to the problem domain. These evaluations can involve measures like accuracy score, precision, recall, F1 score, or other domain-specific evaluation metrics. Overall, the provided table primarily focuses on the fit and test times of different algorithms and does not directly provide a comprehensive comparison of algorithm accuracy. Additional information and evaluations specific to accuracy metrics would be required to compare algorithm accuracy thoroughly.

5. Conclusion and future work

This study presents insights into the performance metrics of various algorithms used in a recommender system. This research examines algorithm metrics and computing efficiency to determine predicted accuracy. These insights can assist academics and practitioners in choosing recommender algorithms. The study emphasizes the importance of fit time, test time, and performance indicators when evaluating algorithms. This study impacts recommendation systems, where algorithm selection improves user satisfaction and engagement. Researchers and developers can improve recommender systems and deliver more accurate and personalized recommendations by understanding algorithm strengths and flaws. To fully comprehend this study, its limitations must be acknowledged. First, the research focused on one dataset or area. Therefore, the results may not apply to others. Validating the findings across contexts requires more studies with diverse datasets. The study only investigated a few algorithms, so others may be important to recommendation systems. This study's findings and limitations suggest various research possibilities. First, comparative research using a bigger set of algorithms can improve recommendation system performance comprehension. Second, examining evaluation data and user satisfaction and engagement might help select algorithms. Explore how dataset factors like sparsity and diversity affect algorithm performance to comprehend their real-world application better. Finally, researching unique methods or hybrid models that combine the capabilities of several algorithms may be interesting. In conclusion, this study summarises research findings regarding algorithm performance in a recommender system. The insights obtained from the analysis of fit time, test time, and performance metrics contribute to the understanding of algorithm selection and its implications in recommendation systems. The limitations of the study highlight the need for further research.

Author contributions

Conceptualization, AS and KS; methodology, KS; software, KS; validation, KS and AS; formal analysis, KS and AS; investigation, KS; resources, KS; data curation, KS; writing—original draft preparation, KS and AS; writing—review and editing, KS and AS; visualization, KS; supervision, AS; project administration, AS. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1. Koren Y, Bell R. Advances in collaborative filtering. In: Ricci F, Rokach L, Shapira B (editors). *Recommender Systems Handbook*. Springer; 2015. pp. 91–142.
2. Kulkarni A, Shivananda A, Kulkarni A, Krishnan VA. Collaborative filtering. In: *Applied Recommender Systems with Python*. Apress Berkeley; 2023. pp. 89–110.
3. Saini K, Singh A. Coherent algorithms of recommender systems in electronic commerce—A retrospection. *Neuro Quantology* 2022; 20(9): 318–330. doi: 10.14704/nq.2022.20.9.NQ440033
4. Isinkaye FO, Folajimi YO, Ojokoh BA. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal* 2015; 16(3): 261–273. doi: 10.1016/j.eij.2015.06.005
5. Jain S, Grover A, Thakur PS, Choudhary SK. Trends, problems and solutions of recommender system. In: Proceedings of the International Conference on Computing, Communication and Automation; 15–16 May 2015; Greater Noida, India. pp. 955–958.
6. Amin SA, Philips J, Tabrizi N. Current trends in collaborative filtering recommendation systems. In: Xia Y, Zhang LJ (editors). *Services—SERVICES 2019*, Proceedings of the SERVICES 2019: World Congress on Services; 8–13 July 2019; Milan, Italy. Springer; 2019. Volume 115117, pp. 46–60.
7. Saini K, Singh A. Hybrid Recommender System for E-Commerce: A Comprehensive Review and Future Direction. Journal of Harbin Engineering University 2023; 44(8): pp 801-809.
8. Ajaegbu C. An optimized item-based collaborative filtering algorithm. *Journal of Ambient Intelligence and Humanized Computing* 2021; 12: 10629–10636. doi: 10.1007/s12652-020-02876-1
9. Bin C. A collaborative filtering recommendation algorithm based on restricted random walk. In: Tuba M, Akashe S, Joshi A (editors). *ICT Systems and Sustainability*. Springer; 2022. pp. 763–773.
10. Ahmed E, Letta A. Book recommendation using collaborative filtering algorithm. *Applied Computational Intelligence and Soft Computing* 2023; 2023: 1514801. doi: 10.1155/2023/1514801
11. Mustafa N, Ibrahim AO, Ahmed A, Abdullah A. Collaborative filtering: Techniques and applications. Available

online: https://www.researchgate.net/publication/341216858_Collaborative_Filtering_Techniques_and_Applications (accessed on7 June 2023).

12. Schafer JB, Frankowski D, Herlocker J, Sen S. Collaborative filtering recommender systems. In: Brusilovsky P, Kobsa A, Nejdl W (editors). *The Adaptive Web*. Springer; 2007. pp. 291–324.
13. Andika HG, Hadinata MT, Huang W, et al. Systematic literature review: Comparison on collaborative filtering algorithms for recommendation systems. In: Proceedings of the 2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT); 3–5 November 2022; Solo, Indonesia. pp. 56–61.
14. Fkih F. Similarity measures for collaborative filtering-based recommender systems: Review and experimental comparison. *Journal of King Saud University—Computer and Information Sciences* 2022; 34(9): 7645–7669. doi: 10.1016/j.jksuci.2021.09.014
15. Snyder H. Literature review as a research methodology: An overview and guidelines. *Journal of Business Research* 2019; 104: 333–339. doi: 10.1016/j.jbusres.2019.07.039
16. Srifi M, Oussous A, Ait Lahcen A, Mouline S. Recommender systems based on collaborative filtering using review texts—A survey. *Information* 2020; 11(6): 317. doi: 10.3390/info11060317
17. Roy D, Dutta M. A systematic review and research perspective on recommender systems. *Journal of Big Data* 2022; 9(1). doi: 10.1186/s40537-022-00592-5
18. Park DH, Kim HK, Choi IY, Kim JK. A literature review and classification of recommender systems research. *Expert Systems with Applications* 2012; 39(11): 10059–10072. doi: 10.1016/j.eswa.2012.02.038
19. Lee J, Sun M, Lebanon G. A comparative study of collaborative filtering algorithms. In: Proceedings of the KDIR 2012—Proceedings of the International Conference on Knowledge Discovery and Information Retrieval; 4–7 October 2012; Barcelona, Spain. pp. 132–137.
20. Chen J, Zhao C, Uliji, Chen L. Collaborative filtering recommendation algorithm based on user correlation and evolutionary clustering. *Complex and Intelligent Systems* 2020; 6(1): 147–156. doi: 10.1007/S40747-019-00123-5/TABLES/5
21. Wang Y, Zhao X, Zhang Z, Zhang LY. A collaborative filtering algorithm based on item labels and Hellinger distance for sparse data. *Journal of Information Science* 2021; 48(6): 749–766. doi: 10.1177/0165551520979876
22. Qingmin Y, Xingyu C. Research on collaborative filtering recommendation algorithm. In: Proceedings of the 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA); 20–22 September 2019; Dalian, China. pp. 741–743.
23. Hug N. Surprise: A Python library for recommender systems. *Journal of Open Source Software* 2020; 5(52): 2174. doi: 10.21105/joss.02174
24. Stephen SC, Xie H, Rai S. Measures of similarity in memory-based collaborative filtering recommender system—A comparison. In: Proceedings of the 4th Multidisciplinary International Social Networks Conference; 17–19 July 2017; Bangkok, Thailand. pp. 1–8.
25. Wang N, Wang H, Jia Y, Yin Y. Explainable recommendation via multi-task learning in opinionated text data. In: Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval; 8–12 July 2018; Ann Arbor, MI, USA. pp. 165–174.
26. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. *Computer* 2009; 42(8): 30–37. doi: 10.1109/mc.2009.263
27. Hu Y, Koren Y, Volinsky C. Collaborative filtering for implicit feedback datasets. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining; 15–19 December 2008; Pisa, Italy. pp. 263–272.