Towards Personalized Recommendation Systems: Domain-Driven Machine Learning Techniques and Frameworks

By

Rabaa Alabdulrahman

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of

Doctorate in Philosophy degree in Digital Transformation and Innovation

School of Electronic Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Recommendation systems have been widely utilized in e-commerce settings to aid users through their shopping experiences. The principal advantage of these systems is their ability to narrow down the purchase options in addition to marketing items to customers. However, a number of challenges remain, notably those related to obtaining a clearer understanding of users, their profiles, and their preferences in terms of purchased items. Specifically, recommender systems based on collaborative filtering recommend items that have been rated by other users with preferences similar to those of the targeted users. Intuitively, the more information and ratings collected about the user, the more accurate are the recommendations such systems suggest.

In a typical recommender systems database, the data are sparse. Sparsity occurs when the number of ratings obtained by the users is much lower than the number required to build a prediction model. This usually occurs because of the users' reluctance to share their reviews, either due to privacy issues or an unwillingness to make the extra effort. Grey-sheep users pose another challenge. These are users who shared their reviews and ratings yet disagree with the majority in the systems. The current state-of-the-art typically treats these users as outliers and removes them from the system. Our goal is to determine whether keeping these users in the system may benefit learning. Thirdly, cold-start problems refer to the scenario whereby a new item or user enters the system and is another area of active research. In this case, the system will have no information about the new user or item, making it problematic to find a correlation with others in the system. This thesis addresses the three above-mentioned research challenges through the development of machine learning methods for use within the recommendation system setting.

First, we focus on the label and data sparsity though the development of the Hybrid Cluster analysis and Classification learning (HCC-Learn) framework, combining supervised and unsupervised learning methods. We show that combining classification algorithms such as k-nearest neighbors and ensembles based on feature subspaces with cluster analysis algorithms such as expectation maximization, hierarchical clustering, canopy, k-means, and cascade k-means methods, generally produces high-quality results when applied to benchmark datasets. That is, cluster analysis clearly benefits the learning process, leading to high predictive accuracies for existing users.

Second, to address the cold-start problem, we present the Popular Users Personalized Predictions (PUPP-DA) framework. This framework combines cluster analysis and active learning, or so-called user-in-the-loop, to assign new customers to the most appropriate groups in our framework. Based on our findings from the HCC-Learn framework, we employ the expectation maximization soft clustering technique to create our user segmentations in the PUPP-DA framework, and we further incorporate Convolutional Neural Networks into our design. Our results show the benefits of user segmentation based on soft clustering and the use of active learning to improve predictions for new users. Furthermore, our findings show that focusing on frequent or popular users clearly improves classification accuracy. In addition, we demonstrate that deep learning outperforms machine learning techniques, notably resulting in more accurate predictions for individual users.

Thirdly, we address the grey-sheep problem in our Grey-sheep One-class Recommendations (GSOR) framework. The existence of grey-sheep users in the system results in a class imbalance whereby the majority of users will belong to one class and a small portion (grey-sheep users) will fall into the minority class. In this framework, we use one-class classification to provide a class structure for the training examples. As a pre-assessment stage, we assess the characteristics of grey-sheep users and study their impact on model accuracy. Next, as mentioned above, we utilize one-class learning, whereby we focus on the majority class to first learn the decision boundary in order to generate prediction lists for the grey-sheep (minority class). Our results indicate that including grey-sheep users in the training step, as opposed to treating them as outliers and removing them prior to learning, has a positive impact on the general predictive accuracy.

Acknowledgments

This journey has been a long, challenging, and exciting one, yet, here I am at the end of one phase in my life, moving to the next one. First and foremost, I would like to thank God for being my strength and guide throughout life. Furthermore, I would like to thank everyone who inspired me and was part of my journey.

My sincere appreciation and respect to my supervisor Prof. Herna Viktor for her continuous support and encouragement. This thesis would not have been possible without all her guidance, knowledge, and advice. Her caring personality encouraged me to continue my studies and research. It has been an honor to work with her.

I kindly thank the academic staff of the Electronic Business Technology program at the University of Ottawa.

In addition, I would like to thank the Saudi Cultural Bureau in Canada for their financial support which helped me achieve my goals.

My mom Aisha, my role model in life; my inspiration when I hit my lowest point. Determination, hard work, and hope are what I learnt from this amazing woman. God bless you, Mom. Know this: I can’t stop thinking of the day when I’m right back beside you to continue my life journey.

My dad, sisters, and brothers, nieces and nephews: a sincere thank you for every WhatsApp joke and FaceTime call. You all kept me going with your loving words. Thank you, my family, for all the support you gave me despite the distance that came as a result of me pursuing my goals.

My brother, Ahmed, you have been here with me from the start; thank you for every cup of coffee you brought me when I had not time to move from my spot, and every meal you prepared when I lost track of time. I am a proud big sister of what you have become. Keep going and chase your dreams, and know there are no limits to what you can achieve.

To Jumanah and Fadi, my family in Canada: it all started nine years ago as a friendship, yet you became my family. Thank you for everything you have done to encourage and motivate me. Jumanah, your words kept me going whenever I became frustrated and when coffee lost its effect on me—my deepest thanks.

Table of Contents

List of Figures

List of Tables

Acronyms

CBF	Content-Based Filtering
CF	Collaborative Filtering
CNN	Convolutional Neural Network
DT	Decision Tree
DLRS	Deep Learning Recommender Systems
EM	Expectation Maximization
FCR	Fuel-Consumption Rating
GSOR	Grey-sheep One-class Recommendation Framework
HC	Hierarchical Clustering
HCC-Learn	Hybrid Cluster analysis and Classification Learning Framework
HT	Hoeffding Tree
IR	Interquartile Range
k-NN	k-Nearest Neighbor
MAE	Mean Absolute Error
ML	Machine Learning
NB	Naïve Bayes
PUPP-DA	Popular User Personalized Prediction Framework
RC	Restaurant-Consumer rating
RSs	Recommendation Systems

Chapter 1. Introduction

1.1 Overview

Recommendation systems (RSs) is an interdisciplinary area of study that has attracted much interdisciplinary research interest, spanning computer science, business, economics, and social sciences. The value these systems add to organizations has led to the development of algorithms that aim to accurately recommend the optimal sets of items to users. Based on these recommendations, browsers may turn into buyers and one-time buyers may turn into loyal customers. After the Netflix prize [4, 5], many research studies have been conducted in this area. Such research not only aims to make the best recommendations, but also intends to identify the best recommendation for individual customers as opposed to blindly displaying lists of the most popular items.

Take Amazon as an example, i.e., a well-known successful business that not only attracts consumers but also business owners [6]. Such business owners trust Amazon to make the best advertisements for them and to reach out to the right customer at the right time. Another well-known example is Shopify [7]. The success of Shopify lies in the fact that they not only present business owners with an easy platform to build their e-commerce website, but also utilize RSs which are specifically created to gain more customers.

Consumers face information overload every time they access the Internet to make a purchase. That is, they are faced with “too many items to choose from.” In today’s fast-paced world, they have neither the time nor the patience to explore all these suggestions. Therefore, the main idea behind RSs is to address the abovementioned problem and aid the users in narrowing down their browsing lists. Thus, these systems work by understanding users’ preferences. This is achieved not only by recognizing their rating for specific items, but in some systems by considering their social and demographic information [8]. Consequently, these systems create a database for both items and users in which the users’ rating and reviews of these items are collected [9]. Intuitively, the more information and ratings collected about the user, the more accurate the recommendation is [10]. However, that is not always the case. For instance, some consumers care about their privacy and will not disclose personal information [8]. Furthermore, some clients may be first-time users of e-business systems, as is the case with the surge in online shopping during the Covid-19 pandemic, leading to their preferences being unknown. Furthermore, an increasing number of users have unique and exotic tastes which makes it harder for the system to match their interests with the current customer base.

Generally speaking, RSs are either content-based filtering (CBF) [11], collaborative filtering (CF) [12], or hybrid approaches [13]. These systems rely on two basic inputs: the set of users in the system, U (also known as customers), and the set of items to be rated by the users, I (also known as the products) [14]. All these systems employ matrices based on past purchase patterns. With CBF, the system focuses on item matrices, whereby it is assumed that if a user liked an item in the past, he or she is more inclined to prefer a similar item in the future [9, 15]. These systems therefore study the attributes of the items [12]. However, CF systems focus on user-rating matrices, recommending items that have been rated by other users with preferences similar to those of the targeted user [16]. Thus, these systems rely on historic data consisting of user rating and similarities across the user network [9]. As hybrid systems employ both CBF and CF approaches, they concurrently consider items based on users' preferences and on the similarity between the items' content [15]. In recent years, research has trended toward hybrid systems [12]. Another growing trend is the use of machine learning (ML) algorithms [17] to identify patterns in users' interests and behaviors [17].

In the literature, there are many challenges facing RSs, such as the lack of rating for some items or the lack of information about users. This problem is commonly known as data sparsity, which affect the system's ability to build a user profile [18]. Furthermore, new users entering the system will cause a so-called cold-start problem, where the system will not be able to build a user profile, due to the initial lack of information about preferences, and will therefore not be able to match them to existing customers. In addition, grey-sheep users, i.e., those with more unusual tastes, also pose a bottleneck for RSs [19, 20]. The challenge for the business is how to keep these atypical customers satisfied. Due to the above-mentioned challenges, it is difficult for RSs to present the users with accurate recommendations. In the e-commerce setting, this mean losing potential customers to another business. This dissertation aims to address these three challenges, which are discussed next.

1.2 Research Challenges

Our aim is to study the three abovementioned research challenges within the RSs context through the development of ML solutions. In our work, we utilize supervised learning algorithms, also referred to as classification techniques, where the labels are known. In addition, we employ unsupervised learning algorithms—specifically cluster analysis approaches—when there is a lack of labels, and one-class techniques when the distributions of labels are skew. Finally, we develop active learning techniques to facilitate scenarios where domain expertise will aid the learning process. Specifically, our goal is to explore three key solution spaces, framed within three research problems.

Sparsity challenge: As discussed earlier, in RSs, sparsity occurs when the number of items a customer typically purchases is much smaller than the number of items for sale [19]. Consequently, the number of ratings, or feedback, will be much lower compared to the number of items in the system. This problem may lead to inaccurate recommendations, as ML algorithms may not be able to generalize well. Furthermore, some ML algorithms, notably the k-nearest neighbor (k-NN) method that has been widely employed in RSs, are more sensitive to data sparsity compared to others [18]. Alasalmi, T., et al. (2015) stated that, naturally, classification algorithms require a complete dataset and most variables values to exist in order to perform well. They also noted that some algorithms represent exceptions from the general population of ML algorithms in their ability to handle missing values, such as Naïve Bayes (NB) [21]. In some research, the use of manual labeling is done by the domain expert, either to address sparsity, or to simply identify these records and remove them from the system. However, involving human experts tend to be time consuming and expensive, and in many cases this is not a realistic option, especially with large systems. In Chapter 3, we address this challenge by using cluster analysis as a preprocessing step prior to classification to guide the learning process to form natural grouping, typically using similar customer profiles, in order to improve predictive accuracy. Our Hybrid Cluster analysis and Classification Learning (HCC-Learn) framework combines content-based analysis in the preprocessing stage and CF in the final prediction stage. We detail the HCC-Learn framework in Chapter 3.

Cold-start challenge: A cold-start refers to new users, with unknown preferences, joining the system. In RSs, users' preferences, historic data in what they like and dislike, and their items' ratings and reviews are used to match them with others in the system. In cold-start situations, it thus follows that such information does not exist, which makes it difficult for the system to calculate a similarity score. Indeed, the tremendous increase in the use of e-commerce websites during the current Covid-19 pandemic has highlighted the importance and difficulty of providing accurate recommendations to many first-time users [22, 23].

To solve this problem, some research tends to use CBF systems, in which information about the items are used to find the best match [24]. Other systems simply present these users with a predefined recommendation list. Although these might be successful solutions for some users, it often results in a redundant list being presented, which causes the user to lose interest [25]. Another solution is the use of conversational learning models, where the new user is presented with a list of questions to build a preference profile [26, 27]. This also might cause the business to lose a potential customer because of their unwillingness to make the extra effort or because of privacy concerns. To address the cold-start challenge, we propose the Popular User Personalized Prediction (PUPP-DA) framework that combines active learning, ML, and deep learning algorithms. In this framework, soft clustering and active learning are used to accurately recommend items to new users in this framework. In addition, we employ deep learning algorithms to improve the overall predictive accuracy. Our PUPP-DA framework is presented in Chapter 5.

Grey-sheep challenge: As mentioned before, grey-sheep users are ones who are difficult to identify or characterize. These kinds of users are willing, to some degree, to share their feedback. However, their preferences will disagree with the majority in the system. In contrast with cold-starts, the system will have the information it might need to calculate a similarity score and produce a recommendation list. However, this list may not be accurate because of their unique tastes and characteristics. Typically, grey-sheep users are treated as outliers and removed from the system [28, 29]. In other research, they will be moved to a separate system where their preferences may then be matched with others [30, 31]. However, this is not realistic in large and online systems, as identifying and moving user/items to a secondary system is time-consuming. Therefore, in Chapter 5 we present our Grey-Sheep One-Class Recommendation (GSOR) framework designed to create accurate prediction models while considering both regular and grey-sheep users. The GSOR framework utilizes one-class classification, whereby the learning process is done using information from the majority class, while predictions are made for the minority class, in our case grey-sheep users. Next, we detail the research questions we aim to address in this dissertation.

1.2.1 Research Questions

As discussed in the previous section 1.2, the RSs research area has many challenges. In this section, we list the main research questions (RQs) related to this dissertation:

RQ1. How can classification algorithms be used to alleviate data sparsity, cold-start, and grey-sheep challenges and improve RS performance?

- How effective and efficient are ML methods in this domain, i.e., when and where should they be used to improve RSs?
- What are the strengths and limitations of existing methods when evaluated against a number of RSs databases?
- Is one-class learning able to address the grey-sheep challenge and produce an accurate recommendation list?

RQ2. What value can unsupervised learning add to the system?

- Can unsupervised learning techniques improve the recommendations accuracy and address data sparsity, cold-starts, and grey sheep?
- If so, which cluster analysis method will be best to use, in various scenarios, with different levels of sparsity and outliers?
- What is the impact of cluster analysis techniques as reported through extensive study?

RQ3. What is the impact of involving a human expert in the learning process?

- Can active learning techniques be used to improve RSs performance?
- If so, which algorithms work best to address cold-start problems?
- How, and when, can the user be involved in the recommendation process without burdening them?

RQ4. How and when can deep learning be used within the RSs framework?

- Is there an added benefit when extending ML techniques with deep learning solutions?
- What is the value of adding deep learning within an active learning framework?

RQ5. What is the impact of grey-sheep users' existence in the systems?

- How severely is the predictions accuracy affected by such users?
- How can items of interests be identified for grey-sheep users?
- Can one-class classification yield better recommendations for grey-sheep users?

Next, we discuss our motivation from an e-commerce perspective.

1.3 Motivation

The main focus of this study is the advancement of RSs in the e-commerce domain, due to the added value they offer to commercial organizations embracing the increasing demand of 24/7 online shopping. Indeed, most organizations have realized that, based on accurate recommendations, they can not only attract new consumers but also expand the shopping lists of their existing consumer base. Therefore, there is a need not only to make the best recommendations, but also to find the most pertinent recommendation for each customer rather than simply displaying a list of the most popular items.

As discussed earlier, RSs are generally divided into CF and CBF systems. However, each category suffers from some limitations, as noted above, and thus a hybrid filtering approach was developed to overcome some of the limitations in these systems and enhance the recommendation accuracy. However, with the development of technology and the increase in the popularity of online businesses, there is a pressing need for more research to address the continued limitations such as data sparsity, cold-start, and grey sheep. The fundamental goal of ML is identifying and recognizing useful and novel patterns in data [32, 33]. Furthermore, in an e-commerce setting, ML algorithms find patterns about user's interests and behaviors [17]. The main principle is to utilize the variables in the system to make predictions [32].

Recently, with the onset of the Covid-19 pandemic, many businesses have turned to e-commerce solutions in an attempt not only to survive but also to thrive post-pandemic [34, 35].

When Covid-19 appeared in late 2019, each government was forced to implement plans for how to face the virus when it arrived in their country. In many countries, lockdown procedures were implemented immediately, leaving citizens with the reality of online shopping.

As Ungerer, C., et al. (2020) observed, e-commerce is the process of selling goods or services online. Before the pandemic, for many users this meant the luxury of importing unique items unavailable in local markets, or the luxury of shopping from the warmth and comfort of their home during the Canadian winter. However, the pandemic transformed e-commerce, for many, into a tool for survival. In many countries, even if a complete lockdown was not enforced, physical distance measures were encouraged. However, as the infection rate began to increase, people started to turn to online ordering to avoid any contact that might cause them to contract the disease. In addition, the movement of the vulnerable and elderly was restricted, leading to a large portion of these customers turning to e-commerce for the first time. Furthermore, in many countries, most non-essential businesses were closed until further notice. For these businesses to survive, this meant reaching out to the customers through the web or social media stores, such as Instagram and Facebook markets.

In terms of general e-commerce, the purpose of online shopping has shifted from convenience of time and location to a necessity. In fact, as the United States started lifting the partial lockdown and opening up the economy again, a survey of consumers' intentions to return to old shopping practices was reported in [36]. The results showed that 24% do not intend to shop in a mall in the next six months, while another 16% stated the same for the next three months. We believe that the same observations hold true in Canada.

The current shift in consumers' habits stresses the importance of meeting customers' demands. Furthermore, it confirms the significance of catering to the right products for the customers, including cold-starts and grey sheep, to avoid losing them to another business and to also streamline supply chains. Online competition is at it peak, and a significant percentage of businesses must address this challenge. Several studies have shown the importance of e-commerce, and notably personalised RSs, during the pandemic across all sectors. For instance, this shift is also relevant in the health care sector, where healthcare providers have had to move to e-commerce to provide tailor-made care and treatments [37].

Although such personalized recommendations have been presented in the literature since 1990s, they have only been widely adopted in e-businesses in recent years. According to Ming-Kuen, C.,

C. Kuo-Hsuan, and C. Chia-Hon (2014), a personalized RS should include data collection, data warehousing, data mining, and data applications [38]. Data mining [1] techniques can make predictions without the need to access users profile information and items; for this reason, they have been used along with RSs to increase the recommendation performance [39, 40]. Many successful businesses have implemented personalized RS approaches in their systems. For instance, Amazon created a personalized recommendation list for each user, followed by other businesses such as Hotels.com which help the user decide based on a smaller suggestion list [41]. Furthermore, studies have shown that using these increases profits [42].

In January 2020, Winkler, N. (2020) reflected on the growth of e-commerce and noted that, in the United States alone, it was expected that sales would top $4.2 trillion USD in 2020 and that 2.1 billion customers will shop online. These numbers and expectations were based on the previously collected data for 2020. However, on April 30th, 2020, Amazon released their first-quarter financial results which describes their total earning as "exceptionally" strong, estimated as $33 USD million an hour in sales for the first three months of this year [43]. In North America alone, sales increased by 29%, about $46.1 billion, compared to last year.

The implications of this trend for the RS research community are manifold. In terms of data sparsity, the number of users increased exponentially, yet many of these users are new. Another aspect of note is that, even for existing users, there will be a shift in their preferences, which will also cause sparsity. Since the pandemic started, many users have switched preferences from "what to buy" to "what is needed," which has left previously popular and frequently rated items being ignored. Furthermore, considering the current situation we live in, many businesses have decided to keep work-from-home practices implemented until the end of this year. Consequently, many consumers have changed their preference from a formal dress, for instance, to comfortable lounge wear. Another challenge centers on cold-start users, where it should be noted that many individuals have turned to e-business for the very first time. This poses a challenge for RSs, since there is a substantial gap in what is known about these users. It may well be that a substantial portion of these new users are indeed grey sheep who typically would not have used e-business during normal times.

[1] We use the terms data mining techniques and ML algorithms interchangeably. However, we note that data mining focuses more on the discovery of patterns, often using ML algorithms.

Considering the three mentioned challenges in RSs, let us now illustrate the current situation with some examples. As discussed earlier, the shift in preferences causes data sparsity, which is a principal challenge for RSs. According to Statista (2019), the lowest two categories by household type who shopped online in Canada prior to the pandemic are either singles, who cohabitate with other adults (e.g., parents or roommates) and single parents [44]; both represent 12% and 3%, respectively, of all participants. For these two types of customers, they have either never shopped online before, or they are using e-commerce now but with different types of demands. In Canada, Millennials and Baby-Boomers had the highest percentage of online consumers sales during 2019 [45]. Today, the shift in preference has turned towards ordering what is necessary for homeschooling or for entertaining children. In fact, a 2019 report by Canada Post indicates that 62% of Canadians shopped for clothing apparel, whereas 41% shopped for computers and electronics using e-commerce [45]. After the pandemic hit, a report by Cision (2020) shows that all e-commerce sales increased, except for clothing (which had the lowest increase of only 21% only). Meanwhile, the sales of electronics increased by 160% [46].

In 2019, it was reported that Pre-Boomers, i.e., those aged 73 and older, as well as Gen Z, i.e., customers in the 18–23 age group, constitute the lowest percentages, 5% in each category, of online shoppers in Canada. These customers represent two very different generations and are thus often difficult to target. For a business to thrive online, it must understand its customers' behaviour and characteristics in order to expand its customer base. Gen Z, for instance, are considered to be the main influencers over buying decisions for most families [47]. According to Forbes (2020), technology is crucial for enhancing the Gen Z shopping experience and providing them with instant and quality services [47]. In Canada, is expected that by 2026, 21% of the population will fall into the 73 years and older category. Older customers often fall in the grey-sheep category and, as discussed in [47], they have a preference for products that provide them with improved life quality [48]. As noted by Retail Insider (2020), this group of customers prefer viewing products physically before buying them [48]. For instance, as the pandemic lockdown started in Canada in mid-March 2020, many grocery stores dedicated special hours for senior shoppers. However, a recent study by Statistics Canada indicates that a large portion of such customers turned to online shopping, with 45% of people aged 75 and older indicating that they did so [49]. The question here is how to target these customers and, as many have turned to e-commerce for the first time, how to maintain their customer base when life returns to normal.

As discussed, an RS is a very important aspect of economic growth for businesses engaging in e-commerce. With the current abrupt shift in their lives, many consumers depend on e-commerce for essential items. This include business owners reaching out to customers, or customers meeting everyday needs. The challenge is how to accommodate the entire customer base, including loyal customers, new users, and those with unique tastes. Our goal in this dissertation is to address the three mentioned challenges, as is discussed next.

1.4 Summary of Contributions

In this section, we summarize the major contribution of this dissertation as we focus on addressing sparsity, cold-start, and grey-sheep users.

1.4.1 HCC-Learn framework

This framework provides a hybrid RS to focus on label and data sparsity. The Hybrid Cluster analysis and Classification Learning (HCC-Learn) framework works by combining supervised and unsupervised learning methods. Different classification algorithms such as k-nearest neighbor (k-NN) and ensemble learning approaches are used along with several cluster analysis algorithms such as expectation maximization (EM), hierarchical clustering (HC), canopy, k-means, and cascade k-means. Our results show that a cluster analysis benefit the learning process and leads to higher predictive accuracies for existing users.

1.4.2 PUPP-DA Framework

The Popular User Personalized Prediction (PUPP-DA) framework is developed to address the previously introduced cold-start problem. In this framework, we incorporate ML and deep learning techniques within the active learning setting. We also employ the EM soft cluster analysis method to create our user segmentations[2] [50]. Our results show the benefit of user segmentation and soft clustering, and the use of active learning to improve the prediction list for new users. We also report that focusing on the popular users clearly improves the classification accuracy. Our

[2] In the HCC-Learn framework, we extensively examine different clustering techniques and conclude that EM soft clustering results in higher accuracies [50].

framework includes a convolution neural network, and our results show that using deep learning technique results in accurate predictions for individual users.

1.4.3 GSOR framework

As noted earlier, in RSs, grey sheep refers to those users with unique tastes for whom it is difficult to develop accurate profiles of their preferences, as the similarity search approach typically followed during the recommendation process fails to yield good results. Most research ignores such users, and thus fails to cater for more exotic tastes and emerging trends, leading to subsequent loss in revenue and opportunity. To address this oversight in the research community, we present the Grey-sheep One-class Recommendation (GSOR) framework, which is designed to create accurate prediction models while considering both regular and grey-sheep users. Our results indicate that one-class learning is a successful solution to generate a recommendation list for grey-sheep users.

1.4.4 Grey-sheep test set

To evaluate our GSOR framework, we create our own test set which may be used as a benchmark for grey-sheep experiments. The process of creating this test set is explained in subsection 0. and we plan to make this benchmark available to the research community.

1.4.5 Scholarly Achievements

In the process of completing this work, the following lists our publications in peer reviewed journals and conference, which validate our work in the research community.

- Refereed journal papers and book chapter:
  - ❖ HCC-Learn Framework for Hybrid Learning in Recommender Systems. HCC-Learn framework has been published in [50].

- Refereed conference papers:
  - ❖ Beyond k-NN: Combining Cluster Analysis and Classification for Recommender Systems. Initial HCC-Learn framework has been published in [51].

- Active Learning and User Segmentation for the Cold-start Problem in Recommendation Systems. The initial PUPP framework, which did not involve deep learning, has been published in [52].

1.5 Thesis Organization

This dissertation is organized as follows. Chapter 2 discusses the fundamental concept of RSs in e-commerce and presents the different types of RSs, its strengths, and its challenges. Furthermore, we describe the ML solutions used in RSs. Next, we present our HCC-Learn framework which focuses on the data sparsity challenge in Chapter 3 and evaluate it by considering the system usefulness and its effectiveness in predicting user responses. Chapter 4 outlines our PUPP-DA framework developed to address cold-starts. We present our third framework in Chapter 5, i.e., the GSOR framework which utilizes one-class learning to target grey-sheep users. Finally, Chapter 6 concludes this dissertation and discusses future work.

Chapter 2. Background

2.1 Recommendation Systems in e-Commerce

The widespread adoption of the Internet has presented sellers with new challenges to keep up with the market. Traditionally, sellers had to compete with local stores within the same city or country. However, we now live in a technological era in which consumers increasingly turn to the Internet for purchases. Subsequently, many technological solutions have been presented to sellers, such as newer software to build their business websites. A major concern is that when a business owner cannot see their customers, it is hard to know what their preferences are. Thus, sellers may lose them to another business. Moreover, the omnipresence of the Internet and e-commerce websites has presented users with a so-called information overload problem [53, 54]. RSs present a technology developed to aid sellers, entrepreneurs, and customers in addressing these challenges. The main goal of these systems is to help the users find a specific product or service by generating useful information [55]. In addition, they predict objects of interest based on the data collected on the users' likes and preferences [56]. Consequently, a browser might be turned into a buyer if the business employs the right software tools [57-59].

To overcome many of the presented challenges such as information overload [12, 60, 61] and customer loyalty [58, 62, 63], e-commerce websites utilize a Customer Relationship Management (CRM) strategy [64]. RSs are the most noted strategy in CRM. In general, these systems can be defined as programs that predict the user's interests and match these interests to the available items. Based on these matching results, a set of recommendations is presented to the users [24]. Nowadays, RSs applications have expanded to include areas such as e-learning, e-government, e-library, and e-health [24]. These systems helps to assist and support human decision making in many applications such as shopping, TV programs, on-demand movies, websites, news, documents, and book selections [65].

RSs rely on two basic entities: the set of users in the system U (also known as customer) and the set of items to be rated by the users I (known also as product) [14]. These systems collect information about user characteristics, user transactions, item attributes, and the relationship between user and item to produce accurate recommendations [54, 66]. Therefore, choosing the right similarity measure to apply on the collected data is a crucial task [67].

In e-commerce, RSs are defined as a "computerized recommendation agents" [62] to aid the customers in the decision-making process and reduce the overwhelming nature of the information overload [62, 68]. These systems have proven their ability to increase profits and reduce the search cost [61, 63, 68-70]. Moreover, RSs enable companies to create personalized marketing [59], and customize targeted advertisement campaigns [71], which lead to an increase in customer loyalty [58, 62, 63].

In this chapter, we introduce the different types of RSs. Their limitations and strengths are also discussed along with some of the solutions available in the literature. We also present some of the ML algorithms used to improve the RS performance. The research in ML is broad; therefore, we focus on ML algorithms and the methods used throughout this dissertation.

2.2 How do Recommendation Systems work?

As we mentioned earlier, an RS has two basic entities: the set of users U and set of items/products I, the main goal being to provide the user with a list of relevant recommendations. Therefore, for one system, a number of items $I = \{I_1, I_2, I_3, \dots, I_n\}$ exists. This list is described by a set of features and attributes. In addition, the set of users in the system is described as $U = \{U_1, U_2, U_3, \dots, U_m\}$. Each user from this list rates a set of items. Then, for every user U in the system, a list of n top recommendation is generated and presented to the user based on similarities with others in the system. However, while generating this list, the system should also consider the previously rated items by the users to avoid redundancy.
Adomavicius, G. and A. Tuzhilin (2005) stated that, for each user in the system, the aim is to choose the items $I' \in I$ that maximize the user's utility as follows:

$$\forall_{\boldsymbol{u}} \in \boldsymbol{U},\ \boldsymbol{I}'_{\boldsymbol{u}} = \boldsymbol{ar\,g} \max_{\boldsymbol{i} \in \boldsymbol{I}} \boldsymbol{s}(\boldsymbol{U}, \boldsymbol{I}) \qquad \textbf{2.1}$$

Where s is the utility function used to measure the usefulness of item I to user U.

In [68], the author divided the recommendation process into three main phases, namely **information collection, learning, and recommending**. During **information collection**, as the name suggests, the aim is to learn more about the users. As many authors have noted, the accuracy of the recommendation is highly related to the quality of information about the user in the system [68, 72-74]. This information enters the system in the form of users' feedback. There are three types of feedback that might exist in the system: *Explicit feedback*, where the user provides a rating through the system interface; *implicit feedback*, where the system monitors the user behavior, history, and purchases; as well as hybrid *feedback*, which is a combination of both types. During the **learning phase**, an algorithm is applied to learn the users' preferences. Finally, the systems involve **predictions,** which show a predicted score that user U_m will be interested in item I_n, or **recommendations,** which is a list of top N items that might be of interest to the user [75].

The following Figure 1 shows the types of RSs and their methods, which are discussed next.

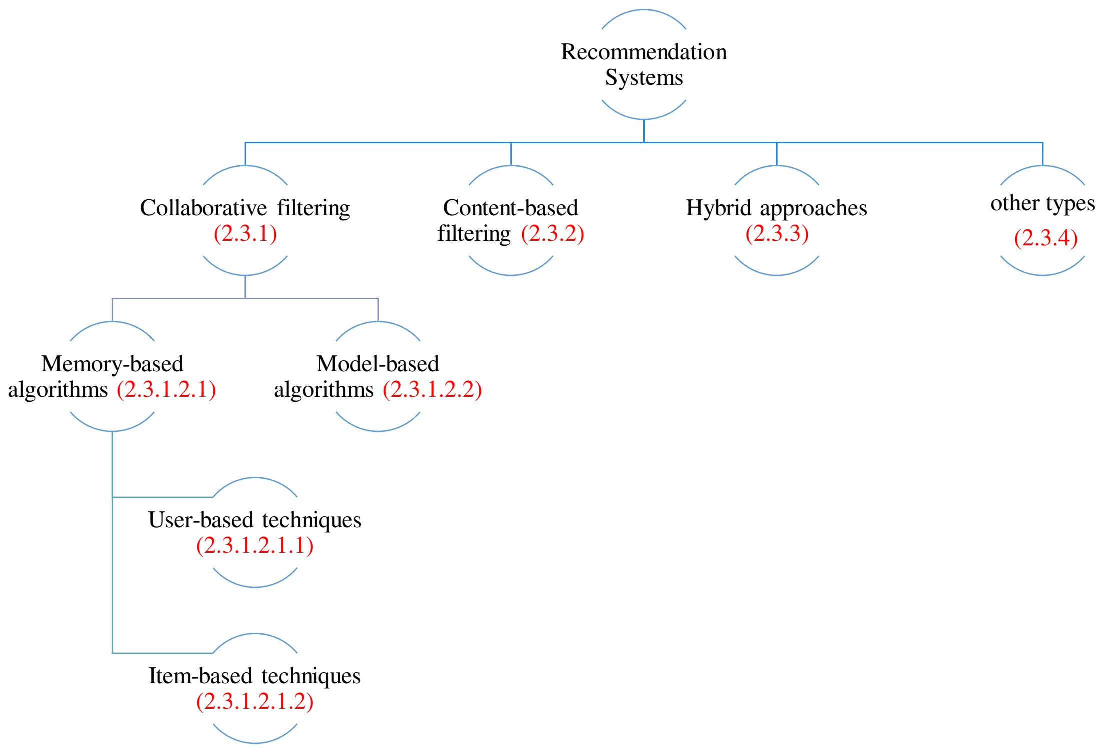

Recommendation Systems
Collaborative filtering (2.3.1)
Content-based filtering (2.3.2)
Hybrid approaches (2.3.3)
other types (2.3.4)
Memory-based algorithms (2.3.1.2.1)
Model-based algorithms (2.3.1.2.2)
User-based techniques (2.3.1.2.1.1)
Item-based techniques (2.3.1.2.1.2)

Figure 1. RSs and their methods, with their correspondence to chapter sections numbers.

2.3 Recommendation System Types

2.3.1 Collaborative Filtering

The CF category, also known as customer-to-customer correlation, is the most widely used RS [69, 70, 76-83], mainly because of its simplicity [84-88] and its wide applicability [12, 89-92], where the recommendations are based on the association between the user decision history and other users' opinions [66]. In general, these systems use the relationship and similarities between the users in the system to compute their recommendations [73, 74, 89, 93-96]. To make useful predictions, the system needs to match two entities, namely, users and items [97]. In a given domain, the users rate items and these values are then used to calculate the similarities between different users [98]. The CF system recommends specific items to the users based on the opinion of the user's "neighbors," who are other users who have shown similar preferences, behaviors, or opinions in the past [99, 100]. The k-NN algorithm was originally considered when creating a CF system to result in the top N recommendations [40, 101]. k-NN uses a similarity measure to calculate the distance between the active user to the k nearest neighbors and is discussed in section 2.5.2.2. In the following Figure 2, the general idea behind CF techniques is depicted.

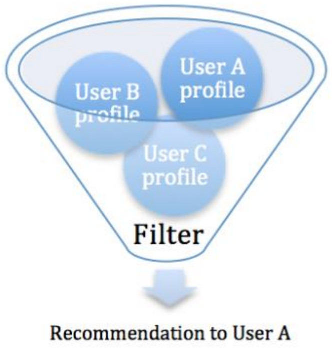
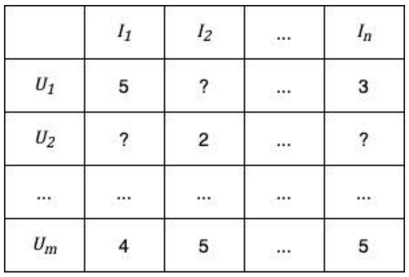


Figure 2. CF system diagram [3].

To provide the user with the most relevant recommendation, three main steps are followed [86, 102]. First, considering the items selected by the user, a new file is assigned to them. The CF system starts by processing the user's inputs to the system. Each user U_m in the system is asked to rate an item I_n that they purchased (e.g. books, clothes or songs). Subsequently, I_n is compared to other users' profiles, and the similarities are calculated using a similarity measure such as the Euclidean distance, Pearson correlation coefficient, and Cosine similarity [103, 104]. The system then matches user U_m to the k nearest neighbors $\{U_1, U_2, U_3, \dots, U_k\}$, retrieves a list of the s items they purchased $I = \{I_1, I_2, I_3, \dots, I_s\}$, and selects the subset of items not previously selected by user U_m . Finally, user U_m is presented with recommendations containing those items that other similar users rated highly. The collected data from the users are stored in the users–items matrix (see Figure 3).



	I_1	I_2	...	I_n
U_1	5	?	...	3
U_2	?	2	...	?
...	...	...	...	...
U_m	4	5	...	5

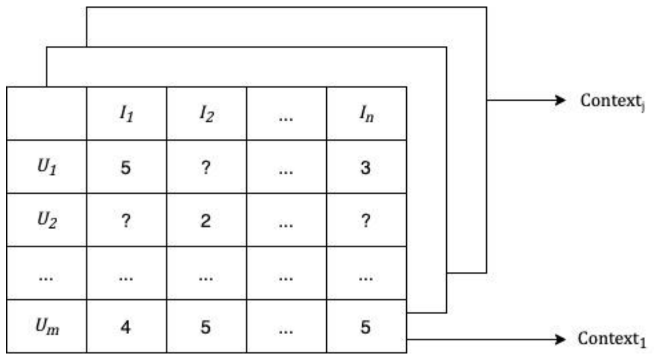
Figure 3. Two-dimensional user-item CF matrix [4].

The two-dimensional user–item matrix in Figure 3 includes the rating of m users $\{U_1, U_2, .., U_m\}$ on n items $\{I_1, I_2, .., I_n\}$. In the case of unrated items, a missing value is shown. Traditionally, RSs can be viewed as the following two-dimensional mapping function [105, 106]:

$$\boldsymbol{R}: \boldsymbol{U} \; \boldsymbol{x} \; \boldsymbol{I} \; \rightarrow \boldsymbol{Ratings} \qquad \textbf{2.2}$$

Where the mapping space S contain only $users$ and $items$. In the multi-dimensional approach, additional information such as time or location are also considered in the matrix [107], as shown in Figure 4. The mapping space S contain several dimensions D, $S = \{D_1, D_2, \dots, D_k\}$, e.g., where the dimensions correspond to different information. The mapping function is defined as [105, 106]:

$$\boldsymbol{R}: \boldsymbol{D_1} \; \boldsymbol{x} \; \boldsymbol{D_2} \; \boldsymbol{x} \dots \boldsymbol{x} \; \boldsymbol{D_k} \; \rightarrow \boldsymbol{Ratings} \qquad \textbf{2.3}$$



	I_1	I_2	...	I_n
U_1	5	?	...	3
U_2	?	2	...	?
...	...	...	...	...
U_m	4	5	...	5

Figure 4. A multidimensional user–item matrix [4].

In general, the accuracy and performance of CFs depend on several factors. Several authors have concluded that the quality of the recommendation is highly influenced by the characteristics of the prediction algorithm used in the system. In addition, the quality and number of ratings available in the system is another factor [81, 91, 108-112]. Intuitively, as stated by Elahi et al (2013), the more information about the user in the system, the more accurate the recommendation is [108].

2.3.1.1 Collaborative Filtering Approaches

In general, CF systems are divided into two main categories, namely *model-based* and *memory-based*. Furthermore, memory-based methods are further divided into *user-based* and *item-based* ones. The following two subsections provide more details.

2.3.1.1.1 Memory-Based CF Methods

Memory-based methods work by storing raw preference information about users in the systems' memory. With large-scale systems, in the current literature, researchers tend to use cloud-computing systems such as Hadoop, which increase the system capacity by providing distributed storage space and parallel processing capabilities [113]. This information is subsequently used to match similar users and make predictions [114]. Memory-based systems exploit user-to-user or item-to-item matrixes. Using the user-to-user matrix means recommendations are made based on items preferred by other users in the same neighbor, whereas item-to-item matches the user purchases with other similar rated items [115].

In memory-based CF techniques, a similarity metrics is used to measure the similarities between users in the system. Depending on the nature of the collected data, there are several similarity measures that may be used; such as the Pearson correlation coefficient, cosine similarity, and Jaccard's similarity [83].

These type of systems are generally considered to be simple and easy to develop, making them successful in real-life applications [68], as they are able to ingest new or additional information about users or items [88]. Memory-based methods are less scalable with the increased information of an online environment and large scale databases [116]. To overcome this challenge, many researchers have turned to parallel and distributed systems [117]. In parallel frameworks, a shared memory is used for parallel computing, such as Pthreads and Java Threads; or shared GPU computing, such as CUDA and OpenCL. As for distribution frameworks, Hadoop has been used as the basis for several frameworks such as Spark and Mahout [117]. In these frameworks, Hadoop distributes the work data into a number of clusters, then performs operations in parallel [118]. In parallel computing, the computer resources may consist of one or more than one computer with multi-processors connected by a local or internet network [119]. In other words, one problem is divided into separate instruction groups that may be executed simultaneously using different processors and by accessing one global address space [119]. However, in distributed frameworks, each processor has their own local memory, which operates independently. For one processor to access the data of another, a programmer must set up the structure for when and how [119].

As mentioned, there are two types of memory-based CF methods, which are discussed next.

2.3.1.1.1.1 User-based CF methods

As stated above, user-based CF approaches provide recommendations based on similar items liked by similar users. It assumes that if two users liked the same item, they will like the same new item later [120]. This method works through three steps to present a recommendation to the user. First, using the item–user matrix, user similarities are computed. Next, a weighted item list for the user is selected from other similar purchases made by other users in the system. Finally, the top N items in the recommendation list that have not yet been purchased by the user are presented [121].

The previously introduced k-NN algorithm is the most widely used algorithm in the user-based methods [84]. Given user U_m, the algorithm will determine the k nearest neighbors of $\{U_1, U_2, U_3, \ldots, U_k\}$. Items in the neighbors' group not rated by U_m are selected using an aggregation approach. Finally, the top (closest) n recommendations from the previous step are presented to the user. The main advantage of this algorithm is its accurate results and simplicity [85]. However, there are some drawbacks to using this method, such as scalability and sensitivity to data sparsity in databases. The k-NN algorithm uses a similarity measure to group each active user with similar k neighbors. In big databases such as those of Netflix or Amazon, generating a neighbor for the active user becomes too slow and the notion of “nearness” becomes problematic in high dimensions [97, 122].

The similarity between the active user and its neighbor is calculated using a similarity measure in the user–item matrix. One method of calculation is the *Pearson correlation matric* [123].

$$\boldsymbol{sim(U_a, U_b)} = \frac{\sum_{l=1}^{n}\left(R(U_a,I_l)-\overline{R}(U_a)\right)\left(R(U_b,I_l)-\overline{R}(U_b)\right)}{\sqrt{\sum_{l=1}^{n}\left(R(U_a,I_l)-\overline{R}(U_a)\right)^2 \sum_{l=1}^{n} R(U_b,I_l)-\overline{R}(U_b)^2}} \qquad \mathbf{2.4}$$

Where $\boldsymbol{U_a}$ and $\boldsymbol{U_b}$ are two users and considered as vectors in the item space where $n = |I_n|$.

2.3.1.1.1.2 Item-based CF methods

Instead of the user–item matrix, these systems build an item-item matrix. Based on the collected data, the algorithms group items based on specific relationship criteria. Predictions are made based on the user’s data, then these data are matched with the matrix. This approach can achieve a similar or higher performance than a user-based CF one [98]. The basic idea of the item-based method is to rate the items in the database, then match these rating values to the ones given by the users. Items with a rating similar to the user score are listed in the recommendation list [124].

This approach is used in the literature to overcome the scalability disadvantage of traditional CF techniques [121, 125] by creating an offline item–item similarity matrix, which enables the recommendations to be produced in less computational time. Through utilizing items features, item-based techniques have become more popular and successful in practice. For example, Amazon.com utilizes this method in many of its RSs which are available to the user [126].

In general, two categories of item-based CF methods are available [127]: static and dynamic. In dynamic methods, user feedback is adopted in the systems to produce recommendations, whereas the static variant presents new users with an already prepared list.

2.3.1.1.2 Model-based CF methods

Model-based CF techniques work by analyzing the data to discover the underlying characteristics and features in order to make predictions. The benefit of this method compared to the previous one is the shorter computational time. Therefore, it works better for RSs which contain a large amount of data. In contrast to the previous method, it groups the user–user, item–item, and user–item relationship into models with a limited number of parameters. The start-up computational time to calculate a relationship between the new user/item and the current one is shorter [53, 115].

Many mechanisms are employed in this type to cluster users or items into groups, such as the social network, Markov decision process, neural network, and Bayesian networks [88, 115, 116]. Therefore, the process of searching the space for similar entities is easier, yet it results in lower accuracy levels compared to memory-based methods. One reason for this is the approach sensitivity to data sparsity [116].

2.3.2 Content-Based Filtering

In CBF, also known as item-to-item correlation, a recommendation is made based on users' historic preference data. By considering the features of items that the user rated in the past, a recommendation about similar items is presented to the user. Thus, this type of system works on the assumption that if user U_i liked item I_j in the past, then they will like items similar to item I_n in the future [9, 12, 68, 75, 77, 128, 129]. This kind of RS is a domain-dependent algorithm which consider only the attributes of the items to make accurate predictions [12, 68, 83]. Therefore, the quality of the recommendation is highly influenced by the amount of information available about the users and items in the system [9, 12].

CBF is based on two aspects, namely the user's preferences and the product features [126]. Similar items to the ones in the user data are recommended [86, 99, 107]. In this system, recommendations are based on a comparison between the candidate items and the previously rated ones by considering purchase data, visited data, content similarities, or the preferences of the common users [86, 130]. Therefore, the analysis step of this approach is limited to the features in the user profile. Figure 5 depicts the CBF process.

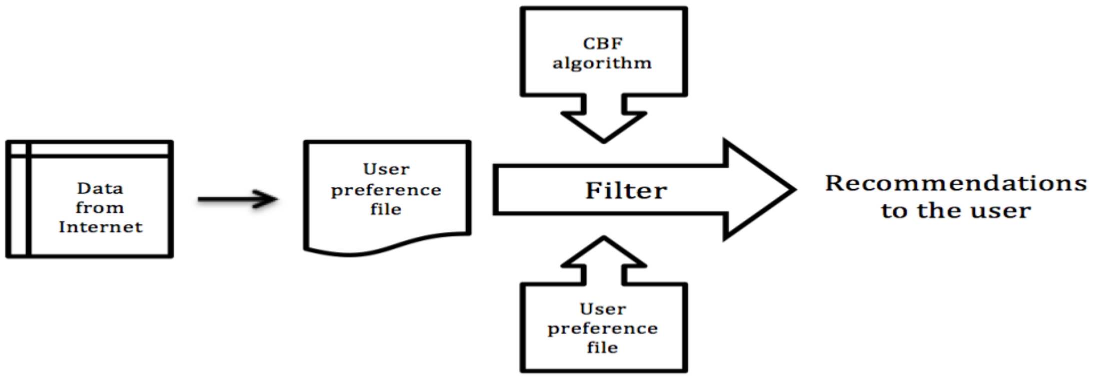


Figure 5. Content-based filtering diagram [3].

Two major challenges in CBF are the limited content involved and overspecialization [131]. In terms of the former, due to limited content the system lacks sufficient information about the items and users to correctly identify the best recommendations. For users, trust and privacy are the most common reasons for not sharing personal information [132]. As for the items, for instance, in music and image RSs, obtaining detailed content might be difficult or expensive [131]. Another example is in text RSs, where distinguishing between a well-written article and a badly one is impossible if both use the same terms [133].

As discussed earlier, item contents play an important role in CBF. Therefore, overspecialization occurs because the system keeps recommending items from the same category [131]. For instance, in a movie RS, movies for the same actors or from the same genre will be repeatedly recommended. A common solution to this challenge is to add random items in the recommended list [19]. Despite the mentioned drawbacks, CBF techniques have been used by many researchers to overcome some of the CF challenges such as sparsity, cold-start, and scalability [134], Table 1 summarizes the strength and limitations of each type of RS that has been presented.

Table 1. RSs techniques, advantages and disadvantages.

Technique	Advantages	Disadvantages
CF	• Machine independent. • Easy to analyze content. • Able to perform with limited content. • Quality can improve over time.	• Scalability. • Computational cost with large database. • Cold-start. • Grey sheep. • Sparsity.
CBF	• No need for other users' profiles. • Fast adjusting user profile change. • Overcomes cold-start problem. • Solve privacy. • Quality can improve over time.	• Limited content analysis. • Overspecialization. • Two items with similar features.
Model-Based CF	• Can handle data sparsity. • Improves CF performance. • Treat scalability. • Shorter computational time.	• Offline in nature.
Memory-Based CF	• Low cost implementation. • Easy to implement. • Shorter computational time. • New data can be added incrementally. • Considers content of items to be recommended.	• Very slow. • Requires scanning the entire data. • Computation complexity. • Limited scalability for large and complex datasets. • Sparsity causes the performance to decrease. • Dependent on human ratings. • Cannot handle cold-start users and items.
User-Based CF	• Accurate results. • Simplicity.	• Limited scalability for large and complex datasets. • Sensitivity to data sparsity. • Time consuming in large database.
Item-Based CF	• Overcomes scalability issue.	• Sensitivity to data sparsity.

2.3.3 Hybrid Filtering algorithms

In hybrid approaches, combinations of CF and CBF are used to increase the recommendation accuracy and overcome a single method's limitations [24, 40]. By combining different filters in the hybrid model, the strength of each one is emphasized, the limitations of one method are removed by another, and the model results in better recommendations [53, 66, 68, 77, 128, 135].

To combine both CF and CBF, there are several approaches used, as reported in the literature [17, 68, 75, 80, 126, 130]. Some researchers tend to implement each method independently and then combine their predictions, whereas others utilize a selected CF characteristic into CBF and vice versa and some integrate both methods into the same model.

Recent research has reported seven hybridization techniques that are used [80, 83, 101, 136]:

i. Weighted: an adjustable weight is given individually to the resulted recommendations from all the available techniques.
ii. Switching: the system switches based on the situation between the available techniques.
iii. Mixed: results in a combined recommendation result from the used techniques.
iv. Feature combination: uses features from different knowledge sources as an input to a single recommendation technique.
v. Feature augmentation: one recommendation technique is used as an input to another one.
vi. Cascade: the output of one recommendation technique is used as an input to the next one.
vii. Meta-level: the resulted model of one technique is used as an input to another one.

Although all these techniques have the same goal of increasing the accuracy of the recommendation, each has its own advantage and disadvantages. Consequently, deciding which hybridization to use is related to the domain and the application of RSs. For instance, in weighted hybridization, the strength of all the used RSs are utilized during the whole process. An online newspaper RS was developed by [137], whereby both CF and CBF are given the same weight, and then adjusted based on the resulted recommendations. Another positive example is the meta-level hybridization. As reported by [68], this kind of system may overcome the data sparsity challenge in CF, as the resultant model provides more valuable information compared to a single model. One reported downside is regarding the switching hybridization techniques. The switching criteria between the recommendation processes introduce more complexity and require predefined parameters to be available [68].

2.3.4 Other types of RSs

Due to the popularity of RSs in e-commerce and the benefit they achieve in the business, several other types of system have been introduced in the literature. For instance, **Social RSs,** also known as community-based RSs, refer to systems where items are recommended based on the social network of the user [129, 138], and it assumes that users within the same network will have similar taste. In addition, the advancement of mobile technologies has prompted the use of **Mobile RSs,** whereby recommendations are made based on the location of the users [129, 139]. Another example is **Graph-Based RSs,** which were originally developed for businesses with massive amounts of data, where the system creates graphs in which entities are represented by nodes and relationships are represented by edges [122]. These types of RSs are relevant for very specific application areas and are not further considered in this dissertation.

2.4 Real-World Applications of RSs

As discussed earlier, the applications of RSs in e-commerce are wide, and numerous researchers and practitioners have reported the benefits of using such systems. In this section, we present some real-world examples to demonstrate the benefits of RSs to business.

LinkedIn is one of the popular businesses for job seekers. On this platform, recommendations are made to suggest some people that the user might know. It helps the customers to narrow down their selection list from a large number of users to a few options. As was reported by [140] in 2018, there are approximately 500 million user in the system. The LinkedIn system is driven by data, information it collects from each user. By considering the different features and attributes of each user, the system then uses CF techniques to generate a short list of possible connections. Consequently, the system is able to suggest people that the user might actually know and grow their network on the site.

Netflix is another popular example of a hybrid RS. The main purpose of this business is to recommend movies and TV shows to customers as a mean to make profit. Using RSs, recommendations are presented to the user in the form of "because you watch this movie you might like this list," or a list of popular movies according to people in the same group age or demographic area. Thus, this system considers both users attributes, such as location, age, and gender; and movies features such as genre, actor, and year.

Netflix is an excellent example of a successful business in terms of attracting a large number of customers. However, it is also a good example of how to increase the business revenue by using RSs. Company executives Carlos A. Gomez-Uribe and Neil Hunt reported that 75% of what users watch comes from their recommendations, which saves the company significant sums of money in the marketing category. Consequently, the company saves approximately $1 billion each year [5].

Amazon is an e-commerce business that not only cares about the buyers but also the sellers. Doing business on Amazon is profitable for both businesses and consumers. Business owners can offer their products without caring about the marketing cost, as Amazon will take care of this. In addition, the customers have the luxury of a good recommendation list tailored and personalized for each customer.

Using CF, information about users' past purchases, items in the shopping cart, and items the customer only viewed are considered in the recommendation process. Using this method, the company credits 35% of their revenue to RSs [141]. In addition, they reported an increase of 29% of their total yearly sales [141], which means they made a total revenue of 177.87 billion in 2017 [142].

Best Buy is a well-known electronic shop. However, in recent years it has also begun an online shop. Customers now have the option to shop online and have the product either delivered to their address or collected from the store. In 2016, the competition increased between Best Buy and Amazon. Hubert Joly, the CEO of Best Buy, revealed in an interview the extent to which their profits are increasing and how Amazon is no longer a threat [143]. In the interview, he noted some of the benefits of using RSs and online stores, such as a faster and narrower delivery window and more relevant products recommendations, and how this helped to increase sales by 23.7 percent for the second quarter of 2016. Best Buy uses CF methods to make recommendations based on individual browsing history and purchases.

All the previous examples highlight the importance of RSs in e-commerce. We also discussed earlier the benefit of RSs to the users themselves, such as addressing the information overload. In their news article about Canadian habits in shopping online vs in-store, Cision, a global media intelligence company, reported that in today's world, and notably in today's settings, most shoppers prefer the comfort of their homes. As early as 2018, a survey finding by Ebates Canada was presented by Cision in [144]. They reported that 82% of Canadian participants made online purchases for clothing and accessories, with 62% of them indicating that the anytime/anywhere convenience of the shopping process was their main motivation. Others favoured the better pricing and deals, and the location of the brand. All this evidence shows the importance of tailoring business to meet the preferences and demands of today's shoppers. As noted earlier, the Covid-19 pandemic has also tremendously impacted the demand for online shopping, and considerably increased the number of cold-start users. It is foreseen that many individuals will continue to use online shopping, making the development of intelligent recommendations a crucial research topic. Next, we review the ML approaches that are typically utilized in personalized RSs.

2.5 Machine Learning Algorithms

There have been many examples in the literature that have utilized ML techniques in the RSs. As mentioned before, the main purpose of ML is to overcome some of the traditional recommendation methods limitations. Although hybrid filtering has been proposed as a solution to address CF and CBF limitations [24, 40], a number of research questions remain and ML for intelligent RSs is thus an active area of research. In the following subsections, several methods and algorithms are discussed. The ML research area is widespread and includes countless techniques. Therefore, we focus our attention on the ones employed in this dissertation.

2.5.1 Cluster analysis methods

In most cluster analysis techniques, which are unsupervised learning methods, the main goal is to create an overview of the entire dataset by partitioning the data into smaller groups. To this end, in ML, clustering techniques are used to discover data distribution, patterns, and the natural groups within data [32, 145, 146]. In an e-commerce setting, the main goal is to maximize the similarity within the same group and minimize the similarity between different groups [147]. For example, for each group, the opinions/likes/ratings of the users can be averaged to predict the recommendations for other users [68].

As discussed earlier, data sparsity is one challenge in the RSs. Therefore, clustering is used to overcome this and to improve the performance and prediction accuracy [89, 145, 146]. Clustering is used to partition the data into subgroups and reduce the high dimensionality in the systems [12].

Generally speaking, clustering methods can be divided into two subgroups: soft clustering and hard clustering [148]. In soft clustering, data points can belong to one or more groups. Depending on the used algorithm, each data point will have an assigned probability or likelihood of belonging to group x, y, or else. In the e-commerce setting, for instance, customer A will have a probability of 40% of belonging to group x, 30% of belonging to group y, and 30% of belonging to group z. However, in hard clustering, overlapping is not allowed, meaning each customer will belong to one group only or none. In RSs, soft clustering is often more desirable, because it aids in capturing more information about the users' interests [73].

The clustering methods are more of a subjective nature, meaning that each technique is used to achieve different goals based on the environment it is applied in. In the literature, there are many different types of algorithms that use different techniques to define the similarity between the data points. Therefore, clustering methods are further divided into three different categories based on the type of algorithm used, see Figure 6 [1]. These categories are further discussed in the following subsections. It should be noted that clustering algorithms include more than the three mentioned categories. As mentioned earlier, in this chapter we only discuss the algorithms and method that are a prerequisite to understanding the work presented in this dissertation.

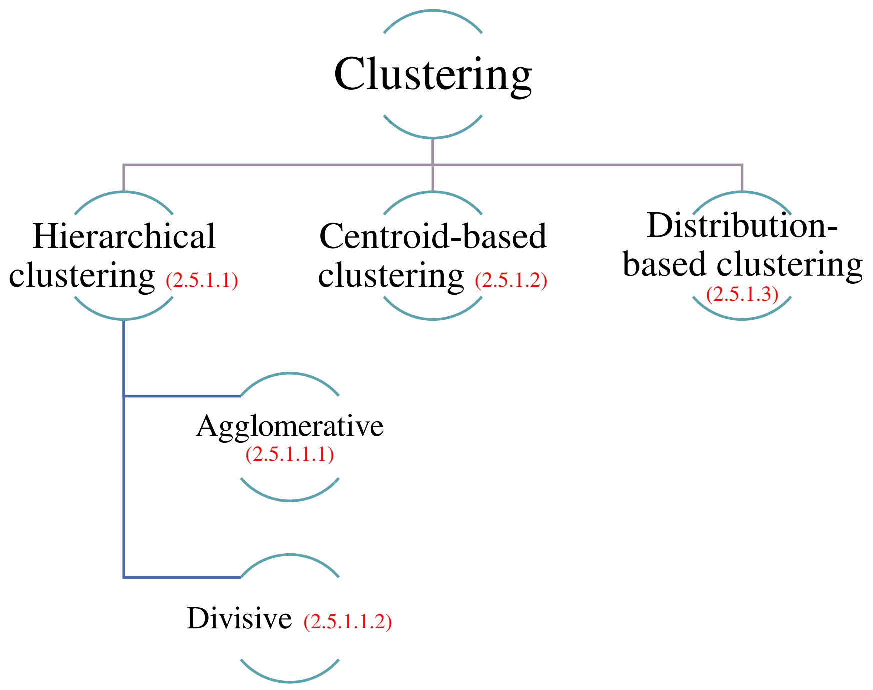

Clustering
Hierarchical clustering (2.5.1.1)
Centroid-based clustering (2.5.1.2)
Distribution-based clustering (2.5.1.3)
Agglomerative (2.5.1.1.1)
Divisive (2.5.1.1.2)

Figure 6. Taxonomy of clustering approaches [1].

2.5.1.1 Hierarchical clustering methods

In this type of clustering, the method creates a sequence of partitions, where each partition is then embedded into the following one in the sequence. Generally, this algorithm works by finding the hierarchical relationship between the data point. Subsequently, based on this finding, clusters are constructed following a tree shape [1]. One advantage of this type of clustering is that the tree can be cut down at the most appropriate level, which helps in reducing the tree size and keeping the required information [1]. The HC algorithms work by breaking down the entire dataset into smaller subsets until some criteria is achieved. Therefore, based on the constructing direction, HC can be divided generally into two types: agglomerative and divisive clustering [32].

In agglomerative clustering, i.e., the bottom-up approach, the method will start with each item being in their own cluster. During the following stages, the algorithm merges each cluster with the nearest one. The algorithm stops once it has a cluster that contains all the top records in the hierarchy [32]. Intuitively, the divisive or top-down method works in the opposite manner, starting with one cluster that contains all the top records then splitting the data down into smaller subsets. Generally, when creating agglomerative clustering, for each pair a distance value is calculated using some distance function such as mean distance, average distance, and distance of the centroid [1, 149]. Then, in every stage, clustering with the minimum distance is merged.

The main advantage of HC is it suitability for datasets with random structure and random attributes. It can easily detect the relationship among the clusters [1]. The downside, however, is the time complexity. The constructing time is highly affected by the number of clusters that require building [1].

2.5.1.2 Centroid-based methods

In contrast to HC, data points here are organized into non-hierarchical clusters. The general idea is to consider the center of the data as the center of the matching cluster [1]. In addition, this type is referred to as partitioning clusters, whereby the database is partitioned into k clusters and each contains N objects. The optimal partition is decided based on the objective function [32]. The quality of this type is overly sensitive to the initial partitioning. Overall, centroid-based methods have low time complexity and are very efficient in term of computing [1].

The most well-known method in this grouping is the k-means algorithm. Given k, the number of clusters, the algorithm will partition the dataset into k sub-populations. The advantage of this method is its reported high accuracy and efficiency, and its easy implementation [89, 150-152]. However, one of the challenges is that the k value must be predefined [153]. Furthermore, k-means produce a hard cluster, which mean that each point in the space can belong to, at most, one cluster

[148]. In addition, the original algorithm works by selecting random seeds and then calculating the centroid (or centre) for each cluster. Therefore, the choice of the initial centre can affect the final cluster [89, 146]. To overcome this challenge, many variants have been proposed in the literature. One example is the Cascade K-Mean, which selects the best k value according to the Calinski–Harabasz criterion, also known as the variance ratio criterion, created by [154].

Canopy clustering is another centroid-based method that involves multiple iterations. It initially starts with a dataset containing D instances and iteratively removes a single instance at random, with a replacement. This instance is added to the "canopy" subset [148]. For each instance, a distance metric is calculated and instances that are closer than T_1 is a distance threshold are added to the same cluster. In addition, the algorithm maintains a second distance function T_2 to remove instances from the original dataset, until the original set is empty [155, 156]. Threshold T_2 is employed to avoid processing new instances that are close to one another [156].

Canopy clustering has been used by many researchers as a pre-clustering step for distance-based algorithms such as k-means, to improve the performance and reduce data complexity. For instance, in [157], a canopy technique was employed to optimize the initial value of k-means. In nature, the canopy algorithm aims to learn from high-dimensional datasets by creating soft clusters, whereby the resulting overlapping subsets are called canopies. A strength of the canopy algorithm is that it requires only one pass over the data, and its ability to learn in batch or incremental mode makes it suitable for large-scale data [149]. In addition, it has been reported to produce clusters with at least the same accuracy as other methods but in a more computational, efficient way [158].

2.5.1.3 Distribution-based methods

In distribution-based methods, the basic idea is that each data point belongs to the same cluster that follows the same distribution [1]. The EM is an example of this type, which is also considered a soft cluster analysis method. Characteristically, the algorithm requires the user to specify the number of clusters. In EM, for each point it assigns a probability that it belongs to a cluster, typically using the range 0 to 1. The EM algorithm then proceeds by re-estimating the assigned probabilities by adjusting the mean and variance values to improve the assignment points, iterating until convergence [148].

Mathematically speaking, EM works by finding the parameters $\hat{\theta}$ that will maximize the log likelihood function of the observed data, as follows [159]:

$$\boldsymbol{p(X|\theta) = \sum_{Y} p(X, Y|\theta)} \qquad \textbf{2.5}$$

Where X represents the set of all observed data $X = \{\overrightarrow{x_1}, \dots, \overrightarrow{x_N}\}$, and Y donates the latent variables set $Y = \{\overrightarrow{y_1}, \dots, \overrightarrow{y_N}\}$. The complete dataset is represented as $(X, Y) = \{(\overrightarrow{x_\iota}, \overrightarrow{y_\iota})\}$ and the joint distribution is formed as $p(\vec{X}, \vec{Y}\middle|\theta)$, which is ruled by a set of parameters.

To reduce the difficulty of this task in cases of too much missing data, EM works by iterating between two steps. First is the *Expectation step (E-step)* where $p(Y|X, \theta^{old})$ is evaluated. Then, during the next step, *Maximization step (M-step),* a new parameter set is introduced $\theta^{new} = \arg max_\theta L(\theta)$. These two steps work toward increasing the objective likelihood function, and they are both repeated until a convergence criterion is satisfied [159].

The advantage of this algorithm is that clusters can overlap, which strengthens the relationship between instances and clusters. More importantly, by nature, this algorithm is very efficient in working with incomplete data [160].

In general, distribution-based algorithms deal with scalable data by changing the distribution on the number of clusters. However, the main challenge is the parameters involved in building the clusters. More importantly, it has been reported that the general performance is highly influenced by the initial setup [1].

2.5.2 Classification techniques

As previously mentioned, supervised learning, or classification, is about learning from examples where the class labels are known. Subsequently, the models built are then used to predict unseen cases [32]. Classification is a well-studied area in ML, and many families of algorithms exist. Throughout this thesis, we evaluate the performance of four well-known classifiers, namely two variants of decision trees, a Naïve Bayesian (NB) learner, and the k-NN approach. These classifiers belong to three ML categories: probabilistic learning, lazy learning, and eager learning, and are discussed next.

2.5.2.1 Probabilistic learning algorithms

The NB learner is based on the well-known Bayesian theorem and is used as our probabilistic classifier. The fundamental idea is to predict which class a given record or data point will belong to. In general, this classifier estimates the probability for each class then chooses the class with the highest probability. Given a data point $x = (x_1, x_2, \dots, x_n)$, the algorithm calculates the probability that c, which is the class, is the correct one using the following equation [161]:

$$\boldsymbol{p(c|x) = \frac{p\,(x|c)P(c)}{p\,(x)}} \qquad \textbf{2.6}$$

Naturally, the NB algorithm has the ability to learn incrementally, as it classifies instances one by one as they arrive. Moreover, it can predict and classify records based on their present features, without considering the absence of some other features. The main assumption is feature independence, which is the Naïve assumption in this algorithm [162, 163]. This assumption has both advantages and disadvantages. A disadvantage is that, in some settings, this assumption might mislead the correct classification by not considering the interplay between dependent features [164, 165]. In contrast, a key advantage is that it helps to reduce the impact of high-dimensional feature distributions, and many researchers have reported that it frequently performs well in practice [162, 163]. Overall, NB has the ability to work with limited memory resources, is robust, and is easy to interpret [162, 163].

2.5.2.2 Lazy learning method

As noted earlier, k-NN has been used extensively in classification and in RSs. This classifier is a lazy learner, which works by processing the instances only when requested [146]. It works by storing the whole training set then classifying new records only if they match one of the training examples [162]. In some situations, the algorithm fails in matching the new records to others in the training set, which was noted as a drawback by [162]. k-NN is generally easy to understand and implement. However, the main challenge in k-NN is computation cost [162].

Generally, k-NN consist of three key elements. First is the training data, which consist of a set or labeled records or data points. Second, is the distance or similarity matric to calculate the distance between the new unlabeled records and the rest of the labeled ones in the training set [162]. In the k-NN algorithm, deciding on which distance function to use to calculate the nearest neighbors is based on the attribute characteristic. It follows that choosing the right function is crucial, as an inappropriate choice may result in poor predictive accuracy [166]. For instance, when considering numeric attributes, the Euclidean distance function is typically used to calculate the neighbor's closeness [167], whereas in other research related to document classification, the Cosine measure has been reported as having better results [162].

One of the challenges in this classifier is deciding the optimal k value. If k is set very small, the algorithm might be sensitive to noise [168]. In addition, it may reduce the chances of exploiting many instances in the training set [169]. A large k value, meanwhile, means including too many instances from other classes. Furthermore, classes that contain a high number of classified instances can overpower the small ones, which biases the results [169]. For this reason, different research has suggested several methods to decide on the optimal value of k. The first is to use experimentation, starting with k equal to 1 then increasing the value by one for each test. Based on the results, the model with the lowest error rate has the best k value [166]. The second method is to use a plot to evaluate the different results of k, the so-called *elbow method* [170]. In this method, the algorithm is trained on the training set over a range of k values. For each value, the model accuracy is evaluated using the test set. The resulted accuracies are then plotted with their corresponding k values. This process is repeated by increasing the k value until the result of the used value is stable [170]. The optimal value to choose is the one that results in the highest accuracy at the elbow point of the graph. The last approach uses *d-fold cross-validation,* where the value of d is usually set to 5 or 10 [171]. This approach works on different iterations. First, the dataset is divided into d subsets of similar size. The first subset is held out to be used as a validation set. Classifier is then trained on the remaining subsets. The same process is repeated for all validation sets. For each value of k, the same process is repeated, and the average error is recorded. Finally, the value of k with the lowest error is chosen.

Earlier in section 2.3.1.1.1.1, we presented a general idea of how k-NN works. In this section, we present a high-level description of this algorithm that was provided by [162]. It states that in a given training set $D = \{(x_1, y_1), \dots, (x_n, y_m)\}$ and a test set of $x = (x', y')$, distance or similarity will be computed between the new record z and all the training records in $(x, y) \in D$. This calculation determines the nearest neighbor of D_z. In a last step, the class of the test instance is decided based on a majority class of the nearest neighbor [162].

2.5.2.3 Eager learning: Tree-based methods

We chose the well-known C45 decision tree (DT) and Hoeffding tree (HT) algorithms for this dissertation due to their interpretability and ability to generate accurate trees. These are examples of eager learners, as they scan the entire dataset and evaluate the different attributes within it [162].

The DT algorithm splits the data into subsets using a *divide and conquer* algorithm to create the final tree, using the information gain ratio measure to guide the decision. Divide and conquer works on two assumptions. First, if all the presented records belong to the same class C or to a very small one, then the leaf is labeled with the most frequent one. Otherwise, it chooses one attribute that have two or more outcomes. Next, this attribute is the root with two branches, one for each outcome. Finally, S is partitioned into different subsets according to the outcome of each. The same procedure is applied repeatedly for each subset [162]. Consequently, the DT algorithm perform multiple scans to partition the dataset. The algorithm terminates when each leaf in the tree has at least one example or until it reaches the point of having two examples with the same value belonging to two different classes [172].

One of the challenges in the DT algorithm is the multiple scans required to grow the tree. This might be hard to achieve in a large-scale application because of computational constraints. An advantage is that this algorithm has the ability to handle different types of attributes, such as continuous, discrete, and missing attributes. Furthermore, the models constructed by DTs are usually easy to explain to human observers.

The HT algorithm was developed by [173]. This incremental technique works by processing data as it arrives and only starts to build a tree when sufficient information about an attribute has been collected. The *Hoeffding Bound* formula is used to make a statistical decision about the split point in each node.

To be precise, the Hoeffding bound decides on the number of samples required to achieve a defined level of confidence. A strength of this is that the resulted tree can be updated at a given time while processing training examples and requires only one scan when compared to the DT approach. This classifier has the ability to examine each example as it arrives in the dataset. Thus, the only memory required is to store the tree itself. However, the memory limitation might still exist in the case of a very large tree for large-scale data.

2.5.3 Ensemble Learning

Ensemble learning can be defined as the process of combining a set of learning algorithms to produce one result [2, 174, 175]. As noted by M. and G. Valentini (2012), the general idea of combining different outputs is to mimic the human behavior of seeking several opinions before making a final decision. Formally, ensembles of classifiers consist of individually trained classifiers [176]. This combination can be of a set of learning machines, learning algorithms, or different views of data [174]. The main goal of an ensemble is to improve the performance of a single algorithm and produce more accurate predictions [2, 174, 175, 177]. However, the performance of an ensemble is highly influenced by the diversity of the base learners included and the characteristics of the data [178].

In this section, we discuss three ensemble learning methods, namely the Bagging, Boosting, and Random Subspace techniques.

2.5.3.1 Bagging ensembles

Created by Breiman in the early 1990s, Bagging ensemble learning is one of the earliest ensemble learning methods [179]. It is widely used due to its many appealing qualities. Bagging aims to maximize prediction accuracy through combining a group of base learners. Suppose we have a dataset, D, and N learners. The Bagging algorithm randomly resamples this dataset to obtain k bootstrap subsets. That is, each of the N learners is trained on different, resampled subsets of D. The model's output is combined by voting for the classification tasks [180]. In Bagging, each classifier is independent of another. Thus, Bagging models can be built in parallel. For a general idea about how Bagging works, see Figure 7 [2].

One of this algorithm's strengths is its computational ability, as it does not store the entire dataset while resampling. This is important in RSs where the dataset size increases over time. Note that because of bootstrapping, Bagging helps reduce variance; however, it is affected by class imbalance [181]. In RSs, users tend to favor one item over another, with the consequence that some items have a higher number of ratings than others. In addition, users' preference change over time, for instance, a teenage user might shift preference within the same year, or a pregnant woman who delivered her baby and is now shopping for regular clothes. For Bagging, this poses a challenge, as small changes in the training data can significantly change the built model [180]. Another challenge in Bagging is that the misclassification rate is highly influenced by the ensemble size; the larger it gets, the more negatively influential it is on the general performance [2].

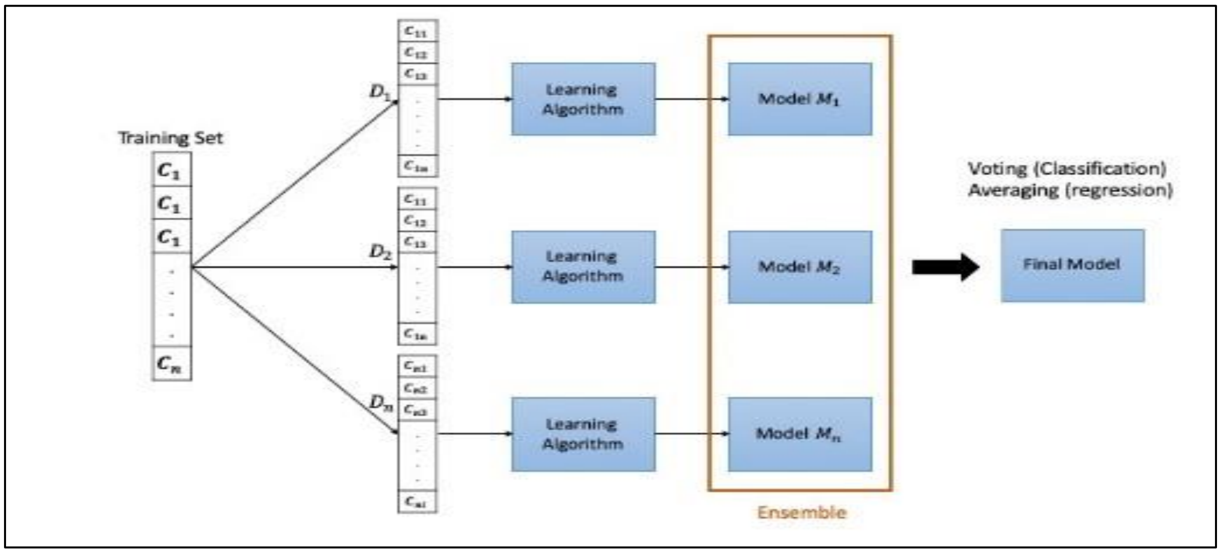


Figure 7. Bagging method illustration [2].

2.5.3.2 Boosting ensembles

The second ensemble we employ is Boosting, named for its ability to boost the performance of a weak learner, causing it to become a stronger one with higher accuracy [182]. A weak learner is defined as the one that achieves minimum improvements compared to random guessing [183]. The goal is to build a series of models using a continuous number of iterations while focusing on hard-to-learn examples. This is done by putting more weight on the examples with high classification errors. In Boosting, a new classifier is trained at each iteration to turn a weak learner into a strong one by emphasizing the examples that are misclassified. Each model helps to improve the performance of the previous one by reducing the weight error for each instance. The final outputs are combined using voting for classification tasks and averaging for the regression ones [180]. Unlike Bagging, Boosting is a sequential ensemble where the weighting of examples depends on the performance of the previous classifier [177]. The following Figure 8 presents a general illustration of how Boosting works.

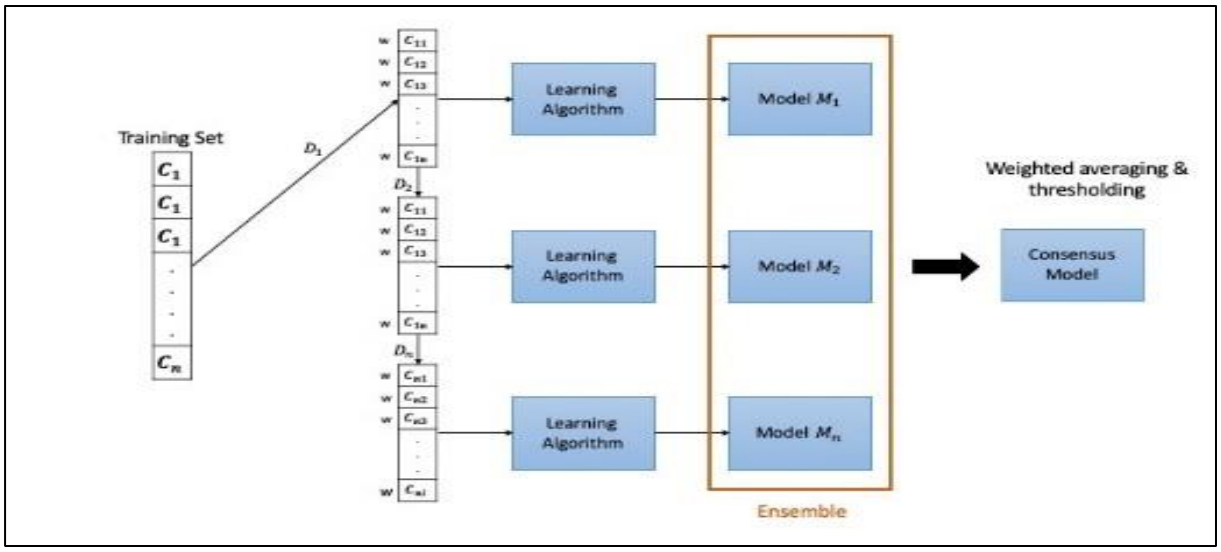


Figure 8. Boosting method illustration [2].

2.5.3.3 Random Subspace ensembles

Bagging methods learn from a created subset of random examples, whereas Boosting assigns a weight to each example which is readjusted during the learning process. However, neither method considers the importance of individual features when constructing a model. To this end, the Random Subspace method was introduced to guide the learning process [184]. Accordingly, the Random Subspace algorithm focuses on the attributes, or features, rather than the examples. With this approach, the subspace subsets are created using feature selection, evaluation, and reduction [185]. In Random Subspace methods, feature subsets are selected randomly with replacements from the training set. Subsequently, each individual classifier learns from the selected attributes in the subspace subsets while considering all training examples [184, 185].

In general, this method, depicted in Figure 9, works by sampling the training set in the feature space; algorithms are learned using a chosen subspace of the original input space [180]. The outputs are then combined using simple majority vote [180] or averaging [180, 186]. The main goal is to reduce the number of input features and reduce the dimensionality [174].

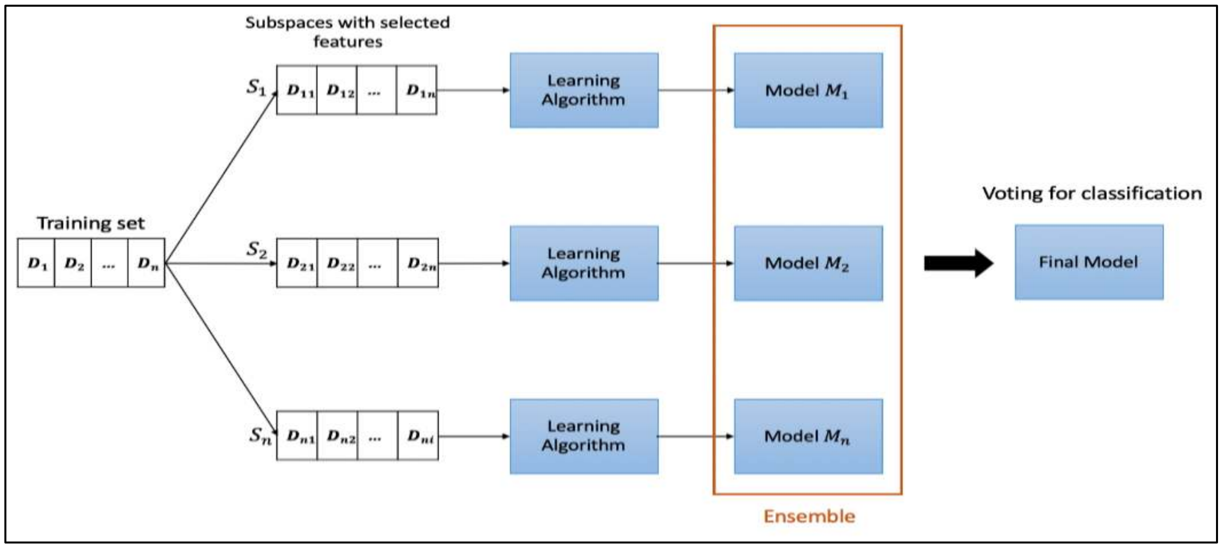

Subspaces with selected features
Training set
D_1 D_2 ... D_n
S_1 D_{11} D_{12} ... D_{1n}
S_2 D_{21} D_{22} ... D_{2n}
S_n D_{n1} D_{n2} ... D_{nl}
Learning Algorithm
Learning Algorithm
Learning Algorithm
Model M_1
Model M_2
Model M_n
Ensemble
Voting for classification
Final Model

Figure 9. Subspace method illustration [2].

The main advantage of the Random Subspace method is its suitability for datasets with high dimensions. The created subspace dimensionality is smaller compared to the original space, and the classifier is constructed in a random smaller subspace. Note that this method reduces the dimensionality size while keeping the number of training instances the same [186]. In the case of redundant features, this method obtains a better performance compared to a single classifier on the original training data in the complete feature space [186]. The challenge with this technique is its computational complexity, as each model must be weighed in accordance with the subset it was built on [187].

2.5.3.4 Discussion

Due to the numerous advantages of ensembles, studies have employed these approaches in problems related to RSs and ML. For instance, an RS for a human resources department employs Bagging and Boosting [188]. In this prior study, the outputs from both models are combined to create a user–interest model to recommend certain employment opportunities to the target user. Another example is presented by Lili, C. (2015), in which a Boosting ensemble is used to increase the recommendation accuracy and the recommender algorithm's ability to adapt to new data [189]. Other than the classification application, Bagging has been used as a post-processing step in a Random Subspace algorithm by [8]. In the following table, we present a summary of the strengths and drawbacks of the different ensemble techniques we use.

Table 2. Summary of ensemble learning methods advantages and disadvantages.

Method	Advantages	Disadvantages
Bagging	• computational ability. • models built in parallel.	• sensitive to data distribution. • performs multiple passes. • requires previous knowledge of data size. • influenced by ensemble size.
Boosting	• Works incrementally. • sequential models.	• performs multiple passes. • requires previous knowledge of data size. • influenced by ensemble size.
Random Subspace	• Reduces the dimensionality.	• Computational complexity.

2.5.4 Deep Learning

The general idea of ML is teaching the machine how to do a task such as classifying new records by learning from given data. With the rapid advance in ML research, several subfields have emerged to further improve the ML algorithms. Deep learning is a subfield of ML inspired by the human brain function in how it understands data and is thus based on artificial neural networks (ANN). In recent years, deep learning has become a trending topic of research within the ML community. Initially, deep learning was advancing and improving the research in image recognition, speech recognition, and natural language processing [190] before expanding to other domains. Generally, deep learning networks have many different networks structures. This mean that the structure can vary between a multilayer, nonlinear, or layer-to-layer structure style [190]. In ML research, the goal is to automatically extract and recognize the feature representation in the training data. Consequently, a combination of feature extraction from low-level will form a denser level of feature abstraction to aid ML [190]. In supervised ML settings, the deep learning networks will be trained using labelled data, whereas in unsupervised settings, these networks will learn from unlabeled data to capture distinguishable patterns.

In the RSs research field, deep learning has been utilized to enhance the recommendations quality and bring more value to the system, and applications of Deep Learning Recommender Systems (DLRS) have shown their ability to capture more complex feature representations compared to traditional RSs [190, 191]. Moreover, DLRSs have the ability to catch the relationship within the data regardless of the data source [190].

The area of deep learning research has been advancing widely. In this section, we focus on the DL models used in our PUPP-DA framework, as is discussed in Chapter 4.

2.5.4.1 Convolutional Neural Networks

A convolutional neural network (CNN) is, in essence, a feedforward neural network with convolution and pooling layers [190]. As the name suggests, a convolutional layer performs a so-called convolution, which is a linear operation, between the previous layer and a kernel (also called a filter or convolution matrix), which is essentially a small window [190, 192, 193]. The convolution is simply the element-to-element product between the parameters of the previous layer and the parameters of the kernel for every possible position of the kernel with respect to the previous layer [194]. The benefits are twofold: The convolution provides translation invariance, and the training is more efficient, as the number of parameters to be estimated for the kernel is much smaller than for its dense layer counterpart. Intuitively, the former entailed a much smaller number of parameters. Multiple kernels or filters are often associated with the same convolutional layer, each one of them in charge of capturing a particular aspect about the data. For instance, in image detection, each filter has a specific task, e.g., to detect eyes, nose, or shapes such as circles and squares. Each filter corresponds to a matrix with a predefined number of rows and column [190, 195]. A convolutional layer is often followed by a subsampling layer, better known as a pooling layer, which applies an aggregative function, such as maximum, minimum, or mean. Pooling is applied to the outcome of the convolution operation for each particular position of the kernel with respect to the previous layer, resulting in nonlinear subsampling and dimensionality reduction [190, 195].

The main advantage of the CNN architecture exists in its convolutional layer. This layer is used to replace the general matrix multiplication that is in standard neural networks [193]. By doing so, CNNs are able to reduce the complexity of the model and increase the model generalization by utilizing a shared weights architecture [190, 193]. Additionally, CNN pooling layers help to reduce neurons number in the model, which also reduces the model complexity [190].

In the RSs setting, a typical CNN consist of the following layers:

1- Convolution layer: to extract features from the input data. Uses a set of mathematical operation to generate a feature map [192].
2- Pooling (subsampling) layer: aims to reduce the dimensionality of the feature map to speed up the processing time [190, 192, 193].
3- Classification layer: the output of both convolutional and pooling layers is used within fully connected layers for the classification task [190, 192].

CNNs' ability to address the cold-start challenge is further discussed in Chapter 4. Next, we turn our attention to one-class learning, to be used when considering grey-sheep users.

2.5.5 One-class learning

Binary or multiclass classification algorithms aim to classify the training examples into pre-defined categories; this poses a challenge when dealing with imbalanced datasets. In this case, very few training examples belong to one class, usually called the *positive* or *target class*, whereas the majority or the rest of the examples belong to the negative class. The important challenge for a classification algorithm is that the positive class may have insufficient data to form a statistical representation in the classification model; it may not have a consistent representation, or it might involve human bias [196]. When these challenges occur, classification algorithms produce a model biased toward the majority class where the data are well characterized. One-class classification can overcome this challenge by using one class only to create a statistical or structural description of the available pattern in the training data [197]. Thus, it uses only the negative class to identify the learning patterns in the dataset [198].

In this approach and for classification problems, the used classifier calculates the similarity between the new instances and the instances in the target class [199]. The main challenge is its dependency on the amount of the training patterns and the degree of the classifier freedom. These two aspects highly influence the within-class and between-class generalization, which in turn influences the one-class classifier performance [200].

Another important application for one-class learning is in outlier detection. Outliers are the observations that do not follow the pattern or relate to the other observations in the system [30]. In this case, one-class classification is used to identify the data points that appears normal and abnormal by considering the training data distribution [201]. For instance, one-class learning is combined with deep learning in work done by [202], using two deep networks that are trained by challenging each other. Simultaneously, both networks collaborate to characterize the target class. This framework is applied to the detection of image and video outliers, with the goal of detecting samples that do not belong to the target class. Another application is in the Internet of things (IoT) application domain. The one-class learning is extended by [203] to retain the information structure of the training data while improving the accuracy of the detected outliers.

As already discussed, there are different uses for one-class in the literature. In addition, each author tends to use different algorithms or methods to improve the performance. One categorization was introduced by Khan, S.S. and M.G. Madden (2009), which distinguishes one-class classification into three types [196]. The first considers the availability of the training data. In this category, the algorithms learn from positive data only or will learn from both positive and unlabeled data. In the second category, one-class task is categorized based on the methodology used; either algorithms based on One Class Support Vector Machines (OSVMs) are used or ones based on other than OSVMs. The last category is based on the domain that the application is applied in. One class learning has been widely used in several domains such as RSs, text, recognition applications, medical analysis, and anomaly detections.

2.6 Summary

In this chapter, we introduced the concepts, algorithms, and methods used in this dissertation. We began by exploring the topic of RSs and their different applications, and then discussed the types of RSs, and their strengths and limitations. We also included several real-world examples of RSs to illustrate the benefits of such systems. This was followed by a discussion of ML algorithms and techniques. In the next chapter, we present our HCC-Learn framework, which addresses the data sparsity challenge.

Chapter 3. HCC-Learn Framework

3.1 Introduction

As discussed in the previous chapter, the use of RSs is an important task because of the added value they offer to the businesses. To meet the increasing demand for 24/7 online shopping, many organizations require more accurate, targeted and personalized recommendations.

A major problem associated with the current RSs solutions is that the number of items within customers' shopping carts typically constitutes only a tiny subset of those for sale. For instance, a customer of an online bookstore usually selects only a small number of books from those available to add to their shopping cart. This data sparsity problem may lead to inaccurate recommendations, as ML algorithms may not generalize well when a large dimensionality is involved. Furthermore, classification algorithms require class labels, which are frequently unavailable or late-arriving, as well as expensive to obtain. Specifically, resorting to manual labelling by domain experts is time-consuming and expensive, and consequently not realistic in an online business environment where the numbers of customers and items are huge.

All RSs face the challenge of collecting relevant information about users; for example, the above-mentioned data sparsity problem [20]. Furthermore, there is a need to group customers who purchase similar items together without having to resort to manual labeling.

As mentioned above, ML techniques are used to enhance recommendations. In particular, ensemble learning is known for its ability to enhance the performance of a single classifier [204] by focusing on hard-to-learn examples through procedures such as instance weighting or instance resampling. In doing so, the ensemble combines the strengths of base-level classifiers to improve the overall performance [205]. In the previous chapter, we discussed three ensemble learning techniques, namely the Bagging, Boosting, and Random Subspace approaches.

In this chapter, we present our HCC-Learn framework. Specifically, we introduce a hybrid cluster analysis and classification learning framework that combines unsupervised and supervised learning to obtain highly accurate classification models. We then study the impact of different cluster analysis techniques and report their impact on classification accuracy. We show that combining diverse classification algorithms, such as k-NN and ensembles, with cluster analysis methods generally produces high-quality results when applied to benchmark datasets [51].

In Section 3.2, we discuss related work to address data sparsity. We then present our HCC-Learn framework in the following Section 3.3. In Section 3.4, we detail the datasets, experiment setup, and evaluation methodology, and we discuss our finding in Section 3.5. Finally, in Section 3.6, we summarize the chapter.

3.2 Related work

In many studies, cluster analysis and classification algorithms have been combined within the same framework. An example is combining social network analysis with the study of human behavior to improve product marketing [206]. Moreover, several studies have reported that using cluster analysis as a preprocessing step, prior to classification, may lead to highly accurate models [51, 207].

Recently, researchers have been studying human behavior in an effort to improve the simulations while increasing the accuracy of ML algorithms, for instance, customer habits and day-to-day activities affect marketing campaigns and revenues. Specifically, in e-business, RSs have been used to gain customer loyalty and increase company profits. For example, Liao and Lee (2016) employed a self-clustering technique that addresses the high dimensionality challenge in the product matrix. By grouping similar products prior to the supervised learning, the classification algorithm produced accurate recommendations to the user while reducing the waiting time to provide an answer [12].

Another aspect that affects recommendation quality is the nature of the collected data. It follows that sparsity has a crucial impact on accuracy. Studies have addressed this problem with different solutions. For instance, Kanagal et al. (2012) introduced the taxonomy-aware latent factor model that combines various taxonomies and latent factor models; the cluster analysis is used to categorize the item matrix with the aid of manual labeling [207]. The objective of this research is to address the “cold-start” (i.e., incorporating unknown, anonymous users, or new items) as well as the data-sparsity problems. Another solution presented by Wang, H., N. Wang, and D. Yeung (2015) used deep learning to address sparsity in the dataset. In this work, Hierarchical Bayesian analysis is used to create a deep learning representation for both the items’ information and users’ ratings [208].

However, privacy and users' unwillingness create another challenge. Users tend to care about protecting their privacy from the unknown, so they prefer not to share personal information—or information they perceive as personal. For this reason, Nikolaenko et al. (2013) employed a hybrid approach with matrix factorization that enables the system to collect additional information about the items while preserving users' privacy [132]. In another research project, Guo, Zhang, and Thalmann (2012) created a simpler approach, in which the system essentially borrows information from the targeted user's neighbors. These neighbors are chosen from the user's trusted social network. The model merges the collected information with those relative to the targeted user to find similar users in the system's network [209].

Furthermore, data are collected continuously in mobile applications. However, few users are interested in rating their experience or the services they received, whereas some users keep returning to previously visited locations. The Rank-GeoFM algorithm was therefore developed, based on this observation, to collect check-in and check-out points that provide additional information to the system [210]. In a similar practice, Lian et al. (2014) created a location-based social network that groups items based on similar points of interest to solve data sparsity [211].

3.3 HCC-Learn Framework

In this section, we present our HCC-Learn which aims to address the following RQs, as explored through extensive study.

RQ1. Can data sparsity be alleviated by using unsupervised learning?
RQ2. What is the impact of cluster analysis on the general accuracy of RSs?
RQ3. How effective is cluster learning in improving the predictions response for existing users?

As described, we address the label and data sparsity problems through the combination of cluster analysis and classification algorithms. We conduct an extensive study of different types of cluster analysis techniques and report their impact on the classification accuracy. In addition, we introduce ensemble learning into the framework to evaluate the performance of a single classifier compared to an ensemble. The following subsection details our framework and introduces its main components.

3.3.1 Framework Components

Figure 10 shows the UML diagram of the HCC-Learn framework, which consists of four stages. In stage one, the original data are merged to obtain integrated information about the items and users. In most rating systems, the items' and users' ratings information are stored in separate matrixes. To this end, the rating matrix includes information about the users and an ID reference to the item. After exploring and understanding our datasets, we proceed with the cleaning and categorizing of the datasets. Data preprocessing is a crucial step, especially when considering the conversion of nominal data, the normalization of numeric data, and the determination of the best distance function, when applicable.

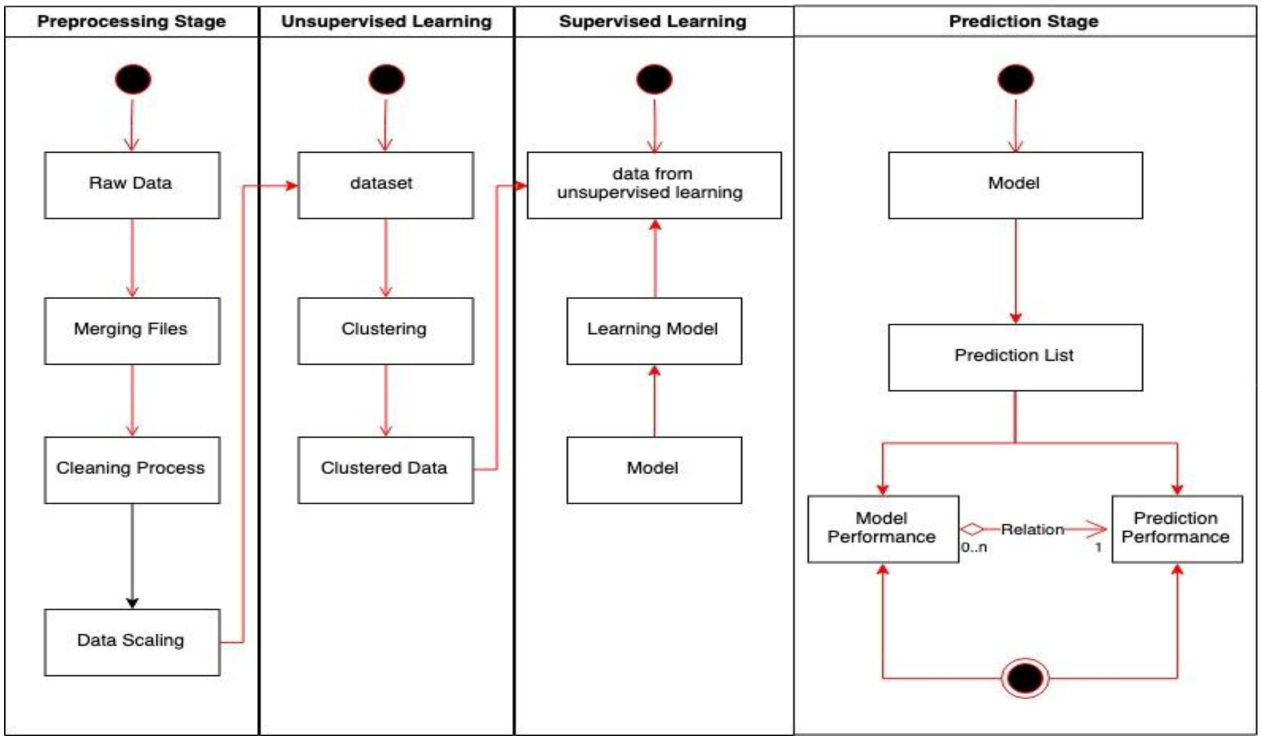


Figure 10. Diagram of the HCC-Learn framework.

Unsupervised learning is done in stage two, where n cluster analysis techniques $(A_1 \dots A_n)$ are applied to the pre-processed datasets. Cluster analysis algorithms group similar items into one cluster, attempting to minimize inter-cluster similarity. These algorithms include distribution-based, hierarchical, and centroid-based algorithms [212], as discussed in subsection 2.5.1.

Algorithm 1: HCC-Learn Recommendation Process.

```
Input
$D$: a set of $d$ class labelled training inputs,
$C_i$: Classifier;
$A_j$: Clustering algorithm;
$k$: Number of clusters;
$Y$: Class label of $d$ ;
$x$: Unknown sample;
p: size of each subset;
e: size of ensemble;
$s_i$: subspace;

Initialization for clustering stage:
   1- $A_j$ discover $k$ objects from $D$ as initial cluster centre
   2- Repeat:
      - (re)assign each object to cluster according to $A_j$ distance measure
      - Update $A_j$
      - Calculate new value
      Until no change
   3- Output models $(M_1, \dots M_n)$
   4- Split dataset into train $t_i$ and test $t_j$.

Initialization for Subspace method stage
   1- set size of p
   2- set size of $e$
   3- set base classifier as $C_i$
   4- create subspace as training set
   5- train the model for each individual $s_i$

prediction stage:
   1- Classify $(t_i, Y, x)$
   2- Output classification model $n$.
   3- Test model on $t_j$.
```

A strength of the HCC-Learn framework is that we employ multiple cluster analysis algorithms with different learning styles. Applying the algorithms (A_1 ... A_n) to the dataset results in n models being built, denoted by (M_1 ... M_n). Next, we conduct a cluster-to-class evaluation for each M_i. That is, each pair of clustering and classification algorithms is considered in the evaluation. That is, resultant dataset will contain an extra attribute that holds the value of the assigned cluster for each record.

The clustered dataset resulting from stage two is then used as an input for step three. In this supervised learning stage, we use m classification algorithms (C_1 ... C_m), once more employing techniques with diverse learning strategies. To this end, we employ probabilistic, eager, and lazy learners [122], as introduced earlier in subsection 2.5.2. The dataset is divided into training and test sets. The classifiers proceed to build models against the training set and test the models accordingly. Furthermore, we evaluate each clustering–classification combination using both a subspace and ensemble setting.

As a next step, each classification model is compared to each of the others. We evaluate the accuracy and select the clustering algorithm which demonstrates the highest improvement rate. It follows that this choice is domain dependent. Finally, we select the clustering–classification pair with the highest predictive accuracy to provide the user with a list of recommendations. Note that our framework is generic, in that it may incorporate many diverse cluster analysis and classification algorithms. In the next section we detail our experimentation evaluation.

The HCC-Learn framework provides one with a solid foundation to aid the decision on the clustering analysis method to be used in the following frameworks, as detailed in Chapter 4 and Chapter 5.

3.4 Experimental Setup

All our experiments were conducted on a desktop computer with an Intel i7 Core 2.7 GHz processor and 16 gigabytes of RAM. We implemented our framework using the WEKA data-mining environment [213].

3.4.1 Data Description

We used three datasets in our experimental evaluation. All datasets were generated using the customer rating for a specific product. Table 3 shows the dataset descriptions.

Table 3. Dataset descriptions.

Dataset	#Sample	#Attributes	#Classes	Used phase
Restaurant-consumer rating (RC)	1161	14	3	1 + 2
Fuel-Consumption rating (FCR) 2017	1056	14	5	1
FCR 2016–2018	3225	14	5	2

The first dataset is the Restaurant and Consumer (RC) dataset, obtained from [214]. The RC dataset contains information about the users and restaurants together with a user–item rating matrix, as shown in Table 4. Based on their overall ratings, customers are divided into three classes. This dataset was collected using a RS prototype to find the top *N* restaurants based on the customers' ratings.

Table 4. Restaurant and consumer (RC) dataset attributes.

User ID Ambience Place ID Transport Service Rating	Accessibility Area Parking (Y/N) Smoking Area Overall Rating	Alcohol Marital Status Price ($) Food Rating

We obtained the second dataset, Fuel Consumption Rating (FCR) from the Government of Canada Open Data Project[3]. Initially, we used the fuel-consumption collected data for only one year, 2017, denoted by FCR-1 and FCR-2, in the first evaluation phase of the HCC-Learn framework. To extend our evaluation of the model's performance and prediction accuracy in phase 2, we utilized an expanded version of this dataset that includes data for three years, 2016–2018 [215]. The details of this dataset are shown in Table 6. This dataset contains information about the fuel consumption of different type of vehicle based on factors such as engine size, number of cylinders, transmission type, etc. In the original dataset, the vehicle make attribute included 42 values. To reduce this number, attribute bonding was performed, and, based on the feedback from domain experts, two versions of the dataset were created. In the first version (FCR-1), we divided the vehicle makes into three categories, North American, European, and Asian. For instance, records for vehicles of makes such as Honda, Kia, and Toyota are all assigned to the Asian category. In the second version (FCR-2), we divided the vehicles into seven categories based on the country where they were designed—the United States, Germany, Italy, Japan, Korea, the United Kingdom, and Sweden. For both versions, vehicles belong to five classes according to their smog rating.

3 https://open.canada.ca/data/en/dataset/98f1a129-f628-4ce4-b24d-6f16bf24dd64

Table 5 shows the attributes information as defined in the original dataset we obtained from the Government of Canada Open Data Project.

Table 5. Fuel Consumption Rating (FCR) dataset attributes –attributes information

Attribute name	Type	Values
Model_Year	int64	2016, 2017, 2018
Make	Object	Ford, Chevrolet, BMW, Mercedes-Benz, GMC, Toyota, Nissan, Porsche, Kia, Jeep, Audi, Honda, Mini, Dodge, Hyundai, Mazda, Volkswagen, Volvo, Cadillac, Lexus, Jaguar, Subaru, Infiniti, Mitsubishi, Lincoln, Fiat, Buick, Chrysler, Ram, Maserati, Bentley, Acura, Rolls-Royce, Lamborghini, Alfa Romeo, Aston Martin, Genesis, Land Rover, Smart
Model	Object	Mustang, Focus FFV, Civic Hatchback, Jetta, Sonic 5, … , Regal AWD, Rogue, S60 Inscription T5, A4 Allroad Quattro, Mazda6 (i-ELOOP)
Vehicle_Class	Object	SUV – small, Mid-Size, Compact, SUV – Standard, Subcompact, Full-Size, Pickup Truck – Standard, Two-Seater, Station Wagon – Small, Mini Compact, Pickup Truck – Small, Minivan, Special Purpose Vehicle, Station Wagon - Mid-Size, Van - Passenger
Engine_Size	float64	2.0, 3.0, 3.5, 3.6, 2.5, 2.4, 5.3, 1.6, 1.5, 1.8, 1.4, 5.0, 4.0, 6.0, 6.2, 4.4, 5.7, 3.7, 2.7, 4.7, 3.3, 2.3, 5.5, 4.3, 3.8, 6.4, 5.2, 5.6, 6.6, 2.8, 4.6, 3.2, 6.5, 1.2, 6.7, 1.0, 6.8, 4.8, 8.4, 2.9, 0.9
Cylinders	int64	4, 6, 8, 12, 3, 10
Transmission	Object	AS6, AS8, M6, A6, A8, AS7, AM7, A9, AV, M5, AM6, AV7, AV6, M7, AS10, A5, AS9, AV8, A7, A4, AS5, AM8, AM9
Fuel_Type	Object	X, Z, E, D
Fuel_Consumption_City (L/100 km)	float64	10.8, 11.0, 12.4, 12.2, 10.2, … ,20.3, 6.4, 4.3, 21.3, 7.3
Fuel_Consumption_CityHWY (L/100 km)	float64	7.8, 8.3, 8.4, 8.5, 8.0, … ,16.5, 18.5, 4.0, 18.0, 5.3
COMB (L/100 km)	float64	8.4, 9.4, 10.4, 8.7, 9.8, … ,4.5, 16.5, 7.1, 19.6, 19.2
COMB (mpg)	int64	29, 22, 30, 27, 34, 26, 24, 25, … ,59, 63, 60, 49, 56, 52, 45, 69
CO2_EMISSIONS	int64	192, 259, 251, 210, 261, 250, 190, 197, 226, 197, 205, …
CO2_Rating	int64	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Smog_Rating	int64	2, 5, 6, 7, 8

Table 6. Fuel Consumption Rating (FCR) dataset attributes after preprocessing.

Vehicle Make.	Vehicle Model.
Engine Size.	Fuel Consumption in City.
Fuel Type.	Fuel Consumption in Highway.
Vehicle Class.	Fuel Consumption Combined.
Cylinders.	Fuel Consumption Combined mpg.
Transmission.	CO2 emissions.
Rating CO2.	Smog rating.

3.4.2 Experimental Setup

In this experimental evaluation, we evaluate the performance of four classifiers individually. We consider the DT and HT, NB, and k-NN learners. Theses classifiers belong to the eager, probabilistic, and lazy learning categories, respectively [122]. It is important to note that most RSs frameworks employ the k-NN algorithm, which is therefore recognized as the benchmark in this field. In addition, we employ the previously introduced Bagging, Boosting and Random Subspace ensemble methods. Note that in this work, we use the Random Subspace method implementation as available in WEKA. The original Random Subspace method developed by [184] uses a decision tree as the base learner. However, in WEKA, this method is a generic one that allows classifiers to be used as the base learner [213].

We employ five cluster analysis algorithms, namely HC, k-means, the cascade k-means, EM method, and the canopy clustering technique. These methods were chosen because of their ability to handle numeric attributes, nominal attributes, and missing values, as well as for the diversity of learning strategies they represent [122]. The number of clusters is set to equal the number of classes in each dataset.

For the k-NN algorithm, we determine that k = 5 is the optimal value for all our datasets. This value was set by experimentation. The number of base learners in ensemble learning is highly domain-dependent; this number was set to 25, in line with [216]. For the subspace size for the subspace method, we evaluate four sizes.

As discussed in Section 3.3 after stage two, cluster analysis, the dataset is divided into a training set (70%) and a test set (30%). To validate our model performance, we use 10-fold cross-validation. Cross-validation provides a realistic performance and results in a valid statistical sample with a smaller variance [204]. Note that the performance as reported in the following section 3.5 is based on the evaluation of the test sets against the resulted models as obtained from the 10-fold-cross-validation step.

3.4.3 Evaluation criteria

The selection of algorithms and parameters, and the evaluation of cluster analysis results remain topics of significant debate [217, 218]. In this work, as the ground truth in our datasets is always available, we evaluate the cluster analysis results using the well-known extrinsic cluster-to-class evaluation method. To evaluate the quality of the classification on the various datasets after clustering, we use the model accuracy as well as the F-score measures, which combines the precision and the recall rates.

However, as we are evaluating a RS framework, it is important to also take the prediction rate into consideration. Most studies of RSs evaluate the overall performance of the system. In line with this approach, we evaluate the overall performance of the system in Section 3.5.1. In addition, we are interested in evaluating the effectiveness and usefulness of the system. To this end, in Section 3.5.2 we consider the prediction rate and evaluate the effectiveness of the system.

3.5 Results and Discussions

Our main goal is to address the data-sparsity problem in RSs. This focus is motivated by the observation that these systems intrinsically contain a large number of items, whereas the requirement is to make a prediction based on a small number of items. As mentioned above, k-NN is commonly used in traditional RSs [80, 89] and thus acts as a baseline in our evaluation.

3.5.1 Cluster-to-Class Evaluation – System Usefulness

In this subsection, our aim is to answer RQ1 and RQ2 by assessing the impact of using cluster analysis via natural groupings in the data as a preprocessing step on classification accuracy. We tested each classifier separately using one of the above-mentioned clustering methods. In addition, we used Bagging, Boosting, and Random Subspace ensemble learners. In total, we tested 72 clustering-classification pairs during phase 1 of this experimentation [51].

Our results for the phase, as shown in Table 7 and Table 8, confirm that cluster analysis improves classification accuracy considerably, between 16.24% and 44.92%, compared to "no clustering.” Across all experiments, accuracies improved by an average of 29.5%. These results also indicate that EM, HC, and cascade k-means return the highest accuracies.

In the second phase, we present additional results when using the subspace method on four subspace sizes—25%, 50%, 75%, and 90%—against the extended FCR dataset, see Table 3. A total of 96 clustering–classification pairs was tested in this phase using the mentioned subspace sizes. As mentioned, this approach constructs models based on a subset of available features [184] with the goal of making more accurate recommendations from a small subset of items to each user. For instance, the system’s recommendation based on a customer’s previous purchase history may be very “accurate” in this particular context but entirely irrelevant to a customer who has experienced a shift in preferences or who has made a series of purchases to address a temporary (not ongoing) need. In these cases, the system would not contribute to the online business improvement, as it would enhance neither the sales, the revenues, or the profits, to mention just a few. Such a use of a Random Subspace method where the learning process focuses on only a small subset of features should prove to be beneficial: our aim is to assess such a scenario. In Table 9, the accuracies for the base learners are very similar to those depicted in Table 7 and Table 8. Again, the soft-clustering EM method resulted in better performance for almost all pairs. As explained earlier, this method employs two steps: an assignment expectation step followed by a re-centering or maximization step. Similar to the k-means algorithm, the covariance matrices and the weight associated with the various Gaussian distributions (clusters) are evaluated [148]. Iteration continues until convergence [148]. The benefit of this method is that it learns using a soft clustering approach, which provides an advantage in RSs [73].

Meanwhile, the HC method follows an agglomerative approach when creating the clusters. That is, it is a bottom-up approach that initiates each cluster with its own observation. Subsequently, pairs of clusters are merged as one progresses through the hierarchy. In our experimental evaluation, following the work of Witten et al. (2016) [33], we use the mean distance to merge these clusters. As mentioned earlier, the cascade k-means is a centroid-based method based on the Calinski–Harabasz criterion [154] that extends the simple k-means algorithm by creating several partitions. The algorithm starts with a small k, which is then cascaded from a small to a large number of groups. In contrast with the HC method, this is a top-down method. The cascade k-means algorithm iterates until it finds the right number of classes, which is an advantage over the k-means algorithm, as confirmed by our experiment.

Table 7. Phase 1 results, in term of accuracies.

Classifier	No-Clustering	HC	k-Means	Cascade k-Means	EM	Canopy	Increase over no-clustering	Dataset
k-NN	67.12	85.93	85.25	**88.23**	86.74	81.87	21.11	FCR-1
	69.82	86.74	87.42	85.12	**87.55**	79.57	17.73	FCR-2
	68.84	93.10	88.18	92.86	89.41	**93.35**	24.51	RC
HT	59.00	87.28	86.47	87.82	**88.50**	76.73	29.50	FCR-1
	59.95	84.84	86.47	81.46	**90.12**	81.46	30.18	FCR-2
	57.02	88.92	73.15	89.16	**98.52**	70.07	41.50	RC
DT	71.58	**96.75**	96.35	96.48	91.75	90.93	71.583	FCR-1
	73.34	**96.35**	92.29	94.72	91.34	89.45	73.342	FCR-2
	72.04	92.00	93.97	95.20	**99.14**	95.07	27.09	RC
NB	59.00	**89.04**	86.60	87.82	88.50	79.84	30.04	FCR-1
	59.95	84.17	86.47	81.46	**90.12**	81.46	30.18	FCR-2
	74.51	92.24	90.52	92.49	**99.14**	91.01	24.63	RC
Bagging - k-NN	68.74	86.74	86.20	**89.18**	87.28	82.14	20.43	FCR-1
	70.23	87.42	**88.23**	86.06	87.42	79.84	17.19	FCR-2
	70.44	92.98	88.55	93.60	90.39	**94.21**	23.77	RC
Bagging-HT	58.46	**89.45**	86.20	88.50	88.63	78.62	30.99	FCR-1
	58.86	84.98	86.06	81.60	**90.66**	77.00	31.80	FCR-2
	60.47	90.39	77.09	90.52	**98.28**	73.15	37.81	RC
Bagging - DT	56.97	**98.11**	96.35	97.70	92.96	90.80	41.14	FCR-1
	57.92	**96.75**	94.59	95.13	92.56	91.20	38.84	FCR-2
	72.54	92.49	94.21	96.31	**99.38**	96.06	26.85	RC
Bagging - NB	59.00	**89.31**	86.20	88.36	88.63	79.16	30.31	FCR-1
	59.00	84.98	86.06	81.19	**90.66**	80.51	31.66	FCR-2
	74.51	92.12	90.03	92.00	**98.89**	91.38	24.39	RC
Boosting -k-NN	67.12	83.36	87.01	**87.69**	**87.69**	79.43	20.57	FCR-1
	69.82	84.84	84.98	84.84	**86.06**	79.57	16.24	FCR-2
	68.84	93.10	88.18	**93.23**	88.42	92.37	24.39	RC
Boosting -HT	59.00	88.63	90.93	**92.83**	92.02	82.54	33.83	FCR-1
	59.95	88.23	88.77	87.01	**94.18**	84.17	34.24	FCR-2
	60.22	88.92	77.46	89.41	**98.52**	70.07	38.30	RC
Boosting - DT	54.53	**97.84**	96.08	97.16	95.54	94.86	43.30	FCR-1
	51.56	**96.48**	94.72	95.54	95.54	90.80	44.93	FCR-2
	70.81	92.61	92.61	96.31	**99.02**	96.68	28.20	RC
Boosting - NB	60.35	**94.72**	92.96	93.91	91.75	85.52	34.37	FCR-1
	60.76	87.82	91.07	89.99	**93.10**	83.76	32.34	FCR-2
	65.03	89.29	91.13	93.97	**98.15**	91.38	33.13	RC

Table 8. F-score results for phase 1.

Classifier	No-Clustering	HC	k-Means	Cascade k-Means	EM	Canopy	Increase over no-clustering	Dataset
k-NN	0.68	0.96	0.92	0.95	0.89	0.93	0.28	FCR-1
	0.00	0.98	0.96	0.97	0.87	0.93	0.98	FCR-2
	0.69	0.93	0.88	0.93	0.90	0.93	0.24	RC
HT	0.61	0.95	0.95	0.97	0.94	0.86	0.36	FCR-1
	0.62	0.85	0.70	0.89	0.89	0.70	0.27	FCR-2
	0.59	0.89	0.74	0.90	0.99	0.72	0.40	RC
DT	0.73	0.99	0.98	0.97	0.95	0.95	0.26	FCR-1
	0.75	0.98	0.93	0.95	0.90	0.86	0.23	FCR-2
	0.73	0.92	0.94	0.95	0.99	0.95	0.26	RC
NB	0.61	0.87	0.80	0.85	0.91	0.76	0.31	FCR-1
	0.62	0.84	0.82	0.82	0.89	0.74	0.27	FCR-2
	0.75	0.92	0.91	0.93	0.99	0.91	0.24	RC
Bagging - k-NN	0.69	0.98	0.96	0.97	0.87	0.93	0.30	FCR-1
	0.71	0.85	0.83	0.82	0.86	0.76	0.14	FCR-2
	0.71	0.93	0.89	0.94	0.91	0.94	0.23	RC
Bagging-HT	0.60	0.90	0.96	0.97	0.97	0.86	0.37	FCR-1
	0.62	0.84	0.80	0.79	0.89	0.74	0.27	FCR-2
	0.61	0.90	0.77	0.91	0.98	0.74	0.37	RC
Bagging - DT	0.57	0.92	0.98	0.98	0.96	0.96	0.41	FCR-1
	0.57	0.98	0.94	0.95	0.92	0.88	0.41	FCR-2
	0.74	0.93	0.95	0.96	0.99	0.96	0.25	RC
Bagging - NB	0.60	0.87	0.80	0.85	0.92	0.75	0.32	FCR-1
	0.61	0.84	0.82	0.82	0.89	0.74	0.27	FCR-2
	0.75	0.93	0.91	0.92	0.99	0.91	0.25	RC
Boosting -k-NN	0.00	0.97	0.96	0.97	0.87	0.93	0.97	FCR-1
	0.00	0.82	0.80	0.78	0.83	0.75	0.83	FCR-2
	0.69	0.93	0.88	0.93	0.89	0.92	0.24	RC
Boosting -HT	0.61	0.96	0.95	0.97	0.96	0.87	0.37	FCR-1
	0.62	0.88	0.73	0.71	0.91	0.75	0.29	FCR-2
	0.60	0.89	0.77	0.89	0.99	0.70	0.38	RC
Boosting - DT	0.55	0.99	0.98	0.98	0.97	0.97	0.45	FCR-1
	0.52	0.98	0.92	0.93	0.94	0.88	0.46	FCR-2
	0.71	0.93	0.93	0.96	0.99	0.97	0.28	RC
Boosting - NB	0.62	0.97	0.86	0.93	0.97	0.90	0.35	FCR-1
	0.63	0.90	0.90	0.89	0.89	0.80	0.27	FCR-2
	0.65	0.89	0.91	0.94	0.98	0.91	0.33	RC

Table 9. Results, in term of accuracies, in phase 2.

Classifier	No-Clustering	HC	k-Means	Cascade k-Means	EM	Canopy	Increase over no-clustering	Dataset
	Subspace of 25%							
k-NN	64.04	89.27	80.13	85.49	**94.64**	84.23	22.71	FCR-1
	60.88	92.74	83.60	88.01	**94.95**	82.97	27.57	FCR-2
	64.08	92.24	87.93	88.22	**94.83**	89.37	26.44	RC
HT	63.09	87.38	68.14	77.92	**93.69**	74.45	17.22	FCR-1
	60.25	88.96	74.45	76.03	**95.27**	78.55	22.40	FCR-2
	75.57	87.64	79.31	85.92	**91.95**	78.16	9.02	RC
DT	62.46	89.91	73.82	80.44	**91.48**	74.76	19.62	FCR-1
	54.57	90.54	79.18	80.44	**96.53**	77.92	30.35	FCR-2
	72.41	83.62	75.86	84.77	**93.10**	81.90	11.44	RC
NB	61.51	86.75	76.03	80.13	**94.01**	81.39	22.15	FCR-1
	58.99	93.69	77.60	82.33	**97.79**	81.07	27.51	FCR-2
	73.28	93.97	82.76	88.22	**95.98**	82.47	15.40	RC
	Subspace of 50%							
k-NN	66.25	90.22	82.33	85.49	**95.90**	88.01	22.15	FCR-1
	68.45	**94.64**	85.17	88.01	94.32	84.86	20.95	FCR-2
	74.71	94.25	91.09	92.24	**95.40**	92.53	18.39	RC
HT	64.04	87.70	74.13	79.18	**97.48**	74.76	18.61	FCR-1
	61.83	93.38	71.61	76.97	**98.74**	75.71	21.45	FCR-2
	61.78	91.95	70.98	84.48	**95.69**	74.43	21.72	RC
DT	67.51	90.85	81.39	79.50	**94.95**	78.86	17.60	FCR-1
	65.93	92.43	79.50	82.65	**94.32**	75.08	18.86	FCR-2
	77.59	93.39	75.29	85.63	**95.69**	91.95	10.80	RC
NB	60.88	88.01	78.86	80.13	**98.74**	82.33	24.73	FCR-1
	60.57	94.01	78.55	81.70	**98.42**	80.44	26.06	FCR-2
	77.59	94.83	86.21	89.66	**97.41**	89.37	13.91	RC
	Subspace of 75%							
k-NN	67.19	91.17	84.54	83.28	**95.90**	84.54	20.69	FCR-1
	66.88	93.69	84.54	86.44	**95.58**	84.86	22.15	FCR-2
	72.13	**95.69**	91.95	90.80	94.83	92.53	21.03	RC
HT	62.46	87.07	76.34	72.87	**99.05**	70.35	18.68	FCR-1
	59.31	92.74	62.46	72.24	**99.05**	70.66	20.13	FCR-2
	58.33	94.54	58.91	89.08	**96.84**	65.52	22.64	RC
DT	70.35	87.70	81.07	81.39	**94.01**	80.76	14.64	FCR-1
	70.66	91.48	79.50	82.97	**92.74**	78.55	14.38	FCR-2
	72.41	92.82	77.30	85.63	**95.11**	93.10	16.38	RC

Table 9. (continued).

NB	61.51	87.07	79.50	78.55	**99.05**	79.81	23.28	FCR-1
	58.68	92.74	76.97	79.81	**99.05**	80.13	27.07	FCR-2
	76.15	93.68	85.34	89.08	**97.41**	87.64	14.48	RC
	Subspace of 90%							
kNN	65.62	90.54	84.86	83.28	**96.53**	83.91	22.21	FCR-1
	67.82	**93.06**	83.28	86.75	92.74	83.28	20.00	FCR-2
	73.56	**95.11**	90.52	91.38	94.54	89.08	18.56	RC
HT	59.94	87.38	78.86	62.15	**98.74**	66.25	18.74	FCR-1
	59.62	92.43	54.26	69.72	**99.05**	67.19	16.91	FCR-2
	58.33	93.68	57.76	89.94	**97.99**	62.93	22.13	RC
DT	69.72	87.70	79.81	80.13	**94.01**	77.60	14.13	FCR-1
	69.72	93.06	78.86	82.97	**93.38**	80.13	15.96	FCR-2
	73.56	93.39	75.86	79.31	95.69	**96.55**	14.60	RC
NB	58.68	87.38	78.55	78.86	**98.74**	79.81	25.99	FCR-1
	59.62	92.43	75.39	79.81	**99.05**	79.81	25.68	FCR-2
	77.01	93.68	86.21	89.66	**97.99**	89.08	14.31	RC

Table 10. F-score results for phase 2.

Classifier	No-Clustering	HC	k-Means	Cascade k-Means	EM	Canopy	Increase over no-clustering	Dataset
	Subspace of 25%							
kNN	0.00	0.89	0.79	0.85	0.95	0.84	0.86	FCR-1
	0.00	0.93	0.83	0.88	0.95	0.82	0.88	FCR-2
	0.63	0.92	0.88	0.88	0.95	0.89	0.27	RC
HT	0.00	0.87	0.68	0.78	0.94	0.74	0.80	FCR-1
	0.00	0.89	0.19	0.76	0.95	0.78	0.72	FCR-2
	0.76	0.87	0.79	0.86	0.92	0.78	0.09	RC
DT	0.00	0.90	0.73	0.81	0.94	0.75	0.82	FCR-1
	0.00	0.90	0.79	0.80	0.97	0.78	0.85	FCR-2
	0.72	0.81	0.75	0.83	0.93	0.82	0.10	RC
NB	0.00	0.87	0.76	0.81	0.94	0.81	0.84	FCR-1
	0.00	0.94	0.78	0.82	0.98	0.81	0.87	FCR-2
	0.73	0.94	0.82	0.88	0.96	0.82	0.15	RC

Table 10. (continued).

	Subspace of 50%							
kNN	0.00	0.90	0.82	0.86	0.96	0.88	0.88	FCR-1
	0.00	0.95	0.85	0.88	0.94	0.85	0.89	FCR-2
	0.72	0.94	0.91	0.92	0.95	0.93	0.21	RC
HT	0.00	0.88	0.74	0.80	0.97	0.75	0.83	FCR-1
	0.00	0.93	0.72	0.77	0.99	0.76	0.83	FCR-2
	0.62	0.92	0.71	0.84	0.96	0.75	0.22	RC
DT	0.00	0.91	0.81	0.80	0.94	0.79	0.85	FCR-1
	0.00	0.92	0.80	0.83	0.94	0.75	0.85	FCR-2
	0.78	0.93	0.75	0.85	0.96	0.92	0.10	RC
NB	0.00	0.88	0.79	0.81	0.99	0.82	0.86	FCR-1
	0.00	0.94	0.79	0.82	0.98	0.80	0.87	FCR-2
	0.78	0.95	0.86	0.90	0.97	0.89	0.14	RC
	Subspace of 75%							
kNN	0.00	0.91	0.85	0.83	0.96	0.84	0.88	FCR-1
	0.00	0.94	0.85	0.87	0.96	0.85	0.89	FCR-2
	0.72	0.96	0.92	0.91	0.95	0.93	0.21	RC
HT	0.62	0.87	0.77	0.73	0.99	0.70	0.19	FCR-1
	0.00	0.93	0.63	0.72	0.99	0.71	0.80	FCR-2
	0.58	0.95	0.59	0.89	0.97	0.66	0.23	RC
DT	0.00	0.88	0.81	0.81	0.95	0.81	0.85	FCR-1
	0.00	0.91	0.80	0.83	0.93	0.78	0.85	FCR-2
	0.73	0.93	0.77	0.85	0.95	0.93	0.16	RC
NB	0.61	0.87	0.79	0.79	0.99	0.80	0.24	FCR-1
	0.00	0.93	0.77	0.80	0.99	0.80	0.86	FCR-2
	0.76	0.94	0.85	0.89	0.97	0.88	0.14	RC
	Subspace of 90%							
kNN	0.00	0.91	0.85	0.84	0.97	0.84	0.88	FCR-1
	0.00	0.93	0.83	0.87	0.93	0.83	0.88	FCR-2
	0.74	0.95	0.91	0.91	0.95	0.89	0.19	RC
HT	0.60	0.88	0.79	0.62	0.99	0.66	0.19	FCR-1
	0.00	0.93	0.55	0.70	0.99	0.67	0.77	FCR-2
	0.58	0.94	0.58	0.15	0.98	0.63	0.07	RC
DT	0.00	0.88	0.80	0.81	0.91	0.77	0.83	FCR-1
	0.00	0.93	0.79	0.83	0.93	0.80	0.86	FCR-2
	0.74	0.93	0.76	0.80	0.96	0.97	0.14	RC
NB	0.58	0.88	0.79	0.79	0.99	0.80	0.27	FCR-1
	0.00	0.93	0.76	0.80	0.99	0.80	0.85	FCR-2
	0.77	0.94	0.86	0.90	0.98	0.89	0.14	RC

A closer examination of the subspace results reveals that the best subspace size used in the experiments depends highly on the domain as well as the base learners, as expected. For instance, the EM and HC cluster analysis algorithms generally perform best when the subspace size is 50%, irrespective of the classification algorithm, whereas a subspace size of 25% resulted in a higher accuracy for canopy analysis than for the other sizes, and better results were achieved for k-means at a subspace size of 75%, meaning that some domains and learners need more or less features to construct accurate models against these datasets. If the feature subspace is too small, many useful features go unconsidered, whereas larger subspaces of 75% and 90% features may lead to lower accuracy, as the algorithms are considering too many redundant features.

Evaluating RSs is generally done via recall and precision, whereby these two measures are used to evaluate the truthfulness level of the model. Recall gives the ratio of the retrieved items considered notable by the user relative to the total number of relevant items, whereas precision provides the ratio of items retrieved by the used method relative to the total number of recommendations [89, 122]. We combine these two metrics into an F-score measure in our evaluation. This measure combines recall and precision into a single measure [204].

$$F - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.1}$$

The results, shown in Table 9 and Table 10, confirm the benefit of adding cluster analysis as a preprocessing step. In addition, by considering Table 9, which shows the accuracies results for all experiments, we can observe that the model performs poorly in most cases where cluster analysis was not used.

3.5.2 Predicting User Responses – System Effectiveness

By considering our results from the two phases, we evaluate the effectiveness of the system by using the three cluster analysis algorithms with the highest overall results, namely EM, HC and cascade k-means. That is, these three clustering methods are chosen based on their good performance, as shown in the previous section. This subsection is mainly intended to answer RQ3 and to further study the effect of clustering on alleviating sparsity, which is our first RQ.

3.5.2.1 Predicting User Responses, clustering-classification pairs

In this section, we consider the RC dataset to determine whether cluster analysis yields highly accurate recommendations to current users. That is, we explore the impact on the recommendation accuracy for existing users using 15 customers randomly selected from our test set. Here, our aim is to study whether we are able to accurately predict the ratings of existing customers, given that the number of items is high, whereas the number of ratings is low (data sparsity).

The number of ratings available for each user is shown in Table 11. It should be noted that the number of ratings by individuals is between 11 (0.95%) and 18 (1.55%), whereas the total number of ratings in the dataset is 1,161.

Table 11. Number of historic records available for the sample users in the RC dataset.

User ID	U1003	U1014	U1036	U1057	U1061	U1081	U1089	U1096
#records	13	11	12	11	18	11	14	11
User ID	U1104	U1106	U1112	U1114	U1122	U1128	U1137	
#records	12	18	13	11	12	11	14	

Table 12. Accuracy of the various clustering algorithms for the RC dataset.

	No clustering	HC	Cascade	EM	Canopy
U1003	69.23	100.00	92.31	100.00	100.00
U1014	54.55	90.91	100.00	90.91	90.91
U1036	66.67	91.67	100.00	91.67	91.67
U1057	54.55	100.00	81.82	100.00	81.82
U1061	77.78	100.00	72.22	88.89	100.00
U1081	63.64	90.91	90.91	100.00	100.00
U1089	92.86	92.86	92.86	100.00	85.71
U1096	72.73	90.91	100.00	90.91	100.00
U1104	58.33	91.67	100.00	91.67	91.67
U1106	61.11	88.89	94.44	94.44	77.78
U1112	76.92	92.31	92.31	100.00	84.62
U1114	81.82	90.91	100.00	100.00	90.91
U1122	66.67	100.00	58.33	100.00	83.33
U1128	100.00	100.00	90.91	100.00	90.91
U1137	71.43	100.00	100.00	92.86	64.29

We report the accuracies for individual user recommendations in Table 12. As already mentioned, in this experiment we are studying the impact of clustering the dataset on the performance of k-NN in RSs. Our results confirm that cluster analysis substantially improves the accuracy for existing user predictions. Indeed, in 14 cases out of the 15 (93.3%), one or more algorithms were able to obtain a perfect score against the test cases. Furthermore, it alleviates the negative effects associated with data sparsity that prevail the RC dataset. In addition, the table shows that cluster analysis algorithms yield comparable results, with EM having the highest accuracy 66.7% of the time. As noted earlier, EM allows for soft membership and does not assume an equal shape and size for the clusters. Indeed, customer recommendations and profiles are typically skewed, which implies that the EM method is highly suitable for such a scenario.

3.5.2.2 Predicting User Responses, clustering–ensemble learning pairs

In this subsection, we use the FCR-2 data from 2016 to 2018. Table 13 shows the number and percentage of ratings taken for each test subject to evaluate the system's effectiveness by predicting each user's choice based on the resulted classification model. In
Table 14, we depict the overall classification accuracy, whereas in Table 16 through Table 19, we illustrate the prediction rate for each test subject (user). Note that, based on our results depicted in section 3.5, the subspace size was set to 50%.

Table 13. Number and percentage of each test subject rating records.

	Total # in original set	% from the original set	# in training set	% from training set	# in test set	% from test set
Ford	304	9%	218	72%	86	28%
Hyundai	80	2%	56	70%	24	30%
Jaguar	65	2%	38	58%	27	42%
Lincoln	47	1%	29	62%	18	38%
Mini	86	3%	49	57%	37	43%
Audi	113	4%	76	67%	37	33%

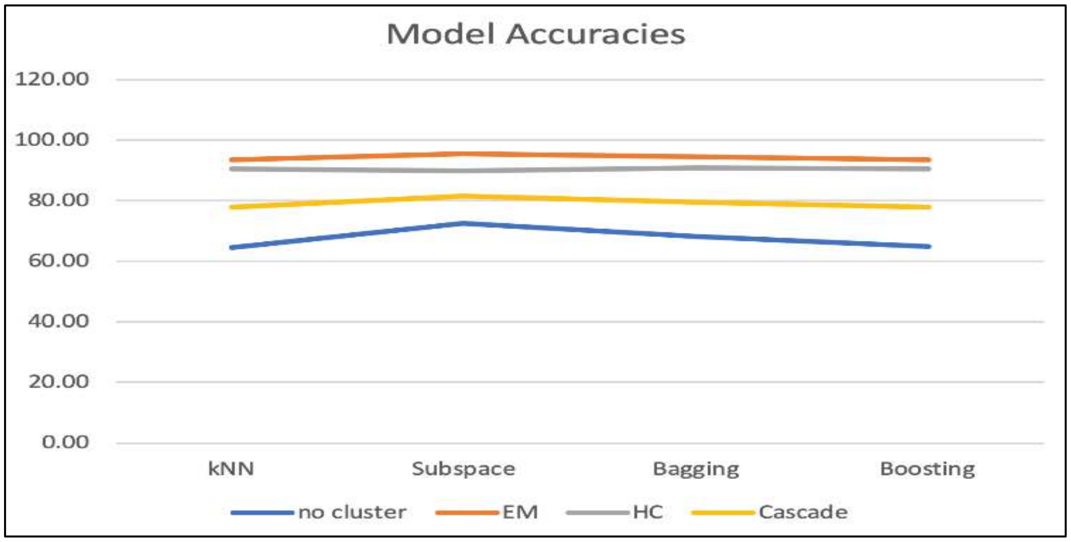


Figure 11. Model accuracies for user predictions.

Table 14. Accuracies for user predicting models.

Cluster \ Method	kNN	Subspace	Bagging	Boosting
No cluster	64.60	72.53	68.19	64.82
EM	93.62	95.57	94.42	93.62
HC	90.65	90.08	90.92	90.65
Cascade	77.98	81.48	79.44	77.98

Table 15. F-score for users predicting models.

Cluster \ Method	kNN	Subspace	Bagging	Boosting
No cluster	0.63	0.71	0.67	0.63
EM	0.94	0.96	0.94	0.94
HC	0.91	0.90	0.91	0.91
Cascade	0.78	0.81	0.79	0.78

In this section we further evaluate our models based on the prediction accuracies per user. To this end, six users were selected at random. As this dataset is about fuel consumption rating, we consider different vehicle makes as our test subject, the goal being to predict their rating, namely Ford, Hyundai, Jaguar, Lincoln, Mini, and Audi.

Table 16 depicts the prediction rate without using cluster analysis as a preprocessing step. The results in Table 17 through Table 19 show the prediction rate for each type of cluster analysis against each user. In this section, we focus on the performance of the ensemble learning method. We investigate the Subspace ensemble method, and therefore test Bagging, Boosting, and the Subspace algorithms.

A review of Table 14 and Table 15 shows that using cluster analysis resulted in better performance for all ensemble methods used in this experimental evaluation. We also note that for this dataset, HC improved the performance for all ensemble methods over the use of no clustering. However, the performance is similar for all ensembles, whereas employing EM resulted in a better performance for all ensembles. From the results shown in Table 14 we can conclude that for this dataset, the best performance is achieved when utilizing the EM–Subspace pair.

The following four tables show the results of the prediction rate for individual users. In these tables, we note that clustering improves the prediction rate for individual users. Furthermore, we observe that Subspace methods improve the prediction rate for both EM and cascade cluster analysis as a pre-processing step. In contrast, the results of Table 18 suggest that the prediction rate is lower when the Subspace method is used together with the HC algorithm.

Table 16. Predictions rates for users without using cluster analysis.

	kNN	Subspace	Bagging	Boosting
FORD	69.00	71.00	72.00	69.00
HYUNDAI	68.00	80.00	72.00	68.00
JAGUAR	20.00	20.00	26.67	20.00
LINCOLN	73.33	73.33	73.33	73.33
MINI	50.00	43.33	46.67	50.00
AUDI	74.29	68.57	74.29	74.29
average	59.10	72.90	60.83	59.10

Table 17. Predictions rates for users using EM analysis.

	kNN	Subspace	Bagging	Boosting
FORD	93.00	95.00	93.00	93.00
HYUNDAI	87.50	87.50	87.50	87.50
JAGUAR	93.33	93.33	86.67	93.33
LINCOLN	100.00	100.00	100.00	100.00
MINI	100.00	96.67	100.00	100.00
AUDI	97.14	100.00	100.00	97.14
average	95.16	95.42	94.53	95.16

Table 18. Predictions rates for users using HC analysis.

	kNN	Subspace	Bagging	Boosting
FORD	86.00	82.00	82.00	86.00
HYUNDAI	88.00	84.00	84.00	88.00
JAGUAR	93.33	84.00	93.33	93.33
LINCOLN	80.00	80.00	80.00	80.00
MINI	100.00	100.00	100.00	100.00
AUDI	85.71	88.57	88.57	85.71
average	88.84	86.43	87.98	88.84

Table 19. Predictions rates for users using cascade analysis.

	kNN	Subspace	Bagging	Boosting
FORD	70.00	73.00	67.00	70.00
HYUNDAI	80.00	84.00	80.00	80.00
JAGUAR	86.67	93.33	86.67	86.67
LINCOLN	80.00	86.67	73.33	80.00
MINI	80.27	82.54	81.20	80.27
AUDI	82.86	77.14	82.86	82.86
average	79.97	82.78	78.51	79.97

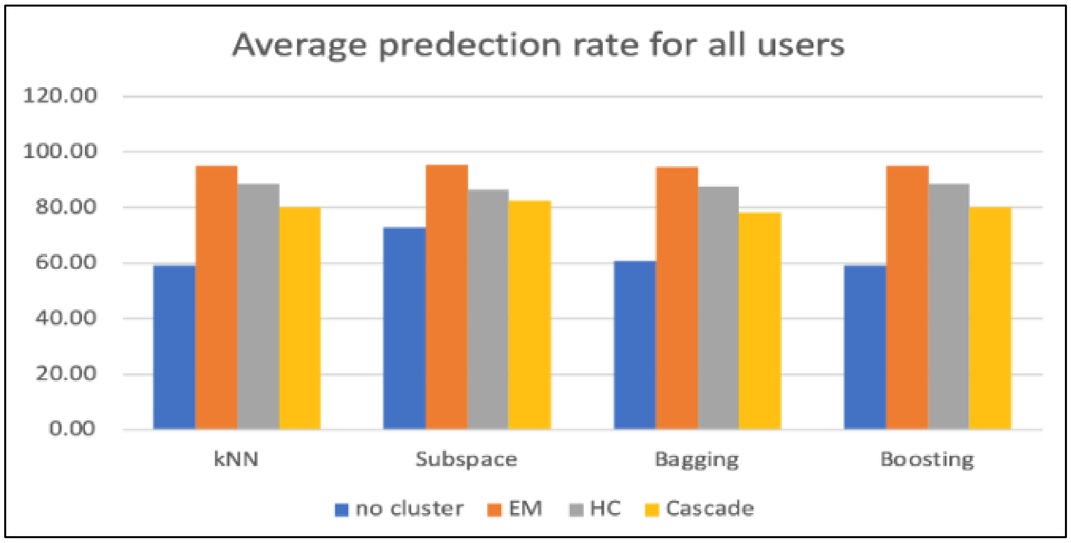


Figure 12. Average prediction rates with and without cluster analysis.

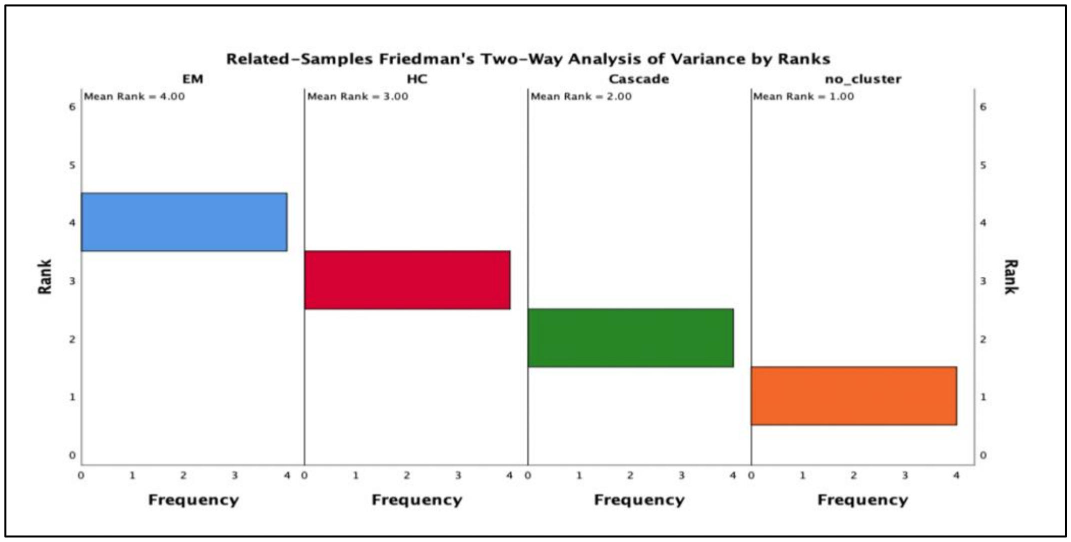


Figure 13. Cluster analysis rank.

3.5.3 Statistical validation

In this section, we use the Friedman's test for our statistical validation step. Friedman's test is a non-parametric equivalent of the repeated measures ANOVA. Both Friedman's test and ANOVA are used for comparison of multiple classifiers. However, as reported in [219], ANOVA's basic assumption is that the drawn samples come from a normal distribution, which is not the case with most ML algorithms. Thus, Friedman's test is a solution in the ML area. This test ranks the algorithms for each dataset separately [219]. The null hypothesis tested here is that the classifiers' performance is the same, and differences among them are based on random. Therefore, if the null-hypothesis is rejected, we can proceed with a Nemenyi's pair-wise comparison post-hoc test. As Demšar, J., (2006) discussed the power of this post-hoc test [219]. According to Demšar, in a ML setting, the Nemenyi's post-hoc test is the correct method to understand the performance of two or more classifiers, especially when we compare all classifiers to each other. Furthermore, it is best to decide if there is a significant difference between the two tested classifiers.

Table 20. Hypothesis test summary.

Null Hypothesis	Sig.	Decision
The distribution of EM, HC, and Cascade are the same	0.007	Reject the Null Hypothesis

The resultant p-value is 0.007383 which indicates a significant difference between the three cluster analysis algorithms. The related-samples Friedman's two-way analysis of variance by ranks, Figure 13, shows the mean ranks for each cluster analysis.

The p value is used to reject the null hypothesis, which states that all groups are from the same distribution and there is no significant difference among them. Subsequently, the alternative hypothesis that one or more algorithm is different is evaluated by using a post-hoc pairwise comparison test. First, we perform a pairwise comparison, as shown in Table 21.

Table 21. Pairwise comparisons.

Method 1- Method 2	Test Statistic	Std. Error	Std. Test Statistic	Sig.
No cluster-Cascade	1.000	0.913	1.095	0.273
No cluster-HC	2.000	0.913	2.191	0.028
No cluster-EM	3.000	0.913	2.286	0.001
Cascade-HC	1.000	0.913	1.095	0.273
Cascade-EM	2.000	0.913	2.191	0.028
HC-EM	1.000	0.913	1.095	0.273

In this table, each row tests the null hypothesis that cluster *method 1* and *method 2* distribution are the same using a significance level of 0.05. We note that the pairs *no cluster-HC, no cluster-EM,* and *Cascade-EM* have significant differences.

Next, we conduct an Nemenyi post-hoc test to determine the critical difference (CD) of where the pair significantly differ from one another, as shown in (eq. 3.2) [220]. Our results reveal that there is a significant difference between the EM and Cascade analysis, as shown in Table 22.

$$\boldsymbol{CD = q_a \sqrt{\frac{k(k+1)}{6n}}} \qquad \textbf{3.2}$$

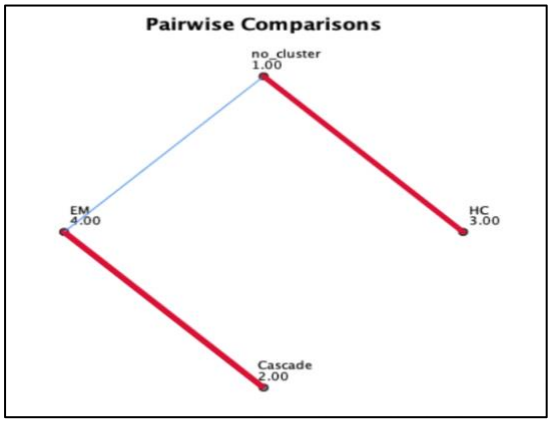


Figure 14. Pairwise comparison of the number of successes shown at each node.

Table 22. Nemenyi *p*-values.

	No cluster	EM	HC
EM	0.005602		
HC	0.125707	0.692333	
Cascade	0.692333	0.125707	0.692333

Our results indicate that pairs using the EM, HC or Cascade cluster analysis algorithms resulted in superior performance compared to other methods when considering all ensemble types. In addition, the statistical test confirms no significant difference between EM and HC when used in collaboration with the Bagging, Boosting, or Feature Subspace ensembles.

3.6 Summary

In this chapter, we introduced the HCC-Learn, a multi-strategy framework that combines multiple cluster analysis and classification algorithms for RSs. In general, RSs are challenged by labeling and data sparsity, and this framework was created to address these challenges. In this chapter, we evaluated the impact of different clustering techniques within the same domain, using the same framework and dataset. We evaluated this framework by using a different combination of cluster analysis and classification algorithms. Specifically, we employed three different types of ensembles to evaluate the value of this method within our framework. Our results confirm that a combination of cluster analysis and classification algorithms generally benefits the learning process. In addition, combining soft clustering with an ensemble based on feature subsets produces superior results when applied to our datasets. Furthermore, when considering all type of clustering we used in this experiment, we can conclude that both soft and hierarchical clustering results in high performance. We conclude that cluster analysis clearly benefits learning, leading to high predictive accuracies for existing users.

In the next chapter, we address the cold-start problem and evaluate the effectiveness of active deep learning on the quality of the recommendations.

Chapter 4. PUPP-DA – Deep Active Learning for new users

4.1 Introduction

As discussed in Chapter 2, a main goal of RSs is to address the information overload users experience and aid the users in narrowing down their purchase options. These systems aim to achieve this objective by understanding their customers' preferences, not only by recognizing the ratings they give for specific items, but also by considering their social and demographic information [8]. Consequently, these systems create a database for both items and users in which ratings and reviews of these items are collected [9]. The more information and ratings are collected about the users, the more accurate are the recommendations the systems make [10].

Deep learning algorithms have had considerable success in industry and academia recently, especially when addressing complex problems that involve big data, focusing on domains such as image processing and text analysis [13, 14]. Notably, deep learning has been employed in RSs which involve movies and music [13, 14, 15]. Deep learning methods are used to extract hidden features and relationships and build on earlier work within the field of neural networks [16]. As mentioned previously, the advantage of deep learning techniques comes from their ability to construct multi-layer, nonlinear, and layer-to-layer network structures [190]. Therefore, in RSs they effectively capture the nonlinear and insignificant user-item relationships [190]. Deep learning technology also has the ability to use diverse data sources to make accurate recommendations and overcome the data sparsity and cold-start problems [221, 222]. Specifically, deep learning based on CNNs [13, 14, 19] has been employed extensively within the RSs domain due to their known success in computer vision and mining domains.

In this chapter, we present the Popular Users Personalized Predictions (PUPP-DA) framework, designed to address the cold-start problem in RSs. We combine cluster analysis and active learning, or so-called user-in-the-loop, to assign new customers to the most appropriate groups in our framework. In this approach we construct user segmentations via cluster analysis. Subsequently, as new users enter the system, classification methods intelligently assign them to the best segment.

Based on this assignment, we apply active learning to describe the groups. That is, cluster analysis is used to group similar user profiles, whereas active learning is employed to learn the labels associated with these groups. Our PUPP-DA framework includes deep learning by incorporating CNNs into the learning process.

The remainder of this chapter is organized as follows. In Section 4.2 we discuss related work, and then we present our PUPP-DA framework in section 4.3 along with its components. In section 4.4 we outline our experimental setup and data preparation and in section 4.5 we detail the results. Finally, we summarize the chapter in Section 4.6 .

4.2 Related Work

In active learning, or user in the loop, a ML algorithm selects the best data samples to present to a domain expert for labelling. These samples are then used to bootstrap the learning process, in that these examples are subsequently used in a supervised learning setting. In RSs, active learning presents a utility-based approach to collect more information about the users [223]. Showing the users a number of questions about their preferences, or asking for more personal information such as age or gender, may benefit the learning process [27], notably for cold-start users.

The literature addressing the cold-start problem [224] is divided into implicit and explicit approaches. On the implicit side, the system uses existing information to create its recommendations by adopting traditional filtering strategies or by employing social network analysis. For instance, Wang et al. (2017) relied on an implicit approach based on questionnaires and active learning to engage the users in a conversation aimed at collecting additional preferences. Based on the previously collected data, the users' preferences, and predictions, they used an active learning method to determine the best questions to be asked [27]. Similarly, standard explicit approaches may be extended by incorporating active learning methods in the data collection phase [224]. Fernandez-Tobias et al. (2016), for example, used an explicit framework to compare three methods based on the users' personal information [225]. First, they included the personal information to improve a CF framework performance; then they used active learning to further improve the performance by adding more personal information from existing domains. Finally, they supplemented the lack of preference data in the main domain using users' personal information from supporting domains.

There are many examples in the literature of ML techniques being utilized in RSs. Although, as noted before, hybrid filtering was proposed as a solution to the limitations of CBF and CF, it does not adequately address cold-starts. To this end, Pereira and Hruschka (2015) proposed a simultaneous co-clustering and learning framework to deal with new users and items. According to their data mining methodology, a cluster analysis approach is integrated in the hybrid RSs, which results in better recommendations [66].

In addition, performances may be improved by implementing classification according to association rule techniques [40]. Such a system was built to deal with sparsity and scalability in both CF and CBF approaches. In [226], clustering and classifications are used to identify criminal behavior. Furthermore, Davoudi and Chatterjee in [59] use clustering to recognize profile injection attacks. Both methods apply clustering techniques to create user segmentations prior to classification.

As discussed, there are many solutions to address the cold-start challenge. Active learning has been a successful one; however, it depends on the user's willingness to provide answers. In this work, we use implicit active learning whereby we utilize the available information about the users. We further incorporate deep learning in an attempt to capture more valuable relationships from within the system. The next subsection provides related work for deep learning in RSs.

4.2.1 Deep Learning in RSs

Deep learning methods, and specifically CNNs, have been successfully used to solve complex computational problems within the RSs domain [227, 228]. For example, in [228], a Siamese CNN architecture was used in a clothing RS to capture the latent feature space from clothing images as available in the system. These features were integrated with other available data, such as personal interest and fashion style, and fed into the personalized recommendation model using probabilistic matrix factorization. A similar approach was employed by [229], where a CNN was also used to extract more features to deal with cold-start recommendations. In this work, the authors employed the CNN approach to extract textual item descriptions that were fed into two different recommendation models based on item, time, and text correlations. In [222], text and images from the users' browsing history were both utilized to make article recommendations. In this study, a CNN is first used to create a text eigenvector, and a Visual Geometry Group method is used to construct the corresponding image eigenvector. This combined eigenvector is input into a multilayer perceptron that outputs the recommendation.

As discussed above, it follows that employing active learning in RSs produces some promising results when considering the customer cold-start problem. However, the integration of active learning within a deep learning paradigm has not been widely explored in this scenario [31]. As reported in [31, 32], the use of deep active learning in detecting cancer through the selection of informative samples and training a CNN showed some success. Similarly, in [33], active learning was used to label new images prior to training. To the best of our knowledge, this is the first work that studies the use of deep active learning to alleviate the cold-start and data sparsity problems in RSs.

4.3 PUPP-DA Framework

The presented PUPP-DA framework focuses on answering the following research questions. It should be noted that the presented results are highly domain-dependent and apply for the datasets used in this study.

RQ1. Can active learning be used to improve the results of RSs?

RQ2. What is the value of deep active learning for RSs?

RQ3. Which ML algorithm would result in the best performance for cold-start users?

RQ4. Comparing ML and deep learning methods, what is the benefit for RSs?

In this section, we introduce our PUPP-DA framework, which employs both ML and deep learning methods to address the cold-start and data sparsity problems.

First, we use soft clustering, the EM method, to create our user segmentation. In the previous chapter we evaluated different clustering techniques. Earlier we concluded that EM is most suitable for the RSs and datasets we used; therefore, a decision was made to use this algorithm in this framework. In addition, we use a k-NN (EM-k-NN framework) and subspace method (EM-subspace framework) with k-NN as a base classifier for the clustered dataset. The results from ML and deep learning are compared with the traditional CF (using k-NN) framework, which constitutes our baseline.

There are different variations of active learning in which the learner will ask the human expert, or so-called *oracle*, to label some unlabeled instances [230, 231]. In this framework, we use the *pool-based sampling*, which is categorized for the way it accesses the unlabeled instances [230]. This approach has been widely used for real-world applications in ML [230, 231]. In pool-based sampling, the learning method is trained on a set of unlabeled instances. At each step, the learner selects the most informative instances and requests their label from the system expert [230, 231]. Following [231], informative instances are defined as those with the ability to reduce the uncertainty of the underlying statistical model [232]. In the pool-based method, the active learning algorithm evaluates the entire set, or so-called pool, of unlabeled instances, then selects the best to query using a predefined strategy [231]. Considering a classification setting, the main goal is to improve the accuracy using as few unlabeled instances as possible, which reduces the cost of obtaining them [231]. As mentioned above, active learning is an effective way to collect more information about the user using either explicit or implicit information extraction. In RSs, several personalized active learning strategies have been introduced in the literature, e.g., influence based, prediction based, and user partitioning [233]. In this framework, we utilize an explicit approach in which we expand the users' profile by adding additional ratings of items which improve the insight into users' preferences. To evaluate the presented unlabeled instances to the human expert, we selected instances by focusing on the items with the highest prediction rates [233, 234]. In this framework, if a new user rates a small number of highly relevant items, that may be sufficient for first analyzing the item's features and then calculating the similarity to other items in the system.

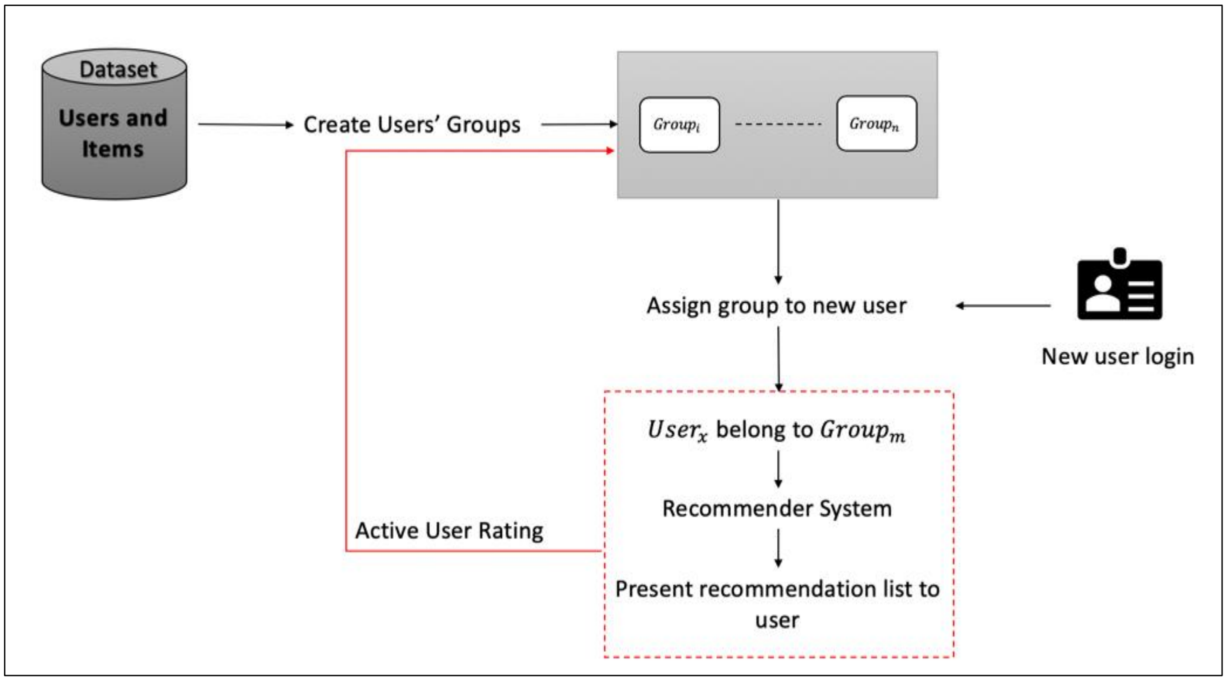


Figure 15. Workflow of our methodology.

Figure 15 shows the steps involved in the ML component of our PUPP-DA framework. Initially, we employ cluster analysis to assign customers to groups using the soft clustering approach [73]. This approach results in overlapping clusters, whereby a user may belong to more than one cluster, and it accurately reflects the human behavioral complexity. Once the groups are created, we apply two splitting methods to generate the training and test sets. We use a random split method—a common practice in ML. In addition, we design an approach that focuses on so-called popular users, as is detailed in section 4.4.4. The cold-start problem is addressed as follows. When a new user logs in to the system, the initial model is employed to find user groups with similar preferences. As stated before, we employ the k-NN algorithm to assign a new user to a given group [80, 89]. We use a ML algorithm to evaluate and potentially improve the group assignment. To this end, a human expert evaluates the predictive outcome and selects two records (for each user) with the highest prediction rate. These are appended to the training set [220, 233]. A new model is trained against the new, enlarged dataset. This process is repeated until a stopping criterion is met.

4.3.1 Framework Components

4.3.1.1 Cluster Analysis Component.

As discussed earlier in subsection 2.5.1, cluster analysis is an unsupervised learning technique used to group data when class labels are unknown [220]. This type of analysis allows the data distribution to be determined while discovering patterns and natural groups [32]. In an e-commerce setting, the goal is to maximize the similarity of individuals within the group while minimizing the similarity of characteristics between groups [147]. Therefore, similarities in opinions, likes, and ratings of the users are evaluated for each group [68].

Numerous options for algorithms are available for cluster analysis. With soft clustering, the groups may overlap; consequently, a data point may belong to more than one group. Intuitively, users' group memberships are often fuzzy in RSs. We therefore use EM clustering in our PUPP-DA framework. As discussed earlier, the EM algorithm proceeds by re-estimating the assigned probabilities, adjusting the mean and variance values to improve the assignment points, and iterating until convergence [148]; further details are discussed in subsection 2.5.1.3.

4.3.1.2 Classification Component.

We use traditional CF as our baseline framework in which k-NN is utilized, as discussed in subsection 2.5.2.2. In addition, we also employ probabilistic, NB classifier, and eager learning methods, DT and HT classifiers, in this framework.

Ensemble learning aims to improve a single algorithm performance and to provide accurate predictions [2, 174, 175, 177]. We utilize the Random Subspace ensemble-based method in this framework, as it resulted in better performance through our extensive experimentation in the HCC-Learn framework. As discussed in subsection 2.5.3.3, the learning process in the Random Subspace method builds its model by focusing on different feature subsets rather than considering instance subsets. This approach, therefore, evaluates all features in the subspace and selects the most informative ones based on the selected features. That is, feature subsets are created randomly with replacements from the training set. Subsequently, each individual classifier learns from the created subsets while considering all training examples [185].

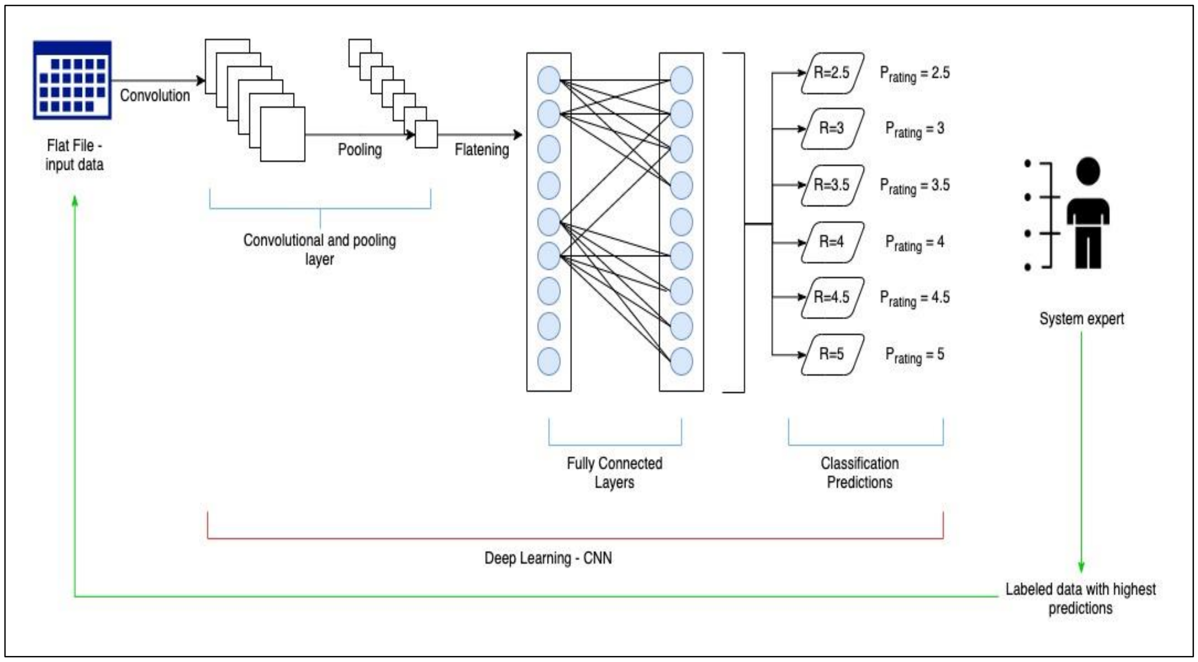


Figure 16. Deep active learning workflow.

As discussed earlier in subsection 2.5.4.1, several works reported on the strength of CNNs in terms of extracting informative features and in terms of finding hidden relationships in data. Therefore, CNNs are employed in our experimentation. In our architectures, the maximum pooling function, a down-sampling strategy, is used. The strength of this method is that it reduces the dimensionality, while allowing the learning process to focus on the features contained in the sub-region [235, 236]. Figure 16 illustrates the PUPP-DA architecture, which, as stated above, uses pool-based sampling during the active learning phrase

4.4 Experimental Setup

To evaluate the framework, we conduct our experimental evaluation using the same environment as in the HCC-Learn framework, see subsection 3.4.

4.4.1 Dataset Description

We used two datasets to evaluate our PUPP-DA framework. We tested our framework on the Serendipity dataset [237], which contains 2,150 movie ratings as well as descriptions of the movies and users' responses to questionnaires about the movies they have rated. The second dataset is the famous MovieLens dataset [238]. This dataset is well-known in RSs research and contains 100,836 ratings on 9,742 movies.

4.4.2 Dataset Pre-Processing

Initially, the movie genres were determined with the help of the statista.com and imdb.com websites, as shown in Table 23. Additional preprocessing steps involved removing all ratings lower than 2.5 out of 5 to focus the recommendations on popular movies. In addition, for the Serendipity dataset, attributes S_1 to q provide information about survey answers. These answers relate to users' experience using the RSs and the movie suggestions presented to them. If fewer than 5 questions were answered, the record was removed for lack of information. We eliminated a total of 18 records.

Table 23. Genre feature coding.

Genre	Code	Genre	Code	Genre	Code
Adventure	1	Thriller (crime)	5	Documentary	9
Action	2	Horror	6	Sci-Fi	10
Drama	3	Romantic Comedy – Romance	7	Musical	11
Comedy	4	Children	8	Animation	12
				Others	13

For the deep learning experiments, we increased the number of features from 6 to 24 features by expanding the genre attribute. We applied one-hot encoding on the genre feature, and this resulted in 19 extra features. We call this dataset MovieLensExpand in our subsequent discussions.

List of Architectures and Parameters

Table 24 shows the architectures of the different deep learning algorithms used for our experimental evaluation. For convenience, our experiments are referred to as architectures 1 to 12. For architectures 1 to 3 and 7 to 9, the neural network has three convolutional layers with 100, 50, and 25 filters, respectively, without a pooling layer. The kernel size is $4x4$. However, for architectures 4 to 6 and 10 to 12, a pooling layer of size $2x2$ with a maximum aggregation function was inserted after the first hidden layer. The sizes of both the kernels and the pooling layer were determined by inspection to maximize the accuracy of the system; the maximum pooling function was used. As expected, adding a pooling layer after each hidden layer resulted in longer training time and lower accuracy, as demonstrated by our experimental results.

Table 24. List of architectures (denoted as Arch.).

	Without clustering		With EM clustering
Arch. 1	1 Layer – 100 Filter	Arch. 7	1 Layer – 100 Filter
Arch. 2	2 Layers – 100, 50 filters	Arch. 8	2 Layers – 100, 50 filters
Arch. 3	3 Layers – 100, 50, 25 filters	Arch. 9	3 Layers – 100, 50, 25 filters
Arch. 4	1 Layer with $4x4$ patch size and $2x2$ pool size	Arch. 10	1 Layer with $4x4$ patch size and $2x2$ pool size
Arch. 5	2 Layers – 100, 50 filter each with $4x4$ patch $2x2$ pool	Arch. 11	2 Layers – 100, 50 filter each with $4x4$ patch $2x2$ pool
Arch. 6	3 Layers – 100, 50, 25 filters each with $4x4$ patch and $2x2$ pool	Arch. 12	3 Layers – 100, 50, 25 filters each with $4x4$ patch - $2x2$ pool

4.4.3 Experimental Setup

In our experimental evaluation, as introduced earlier, we employ the EM cluster analysis algorithm to segment users into potentially overlapping clusters. We initially utilize two baseline classifiers: k-NN and the Random Subspace ensemble method with k-NN as the base learner. The value of *k* is set to 5, while the number of features to be included in a subspace is fixed at 0.50 (50%); both values are set by inspection. In active learning, we proceed in a number of iterations, where in each iteration we select for each user the two records with the highest prediction rate. After labelling, these two records are appended to the original training set and removed from the test set. The number of iterations in the present work is limited to five to process the requests fast. Our model is evaluated using the 10-fold cross-validation approach.

4.4.4 Cold-Start Simulation

This section explains the approach for simulating the cold-start problem. We employ two (2) techniques to split our datasets, random split and popularity split. Initially, each technique is evaluated against the traditional k-NN, EM-k-NN, and EM-Subspace methods.

In the random split method, the dataset is divided randomly between training (70%) and testing (30%) sets, where the training dataset contains the known rating by the system, as already provided by the users. The test set, however, includes unknown ratings. Note that this approach is commonly adopted in the literature [220].

Popularity split evaluates the popularity associated with the users and the items. In this scenario, we consider the users with the highest number of ratings and refer to them as "popular users," i.e., those who use the system frequently. These users are removed from the training set and used as test subjects for cold-start simulations. A removed user must have rated at least five popular movies to be considered for removal; the choice of five movies was determined by inspection. By removing members in this manner, we increase the chance for the system to find similarities among more users' segmentations in the system. This is, as far as we are aware, the first research to use the notion of popular, or frequent, users to guide the determination of the recommendations made to cold-starts. We do so based on the assumption of trends (such as in clothing RSs) and top rating systems for movies or music (such as in Netflix and iTunes).

For a user to be considered as a test subject in the popularity split, the following criteria must be met:

- The user must have a high number of ratings, as opposed to a random split, where the number of items rated by the user is ignored, as shown in Table 25.
- The rated movies must have a rating greater than 2.5 (out of 5).
- The user must have rated popular movies. The unpopular movies create a grey-sheep problem, which refers to users who are atypical. We do not address grey-sheep in this framework.

We illustrate our results with 10 users randomly selected from the test sets of both splits. Table 25 shows information about the selected users in the MovieLens dataset. It is important to stress that we must ensure that each selected user does not have remaining records in the training set. This verification ensures a properly simulated cold-start problem.

4.4.5 Evaluation Criteria

As mentioned, k-NN is widely employed in CF systems. Consequently, we use it as our baseline as well as the base learner in our feature subspace ensemble. We also employ the mean absolute error (MAE) measure, which indicates the deviation between predicted and actual ratings, as a predictive measure [239], and employ the model accuracy and the F-measure (geometric mean of recall and precision) to determine the usefulness of the recommendation list [239].

Table 25. Test subject from MovieLens dataset.

Popular users		Random split	
User ID	#Rating	User ID	#Rating
599	1096	1	226
474	1280	225	67
414	1491	282	190
182	805	304	194
477	772	34	56
603	773	374	32
448	698	412	90
288	724	450	48
274	780	510	74
68	677	602	118

4.5 Results and Discussions

In this section we discuss the performance of the model in terms of accuracy, MAE, and F-measure. Individual users are considered in our evaluation.

4.5.1 System Evaluation

First, we discuss our findings for RQ1. Table 26 and Table 27 show the classification accuracy of the ML methods in system for random and popularity splits. In both cases, active learning improves the performance—by 39.66% for the Serendipity dataset and 59.95% for the MovieLens dataset. When considering the random split results, we note increases of 20.56% for the Serendipity dataset and 42.8% for the MovieLens dataset. These results are obtained using the EM clustering technique.

We also enhance the performance of the CF framework by introducing the Subspace method. As described earlier, instead of using the k-NN algorithm as a single classifier, we apply an ensemble subspace method using k-NN as a base learner and a subspace of 50% features. Again, we note an improvement over the traditional CF system. Specifically, the random split method improves results by 23.91% for the Serendipity dataset and 47.47% for the MovieLens dataset, compared to the traditional framework. In addition, using the popularity split method, the accuracy increases by 40.96% and 60.31%, respectively. The results of Table 26 and Table 27 indicate that the popularity split method always results in much higher accuracy.

Table 30 and Table 31 depict the results for the F-measure, which again confirms the benefit of focusing on popular users while training. The same observation holds when the MAE metric is employed.

Table 32 contains a summary of the improvement in percentage over the CF framework for both datasets. Note that these improvements are calculated only for the first iteration, as we are interested in the immediate, cold-start problem. The outcome of the last four iterations confirms that the system can make appropriate recommendations to new users while performing adequately for existing users.

Table 26. Model accuracy for MovieLens dataset.

		Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Popularity Split	k-NN	38.50	38.43	38.28	38.37	38.44
	EM-k-NN	98.45	98.47	98.44	98.50	98.47
	EM-Subspace	98.81	99.18	98.83	98.86	98.68
Random Split	k-NN	39.08	38.94	38.91	39.11	39.22
	EM-k-NN	81.88	81.76	81.81	81.87	81.83
	EM-Subspace	86.55	88.51	87.23	87.14	86.38

Table 27. Model accuracy for Serendipity dataset.

		Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Popularity Split	k-NN	42.18	42.35	43.29	43.07	44.83
	EM-k-NN	81.84	81.63	81.87	82.58	82.58
	EM-Subspace	83.14	83.18	84.04	84.17	81.60
Random Split	k-NN	43.87	44.18	45.08	45.24	46.51
	EM-k-NN	64.43	65.07	65.23	65.33	66.47
	EM-Subspace	67.78	68.40	69.27	69.21	69.77

Table 28. MAE results for the popularity split test method.

	k-NN		EM-k-NN		EM-Subspace	
	Serendipity	MovieLens	Serendipity	MovieLens	Serendipity	MovieLens
Iteration 1	0.214	0.237	0.120	0.039	0.167	0.106
Iteration 2	0.213	0.237	0.119	0.039	0.170	0.108
Iteration 3	0.211	0.237	0.119	0.039	0.167	0.110
Iteration 4	0.211	0.237	0.118	0.039	0.167	0.110
Iteration 5	0.210	0.237	0.118	0.039	0.167	0.095

Table 29. MAE results for the random split test method.

	k-NN		EM-k-NN		EM-Subspace	
	Serendipity	MovieLens	Serendipity	MovieLens	Serendipity	MovieLens
Iteration 1	0.210	0.237	0.175	0.116	0.204	0.164
Iteration 2	0.210	0.237	0.173	0.116	0.201	0.161
Iteration 3	0.209	0.237	0.171	0.116	0.199	0.168
Iteration 4	0.206	0.237	0.170	0.116	0.201	0.166
Iteration 5	0.205	0.236	0.168	0.116	0.198	0.163

Table 30. F-measure results for the popularity split method.

	k-NN		EM-k-NN		EM-Subspace	
	Serendipity	MovieLens	Serendipity	MovieLens	Serendipity	MovieLens
Iteration 1	0.594	0.352	0.818	0.984	0.830	0.988
Iteration 2	0.595	0.351	0.816	0.985	0.831	0.992
Iteration 3	0.604	0.350	0.819	0.984	0.840	0.988
Iteration 4	0.602	0.351	0.826	0.985	0.841	0.989
Iteration 5	0.619	0.352	0.826	0.985	0.815	0.987

Table 31. F-measure for the random split test method.

	k-NN		EM-k-NN		EM-Subspace	
	Serendipity	MovieLens	Serendipity	MovieLens	Serendipity	MovieLens
Iteration 1	0.610	0.359	0.629	0.817	0.656	0.864
Iteration 2	0.613	0.358	0.636	0.816	0.660	0.884
Iteration 3	0.622	0.357	0.636	0.816	0.671	0.872
Iteration 4	0.623	0.360	0.637	0.817	0.668	0.871
Iteration 5	0.635	0.361	0.650	0.817	0.673	0.863

Table 32. Improvement in predictive accuracy measures for system-wide performance over traditional CF.

Framework	Accuracy Increase by %	F-measure Increase by %	MAE Decrease by %	Dataset
Popularity test method				
EM-CF	39.99	0.224	0.094	Serendipity
	59.95	0.632	0.198	MovieLens
EM-Subspace-CF	40.96	0.236	0.047	Serendipity
	60.31	0.636	0.131	MovieLens
Random split test method				
EM-CF	20.87	0.019	0.035	Serendipity
	42.80	0.581	0.243	MovieLens
EM-Subspace-CF	23.91	0.046	0.006	Serendipity
	47.47	0.628	0.195	MovieLens

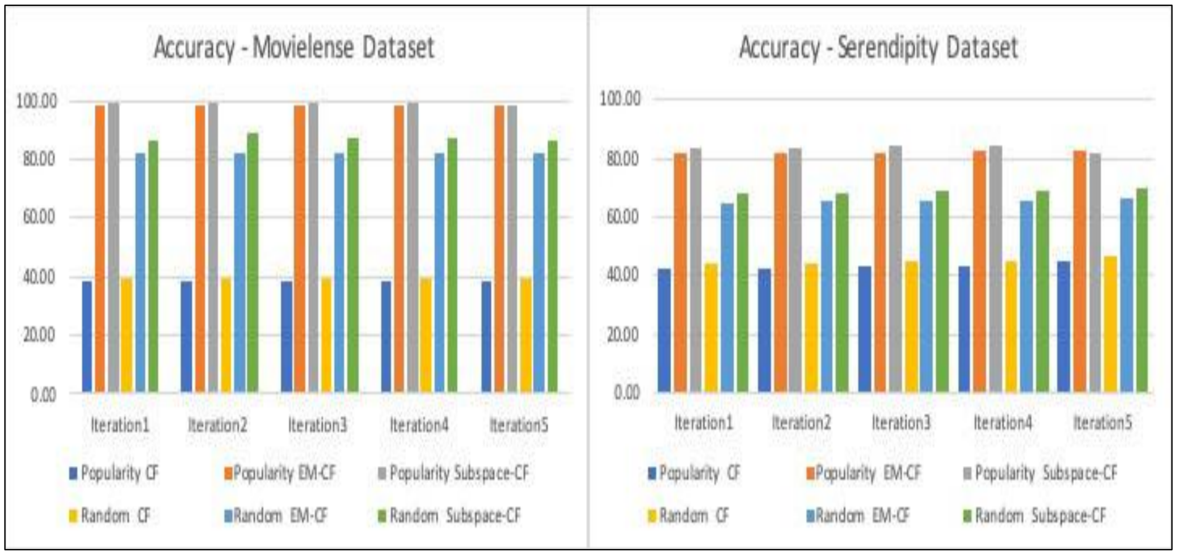


Figure 17. PUPP-DA framework accuracies on MovieLens and Serendipity datasets.

Table 33. Results for random split in all architectures against the MovieLensExpand dataset

	Without Clustering			With EM Clustering		
	Accuracy	MAE	F-measure	Accuracy	MAE	F-measure
	Arch.1: 1 Layer – 100 Filter			Arch.7: 1 Layer – 100 Filter		
Iteration 1	31.808	0.2561	?	98.610	0.0291	0.9860
Iteration 2	31.793	0.2560	?	98.606	0.0292	0.9860
Iteration 3	31.771	0.2562	?	98.626	0.0293	0.9860
Iteration 4	31.720	0.2561	?	98.643	0.0293	0.9860
Iteration 5	31.740	0.2562	?	98.633	0.0290	0.9860
	Arch.2: 2 Layers – 100, 50 filters			Arch.8: 2 Layers – 100, 50 filters		
Iteration 1	31.7011	0.2564	?	98.4412	0.0316	0.9840
Iteration 2	31.4779	0.2564	?	98.3186	0.0317	0.9830
Iteration 3	31.6527	0.2563	?	98.4605	0.0314	0.9840
Iteration 4	31.6280	0.2565	?	98.3600	0.0323	0.9830
Iteration 5	31.5754	0.2564	?	98.5123	0.0312	0.9850
	Arch.3: 3 Layers – 100, 50, 25 filters			Arch.9: 3 Layers – 100, 50, 25 filters		
Iteration 1	31.3238	0.2574	?	98.2671	0.0350	0.9830
Iteration 2	31.2446	0.2574	?	98.2919	0.0350	0.9830
Iteration 3	31.3476	0.2575	?	98.2517	0.0365	0.9820
Iteration 4	31.0654	0.2574	?	98.3252	0.0346	0.9830
Iteration 5	30.9956	0.2574	?	98.1551	0.0368	0.9810
	Arch.4: 1 Layer with 4x4 patch size and 2x2 pool size			Arch.10: 1 Layer with 4x4 patch size and 2x2 pool size		
Iteration 1	31.9727	0.2531	0.2260	93.5398	0.0415	0.9340
Iteration 2	31.9095	0.2533	0.2440	93.5285	0.0416	0.9340
Iteration 3	32.2699	0.2528	0.2360	93.5660	0.0408	0.9340
Iteration 4	32.1882	0.2528	0.2320	93.5153	0.0412	0.9340
Iteration 5	31.9685	0.2525	0.2100	93.5434	0.0411	0.9340
	Arch.5: 2 Layers – 100, 50 filter each with 4x4 patch 2x2 pool			Arch.11: 2 Layers – 100, 50 filter each with 4x4 patch 2x2 pool		
Iteration 1	31.0209	0.2558	?	92.5114	0.0463	0.9230
Iteration 2	31.2586	0.2549	?	92.6466	0.0458	0.9240
Iteration 3	31.0390	0.2548	?	92.5114	0.0457	0.9230
Iteration 4	31.0747	0.2559	?	92.6871	0.0464	0.9250
Iteration 5	31.1985	0.2551	?	92.4755	0.0467	0.9230
	Arch.6: 3 Layers – 100, 50, 25 filters each with 4x4 patch and 2x2 pool			Arch.12: 3 Layers – 100, 50, 25 filters each with 4x4 patch - 2x2 pool		
Iteration 1	31.3621	0.2553	?	19.6546	0.2824	?
Iteration 2	31.3665	0.2549	?	18.6625	0.3134	?
Iteration 3	31.3488	0.2551	?	31.0776	0.2802	0.296
Iteration 4	31.1026	0.2552	?	17.3562	0.3048	?
Iteration 5	31.2530	0.2552	?	27.0162	0.3045	?

Table 34. Results for popularity split in all architectures against the MovieLensExpand dataset.

	Without Clustering			With EM Clustering		
	Accuracy	MAE	F-measure	Accuracy	MAE	F-measure
	Arch.1: 1 Layer – 100 Filter			Arch.7: 1 Layer – 100 Filter		
Iteration 1	32.3328	0.2553	?	98.6855	0.0287	0.9870
Iteration 2	32.3290	0.2554	?	98.6819	0.0283	0.9870
Iteration 3	32.3364	0.2553	?	98.5727	0.0287	0.9860
Iteration 4	32.3878	0.2554	?	98.5865	0.0289	0.9860
Iteration 5	32.3512	0.2553	?	98.5375	0.0286	0.9850
	Arch.2: 2 Layers – 100, 50 filters			Arch.8: 2 Layers – 100, 50 filters		
Iteration 1	32.1711	0.2552	?	98.4277	0.0314	0.9840
Iteration 2	32.3228	0.2554	?	98.4962	0.0313	0.9850
Iteration 3	32.1299	0.2554	?	98.3711	0.0317	0.9840
Iteration 4	32.2562	0.2552	?	98.4459	0.0318	0.9840
Iteration 5	32.1728	0.2553	?	98.3262	0.0321	0.9830
	Arch.3: 3 Layers – 100, 50, 25 filters			Arch.9: 3 Layers – 100, 50, 25 filters		
Iteration 1	31.5767	0.2561	?	98.1899	0.0354	0.9820
Iteration 2	31.6925	0.2566	?	98.2839	0.0351	0.9830
Iteration 3	31.7881	0.2562	?	98.1762	0.0368	0.9820
Iteration 4	31.8678	0.2563	?	98.1674	0.0361	0.9820
Iteration 5	31.5323	0.2563	?	98.1185	0.0398	0.9810
	Arch.4: 1 Layer with 4x4 patch size and 2x2 pool size			Arch.10: 1 Layer with 4x4 patch size and 2x2 pool size		
Iteration 1	32.5612	0.2524	?	93.6614	0.0403	0.9350
Iteration 2	32.5191	0.2522	?	93.6364	0.0408	0.9350
Iteration 3	32.2581	0.2529	?	93.6247	0.0401	0.9350
Iteration 4	32.4751	0.2528	?	93.6772	0.0410	0.9350
Iteration 5	32.3503	0.2528	?	93.6772	0.0410	0.9350
	Arch.5: 2 Layers – 100, 50 filter each with 4x4 patch 2x2 pool			Arch.11: 2 Layers – 100, 50 filter each with 4x4 patch 2x2 pool		
Iteration 1	32.0910	0.2543	?	92.7691	0.0452	0.9260
Iteration 2	31.8594	0.2548	?	92.7830	0.0448	0.9260
Iteration 3	32.0143	0.2549	?	92.6474	0.0450	0.9240
Iteration 4	31.8669	0.2543	?	92.7255	0.0454	0.9250
Iteration 5	31.8077	0.2546	?	92.7314	0.0453	0.9250
	Arch.6: 3 Layers – 100, 50, 25 filters each with 4x4 patch and 2x2 pool			Arch.12: 3 Layers – 100, 50, 25 filters each with 4x4 patch - 2x2 pool		
Iteration 1	32.0923	0.2540	?	37.9286	0.2848	0.2850
Iteration 2	32.0544	0.2544	?	33.4406	0.2839	0.2750
Iteration 3	31.9484	0.2543	?	23.1218	0.3089	?
Iteration 4	32.0119	0.2537	?	30.3928	0.2809	0.2900
Iteration 5	31.8699	0.2543	?	24.4119	0.2692	0.3350

Next, we discuss our results when utilizing CNNs, in order to answer our second RQ. We use 12 different CNN architectures, as depicted in Table 24. When analyzing Table 33 and Table 34, we find that performing user segmentation with EM clustering results in better models for both random and popularity splits. That is, both techniques result in very comparable model performance, therefore providing little additional insight into the cold-start problem. For this reason, we further evaluate these models by examining the individual users' prediction, as reported above. In terms of system performance, the highest accuracy achieved is 98.81% using the subspace method with the popularity split. In the deep learning setting, both the popularity split, and random split methods result in comparable performance, indicating that the deep learning method is successful in capturing relationships in between users.

4.5.2 User Prediction Rates

In this section, we further validate our approach by considering the user prediction rate. This validation assists us in answering our last two RQs. In this section, the prediction rates for the ten (10) users from the MovieLens dataset are presented. The results of Table 35 initially indicate that EM-k-NN has the best prediction rates when employing ML algorithms, rather than deep learning. However, after the third iteration, when a random split is employed, the prediction rate begins to decrease, at least for some users. Furthermore, by considering the overall performance of the system, we can conclude that EM-subspace presents the best performance against these datasets when compared to the other two models.

Table 35. New user prediction accuracy in percentage.

Popular user										
UserID	182	274	288	414	448	474	477	599	603	68
CF	80	100	100	100	100	86	100	100	85	80
EM-CF	100	100	100	100	100	100	100	100	100	100
EM-subspace-CF	91	90	91	92	91	92	91	91	91	90
Random split										
UserID	1	225	282	304	34	374	412	450	510	602
CF	71	52	48	61	41	56	71	50	55	76
EM-CF	100	100	100	100	100	100	100	100	100	100
EM-subspace-CF	63	61	61	62	58	62	59	63	61	63

In what follows, we compare the new users' prediction rates for the first two iterations, between the random split and popularity split, while using different deep active learning architectures. We report the total average for users' prediction rates in iterations 1 and 2. For each type of split, as noted earlier, we test 10 different users, as shown in Table 25. For more details about the individual predictions, the reader is referred to Table 52 in appendix 1.

As reported in Table 36, the prediction rates remain below 50% for almost all users when architectures 1 to 6 are employed. However, with architectures 7 to 11, the models predict the correct product for recommendation with a 100% certainty for at least two items per user. For instance, let us consider architecture 7, where each convolutional layer consists of 100 kernels. In this setting, the prediction rates vary between 76.8% and 100%. Active learning selects the two items with a 100% prediction rate (the two highest prediction rates), labels them, and appends them to the training set. It will be recalled that only the two items with the highest prediction rates are added to the training set during the active learning phase. Therefore, the results for a user are the average of the two movies with the highest prediction rates.

When the popularity split is employed with architectures 7 to 11, accuracies between 92% and 98% are obtained. As depicted in Table 36, during the active learning phase, at least two items had a prediction rate of 100%. Active learning in general improves the deep active learning results for the initial predictions. These initial predictions play a pivotal role in the cold-start setting. While analyzing these results, we further note that the prediction rates for some individual users were improving only in the third, fourth, or even fifth iteration. Consequently, their profiles were more difficult to learn, which means that more iterations were required for a satisfactory conclusion of the active learning process.

Table 36. Average prediction rates for the 10 test users – results in percentage.

	Random Split		Popularity Split	
	Iteration 1	Iteration 2	Iteration 1	Iteration 2
Architecture 1	42.30%	41.30%	39.50%	38.30%
Architecture 2	41.00%	39.80%	39.00%	37.80%
Architecture 3	39.10%	35.20%	37.10%	36.20%
Architecture 4	45.50%	45.70%	41.90%	45.80%
Architecture 5	35.50%	36.00%	37.10%	38.90%
Architecture 6	38.10%	38.70%	38.40%	36.80%
Architecture 7	100.00%	99.81%	100.00%	100.00%
Architecture 8	100.00%	99.85%	100.00%	100.00%
Architecture 9	99.99%	99.90%	100.00%	100.00%
Architecture 10	100.00%	100.00%	100.00%	100.00%
Architecture 11	100.00%	100.00%	100.00%	100.00%
Architecture 12	31.10%	30.90%	29.70%	29.30%

4.5.3 Statistical Validation

In this section, we validate our framework for using only ML, then for using deep active learning. First, we discuss the results of our statistical significance testing using the Friedman test; the confidence level being set to $a = 0.05$. That is, we wish to determine whether there is a statistical significance between the performance of the baseline CF method using k-NN, the two variants of our PUPP system (EM and EM-subspace).

4.5.3.1 PUPP-DA using machine learning

In this validation, the Friedman yields a p-value of 0.000171 for the Serendipity dataset, and a p-value of 0.000139 for the MovieLens dataset. Therefore, the null hypothesis is rejected for both datasets, which means there is a significant difference among the three frameworks. We report the results of the pairwise comparisons in Figure 18 and Figure 19.

Furthermore, to determine if there is a significant difference between each pair, we perform the Nemenyi post-hoc test. As shown in

Table 37, there is a significant difference among three pairs: *EM-k-NN* versus *k-NN*, *EM-subspace* versus *k-NN,* and *k-NN* versus *EM-k-NN*. These results confirm that the system benefits from soft clustering and active learning. There is no statistical difference between the versions that use a baseline learning (k-NN) when compared to an ensemble, which indicates that a single classifier may be employed against these datasets. These results confirm our earlier discussion in which EM-k-NN and EM-subspace, when used with the popularity split method, perform significantly better when compared with the random split method.

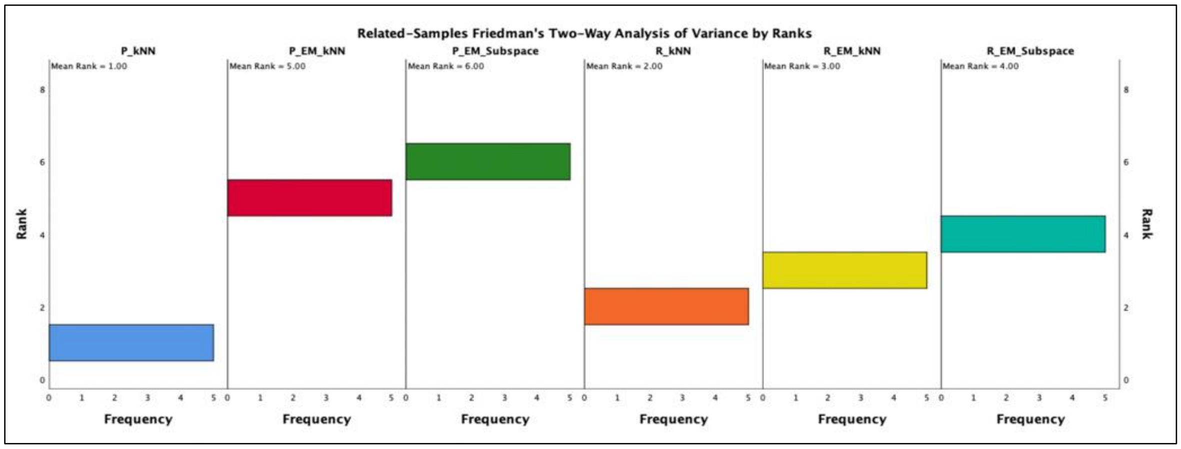


Figure 18. Friedman test mean ranks for the MovieLens dataset.

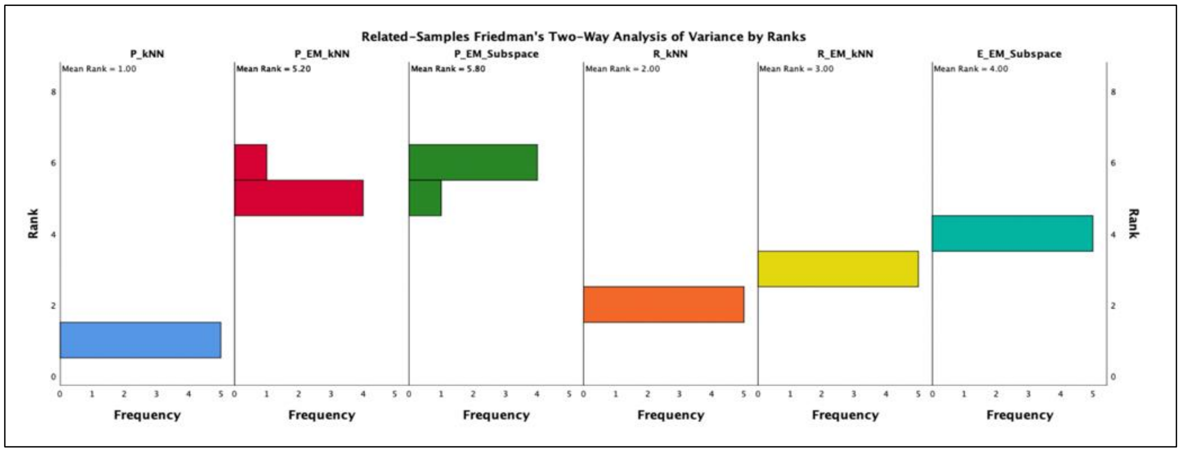


Figure 19. Friedman test mean ranks for the Serendipity dataset.

Table 37. Nemenyi p-values for the PUPP-DA framework [1].

Serendipity Dataset					
	P-k-NN	P-EM-kNN	P-EM-Subspace	R-k-NN	R-EM-kNN
P-EM-kNN	**0.005178**				
P-EM-Subspace	**0.000708**	0.995925			
R-kNN	0.958997	0.074302	**0.016639**		
R-EM-kNN	0.538193	0.427525	0.168134	0.958997	
R-EM-Subspace	0.113891	0.91341	0.65049	0.538193	0.958997
MovieLens Dataset					
	P-kNN	P-EM-kNN	P-EM-Subspace	R-kNN	R-EM-kNN
P-EM-kNN	**0.009435**				
P-EM-Subspace	**0.000343**	0.958997			
R-kNN	0.958997	0.113891	**0.009435**		
R-EM-kNN	0.538193	0.538193	0.113891	0.958997	
R-EM-Subspace	0.113891	0.958997	0.538193	0.538193	0.958997

4.5.3.2 PUPPA-DA using Deep Learning

Friedman Test for All Architectures.

In this section, we first validate all the architectures used in the PUPP-DA framework. We perform a Friedman test with a significance level of 0.05, which, when considering all architectures, results in a significance level of $2.48\text{E}-013 \cong 0.00$. Therefore, we reject the null hypothesis and determine that there is a significant difference between the 24 architectures we tested. The question, however, is which one is the best. To answer this question, we consider the Friedman Two-Way

Analysis of Variance by Rank. Figure 20 shows the mean rank for all architectures. In Table 38, which contains the mean rank of all architectures, we set our threshold at mean rank = 15, based on the mean rank results and the model accuracies. Considering Table 38, we observe that architectures 7 to 11 achieve higher ranks using both splits. This also validates our results in Table 33 and Table 34 where the models achieve an accuracy higher than 90% for these architectures. Next, we explore these 10 architectures to determine whether there is a superior configuration.

Table 38. Friedman's mean rank for all architectures.

Arch. #	Mean Rank	Arch. #	Mean Rank
R-Arch. 7	23.60	P-Arch. 1	13.00
P-Arch. 7	23.40	P-Arch. 4	13.00
R-Arch. 8	21.60	P-Arch. 2	11.40
P-Arch. 8	21.40	R-Arch. 4	10.40
R-Arch. 9	20.00	P-Arch. 6	9.80
P-Arch. 9	19.00	P-Arch. 5	8.80
P-Arch. 10	18.00	R-Arch. 1	7.20
R-Arch. 10	17.00	P-Arch. 3	6.80
P-Arch. 11	16.00	P-Arch. 12	6.40
R-Arch. 11	15.00	R-Arch. 2	6.00
		R-Arch. 6	4.00
		R-Arch. 3	3.00
		R-Arch. 5	3.00
		R-Arch. 12	1.60

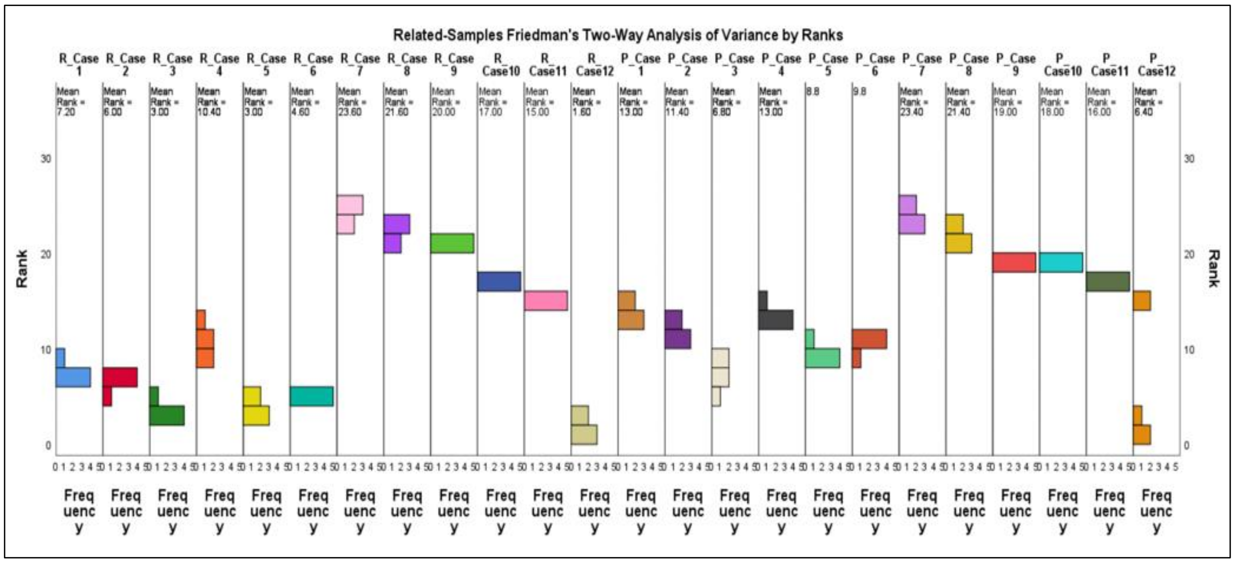


Figure 20. Test mean ranks for the MovieLensExpand dataset.

Wilcoxon Signed Ranks Test.

Earlier we noted that architectures 7 to 11 achieve a higher performance in terms of accuracy and mean rank. However, in this framework, we use two splits to evaluate our model performance: popularity and random splits. Therefore, we proceed to test each architecture for both popularity split (P) and random split (R) to determine whether there is a statistical significance in the results. From the following table, we conclude that the results obtained by architectures 9, 10, and 11 are below the significance level of 0.05. Subsequently, the null hypothesis (that all architectures are equal) is rejected for these three architectures. This implies that, for these architectures, using different splits has an impact on building the learning model. However, the type of split has little influence on the results for architectures 7 and 8.

Table 39. Wilcoxon test Statistics.

	R-Arch. 7 – P-Arch. 7	R-Arch. 8 – P-Arch. 8	R-Arch. 9 – P-Arch. 9	R-Arch. 10 – P-Arch. 10	R-Arch. 11 – P-Arch. 11
Z	-.135b	-.405b	-2.023b	-2.023c	-2.023c
Asymp. Sig. (2-tailed)	0.893	0.686	**0.043**	**0.043**	**0.043**

Friedman Test for Architectures 7 to 11.

In this section, we evaluate all results for architectures 7 to 11, considering both splits. The Friedman test results in a significance of 0.000001, which is lower than the tested level of 0.05. Therefore, the hypothesis that all architectures perform the same on the given dataset is rejected. This means that there is a significant difference between the five tested architectures (7 to 11) for both splits. As mentioned earlier, architectures 7 to 11 resulted in better performance in terms of accuracy and this was validated by considering the Friedman's mean rank.

We report the results of the pairwise comparisons in Figure 21, which illustrates the frequency count for each architecture we tested in the PUPP-DA framework. Furthermore, to determine if there is a significant difference between each pair, we perform the Nemenyi post-hoc test. As shown in Table 40, there is a significant difference among the eight pairs highlighted in bold.

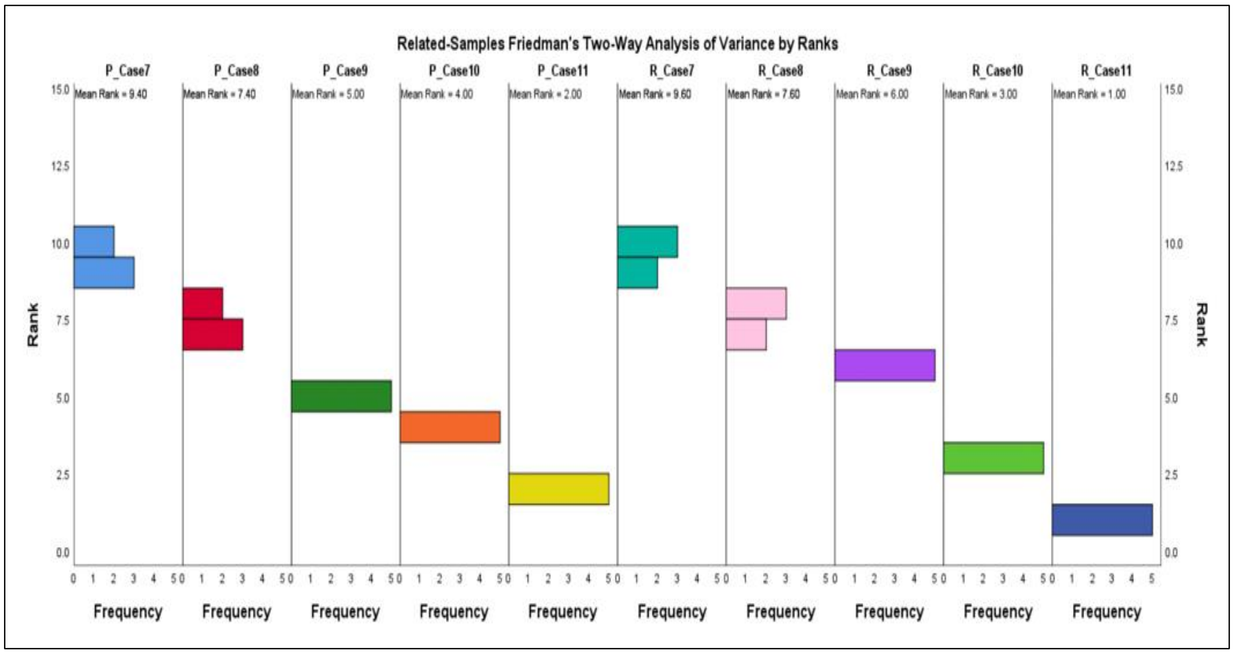


Figure 21. Friedman test mean ranks for the MovieLensExpand dataset.

Table 40. Nemenyi p-values for Arch. 7 to 11.

	P-Arch. 7	P-Arch. 8	P-Arch. 9	P-Arch. 10	P-Arch. 11	R-Arch. 7	R-Arch. 8	R-Arch. 9	R-Arch. 10
P-Arch. 8	0.989497								
P-Arch. 9	0.390244	0.96351							
P-Arch. 10	0.129563	0.750853	0.999959						
P-Arch. 11	**0.004386**	0.129563	0.864285	0.989497					
R-Arch. 7	1	0.979527	0.324212	0.098774	**0.002898**				
R-Arch. 8	0.995153	1	0.939756	0.682781	0.098774	0.989497			
R-Arch. 9	0.750853	0.999319	0.999959	0.989497	0.535342	0.682781	0.998033		
R-Arch. 10	**0.028557**	0.390244	0.989497	0.999959	0.999959	**0.020181**	0.324212	0.864285	
R-Arch. 11	**0.000487**	**0.028557**	0.535342	0.864285	0.999959	**0.000303**	**0.020181**	0.212132	0.989497

To further the summaries in Table 40, we consider the tests in bold that show that there is a significant difference among the results. To this end, in Table 41 we compare these results in terms of accuracy for the first iteration. As explained, our main purpose for the experiment is to alleviate the cold-start problem in RSs. In addition, we have mentioned that adding a pooling layer results in dimensionality reduction. As indicated in Table 33 and Table 34, adding a pooling layer in architectures 10 to 12 actually reduces the model performance. Using a CNN architecture with 1 or 2 hidden layers that contain 100 or 50 filters is a better solution for the dataset on hand, especially when considering the cold-start problem.

Table 41. Final comparison in terms of significance level and accuracy.

Arch. #		P-Arch. 7	P-Arch. 8	P-Arch. 11	R-Arch. 7	R-Arch. 8
	Accuracy	98.6855	98.4277	92.7691	98.6095	98.4412
P-Arch. 11	92.7691	P-Arch. 7				
R-Arch. 7	98.6095			R-Arch. 7		
R-Arch. 10	93.5398	P-Arch. 7			R-Arch. 7	
R-Arch. 11	92.5114	P-Arch. 7	P-Arch. 7		R-Arch. 7	R-Arch. 8

4.6 Summary

In this chapter, we presented the PUPP-DA framework designed to address the cold-start problem in RSs. Our results show the benefits of user segmentation based on soft clustering and the use of active learning to improve predictions for new users. The results also demonstrate the advantages of focusing on frequent or popular users to improve classification accuracy. In addition, our experiments illustrate the value of deep learning algorithms, and notably CNN architectures, when addressing the cold-start problem. In general, our results show that deep learning outperformed the other ML techniques we tested. Another important conclusion is that deep learning resulted in accurate predictions for each user we randomly tested compared to the regular ML in the PUPP-DA framework.

In the next chapter, we discuss the grey-sheep problem. In addition, we present the one-class classification solution to present grey-sheep users with better recommendations.

Chapter 5. GSOR – Catering for Unique Tastes

5.1 Introduction

In general, a profitable business will focus on expanding its customer database and dedicate part of its business budget for marketing strategies and campaigns. One successful marketing technique is *database (target) marketing*, defined as the process of targeting specific types or groups of customers with personalized or special offers [240], such as suggesting seasonal products based on a demographic characteristic (e.g., income, postal code). Furthermore, offer targeting treats the customers as individuals or part of a group. Deciding which groups individuals belong to based on their historical behavior and preferences can be challenging some situations.

Considering RSs in e-commerce settings, it is evident that grey-sheep directly impact the ability to conduct targeted marketing. Grey-sheep users are defined as users with unique or "exotic" tastes, and it is therefore difficult to find their common interests with other users in the system [25]. To target specific groups of customers with advertisements or special offers, companies need to identify both these customers and the items that pique their interest. Therefore, the challenge becomes how to assign a grey-sheep customer to a specific subgroup of user profiles.

According to [240, 241], an RS has many benefits in e-commerce setting, including turning browsers into buyers, increasing cross-sell, up-selling, and building loyalty. Increasing cross-sell happens by suggesting new products to the customer. For instance, some websites recommend new products based on items in the shopping bag or the browsing history. However, to do so, the system must collect sufficient information about the customers to match them with other customers in it. This can be a principal challenge of RSs, whereby it is hard to match new and grey-sheep customers [240]. The main difference between grey-sheep and cold-start customers lies in the user–item database. For grey-sheep, their information is there; however, they have unique tastes that do not match others in the system or, mostly, their opinions/likes disagree with the majority in the system. However, with cold-starts—new users—as the system has zero information, it cannot calculate a similarity score to other users in the database.

Most prior researchers consider grey-sheep users as problematic, and do not consider them in the recommendation process [28]. However, some researchers recognize the value of these customers and find solutions to generate accurate recommendations to them [30]. Nevertheless, these solutions tend to be complex and require additional information that is sometimes not available [28]. Such details may be gathered by using social media or involving human experts in the collection process, which adds more expenses and complexity to the system [28, 224, 242]. In this chapter, we introduce the GSOR framework to address grey-sheep users.

We start by exploring the existence of the grey-sheep users and assessing their impact on model classification accuracy. Next, we study the creation of recommendation lists for grey-sheep users by using the basic information available in the system. Thus, there is no need to burden customers with providing more information than what they already give us willingly. This model also addresses the sparsity problem that exists in RSs. In it, we focus on classifying which groups these unique customers belong to and the right products to recommend.

Note that our framework is based on a model-based CF approach that focuses on creating user-rating matrices where customers are matched based on their preferences. The generated recommendation list comes from the list of items both matched customers liked in the past [25]. As discussed in section 2.3.1.1.2, in model-based algorithms, statistical or ML techniques, such as clustering models and Bayesian networks, are used to make predictions. Furthermore, the relationships between item–item, user–user, and user–item are considered in the algorithm start-up requirement. Consequently, these models consider the users' ratings and other attributes of items or users [28]. With these requirements achieved, model-based methods can make more accurate and faster predictions compared to memory-based methods [243]. They also can potentially overcome the data scalability challenge, as models are already built by the time users request recommendations [28]. In this work, we build our predictive model for each user's preferences using ML techniques.

The main contributions of this work may be summarized as follows. We introduce an intelligent recommendation system targeting grey-sheep users (valuable users with atypical preferences) for tailored marketing in the e-business setting. Our method combines one-class learning algorithms, outlier detection mechanisms and unsupervised learning techniques into one framework. The state-of-the-art practice removes grey sheep users from the system. In contrast, our results confirm that our GSOR system provides highly accurate recommendations both for regular and grey-sheep users. In addition, we introduce a novel movie recommendation benchmark for use in current and future grey-sheep studies.

This chapter is organized as follows. In the next section, we discuss the related work in the grey-sheep area. We then detail our GSOR framework in section 0. The experimental setup and evaluation criteria are discussed in section 5.4. In section 0 we present our results, and we conclude the chapter in section 0.

5.2 Related Work

Recall that the focus of this chapter is the grey-sheep problem that occurs when users' preferences and opinions are unique and do not match those of others [136]. According to [25], in an e-commerce setting, there are three types of users: white-sheep, black-sheep and grey-sheep users. Some users are more engaged with the system than others and are willing to provide feedback and reviews. Users who are more engaged become loyal, and therefore, they are more popular in the system. The recommender system will have enough information about them to easily link them to other users with similar preferences. Their recommendation list can be easily generated and provided, so these types of users are categorized as white-sheep users. In contrast, black-sheep users have scant interaction with the system. These users might be active in browsing the system but have no interest in sharing their feedback or comments. Consequently, it will be hard to find a correlation with other customers in the system [25].

In addition, a few customers have unique, different opinions or more unusual tastes. These customers, in most cases, are willing to share their opinion and give their feedback. However, because they have rare tastes, it is hard to define a correlation criterion to others within the system. Those customers are categorized as grey-sheep customers. Also, because they exist in a small number within the recommender system database, grey-sheep customers create a minority class that introduces a class imbalance problem. In an imbalanced dataset, most of the users belong to one class (majority class), whereas a small number of users are members of the second class (minority class). The learned classification model will be biased toward the majority class, resulting in poor prediction accuracy for the minority class [244].

Many researchers reported that grey-sheep users tend to negatively impact systems. It is challenging to generate a recommendation list for grey-sheep users while assuring that they do not decrease the recommendation accuracy for other users [245]. Therefore, grey-sheep users are often treated as problem cases: most current research aims to identify these users and separate them from the system to limit their negative impact on system performance [28]; [241]. For instance, in [246], the authors isolated grey-sheep users in a separate cluster using an off-line clustering technique. First, the authors clustered the whole dataset; then, within each cluster, a similar threshold was used to isolate grey-sheep users. This resulted in two separate databases. The first database contained the grey-sheep users, where a support vector machine (SVM) regression was used to produce a recommendation list. The second database contained the remainder of the users, and cluster-based CF was used to recommend items. The challenge in this system is that each type of customer has to be treated differently. Consequently, as the customers enter the system, they need to be classified as grey-sheep or regular customers before recommendations can be made.

In another study, Ghorbani and Novin (2016) discussed different clustering techniques that are used in the literature to separate grey-sheep users from the rest of the system to improve the overall system accuracy. The use of clustering analysis, in general, has proven to increase recommender system accuracy, particularly when dealing with sparsity in the dataset. Many researchers, such as [51, 207], use clustering techniques to alleviate the sparsity problem in the system. Grey-sheep and sparsity are two related challenges in recommendation systems that need to be addressed simultaneously.

Another approach is to consider grey-sheep users as outliers, which are defined as observations that do not follow a specific pattern or relate to other observations in the dataset [30]. Inspired by outlier detection methods, [31] suggested a solution using a distribution-based algorithm to identify grey-sheep users. Likewise, [30] used an outlier detection technique based on the distribution of user similarities to identify grey-sheep users. In both works, the users were isolated and separate recommendation lists were generated for them.

The nature of the grey-sheep problem leads to the area of class imbalance [197], where we turn to the supervised learning setting. Intuitively, in a binary classification setting, grey-sheep will be the minority class [247]. With a class imbalance, a machine learning algorithm may produce highly accurate results while failing to learn the minority [178, 247, 248]. One-class classification has shown to be successful in learning from highly skewed data where the minority class instances are few [249]. In this setting, the minority class examples (in our case, grey-sheep items) are considered anomalies, and the one-class algorithm proceeds by learning the decision boundary to accurately classify all instances considered normal (white sheep).

[197, 250] introduced one-class classification as a solution to deal with highly imbalanced datasets. In an imbalanced (or skewed) dataset, one of the classes is usually underrepresented compared to the other classes [251]. Two-class imbalanced datasets consist of a majority class that includes the greatest number of instances and a minority class, which contains a fewer number of instances. In our work, the minority class represents grey-sheep users. One-class classification learns from the majority class to provide a statistical description of the pattern or to provide a class structure for the training sample [250]. Furthermore, [198] reported that one-class classification outperforms some of the benchmark algorithms in the field when generating a recommendation list for a large number of products.

As mentioned above, state-of-art solutions mainly address the grey-sheep challenge in two ways. Often, grey-sheep users are viewed as placing a burden on the system, since these users are perceived to have a negative impact on the model accuracy. Thus, grey-sheep users are removed from the system and placed into a separate set. In other systems, grey-sheep users are kept in the system because researchers recognize those customers for their added value. However, additional information is used based on, for instance, social media profiles and preferences, which adds complexity and expense [28, 224, 242]. Our GSOR framework provides a recommendation list for grey-sheep users while keeping them within the same system. By using this framework, we reduce the complexity of identifying these users and remove them to a separate system. Furthermore, the framework can produce recommendations for both regular and grey-sheep users, which is confirmed through an extensive evaluation.

5.3 Framework

In this section, we present our framework to investigate the following research questions (RQs). (Again, similar to our earlier work, the answers to these questions are **highly** domain-dependent, as is illustrated in our experimental evaluation.)

RQ1. What is the effect of grey-sheep users on the prediction accuracy?

RQ2. How does the system identify items of interest for grey-sheep users, and what is the optimal number of ratings for these items?

RQ3. What is the best machine learning model to produce a recommendation list for both regular and grey-sheep users in the system?

RQ4: How can a novel grey-sheep test set be developed for use as a benchmark in grey-sheep experiments?

Our GSOR framework is evaluated for its ability to make accurate recommendations for grey-sheep users using one-class classification. In this framework, we use unsupervised learning to create a group structure where users with similar preferences belong to the same group. We then apply an outlier detection method to identify grey-sheep users within the training set. The grey-sheep users are assigned to one class, and the rest of the users are assigned to the second (majority) class. We follow previous research using outlier detection to identify grey-sheep users within the training set. Finally, one-class classification is used to learn the final model and make predictions for our test subjects.

After preprocessing the dataset, we proceed by identifying potential items that would be attractive for grey-sheep users. We consider different threshold values for the number of items that we recommend to these users. To identify such items, we consider those that received high ratings but in small quantities, which indicates that the items might have promising commercial potential. In the e-commerce setting, these items could be new products that just entered the system and have not yet been rated by many users. Other examples are unique items such as international movies in a movie recommender system or an emerging trend in a clothing recommender system. The main goal of this step is to cross-reference these items with the available users to create a test set of grey-sheep users. In Section 0, we further explain this step of our framework. Note that these items are identified to study the interplay between the items and users, which will help in deciding whether a user is qualified to be a grey sheep.

Figure 23 illustrates the general workflow of our GSOR framework, which consists of four stages, namely feature engineering, creation of grey-sheep test sets, assessment and one-class learning. During feature engineering, the available characteristics of the users, such as preferences, ratings and demographic information (if available), are used to build user profiles. We subsequently identify grey-sheep users to evaluate and test our approach. In our framework, we consider grey-sheep users as those with atypical preferences, namely, users who give high ratings that are contrary to normal users or who prefer unpopular items. For instance, in a movie recommendation system, grey-sheep users give high ratings to movies that are rarely viewed but still may have won awards. In our work, we experiment with different thresholds regarding the number of items a user is required to rate, as will be discussed in Section 0.

As user profiles act as the inputs to the assessment stage, it follows that the exact features used are dataset- and domain-dependent. Next, we proceed to the assessment stage by first creating user segmentation using unsupervised learning. To this end, a cluster analysis algorithm, detailed in Section 2.5.1, is used to group individuals with similar profiles. After defining these groups, we use an outlier detection method, as explained in Section 5.3.1.1, to identify grey-sheep users within the system. The choice of outlier detection is based on earlier works, as noted in Section 5.2, where anomalous users are typically treated as outliers with atypical characteristics. Note that we create two separate training sets. In the first training set, we keep the grey-sheep users, while they are removed from the second training set. Recall that one of the research questions is to determine the impact of grey-sheep users on the system's accuracy. By maintaining two separate training sets, we are thus able to evaluate the model accuracy with and without grey sheep users. Next, we apply a number of diverse supervised learning methods, discussed 2.5.2, to construct predictive models. We repeat the same process for different item thresholds, as shown in Section 0, and test the accuracy of the resultant classification models against a separate grey-sheep test set. With the results, the impact of grey-sheep users on the prediction accuracy can be assessed to determine whether to keep those users in the system. This step will be the final one in the assessment stage before moving to the recommendation process.

During the recommendation process, we focus on constructing a machine learning model that can produce accurate recommendation lists for both regular and grey-sheep users within the same system. In this final stage of this framework, we apply a one-class classification method, as detailed in Section 5.3.1, to produce the final prediction list for our grey-sheep test subjects.

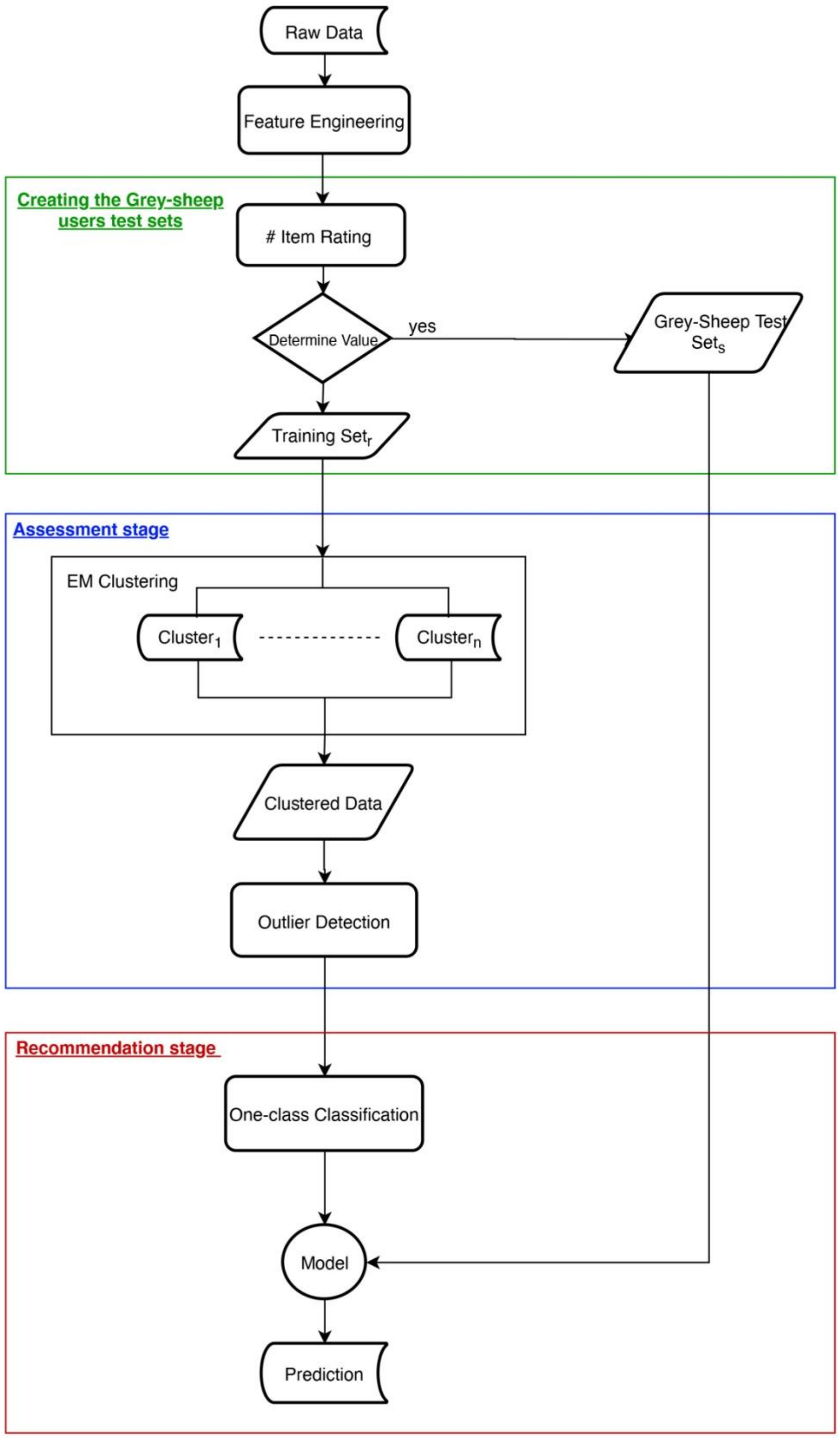


Figure 22. Grey-sheep one-class recommendation (GSOR) framework.

5.3.1 Framework Components

Our GSOR framework includes four components. For supervised learning, the classification algorithms discussed in subsection 2.5, and used in our earlier work, are again included in our framework. As an unsupervised learning component, as in PUPP-DA, we utilize the EM clustering algorithm, as it resulted in higher accuracies against the datasets used during our experimentation for the HCC-Learn framework.

Following on subsection 2.5.5, we discuss how one-class classification is utilized in this framework, and we also detail our outlier detection method.

One-Class Classification

As explained earlier, the existence of grey-sheep users creates an imbalanced dataset. Therefore, we utilize one-class learning in this framework to address this challenge. To this end, we adopt the one-class classifier in WEKA introduced by [201]. This implementation provides a generic approach that enables the adoption of standard binary or multiclass classification algorithms to solve the problem of characterizing the target class. In other words, it reduces the classes which are classified to only one class. It then proceeds by using that class to train the base learner without using information from the other classes. The learning algorithm learns from the training data to define the decision boundaries that separate the two classes [201], and then makes predictions for the grey-sheep users (minority class).

The one-class classifier combines both a density application to form a reference distribution for the target class and a class probability estimator in one method. It works by creating artificial data to reduce the bias toward the majority class. The artificial data are generated using a single Gaussian distribution [201]. One problem with generating these artificial data points is the amount of them: Too much generated data will cause the learning algorithm to always learn from the artificial class. To ensure this amount is sufficient to create the right decision boundary, a reference distribution is used, which is estimated from the available data for the target class. As the data points are not uniformly distributed, this reference ensures that the artificial data distribution is considered when the one-class model membership scores are computed.

Finally, the density function of the reference distribution is combined with the class probability estimator to train the model on the training data [201].

As stated by Hempstalk et al., the density function for the target class T is calculated as:

$$\boldsymbol{P(x|T)} = \frac{\left(1 - P(T)P(\boldsymbol{T}|\boldsymbol{x})\right)}{P(T)\left(1 - P(\boldsymbol{T}|\boldsymbol{x})\right)} \boldsymbol{P(x|A)} \qquad \textbf{5.1}$$

where T denotes the target class, A the artificial data, x a record, and $P(T)$ denotes the prior probability of observing a record from the target class [201].

5.3.1.1 Outlier Detection Component.

Outliers are defined as data points that do not belong to the general pattern in the data distribution. Detecting these outliers is a critical part of the data analysis stage. It is a common practice to identify and remove data points that are outside the majority of the data pattern. The main goal is to minimize these data points' effects on the general model performance. Detection methods have been widely used in many research areas, such as the security sector for the detection of fraud and cybercrime.

In e-commerce settings, outlier detection methods have been used to identify users who disagree with the majority in the system because of their opinions, tastes and more. In recommender systems specifically, outlier detection methods have been used to detect the presence of grey-sheep users to remove or isolate them from the system [30, 31]. In our framework, we use the interquartile range (IQR) outlier detection method. The key advantage of this method is that it is not highly sensitive to data distribution. The IQR method, therefore, can accommodate an imbalanced dataset where the class distributions are not uniform [252]. Also, some research groups have reported that IQR can outperform other methods in imbalance learning settings [253, 254]. Therefore, we decided to use IQR.

The general idea of the method is to divide the dataset into four quartiles [254]. IQR begins by subtracting the first quartile from the third one:

$$\boldsymbol{IQR = Q_3 - Q_1} \qquad \textbf{5.2}$$

Then, for outlier detection, we use the following equations [253]:

$$\boldsymbol{x \leq Q_3 + 3 * IQR \quad or \quad x \geq Q_1 - 3 * IQR} \qquad \textbf{5.3}$$

where: Q_1= 25% quartile, Q_3= 75% quartile, and IQR = the interquartile range.

5.4 Experimental Evaluation

The experimental evaluation was conducted using the same environment as for the HCC-Learn and PUPP-DA frameworks, as detailed in subsections 3.4 and 4.4.

5.4.1 Data Description

To evaluate our framework, we again used the MovieLens dataset [238]. As explained earlier, this dataset was collected during a period of seven months through the MovieLens website. It contains 610 users' ratings on 9,742 movies. In total, there are 100,836 ratings.

5.4.2 Data Pre-Processing

Initially, we merged the movies and ratings into one file. In our work, we are interested in making recommendations that benefit the business overall. Therefore, movies with a rating below three were removed from the system to aid us in focusing on the profitable items in the system. Table 42 shows the final dataset structure.

Table 42. Final dataset structure.

Attribute	Meaning	Example
MovieID	Movie unique id	37386
UserID	User unique id	438
Year	Movie release year	2005
Timestamp	Unix timestamp	964982703
Ratings	Rating given by the user	4
Genres	Movie genre	Adventure|Animation|Children|Comedy|Fantasy

The year feature was collected from the title attribute in the Movie file. This attribute originally contained a string (e.g., *Toy Story* (1995)). Therefore, we extracted only the release year of the movie, since we were not interested in the movie name. Also, for the genres attribute we performed one-hot encoding that resulted in 18 binary attributes (adventure, animation, children, comedy, fantasy, romance, drama, action, crime, thriller, horror, mystery, science fiction, musicals, documentaries, IMAX films, western, and film noir), where one (1) denotes the presence of a genre.

Table 43. List of features used in this experimental evaluation.

Attributes from the original dataset that defines the user profile	Extracted attributes from the movie genre
UserID – MovieID – Year – Timestamp – Ratings	Adventure – Animation – Children – Comedy – Fantasy – Romance – Drama – Action – Crime – Thriller – Horror – Mystery – Sci-Fi – War – Musical – Documentary – IMAX – Western – Film Noir

Following the feature preprocessing mentioned above resulted in 24 attributes, as depicted in Table 43, and 81,763 ratings in total. The final step in the preprocessing stage was the conversion of the nominal data and normalizing the numeric data.

5.4.3 Experimental Setting

In this experimental setting, we evaluated the four previously mentioned individual classifiers: k-NN, NB, and two different decision trees, C4.5 (DT) and HT. We also employed the previously introduced two ensemble learning methods, Bagging and Boosting, using the above-mentioned individual classifiers as base learners. The k value for the k-NN classifier was set to five by inspection. Following [216], the number of base learners in ensemble learning was set to 25. In the cluster analysis, we employed EM to cluster the users into five different groups, as discussed above.

After the data preparation step, the dataset was divided into training and test sets. In the experimental evaluation, test sets were created following the three different scenarios explained in the following section. To validate our models and remove misleading results, we used 10-fold cross-validation [255] to build models against the training set. The final performance is reported by evaluating the performance of the grey-sheep test sets against the model.

5.4.4 Creating the Grey-Sheep Users Benchmark

As far as we are aware, there is no benchmarking testbed for grey-sheep users in the literature. We created our own set by using the following steps:

Step 1: We identified the so-called grey-sheep items, which correspond to movies that may appeal to grey-sheep users. By examining the dataset, we found that the average number of ratings for a movie was equal to 8.39. We considered three different thresholds T, as shown in Table 44, and created three different scenarios. In scenario 1, we took all movies rated less than twice. We also established scenario 2 ($\leq$ 4 times) and scenario 3 ($\leq$ 8 times).

A human expert examined the selected movies to validate if they could be considered movies of interest to grey-sheep users. During this validation process, considerations were given to different aspects, such as movies having a high rating by some users, and also by consulting the imdb.com database to verify their quality. For instance, some movies had received awards and been given high approval on the imdb.com website. However, they were not popular in our dataset because they were movies from the 1970s and 1980s or catered to an international audience.

Table 44. Training set scenarios.

#Scenario	#times movie rated	Training set with grey-sheep users	Training set with no grey-sheep users	Percentage of grey-sheep users
S1	2	76282	72499	5%
S2	4	72400	69362	4%
S3	8	66271	63412	4%

Step 2: We identified the grey-sheep users, defined as a user who rated less than a threshold T of movies in each scenario. By inspection, we set T = 20. That is, users who rated 20 or fewer movies were identified as potential grey-sheep. (Note that 20 is the minimum number of ratings for all users in our dataset).

In this step, we cross-referenced these users with the grey-sheep items identified in step 1. Based on this, we selected only the users who actually gave a high rating to grey-sheep items. This step ensures that the selected users have a unique taste that is validated by inspection. The selected users should have rated movies that were identified in step 1 and some regular movies. We moved the records that contained only ratings for grey-sheep items to the test set. This ensures that these users have a history in the training set and are not cold-start users.

The following Table 45 depicts some examples of movies selected in step 1 to identify users with unique tastes. These users are the ones who disagreed with the majority in the system; however, a few items they liked showed promise in terms of revenue and potential success. As seen in Table 45, these movies have a few ratings in our dataset, but, considering cumulative worldwide gross, the movies had good revenue, and won or were nominated for international film awards.

Table 45. Examples of selected movies as grey-sheep items.

	MovieLens dataset		Imdb.com		Cumulative Worldwide Gross
Movie name	Avg rating	#ratings	Avg rating	#ratings	
Red Cliff (Chi bi)	4	3	7.4	42,680	$ 129,710,514
Released in 2008, in Mandarin This movie won six awards across the 3rd Asian Film Awards, and the 28th Hong Kong Film Awards, and was nominated in another 13 different categories.					
The Illusionist (L'illusionniste)	3.5	2	7.5	32,821	$ 6,007,194
Released in 2010, in French This movie won the 2010 European Film Awards. In addition, it was nominated at the 68th Golden Globe Awards for Best Animated Feature Film. In 2011, it won the first César Award for Best Animated Feature.					
Alan Partridge: Alpha Papa	4	1	6.9	29,069	$ 9,979,601
Released in 2013 in English This movie won the Best Comedy Award at the Empire Awards, UK 2014. It was nominated in seven different categories in different film competitions such as the Chicago International Film Festival 2013, New York Film Festival 2013, and Golden Trailer Awards 2014.					
Identity Thief	4	2	5.7	120,645	$ 173,965,010
Released in 2013, in English Was nominated in seven different categories across the following competitions: the 2013 Teen Choice Awards, 2014 People's Choice Awards, and the 2014 MTV Movie Awards					

Table 46 illustrates an example of a grey-sheep user: a customer who is a repeated user, as they rated 12 movies, but who disagrees with the majority in our system. They are willing to share feedback on the movies watched. However, they have a unique taste that is hard to match to others in the system. As shown, only a few other users were interested in the same movies, regardless of their quality and their awards. For instance, only three other users, out of a total of 610 people, in this dataset rated *Restrepo*, yet this movie won 10 awards and was nominated 21 times. In addition, by considering our user's profile, we can conclude that this user likes a variety of genres, with some current preference for dramas and thrillers. This type of information will help the system match the users to the closest profile when making future recommendations.

Table 46. Profile example of a typical grey-sheep user.

Movie ID	Movie Name	Year	Awards		MovieLens dataset		Imdb.com	
			Won	Nominated	Avg rating	#ratings	Avg rating	#ratings
73023	Crazy Heart	1996	12	5	3.7	5	7.2	82278
Genre: Drama – Romance								
74545	The Ghost Writer	2010	23	22	3.8	7	7.2	153234
Genre: Drama - Mystery – Thriller								
78574	Winter's Bone	2010	29	76	3.8	6	7.2	132434
Genre: Drama – Thriller								
81158	Restrepo	2010	10	21	3.2	3	7.5	21606
Genre: Documentary – War								
81788	The Next Three Days	2010	0	3	3.9	7	7.4	176901
Genre: Drama - Crime - Romance – Thriller								
82667	I Saw the Devil (Akmareul boatda)	2010	15	20	3.9	4	7.8	103699
Genre: Crime – Thriller								
90439	Margin Call	2011	8	24	3.8	7	7.1	112593
Genre: Drama – Thriller								
90600	Headhunters (Hodejegerne)	2011	8	17	3.6	5	7.5	94404
Genre: Action - Crime – Thriller								
96811	End of Watch	2012	3	10	3.9	6	7.6	221981
Genre: Crime - Drama – Thriller								
110387	The Unknown Known	2013	2	9	4	1	7	3792
Genre: Documentary								
121253	The Town that Dreaded Sundown	2014	2	1	3	1	5.6	13066
Genre: Horror – Thriller								
133115	We Could Be King	2014	2	0	4	1	7	372
Genre: Documentary								

5.4.5 Evaluation Criteria

To evaluate the performance of our frameworks, we compared the improvement in the results to a baseline model. The baseline model result was obtained by applying a traditional CF algorithm using k-NN. Recall that k-NN is commonly used in traditional CF systems [80, 89]. The k-NN algorithm uses a similarity or distance measure to calculate the distance between the query user to the k nearest neighbors. In a CF system, given query user U_m, the algorithm will determine the k nearest neighbors $\{U_1, U_2, U_3, \dots, U_k\}$ based on a distance metric such as the Euclidian, Cosine or Manhattan distance. Items in the neighbors' group not rated by U_m are subsequently selected, using an aggregation approach. Finally, the top (closest) N recommendations are presented to user U_m.

As for evaluation measures, we used model accuracy and the mean absolute error (MAE). In deciding which ones to use, we followed the common practice of using model accuracy in one-class classification work [201, 256-258]. In general, model accuracy is calculated using all observations to determine whether they are correctly classified. As stated by [220], the model accuracy is the proportion of correctly classified instances. Thus, it is calculated using the following formula [259]:

$$\boldsymbol{acc = (True\ positive/True\ Negative) * Total} \qquad \textbf{5.4}$$

Where true positive refers to the total number of correctly classified instances in the positive class, and true negative depicts the total number of correctly classified instances from the negative class [260].

To evaluate the recommender system's predictivity, we added the MAE predictive measure [239], which measures the error between the predicted and the actual values. As reported by [204], MAE is not affected by outliers since all error sizes are treated according to their proportion to the dataset and calculated using the following formula:

$$\boldsymbol{MAE = \frac{|p_1 - a_1| + \dots + |p_n - a_1|}{n}} \qquad \textbf{5.5}$$

Here, p is the predicted value on the test instances $p_1, p_2, \dots, p_n$ and a is the actual value for the tested instance $a_1, a_2, \dots, a_n$ [204].

5.5 Results and Discussions

To evaluate the usefulness of our framework, we first apply a basic traditional CF on the preprocessed dataset. The following Table 47 shows that a CF framework results in a low accuracy of 27.95%. This result is used as a baseline in our experimental evaluation and clearly shows that CF does not work well in our context.

Table 47. Baseline model result.

Accuracy	MAE
27.95%	0.3014

5.5.1 Assessment of Grey-Sheep Users' Value

In the assessment stage, we perform a total of 72 experiments. That is, 12 different classifier methods are trained on three different grey-sheep users' scenarios and two different training sets (with/without) grey-sheep users.

The main goal for this assessment stage is to answer our first RQ, which concerns the effects of grey-sheep users on the prediction accuracy for this dataset. As shown in Table 46, having grey-sheep users in the system produces no negative impact on the general performance. Overall, the performance is quite close between the two replicas. This conclusion is important, as most researchers tend to remove these users from the system or isolate them in a different subsystem where recommendations can be made. We conclude that, based on our results and for our framework, the grey-sheep users do not impact the recommendation accuracy negatively. The decision was made to proceed to one-class classification while keeping these users in the system.

Next, we answer RQ2—finding the optimal number of rated records to be considered as grey-sheep items—for the MovieLens dataset. The results indicate that scenario 1 is the best; see Table 44. We conclude this by considering the accuracy results for models built with grey-sheep users in the system. In this case, 7 of the 12 (0.58%) trained classifiers result in higher performance with scenario 1.

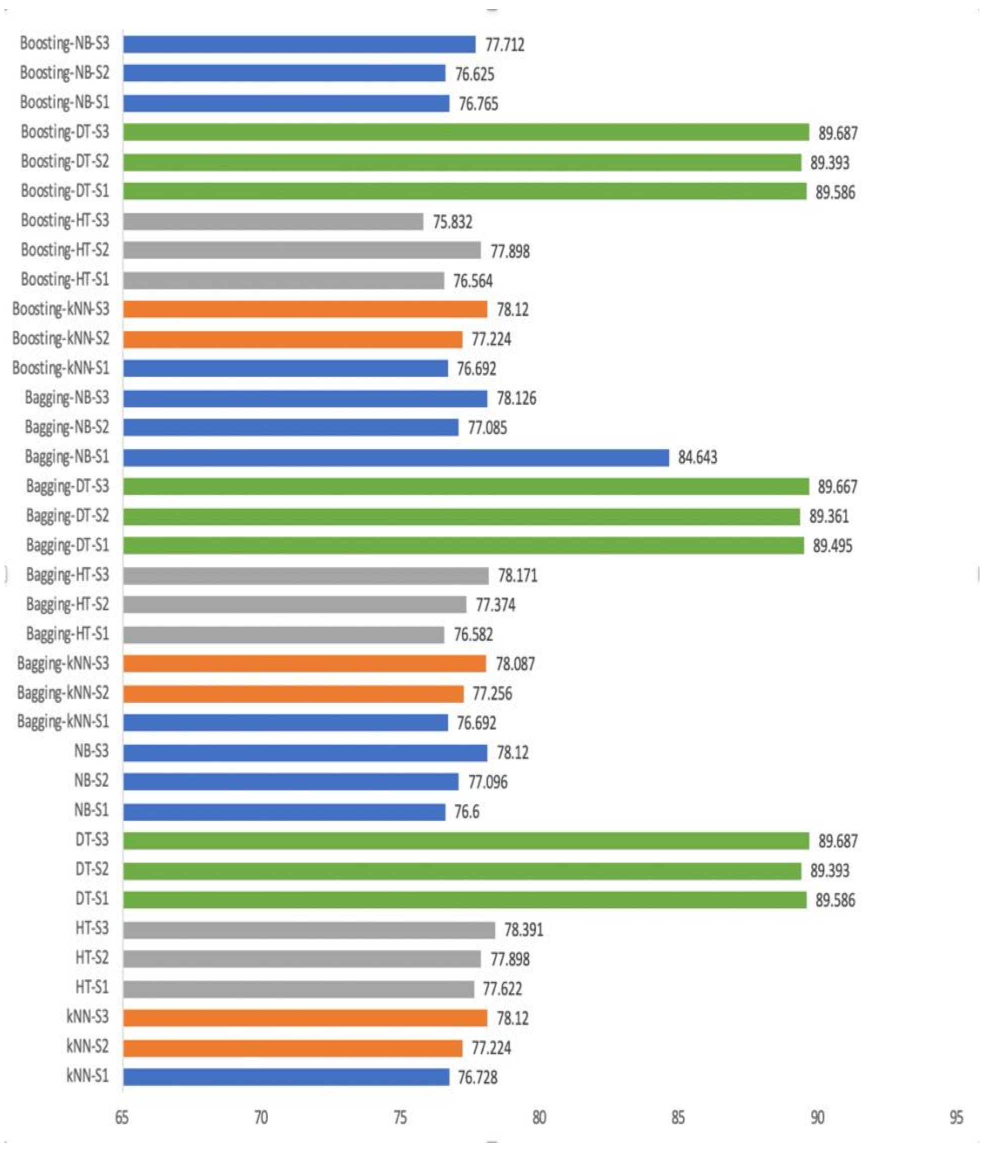


Figure 23. GSOR framework performance, in terms of accuracy.

Table 48. Assessment stage results, in terms of accuracy and MAE.

		Replica 1- with grey-sheep		Replica 2 - No grey-sheep	
		Accuracy	MAE	Accuracy	MAE
k-NN	S1	96.204	0.017	95.353	0.020
	S2	96.077	0.017	95.799	0.017
	S3	94.759	0.021	94.623	0.022
HT	S1	84.966	0.087	84.670	0.102
	S2	84.060	0.097	83.162	0.088
	S3	79.680	0.138	84.276	0.106
DT	S1	97.056	0.012	97.260	0.012
	S2	97.060	0.012	97.242	0.011
	S3	96.540	0.014	96.256	0.015
NB	S1	90.317	0.062	90.206	0.062
	S2	91.848	0.055	81.783	0.091
	S3	83.404	0.087	82.520	0.088
Bagging-k-NN	S1	95.519	0.019	95.316	0.020
	S2	96.034	0.017	95.724	0.018
	S3	94.855	0.021	94.571	0.022
Begging-HT	S1	88.261	0.095	86.114	0.100
	S2	86.134	0.100	86.134	0.100
	S3	84.883	0.105	84.437	0.108
Bagging-DT	S1	97.167	0.012	97.297	0.012
	S2	96.932	0.012	96.932	0.012
	S3	96.534	0.014	96.159	0.015
Bagging-NB	S1	90.243	0.062	90.113	0.063
	S2	81.901	0.090	81.901	0.090
	S3	83.353	0.087	82.468	0.089
Boosting-k-NN	S1	95.501	0.018	95.279	0.019
	S2	95.501	0.018	95.745	0.017
	S3	94.836	0.021	94.668	0.021
Boosting-HT	S1	96.482	0.015	96.834	0.015
	S2	97.434	0.012	97.434	0.012
	S3	96.205	0.021	95.249	0.023
Boosting-DT	S1	98.315	0.007	98.130	0.008
	S2	97.723	0.009	97.915	0.008
	S3	97.696	0.009	97.341	0.011
Boosting-NB	S1	97.667	0.010	97.871	0.010
	S2	97.402	0.012	97.402	0.012
	S3	97.037	0.012	96.256	0.017

Table 49. GSOR framework, in terms of accuracy, MAE, and time.

Classifier	scenario	Accuracy	MAE	Time to build model/ per seconds
k-NN	S1	76.728	0.097	781.47
	S2	77.224	0.095	693.22
	S3	78.120	0.100	578.26
HT	S1	77.622	0.102	42.36
	S2	77.898	0.100	39.23
	S3	78.391	0.099	37.71
DT	S1	89.586	0.104	41.09
	S2	89.393	0.106	39.86
	S3	89.687	0.103	36.52
NB	S1	76.600	0.097	7.97
	S2	77.096	0.095	7.56
	S3	78.120	0.100	6.81
Bagging-k-NN	S1	76.692	0.097	13097.97
	S2	77.256	0.095	10987.91
	S3	78.087	0.100	9685.57
Bagging-HT	S1	76.582	0.097	1011.22
	S2	77.374	0.095	1027.11
	S3	78.171	0.100	898.25
Bagging-DT	S1	89.495	0.104	654.78
	S2	89.361	0.106	650.53
	S3	89.667	0.103	595.70
Bagging-NB	S1	84.643	0.051	152.45
	S2	77.085	0.095	136.11
	S3	78.126	0.100	118.55
Boosting-k-NN	S1	76.692	0.097	15837.63
	S2	77.224	0.095	14112.51
	S3	78.120	0.100	12887.06
Boosting-HT	S1	76.564	0.102	77.27
	S2	77.898	0.100	65.22
	S3	75.832	0.099	74.39
Boosting-DT	S1	89.586	0.104	45.12
	S2	89.393	0.106	42.74
	S3	89.687	0.103	38.09
Boosting-NB	S1	76.765	0.097	115.49
	S2	76.625	0.095	121.36
	S3	77.712	0.099	105.40

5.5.2 GSOR Framework Evaluation

In Table 49, we report our results in terms of accuracy, MAE, and the time taken to build models. In terms of the overall performance, DT results in higher accuracies as a single classifier and when used as a base learner in an ensemble setting in Bagging and Boosting. In general, the DT algorithm achieves accuracies of approximately 90% accuracy for all scenarios. In terms of time required to build the model, DT as a single classifier produces higher accuracy in a shorter building time compared both to traditional CFs and to the ensemble setting. As for the predictivity measure, we note that Bagging with NB as a base learner results in the lowest MAE result. However, this result is achieved only once with scenario 1. By considering all other results, we can conclude that DT has the best performance when measured in terms of accuracy, as the results are consistently high across all scenarios. We further validate this decision in the next section by performing the Friedman's statistical validation test.

As discussed earlier, predictions for grey-sheep users are difficult, and using only traditional CF with k-NN to make predictions for these users results in low performance. As shown in Table 47, using the basic CF system for our dataset results in a low prediction accuracy of 27.95%. To improve the performance for these users, we use one-class classification learning, where the focus is on information from one class, to build the model. In general, the GSOR framework is able to improve the general performance by at least 48%.

To be precise, considering the accuracy and the time, we can conclude that DT as a single classifier is the winner against this dataset. As we discussed in section 5.3.1, one of the DT algorithm's strengths is its successful performance in prediction problems. Another is its ability to handle missing values, which makes it suitable for grey-sheep problems that cause sparsity in the dataset.

As depicted in our evaluation of the GSOR framework, more than one algorithm results in comparable results. To further evaluate these algorithms, we perform a statistical validation, presented in the next section.

5.5.3 Statistical Validation

Similar to the previous two chapters, as shown in subsection 3.5 and subsection 4.5, we validate our results using Friedman's test, a non-parametric statistical test used to measure the significance of a difference in performances of algorithms. Note that we use a confidence level of α = 0.05, following standard practice in the ML community.

For our framework, Friedman's test results in a significance level of 0.012. We set the significance level to 0.05 to reject the null hypothesis; therefore, for this framework we can observe that the null hypothesis is rejected, and there is a significant difference between the algorithms' performances.

Table 50. GSOR framework, mean rank for each algorithm based on Friedman test.

Algorithm	Mean Rank
DT	11.50
Boosting-DT	11.50
Bagging-DT	10.00
HT	8.50
Bagging-NB	6.00
Bagging-HT	5.67
kNN	5.17
Boosting-kNN	4.67
Bagging-kNN	4.50
NB	3.67
Boosting-HT	3.50
Boosting-NB	3.33

As shown in Table 50 and Figure 24, DT, either as a single classifier or in an ensemble of Bagging or Boosting, has the highest mean rank in this test. We then proceed by performing Nemenyi's post-hoc test to evaluate the performance of each of these algorithms against the remainder of algorithms tested in this framework, in order to obtain a better understanding of which algorithms exhibit significant differences in performance. Table 49 reports the results of Nemenyi's post-hoc test, where each row tests the null hypothesis, that the distributions of *classifier 1* and *classifier 2* are the same. The significance level was set to 0.05, which implies that if the significance level is below 0.05, we reject the null hypothesis and conclude that the performance of these two classifiers is different and not based on randomness. In this test, a total of 56 pairs are tested, and we report only the pairs that resulted in significant differences. As mentioned previously, Bagging with NB results in a lower MAE result. As we see from Table 50, there is no significant difference between DT as a single classifier or in ensemble setting, and Bagging with NB. Consequently, we can conclude that DT resulted in the best performance for this framework and this dataset.

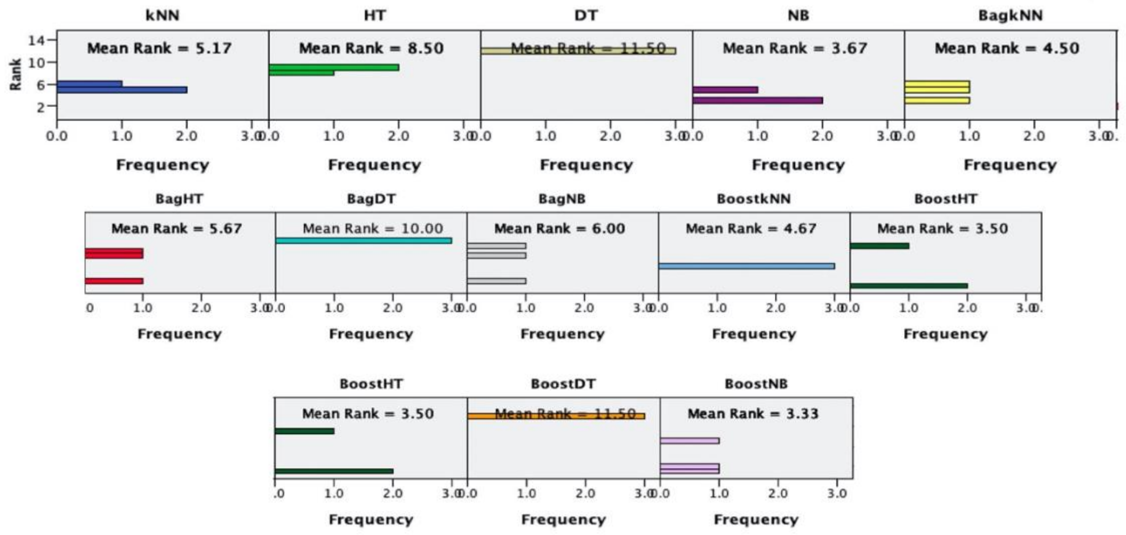


Figure 24. Friedman test mean rank for each algorithm.

Table 51. GSOR framework, Nemenyi's post-hoc results.

Algorithm 1	Algorithm 2	Significance Level
Bagging-DT	Boosting-NB	0.024
Boosting-DT	Boosting-NB	0.006
DT	Boosting-NB	0.006
Bagging-DT	NB	0.031
Boosting-DT	NB	0.008
DT	NB	0.008
Bagging-kNN	Boosting-DT	0.017
Bagging-kNN	DT	0.017
Boosting-DT	Boosting-kNN	0.020
DT	Boosting-kNN	0.020
Boosting-DT	kNN	0.031
DT	kNN	0.031
Boosting-DT	Bagging-HT	0.048
DT	Bagging-HT	0.048

5.6 Summary

In this chapter, we presented our GSOR framework, which is a hybrid, model-based CF framework that combines one-class learning algorithms, outlier detection mechanisms and unsupervised learning techniques. We showed that the use of one-class classification learning in recommendation systems for grey-sheep users could increase the overall performance and prediction accuracy, especially when the DT classifier is used in one-class classification learning. We introduced a unique grey-sheep benchmark and tested different scenarios to find the optimal number of ratings to consider an item as potentially of interest to grey-sheep users. We evaluated the effects of having grey-sheep users in the system and confirmed the positive impact of retaining such users. That is, in many earlier works, grey-sheep users have been considered a problem and removed from the system. Our extensive evaluation confirms that it is often not the case. Indeed, a major advantage of the GSOR framework lies in the ability to create accurate prediction models while considering both regular and grey-sheep users. Specifically, our experimental results confirmed the general benefits of employing one-class classification, whereby the learning process is done using information from the majority class, while predictions are made for the minority class (in our case, grey-sheep users). In the next chapter, we conclude this dissertation and present our future works.

Chapter 6. Conclusion and Future Work

6.1 Conclusions

In this chapter, we conclude the work of this dissertation. Upon beginning our study, the value of RSs was evident in both academia and industry. A number of researchers have been making contributions to ensure that businesses may positively benefit from personalized user experience, both in terms of revenue and customer throughput. As discussed earlier, RSs often lead to customer satisfaction and increase the customer loyalty, which in turn increases business profit. The Covid-19 pandemic has been a shocking, yet eye-opening experience, with a wide impact on e-commerce, technology, and RSs. This impact has, however, not just been negative. For instance, Shopify a well-known Ottawa-based e-commerce business, recently became the most valuable publicly traded company in May 2020 in Canada, even topping the stock value of the Royal Bank of Canada [261]. Shopify's financial results for the first quarter of 2020 increased by 47%, a total revenue of $470 million USD compared to the same period last year. In the merchant solutions which is the RS component of the business, there was growth of 57%, as reported by [262]. Indeed, the Shopify case study reconfirms the value, importance, and growth of RSs.

The main goal of this dissertation was to address three research challenges. We first discussed the data sparsity challenge, which exists in systems where the number of rated items by the users is much lower compared to what is required to build an accurate prediction model. Next, the cold-start challenge was presented, where new users enter the system for the first time and, therefore, the system has no historic information to find the right correlation to other profiles in the system. Finally, we discussed grey-sheep users, who have unique taste that disagree with the majority of users in the system. Three frameworks were presented and evaluated through extensive study.

Our first contribution addresses the data sparsity challenge. We evaluated the HCC-Learn framework to show the benefit of combining diverse classification algorithms with several clustering methods. Different combinations of cluster–classification pairs were evaluated. Our experimental results indicate that a combination of cluster analysis and classification algorithm clearly benefits the learning process. Furthermore, combining soft clustering with an ensemble based on feature subsets result in high performance.

The second contribution focuses on increasing the predictive accuracies for cold-start users. We present our PUPP-DA framework, which combines ML and deep learning within an active learning framework. The novelty of this framework is in constructing the user's segmentations which assign new users to the right group once they enter the system. These groups are constructed by using cluster analysis. Furthermore, we evaluate the influence of popular users' preferences to improve the recommendations for new users. Our results also indicate that focusing on frequent, or popular, users gives an advantage leading to improved classification accuracy when using ML within the active learning framework. As for deep active learning, the users' popularity did not influence the learning process, which indicates that deep learning was able to capture the relationship between users. In PUPP-DA, different CNNs architectures in the active learning setting were evaluated. Our results illustrate the value of deep active learning and show that deep learning outperformed other ML techniques, even when considering new users.

We address the grey-sheep challenge in our final contribution of this dissertation. We present our GSOR framework to make accurate recommendations for grey-sheep users. In this framework, we utilize unsupervised learning to create groups of users with similar preferences. One-class classification is subsequently used to make predictions, as based on group profiles. Our results show that using one-class classification can increase the overall performance and the prediction accuracy. To be precise, DT was able to outperform traditional CF in terms of accuracy and construction time. Furthermore, we report that having grey-sheep users in the system does not always have a negative impact on the learning process.

All presented frameworks were evaluated using a number of supervised and unsupervised learning methods introduced in Chapter 2. To summarize our work, we present general guidelines as obtained from our experimental evaluations. Figure 25 mainly serves as a reminder of our best results and how they may be combined. Note that the thresholds α and β are set by inspection, and it follows that they are highly domain-dependent.

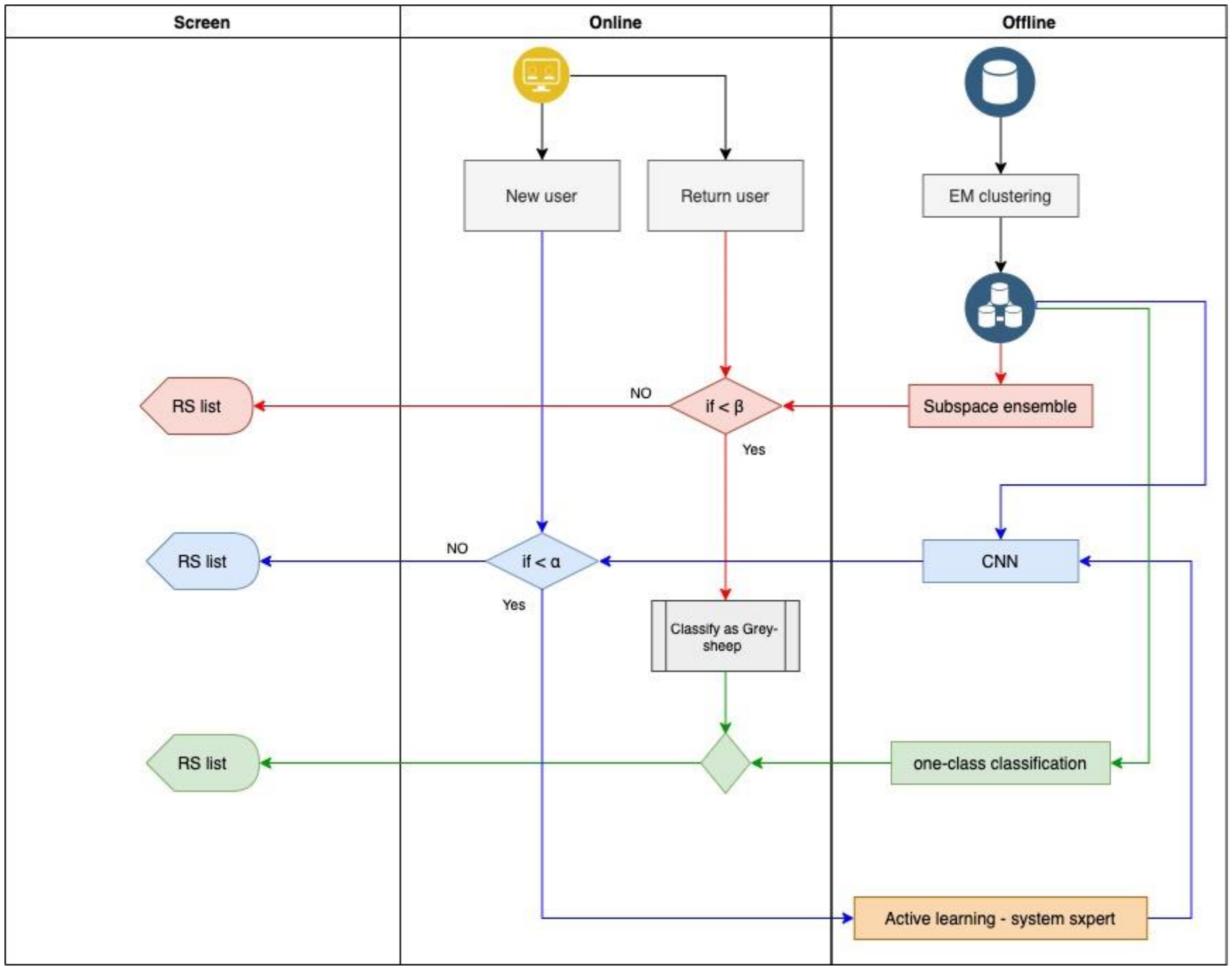


Figure 25. Diagram of suggested workflow.

Initially, a user logs into the system under two categories, namely a new or a returning user. For new users, the system presents the initial recommendation list using the predictions as created, for our benchmark's data, by the CNN algorithm. Subsequently, the accuracy of the prediction is evaluated and, if it is less than the predefined threshold α, the active learning process (which includes the domain expert) is activated. In a real-life system, this threshold might be defined by the domain expert using the learning model accuracy, or, for instance, by using the users' satisfaction feedback. In the case of returning customers, a classifier, and notably the Random Subspace ensemble method, may be used to produce a recommendation list. If the predictive accuracy is below a user-determined threshold β, then this user is categorized as a grey-sheep user and recommendations using the one-class learning algorithm are produced.

6.2 Future Work

Our future work will include the following research directions. As we discussed earlier, data sparsity can be alleviated by using information from social media [263]. However, in practice, there is a significant and increasing portion of users who do not have an interest in sharing their personal preferences on social media. Such users typically do not have a public profile, or have one with very limited information, listing only their professional backgrounds. In addition, many countries now have policies that restrict the collection of personal information without explicit consent. For these types of users, collecting more information through social media will always be a challenge. To address this, we plan to extend our HCC-Learn to include items tags and reviews, as accessible through the web search engines.

Our results in Chapter 4 indicate that deep learning often may result in more accurate recommendations for new and existing users. In this dissertation, we only utilized CNNs. In future work, we plan to further investigate other deep learning architectures. Methods that piqued our interest are the autoencoders and transformer-based deep learning networks. The key advantage of such methods is their ability to reduce the feature dimensionality and to handle sequential data. Consequently, it was reported that the network has the ability to continuously extract useful features and filter out unusable ones [193]. Therefore, we plan to investigate this further, especially if we proceed by collecting data that are tags in text reviews from search engines and social media.

As for the PUPP-DA framework, we further plan to evaluate this framework by labelling more records per iteration. We believe that deep active learning might require a larger sample size of an informative sample to improve its performance when aiming to characterize cold-start users. In addition, it will be interesting to investigate other active learning methods. As mentioned above, more data could be collected from social media or from other RSs. For instance, active transfer learning has been utilized to address data sparsity by collecting highly corelated records [109, 264]. In our case, we plan to turn our attention to social media as an added source, whereby we will use active transfer learning to fill in the gap of our users' records.

Considering grey-sheep users, we intend to investigate additional outlier detection techniques. In the GSOR framework, we show that one-class classification learning has successfully produced accurate recommendations to these users. We further plan to also consider other one-class classification learning methods, such as one-class support vector machines (SVMs) [265] and one-class autoencoders [266]. In addition, combining SVMs with cluster analysis to address class imbalance has been studied by [267]; it will be useful to also evaluate the performance for grey-sheep users. All these suggestions will give us a better idea of how the framework will perform if deployed to a real-world scenario and will demonstrate its ability to handle online learning.

Each of the presented framework aim to address a separate challenge as we discussed in this dissertation; in the future we want to create a system that address different type of users within the same system. Solutions that employ advanced machine learning approaches such as meta-learning [268] and multi-view learning [269] provide us with interesting avenues to explore in our future research.

Bibliography

1. Xu, D. and Y. Tian, *A comprehensive survey of clustering algorithms.* Annals of Data Science, 2015. **2**(2): p. 165-193.
2. Kotsiantis, S.B., *Bagging and boosting variants for handling classifications problems: a survey.* The Knowledge Engineering Review, 2014. **29**(1): p. 78-100.
3. Yuanyuan, J., Z. Haisheng, and Z. Qiaoli. *Application research on personalized recommendation in distance education*. in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*. 2010.
4. Bennett, J. and S. Lanning. *The netflix prize*. in *KDD cup and workshop*. 2007. New York, NY, USA.
5. Gomez-Uribe, C.A. and N. Hunt, *The netflix recommender system: Algorithms, business value, and innovation.* ACM Transactions on Management Information Systems (TMIS), 2016. **6**(4): p. 13.
6. Ritala, P., A. Golnam, and A. Wegmann, *Coopetition-based business models: The case of Amazon. com.* Industrial Marketing Management, 2014. **43**(2): p. 236-249.
7. *Shopify*. 2018 [cited 2018; Available from: https://www.shopify.ca/tour/ecommerce-website.
8. Bhagat, S., et al. *Recommending with an agenda: Active learning of private attributes using matrix factorization*. in *Proceedings of the 8th ACM Conference on Recommender systems*. 2014. ACM.
9. Minkov, E., et al. *Collaborative future event recommendation*. in *Proceedings of the 19th ACM international conference on Information and knowledge management*. 2010. ACM.
10. Karimi, R., et al. *Comparing Prediction Models for Active Learning in Recommender Systems*. in *Comparing Prediction Models for Active Learning in Recommender Systems*. 2015.
11. Tsai, C.-H. *A fuzzy-based personalized recommender system for local businesses*. in *Proceedings of the 27th ACM Conference on Hypertext and Social Media*. 2016. ACM.
12. Liao, C.-L. and S.-J. Lee, *A clustering based approach to improving the efficiency of collaborative filtering recommendation.* Electronic Commerce Research and Applications, 2016. **18**: p. 1-9.
13. Ntoutsi, E., et al. *Strength lies in differences: Diversifying friends for recommendations through subspace clustering*. in *In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 2014. ACM.
14. Bakshi, S., et al., *Enhancing scalability and accuracy of recommendation systems using unsupervised learning and particle swarm optimization.* Applied Soft Computing, 2014. **15**: p. 21-29.
15. Acosta, O.C., P.A. Behar, and E.B. Reategui. *Content recommendation in an inquiry-based learning environment*. in *Frontiers in Education Conference (FIE)*. 2014. IEEE.
16. Saha, T., H. Rangwala, and C. Domeniconi. *Predicting preference tags to improve item recommendation*. in *Proceedings of the 2015 SIAM International Conference on Data Mining*. 2015. SIAM.
17. Bajpai, V. and Y. Yadav, *Survey Paper on Dynamic Recommendation System for E-Commerce.* International Journal of Advanced Research in Computer Science, 2018. **9**(1).

18. Grčar, M., et al. *Data sparsity issues in the collaborative filtering framework*. in *International Workshop on Knowledge Discovery on the Web*. 2005. Springer.
19. Madadipouya, K. and S. Chelliah, *A literature review on recommender systems algorithms, techniques and evaluations.* BRAIN. Broad Research in Artificial Intelligence and Neuroscience, 2017. **8**(2): p. 109-124.
20. Su, X. and T.M. Khoshgoftaar, *A survey of collaborative filtering techniques.* Advances in artificial intelligence, 2009. p. 4.
21. Alasalmi, T., et al. *Classification uncertainty of multiple imputed data*. in *2015 IEEE Symposium Series on Computational Intelligence*. 2015. IEEE.
22. Argaman, O. *Why Customer Re-engagement Has Become More Valuable During COVID-19*. 2020; Available from: https://www.destinationcrm.com/Articles/Web-Exclusives/Viewpoints/Why-Customer-Re-engagement-Has-Become-More-Valuable-During-COVID-19-141234.aspx.
23. Team, T.N. *COVID19: eCommerce is Losing First-Time Shoppers to Customer Journey Hijacking*. 2020; Available from: https://www.namogoo.com/blog/customer-journey-hijacking/covid19-ecommerce-is-losing-first-time-shoppers-to-customer-journey-hijacking/.
24. Lu, J., et al., *Recommender system application developments: A survey.* Decision Support Systems, 2015. **74**: p. 12-32.
25. Kumar, P. and R.S. Thakur, *Recommendation system techniques and related issues: a survey.* International Journal of Information Technology, 2018. **10**(4): p. 495-501.
26. Lamche, B., U. Trottmann, and W. Wörndl. *Active Learning Strategies for Exploratory Mobile Recommender Systems*. in *Proceedings of the 4th Workshop on Context-Awareness in Retrieval and Recommendation*. 2014. ACM.
27. Wang, X., et al. *Interactive social recommendation*. in *Information and Knowledge Management* 2017. ACM.
28. Sánchez-Moreno, D., et al., *A collaborative filtering method for music recommendation using playing coefficients for artists and users.* Expert Systems with Applications, 2016. **66**: p. 234-244.
29. Srivastava, A., P.K. Bala, and B. Kumar, *New perspectives on gray sheep behavior in E-commerce recommendations.* Journal of Retailing and Consumer Services, 2020. **53**.
30. Zheng, Y., M. Agnani, and M. Singh. *Identifying grey sheep users by the distribution of user similarities in collaborative filtering*. in *Proceedings of the 6th Annual Conference on Research in Information Technology*. 2017. ACM.
31. Gras, B., A. Brun, and A. Boyer. *Identifying grey sheep users in collaborative filtering: a distribution-based technique*. in *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. 2016. ACM.
32. Pujari, A.K., K. Rajesh, and D.S. Reddy, *Clustering techniques in data mining—A survey.* IETE Journal of Research, 2001. **47**(1-2): p. 19-28.
33. Mai, J., Y. Fan, and Y. Shen, *A Neural Networks-based Clustering Collaborative Filtering Algorithm in E-commerce Recommendation System*, in *International Conference on Web Information Systems and Mining, 2009.*IEEE. p. 216-619.
34. Fernandes, N., *Economic effects of coronavirus outbreak (COVID-19) on the world economy.* Available at SSRN 3557504, 2020.
35. Ungerer, C., et al., *Recommendations to Leverage E-Commerce During the COVID-19 Crisis*. 2020, World Bank.

36. Columbus, L., *How COVID-19 Is Transforming E-Commerce*. 2020, Forbes Media LLC.
37. Shahzad, A., et al., *Covid-19 Impact on E-Commerce Usage: an Empirical Evidence from Malaysian Healthcare Industry.* 2020.
38. Chen, M.-K., K.-H. Chen, and C.-H. Chen, *The CRM-based Digital Exhibition System for Clothing Industry.* International Journal of Electronic Business Management, 2014. **12**(2): p. 122.
39. Yoon-Joo, P., *The Adaptive Clustering Method for the Long Tail Problem of Recommender Systems.* Knowledge and Data Engineering, IEEE Transactions on, 2013. **25**(8): p. 1904-1915.
40. Lucas, J.P., S. Segrera, and M.N. Moreno, *Making use of associative classifiers in order to alleviate typical drawbacks in recommender systems.* Expert Systems with Applications, 2012. **39**(1): p. 1273-1283.
41. Oestreicher-Singer, G., et al., *The Network Value of Products.* Journal of Marketing, 2013. **77**(3): p. 1-14.
42. Li, S.S. and E. Karahanna, *Online Recommendation Systems in a B2C E-Commerce Context: A Review and Future Directions.* Journal of the Association for Information Systems, 2015. **16**(2): p. 72-107.
43. Kaplan, M. *Amid Covid-19, Amazon's Q1 2020 Earnings Confirm Ecommerce Dominance*. 2020.
44. Statista. *Distribution of online shoppers in Canada as of April 2019, by household type*. 2019 [cited 2020; Available from: https://www.statista.com/statistics/1044443/canada-online-shoppers-by-household-type/.
45. Post, C., *The 2020 Canadian e-commerce report*. 2020. p. 22.
46. inc., A. *COVID-19 Driving Consumers Online: Canadian eCommerce Sales Double in 2 Weeks*. 2020.
47. Goldstein, J. *What Does Gen Z Want From Retail?* 2020.
48. Media, R.I. *How Does Aging Change Canada's Consumer Behavior?* 2020; Available from: https://www.retail-insider.com/articles/2020/2/how-does-aging-change-canadas-consumer-behavior.
49. Canada, S. *Infographic 3, Proportion of crowdsourcing participants who did not go to the grocery store or a drugstore in the week preceding the survey, by age group and by gender* 2020 [cited 2020; Available from: https://www150.statcan.gc.ca/n1/daily-quotidien/200423/g-a003-eng.htm.
50. Alabdulrahman, R., H. Viktor, and E. Paquet. *HCC-Learn Framework for Hybrid Learning in Recommender Systems*. in *Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 2020. Cham: Springer International Publishing.
51. Alabdulrahman, R., H. Viktor, and E. Paquet. *Beyond k-NN: Combining Cluster Analysis and Classification for Recommender Systems.* in *The 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018)*. 2018. Seville, Spain: KDIR 2018.
52. Alabdulrahman, R., H. Viktor, and E. Paquet. *Active Learning and User Segmentation for the Cold-start Problem in Recommendation Systems*. in *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*. 2019. Vienna, Austria: KDIR 2019.

53. Pirasteh, P., D. Hwang, and J.E. Jung, *Weighted Similarity Schemes for High Scalability in User-Based Collaborative Filtering*. Mobile Networks & Applications, 2015. **20**(4): p. 497-507.
54. Li, X., M. Wang, and T.P. Liang, *A multi-theoretical kernel-based approach to social network-based recommendation.* Decision Support Systems, 2014. **65**: p. 95-104.
55. Park, D.H., et al., *A literature review and classification of recommender systems research.* Expert Systems with Applications, 2012. **39**(11): p. 10059-10072.
56. Lü, L., et al., *Recommender systems.* Physics Reports, 2012. **519**(1): p. 1-49.
57. Huang, C.-L., et al., *Incorporating frequency, recency and profit in sequential pattern based recommender systems.* Intelligent Data Analysis, 2013. **17**(5): p. 899-916.
58. Schafer, J.B., J. Konstan, and J. Riedl. *Recommender systems in e-commerce*. in *Proceedings of the 1st ACM conference on Electronic commerce*. 1999. ACM.
59. Davoudi, A. and M. Chatterjee. *Detection of profile injection attacks in social recommender systems using outlier analysis*. in *2017 IEEE International Conference on Big Data (Big Data)*. 2017. IEEE.
60. Valliyammai, C., et al. *An intelligent personalized recommendation for travel group planning based on reviews*. in *2016 Eighth International Conference on Advanced Computing (ICoAC)*. 2017. IEEE.
61. Kim, H.M., et al., *Online serendipity: The case for curated recommender systems.* Business Horizons, 2017. **60**(5): p. 613-620.
62. Ju Jeong, H. and M. Lee, *Effects of recommendation systems on consumer inferences of website motives and attitudes towards a website.* International Journal of Advertising, 2013. **32**(4): p. 539-558.
63. Kaptein, M. and P. Parvinen, *Advancing E-Commerce Personalization: Process Framework and Case Study.* International Journal of Electronic Commerce, 2015. **19**(3): p. 7-33.
64. Kim, Y.S. and B.J. Yum, *Recommender system based on click stream data using association rule mining.* Expert Systems with Applications, 2011. **38**(10): p. 13320-13327.
65. Park, D.H., et al., *A Review and Classification of Recommender Systems Research.* School of Management, KyungHee University, Seoul, Korea, IPEDR vol, 2011. **5**.
66. Pereira, A.L.V. and E.R. Hruschka, *Simultaneous co-clustering and learning to address the cold start problem in recommender systems.* Knowledge-Based Systems, 2015. **82**: p. 11-19.
67. Chung, W.Y. and T.L. Tseng, *Discovering business intelligence from online product reviews: A rule-induction framework.* Expert Systems with Applications, 2012. **39**(15): p. 11870-11879.
68. Isinkaye, F.O., Y.O. Folajimi, and B.A. Ojokoh, *Recommendation systems: Principles, methods and evaluation.* Egyptian Informatics Journal, 2015. **16**(3): p. 261-273.
69. Golubtsov, N., D. Galper, and A. Filchenkov, *Active Adaptation of Expert-Based Suggestions in Ladieswear Recommender System LookBooksClub via Reinforcement Learning*, in *Biologically Inspired Cognitive Architectures (BICA) for Young Scientists*. 2016, Springer. p. 61-69.
70. Pasinato, M.B., C.E. Mello, and G. Zimbrão. *Active learning applied to rating elicitation for incentive purposes*. in *European Conference on Information Retrieval*. 2015. Springer.

71. So, W.T. and K. Yada. *A Framework of Recommendation System Based on In-store Behavior*. in *Proceedings of the 4th Multidisciplinary International Social Networks Conference on ZZZ*. 2017. ACM.
72. Boratto, L. and S. Carta, *The rating prediction task in a group recommender system that automatically detects groups: architectures, algorithms, and performance evaluation.* Journal of Intelligent Information Systems, 2015. **45**(2): p. 221-245.
73. Mishra, R., P. Kumar, and B. Bhasker, *A web recommendation system considering sequential information.* Decision Support Systems, 2015. **75**: p. 1-10.
74. Karimi, R., et al. *Comparing Prediction Models for Active Learning in Recommender Systems*. in *LWA*. 2015.
75. Sharma, L. and A. Gera, *A Survey of Recommendation System: Research Challenges. International Journal of Engineering Trends and Technology (IJETT)*, 2013. **4**(5): p. 1989-1992.
76. Sun, L., F. Luan, and T. Liu. *Research on applying Slopeone collaborative filtering algorithms to building products selection*. in *2011 International Conference on Electronics, Communications and Control (ICECC)*. 2011. IEEE.
77. Acosta, O.C., P.A. Behar, and E.B. Reategui. *Content recommendation in an inquiry-based learning environment*. in *2014 IEEE Frontiers in Education Conference (FIE)*. 2014.
78. Juan, B., *Collaborative Filtering Recommendation Algorithm based onSemantic Similarity of Item*, in *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*. 2012, IEEE. p. 452-454.
79. Karimi, R., et al., *Exploiting the Characteristics of Matrix Factorization for Active Learning in Recommender Systems. Proceedings of the sixth ACM conference on Recommender systems*, 2012: p. 317-320.
80. Sridevi, M., R.R. Rao, and M.V. Rao, *A survey on recommender system.* International Journal of Computer Science and Information Security, 2016. **14**(5): p. 265.
81. Karimi, R., et al., *Non-myopic Active Learning for Recommender Systems Based on Matrix Factorization*, in *2011 IEEE International Conference on Information Reuse and Integration (IRI)*. 2011. p. 299-303.
82. Karimi, R., A. Nanopoulos, and L. Schmidt-Thieme, *A supervised active learning framework for recommender systems based on decision trees.* User Modeling and User-Adapted Interaction, 2015. **25**(1): p. 39-64.
83. Kumar, P. and R.S. Thakur, *Recommendation system techniques and related issues: a survey.* 2018. **10**(4): p. 495-501.
84. Bobadilla, J., et al., *Recommender systems survey.* Knowledge-Based Systems. **46**(Complete): p. 109-132.
85. Alharthi, H. and D. Inkpen, *Content-Based Recommender System Enriched with Wordnet Synsets*, in *Computational Linguistics and Intelligent Text Processing*. 2015, Springer. p. 295-308.
86. Xiaosheng, Y. and S. Shan. *Research on Personalized Recommendation System Based on Web Mining*. in *E-Business and E-Government (ICEE), 2010 International Conference on*. 2010.
87. Ahn, H.J., H. Kang, and J. Lee, *Selecting a small number of products for effective user profiling in collaborative filtering.* Expert Systems with Applications, 2010. **37**(4): p. 3055-3062.

88. Hostler, R.E., V.Y. Yoon, and T. Guimaraes, *Recommendation agent impact on consumer online shopping: The Movie Magic case study.* Expert Systems with Applications, 2012. **39**(3): p. 2989-2999.
89. Katarya, R. and O.P. Verma, *A collaborative recommender system enhanced with particle swarm optimization technique.* Multimedia Tools and Applications, 2016. **75**(15): p. 9225-9239.
90. Terveen, L. and W. Hill, *Beyond Recommender Systems: Helping People Help Each Other.* HCI in the New Millennium, 2001. **1**: p. 487-509.
91. Geuens, S., K. Coussement, and K.W. De Bock, *A framework for configuring collaborative filtering-based recommendations derived from purchase data. European Journal of Operational Research*, 2018. **265**(1): p. 208-218.
92. Xie, L., W. Zhou, and Y. Li, *Application of Improved Recommendation System Based on Spark Platform in Big Data Analysis.* Cybernetics and Information Technologies, 2016. **16**(6): p. 245-255.
93. Stanescu, A., S. Nagar, and D. Caragea. *A Hybrid Recommender System: User Profiling from Keywords and Ratings*. in *In 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. 2013.
94. Dima, M., et al., *Expert Based Prediction of User Preferences*, in *2010 5th International Workshop on Semantic Media Adaptation and Personalization (SMAP)*. 2010, IEEE. p. 112-117.
95. Elahi, M., F. Ricci, and N. Rubens, *Active learning strategies for rating elicitation in collaborative filtering: A system-wide perspective.* ACM Transactions on Intelligent Systems and Technology (TIST), 2013. **5**(1): p. 13.
96. Karimi, R., et al., *Active Learning for Aspect Model in Recommender Systems. 2011 IEEE Symposium Computational Intelligence and Data Mining (CIDM)*, 2011: p. 162-167.
97. Koren, Y. and R. Bell, *Advances in collaborative filtering*, in *Recommender systems handbook*. 2011, Springer. p. 145-186.
98. Lu, J., et al., *A Web-Based Personalized Business Partner Recommendation System Using Fuzzy Semantic Techniques.* Computational Intelligence, 2013. **29**(1): p. 37-69.
99. Sitkrongwong, P., S. Maneeroj, and A. Takasu. *Latent Probabilistic Model for Context-Aware Recommendations*. in *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*. 2013.
100. Guoyong, C., et al. *An Improved Collaborative Method for Recommendation and Rating Prediction*. in *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*. 2014.
101. Albadvi, A. and M. Shahbazi, *Integrating rating-based collaborative filtering with customer lifetime value: New product recommendation technique.* Intelligent Data Analysis, 2010. **14**(1): p. 143-155.
102. Berkovsky, S., T. Kuflik, and F. Ricci, *The impact of data obfuscation on the accuracy of collaborative filtering.* Expert Systems with Applications, 2012. **39**(5): p. 5033-5042.
103. Chen, Z. and Z. Li. *A collaborative recommendation algorithm based on user cluster classification*. in *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*. 2016. IEEE.
104. Nagpal, et al., *FR: A Recommender for Finding Faculty Based On CF Technique*, in *Procedia Computer Science*. 2015. p. 499-507.

105. Adomavicius, G. and A. Tuzhilin. *Extending recommender systems: A multidimensional approach*. in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop on Intelligent Techniques for Web Personalization (ITWP2001), Seattle, Washington, August*. 2001. Citeseer.
106. Bilge, A. and C. Kaleli. *A multi-criteria item-based collaborative filtering framework*. in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. 2014. IEEE.
107. Umberto, P., *Developing a price-sensitive recommender system to improve accuracy and business performance of ecommerce applications.* International Journal of Electronic Commerce Studies, 2015. **6**(1): p. 1-18.
108. Elahi, M., F. Ricci, and N. Rubens, *Active Learning Strategies for Rating Elicitation in Collaborative Filtering: A System-Wide Perspective.* Acm Transactions on Intelligent Systems and Technology, 2013. **5**(1).
109. Zhao, L., et al., *Active Transfer Learning for Cross-System Recommendation.* AAAI, 2013.
110. Karimi, R., et al., *Factorized Decision Trees for Active Learning in Recommender Systems*, in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI)*. 2013, IEEE.
111. Elahi, M., F. Ricci, and N. Rubens, *Adapting to Natural Rating Acquisition with Combined Active Learning Strategies. International Symposium on Methodologies for Intelligent Systems* 2012: p. 254-163.
112. Braunhofer, M., et al., *Context Dependent Preference Acquisition with Personality-Based Active Learning in Mobile Recommender Systems*, in *International Conference on Learning and Collaboration Technologies*. 2014, Springer, Cham. p. 105-116.
113. Zhao, Z.-D. and M.-S. Shang. *User-based collaborative-filtering recommendation algorithms on hadoop*. in *2010 Third International Conference on Knowledge Discovery and Data Mining*. 2010. IEEE.
114. Lampropoulos, A.S. and G.A. Tsihrintzis, *Machine Learning Paradigms: Applications in Recommender Systems*. Vol. 92. 2015: Springer.
115. He, M., C. Ren, and H. Zhang, *Intent-based recommendation for B2C e-commerce platforms.* IBM Journal of Research and Development, 2014. **58**(5/6): p. 5:1-5:10.
116. Hu, L., et al., *CFSF: On Cloud-Based Recommendation for Large-Scale E-commerce.* Mobile Networks & Applications, 2015. **20**(3): p. 380-390.
117. Eirinaki, M., et al., *Recommender systems for large-scale social networks: A review of challenges and solutions.* future generation computer systems, 2018.
118. Verma, J.P., B. Patel, and A. Patel. *Big data analysis: recommendation system with Hadoop framework.* in *2015 IEEE International Conference on Computational Intelligence & Communication Technology*. 2015. IEEE.
119. Barney, B. *Introduction to Parallel Computing*. 2020; Available from: https://computing.llnl.gov/tutorials/parallel_comp/.
120. Gan, M.X. and R. Jiang, *Improving accuracy and diversity of personalized recommendation through power law adjustments of user similarities.* Decision Support Systems, 2013. **55**(3): p. 811-821.
121. Zou, H.T., et al., *TrustRank: a Cold-Start tolerant recommender system.* Enterprise Information Systems, 2015. **9**(2): p. 117-138.
122. Han, J., J. Pei, and M. Kamber, *Data mining: concepts and techniques*. 2011, Elsevier.

123. Lampropoulos, A.S. and G.A. Tsihrintzis, *Review of previous work related to recommender systems*, in *Machine Learning Paradigms*. 2015, Springer. p. 13-30.
124. Wu, J.-w., J.-h. Yu, and X.-h. Yu. *An item-based weighted collaborative recommended algorithm based on screening users' preferences*. in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*. 2010.
125. Gao, M., Z.F. Wu, and F. Jiang, *Userrank for item-based collaborative filtering recommendation.* Information Processing Letters, 2011. **111**(9): p. 440-446.
126. Wen, H., L.P. Fang, and L. Guan, *A hybrid approach for personalized recommendation of news on the Web.* Expert Systems with Applications, 2012. **39**(5): p. 5806-5814.
127. Rokach, L. and S. Kisilevich, *Initial Profile Generation in Recommender Systems Using Pairwise Comparison.* Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews, 2012. **42**(6): p. 1854-1859.
128. Lu, X. and Y. Du. *Interact-Friend Recommender System on Chinese Micro-Blog Based on Structure Balance*. in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. 2016. IEEE.
129. Kumar, B. and N. Sharma, *Approaches, issues and challenges in recommender systems- a systematic review.* a systematic review. I*ndian Journal of Science and Technology*, 2016. **9**(47).
130. Adomavicius, G. and A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.* Knowledge and Data Engineering, IEEE Transactions on, 2005. **17**(6): p. 734-749.
131. Desrosiers, C. and G. Karypis, *A comprehensive survey of neighborhood-based recommendation methods*, in *Recommender systems handbook*. 2011, Springer. p. 107-144.
132. Nikolaenko, V., et al. *Privacy-preserving matrix factorization*. in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013. ACM.
133. Tewari, A.S., A. Kumar, and A.G. Barman. *Book recommendation system based on combine features of content based filtering, collaborative filtering and association rule mining*. in *2014 IEEE International Advance Computing Conference (IACC)*. 2014. IEEE.
134. Kim, H.N., et al., *Collaborative user modeling for enhanced content filtering in recommender systems.* Decision Support Systems, 2011. **51**(4): p. 772-781.
135. Lu, J., et al., *A Web-Based Personalized Business Partner Recommendation System Using Fuzzy Semantic Techniques.* Computational Intelligence, 2013. **29**(1): p. 37-69.
136. Khusro, S., Z. Ali, and I. Ullah, *Recommender systems- issues, challenges, and research opportunities. 2016 Information Science and Applications (ICISA)* 2016: p. 1179-1189.
137. Claypool, M., et al., *Combing content-based and collaborative filters in an online newspaper.* 1999.
138. Torres, J. *Understanding and modeling conversations on microblogs*. in *2017 IEEE, Ecuador Technical Chapters Meeting (ETCM)*. 2017. IEEE.
139. Gonsalves, B. and V. Patil. *Improved web service recommendation via exploiting location and QoS information*. in *2016 International Conference on Information Communication and Embedded Systems (ICICES)*. 2016. IEEE.
140. Aslam, S., *Linkedin by the Numbers: Stats, Demographics & Fun Facts*, in *Omnicore*. 2018.
141. Krawiec, T. *The Amazon Recommendation Secret to Selling More Online*. Rejoiner, 2018.
142. Statista. *Net Sales Revenue of Amazon from 2004 to 2017 (in billion U.S. dollars)*. The Statistics Portal 2018 [cited 2018; Available from: https://www.statista.com/statistics/266282/annual-net-revenue-of-amazoncom/.

143. Gustafson, K., *Best Buy's Surge in Online Sales Show it won't be toppled by Amazon*, in *CNBC*. 2016.
144. Canada, E., *Canadians Prefer Shopping Online for Clothing and Accessories*, in *CISION*. 2018.
145. Das, J., et al. *Clustering-based recommender system using principles of voting theory*. in *2014 international conference on Contemporary computing and informatics (IC3I)*. 2014. IEEE.
146. Amatriain, X., et al. *Data Mining Methods for Recommender Systems*. in *Recommender systems handbook* 2011. Boston, MA: Springer.
147. Cho, Y., and S.P. Jeong. *A Recommender System in u-Commerce based on a Segmentation Method*. in *In Proceedings of the 2015 International Conference on Big Data Applications and Services*. 2015. ACM.
148. Bifet, A. and R. Kirkby, *Data Stream Mining a Practical Approach*. 2009, Citeseer: The University of Waikato. p. 68-69.
149. Frank, E., M.A. Hall, and I.H. Witten, *The WEKA Workbench.* . Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", 2016.
150. Hu, K., et al. *Air pollution exposure estimation and finding association with human activity using wearable sensor network*. in *he MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. 2014. ACM.
151. Dongmeng, G., et al. *Study on dual K-means algorithm in collaborative filtering system based on web log*. in *Proceedings of the The 11th International Knowledge Management in Organizations Conference on The changing face of Knowledge Management Impacting Society*. 2016. ACM.
152. Sowan, B. and H. Qattous, *A Data Mining of Supervised learning Approach based on K-means Clustering.* International Journal of Computer Science and Network Security (IJCSNS), 2017. **17**(1): p. 18.
153. Guo, J. *An improved incremental training approach for large scaled dataset based on support vector machine*. in *Big Data Computing Applications and Technologies (BDCAT)*. 2016. IEEE/ACM 3rd International Conference.
154. Caliński, T. and J. Harabasz, *A dendrite method for cluster analysis.* Communications in Statistics-theory and Methods, 1974. **3**(1): p. 1-27.
155. Kumar, A., et al., *Canopy clustering: a review on pre-clustering approach to K-Means clustering.* Int. J. Innov. Adv. Comput. Sci.(IJIACS), 2014. **3**(5): p. 22-29.
156. McCallum, A., K. Nigam, and L.H. Ungar. *Efficient clustering of high-dimensional data sets with application to reference matching*. in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000.
157. Xia, D., F. Ning, and W. He, *Research on Parallel Adaptive Canopy-K-Means Clustering Algorithm for Big Data Mining Based on Cloud Platform.* Journal of Grid Computing, 2020: p. 1-11.
158. McCallum, A., K. Nigam, and L.H. Ungar. *Efficient clustering of high-dimensional data sets with application to reference matching*. in *the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000. ACM.
159. Saxena, A., et al., *A review of clustering techniques and developments.* Neurocomputing, 2017. **267**: p. 664-681.
160. Do, C.B. and S. Batzoglou, *What is the expectation maximization algorithm?* Nature biotechnology, 2008. **26**(8): p. 897-899.

161. Metsis, V., I. Androutsopoulos, and G. Paliouras. *Spam filtering with naive bayes-which naive bayes?* in *CEAS 2006 - Third Conference on Email and Anti-Spam*. 2006. Mountain View, CA.

162. Wu, X., et al., *Top 10 algorithms in data mining.* Knowledge and information systems, 2008. **14**(1): p. 1-37.

163. Panda, M. and M.R. Patra. *A comparative study of data mining algorithms for network intrusion detection*. in *2008 First International Conference on Emerging Trends in Engineering and Technology*. 2008. IEEE.

164. Read, J., et al., *Batch-incremental versus instance-incremental learning in dynamic and evolving data*, in *Advances in Intelligent Data Analysis XI*. 2012, Springer. p. 313-323.

165. Bouckaert, R.R., *Voting massive collections of bayesian network classifiers for data streams*, in *AI 2006: Advances in Artificial Intelligence*. 2006, Springer. p. 243-252.

166. Han, J., M. Kamber, and J. Pei, *Data mining: concepts and techniques: concepts and techniques*. 2011: Elsevier.

167. Amores, J., N. Sebe, and P. Radeva, *Boosting the distance estimation: Application to the k-nearest neighbor classifier.* Pattern Recognition Letters, 2006. **27**(3): p. 201-209.

168. Aha, D.W., D. Kibler, and M.K. Albert, *Instance-based learning algorithms.* Machine learning, 1991. **6**(1): p. 37-66.

169. Mucherino, A., P.J. Papajorgji, and P.M. Pardalos, *K-nearest neighbor classification*, in *Data mining in agriculture*. 2009, Springer. p. 83-106.

170. Syakur, M.A., et al. *Integration k-means clustering method and elbow method for identification of the best customer profile cluster*. in *IOP Conference Series: Materials Science and Engineering*. 2018. IOP Publishing.

171. Ghosh, A.K., *On optimum choice of k in nearest neighbor classification.* Computational Statistics & Data Analysis, 2006. **50**(11): p. 3113-3123.

172. Wang, H., et al. *Mining concept-drifting data streams using ensemble classifiers*. in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003. ACM.

173. Domingos, P. and G. Hulten. *Mining high-speed data streams*. in *the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000. ACM.

174. Re, M. and G. Valentini *Ensemble methods: a review*. 2012.

175. Aljamaan, H.I. and M.O. Elish. *An empirical study of bagging and boosting ensembles for identifying faulty classes in object-oriented software*. in *IEEE Symposium on Computational Intelligence and Data Mining*. 2009. IEEE.

176. Guo, H. and H.L. Viktor, *Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach.* ACM SIGKDD Explorations Newsletter, 2004. **6**(1): p. 30-39.

177. Bühlmann, P., *Bagging, boosting and ensemble methods*, in *Handbook of computational statistics*. 2012, Springer. p. 985-1022.

178. Jayasree, S. and A.A. Gavya, *Addressing imbalance problem in the class–A survey.*

179. Breiman, L., *Bagging predictors.* Machine learning, 1996. **24**(2): p. 123-140.

180. Sewell, M., *Ensemble methods.* Relatório Técnico RN/11/02, University College London Departament of Computer Science, 2011.

181. Giovanni, S. and E. John, *Ensemble Methods in Data Mining:Improving Accuracy Through Combining Predictions*. Ensemble Methods in Data Mining:Improving Accuracy Through Combining Predictions. 2010: Morgan & Claypool. 126.

182. Freund, Y. and R.E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of computer and system sciences, 1997. **55**(1): p. 119-139.
183. Freund, Y. and R.E. Schapire. *A desicion-theoretic generalization of on-line learning and an application to boosting*. in *European conference on computational learning theory*. 1995. Springer.
184. Ho, T.K., *The random subspace method for constructing decision forests.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998. **20**(8): p. 832-844.
185. Sun, S. *An improved random subspace method and its application to EEG signal classification*. in *International Workshop on Multiple Classifier Systems*. 2007. Springer.
186. Baskin, I.I., et al., *Random subspaces and random forest.* Tutorials in Chemoinformatics, 2017: p. 263-269.
187. Mielniczuk, J. and P. Teisseyre, *Using random subspace method for prediction and variable importance assessment in linear regression.* Computational Statistics & Data Analysis, 2014. **71**: p. 725-742.
188. Cong, Z., et al. *Human resource recommendation algorithm based on ensemble learning and Spark*. in *Journal of Physics: Conference Series*. 2017. IOP Publishing.
189. Lili, C., *Recommender Algorithms Based on Boosting Ensemble Learning.* International Journal on Smart Sensing & Intelligent Systems, 2015. **8**(1).
190. Mu, R., *A survey of recommender systems based on deep learning.* IEEE Access, 2018. **6**: p. 69009-69022.
191. Liu, J.Y. *A Survey of Deep Learning Approaches for Recommendation Systems*. in *Journal of Physics: Conference Series*. 2018. IOP Publishing.
192. Batmaz, Z., et al., *A review on deep learning for recommender systems: challenges and remedies.* Artificial Intelligence Review, 2019. **52**(1): p. 1-37.
193. Liu, W., et al., *A survey of deep neural network architectures and their applications.* Neurocomputing, 2017. **234**: p. 11-26.
194. Zheng, L. *A survey and critique of deep learning on recommender systems*. 2016. 31.
195. Zhang, S., et al., *Deep learning based recommender system: A survey and new perspectives.* ACM Computing Surveys (CSUR), 2019. **52**(1): p. 5.
196. Khan, S.S. and M.G. Madden. *A survey of recent trends in one class classification*. in *Irish conference on artificial intelligence and cognitive science*. 2009. Springer.
197. Lampropoulos, A.S. and G.A. Tsihrintzis. *Evaluation of density one-class classifiers for item-based filtering*. in *IISA 2013*. IEEE.
198. Lampropoulos, A.S., D.N. Sotiropoulos, and G.A. Tsihrintzis, *A music recommender based on artificial immune systems*, in *Intelligent Interactive Multimedia Systems and Services*. 2010, Springer. p. 167-179.
199. Sun, Y., et al., *Cost-sensitive boosting for classification of imbalanced data.* Pattern Recognition, 2007. **40**(12): p. 3358-3378.
200. Moya, M.M. and D.R. Hush, *Network constraints and multi-objective optimization for one-class classification.* Neural Networks, 1996. **9**(3): p. 463-474.
201. Hempstalk, K., E. Frank, and I.H. Witten. *One-class classification by combining density and class probability estimation*. in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2008. Springer.
202. Sabokrou, M., et al. *Adversarially learned one-class classifier for novelty detection*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

203. Deng, X., et al., *An intelligent outlier detection method with one class support tucker machine and genetic algorithm toward big sensor data in Internet of Things.* IEEE Transactions on Industrial Electronics, 2018. **66**(6): p. 4672-4683.
204. Witten, I.H., et al., *Data Mining: Practical machine learning tools and techniques*. 2016: Morgan Kaufmann.
205. Panov, P. and S. Džeroski. *Combining bagging and random subspaces to create better ensembles*. in *International Symposium on Intelligent Data Analysis*. 2007. Springer.
206. Wei, K., J. Huang, and S. Fu. *A survey of e-commerce recommender systems*. in *2007 international conference on Service systems and service management*. 2007. IEEE.
207. Kanagal, B., et al., *Supercharging recommender systems using taxonomies for learning user purchase behavior.* Proceedings of the VLDB Endowment, 2012. **5**(10): p. 956-967.
208. Wang, H., N. Wang, and D.-Y. Yeung. *Collaborative deep learning for recommender systems*. in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015. ACM.
209. Guo, G., J. Zhang, and D. Thalmann. *A simple but effective method to incorporate trusted neighbors in recommender systems*. in *International Conference on User Modeling, Adaptation, and Personalization*. 2012. Springer.
210. Li, X., et al. *Rank-geofm: A ranking based geographical factorization method for point of interest recommendation*. in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2015. ACM.
211. Lian, D., et al. *GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation*. in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014. ACM.
212. Pande, S.R., S.S. Sambare, and V.M. Thakre, *Data clustering using data mining techniques.* International Journal of Advanced Research in Computer and Communication Engineering, 2012. **1**(8): p. 494-9.
213. Frank, E., M.A. Hall, and I.H. Witten, *The WEKA workbench.* Data mining: Practical machine learning tools and techniques, 2016. **4**.
214. Vargas-Govea, B., G. González-Serna, and R. Ponce-Medellın, *Effects of relevant contextual features in the performance of a restaurant recommender system.* ACM RecSys, 2011. **11**(592): p. 56.
215. Canada, N.R., *Fuel Consumption Ratings*. 2018: Open Government Canada
216. Alabdulrahman, R., H. Viktor, and E. Paquet. *An Active Learning Approach for Ensemble-based Data Stream Mining*. in *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 2016. Porto, Portugal: SCITEPRESS-Science and Technology Publications, Lda.
217. Mythili, S. and E. Madhiya, *An analysis on clustering algorithms in data mining.* Journal IJCSMC, 2014. **3**(1): p. 334-340.
218. Zhang, Y. and T. Li, *Dclustere: A framework for evaluating and understanding document clustering using visualization.* ACM Transactions on Intelligent Systems and Technology (TIST), 2012. **3**(2): p. 24.
219. Demšar, J., *Statistical comparisons of classifiers over multiple data sets.* Journal of Machine learning research, 2006. **7**(Jan): p. 1-30.
220. Flach, P., *Machine learning: the art and science of algorithms that make sense of data.* 2012: Cambridge University Press. 3.

221. Zheng, Y., X. Xu, and L. Qi, *Deep CNN-Assisted Personalized Recommendation over Big Data for Mobile Wireless Networks.* Wireless Communications and Mobile Computing, 2019.

222. Yu, B., et al. *Multi-Source News Recommender System Based on Convolutional Neural Networks*. in *Proceedings of the 3rd International Conference on Intelligent Information Processing*. 2018. ACM.

223. Karimi, R., et al., *Towards Optimal Active Learning for Matrix Factorization in Recommender Systems*, in *2011 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. 2011, IEEE. p. 1069-1076.

224. Gope, J. and S.K. Jain, *A Survey on Solving Cold Start Problem in Recommender Systems*. 2017 Ieee International Conference on Computing, Communication and Automation, ed. P.N. Astya, et al. 2017. 133-138.

225. Fernandez-Tobias, I., et al., *Alleviating the new user problem in collaborative filtering by exploiting personality information.* User Modeling and User-Adapted Interaction, 2016. **26**(2-3): p. 221-255.

226. Soundarya, V., U. Kanimozhi, and D. Manjula, *Recommendation System for Criminal Behavioral Analysis on Social Network using Genetic Weighted K-Means Clustering.* JCP, 2017. **12**(3): p. 212-220.

227. Zhou, W., et al., *Deep Learning Modeling for Top-N Recommendation With Interests Exploring.* IEEE Access, 2018. **6**: p. 51440-51455.

228. Sun, G.-L., et al., *Personalized clothing recommendation combining user social circle and fashion style consistency.* Multimedia Tools and Applications, 2018. **77**(14): p. 17731-17754.

229. Xiong, M.T., et al., *TDCTFIC: A Novel Recommendation Framework Fusing Temporal Dynamics, CNN-Based Text Features and Item Correlation.* IEICE Transactions on Information and Systems, 2019. **102**(8): p. 1517-1525.

230. Sun, L.-L. and X.-Z. Wang. *A survey on active learning strategy*. in *2010 International Conference on Machine Learning and Cybernetics*. 2010. IEEE.

231. Settles, B., *Active learning literature survey.* University of Wisconsin, Madison, 2010. **52**(55-66): p. 11.

232. Huang, S.-J., R. Jin, and Z.-H. Zhou. *Active learning by querying informative and representative examples*. in *Advances in neural information processing systems*. 2010.

233. Elahi, M., F. Ricci, and N. Rubens, *Active Learning in Collaborative Filtering Recommender Systems*, in *E-Commerce and Webtechnologies*, M. Hepp and Y. Hoffner, Editors. 2014. p. 113-124.

234. Elahi, M., F. Ricci, and N. Rubens, *A survey of active learning in collaborative filtering recommender systems.* Computer Science Review, 2016. **20**: p. 29-50.

235. Aloysius, N. and M. Geetha. *A review on deep convolutional neural networks*. in *2017 International Conference on Communication and Signal Processing (ICCSP)*. 2017. IEEE.

236. Ahmad, J., H. Farman, and Z. Jan, *Deep learning methods and applications*, in *Deep Learning: Convergence to Big Data Analytics*. 2019, Springer. p. 31-42.

237. Kotkov, D., et al. *Investigating serendipity in recommender systems based on real user feedback*. in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 2018. ACM.

238. Harper, F.M. and J.A. Konstan, *The movielens datasets: History and context.* Acm transactions on interactive intelligent systems (tiis), 2016. **5**(4): p. 19.

239. Chaaya, G., et al., *Evaluating Non-Personalized Single-Heuristic Active Learning Strategies for Collaborative Filtering Recommender Systems*. 2017 16th Ieee International Conference on Machine Learning and Applications, ed. X. Chen, et al. 2017. 593-600.
240. Schafer, J.B., J.A. Konstan, and J. Riedl, *E-commerce recommendation applications.* Data mining and knowledge discovery, 2001. **5**(1-2): p. 115-153.
241. Srivastava, A., P.K. Bala, and B. Kumar, *New perspectives on gray sheep behavior in E-commerce recommendations.* Journal of Retailing and Consumer Services, 2019.
242. Ma, T., et al., *Social network and tag sources based augmenting collaborative recommender system.* IEICE transactions on Information and Systems, 2015. **98**(4): p. 902-910.
243. He, M., C. Ren, and H. Zhang, *Intent-based recommendation for B2C e-commerce platforms.* Ibm Journal of Research and Development, 2014. **58**(5-6).
244. Chawla, N.V., et al. *SMOTEBoost: Improving prediction of the minority class in boosting*. in *European conference on principles of data mining and knowledge discovery*. 2003. Springer.
245. Ghorbani, S. and A.H. Novin, *An introduction on separating gray-sheep users in personalized recommender systems using clustering solution.* International Journal of Computer Science and Software Engineering, 2016. **5**(2): p. 14.
246. Ghazanfar, M.A. and A. Prügel-Bennett, *Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems.* Expert Systems with Applications, 2014. **41**(7): p. 3261-3275.
247. Longadge, R. and S. Dongre, *Class Imbalance Problem in Data Mining Review.* arXiv preprint arXiv:1305.1707, 2013.
248. Kotsiantis, S. and P. Pintelas, *Mixture of expert agents for handling imbalanced data sets.* Annals of Mathematics, Computing & Teleinformatics, 2003. **1**(1): p. 46-55.
249. Li, X. and H.C. Chen, *Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach.* Decision Support Systems, 2013. **54**(2): p. 880-890.
250. Li, X. and H. Chen, *Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach.* Decision Support Systems, 2013. **54**(2): p. 880-890.
251. Mollineda, R.A., R. Alejo, and J.M. Sotoca. *The class imbalance problem in pattern classification and learning*. in *II Congreso Español de Informática (CEDI 2007). ISBN*. 2007. Citeseer.
252. Swallow, W.H. and F. Kianifard, *Using robust scale estimates in detecting multiple outliers in linear regression.* Biometrics, 1996: p. 545-556.
253. Ndirangu, D., W. Mwangi, and L. Nderu. *An Ensemble Filter Feature Selection Method and Outlier Detection Method for Multiclass Classification*. in *Proceedings of the 2019 8th International Conference on Software and Computer Applications*. 2019.
254. Nair, P. and I. Kashyap. *Hybrid Pre-processing Technique for Handling Imbalanced Data and Detecting Outliers for KNN Classifier*. in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. 2019. IEEE.
255. Witten, I.H., E. Frank, and M.A. Hall, *Data Mining: Practical machine learning tools and techniques*. 3rd edition ed. 2011: Morgan Kaufmann.
256. Brew, A., M. Grimaldi, and P. Cunningham, *An evaluation of one-class classification techniques for speaker verification.* Artificial Intelligence Review, 2007. **27**(4): p. 295-307.

257. Perdisci, R., G. Gu, and W. Lee. *Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems*. in *Sixth International Conference on Data Mining (ICDM'06)*. 2006. IEEE.
258. Alashwal, H., S. Deris, and R.M. Othman, *One-class support vector machines for protein-protein interactions prediction.* International Journal of Biological and Medical Sciences, 2006. **1**(2).
259. Markham, K. *Simple guide to confusion matrix terminology*. 2014 [cited 2020; Available from: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/.
260. developer, G. *Classification: True vs. False and Positive vs. Negative*. 2020; Available from: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative.
261. Deschamps, T. *'2030 overnight': Shopify offers help to firms grappling with COVID-19*. 2020; Available from: https://www.bnnbloomberg.ca/2030-overnight-shopify-offers-help-to-firms-grappling-with-covid-19-1.1438934.
262. Simpson, M. *Shopify Sees 47 Percent Revenue Growth in Q1 2020 Amid Impacts of Covid-19*. 2020; Available from: https://betakit.com/shopify-sees-47-percent-revenue-growth-in-q1-2020-amid-impacts-of-covid-19/.
263. Ha, I., K.J. Oh, and G.S. Jo, *Personalized advertisement system using social relationship based user modeling.* Multimedia Tools and Applications, 2015. **74**(20): p. 8801-8819.
264. Zhao, L.L., S.J. Pan, and Q. Yang, *A unified framework of active transfer learning for cross-system recommendation.* Artificial Intelligence, 2017. **245**: p. 38-55.
265. Seo, K.-K., *An application of one-class support vector machines in content-based image retrieval.* Expert Systems with Applications, 2007. **33**(2): p. 491-498.
266. Nicolau, M. and J. McDermott. *A hybrid autoencoder and density estimation model for anomaly detection*. in *International Conference on Parallel Problem Solving from Nature*. 2016. Springer.
267. Moulton, R.H., *Clustering to Improve One-Class Classifier Performance in Data Streams* 2018.
268. Vartak, M., et al. *A meta-learning perspective on cold-start recommendations for items*. in *Advances in neural information processing systems*. 2017.
269. Elkahky, A.M., Y. Song, and X. He. *A multi-view deep learning approach for cross domain user modeling in recommendation systems*. in *Proceedings of the 24th International Conference on World Wide Web*. 2015.

Appendix 1

Table 52. Prediction rates in details for the PUPP-DA framework – results in percentage.

Random Split			Popularity Split		
User ID	Iteration 1	Iteration 2	User ID	Iteration 1	Iteration 2
Architecture 1					
1	44%	44%	68	36%	34%
225	45%	43%	182	34%	39%
282	31%	41%	274	47%	34%
304	50%	44%	288	49%	42%
34	31%	32%	414	44%	42%
374	49%	43%	448	46%	41%
412	44%	45%	474	34%	42%
450	44%	43%	477	18%	33%
510	32%	32%	599	36%	32%
602	53%	46%	603	51%	44%
Architecture 2					
1	44%	44%	68	33%	32%
225	44%	43%	182	38%	37%
282	31%	30%	274	32%	32%
304	45%	44%	288	45%	43%
34	32%	30%	414	45%	43%
374	46%	43%	448	43%	41%
412	45%	44%	474	45%	43%
450	44%	44%	477	32%	32%
510	32%	31%	599	32%	32%
602	47%	45%	603	45%	43%
Architecture 3					
1	45%	43%	68	30%	30%
225	44%	42%	182	34%	32%
282	29%	28%	274	30%	30%
304	44%	42%	288	44%	42%
34	29%	28%	414	44%	42%
374	41%	39%	448	42%	41%
412	45%	21%	474	44%	42%
450	44%	42%	477	30%	30%
510	29%	28%	599	30%	30%
602	41%	39%	603	43%	43%

Table 52. (continued).

Random Split				Popularity Split		
User ID	Iteration 1	Iteration 2		User ID	Iteration 1	Iteration 2
Architecture 4						
1	53%	48%		68	33%	32%
225	52%	47%		182	49%	47%
282	30%	31%		274	33%	31%
304	57%	57%		288	54%	58%
34	29%	29%		414	53%	57%
374	54%	54%		448	54%	57%
412	48%	46%		474	53%	54%
450	46%	48%		477	32%	32%
510	30%	30%		599	34%	34%
602	56%	67%		603	24%	56%
Architecture 5						
1	40%	41%		68	27%	30%
225	38%	39%		182	42%	44%
282	27%	26%		274	27%	30%
304	39%	40%		288	46%	45%
34	27%	26%		414	46%	45%
374	40%	42%		448	42%	45%
412	38%	39%		474	44%	45%
450	38%	39%		477	27%	30%
510	27%	26%		599	27%	30%
602	41%	42%		603	43%	45%
Architecture 6						
1	43%	42%		68	27%	30%
225	43%	41%		182	41%	23%
282	29%	27%		274	27%	30%
304	43%	45%		288	47%	44%
34	29%	27%		414	48%	46%
374	40%	46%		448	46%	45%
412	42%	42%		474	47%	46%
450	43%	42%		477	27%	30%
510	29%	27%		599	27%	30%
602	40%	48%		603	47%	44%

Table 52. (continued).

Random Split				Popularity Split		
User ID	Iteration 1	Iteration 2		User ID	Iteration 1	Iteration 2
Architecture 7						
1	100%	100%		68	100%	100%
225	100%	99.50%		182	100%	100%
282	100%	100%		274	100%	100%
304	100%	100%		288	100%	100%
34	100%	99.50%		414	100%	100%
374	100%	99.50%		448	100%	100%
412	100%	99.90%		474	100%	100%
450	100%	99.90%		477	100%	100%
510	100%	99.85%		599	100%	100%
602	100%	99.90%		603	100%	100%
Architecture 8						
1	100%	100%		68	100%	100%
225	100%	99.25%		182	100%	100%
282	100%	100%		274	100%	100%
304	100%	100%		288	100%	100%
34	100%	99.75%		414	100%	100%
374	100%	99.60%		448	100%	100%
412	100%	100%		474	100%	100%
450	100%	99.95%		477	100%	100%
510	100%	99.95%		599	100%	100%
602	100%	100%		603	100%	100%
Architecture 9						
1	100%	100%		68	100%	100%
225	100%	99.65%		182	100%	100%
282	100%	100%		274	100%	100%
304	100%	100%		288	100%	100%
34	99.90%	99.70%		414	100%	100%
374	100%	99.80%		448	100%	100%
412	100%	99.90%		474	100%	100%
450	99.95%	100%		477	100%	100%
510	100%	99.95%		599	100%	100%
602	100%	100%		603	100%	100%

Table 52. (continued).

Random Split			Popularity Split		
User ID	Iteration 1	Iteration 2	User ID	Iteration 1	Iteration 2
Architecture 10					
1	100%	100%	68	100%	100%
225	100%	100%	182	100%	100%
282	100%	100%	274	100%	100%
304	100%	100%	288	100%	100%
34	100%	100%	414	100%	100%
374	100%	100%	448	100%	100%
412	100%	100%	474	100%	100%
450	100%	100%	477	100%	100%
510	100%	100%	599	100%	100%
602	100%	100%	603	100%	100%
Architecture 11					
1	100%	100%	68	100%	100%
225	100%	100%	182	100%	100%
282	100%	100%	274	100%	100%
304	100%	100%	288	100%	100%
34	100%	100%	414	100%	100%
374	100%	100%	448	100%	100%
412	100%	100%	474	100%	100%
450	100%	100%	477	100%	100%
510	100%	100%	599	100%	100%
602	100%	100%	603	100%	100%
Architecture 12					
1	31.10%	30.90%	68	29.70%	29.30%
225	31.10%	30.90%	182	29.70%	29.30%
282	31.10%	30.90%	274	29.70%	29.30%
304	31.10%	30.90%	288	29.70%	29.30%
34	31.10%	30.90%	414	29.70%	29.30%
374	31.10%	30.90%	448	29.70%	29.30%
412	31.10%	30.90%	474	29.70%	29.30%
450	31.10%	30.90%	477	29.70%	29.30%
510	31.10%	30.90%	599	29.70%	29.30%
602	31.10%	30.90%	603	29.70%	29.30%