# Parallel Implementation Of Recommendation System For Social Media Using Machine Learning and Deep Learning.

M. Aljabri , F. Alnasser, J. Mustafa, R. Alzahrani, N. Almaswdy, N. Alshehri, and S.Almatrafi

***Abstract*— Social media platforms collects large amounts of data related to user interactions, providing a resource for creating personalized content recommendations. Such recommendations are integral to keeping users engaged to these platforms. Even though, recommendation algorithms often encounter difficulties, especially when dealing with the complexities of social networks. Parallel computing comes up as a solution, enabling the concurrent execution of multiple tasks by using the power of multiple processors. Prior research in the field of recommendation systems has mostly focused on collaborative filtering and deep learning algorithms approaches offering various solutions to enhance recommendation system performance. However, a notable gap is the limited exploration of Support Vector Machines (SVM) together with parallel computing techniques to further boost recommendation system performance. Our study aims to present the benefits of parallel computing by improving the performance of recommendation systems through training and prediction processes. Furthermore, the code used in this study shows the application of parallelism through multiprocessing and uses (SVM) algorithm which is well-known for its ability to solve classification problems and make accurate predictions in a wide range of scenarios to achieve recommendation results. Overall, this approach achieved improved efficiency results and speedup.**

***Index Terms*— recommendation system; SVM; Parallel execution; Social Networks; Multiprocessing**

## I. INTRODUCTION

In recent times , as sensor technology, storage technology, computer technology, and network technology have rapidly advanced, the amount of data being generated has significantly increased . [1]. However as the quantity of data grows, individuals are confronted with the challenge of dealing with an overwhelming amount of information, leading to increased difficulty in making right decisions . This occurrence is commonly referred to as information overload. [2] . To solve the problem of information overload, recommender system emerges as the times require. Its main idea is to analyze the user's historical behavior and preference information, establish a model, and automatically recommend the items or products of interest to the user, then get a personalized list for the user [3]. The proposed recommendation system is based on information which the user gave to the system in the past. The given information may have many ratings which show that the aim of the user is to get information from a particular domain—e.g., research area, documents, tweets, etc. Based on the recommendation system architecture, the given information can be used as training data, either supervised learning or unsupervised learning—e.g., clustering or document classification problems [4,5].
The increasing demand for faster recommendation systems in social media has prompted the need for efficient parallel algorithms to process data and provide quick recommendations to users based on their interests. In this research, we aim to address the challenges posed by the increasing volume of data in social media platforms and develop an effective parallel algorithm for recommendation systems. Traditional sequential algorithms are not suitable for processing large data sets in a timely manner. Hence, there is a need for a recommendation

First M. Aljabri is with the Department of Computer Science and Artificial Intelligence, College of Computers, Umm Al-Qura University, Makkah 21955, Saudi Arabia.

Second F. Alnasser is with the Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia

Third J. Mustafa is with the Department of Computer Science and Artificial Intelligence, College of Computers, Umm Al-Qura University, Makkah 21955, Saudi Arabia.

Fourth R. Alzahrani is with the Department of Computer Science and Artificial Intelligence, College of Computers, Umm Al-Qura University, Makkah 21955, Saudi Arabia.

Fifth N. Almaswdy is with the Department of Computer Science and Artificial Intelligence, College of Computers, Umm Al-Qura University, Makkah 21955, Saudi Arabia.

Sixth N. Alshehri is with the Department of Computer Science and Artificial Intelligence, College of Computers, Umm Al-Qura University, Makkah 21955, Saudi Arabia.

Seventh S.Almatrafi is with the Department of Computer Science and Artificial Intelligence, College of Computers, Umm Al-Qura University, Makkah 21955, Saudi Arabia.

system that can quickly process data and provide timely recommendations to users based on their interests. Previous works in the field of recommender systems have made significant contributions. However, most existing systems still face difficulties in providing scalable and timely recommendations. These limitations affect the user experience and may result in a delay in discovering content. Therefore, there is a clear gap in research on developing efficient and scalable recommendation algorithms that can handle large data sets effectively. In this paper, we propose an efficient parallel algorithm for recommendation systems in social media. Our approach aims to enhance the performance of recommender systems and reduce data processing time. We achieve this by leveraging the power of parallel computing and using a support vector machine (SVM) model. The proposed algorithm uses the SVM model as a machine learning algorithm to classify textual data from the dataset. We convert the serial recommender system algorithm into a parallel algorithm by using the Parallel_backend function, which enables parallel processing using a multiprocessing backend with different numbers of parallel jobs. An SVM model is trained using a training dataset, where text features are mapped and used to predict the test dataset. To process the dataset, we use Term Inverse Document Frequency (TF-IDF) technology, which converts textual data into a digital representation, which helps in efficient processing. The main motivation behind this research is to address the limitations of existing recommender systems in terms of efficiently processing large data sets. By using parallel computing techniques, we aim to reduce training and prediction times, ensuring that the recommendation model can be updated and retrained quickly. This approach enables realtime predictions and scalability to handle comprehensive datasets without compromising performance. By improving content retrieval and recommendation processes, we can provide users with personalized and valuable recommendations that match their interests and preferences in real time. The remainder of this work is organized as follows. Section 2 contains a review of the relevant literature. Section 3 Methodology and describes the experimental studies, including a description of the data set, the experimental setup. Section 5 presents the results and discussion, while Section 6 conclusion of the paper .

## II. Literature Review

Recommendation systems play an essential role in personalized content recommendations and keeping users engaged and returning for more on social media platforms. However traditional recommendation algorithms often face challenges particularly when dealing with social networks. The reason behind this is their need to process amounts of data and comprehend relationships between users and content, evolving with innovations in machine learning and deep learning. This literature review aims to explore the intersection of recommender systems and parallel computing. Moreover, the review unveils emerging trends, novel techniques, and applications while identifying research gaps by analyzing a wide range of papers. Consequently, this review helps identify existing gaps in the field. Additionally, this literature review covers recent research from 2019 to 2020, while providing crucial insights for researchers and practitioners. Notably, it emphasizes the critical importance of using parallelism in recommendation systems for scalability and efficiency, showcasing its impact and paving the way for future research.

Most of the studies we reviewed presented recommendation systems with a focus on either collaborative filtering or deep learning algorithms. Firstly, Tao et al. [6] proposed a Spark-based algorithm called the collaborative filtering recommendation algorithm. The paper focuses on the issue of accurately and efficiently delivering information to users. This challenge arises due to the increasing volume of data available. However, the paper primarily employed parallelism through the Apache Spark distributed platform to enhance the efficiency of the collaborative filtering recommendation algorithm based on Alternating Least Squares (ALS) and focused on leveraging Spark for distributed computation. In terms of the experimental setup, the researchers use the MovieLens dataset provided by the GroupLens Lab. They implement their recommendation algorithm on both Spark distributed clusters and a single node. Techniques involved likely encompass collaborative filtering, ALS, item similarity, and cold start strategies. The paper demonstrated three tests to evaluate the improved algorithm which resulted in the following, the first experiment showed that Root Mean Squared Error RMSE decreased with larger data sizes. Meanwhile, the second experiment showed that the improved algorithm had better prediction accuracy than the traditional algorithms. Next, experiment showed that Spark-based distributed computing was more efficient as the data size increased. In conclusion, parallel recommendation algorithm on Spark, significantly improves efficiency and recommendation accuracy. Moving on, the future research direction identified in the paper involves combining implicit feedback with larger datasets to enhance the quality of recommendation systems.

Simultaneously, Fu et al.[7], presented a novel deep learning approach to enhance collaborative filtering recommendation systems. The method uses neural network to generate predictions from pre-learned embedding of users and items. However, this approach addresses the existing or used collaborative filtering based Algorithm which can capture a single type of relation. Moreover, the model represents two major stages: representation learning and training the prediction neural networks. In the representation learning phase, the model captures inter-relations between user-user and item-item matrices. It does so through the constraint model (CM) for global co-occurrence and the rating independent model (RIM) for local co-occurrence. In the second stage, neural networks are trained to predict user-item interactions effectively from the pre-learned or pre-trained CM and RIM models. For evaluation, experiments utilized MovieLens datasets, with performance assessed using root mean square error (RMSE). Comparing the proposed multiview neural networks with existing models revealed superior performance, indicating higher prediction accuracy. Overall, the results show that the RMSE for the multiview neural networks was smaller which means more accuracy of the prediction. distinctly, the multiview neural

networks can be more efficient then the models used nowadays in collaborative filtering recommendation systems. Finally, The study highlights key directions for future research, including the exploration of end-to-end neural networks, consideration of content information, and improvements in training efficiency. Using the same data sources as [6] and [7] Shen Jian et al. [8] introduced a movie recommendation algorithm that can be applied to types of e commerce. The paper primarily focused on addressing the challenges of movie recommendations by utilizing users past behavior data to understand their preferences and provide them movie suggestions. Specifically they implemented a recommendation algorithm based on item based filtering methods using Java language on an Ubuntu system, Hadoop model, MapReduce framework and the MovieLens dataset which contain 1000 row . By using the capabilities of Hadoops distributed file system and MapReduce the algorithm was able to store and process amounts of data and normalize the results obtained so as to increase the accuracy of recommendation resulting in improved performance and faster response times. as for future work the researchers suggested conducting studies to enhance the effectiveness of their recommendations.

Next, Tahmasbi, et al. [9], introduced the tscmf model which enhances collaborative filtering recommender systems used deep learning techniques to captured temporal dynamics and social influence. The model combines collective matrix factorization with a transition matrix to effectively model user preference dynamics over time. Also to optimize the tscmf model parallel computing techniques such as multi-core processing and multiple threads were utilized to improved efficiency and speed. Moreover, the paper presents an efficient optimization algorithm based on stochastic gradient descent. Furthermore, the performance of the tscmf model was evaluated used the "Epinions" dataset which was real-world data demonstrating superior recommendation accuracy. Finally, the authors suggested future improvements including scaling the model for larger datasets and exploring its applicability in different domains.

Lastly, Christos Sardianos et al. [10] propose a distributed, parallel framework to enhance the scalability and performance of collaborative filtering (CF) algorithms. They utilize the SVD++ algorithm for CF and apply graph partitioning for distributed processing. Custom edge selection strategies are employed to refine the partitions. The study uses a small subset of the Epinions product ratings dataset, consisting of 50,000 ratings from 9,435 users and 16,288 items. Evaluation is done through cross-validation and RMSE analysis. The results demonstrate the effectiveness of the proposed techniques in improving the scalability and accuracy of parallel CF algorithms. The study provides insights into optimizing parallel CF algorithms on partitioned graphs for improved scalability and accuracy, while also acknowledging the issue of sparsity in resulting partitions and recommending strategies to reduce it through edge replication.

TABLE 1
SUMMARY OF THE REVIEWED FINDINGS

| Reference No. | Year | Paper Name | Algorithm | Dataset | Performance measures |
|---|---|---|---|---|---|
| [6] | 2019 | Collaborative Filtering Recommendation Algorithm based on Spark | Collaborative Filtering Recommendation Algorithm | MoveLens | Accuracy, Efficiency |
| [7] | 2020 | A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System | for learning representation CM-RIC for prediction: -multiview neural networks | MovieLens 1M dataset MovieLens 10M dataset | RMSE |
| [8] | 2020 | Collaborative filtering-based recommendation system for big data | MapReduce framework. and an item-based collaborative filtering algorithm | MovieLens | accuracy, response speed |
| [9] | 2020 | TSCMF: Temporal and social collective matrix factorization model for recommender systems | TSCMF | Real world | Recommendatio accuracy, scalability |
| [10] | 2019 | Optimizing parallel collaborative filtering approaches for improving recommendation system performance | -SVD++ for collaborative filtering -Graph partitioning for distributed processing - Custom edge selection strategies to refine partitions - Cross validation and RMSE for evaluation | Used a small subset of the full Epinions product ratings dataset with 50,000 ratings between 9,435 users and 16,288 items. | - Precision Computational Efficiency |

## GAP ANALYSIS

In recommendation systems, the efficient processing of large datasets for content recommendation remains a significant challenge. Most existing systems encounter difficulties in providing timely and scalable recommendations. This limitation hinders the user experience and may lead to delays in content discovery. In our study we aim to demonstrate the benefits of parallel computing in the context of

recommendation systems. We specifically focused on First, Parallel Training, it leverages parallel computing (multiprocessing) to train a Support Vector Machine (SVM) model efficiently. By distributing the computational load, it significantly reduces training time, ensuring that the recommendation model can be updated and retrained swiftly. Second, Real-time Predictions, parallel computing enables faster prediction of recommendations. Third, Scalability, it showcases how parallelism can be applied to scale recommendation systems, allowing for the processing of extensive datasets without compromising performance. Finally, Optimized Content Retrieval, by accelerating the process of content retrieval and recommendation, we can enhance the user experience by delivering personalized and valuable recommendations that align with users' interests and preferences in real time.

## III. Methodology

These days, many social media applications have been used by users so the need for a faster recommendation system in social media to process the data and make a quick recommendation to users based on their interest. So, applications need an efficient parallel algorithm of recommendation systems to face the challenges of speed in processing the data. The Traditional sequential algorithms are not suitable for a large range of datasets because it will take a lot of time to process data.

In general, The proposal of this research paper is an efficient parallel algorithm of recommendation systems in social media using SVM model to enhance the performance of recommendation systems in social media, and reduce the time in processing the data in the recommendation system. So, in this research paper we convert the sequential algorithm of recommendation system in the social media to an efficient parallel algorithm by using SVM model to make the recommendation system faster in processing the data.

### *A. Dataset collection method*

This research paper used secondary dataset found on [11] and added some modifications on the dataset. Moreover, the data was qualitative data which refers to non-numerical data because the data in social media is represented as text in most situations. So, the algorithm needed to analyze the context of the post to recommend similar posts to the users. However, the algorithm needed categories to filter the recommendation for users, this research paper considered 17 categories for data such as: art, science, banking, drawing, GST, HR Management, dance, programming, business, zoology, mathematics, politics, graphic, craft, political, operating system, photography. However, 20% of the dataset is for training the SVM model and the rest for testing the SVM model. The dataset is loaded and set up by using the pandas library.

### *B. SVM model*

In this research paper, the proposed algorithm used a SVM model as a machine learning algorithm which was used to classify the texts from the dataset. Moreover, this research paper found a sequential code work as a recommendation system and converted it into parallel code by using parallel_backend function which is responsible for backend parallel processing that works using multiprocessing backend with different numbers of parallel jobs. However, the algorithm has two stages: First, Train the SVM Model: the SVM model received A 20% data from the Dataset to train the model and mapping the text features. Sconed, Testing the SVM model: the SVM model now can predict from the testing dataset which is about 80% data from the Dataset and map the input text features to predict closest posts based on the similarities of titles in the dataset.Also, this research paper used the term frequency - inverse document frequency (TF-IDF) which is used to convert the text data to numerical representation and help in processing the dataset.

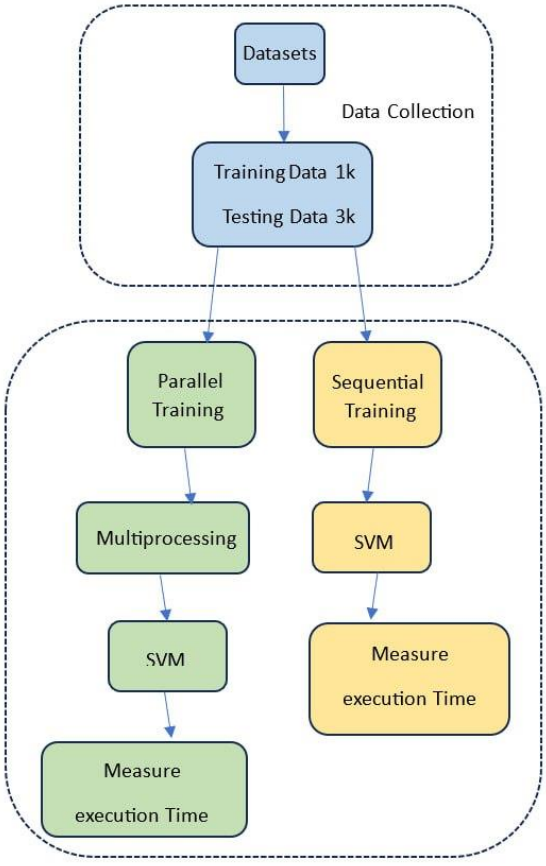


Fig. 1. Main Phases Diagram

TABLE 2
SVM MODEL TRAINING AND TESTING COMPONENTS

| Step | Description |
|---|---|
| 1. Import Libraries | Import necessary libraries. |
| 2. Load Dataset | Load the dataset from CSV file for training and testing. |
| 3. Split Data | Split dataset into training and testing. |
| 4. Feature Engineering | Extract and preprocess text data, using techniques such as TF-IDF vectorization |
| 5. Initialize SVM Model | Initialize the SVM model, and specify parameters (kernel). |
| 6. Train SVM Model | Train the SVM model using the Model training dataset and the chosen features. |
| 7. Test SVM Model | Use the testing dataset to evaluate the SVM model's performance |
| 8. Evaluate Performance | Calculate total execution time for training and testing. |

## IV. RESULTS AND DISCUSSION

### *A. Experimental Setup*

To perform the explained experiment, we used Python version 3 on Google Colab notebook platform. The device used has a Windows 11 Home 64-bit operating system. Moreover, the device has core i7 of the 11th generation, which means it works with four cores and eight logical processors. The dataset we worked on was balanced and allowed us to work with 3 K training data, and 1 K testing data. Furthermore, after utilizing the TF-IDF vectorization technique to convert text data into numerical features. We start to train and test the SVM model as shown in Table 2. The detailed settings of parameters applied for the model are shown in Table 3. Eight experiments were conducted in total using SVM algorithm on (Sequential code) and (2, 3, 4, 5, 6, 7, 8) cores on parallel code. The sequential part was implemented on the given hardware and recorded the execution time. Followed by the next ten experiments based on parallel computing by multiprocessing using Python parallel backend and number of jobs with n_jobs that equals different number of cores each time, from 2 to 8 cores manually updated. results are discussed below.

TABLE 3
SVM PARAMETERS

| Model | Parameter | Value |
|---|---|---|
| SVM | kernel | linear |

### *B. Results*

After making use of the parallel techniques described in the methodology section, the efficiency, execution time, and speedup we have measured the Efficiency and the speedup of the multi-processing from 2-8 cores check Table 4, 5, and 6. The results varied and changed between sequential and parallel for each number of cores as shown in the Fig.2 below and the following table illustrates a summary of the evaluation results obtained. Speedup is a measure of how much faster the computation is in parallel compared to a single processor. The highest speedup is achieved with four processors, indicating a substantial performance boost. However, as the number of processors increases beyond four, the speedup drops off, suggesting diminishing returns and potential overhead. Efficiency, which measures how well processors are utilized, is highest with two processors but decreases with more processors, indicating underutilization. The lowest performance is observed with six processors, where speedup is significantly less than 1. These results illustrate the trade-off between adding processors for parallel processing and ensuring their efficient utilization.

TABLE 4
SEQUENTIAL EXPERIMENTS

| Training execution time | Testing execution time |
|---|---|
| 0.0382695198059082 | 0.19955134391784668 |

TABLE 5
PARALLEL EXPERIMENTS.1

| Cores | Execution time | |
|---|---|---|
| | Training | Testing |
| 2 | 0.032033443 | 0.164228439 |
| 3 | 0.065299749 | 0.191431046 |
| 4 | 0.031348944 | 0.159118652 |
| 5 | 0.031821251 | 0.168299198 |
| 6 | 0.062337399 | 0.198587179 |
| 7 | 0.040993214 | 0.158982277 |
| 8 | 0.039781809 | 0.157906532 |

TABLE 6
PARALLEL EXPERIMENTS.2

| Cores | Efficiency | Speedup |
|---|---|---|
| 2 | 0.597337 | 1.194674 |
| 3 | 0.195353 | 0.586058 |
| 4 | 0.305185 | 1.220741 |
| 5 | 0.240527 | 1.202633 |
| 6 | 0.010232 | 0.061391 |
| 7 | 0.133364 | 0.933549 |
| 8 | 0.120249 | 0.961992 |

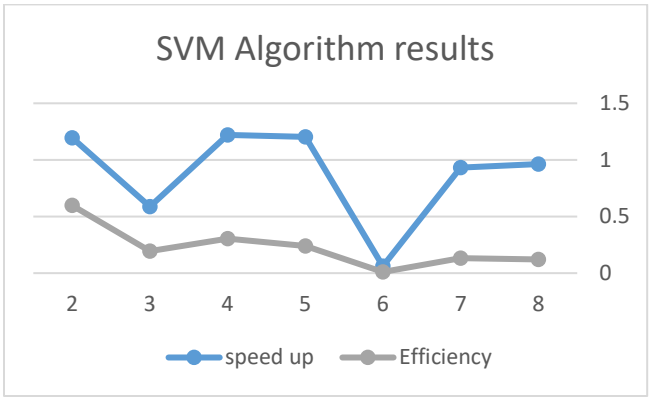


Fig. 2. Speed up and Efficiency Results

Moreover, we also calculated the execution times for each of training and testing with different processor counts. As shown in Fig.3 for training data using four processors shows the most efficient performance with the shortest training time at around 0.0313 units. Two and eight processors maintain efficiency, with training times at approximately 0.0320 and 0.0398 units, respectively. However, a six processors doubles the training time to 0.0623 units. And seven processors increase the time to around 0.0410 units, while five processors maintain efficiency at 0.0318 units, making them the second fastest configuration after four processors.


Fig. 3. Training time over several number of cores

Next, the results for testing data was as shown in Fig.4 when utilizing eight processors, the testing time is the most efficient at approximately 0.157 units. Similarly, configurations with two, four, and five processors maintain efficiency, with testing times at around 0.164, 0.1591, and 0.1683 units, respectively. Surprisingly, using six processors nearly doubles the testing time to approximately 0.1986 units. With seven processors, the testing time slightly increases to around 0.1590 units, positioning it as the second fastest configuration after eight processors. These results emphasize the trade-off between processor counts and efficient utilization for optimizing testing time.

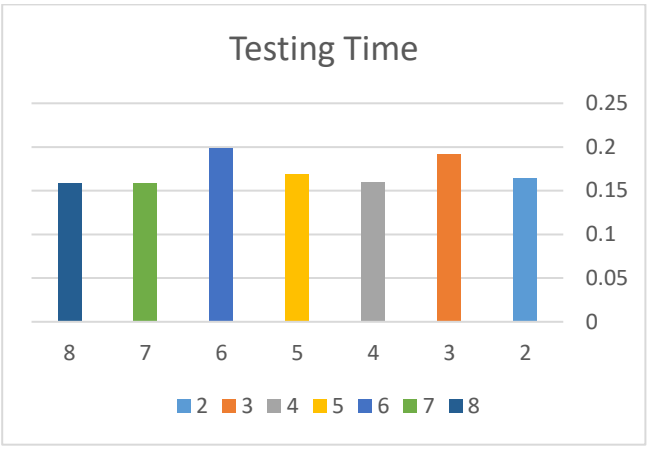


Fig. 4. Testing time over several number of cores

## V. Conclusion

In summary, our research has demonstrated the effectiveness of solving the efficiency problem by developing a parallel recommendation system that processes large datasets of content and provides timely and scalable recommendations, thus enhancing the user experience by delivering personalized and valuable recommendations that align with users' interests and preferences in real time. However, it is important to acknowledge its limitations. Which was the sample size being relatively small, limiting the generalizability of the findings. Future research should aim to include a larger and more diverse sample to enhance the validity of the results. Despite this limitation, our study contributes to the existing body of knowledge and offers valuable output into the Parallel Implementation Of Recommendation systems for Social Media Using Machine Learning, laying the groundwork for future research to address these limitations and expand upon our findings.

## References

[1] N. Marz and J. Warren, Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Greenwich, CT, USA: Manning Publications, 2015.
[2] J. Gantz and D. Reinsel, ‘‘The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east,’’ in Proc. IDC iView, IDC Analyze Future, 2012, pp. 1–16.
[3] H. Wang, N. Wang, and D.-Y. Yeung, ‘‘Collaborative deep learning for recommender systems,’’ in Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Sydney, NSW, Australia, 2015, pp. 1235–1244.
[4]Pornwattanavichai, A.; Jirachanchaisiri, P.; Kitsupapaisan, J.; Maneeroj, S. Enhanced Tweet Hybrid Recommender System Using Unsupervised Topic Modeling and Matrix Factorization-Based Neural Network. In Supervised and Unsupervised Learning for Data Science; Springer: Berlin/Heidelberg, Germany, 2020; pp. 121–143. 3.

[5]Yan, L.; Liu, Y. An Ensemble Prediction Model for Potential Student Recommendation Using Machine Learning. Symmetry 2020, 12, 728.

[6] Tao, J., Gan, J., & Wen, B. (2019). Collaborative Filtering Recommendation Algorithm based on Spark. Int J Performability Eng. http://www.ijpe-online.com/EN/10.23940/ijpe.19.03.p22.930938.

[7] M. Fu, H. Qu, Z. Yi, L. Lu and Y. Liu, "A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System," in IEEE Transactions on Cybernetics, vol. 49, no. 3, pp. 1084-1096, March 2019, doi: 10.1109/TCYB.2018.2795041. https://ieeexplore.ieee.org/document/8279660

[8] Jian Shen, Tianqi Zhou, and Lina Chen , Collaborative filtering-based recommendation system for big data ,International Journal of Computational Science and Engineering 2020 21:2, 219-225 https://doi.org/10.1504/IJCSE.2020.105727

[9] Tahmasbi, H., Jalali, M., & Shakeri, H. (2020). TSCMF: Temporal and social collective matrix factorization model for recommender systems. Journal of Intelligent Information Systems, 1-24. DOI: 10.1007/s10844-020-00613-w. https://link.springer.com/article/10.1007/s10844-020-00613-w

[10] Christos Sardianos, Grigorios Ballas Papadatos and Iraklis Varlamis. (2019). Optimizing Parallel Collaborative Filtering Approaches for Improving Recommendation Systems Performance. Information 2019, 10(5), 155. https://doi.org/10.3390/info10050155

[11]dataset https://www.kaggle.com/datasets/vatsalparsaniya/post-pecommendation/data