ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master Thesis Data Science and Marketing Analytics

Enhancing rating predictions with collaborative filtering

The effect of individually implementing or combining collaborative filtering methods on rating predictions

---

Name student: Adriaan Zantman

Student ID number: 647049az

Supervisor: drs. Jeffrey Durieux

Second assessor: dr. Vardan Avagyan

Date final version: August 13, 2023

Abstract

With the rise of the internet and booming consumerism, consumers now have more options than ever and rely on recommendations from others. Companies, such as digital platforms like streaming services or retail platforms, search for methods improving the recommendations for their customers. Collaborative filtering (CF) models are used to recommend items to a customer by predicting ratings of items that a customer has not interacted with yet. CF methods can be divided in two main techniques: neighbourhood approaches and latent factor models. Neighbourhood approaches, with or without clustering, and latent factor models have both been extensively researched and applied separately but their combined application is still largely unexplored in the literature. Nevertheless, these methods could potentially benefit from a combination with the other. In this study, a novel integration of an item-based neighbourhood approach with item clustering and matrix factorization is introduced, with the goal of investigating whether the performance accuracy and computational expense of the combined model has improved compared to the individual models. The combination is performed by the ensemble methods bagging, boosting and Extreme Gradient Boosting (XGBoost). These individually implemented and combined CF models are applied to the MovieLens and Netflix datasets containing movie ratings and to the Amazon dataset containing book ratings. The results reveal that, for each dataset, the individual implementation of Singular Value Decomposition (SVD) outperforms any individual CF model and any combination of CF models based on performance accuracy and computational expense. The findings underscore the significance of datasets with high rating density and low skewness as crucial factors for optimal performance of collaborative filtering models. Knowing that SVD predicts product ratings most accurately and time efficiently, is valuable from a managerial perspective because it enables companies to predict what kind of products are potentially preferred by their customers and therefore likely to be purchased.

Table of Contents

1 Introduction

With the rise of the internet and booming consumerism, consumers now have more options than ever, whereas advertisers are challenged with customizing their advertising campaigns (Melville & Sindhwani, 2010). The authors agree that nowadays, businesses must gather substantial amounts of customer transaction data to better understand the customers' interactions with their product offerings. Customers, however, typically rely on recommendations from others since they have little first-hand knowledge with the alternatives (Resnick & Varian, 1997). Recommendation systems have progressed to automate the creation of recommendations based on data analysis in order to satisfy this inherent dual desire of consumers and sellers (Melville & Sindhwani, 2010; Resnick & Varian, 1997). The popularity of recommendation systems is reflected by MacKenzie et al. (2013) stating that personalised recommendations provided by recommendation systems have driven at least 35% of purchases on Amazon and 75% of viewing choices on Netflix in 2013, acknowledged by Donnelly et al. (2023) and Zhou and Zou (2023a).

Collaborative Filtering (CF) is a widely known recommendation system that uses feedback of all users to recommend items to individual users (Goldberg et al., 1992). CF models are used to predict a user's ratings for items that he or she has not interacted with yet. To make item recommendations, the company can represent a user with a personalised ranking of predicted top rated items by that particular user. Nowadays, CF is one of the most popular and effective techniques for automatic product suggestion online (Ahn, 2008). CF models are used as recommendation system in a wide variety of areas where customer ratings are available, such as digital platforms like streaming services or retail platforms that offer an extensive collection of item ratings. For instance, CF models are used on retail platforms like Bol.com to recommend products to customers using product reviews (Kersbergen & Schelter, 2021) and on streaming services like Netflix to suggest which movies or series to watch next based on content ratings (Bennett & Lanning, 2007).

Herlocker et al. (2004) and Koren et al. (2021) state that CF techniques can be divided in two main techniques: neighbourhood approaches and latent factor models. Neighbourhood approaches attempt to predict a target user's ratings based on the relationship between items (item-based) or based on the relationship between users (user-based) (Koren et al., 2021; Sarwar et al., 2001). For predicting the item rating by a user, the authors state that an item-based neighbourhood technique identifies items that are similar to the target item. The item-based approach recommends the target item to a user if the item is similar to items that the user has rated favourably. The similarity between these items is determined by the ratings given by other

users. On the other hand, a user-based neighbourhood approach identifies users that provide similar ratings to the target user. The user-based neighbourhood approach recommends a new item to the target user if these similar users have rated it favourably. Latent factor models (also called matrix factorization) are CF models that attempt to identify hidden or latent factors that capture underlying patterns and preferences in user-item interactions, similar to user ratings of an item. These latent factor models convert users and items into the same latent factor space and attempt to predict ratings by characterizing both users and items on factors (Koren et al., 2021; Kumar et al., 2015).

Additionally, CF models suffer from scalability issues as there is an ever increasing amount of data available (Gong, 2010). Clustering techniques are used in combination with neighbourhood approaches to solve this scalability problem. Clustering is an unsupervised learning method that segments a dataset in different partitions of data points, so-called clusters (Kaufman & Rousseeuw, 1990). Clustering techniques are not used to predict outcomes but can be used in combination with supervised learning methods to optimize their predictions. For item-based neighbourhood approaches, items can be clustered together based on similar ratings. The item-based neighbourhood approach then searches through this created cluster of related items rather than the whole item database to locate objects that are similar to the target item (Gong, 2010; Ungar & Foster, 1998). The authors state that for user-based neighbourhood approaches, similar users are clustered together such that the neighbourhood that the CF model has to search through, decreases drastically.

The quality of such recommendation systems is generally measured in performance accuracy of the predicted ratings and computational expense (Koren et al., 2021). Factorization techniques are typically more accurate than neighbourhood approaches (Koren, 2009; Koren et al., 2021) and generally neighbourhood approaches benefit from clustering in terms of accuracy and computation time (O'Connor & Herlocker, 1999; Ungar & Foster, 1998). Even though a combination of these methods could result in a more accurate and computational efficient collaborative filtering model, most research papers investigate the use of an individually implemented CF model, such as solely investigating a neighbourhood approach or latent factor model (Koren et al., 2021). This limited research on the combination of CF models presents the opportunity to investigate their potential benefits.

The focus of this thesis is to identify which techniques contribute to a better performing CF model in terms of prediction accuracy and computational expense, by evaluating neighbourhood approaches, clustering and matrix factorization techniques separately and combined.

1.1 Research question

In this research the MovieLens and the Netflix Prize competition data, both containing movie ratings, and the Amazon dataset containing book ratings, will be used to create collaborative filtering models. This research will investigate the contribution of individually implemented or combined collaborative filtering methods for each dataset. This results in the following main research question:

"To what extent does the combination of an item-based neighbourhood approach with item clustering and a latent factor model enhance the performance accuracy and computational efficiency of the recommendation system compared to the individually implemented collaborative filtering methods?"

The focus of this research is three-fold by investigating the effect of applying item clustering to an item-based approach, investigating which latent factor models performs best and investigating how best to combine the neighbourhood approach with clustering and matrix factorization. The main research question is therefore divided in the following sub-questions:

(a) *"Which item clustering technique enhances the performance accuracy and the computational expense of an item-based collaborative filtering model most?"*

(b) *"Which latent factor model achieves the highest level of performance accuracy and computational efficiency?"*

(c) *"Which combination of an item-based collaborative filtering model with item clustering and a latent factor model acquires the best performance accuracy and computational expense, and which method is best used to combine them?"*

The results in this research will reveal that the individual implementation of the latent factor model Singular Value Decomposition (SVD) outperforms any individually implemented or combination of collaborative filtering models based on performance accuracy and computational expense. The findings will underscore the significance of datasets with high rating density and low skewness as crucial factors for optimal performance of collaborative filtering models.

1.2 Relevance

1.2.1 Managerial perspective

From a managerial perspective, it is valuable to accurately predict which products are potentially preferred by customers and therefore likely to be purchased. Models that predict item ratings, such as collaborative filtering, contribute to better understanding which products customers prefer and are therefore more likely to purchase. Therefore, a study that compares different CF models to determine which technique predicts item ratings most accurately and time efficiently, has managerial value.

This managerial value can manifest itself at various levels and aspects. For instance on an operational level, determining which CF model constitutes to more accurate rating predictions, is beneficial to tasks such as sales strategies, procurement decisions and inventory management. Better rating predictions enable sales teams to accurately recommend items to the customers that they are likely to purchase, underpinned by the research of Kawaguchi et al. (2019) stating that product recommendations increase the sales of these recommended products. Also, for example procurement decisions can benefit from more accurate rating predictions, since they provide better product demand forecasts allowing the company to adjust its procurement quantities. In addition, accurate rating predictions could for instance benefit managers in optimizing inventory management in preserving an adequate amount of inventory for goods that are anticipated to see increased demand, reducing excess inventory costs or lack of stock.

On a strategic level for example, models that more accurately predict item ratings could provide input data to enhance a competitive advantage, customer retention and campaign strategies. Accurate rating predictions could provide the company with insights which improve its strategic positioning compared to its competition. They could also benefit customer retention as Bennett and Lanning (2007) have stated that the length of users' services is closely related to the number of items they enjoy, since subscribers gravitate towards leaving the platform if they fail to find interesting items. Thus, improving the recommendations connects users to more preferable items, increases customer satisfaction and benefits the company by decreasing the likelihood of their users leaving the platform. In addition, based on more accurate rating predictions, managers can identify which items are most popular amongst the customers and guide targeted marketing campaigns in line with the company's strategy.

1.2.2 Academic perspective

Item-based neighbourhood and matrix factorization approaches have both been extensively researched and applied separately but their combined application is still largely unexplored in the literature, even though these methods could benefit from a combination with the other (Bell & Koren, 2007). Also, while clustering and neighbourhood approaches have been combined in the literature, the addition of matrix factorization to this combination is rather unique. In this study, a novel integration of an item-based neighbourhood approach with item clustering and matrix factorization is introduced, with the goal of investigating whether the performance accuracy and computational expense of the combined model has improved compared to the separate models. This research contributes to the existing literature by constructing a novel combination of CF techniques and analysing the contribution of each technique.

1.3 Thesis structure

The research question of this thesis and its corresponding sub-questions are stated in Section 1.1 and their business and academic relevance are stated in Section 1.2. In Section 2 the different collaborative filtering models and their marketing implications are related to the literature and a comprehensive overview of the underlying theories behind these models is provided in Section 3. Section 4 describes the datasets used for this research after which the methods employed to achieve the research objectives are elaborated on, in Section 5. The results of the research are provided and used to answer the main research question and corresponding sub-questions in Section 6. After concluding these results in Section 7, the limitations, implications and future research of this research are discussed in Section 8.

2 Literature Review

This section introduces a general overview of the most widely used collaborative filtering methods and how these can be combined in one model.

2.1 Collaborative filtering

The very first recommender system created by Goldberg et al. (1992) was used to suggest mailing lists to its users. Here, filters were built on user feedback to select and recommend only the documents of interest to a particular user. Goldberg et al. (1992) have labelled this recommender system as "Collaborative Filtering" (CF), stating that users collaborate to help the filtering process by documenting their feedback on the product. Since this initial system, the ever-increasing amount of information on the internet enlarges the demand of relevant recommendation systems (Kersbergen & Schelter, 2021).

Of all these automated product recommendations, the two main techniques are collaborative filtering and content-based filtering (Ahn, 2006). Content-based techniques construct product and customer profiles using content information or product attributes, and then determine how well a given customer and product match (Herlocker et al., 2004; Kim & Kim, 2001). In contrast, collaborative filtering identifies customer or product commonalities by using product ratings from customers rather than content data and by using this data, it suggests highly rated goods to a target customer who has purchased comparable things (Herlocker et al., 2004; Vézina & Militaru, 2004). The authors state that the type of utilized data is the major distinction between these recommendation systems where content-based filtering recommends items similar in content to items the customer has liked in the past or matched to attributes of the customer; collaborative filtering recommends items based on the past ratings of all customers collectively. According to Herlocker et al. (2004) and Sarwar et al. (2001), this distinction results in collaborative filtering having the potential for finding an unexpectedly intriguing item for the user that he may not have discovered otherwise (referred to by the authors as a serendipitous recommendation) by relying on ratings from other users for their recommendations. Herlocker et al. (2004) state that standard content-based information filtering systems lack this potential for serendipitous recommendations since they recommend items similar in content to a user.

Despite being utilized more and more, CF models still face the following three issues (Ahn, 2006). Firstly, the data used by recommender systems are often matrices with a lot of missing values because not all customers give their purchased items a rating and, when they do, they only do so for a small set of products (Ahn, 2006). This problem of *data sparsity* causes CF

systems to recommend items based on limited available data. This results in the second challenge for CF models: *cold-start* problems. Cold-start problems refer to the issue of referring items to a new customer who has little to no purchase history since the CF model cannot rely on the customer's feedback to recommend items (Maltz & Ehrlich, 1995). Cold-start problems also occur when a new item with few records ratings needs to be recommended to a customer. Lastly, CF models can get computationally very expensive and expand non-linearly with a large dataset (*scalability*), especially with millions of users and items for a typically web-based recommendation system, resulting in lack of understandability (Ahn, 2006; Sarwar et al., 2001).

This interest in which techniques are most beneficial for CF models especially peaked among researchers when in October 2006, Netflix initiated its so-called "Netflix Prize competition" by releasing over 100 million anonymous movie ratings. The team that could construct a CF model to suggest movie recommendations improving the accuracy from Cinematch, the technology used at the time by Netflix to create movie recommendations, by 10%, would be awarded a grand prize of 1 million dollars (Bennett & Lanning, 2007). In 2009, one team called "BellKor's Pragmatic Chaos" finally managed to achieve a 10% improvement and therefore received the grand prize of 1 million dollars (Koren et al., 2021).

2.2 Neighbourhood approach

Neighbourhood techniques are a subset of collaborative filtering algorithms that search through item or user neighbourhoods to identify similar items or users to the target based on their received or given ratings. These ratings are then leveraged to make predictions of unknown ratings. The main advantages of neighbourhood approaches are the results of the local view it provides between items or between users. Firstly, Bell and Koren (2007) note the importance of this local view by searching through local neighbourhoods, making the method easy to understand. For example, suppose that a dataset of user reviews for movies reveals that those who give "Lord of the Rings 1-2" high ratings also gave high ratings for "Lord of the Rings 3", then a new user who has rated "Lord of the Rings 1-2" highly, will probably like the third film as well (Koren, 2010). Secondly, when new items are added to the dataset, neighbourhood models can quickly offer updated suggestions and have the ability to handle new users as soon as they provide new input, without having to retrain the model and estimate new parameters (Koren et al., 2021). Lastly, by utilizing the preferences of similar users or products, neighbourhood techniques are able to capture serendipitous recommendations since they recommend newly discovered items to users with similar taste (Herlocker et al., 2004).

The main disadvantage of neighbourhood approaches is its scalability. As the dataset grows larger, the computational complexity of the neighbourhood approach increases. The neighbourhood method's search of each neighbourhood of similar items results in a computationally expensive model, especially for a large dataset such as the Netflix Prize and MovieLens datasets (Gong, 2010). Ding and Li (2005) note that neighbourhood approaches are extremely time intensive where the computational cost increases quadratically with the number of items.

Neighbourhood approaches require a user-item matrix with the number of rows corresponding to the number of unique users and the number of columns corresponding to the number of unique items. Each row thus represents the ratings of all movies rated by a particular user. Due to the data sparsity issue, users have only rated a selection of movies resulting in a substantial amount of zero entries in the user-item matrix.

Depending on whether the similarities are determined between users or items, these strategies may be further separated into user-based and item-based techniques. Both variants of neighbourhood approaches share the same advantages and disadvantages as mentioned before. However, item-based neighbourhood approaches are favoured in most cases due to improved accuracy and better scalability (Sarwar et al., 2001). Also, item-based approaches are often better interpretable since users are familiar with their previously preferred items and not with other likeminded users (Koren et al., 2021). Considering these advantages, an item-based neighbourhood approach emerges as the more promising choice for this research, enhancing scalability and offering better interpretability. This aligns with this research's objective to contribute as best as possible to the field of collaborative filtering. Hence, an item-based neighbourhood approach is used in this research.

Commonly used as item-based (or user-based) approach, is the k-nearest neighbour method (Park et al., 2015; Sarwar et al., 2002). Identifying the k most similar neighbours to a target user or item and using their preferences to generate recommendations is the fundamental notion underlying the k-nearest neighbour method (kNN) in collaborative filtering (Koren, 2008). So for item-based collaborative filtering, kNN identifies the k most similar items to a target item based on user ratings (Sarwar et al., 2001). This value of k must be determined beforehand.

2.3 Clustering

To solve the aforementioned scalability and computational complexity issues of CF models, several researchers have investigated the use of clustering in combination with neighbourhood approaches. Since clustering is generally used as unsupervised learning method, clustering techniques do not predict themselves but can be used in combination with supervised methods. Gong (2010) has combined item-based neighbourhood search with item clustering to make rating predictions for the MovieLens dataset which is smaller than the Netflix dataset but both contain movie ratings. Here, the author used a k-means clustering algorithm to cluster users based on similar item ratings. The predicted rating of item i by user u is calculated as the average rating of users similar to u in the neighbourhood cluster weighted by the similarity of each user to the target user u (Gong, 2010). The author has concluded that joining item clustering with neighbourhood search is more scalable and more accurate than the traditional neighbourhood approach. Gong (2010) explains this improved time efficiency by stating that the neighbourhood approach needs to consider a smaller portion of neighbours due to clustering while the neighbourhood approach without clustering needs to search through all possible neighbours.

Sarwar et al. (2002) have used k-means clustering to cluster users instead of items in combination with a user-based neighbourhood approach for the MovieLens dataset. Sarwar et al. (2002) cluster users in order to search through similar users, instead of item clustering for similar items performed by Gong (2010). Sarwar et al. (2002) have also concluded that using the clustered approach results in a substantially faster computational time and better performance accuracy compared to using a neighbourhood approach without clustering. Similarly, O'Connor and Herlocker (1999) and Ungar and Foster (1998) mention that by clustering similar items or users together, the prediction accuracy could increase because the noise generated by ratings on items of dissimilar user interest is removed.

Both Gong (2010) and Sarwar et al. (2002) used k-means clustering. For k-means clustering, the number of clusters k is pre-defined and aims to group similar data points together while keeping dissimilar data points in separate clusters, as follows (Hartigan, 1975; Hartigan & Wong, 1979; Ungar & Foster, 1998): First, k data points are selected as initial centroids. Next, each data point is assigned to the nearest centroid based on Euclidean distance and together form a cluster. K-means clustering identifies the nearest centroid by minimizing the Euclidean distance between an observation (vector of n numerical variables) x and centroid μ as: $d_{euc}(x, \mu) = \sqrt{\sum_{i=1}^{n}(x_i - \mu_i)^2}$ (Hartigan, 1975). The new centroids are defined as the averages of points contained within the cluster (Hartigan & Wong, 1979). This process is repeated until the position of the data points no longer changes. Sarwar et al. (2002) note that future research

could use better clustering algorithms, besides k-means, to improve the prediction quality and time efficiency for the MovieLens dataset but have left this for future research.

An alternative clustering technique is hierarchical clustering. Hierarchical clustering builds a hierarchy of clusters by iteratively merging or splitting clusters based on their similarities (Nielsen, 2016). The result is a dendrogram, which is a tree-like structure that represents the clustering hierarchy. The dendogram can be cut at a specific level to obtain a desired number of clusters that provides a good balance between within-cluster similarity and between-cluster dissimilarity (Nielsen, 2016). The author states that hierarchical clustering (HC) can be divided in agglomerative HC (bottom-up) and divisive HC (top-down): agglomerative HC starts with each data point as an individual cluster and iteratively merges the closest pairs of clusters based on a similarity measure. Divisive HC starts with all data points in a single cluster and iteratively splits the clusters based on dissimilarities into smaller clusters until each data point is in its own cluster (Nielsen, 2016). Similarities and dissimilarities between clusters, as expressed in distances, are calculated differently by certain linkage methods for HC. The single linkage calculates the distance between two clusters based on the shortest distance between any two data points, one from each cluster; complete linkage computes this distance based on the maximum distance between any two data points; average linkage computes this cluster distance based on the average distance between any two data points; and ward's linkage minimizes the within-cluster variation by aiming to merge clusters that have the least increase in overall within-cluster variance (Yim & Ramdeen, 2015). The ward linkage tends to form clusters of more equal size and is less sensitive to outliers. The process continues until all data points are in a single cluster.

Nielsen (2016) notes that the biggest disadvantage of HC is its computational complexity where HC yields a cubic time complexity. This implies that the algorithm's computation cost increases cubically with an increase in the input size. On the contrary, the k-means algorithm is known to have a quadratic time complexity. Also, the main difference between HC and k-means clustering is the fact that for HC, the number of clusters k does not need to be pre-defined. To avoid confusion, in the rest of this paper the number of nearest neighbours for kNN will be denoted by k_{kNN}, the amount of clusters for k-means by k_{kmeans} and the amount of clusters for hierarchical clustering by k_{HC}.

Most papers have investigated k-means algorithms for a large database of movie ratings (such as the MovieLens dataset) and conclude that the addition of clustering provides improved prediction accuracies and more computationally efficient CF models (Sarwar et al., 2002). This research will apply k-means and hierarchical clustering to investigate which is most computationally efficient or improves the prediction quality the most.

2.4 Latent factor models

Latent factor models predict user ratings by splitting the user-item matrix into two lower-rank matrices, a user matrix and an item matrix. Each user is represented as a vector of f latent factors in the user matrix and each item is represented as a vector of f latent factors in the item matrix (Koren et al., 2009). These hidden characteristics are discovered from the data's observed user-item interactions (Koren et al., 2021; Kumar et al., 2015). By breaking the original user-item matrix down into smaller, lower-dimensional matrices, matrix factorization techniques minimize the dimensionality of the matrix (Koren, 2008). This decrease in dimensionality makes it easier to manage large datasets. This is a major advantage in contrast to the computation required by neighbourhood approaches which increases substantially in cost as the dataset gets larger (Gong, 2010).

In contrast to the local view provided by neighbourhood approaches, matrix factorization techniques give a global view by characterizing both users and items on latent factors (Bell & Koren, 2007). Besides the benefit of dimensionality reduction, this global view does make the results less uninterpretable compared to the local view of neighbourhood approaches. For example, when applying matrix factorization to movie ratings, latent factors might measure obvious dimensions such as comedy level, amount of action or orientation to children. However, less well defined dimensions such as depth of character development or plot quality are less interpretable dimensions (Koren, 2008).

There are several matrix factorization techniques used in latent factor models. Singular Value Decomposition (SVD) is a popular matrix factorization technique. SVD requires a user-item matrix, similar to an item-based neighbourhood approach, as mentioned in Section 2.2. For m users and n items, SVD decomposes the original user-item data matrix $m \times n$ A, with rank r, into three separate matrices: $m \times m$ U, $m \times n$ Σ and $n \times n$ V^T (Kumar et al., 2015). Since rhe original user-item matrix A contains a lot of missing values (data sparsity issue), the factorization is only applied to the observed ratings in A according to the following formula:

$$A = U \, \Sigma \, V^T \tag{1}$$

In matrix U, each vector corresponds to a user and captures the latent features of the users where the rows represent the users and the columns the latent features (Kumar et al., 2015). The authors state that in the V^T matrix, each vector corresponds to an item and captures the latent features of the items where the rows represent the items and the columns the latent features. Lastly, matrix Σ with non-negative numbers on the diagonal, reflects the singular

values that describe the weight of each latent characteristic (Kumar et al., 2015). The rank of A is r, denoting the number of linearly independent rows or columns. When solely focusing on the r singular values, the matrices reduce into $m \times r$ for U, $r \times r$ Σ and $r \times n$ V^T. SVD provides the best low-rank approximation of matrix A by keeping the first f singular values ($f < r$) of Σ and removing the other $r - f$, which results in the dimensionality of $(f \times f)$ for Σ (Vozalis & Margaritis, 2006). Since, matrix Σ is sorted, the f largest singular values are kept. The authors state that thus SVD reduces the dimensionality of the data and attempts to capture the most important f latent relations in matrix A. This results in removing the first $r - f$ columns of U resulting in a $(m \times f)$ dimensionality and removing the first $r - f$ rows from V^T resulting in a $(f \times n)$ dimensionality. Hence, the closest linear approximation of original matrix A with a reduced rank f is formulated as follows (Vozalis & Margaritis, 2006):

$$A_f = U_f \, \Sigma_f \, V_f^T \tag{2}$$

A lower value for f results in a lower ranked approximation where less important factors are discarded resulting in a more compact representation of A. In summary, for SVD first the original matrix is decomposed according to equation 1, then only the f first singular vectors and values are kept reducing the matrices U, Σ and V^T to U_f, Σ_f and V_f^T respectively, and lastly a low-rank approximation of original matrix A is made by using the first f singular vectors according to equation 2 (Vozalis & Margaritis, 2006).

Non-Negative Matrix Factorization (NMF) is also a popular matrix factorization technique. NMF requires the input data to be non-negative and decomposes the original user-item data matrix $m \times n$ A into two separate matrices: basis matrix $m \times f$ W and coefficient matrix $f \times n$ H (Wang & Zhang, 2012). Similar to SVD, due to the data sparsity issue, the factorization is only applied to the observed ratings in A. Basis matrix W represents the basis vectors or weightings of the latent factors in the original matrix and comprises of non-negative values. It has the dimensions $(m \times f)$, where m represents the users and f is the total number of desired latent factors and the weightings of the latent factors for a particular user are represented by each row of matrix W (Wang & Zhang, 2012). The coefficient matrix H of dimensionality $(f \times n)$ where n denotes the items, contains the non-negative coefficients of the latent factors for a given item (Wang & Zhang, 2012). NMF reconstructs the original matrix A as follows:

$$A \approx W \, H \tag{3}$$

According to Wang and Zhang (2012), the factor matrices W and H are randomly initialized at the beginning of the NMF procedure. A selection loss function is then minimized by progressively refining these factor matrices. Through optimization methods like gradient descent or alternating least squares, this minimization is accomplished (Wang & Zhang, 2012). The authors state that the factor matrices W and H are updated by NMF at each iteration so that their product approximates the original data matrix A while making sure that W and H are both non-negative. Until the approximation error converges to a desirable level or a certain number of iterations is reached, the optimization process is repeated. The generated factor matrices W and H give a non-negative approximation of the initial data matrix and capture its latent properties.

SVD and NMF are unsupervised matrix factorization techniques that apply factorization on the observed ratings in the original matrix. The number of factors f needs to be pre-determined (Koren, 2009). The f factors developed by SVD or NMF are utilized by CF models to make predictions. Thus, SVD and NMF as CF models use the dimensionality reduction to predict ratings, as will be shown in Section 3.3. The main similarity between SVD and NMF is that both decompose the original matrix in separate matrices, and therefore reducie dimensionality, after which these separate matrices are used to reconstruct the original matrix. According to Kumar et al. (2015) and Wang and Zhang (2012), the main difference between SVD and NMF is that SVD offers a low-rank approximation of A while NMF does not reduce the rank of A for its approximation, as shown in equations 2 and 3 respectively. Moreover, SVD allows for negative values in the matrix while NMF only allows for non-negative values (Kumar et al., 2015; Wang & Zhang, 2012). Nevertheless, this advantage of SVD neutralizes for datasets including ratings since these often do not contain negative ratings. Both authors do conclude that both SVD and NMF attempt to improve accuracy and prediction performance of the CF model by reducing dimensionality.

Another popular matrix factorization technique is Principal Components Analysis (PCA) that attempts to transform the observations of potentially correlated variables into a new set of linearly uncorrelated variables, principal components (Kumar et al., 2015). Abdi (2007) have shown that PCA is a form of SVD because it also gives an estimate of an input matrix by a lower rank matrix of the same dimensions. This research will therefore solely apply SVD and NMF as latent factor models.

2.5 Ensemble methods combining neighbourhood approach and latent factor model

Due to its local view, neighbourhood approaches tend to emphasize on the most prominent and similar ratings and thus potentially overlook the less dominant ratings. This shortcoming is overcome by most latent factor models due to estimating a global structure over all items. However, factorization models poorly detect present relations among a small set of closely associated items (e.g. trilogy movies) in which neighbourhood methods excel (Bell & Koren, 2007). Even though neighbourhood approaches account better for newly added ratings and are better interpretable, factorization techniques are typically more accurate than neighbourhood approaches (Koren, 2009; Koren et al., 2021). This can also be seen for the Netflix Prize dataset where solely using a latent factor model improves Cinematch's performance more than an item-based neighbourhood method (Bell & Koren, 2007).

Neighbourhood approaches can benefit from the addition of matrix factorization and vice versa. Similarly, Bell and Koren (2007) stated that there was no perfect individual model for the Netflix Prize competition and that the best results were achieved from a combination of prediction models that complemented the shortcomings of each other. Nevertheless, similar to most authors, Bell and Koren (2007) did not attempt to combine factorization models and neighbourhood approaches due to it possibly requiring too many latent factors. Koren (2008) do combine both these regional and local models through a unifying model. The author combined an item-based neighbourhood approach with matrix factorization technique SVD by using the ensemble method Extreme Gradient Boosting (XGBoost).

Ensemble methods are machine learning methods that combine the predictions of separate models (weak learners) to construct final predictions. The premise behind ensemble approaches is that by merging the results of several models, the advantages of each individual model could be combined, resulting in better overall performance (Dietterich, 2000). Ensemble methods, such as bagging, random forests, boosting and XGBoost, are popular methods since they outperform any single classifier within the ensemble, where decision trees are commonly used as weak learners (Dietterich, 2000; Schapire, 2003).

In the case of using decision trees as weak learners, the difference between bagging and (gradient) boosting is as follows (James et al., 2013): for bagging, multiple decision trees are trained independently on different subsets of the training data, typically obtained through resampling with replacement. The final prediction is obtained by aggregating the predictions of all these trees through averaging. On the other hand, for boosting there are no multiple models involved but each decision tree is grown sequentially on the adjusted original data based on the results

from the previously grown trees (James et al., 2013). So for boosting the information from the previously built decision trees is used. Bagging therefore decreases variance, not bias, and solves overfitting problems while boosting decreases bias, not variance. Moreover, since fitting only one large decision tree to the current model would probably result in overfitting, bagging and boosting methods learn slowly by using separate decision trees (bagging) or adding a shrunken decision tree to its model at each step of the process (boosting) (Dietterich, 2000; James et al., 2013).

James et al. (2013) note that random forests are an extension of bagging. The authors state that, similarly to bagging, random forests build multiple decision trees but choose a random sample of features at each split in the tree to diversify the tree further. This research will apply bagging instead of random forests. XGBoost is an implementation of the gradient boosting method. As implementation of gradient boosting, XGBoost follows the same principle and methodology as gradient boosting. In addition, XGBoost uses regularization where it shrinks the variable coefficients towards zero (Carmona et al., 2019). XGBoost provides a regression model penalised by the L1 regularization (the variable coefficients are often shrunk to exactly zero, useful for variable selection) and L2 regularization, also known as ridge regression (the coefficients are shrunk towards zero but never set to zero). Often, XGBoost solely uses L2 regularization and thus does not make use of variable selection (Carmona et al., 2019). By using L2 regularization and therefore shrinking variable coefficients towards zero, the variance in predictions substantially decreases at the expense of a bias increase for XGBoost (James et al., 2013). By using regularization, XGBoost helps reduce overfitting, often outperforms gradient boosting and is more computationally efficient than gradient boosting (Carmona et al., 2019).

Koren (2008) uses XGBoost as ensemble method to combine the predictions of an item-based neighbourhood approach and a SVD. Besides XGBoost, this research will also investigate the use of boosting and bagging as ensemble methods to combine the predictions of an item-based neighbourhood approach with item clustering and the predictions of a matrix factorization model.

2.6 Marketing benefits and implications

The aforementioned collaborative filtering techniques provide companies with rating predictions. These predictions can be used to make personalised recommendations by providing a personalised ranking of predicted top rated items that the user has not interacted with yet. However, even though most recommendation algorithms are made to direct customers to the most relevant items, retailers can decide to suggest to them items that are expected to generate the highest

profit for the retailer or marketplace at the expense of the customer's prefences (Donnelly et al., 2023; Zhou & Zou, 2023a). According to the authors, profit-oriented product recommendations are partly based on customer's preferences, suggesting to be customer-oriented personalised recommendations. Nonetheless, their main goal is to increase the marketplace's profit, not to solely tailor the recommendations to the preferences of the consumer resulting in reduced customer focus. Donnelly et al. (2023) have performed a controlled experiment where the treatment group received personalised recommendations and the control group received non-personalised recommendations based on historical popularity. The authors discovered that personalised rankings stimulate more active customer search for more items, significantly raise the likelihood of purchase and shift product demand towards relatively unpopular items. This demand shift towards new and less discovered items, utilizes the potential of collaborative filtering models of making serendipitous recommendations that the customer might not have discovered otherwise, as mentioned previously by Herlocker et al. (2004) and Sarwar et al. (2001). Finally, Donnelly et al. (2023) conclude that personalised recommendations increase the overall benefit consumers gain from a transaction and increase the retailer's revenues. The authors therefore did not find strong evidence of the retailers requiring profit-oriented recommendations to increase profitability since the personalised suggestions increase both profitability and the customers' welfare. Dzyabura and Hauser (2019) extrapolate these findings, asserting that personalised item recommendations result in increased customer satisfaction and higher profits for not only retailers but for companies from a wide range of sectors such as service providers like streaming services and e-commerce platforms.

For online marketplaces, Zhou and Zou (2023a) further investigate the implications of providing profit-oriented recommendations or price-neutral recommendations that are tailored to the customer's preferences and do not account for profit margins. For online marketplaces, Zhou and Zou (2023a) suggest that the marketplace's and sellers' profits are often lower under the profit-based system than under the price-neutral system, if the sellers set their product prices strategically. The authors' reasoning behind this conclusion is that under a profit-oriented market, recommending products with the highest expected profit maximises the marketplace's expected profit resulting in marketplaces that favour recommendations of more expensive goods, which could impair customer welfare. Since sellers in a profit-based system thus have an incentive to raise their pricing when the marketplace is more likely to recommend more expensive goods, the conversion rate and the marketplace's profit are likely to decrease (Zhou & Zou, 2023a). In contrast, the authors mention that in a price-neutral marketplace, the marketplace's profit is maximised by optimizing the conversion rate since customers are more likely to purchase

the fairer priced products as sellers have no incentive to raise their prices. This conclusion is confirmed and strengthened by Liu and Huang (2023) and Zhou and Zou (2023b).

In this research, CF models are used to predict book ratings from the marketplace Amazon. Since the ratings of the Amazon dataset are predicted by not using costs or profits per product in this research, the personalised recommendations that can be acquired from this research are price-neutral, aligning with the preference of price-neutral recommendations over profit-oriented from Donnelly et al. (2023) and Zhou and Zou (2023a).

Moreover, Kawaguchi et al. (2019) have found that personalised recommendations do not solely increase the purchases of the recommended products but also of the non-recommended products. The authors use several economic models to show that product recommendations often drive customers to carefully consider other non-recommended products and generate a spill over of purchases to these non-recommended items. As mentioned previously, Bennett and Lanning (2007) have shown that connecting users with appealing items, leads to an increase in customer satisfaction and retention rate. Due to this spillover effect, users interact with the recommended as well as the appealing non-recommended items, resulting the likelihood of them leaving being further decreased.

3 Theoretical Framework

This section aims to build a theoretical understanding of the collaborative filtering methods mentioned in the literature, Section 2, creating the framework for the subsequent model evaluations in this research.

3.1 Baseline estimation

To predict item ratings of a particular user, the item-based neighbourhood approach starts by identifying similarities between items. Many of the observed ratings are affected by either items or users independently of their interaction, for example some items systematically tend to receive higher ratings than others and some users tend to provide higher ratings than others (Koren, 2009; Koren et al., 2021). These effects are incorporated in the following *baseline predictors* such that the true user-item interaction can be isolated.

A rating of item i provided by user u is denoted by r_{ui}. A baseline prediction for unknown rating r_{ui} accounts for the user-item interaction and is denoted by b_{ui} (Koren, 2009; Koren et al., 2021):

$$b_{ui} = \mu + b_u + b_i \tag{4}$$

Here, μ is the average rating over all movies, b_u is the observed deviation from the average of user u (this incorporates if for example a user systematically gives higher ratings) and b_i is the observed deviation from the average of item i (this incorporates if e.g. an item is systematically rated higher). Both authors state that b_u and b_i can be estimated by solving the following least squares problem:

$$\min_{b^*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 (\sum_u b_u^2 + \sum_i b_i^2) \tag{5}$$

Here, $\mathcal{K}$ is the set of all user-item pairs for which rating r_{ui} is known: $\mathcal{K} = \{(u,i)|r_{ui} \text{ is known}\}$ (Koren, 2009; Koren et al., 2021). The authors state that $\min_{b^*} \sum_{(u,i)\in\mathcal{K}} (r_{ui}-\mu-b_u-b_i)^2$ attempts to acquire b_u and b_i that fit the provided ratings and that $\lambda_1(\sum_u b_u^2 + \sum_i b_i^2)$ is the regularization term with parameter λ_1 attempting to avoid overfitting by L2 penalisation. A high value of regularization parameter λ_1 results in the L2 penalty term becoming more significant in the cost function, equation 5, resulting in more aggressive shrinkage of the coefficients towards zero (Koren et al., 2021). The model becomes more constrained resulting into a simpler and more regularized model less prone to overfitting. A low value of λ_1 results in a less constrained model making it more flexible to capture more complex patterns in the data.

3.2 Item-based neighbourhood approach with item clustering

To calculate the predicted rating of item i by user u, denoted by $\hat{r}_{ui}$, with an item-based neighbourhood approach, first the similarity between two different items i and j needs to be computed. There are different similarity metrics. In this research the cosine similarity is used. By combining the work of Hong and Tsamis (2006) and Chiny et al. (2022), the cosine similarity between items i and j, denoted by $cos_{i,j}(\theta)$ is calculated as follows:

$$cos_{i,j}(\theta) = \frac{\mathbf{I} \cdot \mathbf{J}}{||\mathbf{I}||\,||\mathbf{J}||} = \frac{\sum_{u \in U(i,j)} I_u J_u}{\sqrt{\sum_{u \in U(i,j)} I_u^2}\sqrt{\sum_{u \in V(i,j)} J_u^2}} \tag{6}$$

Here, vector $\mathbf{I}$ denotes the vector of ratings for item i where vector $\mathbf{J}$ denotes the vector of ratings for item j. θ is the angle between the two vectors $\mathbf{I}$ and $\mathbf{J}$ and $U(i,j)$ denotes the set of all users that have rated both items i and j. I_u denotes element u from vector $\mathbf{I}$, so the rating that user u provided for item i.

As shown in equation 6, the cosine similarity measures the similarity between items i and j based on the ratings they have received from common users. The cosine similarity is a normalized similarity measure that considers the magnitude of the compared vectors. It produces a range from -1 to 1 by dividing the dot product of the vectors by the product of their magnitudes. By reducing the influence of different scales or magnitudes in the user-item evaluations, this normalization makes the similarity computation more accurate compared to a similarity measure such as the Pearson correlation coefficient (Hong & Tsamis, 2006). The authors state that since data sparsity is an issue for the dataset, there are limited ratings available for each user-item pair and a small number of ratings can have a significant impact on the Pearson correlation coefficient, which could result in a misleading similarity metric. Thus, the cosine similarity is used to improve the robustness of the similarity calculation.

The item-based neighbourhood approach estimates $\hat{r}_{ui}$ as follows (Gong, 2010; Koren et al., 2021):

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S^{k_{kNN}}(i;u)} (r_{uj} - b_{uj})\, cos_{i,j}(\theta)}{\sum_{j \in S^{k_{kNN}}(i;u)} cos_{i,j}(\theta)} \tag{7}$$

In equation 7, $S^{k_{kNN}}(i;u)$ denotes the set of k_{kNN} items rated by user u that are most similar to i. The similarity is determined by the cosine similarity that averages the ratings of all users who have rated both items i and j. The parameter k_{kNN} is used by the k-nearest neighbour method as the k most similar neighbours to the target item. Hence, the predicted rating of item i by user u ($\hat{r}_{ui}$) is calculated as the average of the ratings of neighbouring items most similar to item i weighted by the similarity of each item to the target item i (Gong, 2010).

When initializing a k-nearest neighbourhood approach, the following hyperparameters need to be estimated or set: the shrinkage parameter λ_1 from equation 5 and the number of nearest neighbours k_{kNN} to consider. The value of k_{kNN} is often k-fold cross validated. k-fold cross validation is a resampling technique that splits the dataset into k subsets or folds (Stone, 1974). The model, in this case kNN, is trained on the $k-1$ training folds and evaluated on the remaining fold, the validation set. This is done k times using a different fold as the validation set each time. The model's performance is evaluated on each validation and the performance metrics are averaged from all iterations to get an overall performance score (Stone, 1974). The values of k_{kNN} and k are unrelated.

For predicting the rating of item i by user u in item-based neighbourhood approaches, the correlation between the target item i and its neighbours who are rated by u ($S(i;u)$) is used, as shown in equation 7. Clustering the items before applying the item-based neighbourhood approach, divides the set of items in smaller partitions (clusters) based on similar rating behaviour. Instead of having to search through all items in the dataset similar to user u that have rated item i to find k_{kNN} nearest neighbours, item clustering sets this neighbourhood $S(i;u)$ to the cluster of item i (Sarwar et al., 2002). Thus, by clustering items before applying item-based neighbourhood approach, predictions for a single item can be made by averaging the ratings of neighbouring items in that cluster (Sarwar et al., 2002). In each cluster k_{kmeans} or k_{HC}, the k-nearest neighbour technique searches for the k_{kNN} nearest neighbours. If k_{kNN} is larger than the number of items in cluster k_{kmeans} or k_{HC}, than the k-nearest neighbour technique will search through the total number of items in that cluster.

Hierarchical clustering defines the optimal number of clusters k_{HC} itself. To determine the optimal number of clusters k_{kmeans} , the silhouette score is evaluated for each value of k_{kmeans} and the best is selected. The silhouette index s_i relates the average distance from object i to all objects in the same cluster, to the best (smallest) alternative distance in the cluster and is defined as follows (Rousseeuw, 1987):

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)} \tag{8}$$

Here, a_i is the average distance between data point i and all other points within the same cluster and b_i is the average distance between point i and all points in the nearest neighbouring cluster (Rousseeuw, 1987). If the cluster only contains data point i, the silhouette score equals 0. The silhouette score thus ranges between -1 and 1 and high values characterize good clustering since then the observations are well matched to their own clusters. So, the value of k_{kmeans} resulting in the highest silhouette score is selected as optimal number of clusters for k-means.

3.3 Latent factor models

Latent factor models or matrix factorization techniques, try to explain ratings by characterizing both items and users on factors by mapping them on a joint latent factor space of dimensionality f (Koren et al., 2021). Each user u is associated with a vector $p_u \in \mathbb{R}^f$ with dimensionality $(f \times 1)$ whose elements measure how much interest each user u has in the items that relate highly to the corresponding factors and each item i is associated with vector $q_i \in \mathbb{R}^f$ with dimensionality $(f \times 1)$ whose elements measure how much the item relates to the factors (Koren, 2008; Koren et al., 2021). The elements in p_u and q_i can be both positive or negative. The authors state that according to these latent factor models, a rating is predicted according to:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{9}$$

Here, the dot product $q_i^T p_u$ denotes the interest of the user in the aspects of an item (the interaction between user u and item i). The matrix factorization techniques SVD and NMF determine the latent factors f as shown in equations 2 and 3 respectively. These f are then used to construct vectors p_u and q_i to be able to predict missing ratings using equation 9 (Aghdam et al., 2017; Koren et al., 2021). The values for q_i and p_u can be calculated by applying a stochastic gradient descent method similar to equation 5 (Koren et al., 2021):

$$\min_{b^*,q^*,p^*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda_2(b_u^2 + b_i^2 + ||q_i||^2 + ||p_u||^2) \tag{10}$$

Here, λ_2 controls the regularization for learning p_u and q_i to avoid overfitting. A low value of λ_2 results in a less constrained model which allows it to fit the training data more closely. Koren et al. (2021) state that the calculation of λ_2 (regularization constant) is usually performed by applying cross validation and that the minimization in equation 10 is usually performed by stochastic gradient descent as practised by Paterek (2007).

3.4 Combining neighbourhood approach, item clustering and matrix factorization

As mentioned previously, neighbourhood approach methods and latent factor models could potentially complement each other. Koren (2008) suggests a model where these two methods are combined by summing the predictions from equations 7 and 9 resulting in the following equation:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u + \frac{\sum_{j \in S^{k_{kNN}}(i;u)} (r_{uj} - b_{uj}) cos_{i,j}(\theta)}{\sum_{j \in S^{k_{kNN}}(i;u)} cos_{i,j}(\theta)} \tag{11}$$

Gradient descent is used to minimize the related regularized squared error function for the item-based neighbourhood approach and matrix factorization separately, following equations 5 and 10, in order to find the values of the model's parameters b_u, b_i, q_i and p_u. Again, item clustering can be applied to decrease the size of neighbourhood $S^{k_{kNN}}(i;u)$. Equation 11 consists of three components: $\mu + b_u + b_i$ describes the overall characteristics of the item and user but does not take into account any interactions between them; $q_i^T p_u$ facilitates the interaction between the item and user profile; and $\frac{\sum_{v \in S^k(u;i)} (r_{vi} - b_{vi}) cos_{u,v}(\theta)}{\sum_{v \in S^k(u;i)} cos_{u,v}(\theta)}$ adds difficult to-profile and specific adjustments.

As mentioned in subsection 2.5, the item-based neighbourhood approach and matrix factorization technique can be combined by ensemble methods according to Koren (2008). The author states that the ensemble methods combine the predicted values of the item-based neighbourhood approach and the matrix factorization technique to create a strong learner by using decision trees as weak learners. Item clustering can be added to the item-based neighbourhood approach. The ensemble method then combines the predicted values of the extended neighbourhood approach with clustering and the predictions of a latent factor model.

Bagging (bootstrap aggregating) is an ensemble method that attempts to reduce variance by averaging the predictions from all the weak learners over separate training sets (James et al., 2013). To do so, bagging takes B random samples with replacement from the original training data where each new sample has the same size as the original training data. This sampling method is called bootstrapping (James et al., 2013). Next, bagging constructs a separate prediction model on each bootstrapped sample. When using decision trees as weak learners, each decision tree from the ensemble makes a prediction on each of the B bootstrapped sample, denoted as $\hat{f}^{*b}(x)$ (James et al., 2013). The authors state that bagging averages the predictions of the decision trees from each bootstrapped sample to obtain a low-variance model that outputs the predicted values of the response variable as follows:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{12}$$

Boosting on the other hand, does not fit B decision trees to B bootstrapped samples, but subsequently adds decision trees to the same original data (James et al., 2013). The authors state that the goal of boosting is to minimize the residuals, which is the difference between the predicted value, denoted by $\hat{f}(x)$, and the observed values of the response variable. Thus,

boosting attempts to predict the response variable as close as possible to its actual value and does this as follows (James et al., 2013). First the residuals are calculated by using the average of the actual value of the response variable as initial prediction and substracting that from the observed values. Next, a decision tree, denoted by $\hat{f}^b$ with $b \in B$, is fit to these residuals and predicts the value of the response variable. The residuals are again updated by subtracting the updated predicted values of the current model, $\hat{f}(x)$, from the actual observed values of the response variable. After updating the residuals, a new decision tree is fit to these updated residuals shrunken by a certain shrinkage rate λ. This shrinkage rate lies between 0 and 1 and prevents overfitting by simply fitting a single decision tree to the residuals (James et al., 2013). By adding a decision tree to the model each tree is grown using the information of the previously grown trees and by shrinking each added tree, the learning process is slowed down. Boosting shrinks the added tree, $\hat{f}^b$, with shrinkage rate λ to the current predicted value of the response variable $\hat{f}(x)$ as follows (James et al., 2013):

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b \tag{13}$$

The gradient boosting model subsequently adds new decision trees to the model until the total number of B decision trees has been added, after which the predicted values are outputted. This boosting process can be formulated by (James et al., 2013):

$$\hat{f}_{boost}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x) \tag{14}$$

XGBoost applies the same technique as gradient boosting in addition to L1 or L2 regularization. The learning rate λ is often cross validated for boosting and XGBoost where a higher value for λ reduces the impact of each weak learner. Moreover, the number of splits in the added decision tree during the boosting or XGBoost process is denoted as parameter s, often referred to as the interaction depth. A higher value of s allows the boosting and XGBoost methods to add larger tree to the current model that capture more complex details in the data. For boosting and XGBoost the value of s can also be cross validated.

Since the ensemble methods combine the predicted values of the individually implemented CF models, the ensemble methods are necessarily more computationally expensive than the individually implemented CF models because the individually implemented CF models first need to make their predictions after which the ensemble methods can combine these predictions. The combined CF model can still be preferred over the individually implemented models besides the increased computational expense if its performance accuracy is better. Some companies

could prefer the use of a CF model that takes more time but makes more accurate rating predictions. The trade-off between performance accuracy and computational expense of the CF model depends on the objectives of a company.

3.5 Evaluation metrics

Collaborative filtering methods are supervised methods and make predictions. The performance of CF methods is evaluated on their prediction accuracy and computational expense in this thesis. The computational expense equals the running time of each each method from start to end with the optimal parameter settings. Thus parameter tuning processes are not included in these running times since these processes vary per method and are therefore not comparable.

The prediction accuracy of CF methods can be evaluated by several metrics. Koren et al. (2021) state that a popular performance metric used for neighbourhood approaches and matrix factorization techniques is the Root Mean Squared Error (RMSE). The RMSE value of a model is defined as follows (Koren et al., 2021):

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{15}$$

In equation 15, y_i denotes the dependent variable (item ratings) and its predicted values $\hat{y}_i$ for n observations. So RMSE measures the average magnitude of the differences between the predicted values and the actual values and a low RMSE value is desirable because it indicates that the model's predictions are close to the actual values (Koren et al., 2021). If the RMSE value is 1 for example, that means that, on average, the predictions of the model deviate from the actual values by approximately 1 unit. These units differ per dataset and consider the scale and context of the data.

Another popular performance metric for collaborative filtering is the so-called Mean Absolute Error (MAE) which is defined as follows (Sarwar et al., 2001):

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{16}$$

MAE measures the average magnitude of the differences between the predicted values and the actual values, without considering the direction of the differences, as depicted in equation 16. A low MAE is desirable because it indicates that the model's predictions have a smaller average absolute difference from the actual values (Sarwar et al., 2001).

To make the MAE metric more interpretable, it can be expressed as a percentage via the Mean Absolute Percentage Error (MAPE) which provides a measure of the average relative error in percentage terms and is defined as follows (De Myttenaere et al., 2016):

$$MAPE = \frac{1}{n} \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{y_i} * 100\% \tag{17}$$

Equation 17 shows that a low MAPE is desirable because it suggests a better performance accuracy of the model (De Myttenaere et al., 2016). If the MAPE value is 10% for example, that means that, on average, the predictions of the model are accurate within $\pm$ 10% of the observed values. In this research, the RMSE and MAPE are used as error metrics to evaluate the different collaborative filtering models.

4 Data Description

The collaborative filtering methods used in this research make use of explicit feedback which is data where users directly report on their interest in products such as providing ratings (Koren et al., 2021). In this research, collaborative filtering models are tested on three different datasets to identify whether their performances are consistent and robust or if they are sensitive to specific dataset properties. The MovieLens and Netflix datasets containing movie ratings are used in this research since prior research has shown these datasets to be suitable for training and evaluating CF models (Gong, 2010; Koren et al., 2021). Furthermore, the Amazon books dataset containing book ratings is used to assess the generalizability of the proposed methods beyond movie ratings and to investigate whether the models are sensitive to specific data properties. All three datasets contain three variables (user ID, item ID and rating) where each row corresponds to a user-item pair with the corresponding rating value. Evidently, the number of user-item pairs in each dataset (the number of rows) equals the total number of ratings provided.

In this research the complete datasets are used in their original format without applying any form of feature selection beforehand. Collaborative filtering models merely require the aforementioned three variables for the user-item pairs and their rating values. Contextual information of each ratings, such as the personal details of the users or characteristics of the items, are not required and thus not provided. By utilizing all three variables in the datasets, no variable selection is applied. Moreover, no feature selection is applied to obtain the ratings of a certain user or item segment such as solely evaluating users that provide high ratings or items that receive high ratings. This research aims to provide a thorough and objective approach to the collaborative filtering techniques used and therefore bases its recommendations on all available data.

4.1 MovieLens

For this research, the MovieLens dataset is used containing ratings r_{ui} of user u rating movie i. The dataset consists of a user ID, item ID and rating for each user-item pair. The MovieLens dataset, is a dataset that keeps getting updated with new and more movie ratings (Herlocker et al., 2004; Sarwar et al., 2002) This dataset is obtained from Kaggle and consists of over 20 million movie ratings provided by 138493 unique users about 26744 movie titles. Movie ratings are expressed in stars. The movie ratings are on a scale from 0.5 to 5 stars where half stars can be given. These ratings are collected between January 09, 1995 and March 31, 2015. These users were randomly selected and have all provided at least 20 ratings. Random sampling was

used to obtain a representative subset of the entire population of users in the MovieLens dataset which helps to reduce selection bias. The minimum of 20 ratings for each user reduces the data sparsity in the dataset.

Figure 4.1 displays how many reviews are provided with the 10 different rating values and shows that most movie ratings provided in the dataset are 3 or 4 stars. Ratings from 0.5 to 2 stars only consist of 13% of the total number of ratings, indicating that most movie reviews are positive. Where stronger emotions and lasting impact are typically evoked and left by extraordinary good or poor movies, it is often thought that consumers are thus more likely to rate a movie with a strong favorable or negative reaction. Nevertheless, Figure 4.1 shows that most ratings are either neutral (3 stars) or positive (4 or 5 stars).

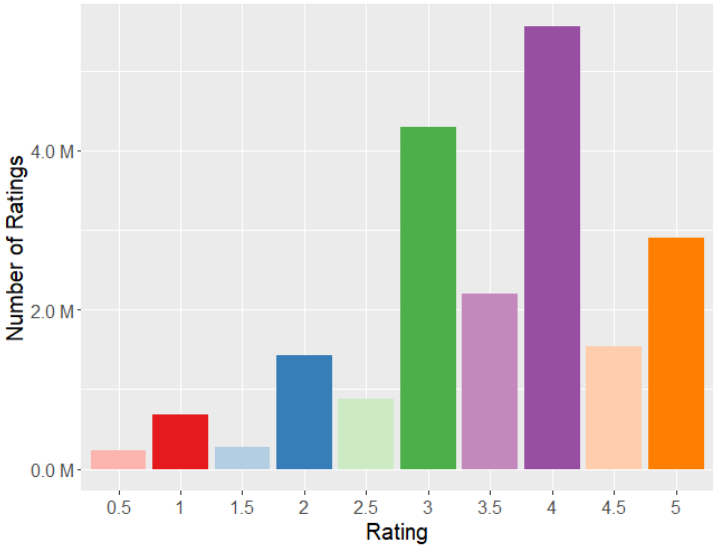

Figure 4.1: Rating distribution of the MovieLens dataset

Gong (2010) has evaluated the performance of CF models for different ratios of train and test set divisions of the MovieLens dataset. The author ranged this ratio from 80% train and 20% test set to 20% train and 80% test set with steps of 10%. Gong (2010) has concluded that the quality of prediction for CF models is best when the dataset is divided in 80% train and 20% test data set. This research will therefore also divide the MovieLens dataset in 80% train and 20% test data set.

4.2 Netflix Prize competition

For this research paper, the Netflix Prize competition dataset is used. This dataset is obtained from Kaggle and consists of over 100 million movie ratings provided by 480189 users about 17770 movie titles (Bennett & Lanning, 2007). The data set consists of user IDs, item IDs and ratings for each user-item pair. Similarly to the MovieLens dataset, the movie ratings are expressed in

stars. The ratings are on an integer scale from 1 to 5 stars and are collected between October 1998 and December 2005. These ratings represent the distribution of all movie ratings from users who have at least provided 20 ratings in this time period (Bennett & Lanning, 2007). Similar to the MovieLens dataset, these users are randomly selected by Netflix, each with a minimum of 20 rated movies.

The whole dataset is divided in a probe, quiz and test set (Bennett & Lanning, 2007). The probe set contains the actual movie ratings such that contestants can compute and compare their RMSE values. The quiz and test set do not contain the actual movie ratings. The contestants' predicted ratings from the quiz set are evaluated by Netflix and their corresponding RMSE values posted on the leader board. Their predicted ratings for the test set are not reported by Netflix but used to identify potential winners of the Netflix Prize Competition (Bennett & Lanning, 2007). For this research, the probe set is used since it contains the actual movie ratings which enables the evaluation of a model's performance accuracy.

Figure 4.2 displays how many reviews are provided with the 5 different rating values. It can be seen that most movie ratings provided in the dataset are 4-star or 3-star ratings. Ratings with 1 or 2 star only consist of 15% of the total number of reviews, indicating that most movie reviews are positive. Similar to Figure 4.1, Figure 4.2 shows that most ratings are either neutral (3 stars) or positive (4 or 5 stars) in contrast to the common thought that consumers are more likely to evoke strong negative emotions as well. Similar to the MovieLens dataset, this probe set is divided in 80% training set and 20% test set, according to the findings of Gong, 2010.

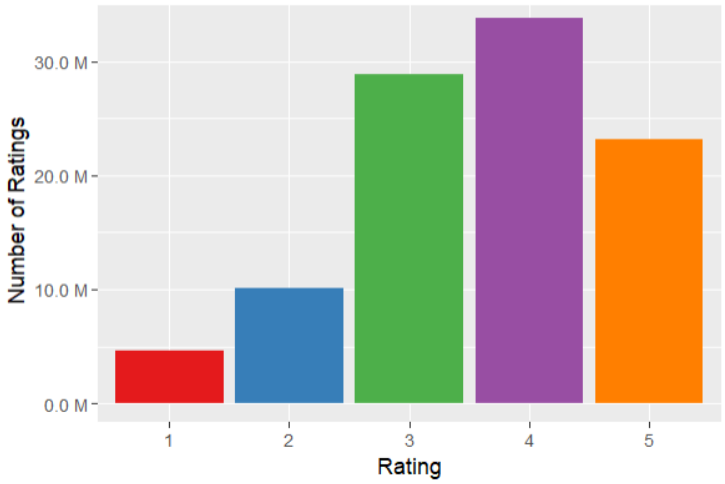

Figure 4.2: Rating distribution of the Netflix Prize competition dataset

4.3 Amazon books

Whereas the MovieLens and Netflix dataset contain movie ratings, this research seeks to extend its results to other fields for item recommendations. Hence, the Amazon Books dataset is used, obtained from Kaggle which consists of around 2.4 million book ratings after the removal of duplicates, see subsection 5.1. This dataset contains user IDs, item IDs and rating values. 1008972 unique users provided these 2.4 million ratings for 216023 books. The book ratings are, similar to the Netflix dataset, on an integer scale from 1 to 5 stars rated between May 1996 and July 2014. This dataset is a subset of around 140 million product reviews and metadata published by Amazon. Similar to the MovieLens and Netflix datasets, the Amazon books dataset is divided in 80% train data and 20% test data in light of the findings of Gong (2010). The rating distribution of the whole Amazon books dataset is provided in Figure 4.3 which shows that a large portion of the ratings have a value of 4 or 5, namely 20% and 60% of the ratings respectively. The Amazon dataset shows a high right-skewed rating distribution compared to the rating distribution of the MovieLens and Netflix datasets, in Figures 4.1 and 4.2. This highly skewed data could result in algorithms emphasizing on the patterns of the majority class and failing to generalize well to the less represented rating values, which could increase the risk of overfitting.

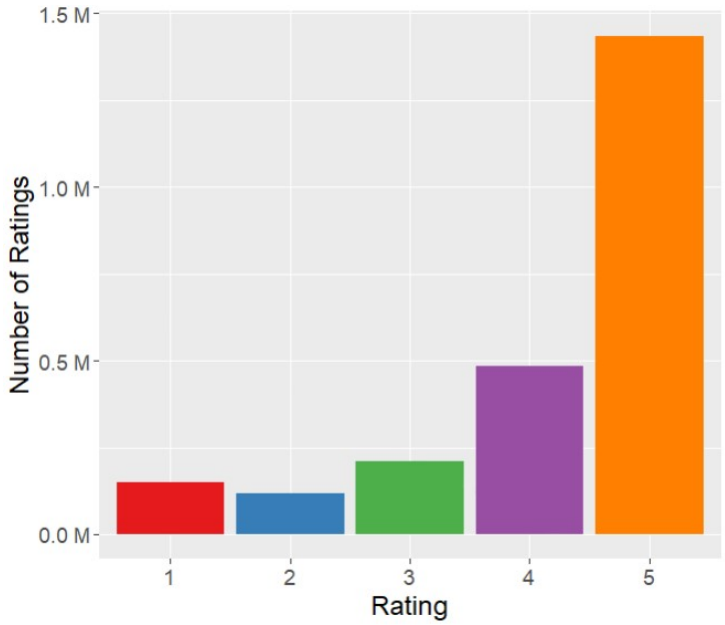

Figure 4.3: Rating distribution of the Amazon books dataset

In contrast to the MovieLens and Netflix dataset, the users for the Amazon book ratings dataset did not all provide at least 20 ratings. On average each user only provides 2 ratings since there are 2.4 million book rated by approximately 1 million users. The MovieLens and Netflix

datasets exhibit a much denser structure with a high volume of ratings compared to the Amazon dataset. This necessitates a much larger subset of users and items from the Amazon dataset to match the number of ratings of the MovieLens and Netflix datasets. Handling the larger size of the Amazon dataset needs more resources and might result in processing and analysis constraints, which makes it less suitable for some memory-constrained environments. Thus, to match the number of users and items as the MovieLens or Netflix data, the subsetted Amazon dataset will contain far less item ratings. This lower number of ratings could result in poorer performances of the CF models since they may not have enough information to make accurate predictions. This less dense rating structure and highly right-skewed rating distribution could result in less robust conclusions deviating from the conclusions drawn from the MovieLens and Netflix datasets.

5 Methodology

This section covers the steps taken for processing the data and covers the machine learning models used to make predictions based on the processed data. The code provided by GitHub (2018) in line with the research of Koren (2008), served as a helpful foundation for the coding of this research.

5.1 Data processing

In this research, various CF models are constructed and used as recommendation systems for the MovieLens, Netflix and Amazon datasets. The large dataset of movie ratings is cleaned by removing all missing values (NaN) and duplicates. NaN values respresent missing data and might lead to inconsistencies in the dataset. Leaving NaN values in the dataset might result in inaccurate computations, have an impact on statistical analysis and yield findings that are not trustworthy. Therefore, the NaN values are removed from the data. Duplicate entries can unexpectedly increase or decrease the importance of specific user-item interactions, which might result in biased recommendations. Also, if duplicates are present in both the training and test sets, they leak information between the sets and result in the model having prior knowledge of the user-item interactions when testing. Duplicates are therefore also removed from each dataset. Since the datasets are each separated in 80% training data and 20% test data, *cold-start* issues are addressed. The whole dataset contains user-item pairs and thus by randomly taking a training and test set, the training and test set both contain approximately the same users and items but with different ratings. Hence, the test set does not necessarily contain new users or items which are not present in the train set on which the models are built.

An item-based neighbourhood approach and SVD require a user-item matrix as input, as mentioned in Sections 2.2 and 2.4 respectively. Thus, for each dataset, the train and test sets are transformed in user-item matrices. Due to a sufficient number of zero entries in the user-item matrices (data sparsity issue), sparse matrices are applied. These matrices produce a significantly more memory-efficient representation by storing only the non-zero values and their indices. Similar to the user-item matrix, for these sparse matrices the number of columns corresponds to the number of unique items and the number of rows corresponds to the number of unique users. Each row thus represents the ratings of all movies rated by a particular user. For the MovieLens and Netflix dataset, the items correspond to movies and for the Amazon dataset the items correspond to books.

The memory and computing capability restrictions of the hardware employed in this study, limit the analysis of each dataset to a certain number of users and items. The size of the sparse matrices grow when increasing the number of items or users but not when increasing the number of ratings. Thus, the capability of the used hardware mainly restricts the number of items and users used for the analysis instead of the number of ratings. As a consequence, the datasets are limited to subsets of the users and items that could be processed and analysed effectively using the technology that was available. The train and test set of the datasets are subsetted separately by randomly selecting a number of items and subsequently selecting all the users rating these items. The items are randomly sampled to obtain a representative subset of the entire population. Selecting all users that have rated these items preserves a diverse range of user-item interactions within the reduced dataset compared to random sampling both items and users which could result in a set of items that have very few ratings.

The MovieLens dataset with approximately 20 million ratings by 138 thousand users for 27 thousand items could be processed as a whole and therefore did not need to be subsetted. The train set contains approximately 16 million ratings (80 %) by 112 thousand users for 124 thousand items and the test set 4 million ratings (20 %) by 32 thousand users for 26 thousand items. The Netflix dataset contains approximately 480 thousand users and 18 thousand items. For the Netflix dataset, the train set is subsetted to a set of 120 thousand users and 17 thousand items and the test set is subsetted to a set of 100 thousand users and 17 thousand items. The subsetted train set contains approximately 24 million ratings and the test set 6 million ratings, retaining the 80-20 ratio.

The Amazon dataset has a far lower rating density compared to the MovieLens and Netflix datasets with approximately 2.4 million ratings by 1 million users for 200 thousand items. For the Amazon dataset, the train set is subsetted to approximately 300 thousand users and 45 thousand items and the test set is subsetted to 100 thousand users and 12 thousand items. The subsetted train set contains approximately 500 thousand ratings and the subsetted test set 130 thousand, roughly retaining this 80-20 rating. This shows the importance of rating density for the used datasets where the hardware capabilities allow the CF models to be trained on 16 million ratings for the MovieLens dataset, 24 million ratings for the Netflix dataset but only 0.5 million ratings for the Amazon dataset. Training the CF models on a lower number of ratings could result in poorer performances since they have limited information to make accurate predictions.

5.2 Machine learning models

In this section, the methods employed in this research to address the research questions are elaborated. This section includes a thorough explanation of the techniques applied to conduct the analysis and obtain the results. Section 6.5 displays a detailed demonstration to clarify these methods.

5.2.1 Item-based neighbourhood approach with item clustering

In light of sub-question (a), an item-based neighbourhood approach is applied to predict $\hat{r}_{ui}$ according to equation 7. The k-nearest neighbour approach is used as item-based neighbourhood approach where the values of b_u and b_i are estimated according to equation 5 by using a stochastic gradient descent method for this baseline estimation. The regularization parameter λ_1 from equation 5 is set to 0.005, in line with Koren et al. (2021). Furthermore, 10-fold cross validation is performed on the train set to determine the optimal number of nearest neighbours (k_{kNN}). According to Bell et al. (2007), typical values for k_{kNN} lie between 40 and 100 neighbours while Koren et al. (2021) argue that typical values of k_{kNN} lie in the range of 20 to 50. To incorporate both, in this research k_{kNN} is cross validated for the value set of (20, 40, 60, 80 and 100). More values of k_{kNN} could not be tested due to it being too intensive for the hardware. The optimal value of k_{kNN} that results in the lowest RMSE and MAPE values is used by the neighbourhood approach to recommend movies for all users in the test set. The RMSE and MAPE values for each dataset are stored.

Considering sub-question (a), item clustering is applied to the previously constructed neighbourhood approach model. The CF model that combines the item-based neighbourhood approach and item clustering is based on equation 7 where clustering sets the neighbourhood $S(i;u)$ of nearest neighbours to the cluster of the particular item. Item clustering is performed by applying k-means clustering and hierarchical clustering separately to the train set to group similar items together per cluster. The implementation of the separate clustering methods are relatively similar and in line with their concepts discussed in Section 2.3. To best determine the number of clusters for k-means, namely k_{kmeans}, the silhouette score is calculated for each different number of clusters and the value of k_{kmeans} with the highest silhouette score is selected. Similar to the values used by Sarwar et al. (2002) to determine k_{kmeans} for the MovieLens dataset, the silhouette score is determined for values of k_{kmeans} from 10 to 100 clusters with steps of 10, including the possibility of 1 cluster. K-means clustering iteratively assigns the items to the nearest centroids by updating the centroid centres until the position of each data point no longer changes. For hierarchical clustering an agglomerative nature is used. The bottom-up

technique starts with each data point as individual cluster and iteratively merges the closest pairs of clusters together. The ward linkage method is applied for this agglomerative HC which minimizes the within-cluster variance and aims to create compact clusters. Hierarchical clustering has the ability to determine the number of clusters itself and therefore does not need to pre-define k_{HC}.

After k-means clustering or hierarchical clustering have assigned all items to the optimal number of clusters, each item is assigned a cluster label based on the cluster they belong to. The neighbourhood approach algorithm is modified such that it takes these item clusters into account according to equation 7. So, only the items in the same cluster as the target item are considered in the neighbourhood rather than looking through the complete user dataset for nearest neighbours. The RMSE and MAPE values for this item-based neighbourhood approach with item clustering for both the train as test set are stored.

5.2.2 Latent factor models

In light of sub-question (b), two different latent factor models are constructed following equation 9. In this research, SVD and NMF are used as latent factor models. SVD decomposes the user-item train data into three separate matrices that capture the underlying latent factors f and reconstructs the original matrix. Since the input data for all three datasets only contains non-negative ratings, as shown in Section 4, NMF can be used as latent factor model. NMF decomposes the original train data into two non-negative matrices with dimensions affected by the number of factors f and also reconstructs the original matrix, as described in Section 2.4 and equation 3. The dimensionality reduction performed by SVD and NMF, results in f latent factors. In line with equation 9, the CF models utilize these latent factors to provide rating predictions.

For the SVD and NMF model, the number of latent factors f needs to be pre-defined. 10-fold cross validation is used separately for the train set to determine the optimal number of factors f and the optimal value of the regularization term λ, shown as λ_2 in equation 10. Where Koren et al. (2021) use 10-fold cross validation for the value set of (10, 20, 50, 100 and 200) for f, Koren (2008) uses the value set of (50, 100 and 200). To combine both papers, in this research the number of latent factors f for SVD and NMF are separately 10-fold cross validated over the value set of (20, 50, 100 and 200). The value of regularization term λ for SVD and NMF is separately 10-fold cross validated over the values (0.01, 0.02 and 0.05) which is in line with Koren et al. (2021) who use the value of 0.02 for this regularization term λ and Bell and Koren (2007) who set λ to 0.05. More values for f and λ could not be tested due to it being too intensive

for the hardware. The optimal values of the parameters are used by the SVD or NMF model separately to predict movie ratings according to equation 9. The RMSE and MAPE values for the train and test set are stored.

5.2.3 Ensemble methods

Considering sub-question (c), different CF models are combined. The item-based neighbourhood approach is combined with the most beneficial latent factor model. Also, the item-based neighbourhood approach with the most beneficial item clustering technique is combined with the most beneficial latent factor model to calculate the predicted ratings $\hat{r}_{ui}$ according to equation 11. The merging of these combination models is performed by three different ensemble methods, namely bagging, boosting and XGBoost. These ensemble methods use as train data the predicted item ratings from each individually implemented CF model from the train set and the actual ratings from the train set. Since clustering is an unsupervised method and not a CF model but an extension, when combining the item-based neighbourhood approach with clustering and a latent factor model, the ensemble methods use the predictions of the item-based approach with clustering and the predictions of the latent factor model. This enables the execution of equation 11 for combining the individual CF techniques. For example, for the merging of the item-based neighbourhood approach with k-means clustering and SVD, the ensemble method would use the observed rating values, the predicted ratings of the item-based neighbourhood approach with k-means clustering of the train set and the predicted ratings of SVD of the train set, as its training data. The ensemble methods are trained on this train data by subsequently (boosting and XGBoost) or separately (bagging) fitting decision trees to this data in order for them to make predictions.

For bagging, boosting and XGBoost, the values for the number of trees B is 5-fold cross validated on the train set for the values of 50, 100 and 200 trees, in line with James et al., 2013. For boosting and XGBoost, the value for the interaction depth s and the shrinkage rate λ, shown in Section 3.4, are 5-fold cross validated on the train data for the value sets of (1, 3, 5) and (0.1, 0.01, 0.001) respectively, in lines with James et al., 2013. More folds for the cross validation and more values for B, s or λ could not be tested due to it being too intensive for the hardware. Moreover, XGBoost makes use of L2 regularization by enabling the shrinkage of coefficients towards zero but never to exactly zero. Since L2 keeps all features in the model by not shrinking any of their coefficients to exactly zero, it is more suitable than L1 regularization because there is no evidence that any features are irrelevant.

The trained models are evaluated on the test set which consists of the observed ratings and predicted ratings from each individual CF model from the test set. The RMSE and MAPE values of the predicted ratings for the different merged CF models by the ensemble methods are stored.

5.3 Conceptual framework

To provide a thorough understanding of the methodology used in this research, a conceptual framework flowchart covering all the individually implemented and combined methods applied in this study, is provided in Figure 5.1. The flowchart gives a visual depiction of how the different approaches relate to each other and contribute to answering the three different sub-questions and eventually the main research question. As Figure 5.1 shows, for each dataset a total of 11 methods will be used to predict ratings, indicated by the 11 green stars, and evaluated on the test set in Section 6.

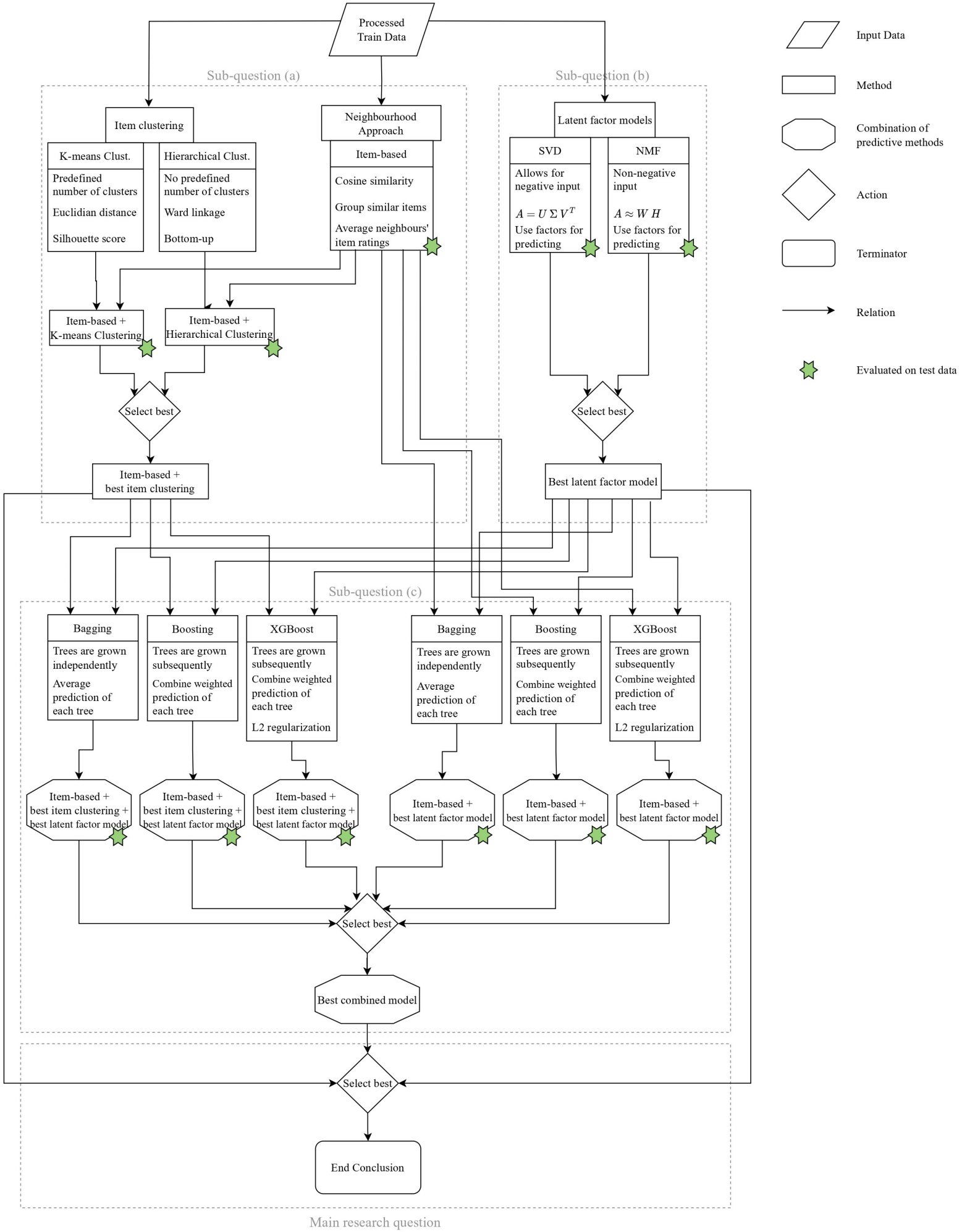

Figure 5.1: Conceptual framework in answering the sub-questions and main research question for each dataset

6 Results

Table 6.1 displays the performance accuracies and computational complexities of predicting item ratings from the MovieLens, Netflix Prize Competition and Amazon book datasets for the in-sample train data (IS) and the out-of-sample test data (OOS) with the methods as discussed in Section 5.2, namely an item-based neighbourhood approach (NbhdApp); NbhdApp with item clustering by k-means clustering or hierarchical clustering; and the latent factor models SVD and NMF. The performance accuracies of the CF models are compared based on the OOS data instead of the IS data, since the models are not exposed to unseen data during training. Comparing OOS results is a more reliable indicator of how well the models generalize to unseen data and reflect handle real-world scenarios. The running times provided in Table 6.1 display the running time of the model without cross validation to enable a fair comparison between models since a different amount and combination of parameter settings are cross validated between the models.

Table 6.1: Performance metrics of individually implemented CF models for each dataset

	CF Model	RMSE IS (MAPE)	RMSE OOS (MAPE)	Running Time
MovieLens	NbhdApp	0.705 (22.31%)	0.939 (34.03%)	3h
	NbhdApp & K-means	0.700 (22.20%)	0.936 (33.55%)	2h
	NbhdApp & Hier. Clust.	0.700 (22.20%)	0.938 (34.02%)	3h
	SVD	0.550 (16.68%)	0.931 (32.20%)	1.5h
	NMF	0.855 (29.33%)	1.010 (36.96%)	2.5h
Netflix	NbhdApp	0.725 (21.41%)	1.027 (31.44%)	5.5h
	NbhdApp & K-means	0.721 (21.29%)	1.026 (31.25%)	3.5h
	NbhdApp & Hier. Clust.	0.721 (21.29%)	1.026 (31.25%)	4.5h
	SVD	0.603 (17.12%)	1.019 (30.03%)	2h
	NMF	0.915 (29.02%)	1.081 (33.68%)	3h
Amazon	NbhdApp	0.208 (1.66%)	1.165 (36.68%)	10min
	NbhdApp & K-means	-	-	-
	NbhdApp & Hier. Clust.	0.235 (2.56%)	1.171 (38.94%)	12min
	SVD	0.535 (15.56%)	1.165 (38.68%)	1min
	NMF	0.200 (2.50%)	1.188 (39.60%)	6min

6.1 Item-based neighbourhood approach with item clustering

Table 6.2: Tuned parameter values for NbhdApp, k-means and hierarchical clustering of each dataset

Parameter	MovieLens	Netflix	Amazon
k_{kNN}	20	20	20
k_{kmeans}	10	10	1
k_{HC}	20	40	2

Sub-question (a) of this research is the following:

"Which item clustering technique enhances the performance accuracy and the computational expense of an item-based collaborative filtering model most?"

To answer sub-question (a), the performances accuracies and computational expenses of the item-based neighbourhood approach (NbhdApp), the combination of NbhdApp with k-means clustering (NbhdApp & K-means) and the combination of NbhdApp with hierarchical clustering (NbhdApp & Hier. Clust.) are compared per dataset.

The results for the different values provided in Table 6.1, are based on the models with their optimal parameter settings which are provided in Table 6.2. As mentioned before, the value of parameter k_{kNN} is 10-fold cross validated for the values 20 to 100 with steps of 20. For the MovieLens, Netflix and Amazon datasets, the optimal value of k_{kNN} equals 20 neighbours. Between the different datasets it can be seen in Table 6.1 that the NbhdApp for Amazon took the least amount of time for the predictions (10min) compared to the most amount of time for the Netflix dataset (5.5h). This difference is the result of the differences in size of the datasets where the Netlix data for IS and OOS contains approximately 30 million ratings and the IS and OOS Amazon datasets contains merely 630 thousand ratings (Section 4). Moreover, the RMSE IS and OOS for the NbhdApp methods show for all three datasets that the NbhdApp performs better on the train set than the test set, which is expected since they are trained on the IS data. However, for the Amazon dataset the in-sample performance accuracies for the NbhdApp (0.208 and 1.66%) is substantially lower than OOS (1.165 and 36.68%) showing signs of overfitting since the model performs well on the training data but poorly generalizes to new or different data, the test data.

Since the clustering methods are additions to the NbhdApp, the optimal value of k_{kNN}, 20, is used for the NbhdApp models with clustering additions. Furthermore, the optimal value for k_{kmeans} is determined based on the silhouette score. For k_{kmeans} the silhouete scores are

calculated for the values of 10 to 100 clusters with steps of 10, including the possibility of 1 cluster, for NbhdApp & K-means. These silhouette scores are provided in Section 9 and the number of clusters with the highest silhouette score is selected. The hierarchical clustering model is able to determine the optimal value of k_{HC} itself.

For the MovieLens dataset, Table 6.2 shows that the optimal value of k_{kmeans} equals 10 clusters and k_{HC} equals 20 clusters. Table 6.1 shows that the addition of k-means clustering or hierarchical clustering to the NbhdApp for the MovieLens dataset is slightly beneficial since the OOS performance accuracies of the NbhdApp with either clustering technique (0.936 and 33.55% for k-means, 0.938 and 34.02% for HC) are lower than those of the NbhdApp without clustering (0.939 and 34.03%). The computational expense of the NbhdApp did not benefit from the addition of hierarchical clustering since it remained 3 hours. It did improve by the addition of k-means clustering, resulting in a running time of 2 hours. Hence, in light of sub-question (a), by incorporating k-means clustering the performance accuracy and computational expense of the NbhdApp benefit most, for the MovieLens dataset. This is in line with the conclusions presented in prior research by Gong (2010) and Sarwar et al. (2002).

For the Netflix dataset Table 6.2 displays that k_{kmeans} also equals 10 (Section 9) and k_{HC} equals 40 clusters. The results in Table 6.1 show that the addition of k-means clustering to the NbhdApp results in the same IS and OOS performance accuracies as adding HC clustering to the NbhdApp for the Netlix dataset. Nevertheless, the addition of either clustering method to the NbhdApp is beneficial for the performance accuracy since the OOS RMSE values are slightly lower than not including clustering (1.026 compared to 1.027). So, for the Netflix dataset, the predictions of the NbhdApp with either clustering technique, on average, deviate from the actual values by approximately 1.026 stars, since the ratings are provided in stars. The computational expense of the NbhdApp model benefits from the addition of both k-means or hierarchical clustering. However, since the addition of k-means is more computationally efficient than hierarchical clustering (3.5 compared to 4.5 hours), it can be concluded, in light of sub-question (a), that the performance accuracy and computational expense of the NbhdApp benefit most from the addition of k-means clustering. This is in line with the conclusions presented in prior research by Gong (2010) and Sarwar et al. (2002).

For the Amazon dataset, k-means clustering could only create one cluster, so the value of k_{kmeans} resulted in only one cluster, as shown in Table 6.2. No silhouette scores could be calculated for different cluster amounts. Since the clustering method suggests that the data points could not be effectively separated into distinct groups or clusters, k-means clustering has no additional value to the NbhdApp and is therefore omitted from Table 6.1. This lack

of distinct clusters indicates the complexity of the Amazon data since it does not possess clear inherent cluster structures. Nevertheless, the hierarchical clustering method set k_{HC} to the optimal value of 2 clusters, shown in Table 6.2. The k_{HC} value of merely 2 clusters could also possibly be explained by the data complexity. Furthermore, similar to the NbhdApp, the NbhdApp combined with hierarchical clustering shows strong signs of overfitting in the results of Table 6.1 with much lower IS RMSE and MAPE values compared to OOS, namely 0.235 and 2.56% compared to 1.171 and 38.94%. This shows that the predictions of the NbhdApp combined with hierarchical clustering, on average, are within ± 2.56% of the observed values in the train set but within ± 38.94% of the observed values in the test set. Lastly, the performance accuracy of the NbhdApp did not benefit from the addition of hierarchical clustering, with higher OOS RMSE and MAPE values of 1.171 and 38.94% when HC is included compared to 1.165 and 36.68% without HC. The computational expense increases by the addition of clustering (12 compared to 10 minutes). Thus, considering sub-question (a), the performance accuracy and computational expense of the NbhdApp do not benefit from either k-means or hierarchical clustering. This diverges from the conclusions presented in prior research by Gong (2010) and Sarwar et al. (2002).

6.2 Latent factor models

Table 6.3: Tuned parameter values for SVD and NMF of each dataset

	MovieLens		**Netflix**		**Amazon**	
CF Model	λ	f	λ	f	λ	f
SVD	0.01	200	0.01	200	0.01	200
NMF	0.05	20	0.05	20	0.05	20

Sub-question (b) of this research is the following:

"Which latent factor model achieves the highest level of performance accuracy and computational efficiency?"

To be able to answer sub-question (b), the performance accuracies and computational expenses of the SVD and NMF models are compared per dataset which are provided in Table 6.1. These results are based on the models with their optimal parameter settings which are shown in Table 6.3. For the SVD and NMF models, the values of λ_2 from equation 10 and the number of latent factors f are 10-fold cross validated for each dataset. For the SVD model of all three datasets, the values of the regularization parameter λ and the number of factors f are

the same, 0.01 and 200 respectively. With a low value of λ the SVD model is less constrained by regularization and tends to fit the training data closely whereas a high number of factors f enables the SVD model to take into account more complex relationships and finer features in the data. Both values could thus lead to signs of overfitting. For the NMF model of all three datasets, the values of λ and f are also the same, 0.05 and 20 respectively. Here, the regularization parameter λ has a high value implying that the NMF model has more stringent constraints and is penalised for complex solutions whereas the low number of factors f of the NMF model results in a simplified representation since it captures fewer underlying patterns or dimensions. This combination of parameters facilitates a more comprehensible and broadly applicable depiction of the data by the NMF model. It is interesting that the optimal values for λ and f are the complete opposite for SVD and NMF from the possible parameter values they are tuned for. There is no clear explanation for this. Possibly, the SVD models could emphasize more on performance accuracy whereas the NMF models could focus more on interpretability.

The RMSE and MAPE values provided in Table 6.1 show that again, as expected, the running time of the SVD and NMF models are highest for the Netflix dataset and lowest for the Amazon dataset due to the dataset sizes. For the MovieLens and Netflix dataset the IS RMSE and MAPE values are considerably lower than the OOS values, hinting at the possibility of overfitting which, for SVD, could be caused by the low λ and high f values. For the Amazon dataset, the IS performance accuracies of the SVD (0.535 and 15.56%) and especially the NMF model (0.200 and 2.50%) are substantially lower than their OOS values (1.165 and 38.68% for SVD, 1.188 and 39.60% for NMF). This shows strong signs of overfitting of the SVD and NMF model.

For the MovieLens dataset, Table 6.1 shows that the OOS RMSE and MAPE values of the SVD model (0.931 and 32.20%) are lower than those of the NMF model (1.010 and 36.96%). Also, the computational expense of the SVD model (1.5 hours) is lower than for the NMF model (2.5 hours) for the MovieLens dataset. Thus, in sight of sub-question (b), the latent factor model SVD achieves the highest level of performance accuracy and computational efficiency compared to NMF. Similarly, for the Netflix and Amazon datasets the OOS RMSE and MAPE values are lower for the SVD model than for the NMF model (1.019 and 30.03% compared to 1.081 and 33.68% for the Netflix dataset; and 1.165 and 38.68% compared to 1.188 and 39.60% for the Amazon dataset) and the computational expenses of the SVD models are lower than for the NMF model (2 hours compared to 3 hours for the Netflix dataset; and 1 minute compared to 6 minutes for the Amazon dataset). Thus, considering sub-question (b), also for the Netflix and Amazon datasets, SVD achieves a higher level of performance accuracy and computational

efficiency compared to NMF. SVD being the preferred latent factor model is in line with the conclusions presented in prior research by Koren (2009) and Koren et al. (2021).

6.3 Ensemble methods

Table 6.4: Performance metrics of CF model combinations merged by ensemble methods for each dataset

	Ensemble	Combined CF model	RMSE IS (MAPE)	RMSE OOS (MAPE)	Running Time
MovieLens	Bagging	NbhdApp & SVD	0.491 (15.43%)	1.115 (39.93%)	3h
		NbhdApp & K-means & SVD	0.482 (15.22%)	1.109 (39.26%)	3.5h
	Boosting	NbhdApp & SVD	1.060 (38.01%)	1.020 (37.56%)	1h
		NbhdApp & K-means & SVD	1.060 (38.01%)	1.020 (37.55%)	1h
	XGBoost	NbhdApp & SVD	1.060 (38.01%)	1.020 (37.54%)	2min
		NbhdApp & K-means & SVD	1.060 (38.00%)	1.021 (37.58%)	2min
Netflix	Bagging	NbhdApp & SVD	0.506 (14.42%)	1.207 (35.79%)	5h
		NbhdApp & K-means & SVD	0.493 (14.14%)	1.185 (35.21%)	5h
	Boosting	NbhdApp & SVD	1.084 (33.80%)	1.091 (34.20%)	1.5h
		NbhdApp & K-means & SVD	1.084 (33.80%)	1.091 (34.19%)	1.5h
	XGBoost	NbhdApp & SVD	1.084 (33.80%)	1.091 (34.19%)	8min
		NbhdApp & K-means & SVD	1.084 (33.79%)	1.091 (34.20%)	8min
Amazon	Bagging	NbhdApp & SVD	0.564 (16.65%)	1.234 (40.58%)	25min
		NbhdApp & Hier. Clust. & SVD	0.554 (16.46%)	1.288 (41.07%)	30min
	Boosting	NbhdApp & SVD	1.195 (40.04%)	1.188 (39.59%)	20min
		NbhdApp & Hier. Clust. & SVD	1.195 (40.04%)	1.188 (39.59%)	25min
	XGBoost	NbhdApp & SVD	1.195 (40.04%)	1.188 (39.52%)	15min
		NbhdApp & Hier. Clust. & SVD	1.195 (40.04%)	1.188 (39.52%)	15min

In light of sub-question (a) the results in Table 6.1 have shown that k-means clustering is a beneficial addition to the item-based neighbourhood approach for the MovieLens and Netflix datasets whereas no clustering method is a beneficial addition for the Amazon dataset. Considering sub-question (b), the results in Table 6.1 have shown that SVD achieves a higher level of performance accuracy and computational efficiency compared to NMF, for all three datasets.

Since the objective of this research is to enhance the performance accruacy and computational efficiency of the collaborative filtering model, the most beneficial item clustering model and latent factor model are combined with the NbhdApp per dataset. So, for the MovieLens and Netflix datasets a combination of NbhdApp and SVD (NbhdApp & SVD) and a combination of NbhdApp with k-means clustering and SVD (NbhdApp & K-means & SVD) are constructed. Their performance metrics are displayed in Table 6.4. For the Amazon dataset, the combination of NbhdApp and SVD is also evaluated and even though no addition of item clustering technique is beneficial for the NbhdApp of the Amazon dataset, the combination of NbhdApp with hierarchical clustering and SVD (NbhdApp & Hier. Clust. & SVD) is evaluated to asses the overall impact of hierarchical clustering on the combined model.

Sub-question (c) of this research is the following:

"Which combination of an item-based collaborative filtering model with item clustering and a latent factor model acquires the best performance accuracy and computational expense, and which method is best used to combine them?"

To be able to answer sub-question (c), three ensemble methods are used and evaluated to combine the CF models, namely bagging, boosting and XGBoost. For all ensemble methods, the parameter value for the number of trees B is tuned by 5-fold cross validation and for the boosting and XGBoost models the interaction depth s and the shrinkage rate λ are also 5-fold cross validated. The optimally tuned parameter values are provided in Table 6.5.

Table 6.5: Tuned parameter values for the ensemble methods for each combined CF model of each dataset

		MovieLens			**Netflix**			**Amazon**		
Ensemble	**Combined CF model**	B	s	λ	B	s	λ	B	s	λ
Bagging	NbhdApp & SVD	200	-	-	200	-	-	200	-	-
	NbhdApp & Clustering & SVD	200	-	-	200	-	-	200	-	-
Boosting	NbhdApp & SVD	50	1	0.01	50	1	0.001	200	1	0.001
	NbhdApp & Clustering & SVD	50	1	0.01	50	1	0.001	200	1	0.001
XGBoost	NbhdApp & SVD	100	1	0.1	100	1	0.1	100	1	0.1
	NbhdApp & Clustering & SVD	100	1	0.1	100	1	0.1	100	1	0.1

The results in Table 6.5 show that the optimally tuned parameter values for each ensemble method do not differ between the NbhdApp & SVD model and the NbhdApp & Clustering & SVD. So, despite the differences in these two combined CF models, the ensemble methods converged to the same optimally tuned parameter values. Moreover, the optimally tuned parameter values show that bagging generally uses more trees B than the other ensemble methods, for each dataset. By combining a high number of weak learners, bagging could potentially achieve higher performance efficiency. Furthermore, the interaction depth s for all combined CF models and datasets is 1 for boosting and XGBoost indicating that each decision tree in the boosting and XGBoost process considers only one feature at a time. This interaction depth s of 1 emphasizes the focus on simple relationships in the data. Lastly, the results in Table 6.5 show that boosting uses a relatively lower shrinkage rate λ compared to XGBoost (0.01 and 0.001 compared to 0.1) for all datasets, indicating that each decision tree has a stronger influence on the final prediction for boosting compared to XGBoost.

Given these optimally tuned parameter values for the ensemble methods, the IS and OOS performance accuracies and computational expenses of the ensemble methods for each combination of collaborative filtering models per dataset are provided in Table 6.4. Again, it can be seen that the CF models for the Netflix dataset need the most time to run compared to the Amazon dataset with the least amount of time due to the data sizes. Also, the computational expenses of the CF models approximately remains the same after adding item clustering to the item-based approach (k-means for MovieLens and Netflix, hierarchical clustering for Amazon). This is as expected, since the number of data points used by the ensemble methods does not increase or decrease. By either combining the predictions of an item-based approach and the predictions of a latent factor model or by combining the predicitons of an item-based approach with clustering and the predictions of a latent factor model, the amount of training and test data used by the ensemble methods does not change. Moreover, for the three datasets, the results in Table 6.4 show that the IS performance accuracies for the models combined by bagging is substantially lower than their OOS values whereas the performance accuracies for the models combined by boosting or XGBoost are similar to their OOS values. This substantial drop in RMSE and MAPE values between the OOS and IS data from the bagging model shows strong signs of overfitting which could potentially be caused by the high number of fitted decision trees B, as displayed in Table 6.5, compared to the boosting and XGBoost methods. Furthermore, the performance accuracies IS and OOS of the boosting and XGBoost models do not differ much even though their optimal parameter settings do differ, as shown in Table 6.5.

For the MovieLens dataset, the results in Table 6.4 show that for bagging the OOS performance accuracy improves when k-means is added to the combined CF model (1.109 and 39.26% compared to 1.115 and 39.93%), for boosting the OOS performance accuracy does not differ much and for XGBoost the OOS performance accuracy is slightly higher when k-means is merged with the combined model. The CF model that combines NbhdApp and SVD with XGBoost acquires the lowest OOS RMSE and MAPE values of 1.020 and 37.54% and the lowest running time of 2 minutes. Hence, in light of sub-question (c), for the MovieLens dataset the combination of an item-based neighbourhood approach and SVD acquires the best performance accuracy and computational expense and XGBoost is the best model to combine them.

For the Netflix dataset, the results in Table 6.4 display that for bagging the performance accuracy improves when k-means is added to the combined CF model (1.185 and 35.21% compared to 1.207 and 35.79%). The performance accuracies for the boosting and XGBoost methods do not differ much when adding clustering. The lowest OOS RMSE and MAPE value are acquired when boosting combines NbhdApp with k-means and SVD or when XGBoost combines NbhdApp and SVD (RMSE of 1.091 and MAPE of 34.19%). Between these two models, XGBoost combining NbhdApp and SVD uses the least amount of running time (only 8 minutes compared to 1.5 hours). Hence, considering sub-question (c), the combination of an item-based neighbourhood approach and SVD acquires the best performance accuracy and computational expense and XGBoost is the best method to combine them.

For the Amazon dataset, the OOS performance accuracy of the bagging model decreases when including hierarchical clustering. The performances of the boosting and XGBoost model do not differ when adding clustering. The best OOS performance accuracy is acquired by using XGBoost to either combine NbhdApp and SVD or to combine NbhdApp with hierarchical clustering and SVD. The running time of both these models is 15 minutes, which is lower than for the other ensemble methods. Hence, in view of sub-question (c), both the combinations of an item-based neighbourhood approach and SVD or an item-based neighbourhood approach with hierarchical clustering and SVD acquire the best performance accuracy and computational expense and XGBoost is the best method to combine either of them.

6.4 Individual and combined CF models

The previous results have shown for each dataset, the addition of which item clustering technique to an item-based neighbourhood approach is most beneficial for the performance accuracy and computational expense (sub-question (a)), which latent factor model achieves the highest level of performance accuracy and computational efficiency (sub-question (b)) and which combination of

item-based collaborative filtering model with item clustering and a latent factor model acquires the best performance accuracy and computational expense, and which method is best used to combine them (sub-question (c)). Here, the main research question is restated to reinforce the primary focus of this study and provide a clear connection to the analysis from the sub-questions:

"To what extent does the combination of an item-based neighbourhood approach with item clustering and a latent factor model enhance the performance accuracy and computational efficiency of the recommendation system compared to the individually implemented collaborative filtering methods?"

For the MovieLens dataset, the combination of NbhdApp and SVD merged by XGBoost acquires the best performance accuracy and computational expense of the combined CF models. The results in Table 6.4 show that the performance accuracy of this model is represented by an OOS RMSE value of 1.020 and MAPE value of 37.54% which is less accurate than the best individual CF model for the MovieLens dataset, namely SVD with an OOS RMSE value of 0.931 and MAPE value of 32.20%, as shown in Table 6.1. Moreover, the ensemble models predict based on the predictions from the individual CF models indicating that the XGBoost model will necessarily have a longer running time than any of the individual implementations. Thus, in view of the main research question, for the MovieLens dataset the individual implementation of SVD enhances the performance accuracy and computational efficiency more than a combination of NhbdApp with clustering and a latent factor model.

For the Netflix dataset, the combination of NbhdApp with SVD acquires the best performance accuracy and computational expense merged by XGBoost with an OOS RMSE value of 1.091 and MAPE value of 34.19%, shown in Table 6.4. This performance accuracy is worse than the individual implementation of SVD for the Netflix dataset, namely an RMSE value of 1.019 and MAPE value of 30.03% as shown in Table 6.1. Also, SVD as individual model uses less running time than one of the ensemble methods. Hence, considering the main research question, also for the Netflix dataset the individual implementation of SVD enhances the performance accuracy and computational efficiency more than a combination of NhbdApp and a latent factor model.

The results for Amazon differ from the other datasets. The individual implementations of the CF models show substantially stronger signs of overfitting. K-means clustering suggests that the data points could not be effectively separated into clusters and hierarchical clustering could cluster the data points into merely 2 clusters. The combination of either NbhdApp and SVD or NbhdApp with hierarchical clustering and SVD merged by XGBoost acquires the best performance accuracy of an RMSE value of 1.188 and MAPE value of 39.52%. This combination

is however outperformed by the individual implementations of NbhdApp and SVD, both with an RMSE value of 1.165 and MAPE value of 38.68%. SVD outperforms the NbhdApp model in terms of computational efficiency since it takes 1 minutes compared to 10 mintues. So, in sight of the main research question, for the Amazon dataset, the individual implementation of SVD enhances the performance accuracy and computational efficiency more than a combination of NhbdApp with clustering and a latent factor model.

6.5 Prediction example

Table 6.6: Predicted ratings by individually implemented and combined CF models for user 1385 rating movie 2452 from the Netflix dataset, with an observed rating of 5

Ensemble	**CF model**	**Predicted Rating**
	NbhdApp	4.34
	NbhdApp & K-means	4.41
	NbhdApp & Hier. Clust.	4.41
	SVD	4.25
	NMF	4.18
Bagging	NbhdApp & SVD	3.40
	NbhdApp & K-means & SVD	3.50
Boosting	NbhdApp & SVD	3.58
	NbhdApp & K-means & SVD	3.58
XGBoost	NbhdApp & SVD	3.58
	NbhdApp & K-means & SVD	3.59

As an illustration, one user-item pair is selected from the test set where the predictions of each collaborative filtering model, individual or combined implementation, are provided in Table 6.6. The user-item pair is selected from the test set to gather more reliable evaluations of the CF models. The user from this example has ID 1385 and has given the movie with ID 2452 from the Netflix dataset a rating of 5 stars. This movie ID corresponds to the movie titled “Lord of the Rings: The Fellowship of the Ring” which is the first movie in the Lord of the Rings trilogy and is known to have genres such as action, fiction and adventure. User 1385 is randomly selected from the users in the test set who have rated “Lord of the Rings: The Fellowship of the Ring”. With an actual rating of 5 stars, the user highly appreciated the movie. The CF methods used

to predict the rating of this user-item pair from the Netflix dataset, use their optimal parameter values, as provided in Tables 6.2, 6.3 and 6.5. In this example it can be seen that the individual implementations of the collaborative filtering model predict a rating closest to the actual rating, namely NbhdApp with or without clustering, SVD and NMF. The combined CF models predict a lower rating compared to the actual rating.

This prediction example can be used to explain the inner workings of some of the collaborative filtering methods. The item-based neighbourhood approach identifies the 20 most similar neighbours ($k_{kNN} = 20$) for the target item with ID 2452 by calculating the cosine similarity as presented in equation 6 between the target item and all other items and selecting the 20 items with the highest similarity. As an illustration, the first 5 of these 20 neighbours are the movies "Lord of the Rings: The Two Towers", "Lord of the Rings: The Return of the King", "Pirates of the Caribbean: The Curse of the Black Pearl", "The Matrix" and "Spider-Man". The data used by the NbhdApp only consists of the item ID, user ID and date, see Section 4.2. Without knowing the movie title or genres it computes that the most similar movies to "Lord of the Rings: The Fellowship of the Ring" are the second and third movies in the trilogy, who evidently are similar types of movies. Moreover, the NbhdApp computes that a "Pirates of the Caribbean" movie, "The Matrix" and "Spider-Man" are similar to the target movie. These three movies are known to be similar to the target item since they are known to have genres such as action, adventure and fiction. Next, the NbhdApp uses all 20 neighbours with equation 7 to predict the rating of user 1385 for item 2452, which resulted in a predicted rating of 4.34.

The addition of k-means or hierarchical clustering reduces the set of all items for which the NbhdApp needs to find 20 similar neighbours by clustering the items beforehand. The NbhdApp with clustering needed to search for 20 similar neighbours in the cluster in which item 2452 is placed, while the individual implementation of NbhdApp needed to search through all items in the dataset. The addition of k-means and hierarchical clustering resulted in the same predicted rating for this user-item pair of 4.41 by using equation 7. The SVD and NMF models use 200 and 20 latent factors respectively for the Netflix dataset, for predicting the rating of user 1385 for movie 2452. SVD and NMF construct these latent factors according to equations 2 and 3. These factors can have several different meanings and perceptions and the interpretation of these factors is out of the scope of this research. These factors are used in equation 9 in order to make rating predictions. The SVD model predicts a rating of 4.25 and the NMF model predicts a rating of 4.18 for this user-item pair.

Lastly, the ensemble methods use the predicted values of the methods they combine to predict the rating. For example, the bagging model that combines NbhdApp and SVD uses the observed values from the train set, the predicted rating values of NbhdApp and predicted ratings of SVD in the train set as training data. Hence, the ensemble methods combine the individually implemented CF models with equation 11. How the decision trees are fit to the data, depends on the ensemble method, see Section 3.4. The bagging model, similar to the other ensemble methods, uses the predicted rating values of NbhdApp and predicted ratings of SVD to construct split nodes in the decision trees. The bagging method trains 200 decision trees, see Table 6.5, independently on different subsets of this training data by resampling with replacement. By aggregating the predictions of all these trees through averaging, it predicts a rating of 3.40 by user 1385 for item 2452 by combining NbhdApp and SVD and predicts a rating of 3.50 when combining NbhdApp with k-means clustering and SVD. Boosting subsequently fits 50 decision trees, each considering only one feature, to the model with a shrinkage rate of 0.001. Boosting predicts that user 1385 rates movie 2452 a value of 3.58 for both combined CF models. Lastly, XGBoost subsequently fits 100 decision trees to the model, each considering one feature with a shrinkage rate of 0.1. By combining the NbhdApp and SVD, XGBoost predicts a rating of 3.58 and by combining the NbhdApp with k-means clustering and SVD, XGBoost predicts a rating of 3.59, as shown in Table 6.6. The combined CF models predict a less accurate rating than the individual CF models.

7 Conclusion

The research question of this thesis is: *"To what extent does the combination of an item-based neighbourhood approach with item clustering and a latent factor model enhance the performance accuracy and computational efficiency of the recommendation system compared to the individually implemented collaborative filtering methods?"*. The results in this research for the MovieLens, Netflix and Amazon datasets have all shown that the individual implementation of SVD enhances the performance accuracy and computational efficiency more than a combination of NhbdApp with clustering and a latent factor model. The results have shown however that the different CF models and combination of CF methods are sensitive to rating densities and skewness in the data, performing best on datasets with high rating density and less skewness such as the MovieLens and Netflix dataset. For less dense and more skewed datasets like the Amazon dataset, the results of the CF methods are less robust since all individually implemented CF models have shown strong signs of overfitting and clustering techniques have little to no additional value due to the lack of distinct clusters.

This conclusion to the research question has been shown by the results for each sub-question. The first sub-question is: *"Which item clustering technique enhances the performance accuracy and the computational expense of an item-based collaborative filtering model most?"*. For the MovieLens and Netflix datasets, by incorporating k-means clustering the performance accuracy and computational efficiency of the item-based neighbourhood approach benefit most. The results for the Amazon dataset deviate from this since k-means suggests that no clusters could be made where hierarchical clustering only segmented the data points in 2 clusters and strong signs of overfitting are present for the item-based neighbourhood approach including and not including hierarchical clustering. For the Amazon dataset, the item-based neighbourhood approach did not benefit from either clustering methods.

The second sub-question is: *"Which latent factor model achieves the highest level of performance accuracy and computational efficiency?"*. For the MovieLens, Netflix and Amazon dataset SVD achieves the highest level of performance accuracy and computational efficiency compared to NMF where the latent factor model results for the Amazon dataset show strong signs of overfitting.

The third sub-question is: *"Which combination of an item-based collaborative filtering model with item clustering and a latent factor model acquires the best performance accuracy and computational expense, and which method is best used to combine them?"*. For the MovieLens and Netflix dataset, the combination of an item-based neighbourhood approach and SVD acquires

the best performance accuracy and computational expense and XGBoost is the best method to combine them. The results of the Amazon dataset deviate by showing that both the combinations of an item-based neighbourhood approach and SVD or an item-based neighbourhood approach with hierarchical clustering and SVD acquire the best performance accuracy and computational expense and XGBoost is the best method to combine either of them. Thus for all three datasets, XGBoost is considered the best ensemble method to combine CF methods but the best combination of methods differs between the MovieLens and Netflix dataset, and the Amazon dataset.

The less robust and deviating results from the Amazon dataset compared to the results from the MovieLens and Netflix datasets relate to the data sparsity issue of collaborative filtering models where datasets contain few ratings compared to the number of users and items. CF models recommend items based on limited data and encounter issues when predicting ratings for the Amazon dataset as a result of its low rating density and data sparsity. Also the CF models show strong signs of overfitting for the Amazon dataset. This could be caused by the high level of skewness in the data compared to the other datasets. Thus, collaborative filtering models are more suitable for datasets with high rating density and low skewness where SVD enhances the performance accuracy and computational expense of the recommender system the most, compared to both individual and combined CF models.

Thus, this research concludes that companies acquire the most accurate and time efficient predicted item ratings by applying SVD as collaborative filtering model, as long as the data upholds the restrictions of high rating density and low skewness.

8 Discussion

8.1 Limitations

A limitation of this research is that the models' performance accuracy confidence intervals were not estimated. As a result, the conclusions might not have adequately accounted for the uncertainty surrounding the given performance metrics. The RMSE and MAPE values provided in Tables 6.1 and 6.4, representing the performance accuracies of the different CF models, lie relatively close to each other in the results. The results of this research are less robust due to the absence of confidence intervals which limits a more thorough examination of the variability in each model's performance.

Also, a limitation of the conclusions drawn from this research is the presence of skewness in the MovieLens, Netflix and Amazon datasets. The datasets show skewed rating distributions with the Amazon dataset having a highly right-skewed rating distribution. Skewed rating distributions might result in misleading results. When one rating value dominates the dataset, it is possible that the collaborative filtering models are more likely to predict the majority rating value resulting in poorer performance of the less represented rating values, indicated by misleadingly low RMSE values. Skewness in the datasets used for this research could result in the CF models failing to generalize well to the less represented rating values, which increases the risk of overfitting. The results of this research substantiate this by showing strong signs of overfitting for the Amazon dataset which displays a high level of skewness compared to the other datasets.

Furthermore, the performances of the CF models are limited to certain data restrictions. The results of the collaborative filtering models for the Amazon dataset are less robust and deviate from the literature and from the other two datasets due to strong signs of overfitting and the inability to cluster the data point in a sufficient number of clusters. As mentioned before, this could be caused by a number of things, such as the low rating density or high level of skewness in the Amazon dataset. This low rating density and data sparsity complicates the accurate capturing of underlying patterns due to the small dataset size. As a result, the models' functioning is impaired, leading to fewer reliable outcomes. Thus, the individually implemented and combined CF models proposed in this research are limited to be preferably performed on datasets similar to the MovieLens and Netflix datasets with high rating densities and less data skewness.

8.2 Implications

8.2.1 Managerial perspective

The conclusions presented in this research paper show that from all tested CF methods, SVD predicts product ratings most accurately and time efficiently. Knowing that SVD predicts product ratings most accurately and time efficiently, is valuable from a managerial perspective because it enables companies to predict what kind of products are potentially preferred by their customers and therefore likely to be purchased, as shown by Kawaguchi et al. (2019). This is not only impactful for, for instance, the product sales but also for the effects it has on for example procurement decisions and inventory management (operational level), and on a strategic level such as a competitive advantage, enhanced customer retention and improved campaign strategies. Depending on the structure of the company's business model, the value of accurately predicting ratings can be explored further. For example, Netflix's profits are based on the number of subscriptions and could therefore benefit from accurate rating predictions by SVD to increase or sustain their number of subscriptions, whereas Amazon's profits are based on the revenues of each sold product on their marketplace and could increase the sales of their products with improved rating predictions.

Moreover, from a managerial perspective it is important that this research concluded that collaborative filtering models are sensitive to rating densities and skewness in the data, performing best on datasets with high rating density and low skewness, which is in line with the common issue of data sparsity for CF models. This implies that companies must ensure their data fits these restrictions in order to make optimal use of the CF models or their data should be tailored to align with these data restrictions.

The findings of this research can be extrapolated to a wide range of businesses where item ratings are available and uphold these data restrictions. For example, other streaming services such as the music streaming service Spotify or online review platforms such as TripAdvisor can improve their businesses on operational and strategic level by using this research's findings that SVD outperforms the tested CF models, to make relatively accurate and time efficient rating predictions.

8.2.2 Academic perspective

From an academic perspective, several findings from this research align with the conclusions drawn in prior research, as shown in Section 2. The fact that k-means clustering is less computational expense than hierarchical clustering is in line with the findings of Nielsen (2016).

Moreover, the results show that clustering is a beneficial addition to an item-based neighbourhood approach, in line with Gong (2010), the results show that SVD outperforms an item-based neighbourhood approach, similar to the findings of Koren (2009), and that XGBoost outperforms gradient boosting, similarly concluded by Carmona et al. (2019). However, the conclusion drawn from this research that the combination of an item-based neighbourhood approach with item clustering and a latent factor model merged by an ensemble method is not more beneficial than the individual implementation of SVD, is a novel addition to the literature. Furthermore, the results in this research have shown that CF models are better suited for datasets with high rating density and low skewness. Thus, the findings from this research contribute to the existing literature by concluding that combining CF models is less beneficial than the individual implementation of SVD and by underscoring the significance of datasets with high rating density and low skewness as crucial factors for optimal performance of collaborative filtering models.

8.3 Future research

The collaborative filtering methods used in this research make use of explicit feedback. Implicit feedback is data which indirectly reflects opinions through observing user behavior such as purchase history, browsing history, search patterns or even mouse movement (Koren et al., 2021). The MovieLens, Netflix and Amazon datasets provide a certain implicit feedback of which movies or products a user has rated regardless of how they rated it (Koren et al., 2021). By choosing to express his or her view and give a high or low rating, a user provides an implicit indication of his or her preferences. Several researches have incorporated this implicit feedback in the SVD model and dubbed this model as SVD++ (Koren, 2008, 2009; Koren et al., 2021). For SVD++, $N(u)$ is the set of all items for which user u provided an implicit preference. The authors state that a rating is predicted by this SVD++ model according to:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_i) \tag{18}$$

Here, in contrast to predictions made by SVD, equation 9, a second set of item factors that connects each item i to a factor vector $y_i \in \mathbb{R}^f$ is introduced. Users are characterized by these new item factors based on the variety of items they rated. User u is modeled as $p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_i$ where $|N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_i$ reflects the perspective of implicit feedback. Koren (2008) and Koren et al. (2021) have shown that SVD++ outperforms SVD in terms of performance accuracy.

Since the results of this research have shown that the individual implementation of SVD outperforms the combination of several CF models, the implementation of SVD++ could pos-

sibly outperform all individual and combined CF models, including SVD. This research paper has opted to concentrate on the effectiveness of traditional latent factor models and its combination with an item-based neighbourhood approach without adding the complexity of implicit feedback. Future research could investigate the combination of SVD++ with an item-based neighbourhood approach and user clustering with the use of ensemble methods. Future research endeavors could focus on the implementation of such a combination model with SVD++. It is suggested to combine these CF models similarly to equation 11, as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_i) + \frac{\sum_{j \in S^{k_{kNN}}(i;u)}(r_{uj} - b_{uj}) cos_{i,j}(\theta)}{\sum_{j \in S^{k_{kNN}}(i;u)} cos_{i,j}(\theta)} \tag{19}$$

Future research could evaluate the performance accuracy and computational expense of such a combination of an item-based neighbourhood approach with user clustering and SVD++ in various contexts and datasets, to determine whether this combination enhances the performance accuracy and computational efficiency of the collaborative filtering model compared to the individual implementations of SVD or SVD++.

In addition, this research roughly indicates the restrictions of a dataset for which the CF models are best suited, namely high rating density and low skewness. Future research could delve deeper into finding more specific and refined requirements for the datasets. Also, future research could investigate how to transform data with low rating density and/or high skewness such that they can be optimally used for the evaluation of CF models. Lastly, instead of having to transform the data, research could delve deeper into solving the general issues of CF models such as data sparsity to enable CF models to deal with low rating densities and/or high skewness in the data.

In summary, this research combines multiple collaborative filtering models with the use of ensemble methods and reveals that the individual implementation of SVD outperforms any combination of CF methods based on performance accuracy and computational expense. The findings underscore the significance of datasets with high rating density and low skewness as crucial factors for optimal performance of collaborative filtering models. Knowing that SVD predicts product ratings most accurately and time efficiently, is valuable from a managerial perspective because it enables companies to predict what kind of products are potentially preferred by their customers and therefore likely to be purchased.

References

Abdi, H. (2007). Singular value decomposition (svd) and generalized singular value decomposition. *Encyclopedia of measurement and statistics*, *907*, 912.

Aghdam, M. H., Analoui, M., & Kabiri, P. (2017). Collaborative filtering using non-negative matrix factorisation. *Journal of Information Science*, *43*(4), 567–579.

Ahn, H. J. (2006). Utilizing popularity characteristics for product recommendation. *International Journal of Electronic Commerce*, *11*(2), 59–80.

Ahn, H. J. (2008). A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information sciences*, *178*(1), 37–51.

Bell, R. M., & Koren, Y. (2007). Lessons from the netflix prize challenge. *Acm Sigkdd Explorations Newsletter*, *9*(2), 75–79.

Bell, R. M., Koren, Y., & Volinsky, C. (2007). The bellkor solution to the netflix prize. *KorBell Team's Report to Netflix*.

Bennett, J., & Lanning, S. (2007). The netflix prize. *Proceedings of KDD cup and workshop*, *2007*, 35.

Carmona, P., Climent, F., & Momparler, A. (2019). Predicting failure in the us banking sector: An extreme gradient boosting approach. *International Review of Economics & Finance*, *61*, 304–323.

Chiny, M., Chihab, M., Bencharef, O., & Chihab, Y. (2022). Netflix recommendation system based on tf-idf and cosine similarity algorithms. *no. Bml*, 15–20.

De Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean absolute percentage error for regression models. *Neurocomputing*, *192*, 38–48.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, 2000 Proceedings 1*, 1–15.

Ding, Y., & Li, X. (2005). Time weight collaborative filtering. *Proceedings of the 14th ACM international conference on Information and knowledge management*, 485–492.

Donnelly, R., Kanodia, A., & Morozov, I. (2023). Welfare effects of personalized rankings [Forthcoming]. *Marketing Science*.

Dzyabura, D., & Hauser, J. R. (2019). Recommending products when consumers learn their preference weights. *Marketing Science*, *38*(3), 417–441.

GitHub. (2018). *yehuda_koren_recommender*. https://github.com/asingh9530/yehuda_koren_recommender

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–70.

Gong, S. (2010). A collaborative filtering recommendation algorithm based on user clustering and item clustering. *J. Softw.*, *5*(7), 745–752.

Hartigan, J. A. (1975). *Clustering algorithms*. John Wiley & Sons, Inc.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, *28*(1), 100–108.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5–53.

Hong, T., & Tsamis, D. (2006). Use of knn for the netflix prize. *CS229 Projects*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis*. John Wiley & Sons, Inc.

Kawaguchi, K., Uetake, K., & Watanabe, Y. (2019). Effectiveness of product recommendations under time and crowd pressures. *Marketing Science*, *38*(2), 253–273.

Kersbergen, B., & Schelter, S. (2021). Learnings from a retail recommendation system on billions of interactions at bol. com. *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2447–2452.

Kim, B.-D., & Kim, S.-O. (2001). A new recommender system to combine content-based and collaborative filtering systems. *Journal of Database Marketing & Customer Strategy Management*, *8*, 244–252.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 426–434.

Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix prize documentation*, *81*(2009), 1–10.

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *4*(1), 1–24.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

Koren, Y., Rendle, S., & Bell, R. (2021). Advances in collaborative filtering. *Recommender systems handbook*, 91–142.

Kumar, D., Girase, S., & Mukhopadhyay, D. (2015). Role of matrix factorization model in collaborative filtering algorithm: A survey. *CoRR, abs/1503.07475*.

Liu, J., & Huang, T. (2023). Erratum on "competing for recommendations" model by zhou and zou (2023) [Forthcoming]. *Marketing Science*.

MacKenzie, I., Meyer, C., & Noble, S. (2013). How retailers can keep up with consumers. *McKinsey & Company*, *18*(1).

Maltz, D., & Ehrlich, K. (1995). Pointing the way: Active collaborative filtering. *Proceedings of the SIGCHI conference on Human factors in computing systems*, 202–209.

Melville, P., & Sindhwani, V. (2010). Recommender systems. *Encyclopedia of machine learning*, *1*, 829–838.

Nielsen, F. (2016). Hierarchical clustering. *Introduction to HPC with MPI for Data Science*, 195–211.

O'Connor, M., & Herlocker, J. (1999). Clustering items for collaborative filtering. *Proceedings of the ACM SIGIR workshop on recommender systems*, *128*.

Park, Y., Park, S., Jung, W., & Lee, S.-g. (2015). Reversed cf: A fast collaborative filtering algorithm using a k-nearest neighbor graph. *Expert Systems with Applications*, *42*(8), 4022–4028.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD cup and workshop*, *2007*, 5–8.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, *40*(3), 56–58.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, 53–65.

Sarwar, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th international conference on World Wide Web*, 285–295.

Sarwar, G., Konstan, J., & Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. *Proceedings of the fifth international conference on computer and information technology*, *1*, 291–324.

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, 149–171.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, *36*(2), 111–133.

Ungar, L. H., & Foster, D. P. (1998). Clustering methods for collaborative filtering. *AAAI workshop on recommendation systems*, *1*, 114–129.

Vézina, R., & Militaru, D. (2004). Collaborative filtering: Theoretical positions and a research agenda in marketing. *International Journal of Technology Management*, *28*(1), 31–45.

Vozalis, M. G., & Margaritis, K. G. (2006). Applying svd on generalized item-based filtering. *Int. J. Comput. Sci. Appl.*, *3*(3), 27–51.

Wang, Y.-X., & Zhang, Y.-J. (2012). Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on knowledge and data engineering*, *25*(6), 1336–1353.

Yim, O., & Ramdeen, K. T. (2015). Hierarchical cluster analysis: Comparison of three linkage measures and application to psychological data. *The quantitative methods for psychology*, *11*(1), 8–21.

Zhou, B., & Zou, T. (2023a). Competing for recommendations: The strategic impact of personalized product recommendations in online marketplaces. *Marketing Science*, *42*(2), 360–376.

Zhou, B., & Zou, T. (2023b). Rejoinder on "competing for recommendations: The strategic impact of personalized product recommendations in online marketplaces" [Forthcoming]. *Marketing Science*.

9 Appendix

	$k_{kmeans} = 10$	$k_{kmeans} = 20$	$k_{kmeans} = 30$	$k_{kmeans} = 40$	$k_{kmeans} = 50$
Silhouette Score	0.531	0.463	0.484	0.507	0.449
	$k_{kmeans} = 60$	$k_{kmeans} = 70$	$k_{kmeans} = 80$	$k_{kmeans} = 90$	$k_{kmeans} = 100$
Silhouette Score	0.411	0.466	0.402	0.496	0.478

Table 9.1: Cross validation K-means MovieLens

	$k_{kmeans} = 10$	$k_{kmeans} = 20$	$k_{kmeans} = 30$	$k_{kmeans} = 40$	$k_{kmeans} = 50$
Silhouette Score	0.572	0.493	0.483	0.453	0.429
	$k_{kmeans} = 60$	$k_{kmeans} = 70$	$k_{kmeans} = 80$	$k_{kmeans} = 90$	$k_{kmeans} = 100$
Silhouette Score	0.423	0.425	0.421	0.425	0.429

Table 9.2: Cross validation K-means Netflix