Annals of Data Science (2024) 11(5):1705–1739
https://doi.org/10.1007/s40745-023-00474-4

Artificial Intelligence Algorithms for Collaborative Book Recommender Systems

Clemens Tegetmeier[1] · Arne Johannssen[1] · Nataliya Chukhrova[2]

Received: 24 November 2022 / Revised: 10 May 2023 / Accepted: 13 May 2023 /
Published online: 8 June 2023

Abstract

Book recommender systems provide personalized recommendations of books to users based on their previous searches or purchases. As online trading of books has become increasingly important in recent years, artificial intelligence (AI) algorithms are needed to recommend suitable books to users and encourage them to make purchasing decisions in the short and the long run. In this paper, we consider AI algorithms for so called collaborative book recommender systems, especially the matrix factorization algorithm using the stochastic gradient descent method and the book-based k-nearest-neighbor algorithm. We perform a comprehensive case study based on the Book-Crossing benchmark data set, and implement various variants of both AI algorithms to predict unknown book ratings and to recommend books to individual users based on the highest predicted ratings. This study aims to evaluate the quality of the implemented methods in recommending books by using selected evaluation metrics for AI algorithms.

Keywords Artificial intelligence · Book recommender systems · $k$nn algorithm · Machine learning · Matrix factorization algorithm · Stochastic gradient descent method

✉ Arne Johannssen
arne.johannssen@uni-hamburg.de

Clemens Tegetmeier
clemens.tegetmeier@t-online.de

Nataliya Chukhrova
nataliya.chukhrova@hcu-hamburg.de

[1] University of Hamburg, Hamburg, Germany

[2] HafenCity University, Hamburg, Germany

1 Introduction

Book recommender systems are often used by companies to present interesting and personalized book recommendations to their customers. The recommendations are supposed to convince the customer to buy books in the short run and to use the book recommender system for further purchases in the long run. As online trading of books has considerably increased in recent years [1], book recommender systems have become more important. Online booksellers such as Amazon, Barnes & Noble, Waterstones, and Thalia have played an important role in this development. For instance, current challenges for recommender systems are taking into account the user's context, e.g., time or mood [2–4], ensuring diversity [5] and including implicit ratings to a greater extent [6] when generating book recommendations. Generally, the research focus has turned to Artificial Intelligence (AI) algorithms, e.g., the number of papers considering deep learning techniques has increased significantly in recent years and is frequently applied to recommender systems [7–9]. Note that AI, like machine learning, deep learning, data mining, and Big Data analytics, is based on Data Science techniques, so these areas are closely related [10–12]. While AI refers to the development of intelligent techniques that can perform tasks that typically require human intelligence [13], Data Science is an interdisciplinary field that involves the extraction, processing, analysis, and interpretation of large and complex data sets. In particular, deep learning and various variants of neural networks offer a new way to address current challenges of recommender systems [14] and beyond [15–19]. However, when using these black box algorithms, the problem of missing explainability of how the recommendations are generated needs to be considered [20–22].

In book recommender systems, AI algorithms have the task of suggesting books that buyers are potentially interested in and that have not been read by them. Depending on how the AI algorithms are supposed to recommend books, a distinction is made between *collaborative*, *content-based*, and *hybrid* book recommender systems. In a collaborative book recommender system, AI algorithms access all book ratings that have been submitted by users of the book recommender system. Based on the submitted book ratings, the AI algorithms predict for each user the ratings for the books they have not yet rated. Then, the books with the highest predicted ratings can be recommended to each user [23]. Users mostly have rated a very small proportion of the books that exist in the data. Thus, AI algorithms have to predict more of the book ratings than are known. In this paper, we focus on AI algorithms in collaborative recommender systems.

Two popular AI algorithms in collaborative recommender systems are the *matrix factorization algorithm* using the stochastic gradient descent method and the *book-based k-nearest-neighbor (knn) algorithm* [24]. In this paper, both these algorithms are considered in the framework of the modified Book-Crossing data set. This data set from Cai–Nicolas Ziegler [25] is a kind of benchmark data basis for research on AI algorithms in collaborative recommender systems [26]. We investigate a subset consisting of 42,137 explicit ratings of the Book-Crossing data set that forms the data basis. The task of both AI algorithms is to predict the unknown book ratings of the modified Book-Crossing data set, and then to recommend the books with the highest predicted ratings to each user. By using different variants of both AI algorithms, this paper aims to evaluate both these algorithms in recommending books based on the modified Book-Crossing data set. For this aim, the quality of both AI algorithms is measured by selected evaluation metrics for AI algorithms.

This paper is organized as follows. In Sect. 2, after introducing the basics of collaborative book recommender systems, a short overview of AI algorithms and common evaluation metrics is given. Section 3 presents the book-based *k*nn-algorithm and the matrix factorization algorithm using the stochastic gradient descent method. In Sect. 4, we provide a comprehensive case study based on the Book-Crossing data set. In particular, we establish modifications to the data basis, present our methodology and proposed procedure, give the results of the study, and discuss them in detail. In the framework of the case study we show how the quality of both AI algorithms is measured using selected evaluation metrics. For this purpose, the statistical software `R` and the corresponding package `Recommenderlab`, which was developed for collaborative recommender systems [27], is used. Finally, Sect. 5 concludes the paper.

2 AI Algorithms in Collaborative Book Recommender Systems

2.1 Essentials

AI algorithms in collaborative book recommender systems are equivalent to AI algorithms that are generally used in entertainment recommender systems (e.g., movie recommender systems), where the entertainment products are referred to as items. The research on book recommender systems depends very much on the research flow on entertainment recommender systems, and most of the results are transferable to book recommender systems. AI algorithms need data about the book ratings by users, which can be explicit or implicit book ratings. A book rating is called an *explicit book rating* if a user actively assigns a rating on a specific scale (e.g., a scale from 1 to 10, where 10 represents the most positive experience and 1 is the most negative experience) to a book, see Table 1.

In contrast, an *implicit book rating* is not directly given by a user. Instead, the book ratings are predicted based on the user's behavior [28]. For example, a rating of 1 is assigned to a book if a user reads the complete book whereas the book gets a rating of 0 if a user only spends a short time with the book. In the following, however, we focus on explicit book ratings.

Table 1 Example of explicit book ratings by users (on a scale from 1 to 10)

	Book 1	Book 2	Book 3	Book 4	Book 5
User 1	9	–	–	1	–
User 2	–	–	–	6	4
User 3	10	10	–	1	–
User 4	–	10	9	–	9
User 5	–	10	9	–	–

A collaborative book recommender system includes data about p book ratings. There are n users $u_1, \ldots, u_n$ who have rated books, and m books $i_1, \ldots i_m$ that have received a rating by a user. The p book ratings are represented in an $n \times m$ *user-book matrix* B, see (2.1).

$$B = \begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ u_n \end{array} \begin{array}{c} \begin{array}{cccc} i_1 & i_2 & \cdots & i_m \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & & \ddots & \vdots \\ r_{n1} & \cdots & \cdots & r_{nm} \end{pmatrix} \end{array} \tag{2.1}$$

A user can rate a book only once. Every row shows the ratings by one user and every column represents the ratings for one book. Thus, every entry in the user-book matrix is a rating by one user for one book. Formally this means that the entry r_{ui} is the rating by user u for book i. Rating all m books is the maximum amount of ratings a user can give. A book can receive a maximum of n ratings meaning that every user has rated this book.

In this paper, we consider rating predictions for users who have already rated at least one book and for books that have already received at least one rating by a user. Therefore, the *cold start problem* [29] that deals with the question of how to predict a rating of a user or for a book without any knowledge about past ratings is not addressed. Most users have rated only a small percentage of the m books, which implies that B is a sparse matrix. The density

$$D_B = \left(\frac{p}{nm}\right) \cdot 100$$

of the user-book matrix measures the percentage of the known p book ratings in relation to the theoretically possible book ratings (i.e., nm ratings). It is important for the AI algorithms to be able to predict the large number of unknown ratings by a small number of known ratings.

It should be noted that each user has a different view on the question of which rating corresponds to a certain book quality. One user may argue that the rating 6 is a good rating on a scale from 1 to 10, whereas another user considers only ratings greater or equal to 9 as good ratings. The mean $\overline{r}_u$ represents the average book rating by a user. Based on the standard deviation $\sigma\ (r_u)$ of a user's book ratings, conclusions can be drawn whether a user has given similar book ratings (low standard deviation) or varying book ratings (high standard deviation). The mean of the ratings of a book $\overline{r}_i$ indicates how well users have rated the book on average. Additionally, the standard deviation $\sigma\ (r_i)$ describes the size of the rating range for a specific book. The mean of all p ratings of the user-book matrix is given by μ.

Based on the given book ratings, AI algorithms predict the unknown ratings of the user-book matrix B. This enables the AI algorithms to create an ordered list for each user with the N books that received the highest prediction. As a consequence, the books on the list are recommended to each user. How AI algorithms deal with different perceptions of the rating scale by users is explained in Sects. 3.1 and 3.2.

2.2 Memory-Based Versus Model-Based AI Algorithms

The AI algorithms in book recommender systems are distinguished between memory-based and model-based AI algorithms. *Memory-based AI algorithms* access the entire user-book matrix to recommend books to users [23, 30]. In contrast, *model-based AI algorithms* create a model from the user-book matrix. Based on this model, users get book recommendations. Within the memory-based AI algorithms, a distinction is made between user-based and book-based AI algorithms. They employ two different approaches to forecast the unknown ratings of the user-book matrix. User-based AI algorithms predict the missing ratings of every user based on similar user ratings. In contrast, book-based AI algorithms forecast the unknown ratings of every book by considering the ratings of similar rated books. Based on the forecasts of the unknown book ratings every user gets the books recommended that received the highest predicted ratings. The *k*nn-algorithm is a popular memory-based AI algorithm in collaborative book recommender systems [23, 31].

Model-based AI algorithms can be classified into the fields of regression, clustering, neural networks, deep learning and dimensionality reduction [31]. Model-based AI algorithms are mostly dimension-reducing algorithms, and matrix factorization algorithms are often applied in this context. The matrix factorization algorithm using the stochastic gradient descent method and the matrix factorization algorithm using the alternating least squares method are two popular matrix factorization algorithms [23, 24]. Additionally, research on neural networks and deep learning in collaborative recommender systems has increased significantly in recent years and can also be applied to collaborative book recommender systems [7–9].

2.3 Evaluation Metrics for AI Algorithms in Book Recommender Systems

AI algorithms in book recommender systems predict the unknown book ratings of the user-book matrix B. Based on the predictions, the AI algorithms suggest N books to every user as an ordered list. The quality of the AI algorithms depends on the grade of satisfaction of the users in relation to the proposed books. However, the satisfaction is hardly measurable in reality. Thus, the quality of the algorithms can only be approximated by online or offline tests [28, 32]. In the following, we focus on offline tests and the corresponding evaluation measures. The evaluation metrics can be divided into the fields of *prediction accuracy*, *classification accuracy* and *diversity* [23]. The values of the evaluation metrics depend on the characteristics of the considered data set (e.g., the range of the rating scale). Therefore, it is important to compare the quality of different AI algorithms using the same data set [28].

Split, *bootstrapping* and *cross-validation* are methods that can be used to evaluate the AI algorithms [27]. In particular, in a cross-validation, users are divided into a predetermined number of groups of equal size. The number of groups is equivalent to the number of iterations performed. In each iteration, one group is the test group and all other groups are considered to be the training groups. The test group is changed in every iteration, so that after all iterations each user was in the test group once. The users of the training groups are referred to as training users and the users of the test group

Table 2 Overview of the classification accuracy (confusion matrix)

	Recommended	Not recommended	All
Relevant books	m_{re} (TP)	m_{rn} (FN)	m_r
Irrelevant books	m_{ie} (FP)	m_{in} (TN)	m_i
All	m_e	m_n	m

are referred to as test users. In each iteration, a model is developed based on the given ratings of the training users. This model is then tested on the test users. In this process, some of the known ratings are employed to test the model and some of the known ratings of the test users are withheld to validate the model. Most offline evaluation metrics are measured by the predicted values for the withheld ratings. This is mostly done by taking the mean of the values regarding the offline assessment metrics in all iterations [27].

Prediction accuracy metrics measure how precisely an AI algorithm estimates the ratings. The larger the deviation of the predicted value from the true value is, the larger is the value of the metrics, and the worse the prediction accuracy of the AI algorithm. In the following, we consider three common prediction accuracy metrics, i.e., the Root Mean Square Error (RMSE), the Mean Square Error (MSE), and the Mean Absolute Error (MAE):

$$\text{RMSE} = \sqrt{\frac{\sum_{r_{ui} \in r_{\text{test}}} (\hat{r}_{ui} - r_{ui})^2}{p_{\text{test}}}}$$

$$\text{MSE} = \frac{\sum_{r_{ui} \in r_{\text{test}}} (\hat{r}_{ui} - r_{ui})^2}{p_{\text{test}}}$$

$$\text{MAE} = \frac{\sum_{r_{ui} \in r_{\text{test}}} |\hat{r}_{ui} - r_{ui}|}{p_{\text{test}}}$$

Here, r_{test} denotes the ratings of the test data set that need to be validated and p_{test} denotes their number. The real ratings are referred to as r_{ui}, whereas $\hat{r}_{ui}$ represents the predicted ratings of user u for book i.

Classification accuracy metrics determine whether a user receives book recommendations that are relevant to the user [23]. The larger the value of a classification accuracy metric, the larger the classification accuracy. *Precision* and *Recall* are the two most popular classification accuracy metrics and can be computed based on the confusion matrix given in Table 2:

$$\text{Precision} = \frac{m_{re}}{m_e}$$

$$\text{Recall} = \frac{m_{re}}{m_r}$$

Note that it holds $0 \leq \text{Precision}, \text{Recall} \leq 1$.

In Table 2, the following acronyms are used: True Positives (TP), False Negatives (FN), False Positives (FP), True Negatives (TN). Following Table 2, the recommended but irrelevant number of books m_{ie} corresponds to the type I error. Additionally, the number of books m_{rn} that is not recommended but relevant can be interpreted as type II error. Precision measures how many of the recommended books are relevant, whereas Recall specifies how many of the relevant books are recommended. Therefore, Precision minimizes the type I error, while Recall minimizes the type II error. Note that minimizing one error type increases the other error type in many cases. This leads to a trade-off between the maximization of Recall on the one hand and the maximization of Precision on the other hand [23].

Another important evaluation metric is *Diversity*. The main idea is that users do not appreciate to have the same books suggested over and over again. Diversity can be measured in different ways [5, 33]. One approach is to determine how many of the m books of the data set are recommended to the users. This ability is called *Coverage*. If there are many books that are not recommended to any user, this could mean a lack of diversification. Bobadilla et al. [34] defined a user's coverage as the proportion of books not rated by the user that have been rated by one of the user's nearest neighbors (a user's nearest neighbors are users who have rated books similarly to the considered user). Yang et al. [23] presented a different approach to measure Diversity that takes into account the similarity of books recommended to one user and the similarity of books recommended to two different users.

3 AI Algorithms

3.1 Book-Based *k*nn Algorithm

The $k$nn algorithm is a nonparametric algorithm [35]. In collaborative book recommender systems, it is used as a regression algorithm to estimate the missing values of the user-book matrix B. A distinction is made between the user-based and the book-based $k$nn-algorithm. We will mainly focus on the book-based $k$nn-algorithm in the following.

First, for every book, the similarity to all other books is measured by a similarity measure. Two books are considered to be similar, if users have given them a similar rating. The k nearest neighbors of a book are the k books that are most similar to the book. Every book has users who have not rated the book. Based on one user's ratings for the k nearest neighbors of the book the user's rating for the book can be predicted. Therefore, the user's ratings for the k nearest neighbors are weighted with the value of the corresponding similarity measure. For example, in Table 1, books 2 and 3 have received similar ratings by users 4 and 5. Book 2 has been highly rated by user 3. Therefore, the predicted rating of user 3 for book 3 could also be high. With this approach, the $k$nn algorithm tries to predict all unknown book ratings. After obtaining the predictions, the N books with the highest predicted ratings are suggested to each user [23].

The similarity of two books is measured by a similarity measure. A common similarity measure is the *Bravais-Pearson correlation coefficient* [36]

$$w^i_{gh} = \frac{\sum_{u\in U}\left(r_{ug}-\overline{r}_g\right)\cdot\left(r_{uh}-\overline{r}_h\right)}{\sqrt{\sum_{u\in U}\left(r_{ug}-\overline{r}_g\right)^2}\cdot\sqrt{\sum_{u\in U}\left(r_{uh}-\overline{r}_h\right)^2}} \tag{3.1}$$

with $-1 \le w^i_{gh} \le 1$, where w^i_{gh} indicates the similarity of the books i_g and i_h measured by the Bravais-Pearson correlation coefficient. Note that only the users who have rated both books (i.e., $u \in U$) are used for the calculation. Some books may generally have been rated higher than other books. This is taken into account by subtracting the respective mean. The mean of book i_g is given by $\overline{r}_g$ and the mean of book i_h is given by $\overline{r}_h$. Therefore, a user's rating counts as a positive rating only if it exceeds the mean of the book's ratings. The denominator contains the standard deviation of the ratings from the mean. A large standard deviation indicates that a book has received different ratings. In contrast, a low standard deviation means that a book has received mostly the same ratings. In this way the differences of the ratings from the mean considered in the numerator are scaled. Thus, the Bravais-Pearson correlation coefficient takes into account general rating differences between books. A value of "1" implies a high similarity between two books, whereas a value of "−1" means that two books have received opposite ratings by users and are therefore not similar. The *adjusted cosine similarity* and the *Euclidean distance* are also popular similarity measures but are not considered in this paper [37].

To improve the quality of the similarity measure, the number of users who have rated both books could be considered. This ensures that books are only counted as similar if they have been rated similarly by multiple users [23, 37]. This approach can be represented for the calculation of the similarity of two books as follows [23]:

$$w'^i_{gh} = \frac{2\cdot|U_g\cap U_h|}{|U_g|+|U_h|}\cdot w^i_{gh} \tag{3.2}$$

In (3.2), $|U_g|$ represents the number of users who have rated the book i_g, $|U_h|$ indicates how many users have rated the book i_h and $|U_g \cap U_h|$ corresponds to the number of users who have rated both books. The fraction gets smaller if fewer users have rated both books. Multiplying the fraction by w^i_{gh} ensures that very few common user ratings result in a lower similarity of two books [23].

After calculating the similarity measures for all books, the k nearest neighbors are determined for each book. The k nearest neighbors of a book are the books that have the highest similarity value [23]. The unknown ratings of users for a book are determined by weighting the ratings of these users at the k nearest neighbors of the book with the similarity measure [36, 37]:

$$P(ug) = \overline{r}_g + \sigma(r_g)\cdot\frac{\sum_{g_a\in N(g)}\left(\frac{r_{ug_a}-\overline{r}_{g_a}}{\sigma(r_{g_a})}\right)\cdot w^i_{gg_a}}{\sum_{g_a\in N(g)} w^i_{gg_a}} \tag{3.3}$$

Here, $P(ug)$ is the prediction for the rating of a user for book i_g. The k most similar books to book i_g are in the set $N(g)$. These books are denoted by $g_a, \ldots, g_k$. Only the books that have been rated by a given user are considered as nearest neighbors in the prediction. To account for differences in ratings between books, the ratings of the books are normalized. Additionally, the normalized ratings are weighted by the similarity measure $w^i_{gg_a}$.

The user's weighted normalized ratings for the books are transformed into the book's rating scale by multiplying the standard deviation $\sigma(r_g)$ of the book's ratings. The resulting value is added to the mean $\overline{r}_g$ of book i_g. If book i_g has a large standard deviation, the multiplication by the standard deviation ensures that a positive value should cause a greater deviation of the predicted value from the mean $\overline{r}_g$ of book i_g. This approach is known as *z-score*. Another common approach is given by the *deviation from the mean*. This approach does not take into account the standard deviation of the considered book and the book's nearest neighbors [36, 37]. In this way, the unknown ratings of users are estimated for each book. As explained later in this section, it may not be possible to predict all ratings. For each user, the books not yet rated by the user are sorted in a descending order according to the height of the predicted rating.

The book-based $k$nn algorithm can be classified as either a memory-based [23, 31] or a model-based AI algorithm [38]. It depends on whether each time a list of book recommendations is created for a user, the similarity measures are recalculated using the user-book matrix. If this is true, the book-based $k$nn algorithm is a memory-based AI algorithm. In contrast, the book-based $k$nn algorithm can be considered as a model-based algorithm if the similarity measures are recomputed only at regular intervals. The model is the similarity matrix that contains the similarity between the books. The classification of the book-based $k$nn algorithm as a model-based AI algorithm is supported by research results showing that the similarities between the books are stable over time [38].

The quality of the $k$nn algorithm depends on the choice of the similarity measure and the possible consideration of the number of common users of two books. Additionally, the choice of the number of k nearest neighbors plays an important role: choosing a small number of nearest neighbors could result in an overfitting to the ratings of the nearest neighbors [28]. Moreover, there is a risk that the nearest neighbors have not received any ratings by the user. This implies that it is not possible to predict a rating for the book. Coverage (see Sect. 2.3) is a measure to determine the extent of the problem [34, 36]. In contrast, choosing a large number of nearest neighbors could lead to the problem of underfitting [35]. User's ratings for books that are not similar enough to the book might influence the prediction too much. In extreme cases this could lead to unsatisfactory book recommendations. Therefore, it is often suggested to take a value in the range between 20 and 50 for the number of k nearest neighbors to solve the trade-off between overfitting and underfitting [36, 37].

3.2 Matrix Factorization Algorithm Using the Stochastic Gradient Descent Method

The matrix factorization algorithm using the stochastic gradient descent method is a model-based AI algorithm. The main assumption behind the matrix factorization is that the book ratings of users can be explained by j latent factors [24]. For every user, it is predicted how important the occurrence of a latent factor in a book is to the user. The prediction is based on the p known book ratings. Additionally, for every book, the known ratings are used to estimate the extent to which a latent factor occurs in the book [24].

The information about how important a latent factor is to a user is stored in the $n \times j$ dimensional *user-factor matrix* P. Additionally, the information about the occurrence of the latent factors in the books is stored in the $m \times j$ dimensional *book-factor matrix* Q. Every row p_u of P corresponds to one user of the user-book matrix B and every column of P represents a latent factor. Every column q_i of the book-factor matrix Q belongs to one book and every row represents one latent factor. The number of latent factors is much smaller than the number of books m and the number of users n. Thus, the storage in the two matrices P and Q leads to a strong reduction of the dimensions in which the information is available. Therefore, the matrix factorization algorithm is classified as a dimension-reducing algorithm (see Sect. 2.2). It can be described as an attempt to map the information of the user-book matrix by the information about the latent factors of the users and the books [24]. The matrix factorization algorithm estimates the user-book matrix B by the product of the user-factor matrix P with the transposed book-factor matrix Q:

$$\widehat{B} = PQ^T$$

The estimated user-book matrix $\widehat{B}$ is of the same dimension ($n \times m$) as the real user-book matrix B.

In reality, the meaning of the latent factors is not known and it is not known how many latent factors are necessary to map the information of the user-book matrix B. Additionally, it is not known which values the users and books have in the latent factors. In the following, for a better illustration, it is assumed that the meanings of the latent factors are known. It is explained how for a given number of j latent factors, the entries of P and Q can be determined using the stochastic gradient descent method.

Each entry of P expresses how important the user considers the occurrence of a latent factor in a book. A high value means that a user has a preference for the latent factor, whereas a low value indicates that the occurrence of the factor is irrelevant for the user. Each entry of Q indicates the extent to which a book contains a latent factor. A high value means that the latent factor occurs in the book. In contrast, a low value implies that the book does not contain the latent factor. To predict the book rating r_{ui}, the scalar product of the row p_u with the transposed row q_i is calculated [23]:

$$\widehat{r}_{u,i} = p_u q_i^T \tag{3.4}$$

The row p_u contains the values of the latent factors of the user u and the transposed row q_i includes the values of the latent factors of the book i. Thus, the user's preferences for the latent factors are each weighted by the extent of the corresponding latent factor in the book. A high book rating is predicted if the latent factors are rated highly by the user.

Example 3.1 Fantasy, thriller, humor and history are latent factors in books. Equation (3.5) shows that the user likes fantasy books. Additionally, it reveals that the user does not really care whether a book is a historical book or not. Since Harry Potter is a fantasy book, a high rating for this book is predicted for the first user. In contrast to the first user, the second user likes historical books and does not care whether a book is a fantasy book or not. Thus, a high rating is estimated for the Robinson Crusoe book.

$$\begin{aligned} \widehat{B} &= \begin{array}{c} \\ \text{user 1} \\ \text{user 2} \end{array} \begin{array}{cc} \text{Fantasy} & \text{Historic} \\ \left(\begin{array}{c} 5 \\ 2 \end{array} \right. & \left. \begin{array}{c} 1 \\ 4 \end{array} \right) \end{array} \cdot \begin{array}{c} \\ \text{Fantasy} \\ \text{Historic} \end{array} \begin{array}{cc} \text{Harry Potter} & \text{Robinson Crusoe} \\ \left(\begin{array}{c} 1 \\ 0 \end{array} \right. & \left. \begin{array}{c} 0.4 \\ 1 \end{array} \right) \end{array} \\ &= \begin{array}{c} \\ \text{user 1} \\ \text{user 2} \end{array} \begin{array}{cc} \text{Harry Potter} & \text{Robinson Crusoe} \\ \left(\begin{array}{c} 5 \\ 2 \end{array} \right. & \left. \begin{array}{c} 3 \\ 4.8 \end{array} \right) \end{array} \end{aligned} \tag{3.5}$$

In the following, it is described how to estimate the entries of P and Q using the stochastic gradient descent method. Therefore, a number j of latent factors is determined, and the stochastic gradient descent method is performed for different values of j. Thus, the optimal value of j can be determined by cross-validation. The estimation of the two matrices is based on the p known book ratings that are referred to as $r_{ui} \in K$. The estimation tries to achieve two different goals, which can be mapped into a loss function L that needs to be minimized:

$$L = \sum_{(r_{ui}) \in K} \left(r_{ui} - p_u q_i^T \right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2 \right) \tag{3.6}$$

The value of the loss function depends on the choice of P and the entries of Q. On the one hand, minimizing (3.6) implies choosing entries such that the known p book ratings in $\widehat{B}$ have as much as possible the same value as in B. The first part of (3.6) consists of the sum of these squared deviations. This prevents underfitting the known book valuations. On the other hand, to avoid overfitting, the known p ratings, the entries of the two matrices should not be chosen too large. Thus, in the second part of (3.6), high values for the latent factors are penalized by L2 regularization. The value of the parameter $\lambda > 0$ determines to which extent overfitting to the known valuations is penalized. Additionally, it decides the outcome of the trade-off between overfitting and underfitting. The optimization problem of the matrix factorization algorithm with (3.6) to be minimized can be represented as follows [24]:

$$\min_{q^*, p^*} \sum_{(r_{ui}) \in K} \left(r_{ui} - p_u q_i^T \right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2 \right) \tag{3.7}$$

Example 3.2 To better illustrate the trade-off between overfitting and underfitting, a user with the following preferences is considered. The user has given a good rating to many non-fiction books as well as to a fantasy book with the Harry Potter book. In this case, fantasy and science are the latent factors. This leads to the conclusion that the user prefers scientific books with a high probability, whereas a preference for fantasy books can only be inferred with a small probability. The first part of (3.6) prevents underfitting by ensuring that a high value is chosen for the latent factor science. Otherwise the predicted ratings for the non-fiction books would be too low. This would result in high deviations from the original ratings. Without the second part of (3.6) for the latent factor fantasy, a high value would have to be selected as well. The second part of L prevents overfitting to the known rating for Harry Potter by penalizing too high values for the latent factor fantasy.

The stochastic gradient descent is a method to estimate the entries of P and Q for a given number j of latent factors so that (3.6) is sufficiently minimized. Ng/Soo [39] stated that the stochastic gradient descent method can only find a solution that is close to optimality, but not necessarily an optimal solution. Before using the stochastic gradient descent method it is necessary to determine the regularization parameter λ, the learning rate γ, and the termination criteria. First, random values are chosen for the entries of P and Q. Then, one of the p known ratings is randomly selected to update the latent factors p_u of the corresponding user and the latent factors q_i of the corresponding book in multiple iterations. The procedure for each iteration can be described as follows: at the start of each iteration, it is calculated to what extent the estimation of the known book ratings based on the latent factors (see (3.4)) deviates from the true value of the known book rating. In the first iteration, these values are the randomly chosen values described above for the entries of P and Q. The deviation is denoted as e_{ui} and is defined as follows [23, 24]:

$$e_{ui} = r_{ui} - q_i^T p_u \tag{3.8}$$

The latent factors belonging to the book rating should be modified to minimize the part of the loss function belonging to the book rating. Thus, the negative gradient with the corresponding partial derivatives of the latent factors is required as the negative gradient points in the direction of the steepest descent of the part of (3.6) belonging to the book rating [24]:

$$\frac{\partial L_u}{\partial q_i} = 2(\lambda q_i - e_{ui} p_u) \tag{3.9}$$

$$\frac{\partial L_u}{\partial p_u} = 2(\lambda p_u - e_{ui} q_i) \tag{3.10}$$

The negative gradient indicates the direction in which the latent factors have to move to minimize the corresponding part of the loss function. The learning rate γ is initially set and determines how far the part of the loss function moves in the direction of the steepest descent, and thus, how much the latent factors change. On the one hand, the learning rate γ has to be large enough to find the minimum in a short time. On the other hand, it should be not too large to achieve convergence. To achieve convergence, a small value in the range of a few hundredths is mostly chosen for γ [40]. Thus, in each iteration, the latent factors of the user and the book are modified as shown in the following [24]:

$$q_i \leftarrow q_i + \gamma (e_{ui} p_u - \lambda q_i) \tag{3.11}$$

$$p_u \leftarrow p_u + \gamma (e_{ui} q_i - \lambda p_u) \tag{3.12}$$

Here, the value "2" from both partial derivatives (3.9) and (3.10) is considered in the value for γ.

Afterwards, the next iteration is performed. In this iteration, the error (3.8) and all following steps of this iteration are calculated with the modified values of q_i (see (3.11)) and p_u (see (3.12)). Several iterations are performed for the known book rating until a termination criterion is hit. A termination criterion can be a maximum number of iterations or an insufficient improvement of the corresponding part of the loss function. The stochastic gradient descent method ends when this process has been performed for all known book ratings. After estimating P and Q, all entries of $\widehat{B}$ are computed using (3.4). Out of the books not yet rated by the user, the user gets suggested the N books that have the highest predicted ratings.

The presented estimation of the ratings by multiplying the latent factors of the book by the latent factors of the user (see (3.4)) does not take into account the different perceptions of the rating scale by the users in Sect. 2.1 [24]. Thus, when estimating B, a common approach is to also include the bias of a book b_i, the bias of a user b_u and the average book rating of all users μ. The bias b_i indicates the deviation of the mean of the ratings of the book i from μ, and b_u indicates the deviation of the mean of the ratings of the user u from μ [23, 24].

4 Applying AI Algorithms to the Book-Crossing Data Set

4.1 The Book-Crossing Data Set

Cai-Nicolas Ziegler collected the Book-Crossing data set in August and September 2004 by capturing anonymized data from the Book-Crossing Community within four weeks [41], see the Book-Crossing website https://www.bookcrossing.com/about for a detailed description of the organization. The data set is available at http://www2.informatik.uni-freiburg.de/~cziegler/BX/. It consists of three tables, BX-Users, BX-Books, and BX-Book-Ratings. The table BX-Users contains anonymized demographic information about the 278,858 users. It has three features, *User-ID*, *location*, and *age*. User-ID as well as age are coded as integer and location is coded as string. Further information about the 271,379 books are provided in the BX-Books table. The books have a unique ISBN (coded as string) by which they are identified. Additional features provided for every book are the book's title, the book's first author, the name of the publisher (all coded as string), and the publication year (coded as integer). The BX-Book-Ratings table (see Table 3 for exemplary entries) consists of 1,149,780 ratings for 271,379 books by 278,858 users, the ratings (coded as integer) can be implicit or explicit. Every implicit rating has the value 0 because of the assumption that a missing rating implies a bad rating. The explicit book ratings are on a scale from 1 to 10,

Table 3 First five entries of the original BX-Book-Ratings data set (the book title was added for illustrative purposes)

User-ID	ISBN	Book title	Rating
276725	034545104X	*Flesh Tones: A Novel*	0
276726	0155061224	*Rites of Passage*	5
276727	0446520802	*The Notebook*	0
276729	052165615X	*Help!: Level 1*	3
276729	0521795028	*The Amsterdam Connection: Level 4*	6

Table 4 Descriptive statistics related to explicit ratings of the Book-Crossing data set

Minimum	1st Quartile	Median	Mean	3rd Quartile	Maximum	NA's
1.000	7.000	8.000	7.601	9.000	10.000	4

Table 5 Overview of the frequencies of the explicit ratings at the initial data set BX-Books-Ratings as part of the Book-Crossing data set

1	2	3	4	5	6	7	8	9	10
1770	2759	5996	8904	50,974	36,924	76,456	103,734	67,540	78,610

with 1 being the worst and 10 being the best possible rating. The 271,379 books are represented by their ISBN and the 278,858 users by their User-ID [25].

AI algorithms need data about the book ratings by users to predict a book rating. Thus, only the BX-Book-Ratings data subset containing this required information is considered in the following. The BX-Book-Ratings data set consists of 716,109 implicit and 433,671 explicit book ratings. Below, we focus on the 433,671 explicit book ratings. As for descriptive statistics on the explicit book ratings see Tables 4 and 5.

Note that many users have rated a small number of books, and many books have received a small number of ratings by users. As a consequence, the BX-Book-Ratings data set was reduced by only considering users who have rated at least 10 books and books that have received at least 10 ratings. The remaining data set consists of 42,137 explicit ratings for 2065 books by 1842 users. The corresponding user-book matrix has 3,803,730 entries and a density of around 1.11%. Most of the books (around 50%) have received 10–15 ratings, whereas most of the users (around 50%) have rated 10 to 16 books. The largest number of given ratings by a user is 964 and the book with the most ratings has got 225 ratings (see Figs. 5 and 6 in Appendix A). The mean of the ratings is 7.995 and the median is equal to 8 (see Fig. 1).

The median of the individual user's and book's rating is in the range of 7.5–8 (see Figs. 7 and 8 in Appendix A). Additionally, the mean of the individual's user's and book's rating is in the range of 7.5-—8.5 (see Figs. 9 and 10 in Appendix A). The most popular rating is 8. Only 25% of the users have given an explicit rating equal or below the value of 7 (see Fig. 2).

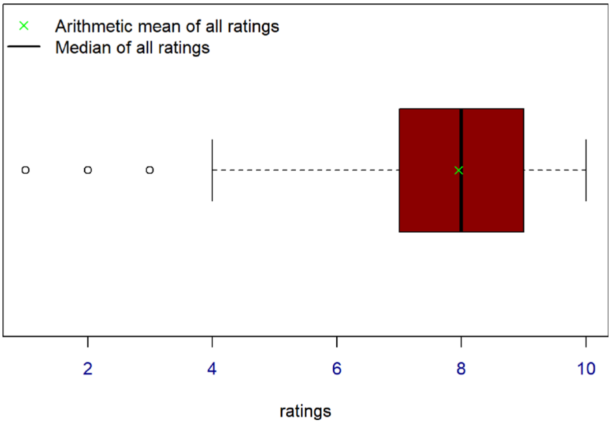

Fig. 1 Boxplot of the modified Book-Crossing data set

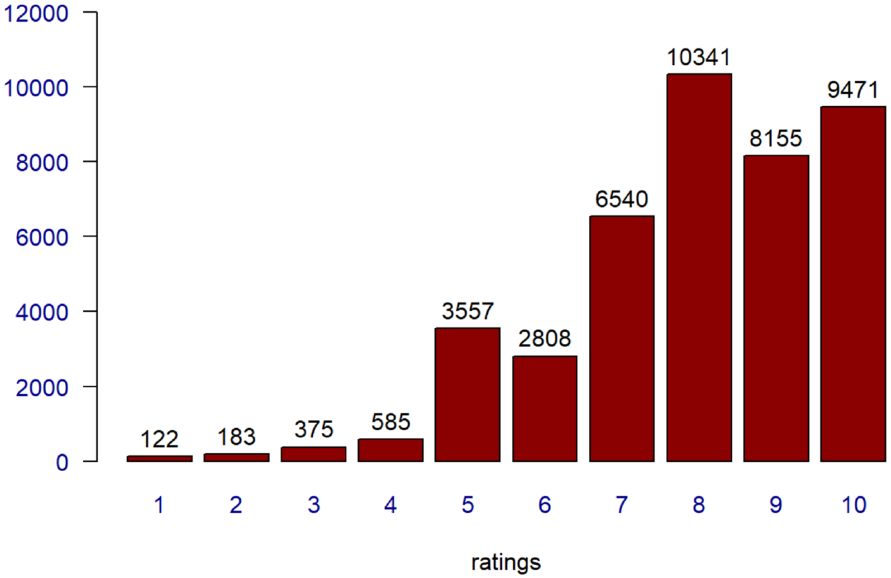

Fig. 2 Frequencies of the ratings of the modified Book-Crossing data set

The standard deviation of the ratings is around 1.74. Considering the standard deviation from the user's and book's perspective results in both cases in a value in the range between 1.5 and 2 (see Figs. 11 and 12 in Appendix A). This implies that a user has mostly given similar ratings to different books and that a book has mostly received similar ratings by different users (see Sect. 2.1).

Remark 4.1 The Book-Crossing data set is frequently used in the research field of book recommender systems as well as in the research about recommender systems in general. For instance, Ziegler et al. [25] as the originator used a subset of the data set containing 10,339 users, 6708 books and 361,349 ratings as one main basis to examine the impact of diversity in book recommendations on user's satisfaction with the recommended books. Adamopoulos/Tuzhilin [42] considered different subsets of the data set and proposed unexpectedness as an additional opportunity to improve the recommendation quality. Park/Tuzhilin [43] presented an approach for solving the *long-tail* problem (e.g., how to deal with books with few ratings) of recommender systems. Deldjoo et al. [44] used the data set as part of their study which examined the influence of data characteristics on the accuracy and fairness (e.g., measuring to what extent the quality of the recommendation depends on being in a specific group as age) of recommender systems.

4.2 Procedure and Methodology

In this section, the methodology and the procedure to analyze the quality of the book-based *k*nn-algorithm and the matrix factorization algorithm using the stochastic gradient descent method to recommend books from the modified Book-Crossing data set is presented.

4.2.1 Procedure

On the one hand, the quality of 31 variants of book-based *k*nn-algorithms, in which the number of *k* nearest neighbors is varied from 20 to 50, is measured. On the other hand, the quality of 11 variants of the matrix factorization algorithm using the stochastic gradient descent method, in which the number of latent factors is varied from 5 to 15, is measured. In order to measure the quality of the variants of both algorithms, the values of the prediction accuracy metrics RMSE, MSE, and MAE and of the classification metrics Precision and Recall (see Sect. 2.3) are considered. As an additional check on the quality of the variants, they are compared with the values of the evaluation metrics in a "random" algorithm (recommends books randomly) and a "popular" algorithm (recommends frequently rated books). The values of the evaluation metrics are determined using the `R`-package `Recommenderlab` that was developed by Michael Hahsler to test and evaluate collaborative recommender systems [27].

To compare the variants of both AI algorithms, they need to have the same training and test data set. Additionally, more than one training data set and one test data set should be used for the evaluation. This may reduce the risk that the division into a training and a test data set would affect the quality of the algorithms. To satisfy these important requirements for quality comparability, the option of the `Recommenderlab` package to determine an evaluation scheme is employed. Using the command `set.seed` ensures that the evaluation scheme is the same for all tested variants.

We apply cross-validation with 10 partitions and 10 iterations for each variant of both AI algorithms. Therefore, the 1842 users of the modified Book-Crossing data set were divided into 10 partitions consisting of about 184 users. In each iteration, users of 9 partitions form the training data set and develop a model. This model is tested using the test data set, which consists of the users of one partition. Thus, each user is nine times in the training data set and once in the test data set (see Table 6).

Table 6 Overview of the settings at the rating scheme

Method	Percentage of training users	Iterations	`Given`	`GoodRating`
Cross-validation	90%	10	9	9

4.2.2 Evaluation Metrics

To be able to measure the evaluation metrics, the rating scheme uses the option `Given` to specify how many of a test user's known ratings should be used for testing and how many should be used for validation. The value "9" is set for `Given`. Thus, from each test user, 9 of the known ratings are utilized to test the model developed by the training data set. Based on the estimation of the remaining known ratings of the test users, the algorithms are validated. Since each user has submitted at least 10 ratings in the modified Book-Crossing data set, at least one rating is used for validation for every test user. The prediction accuracy metrics RMSE, MSE, and MAE are measured by the known ratings used for validation. After 10 iterations, the prediction accuracy metrics are determined as the mean of their values from these iterations.

To measure the classification accuracy metrics Precision and Recall, the value for `GoodRating` was decisive for the rating scheme. The value of `GoodRating` indicates the rating from which on a book belonging to a validating rating is so relevant for a test user that it should be recommended to the test user. This is a hypothetical assumption since, in reality, the user has already rated the book. The book ratings of the modified Book-Crossing data set are on a scale of 1 to 10, where a value of "9" is chosen for `GoodRating`. This choice is based on the assumption that the user would like to receive a recommendation for a book that the user has rated 9 or 10. Additionally, this assumes that the user would not know the book yet. The number of ratings to validate with a rating of 9 or 10 is determined for each test user. Then, for each test user, the ratings for the 2056 remaining books are predicted, since 9 of the known ratings are used to test the model. Of the 2056 books, each test user was recommended once the 10 and once the 20 books with the highest predicted ratings. The main assumption for choosing the two list sizes is that a user would mostly only look at the recommendations placed at the top of the list.

For each test user, Precision is measured as the proportion of books in the list from recommended books that were previously determined to be relevant books to validate. Recall is measured as the proportion of the relevant books to be validated. After 10 iterations, the values for Recall and Precision are taken as the mean of the results from the 10 iterations.

4.2.3 Book-Based *k*nn Algorithm

Previous research on collaborative recommender systems (as discussed in Sect. 3.1) considers a number of 20–50 k nearest neighbors as optimum. Based on this suggestion, 31 variants with values for k from $k = 20$ to $k = 50$ of the book-based *k*nn algorithm are tested.

Table 7 Overview of the variants of the book-based *k*nn-algorithm

na.as.zero	k	Normalize	Method	normalize_sim_matrix	alpha
FALSE	20–50	z-score	Pearson	FALSE	0.5

Table 8 Overview of the variants of the matrix factorization algorithm using the stochastic gradient descent method

k	γ	λ	min_epochs	max_epochs	min_improvement	Normalize
5–15	0.001	0.015	50	200	0.000001	z-score

The Bravais-Pearson correlation coefficient (3.1) is chosen as similarity measure. For normalization, the z-score approach (3.3) is used. The unknown book ratings are not set to 0 (option na.as.zero) because the similarity of two books in the Bravais-Pearson correlation coefficient is only based on the users who rated both books. The meaning of alpha is not defined in the Recommenderlab package and related instructions, so the value was left at the default value of 0.5 (note that pre-tests showed no change in the scoring metrics at different values for alpha). The option to normalize the similarity matrix of the books is not set, as general differences in the ratings are already taken into account when calculating the similarities of the books (see Sect. 3.1). For an overview of the settings see Table 7.

4.2.4 Matrix Factorization Algorithm Using the Stochastic Gradient Descent Method

For the variants of the matrix factorization algorithm using the stochastic gradient descent method, the number of latent factors is varied from 5 to 15 latent factors. Funk [40], as the founder of the method, stated in his blog entry 25 and 40 as values for a reasonable number of latent factors for the Netflix data set, where the user-item matrix has a size of 8.5 billion entries. Koren et al. [24] mentioned a number of 20 to 100 latent factors for the same data set. Since the modified Book-Crossing data set has approximately 3.8 million entries, values between 5 and 15 are chosen for the number of latent factors.

For normalization, the z-score is used, as for the variants of the book-based $k$nn-algorithm. The z-score is chosen, since the common approach of considering the bias b_u, the bias b_i, and the mean of all known ratings μ in the prediction of the ratings could not be selected, see Sect. 3.2. The other parameters (see Table 8) are left at the default values [27, 40].

4.3 Results

In this section, the results from all variants of both AI algorithms are presented for the prediction accuracy metrics RMSE, MSE, and MAE as well as the classification accuracy metrics Precision and Recall. In order to additionally check the quality of these AI algorithms critically, they are also compared with the results of the two control

Table 9 Comparison of the values of the best variants of both AI algorithms for RMSE, MSE, and MAE with the values of the control algorithms

	RMSE	MSE	MAE
Random_1	2.092	4.383	1.563
Popular_1	1.562	2.443	1.178
SVDF_15	1.555	2.420	1.148
kNN_24	2.040	4.180	1.396
kNN_50	2.036	4.157	1.407

algorithms "popular" and "random". The results are rounded to the third decimal place. The values of the evaluation metrics for all variants of both AI algorithms (see Tables 12–16) and the two control algorithms "popular" and "random" (see Tables 17–18) can be found in Appendix B.

For the matrix factorization algorithm, the variant with 15 latent factors received the lowest values (RMSE = 1.555, MSE = 2.420, MAE = 1.148, see Table 9) and the variant with 5 latent factors the highest values (RMSE = 1.560, MSE = 2.437, MAE = 1.159, see Table 12) for all three predictive accuracy metrics. In each case, the increase in a latent factor slightly improved the predictive accuracy metrics. Therefore, the difference between the worst and the best variant is about 0.005 for RMSE, about 0.017 for MSE, and about 0.011 for MAE. For the book-based *k*nn algorithm, the variant with 24 nearest neighbors received the best value for the prediction accuracy metric MAE with a value of 1.396. The variant with 50 nearest neighbors performed the best for the predictive accuracy metrics RMSE (2.036) and MSE (4.157) (see Table 9). Table 9 and Fig. 3 show the variants of both AI algorithms with the best values for RMSE, MSE and MAE and the values of the control algorithms.

The matrix factorization algorithm using the stochastic gradient descent method achieved for all variants lower values in the prediction accuracy metrics compared to all variants of the book-based *k*nn algorithm and than the control algorithms. All variants of the book-based *k*nn algorithm had lower scores on the three predictive accuracy metrics than the control algorithm " random". Compared to the control algorithm "popular", all variants of the book-based *k*nn algorithm had higher values. The minimum difference between the prediction accuracy metrics of both AI algorithms is 0.476 for RMSE, 1.72 for MSE, and 0.248 for MAE. The maximum difference is 0.506 for RMSE, 1.854 for MSE, and 0.262 for MAE (see Tables 12 and 14).

In the following, the results for Precision and Recall are discussed for both a list of 10 and 20 recommended books. As for the book-based *k*nn algorithm, for the list with 10 recommended books, the variant with 29 nearest neighbors achieved the best values for Precision (0.006) and Recall (0.017). In contrast, for the list of 20 recommended books, the variant with 22 nearest neighbors received the highest value for Precision (0.006), and the variant with 26 nearest neighbors received the highest value for Recall (0.024) (see Tables 10–11). For the matrix factorization algorithm using the stochastic gradient descent method, the variant with 5 latent factors obtained the highest values for Precision (0.016) and Recall (0.035) for the list of 10 recommended books. For the list of 20 recommended books, the variant with 5 latent factors at Precision (0.013) and the variant with 9 latent factors at Recall (0.060) performed best (see Tables 10–11).

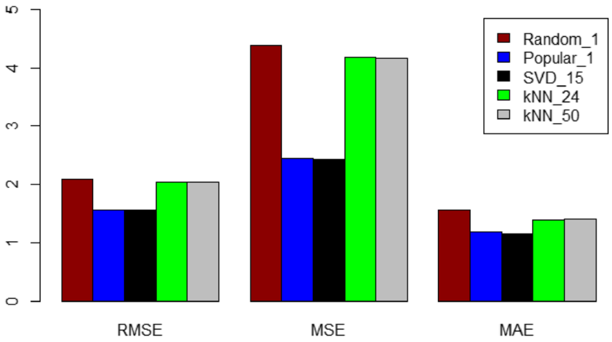

Fig. 3 Graphical comparison of the values of the best variants of both AI algorithms for RMSE, MSE, and MAE with the values of the control algorithms

Table 10 Comparison of the values of the best variants of both algorithms for Precision and Recall at the top 10 list with the values of the control algorithms

	TP	FP	FN	TN	*N*	Precision	Recall
Random_1	0.020	9.980	5.740	2040.260	2056	0.002	0.003
Popular_1	0.249	9.751	5.511	2040.489	2056	0.025	0.059
SVDF_5	0.158	9.842	5.603	2040.397	2056	0.016	0.035
kNN_29	0.061	9.751	5.699	2040.489	2056	0.006	0.017

Table 11 Comparison of the values for Precision and Recall of the best variants of both AI algorithms at the top 20 list with the values of the control algorithms

	TP	FP	FN	TN	*N*	Precision	Recall
Random_1	0.054	19.946	5.706	2030.294	2056	0.003	0.010
Popular_1	0.360	19.640	5.400	2030.600	2056	0.018	0.082
SVDF_5	0.264	19.736	5.496	2030.504	2056	0.013	0.057
SVDF_9	0.264	19.736	5.496	2030.504	2056	0.013	0.060
kNN_22	0.111	19.511	5.649	2030.729	2056	0.006	0.023
kNN_26	0.104	19.519	5.656	2030.720	2056	0.005	0.024

All variants of the matrix factorization algorithm using the stochastic gradient descent method had higher values for Precision and Recall than all variants of the book-based *k*nn algorithm. Additionally, all variants of both AI algorithms had higher values for Precision and Recall than the control algorithm random (see Tables 10–11).

For the list of 10 recommended books, the minimum difference between both AI algorithms for Precision is 0.007, while the minimum difference for Recall is 0.012. The maximum difference is 0.011 for Precision and 0.021 for Recall. For the list of 20 recommended books, the minimum difference between the AI algorithms is 0.006 for Precision and 0.029 for Recall. The maximum difference is 0.008 for Precision and 0.038 is Recall (see Tables 13, 15, 16).

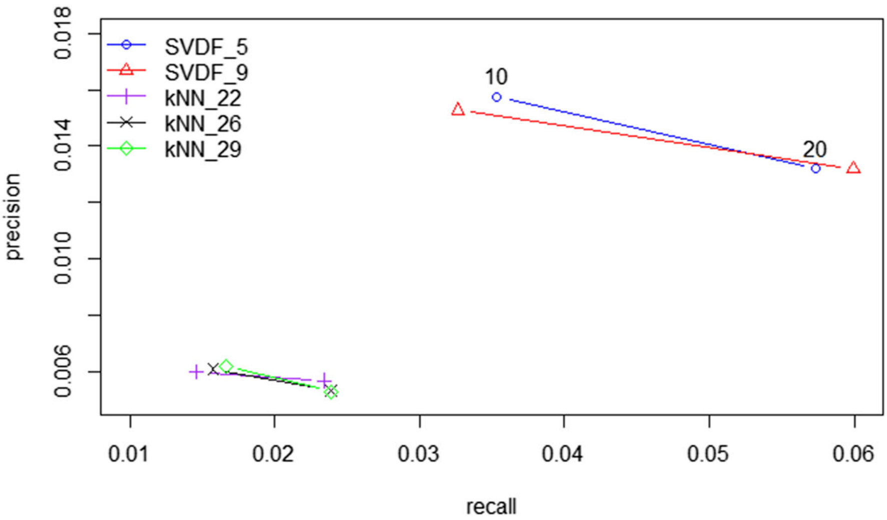

Fig. 4 Graphical comparison of the values of the best variants of both AI algorithms for Precision and Recall

For both AI algorithms, a higher value for Recall is observed for all variants for the list of 20 recommended books. The value for Precision is higher for all variants in the matrix factorization algorithm using the stochastic gradient descent method for the list with 10 recommended books. For the book-based *k*nn algorithm, this is true for most variants, although the difference is much smaller here. Figure 4 shows this tendency by looking at the variants of the AI algorithms that scored the highest for Recall or Precision.

4.4 Discussion

For the chosen settings of both AI algorithms, all variants of the matrix factorization algorithm using the stochastic gradient descent method show superior performance compared to all considered variants of the book-based *k*nn algorithm.

The matrix factorization algorithm using the stochastic gradient descent method led to better results for the prediction accuracy metrics compared to both control algorithms. As for Precision and Recall, the matrix factorization algorithm showed a better performance than the "random" algorithm and a worse performance than the "popular" algorithm. Thus, the quality of the matrix factorization algorithm applied to the modified Book-Crossing data set can be considered as good. The variants of the *k*nn algorithm led to better scores than the control algorithm "random" and worse scores than the control algorithm "popular" on the prediction accuracy and classification accuracy metrics. One reason for the poor performance of the book-based *k*nn algorithm could be a possibly low coverage of the books of the modified Book-Crossing data set [34]: the coverage of a book is the proportion of users who have not rated a book and at the same time have rated one of the *k* nearest neighbors of the book (see

Sect. 2.3). Low coverage means a high probability that if a user has not rated a book, the user has rated only a very small fraction of the k nearest neighbors, or in extreme cases, none of the k nearest neighbors of the book. In the first case, the problem of overfitting the predicted rating to the user's rating at the few k nearest neighbors may occur [28]. In the second case, no prediction can be made for the rating. The good results of the control algorithm "popular" on the classification accuracy metrics recall and precision and on the prediction accuracy metrics might be related to the fact that users might like books that have been rated by many users. Based on Fig. 4, where the best values of both AI algorithms for the lists of 10 and 20 recommended books are plotted, the trade-off between a high value for Recall and a high value for Precision described in Sect. 2.3 can be seen.

Moreover, the computation of Precision and Recall in the `Recommenderlab` package can be considered to be critical: when creating the list of recommended books for a test user, the 10 or 20 books with the highest predicted ratings were recommended. This involves predicting ratings for books with a rating to be validated and ratings for books where the true rating is unknown. Here, an AI algorithm lead to a high value for Recall if a large proportion of the books are recommended with a relevant rating to validate and a high value for Precision if a large proportion of the recommended books are books with a relevant rating to validate (see Sect. 4.2). Here, it is not possible to state with certainty, whether the user might find the recommended books with the predicted rating more interesting than the ratings to be validated of the books for which the user has given a rating of 9 or 10. The values for Recall and Precision were therefore calculated but should only be interpreted with caution.

5 Conclusions

In this paper, we investigated the performance of two popular AI algorithms for collaborative book recommender systems using the Book-Crossing benchmark data set. We implemented different variants of the book-based $k$nn algorithm and the matrix factorization algorithm using the stochastic gradient descent method based on selected prediction and classification accuracy metrics as well as using two control algorithms. These variants are characterized by variations in the number of k nearest neighbors in the book-based $k$nn algorithm and in the number of j latent factors in the matrix factorization algorithm using the stochastic gradient descent method. We performed a comprehensive case study to analyze the quality of both AI algorithms for collaborative book recommender systems to recommend books from the modified Book-Crossing data set.

For the investigated variants of both AI algorithms, the variants of the matrix factorization algorithm using the stochastic gradient descent method showed superior performance. In contrast, the book-based variants performed worse than the variants of the matrix factorization algorithm using the stochastic gradient descent method and than the control algorithm "popular". It seems that the poor performance of the book-based $k$nn algorithm might be related to the problem of poor coverage of the book-based $k$nn algorithm.

This paper considered users who have already rated books and books that have already received ratings. For AI algorithms, there is also the question of how to deal with new users who have not yet submitted ratings and new books that have not yet received ratings. This problem is known as the *cold start problem*. It deals with the question of which books are suggested to a new user and to which users a new book is suggested. Another interesting question is how AI algorithms deal with the *grey sheep problem*. This problem deals with users whose rating behavior is difficult to explain by any patterns, which makes it very difficult for AI algorithms to recommend suitable books to them.

In addition to these aspects, future research could focus on the performance of both algorithms when tested on other recent book data sets such as the Goodbooks-10k data set [45] and the Goodreads data set [46, 47], which are also frequently used in research about book recommender systems [6, 21].[1]

Acknowledgements The authors thank both anonymous reviewers for their valuable feedback and suggestions, which were important and helpful to improve the paper.

Author contributions Clemens Tegetmeier: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Visualization Arne Johannssen: Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Supervision, Project administration Nataliya Chukhrova: Validation, Formal analysis, Investigation, Writing - Review & Editing, Supervision.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availibility The data that support the findings of this study are available from the respective references as mentioned in the main text.

Code Availability The code is available from the authors upon request.

Declarations

Compliance with Ethical Standards This article does not contain any studies with human participants or animals performed by the authors.

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Disclosure of potential conflicts of interest (i) This manuscript is the authors' original work, which has not been published nor submitted simultaneously elsewhere; (ii) all authors have checked the manuscript and agreed to the submission, and (iii) there is no conflict of interest.

[1] The data sets are available at http://fastml.com/goodbooks-10k-a-new-dataset-for-book-recommendations/ and https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home?pli=1.

Appendix A

See Figs. 5, 6, 7, 8, 9, 10, 11 and 12.

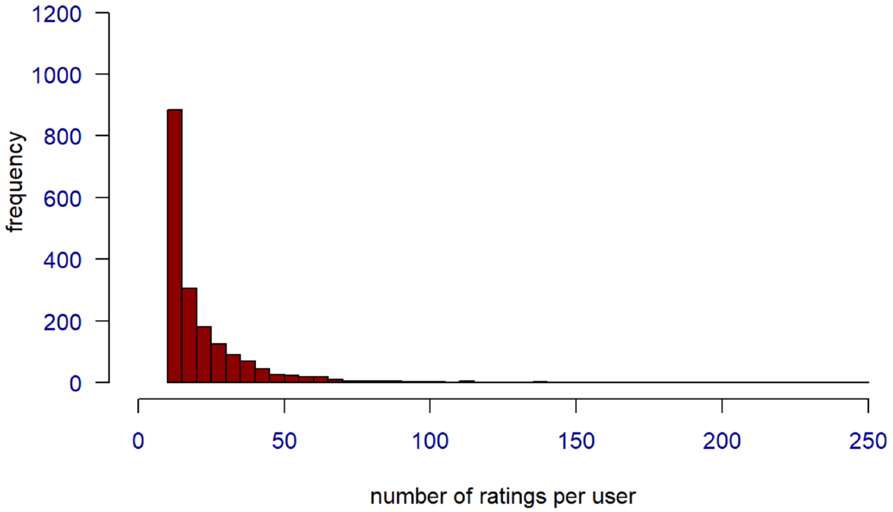

Fig. 5 Frequencies of the number of ratings per user in the modified Book-Crossing data set (the user with 964 submitted ratings is not included in this figure)

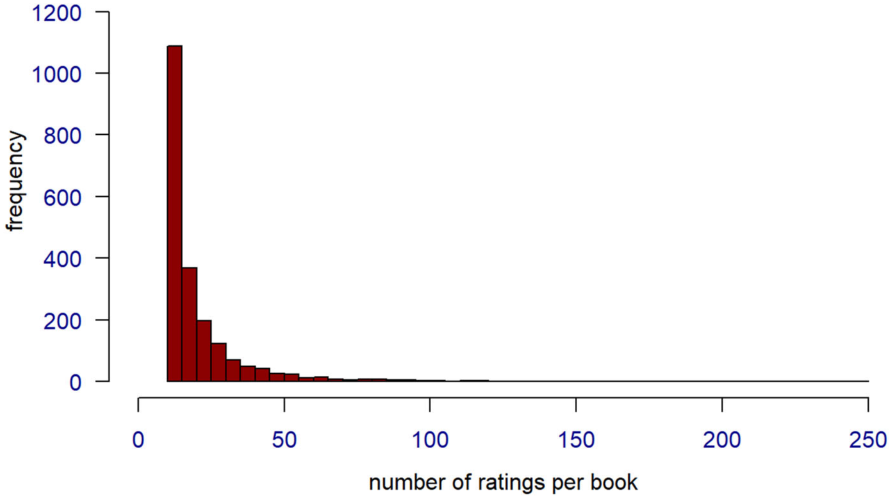

Fig. 6 Frequencies of the number of ratings per book in the modified Book-Crossing data set

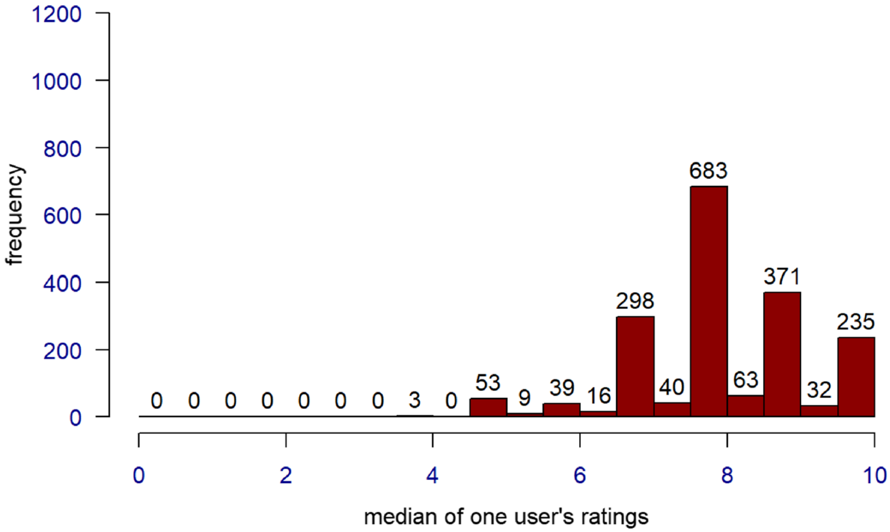

Fig. 7 Frequencies of the medians of one user’s ratings in the modified Book-Crossing data set

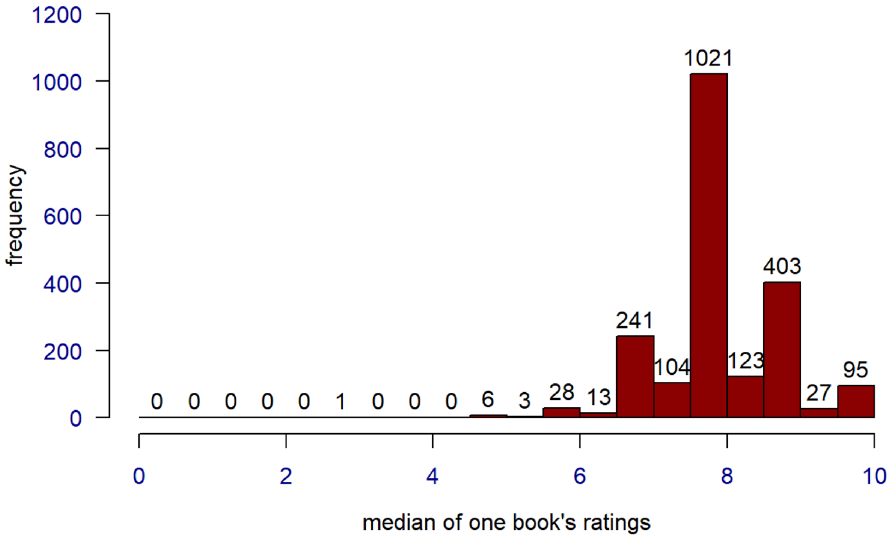

Fig. 8 Frequencies of the medians of one book’s ratings in the modified Book-Crossing data set

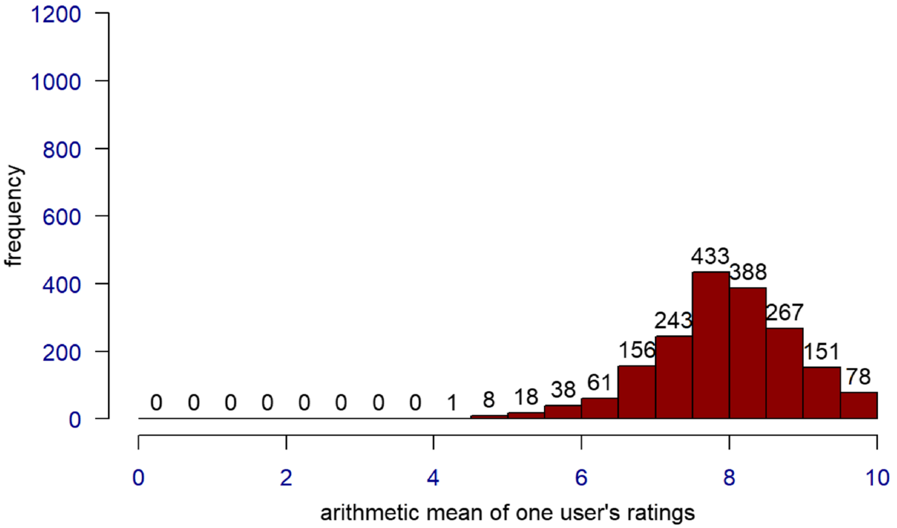

Fig. 9 Frequencies of the means of one user's ratings in the modified Book-Crossing data set

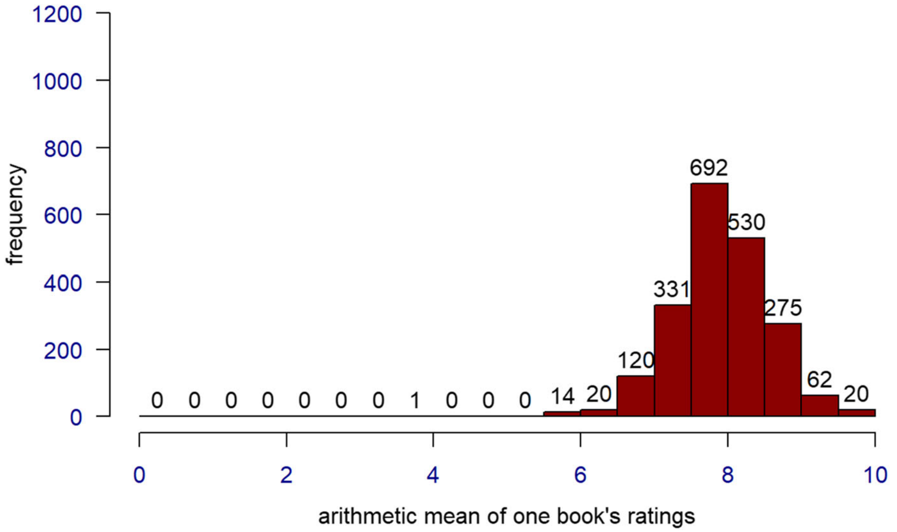

Fig. 10 Frequencies of the means of one book's ratings in the modified Book-Crossing data set

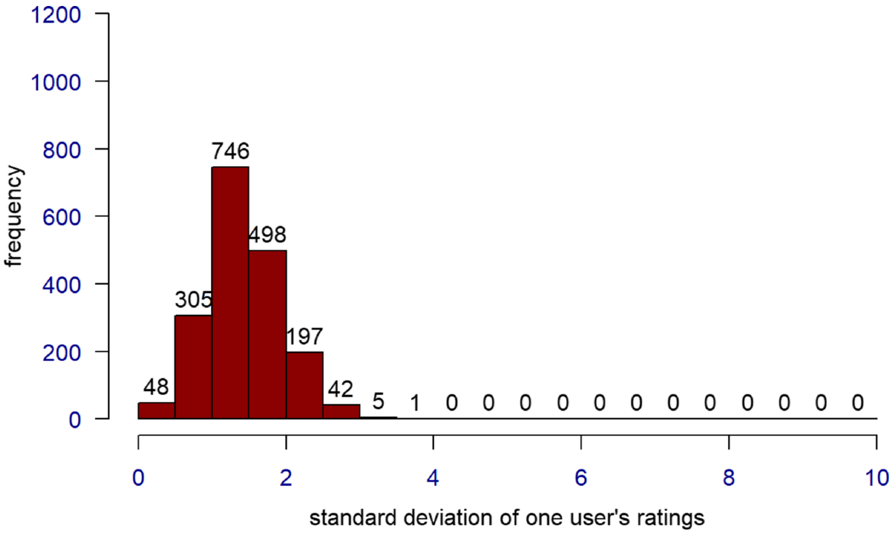

Fig. 11 Frequencies of the standard deviations of one user's ratings in the modified Book-Crossing data set

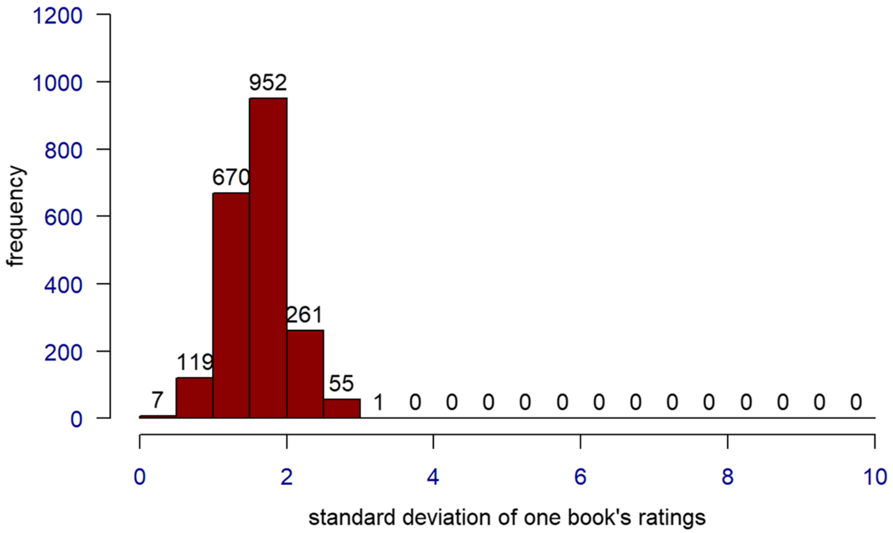

Fig. 12 Frequencies of the standard deviations of one book's ratings in the modified Book-Crossing data set

Appendix B

See Tables 12, 13, 14, 15, 16, 17 and 18.

Table 12 Values for RMSE, MSE, and MAE of all variants of the matrix factorization algorithm using the stochastic gradient descent method

	RMSE	MSE	MAE
SVDF_5	1.560	2.437	1.159
SVDF_6	1.559	2.435	1.157
SVDF_7	1.559	2.433	1.156
SVDF_8	1.558	2.431	1.155
SVDF_9	1.558	2.430	1.154
SVDF_10	1.557	2.427	1.153
SVDF_11	1.556	2.426	1.152
SVDF_12	1.556	2.425	1.151
SVDF_13	1.556	2.423	1.150
SVDF_14	1.555	2.421	1.149
SVDF_15	1.555	2.420	1.148

Table 13 Values for Precision and Recall of all variants of the matrix factorization algorithm using the stochastic gradient descent method

	TP	FP	FN	TN	*N*	Precision	Recall	*n*
SVDF_5	0.158	9.842	5.603	2040.397	2056	0.016	0.035	10
SVDF_5	0.264	19.736	5.496	2030.504	2056	0.013	0.057	20
SVDF_6	0.156	9.844	5.604	2040.396	2056	0.016	0.035	10
SVDF_6	0.262	19.738	5.498	2030.502	2056	0.013	0.057	20
SVDF_7	0.153	9.847	5.607	2040.393	2056	0.015	0.034	10
SVDF_7	0.261	19.739	5.499	2030.501	2056	0.013	0.057	20
SVDF_8	0.151	9.849	5.610	2040.390	2056	0.015	0.034	10
SVDF_8	0.263	19.737	5.497	2030.503	2056	0.013	0.059	20
SVDF_9	0.153	9.847	5.608	2040.392	2056	0.015	0.033	10
SVDF_9	0.264	19.736	5.496	2030.504	2056	0.013	0.060	20
SVDF_10	0.147	9.853	5.613	2040.387	2056	0.015	0.031	10
SVDF_10	0.261	19.739	5.499	2030.501	2056	0.013	0.059	20
SVDF_11	0.146	9.854	5.615	2040.385	2056	0.015	0.031	10
SVDF_11	0.259	19.741	5.502	2030.498	2056	0.013	0.059	20
SVDF_12	0.144	9.856	5.617	2040.383	2056	0.014	0.031	10
SVDF_12	0.246	19.754	5.515	2030.485	2056	0.012	0.054	20
SVDF_13	0.145	9.855	5.616	2040.384	2056	0.014	0.032	10
SVDF_13	0.251	19.749	5.509	2030.491	2056	0.013	0.056	20
SVDF_14	0.141	9.859	5.619	2040.381	2056	0.014	0.032	10
SVDF_14	0.246	19.754	5.514	2030.486	2056	0.012	0.055	20
SVDF_15	0.131	9.869	5.629	2040.371	2056	0.013	0.029	10
SVDF_15	0.240	19.760	5.520	2030.480	2056	0.012	0.053	20

Table 14 Values for RMSE, MSE, and MAE of all variants of the book-based *k*nn algorithm

	RMSE	MSE	MAE
kNN_20	2.061	4.274	1.407
kNN_21	2.052	4.237	1.402
kNN_22	2.051	4.226	1.404
kNN_23	2.051	4.228	1.402
kNN_24	2.040	4.180	1.396
kNN_25	2.040	4.181	1.397
kNN_26	2.043	4.189	1.403
kNN_27	2.056	4.242	1.410
kNN_28	2.047	4.205	1.403
kNN_29	2.038	4.168	1.398
kNN_30	2.044	4.188	1.404
kNN_31	2.044	4.188	1.404
kNN_32	2.048	4.207	1.407
kNN_33	2.049	4.209	1.405
kNN_34	2.048	4.207	1.404
kNN_35	2.051	4.216	1.406
kNN_36	2.053	4.227	1.408
kNN_37	2.049	4.209	1.407
kNN_38	2.047	4.199	1.406
kNN_39	2.045	4.190	1.405
kNN_40	2.041	4.176	1.404
kNN_41	2.040	4.171	1.403
kNN_42	2.045	4.193	1.405
kNN_43	2.044	4.190	1.407
kNN_44	2.043	4.184	1.408
kNN_45	2.041	4.177	1.406
kNN_46	2.040	4.171	1.406
kNN_47	2.042	4.181	1.409
kNN_48	2.043	4.183	1.410
kNN_49	2.040	4.174	1.409
kNN_50	2.036	4.157	1.407

Table 15 Values for Precision and Recall for the book-based *k*nn algorithm for 20–35 neighbors

	TP	FP	FN	TN	*N*	Precision	Recall	*n*
kNN_20	0.059	9.753	5.702	2040.487	2056	0.006	0.015	10
kNN_20	0.111	19.510	5.649	2030.730	2056	0.006	0.023	20
kNN_21	0.058	9.754	5.702	2040.486	2056	0.006	0.015	10
kNN_21	0.108	19.514	5.653	2030.726	2056	0.005	0.023	20
kNN_22	0.059	9.753	5.702	2040.487	2056	0.006	0.015	10
kNN_22	0.111	19.511	5.649	2030.729	2056	0.006	0.023	20
kNN_23	0.058	9.754	5.703	2040.485	2056	0.006	0.014	10
kNN_23	0.110	19.513	5.650	2030.727	2056	0.006	0.023	20
kNN_24	0.058	9.754	5.702	2040.486	2056	0.006	0.015	10
kNN_24	0.109	19.515	5.651	2030.725	2056	0.006	0.023	20
kNN_25	0.059	9.753	5.702	2040.487	2056	0.006	0.014	10
kNN_25	0.106	19.518	5.654	2030.722	2056	0.005	0.022	20
kNN_26	0.060	9.752	5.701	2040.488	2056	0.006	0.016	10
kNN_26	0.104	19.519	5.656	2030.720	2056	0.005	0.024	20
kNN_27	0.061	9.751	5.699	2040.489	2056	0.006	0.016	10
kNN_27	0.101	19.523	5.659	2030.717	2056	0.005	0.023	20
kNN_28	0.060	9.752	5.701	2040.488	2056	0.006	0.016	10
kNN_28	0.103	19.521	5.658	2030.719	2056	0.005	0.023	20
kNN_29	0.061	9.751	5.699	2040.489	2056	0.006	0.017	10
kNN_29	0.104	19.520	5.656	2030.720	2056	0.005	0.024	20
kNN_30	0.060	9.752	5.700	2040.488	2056	0.006	0.016	10
kNN_30	0.103	19.520	5.657	2030.719	2056	0.005	0.023	20
kNN_31	0.058	9.754	5.703	2040.485	2056	0.006	0.016	10
kNN_31	0.102	19.522	5.659	2030.718	2056	0.005	0.023	20
kNN_32	0.055	9.756	5.705	2040.483	2056	0.006	0.014	10
kNN_32	0.100	19.524	5.660	2030.716	2056	0.005	0.022	20
kNN_33	0.055	9.756	5.705	2040.483	2056	0.006	0.014	10
kNN_33	0.099	19.525	5.661	2030.715	2056	0.005	0.022	20
kNN_34	0.054	9.758	5.706	2040.482	2056	0.006	0.014	10
kNN_34	0.099	19.525	5.661	2030.715	2056	0.005	0.022	20
kNN_35	0.053	9.759	5.707	2040.481	2056	0.005	0.014	10
kNN_35	0.097	19.526	5.663	2030.713	2056	0.005	0.022	20

Table 16 Values for Precision and Recall for the book-based *k*nn-algorithm for 36–50 neighbors

	TP	FP	FN	TN	*N*	Precision	Recall	*n*
kNN_36	0.054	9.758	5.706	2040.482	2056	0.005	0.014	10
kNN_36	0.099	19.524	5.661	2030.716	2056	0.005	0.022	20
kNN_37	0.054	9.758	5.706	2040.482	2056	0.006	0.014	10
kNN_37	0.101	19.523	5.660	2030.717	2056	0.005	0.022	20
kNN_38	0.052	9.760	5.708	2040.480	2056	0.005	0.014	10
kNN_38	0.098	19.525	5.662	2030.715	2056	0.005	0.022	20
kNN_39	0.051	9.761	5.709	2040.479	2056	0.005	0.014	10
kNN_39	0.099	19.525	5.661	2030.715	2056	0.005	0.022	20
kNN_40	0.052	9.760	5.708	2040.480	2056	0.005	0.014	10
kNN_40	0.097	19.527	5.663	2030.713	2056	0.005	0.022	20
kNN_41	0.053	9.759	5.707	2040.481	2056	0.005	0.015	10
kNN_41	0.098	19.526	5.662	2030.714	2056	0.005	0.023	20
kNN_42	0.056	9.755	5.704	2040.484	2056	0.006	0.016	10
kNN_42	0.099	19.524	5.661	2030.716	2056	0.005	0.023	20
kNN_43	0.058	9.754	5.703	2040.485	2056	0.006	0.016	10
kNN_43	0.101	19.523	5.660	2030.717	2056	0.005	0.024	20
kNN_44	0.058	9.754	5.702	2040.486	2056	0.006	0.016	10
kNN_44	0.099	19.524	5.661	2030.716	2056	0.005	0.023	20
kNN_45	0.059	9.753	5.702	2040.487	2056	0.006	0.016	10
kNN_45	0.101	19.523	5.660	2030.717	2056	0.005	0.023	20
kNN_46	0.060	9.752	5.700	2040.488	2056	0.006	0.016	10
kNN_46	0.102	19.522	5.658	2030.718	2056	0.005	0.024	20
kNN_47	0.058	9.754	5.703	2040.485	2056	0.006	0.016	10
kNN_47	0.103	19.521	5.658	2030.719	2056	0.005	0.024	20
kNN_48	0.059	9.753	5.702	2040.487	2056	0.006	0.015	10
kNN_48	0.104	19.520	5.656	2030.720	2056	0.005	0.024	20
kNN_49	0.059	9.753	5.702	2040.487	2056	0.006	0.015	10
kNN_49	0.103	19.521	5.658	2030.719	2056	0.005	0.023	20
kNN_50	0.057	9.755	5.703	2040.485	2056	0.006	0.015	10
kNN_50	0.101	19.523	5.659	2030.717	2056	0.005	0.023	20

Table 17 Values for RMSE, MSE, and MAE for the control algorithms “random” and “popular”

	RMSE	MSE	MAE
Random_1	2.092	4.383	1.563
Popular_1	1.562	2.443	1.178

Table 18 Values for Precision and Recall for the control algorithms “random” and “popular”

	TP	FP	FN	TN	*N*	Precision	Recall	*n*
Random_2	0.020	9.980	5.740	2040.260	2056	0.002	0.003	10
Random_2	0.054	19.946	5.706	2030.294	2056	0.003	0.010	20
Popular_2	0.249	9.751	5.511	2040.489	2056	0.025	0.059	10
Popular_2	0.360	19.640	5.400	2030.600	2056	0.018	0.082	20

References

1. Börsenblatt (2021) Online-Buchhandel mit hohen Wachstumsraten. https://www.boersenblatt.net/news/buchhandel-news/online-buchhandel-mit-hohen-wachstumsraten-185281
2. Adomavicius G, Tuzhilin A (2011) Context-aware recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook. Springer, Boston, pp 217–253
3. Villegas NM, Sánchez C, DÍaz-Cely J, Tamura G (2018) Characterizing context-aware recommender systems: a systematic literature review. Knowl Based Syst 140:173–200. https://doi.org/10.1016/j.knosys.2017.11.003
4. Raza S, Ding C (2019) Progress in context-aware recommender systems—an overview. Comput Sci Rev 31:84–97. https://doi.org/10.1016/j.cosrev.2019.01.001
5. Kunaver M, PoŽrl T (2017) Diversity in recommender systems—a survey. Knowl Based Syst 123:154–162. https://doi.org/10.1016/j.knosys.2017.02.009
6. Ramakrishnan G, Saicharan V, Chandrasekaran K, Rathnamma MV, Ramana VV (2020) Collaborative filtering for book recommendation system. In: Das KN, Bansal JC, Deep K, Nagar AK, Ponnambalam P, Naidu RC (eds) Soft computing for problem solving. Springer, Singapore, pp 325–338
7. Da'u A, Salim N (2019) Recommendation system based on deep learning methods: a systematic review and new directions. Artif Intell Rev 53(4):2709–2748. https://doi.org/10.1007/s10462-019-09744-1
8. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv 52:1. https://doi.org/10.1145/3285029
9. Wu L, He X, Wang X, Zhang K, Wang M (2021) A survey on neural recommendation: from collaborative filtering to content and context enriched recommendation. arXiv preprint arXiv: 2104.13030
10. Olson DL, Shi Y (2007) Introduction to business data mining. McGraw-Hill/Irwin, New York
11. Shi Y, Tian Y, Kou G, Peng Y, Li J (2011) Optimization based data mining: theory and applications. Springer, Berlin
12. Shi Y (2022) Advances in big data analytics. https://doi.org/10.1007/978-981-16-3607-3
13. Tien JM (2017) Internet of things, real-time decision making, and artificial intelligence. Ann Data Sci 4:149–178. https://doi.org/10.1007/s40745-017-0112-5
14. Batmaz Z, Yurekli A, Bilge A, Kaleli C (2018) A review on deep learning for recommender systems: challenges and remedies. Artif Intell Rev 52(1):1–37. https://doi.org/10.1007/s10462-018-9654-y
15. Foerch Brenes R, Johannssen A, Chukhrova N (2022) An intelligent bankruptcy prediction model using a multilayer perceptron. Intell Syst Appl 16:200136. https://doi.org/10.1016/j.iswa.2022.200136
16. Yeganeh A, Shadman A, Abbasi SA, Pourpanah F, Johannssen A, Chukhrova N (2022) An ensemble neural network framework for improving the detection ability of a base control chart in non-parametric profile monitoring. Expert Syst Appl 204:117572. https://doi.org/10.1016/j.eswa.2022.117572
17. Kurani A, Doshi P, Vakharia A, Shah M (2023) A comprehensive comparative study of artificial neural network (ANN) and support vector machines (SVM) on stock forecasting. Ann Data Sci 10(1):183–208. https://doi.org/10.1007/s40745-021-00344-x

18. Yeganeh A, Chukhrova N, Johannssen A, Fotuhi H (2023) A network surveillance approach using machine learning based control charts. Expert Syst Appl 219:119660. https://doi.org/10.1016/j.eswa.2023.119660
19. Yeganeh A, Johannssen A, Chukhrova N, Abbasi SA, Pourpanah F (2023) Employing machine learning techniques in monitoring autocorrelated profiles. Neural Comput Appl. https://doi.org/10.1007/s00521-023-08483-3
20. Zhang Y, Chen X (2020) Explainable recommendation: a survey and new perspectives. Found Trends Inf Retrieval 14(1):1–101. https://doi.org/10.1561/1500000066
21. Ghazimatin A, Balalau O, Saha Roy R, Weikum G (2020) PRINCE: provider-side interpretability with counterfactual explanations in recommender systems. In: Proceedings of the 13th international conference on web search and data mining. Association for Computing Machinery, pp 196–204
22. Li X, Xiong H, Li X, Wu X, Zhang X, Liu J, Bian J, Dou D (2022) Interpretable deep learning: interpretation, interpretability, trustworthiness, and beyond. Knowl Inf Syst 64(12):3197–3234. https://doi.org/10.1007/s10115-022-01756-8
23. Yang Z, Wu B, Zheng K, Wang X, Lei L (2016) A survey of collaborative filtering- based recommender systems for mobile internet applications. IEEE Access 4:3273–3287. https://doi.org/10.1109/access.2016.2573314
24. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37. https://doi.org/10.1109/mc.2009.263
25. Ziegler C-N, McNee S, Konstan J, Lausen G (2005) Improving recommendation lists through topic diversification. In: Proceedings of the 14th international World Wide Web conference. ACM Press, London
26. Alharthi H, Inkpen D, Szpakowicz S (2017) A survey of book recommender systems. J Intell Inf Syst 51(1):139–160. https://doi.org/10.1007/s10844-017-0489-9
27. Hahsler M (2021) recommenderlab: lab for developing and testing recommender algorithms. R package version 0.2-7. https://github.com/mhahsler/recommenderlab
28. Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53. https://doi.org/10.1145/963770.963772
29. Lika B, Kolomvatsos K, Hadjiefthymiades S (2014) Facing the cold start problem in recommender systems. Expert Syst Appl 41(4 Part 2):2065–2073. https://doi.org/10.1016/j.eswa.2013.09.005
30. Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the fourteenth conference on uncertainty in artificial intelligence. Morgan Kaufmann, pp 43–52
31. Wenga C, Fansi M, Chabrier S, Mari J-M, Gabillon A (2021) A comprehensive review on non-neural networks collaborative filtering recommendation systems. In: arXiv preprint arXiv:2106.10679 .https://doi.org/10.48550/arXiv.2106.10679
32. Beel J, Langer S (2015) A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. In: Kapidakis S, Mazurek C, Werla M (eds) Proceedings of the 19th international conference on theory and practice of digital libraries. Lecture Notes in Computer Science, vol 9316, pp 153–168
33. Bradley K (2001) Improving recommendation diversity. In: Proceedings of the AICS '01
34. Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowl Based Syst 46:109–132. https://doi.org/10.1016/j.knosys.2013.03.012
35. James G, Witten D, Hastie T, Tibshirani R (2013) An introduction to statistical learning. Springer, New York
36. Desrosiers C, Karypis G (2010) A comprehensive survey of neighborhood-based recommendation methods. In: Recommender systems handbook. Springer, Berlin, pp 107–144
37. Herlocker J, Konstan JA, Riedl J (2002) An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf Retrieval 5(4):287–310. https://doi.org/10.1023/A:1020443909834
38. Sarwar B, Karypis G, Konstan J, Reidl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the tenth international conference on World Wide Web—WWW '01. ACM Press, London
39. Ng A, Soo K (2017) Numsense! Data science for the Layman: no math added. Lightning Source Inc
40. Funk S (2006) Netflix update: try this at home. https://sifter.org/~simon/journal/20061211.html
41. BookCrossing, About BookCrossing. (2021). https://www.bookcrossing.com/about

42. Adamopoulos P, Tuzhilin A (2014) On unexpectedness in recommender systems: or how to better expect the unexpected. ACM Trans Intell Syst Technol 5/4:1–32. https://doi.org/10.1145/2559952
43. Park Y-J, Tuzhilin A (2008) The long tail of recommender systems and how to leverage it. In: Proceedings of the 2008 ACM conference on recommender systems. Association for Computing Machinery, pp 11–18
44. Deldjoo Y, Bellogin A, Di Noia T (2021) Explaining recommender systems fairness and accuracy through the lens of data characteristics. Inf Process Manag 58:5. https://doi.org/10.1016/j.ipm.2021.102662
45. Zajac Z (2017) Goodbooks-10k: a new dataset for book recommendations. In: FastML
46. Wan M, McAuley JJ (2018) Item recommendation on monotonic behavior chains. In: Pera S, Ekstrand MD, Amatriain X, O'Donovan J (eds) Proceedings of the 12th ACM conference on recommender systems. Association for Computing Machinery, pp 86–94
47. Wan M, Misra R, Nakashole N, McAuley JJ (2019) Fine-grained spoiler detection from large-scale review corpora. In: Korhonen A, Traum DR, Màrquez L (eds) Proceedings of the 57th conference of the association for computational linguistics. Association for Computational Linguistics, pp 2605–2610

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.