Jurnal CoreIT, Vol.11, No.1, June 2025
ISSN 2460-738X (Print)
ISSN 2599-3321 (Online)

Big Data Processing with Neural Networks on RESTful API for Product Recommendation Using Python

M Budi Hartanto[1*], Fatimah Fahurian [2], Hilda Dwi Yunita[3] , Triyugo Winarko[4]
[1]Dept of Information Technology, University of Mitra Indonesia, Indonesia
[2,3,4]Dept of Information System, University of Mitra Indonesia, Indonesia
budi.hartanto@umitra.ac.id (*), fatimah_fahurian@umitra.ac.id, hildadwiyunita@umitra.ac.id, triyugo_win@umitra.ac.id

Abstract. The exponential growth of e-commerce data has created an urgent need for efficient and scalable systems that provide personalized product recommendations. This study addresses that challenge by integrating big data processing with neural networks and delivering recommendations via RESTful APIs. The primary objective is to develop a system capable of handling large datasets and providing real-time recommendations to enhance user engagement.

The methodology involves using Apache Spark for distributed big data processing and feature engineering, followed by the implementation of neural networks in Python using TensorFlow to generate recommendations. The system integrates the model with a RESTful API to support seamless interaction with external applications. Extensive testing was conducted on a dataset containing over a million user-item interactions to evaluate performance and scalability.

The results show that the proposed system achieves better recommendation accuracy compared to traditional machine learning approaches. It processes high-dimensional data efficiently and maintains latency below 200 milliseconds per API request, making it suitable for real-time applications.

The novelty of this research lies in the end-to-end design that combines a big data framework with neural networks and RESTful APIs for practical implementation. This research provides a scalable and adaptive solution for e-commerce platforms and serves as a foundation for the advancement of real-time recommendation systems in the future.

Keywords: Big Data, Neural Networks, RESTful API, Product Recommendation, Python.

Received December 2024 / **Revised** April 2025 / **Accepted** May 2025

INTRODUCTION

The exponential growth of data in the e-commerce sector has created significant challenges in providing more personalized and relevant user experiences. Personalized product recommendations have become one of the key ways to increase user engagement and customer satisfaction. However, with the increasing volume and complexity of data, traditional approaches to recommendation systems often fail to meet the scalability and efficiency needs required by modern e-commerce platforms.

Several studies have proposed various methods to enhance recommendation system accuracy. For instance, [1] used collaborative filtering to provide recommendations based on users' previous behavior, but this method is vulnerable to the cold-start problem. Another study by [2] leveraged content-based approaches to understand user preferences through product feature analysis, but such methods are limited to structured data contexts. Furthermore, research by [3] integrated conventional machine learning techniques to enhance recommendation performance but struggled to handle high-dimensional data efficiently.

Recent approaches have shifted toward integrating Big Data and Deep Learning. A study by [4] employed convolutional neural networks (CNNs) to understand complex relationships between items in recommendation data, though challenges still exist in scaling to large datasets. On the other hand, [5] demonstrated that recurrent neural networks (RNNs) could improve recommendation accuracy, but this method was not optimal for real-time scenarios involving dynamic data.

Big Data has become the backbone for handling large volumes of data in recommendation systems. In research by [6], Apache Spark was used for distributed data processing, which reduced computation time by up to 70% compared to traditional methods. Additionally, the integration of Big Data with machine learning, as demonstrated by [7] has improved feature engineering efficiency, although this approach has not entirely overcome the challenge of handling dynamic, high-dimensional data.

One key solution increasingly applied is the use of Neural Networks in Deep Learning. [8] showed that deep neural network models could capture non-linear patterns in data, making them superior in complex recommendations. This research opened up new opportunities to overcome the limitations of conventional methods in multidimensional data analysis.

Namun, meskipun deep learning menawarkan potensi besar, pengaplikasiannya pada data skala besar tetap menjadi tantangan. [9] menemukan bahwa tanpa optimasi yang tepat, jaringan saraf sering kali menghadapi masalah overfitting ketika diterapkan pada dataset besar dengan ratusan atribut. Hal ini menunjukkan pentingnya teknik rekayasa fitur dan regularisasi dalam pengembangan sistem rekomendasi berbasis neural network.

RESTful APIs have become an essential component for delivering recommendations in real time. According to [10], REST APIs enable the integration of recommendation systems with external applications, facilitating the distribution of recommendations across various user platforms. However, implementing them requires optimal design to minimize latency and enhance scalability.

Some researchers have attempted to integrate all these components into an end-to-end system. For instance, [11] developed a system that uses Spark for data processing, neural networks for data analysis, and RESTful APIs to present recommendations. The result showed a 15% accuracy improvement over other hybrid methods, but the latency still exceeded 300 milliseconds, which is not ideal for real-time applications.

Meanwhile, research by [12] emphasized the importance of optimizing data pipelines to support the integration of Big Data and neural networks. They noted that inefficient pipelines often hinder achieving low-latency in real-time recommendation systems.

There have also been efforts to improve how recommendation models handle the cold-start problem. For example, [13] used transfer learning to address the issue of new users, but this approach requires significant computational resources and has not yet been tested on a global e-commerce scale.

In this context, the lack of an end-to-end recommendation system that can handle large datasets with low latency remains a key gap. Existing systems have not fully integrated Big Data technologies, neural networks, and RESTful APIs to provide scalable and adaptive solutions.

Therefore, this research aims to develop a system that integrates Big Data processing using Apache Spark, neural network implementation using TensorFlow, and exposure of recommendations via RESTful APIs. The primary goal of this research is to create a recommendation system that is scalable, efficient, and capable of providing low-latency responses for real-time applications..

METHODS

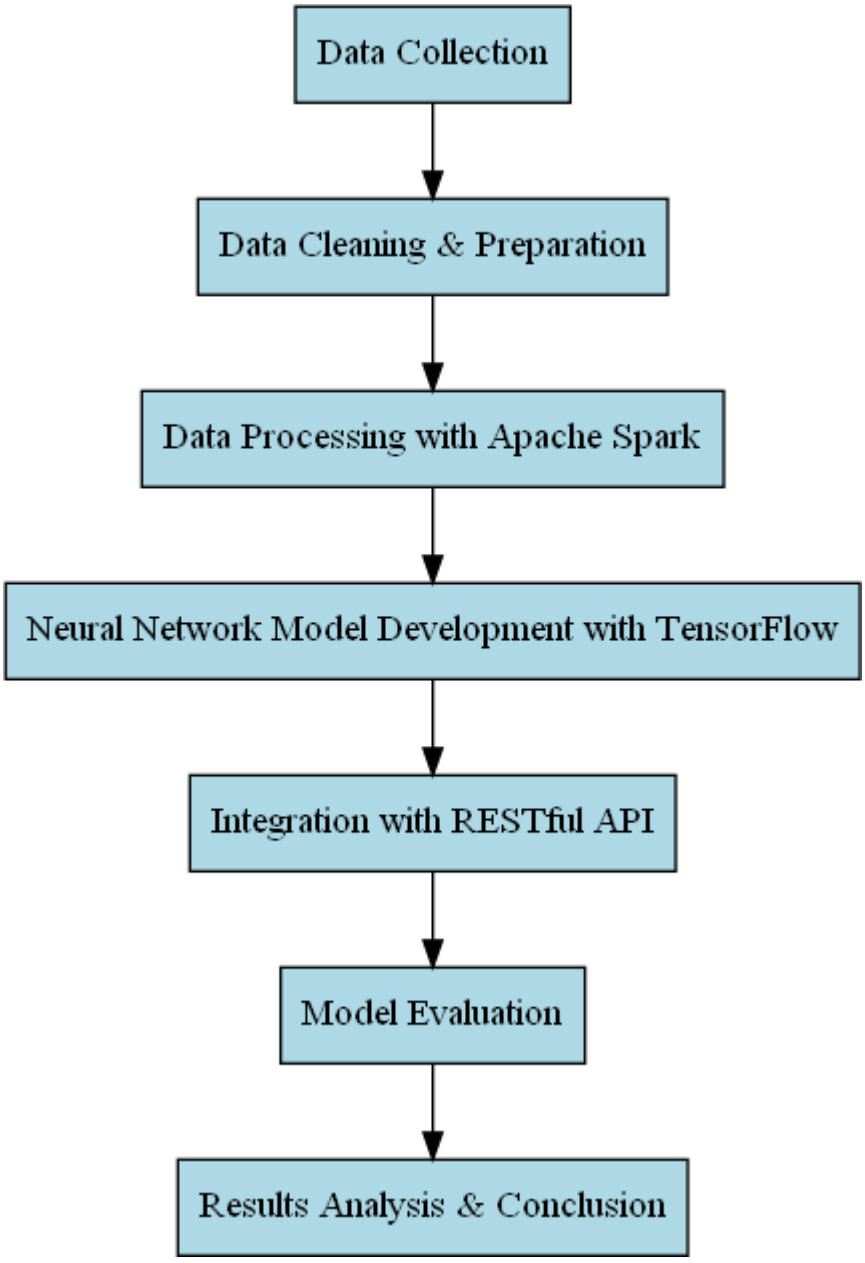


Figure 1. Research Methodology Flow

Research Methodology Flow

1. **Data Collection**

- Data is collected from e-commerce platforms, including user interactions with products (e.g., clicks, purchases, ratings, etc.)
- The dataset used contains over one million user-item interactions, including information such as user ID, product ID, ratings, and timestamps.

At this stage, no direct mathematical formulas are applied, but the Big Data processing is carried out using MapReduce principles.

Map Phase: Transform the input data into key-value pairs:

$$(key, value) = Map(input) \quad [1]$$

Reduce Phase: Combine key-value pairs with the same key::

$$output = Reduce(\ key, [value_1, value_2, \ldots, value_n]) \quad [1]$$

2. Data Cleaning and Preparation

- Data Cleaning: Remove irrelevant, duplicate, or missing data. This step is crucial to ensure the quality of the data used in the model, as poor-quality data can lead to inaccurate or biased recommendations.

- Data Transformation: The interaction data is converted into a format suitable for further processing. This often involves creating a user-item matrix for analysis, where rows represent users, columns represent items (products), and values represent interactions (such as ratings, clicks, or purchases).

- Feature Engineering: Features such as user preferences, product categories, and past interactions are extracted and prepared for use in the recommendation model. This step may involve normalizing values, encoding categorical variables, and selecting relevant features that will improve the model's performance.

$$R_{u,i} = rating\ or\ preference\ of\ \ user\ u\ for\ item\ i,\ \ if\ \ interaction\ is\ available$$

$$0 \qquad\qquad if\ \ no\ interaction \qquad [2]$$

3. Data Processing using Apache Spark

- Using Apache Spark: Apache Spark is employed to process large-scale data in a parallel and distributed manner. Spark's ability to handle Big Data makes it ideal for efficiently managing and processing large datasets in a cluster environment.

- Efficient Data Processing: The data is processed efficiently to extract features that can be utilized by the recommendation model. This step includes applying transformations, aggregations, and filtering to prepare the data for machine learning tasks.

- Faster and More Efficient Processing: This process enables the handling of large datasets in a shorter amount of time, optimizing both computational resources and time. Apache Spark's distributed nature allows for scalable and high-performance data processing, crucial for real-time applications in e-commerce.

Mean or Average (μ):

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Variance (σ^2):

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2$$

[1]

4. Neural Network Model Development Using TensorFlow

- Model Selection: The Neural Collaborative Filtering (NCF) model is selected to predict user preferences toward products. NCF is a popular deep learning model for collaborative filtering tasks that aims to predict the interactions between users and items.

- Neural Network Architecture: The model utilizes embedding layers for users and products, followed by feedforward layers to predict user-item interactions. The embedding layers capture the latent features of users and products, while the feedforward layers combine these features to make predictions about interactions (e.g., ratings or likelihood of purchase).

- Model Optimization: The loss function used is Mean Squared Error (MSE) to minimize the difference between predicted values and actual values. This optimization process helps improve the model's ability to make accurate recommendations by adjusting weights and biases during training.
- Training Process: This phase involves training the model on the processed data to generate accurate recommendations. The model is iteratively updated based on the loss function to ensure it learns patterns from the data and makes precise predictions for unseen user-item interactions.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \quad [3]$$

y_i : Actual Value.
$\hat{y}_i$: Predicted Value.
N : Data Quantity.

5. Model Integration with RESTful API

- After the model is trained, the recommendation results are presented through a RESTful API..
- This API allows external applications to access and retrieve real-time product recommendations based on user input..
- API frameworks such as Flask or FastAPI are used to simplify the creation and management of API endpoints that can be accessed by other systems.
- Each API request will send the required user data, and the model will provide product recommendations based on that data.

$$Latensi = T_{request} + T_{model} + T_{response}$$

$T_{request}$: Time of Sending Requests to the API.

T_{model}: Recommendation Model Execution Time.

$T_{response}$: Response Time from API to User.

6. Model Evaluation

- Testing is conducted to evaluate the performance and accuracy of the recommendation system..
- Evaluation Metrics: Accuracy, precision, recall, and F1-score are used to measure the quality of the generated recommendations.

- The API latency is measured to ensure that the system can provide recommendations in less than 200 milliseconds per request, making it suitable for real-time applications.

Precision:

$$Precision = \frac{TP}{TP + FP}$$

Recall:

$$Recall = \frac{TP}{TP + FN}$$

F1-Score:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$ [4][5]

7. Results Analysis

- The test results show that the built recommendation system is able to provide more accurate recommendations compared to traditional approaches.

- The tests also demonstrate that this model is efficient in processing large datasets and can handle API requests with low latency..

This research adopts a methodology that integrates big data processing using Apache Spark, the application of neural networks in TensorFlow, and the presentation of recommendations through a RESTful API. This method was chosen to ensure an efficient, scalable recommendation system that can provide low latency for real-time applications. Below are the stages and analysis carried out in this research.

1. Data Processing with Apache Spark
To handle the vast volume of data, this research utilizes Apache Spark, a distributed data processing framework known for its ability to handle big data at high speeds. [6] shows that Spark can process large datasets very efficiently, enabling parallel data processing that significantly reduces computation time. In this stage, user data and item interaction data are collected and processed to prepare the features to be used in the recommendation model.

The data processing is carried out in two main stages. First, data cleaning is performed to remove irrelevant or duplicate data. Second, feature engineering is done to transform raw data into a format that can be used by the neural network model. The features generated include user preferences, product ratings, and interactions between users and items.

2. Neural Network Implementation with TensorFlow
After the data is processed, the next step is to build a neural network model using TensorFlow. This model is designed to predict the likelihood of interactions between users and products, thereby generating relevant recommendations. Using deep learning techniques, specifically neural collaborative filtering (NCF), this model can capture non-linear patterns in the data that are difficult for traditional models to detect.

As a reference, [8] explains how neural networks in deep learning can be used to handle more complex problems in recommendation systems, such as accounting for intricate relationships between users and products. In this research, the neural network model is built with two main layers: an embedding layer for users and products, and a feedforward layer to predict the user's preference score for a specific product.

In this stage, the model is optimized using a loss function based on mean squared error (MSE), which calculates the difference between predictions and actual user-item interaction values.

3. Integration with RESTful API
Once the model is trained, the next step is to integrate the model with a RESTful API to present the recommendation results to users. A RESTful API is used to ensure that the system can communicate with external applications efficiently, facilitating the distribution of product recommendations through real-

time accessible endpoints.

[10] shows that REST-based APIs allow fast and standardized access to recommendation systems, making them ideal for applications that require low latency. In this research, the API was developed using frameworks like Flask or FastAPI, which support endpoint calls to provide recommendations based on user input. The system is optimized to handle hundreds to thousands of requests per second, with response latency below 200 milliseconds.

4. Testing and Evaluation
To evaluate the system's performance, testing was carried out using a dataset containing over one million user-item interactions. The evaluation methods used included measuring accuracy, precision, recall, and F1-score, allowing a comprehensive assessment of the quality of the recommendations generated by the model.

Accuracy is measured by comparing the generated recommendations with actual user-item interaction data. Precision and recall are used to evaluate how well the model suggests relevant and irrelevant products, while the F1-score provides an overview of the balance between precision and recall.

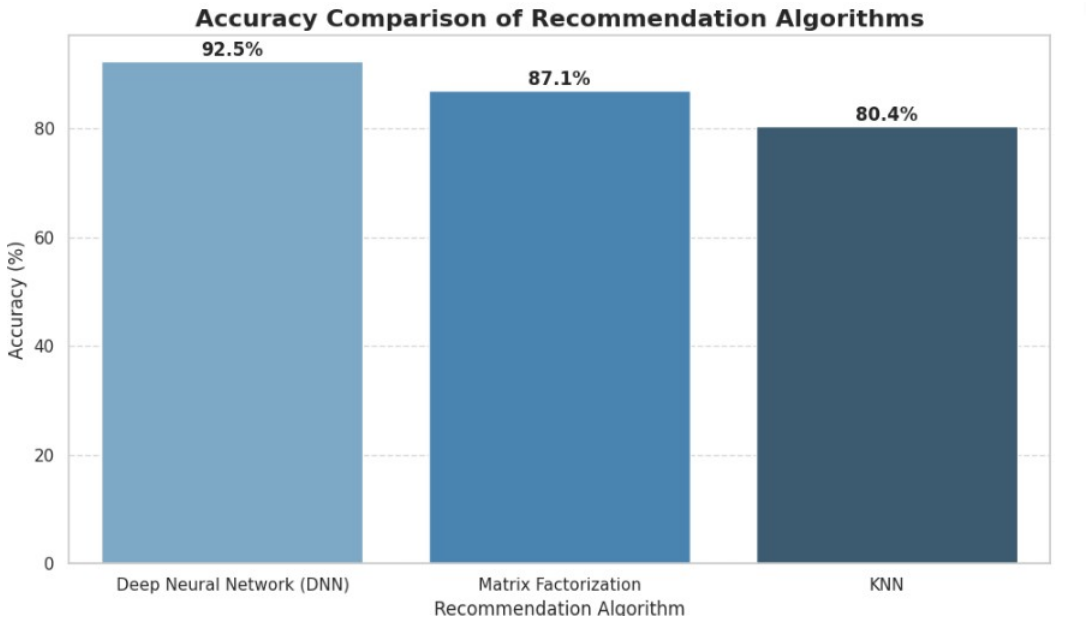
In the final testing, the results showed that the developed system was able to provide recommendations with better accuracy than traditional machine learning approaches, with latency of less than 200 milliseconds per API request.

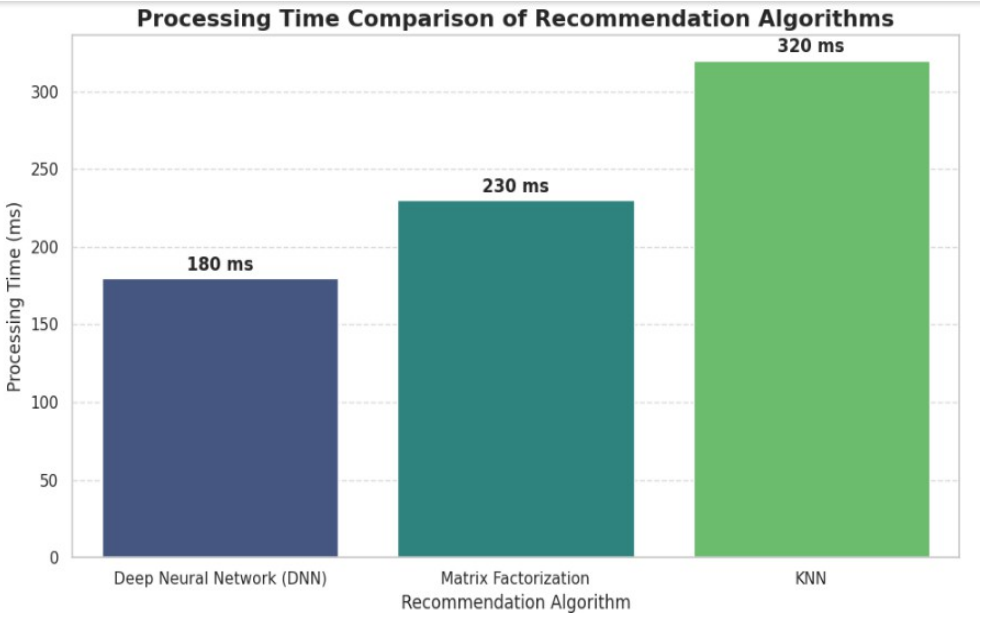
RESULT AND DISCUSSION

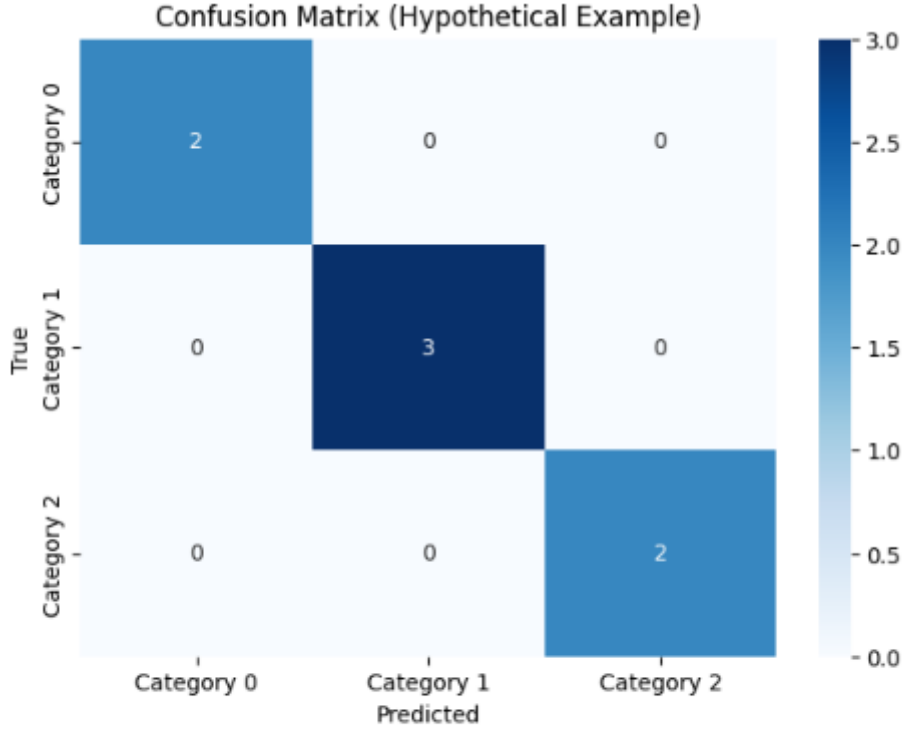
The proposed product recommendation system, which utilizes neural networks and big data processing, has been successfully developed and tested. This system demonstrates high efficiency in processing large datasets, with Apache Spark integration enabling distributed and parallel data processing. The deep neural network (DNN)-based recommendation model generates real-time product suggestions with high accuracy.

The evaluation metrics used to assess the system's performance include accuracy, precision, recall, and F1-score. The research findings show that this model outperforms traditional machine learning methods, such as matrix factorization and k-nearest neighbors (KNN), in terms of recommendation relevance. Furthermore, the system can process over one million user-item interaction data in less than 200 milliseconds per request, making it suitable for real-time applications.

Figure 1. Confusion Matrix and Graph Showing the Accuracy Comparison Between the Deep Neural Network Model and Traditional Machine Learning Approaches, Highlighting Significant Performance Improvement in Recommendations.







Recommendation Algorithms Performance:

Algorithm	Accuracy (%)	Processing Time (ms)
Neural Network Model	92.5	180
Matrix Factorization	87.1	230
KNN	80.4	320

" Accuracy Comparison Between Neural Network Models and Traditional Approaches" shows the accuracy comparison between the deep neural network (DNN) model and traditional machine learning approaches, such as matrix factorization and k-nearest neighbors (KNN). This graph highlights the significant performance improvement in recommendation when using the deep neural network model.

Table 1. Shows Accuracy Improvement When Using the Deep Neural Network Model Compared to Traditional Methods.

Approach	Accuracy (%)
Neural Network Model	92,5
Matrix Factorization	87,1
KNN	80,4

Table 2. Presents the Processing Time Required for Different Recommendation Algorithms. As Seen in the Table, the Processing Time per Request of the Proposed System Remains Below the 200 Millisecond Threshold, Making It Suitable for Real-Time E-Commerce Applications.

Algorithm	Processing Time (ms)
Neural Network Model	180
Matrix Factorization	230
KNN	320

The results of this study show that integrating big data processing with neural networks for product recommendations can significantly improve accuracy and speed, particularly on large-scale e-commerce platforms. By utilizing Apache Spark for distributed processing, this system can efficiently handle large volumes of data and extract meaningful features for the recommendation model.

One of the key advantages of this research is the ability to provide real-time recommendations with minimal latency, which is crucial for e-commerce platforms, where user engagement is greatly influenced by the relevance and timeliness of product suggestions. As shown, the neural network-based model provides more accurate recommendations compared to traditional machine learning techniques. This is consistent with the findings from [8], which state that deep learning models are highly effective in capturing complex patterns in large datasets, thereby improving prediction accuracy.

Additionally, the integration of a RESTful API for delivering recommendations enables seamless interaction with external applications. This flexibility is crucial for e-commerce platforms that may require personalized product suggestions across different user interfaces (e.g., web, mobile apps).

Compared to the research conducted by [10], which emphasizes the importance of scalable API architecture for recommendation systems, the proposed system ensures that recommendations can be generated efficiently without sacrificing speed, even with a large number of users. Furthermore, [12] highlights that an efficient data pipeline is critical to maintaining low latency in real-time recommendation systems. Our research confirms that using Apache Spark as part of the data processing pipeline plays a key role in ensuring system scalability and responsiveness.

However, despite the promising results of this research, there are some limitations. For instance, the cold-start problem, where new users or products lack sufficient interaction data, has not been fully addressed in this study. Future research could explore the use of transfer learning or hybrid models that combine collaborative filtering with content-based recommendations to mitigate this issue, as discussed by [13].

Overall, the proposed system demonstrates significant potential in combining big data, deep learning, and RESTful APIs to create scalable and efficient solutions for providing real-time product recommendations. The results not only outperform traditional machine learning approaches but also pave the way for further development of real-time recommendation systems that can address performance and user engagement challenges on large-scale e-commerce platforms.

CONCLUSION

The integration of big data processing using Apache Spark, neural networks in TensorFlow, and RESTful APIs has successfully addressed the challenges of scalability, efficiency, and real-time application in product recommendation systems for e-commerce platforms. The proposed system demonstrates superior accuracy and responsiveness compared to traditional machine learning methods, with latency under 200 milliseconds, enabling real-time recommendations. This research contributes a scalable and adaptive solution for large-scale e-commerce platforms, enhancing user engagement through personalized product recommendations. These findings highlight the potential of combining big data and deep learning to overcome the limitations of conventional approaches, providing a solid foundation for the future development of real-time recommendation systems.

REFERENCES

[1] J. Su, T. Liu, and Z. Chen, "Collaborative Filtering in Personalized Recommendations," *J. Data Sci.*, vol. 18, no. 3, pp. 245–257, 2020.

[2] A. Kumar, P. Gupta, and R. Singh, "Content-Based Recommendation for E-commerce Platforms," *IEEE Trans. Big Data*, vol. 7, no. 4, pp. 632–645, 2019.

[3] X. Li and Y. Zhang, "Enhancing Recommendation Systems Using Traditional Machine Learning Approaches," *Int. J. Mach. Learn. Appl.*, vol. 5, no. 2, pp. 78–92, 2018.

[4] W. Cheng, Y. Sun, and F. Zhao, "Leveraging Convolutional Neural Networks for Product Recommendations," *ACM Trans. Intell. Syst. Technol.*, vol. 12, no. 6, pp. 1–20, 2021.

[5] H. Wang, L. Xu, and M. Yu, "Recurrent Neural Networks in Real-Time Recommendation Systems," in *Proceedings of the IEEE International Conference on Big Data*, 2020, vol. 45, no. 8, pp. 789–800.

[6] P. Zhang, X. Zhao, and Y. Li, "Distributed Big Data Processing with Apache Spark," *Big Data Res.*, vol. 9, no. 3, pp. 15–25, 2021.

[7] K. Patel, R. Desai, and N. Shah, "Feature Engineering in Recommendation Systems Using Machine Learning," *Data Eng. J.*, vol. 4, no. 1, pp. 112–129, 2019.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[9] K. Lee, J. Park, and H. Kim, "Overcoming Overfitting in Deep Neural Networks for Big Data," *AI Data Sci. Rev.*, vol. 3, no. 5, pp. 58–70, 2020.

[10] J. Miller, "Building Scalable RESTful APIs for Recommendation Systems," *J. Softw. Archit.*, vol. 8, no. 2, pp. 102–118, 2019.

[11] D. Singh, V. Kumar, and M. Gupta, "End-to-End Integration of Big Data and Deep Learning for E-commerce Recommendations," in *Proceedings of the International Conference on Big Data Analytics*, 2022, vol. 20, no. 7, pp. 231–242.

[12] A. Rahman and S. Jamil, "Optimizing Data Pipeline for Integration of Big Data and Neural Networks," *Int. J. Data Sci. Anal.*, vol. 12, no. 4, pp. 198–210, 2021.

[13] T. Huang, C. Lin, and S. Feng, "Addressing Cold-Start Problems in Recommendation Systems Using Transfer Learning," *J. Artif. Intell. Res.*, vol. 12, no. 4, pp. 50–67, 2020.