International Journal on Advanced Science Engineering Information Technology

Vol.14 (2024) No. 6
ISSN: 2088-5334

Generative AI Recommender System in E-Commerce

Nur Anis Nabila Binti Mohd Romzi [a], Su-Cheng Haw [a,*], Wan-Er Kong [a], Heru Agus Santoso [b], Gee-Kok Tong [a]

[a] *Faculty of Computing and Informatics, Multimedia University, Jalan Multimedia, Cyberjaya, Malaysia*
[b] *Department of Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang, Indonesia*
Corresponding author: [*]sucheng@mmu.edu.my

***Abstract*—In today's information-rich world, recommender systems are essential for helping consumers find relevant products and content. The development of efficient recommender systems is still a challenging endeavor even with their broad use. This research explores different approaches to building recommender systems, emphasizing the use of generative AI to overcome underlying difficulties. Conventional recommender systems, like collaborative filtering, struggle with problems like sparsity limitations and the cold start problem. This paper aims to provide a comprehensive overview of recommender system techniques and algorithms, identify the limitations of existing methods, and highlight open research questions and directions for future development. A thorough and comprehensive literature analysis of recommender system algorithms is part of the process, ensuring the validity and reliability of the research. The Autoencoder technique—which has shown to be highly significant and effective—is used for the evaluation. The review will provide a detailed analysis of potential for improving research on recommender systems, while also thoroughly addressing the primary challenges and drawbacks of current methodologies. Furthermore, by providing insights into the usage of Generative AI—more especially, the Autoencoder technique— to improve recommender system accuracy. The study hopes to make a substantial contribution to the area. Through the identification and resolution of current methods' shortcomings, particularly regarding the incorporation of Generative AI, the study endeavors to widen up the opportunity for recommendations that are more precise, varied, and focused on individuals. It is anticipated that the assessment process's use of Autoencoder will highlight the usefulness and efficiency of the suggested strategy and highlight its significance in the continuous development of recommender systems.**

***Keywords*—Machine learning; generative AI; recommendation system; autoencoder; e-commerce.**

Manuscript received 8 Dec. 2023; revised 14 Apr. 2024; accepted 22 Sep. 2024. Date of publication 31 Dec. 2024.

I. INTRODUCTION

The importance of recommendation systems has grown, mainly since websites like YouTube, Amazon, and Netflix emerged. Recommendation systems assess customer preferences and make proactive suggestions for things they are likely to purchase based on product information, customer behavior, and other data. Numerous research investigations have been carried out to create these recommendation systems, and countless valuable systems have been effectively used in a range of industries [1]. Nowadays, the majority of services and products are sold online, making it challenging to build relationships with clients. As a result, these complex algorithms are meticulously designed to offer consumers customized product recommendations. One creative way to get beyond the constraints of e-commerce services is using recommendation algorithms.

Designed to provide relevant and tailored experiences, recommender systems are vital tools for users navigating through the abundance of content available in today's world of information. The rapidly developing field of Generative AI in artificial intelligence is a promising area with the potential to change recommender systems completely. This innovative field of artificial intelligence promises to provide novel aspects to user-centric content recommendations while addressing the shortcomings of conventional recommender systems. Evidence of this transformative technique's success is observable in various platforms such as Amazon, Netflix, healthcare [2], food recommendation [3], and e-commerce [4], where Generative AI has been effectively employed in recommender systems to enhance the precision and personalization of content suggestions [5]. Customers find it very important in the e-commerce version because, for example, generative AI makes it possible to create highly personalized suggestions by evaluating each user's unique behavior and interests [6]. By showing customers products

that closely match their interests, personalization improves the whole shopping experience and helps customers save time and effort throughout their search. Customers are more likely to be happy with their purchases when they obtain recommendations that match their tastes. This optimized process can promote client loyalty and repeat business as consumers gain confidence in the platform's ability to comprehend and meet their demands. By integrating powerful generative AI into recommender systems, e-commerce firms can gain a competitive advantage. In a competitive marketplace, platforms that provide an exceptional, customized purchasing experience are more likely to draw in and keep users. Generative artificial intelligence (AI) in e-commerce recommender systems helps individual customers and makes e-commerce platforms more successful and competitive.

Traditional RS can be broadly grouped into Content-based (CB), collaborative filtering (CF), Hybrid-based (HB), and Knowledge-based (KB). CB solutions are widely utilized in various industries, where they are arguably the most popular approach. A few examples are websites like Google Play Store and Amazon.com. Recommendations supplied by the user, either directly or through interacting with the interface, are generated by a content-based recommender system. Data can be utilized to produce suggestions for the user once the customer profile has been created. As more data sources are provided by the customer or suggested activities are accepted, the engine gets more and more accurate. A content recommendation system often looks at the user's past interests and suggests related items or services.

By leveraging the interests of similar users, CF technologies go above and beyond. Accurate recommendations are made possible using the preferences of individuals with similar tastes. Locating a group of individuals with whom the target user shares interests is the basic tenet of collaborative filtering [7]. The algorithm forecasts the interests of the target user by considering these neighbors' preferences. Amazon is a well-known e-commerce company that employs collaborative filtering to suggest products to its customers. When the algorithm locates a neighbor user for the target user, recommendations can be produced based on the things these neighbors like. This neighboring user group functions as a benchmark for item recommendations, encapsulating the essence of collaborative filtering in finding a user cluster with interests similar to those of the target user. Memory-based approaches operate under the assumption that there is no pre-existing model for predictions. Instead, they rely on choices derived directly from the user-item interaction matrix. This approach looks for similarities between a new user and existing "profiles" by mapping their regular interactions to determine what products to present them. By utilizing the popularity of similar items among users who have comparable experiences with the item, the goal is to anticipate which item will be best for the new user. The concept behind item-item recommendation is to pinpoint items a user might find appealing based on their positive interactions with other items. If most users engaged with two separate items (a product, page, or email) comparably, then the two products are deemed similar. In contrast to user-user recommendation, this approach focuses on finding commonalities in interactions between items in an item matrix as opposed to between users. Therefore, combining many recommender approaches is one of the best ways to increase sales and enhance client retention. Combining the best features of both methods, HB RS adds content information to collaborative models, weights the average recommendation from collaborative and content sources, and generates final recommendations based on the combined rankings. HB RS is the result of combining CB and CF [8].

KB or Semantics-based recommender techniques make up the other group. Context-based and ontology-based methods are among them. Systems are knowledge-based and use ontologies to frame knowledge about stakeholders and material in the recommendation process. For example, e-learning means that these systems use relational knowledge to map learner-relevant learning resources [9].

On the other hand, generative AI Intelligence (GAI) is powered by foundation models (large AI models), enabling them to handle various tasks beyond conventional boundaries seamlessly, including summarization, Q&A, classification, and others. GAI can create customized products to satisfy users' unique information requirements, and the recently released ChatGPT greatly helps users express information demands more precisely through natural language commands. GAI is purposefully designed for content generation and the development of robust recommendation systems. Leveraging supervised learning, the model undergoes a dynamic learning process directly from the data it encounters. Also, implementing embeddings is integral to the system, facilitating the computation of similarity between recommendation embeddings. A higher degree of similarity between these embeddings indicates a closer semantic relationship, which improves the system's capacity to make more contextually relevant recommendations. E-commerce platforms like Amazon employ generative AI algorithms to provide tailored product recommendations that are predicated on consumers' past purchases and browsing patterns.

II. Materials and Method

A. Phases in RS

1) Traditional RS: To give consumers individualized ideas, recommendation systems must be created using several crucial procedures. First, gather data through explicit (such as ratings) and implicit (such as clicks) input. Afterward, enter the learning phase, during which the system uses this information to comprehend user preferences. To do this, profiles that reflect user preferences and objects' features must be created. Lastly, the system applies all the knowledge it has gathered to forecast and recommend products that consumers may find interesting but haven't seen. This procedure enhances user satisfaction by ensuring that recommendations are customized to each user's preferences [10], [11]. The implementations underlying the recommendation phases are depicted in Fig. 1. The Information collection phase requires a robust user profile or model. This first step collects relevant user data, including the user's attributes, behaviors, and the content of the resources they visit, to build a user profile or model for taste prediction. To successfully make recommendations that are appropriate for the user's

preferences, the system requires as much information from the user as possible.

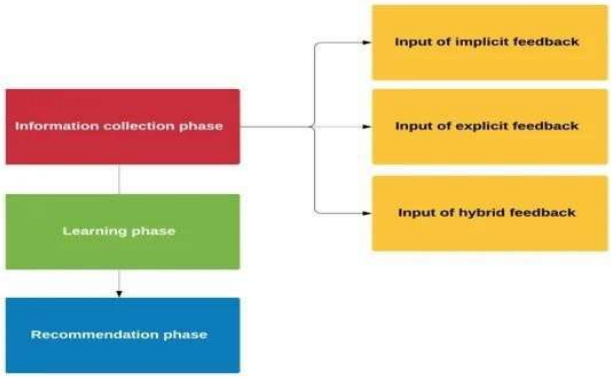

Fig. 1 Phases in RS

Therefore, the model's capacity to represent users' current choices or preferences is a crucial factor in the effectiveness of a recommender system or recommendation. User input data can be gathered in three ways: explicit, implicit, and hybrid feedback. With implicit feedback, the user's preferences are automatically set by the system based on an analysis of their past browsing patterns, purchasing patterns, clicked links, and amount of time spent on various websites. The user doesn't have to do anything; instead, it analyses and gives recommendations on its own, as previously said. This approach is frequently regarded as less precise even if it does not demand the same level of user effort. This input method's benefit is that it makes data collection more accessible and less demanding for the user [12].

Since explicit feedback solicits the user's direct input regarding product preferences, its efficacy depends on accuracy. This input requests ratings for multiple products from users via the system interface, which helps build and improve the recommendation model. The quantity and caliber of these user-provided ratings directly affect the recommendations' accuracy. In other words, recommendations get more precise and well-rounded the more perceptive and numerous opinions are. Since explicit feedback does not involve deriving preferences from actions, it still provides more reliable data even though it necessitates more user work. Additionally, because it offers transparency into the recommendation process, it raises perceived recommendation quality and increases confidence in the recommendations [12].

Additionally, users who wish to show their interest directly can only provide feedback through hybrid feedback, which combines explicit and implicit feedback ratings. This approach empowers users by allowing them to provide input explicitly, emphasizing the importance of their active expression of interest in a choice. Within this hybrid framework, the system adeptly incorporates indirect data as a valuable attribute for recommendation generation, ensuring a comprehensive understanding of user preferences. This feedback can be obtained by letting consumers give direct input and ratings while using indirect data as a recommendation attribute.

In the learning phase, the user data collected during the information-gathering phase is subjected to learning algorithms. This phase's trained data offers specific patterns to predict the user's future actions or interests. During the suggestion stage, the learning algorithms assist in identifying the appropriate patterns that are pertinent for application [5]

The recommendation phase suggests the kinds of products a customer or user would find appealing. Recommendations may be given directly from the dataset gathered in the information gathering stage (model—or memory-based) or indirectly via the system's observation of users' browsing histories. The available filtering techniques will be discussed in Section II (B).

2) Generative AI: Within the swiftly evolving landscape of artificial intelligence, Generative AI emerges as a rapidly advancing subfield with the transformative potential to *revolutionize* recommender systems. Its forward momentum promises to surmount existing limitations and elevate the capabilities of these systems to new heights. Generative AI models, such as Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE), demonstrate proficiency in producing innovative and high-quality data by learning from existing samples. Their aptitude for crafting new data holds substantial promise for recommender systems, as these systems heavily depend on data to understand user preferences and deliver precise recommendations.

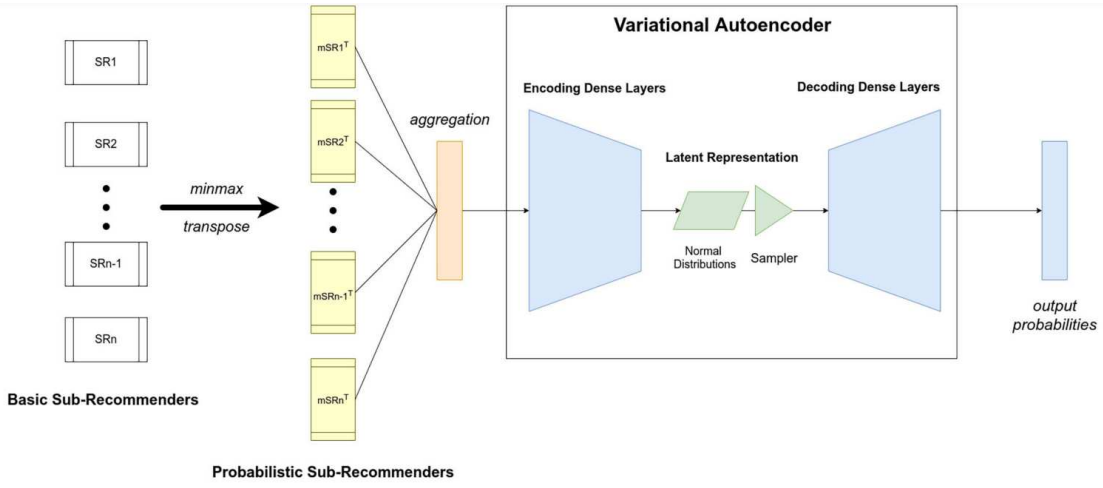

Fig. 2 Architecture of generative AI in recommender system

In the Data Collection and Pre-processing Step, there is data collection and pre-processing, a fundamental step in which unprocessed data is carefully cleaned up and arranged. The data must be cleaned, formatted, and transformed in this first step to be effectively used later. Gathering a diverse and representative dataset, such as high-quality images, is crucial to align with the desired output domain. The data will be pre-processed later to ensure consistency, remove noise, and prepare for model training. Pre-processing, a crucial stage in which raw data is meticulously cleansed and organized, comes before data gathering. For the data to be used efficiently in subsequent steps, it must be cleaned, formatted, and converted in this initial step. A varied and representative dataset—such as high-quality images—must be gathered to correspond with the intended output domain. Later, the data will undergo preprocessing to guarantee consistency, eliminate noise, and prepare for model training.

Choosing an appropriate model is crucial to get the desired results in the Model Architecture Selection Phase. Various models have unique features that are appropriate for particular applications. Variational Autoencoders (VAEs) are a popular model architecture in generative AI that may be used to learn the dataset's distribution. After that, realistic visuals and other creative content are produced using Generative Adversarial Networks (GANs), and lastly, the following word in a sequence can be predicted by Autoregressive models. Using GANs or VAEs to produce new interaction recommendations, generative model layers train the model to recognize patterns in user-item interactions. While VAEs concentrate on encoding and decoding latent representations, GANs have a generator that generates fresh samples and a discriminator that provides feedback.

Following the selection of the model architecture, the training process begins in the Model Training phase. The model is trained using a dataset that contains input data. The training process depends on the selected model and usually involves regularization, gradient descent, and backpropagation to maximize model parameters. Reducing the difference between the output of the model and the actual data, or ground truth, is the primary goal of training. Thanks to this optimization process, the model can generate material that closely resembles the patterns found in the training data. As a result, the model can produce material similar to the training set.

This is accomplished by comparing it to predetermined criteria and benchmarks during the Evaluate and Refine process. Using these measures, you can evaluate the created output's quality, coherence, and realism. If the outcomes don't meet your expectations, you can adjust the model iteratively. This iterative procedure may involve tweaking the dataset, changing the training parameters, or altering the architecture. Remember that reaching optimal performance is a process that requires incremental steps, and it may take several attempts to get the necessary level of satisfaction with the model's output.

After being satisfied with how the generative AI model performs in the Test and Validate phase, it becomes essential to conduct thorough testing and validation. It can be done by testing it on a separate dataset that has not been seen before to see how well it can generalize to new data. Additionally, assess the model's capability to produce varied and coherent outputs in diverse scenarios. Lastly, the results are compared against human judgment and domain-specific criteria to ensure the model's reliability and effectiveness.

The deployment and integration layer are where the generative AI process ends in the Deploy and Integrate phase. The improved model is incorporated into proper systems, platforms, or applications, enabling end users to access its creative results. While integration guarantees smooth communication with current frameworks or technologies, deployment entails the purposeful use of those changes. Gather user feedback and opinions, monitor the model's performance, and make changes. Continuous improvement will always help to keep the model relevant and effective. Fig. 3 shows the end-to-end process of generative AI.

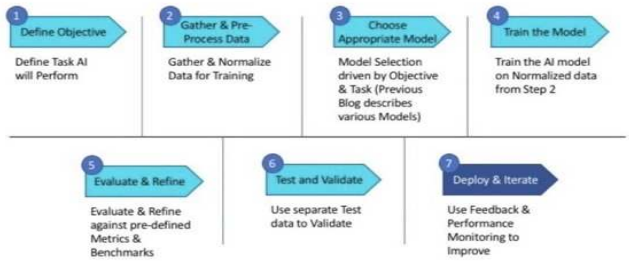

Fig. 3 End-to-end process of generative AI

B. RS Techniques

1) CB: CB approaches leverage item or user metadata to formulate tailored recommendations. This involves examining the user's purchasing history as a critical factor. For instance, a user is assumed to prefer a certain author or brand if they have interacted with that author's book or brand's product. Additionally, there's a chance they'll purchase a comparable item later.

The three components of CB RS's architecture are a content analyzer, a profile learner (generator), and a filtering component. Fig. 4 illustrates this architecture. The Content Analyzer uses feature extraction techniques to gather item content from various information sources and derives item representations. The Profile Learner generates a user profile based on previous likes and dislikes by generalizing user input and applying machine learning algorithms to the item representation. The filtering component matches the user profile with recommended items in the final stage [14].

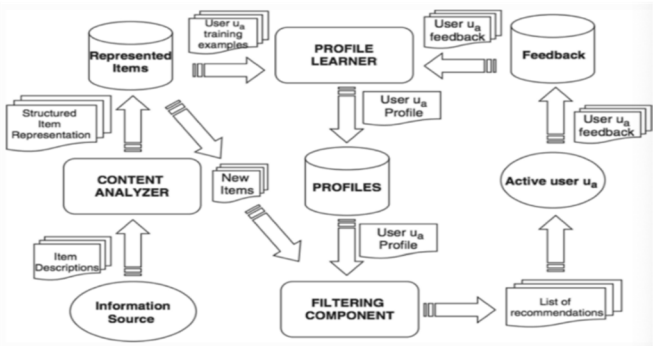

Fig. 4 Architecture of content-based

A content-based book recommendation engine that uses online book information is also shown. The system creates a user profile by analyzing data taken from the web using a

Naïve Bayes classifier. Using user-provided training examples, this profile plays a crucial role in producing a prioritized list of book titles. By enumerating the attributes that contribute to the top ratings, the system can explain any suggestions it makes to consumers. This gives people complete confidence in the advice the system gives them.

2) CF: Collaborative filtering examines user relationships and product interdependencies to find new user-item correlations. User behavior or user evaluations of recommended things are the foundation for CF recommendations. It investigates a variety of potential materials and suggests items loved by users who are similar to you [15]Through a learner profile, RS can obtain data like the learner's age, nationality, past educational experiences, and educational background, among other things. Memory-based and model-based collaborative filtering are the two primary categories. Particularly accurate collaborative filtering systems consider data from multiple users as opposed to just one. One benefit of the CF is that it is independent of the content. As a result, it may recommend sophisticated products like films without the requirement for metadata analysis.

When making suggestions, memory-based recommenders depend on the direct similarities between users or items. Typically, these systems leverage unprocessed historical user interaction data, like user-item ratings or purchase histories, to discern likenesses between users or items and formulate personalized recommendations. User-based and item-based collaborative filtering are the primary categories into which memory-based recommenders fall.

The user-based approach involves suggesting items to the target user by identifying others with similar behavior or preferences. This entails finding users who closely resemble the target user based on past interactions with items. This results in recommendations like "users who are similar to you also liked..." such as Jenny and Tom, both avid fans of sci-fi books. In practical terms, if a new sci-fi book surfaces and Jenny purchases it, the same book would be recommended to Tom because of his shared fondness for sci-fi books.

Item-based collaborative filtering is a technique that finds things that resemble the ones the target user has already engaged with to make suggestions. The goal is to identify products with comparable user experiences and suggest those products to the intended user. This can include recommendations of the kind "users who liked this item also liked...". For example, let's say that Fahrenheit 451 and The Time Machine are two science fiction novels for which John, Robert, and Jenny gave five stars. As a result, the system suggests The Time Machine to Tom after he purchases Fahrenheit 451 since it believes it to be comparable based on user ratings.

Next, model-based recommenders use machine learning models to produce recommendations. These systems carefully extract patterns, correlations, and linkages from past user-item interaction data. By absorbing insights from this data, model-based recommenders can forecast a user's preferences for products they haven't interacted with before. Several approaches are used in the field of model-based recommenders, demonstrating the adaptability of this methodology. These methods include neural networks, matrix factorization, and Singular Value Decomposition (SVD), each with a unique advantage in producing precise and customized recommendations.

Despite this, matrix factorization is a popular method for collaborative filtering. The models are better than conventional nearest-neighbor methods for suggesting products because they can include extra data such as implicit feedback, temporal effects, and confidence levels. Some of the best realizations of latent factor models are based on matrix factorization.

Matrix factorization, in its most basic form, uses vectors of factors created from patterns in item ratings to represent both items and users. The ability to include more information is one of matrix factorization's strengths. Recommender systems can infer user preferences from implicit feedback when no explicit feedback is available. Indirect feedback is the expression of opinions through user behavior, such as past purchases, browsing habits, search activity, or even mouse movements. Implicit feedback is typically shown by the existence or non-existence of an event, which is sometimes depicted as a heavily packed matrix.

3) HB: Combining multiple recommender methods to provide more personalized, different, and effective recommendations is one of the best ways to increase sales and improve client retention. Especially when recommendations are made in real life, they generate more dependable, precise, and adaptable ideas. Content-based and collaborative filtering are combined, and both techniques are used to classify and recommend products to customers. This method was created to address the shortcomings of the CF methodology, such as sparsity and diversity, while also utilizing the benefits of memory and model-based CF techniques. Depending on the unique needs and limitations of the recommendation system, as well as the type of data that is accessible, a hybrid method may be chosen. Netflix exemplifies hybrid filtering by suggesting similar films based on user ratings (content-based filtering) and comparing a user's past with other users who are similar to them (collaborative filtering). The features produced by one recommendation technique are fed into another recommendation strategy in the feature-combination hybrid technique. The feature augmentation technique creates a model that is always richer in terms of information usage than a single rating by using the ratings and other pertinent data generated by a prior recommendation system as input for another recommender [12]. The metalevel hybrid technique learns a second recommendation algorithm from a first algorithm by using the full model as input [16] which is shown in Fig. 5 below on how the HB filtering works.

One good consequence of using hybrid recommenders is that they are frequently robust in managing different suggestion scenarios. They can adjust to user behaviors, data features, and recommendation difficulties. Such adaptability is helpful for practical recommendation systems. It also can mitigate the limitations of individual recommendation techniques. For example, they can tackle the "cold-start" challenge for new users and items by incorporating content-based recommendations, offering unexpected suggestions, and mitigating popularity bias.

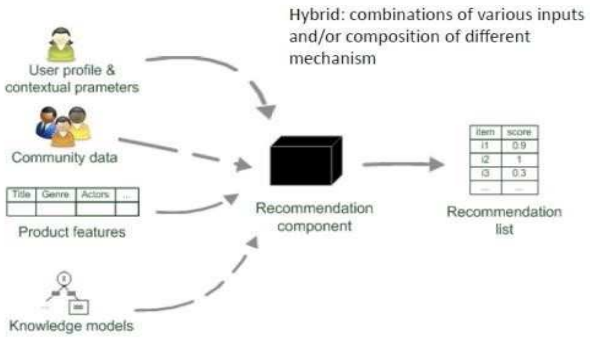

Fig. 5 HB filtering

4) Generative AI: Recently, recommender systems have benefited from applying various AI techniques, improving user happiness and the overall user experience. AI makes recommendations of a higher caliber than those made with traditional techniques. Automation of intelligent behavior is the aim of AI technology development. These six domains include knowledge engineering, reasoning, planning, communication, perception, and motion [17]. Hence, the next step will explain techniques to achieve the desired outcome. Fig. 6 explains the AI areas and the techniques used.

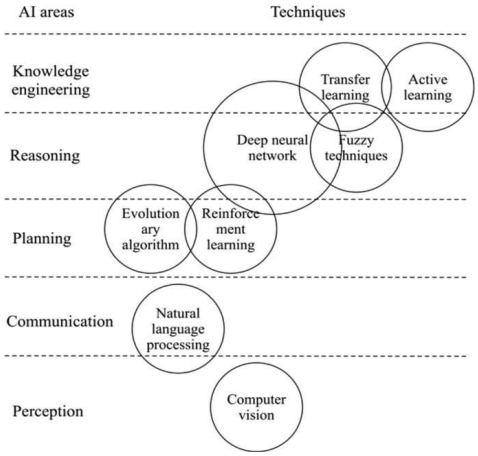

Fig. 6 The AI areas and the techniques used

Transfer learning has emerged as a method for knowledge engineering, whereby information is transferred transferring data from a large source domain to a small target domain [18]. According to this definition, transfer learning uses information from one or more source data to help with target data in a learning activity. Transfer learning methodologies can be divided into three main categories: unsupervised transfer learning, transductive transfer learning, and inductive transfer learning. The fundamental principle of active learning is to select training data with the purpose of maximizing machine learning performance with minimal information requirements. Users may be asked to name occurrences not labeled by an active learning system. Active learning has useful applications in different AI disciplines, especially well-suited for online systems since labeling can be expensive, time-consuming, and sometimes impractical. For reasoning, the techniques used are deep neural networks (DDNs) and fuzzy. DNNs can model nonlinear relationships effectively. Recommender systems often involve non-linear dependencies between user preferences and item characteristics, and DNNs can capture these nuances. DNNs can be used for matrix factorization, a common technique in recommender systems. They can learn latent factors representing user and item embeddings, capturing implicit relationships in the data. Since fuzzy techniques can replicate real-world concepts that cannot be adequately described, they are frequently used in artificial intelligence (AI). Fuzzy approaches have drawn much interest in the literature. For example, fuzzy distance has been used to describe similar instances and fuzzy sets have been utilized by researchers to represent linguistic variables when feature values cannot be adequately expressed in numerical values [18].

Afterward, reinforcement learning and evolutionary algorithms participate in the planning domain. Evolutionary algorithms (EA) are a subclass of population-based search algorithms for global optimization in artificial intelligence that is inspired by natural processes. Some potential answers to a problem that needs to be solved make up an initial population, also known as the parent population. An evolutionary algorithm starts at this point. Genetic operators, such as crossover and mutation, are applied to parent individuals to create new solutions, referred to as offspring. The selection of individuals who will become parents is based on their suitability as future parents. This process continues until a few conditions are met to end it.

The recommender system is seen as a learning agent in reinforcement learning; user behaviors match states, and user actions are suggestions generated by the system. The prize is the feedback that users leave on the suggestion results, such as the frequency of click-throughs or the duration of time spent on the page. The aim is to find a value function or method that enables consumers to maximize long-term advantages.

In order to address the problem of data sparsity in communication, most recommender systems also enrich the rating matrix with review data acquired via natural language processing. In extreme cases, virtual ratings are generated utilizing the emotion polarity obtained from review classification without ratings. Topic models evaluate item metadata in "bag-of-words" representation and use matrix factorization techniques to handle both warm-start and cold-start circumstances. The researcher improved suggestions with feature sentiment and product experience by mining feature-based product descriptions from reviews, resulting in better items based on user inquiries [19].

In summary, the direct application of computer vision in picture recommendation—mapping images to user preferences—contributes to recommender systems. Deep neural networks were used to extract information from photographs in early e-commerce suggestions, which were then integrated into pre-existing techniques for apparel recommendations [17]. This was extended in a later study by using low-level data, like color qualities, that mimicked parts of the human visual system. This provided a fresh viewpoint on user preferences in movie recommendations by creatively integrating visual elements from still frames and movie posters with a matrix factorization algorithm. Point-of-interest recommendations have also used visual content, using the wealth of landmarks included in pictures and images uploaded by users.

C. Related Works

In the study by Ferreira et al. [20], challenges were encountered when employing matrix factorization techniques, such as SVD, due to the utilization of a dataset characterized by its sparsity and large size. The project intends to enhance some recommendation system-related areas. The task's loss function had to be Mean Squared Error (MSE) since it did not involve a classification problem where binary cross-entropy could be used. The fact that ADAM stands out as one of the most significant choices and is renowned for its speed, low memory usage, and ability to handle big datasets impacted the optimizer selection—a consistent strategy with this article. The evolution of loss and validation loss (val_loss) show a declining trend over each epoch, ultimately stabilizing at a point. The absence of an intersection between the loss curve and val_loss is attributed to the addition of the dropout layer, which exclusively affects the training dataset, leading to the observed disparity between them. Despite the inherent challenges associated with very sparse datasets and the application of a collaborative filtering approach, the study anticipated problems but found that the autoencoder model effectively overcame them. The Root Mean Squared Error (RMSE) value of 0.996 indicates the model's success in providing recommendations aligned with users' interests, unaffected by the data sparsity problem. The obtained results are promising, showcasing an RMSE value of 0.029 for the first and 0.010 for the second datasets.

In the following study, Tran et al. [21]demonstrated outstanding potential by implementing a deep Autoencoder (DAE) for recommendation systems in an efficient manner. They built a flexible deep neural network model, called the FlexEncoder model, that combines characteristics and methods from multiple sources into an all-encompassing DAE model. Characterized by unique features and tunable parameters, this FlexEncoder model enables a comprehensive examination of parameter impacts on recommender system prediction accuracy. This method incorporates novel features from previous publications to make identifying the best performance parameters for a particular dataset easier. The ADAM optimizer, the SELU activation function, and one round of dense refeeding with mean normalization enabled were among the standard parameters used in the study that contributed to an RMSE in the range of 0.90 to 0.91. The authors conducted a thorough evaluation analysis to substantiate their assertion that the parameters significantly selected impact the DAE model's prediction accuracy. The FlexEncoder model outperformed other cutting-edge recommendation algorithms and showed the lowest RMSE when tested against current methods on the MovieLens 100K dataset. In particular, the FlexEncoder outperformed AutoRec (0.887) and SVD++ (0.903) with an RMSE score of 0.833. These comparative RMSE findings were taken from another study.

Other than that, Zhang et al. [18] emphasized the widespread use of conventional recommender systems, encompassing knowledge-based, collaborative filtering, content-based, and hybrid models created in the past ten years. Even still, these models suffer from issues related to cold start and data sparsity, which causes a large reduction in recommendation performance in sparse user-item interactions. The inability of recommendation systems to offer suggestions for new users and things gives rise to the cold start issue. Researchers have developed innovative recommendation methods that use side data about individuals or objects to address these problems. However, these models' limitations in collecting customers' preferences and item attributes frequently limit the improvement in recommended performance. As a result, Autoencoder (AE) has become a powerful method for extracting essential features from data, transforming recommendation structures, and providing improved user experiences. The authors stressed the widespread use of recall and Root Mean Square Error (RMSE) as evaluation measures. Mean Average Error (MAE) and RMSE are commonly used for rating prediction assessment. While accuracy and recall are still frequently employed to evaluate classification results, recall, Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG) are preferred for giving correctly indicated items in top ranks more credit.

In the following paper, Rama et al. [22] integrated embeddings into a deep neural network using autoencoder features for recommender system rating prediction. They demonstrated a method for establishing a benchmark for prediction accuracy using DAFERec (Deep Autoencoders for Feature Learning for Recommendations). DAFERec uses the innermost layer activations of a deep autoencoder as features in a deep neural network for recommendation. This technique combines the innermost activations of the autoencoders with embeddings to incorporate higher-order innermost nonlinear latent characteristics from both user and item autoencoders. Using latent variables, one can learn a compressed higher-order representation of user preferences or item attributes in recommender systems, enabling the Deep Neural Network (DNN) to learn it. Like the approaching method in this study, they started with a list of user-item-rating tuples and turned it into a rating matrix. The details of data preparation were not explored; instead, the standard pivot operation in several programming languages was used. The authors employed the tenfold cross-validation technique to confirm the dependability of their findings.

Furthermore, based on Loukili et al.[23], they have succeeded in a significant milestone by developing an algorithm to provide personalized recommendations to customers, employing association rules through the utilization of the Frequent Pattern Growth algorithm. This new method has shown outstanding results, with a high chance that customers will buy the following product suggested by the system. The study focuses on assessing how well the recommendation system performs, which is done by calculating the average probability (Paverage) connected to the chance that customers will buy the following suggested product. The evaluation metric is a pivotal indicator of the system's effectiveness in accurately suggesting items that align with individual customer preferences, ultimately contributing to an enhanced user experience and increased customer satisfaction. The success of this algorithm not only underscores its potential in optimizing recommendation systems but also signifies a promising step towards delivering more tailored and relevant suggestions to users in various domains.

Additionally, retailers and service providers invest more money in social media, e-commerce, mobile, and internet

channels to improve customer engagement and communication with current and potential customers and increase sales. Thus, studies conducted by Grewal et al. [24] explained in detail how predictive analytics helps estimate people's product preferences, price sensitivity, and expected next steps on the customer journey—all critical for improving business activities. This is accomplished by utilizing big data and advanced analytics tools to obtain knowledge and generate tailored recommendations based on client information. Furthermore, clear suggestions and insight into the AI system are made possible by explainable AI (XAI) techniques, which eventually boost confidence and lessen algorithmic biases. However, the study also discusses the possible dangers of the "big data revolution," especially as they relate to data security, and it highlights the significance of having better data that is influenced by the arts and sciences of marketing and creativity in addition to the science of robotics, AI, and machine learning.

Then, the techniques used for the recommendation system on Amazon are discussed by Rybakov et al. [25] mentioned in this study. The use of neural networks in a personalized recommender system to make product recommendations based on implicit feedback from customers—such as past purchases, listens, or watches—is covered in this research. Additionally, offline assessment metrics—like Product Converted Coverage (PCC) at K and Precision at K—that are frequently employed in real-world recommender system applications are presented in this work. The use of a deep learning package that facilitates parallel training with the multi-GPU model is also mentioned in the research. This is a crucial feature for developing neural network-based recommenders with massive input and output data dimensionality. Additionally, the study mentions the Deep Scalable Sparse Tensor Network Engine (DSSTNE) at Amazon and the utilization of Apache Spark for recommendation generation at Amazon scale, demonstrating the usefulness of the methods covered in the research at Amazon. Among these, the advantages are increased accuracy metrics as the research has shown that using a hybrid method that combines predictor and auto-encoder models leads to improvements in accuracy metrics like Product Converted Coverage (PCC) and precision in a variety of digital product categories.

Consequently, Yu et al. [26] claimed that collaborative filtering frequently faces difficulties while handling sparse data. Model-based Collaborative Filtering Algorithm Based on Stacked AutoEncoder (MCFSAE), By first converting the rating matrix into a high-dimensional classification dataset the same size as the entire number of ratings, the suggested solution effectively eliminates this problem. Because the ratings are typically broad, this method guarantees strong categorization performance. The authors utilize Stacked AutoEncoder, a skilled nonlinear feature reduction approach, to leverage the high-dimensional classification dataset and produce an elevated low-dimensional feature depiction. Experimental results show that MCFSAE outperforms other collaborative filtering (CF) models, particularly in the case of sparse rating matrices. Even with the large-scale training dataset that was gathered, MCFSAE effectively resolves the sparsity issue that existed in the initial recommendation task. Furthermore, v. To assist in creating a high level, low-dimensional feature representation of the original data, stacked autoencoders (SAE) are used. According to experimental findings, the suggested MCFSAE operates better than the most advanced CF approaches in rating prediction, especially when dealing with sparse rating matrices, because of the remarkable representation performance of SAE.

Next, Lacic et al. [27] recommendation approach shown in this work encodes sessions inside the job domain using several autoencoder architectures. The inferred latent session representations are used in a k-nearest neighbor fashion to recommend jobs within a session. Three autoencoder types are used in the study: variational autoencoder (VAE), denoising autoencoder (DAE), and classical autoencoder (AE). A single hidden layer separates the input and output of the most basic AE. By corrupting the input on one or more layers prior to computing the final output, DAE, on the other hand, develops representations that are resilient to minute, insignificant changes in the input. Alternatively, VAE uses variational inference to retrieve latent representations. Similar to our methodology in this research, the authors performed a grid search on hyperparameters for baseline approaches using a validation set. They then assessed the models on three datasets using NCDG, MRR, Session-based novelty (EPD), and System-based novelty (EPC).

Lastly, Sachdeva et al. [28] presented a recurrent variant of the Variational Autoencoder (VAE), in which they chose to run a recurrent neural network (RNN) on the consumption sequence subset rather than running a fraction of the full history without taking temporal dependencies into account. The sequence passes through several fully connected layers at each RNN time step in this configuration. The probability distribution of the most likely future preferences is modeled by the output from these layers. The authors show that adding temporal information is essential to improving the VAE's accuracy. Their model achieves significant improvements over the state-of-the-art, demonstrating its capacity to employ the recurrent encoder to capture temporal relationships inside the user-consumption sequence while adhering to the fundamental ideas of variational autoencoders. They also perform sensitivity analysis concerning the configurations/contour circumstances under which Sequential Variational Autoencoder (SVAE) performs best. The main finding from their tests is that, for the top-N recommendation task, SVAE outperforms the state-of-the-art in several criteria. The evaluation concentrates on top-N recommendations while considering implicit preferences, using measures like NCDG, Precision, and Recall. On the MovieLens-1M and Netflix datasets, SVAE continuously beats rivals, exhibiting notable gains in all metrics.

III. Results and Discussion

A. Theoretical Framework

1) Autoencoder Architecture (AE): Neural networks designed for efficient coding, dimensionality reduction, and generative modeling are good candidates for unsupervised learning tasks, such as auto-encoders. It has been useful in learning underlying feature representation in some domains, including computer vision, speech recognition, and language modelling. With this knowledge in mind, autoencoders have

been incorporated into new suggestion designs, expanding the possibilities for developing creative user experiences that appeal to customers. An auto-encoder consists of three layers: input, hidden, and output, as shown in Fig. 7. The input layer is where the data is received. The input layer and the hidden layer combine to form an encoder. The output layer and the hidden layer combine to form a decoder [29].The output layer is where the output data exits.

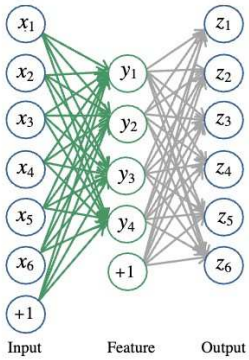

Fig. 7 Autoencoder architecture

The encoder encodes the high-dimensional input data x into a lower-dimensional hidden representation h with a function f in Equation (1).

$$h = f(x) = Sf\ (Wx + b) \quad (1)$$

where *Sf* is an activation function, W is the weight matrix, and b is the bias vector.

The decoder decodes the hidden representation h back to a reconstruction x' by another function g in Equation (2).

$$x' = g(h) = Sg\ (W'h + b') \quad (2)$$

where *Sg* is an activation function, W' is the weight matrix, and b' is the bias vector.

Non-linear options for *Sf* and *Sg* include Sigmoid, TanH, and ReLU. The auto-encoder can learn More meaningful features as a result, compared to other unsupervised linear techniques like Principal Component Analysis. The auto-encoder was trained to use the squared error for regression tasks or the cross-entropy error for classification tasks to minimize the reconstruction error between x and x's.

The formula for the squared error as per Equation (3).

$$SE\ (x, x'\ ) = \|x - x'\ \|2 \quad (3)$$

2) Variational Autoencoder Architecture (*VAE):* The extension of AE is called VAE [30] As shown in Fig. 8, the bottleneck will have a sampling layer rather than a straightforward dense layer. This layer will employ the mean and variance from the previous encoder layer to create a Gaussian sample that will be used as input for the decoder. The first layer of the AE also uses dropout.

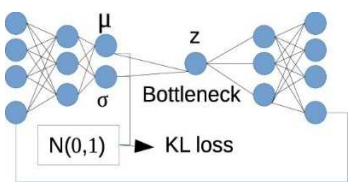

Fig. 8 VAE architecture

In the sampling layer, a latent vector z will be generated by sampling from a Gaussian distribution with mean mu and standard deviation sigma. This is the reparameterization trick, which allows the network to be trained using gradient descent. Equation (4) shows the sampling layer.

$$Z = \mu + \sigma \odot \epsilon \quad (4)$$

where μ = is the mean vector, σ = is the standard deviation vector, ϵ = is a random sample from a standard normal distribution.

Then, to encourage the distribution of latent vectors to be close to a standard normal distribution, a term related to the Kullback-Leibler (KL) divergence is added to the loss function, as shown in Equation (5).

$$KL\ loss = -21 \sum i=1K(1+\log(\sigma i2)-\mu i2-\sigma i2) \quad (5)$$

Next, dropout is a regularization method that neural networks frequently employ to avoid overfitting. During training, it arbitrarily sets a portion of the input units to zero. Dropout is applied to the encoder's initial layer in the VAE scenario.

$$Output = Input \odot Mask$$

where Input = input vector, Mask = binary mask with values randomly set to 0 or 1 during training.

Thus, VAE introduces a sampling layer in the bottleneck, which involves generating a latent vector by sampling from a Gaussian distribution. Additionally, dropout is applied to the first layer of the encoder for regularization. The KL divergence loss encourages the learned latent space to approximate a standard normal distribution, promoting a continuous and structured latent space. The equations provided are simplified representations, and the actual implementation might include additional details for practical considerations.

B. Dataset

The dataset for model training in this prototype is Amazon Consumer Review, obtained from the Kaggle Website provided by Datafiniti's Product Database. The dataset is loaded into a pandas DataFrame after importing the necessary libraries for developing the recommendation system. This dataset consisted of 22 columns of features, but only five columns were used for further exploration, as listed in Table I.

TABLE I
DESCRIPTION ON DATASET

Column Name	Description
id	Unique ID for every product purchase
name	Name of each product
asins	A list of ASINs (Amazon Standard Identification Numbers) associated with the product.
categories	Categories for every product, indicating its classification
reviews.ratings	Overall ratings given by users in product review
reviews.userna me	People that rating for every product they reviewed
brand	Brand for each product
keys	Universal Product Code (UPC), unique product barcode used for retail and online sales

Column Name	Description
manufacturer	The company or entity responsible for manufacturing the product.
reviews.date	The date when a review was posted.
reviews.dateA dded	The date when the review was added to the system or dataset.
reviews.dataSe en	The date when the review data was last seen or accessed.
reviews.didPur chase	Indicates whether the reviewer claims to have purchased the product.
reviews.doRec ommend	Indicates whether the reviewer recommends the product.
reviews.id	A unique identifier for each review.
reviews.numH elpful	The number of users who found the review helpful.
reviews.rating	The specific rating given by the reviewer in the review.
reviews.source URLs	URLs pointing to the source of the reviews.
reviews.text	The text content of the review.
reviews.title	The title or heading of the review.
reviews.userCi ty	The city or location associated with the reviewer.

C. Data Cleaning and Data Preprocessing

After loading the dataset in the prototype, data cleaning and preprocessing are carried out to remove errors and inconsistencies and improve its quality. The Pandas.drop() method is used to remove unnecessary columns before continuing further analysis. The columns selected were "name," "as is," "categories," "reviews.rating," and "reviews.username."

After that, cleaning data was resumed by removing the missing values (NaNs). This delicate process is essential because it enables the models to extract more significant insights from the dataset, providing a solid basis for further analysis. The dataset becomes more complete by filling in these gaps, enabling the models to produce more detailed representations from the available data. Fig. 9 shows how to handle the missing values by using the dropna() method.

```
data = data.dropna()

print(data.isna().sum())
```

```
name              0
asins             0
categories        0
reviews.rating    0
reviews.username  0
dtype: int64
```

Fig. 9 Handling missing values with dropna() method

Furthermore, since column "name" in this data frame is a bit messy and needs to be cleaned up, the unique() method has been used first to identify the unique product names available in the dataset. Then, to standardize it, str.strip() and str.replace() are also being used to remove leading and trailing whitespaces(spaces, tabs, or newline characters) from a string. This method returns a new string with the starting and trailing whitespaces removed, without altering the original string. Python's str.replace the (old, new) function to swap out instances of one substring (old) in a given string for another (new). In this case, str.replace (',,,' , ',') were used to replace text sequences containing three consecutive commas with a single comma. After that, the original column "name", replaced with new column name which is "cleaned_name".

Next, since column "cleaned_name" and "reviews.username" are categorical, it need to be coverted into numerical format first by using LabelEncoder(). For further work, only three columns will be used wich are "username_encoded", "product_name_encoded" and "reviews.ratings" Fig. 10 shows how the categorical columns were converted into numerical format.

```
Name: product_name_encoded, Length: 27864, dtype: int64
```

Fig. 10 Convert categorical data into numerical formats

D. E-commerce Products Recommender System

After the preprocessing part is done, the cleaned datasets are loaded using the surprise library, which imports Dataset and Reader. Beside surprise library, other necessaries libraries are also imported in this part including numpy, tensorflow.keras.models, tensorflow.keras.layers, keras.models, keras.optimizers, keras.layers and surprise.model_selection. After defining autoencoder model, the model then compiles it. The optimizer used is Adam optimizer and loss function is mean_squared_logarithmic_error. The data is then split into training and testing data, which appears to be initializing matrices to represent training and test data in a recommendation system. The initialization with zeros indicates that there is no known interaction or preference between users and items. These matrices will be populated with actual interaction data during the training and testing phases of the recommendation system.

In addition, the autoencoder model uses the specified training data, the 'train_data', so that it can learn a representation of the input data that captures its important features. By using the same data for input and target output, the autoencoder is trained to reconstruct the original data. Fig. 11 shows the training process repeated for 30 epochs, with each epoch consisting of batches of 52 samples.

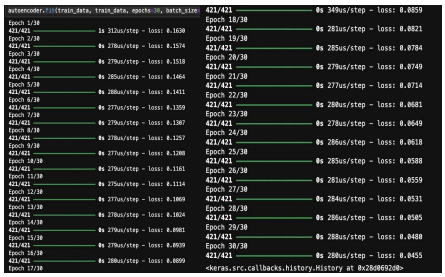

Fig. 11 The Training process repeated for 30 epochs consisting of 52 batches of samples.

The user's attributes or characteristics are captured in the subsequent function by the vector representing the user in the learned latent space derived from the autoencoder model. The representation of an item in the learned latent space is defined by the matrix for each row, which is obtained by transposing and accessing the weights of the second layer. Next, the user representation vector and the transpose of the item representation matrix are computed as the dot product. The outcome of this operation is a vector of similarity scores, where each score denotes how similar the user is to a

particular object. Lastly, they are sorted based on similarity scores in descending order, indicating the objects that are most similar to the user. Recommendations are based on the top-N items with the highest similarity scores.

Presumably, each item in the learned latent space is represented by a similarity score that indicates how similar the user is to it. It is calculated using the transpose of the item representation matrix and the dot product of the user's representation vector. Higher similarity scores indicate that an item is more relevant or similar to the user and is recommended accordingly. Lastly, for accurate visualization, part of the code ensures that the label-encoded item IDs obtained from the recommendations are transformed back to their original item names before displaying the recommendations to the user. This decoding step provides human-readable information about the recommended items. Fig. 12 illustrates the human-readable version of this recommendation where the prototype will show previous items bought by user ID prompted and recommend new products.

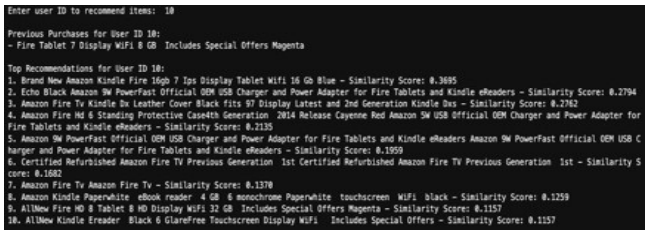

Fig. 12 Example of product recommendation

E. User Interface

This prototype also includes a Graphic User Interface (GUI). As shown in Fig. 13, the prototype can be started by opening the "Autoencoder in E-commerce RecSys" in a Jupyter notebook or another online or local application that supports Python and its libraries.

Fig. 13 IPYNB file of the prototype

The user will first see the pop-up window prompting user ID (1-32) only because the number of other user IDs has been compressed earlier using the autoencoder model, as shown in Fig. 14.

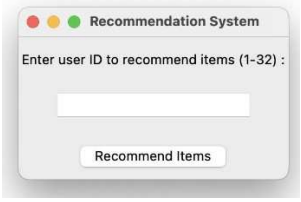

Fig. 14 GUI to prompt user ID

After the User ID is prompted, another window will pop up to show the recommendation results. The window will first show the User ID and the previous item bought. Then, the top recommendation for the user ID will be displayed along with the similarity scores, as shown in Fig. 15.

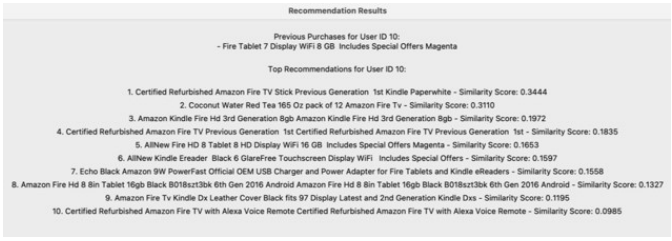

Fig. 15 Recommendation results

F. Evaluation Results

Mean absolute error (MAE) is a popular metric for calculating the average absolute difference between expected and actual values. It offers an easy-to-understand method for determining the accuracy of a predictive model. The MAE is computed by adding together all of the absolute differences between each observed and forecasted value and dividing the result by the total number of observations. Regardless of the direction of the errors, the MAE shows how much predictions, on average, differ from the actual data. A lower MAE indicates better model accuracy.

On the other hand, the loss function of a machine learning model is represented graphically by loss curves, which are usually plotted against the total number of training epochs. As the optimization goal during training, the loss function measures the discrepancy between the target values and the model's predictions. When tracking the learning process and deciding on the model's performance, loss curves are a crucial tool. Training and validation losses are typically plotted over epochs in a loss curve. While the training loss indicates how well the model matches the training data, the validation loss suggests how well the model generalizes to new data. A diminishing training loss is expected as the model improves at reducing the error on the training set. The loss function, often denoted as $I(\emptyset)$,is a mathematical expression that measures the discrepancy between the predicted values of a machine learning model and the true target values. The goal during training is to find the values of the parameters θ that minimize this loss function, typically achieved through optimization algorithms such as gradient descent.

Similarity scores measure how similar or similar two entities—users or items—are in the context of machine learning, especially in recommender systems. These scores are essential for assessing how relevant an item is to a user or how similar users are to one another regarding preferences or behavior. It is possible to employ a variety of similarity metrics, such as Jaccard similarity, Pearson correlation, and cosine similarity [31],[32],[33]. Overall, similarity scores play a crucial role in enhancing the personalization and effectiveness of recommendation systems by quantifying the relationships between entities within the system.

Fig. 16 depicts the MAE result for both the training and test sets. Consistent MAE values between the training and testing sets suggest a well-generalized model. This encouraging result shows that the model has successfully discovered underlying patterns in the data without committing individual training examples to memory. A balanced performance on both sets suggests that the model is ready to make good predictions on unobserved data by collecting key properties and avoiding overfitting pitfalls. Table II shows the score obtained from the previous technique used.

Next, loss curves, specifically training and validation loss curves, provide information about how well the model is learning over epochs. Monitoring the loss curves helps detect overfitting, find the optimal epoch for early stopping, and understand the model's convergence. The result is shown in Fig. 17.

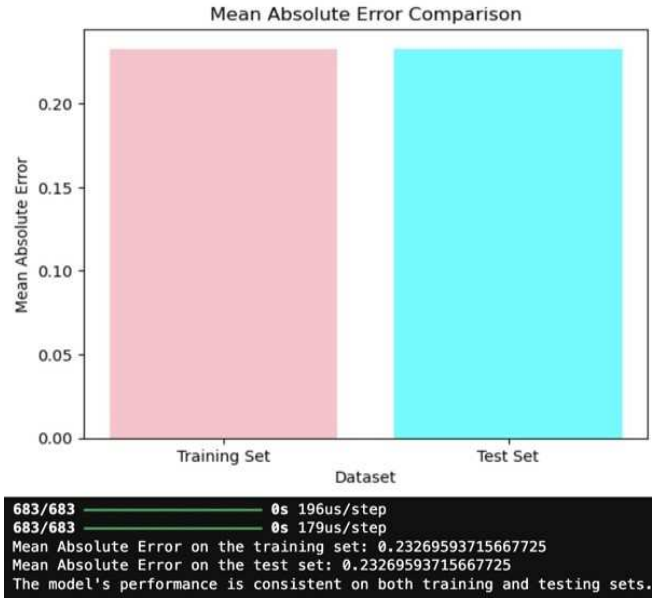

Fig. 16 Result obtained

TABLE II
MAE SCORE

Evaluation Metric	Training set	Test Set
MAE Score	0.2327	0.2327

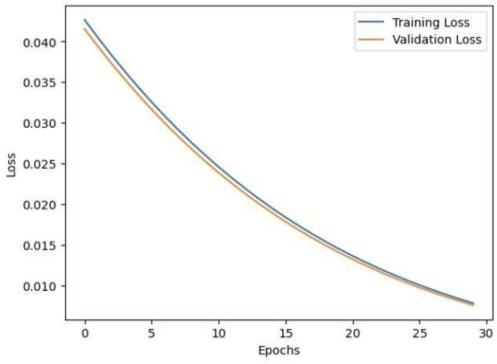

Fig. 17 Loss curves

A decreasing training loss is expected as the model learns to minimize the error on the training set. However, the validation loss helps identify potential overfitting or underfitting issues. This can be described as a good model because it demonstrates a decreasing training loss, which is ideally a decreasing validation loss without significant divergence.

IV. CONCLUSIONS

The investigation of recommendation techniques has highlighted the advantages and disadvantages of three well-known approaches: CB, CF, and hybrid. Apart from these well-known methods, a new method incorporating generative AI— more precisely, the AE technique—has been looked at in detail. This research will primarily focus on this novel approach, with many models to be implemented and assessed. A basic GUI prototype has been created to show off the expected features of the program. Metrics like MAE, Loss Curves, and Similarity Scores will be used to measure how well the models predict ratings within the dataset.

The next step will involve implementing new models, such as VAE, to improve the recommendation system even further. Metrics like precision and recall will be included in the evaluation to examine the models' performances thoroughly. Concurrently, efforts will be focused on finishing and improving the system prototype and GUI development. This iterative approach aims to guarantee the completeness and efficiency of each recommendation system function. The dedication to providing a reliable and user-friendly recommendation system is demonstrated by the ongoing development of both the user interface and the underlying models.

REFERENCES

[1] S. Wei, X. Zheng, D. Chen, and C. Chen, "A hybrid approach for movie recommendation via tags and ratings," *Electron Commer Res Appl*, vol. 18, 2016, doi: 10.1016/j.elerap.2016.01.003.

[2] S.-K. Tan, S.-C. Chong, K.-K. Wee, and L.-Y. Chong, "Personalized Healthcare: A Comprehensive Approach for Symptom Diagnosis and Hospital Recommendations Using AI and Location Services," *Journal of Informatics and Web Engineering*, vol. 3, no. 1, pp. 117–135, Feb. 2024, doi: 10.33093/jiwe.2024.3.1.8.

[3] P. Mahajan and P. D. Kaur, "A Systematic Literature Review of Food Recommender Systems," 2024. doi: 10.1007/s42979-023-02537-y.

[4] S. C. Haw, L. J. Chew, K. W. Ng, P. Naveen, A. Gandhi, and A. S. D. Martha, "Ontology-based Recommender System with Descriptive Analytics in e-Commerce," in *Proceedings - 2022 2nd International Conference on Big Data Engineering and Education, BDEE 2022*, 2022. doi: 10.1109/BDEE55929.2022.00015.

[5] W. Chang and J. Park, "A comparative study on the effect of ChatGPT recommendation and AI recommender systems on the formation of a consideration set," *Journal of Retailing and Consumer Services*, vol. 78, 2024, doi: 10.1016/j.jretconser.2024.103743.

[6] R. J. Zwanka and M. M. Zondag, "Tired or Inspired: A Conceptual Model for Using Regenerative Artificial Intelligence to Create Context, User, and Time-Aware Individualized Shopping Guidance," *J Int Consum Mark*, vol. 36, no. 3, 2024, doi:10.1080/08961530.2023.2266897.

[7] L. Jiang, Y. Cheng, L. Yang, J. Li, H. Yan, and X. Wang, "A trust-based collaborative filtering algorithm for E-commerce recommendation system," *J Ambient Intell Humaniz Comput*, vol. 10, no. 8, 2019, doi: 10.1007/s12652-018-0928-7.

[8] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition," *Inf Sci (N Y)*, vol. 180, no. 22, 2010, doi: 10.1016/j.ins.2010.07.024.

[9] J. K. Tarus, Z. Niu, and G. Mustafa, "Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning," *Artif Intell Rev*, vol. 50, no. 1, 2018, doi:10.1007/s10462-017-9539-5.

[10] E. A. Anaam, S.-C. Haw, K.-W. Ng, P. Naveen, and R. Thabit, "Utilizing Fuzzy Algorithm for Understanding Emotional Intelligence on Individual Feedback," *Journal of Informatics and Web Engineering*, vol. 2, no. 2, 2023, doi: 10.33093/jiwe.2023.2.2.19.

[11] Y. Lim, K.-W. Ng, P. Naveen, and S.-C. Haw, "Emotion Recognition by Facial Expression and Voice: Review and Analysis," *Journal of Informatics and Web Engineering*, vol. 1, no. 2, 2022, doi:10.33093/jiwe.2022.1.2.4.

[12] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," 2015. doi:10.1016/j.eij.2015.06.005.

[13] A. Drif, H. E. Zerrad, and H. Cherifi, "Ensvae: Ensemble variational autoencoders for recommendations," *IEEE Access*, vol. 8, 2020, doi:10.1109/access.2020.3030693.

[14] F. Narducci, C. Mustoy, M. Polignano, M. De Gemmis, P. Lops, and G. Semeraro, "A recommender system for connecting patients to the right doctors in the healthnet social network," in *WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web*, 2015. doi: 10.1145/2740908.2742748.

[15] S. M. Al-ghuribi, S. Azman, and M. Noah, "A Comprehensive Overview of Recommender System and Sentiment Analysis. (arXiv:2109.08794v1 [cs.AI])," *arXiv Computer Science*, 2016.

[16] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, and R. Kashef, "Recommendation systems: Algorithms, challenges, metrics, and business opportunities," *Applied Sciences (Switzerland)*, vol. 10, no. 21, 2020, doi: 10.3390/app10217748.

[17] Q. Zhang, J. Lu, and Y. Jin, "Artificial intelligence in recommender systems," *Complex & Intelligent Systems*, vol. 7, no. 1, pp. 439–457, Feb. 2021, doi: 10.1007/s40747-020-00212-w.

[18] G. Zhang, Y. Liu, and X. Jin, "A survey of autoencoder-based recommender systems," 2020. doi: 10.1007/s11704-018-8052-6.

[19] G. Zhang, Y. Liu, and X. Jin, "Adversarial Variational Autoencoder for Top-N Recommender Systems," in *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 2018. doi: 10.1109/ICSESS.2018.8663730.

[20] D. Ferreira, S. Silva, A. Abelha, and J. Machado, "Recommendation system using autoencoders," *Applied Sciences (Switzerland)*, vol. 10, no. 16, 2020, doi: 10.3390/app10165510.

[21] D. H. Tran, Z. Hussain, W. E. Zhang, N. L. D. Khoa, N. H. Tran, and Q. Z. Sheng, "Deep autoencoder for recommender systems: Parameter influence analysis," in *ACIS 2018 - 29th Australasian Conference on Information Systems*, 2018. doi: 10.5130/acis2018.aj.

[22] K. Rama, P. Kumar, and B. Bhasker, "Deep autoencoders for feature learning with embeddings for recommendations: a novel recommender system solution," *Neural Comput Appl*, 2021, doi: 10.1007/s00521-021-06065-9.

[23] M. Loukili, F. Messaoudi, and M. El Ghazi, "Machine learning based recommender system for e-commerce," *IAES International Journal of Artificial Intelligence*, vol. 12, no. 4, 2023, doi:10.11591/ijai.v12.i4.pp1803-1811.

[24] D. Grewal, J. Hulland, P. K. Kopalle, and E. Karahanna, "The future of technology and marketing: a multidisciplinary perspective," 2020. doi: 10.1007/s11747-019-00711-4.

[25] O. Rybakov *et al.*, "The effectiveness of a two-layer neural network for recommendations," in *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2018.

[26] M. Yu, T. Quan, Q. Peng, X. Yu, and L. Liu, "A model-based collaborate filtering algorithm based on stacked AutoEncoder," *Neural Comput Appl*, vol. 34, no. 4, 2022, doi: 10.1007/s00521-021-05933-8.

[27] E. Lacic, M. Reiter-Haas, D. Kowald, M. Reddy Dareddy, J. Cho, and E. Lex, "Using autoencoders for session-based job recommendations," *User Model User-adapt Interact*, vol. 30, no. 4, 2020, doi:10.1007/s11257-020-09269-1.

[28] N. Sachdeva, E. Ritacco, G. Manco, and V. Pudi, "Sequential variational autoencoders for collaborative filtering," in *WSDM 2019 - Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, 2019. doi: 10.1145/3289600.3291007.

[29] S. Chen and W. Guo, "Auto-Encoders in Deep Learning—A Review with New Perspectives," 2023. doi: 10.3390/math11081777.

[30] L. Vu, V. L. Cao, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Learning Latent Distribution for Distinguishing Network Traffic in Intrusion Detection System," in *IEEE International Conference on Communications*, 2019. doi:10.1109/ICC.2019.8762015.

[31] P. M. LeBlanc, D. Banks, L. Fu, M. Li, Z. Tang, and Q. Wu, "Recommender Systems: A Review," *J Am Stat Assoc*, vol. 119, no. 545, pp. 773–785, Jan. 2024, doi: 10.1080/01621459.2023.2279695.

[32] Y. Y. Chow, S. C. Haw, P. Naveen, E. A. Anaam, and H. Bin Mahdin, "Food Recommender System: A Review on Techniques, Datasets and Evaluation Metrics," *Journal of System and Management Sciences*, vol. 13, no. 5, 2023, doi: 10.33168/JSMS.2023.0510.

[33] S. T. Lim, J. Y. Yuan, K. W. Khaw, and X. Chew, "Predicting Travel Insurance Purchases in an Insurance Firm through Machine Learning Methods after COVID-19," *Journal of Informatics and Web Engineering*, vol. 2, no. 2, 2023, doi: 10.33093/jiwe.2023.2.2.4.