Medicon Engineering Themes
Volume 9 Issue 2 August 2025
Article Type: Review Article
ISSN: 2834-7218

Enhanced Recommender Systems for Big Data: A Feature Engineering Approach Using Apache Spark

Balaji GN, Parthasarathy Govindarajan* and Balaji N

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India

***Corresponding Author:** Parthasarathy Govindarajan, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India.

Received: May 22, 2025; **Published:** September 29, 2025

Abstract

In the era of big data, recommender systems play a crucial role in addressing information overload by suggesting relevant items to users based on their preferences. However, traditional recommendation methods face significant challenges in processing vast and complex data, leading to computational inefficiencies and reduced prediction accuracy. This paper introduces a novel recommender system designed specifically for big data environments, leveraging the Apache Spark platform to enhance scalability and address data sparsity issues.

Motivation: The increasing volume and complexity of user data in real-world applications create a need for more robust and scalable recommender systems. Traditional systems often struggle with performance and accuracy when applied to such large datasets. Our goal is to develop a system that can efficiently handle massive data while providing accurate recommendations.

Methods: We propose a BDMFE (Big Data Model with Feature Engineering) approach that optimizes data processing within a distributed computing environment, utilizing the capabilities of Apache Spark. Our model tackles both data sparsity and scalability issues by leveraging Spark's in-memory processing and parallelism.

Results: The proposed system was evaluated using three real-world datasets, demonstrating superior performance over existing recommendation models. In particular, it showed significant improvements in prediction accuracy, as measured by Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

Keywords: Machine Learning; Prediction; Feature Engineering; Big Data

Introduction

Big data is considered a valuable resource in making choices across different business niches. This allows organizations to make better decisions and improve their processes. Nevertheless, as data grows in size and complexity, we have more challenges in traditional computation techniques. Big data is often defined by its four Vs: volume, velocity, variety and veracity which makes it a hard problem to handle with traditional tools [1]. Such problems are particularly recognizable in processing the enormous bulk of user data being created on a daily basis across a range of platforms including ecommerce, social networks or streaming services.

One domain where big data transformation stands out is that of recommender systems. Recommender systems, or RSs, have become critical for helping users sift through mountains of data and finally identify the right products or services, based on what an individual likes or dislikes. The issue with this is that as the volume of digital information grows, it becomes difficult to find what is relevant and not; this spans from products on retail websites to movies on streaming services and content in other online marketplaces. RSs solve this problem because they use past user behaviour and historical data to provide personalized recommendations, thus making it easy for the user and driving engagement [2].

The fundamental idea behind any recommendation system is the ability to assess a user's tastes. To this date, one of the proven methodologies in this domain is Collaborative Filtering (CF). This method is predicated on the fact that if two or more users rated some items in the same way, their preferences would be aligned for future items as well. Ties the user action or item action together based on some similarity and does recommendation based on that [3]. But when CF techniques are applied to real world datasets, they are confronted with numerous limitations. For instance, platforms such as Amazon present more than 18.6 million products for over 20 million users and therefore generates immense amounts of data that traditional CF Modelling cannot handle. Therefore, this leads to the problems of reach, prediction precision, and data overfitting and lack of information [4].

Data sparsity refers to the problem where users interact with only a small subset of available items, leaving large parts of the user-item interaction matrix empty. This makes it difficult for CF models to accurately predict user preferences, especially when there is little overlap between user histories. Scalability becomes another issue as the size of user-item matrices grows exponentially with the increase in data, leading to long processing times and high computational costs. These limitations underscore the need for advanced recommender system models that can handle massive datasets while overcoming the common issues associated with CF.

In order to tackle these problems, we present a recommender system design meant for big data environments that integrates collaborative filtering approaches with feature engineering. In a nutshell, feature engineering improves the performance of a machine learning model by inventing new features out of existing raw data from some task. Example of such a task could be recommendation system feature engineering, where one wants to excavate intrinsic characteristics and correlations from user and item interaction exchange content. Nonetheless, it is possible to enhance the performance of content-based filtering approaches quite significantly by employing some additional features for the user, item, and correlation data.

In previous work, we introduced a feature engineering approach that integrated attribute engineering with collaborative filtering, using supervised learning algorithms to enhance prediction accuracy. The results demonstrated a notable improvement in prediction quality, particularly by reducing the mean absolute error (MAE) of the model [5]. This laid the foundation for the development of a scalable and accurate recommender system designed to handle big data.

Building on these insights, our proposed system leverages Apache Spark, a powerful distributed computing platform, to process large-scale datasets efficiently. Spark's in-memory processing capabilities and parallel computing framework make it ideal for handling the vast amounts of data involved in modern recommendation systems. The combination of Spark with our advanced feature engineering techniques enables the system to tackle both the scalability and data sparsity challenges that are inherent in big data environments.

To further improve prediction accuracy, we employ gradient boosting regression, a powerful machine learning technique that iteratively improves model performance by minimizing prediction errors. By combining this with feature engineering, we aim to develop a recommender system that not only scales effectively but also provides highly accurate predictions for user preferences.

The rest of the paper is structured as follows: Section 2 reviews the advancements made in the recommendation systems, big data processing and feature engineering techniques. Section 3 provides the background information on recommender systems within the big data context as well as the issues that they encounter. Section 4 presents the details of the approach that we put forward, including feature engineering and various strategies of collaborative filtering. In Section 5 we describe the experiments we have conducted on

real data sets while Section 6 analyses the result in terms of effectiveness and efficiency of the system. Finally, in Section 7 we present the implications of the work and recommend future research activities.

Related Work

The advent of big data and its rapid acceptance in different fields has revolutionized the working of businesses and organizations by promoting the use of data in making decisions. However, the influx of big data into the system with its massive volume presents a major challenge in the management and processing of information timely, especially in the area of recommendation systems (RSs). As recommender systems are integrated into e-commerce, social networking and content streaming services, novel techniques for handling large volumes of data have been developed.

Historically, recommender systems relied on data obtained from static, centralized data warehouses. These systems were effective for small-scale datasets but struggled to cope with the volume, velocity, and variety of modern data. To meet the increasing demands of users, recommender systems shifted to distributed computing environments to enhance scalability and performance. The nature of recommendation tasks, which involves processing dynamic user data and providing real-time results, necessitates parallelism in computation [6].

Xiaolong Jin et al. [7] identified two critical challenges in big data: data complexity and computational complexity. Data complexity arises from the diverse types, structures, and patterns within datasets, which are difficult to analyze using traditional methods. Computational complexity, on the other hand, refers to the difficulties in processing and analyzing these large datasets due to their sheer size. This complexity makes conventional machine learning and data mining techniques inadequate, requiring new strategies such as distributed computing frameworks like Apache Hadoop and Apache Spark.

In their work, R. Patgiri et al. [8] discussed the broad applications of big data across multiple disciplines, such as neuroscience, climate research, and cancer studies. NASA, for instance, uses big data to process petabytes of climate data for weather forecasting. This demonstrates the multidisciplinary importance of big data and its future impact on all aspects of human life. As data continues to grow, traditional recommender systems are unable to cope with the computational requirements, necessitating new approaches.

Traditional memory-based collaborative filtering (CF) algorithms, which were once the backbone of recommendation systems, have become inefficient in the big data era. The CF approach struggles with the scalability and data sparsity problems, where users only rate a small portion of available items, leading to large portions of the user-item matrix being empty. To address these issues, Hafed Zarzour et al. [9] proposed combining CF with k-means clustering and principal component analysis (PCA) to improve recommendation accuracy. PCA helps in reducing the dimensionality of the data, thus enhancing the performance of k-means clustering in large datasets.

Another important contribution in this domain is the work of Robin Genuer et al. [10], who tackled the classification problem in big data using random forests. The authors addressed the challenges of computational efficiency by implementing parallel settings and "divide-and-conquer" strategies, thereby improving the performance of random forest models. Their study applied random forest techniques to large, real-world datasets and demonstrated that parallel random forest algorithms could provide high-quality results without the overfitting issues common in other models.

Apache Hadoop and Apache Spark are two major big data frameworks that have been extensively adopted in recommendation system research. Hsieh, Weng, and Li [11] introduced a keyword-aware recommendation engine using Apache Hadoop. Their model handled large-scale datasets efficiently by utilizing keywords extracted from textual data related to users and items. This method not only optimized system performance but also addressed the cold-start problem, which occurs when there is insufficient data for new users or items. However, while Hadoop facilitates parallel computation, Apache Spark has proven to be faster due to its in-memory processing capabilities. Badr Ait Hammou [12] demonstrated a collaborative filtering system built on Apache Spark, showing that Spark's in-memory operations significantly outperformed Hadoop-based systems for large-scale recommendations. Hammou's system efficiently processed vast amounts of data, leveraging Spark's capability to handle both structured and unstructured data in real-time.

Building on this, Hammou et al. [13] proposed a distributed recommendation system designed to address the limitations of traditional RSs, particularly the problems of data sparsity and scalability in big data environments. Their system incorporated distributed training methods to increase performance and improve prediction quality in recommendation tasks, utilizing the inherent parallelism in Spark to process large datasets faster.

Recommender systems have also seen applications in specialized fields like e-learning. For example, a study on distributed course recommendations utilized association rule mining to identify correlations between students' learning activities and suggest the most appropriate resources for them. The model leveraged FP-growth and the Hadoop ecosystem to mine frequent patterns from students' activity logs and predict relevant courses. This study further demonstrated the potential of big data techniques in personalizing learning experiences on a large scale [14].

Significant contributions to RS research have also come from the KDD Cup competition, where participants addressed challenges in binary classification for Yahoo! Music's recommendation system. The top teams employed gradient-boosted decision trees and feature engineering, highlighting the importance of creating domain-specific features to improve prediction accuracy [15]. This competition underscored the need for advanced machine learning models combined with feature engineering techniques to effectively tackle the complexities of big data.

Feature engineering, as a technique for extracting and transforming features from raw data, has become a powerful tool for improving recommendation accuracy in big data applications. Traditional models often struggle with raw data, but by generating meaningful features—such as user behaviour patterns, item metadata, and temporal information—feature engineering can substantially improve a system's predictive performance. Various techniques, including log transformations, normalization, aggregation, and dimensionality reduction, have been used to refine data for machine learning models [16].

Several studies have applied feature engineering in more specialized areas. For instance, R. Lou et al. [18] used natural language processing (NLP) combined with machine learning to detect follow-up recommendations in radiology reports. By extracting key features from textual data, they improved the model's ability to predict follow-up actions. Similarly, in the Internet of Things (IoT) domain, feature engineering has proven critical in handling the massive and diverse datasets generated by IoT systems. Studies have shown that models incorporating advanced feature engineering techniques consistently outperform models that rely solely on raw data [19].

Finding patterns in end-user electrical thefts requires the application of feature engineering. A feature engineering-based approach was suggested by Wei Zhang et al. to extract electricity consumption features from the ensemble and raw data. Features having distinct physical interpretations and a strong correlation with abnormal behaviour are chosen using a machine learning algorithm [31].

In summary, the studies reviewed indicate that recommender systems are not static but rather dynamic and continuously evolving within the limitations of big data. With the increasing complexity and amount of data, basic recommendation models come with inherent problems like scalability, data sparsity, high computational costs among others. In order to address these issues, within the past few years, considerable effort has been made on building distributed, and parallelized solvers, such as Apache Spark, and also optimization methods like feature engineering, dimensionality reduction, and gradient-boost techniques. These developments have made great impacts on the performance and precision of recommendation systems enabling them to function properly even in a large data setting. Nevertheless, there are still unresolved issues especially in the management of complex data types while still maintaining the real-time performance capabilities. This article addresses the importance of feature engineering in the performance optimizations of recommender systems in general and in the context of big datasets in particular and predicts that more research should be undertaken on efficient and scalable systems in the present technologies in the next phase.

Preliminaries

Problem Definition

The discussion of prediction models explained in the context of Big Data in this article. Traditional computing systems have a difficult time dealing with vast and complicated data structures. Let U = u_1, u_2,..., uM and I = i_1, i_2,..., i_N be the sets of users and things, respectively, and each user given his rating on the item is denoted by $r_{u,i}$. Parallel processing produces superior results when compared to a single system, hence resolving the scalability issue. Increased number of comparable users minimizes data sparsity in suggestions. The primary goal of our study is to examine the features of large datasets and to apply feature engineering techniques to them.

Feature Engineering

Feature engineering is the practice of reusing existing features to create new ones. There are several approaches for creating features, including imputation, binning, log transform, one-hot encoding, grouping, feature splitting, scaling, and data extraction. In our work, we generate features in three categories based on the supplied data. To begin, statistical characteristics are formed using statistical computations, and item-related features are derived from statistical features. Thirdly, user-related characteristics are generated from user-related characteristics. Mean and Standard Deviation are used to calculate the statistical features. Similarly, feature engineering approaches such as Feature Split and Binning are used to construct item-related features. Upper Confidence Level (UCL) and Lower Confidence Level (LCL) are used to build user-related features (LCL). Table 1 illustrates how statistical measurements such as mean and standard deviation are chosen and applied to the user's assessment.

Algorithm 1	***Confidence Level Calculation***
1	Gather the sample data
2	Calculate the sample mean $\bar{x}$
3	Determine whether a population's standard deviation is known or unknown
4	If a population's standard deviation is known, we can use a z-score for the corresponding confidence level
5	If a population's standard deviation is unknown, we can use a t-statistic for the corresponding confidence level.
6	Find the lower and upper bounds of the confidence interval using the following formulas
7	$Lower\ bound = \bar{X} - t \times \frac{s}{\sqrt{n}}$ $Upper\ bound = \bar{X} + t \times \frac{s}{\sqrt{n}}$

Table 1

Algorithm 2: Spark Jobs	
Input: RDD containing pre-processed data	
Output: Prediction results	
1	Apply the feature engineering technique to the RDD
2	Assemble the vector using Spark's VectorAssembler
3	Split the dataset into training and testing sets using random split
4	Train a Gradient Boosted Trees (GBT) Regression Model on the training data.
5	Generate predictions on the test data
6	Output the final prediction results

Table 2

Feature Name	Technique	Notation	Description
S_1	Mean	$\{R_{u,m}>=\}$	User ratings on movie greater than or equal to mean Ratings
S_2	Standard Deviation	$\{R_{u,m}>=\}$	User ratings on movie greater than or equal to standard deviation of Ratings

Table 1: Statistical Features.

Feature Name	Technique	Notation	Description
I_1	Feature Splitting	$M\{y(n,y)\}$	Find the release year and title
I_2	Binning	$M\{G(g)\}$	Reduce the number of genres

Table 2: Item related Features.

Table 2 shows the features are derived from the item-related information. Two features namely, I_1, I_2, are created using feature splitting and binning techniques.

Feature Name	Technique	Notation	Description
U_1	Lower Confidence Level	$LCL\{R\}$	The lower Confidence level of the user on rating
U_2	Upper Confidence Level	$UCL\{R\}$	Upper confidence Level of the user on rating

Table 3: User related Features.

The two user-related features are created using user-related information and are depicted in Table 3. The confidence levels of users are calculated and applied in the existing features.

A confidence interval is a statistical term that refers to an estimate of an interval that may contain a population parameter [20]. Confidence Levels are determined using sample data and are expressed as sample means. Calculate the standard deviation and the corresponding z-score. The unknown population parameter is determined by calculating a sample parameter from the collected data. For instance, the population mean is determined by the sample mean x. In most cases, the interval is defined by its bottom and upper boundaries. Confidence intervals are stated in percentage terms (the most frequently quoted percentages are 90 percent, 95 percent, and 99 percent). The percentage indicates the degree of confidence and the steps in the process depicted in Algorithm 1.

Big data Framework

There are tools available for configuring recommendations in the context of big data. Big Data is a massive collection of data sets, measured in zettabytes, that originates from a range of sources [21]. Among the several big data tools available, Apache spark supports data analytics, machine learning methods, and cluster computing.

Proposed Methodology

In this section, we present our model for movie recommendations through big data tools and techniques. In this proposed model, the gradient boost regression algorithm is chosen to handle the large data using Apache Spark. Figure 1 shows the different stages of the proposed model. (i) Apache spark (ii) Preprocessing, (iii) Feature Engineering, (iv) Model Creation (v) Prediction.

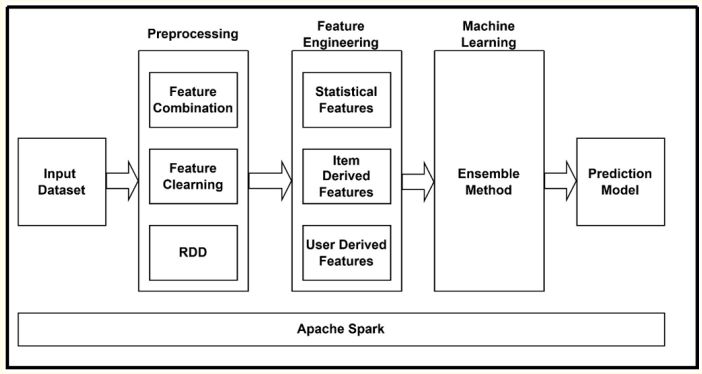

Figure 1

Apache Spark

Cluster computing frameworks have grown in popularity over the previous decade due to their outstanding solutions for processing large amounts of data. Map Reduce [22] and Dryad [23] are two popular cluster computing frameworks for speeding up parallel calculations. However, they have one limitation: they are inefficient at reusing intermediate results over multiple calculations. Reusing intermediate results is a critical stage in the development of many algorithms, including machine learning and graph algorithms [24]. The techniques listed above are finding new applications in areas such as K-means clustering, Page Rank, and logistic regression. Iterative algorithms and interactive data mining tools are inefficient without in-memory computations. Spark's RDDs concept was motivated by these two types of applications [25]. Apache Spark is a platform for cluster computing that enables the management of Resilient Distributed Datasets (RDDs). The processing of tasks represented by a Directed Acyclic Graph (DAG) is expanded from a bipartite MapReduce paradigm using Spark pipelines. When compared to standard Hadoop, the primary advantage of Spark is the way it handles intermediate calculations [25]. At first, Spark reads data from a distributed storage system and caches it as persisted RDDs in local RAM. As a result, Spark effectively utilizes RAM to store intermediate processing results, hence optimizing performance by dramatically lowering overhead. For calculations, Spark requires a cluster manager and a distributed storage system. At the moment, Spark supports three distinct cluster managers: Standalone, Apache Mesos, and Hadoop YARN [26, 27].

The driver software is critical because, in addition to its primary responsibilities, it also generates the RDD with all of its parallel processes (transformations and actions). Additionally, the driver application is responsible for running all of the aforementioned parallel activities on a cluster [25].

Preprocessing

Several steps should be followed while building any machine learning model, more so when the problem requires working with a large data set. Preprocessing is one of the most important one, aimed at improving the quality of data as well as allowing a better performance of the model. Many times, the raw datasets are unclean, incomplete, noisy and even inconsistent, all of which if not taken care of could result in adverse consequences. Preprocessing is designed to deal with such issues using a number of approaches such as data cleaning, data integration, data reduction, data transformation and discretization of data.

Missing values constitute incomplete data for an attribute whereas noisy data is that which contains values that are inaccurate or are extreme. Illogical or contradictory data results from non-uniformity in names or coding. Hence, it can be concluded that data preprocessing is the imperative process in data mining and it embraces many techniques. Dimensionality reduction decreases the number

of features in a dataset, enabling faster computations while preserving essential information. Numerosity reduction minimizes data volume through techniques like clustering or sampling. Data compression reduces file sizes while maintaining the core essence of the data. Data transformation converts raw data into suitable formats for analysis, such as normalization or aggregation. Lastly, discretization simplifies analysis by dividing continuous data into discrete segments, making patterns easier to interpret.

In this dataset, we concentrated on two preprocessing activities, namely feature combination and feature cleaning. Enhancing features and eliminating inconsistencies prepared the datasets for the training of the model.

Resilient Distributed Dataset (RDD)

In the Apache Spark ecosystem, parallel programming is implemented using two key abstractions: Resilient Distributed Datasets (RDDs) and parallel operations on RDDs. An RDD is an immutable collection of objects that can be divided across multiple nodes of a cluster and processed in-memory for faster computation. This data structure allows Spark to perform distributed computing efficiently.

RDDs offer the advantages of distributed shared memory without suffering from the associated latency. There are two main types of operations on RDDs:

i. ***Transformations***: Operations like map, filter, and group by that define a new RDD based on the old one. These are lazy, meaning they don't immediately execute until an action is called.
ii. ***Actions***: Operations like count, collect, and save that trigger the execution of transformations.

In this work, RDDs were used to partition and process large datasets in parallel, ensuring the efficiency and scalability of the recommendation system.

Feature Engineering

Feature engineering is one of the most important steps in building machine learning models, as it helps convert raw data into meaningful input for algorithms. In this study, we employed various techniques to extract and generate features that enhance the prediction accuracy of our model. Common feature engineering methods include, imputation, binning, log transform, one hot encoding, grouping, feature splitting and scaling. We generated three types of features in our experiment:

i. ***Statistical Features***: Features derived from basic statistical calculations such as mean, median, and variance.
ii. ***Item-Related Features***: Features related to the properties of the items (e.g., movies) being recommended.
iii. ***User-Related Features***: Features derived from the relationships between users and items, such as user preferences and behaviours.

Feature engineering is quite possibly among the most critical activities whenever constructing machine learning models as it enables the transformation of unclean raw data into relevant input suitable for the algorithms. In this research work, we adopted some methods to extract and create additional features that improve the accuracy of predictions made by the model. Typical feature engineering practices involve, completing data points that are missing called imputation, shifting from continuous variables to categorical variables, called binning, restructuring the values due to skewed data, named as log transform, combines like features into one, called as grouping, by taking one feature and dividing it into parts in order to evaluate each of the parts, called feature splitting and scaling is another type where enhancing the model by bringing all the features to the same level.

Three main elements of expansion features were demonstrated in the conducted experimental work:

i. ***Statistical Features***: Features that can be computed using some standard statistics such as average, mode, and standard deviation.

ii. ***Item-Related Features***: Looking at the items that are recommended (in this case - movies).

iii. ***User-Related Features***: Concerning and related to the interaction of users and items, their preferences and actions respectively.

Model Selection

Based on our prior work, we identified that ensemble methods tend to deliver higher accuracy compared to other algorithms [5]. Therefore, we selected an ensemble technique, specifically Gradient Boosting for our experiment. Ensemble methods combine multiple base models to create a single, highly accurate predictive model.

Gradient Boosting is a machine learning technique that iteratively builds an ensemble of weak learners, often decision trees, to improve prediction performance. By gradually correcting errors made by earlier trees, Gradient Boosting is highly effective for both classification and regression tasks. The flow of our proposed prediction model is shown in Algorithm 2.

Experiments

To evaluate the proposed methodologies, we conducted experiments using three real-world datasets. In this section, we describe the datasets, experimental setup, evaluation metrics, and results.

Dataset

We used three actual datasets from the Movielens repository: Movielens 1M, 10M, and 20M. Each of these datasets consists of user ratings of movies as well as several items and users related metadata. The datasets were compiled by the GroupLens Research Group based at the University of Minnesota. The Movielens 1 million dataset consists of over 1,000,000 ratings by 6,040 users on 3,900 movies such that every user has at least rated 20 movies on a scale of one to five with, when rated, five being the highest.

The two other datasets Movielens 10M and 20M are scalable up to 10 million and 20 million ratings respectively and they also provide room for improvement in the evaluation of the system due to the depth of challenge in the recommendation system one users spatial. The primary highlights of these datasets are that they are the right practical test beds for evaluating the performance of recommendation systems because of their realistic characteristics and degree of sparsity of data contained within them.

Experimental Setup

For our implementation, 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. The methods were implemented using the following frameworks: Apache Spark 3.4.3, and Scala-3.3.

Evaluation Metrics

Prediction is the process of forecasting the dependent variable using the independent variables. There are different prediction measures like MAE, RMSE, and MSE. It is defined as the difference between the prediction of a rating of user u on item $i(p_{u,i})$ and the real rating of user u on item $i(r_{u,i})$). The calculation of MAE is shown in Equation 1.

$$MAE = \frac{1}{N}\sum_{u,i} | p_{u,i} - r_{u,i} | \quad (1)$$

The mean squared error tells you how close a regression line is to a set of points. It is the average of a set of errors which is shown in Equation 2. The squaring is necessary to remove any negative signs.

$$MSE = \sum_{i=1}^{n} \frac{\left(y_i - \overline{y_i}\right)^2}{n} \quad (2)$$

Equation 3 shows root mean squared error, which is the standard deviation of the residuals. Residuals are measures of how far from the regression line data points are plotted.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{\left(y_i - \overline{y_i}\right)^2}{n}} \quad (3)$$

Experimental results

The experiment is carried out in 16 CPU cores, 32 GB RAM, SSD, environment. To evaluate the performance and scalability of our proposed recommendation system, we used three versions of the MovieLens dataset: the 1M, 10M, and 20M datasets. These datasets are widely used in recommendation system research and contain extensive information about user preferences, item (movie) metadata, and various interaction features. Each dataset contains ratings provided by users for movies, along with additional metadata such as genres, timestamps, and external identifiers like IMDb and TMDb IDs. The user-related and item-related features are stored in separate files, and the datasets vary in size to test how well the system scales. The experiment is carried out in 16 CPU cores, 32 GB RAM with SSD.

- ***MovieLens 1M***: Contains 1,000,209 ratings from 6,040 users across 3,900 movies.
- ***MovieLens 10M***: Contains 10,000,054 ratings from 69,878 users across 10,681 movies.
- ***MovieLens 20M***: Contains 20,000,263 ratings from 138,493 users across 27,278 movies.

After performing exploratory data analysis (EDA) to understand the distributions, missing values, and relationships in the dataset, we integrated all features into a single dataset. The integrated dataset includes user interactions with movies (ratings), movie metadata (genres, IMDb and TMDb IDs), and additional features like tags and relevance scores, as shown in Table 4. This combined dataset was then loaded into the big data environment for processing and analysis using Apache Spark.

userId	*movieId*	*rating*	*timestamp*	*genres*	*imdbId*	*tmdbId*	*tagId*	*relevance*	*tag*	*title_count*
76480	165	3.0	836488713	45	112864	1572.0	190	0.02725	6	1

Table 4: Sample Dataset in Big data Environment.

The sample dataset contains columns representing the userId (the unique identifier for a user), movieId (the unique identifier for a movie), the rating provided by the user, the timestamp of the rating, as well as additional metadata such as genres, imdbId, tmdbId (for cross-referencing with external databases), and tag-related features like tagId, relevance, and title_count. These features are critical for building a comprehensive recommendation model that can accurately predict user preferences.

This integrated dataset allows us to handle complex relationships and perform advanced feature engineering techniques, leading to improved prediction accuracy when recommending movies to users in the big data environment. The dataset is also well-suited for testing the system's scalability and performance with larger data volumes, as represented by the 10M and 20M datasets.

In the preprocessing step, the vector assembler is used to transform the dataset by consolidating the independent variables (or features) into a single vector, while the dependent variable (rating) remains separate. This transformation is essential for efficiently applying machine learning algorithms in a big data environment, as it organizes the data into a structured format suitable for model training and evaluation.

Each row in the Features column is a vector of combined features that include various user, item, and statistical characteristics. These vectors represent the assembled independent variables for each data point (user-movie interaction), and the corresponding Rating column shows the dependent variable (the rating provided by the user) shown in Table 5. As described in Section 3, feature engineering techniques were applied to enhance the dataset by generating new features based on existing information.

Features	*rating*
[52349.0,502.0,6....]	3.0
[46663.0,502.0,0....]	3.0
[116590.0,502.0,1...]	2.0

Table 5: Vector Assembling.

The dataset is split into training and test sets with an 80/20 ratio respectively, further the training samples are given as input to the feature engineering step. During the step, the training data is combined with set of additional features which includes statistical, item and user related features. In the final step, data after the feature engineering is fed as an input to the ensemble method. In case, if the feature engineering is carried out on training data, the number of features are more while comparing with the number of features in the test data. So, the trained model can not fit for testing purpose.

Experiments and Evaluation

The feature-engineered dataset was applied to the MovieLens 1M, 10M, and 20M datasets, following the same procedure for all datasets. The aim was to evaluate the model's prediction accuracy after the application of feature engineering techniques and compare the results before and after feature engineering in Table 6.

Measures	*MAE 1M*	*MAE 10 M*	*MAE 20 M*
All Features	0.8010	0.8488	0.8054
Statistical FE	0.6156	0.5574	0.5737
Item Related FE	0.8009	0.8038	0.8072
User Related FE	0.4295	0.4284	0.4170

Table 6: MAE Values.

The results show that the application of feature engineering techniques significantly improved the model's performance across all three datasets. The Mean Absolute Error (MAE), which measures the average magnitude of prediction errors, was reduced after feature engineering, with user-related features contributing the most to the improvement. These results validate the effectiveness of feature engineering in enhancing prediction accuracy, particularly in large-scale data environments.

The experiments highlight that the proposed system is capable of handling vast amounts of data while improving the quality of recommendations. By utilizing techniques such as vector assembling, feature engineering, and cross-validation, we were able to achieve a more scalable and accurate recommendation system for big data applications.

The experiment was repeated using different categories of feature engineering techniques, and the Root Mean Squared Error (RMSE) values were recorded across the MovieLens 1M, 10M, and 20M datasets. Table 7 summarizes the RMSE values for the overall feature set, as well as the statistical, item-related, and user-related feature engineering (FE) approaches.

Measures	*RMSE 1M*	*RMSE 10 M*	*RMSE 20 M*
All Features	0.9926	1.0219	1.0514
Statistical FE	0.7258	0.6539	0.6867
Item Related FE	0.9932	0.9975	1.0175
User Related FE	0.4295	0.4710	0.4708

Table 7: RMSE Values.

It shows that feature engineering plays a major role and low prediction error values. The results on these input datasets give consistent results and also lower values. Now our model is compared with state of the art technologies. The average of our experiment results compared with the following two papers. Table 8 shows our proposed model compared with the state of the art technologies.

The results indicate that user-related feature engineering achieved the lowest RMSE across all three datasets, demonstrating the significant impact of user behaviour data on prediction accuracy. The overall feature set, which includes all categories of feature engineering, produced slightly higher RMSE values, but it still performed competitively.

The results also highlight that feature engineering plays a crucial role in reducing prediction errors. By improving how the data is structured and extracting meaningful insights from user interactions, the model consistently produces lower error rates. Comparison with State-of-the-Art Technologies is followed.

- To further validate the effectiveness of the proposed model, the results were compared with two state-of-the-art approaches from the literature:
- ***APRA***: A collaborative filtering-based recommendation approach that emphasizes scalability in big data environments.
- ***DPM***: A distributed prediction model that uses a combination of collaborative filtering and big data techniques to address data sparsity and scalability issues.

Table 8 presents the Mean Absolute Error (MAE) values for the MovieLens 10M and 20M datasets for these methods, compared with our BDMFE (Big Data Model with Feature Engineering).

Method	***MAE 10M***	***MAE 20 M***
APRA (AitHammou et al., 2018)	0.6677	0.6557
DPM(BadrAitHammou et al.,2019)	0.6591	0.6490
BDMFE (Proposed)	0.5965	0.5993

Table 8: Performance Comparison on Movielens.

The results demonstrate that the BDMFE model outperforms both the APRA and DPM models, with significantly lower MAE values on both the 10M and 20M datasets. This shows that the integration of feature engineering techniques with collaborative filtering improves the predictive accuracy and scalability of the recommendation system in big data environments.

The reduced MAE values in our model confirm the efficiency of our feature engineering approach, especially in handling the complexities of large datasets while minimizing computational costs. The experiment results, combined with the performance comparison, clearly establish our model as an effective and scalable recommendation solution for big data.

Discussion

The objective of this work was to propose a novel distributed prediction model, BDMFE (Big Data Model with Feature Engineering), utilizing Gradient Boosted Trees (GBT) alongside feature engineering techniques. The model effectively leverages the capabilities of Apache Spark to enhance performance in a Big Data environment.

As demonstrated in Figure 2, the proposed model, when enhanced with feature engineering techniques in various datasets, significantly improves prediction quality. Notably, the BDMFE model utilizing user-related features outperforms the DPM (Distributed Prediction Model) and APRA (Adaptive Probabilistic Recommendation Algorithm), achieving lower error rates. This improvement can be attributed to the careful crafting of features, including statistical, user, and item characteristics, which collectively enhance the accuracy of the prediction model.

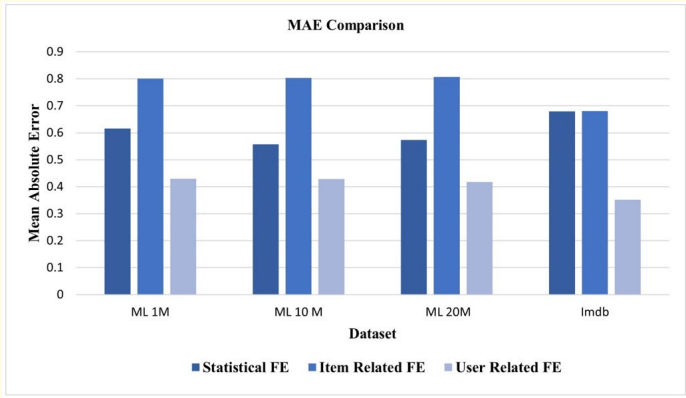

Figure 2

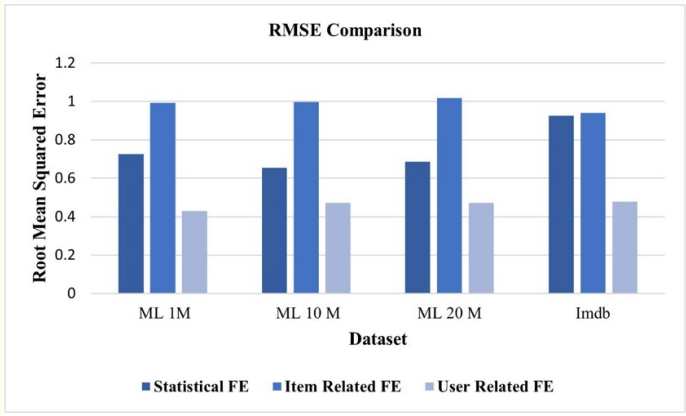

Figure 3

In Figure 3, the Root Mean Squared Error (RMSE) values further corroborate the superior performance of the BDMFE model. The lower RMSE observed with user-related features indicates that the predicted values are closer to the actual regression line for the various datasets [29, 30]. This reinforces the efficacy of our feature engineering techniques in yielding more reliable predictions. In the Movielens and Imdb datasets, the suggested BDMFE model provides a low error rate. To measure the stability of the proposed model, experiment is conducted with other dataset and compared. The model produces consistent findings across the various datasets based on the results obtained which is show in Figure 2 and 3.

Figures 4.a and 4.b illustrate the Mean Absolute Error (MAE) comparisons for the 10M and 20M MovieLens datasets, respectively. Our proposed model consistently demonstrates better performance compared to existing state-of-the-art prediction models. This success underscores the advantages of our approach in enhancing prediction quality, effectively managing large-scale datasets, and mitigating data sparsity challenges.

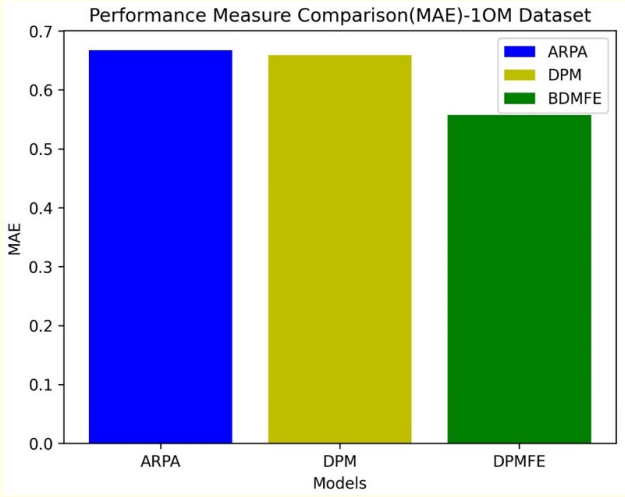

Figure 4.a

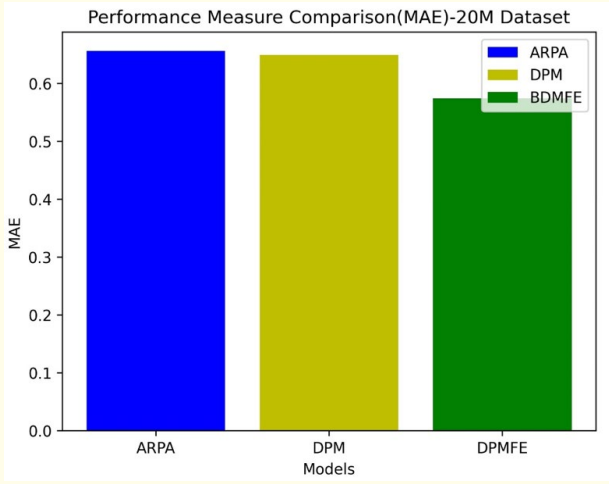

Figure 4.b

Conclusion and Future work

The experimental results demonstrate that our proposed model substantially outperforms existing recommendation methods regarding prediction error. Traditional recommender systems struggle with efficiency when confronted with vast amounts of data. To overcome this limitation, we have designed a feature engineering-based model specifically tailored for the Big Data environment. A critical focus of our work is the scalability issue, which is crucial for maintaining performance in recommendation systems when dealing with large datasets. We implemented feature engineering techniques across three distinct categories: statistical, user-related, and item-related. These engineered features were integrated into our proposed model, which operates seamlessly within a Big Data

platform. The results, summarized in tabulated measures, clearly indicate that the feature engineering strategies significantly contributed to the improved performance of our model in this context. The improvement in accuracy shows that performance complexity is decreased by introducing Big Data Platform and sparsity issues are minimised through the feature engineering.

Looking ahead, future work could explore the development of an efficient hybrid mechanism designed to handle extensive datasets utilizing contemporary tools and techniques. This hybrid approach could integrate multiple recommendation strategies to further enhance prediction accuracy and system robustness. Additionally, we aim to test our model against various large-scale datasets to evaluate its effectiveness in hybrid recommendation scenarios comprehensively. By expanding the scope of our research, we hope to contribute further to the advancement of recommendation systems in the era of Big Data.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Competing Interests

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.

Author Contributions

The study importance and challenges of the recommender system done by Parthasarathy G. Material preparation, data collection, and analysis were performed by Dr. Balaji GN. The first draft of the manuscript was written by Dr.Balaji N. All the authors read and approved the final manuscript.

Data Availability

The data that support the findings of this study are available from the corresponding author, Parthasarathy Govindarajan, upon reasonable request.

References

1. Nair A and Mathews R. “Challenges and Solutions in Recommender Systems”. In: Pandian A., Palanisamy R., Ntalianis K. (eds) Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2019). ICCBI 2019. Lecture Notes on Data Engineering and Communications Technologies 49 (2020).
2. Ricci F., Rokach L and Shapira B.” Recommender Systems: Introduction and Challenges”. Recommender Systems Handbook. Springer, Boston, MA (2015).
3. Charu C Aggarwal. “Recommender Systems” (2016).
4. JA Konstan and J Riedl. “Recommender Systems: from algorithms to user experience”. User Model User-Adapt Interact 22 (2011) 101-123.
5. Sathiya Devi S and Parthasarathy G. “Feature Engineering based Approach for Prediction of Movie Ratings”. I.J. Information Engineering and Electronic Business 6 (2019): 24-31.
6. Murale Narayanan and Aswani Kumar Cherukuri. “A study and analysis of recommendation systems for location-based social network(LBSN) with big data”. IIMB Management Review 28 (2016): 25-30.
7. Xiao long Jin., et al. “Significance and Challenges of Big Data Research”. Journal Big Data Research 2 (2015): 59-64.
8. Patgiri R. “Issues and Challenges in Big Data: A Survey”. In: Negi A., Bhatnagar R., Parida L. (eds) Distributed Computing and Internet Technology. ICDCIT 2018. Lecture Notes in Computer Science (2018): 10722.
9. Hafed Zarzour., et al. “An Improved Collaborative Filtering Recommendation Algorithm for Big Data”. International Federation for Information Processing, AICT 522 (2018): 660-668.

10. Robin Genuer., et al. "Random Forests for Big Data". Jounral of Big Data Research 9 (2017): 28-46.
11. M-Y Hsieh., et al. "A keyword-aware recommender system using implicit feedback on Hadoop". J. Parallel Distrib. Comput (2018).
12. Badr Ait Hammou, Ayoub Ait Lahcen and Salma Mouline. "APRA: An Approximate Parallel Recommendation Algorithm for Big Data". Knowledge-Based Systems (2018).
13. Badr Ait Hammou, Ayoub Ait Lahcen and Salma Mouline. "An effective distributed predictive model with Matrix factorization and random forest for Big Data recommendation systems". Expert Systems with Applications (2019): 253-265.
14. Dahdouh K., et al. "Large-scale e-learning recommender system based on Spark and Hadoop". J Big Data 6.2 (2019).
15. Jianjun Xie., et al. "Feature Engineering in User's Music Preference Prediction" JMLR: Workshop and Conference Proceedings 18 (2012): 183-197.
16. Udayan Khurana, Horst Samulowitz and Deepak Turaga. "Feature Engineering for Predictive Modeling Using Reinforcement Learning". Association for the Advancement of Artificial Intelligence (www.aaai.org) (2018).
17. Soares E. F. de S., et al. "Online travel mode detection method using automated machine learning and feature engineering". Future Generation Computer Systems (2019).
18. Lou R., et al. "Automated Detection of Radiology Reports that Require Follow-up Imaging Using Natural Language Processing Feature Engineering and Machine Learning Classification". Journal of Digital Imaging (2019).
19. Devarshi Shah, Jin Wang and Q Peter He. "Feature Engineering in Big Data Analytics for IoT-Enabled Smart Manufacturing - Comparison between Deep Learning and Statistical Learning". Computers and Chemical Engineering (2020).
20. ITRC (Interstate Technology & Regulatory Council). Groundwater Statistics and Monitoring Compliance, Statistical Tools for the Project Life Cycle. GSMC-1. Washington, D.C.: Interstate Technology & Regulatory Council, Groundwater Statistics and Monitoring Compliance Team (2013).
21. Mccaffrey P. "Overview of big data tools: Hadoop, Spark, and Kafka". An Introduction to Healthcare Informatics (2020): 291-305.
22. M AlJame and I Ahmad. "DNA short read alignment on apache spark". Applied Computing and Informatics.
23. M Isard., et al. "Dryad: distributed data-parallel programs from sequential building blocks, ACM SIGOPS operating systems review". ACM 41 (2007): 59-72.
24. M Zaharia., et al. "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing". in: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, USENIX Association (2012): 2-2.
25. K Hwang. "Cloud Computing for Machine Learning and Cognitive Applications: A Machine Learning Approach". MIT Press (2017).
26. B Hindman., et al. "Mesos: a platform for fine-grained resource sharing in the data center". NSDI 11 (2011): 22-22.
27. VK Vavilapalli., et al. "Apache Hadoop YARN: yet another resource negotiator". in: Proceedings of the 4th annual Symposium on Cloud Computing, ACM (2013): 5.
28. M Zaharia., et al. "Stoica, Spark: cluster computing with working sets". Hot Cloud 10 (2010): 95.
29. https://www.kaggle.com/datasets/imdb-movies-dataset
30. Maxwell Harper and Joseph A Konstan. "The MovieLens Datasets: History and Context". ACM Transactions on Interactive Intelligent Systems (TiiS) (2015).
31. Wei Zhang and Yu Dai. "A multiscale electricity theft detection model based on feature engineering". Big Data Research 36 (2024).

Volume 9 Issue 2 August 2025