

<p>An effective distributed predictive model with Matrix factorization and random forest for Big Data recommendation systems</p>	<p>Эффективная модель распределенного прогнозирования с матричной факторизацией и случайным лесом для систем рекомендаций на основе больших данных</p>
<p>a b s t r a c t</p>	<p>Аннотация</p>
<p>Recommendation systems have been widely deployed to address the challenge of overwhelming information. They are used to enable users to find interesting information from a large volume of data. However, in the era of Big Data, as data become larger and more complicated, a recommendation algorithm that runs in a traditional environment cannot be fast and effective. It requires a high computational cost for performing the training task, which may limit its applicability in real-world Big Data applications.</p>	<p>Системы рекомендаций широко используются для решения проблемы подавляющей информации. Они используются, чтобы позволить пользователям находить интересную информацию из большого объема данных. Однако в эпоху больших данных, поскольку данные становятся все больше и сложнее, алгоритм рекомендаций, который работает в традиционной среде, не может быть быстрым и эффективным. Это требует высоких вычислительных затрат для выполнения учебного задания, что может ограничить его применимость в реальных приложениях больших данных.</p>
<p>In this paper, we propose a novel distributed recommendation solution for Big Data. It is designed based on Apache Spark to handle large-scale data, improve the prediction quality, and address the data sparsity problem. In particular, thanks to a novel learning process, the model is able to significantly speed up the distributed training, as well as improve the performance in the context of Big Data. Experimental results on three real-world data sets demonstrate that our proposal outperforms existing recommendation methods in terms of Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and computational time.</p>	<p>В этой статье мы предлагаем новое решение с распределенными рекомендациями для больших данных. Он разработан на основе Apache Spark для обработки крупномасштабных данных, улучшения качества прогнозирования и решения проблемы разреженности данных. В частности, благодаря новому процессу обучения, модель способна значительно ускорить распределенное обучение, а также повысить производительность в контексте больших данных. Экспериментальные результаты на трех реальных наборах данных демонстрируют, что наше предложение превосходит существующие методы рекомендаций с точки зрения средней абсолютной ошибки (MAE), среднеквадратической ошибки (RMSE) и времени вычислений.</p>
<p>1. Introduction</p>	<p>1. Введение</p>
<p>In recent years, Big Data has emerged as the key to success in many companies, scientific disciplines, and government endeavors. The term Big Data generally refers to data sets whose volume is beyond the capacity of conventional tools to capture, manage, and process data within an acceptable computational time. The challenge is to explore and analyze massive data to extract relevant information needed for specific objectives (Hu, Dou, & Liu, 2014; Takaishi, Nishiyama, Kato, &</p>	<p>В последние годы большие данные стали ключом к успеху во многих компаниях, научных дисциплинах и усилиях правительства. Термин «большие данные» обычно относится к наборам данных, объем которых превышает возможности традиционных инструментов для сбора, управления и обработки данных в течение приемлемого вычислительного времени. Задача состоит в том, чтобы исследовать и анализировать массивные данные для извлечения соответствующей</p>

<p>Miura, 2014; Zhang, Zhou, Wang, Sun, & He, 2018).</p>	<p>информации, необходимой для достижения конкретных целей (Ху, Доу и Лю, 2014; Такаиси, Нисияма, Като и Миура, 2014; Чжан, Чжоу, Ван, Сунь и Хе 2018).</p>
<p>Due to the explosive growth of information available on the Web, users confront the crucial challenge of overwhelming choices, which is known as the information overload problem (Ait Hammou, Ait Lahcen, & Mouline, 2019b; Sun, Wu, Wu, & Ye, 2019). It is difficult for users to find interesting information. This problem has increased the need for efficient information filtering technology to help users find the appropriate items like books, movies, music according to their needs (Ait Hammou & Ait Lahcen, 2017; Kaleli, 2014; Ma, Lu, Gan, & Zhao, 2016).</p>	<p>Из-за взрывного роста информации, доступной в Интернете, пользователи сталкиваются с критической проблемой подавляющего выбора, которая известна как проблема информационной перегрузки (Ait Hammou, Ait Lahcen, & Mouline, 2019b; Sun, Wu, Wu, & Ye, 2019). Пользователям сложно найти интересную информацию. Эта проблема увеличила потребность в эффективной технологии фильтрации информации, чтобы помочь пользователям находить нужные предметы, такие как книги, фильмы, музыка, в соответствии с их потребностями (Ait Hammou & Ait Lahcen, 2017; Kaleli, 2014; Ma, Lu, Gan, & Чжао, 2016).</p>
<p>Recommendation systems have emerged as powerful tools to address the mentioned problem. They aim to provide users with personalized suggestions based on their past preferences and interests (Borràs, Moreno, & Valls, 2014; Lu, Wu, Mao, Wang, & Zhang, 2015; Mao, Lu, Zhang, & Zhang, 2017; Wang, Cheng, Jiang, & Lou, 2019). Examples of real-world applications include e-government, social network, e-commerce, e-learning, and so on (Ait Hammou et al., 2019b; Al-Shamri, 2014; Bobadilla, Ortega, Hernando, & Gutiérrez, 2013; Ortega, Hernando, Bobadilla, & Kang, 2016; Wu, Lu, & Zhang, 2015; Zhang et al., 2016).</p>	<p>Системы рекомендаций стали мощными инструментами для решения упомянутой проблемы. Они стремятся предоставить пользователям персонализированные предложения, основанные на их прошлых предпочтениях и интересах (Borràs, Moreno, & Valls, 2014; Lu, Wu, Mao, Wang, & Zhang, 2015; Mao, Lu, Zhang, & Zhang, 2017 Wang, Cheng, Jiang & Lou, 2019). Примеры реальных приложений включают электронное правительство, социальные сети, электронную коммерцию, электронное обучение и т. Д. (Ait Hammou et al., 2019b; Al-Shamri, 2014; Bobadilla, Ortega, Hernando, & Gutiérrez, 2013 ; Ortega, Hernando, Bobadilla, & Kang, 2016; Wu, Lu, & Zhang, 2015; Zhang et al., 2016).</p>
<p>Collaborative filtering (CF) is one of the most successful approaches used to build recommendation systems (Osadchiy, Po liakov, Olivier, Rowland, & Foster, 2019; Turk & Bilge, 2019; Zhang et al., 2016). In general, the CF methods are divided into two categories: model-based and memory-based CF. The model based methods provide recommendations based on a mathematical model, while memory-based methods such as user-based and item-based CF predict the unknown ratings by aggregating the preferences of the most similar users, or items, respectively (Mazurowski, 2013; Turk & Bilge, 2019; Zhang et al., 2016; Zhang, Lu, Chen, Liu, & Ling, 2017a).</p>	<p>Коллаборативная фильтрация (CF) является одним из наиболее успешных подходов, используемых для построения систем рекомендаций (Осадчий, Поляков, Оливье, Роуланд и Фостер, 2019; Turk & Bilge, 2019; Zhang et al., 2016). В целом, методы CF делятся на две категории: основанные на модели и основанные на памяти CF. Методы, основанные на модели, предоставляют рекомендации, основанные на математической модели, в то время как методы, основанные на памяти, такие как пользовательский и основанный на элементах CF, предсказывают неизвестные оценки, агрегируя предпочтения наиболее похожих пользователей или элементов,</p>

	соответственно. (Mazurowski, 2013; Turk & Bilge, 2019; Zhang et al., 2016; Zhang, Lu, Chen, Liu & Ling, 2017a).
Currently, like many real-world Big data applications, the rapid increase in the number of users, items, and other information generated has created critical issues for traditional recommendation systems (Zhang et al., 2018). The need to analyze the preferences and interests of users has made it necessary to process an enormous amount of information (Hu et al., 2014; Zhang et al., 2018).	В настоящее время, как и во многих реальных приложениях для работы с большими данными, быстрое увеличение числа пользователей, предметов и другой информации, созданной, создало критические проблемы для традиционных систем рекомендаций (Zhang et al., 2018). Необходимость анализа предпочтений и интересов пользователей привела к необходимости обработки огромного количества информации (Hu et al., 2014; Zhang et al., 2018).
Although several recommendation techniques have proved good performance for small-scale data, they are difficult to be applied in the context of Big Data (Zhang et al., 2018). In particular, the computational cost for performing the training task can be computationally prohibitive on large-scale data, which may limit the applicability in real-world scenarios (Hu et al., 2014; Salah, Rogovschi, & Nadif, 2016; Zhang et al., 2018). Furthermore, taking into account the new preferences in the system involves performing the offline computation. This task costs much time with increasing data volume (Aggarwal, 2016; Ait Hammou & Ait Lahcen, 2017; Zhou, He, Huang, & Zhang, 2015). In addition to this, the data sparsity is another critical problem, which has a negative impact on the prediction quality (Ait Hammou et al., 2019b; Zhou et al., 2015).	Хотя некоторые методики рекомендаций доказали хорошую производительность для небольших данных, их трудно применять в контексте больших данных (Zhang et al., 2018). В частности, вычислительные затраты на выполнение учебного задания могут быть вычислительно непомерными для крупномасштабных данных, что может ограничивать применимость в реальных сценариях (Hu et al., 2014; Salah, Rogovschi, & Nadif, 2016; Zhang et al., 2018). Кроме того, принимая во внимание новые предпочтения в системе, требуется выполнение одних вычислений. Эта задача стоит много времени при увеличении объема данных (Aggarwal, 2016; Ait Hammou & Ait Lahcen, 2017; Zhou, He, Huang, & Zhang, 2015). В дополнение к этому, разрозненность данных является еще одной критической проблемой, которая оказывает негативное влияние на качество прогноза (Ait Hammou et al., 2019b; Zhou et al., 2015).
Therefore, building large-scale recommendation systems requires the consideration of different issues such as coping with the data sparsity, shortening the computational time, improving the quality of predictions, and handling large-scale data efficiently (Ait Hammou, Ait Lahcen, & Mouline, 2018; Hu et al., 2014; Ma et al., 2016; Salah et al., 2016; Zhang et al., 2018).	Таким образом, создание крупномасштабных систем рекомендаций требует рассмотрения различных вопросов, таких как преодоление разреженности данных, сокращение времени вычислений, повышение качества прогнозов и эффективная обработка крупномасштабных данных (Ait Hammou, Ait Lahcen, & Mouline, 2018; Hu et al., 2014; Ma et al., 2016; Salah et al., 2016; Zhang et al., 2018).
This paper represents a continuation of our previous work (Ait Hammou et al., 2018). In particular, it presents an effective distributed predictive model for Big Data recommendation based on Apache Spark. The main objective is to address the aforementioned shortcomings.	Эта статья представляет собой продолжение нашей предыдущей работы (Ait Hammou et al., 2018). В частности, он представляет эффективную модель распределенного прогнозирования для рекомендации больших данных на основе Apache Spark. Основной целью является устранение вышеуказанных недостатков.

The contributions of this work are summarized as follows.	Вклад этой работы резюмируется следующим образом.
<ul style="list-style-type: none"> We propose a distributed recommendation model based on a data partitioning strategy and a novel learning process to speed up the training task, tackle data sparsity, improve the prediction quality, and handle large-scale data effectively. 	<ul style="list-style-type: none"> Мы предлагаем модель распределенных рекомендаций, основанную на стратегии разделения данных и новом процессе обучения, чтобы ускорить учебную задачу, решить проблему редкости данных, улучшить качество прогнозирования и эффективно обрабатывать крупномасштабные данные.
<ul style="list-style-type: none"> To further enhance the overall performance, we devise an approach based on distributed Matrix factorization and Random forest to provide high-quality results. 	<ul style="list-style-type: none"> Для дальнейшего повышения общей производительности мы разрабатываем подход, основанный на факторизации распределенной матрицы и случайного леса, чтобы обеспечить высококачественные результаты.
<ul style="list-style-type: none"> We parallelize the proposed model using the Apache Spark platform. The data are stored in the Hadoop Distributed File System (HDFS). Every operation is performed based on the Resilient Distributed Dataset (RDD), which implies that the operations are efficient and fully scalable. 	Мы распараллеливаем предложенную модель, используя платформу Apache Spark. Данные хранятся в распределенной файловой системе Hadoop (HDFS). Каждая операция выполняется на основе Resilient Distributed Dataset (RDD), что означает, что операции эффективны и полностью масштабируемы.
<ul style="list-style-type: none"> We conduct extensive experiments on three real-world data sets. The experimental results demonstrate the effectiveness of our proposal in comparison with existing state-of-the-art methods. 	Мы проводим обширные эксперименты на трех реальных наборах данных. Экспериментальные результаты демонстрируют эффективность нашего предложения по сравнению с существующими современными методами.
The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 presents the preliminaries. Section 4 describes our proposal. Section 5 details the experiments. Section 6 presents the discussion. Finally, Section 7 concludes this paper.	Остальная часть этой статьи организована следующим образом. Раздел 2 представляет соответствующую работу. Раздел 3 представляет предварительные условия. Раздел 4 описывает наше предложение. Раздел 5 подробно описывает эксперименты. Раздел 6 представляет обсуждение. Наконец, Раздел 7 завершает эту статью.
2. Related work	2. Связанная работа
With the advent of Big Data applications, the huge amount of data has brought several challenges for recommendation systems. These challenges have inspired many researchers to propose several approaches to improve the performance of recommender systems in the context of Big Data. For instance, Hsieh, Weng, and Li (2018) proposed a keyword-aware recommendation system, which is designed to handle large-scale data sets using Apache Hadoop. The main idea behind this system is to exploit the keywords extracted from the textual data of users and items, to improve the performance and alleviate the cold-start problem. Xu, Sun, Ma, and Du (2016) developed a personalized recommender system based on MapReduce. Its purpose is to help researchers and practitioners find the appropriate R&D project opportunities launched by	С появлением приложений для больших данных огромный объем данных принес несколько проблем для систем рекомендаций. Эти проблемы вдохновили многих исследователей предложить несколько подходов для повышения производительности рекомендательных систем в контексте больших данных. Например, Hsieh, Weng и Li (2018) предложили систему рекомендаций с учетом ключевых слов, которая предназначена для обработки крупномасштабных наборов данных с использованием Apache Hadoop. Основная идея этой системы состоит в том, чтобы использовать ключевые слова, извлеченные из текстовых данных пользователей и элементов, для повышения производительности и облегчения

governments and enterprises.	проблемы холодного запуска. Xu, Sun, Ma и Du (2016) разработали персонализированную систему рекомендаций на основе MapReduce. Его цель помочь исследователям и практикам найти подходящие возможности для проектов НИОКР, запущенных правительствами и предприятиями.
<p>Chen et al. (2018a) designed a Disease Diagnosis and Treatment Recommendation System (DDTRS) based on Big Data mining and Cloud Computing. Its main purpose is to recommend medical treatments based on the inspection reports of patients. Lee and Lin (2017) developed a restaurant recommender system based on lambda architecture. It is designed to handle a large amount of data by taking advantage of both batch and stream processing. While Ait Hammou et al. (2018) proposed an approximate parallel recommendation algorithm for Big Data called APRA, which is developed based on Apache Spark to process large-scale data efficiently.</p>	<p>Чен и соавт. (2018a) разработал систему рекомендаций по диагностике и лечению заболеваний (DDTRS), основанную на интеллектуальном анализе больших данных и облачных вычислениях. Его основная цель рекомендовать лечение на основании отчетов об осмотре пациентов. Ли и Лин (2017) разработали систему рекомендаций для ресторанов, основанную на лямбда-архитектуре. Он предназначен для обработки большого объема данных, используя преимущества как пакетной, так и потоковой обработки. В то время как Ait Hammou et al. (2018) предложили приблизительный алгоритм параллельных рекомендаций для больших данных, названный APRA, который разработан на основе Apache Spark для эффективной обработки крупномасштабных данных.</p>
<p>However, due to the sparsity and scalability issues, Zhang et al. (2018) proposed a scalable approach called Covering Algorithm based on Quotient space Granularity analysis on Spark (CA-QGS), which is designed to suggest Web services in a Big Data environment. Hwang, Lee, Kim, Won, and Lee (2016) designed novel recommendation methods, which are based on the notion of category experts rather than neighbors' of neighborhoodbased methods. The idea behind this work is that the users who have evaluated many items in a specific category are considered as knowledgeable about the category, thereby their opinions are useful for other users. Yin, Wang, and Park (2017) proposed two recommendation algorithms for Big Data. The first algorithm (CFRAT) is based on the trust in sociology. While the second one (HRAT) is based on the trust and similarity. Kupisz and Unold (2015) developed a solution for Big Data, which is tailored to speed up the parallel item-based collaborative filtering using Tanimoto coefficient. Zhao and Shang (2010) implemented User-based Collaborative Filtering algorithm to solve the scalability problem using Apache Hadoop. Jiang, Lu, Zhang, and Long (2011) developed a scaling-up item-based collaborative filtering algorithm based on MapReduce, which aims to split the costly computations into four Map-Reduce phases, in order to improve the parallel computation in each phase and minimize the communication cost.</p>	<p>Однако из-за проблем разреженности и масштабируемости Zhang et al. (2018) предложили масштабируемый подход, называемый алгоритмом покрытия, основанный на анализе гранулярности фактор-пространства в Spark (CA-QGS), который предназначен для предложения веб-сервисов в среде больших данных. Хван, Ли, Ким, Вон и Ли (2016) разработали новые методы рекомендаций, которые основаны на понятии экспертов по категориям, а не на соседских методах, основанных на соседстве. Идея, лежащая в основе этой работы, заключается в том, что пользователи, которые оценили многие элементы в определенной категории, считаются осведомленными в этой категории, поэтому их мнения полезны для других пользователей. Инь, Ван и Парк (2017) предложили два рекомендательных алгоритма для больших данных. Первый алгоритм (CFRAT) основан на доверии к социологии. В то время как второй (HRAT) основан на доверии и сходстве. Купиш и Унольд (2015) разработали решение для больших данных, которое адаптировано для ускорения параллельного совместного фильтрования на основе элементов с использованием коэффициента Танимото. Чжао и Шан (2010) реализовали алгоритм совместной фильтрации на основе</p>

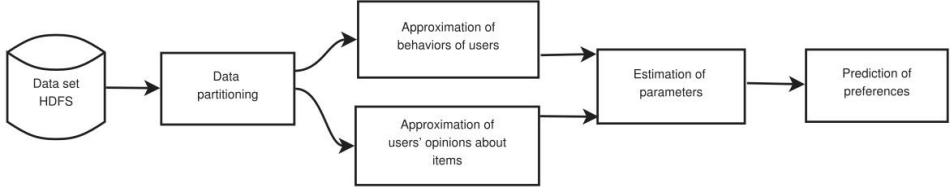
<p>Hu et al. (2014) proposed a Clustering-based Collaborative Filtering approach for Big Data (ClubCF), which employs the clustering to divide data into manageable parts, and item-based collaborative filtering to perform the rating prediction.</p>	<p>пользователей для решения проблемы масштабируемости с использованием Apache Hadoop. Jiang, Lu, Zhang и Long (2011) разработали масштабируемый алгоритм совместной фильтрации на основе элементов, основанный на MapReduce, целью которого является разбиение дорогостоящих вычислений на четыре фазы Map-Reduce, чтобы улучшить параллельные вычисления в каждый этап и минимизировать стоимость связи. Ху и соавт. (2014) предложили подход коллективной фильтрации на основе кластеров для больших данных (ClubCF), который использует кластеризацию для разделения данных на управляемые части и совместную фильтрацию на основе элементов для прогнозирования рейтинга.</p>
<p>Additionally, Winlaw, Hynes, Caterini, and De Sterck (2015) proposed a distributed method based on nonlinear conjugate gradient, which is devoted to speeding up the convergence of the parallel Alternating Least Squares (ALS) algorithm for Big Data Recommendation systems. Kim, Kim, and Min (2019) designed a parallel algorithm based on MapReduce called ConSimMR, which is tailored to construct the similarity matrix for improving collaborative filtering. Li and He (2017) developed an optimized item-based CF algorithm based on MapReduce for big data. Its main objective is improving the scalability and processing efficiency. Singh and Mehrotra (2018) studied the impact of Biclustering based Collaborative Filtering (BBCF) approach on the Performance of recommendation systems. Duma and Twala (2019) designed a collaborative filtering approach based on genetic algorithms and nearest neighbor artificial immune. Hernando, Bobadilla, and Ortega (2016) developed a non-negative matrix factorization based on a Bayesian probabilistic model, which aims to improve the performance of recommendation systems. Yuan, Han, Qian, Xu, and Yan (2019) proposed a recommendation approach called imputation-based SVD (ISVD), which aims to incorporate imputed data into the Singular value decomposition (SVD) model for alleviating data sparsity problem.</p>	<p>Кроме того, Winlaw, Hynes, Caterini и De Sterck (2015) предложили распределенный метод, основанный на нелинейном сопряженном градиенте, который предназначен для ускорения сходимости параллельного алгоритма с чередованием наименьших квадратов (ALS) для больших систем рекомендаций данных. Ким, Ким и Мин (2019) разработали параллельный алгоритм на основе MapReduce, который называется ConSimMR, который предназначен для построения матрицы подобия для улучшения совместной фильтрации. Ли и Хэ (2017) разработали оптимизированный алгоритм CF на основе элементов на основе MapReduce для больших данных. Его главная цель повышение масштабируемости и эффективности обработки. Сингх и Мехротра (2018) изучили влияние подхода коллаборативной фильтрации (BBCF) на основе бикластеризации на производительность систем рекомендаций. Duma и Twala (2019) разработали подход совместного фильтрования, основанный на генетических алгоритмах и искусственном иммунитете ближайшего соседа. Эрнандо, Бобадилла и Ортега (2016) разработали неотрицательную матричную факторизацию на основе байесовской вероятностной модели, которая направлена на повышение производительности рекомендательных систем. Юань, Хан, Цянь, Сюй и Ян (2019) предложили рекомендательный подход, названный SVD (ISVD) на основе вменения, который нацелен на включение вмененных данных в модель разложения по сингулярным значениям (SVD) для решения проблемы разреженности данных.</p>

<p>On the other hand, da Costa, Manzato, and Campello (2019) presented an ensemble-based co-training approach called ECoRec, which employs two or more recommendation approaches to boost the performance of the system. Ait Hammou et al. (2019b) designed a recommendation approach called FRAIPA v2, which is based on self-adaptation and multithresholding. It is tailored to alleviate data sparsity, reduce the computational time, and improve the prediction quality. Godoy-Lorite, Guimerà, Moore, and Sales-Pardo (2016) developed a recommendation model based on expectation maximization method. It aims to infer user preferences based on explicit probabilistic hypotheses about user behavior. Papadakis, Panagiotakis, and Fragopoulou (2017) designed a Synthetic Coordinate based Recommendation system (SCoR) for generating more accurate predictions. The key idea is to place the users and items in a multi-dimensional Euclidean space using the Vivaldi synthetic coordinates algorithm. Then, the Euclidean distance between an item and a user is adopted to predict the unknown preference. Meanwhile, Tran, Lee, Liao, and Lee (2018) proposed a Regularized Multi-Embedding recommendation model (RME), which employs weighted matrix factorization, with user embedding, co-disliked item embedding, and co-liked item embedding to achieve high prediction accuracy.</p>	<p>С другой стороны, да Коста, Манзато и Кампельо (2019) представили основанный на ансамбле подход совместного обучения под названием ECoRec, который использует два или более рекомендаций для повышения производительности системы. Айт Хамму и соавт. (2019b) разработали рекомендательный подход под названием FRAIPA v2, который основан на самоадаптации и многозадачности. Он предназначен для уменьшения разреженности данных, сокращения времени вычислений и улучшения качества прогнозирования. Годой-Лорит, Гимера, Мур и Сэйлс-Пардо (2016) разработали модель рекомендаций, основанную на методе максимизации ожидания. Он направлен на вывод пользовательских предпочтений на основе явных вероятностных гипотез о поведении пользователей. Papadakis, Panagiotakis и Fragopoulou (2017) разработали систему рекомендаций на основе синтетических координат (SCoR) для создания более точных прогнозов. Основная идея заключается в том, чтобы поместить пользователей и предметы в многомерное евклидово пространство, используя алгоритм синтетических координат Вивальди. Затем евклидово расстояние между элементом и пользователем принимается для прогнозирования неизвестного предпочтения. Тем временем Tran, Lee, Liao и Lee (2018) предложили модель рекомендаций Regularized Multi-Embedded (RME), в которой используется взвешенная матричная факторизация, с вложением пользователя, вложением элемента с одинаковым неприятием и вложением элемента с одинаковым вкусом для достижения высокого уровня. точность прогноза.</p>
<p>Table 1</p>	<p>Таблица 1</p>

Notation	Description	
U	The set of users	
M	The number of users	
R_u	The set of ratings expressed by the user u	
I	The set of items	
N	The number of items	
R_i	The set of ratings expressed for the item i	
$[r_{\min}, r_{\max}]$	The rating domain of the data set	
$\hat{r}_{u,i}$	The user u 's predicted rating on item i	
$r_{u,i}$	The user u 's rating on item i	
$ \cdot $	The number of elements in the set	
<p>The various challenges of Big Data and recommendation systems have motivated us to propose an effective distributed predictive model, which is tailored to handle large-scale data, alleviate data sparsity, reduce the computational time, and enhance the prediction quality.</p>		<p>Различные проблемы больших данных и систем рекомендаций побудили нас предложить эффективную модель распределенного прогнозирования, предназначенную для обработки крупномасштабных данных, уменьшения разреженности данных, сокращения времени вычислений и повышения качества прогнозирования.</p>
3. Preliminaries		3. Подготовительные мероприятия
<p>This section presents the necessary background for understanding the remainder of this paper, including the problem definition, matrix factorization and the big data frameworks Apache Hadoop and Apache Spark used to implement our proposal.</p>		<p>В этом разделе представлены необходимые предпосылки для понимания оставшейся части этой статьи, включая определение проблемы, матричную факторизацию и структуры больших данных Apache Hadoop и Apache Spark, использованные для реализации нашего предложения.</p>
3.1. Problem definition		3.1. Определение проблемы
<p>In this paper, we dealt with the personalized recommendation problem in the context of Big Data. Let $U = \{u_1, u_2, \dots, u_M\}$ and $I = \{i_1, i_2, \dots, i_N\}$ be the sets of users and items, respectively. Let M and N be the number of users and items in the system. The preferences provided by each user u are represented as a rating vector $R_u = (r_{u,i_1}, r_{u,i_2}, \dots, r_{u,i_N})$, where $r_{u,i}$ denotes the rating expressed by the user u for the item i. All users' preferences for items are represented using the user-item rating matrix $R \in \mathbb{R}^{M \times N}$.</p>		<p>В этой статье мы рассмотрели проблему персональных рекомендаций в контексте больших данных. Пусть $U = \{u_1, u_2, \dots, u_M\}$ и $I = \{i_1, i_2, \dots, i_N\}$ наборы пользователей и элементов соответственно. Пусть M и N количество пользователей и элементов в системе. Предпочтения, предоставляемые каждым пользователем u, представляются в виде вектора рейтинга $R_u = (r_{u,i_1}, r_{u,i_2}, \dots, r_{u,i_N})$, где $r_{u,i}$ обозначает рейтинг, выраженный пользователем u для элемента i. Все предпочтения пользователей для элементов представлены с использованием матрицы рейтинга элементов пользователя $R \in \mathbb{R}^{M \times N}$.</p>

As each user $u \in U$ rates only a very small number of items, $r_{u,i}$ is unknown for most pairs (u, i) in the rating matrix R , which is the principal cause of data sparsity. The main objective of our work is to analyze the past preferences of users to infer the preferences for the unseen items. Table 1 presents the descriptions of notations used in the rest of this paper.	Поскольку каждый пользователь $u \in U$ оценивает только очень небольшое количество элементов, $r_{u,i}$ неизвестно для большинства пар (u, i) в матрице оценок R , что является основной причиной разреженности данных. Основная цель нашей работы проанализировать прошлые предпочтения пользователей, чтобы вывести предпочтения для невидимых предметов. В таблице 1 представлены описания обозначений, используемых в остальной части этой статьи.
3.2. Matrix factorization	3.2. Матричная факторизация
Matrix factorization (MF) is a dimensionality reduction technique, which belongs to the class of latent factor models. It has gained great popularity due to superior performance in terms of scalability and recommendation quality. The main idea behind Matrix factorization is that the items and the users can be represented by a small number of latent factors inferred from the rating matrix (Fernández-Tobías, Cantador, Tomeo, Anelli, & Di Noia, 2019; Koren, Bell, & Volinsky, 2009; Liu, 2007). Specifically, given the user-item rating matrix R and the number of latent factors $C < \min(M, N)$, the MF model learns an approximation of the matrix $R \in \mathbb{R}^{M \times N}$ as the product of two low-rank matrices $E \in \mathbb{R}^{M \times C}$ and $W \in \mathbb{R}^{C \times N}$ as follows:	Матричная факторизация (MF) это метод уменьшения размерности, который относится к классу моделей скрытых факторов. Он приобрел большую популярность благодаря превосходной производительности с точки зрения масштабируемости и качества рекомендаций. Основная идея факторизации матрицы состоит в том, что элементы и пользователи могут быть представлены небольшим количеством скрытых факторов, выведенных из матрицы рейтинга (Фернандес-Тобиас, Кантадор, Томео, Анелли и Ди Нойя, 2019; Корен, Белл, & Volinsky, 2009; Liu, 2007). В частности, учитывая матрицу R рейтинга элементов пользователя и количество скрытых факторов $C < \min(M, N)$, модель MF изучает приближение матрицы $R \in \mathbb{R}^{M \times N}$ как произведение двух матриц низкого ранга $E \in \mathbb{R}^{M \times C}$ и $W \in \mathbb{R}^{C \times N}$ следующим образом:
$R \approx EW$	
where each row E_u of the user-factor matrix $E \in \mathbb{R}^{M \times C}$ is a C -dimensional vector associated to a user u . Each row W_i of the item-factor matrix $W \in \mathbb{R}^{C \times N}$ is associated to an item i .	где каждая строка E_u матрицы пользовательских факторов $E \in \mathbb{R}^{M \times C}$ является многомерным вектором, связанным с пользователем u . Каждая строка W_i матрицы фактора $W \in \mathbb{R}^{C \times N}$ связана с элементом i .
As a result, predicting an unknown preference $\hat{r}_{u,i}$ of a user u on an item i can be computed as follows:	В результате, прогнозирование неизвестного предпочтения $r_{u,i}$ пользователя u для элемента i может быть вычислено следующим образом:
$\hat{r}_{u,i} = E_u W_i$	
where E_u and W_i are the latent vectors of user u and item i .	где E_u и W_i скрытые векторы пользователя u и элемента i .
For the training task, the common approach is to minimize the following objective function:	Для задачи обучения общий подход заключается в минимизации следующей целевой функции:
$L = \sum (r_{u,i} - E_u W_i)^2 + \lambda_m (\ E_u\ ^2 + \ W_i\ ^2)$	
where $r_{u,i}$ is a known preference in the user-item rating matrix R , λ_m is	где $r_{u,i}$ известное предпочтение в матрице рейтинга элементов

the regularization parameter.	пользователя R , λm параметр регуляризации.
3.3. Big Data frameworks	3.3. Платформы больших данных
Big Data has attracted a lot of attention from both academia and industry. Several frameworks for distributed computing have been developed thanks to companies like Google and Yahoo. Specifically, Google designed the MapReduce programming model for processing large-scale data (Dean & Ghemawat, 2008). Then, Apache Hadoop (Hadoop, 2019) was developed by Yahoo. It is a popular open source implementation of the MapReduce programming paradigm. It provides a distributed file system called HDFS (Hadoop Distributed File System), to store massive data, and provide highly fault-tolerant storage (Glushkova, Jovanovic, & Abelló, 2019; White, 2012).	Большие данные привлекли большое внимание как научных кругов, так и промышленности. Несколько структур для распределенных вычислений были разработаны благодаря таким компаниям, как Google и Yahoo. В частности, Google разработала программную модель MapReduce для обработки крупномасштабных данных (Dean & Ghemawat, 2008). Затем Apache Hadoop (Hadoop, 2019) был разработан Yahoo. Это популярная открытая реализация парадигмы программирования MapReduce. Он предоставляет распределенную файловую систему, называемую HDFS (распределенная файловая система Hadoop), для хранения массивных данных и обеспечения отказоустойчивого хранилища (Глушкова, Йованович и Абелло, 2019; Белый, 2012).
However, due to the inefficiency of MapReduce for applications that share data across multiple steps, Apache Spark (Spark, 2019) has recently emerged to overcome the weaknesses of Hadoop (Ait Hammou et al., 2018; Galicia, Torres, Martínez-Álvarez, & Troncoso, 2018; Mailló, Ramírez, Triguero, & Herrera, 2017). Indeed, it can run programs up to 100x faster than Hadoop MapReduce if data fits in memory, or 10x faster on disk. The key part in Spark is an abstraction called the resilient distributed data set (RDD), which is an immutable and partitioned collection of elements that can be processed in a distributed way. RDD is designed to be fault tolerant, i.e., in case of node failure, Spark is able to reconstruct the lost RDD partitions thanks to the lineage information (Zaharia et al., 2012; Zaharia, Chowdhury, Franklin, Shenker, & Stoica, 2010).	Однако из-за неэффективности MapReduce для приложений, которые совместно используют данные на нескольких этапах, недавно появился Apache Spark (Spark, 2019) для преодоления слабых мест Hadoop (Ait Hammou et al., 2018; Галисия, Торрес, Мартинес-Альварес, & Troncoso, 2018; Mailló, Ramírez, Triguero & Herrera, 2017). Действительно, он может запускать программы в 100 раз быстрее, чем Hadoop MapReduce, если данные помещаются в память, или в 10 раз быстрее на диск. Ключевой частью в Spark является абстракция, называемая эластичным распределенным набором данных (RDD), которая представляет собой неизменный и разделенный набор элементов, которые могут обрабатываться распределенным способом. RDD спроектирован так, чтобы быть отказоустойчивым, т. Е. В случае сбоя узла Spark может восстановить потерянные разделы RDD благодаря информации о происхождении (Zaharia et al., 2012; Zaharia, Chowdhury, Franklin, Shenker & Stoica, 2010).).
4. Our proposal	4. Наше предложение
In this section, we present three distributed recommendation approaches for Big Data. We denote our proposed models as DPM, DPMI, DPMF. As stated in the introduction, several factors may impact the performance of recommendation systems such as sparsity, prediction quality, and the expensive computational time. Therefore, the objective of our solution is to address the data sparsity problem, improve the prediction quality, reduce the computational time, and handle large-scale data effectively.	В этом разделе мы представляем три подхода распределенных рекомендаций для больших данных. Мы обозначаем наши предлагаемые модели как DPM, DPMI, DPMF. Как указано во введении, на эффективность систем рекомендаций могут влиять несколько факторов, таких как разреженность, качество прогнозирования и дорогостоящее время вычислений. Поэтому целью нашего решения является решение проблемы

	разреженности данных, повышение качества прогнозирования, сокращение времени вычислений и эффективная обработка крупномасштабных данных.
<p>Fig. 1. Flowchart of the proposed model DPM.</p> 	<p>Рис. 1. Блок-схема предлагаемой модели DPM.</p>
<p>4.1. DPM: A distributed predictive model for personalized recommendations</p>	<p>4.1. DPM: распределенная модель прогнозирования для персонализированных рекомендаций</p>
<p>This section presents our proposed distributed predictive model for recommendation systems called DPM. It is designed based on Apache Spark. It is based on three steps. The first step is data partitioning, which aims to divide data into an optimal number of partitions. The second step is intended to train our model by employing a novel learning process for improving the recommendation quality while reducing the computational time. The third step is the prediction of the preferences. Fig. 1 depicts the flowchart of the model DPM.</p>	<p>В этом разделе представлена предлагаемая нами модель распределенного прогнозирования для систем рекомендаций, которая называется DPM. Он разработан на основе Apache Spark. Он основан на трех этапах. Первым шагом является разделение данных, целью которого является разделение данных на оптимальное количество разделов. Второй шаг предназначен для обучения нашей модели с использованием нового процесса обучения для улучшения качества рекомендаций при одновременном сокращении времени вычислений. Третий шаг - прогнозирование предпочтений. На рис. 1 изображена блок-схема модели DPM.</p>
<p>4.1.1. Data partitioning</p>	<p>4.1.1. Разделение данных</p>
<p>The distribution of data across nodes is of critical importance to the efficiency of the parallel and distributed computations. The principal objective of this step is to partition the training data RDD train into an optimal number of partitions, which enables the proposed model to accelerate the parallel and distributed training task.</p>	<p>Распределение данных по узлам имеет решающее значение для эффективности параллельных и распределенных вычислений. Основная цель этого шага состоит в том, чтобы разделить последовательность данных обучения RDD на оптимальное количество секций, что позволяет предлагаемой модели ускорить параллельную и распределенную задачу обучения.</p>
<p>Let N_p be the set of the possible number of partitions, and $Time(RDD_{train}, n_p)$ be a function that represents the computational time required to perform the training task according to the parameter n_p. The problem can be defined as follows:</p>	<p>Пусть N_p будет набором возможного количества разделов, а $Time(RDD_{train}, n_p)$ будет функцией, которая представляет вычислительное время, необходимое для выполнения обучающей задачи в соответствии с параметром n_p. Проблема может быть определена следующим образом:</p>
$n_p^* = \underset{n_p}{\operatorname{argmin}}(Time(RDD_{train}, n_p)), \forall n_p \in N_p$ $s.t. RDD_{train} = \left(RDD_{train}^{(1)} \cup \dots \cup RDD_{train}^{(n_p^*)} \right) \quad (1)$	

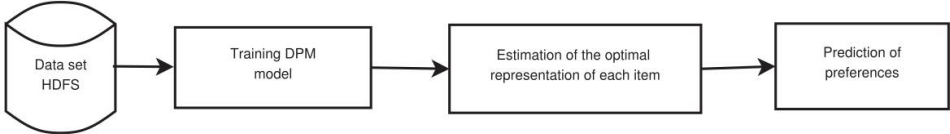
where the parameter $n * p$ is the optimal number of partitions. (1) RDD train refers to a partition. Time (RDD train $n * p$) represents the smallest computational time.	где параметр $n * p$ - оптимальное количество разбиений. (1) Поезд RDD относится к разделу. Время (RDD поезд $n * p$) представляет наименьшее время вычислений.
For example, given a training set which composed of 50 instances, and a set of two possible number of partitions $N_p = (2, 5)$. Suppose that the optimal number of partitions is $n * p = 5$, this means that the training set should be divided into 5 partitions, where each one contains 10 instances.	Например, дан обучающий набор, состоящий из 50 экземпляров, и набор из двух возможных чисел разбиения $N_p = (2, 5)$. Предположим, что оптимальное количество разделов равно $n * p = 5$, это означает, что обучающий набор должен быть разделен на 5 разделов, каждый из которых содержит 10 экземпляров.
4.1.2. Training step	4.1.2. Шаг обучения
The proposed model can be represented as a directed acyclic graph. Each node represents a function that takes input data and produces output. The output of each node is used as input for the next node, and so on. To illustrate the sequence of steps, Fig. 2 represents the overall structure of the model DPM.	Предлагаемая модель может быть представлена в виде ориентированного ациклического графа. Каждый узел представляет функцию, которая принимает входные данные и производит вывод. Выход каждого узла используется как вход для следующего узла и т. Д. Чтобы проиллюстрировать последовательность шагов, на рис. 2 представлена общая структура модели DPM.
Let p_r be the probability of expressing the rating $r \in [r_{\min}, r_{\max}]$ by the users. The set of relevant probabilities can be defined as follows:	Пусть p_r - вероятность выражения пользователями рейтинга $r \in [r_{\min}, r_{\max}]$. Набор соответствующих вероятностей может быть определен следующим образом:
$P = (p_{r(1)}, p_{r(2)}, \dots, p_{r(k)})$ $s.t. \sum_{j=1}^k p_{r(j)} = 1 - \sum_{j=k+1}^o p_{r(j)}$ $P \cap (p_{r(k+1)}, \dots, p_{r(o)}) = \emptyset \quad (2)$	
where k denotes the number of probabilities adopted to reflect the opinions of users. Note that the probability of expressing the rating r (1) as a preference is higher than the probability of expressing r (2), and so on. The symbol o represents the number of possible ratings in the system. For example, $o = 5$ when the ratings are on a scale from 1 to 5.	где k обозначает количество вероятностей, принятых для отражения мнений пользователей. Обратите внимание, что вероятность выражения рейтинга r (1) в качестве предпочтения выше, чем вероятность выражения r (2) и т. Д. Символ o представляет количество возможных оценок в системе. Например, $o = 5$, когда рейтинги находятся по шкале от 1 до 5.
As depicted in Fig. 2, the model maps each input instance $X \in R X $ to a relevant representation $A \in R A $ as follows:	Как показано на рис. 2, модель отображает каждый входной экземпляр $X \in R X $ соответствующему представлению $A \in R A $ следующим образом:

$A_u = T(X = R_u) = (r_{u,i_1}, \dots, r_{u,i_{ I }})$ $A_i = T(X = R_i) = (r_{u_1,i}, \dots, r_{u_{ U },i})$ $s.t. \quad \forall r_{u,i} \in R_u, \quad r_{u,i} \in A_u \quad \text{if } p_{r_{u,i}} \in P$ $\quad \forall r_{u,i} \in R_i, \quad r_{u,i} \in A_i \quad \text{if } p_{r_{u,i}} \in P \quad (3)$	
<p>where X represents either the set of ratings R_u expressed by a user u, or the set of ratings R_i provided for an item i. The main idea behind the function $T(X)$ is to select only each rating $r_{u,i} \in X$ for which the condition $p_{r_{u,i}} \in P$ is true.</p>	<p>где X представляет собой либо набор оценок R_u, выраженный пользователем u, либо набор оценок R_i, предоставленных для элемента i. Основная идея функции $T(X)$ состоит в том, чтобы выбрать только каждый рейтинг $r_{u,i} \in X$, для которого выполняется условие $p_{r_{u,i}} \in P$.</p>
<p>The resulting representations A_u and A_i are aggregated according to each user u, and item i as follows:</p>	<p>Результирующие представления A_u и A_i агрегируются в соответствии с каждым пользователем u и элементом i следующим образом:</p>
$V_u = S(A_u) = \frac{\sum_{r_{u,i} \in A_u} r_{u,i}}{ R_u }$ $V_i = S(A_i) = \frac{\sum_{r_{u,i} \in A_i} r_{u,i}}{ R_i } \quad (4)$	
<p>where $V_u \in R$ generalizes the rating behavior of a user u. $V_i \in R$ approximates the preferences of users about an item i. The principal goal of the Eq. (4) is to reflect optimism, neutrality, or pessimism of users. For instance, in the range $[1,5]$, $V_{u1} = 4$ means that the user $u1$ can be considered an optimistic user because $V_{u1} > 3$. Similarly, for an item $i3$, $V_{i3} = 2$ means that most users did not like the item $i3$, which corresponds to $V_{i3} < 3$.</p>	<p>где $V_u \in R$ обобщает рейтинговое поведение пользователя u. $V_i \in R$ аппроксимирует предпочтения пользователей относительно элемента i. Основная цель уравнения. (4) должен отражать оптимизм, нейтралитет или пессимизм пользователей. Например, в диапазоне $[1,5]$ $V_{u1} = 4$ означает, что пользователь $u1$ может считаться оптимистичным пользователем, поскольку $V_{u1} > 3$. Аналогично, для элемента $i3$, $V_{i3} = 2$ означает, что большинству пользователей не понравился элемент $i3$, что соответствует $V_{i3} < 3$.</p>
<p>Due to the different challenges of Big Data, and the sparse nature of the rating matrix. It is essential to adopt a more sophisticated mechanism to efficiently process a large amount of data and address the sparsity issue.</p>	<p>Из-за различных проблем больших данных и редкого характера матрицы рейтинга. Важно принять более сложный механизм для эффективной обработки большого количества данных и решения проблемы нехватки.</p>
<p>Therefore, for each R_u, the model enables the connections only from the unit of the user u (i.e., V_u), and the unit of each item rated by u (i.e., V_i), where $r_{u,i} \in R_u$ and $r_{u,i} = 0$. After that, the model decomposes the problem into a set of subproblems, where each subproblem is optimized based on a set of consecutive objective functions. Since the users in the system are independent. The model splits the problem into U</p>	<p>Следовательно, для каждого R_u модель разрешает соединения только от единицы пользователя u (т. Е. V_u) и от единицы каждого элемента, оцениваемой u (т. Е. V_i), где $r_{u,i} \in R_u$ и $r_{u,i} = 0$. После этого модель разбивает задачу на набор подзадач, где каждая подзадача оптимизируется на основе набора последовательных целевых функций. Так как пользователи в</p>

<p>optimization subproblems as follows:</p>	<p>системе независимы. Модель разбивает задачу на U Оптимизация подзадач осуществляется следующим образом:</p>
$ \begin{aligned} (\gamma_{u_1}, \beta_{u_1}, \omega_{u_1}^*) &= \text{subproblem}_{u_1}(V_{u_1}, V_{i_1}, \dots, V_{i_{ I }}, R_{u_1}) \\ (\gamma_{u_2}, \beta_{u_2}, \omega_{u_2}^*) &= \text{subproblem}_{u_2}(V_{u_2}, V_{i_1}, \dots, V_{i_{ I }}, R_{u_2}) \\ &\vdots \\ (\gamma_{u_{ U }}, \beta_{u_{ U }}, \omega_{u_{ U }}^*) &= \text{subproblem}_{u_{ U }}(V_{u_{ U }}, V_{i_1}, \dots, V_{i_{ I }}, R_{u_{ U }}) \end{aligned} \quad (5) $	
<p>where learning the parameters $(\gamma_u, \beta_u, \omega_u^*)$ is performed by solving a set of objective functions with respect to subproblem u. Each subproblem u is related to a user u. The main goal of defining independent subproblems is to carry out efficient parallel and distributed learning on large-scale data.</p>	<p>где изучение параметров $(\gamma_u, \beta_u, \omega_u^*)$ выполняется путем решения набора целевых функций относительно подзадачи u. Каждая подзадача u связана с пользователем u. Основной целью определения независимых подзадач является проведение эффективного параллельного и распределенного обучения на крупномасштабных данных.</p>
<p>Fig. 2. DPM (Training step).</p>	<p>Рис. 2. DPM (шаг обучения).</p>
<p>In general, the interaction between the units is determined based on minimizing the following objective function:</p>	<p>В общем, взаимодействие между подразделениями определяется на основе минимизации следующей целевой функции:</p>
$ z = \sum_{u \in U} \sum_{i \in I} \left(r_{u,i} - \left(\frac{V_u + V_i}{\gamma_u} \right) \right)^2 \quad (6) $	

The partial derivative of z is written as follows:	Частная производная от z записывается следующим образом:
$\frac{\partial z}{\partial \gamma_u} = 2 \sum_{r_{u,i} \in R_u} \left(r_{u,i} - \left(\frac{V_u + V_i}{\gamma_u} \right) \right) \left(\frac{V_u + V_i}{\gamma_u^2} \right) \quad (7)$	
According to $\frac{\partial z}{\partial \gamma_u} = 0$, the interaction between the enabled units for each user u is defined as follows:	В соответствии с $\frac{\partial z}{\partial \gamma_u} = 0$, взаимодействием между включенными блоками для каждого пользователя u определяется следующим образом:
$\gamma_u = Z = \frac{\sum_{r_{u,i} \in R_u} (V_u + V_i)^2}{\sum_{r_{u,i} \in R_u} (r_{u,i} (V_u + V_i))} \quad (8)$	
where the parameter $\gamma_u \in R$ represents the optimal value, which controls the interaction between units with respect to the user u .	где параметр $\gamma_u \in R$ представляет собой оптимальное значение, которое управляет взаимодействием между единицами по отношению к пользователю u.
In order to take into account the personalized behavior of each user u , with the impact of units. The parameter β_u is estimated directly by minimizing the following loss function:	Для того, чтобы принять во внимание персонализированное поведение каждого пользователя, с воздействием единиц. Параметр β_u оценивается непосредственно путем минимизации следующей функции потерь:
$g = \sum_{u \in U} \sum_{i \in I} \left(r_{u,i} - \left(\frac{2((1 - \beta_u)V_u + \beta_u V_i)}{\gamma_u} \right) \right)^2 \quad (9)$	
The partial derivative of the function g with respect to the parameter β_u is determined as follows:	Частная производная функции g по параметру β_u определяется следующим образом:
$\frac{\partial g}{\partial \beta_u} = 2 \sum_{r_{u,i} \in R_u} \left(r_{u,i} - \left(\frac{2((1 - \beta_u)V_u + \beta_u V_i)}{\gamma_u} \right) \right) \left(\frac{-2(V_i - V_u)}{\gamma_u} \right) \quad (10)$	
At the minimizing point $\frac{\partial g}{\partial \beta_u} = 0$, the optimal value of $\beta_u \in R$ is computed as follows:	В точке минимизации $\frac{\partial g}{\partial \beta_u} = 0$, оптимальное значение $\beta_u \in R$ вычисляется следующим образом:
$\beta_u = G = \frac{\sum_{r_{u,i} \in R_u} \frac{2(V_i - V_u)}{\gamma_u} \left(r_{u,i} - \frac{2V_u}{\gamma_u} \right)}{\sum_{r_{u,i} \in R_u} \left(\frac{2(V_i - V_u)}{\gamma_u} \right)^2} \quad (11)$	
The main idea behind the parameter $\beta_u \in R$ is to adjust the expected preferences according to the user's behavior.	Основная идея параметра $\beta_u \in R$ состоит в том, чтобы настроить ожидаемые предпочтения в соответствии с поведением пользователя.

On the other hand, to provide the more appropriate recommendation, it is essential to update the estimated personalized behavior of each user u (i.e., V_u), based on the optimal parameters γ_u , β_u and the past preferences. Thus, the new objective is to optimize the following loss function:	С другой стороны, чтобы предоставить более подходящую рекомендацию, важно обновить оценочное персонализированное поведение каждого пользователя u (то есть, V_u) на основе оптимальных параметров γ_u , β_u и прошлых предпочтений. Таким образом, новая цель заключается в оптимизации следующей функции потерь:
$f = \sum_{u \in U} \sum_{i \in I} \left(r_{u,i} - \left(\frac{2((1 - \beta_u)(V_u + \omega_u) + \beta_u V_i)}{\gamma_u} \right) \right)^2 \quad (12)$	
The partial derivative of the function f with respect to the parameter ω_u is defined as follows:	Частная производная функции f по параметру ω_u определяется следующим образом:
$\frac{\partial f}{\partial \omega_u} = 2 \sum_{r_{u,i} \in R_u} \left(r_{u,i} - \left(\frac{2((1 - \beta_u)(V_u + \omega_u) + \beta_u V_i)}{\gamma_u} \right) \right) \left(\frac{-2(1 - \beta_u)}{\gamma_u} \right) \quad (13)$	
According to $\partial \omega = 0$, each parameter w_u is computed as follows:	Согласно $\partial \omega = 0$ каждый параметр w_u вычисляется следующим образом:
$\omega_u = F = \frac{\sum_{r_{u,i} \in R_u} \frac{2(1 - \beta_u)}{\gamma_u} \left(r_{u,i} - \frac{2(\beta_u(V_i - V_u) + V_u)}{\gamma_u} \right)}{\sum_{r_{u,i} \in R_u} \left(\frac{2(1 - \beta_u)}{\gamma_u} \right)^2} \quad (14)$	
Once the initial parameter $\omega_u \in R$ is calculated according to the user u , the next goal is to learn the optimal value $\omega_u^* \in R$, which is defined as follows:	Как только начальный параметр $\omega_u \in R$ рассчитывается в соответствии с пользователем u , следующая цель состоит в том, чтобы узнать оптимальное значение $\omega_u^* \in R$, которое определяется следующим образом:
$\omega_u^* = \omega_u + B(x) \quad (15)$	
The key consideration of ω_u^* is to determine the optimal representation of the personalized behavior of the user u . The term $B(x) \in R$ represents the relevant value to improve the estimated behavior.	Ключевым соображением ω_u^* является определение оптимального представления персонализированного поведения пользователя u . Член $B(x) \in R$ представляет соответствующее значение для улучшения оценочного поведения.
In general, finding the parameter ω_u^* based on $B(x)$ involves considering multiple objective functions, which are described as follows:	В общем, нахождение параметра ω_u^* на основе $B(x)$ включает рассмотрение нескольких целевых функций, которые описываются следующим образом:

$y(x) = [y^{(1)}(x), y^{(2)}(x)]$ $y^{(1)}(x) = \frac{\sum_{r_{u,i} \in R_u} \left r_{u,i} - \frac{2((1-\beta_u)(V_u + \omega_u^*) + \beta_u V_i)}{\gamma_u} \right }{ R_u }$ $y^{(2)}(x) = \sqrt{\frac{\sum_{r_{u,i} \in R_u} \left(r_{u,i} - \frac{2((1-\beta_u)(V_u + \omega_u^*) + \beta_u V_i)}{\gamma_u} \right)^2}{ R_u }} \quad (16)$	
<p>Fig. 3. Flowchart of the proposed model DPMI.</p>  <pre> graph LR A[(Data set HDFS)] --> B[Training DPM model] B --> C[Estimation of the optimal representation of each item] C --> D[Prediction of preferences] </pre>	<p>Рис. 3. Блок-схема предлагаемой модели DPMI.</p>
<p>where $(y(i)(x), i = 1, 2)$ refer to the two objective functions.</p> <p>Therefore, the estimation process is performed by considering the trade-off between the objectives. Mathematically, this task is formulated as follows:</p>	<p>где $(y(i)(x), i = 1, 2)$ относится к двум целевым функциям.</p> <p>Поэтому процесс оценки выполняется с учетом компромисса между целями. Математически эта задача формулируется следующим образом:</p>
$B(x) = \operatorname{argmin}_x \left(\frac{y^{(1)}(x) - \min(Y^{(1)})}{\max(Y^{(1)})} + \frac{y^{(2)}(x) - \min(Y^{(2)})}{\max(Y^{(2)})} \right), \forall x \in O$ $Y^{(1)} = (y^{(1)}(x), x \in O)$ $Y^{(2)} = (y^{(2)}(x), x \in O) \quad (17)$	
<p>where $(y(i)(x), i = 1, 2)$ refer to the two objective functions.</p> <p>Therefore, the estimation process is performed by considering the trade-off between the objectives. Mathematically, this task is formulated as follows:</p>	<p>где $(y(i)(x), i = 1, 2)$ относится к двум целевым функциям.</p> <p>Поэтому процесс оценки выполняется с учетом компромисса между целями. Математически эта задача формулируется следующим образом:</p>
$B(x) = \operatorname{argmin}_x \left(\frac{y^{(1)}(x) - \min(Y^{(1)})}{\max(Y^{(1)})} + \frac{y^{(2)}(x) - \min(Y^{(2)})}{\max(Y^{(2)})} \right), \forall x \in O$ $Y^{(1)} = (y^{(1)}(x), x \in O)$ $Y^{(2)} = (y^{(2)}(x), x \in O) \quad (17)$	
<p>where O denotes the set of possible values to estimate $B(x)$. $\min(Y^{(1)})$ and $\max(Y^{(1)})$ represent the minimum and maximum value of $Y^{(1)}$. Based on the Eqs. (1)–(17), the training step of the model DPM is shown in Algorithm 1.</p>	<p>где O обозначает множество возможных значений для оценки $B(x)$. $\min(Y^{(1)})$ и $\max(Y^{(1)})$ представляют минимальное и максимальное значение $Y^{(1)}$. На основании формул. (1) - (17), шаг обучения модели DPM показан в алгоритме 1.</p>

Algorithm 1 : DPM (Training step).

Input: RDD_{train} : Training data in HDFS, n_p^* : Optimal number of partitions, k : Dimension of the behavior

Output: RDD_{DPM} : RDD of the estimated parameters

- 1: Partition RDD_{train} into n_p^* partitions $RDD_{train} = (RDD_{train}^{(1)} \cup \dots \cup RDD_{train}^{(n_p^*)})$;
- 2: Identify the set of relevant probabilities $P = (p_{r(1)}, p_{r(2)}, \dots, p_{r(k)})$ using Eq. (2);
- 3: Combine the set of preferences R_u provided by each user $u \in U$;
- 4: Combine the set of users' opinions R_i provided for each item $i \in I$;
- 5: **for** each user $u \in U$ **do**
- 6: Represent R_u as A_u based on P by Eq. (3);
- 7: Approximate the rating behavior V_u of the user u by Eq. (4);
- 8: **end for**
- 9: **for** each item $i \in I$ **do**
- 10: Represent R_i as A_i based on P by Eq. (3);
- 11: Approximate the users' opinions V_i about the item i by Eq. (4);
- 12: **end for**
- 13: **for** each $subproblem_u$ **do**
- 14: Compute γ_u based on $\frac{\partial z}{\partial \gamma_u} = 0$ using Eq. (8);
- 15: Compute β_u based on $\frac{\partial g}{\partial \beta_u} = 0$ using Eq. (11);
- 16: Compute ω_u based on $\frac{\partial f}{\partial \omega_u} = 0$ using Eq. (14);
- 17: Compute the optimal value ω_u^* using Eq. (15);
- 18: **end for**
- 19: $RDD_{DPM} < -((u_1, (\gamma_{u_1}, \beta_{u_1}, \omega_{u_1}^*)), \dots, (u_M, (\gamma_{u_M}, \beta_{u_M}, \omega_{u_M}^*)))$;
- 20: **return** RDD_{DPM} ;

4.1.3. Prediction step

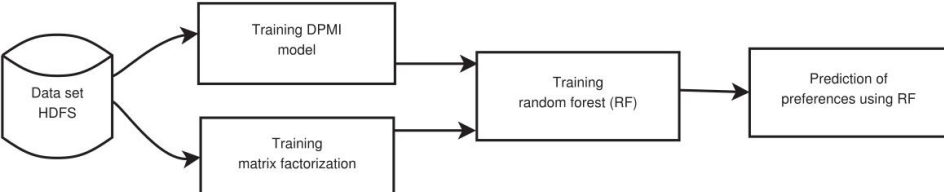
Given the estimated parameters the Model DPM, the predicted rating of a user u , for an item i is computed as follows:

$$\hat{r}_{u,i} = \frac{2((1 - \beta_u)(V_u + \omega_u^*) + \beta_u V_i)}{\gamma_u} \quad (18)$$

4.1.3. Шаг прогнозирования

Учитывая оценочные параметры модели DPM, прогнозируемый рейтинг пользователя u для элемента i вычисляется следующим образом:

Here β_u , γ_u and ω_u^* are the parameters of the trained model. In general, the main assumption behind the Eq. (18) is that, predicting an unknown preference $\hat{r}_{u,i}$ of a user u involves taking into account the optimal representation of the personalized behavior of the user u (i.e., $(V_u + \omega_u^*)$), and the opinions of most users on the item i denoted as V_i .	Здесь β_u , γ_u и ω_u^* - параметры обучаемой модели. В общем, главное предположение за формулой. (18) заключается в том, что для прогнозирования неизвестного предпочтения пользователя u , i , необходимо учитывать оптимальное представление персонализированного поведения пользователя u (т. Е. $(V_u + \omega_u^*)$) и мнения большинство пользователей на элемент, который я обозначил как V_i .
4.2. DPMI: A distributed predictive model based on adjusting the representation of items	4.2. DPMI: модель распределенного прогнозирования, основанная на корректировке представления элементов
This section presents our second model called DPMI, which is an improved variant of the model described in Section 4.1. It is tailored to take advantage of the optimal parameters of DPM with adjusting the estimated representation of users' opinions for each item. Fig. 3 illustrates the flowchart of the model DPMI.	В этом разделе представлена наша вторая модель под названием DPMI, которая является улучшенным вариантом модели, описанной в разделе 4.1. Он приспособлен для использования оптимальных параметров DPM с корректировкой предполагаемого представления мнений пользователей по каждому пункту. Рис. 3 иллюстрирует блок-схему модели DPMI.
4.2.1. Training step	4.2.1. Шаг обучения
To produce more accurate predictions, DPMI assumes that there exists a relevant representation that can reflect the opinions of users for each item. Therefore, it is necessary to take into account the optimal parameters of the model DPM and analyze the opinions of users to update the estimated representation (i.e., V_i) with respect to each item i .	Для получения более точных прогнозов DPMI предполагает наличие соответствующего представления, которое может отражать мнения пользователей по каждому элементу. Следовательно, необходимо учитывать оптимальные параметры модели DPM и анализировать мнения пользователей для обновления оценочного представления (т.е. V_i) по каждому элементу i .
As there are a large number of opinions expressed for each item, the model DPMI decomposes the problem into $ I $ subproblems, where each subproblem i corresponds to an item $i \in I$. Then, it solves each subproblem independently based on the estimated parameters $((\gamma_u, \beta_u, \omega_u^*), u \in U)$.	Поскольку по каждому пункту высказано большое количество мнений, модель DPMI разбивает проблему на $ I $ подзадачи, где каждой подзадаче i соответствует элемент i . Затем он решает каждую подзадачу независимо на основе оценочных параметров $((\gamma_u, \beta_u, \omega_u^*), u \in U)$.
In particular, the goal is to optimize the following objective function:	В частности, целью является оптимизация следующей целевой функции:
$h = \sum_{u \in U} \sum_{i \in I} \left(r_{u,i} - \left(\frac{2((1 - \beta_u)(V_u + \omega_u^*) + \beta_u(V_i + \lambda_i))}{\gamma_u} \right) \right)^2$ <div style="text-align: right;">(19)</div>	
The partial derivative of h with respect to the parameter λ_i is defined as follows:	Частная производная от h по параметру λ_i определяется следующим образом:

$\frac{\partial h}{\partial \lambda_i} = 2 \sum_{r_{u,i} \in R_i} \left(r_{u,i} - \left(\frac{2((1 - \beta_u)(V_u + \omega_u^*) + \beta_u(V_i + \lambda_i))}{\gamma_u} \right) \right) \left(\frac{-2\beta_u}{\gamma_u} \right)$ <p style="text-align: right;">(20)</p>	
<p>From Eq. (20) , the parameter $\lambda_i \in R$ is computed as follows:</p>	<p>Из уравнения (20) параметр $\lambda_i \in R$ вычисляется следующим образом:</p>
$\lambda_i = H = \frac{\sum_{r_{u,i} \in R_u} \frac{2\beta_u}{\gamma_u} \left(r_{u,i} - \frac{2((1 - \beta_u)(V_u + \omega_u^*) + \beta_u V_i)}{\gamma_u} \right)}{\sum_{r_{u,i} \in R_u} \left(\frac{2\beta_u}{\gamma_u} \right)^2}$ <p style="text-align: right;">(21)</p>	
<p>Fig. 4. Flowchart of the proposed model DPMF.</p>  <pre> graph LR A[(Data set HDFS)] --> B[Training DPMI model] A --> C[Training matrix factorization] B --> D[Training random forest RF] C --> D D --> E[Prediction of preferences using RF] </pre>	
<p>For each item i , the optimal parameter $\lambda^* i \in R$ is defined as follows:</p>	<p>Для каждого элемента i оптимальный параметр $\lambda^* i \in R$ определяется следующим образом:</p>
$\lambda_i^* = \lambda_i + D(x)$ <p style="text-align: right;">(22)</p>	
<p>The term $D(x) \in R$ denotes the estimated value, which is devoted to improving the estimation of users' opinions. The intuition behind $\lambda^* i$ is to determine the optimal representation of the opinions of users for an item i .</p>	<p>Термин $D(x) \in R$ обозначает оценочное значение, которое предназначено для улучшения оценки мнений пользователей. Интуиция, лежащая в основе $\lambda^* i$, заключается в определении оптимального представления мнений пользователей для элемента i.</p>
<p>In order to find the most appropriate value $D(x)$, DPMI employs the following objective functions:</p>	<p>Чтобы найти наиболее подходящее значение $D(x)$, DPMI использует следующие целевые функции:</p>
$q(x) = [q^{(1)}(x), q^{(2)}(x)]$ $q^{(1)}(x) = \frac{\sum_{r_{u,i} \in R_i} \left r_{u,i} - \frac{2((1 - \beta_u)(V_u + \omega_u^*) + \beta_u(V_i + \lambda_i^*))}{\gamma_u} \right }{ R_i }$ $q^{(2)}(x) = \sqrt{\frac{\sum_{r_{u,i} \in R_i} \left(r_{u,i} - \frac{2((1 - \beta_u)(V_u + \omega_u^*) + \beta_u(V_i + \lambda_i^*))}{\gamma_u} \right)^2}{ R_i }}$ <p style="text-align: right;">(23)</p>	
<p>where $(q(i)(x), i = 1, 2)$ are the 2-objective functions.</p>	<p>где $(q(i)(x), i = 1, 2)$ - 2-целевые функции.</p>
<p>By taking into account these two objectives, the solution that satisfies the</p>	<p>Принимая во внимание эти две цели, решение, которое</p>

best trade-off is formulated as follows:	удовлетворяет наилучшему компромиссу, формулируется следующим образом:
$D(x) = \operatorname{argmin}_x \left(\frac{q^{(1)}(x) - \min(Q^{(1)})}{\max(Q^{(1)})} + \frac{q^{(2)}(x) - \min(Q^{(2)})}{\max(Q^{(2)})} \right), \forall x \in O$ $Q^{(1)} = (q^{(1)}(x) : x \in O)$ $Q^{(2)} = (q^{(2)}(x) : x \in O) \quad (24)$	
where O represents the set of possible values to estimate D (x). min (Q (1)) and max (Q (1)) are the minimum, maximum value of Q (1) , respectively.	где O представляет собой набор возможных значений для оценки D (x). min (Q (1)) и max (Q (1)) являются минимальным, максимальным значением Q (1), соответственно.
Based on the model DPM and Eqs. (19)–(24) , the training step is shown in Algorithm 2 .	На основе модели DPM и формул. (19) - (24), шаг обучения показан в алгоритме 2.
Algorithm 2 : DPML (Training step). Input: RDD_{train} : Training data in HDFS, n_p^* : Optimal number of partitions, k : Dimension of the behavior Output: RDD_{DPM} , RDD_{DPML} : RDDs of the estimated parameters 1: $RDD_{DPM} < -DPM(RDD_{train}, n_p^*, k)$ 2: for each $subproblem_i$ do 3: Compute λ_i based on $\frac{\partial h}{\partial \lambda_i} = 0$ using Eq. (21); 4: Compute the optimal value λ_i^* using Eq. (22); 5: end for 6: $RDD_{DPML} < -((i_1, \lambda_{i_1}^*), \dots, (i_N, \lambda_{i_N}^*))$; 7: return RDD_{DPM} , RDD_{DPML} ;	
4.2.2. Prediction step	4.2.2. Шаг прогнозирования
Given the estimated parameters of the model DPML, the predicted rating of user u , for an item i is defined as follows:	Учитывая оценочные параметры модели DPML, прогнозируемый рейтинг пользователя u для элемента i определяется следующим образом:
$\hat{r}_{u,i} = \frac{2((1 - \beta_u)(V_u + \omega_u^*) + \beta_u(V_i + \lambda_i^*))}{\gamma_u} \quad (25)$	
where β_u , γ_u , ω_u^* and λ_i^* are the parameters of the trained model. The underlying rationale behind Eq. (25) is the assumption that predicting a rating $\hat{r}_{u,i}$ can be performed by considering the optimal representation of rating behavior for a user u , with the relevant representation of users' opinions for an item i .	где β_u , γ_u , ω_u^* и λ_i^* - параметры обученной модели. Основное обоснование уравнения. (25) является допущением, что прогнозирование рейтинга $\hat{r}_{u,i}$ может быть выполнено путем рассмотрения оптимального представления поведения рейтинга для пользователя u с соответствующим представлением мнений пользователей для элемента i.

4.3. DPMF: A distributed predictive model with matrix factorization and random forest	4,3. DPMF: распределенная прогнозирующая модель с матричной факторизацией и случайным лесом
This section presents our third approach called DPMF, which is designed to take advantage of the model DPMI presented in Section 4.2 , with Matrix factorization and Random forest models for improving the recommendation quality. The basic idea is to represent each known rating $r_{u,i} > 0$ in the training set as features and label. Then, solving the rating prediction task as a regression problem. Fig. 4 shows the flowchart of the model DPMF.	В этом разделе представлен наш третий подход, названный DPMF, который предназначен для использования преимуществ модели DPMI, представленной в разделе 4.2, с матричной факторизацией и моделями случайных лесов для повышения качества рекомендаций. Основная идея состоит в том, чтобы представить каждый известный рейтинг $r_{u,i} > 0$ в обучающем наборе в виде признаков и метки. Затем решение задачи прогнозирования рейтинга как задачи регрессии. На рис. 4 приведена блок-схема модели DPMF.
Let $C = \{ (x_j, y_j), j = 1, \dots, C \}$ be the training data set, which is composed of $ C $ instances. $ C $ is the number of non-zero rating the user-item rating R . Each $x_j \in R^{\beta+1}$ and $y_j \in R$ denote the features of an instance j (i.e., the generated representation) and the label (i.e., the ground-truth rating), respectively.	Пусть $C = \{(x_j, y_j), j = 1, \dots, C \}$ будет набор данных обучения, который состоит из $ C $ экземпляров. $ C $ это число ненулевой оценки пользовательского рейтинга R . Каждое $x_j \in R^{\beta+1}$ и $y_j \in R$ обозначают признаки экземпляра j (то есть сгенерированного представления) и метки (то есть рейтинг истинности основания) соответственно.
Let β be the number of matrix factorization models, the main objective is to train β matrix factorization models with our second approach DPMI, then employ the learned models to generate a representation for each preference $r_{u,i}$ in the training set as follows:	Пусть β будет количеством моделей матричной факторизации, основная цель - обучить модели матричной факторизации β с помощью нашего второго подхода DPMI, а затем использовать изученные модели для генерации представления для каждого предпочтения $r_{u,i}$ в обучающем наборе следующим образом:
$L(r_{u,i}) = (x_j, y_j)$ $x_j = (E_u^{(1)}W_i^{(1)}, \dots, E_u^{(\beta)}W_i^{(\beta)}, \hat{r}_{u,i})$ $y_j = r_{u,i}$ (26)	
where the function $L(.)$ is devoted to represent the ratings using the learned models. $x_j \in R^{\beta+1}$ denotes the features, y_j is the label, and $\hat{r}_{u,i}$ refers to the rating predicted using Eq. (25). $(E_u^{(1)}, W_i^{(1)})$ and $(E_u^{(2)}, W_i^{(2)})$ represent the latent factors estimated using the first and the second Matrix factorization models, respectively. The underlying assumption behind Eq. (26) is that generating a representation that characterizes each preference, and exploiting these representations by taking advantage of a machine learning algorithm may lead to better results.	где функция $L(.)$ предназначена для представления оценок с использованием изученных моделей. $x_j \in R^{\beta+1}$ обозначает характеристики, y_j обозначает метку, а $\hat{r}_{u,i}$ обозначает рейтинг, прогнозируемый по формуле. (25). $(E_u^{(1)}, W_i^{(1)})$ и $(E_u^{(2)}, W_i^{(2)})$ представляют скрытые факторы, оцененные с использованием первой и второй матричных моделей факторизации, соответственно. Основное предположение за формулой. (26) заключается в том, что генерация представления, которое характеризует каждое предпочтение, и использование этих представлений с использованием алгоритма машинного обучения может привести к лучшим результатам.
Next, the rating prediction task is considered as a regression problem. It	Далее задача прогнозирования рейтинга рассматривается как

can be solved by training a machine learning model using the generated representations with the pre-defined labels.	проблема регрессии. Это может быть решено путем обучения модели машинного обучения с использованием сгенерированных представлений с предопределенными метками.
Random forest (RF) is one of the most powerful machine learning (ML) algorithms. It is a supervised learning method, which is widely used for both regression and classification problems. Generally, the random forest is typically an ensemble of decision trees, where the training process is based on the bagging method (Breiman, 2001; Chen et al., 2016; Genuer, Poggi, TuleauMalot, & Villa-Vialaneix, 2017; Wyner, Olson, Bleich, & Mease, 2017). Because of its advantages over other existing machine learning approaches, we have adopted random forest to perform the rating prediction task. Particularly, after training the random forest, the learned model is adopted to predict the unknown preferences.	Случайный лес (RF) - один из самых мощных алгоритмов машинного обучения (ML). Это контролируемый метод обучения, который широко используется для решения задач регрессии и классификации. Как правило, случайный лес, как правило, представляет собой совокупность деревьев решений, где процесс обучения основан на методе создания пакетов (Breiman, 2001; Chen et al., 2016; Genuer, Poggi, TuleauMalot, & Villa-Vialaneix, 2017; Wyner, Olson, Bleich & Mease, 2017). Из-за его преимуществ по сравнению с другими существующими подходами машинного обучения мы приняли случайный лес для выполнения задачи прогнозирования рейтинга. В частности, после обучения случайного леса, изученная модель используется для прогнозирования неизвестных предпочтений.
5. Experiments	5. Эксперименты
In this section, we will evaluate the effectiveness of our proposed methods using three real-world data sets. Section 5.1 presents the data sets. Section 5.2 describes the experimental setup. Section 5.3 presents the evaluation metrics. Section 5.4 summarizes the comparison methods. Section 5.5 details the parameter setting. Section 5.6 presents the implementation detail. Finally, Section 5.7 details the experimental results.	В этом разделе мы оценим эффективность предложенных нами методов с использованием трех реальных наборов данных. В разделе 5.1 представлены наборы данных. Раздел 5.2 описывает экспериментальную установку. В разделе 5.3 представлены показатели оценки. Раздел 5.4 обобщает методы сравнения. Раздел 5.5 подробно описывает настройку параметров. Раздел 5.6 представляет детали реализации. Наконец, раздел 5.7 детализирует экспериментальные результаты.
5.1. Data sets	5.1. Наборы данных
In this paper, the experiments are conducted using three realworld data sets, namely Movielens 10M, Movielens 20M (Harper & Konstan, 2016), and Yelp (2019) .	В этой статье эксперименты проводятся с использованием трех наборов данных реального мира, а именно Movielens 10M, Movielens 20M (Harper & Konstan, 2016) и Yelp (2019).
MovieLens 10M 1 is a real-world data set, which was collected by GroupLens Research at the University of Minnesota. It consists of 10,0 0 0,054 ratings, given by 71,567 users for 10,681 movies of the online movie recommender service. Each user has rated at least 20 movies using a discrete score on the scale [0.5,5].	MovieLens 10M 1 - это набор данных реального мира, который был собран исследовательской группой GroupLens в Университете Миннесоты. Он состоит из 10,0 0 0,054 оценок, предоставленных 71 567 пользователями для 10 681 фильма службы онлайн-рекомендации фильмов. Каждый пользователь оценил не менее 20 фильмов с использованием дискретной оценки по шкале [0,5,5].
MovieLens 20M 2 is another benchmark data set published by GroupLens research. It contains 20,0 0 0,263 ratings of approximately 27,278 movies	MovieLens 20M 2 - это еще один набор данных, опубликованный в исследовании GroupLens. Он содержит 20,0 0 0,263 оценок из

made by 138,493 users between January 09, 1995 and March 31, 2015. Each user has rated at least 20 movies. The ratings are given on a 5-star scale with half-star increments.	примерно 27 278 фильмов, сделанных 138 493 пользователями в период с 9 января 1995 года по 31 марта 2015 года. Каждый пользователь оценил не менее 20 фильмов. Рейтинги приведены по 5-звездочной шкале с шагом в пол-звезды.
Yelp 3 is one of the most popular user review sites in the United States, a local directory service with social networking features. Customers rate the businesses, share their personal experiences and comments. Yelp data set is composed of 6,685,900 interactions expressed by 1,637,138 users for 192,609 businesses in 10 metropolitan areas. As the original data is highly sparse, we further process the data set by retaining the items and users with at least 30 preferences. The final data set contains approximately 15,139 users, 27,307 items and 1,210,783 preferences.	Yelp 3 - один из самых популярных сайтов для обзора пользователей в Соединенных Штатах, локальная служба каталогов с функциями социальных сетей. Клиенты оценивают бизнес, делятся своим личным опытом и комментариями. Набор данных Yelp состоит из 6 685 900 взаимодействий, выраженных 1 637 138 пользователями, для 192 609 предприятий в 10 мегаполисах. Поскольку исходные данные крайне скудны, мы дополнительно обрабатываем набор данных, сохраняя элементы и пользователей как минимум с 30 предпочтениями. Окончательный набор данных содержит приблизительно 15 139 пользователей, 27 307 элементов и 1 210 783 предпочтений.
5.2. Experimental setup	5.2. Экспериментальная установка
For the experimental study, we split each data set according to the methodologies adopted in existing research works (Ait Hammou et al., 2018; Frémal & Lecron, 2017; Hernando et al., 2016; Zheng, Haihong, Song, & Song, 2016).	Для экспериментального исследования мы разделили каждый набор данных в соответствии с методологиями, принятыми в существующих исследовательских работах (Ait Hammou et al., 2018; Frémal & Lecron, 2017; Hernando et al., 2016; Zheng, Haihong, Song & Song, 2016).
Methodology 1: the data set is randomly partitioned into two parts, 80% of ratings are used as the training set, and the remaining 20% as the test set. This procedure is repeated 20 times. Then, the quality metrics of all the tests are averaged.	Методология 1: набор данных случайным образом разбивается на две части, 80% оценок используются в качестве обучающего набора, а оставшиеся 20% - в качестве тестового набора. Эта процедура повторяется 20 раз. Затем показатели качества всех тестов усредняются.
Methodology 2: the data set is randomly divided into two parts, 80% of the data are used as the training set and the rest as the test set.	Методология 2: набор данных случайным образом делится на две части, 80% данных используются в качестве обучающего набора, а остальные - в качестве тестового набора.
Methodology 3: the data set is randomly split into 10 folds. In each run, 9 folds are utilized as the training set and the remainder as the test set. The results of the 10 runs are then averaged.	Методология 3: набор данных случайным образом разбивается на 10 раз. В каждом прогоне 9 сгибов используются в качестве тренировочного набора, а остальные - в качестве тестового набора. Результаты 10 прогонов затем усредняются.
5.3. Evaluation metrics	5.3. Метрики оценки
To evaluate the performance of recommendation algorithms, we employed two well-known metrics, namely Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). They are widely used to measure the prediction accuracy (Ait Hammou, Ait Lahcen, & Mouline, 2019a; Al-Hassan, Lu, & Lu, 2015; Ar & Bostanci, 2016; Chen, Wang,	Чтобы оценить эффективность алгоритмов рекомендации, мы использовали две хорошо известные метрики, а именно среднеквадратическую ошибку (RMSE) и среднюю абсолютную ошибку (MAE). Они широко используются для измерения точности прогноза (Ait Hammou, Ait Lahcen, & Mouline, 2019a;

Yan et al., 2018b; Da Costa, Manzato, & Campello, 2018; Hu et al., 2014; Li, Chen, Chen, & Tong, 2017; Ma et al., 2016; Mao et al., 2017; Parvin, Moradi, & Esmaeili, 2019; Wu, Chang, & Liu, 2014; Zhang, Wu, Lu, Liu, & Zhang, 2017b; Zhang et al., 2018; Zhao, Qian, & Xie, 2016). RMSE and MAE are defined as follows:	Al-Hassan, Lu, & Lu, 2015; Ar & Bostanci, 2016; Chen, Wang, Yan et al., 2018b; Da Costa, Manzato, & Campello, 2018; Hu et al., 2014; Li, Chen, Chen, & Tong, 2017; Ma et al., 2016; Mao et al., 2017; Parvin, Moradi & Esmaeili, 2019; Wu, Chang, & Liu, 2014; Zhang, Wu, Lu, Liu, & Zhang, 2017b; Zhang et al., 2018; Zhao, Qian, & Xie, 2016). RMSE и MAE определены следующим образом:
$RMSE = \sqrt{\frac{\sum_{u \in U, i \in I} (r_{u,i} - \hat{r}_{u,i})^2}{n}} \quad (27)$	
$MAE = \frac{\sum_{u \in U, i \in I} r_{u,i} - \hat{r}_{u,i} }{n} \quad (28)$	
where n is the number of ratings in the test set, $r_{u,i}$ refers to the ground-truth rating assigned by the user u to the item i , and $\hat{r}_{u,i}$ represents the predicted rating. Note that, a smaller error value indicates a better recommendation accuracy.	где n - это число оценок в наборе тестов, $r_{u,i}$ относится к оценке истинности земли, назначенной пользователем u для элемента i, а $\hat{r}_{u,i}$ представляет прогнозируемый рейтинг. Обратите внимание, что меньшее значение ошибки указывает на лучшую точность рекомендации.
5.4. Comparison methods	5.4. Методы сравнения
We compared our proposal with the following state-of-the-art recommendation methods:	Мы сравнили наше предложение со следующими современными методами рекомендаций:
APRA (Ait Hammou et al., 2018): it is a parallel approximate recommendation method, which adopts a random sampling technique and a special learning process to accelerate the training task and handle large-scale data.	APRA (Ait Hammou et al., 2018): это метод параллельных приближенных рекомендаций, который использует метод случайной выборки и специальный процесс обучения для ускорения учебной задачи и обработки крупномасштабных данных.
ITAN (Kupisz & Unold, 2015): Item-based collaborative filtering is a state-of-the-art recommendation method, which measures the similarities between items using the Tanimoto coefficient. Then, it estimates the preferences based on the identified neighbors.	ITAN (Kupisz & Unold, 2015): основанная на элементах совместная фильтрация - это современный метод рекомендаций, который измеряет сходства между элементами с использованием коэффициента Танимото. Затем он оценивает предпочтения на основе идентифицированных соседей.
UPCC (Zhang et al., 2016): User-based collaborative filtering method calculates the similarities between users using the Pearson correlation coefficient. Then, the most similar users are employed to predict the preferences.	UPCC (Zhang et al., 2016): основанный на пользователях метод совместной фильтрации вычисляет сходства между пользователями, используя коэффициент корреляции Пирсона. Затем наиболее похожие пользователи используются для прогнозирования предпочтений.
FRAIPA (Ait Hammou & Ait Lahcen, 2017): It is a recommendation approach, which considers the rating behavior of each user and the	FRAIPA (Ait Hammou & Ait Lahcen, 2017): это рекомендательный подход, который рассматривает поведение

opinions of users on each item as a set of probabilities. Then, it employs the estimated probabilities to predict the preferences.	рейтинга каждого пользователя и мнения пользователей по каждому элементу как набор вероятностей. Затем он использует оценочные вероятности для прогнозирования предпочтений.
CESP (Hwang et al., 2016): This method computes the similarities between users, by considering the concept of category experts instead of all the users. It predicts the preferences by employing the similarity between each user and the category experts, with the category interest of the user.	CESP (Hwang et al., 2016): этот метод вычисляет сходства между пользователями, рассматривая концепцию экспертов по категориям вместо всех пользователей. Он прогнозирует предпочтения, используя сходство между каждым пользователем и экспертами категории с интересом категории пользователя.
CMPTF (Zheng et al., 2016): This is a contextual modeling method based on probabilistic tensor factorization, which exploits the social relationships, ratings, item contents and contextual information to generate recommendations.	CMPTF (Zheng et al., 2016): это метод контекстного моделирования, основанный на вероятностной тензорной факторизации, который использует социальные отношения, рейтинги, содержимое элемента и контекстную информацию для выработки рекомендаций.
NMF (Lee & Seung, 1999): It is a dimensionality reduction approach, which decomposes the rating matrix into two low-rank non-negative matrix factors. Then, it predicts the preferences using the latent factors associated with the users and items.	NMF (Lee & Seung, 1999): это подход к уменьшению размерности, который разбивает матрицу оценок на два неотрицательных матричных фактора низкого ранга. Затем он прогнозирует предпочтения, используя скрытые факторы, связанные с пользователями и предметами.
BMF (Hernando et al., 2016): It is a non negative matrix factorization based on a Bayesian probabilistic model for personalized recommendations.	BMF (Hernando et al., 2016): это неотрицательная матричная факторизация, основанная на байесовской вероятностной модели для персонализированных рекомендаций.
MLR, CM II, SW I (Frémal & Lecron, 2017): These three strategies proposed by Frémal and Lecron (2017) are called Multiple Linear Regression, Confusion matrices II, Smart Weights I, respectively. The main idea is to combine content-based clustering with collaborative filtering. Specifically, after performing the clustering task according to item genre, a weighting strategy is used to predict the preferences.	MLR, CM II, SW I (Frémal & Lecron, 2017). Эти три стратегии, предложенные Frémal и Lecron (2017), называются множественной линейной регрессией, матрицами смешения II, умными весами I соответственно. Основная идея состоит в том, чтобы объединить кластеризацию на основе контента с совместной фильтрацией. В частности, после выполнения задачи кластеризации в соответствии с жанром элемента для прогнозирования предпочтений используется весовая стратегия.
PLSA-CF (Hofmann, 2004): This is a model-based collaborative filtering approach, which is a generalization of the statistical approach called probabilistic Latent Semantic Analysis (pLSA).	PLSA-CF (Hofmann, 2004): это основанный на модели подход совместной фильтрации, который является обобщением статистического подхода, называемого вероятностным скрытым семантическим анализом (pLSA).
KNN JMSD (Bobadilla, Serradilla, & Bernal, 2010): This method identifies the neighbors using a similarity metric called JMSD. Then, it predicts the ratings by aggregating the preferences of the neighbors.	KNN JMSD (Bobadilla, Serradilla, & Bernal, 2010): этот метод идентифицирует соседей, используя метрику сходства, называемую JMSD. Затем он прогнозирует рейтинги путем агрегирования предпочтений соседей.
DPM : This is our approach described in Section 4.1 .	DPM: это наш подход, описанный в разделе 4.1.

DPMI : This is an enhanced version of our proposed approach DPM, which adjusts the estimated representation of users' opinions for each item.	DPMI: это расширенная версия предлагаемого нами подхода DPM, который корректирует приблизительное представление мнений пользователей по каждому пункту.
DPMF : This is our proposed approach presented in Section 4.3 , which adopts the model DPMI with Matrix factorization and Random forest to perform the rating prediction task.	DPMF: это наш предлагаемый подход, представленный в разделе 4.3, в котором для выполнения задачи прогнозирования рейтинга используется модель DPMI с факторизацией матрицы и случайным лесом.
5.5. Parameter setting	5.5. Установка параметра
For each method, there are a number of parameters to tune. In this work, several experiments were conducted to find the optimal parameters. Only the parameters that yield the best results are reported.	Для каждого метода есть ряд параметров для настройки. В этой работе было проведено несколько экспериментов для нахождения оптимальных параметров. Только параметры, которые дают лучшие результаты сообщаются.
For MovieLens 10M and MovieLens 20M data sets, the number of nearest-neighbors of UPCC is set to 300. For FRAIPA, the parameter α used to predict the preferences with respect to each user lies in the interval $[1.5, 2]$. The parameter λ adopted to control the prediction error and the running time is set to 0.025. For CESP, the number of category experts is fixed at 100. For APRA, the values of the random sampling parameter ϕ used to approximate data are $\{1.0, 0.8, 0.6, 0.5, 0.4\}$. The dimension of the behavior is set to 6. The value of the parameter θ u adopted to emphasize the predicted opinions regarding the personalized behavior of users lies in the interval $[0.1, 0.2, \dots, 0.8, 0.9]$. While for our proposed methods DPM and DPMI, the number of possible partitions is set to $N_p = [8, 16, 24, 32]$. The number of relevant probabilities k is fixed at 6, and $O = [-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3]$. As the proposed approach named DPMF is based on DPMI with Matrix Factorization (MF) and Random Forest, the optimal parameters for DPMI are also used for DPMF. In addition, the parameter β which represents the number of MF models is fixed at 3. For each Matrix Factorization model, the rank is set to 10, the learning rate is fixed at 0.05, and the number of iterations for performing the training task is set to 15. For Random Forest, the number of trees is set to 5. Besides, it is important to note that, for more details about the optimal parameters of the other methods, readers could refer to the original papers.	Для наборов данных MovieLens 10M и MovieLens 20M количество ближайших соседей UPCC установлено равным 300. Для FRAIPA параметр α , используемый для прогнозирования предпочтений по отношению к каждому пользователю, находится в интервале $[1.5, 2]$. Параметр λ принят для контроля ошибки прогнозирования, а время работы установлено на 0,025. Для CESP число экспертов по категориям установлено равным 100. Для APRA значения параметра случайной выборки ϕ , используемые для аппроксимации данных, составляют $\{1,0, 0,8, 0,6, 0,5, 0,4\}$. Измерение поведения установлено на 6. Значение параметра θ u, принятого для подчеркивания предсказанных мнений относительно персонализированного поведения пользователей, лежит в интервале $[0.1, 0.2, \dots, 0.8, 0.9]$. В то время как для предложенных нами методов DPM и DPMI число возможных разделов установлено в $N_p = [8, 16, 24, 32]$. Число релевантных вероятностей k фиксировано и равно 6, а $O = [-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3]$. Поскольку предлагаемый подход с именем DPMF основан на DPMI с матричной факторизацией (MF) и случайным лесом, оптимальные параметры для DPMI также используются для DPMF. Кроме того, параметр β , который представляет количество моделей MF, установлен на 3. Для каждой модели матричной факторизации ранг установлен на 10, скорость обучения установлена на 0,05, а количество итераций для выполнения учебного задания равно для случайного леса установлено число 15. Кроме того, важно отметить, что для получения более подробной информации об оптимальных параметрах других методов читатели могут

	обратиться к оригинальным статьям.
On the other hand, the optimal parameters of the different methods adopted for MovieLens10M and MovieLens 20M data sets are also used for Yelp data set, except for APRA, the parameter of random sampling ϕ is set to 1, the dimension of the behavior is set to 3, and the value of the parameter θ_u lies in the interval $[0.1, 0.2, \dots, 0.5]$. For the proposed methods DPM and DPMI, the parameters are fixed at $k = 3$, $O = [-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3]$ and $N_p = [8, 16, 24, 32]$.	С другой стороны, оптимальные параметры различных методов, принятых для наборов данных MovieLens10M и MovieLens 20M, также используются для набора данных Yelp, за исключением APRA, параметр случайной выборки ϕ установлен на 1, размерность поведения установлена 3, а значение параметра θ_u лежит в интервале $[0.1, 0.2, \dots, 0.5]$. Для предложенных методов DPM и DPMI параметры фиксированы при $k = 3$, $O = [-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3]$ и $N_p = [8, 16, 24, 32]$.
5.6. Implementation detail	5.6. Детали реализации
The experiments were conducted on a cluster consisting of two nodes: a master node and a slave node. Each node runs Ubuntu 16.04. It is equipped with a processor 2 cores (4 threads), a clock speed of 2.93 GHz, 4 GB of RAM.	Эксперименты проводились на кластере, состоящем из двух узлов: главного узла и подчиненного узла. Каждый узел работает под управлением Ubuntu 16.04. Он оснащен процессором с 2 ядрами (4 потока), тактовой частотой 2,93 ГГц, 4 ГБ оперативной памяти.
The methods were implemented using the following frameworks: Apache Hadoop-2.7.4, Apache Spark 2.1.0, Apache Mahout 0.13.0 and Scala-2.11.8.	Методы были реализованы с использованием следующих сред: Apache Hadoop-2.7.4, Apache Spark 2.1.0, Apache Mahout 0.13.0 и Scala-2.11.8.
5.7. Experimental results	5.7. Результаты экспериментов
This section is intended to compare the performance of the proposed models described in Section 4 with the state-of-the-art recommendation methods.	Этот раздел предназначен для сравнения характеристик предлагаемых моделей, описанных в разделе 4, с современными методами рекомендаций.
For the three proposed approaches, data partitioning is an important step which aims to determine the number of partitions to accelerate the training step. Selecting the optimal number of partitions is essential for improving the performance of the proposed models. In particular, the first goal is to conduct experiments according to methodology 1, to study the impact of the number of partitions n_p on the performance in terms of computational time.	Для трех предложенных подходов разделение данных является важным этапом, целью которого является определение количества разделов для ускорения этапа обучения. Выбор оптимального количества разделов имеет важное значение для повышения производительности предлагаемых моделей. В частности, первой целью является проведение экспериментов в соответствии с методологией 1, чтобы изучить влияние числа разделов n_p на производительность с точки зрения вычислительного времени.
Fig. 5 and Fig. 6 show the variation of computational time for different values of n_p on MovieLens 10M and MovieLens 20M data sets. Particularly, the x-axis presents the number of partitions n_p tested $N_p = (8, 16, 24, 32)$, and the y-axis depicts the computational time in seconds.	На рис. 5 и рис. 6 показано изменение вычислительного времени для разных значений n_p для наборов данных MovieLens 10M и MovieLens 20M. В частности, ось X представляет количество проверенных разделов n_p $N_p = (8, 16, 24, 32)$, а ось Y показывает время вычисления в секундах.
Fig. 5. The impact of the number of partitions n_p on the runtime (DPM).	Рис. 5. Влияние количества разделов n_p на время выполнения (DPM).

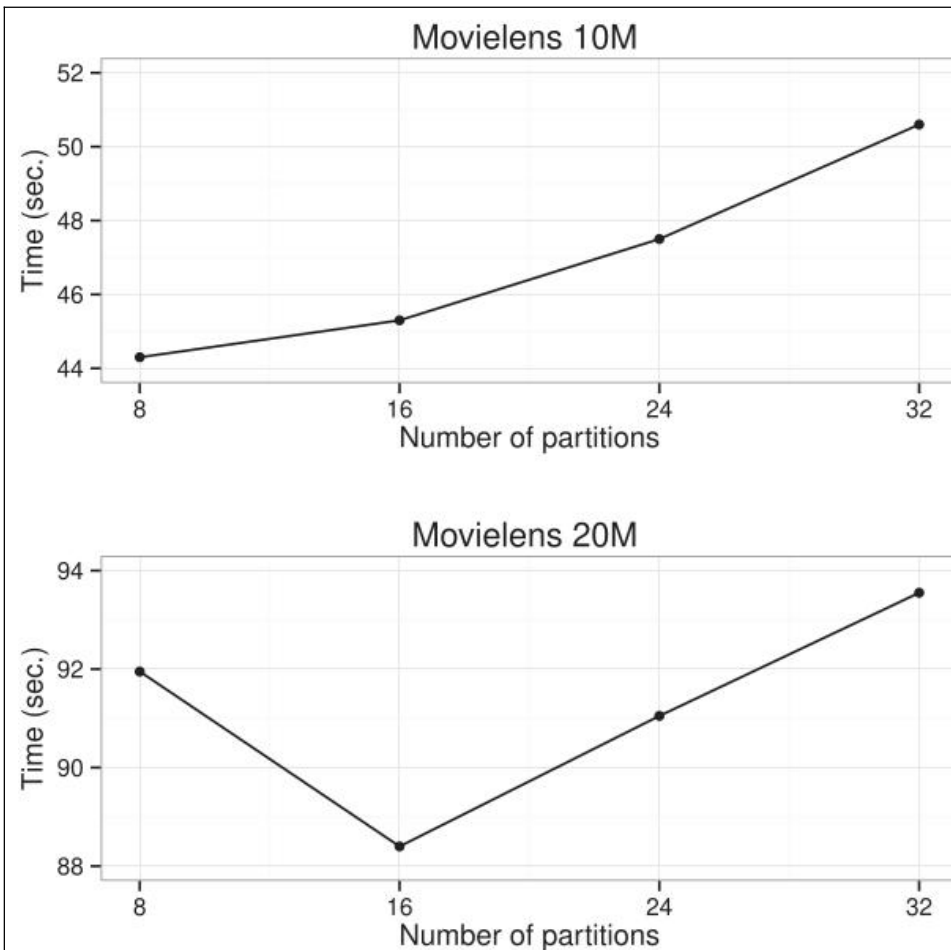


Fig. 6. The impact of the number of partitions n_p on the runtime (DPMI).

Рис. 6. Влияние количества разделов n_p на время выполнения (DPMI).

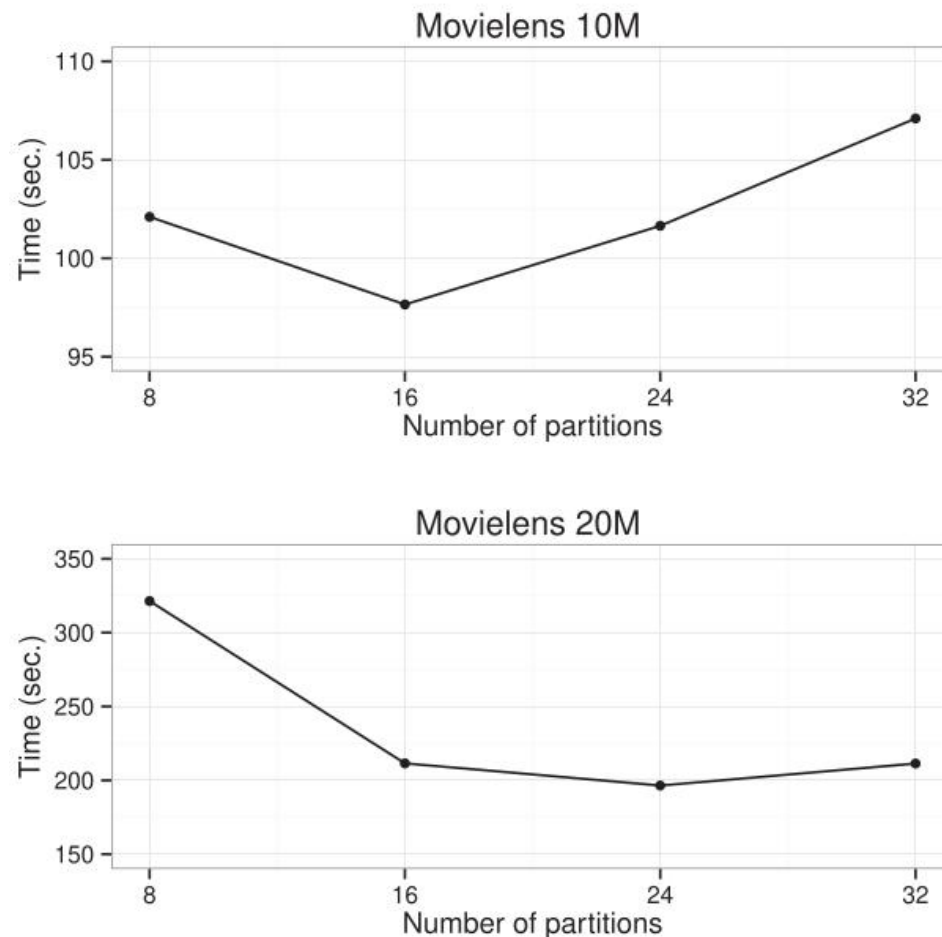


Table 2 The impact of the number of partitions n p on the acceleration performance (methodology 1).

Method	Number of partitions	Movielens 10M	Movielens 20M
		Elapsed time (sec.)	Elapsed time (sec.)
DPM	8	44.3	91.95
	16	45.3	88.40
	24	47.5	91.05
	32	50.6	93.55
DPMI	8	102.10	321.4
	16	97.65	211.5
	24	101.65	196.5
	32	107.10	211.4

Таблица 2 Влияние количества разделов n p на показатели ускорения

Table 3 Performance comparisons on Movielens 10M and Movielens

20M	data	sets	(methodology		2).
Method	ϕ	Movielens 10M		Movielens 20M	
		MAE	RMSE	MAE	RMSE
UPCC (Zhang et al., 2016)	–	0.8218	1.0537	0.8261	1.0621
ITAN (Kupisz & Unold, 2015)	–	0.7308	0.9370	0.7131	0.9225
FRAIPA (Ait Hammou & Ait Lahcen, 2017)	–	0.6758	0.8892	0.6643	0.8822
CMPTF (Zheng et al., 2016)	–	–	–	0.6343	0.8182
CESP (Hwang et al., 2016)	–	0.6870	0.8988	–	–
	1.0	0.6623	0.8735	0.6519	0.8674
	0.8	0.6638	0.8754	0.6534	0.8694
APRA (Ait Hammou et al., 2018)	0.6	0.6662	0.8790	0.6557	0.8728
	0.5	0.6681	0.8818	0.6576	0.8755
	0.4	0.6711	0.8858	0.6604	0.8796
DPM	–	0.6591	0.8596	0.6490	0.8527
DPMI	–	0.6534	0.8570	0.6440	0.8502
DPMF	–	0.6068	0.7928	0.5977	0.7854

Table 4 Performance comparisons on Yelp data set (methodology 2).				
Method	ϕ	Yelp		
		MAE	RMSE	
UPCC (Zhang et al., 2016)	–	0.9748	1.2965	
ITAN (Kupisz & Unold, 2015)	–	0.8410	1.0692	
FRAIPA (Ait Hammou & Ait Lahcen, 2017)	–	0.7958	1.0442	
APRA (Ait Hammou et al., 2018)	1.0	0.7922	1.0357	
DPM	–	0.7949	1.0335	
DPMI	–	0.7887	1.0383	

<p>From Fig. 5 , it is clear that the number of partitions has a prominent impact on the performance of the model DPM. Specifically, for the MovieLens 10M data set, the elapsed time spent in the training task grows as the number of partitions increased. The model DPM achieves the optimal performance when the number of partitions is equal to 8. While, for the MovieLens 20M data set, it can be observed that, with the increase of the number of partitions n p from 8 to 32, the computational time of DPM first decreases and reach minima when n * p = 16 , and then grow when n p continues to increase.</p>	<p>Из рис. 5 видно, что количество разделов оказывает заметное влияние на производительность модели DPM. В частности, для набора данных MovieLens 10M затраченное время, затрачиваемое на учебное задание, увеличивается с увеличением количества разделов. Модель DPM достигает оптимальной производительности, когда число разделов равно 8. В то время как для набора данных MovieLens 20M можно наблюдать, что при увеличении числа разделов n p с 8 до 32 вычислительное время DPM сначала уменьшается и достигает минимумов, когда n * p = 16, а затем растет, когда n p продолжает расти.</p>
<p>Fig. 6 presents how the number of partitions influences the performance of DPMI model. It is obvious that the computational time is also sensitive to the choice of the number of groups n p . For MovieLens 10M, the model DPMI achieves the optimal performance when n p is equal to 16. However, for MovieLens 20M, DPMI can considerably reduce the computational time when the number of partitions is approximately equal</p>	<p>На рис. 6 показано, как количество разделов влияет на производительность модели DPMI. Очевидно, что вычислительное время также чувствительно к выбору количества групп n p. Для MovieLens 10M модель DPMI достигает оптимальной производительности, когда n p равно 16. Однако для MovieLens 20M DPMI может значительно сократить</p>

to 24.	вычислительное время, когда число разделов приблизительно равно 24.																																														
Table 2 shows in detail, for the models DPM and DPMI, how the computational time changes according to the number of partitions. The best results are in bold.	Таблица 2 подробно показывает, для моделей DPM и DPMI, как вычислительное время изменяется в зависимости от количества разделов. Лучшие результаты выделены жирным шрифтом.																																														
In can be observed that DPM spends relatively less computational time than DPMI, which is consistent with the fact that DPMI is tailored to consider the optimal parameters of DPM to improve the quality of predictions.	Можно заметить, что DPM тратит относительно меньше вычислительного времени, чем DPMI, что согласуется с тем фактом, что DPMI приспособлен для учета оптимальных параметров DPM для улучшения качества прогнозов.																																														
To effectively evaluate the performance of the proposed models, it is necessary to compare them with other competitors. Table 3 reports the experimental results on MovieLens 10M and MovieLens 20M data sets, according to methodology 2. Table 4 shows the experimental results on Yelp data set using the methodology 2. The columns list the MAE and RMSE values. Comparative results show that the proposed models are better than other competitors in terms of prediction quality. Note that, the model DPMF adopts the same optimal parameters of DPMI.	Чтобы эффективно оценить производительность предлагаемых моделей, необходимо сравнить их с другими конкурентами. В таблице 3 приведены экспериментальные результаты для наборов данных MovieLens 10M и MovieLens 20M в соответствии с методологией 2. В таблице 4 показаны экспериментальные результаты для набора данных Yelp с использованием методологии 2. В столбцах приведены значения MAE и RMSE. Сравнительные результаты показывают, что предлагаемые модели лучше других конкурентов с точки зрения качества прогноза. Обратите внимание, что модель DPMF принимает те же оптимальные параметры DPMI.																																														
Table 5 shows the experimental results on the MovieLens 10M and MovieLens 20M data sets, according to 10-fold cross-validation (i.e., methodology 3). It can be seen from the results, our proposal achieves the best results on both data sets.	В таблице 5 показаны экспериментальные результаты для наборов данных MovieLens 10M и MovieLens 20M в соответствии с 10-кратной перекрестной проверкой (т.е. методология 3). Как видно из результатов, наше предложение достигает лучших результатов для обоих наборов данных.																																														
On the other hand, Table 6 illustrates the comparison of MAE, RMSE, standard deviation (i.e., σ), and elapsed time according to methodology 1. The results reveal that the proposed models can significantly outperform the state-of-the-art methods in terms of MAE, RMSE and computational time.	С другой стороны, таблица 6 иллюстрирует сравнение MAE, RMSE, стандартного отклонения (то есть, σ) и прошедшего времени согласно методике 1. Результаты показывают, что предлагаемые модели могут значительно превзойти современные методы с точки зрения MAE, RMSE и вычислительного времени.																																														
Table 5 Performance comparisons on Movielens 10M and Movielens 20M data sets (10-fold cross-validation).	Таблица 5 Сравнение производительности для наборов данных Movielens 10M и Movielens 20M (10-кратная перекрестная проверка)																																														
<table><tr><th rowspan="2">Method</th><th rowspan="2">ϕ</th><th colspan="2">Movielens 10M</th><th colspan="2">Movielens 20M</th></tr><tr><th>MAE $\pm \sigma$</th><th>RMSE $\pm \sigma$</th><th>MAE $\pm \sigma$</th><th>RMSE $\pm \sigma$</th></tr><tr><td>MLR (Fr�mal & Lecron, 2017)</td><td>–</td><td>0.7106 \pm 0.0006</td><td>0.8374 \pm 0.0015</td><td>0.7027 \pm 0.0004</td><td>0.8309 \pm 0.0010</td></tr><tr><td>CM II (Fr�mal & Lecron, 2017)</td><td>–</td><td>0.7141 \pm 0.0005</td><td>0.8296 \pm 0.0012</td><td>0.7064 \pm 0.0004</td><td>0.8216 \pm 0.0010</td></tr><tr><td>SW I (Fr�mal & Lecron, 2017)</td><td>–</td><td>0.7136 \pm 0.0005</td><td>0.8296 \pm 0.0013</td><td>0.7054 \pm 0.0004</td><td>0.8207 \pm 0.0010</td></tr><tr><td>DPM</td><td>–</td><td>0.6592 \pm 0.0004</td><td>0.8593 \pm 0.0007</td><td>0.6485 \pm 0.0003</td><td>0.8521 \pm 0.0004</td></tr><tr><td>DPMI</td><td>–</td><td>0.6534 \pm 0.0005</td><td>0.8565 \pm 0.0007</td><td>0.6434 \pm 0.0002</td><td>0.8497 \pm 0.0003</td></tr><tr><td>DPMF</td><td>–</td><td>0.6053 \pm 0.0008</td><td>0.7903 \pm 0.0010</td><td>0.5952 \pm 0.0004</td><td>0.7819 \pm 0.0005</td></tr></table>	Method	ϕ	Movielens 10M		Movielens 20M		MAE $\pm \sigma$	RMSE $\pm \sigma$	MAE $\pm \sigma$	RMSE $\pm \sigma$	MLR (Fr�mal & Lecron, 2017)	–	0.7106 \pm 0.0006	0.8374 \pm 0.0015	0.7027 \pm 0.0004	0.8309 \pm 0.0010	CM II (Fr�mal & Lecron, 2017)	–	0.7141 \pm 0.0005	0.8296 \pm 0.0012	0.7064 \pm 0.0004	0.8216 \pm 0.0010	SW I (Fr�mal & Lecron, 2017)	–	0.7136 \pm 0.0005	0.8296 \pm 0.0013	0.7054 \pm 0.0004	0.8207 \pm 0.0010	DPM	–	0.6592 \pm 0.0004	0.8593 \pm 0.0007	0.6485 \pm 0.0003	0.8521 \pm 0.0004	DPMI	–	0.6534 \pm 0.0005	0.8565 \pm 0.0007	0.6434 \pm 0.0002	0.8497 \pm 0.0003	DPMF	–	0.6053 \pm 0.0008	0.7903 \pm 0.0010	0.5952 \pm 0.0004	0.7819 \pm 0.0005	
Method			ϕ	Movielens 10M		Movielens 20M																																									
	MAE $\pm \sigma$	RMSE $\pm \sigma$		MAE $\pm \sigma$	RMSE $\pm \sigma$																																										
MLR (Fr�mal & Lecron, 2017)	–	0.7106 \pm 0.0006	0.8374 \pm 0.0015	0.7027 \pm 0.0004	0.8309 \pm 0.0010																																										
CM II (Fr�mal & Lecron, 2017)	–	0.7141 \pm 0.0005	0.8296 \pm 0.0012	0.7064 \pm 0.0004	0.8216 \pm 0.0010																																										
SW I (Fr�mal & Lecron, 2017)	–	0.7136 \pm 0.0005	0.8296 \pm 0.0013	0.7054 \pm 0.0004	0.8207 \pm 0.0010																																										
DPM	–	0.6592 \pm 0.0004	0.8593 \pm 0.0007	0.6485 \pm 0.0003	0.8521 \pm 0.0004																																										
DPMI	–	0.6534 \pm 0.0005	0.8565 \pm 0.0007	0.6434 \pm 0.0002	0.8497 \pm 0.0003																																										
DPMF	–	0.6053 \pm 0.0008	0.7903 \pm 0.0010	0.5952 \pm 0.0004	0.7819 \pm 0.0005																																										

Method	ϕ	Movielens 10M			Movielens 20M		
		MAE $\pm \sigma$	RMSE $\pm \sigma$	Elapsed time (sec.)	MAE $\pm \sigma$	RMSE $\pm \sigma$	Elapsed time (sec.)
BMF (Hernando et al., 2016)	—	0.676	—	—	—	—	—
PLSA-CF (Hofmann, 2004)	—	0.823	—	—	—	—	—
KNN JMSD (Bobadilla et al., 2010)	—	0.753	—	—	—	—	—
NMF (Lee & Seung, 1999)	—	1.356	—	—	—	—	—
FRAIPA (Ait Hammou & Ait Lahcen, 2017)	—	0.6764 \pm 0.0003	0.8899 \pm 0.0004	206.9	0.6644 \pm 0.0002	0.8823 \pm 0.0003	464.55
	1.0	0.6628 \pm 0.0003	0.8740 \pm 0.0003	239.2	0.6519 \pm 0.0002	0.8676 \pm 0.0003	489.2
	0.8	0.6643 \pm 0.0003	0.8761 \pm 0.0003	190.25	0.6534 \pm 0.0002	0.8696 \pm 0.0003	423.00
APRA (Ait Hammou et al., 2018)	0.6	0.6667 \pm 0.0003	0.8794 \pm 0.0004	130.65	0.6557 \pm 0.0002	0.8729 \pm 0.0003	237.75
	0.5	0.6687 \pm 0.0003	0.8822 \pm 0.0003	106.35	0.6577 \pm 0.0002	0.8756 \pm 0.0004	186.00
	0.4	0.6715 \pm 0.0002	0.8861 \pm 0.0003	84.05	0.6605 \pm 0.0002	0.8795 \pm 0.0003	142.15
DPM	—	0.6598 \pm 0.0004	0.8602 \pm 0.0005	44.3	0.6491 \pm 0.0001	0.8530 \pm 0.0002	88.40
DPMI	—	0.6541 \pm 0.0004	0.8576 \pm 0.0005	97.65	0.6440 \pm 0.0001	0.8506 \pm 0.0005	196.5
DPMF	—	0.6072 \pm 0.0004	0.7934 \pm 0.0005	49.99	0.5972 \pm 0.0004	0.7849 \pm 0.0004	1185.2

The objective of this work is to propose a novel distributed recommendation model with matrix factorization and random forest, which adequately exploits the strengths Apache Spark and Hadoop frameworks, to improve the performance in Big Data environment.

In addition to the prediction quality, the computational time represents an important issue in building recommendation systems for Big Data. The obtained results show that DPM is faster than DPMI, DPMF and other competitors. The improvement in terms of the computational time is significant. In fact, DPM can save between 47.29% and 81.47% of the computational time for MovieLens 10M, and the improvement up to 81.92% for MovieLens 20M.

Таблица 6 Сравнение производительности для наборов данных Movielens 10M и Movielens 20M (методология 1)

Цель этой работы - предложить новую модель распределенных рекомендаций с матричной факторизацией и случайным лесом, которая адекватно использует сильные стороны фреймворков Apache Spark и Hadoop, для повышения производительности в среде больших данных.

Помимо качества прогноза, время вычислений представляет собой важную проблему при создании систем рекомендаций для больших данных. Полученные результаты показывают, что DPM быстрее, чем DPMI, DPMF и другие конкуренты. Улучшение с точки зрения вычислительного времени является значительным. Фактически, DPM может сэкономить от 47,29% до 81,47% вычислительного времени для MovieLens 10M, а улучшение - до 81,92% для MovieLens 20M.

Кроме того, алгоритм APRA принимает случайную выборку, чтобы уменьшить размерность данных и ускорить учебную задачу. Результаты доказывают, что это может значительно сократить время вычислений с небольшим увеличением ошибки прогнозирования. Тем не менее, благодаря новому процессу обучения, предложенные модели могут быть эффективными с

	точки зрения вычислительного времени без снижения качества предсказаний.
The main advantages of this work are the ability to reduce the computational cost, the capacity to effectively improve the quality of predictions, handle large-scale data sets, and alleviate data sparsity. Furthermore, the proposed solution relies only on the user-item rating matrix to perform the training and prediction tasks, therefore it can be easily utilized as a recommendation system of real-world e-commerce companies. In addition, our models solve the recommendation problem as a set of independent subproblems, which allows estimating the parameters efficiently, and provides the flexibility to consider the new ratings in the system without offline computation. These facts represent a clear indication that our models can contribute to a more efficient design of recommendation systems in the context of Big Data.	Основными преимуществами этой работы являются возможность уменьшить вычислительные затраты, способность эффективно улучшать качество прогнозов, обрабатывать крупномасштабные наборы данных и уменьшать разреженность данных. Кроме того, предлагаемое решение опирается только на матрицу оценки элементов пользователя для выполнения задач обучения и прогнозирования, поэтому оно может быть легко использовано в качестве системы рекомендаций реальных компаний электронной коммерции. Кроме того, наши модели решают проблему рекомендаций в виде набора независимых подзадач, что позволяет эффективно оценивать параметры и обеспечивает гибкость для учета новых оценок в системе без автономных вычислений. Эти факты ясно свидетельствуют о том, что наши модели могут способствовать более эффективному проектированию систем рекомендаций в контексте больших данных.
7. Conclusions and future work	7. Выводы и будущая работа
In this paper, a novel distributed predictive model is proposed for the personalized recommendation in the context of Big Data. It is designed based on Spark to address several challenges related to recommendation systems and Big Data.	В этой статье предлагается новая модель распределенного прогнозирования для персонализированной рекомендации в контексте больших данных. Он разработан на основе Spark для решения нескольких проблем, связанных с системами рекомендаций и большими данными.
The key point of this proposal lies in the adoption of data partitioning strategy to speed up Big Data processing. Secondly, the representation of the rating behavior for each user and the opinions of users for each item based on a more sophisticated procedure. Thirdly, the division of the optimization problem into a set of independent subproblems, where each subproblem can be resolved based on a series of consecutive objective functions. In particular, these cost functions are tailored to efficiently learn the parameters in a parallel and distributed way, and overcome the data sparsity problem. The experimental results show that the proposed solution significantly outperforms existing recommendation methods in terms of prediction error and computational time.	Ключевым моментом этого предложения является принятие стратегии разделения данных для ускорения обработки больших данных. Во-вторых, представление рейтингового поведения для каждого пользователя и мнений пользователей для каждого элемента основано на более сложной процедуре. В-третьих, разделение задачи оптимизации на набор независимых подзадач, где каждая подзадача может быть решена на основе ряда последовательных целевых функций. В частности, эти функции стоимости предназначены для эффективного изучения параметров в параллельном и распределенном режиме и преодоления проблемы разреженности данных. Результаты эксперимента показывают, что предлагаемое решение значительно превосходит существующие методы рекомендаций с точки зрения ошибки прогнозирования и времени вычислений.
For future work, it would be interesting to design an efficient mechanism	Для будущей работы было бы интересно разработать

to further reduce the computational cost of the proposed approach DPMF. As a second future work, we plan to extend our approach by incorporating various information such as the social contexts of users, and the content-based features to improve the prediction quality. In addition, other large-scale data sets will be utilized to evaluate the effectiveness of our proposal in other recommendation scenarios like social recommendations, and group recommendations.	эффективный механизм для дальнейшего снижения вычислительных затрат предлагаемого подхода DPMF. В качестве второй будущей работы мы планируем расширить наш подход, включив в него различную информацию, такую как социальный контекст пользователей и контентные функции, для улучшения качества прогнозирования. Кроме того, другие крупномасштабные наборы данных будут использоваться для оценки эффективности нашего предложения в других сценариях рекомендаций, таких как социальные рекомендации и групповые рекомендации.
Declaration of competing interest	Декларация о конкурирующих интересах
We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.	Мы хотим подтвердить, что с этой публикацией не было никаких известных конфликтов интересов, и не было значительной финансовой поддержки этой работы, которая могла бы повлиять на ее результаты.
Credit authorship contribution statement	
Badr Ait Hammou: Writing original draft, Conceptualization, Methodology, Data curation, Formal analysis, Investigation, Writing review & editing, Software. Ayoub Ait Lahcen: Supervision, Investigation, Validation. Salma Mouline: Supervision, Investigation, Validation.	
Acknowledgments	
This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.	
References	
Aggarwal, C. C. (2016). Recommender systems: the textbook . Springer . Ait Hammou, B. , & Ait Lahcen, A. (2017). Fraipa: A fast recommendation approach with improved prediction accuracy. Expert Systems with Applications, 87 , 90–97 . Ait Hammou, B. , Ait Lahcen, A. , & Mouline, S. (2018). Apra: An approximate parallel recommendation algorithm for big data. Knowledge-Based Systems, 157 , 10–19 . Ait Hammou, B., Ait Lahcen, A., & Mouline, S. (2019a). A distributed group recommendation system based on extreme gradient boosting and big data technologies. Applied Intelligence . doi: 10.1007/s10489019014829 . Ait Hammou, B. , Ait Lahcen, A. , & Mouline, S. (2019b). Fraipa version 2: A fast recommendation approach based on self-adaptation and multi-thresholding. Expert Systems with Applications, 118 , 209–216 . Al-Hassan, M. , Lu, H. , & Lu, J. (2015). A semantic enhanced hybrid recommendation approach: A case study of e-	

government tourism service recommendation system. *Decision Support Systems*, 72 , 97–109 .

Al-Shamri, M. Y. H. (2014). Power coefficient as a similarity measure for memory-based collaborative recommender systems. *Expert Systems with Applications*, 41 (13), 56 80–56 88 .

Ar, Y. , & Bostanci, E. (2016). A genetic algorithm solution to the collaborative filtering problem. *Expert Systems with Applications*, 61 , 122–128 .

Bobadilla, J. , Ortega, F. , Hernando, A. , & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, 46 , 109–132 .

Bobadilla, J. , Serradilla, F. , & Bernal, J. (2010). A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, 23 (6), 520–528 .

Borràs, J. , Moreno, A. , & Valls, A. (2014). Intelligent tourism recommender systems: A survey. *Expert Systems with Applications*, 41 (16), 7370–7389 .

Breiman, L. (2001). Random forests. *Machine learning*, 45 (1), 5–32 .

Chen, J. , Li, K. , Rong, H. , Bilal, K. , Yang, N. , & Li, K. (2018a). A disease diagnosis and treatment recommendation system based on big data mining and cloud computing. *Information Sciences*, 435 , 124–149 .

Chen, J. , Li, K. , Tang, Z. , Bilal, K. , Yu, S. , Weng, C. , & Li, K. (2016). A parallel random forest algorithm for big data in a spark cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 28 (4), 919–933 .

Chen, J. , Wang, H. , Yan, Z. , et al. (2018b). Evolutionary heterogeneous clustering for rating prediction based on user collaborative filtering. *Swarm and Evolutionary Computation*, 38 , 35–41 .

da Costa, A. F. , Manzato, M. G. , & Campello, R. J. (2019). Boosting collaborative filtering with an ensemble of co-trained recommenders. *Expert Systems with Applications*, 115 , 427–441 .

Da Costa, A. F. , Manzato, M. G. , & Campello, R. J. (2018). Corec: a co-training approach for recommender systems. In *Proceedings of the 33rd annual acm symposium on applied computing* (pp. 696–703). ACM .

Dean, J. , & Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51 (1), 107–113 .

Duma, M. , & Twala, B. (2019). Sparseness reduction in collaborative filtering using a nearest neighbour artificial immune system with genetic algorithms. *Expert Systems with Applications*, 132 , 110–125 .

Fernández-Tobías, I. , Cantador, I. , Tomeo, P. , Anelli, V. W. , & Di Noia, T. (2019). Addressing the user cold start with cross-domain collaborative filtering: Exploiting item metadata in Matrix factorization. *User Modeling and User-Adapted Interaction* , 1–44 .

Frémal, S. , & Lecron, F. (2017). Weighting strategies for a recommender system using

item clustering based on genres. *Expert Systems with Applications*, 77 , 105–113 . Galicia, A. , Torres, J. F. , Martínez-Álvarez, F. , & Troncoso, A. (2018). A novel spark-based multi-step forecasting algorithm for big data time series. *Information Sciences*, 467 , 800–818 . Genuer, R. , Poggi, J.-M. , Tuleau-Malot, C. , & Villa-Vialaneix, N. (2017). Random forests for big data. *Big Data Research*, 9 , 28–46 . Glushkova, D. , Jovanovic, P. , & Abelló, A. (2019). Mapreduce performance model for Hadoop 2. x. *Information Systems*, 79 , 32–43 . Godoy-Lorite, A. , Guimerà, R. , Moore, C. , & Sales-Pardo, M. (2016). Accurate and scalable social recommendation using mixed-membership stochastic block models. *Proceedings of the National Academy of Sciences*, 113 (50), 14207–14212 . Hadoop (2019). Apache Hadoop. <https://hadoop.apache.org/> . Accessed 02 june 2019. Harper, F. M. , & Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5 (4), 19 . Hernando, A. , Bobadilla, J. , & Ortega, F. (2016). A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. *Knowledge-Based Systems*, 97 , 188–202 . Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22 (1), 89–115 . Hsieh, M.-Y. , Weng, T.-H. , & Li, K.-C. (2018). A keyword-aware recommender system using implicit feedback on Hadoop. *Journal of Parallel and Distributed Computing*, 116 , 63–73 . Hu, R. , Dou, W. , & Liu, J. (2014). Clubcf: A clustering-based collaborative filtering approach for big data application. *IEEE Transactions on Emerging Topics in Computing*, 2 (3), 302–313 . Hwang, W.-S. , Lee, H.-J. , Kim, S.-W. , Won, Y. , & Lee, M.-s. (2016). Efficient recommendation methods using category experts for a large dataset. *Information Fusion*, 28 , 75–82 . Jiang, J. , Lu, J. , Zhang, G. , & Long, G. (2011). Scaling-up item-based collaborative filtering recommendation algorithm based on Hadoop. In *Proceedings of the IEEE world congress on services (services)* (pp. 4 90–4 97). IEEE . Kaleli, C. (2014). An entropy-based neighbor selection approach for collaborative filtering. *Knowledge-Based Systems*, 56 , 273–280 . Kim, S. , Kim, H. , & Min, J.-K. (2019). An efficient parallel similarity matrix construction on mapreduce for collaborative filtering. *The Journal of Supercomputing*, 75 (1), 123–141 . Koren, Y. , Bell, R. , & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer* , (8), 30–37 . Kupisz, B. , & Unold, O. (2015). Collaborative filtering recommendation algorithm based on Hadoop and

spark. In Proceedings of the IEEE international conference on industrial technology (ICIT) (pp. 1510–1514). IEEE . Lee, C.-H. , & Lin, C.-Y. (2017). Implementation of lambda architecture: A restaurant recommender system over apache mesos. In Proceedings of the IEEE 31st international conference on advanced information networking and applications (AINA) (pp. 979–985). IEEE . Lee, D. D. , & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401 (6755), 788 . Li, C. , & He, K. (2017). CBMR: An optimized mapreduce for item-based collaborative filtering recommendation algorithm with empirical analysis. *Concurrency and Computation: Practice and Experience*, 29 (10), e4092 . Li, J. , Chen, C. , Chen, H. , & Tong, C. (2017). Towards context-aware social recommendation via individual trust. *Knowledge-Based Systems*, 127 , 58–66 . Liu, B. (2007). *Web data mining: Exploring hyperlinks, contents, and usage data* . Springer Science & Business Media . Lu, J. , Wu, D. , Mao, M. , Wang, W. , & Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems*, 74 , 12–32 . Ma, X. , Lu, H. , Gan, Z. , & Zhao, Q. (2016). An exploration of improving prediction accuracy by constructing a multi-type clustering based recommendation framework. *Neurocomputing*, 191 , 388–397 . Maillo, J. , Ramírez, S. , Triguero, I. , & Herrera, F. (2017). Knn-is: An iterative spark-based design of the k-nearest neighbors classifier for big data. *Knowledge-Based Systems*, 117 , 3–15 . Mao, M., Lu, J., Zhang, G., & Zhang, J. (2017). Multirelational social recommendations via multigraph ranking. *IEEE Transactions on Cybernetics*, 47 (12), 4049– 4061. doi: 10.1109/TCYB.2016.2595620 . Mazurowski, M. A. (2013). Estimating confidence of individual rating predictions in collaborative filtering recommender systems. *Expert Systems with Applications*, 40 (10), 3847–3857 . Ortega, F. , Hernando, A. , Bobadilla, J. , & Kang, J. H. (2016). Recommending items to group of users using matrix factorization based collaborative filtering. *Information Sciences*, 345 , 313–324 . Osadchiy, T. , Poliakov, I. , Olivier, P. , Rowland, M. , & Foster, E. (2019). Recommender system based on pairwise association rules. *Expert Systems with Applications*, 115 , 535–542 . Papadakis, H. , Panagiotakis, C. , & Fragopoulou, P. (2017). SCOR: A synthetic coordinate based recommender system. *Expert Systems with Applications*, 79 , 8–19 . Parvin, H. , Moradi, P. , & Esmaeili, S. (2019). TCFACO: Trust-aware collaborative filtering method based on ant colony optimization. *Expert Systems with Applications*, 118 , 152–168 . Salah, A. , Rogovschi, N. , &

Nadif, M. (2016). A dynamic collaborative filtering system via a weighted clustering approach. *Neurocomputing*, 175 , 206–215 .

Singh, M. , & Mehrotra, M. (2018). Impact of biclustering on the performance of biclustering based collaborative filtering. *Expert Systems with Applications*, 113 , 443–456 .

Spark (2019). Apache spark. <https://spark.apache.org> . Accessed 02 june 2019.

Sun, Z. , Wu, B. , Wu, Y. , & Ye, Y. (2019). APL: Adversarial pairwise learning for recommender systems. *Expert Systems with Applications*, 118 , 573–584 .

Takaishi, D. , Nishiyama, H. , Kato, N. , & Miura, R. (2014). Toward energy efficient big data gathering in densely distributed sensor networks. *IEEE Transactions on Emerging Topics in Computing*, 2 (3), 388–397 .

B. Ait Hammou, A. Ait Lahcen and S. Mouline / *Expert Systems With Applications* 137 (2019) 253–265

Tran, T. , Lee, K. , Liao, Y. , & Lee, D. (2018). Regularizing matrix factorization with user and item embeddings for recommendation. In *Proceedings of the 27th acm international conference on information and knowledge management* (pp. 687–696). ACM .

Turk, A. M. , & Bilge, A. (2019). Robustness analysis of multi-criteria collaborative filtering algorithms against shilling attacks. *Expert Systems with Applications*, 115 , 386–402 .

Wang, R. , Cheng, H. K. , Jiang, Y. , & Lou, J. (2019). A novel matrix factorization model for recommendation with LOD-based semantic similarity measure. *Expert Systems with Applications*, 123 , 70–81 .

White, T. (2012). *Hadoop: The definitive guide* . O'Reilly Media, Inc .

Winlaw, M. , Hynes, M. B. , Caterini, A. , & De Sterck, H. (2015). Algorithmic acceleration of parallel ALS for collaborative filtering: Speeding up distributed big data recommendation in spark. In *Proceedings of the IEEE 21st international conference on parallel and distributed systems (icpads)* (pp. 682–691). IEEE .

Wu, D., Lu, J., & Zhang, G. (2015). A fuzzy tree matching-based personalized elearning recommender system. *IEEE Transactions on Fuzzy Systems*, 23 (6), 2412– 2426. doi: 10.1109/TFUZZ.2015.2426201 .

Wu, M.-L. , Chang, C.-H. , & Liu, R.-Z. (2014). Integrating content-based filtering with collaborative filtering using co-clustering with augmented matrices. *Expert Systems with Applications*, 41 (6), 2754–2761 .

Wyner, A. J. , Olson, M. , Bleich, J. , & Mease, D. (2017). Explaining the success of adaboost and random forests as interpolating classifiers. *The Journal of Machine Learning Research*, 18 (1), 1558–1590 .

Xu, W. , Sun, J. , Ma, J. , & Du, W. (2016). A personalized information recommendation system for r&d project opportunity finding in big data contexts. *Journal of Network and Computer Applications*, 59 , 362–369 .

Yelp (2019). Yelp

dataset. <https://www.yelp.com/dataset> . Accessed 02 june 2019. Yin, C. , Wang, J. , & Park, J. H. (2017). An improved recommendation algorithm for big data cloud service based on the trust in sociology. *Neurocomputing*, 256 , 49–55 . 265 Yuan, X. , Han, L. , Qian, S. , Xu, G. , & Yan, H. (2019). Singular value decomposition based recommendation using imputed data. *Knowledge-Based Systems*, 163 , 4 85–4 94 . Zaharia, M. , Chowdhury, M. , Das, T. , Dave, A. , Ma, J. , McCauley, M. , . . . Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th unix conference on networked systems design and implementation* . USENIX Association . (pp. 2–2) Zaharia, M. , Chowdhury, M. , Franklin, M. J. , Shenker, S. , & Stoica, I. (2010). Spark: Cluster computing with working sets.. *HotCloud*, 10 (10-10), 95 . Zhang, F. , Gong, T. , Lee, V. E. , Zhao, G. , Rong, C. , & Qu, G. (2016). Fast algorithms to evaluate collaborative filtering recommender systems. *Knowledge-Based Systems*, 96 , 96–103 . Zhang, F. , Lu, Y. , Chen, J. , Liu, S. , & Ling, Z. (2017a). Robust collaborative filtering based on non-negative matrix factorization and r_1 -norm. *Knowledge-based systems*, 118 , 177–190 . Zhang, Q. , Wu, D. , Lu, J. , Liu, F. , & Zhang, G. (2017b). A cross-domain recommender system with consistent information transfer. *Decision Support Systems*, 104 , 49–63 . Zhang, Y.-w. , Zhou, Y.-y. , Wang, F.-t. , Sun, Z. , & He, Q. (2018). Service recommendation based on quotient space granularity analysis and covering algorithm on spark. *Knowledge-Based Systems*, 147 , 25–35 . Zhao, G., Qian, X., & Xie, X. (2016). User-service rating prediction by exploring social users' rating behaviors. *IEEE Transactions on Multimedia*, 18 (3), 496–506. doi: 10. 1109/TMM.2016.2515362 . Zhao, Z.-D. , & Shang, M.-S. (2010). User-based collaborative-filtering recommendation algorithms on Hadoop. In *Proceedings of the third international conference on Knowledge discovery and data mining WKDD'10*. (pp. 478–481). IEEE . Zheng, C. , Haihong, E. , Song, M. , & Song, J. (2016). CMPTF: Contextual modeling probabilistic tensor factorization for recommender systems. *Neurocomputing*, 205 , 141–151 . Zhou, X. , He, J. , Huang, G. , & Zhang, Y. (2015). Svd-based incremental approaches for recommender systems. *Journal of Computer and System Sciences*, 81 (4), 717–733 .