GENERATIVE AI RECOMMENDER SYSTEM IN E-COMMERCE

NUR ANIS NABILA BT MOHD ROMZI

SESSION 2023/ 2024

FACULTY OF COMPUTING AND INFORMATICS
MULTIMEDIA UNIVERSITY
FEBRUARY 2024

GENERATIVE AI RECOMMENDER SYSTEM IN E-COMMERCE

BY

NUR ANIS NABILA BT MOHD ROMZI

SESSION
2023/ 2024

THIS PROJECT REPORT IS
PREPARED FOR

FACULTY OF COMPUTING AND INFORMATICS
MULTIMEDIA UNIVERSITY
IN PARTIAL FULFILLMENT
FOR

BACHELOR OF COMPUTER SCIENCE
B.CS (HONS) DATA SCIENCE

FACULTY OF COMPUTING AND
INFORMATICS
MULTIMEDIA UNIVERSITY
JULY 2024

Declaration

I hereby declare that the work has been done by myself and no portion of the work contained in this thesis has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

_____

NUR ANIS NABILA BT MOHD ROMZI

Faculty of Computing & Informatics

Multimedia University

Date : 26 June 2024

Acknowledgement

I want to express my sincere gratitude to everyone who has helped to bring these complex ideas. I am truly touched by the contributions. My deep appreciation is extended to Dr. Haw Su Cheng, my supervisor for the Final Year Project (FYP), whose direction and mentoring gave me the priceless chance to explore the intriguing field of the Generative AI Recommender System for streamlining corporate operations. In addition to being educational, working on this project has given me the opportunity to conduct in-depth study and gain a variety of new abilities and perspectives. I sincerely appreciate all of the help and encouragement I have received. Realising that every accomplishment, no matter how big or small, takes a team to complete, I would like to sincerely thank my parents and friends. Throughout this journey, their unwavering belief in my ability and support have been a source of inspiration.

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 endeavour even with their broad use. This research explores different approaches to building recommender systems, emphasising 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 literature analysis of recommender system algorithms is part of the process, and the Autoencoder (AE) 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 (AE) 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 with regard to the incorporation of Generative AI, the study endeavours 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 (AE) will highlight the usefulness and efficiency of the suggested strategy and highlight its significance in the continuous development of recommender systems.

Table of Contents

List of Tables

List of Figures

List of Equations

Chapter 1 : INTRODUCTION

1.1 Background

In the modern era of vast informations and knowledges, recommender systems stand as essential aids for users navigating through vast content offerings, aiming to deliver pertinent and personalized experiences. The swiftly evolving realm of Generative AI within artificial intelligence presents an exciting prospect, holding the transformative power to revolutionize recommender systems. This progressive subfield within artificial intelligence addresses the challenges faced by traditional recommender systems and holds the promise of introducing innovative dimensions to user-centric content recommendations. Evidence of this transformative technique's success is observable in various platforms such as Amazon, Netflix, Food Recommendation and E-commerce, where Generative AI has been effectively employed in recommender systems to enhance the precision and personalization of content suggestions.

Customers find it very important in the e-commerce version because, for example, generative AI makes it possible to create highly personalised suggestions by evaluating each user's unique behaviour and interests. By showing customers products that closely match their interests, personalisation 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 closely match their tastes. This optimised 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, customised 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 overall.

1.2 Problem Statement

At the beginning of this project, some problems were addressed and needed to be researched. The research questions are as follows:

1. What are the commonly used recommendation techniques nowadays?

   The user experience is directly impacted by the recommendation strategies selected. Designing systems that provide more precise and tailored recommendations is made possible by experience with various techniques, which raises user happiness and engagement levels. Different recommendation techniques have varying computational requirements and performance characteristics. Knowing these factors makes it easier to choose methods that fit the resources at hand and the intended level of system performance.

2. What recommendation phases need to be carried out to develop an e-commerce recommender system?

   For efficient system design, algorithm selection, user experience improvement, and ongoing system refining, it is essential to comprehend the recommendation phases while creating an e-commerce recommender system. It offers an organised method for creating a system that accommodates consumer preferences and helps e-commerce platforms succeed.

3. What evaluation requirements can be utilised for assessing the model's effectiveness?

Recommendation system optimisation is guided by evaluation metrics. Developers can update the model to offer better user experiences and recommendations by identifying areas for development. Metrics, such as enhanced revenue, higher conversion rates, or increased user engagement, frequently show how the model has affected the firm. Comprehending these data helps match the model with more general company goals.

1.3 Project Objectives

There are various ways in which generative AI could help employees make decisions about customer service. Employees may receive real-time suggestions and ideas from generative AI depending on the specifics related to the customer's engagement. For example, it can appropriately respond to commonly requested questions or issues raised by clients. Step-by-step guides or instructions can be made to help employees recognise and address customer problems. Additionally, generative AI can produce answers or solutions based on information already in the public domain. In this project, an AI generator are used to give staff recommendations on how to do better with service customers. Generative Adversarial Networks (GANs), Recurrent Neural Networks (RNN), Autoencoders, Variational Autoencoders (VAEs), Transformer Model Markov Chains, and so forth are a few examples of potential generative artificial intelligence. Hence, this project aims to:

1. To explore the various Generative AI in recommendation system.
2. To design and implement the selected Generative AI.
3. To evaluate the algorithm through user testing or performance evaluation.

1.4 Scope

This study aims to implement Generative AI in recommender systems in e-commerce sector. The system will generate recommendations for new items based on existing users' purchases. The proposed recommender system will leverage techniques including collaborative filtering applied to an e-commerce dataset. Additionally, Generative AI techniques such as Autoencoders and Long-Short-Term Memory(LSTM) will be incorporated. The system's performance will be assessed using evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Loss Curve Validation, Precision, Recall, F1-Score, Normalized Discounted Cumulative Gain (NDCG) and Similarity Score. The outcomes will be presented through data visualization methods, including charts graphs, and streamlit. This project is centered on utilizing Generative AI techniques within a recommender system to provide tailored recommendations that closely align with existing customers' preferences.

1.5 Organization of Chapters

The following summarises the six chapters, as summarised below :

- Chapter 1 introduces the proposed project and a general summary of this project. Overall overview is given in this chapter. It contains the background of problem descriptions, project goals, project scope, and chapter organization.
- Chapter 2 covers the literature review of the project. It includes an overview of the traditional recommendation systems and the use of Generative AI in enhancing it, recommendation techniques and some related works on the implementation of Generative AI in recommender systems to optimize business processes.
- Chapter 3 covers the theoretical framework for the technique used.
- Chapter 4 proposes the research methodology of the project. This chapter will show how Generative AI technique will be implemented. The evaluation metrics used will also be discussed in this chapter.
- Chapter 5 will illustrate of how the recommendation prototype are used.
- Chapter 6 consists the result of evaluation metrics and testing experienced
- Chapter 7 will wrap up overall research paper, a conclusion and plan for future work activities.

Chapter 2 : LITERATURE REVIEW

2.1 Overview of Recommender System

The importance of recommendation systems (RS) has grown, especially since the emergence of websites like YouTube, Amazon, and Netflix. Recommendation systems assess customer preferences and make proactive suggestions for things that they are likely to purchase based on product information, customer behaviour, and other data. Numerous research investigations have been carried out to create these recommendation systems, and numerous useful systems have been effectively used in a range of industries (Wei et al., 2016). Nowadays, the majority of both 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 customised product recommendations. One creative way to get beyond the constraints of e-commerce services is using recommendation algorithms.

Next, content-based (CB) solutions are widely utilised in a variety of industries, where they are arguably the most popular approach. Just a few of the 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. Taking into account that data can be utilised 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 will often look at the user's past interests and suggest related items or services.

Besides, Collaborative Filtering (CF) systems go beyond by utilising the interests of users similar to you. Drawing from the preferences of users who share similar tastes enables the provision of precise recommendations. The fundamental idea behind collaborative filtering is to find a set of people with whom the target user has common interests.

(Jiang et al., 2019). By considering the preferences of these neighbours, the algorithm predicts the target user's interests. One well-known example is Amazon, a well-known e-commerce site that uses collaborative filtering to recommend items to its consumers. Recommendations based on the things these neighbours like can be made when the system finds a neighbour user for the target user. This group of neighbouring users serves as a standard for recommending items, capturing the essence of collaborative filtering in identifying a cluster of users with shared interests akin to the target user's preferences.

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 utilising 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 the majority of users engaged with two separate items (a product, page, or email) in a comparable way, 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, one of the best ways to increase sales and enhance client retention is to combine many recommender approaches. Combining the best features of both approaches, hybrid 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. Hybrid RS is the result of combining CB and CF (Barragáns-Martínez et al., 2010).

Knowledge-Based (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, in e-learning, this means that these systems use relational knowledge to map learner-relevant learning resources (Tarus et al., 2018).

However, generative AI Intelligence (GAI) is powered by foundation models (large AI models), enabling them to handle a variety of tasks beyond conventional boundaries seamlessly, including summarization, Q&A, classification, and others.

Additionally, GAI can create customised products to satisfy users' unique information requirements, and the recently released ChatGPT greatly helps users express information demands more precisely through natural language commands.

Generative AI Intelligence (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. Additionally, 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 recommendations that are more contextually relevant. Generative AI algorithms are employed by e-commerce platforms like Amazon to provide tailored product recommendations that are predicated on consumers' past purchases and browsing patterns.

2.2 Phases in the Recommendation Process

2.2.1 Traditional Recommendation System

Creating recommendation systems involves several key steps to provide users with personalised suggestions. First, we collect information using explicit (like ratings) and implicit (like clicks) feedback. Then, we move on to the learning phase, where the system uses this data to understand user preferences. This involves creating profiles that represent what users like and what items offer. Finally, in the recommendation phase, the system uses what it learned to predict and suggest items users might enjoy but have not seen yet. This process ensures users receive recommendations tailored to their unique tastes, making their experience more enjoyable. Figure 2.1 shows the implementations behind the recommendations phases.

Commented [d1]: First figure in Chapter 2, should be Figure 2.1, follow by Figure 2.2, etc.

Figure and Table is labelled based on Chapter as prefix.running number.

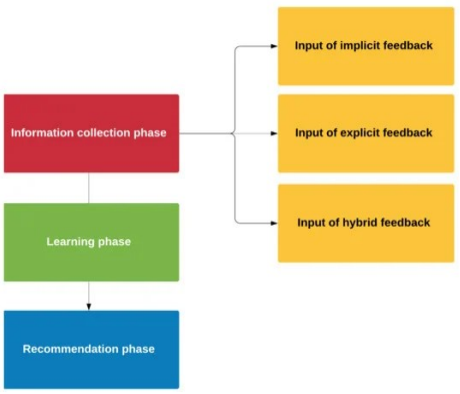


Figure 2. 1 Phases in Recommendation System

1. Information Collection Phase

A robust user profile or model is a prerequisite for an accurate recommendation agent. In order to create a user profile or model for the prediction of taste, this first stage gathers pertinent user data, such as the user's attributes, behaviours, and the content of the resources they visit. To successfully make recommendations that are appropriate for the user's preferences, the system requires as much information from the user as possible. Therefore, the model's capacity to represent users' current choices or preferences is a key factor in the effectiveness of a recommender system or recommendation. User's input data can be gathered in three ways, which include explicit, implicit and hybrid feedback.

Commented [d2]: Should be indent same as the start of the start of Chapter. Please change for all.

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 easier and less demanding for the user (Isinkaye et al., 2015)

Since explicit feedback solicits the user's direct input regarding product preferences, its efficacy depends on its accuracy. This kind of input requests ratings for multiple products from users via the system interface, which helps build and

improve the recommendation model. The quantity and calibre of these user-provided ratings directly affect how accurate the recommendations are. 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 is still regarded as providing more reliable data even though it necessitates more work from users. Additionally, because it offers transparency into the recommendation process, it raises perceived recommendation quality and increases confidence in the recommendations (Isinkaye et al., 2015)

Additionally, Users who wish to show their interest directly can only provide feedback through hybrid feedback, which combines explicit and implicit feedback rating. This approach empowers users by allowing them to provide feedback explicitly, emphasising 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 feedback and ratings while using indirect data as a recommendation attribute.

2. Learning Phase

The user data collected during the information gathering phase is subjected to learning algorithms in this phase. This phase's trained data offers specific patterns that are used 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 (Isinkaye et al., 2015).

3. Recommendation Phase

The phase of recommendation suggests the kinds of products that a customer or user would find appealing. Recommendations may be given directly from the dataset gathered in the information gathering stage (which may be model- or memory-based), or indirectly via the system's observation of users' browsing histories. The available filtering techniques will be discussed in Section 2.3.

2.2.2 Generative AI

Within the swiftly evolving landscape of artificial intelligence, Generative AI emerges as a rapidly advancing subfield with the transformative potential to revolutionise 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.

1. Data Collection and Pre-processing

Commented [d3]: Change to proper alignment (indentation as commented earlier)

First, 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 for it to be effectively used in later stages. 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.

2. Model Architecture Selection

Selecting a suitable model is essential in order to achieve the desired results. Different models offer distinct capabilities that suit specific tasks. Some of the common model architectures used in generative AI are Variational Autoencoders (VAEs) that can be used to learn the distribution of the dataset. Next, Generative Adversarial Networks (GANs) generate realistic images and other creative content and finally, Autoregressive models that can predict the next word in sequence. Generative model layers train the model to understand patterns in user-item interactions by using GANs or VAEs to generate new interaction recommendations. GANs involve a generator creating new samples and a discriminator providing feedback, while VAEs focus on encoding and decoding latent representations.

3. Model Training

The training procedure starts after the model architecture is chosen. A dataset containing input data is used to train the model. The training procedure is contingent upon the chosen model, typically encompassing techniques such as gradient descent, backpropagation, and regularisation to optimise model parameters. The primary aim during training is to minimise the disparity between the model's output and the actual data, known as the ground truth. This optimization process enables the model to produce content that closely mirrors the patterns present in the training data. This allows the model to generate content that resembles the training data.

4. Evaluate and Refine

Now, it is the time to evaluate the output. This is done by comparing it to pre-defined metrics and benchmarks. These metrics will help you assess the generated output's quality, coherence, and realism. If the results fall short of your expectations, you can iterate over the model for refinement. This iterative process might entail modifying the architecture, adjusting training parameters, or refining the dataset. Keep in mind that achieving optimal performance is an incremental journey, and it is important to go through this process multiple times before achieving the desired satisfaction with the model's output.

5. Test and Validate

After being satisfied on how the generative AI model is performing, it becomes important 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 generalise to new data. Additionally, assess the model's capability to produce varied and coherent outputs in diverse scenarios. Lastly, compare the generated results against human judgement and domain-specific criteria to ensure the model's reliability and effectiveness.

6. Deploy and Iterate

Finally, the deployment and integration layer is where the generative AI process ends. The improved model is incorporated into useful 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 feedback and opinions from users, monitor model’s performance and make changes in the meantime. Continuous improvement will always help to keep the model relevant and effective. Figure 2.2 shows the end-to-end process of generative AI.

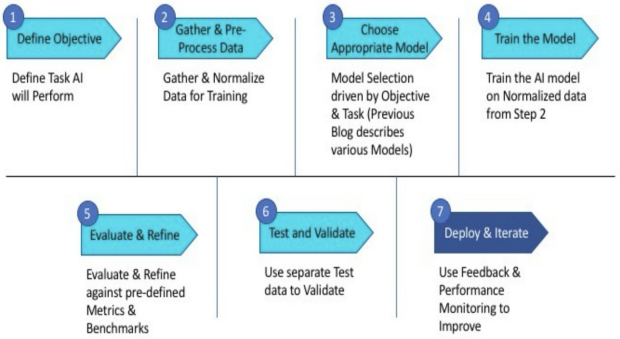


Figure 2. 2 End-to-end process of generative AI

2.3 Recommendation System Techniques

2.3.1 Traditional Recommendation System

Recommendation systems employ various techniques to analyse user preferences and behaviours, providing tailored suggestions. Common techniques used include collaborative filtering, which identifies patterns based on user-item interactions, and content-based filtering, which makes suggestions for products based on what a user has already expressed interest in. Hybrid methods also incorporate several strategies to improve the precision of recommendations. The structure of several suggestion filtering methods is depicted in Figure 2.3.

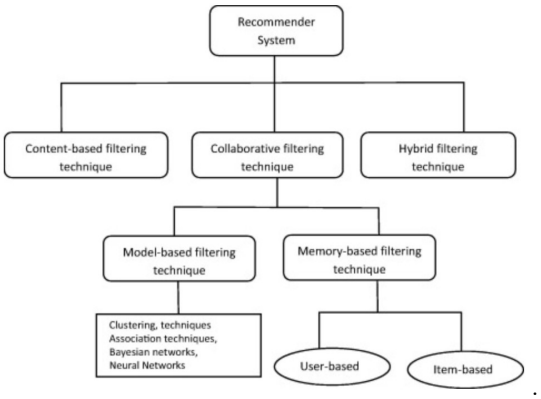


Figure 2. 3 structure of several suggestion filtering methods

1. Content-based filtering (CB) (user-item)

Commented [HSC4]: Content-based is CB, not CF.

Content-based approaches leverage item or user metadata to formulate tailored recommendations. This involves examining the user's purchasing history as a key factor. For instance, It is assumed that a user prefers a certain author or brand if they have interacted with that author's book or brand's product in the past. Additionally, there's a chance they'll purchase a comparable item later on. A basic illustration of a content-based recommender system is shown in Figure 2.4.

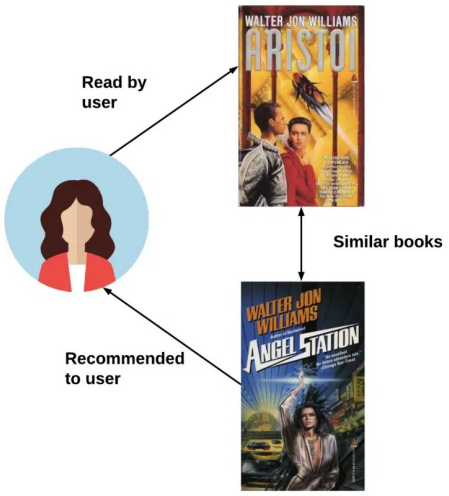


Figure 2. 4 Content-based recommender system

Consider Jenny, who loves sci-fi books and has a particular fondness for the works of Walter Jon Williams. If she indulges in the Aristoi book, her subsequent recommended read would be Angel Station, another sci-fi creation by Walter Jon Williams. This practical illustration mirrors the concept of content-based filtering in action. The advantage gained from this approach is it contains transparency, which allows users to understand why a recommendation is made since it is predicated on the characteristics and substance of things they have already engaged with. It can then offer a variety of recommendations. A movie recommendation system, for instance, might base suggestions on the performers, director, and genre.

The three components of CB RS's architecture are a content analyzer, a profile learner (generator), and a filtering component. Figure 2.5 illustrates this architecture. Using feature extraction techniques, the Content Analyzer gathers item content from various information sources and derives item representations. The Profile Learner generates a user profile based on previous likes and dislikes by generalising 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 (Narducci et al., 2015)

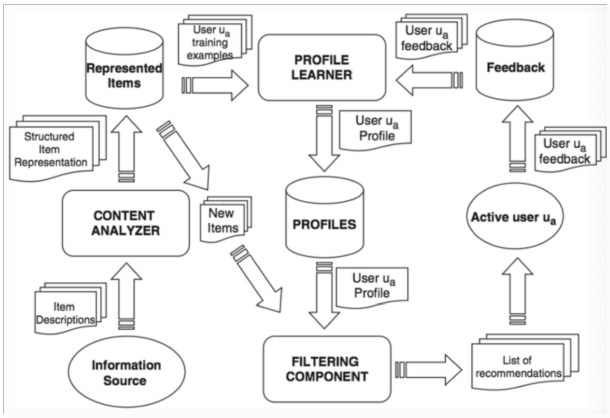


Figure 2. 5 Architecture of Content-based

Additionally shown is a content-based book recommendation engine that uses online book information. The system creates a user profile by analysing data taken from the web using a Naïve Bayes classifier. With the use of user-provided training examples, this profile plays a crucial role in producing a prioritised 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. Collaborative filtering (CF) (user-item)

In order to find new user-item correlations, collaborative filtering examines user relationships and product interdependencies. User behaviour or user evaluations of recommended things serve as the foundation for CF recommendations. It investigates a variety of potential material and suggests items loved by users who are similar to you (Al-ghuribi et al., 2016). 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 are those that take into account 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.

Memory-based recommenders

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 personalised recommendations. User-based and item-based collaborative filtering are the two primary categories into which memory-based recommenders fall. User-based collaborative filtering is depicted in Figure 2.6.

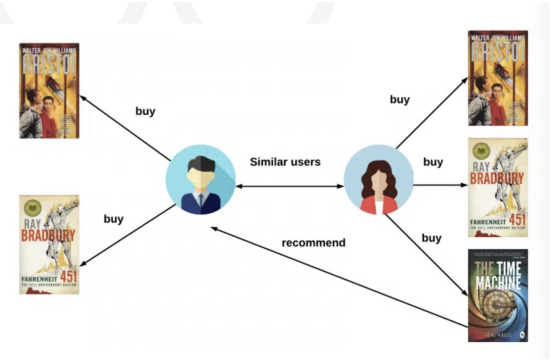


Figure 2. 6 User-based Collaborative filtering

The user-based approach involves suggesting items to the target user by identifying others with similar behaviour or preferences. This entails finding users who closely resemble the target user based on their 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 recommenders

Figure 2.7 shows the item-based collaborative filtering. This technique 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.

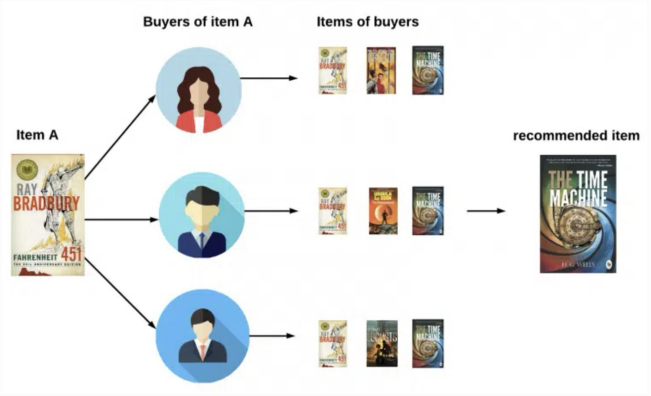


Figure 2. 7 Item-based Collaborative filtering

Next, model-based recommenders make use of machine learning models to generate recommendations. These systems delve into historical user-item interaction data, diligently extracting patterns, correlations, and relationships. By assimilating insights from this data, the model-based recommenders can predict a user's preferences for items they have not engaged with previously. Within the realm of model-based recommenders, various techniques come into play, showcasing the versatility of this approach. Among these techniques are matrix factorization, Singular Value Decomposition (SVD), and neural networks, each contributing distinct strengths to generating accurate and tailored recommendations.

Matrix factorization is a common approach of performing Collaborative Filtering in spite of this. The models can incorporate additional information like implicit feedback, temporal effects, and confidence levels, making them preferable to traditional nearest-neighbor strategies for making product suggestions. Matrix factorization forms the basis of some of the most effective latent factor model realisations. Matrix factorization, in its most basic form, uses vectors of factors that are 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 there isn't any explicit feedback available. Indirect feedback is the expression of opinions through user behaviour, 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. Hybrid-based filtering

Combining multiple recommender methods to provide more personalised, differed, and effective recommendations is one of the best ways to increase sales and improve client retention. Especially in instances where recommendations are made in real life, they might generate more dependable, precise, and adaptable ideas. Content-based and collaborative filtering are combined, and both filtering 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 utilising 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 films based on user ratings that are similar (content-based filtering) and by 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 (Isinkaye et al., 2015).The metalevel hybrid technique learns a second recommendation algorithm

from a first algorithm by using the full model as input(Fayyaz et al., 2020). Figure 2.8 shows how the hybrid-based filtering works.

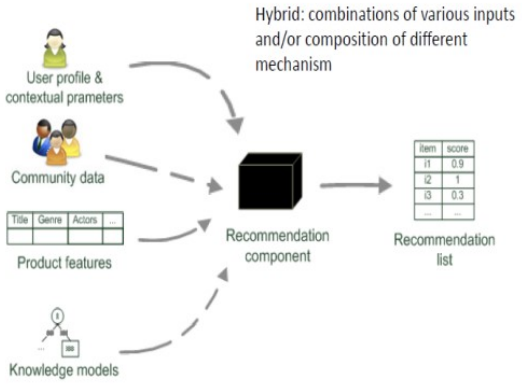


Figure 2. 8 Hybrid-based filtering

Some of the good consequences that we can gain from using hybrid recommenders are that it is frequently robust in managing different suggestion scenarios. They can adjust to various user behaviours, data features, and recommendation difficulties. Such adaptability is useful 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.

2.3.2 Generative AI Recommendation Techniques

Recently, recommender systems have benefited from the application of various AI techniques, which have improved user happiness and the overall user experience. AI makes recommendations that are of a higher calibre than those made with traditional techniques. Automation of intelligent behaviour is the aim of AI technology development. These six domains include knowledge engineering, reasoning, planning, communication, perception, and motion. (Zhang et al., 2021).Hence, the next step will explain techniques that are applied to achieve the desired outcome. Figure 2.9 explains the AI areas and the techniques used.

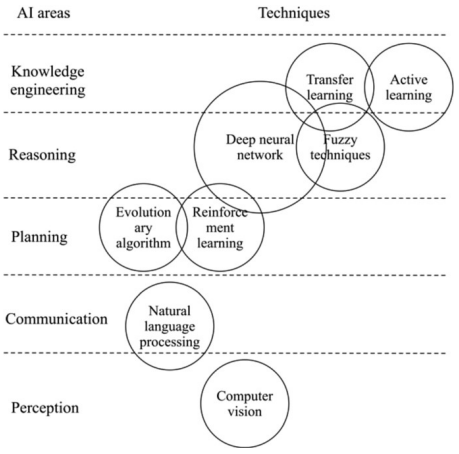


Figure 2. 9 The AI areas and the techniques used

Transfer learning has emerged as a method for knowledge engineering, whereby information is transferred transfering data from a large source domain to a small target domain (Zhang et al., 2021).According to this definition, transfer learning is using 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 purpose in order to maximise machine learning performance with minimal information requirements. Users may be asked to name occurrences that are not labelled by an active learning system. Active learning has useful applications in different AI disciplines, especially well-suited for online systems, since labelling can be expensive, time-consuming, and sometimes impractical. For reasoning, the techniques used are deep neural networks(DDNs) and fuzzy. DNNs can model non-linear 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 be used to replicate real-world concepts that cannot be properly described, they are frequently used in the field of artificial intelligence (AI). Fuzzy approaches have drawn a lot of interest in the literature. For example, fuzzy distance has been used to describe similar instances, and fuzzy sets have been utilised by researchers to represent linguistic variables when feature values cannot be adequately expressed in numerical values (Zhang et al., 2021).

Afterwards, reinforcement learning and evolutionary algorithms participate in the planning domain. Evolutionary algorithms (EA) are a subclass of population-based search algorithms for global optimisation in artificial intelligence that are inspired by natural processes. A number of 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 put an end to it. The recommender system is seen as a learning agent in reinforcement learning, user behaviours 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. Finding a value function or method that enables consumers to maximise long-term advantages is the aim.

In order to address the problem of data sparsity in communication, the majority of recommender systems also enrich the rating matrix with review data that is acquired via natural language processing. In extreme cases, virtual ratings are generated utilising the emotion polarity that is obtained from review classification in the absence of 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. (Q. Zhang et al., 2021)

In summary, the direct application of computer vision in picture recommendation—that is, the mapping of 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 (Zhang et al., 2021). This was extended in later study by using low-level data, like colour qualities, that mimicked parts of the human visual system. Zhao et al. 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. (Zhao et al., 2016). Point-of-interest recommendations have also made use of visual content, taking use of the wealth of landmarks included in pictures and images uploaded by users.

2.4 Related Works

In the study by (Ferreira et al., 2020) 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 a number of 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 greatest choices and is renowned for its speed, low memory usage, and ability to handle big datasets had an impact on the optimizer selection—a strategy that is consistent with this article. The evolution of loss and validation loss (val_loss) show a declining trend over the course of each epoch, ultimately stabilising 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 achieved 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 dataset and 0.010 for the second dataset.

In the following study, (Tran et al., 2018) 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. Characterised by unique features and tuneable 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 it easier to identify the best performance parameters for a particular dataset. The ADAM optimizer, the SELU activation function, and one round of dense refeeding with mean normalisation enabled were among the common 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 DAE model's prediction accuracy is greatly impacted by the parameters selected. 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 (G. Zhang et al., 2020) emphasised 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 important 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., 2021) 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). The innermost layer activations of a deep autoencoder are used by DAFERec 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, thereby 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 tenfold cross-validation technique was employed by the authors to confirm the dependability of their findings

Furthermore, based on (Loukili et al., 2023), they have succeeded 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 really good results, with a high chance that customers will actually buy the next product suggested by the system. The focus of the study is 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 next suggested product. The evaluation metric serves as 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 are investing more money in social media, e-commerce, mobile, and internet channels to improve customer engagement and communication with both current and potential customers and increase sales. Thus, studies conducted by (Grewal et al., 2020) explains in detail how predictive analytics is useful for estimating people's product preferences, price sensitivity, and expected next steps on the customer journey—all of which are critical for improving business activities. This is accomplished by utilising 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 tecniques used for recommendation system on Amazon discussed by (Rybakov et al., 2018)mentioned in this study. The use of neural networks in a

personalised 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, the usage of offline assessment metrics—like Product Converted Coverage (PCC) at K and Precision at K—that are frequently employed in real-world recommender system applications is presented in this work. The usage of a deep learning package that facilitates multi-GPU model parallel training 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 utilisation 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 it increased accuracy metrics such as the research has demonstrated 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., 2022)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. Due to

the fact that the ratings are typically broad, this method guarantees strong categorization performance. The authors utilise Stacked AutoEncoder, a skilled nonlinear feature reduction approach, to leverage the high-dimensional classification dataset and produce a 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, the mapping relationship between the output ratings and the high-level representation is described by a softmax model. 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.

Additionaly, (Namazi et al., 2022) proposed that utoencoders, when utilized as generative models in recommendation systems, can significantly enhance the personalization and accuracy of the recommendations. When it comes to creating fresh user-item interaction data, these models are particularly good at predicting possible interests or preferences that the user may not have clearly stated in their initial interactions. Generative autoencoders, for example, are useful for data imputation jobs because they can accurately forecast missing entries in sparse user-

item matrices. By learning to rebuild known portions of the data and infer missing values, this feature is especially useful for tackling the prevalent problem of data sparsity in recommendation systems, resulting in more comprehensive user profiles. Additionally, by creating new, synthetic user-item interaction data based on the latent characteristics discovered during training, generative autoencoders enable improved feature learning. With less past interaction data available for new users or objects, this feature helps solve cold-start issues and improves the system's ability to offer pertinent recommendations from the outset.

Next, (Lacic et al., 2020) recommendation approach shown in this work encodes sessions inside the job domain using several autoencoder architectures. To recommend jobs within a session, the inferred latent session representations are used in a k-nearest neighbour fashion. Three autoencoder types are used in the study: variational autoencoder (VAE), denoising autoencoder (DAE), and classical autoencoder (AE). The input and output of the most basic AE are separated by a single hidden layer. 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, variational inference is used by VAE 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).

Long-Short Term memory also one of the Generative AI or Generative model that could be use in the recommendation system based on (Noorian et al., 2024) which it clarifies that the Neural Network-Long Short-Term Memory (LSTM) POI recommendation system can determine user similarity based on preferences and opinions. Furthermore, it offers a deep learning approach for finding sequential travel suggestions with Bidirectional Encoder Representations from Transformer (BERT). LSTM also can extract the sequential as well as semantic information from social media messages. The suggested model can deliver consumers recommendations that are more relevant and accurate by taking advantage of the contextualised dependencies of LSTM and the sequential representations of BERT.

To back up this concern, (Reddy & Kumar, 2023) suggest a CNN-Bi-LSTM model-based personalised health product recommendation system for online sales. The Bi-LSTM serves as a numerical function to provide ratings, and our system uses pre-trained CNN-based transfer learning models, such as AlexNet, Google Net, and ResNet-50, to forecast health items. By using attribute-specific representation extraction techniques, the CNN-Bi-LSTM recommendation system produces a stronger compatibility model and more effective health product recommendations. Bi-LSTM aids in product numerical rating. Sequence processing models with two LSTMs that receive input both forward and backward are called bidirectional LSTMs, or BiLSTMs. BiLSTMs improve algorithm context by providing the network with more data. Both sides' information can be utilised by Bi-LSTM. Product data is stored in the convolutional layer, which also projects the CNN.

This work builds a strong recommendation system using CNN and Bi-LSTM. The goal of the Bi-LSTM architecture is to develop a recommendation system that is data-efficient. While the application of deep learning to recommendations is not new, it is to e-commerce.

Lastly, (Sachdeva et al., 2019) 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 through without taking temporal dependencies into account. In this configuration, the sequence passes through a number of fully-connected layers at each RNN time-step. The probability distribution of the most likely future preferences is modelled 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 with respect to 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 a number of criteria. The evaluation concentrates on top-N recommendation while taking implicit preferences into account, using measures like NCDG, Precision, and Recall.

2.5 Summary of Related Works

Commented [d5]: I noticed most of your related works are not in business processes. You may need to replace.

Keep in mind: In this project, we propose to implement an AI generator to provide recommendation to employee to serve customer better.

So, you need GAI for recommender system in customer service support, etc.

Table 2.1 summarizes the related work reviwed. The table shows that AE and VAE techniques are most applied in Geneerative AI for recommendation system. The techniques outperformed other filtering techniques and accuracy including the traditional one. Besides, compared with other evaluation metrics, most of the studies uses MAE, MSE and RMSE metrics to assess the recommendation system's performance and accuracy.

Table 2. 1 Summarize of Related Work

References	**Title**	**Findings**	**Dataset**	**Evaluation Metrics**
(Ferreira et al., 2020)	Recommendation Systems Using Autoencoders	Challenges arose while using matrix factorization techniques such as SVD due to large dataset but autoencoder was used to overcome this issues and achieve promising results for RMSE.	Amazon, Macys, Shop_norstrom	RMSE

(Tran et al., 2018)	Deep autoencoder for recommender systems: Parameter influence analysis	The findings implemented a deep AutoEncoder(DAE), known as FlexEncoder model for recommendation systems where the model integrates features from various sources, allowing a thorough analysis of parameter influences on prediction accuracy. The model successfully achieved lower RMSE compared to AutoRec and	MovieLens 100K	RMSE loss

		SVD++		
(G. Zhang et al., 2020)	A survey of autoencoder-based recommender systems	This paper analysed that previous traditional recommender system has some issues with cold start and data sparsity which cause large reduction in recommender system. Hence, the use of AE has become a powerful method extracting important features from data. They also analysed evaluation metric that frequently	None	None

		used for AE are recall, MAP and NDCG.		
(Rama et al., 2021)	Deep autoencoders for feature learning with embeddings for recommendations: a novel recommender system solution	A method called DAFERec which combines features from autoencoders and embeddings in order to improve the learning pattern of items that customer interested in. They also make sure their method is reliable by using technique called tenfold cross-validation.	MovieLens 100K, MovieLens 1M, FilmTrust, BookCrossing	MAE, MSE, RMSE
(Loukili et al., 2023)	Machine learning based recommender system for e-	This article discusses the development of an e-commerce	Online Retail	P average

	commerce	recommender system based on machine learning, with an emphasis on the use of association rules and the Frequent-Pattern-Growth algorithm. In addition to discussing the difficulties in giving customers accurate recommendations, the article analyses relevant work in the field of recommender systems and illustrates the significance of personalised		

		recommendations in e-commerce. The performance of the recommendation system is also evaluated, and the study's flaws are covered.		
(Grewal et al., 2020)	The future of technology and marketing: a multidisciplinary perspective	Using large data and sophisticated analytics tools to obtain insights and provide tailored suggestions based on consumer information is how artificial intelligence (AI) is being implemented in recommender systems to	None	None

		optimise business processes. Predictive analytics, for example, can be used to forecast people's price sensitivity, product preferences, and likely next actions on the customer journey, so optimising business operations. 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.		
(Rybakov et al., 2018)	The effectiveness of a two-layer neural network for recommendations	The study describes a two-layer neural network-based personalised recommender system that makes product recommendations based on past implicit customer input. Furthermore, the authors made their deep learning library available to the research and development community	Amazon history dataset	Precision at K, Product Converted Coverage(PCC)

		through open-sourcing. This library facilitates multi-GPU model parallel training. Additionally, the study contrasts several implicit feedback-based recommendation models, including analysis on how well they work and whether or not they are appropriate for real-world use.		
(Yu et al., 2022)	A model-based collaborate filtering algorithm based on	They proposed a solution named MCFSAE, a method that transforms the available	EachMovie, MovieLens	MAE, RMSE

	stacked AutoEncoder	information into a format that's easier to work with because of data sparsity problem. The method uses Stacked AutoEncoder to process information and provide helpful information.		
(Namazi et al., 2022)	Autoencoders in generative modeling, feature extraction, regression, and classification	By enabling the creation of fresh user-item interaction data that can reveal prospective interests or preferences not explicitly represented in the	None	None

		user's original interactions, autoencoders—especially when used as generative models—offer interesting possibilities in recommendation systems. This generative capacity can greatly improve suggestion accuracy and personalisation.		
(Lacic et al., 2020)	Using autoencoders for session-based job recommendations	The paper applied autoencoders to understand and represent job-related recommendations	Austrian student job portal Studo Jobs, RecSys 2017 Challenge,	MRR, EPD, EDC, NCDG

		by using AE, DAE and VAE. Final result state that When compared to cutting-edge session-based recommendation algorithms, VAE's job recommendations are more accurate.	Anonymized job applications	
(Noorian et al., 2024)	A sequential neural recommendation system exploiting BERT and LSTM on social media posts	This study proposes a hybrid LSTM and BERT-based POI recommendation system that uses tourist opinions and demographic data to create personalized trip plans, addressing the cold start	Tripadvisor, Yelp	F-Score, NDCG, RMSE, MAP

		issue. Experimental results on Tripadvisor and Yelp datasets demonstrate its superior performance in key metrics compared to traditional methods.		
(Reddy & Kumar, 2023)	An E-Commerce Based Personalized Health Product Recommendation System Using CNN-Bi-LSTM Model	This study introduces a CNN-Bi-LSTM-based E-commerce recommendation system for personalized health products, enhancing accuracy and transparency by	Health-product dataset, Flipkart health product dataset	Accuracy, Recall, User Coverage

		using pre-trained CNN models and Bi-LSTM for ratings. Evaluated on two health product datasets, the system outperforms existing methods, achieving higher accuracy, recall, and user coverage.		
(Sachdeva et al., 2019)	Sequential variational autoencoders for collaborative filtering	This paper introduced new and improved version of VAE by considering the order and timing of activities like watching movies or reading articles rather than just looking at the	MovieLens 1M, Netflix	NDCG, Presicion, Recall

		entire history without paying attention to when things happened. They used RNN to do that. This improved model performed much better than the existing top-notch methods.		

2.6 Summary of chapter

In this chapter, the insightful overview of the generative AI recommender system and its role in enhancing business processes, with a particular focus on the dynamic landscape of e-commerce has been provided. Also, this chapter delves into the various phases of the recommendation process, discussing each in detail. Moreover, it is also highlighted the importance of related work, emphasizing potential methodologies that pave the way for further exploration in the subsequent chapter. By suggesting the integration of conventional recommender systems, which include content-based, collaborative, and hybrid filtering, this study adopts a progressive stance. with generative AI. The aim is to leverage the strengths of each to achieve optimal results, marking a significant advancement in the recommendation system.

Chapter 3 : THEORETICAL FRAMEWORK

3.1 Matrix Factorization

Matrix factorization is a method that involves multiplying matrices that represent two distinct types of things in order to produce latent characteristics. Matrix factorization is used in collaborative filtering to determine the links between items and users. Our goal is to forecast users' evaluations of other goods by using their ratings of shop items as input. This will allow us to provide personalised suggestions based on these predictions.

Mathematics concept of matrix factorization can be defined as a set of Users (U), items (D), R size of |U|, and |D|. The matrix |U|*|D| includes all the ratings given by users. The goal is to discover K latent features. Given with the input of two matrics matrices P (|U|*k) and Q (|D|*k), it would generate the product result R as shown in Equation 3.1:

$$\mathbf{R \approx P \times QT = R\hat{}}$$

Equation 3. 1

3.1.1 Singular Value Decomposition

SVD is a factorization of a real (or) complex matrix that applies an extension of the polar decomposition to any m x n matrix, generalising the eigen decomposition of a square normal matrix. To put it another way, as shown in Equation 3.2 below,

SVD approximates any dimensions matrix into three smaller dimensional matrices while "rotating and scaling" to preserve the maximum variance in the form of matrices USV with "top-k eigen vectors and eigen values."

$$A = U\ S\ V'(Final\ rotation\ ||\ Scaling\ ||\ Initial\ rotation)$$

Equation 3. 2

where:

U = left singular valued matrix ,

S = sigular valued matrix,

V = right singular valued matrix.

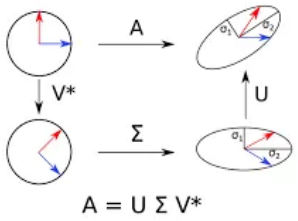


Figure 3. 1 Rotated Metrices

Figure 3.1 depicts the rotated matrices consists of transformed data points, whose eigen vectors make up singular matrices U or V while the Scaled matrix have values scaled by a factor. U,V are orthogonal matrices, represents the rotations or reflection of the space which are eigen vectors that are orthonormal to each other.

3.2 Generative Model

3.2.1 Autoencoder(AE)

Neural networks are designed for efficient coding, dimensionality reduction, and generative modelling are good candidates for unsupervised learning tasks, such as auto-encoders. It has been useful in learning underlying feature representation in a number of 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 Figure 3.1 below. 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. The output layer is where the output data exits.

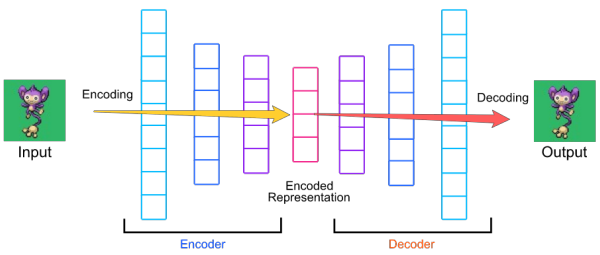


Figure 3. 2 Autoencoder architecture

The encoder encodes the high-dimensional input data x into a lower-dimensional hidden representation h with a function f in Equation 3.3 below:

$$h = f(x) = Sf(Wx + b)$$

Equation 3. 3

where s_f 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 3.4:

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

Equation 3. 4

Where :

s_g is an activation function, W' is the weight matrix, and b' is the bias vector.Non-linear options for s_f and s_g include Sigmoid, TanH, and ReLU. More meaningful features can be learned by the auto-encoder 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 in order to minimise the reconstruction error between x and x'. The formula for the squared error as per Equation in 3.5.

$$SE(x, x') = ||x - x'||^2$$

Equation 3. 5

3.2.2 Long Short-Term Memory (LSTM)

LSTM can be used as a generative model to understand the sequence in which users interact with items. Once the model has learned these patterns, it can take a user's recent activity and predict and recommend items the user might like next based on these learned patterns. For instance, imagine having a large dataset showing user interactions with items, such as movies they've watched or products they've purchased. An LSTM model can be trained on this data to understand and predict user preferences effectively. A generative LSTM could conceivably use in a simple Vanilla LSTM. Figure 3.3 ilustrates the architecture of Vanilla LSTM Architecture for Generative Models.

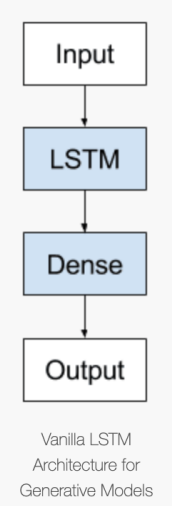


Figure 3. 3 Vanilla LSTM Architecture for Generative Models

LSTM architecture also involves memory cell which is controlled by three gates that consist the input gate, the forget gate, and the output gate in control a memory cell in

LSTM systems. Information added to, deleted from, and output from a memory cell are all controlled by the input gate, forget gate, and output gate, respectively. By using this approach, long-term dependencies can be learned by LSTM networks by allowing them to choose retain or discard information as it flows through the network. In addition, the LSTM serves as the network's short-term memory by maintaining a hidden state that is updated in response to input, the hidden state from before, and the current state of the memory cell.

Forget gate decides what information to discard from the cell state. It takes the current input (xtx_txt) and the previous output (ht−1h_{t-1}ht−1), combines them, and processes the result through a sigmoid activation function. If the output is 0, the information is forgotten; if it is 1, the information is retained. Equation 3.6 below explain mathematics behind forget gate that being used in LSTM architecture model.

$$\boldsymbol{ft = \sigma(Wf \cdot [ht - 1, xt] + bf)}$$

Equation 3. 6

where:

$\boldsymbol{Wf}$ = Weight matrix for the forget gate
$\boldsymbol{[ht - 1, xt]}$ = Concatenation of previous hidden state and current input
$\boldsymbol{bf}$ = Bias term
$\boldsymbol{\sigma}$ = Sigmoid activation function

Next, input gate adds useful information to the cell state. It regulates the input using a sigmoid function and creates a vector of potential values with a tanh function.

The results are multiplied to update the cell state as depicts in Equation 3.7 and 3.8 below.

$$\boldsymbol{it} = \boldsymbol{\sigma}(\boldsymbol{Wi} \cdot [\boldsymbol{ht} - \mathbf{1}, \boldsymbol{xt}] + \boldsymbol{bi})$$

Equation 3. 7

$$\boldsymbol{Ct}\text{^} = tanh(\boldsymbol{Wc} \cdot [\boldsymbol{ht} - \mathbf{1}, \boldsymbol{xt}] + \boldsymbol{bc})$$

Equation 3. 8

where:

$\boldsymbol{it}$ = Input gate's output

$\boldsymbol{Ct}\text{^}$ = New candidate values

$\boldsymbol{Wi}, \boldsymbol{Wc}$ = Weight matrices

$\boldsymbol{bi}, \boldsymbol{bc}$ = Bias terms

$\boldsymbol{tanh}$ = Tanh activation function

Additionally, cell state update combines the old cell state and new information to update the cell state as shown in Equatio 3.9 below.

$$\boldsymbol{Ct} = \boldsymbol{ft} \odot \boldsymbol{Ct} - \mathbf{1} + \boldsymbol{it} \odot \boldsymbol{Ct}\text{^}$$

Equation 3. 9

where:

$\odot$ = Element-wise multiplication

Finally, output gate determines the output from the cell state. It applies a tanh function to the cell state and uses a sigmoid function to filter the result as represent in Equation 3.10

$$\boldsymbol{ot} = \boldsymbol{\sigma}(\boldsymbol{Wo} \cdot [\boldsymbol{ht} - \mathbf{1}, \boldsymbol{xt}] + \boldsymbol{bo})$$

Equation 3. 10

where:

$\boldsymbol{ot}$ = Output gate's output

$\boldsymbol{Wo}$ = Weight matrix for the output gate

$\boldsymbol{bo}$ = Bias term

Lastly, the hidden state (*ht*) acts as hort-term memory, is then updated based on the cell state and the output gate:

$$\boldsymbol{ht} = \boldsymbol{ot} \odot tanh(\boldsymbol{Ct})$$

Equation 3. 11

3.3 Summary Of Chapter

In summary, Autoencoders (AE) and Long Short-Term Memory (LSTM) networks are powerful tools in generative AI for recommendation systems. Autoencoders, which consist of an encoder and decoder, are used to learn efficient representations of data by reducing the dimensionality and then reconstructing the original input. In recommendation systems, autoencoders are employed for collaborative filtering by generating user or item embeddings, capturing the underlying structure of user-item interactions, and predicting missing ratings or preferences. They are also useful in content-based recommendations by encoding content features to identify semantic similarities between items. LSTM networks, a type of recurrent neural network (RNN) designed to remember long-term dependencies, are particularly effective in sequence prediction tasks. In recommendation systems, LSTMs can model user behavior over time, capturing temporal dynamics and sequential patterns in user interactions. This enables more accurate and personalized recommendations based on the user's sequential behavior. Both autoencoders and LSTMs handle complex, non-linear relationships in data, making them essential components in modern recommendation systems that aim to provide tailored and context-aware suggestions to users.

Table 3. 1 Avantages and Disadvantages of Matrix Factorization and Generative Models

Model	Advantages	Disadvantages
Matrix Factorization	- Efficient at capturing latent factors from large, sparse datasets - Widely used and well-understood with robust implementations - Provides interpretable latent factors	- Assumes linear interactions between latent factors, which may limit capturing complex relationships - Sensitive to data sparsity and may require additional regularization techniques - Requires careful tuning of hyperparameters like the number of latent factors
Generative Models	- Can generate realistic synthetic data, enhancing diversity in	- Risk of mode collapse (generative models may produce

	recommendations - Effective in learning representations from multimodal data sources - - Handles various types of data (e.g., images, text)	limited diversity in generated samples) - Interpretability can be a challenge compared to traditional methods like matrix factorization - Requires large amounts of training data and extensive hyperparameter tuning

Chapter 4 : RESEARCH METHODOLOGY

4.1 Research Methodology

Theoretical ideas has been delved in the previous chapter, concentrating on Autoencoder and Variational Autoencoder. The foundation of the useful actions described in this chapter are these models. The following parts will offer a thorough explanation and verification of the theoretical concepts presented previously. The research methodological flow used for this project, which directed the application and validation of the principles mentioned, is depicted in Figure 4.1.

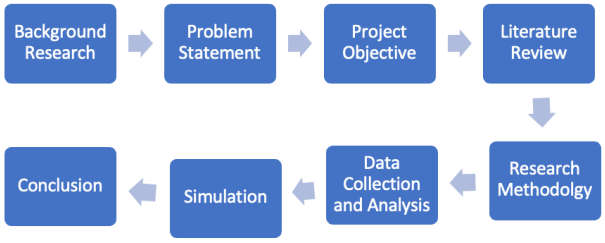


Figure 4. 1 Research Methodology Flow

Based on Figure 4.1, the research methodology begins with a thorough background investigation. During this stage, a lot of discussions was had related to recommender systems, which are essential tools that help users navigate large amounts of content in order to provide relevant and personalised experiences. The integration of "Generative AI in recommender systems", “E-commerce recommendation” and “Gen-AI in Optimiziming Recommendation Product” were

investigated using focused keyword searches at first on reputable academic search engines, including ScienceDirect, Research Gate, and Google Scholar. Additional perspectives were obtained from pertinent news and articles obtained from reliable sites such as Towards Data Science, Medium, and Analytics Vidhya. The thorough analysis of many papers improved knowledge and made it easier to define project goals, identify the problem statement, and define the parameters of the research study that would follow.

Subsequently, a thorough literature study was then conducted, covering discussions on important findings, method selection, and assessment matrices to provide an all-encompassing viewpoint on articles pertaining to the research issue. Significant insights and ideas were gained by closely examining and reviewing research papers written by subject area experts. This allowed for a more efficient and successful structuring of the research method. In order to progress the research, essential steps such as data collecting, exploratory data analysis, and the implementation phase were methodically carried out. Ultimately, a final report was written that summarised all of the processes that were followed, the achievements that were made, and the future works that needed to be explored going forward.

4.1.1 Hardware and Software Requirements to Develop the Protoype

Using the selected RS technique, the recommendation system will be created in this project to provide users with suggestions. To find out whether a system can

create or run software, system requirements are crucial. Table 4.1 is a list of the hardware requirements that were used in the development of the prototype.

Table 4. 1 Hardware equirements to develop a prototype

Hardware	**Description**
• Laptop with the operating system Windows 10 • 2.0 GHz AMD Ryzen 5 2500U Processor • 8GB RAM • 64-bit operating system	Used to run the operating system, write and edit code, host database, and write documents.
• Full HD resolution, ideally 1920x 1080	Provides clear and vivid graphics

Moreover, Table 4.2 illustrates the multiple software requirements which are used to implement this prototype.

Table 4. 2 Software requirements to develop the prototype

Software	**Description**
Microsoft Excel	Microsoft Excel can serve as a valuable tool for saving and organizing data that is destined for use in machine learning projects. In the context of machine learning, where data preparation and preprocessing are critical steps, Excel provides a user-friendly interface for managing and structuring datasets.
Phyton 3.6 and above	Phyton is a programming language with a high abstraction level, is object-oriented, and uses dynamic emantics. It is compatible with multiple platforms such as Microsoft Windows, Mac, Linux, and so on. Multiple packages are needed to implement this software, for instance,

	Pandas, Numpy, Surprise, Scikit-learn and Matplotlib.
Jupyter Notebook	Jupyter Notebook is a web-based tool that enables users to create and share document with interactive code, equations, visualisations and narrative text. It is commonly ultilized for tasks related to data science and machine learning, as well as for educational purposes.

4.1.2 Hardware and Software Requirements to Run the Protoype

The hardware and software requirements to run the prototype are listed in Table 4.3.

Table 4. 3 Hardware and Software requirements to run the prototype

Operating System	Windows 10(64-bit)
Processor	2.0 GHz AMD Ryzen 5 2500U or above
Memory/ RAM	8GB or above

4.2 Implementation (Overall Process Flow)

Since the recommendation based on user-item, so this project proptotype will recommend new product based on item the users bought. The prototype will first ask to prompt User ID (1-32) only because the list of previous users has been compressed using autoencoder earlier. Next, the prototype will perform tasks in the background, such as data cleaning, preprocessing, model training, prediction and evaluation. In the end, the similarity scores along with the recommended product prediction will be displayed. The GUI will also be implemented to show the process of the simulation. Figure 4.2 visualizes the flowchart of the prototype.

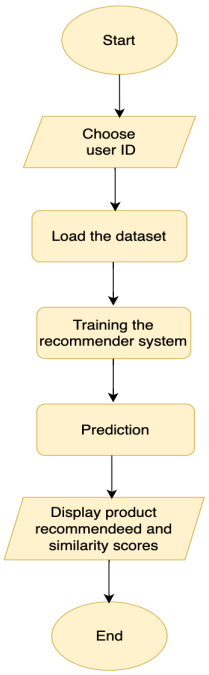


Figure 4. 2 Flowchart of the protoype

4.2.1 Dataset

The dataset for model training in this prototype is Amazon Consumer Review that has been obtained from 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 23 columns features and only five columns being used for further exploration which is listed in Table 4.4.

Table 4. 4 Columns of Amazon Consumer Review dataset

Column Name	**Description**
id	Unique ID for every product purchase
name	Name of the 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 give by users in product review
Product_image	Image of the products
reviews.username	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
manufacturer	The company or entity responsible for manufacturing the product.
reviews.date	The date when a review was posted.
reviews.dateAdded	The date when the review was added to the system or dataset.
reviews.dataSeen	The date when the review data was last seen or accessed.
reviews.didPurchase	Indicates whether the reviewer claims to have purchased the product.
reviews.doRecommend	Indicates whether the reviewer recommends the product.
reviews.id	A unique identifier for each review.
reviews.numHelpful	The number of users who found the review helpful.
reviews.rating	The specific rating given by the reviewer in the review.
reviews.sourceURLs	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.userCity	The city or location associated with the reviewer.

4.2.2 Recommender System

I. Autoencoder (AE)

After the preprocessing part are done, cleaned dataset are load using surprise library which import 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 compile it. The optimizer used is Adam optimizer and loss function is mean_squared_logarithmic_error. The data then splitted into training and testing data where it appears to be initializing matrices to represent training and test data in a recommendation system. The initialization with zeros indicates that, initially, 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, autoencoder model using the specified training data which is the 'train_data' so that the model 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. Figure 4.3 shows the training process is repeated for 30 epochs, with each epoch consisting of batches of 52 samples.

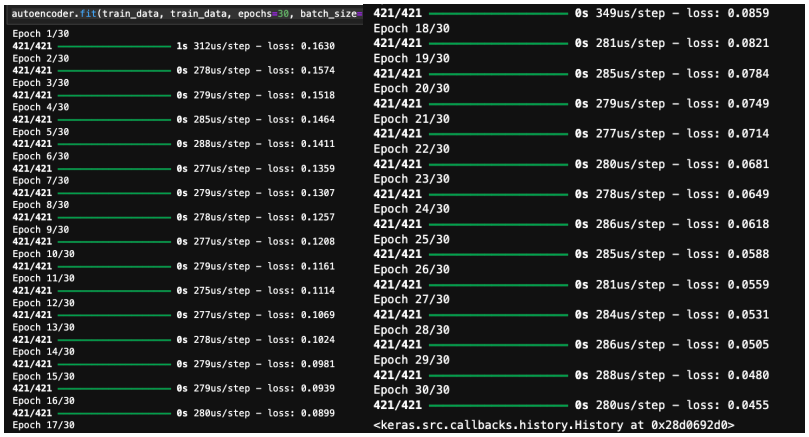


Figure 4. 3 Training process repeated for 30 epochs consisting of 52 batches samples

The user's attributes or characteristics are captured in the subsequent function by the vector that represents the user in the learnt latent space, which is derived from the autoencoder model. The representation of an item in the learned latent space is represented 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 an avector of similarity scores, where each score denotes how similar the user is to a particular object. Lastly, they are being sorted based on similarity scores in descending order, in dication 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 learnt 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 they are recommended accordingly. Lastly, for a real visualisation, 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. Figure 4.4 illustrates the human-readable version of this recommendation where the prototype will show previous item bought by user id prompted and recommend new products.

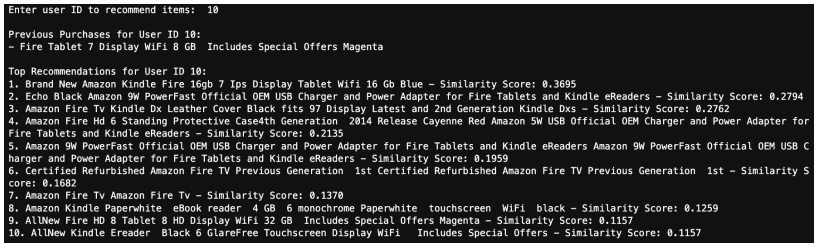


Figure 4. 4 Autoencoder product recommendation

II. Long short-term memory (LSTM)

The first step involves preprocessing the data to make it suitable for the LSTM model. This is done by encoding the usernames and product names using LabelEncoder. This converts the categorical user and product identifiers into

numerical form. The processed data is then loaded into the Surprise library, which helps in creating a structured dataset for recommendation tasks. The interaction matrix, representing user-item interactions, is constructed, where each cell indicates the rating a user has given to an item.

Then, The interaction matrix is further expanded to include an additional dimension, making it compatible with the LSTM model's requirements. This matrix now has three dimensions: users, a single time step, and items. The data is split into training and test sets to enable the evaluation of the model's performance. This splitting ensures that the model can be trained on one portion of the data and tested on another, simulating its performance on unseen data.

Next, the LSTM model is defined using the Keras library. A Sequential model is set up, and an LSTM layer is added with 100 units and a ReLU activation function. This layer processes the input sequences and captures the temporal dependencies in user interactions. Following the LSTM layer, a Dense layer is added to produce the output with the same number of features as the input, ensuring that the model can predict the user's preferences for all items. The model is compiled using the Adam optimizer and Mean Squared Error (MSE) as the loss function.

Furthermore, the model is then trained on the training data. The input (trainX) is used as both the input and target for the model, enabling it to learn to predict the next item in the sequence based on the given input. The training process involves multiple epochs, where the model iteratively adjusts its weights to minimize the loss. Batch

size determines the number of samples processed before the model's internal parameters are updated. After training, the model can predict user preferences for the items in the dataset.

Finally, the trained model is used to generate recommendations. A function is defined to get the top N recommendations for a specific user. This function retrieves the predicted preferences for the user, sorts them to identify the top items, and returns these as recommendations. The recommended product IDs are then decoded back into their original names using LabelEncoder. Additionally, previous purchases of the user are retrieved and displayed alongside the new recommendations, providing a comprehensive view of the user's interaction history and suggested items. Figure 4.5 illustrates the recommendation of LSTM model.

Figure 4. 5 LSTM product recommendation

4.3 Evaluation Metrics

Evaluation metrics are essential for evaluating and optimising model performance in machine learning, particularly in recommendation systems. Assessment metrics offer numerical assessments of a recommendation system's performance concerning user satisfaction, relevancy, and accuracy. They give a systematic mechanism to compare different algorithms, guide model optimization, and inform decision-making. Recommendation systems can be improved to better match user tastes, improve the quality of recommendations, and ultimately provide a more fulfilling user experience by utilising metrics like Mean Absolute Error (MAE), precision, recall, and similarity scores. The capacity to objectively measure the system's efficacy, direct the creation of models, and guarantee that the recommended products satisfy users' requirements and expectations are the main advantages of utilising evaluation metrics in recommendation systems.

4.3.1 Mean Aboslute Error

A popular metric for calculating the average absolute difference between expected and actual values is mean absolute error, or MAE. It offers an easy-to-understand method for determining how accurate a predictive model is. The following Equation 4.1 is the formula for MAE:

$$\boldsymbol{MAE = n1\sum i = 1n \mid yi - y\hat{}i}$$

Equation 4. 1

Where :

$\boldsymbol{n}$ = number of observations

$\boldsymbol{yi}$ = observed value for observations $\boldsymbol{i}$

$\boldsymbol{y\hat{}i}$ = predicted value for observation $\boldsymbol{i}$

The mean absolute error (MAE) is computed by adding together all of the absolute differences between each observed and forecasted value, dividing the result by the total number of observations. Regardless of the direction of the errors, the mean absolute error (MAE) shows how much predictions, on average, differ from the actual data. Better model accuracy is indicated by a lower MAE.

4.3.2 Validation Loss Curves

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 optimisation 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 performance of the model, loss curves are a crucial tool. The training loss and validation loss 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 indicates how well the model generalises to new data. A diminishing training loss is expected as the model gets better at reducing the error on the training set.

The loss function, often denoted as $\boldsymbol{J(\theta)}$, is a mathematical expression that measures the discrepancy between the predicted values of a machine learning model and the true target values. The general form of loss function can be expressed as Equation 4.2 below:

$$\boldsymbol{J(\theta) = 2m1\sum i = 1m(h\theta(x(i)) - y(i))2}$$

Equation 4. 2

Where :

$\boldsymbol{m}$ = number of training examples

$\boldsymbol{x(i)}$ = input feature vector of the ith training example

$\boldsymbol{y(i)}$ = true target value of the ith training example

$\boldsymbol{h\theta(x(i))}$ = model's prediction for the ith training example based on parameters 0.

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.

4.3.3 Similarity Score

Similarity scores measure how similar or similar two entities—users or items—are in the context of machine learning, and 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 in terms of preferences or behaviour. It is

possible to employ a variety of similarity metrics, such as Jaccard similarity, Pearson correlation, and cosine similarity. The following shown in Equation 4.3 is the formula for the cosine similarity between two vectors, A and B:

$$\boldsymbol{Cosine\ Similarity}(\boldsymbol{A}, \boldsymbol{B}) = \boldsymbol{A} \cdot \boldsymbol{B} \,/\, \| \boldsymbol{A} \| \cdot \| \boldsymbol{B} \|$$

Equation 4. 3

Where :

$\boldsymbol{A} \cdot \boldsymbol{B}$ = the dot product of vectors A and B

$\| \boldsymbol{A} \| \ \boldsymbol{and} \ \| \boldsymbol{B} \|$ = Euclidean norms (lengths) of vectors A and B, respectively

This formula yields a similarity score between -1 and 1, with 1 indicating perfect similarity and -1 indicating perfect dissimilarity. 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.

4.3.4 Precision

Of all the items the system suggests, precision calculates the percentage of pertinent items (true positives) (true positives plus false positives). In essence, precision provides an answer to the following query: "How many of the items the system recommended are actually relevant?" The precision calculation formula is as Equation 4.4 below:

$$\boldsymbol{Precision} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ positives + False\ Positives}}$$

Equation 4. 4

A higher precision indicates that the recommendation system is efficient at providing relevant items to the user, which is crucial for maintaining user satisfaction and trust. For example, in an e-commerce recommendation system, high precision means that most of the suggested products are ones that the user is likely to find useful or interesting, reducing the clutter of irrelevant items.

4.3.5 Recall

By calculating the percentage of appropriate items (true positives) that have been successfully recommended out of all the relevant items available (true positives plus false negatives), recall is a crucial metric used to evaluate the efficacy of a recommendation system or classification model. Recall essentially provides an answer to the following query: "How many of the relevant products available did the system successfully recommend?" The recall calculation formula is as Equation 4.5 below:

$$\boldsymbol{Recall} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives + False\ Negatives}}$$

Equation 4. 5

A recommendation system with a better recall rate may retrieve a larger proportion of relevant items, which guarantees that users are exposed to the majority of items that are relevant to their requirements or interests. For example, strong recall in a movie recommendation system indicates that the system is able to recommend a

lot of movies that the user would probably like, even while it also includes some irrelevant ones. Recall is important to ensure that all relevant topics are covered, but if recall is the only factor considered, it could lead to a lot of irrelevant recommendations. For this reason, recall is frequently carefully balanced to give a more comprehensive evaluation of the system's effectiveness.

4.3.6 F1-Score

The F1-Score is a crucial metric used to evaluate the performance of a recommendation system or classification model by balancing the trade-off between precision and recall. It is the harmonic mean of precision and recall, providing a single measure that captures both the relevance and the comprehensiveness of the recommendations. The formula for calculating the F1-Score is as Equation 4.6 below:

$$\boldsymbol{F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}}$$

Equation 4. 6

An improved precision and recall balance is indicated by a higher F1-Score, indicating that the system is more accurate and comprehensive in its suggestions. at the context of spam email filters, for instance, a high F1-Score indicates that the filter is efficient at identifying the majority of spam emails (high recall) and reducing the quantity of valid emails that are mistakenly classified as spam (high precision). The F1-Score offers a more thorough evaluation of the model's overall performance and is especially helpful in situations when you need to make sure that a healthy balance

is maintained between accurately recognizing important items and limiting the detection of irrelevant ones.

4.3.7 Normalized Discounted Cumulative Gain (NDCG)

Normalized Discounted Cumulative Gain (NDCG) is a widely used metric for evaluating the quality of rankings in recommendation systems and search engines. NDCG measures the relevance of recommended items, considering the position of each item in the recommendation list. Higher-ranked items contribute more to the NDCG score, making it sensitive to the order of recommendations. The formula for NDCG involves two main steps: calculating the Discounted Cumulative Gain (DCG) and then normalizing it as show in Equation 4.7, 4.8 and 4.9 below:

1. Discounted Cumulative Gain (DCG)

$$\boldsymbol{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel} i - 1}{log_2(i+1)}$$

Equation 4. 7

where:

$\boldsymbol{rel_i}$ = relevance score of the item at position *i*
$\boldsymbol{p}$ = up to which the DCG in computed

2. Ideal Discounted Cumulative Gain (IDCG)

$$\boldsymbol{IDCG_p} = \sum_{i=1}^{p} \frac{\boldsymbol{2^{rel*} i - 1}}{\boldsymbol{log_2(i+1)}}$$

Equation 4. 8

where:

$\boldsymbol{rel_i^*}$ = relevance score of the item at position i in the ideal (perfectly ordered) ranking

3. Normalized DCG (NDCG):

$$\boldsymbol{NDCG_P = \frac{DCG_p}{IDCG_p}}$$

Equation 4. 9

NDCG ranges from 0 to 1, where 1 indicates a perfect ranking. A higher NDCG score signifies that the recommendation system not only identifies relevant items but also ranks them in an order that places the most relevant items at the top. For instance, in a music recommendation system, a high NDCG score means that the most preferred songs by the user are ranked higher in the recommendation list. This metric is particularly valuable in scenarios where the position of the recommended items significantly impacts user satisfaction.

4.3.8 Root Mean Squared Error

Root Mean Squared Error (RMSE) is a commonly used metric for evaluating the accuracy of predictions in recommendation systems and regression models. RMSE measures the square root of the average squared differences between the predicted values and the actual values. It provides a direct indication of the magnitude of the prediction errors, with lower values indicating better model performance. The formula for calculating RMSE is as Equation 4.10 below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y\ _i)\text{^}2}$$

Equation 4. 10

where:

$\boldsymbol{y_i}$ = represents the actual value,

$\hat{y}_i$ = represents the predicted value,

n = number of observations

When analyzing how well a recommendation system forecasts user ratings or preferences, RMSE is especially helpful. For example, RMSE quantifies the degree to which consumers' actual ratings and the anticipated ratings of movies in a movie recommendation system agree. A smaller root mean square error (RMSE) suggests that the recommendation system is effectively capturing user preferences because the projected and actual ratings are relatively similar. But because bigger errors have a correspondingly bigger effect on the score, RMSE is prone to outliers. Because of this, RMSE is an important indicator for making sure the recommendation system continuously generates correct predictions and minimizes huge errors that could have a negative impact on user happiness.

4.4 Summary of Chapter

In conclude this chapter, this chapter brief the details of research methodology, building on the theoretical concepts of Autoencoder and LSTM. It begins with comprehensive background research on recommender systems, focusing on "Generative AI in recommender systems," "E-commerce recommendations," and "Gen-AI in optimizing recommendation products," using reputable academic and reliable sources. A thorough literature review helps structure the research method, leading to steps such as data collection, exploratory data analysis, and implementation. The chapter specifies hardware and software requirements for developing and running the prototype, including laptops, Microsoft Excel, Python, and Jupyter Notebook. The implementation section outlines the prototype's process flow, from user-item recommendations based on past purchases to data cleaning, preprocessing, model training, prediction, and evaluation. The Amazon Consumer Review dataset from Kaggle is used for model training. The recommender system uses Autoencoder and Long Short-Term Memory (LSTM) models, detailing their preprocessing steps, model definitions, and recommendation processes. Evaluation metrics such as Mean Absolute Error, Validation Loss Curves, Similarity Scores, Precision, Recall, F1-Score, Normalized Discounted Cumulative Gain (NDCG), and Root Mean Squared Error (RMSE) are discussed to ensure accurate, relevant, and satisfying recommendations.

Chapter 5 : PROTOTYPE

5.1 Data Preparation

5.1.1 Data cleaning and Preprocessing

After loading the dataset in the prototype, the process of data cleaning and data preprocessing are carried out to remove the errors and inconsistencies that occur in the datset as well as improves its quality. Pandas.drop() method is use to remove unnecessary columns before continue further analysis. Columns selected were “name”, “asins”, “categories”, “reviews.rating” and “reviews.username”. After that, process of cleaning data subsequently resumed with remove the missing values(NaNs). This process is essential because it enables the models to extract more significant insights from the dataset, providing a strong basis for further analysis. By filling in these gaps, the dataset becomes more complete, which enables the models to produce more detailed representations from the data that is available. Figure 5.1 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
```

Figure 5. 1 Handling missing values with dropna() method

Furthermore, since column “name” in this dataframe is a bit messy and need to cleanup, unique() method has been used first to identify unique product name available in the dataset. Then, to standardize it, str.strip() and str.replace() also being used to remove leading and trailling 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” Figure 5.2 shows how the categorical columns were converted into numerical format.

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['username_encoded'] = label_encoder.fit_transform(data['reviews.username'])
data['product_name_encoded'] = label_encoder.fit_transform(data['cleaned_name'])
```

```
[117]:
print(data['username_encoded'])
```

```
0          453
1        21447
2         3711
3        13197
4        17532
         ...
27895     6845
27896    20740
27897    14743
27898     8625
27899    14576
Name: username_encoded, Length: 27864, dtype: int64
```

```
[118]:
data['product_name_encoded']
```

```
[118]:
0         0
1         0
2         0
3         0
4         0
         ..
27895    47
27896    47
27897    47
27898    47
27899    46
Name: product_name_encoded, Length: 27864, dtype: int64
```

Figure 5. 2 Covert categorical data into numerical formats

5.2 User Interface

To enhance the Graphical User Interface (GUI) of this prototype, we will utilize Streamlit. The prototype involves unpickling and loading a model in a Python file. Figure 5.3 illustrates the location of the Python file that will be used for deployment in Streamlit.

Figure 5. 3 Python file for streamlit

5.2.1 Interface for Product Recommendation System

For the recommendation process, users need to start by selecting their user ID. For instance, if a user selects user ID = 30, the system will display the previous purchases made by this user. Following that, the system will present a list of recommended products, complete with product images and names, providing a seamless and visually appealing experience for the user. Figure 5.4 will firstly show user ID was prompted and the previous purchase made by him.

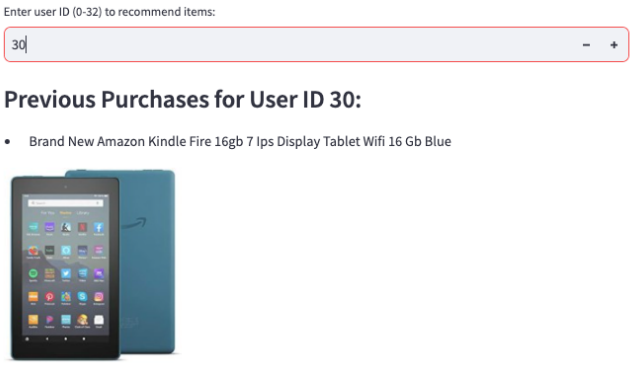


Figure 5. 4 Previous purchase by user ID = 30

Then, figure 5.5 will show the 10 products that highly recommended tailored to his previous purchase.

Highly Recommended Product:

1. AllNew Fire HD 8 Tablet 8 HD Display WiFi 16 GB Includes Special Offers Magenta

2. Amazon Fire Kids Edition Tablet 7 Display WiFi 16 GB Blue KidProof Case Blue

3. Echo Black Echo Black

4. Amazon Echo and Fire TV Power Adapter Amazon Echo and Fire TV Power Adapter

5. Amazon Kindle Fire Hd 3rd Generation 8gb Amazon Kindle Fire Hd 3rd Generation 8gb

6. Fire HD 8 Tablet with Alexa 8 HD Display 32 GB Tangerine with Special Offers

7. Certified Refurbished Amazon Fire TV Previous Generation 1st Certified Refurbished Amazon Fire TV Previous Generation 1st

8. Brand New Amazon Kindle Fire 16gb 7 Ips Display Tablet Wifi 16 Gb Blue

9. Certified Refurbished Amazon Fire TV Stick Previous Generation 1st Kindle Paperwhite

10. Fire Tablet 7 Display WiFi 8 GB Includes Special Offers Magenta

Figure 5. 5 Top 10 highly recommended products

5.3 Pseudocode

5.3.1 Pseudocode for Autoencoder in recommendation system

Algorithm 1: Autoencoder Model Training

```
{
  {
    Encode 'reviews.username' to 'username_encoded' using LabelEncoder
    Encode 'product_name' to 'product_name_encoded' using LabelEncoder

    Create a Reader object with rating scale (1, 5)
    Load data into Surprise Dataset

    Define num_users and num_items from the dataset
    Set latent_dim for encoding

    Define input layer with shape (input_shape)
    Define encoded layer with latent_dim neurons
    Define decoded layer with input_shape dimension

    Create and compile Autoencoder model with Adam optimizer and
mean_squared_logarithmic_error loss

    Split Surprise data into training and testing sets
    Prepare training and testing data arrays

    Train Autoencoder model on training data with specified epochs and batch size
    }

   {
   Define recommend_items_for_user function
    {
       Validate user ID

       Calculate similarity scores between user and items
       Exclude already interacted items
       Get top N recommendations based on similarity scores
    }

   Retrieve user and item representations from the model
   Encode users and items using LabelEncoder

   Prompt user for user ID and validate input
```

```
  Call recommend_items_for_user function to get recommendations

  Display top recommendations with product names and similarity scores
 }
}
```

5.3.2 Pseudocode for LSTM in recommendation system

Algorithm 2: LSTM Model Training

```
{
 {
  Encode 'reviews.username' to 'username_encoded' using LabelEncoder
  Encode 'product_name' to 'product_name_encoded' using LabelEncoder

  Create a Reader object with rating scale (1, 5)
  Load data into Surprise Dataset

  Build training set and create interaction matrix
  Initialize interaction matrix with num_users and num_items

  Fill interaction matrix with user-item ratings

  Expand interaction matrix dimensions
  Split data into training and test sets

  Define LSTM model with Sequential
  Add LSTM and Dense layers to the model
  Compile the model with Adam optimizer and MSE loss

  Train the model on training data with specified epochs and batch size

  Predict user preferences using the trained model
 }
 {
  Define get_top_n_recommendations function
  {
    Retrieve user ID from username using LabelEncoder
    Get user's predicted preferences
    Sort preferences to get top N items
  }

  Encode product names using LabelEncoder
```

```
    Prompt user for username and get recommendations
    Decode recommended product IDs back to original product names

    Retrieve and display previous purchases for the user

    Display top recommendations with product names
  }
}
```

5.4 Challenges

This dataset consist of more than 20k rows that may lead to slow performance in the interface. Furthermore, Autoencoders can easily overfit, especially when the model is too complex relative to the amount of available training data. This leads to poor generalization to new, unseen data.

5.5 Summary of Chapter

Chapter 5 covers data preparation and the user interface. Data cleaning and preprocessing involve removing unnecessary columns, handling missing values with `dropna()`, and converting categorical data to numerical format. The user interface is enhanced using Streamlit, allowing users to select their ID to view previous purchases and receive product recommendations. Pseudocode for Autoencoder and LSTM models outlines steps for training, validating user IDs, and generating top product recommendations.

Chapter 6 : TESTING

6.1 Evaluation Metrics Score

Using a wide range of evaluation measures, the effectiveness of the generative models used in this recommendation system is thoroughly evaluated. These outcomes serve as important to determine how effectively the models function during the whole training procedure. Metrics like Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) were first used to evaluate the accuracy of traditional models. However, the usefulness of these conventional measurements has decreased with the advent of advanced generative models. Instead, further metrics including recall, F1-score, precision, and Normalised Discounted Cumulative Gain (NDCG) have been added to better represent the capabilities and performance of the generative models. These indicators offer a more thorough evaluation, ensuring that the performance of the recommendation system is not only precise but also extremely pertinent and efficient. This change in evaluation approach highlights the advantages of generative models over traditional ones and shows how they can greatly improve the overall performance of the recommendation system.

In this research paper, we introduce two sophisticated generative models: Autoencoder and Long Short-Term Memory (LSTM). Each model utilizes distinct techniques to enhance the recommendation process, significantly contributing to improved performance in recommendations.

6.1.1 Autoencoder (AE) model

The first model employs in this recommendation system is autoencoder which it firstly utilized to extract meaningful representations of items and users. To facilitate mapping between user and item IDs and their encoded forms, LabelEncoder from the sklearn.preprocessing module is employed. This involves transforming the usernames and product names in the original dataframe into numerical labels. For user input, the system prompts the user to enter a user ID for whom recommendations are to be generated, incorporating error handling to ensure a valid ID is provided. Then, the recommend_items_for_user function is called, the user and item representations to calculate similarity scores and identify the top recommended items. The results are then displayed, converting the encoded item IDs back to their original form using the inverse transform method of LabelEncoder, and presenting the top recommendations along with their respective similarity scores. This approach showcases the autoencoder's ability to capture latent features in the data, making it a powerful tool for generating personalized and relevant product recommendations. Figure 6.1 below show the products recommendation for user ID = 2 and the similarity score obtained.

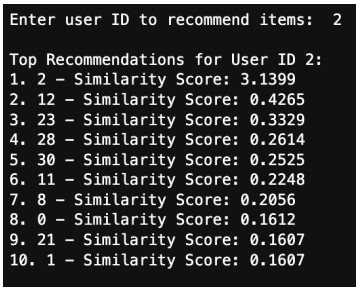


Figure 6. 1 Example of product recommendation with similarity score

For better representation, Once a valid user ID is provided by prompting desired user ID, the system retrieves and displays the items previously purchased by the user, converting the encoded user ID back to its original form using LabelEncoder. The recommend_items_for_user function is then called to generate similarity scores and identify the top recommended items based on the user and item representations derived from the autoencoder model. After generating the recommendations, the encoded item IDs are converted back to their original product names using LabelEncoder and the dataframe, ensuring that the recommendations are presented in a user-friendly manner. The code then displays both the user's previous purchases and the top recommended items, complete with similarity scores, providing a comprehensive and readable output for the user. This approach highlights the autoencoder model's ability to generate personalized recommendations by leveraging latent features in the data and ensures the output is easily understandable by converting encoded information back to its original form. Figure 6.2 displays the items previously purchased by the user, the products recommendation converting the encoded user ID back to its original form along with the similarity score.

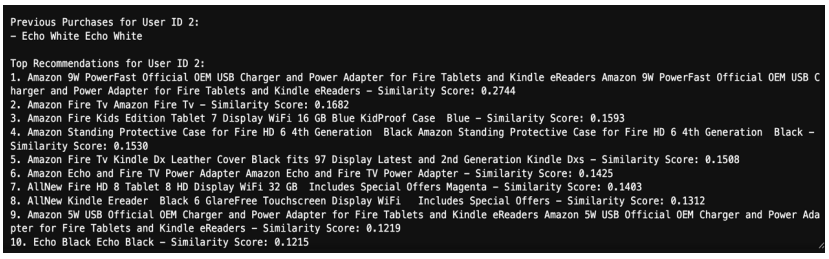


Figure 6. 2 Products recommendation and previous purchase encoded

The performance of first recommender system was evaluated using MAE, RMSE, Validation loss, Precision, Recall, F1-Score and NDCG which are most prefererable metrics to evaluate recommendation that consist generative models. Table 6.1 below shows the result obtained from first model meanwhile Figure 6.3 visualizes the validation loss over epochs while train the model.

Table 6. 1 Evaluation Metrics for AE model

MAE	**RMSE**	**Precision**	**Recall**	**F1-Score**	**NDCG**
0.5824	0.7473	0.9760	1.0	0.9879	0.9966

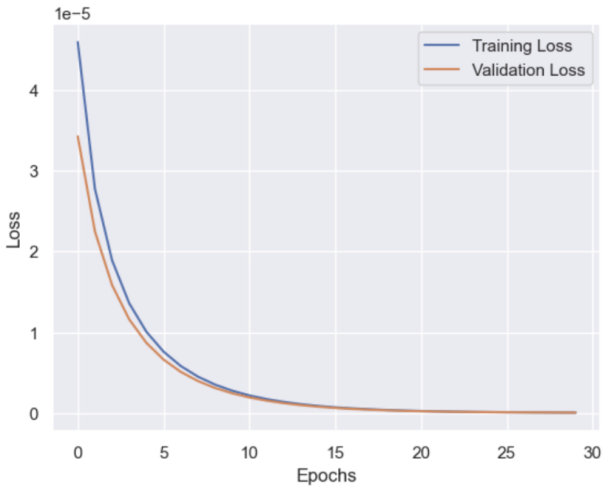


Figure 6. 3 Validation Loss over epochs

6.1.2 Long short-term memory (LSTM) model

Second model of generative model which is LSTM model also predict user preferences based on past interactions. LSTM networks are capable of learning long-term dependencies, making them suitable for sequential data. In a recommendation system, user interaction history is sequential data. By training on sequences of user interactions, the LSTM model learns patterns and trends in user behavior. The trained LSTM model can generate a vector of predicted preferences for items, which can be used to recommend items the user is likely to interact with next. By using an LSTM model, the recommendation system can effectively capture temporal dynamics in user behavior, providing more accurate and personalized recommendations.

The function `get_top_n_recommendations` in an LSTM-based recommendation system makes use of the model's capacity to identify temporal trends in user interaction data to provide customized recommendations. Using a `LabelEncoder}, the username is first transformed into a user ID to make sure it matches the encoded format used in the model. After been trained on user interaction sequences in the past, the LSTM model predicts a preference vector for the user that represents their likely interest in different items. After sorting the preferences in decreasing order and choosing the top N indices, this preference vector is processed to determine the top N things that the user is most likely to interact with. To make the recommendations more comprehensible, these indices are transferred back to the item names using an additional {LabelEncoder}. To add context to the recommendations, the code also pulls the user's previous purchases from the dataset. A tailored and relevant list of recommended items is produced by combining prior

purchase data with forecasted preferences. Figure 6.4 demonstrates how well-suited LSTM models are for producing precise suggestions based on the temporal patterns of user interactions because of their prowess in handling sequential data.

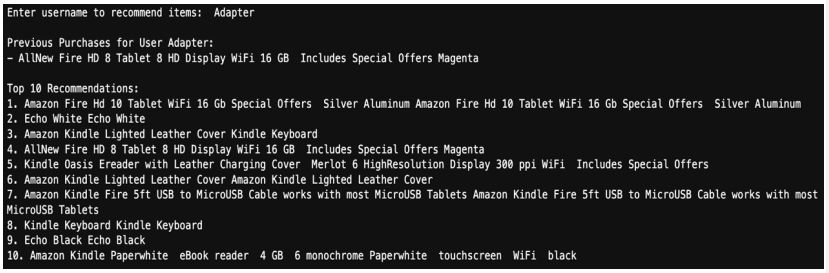


Figure 6. 4 Product recommendation using LSTM model

For an LSTM model, the approach differs from that of an autoencoder. An autoencoder can directly map input data to a compressed representation, which can then be encoded into labels, allowing users to prompt by user ID. In contrast, an LSTM model processes sequential data and is designed to predict the next item in a sequence, making it more suitable for tasks that involve time series or ordered data. Therefore, for an LSTM model, instead of prompting by user ID, it prompts by username. For instance, in the example above, the system retrieves the username to provide recommendations, ensuring that the LSTM model leverages the sequential patterns in the user's interaction history to make accurate predictions. This is why the function accepts a username input and then converts it to a user ID internally, allowing the model to generate personalized recommendations based on the user's past interactions. For example, if user ID 2 corresponds to the username 'Adapter', the system will use 'Adapter' to generate recommendations for him.

For the evaluation of the recommendation performance of LSTM model, same as first model, it will be assessed using MAE, RMSE, Precision, Recall, F1-Scorem NDCG and Validation loss over epochs while training the model. Table 6.2 show the result score and figure 6.5 will illustrates the validation loss over epochs.

Table 6. 2 Result score of LSTM models

MAE	RMSE	Precision	Recall	F1-Score	NDCG
0.1273	0.4644	0.1090	0.9596	0.1916	0.3455

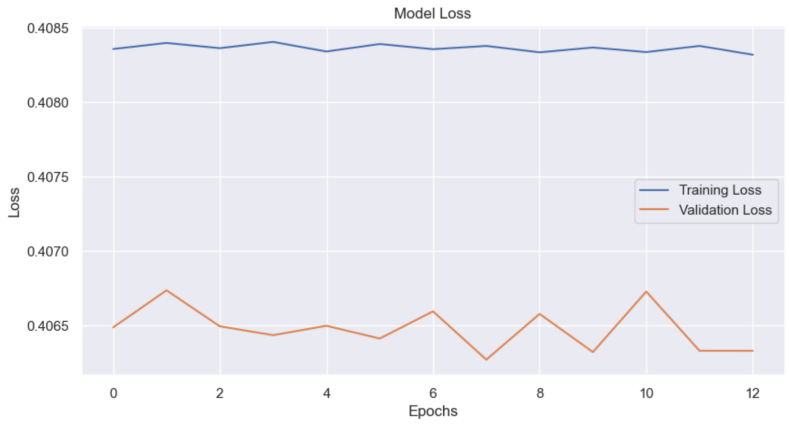


Figure 6. 5 Validation loss over epochs

6.2 Comparative Analysis

A thorough comparison of each model's evaluation measure scores can be found in this part. The analysis that follows helps determine which model will work best in the prototype. A bar chart comparing the effectiveness of three different recommendation system models is shown in Figure 6.6. These models comprise two generative models, the Autoencoder and the LSTM model, as well as the conventional collaborative filtering technique that makes use of Singular Value Decomposition (SVD). The comparative analysis centers on diverse evaluation metrics, facilitating an exhaustive review of every model's efficacy. This comprehensive analysis is essential to identify the model that provides the optimal ratio of precision, effectiveness, and expandability, consequently directing the choice process for the prototype's deployment.

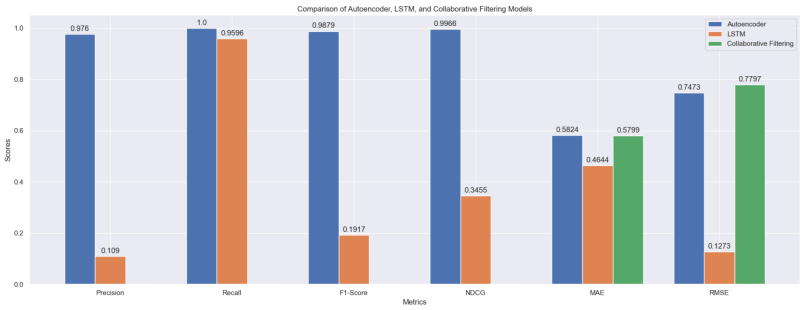


Figure 6. 6 Comparison of evaluation metrics for models.

Since collaborative filtering is a traditional method, it does not incorporate sophisticated evaluation metrics like precision, recall, F1-Score, and NDCG into its standard evaluation of performance. This is because these metrics are typically associated with generative models that are being used in this paper. Metrics like mean absolute error (MAE) or root mean square error (RMSE) are the main tools used in traditional collaborative filtering to testing the accuracy. These measures quantify the discrepancy between expected and actual scores, but they don't offer an accurate representation of how well the model performed in terms of relevance or ranking quality. In contrast, these sophisticated evaluation metrics can be used by generative models, such as Autoencoders and LSTM models, to give a more thorough and detailed picture of their performance. Table 6.3 depicts the clear comparison between these three models.

Table 6. 3 Comparison among three models

Model	**Collaborative Filtering (SVD)**	**Autoencoder (AE)**	**Long short-term memory (LSTM)**
MAE	0.5799	0.5824	0.4644
RMSE	0.7797	0.7473	0.1273
Precision	-	0.976	0.109
Recall	-	1.0	0.9596
F1-Score	-	0.9879	0.1917
NDCG	-	0.9906	0.3455

For MAE and RMSE, LSTM achieves the lowest MAE (0.4644) and RMSE (0.1273), indicating it has the best accuracy among the three models in terms of error metrics. This suggests that LSTM is highly effective in minimizing prediction errors. Also, it performs well in Recall (0.9596), but it falls short in Precision (0.109), F1-Score (0.1917), and NDCG (0.3455), indicating that while it is good at capturing relevant items, it struggles with ranking and balancing precision and recall. However, Autoencoder (AE) excels in advanced evaluation metrics. With a Precision of 0.976, Recall of 1.0, F1-Score of 0.9879, and NDCG of 0.9906, it demonstrates superior performance in identifying relevant items and ranking them appropriately.

In conclusion, The Autoencoder (AE) appears as the best generative model when these models are compared in the context of recommendation systems. It greatly outperforms in all advanced assessment measures and equals the error metrics with the LSTM. The Autoencoder is quite good at finding relevant recommendations and correctly rating them, as evidenced by the high Precision, flawless Recall, strong F1-Score, and NDCG scores. Because of this, the Autoencoder is an extremely dependable and effective option for recommendation systems, particularly when the objective is to optimize the relevance and accuracy of the suggestions made to users. Table 6.4 shows the advantages and disadvantages of these three different models.

Table 6. 4 Advantages and disadvantages of three different models

Model	**Advantages**	**Disadvantages**
SVD	- Simple and interpretable. - Well-understood and widely used in traditional systems. - Effective for smaller datasets with explicit ratings.	- Lacks advanced metrics like Precision, Recall, F1-Score, and NDCG. - May not capture complex patterns in user behavior. - Can struggle with sparsity and scalability issues.
Autoencoder (AE)	- High Precision (0.976), Recall (1.0), F1-Score (0.9879), and NDCG (0.9906). - Excels in capturing complex user behavior and interactions. - Generates highly relevant and well-ranked	- Slightly higher MAE (0.5824) compared to LSTM. - More computationally intensive compared to SVD. - Requires a larger amount of data for effective training.

	recommendations.	
Long Short-Term Memory (LSTM)	- Lowest MAE (0.4644) and RMSE (0.1273), indicating high accuracy. - Effective in capturing temporal patterns in user behavior. - Good Recall (0.9596) shows ability to identify relevant items.	- Low Precision (0.109) and F1-Score (0.1917) highlight issues with relevance ranking. - Higher complexity and computational requirements. - Less effective in ranking compared to AE, as shown by lower NDCG (0.3455).

Chapter 7 : CONCLUSION AND FUTURE WORK

7.1 Conclusion

In conclusion, significant progress has been made in the implementation of the recommendation system. The core information for the succeeding design and implementation stages has been established through the completion of the groundwork for the study on the recommendation system's background and a thorough review of relevant works. The investigation of recommendation techniques has brought the spotlight on the advantages and disadvantages of three well-known approaches: content-based (CB), collaborative (CF), and hybrid (HF). Apart from these well-known methods, a new method that incorporates generative AI—more precisely, the Autoencoder (AE) technique and Long Short-term memory (LSLTM) has been looked at in detail. AE has been chosed for further implementation in recommendation system because of the evaluation achieved. Metrics like MAE, Loss Curves, Precicion, Recall, F1-Score, RMSE, NDCG and Similarity Score are used to measure how well the models perform within the dataset absed on existing user. Lastly, the user-friendly recommendation system is demonstrated by developing user interface that has been deployed in streamlit application.

7.2 Future Works

In future, the Autoencoder (AE) and Long Short-Term Memory (LSTM) models will be optimized and fine-tuned. Other generative AI techniques, like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), will be investigated for improved recommendation accuracy and diversity. Finally, real-time data streams will be integrated to guarantee the most recent recommendations. User engagement will be increased by creating a more complex and user-friendly user interface with elements like interactive visualizations, user feedback systems, and dynamic information updates. The system will undergo extensive performance and scalability testing to make sure it can manage high data volumes and multiple user interactions at once. Recommendations based on user preferences and behaviors will continually be improved by adding user feedback into the algorithms. The system will be further advanced by conducting longitudinal user studies, investigating cross-domain recommendations, fortifying security and privacy safeguards, and working with academic institutions and industrial partners. Lastly, to guarantee ongoing and dependable functioning, preparation for deployment in a production setting and the establishment of strong maintenance processes, including regular updates, monitoring, and troubleshooting methods, are essential.

References

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

Barragáns-Martínez, A. B., Costa-Montenegro, E., Burguillo, J. C., Rey-López, M., Mikic-Fonte, F. A., & Peleteiro, A. (2010). A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, *180*(22). https://doi.org/10.1016/j.ins.2010.07.024

Fayyaz, Z., Ebrahimian, M., Nawara, D., Ibrahim, A., & Kashef, R. (2020). Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *Applied Sciences (Switzerland)*, *10*(21). https://doi.org/10.3390/app10217748

Ferreira, D., Silva, S., Abelha, A., & Machado, J. (2020). Recommendation system using autoencoders. *Applied Sciences (Switzerland)*, *10*(16). https://doi.org/10.3390/app10165510

Grewal, D., Hulland, J., Kopalle, P. K., & Karahanna, E. (2020). The future of technology and marketing: a multidisciplinary perspective. In *Journal of the Academy of Marketing Science* (Vol. 48, Issue 1). https://doi.org/10.1007/s11747-019-00711-4

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. In *Egyptian Informatics Journal* (Vol. 16, Issue 3). https://doi.org/10.1016/j.eij.2015.06.005

Jiang, L., Cheng, Y., Yang, L., Li, J., Yan, H., & Wang, X. (2019). A trust-based collaborative filtering algorithm for E-commerce recommendation system. *Journal of Ambient Intelligence and Humanized Computing*, *10*(8). https://doi.org/10.1007/s12652-018-0928-7

Lacic, E., Reiter-Haas, M., Kowald, D., Reddy Dareddy, M., Cho, J., & Lex, E. (2020). Using autoencoders for session-based job recommendations. *User Modeling and User-Adapted Interaction*, *30*(4). https://doi.org/10.1007/s11257-020-09269-1

Loukili, M., Messaoudi, F., & Ghazi, M. El. (2023). Machine learning based recommender system for e-commerce. *IAES International Journal of Artificial Intelligence*, *12*(4). https://doi.org/10.11591/ijai.v12.i4.pp1803-1811

Namazi, M., Karimi-Jafari, M. H., Qassemi, F., & Ghasemi, J. B. (2022). Autoencoders in generative modeling, feature extraction, regression, and classification. In *Machine Learning and Pattern Recognition Methods in Chemistry from Multivariate and Data Driven Modeling*. https://doi.org/10.1016/B978-0-323-90408-7.00007-1

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

Noorian, A., Harounabadi, A., & Hazratifard, M. (2024). A sequential neural recommendation system exploiting BERT and LSTM on social media posts. *Complex and Intelligent Systems*, *10*(1). https://doi.org/10.1007/s40747-023-01191-4

Rama, K., Kumar, P., & Bhasker, B. (2021). Deep autoencoders for feature learning with embeddings for recommendations: a novel recommender system solution. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-021-06065-9

Reddy, B. R., & Kumar, R. L. (2023). An E-Commerce Based Personalized Health Product Recommendation System Using CNN-Bi-LSTM Model. *International Journal of Intelligent Engineering and Systems*, *16*(6), 398–410. https://doi.org/10.22266/ijies2023.1231.33

Rybakov, O., Mohan, V., Misra, A., LeGrand, S., Joseph, R., Chung, K., Singh, S., You, Q., Nalisnick, E., Dirac, L., & Luo, R. (2018). The effectiveness of a two-layer neural network for recommendations. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*.

Sachdeva, N., Ritacco, E., Manco, G., & Pudi, V. (2019). Sequential variational autoencoders for collaborative filtering. *WSDM 2019 - Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. https://doi.org/10.1145/3289600.3291007

Tarus, J. K., Niu, Z., & Mustafa, G. (2018). Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial Intelligence Review*, *50*(1). https://doi.org/10.1007/s10462-017-9539-5

Tran, D. H., Hussain, Z., Zhang, W. E., Khoa, N. L. D., Tran, N. H., & Sheng, Q. Z. (2018). Deep autoencoder for recommender systems: Parameter influence analysis. *ACIS 2018 - 29th Australasian Conference on Information Systems*. https://doi.org/10.5130/acis2018.aj

Wei, S., Zheng, X., Chen, D., & Chen, C. (2016). A hybrid approach for movie recommendation via tags and ratings. *Electronic Commerce Research and Applications*, *18*. https://doi.org/10.1016/j.elerap.2016.01.003

Yu, M., Quan, T., Peng, Q., Yu, X., & Liu, L. (2022). A model-based collaborate filtering algorithm based on stacked AutoEncoder. *Neural Computing and Applications*, *34*(4). https://doi.org/10.1007/s00521-021-05933-8

Zhang, G., Liu, Y., & Jin, X. (2020). A survey of autoencoder-based recommender systems. In *Frontiers of Computer Science* (Vol. 14, Issue 2). https://doi.org/10.1007/s11704-018-8052-6

Zhang, Q., Lu, J., & Jin, Y. (2021). Artificial intelligence in recommender systems. *Complex and Intelligent Systems*, *7*(1). https://doi.org/10.1007/s40747-020-00212-w

Zhao, L., Lu, Z., Pan, S. J., & Yang, Q. (2016). Matrix factorization+ for movie recommendation. *IJCAI International Joint Conference on Artificial Intelligence*, *2016-January*.

Appendix A : Research Paper

Leave this box blank

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, 63100 Cyberjaya, Malaysia.*

[b] *Department of Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang, Indonesia.*

E-mail: 1211303587@student.mmu.edu.my, sucheng@mmu.edu.my, kong.wan.er@student.mmu.edu.my, heru.agus.santoso@dsn.dinus.ac.id, gktong@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 endeavour 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 literature analysis of recommender system algorithms is part of the process, and 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 endeavours to widen up the opportunity for recommendations that are more precise, 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**

I. Introduction

The importance of recommendation systems has grown, especially since the emergence of websites like YouTube, Amazon, and Netflix. Recommendation systems assess customer preferences and make proactive suggestions for things that they are likely to purchase based on product information, customer behaviour, and other data. Numerous research investigations have been carried out to create these recommendation systems, and numerous useful systems have been effectively used in a range of industries [1]. Nowadays, the majority of both 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 customised product recommendations. One creative way to get beyond the constraints of ecommerce 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 completely change recommender systems. 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, Food Recommendation and E-commerce, where Generative AI has been effectively employed in recommender systems to enhance the precision and personalization of content suggestions. Customers find it very important in the e-commerce version because, for example, generative AI makes it possible to create highly personalised suggestions by evaluating each user's unique behaviour and interests. By showing customers products that closely match their interests, personalisation 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 closely match their tastes. This optimised 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, customised 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 overall.

Traditional RS can be broadly grouped into Content-based (CB), Collaborative-Filtering (CF), Hybrid-based (HB) and Knowledge-based (KB). CB solutions are widely utilised in a variety of industries, where they are arguably the most popular approach. Just a few of the 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. Taking into account that data can be utilised 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 will often look at the user's past interests and suggest related items or services.

By leveraging the interests of users who are similar, CF technologies go above and beyond. Giving accurate recommendations is made possible by using the preferences of individuals with similar tastes. Locating a group of individuals who the target user shares interests in is the basic tenet of collaborative filtering [2]. The algorithm forecasts the interests of the target user by taking these neighbours' preferences into account. Amazon is a well-known e-commerce company that employs collaborative filtering to suggest products to its customers. When the algorithm locates a neighbour user for the target user, recommendations based on the things these neighbours like can be produced. This neighbouring 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 utilising 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 the majority of users engaged with two separate items (a product, page, or email) in a comparable way, 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, one of the best ways to increase sales and enhance client retention is to combine many recommender approaches. Combining the best features of both approaches, Hybrid-based Recommender System (HBRS) 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 [3].

KB or Semantics-based recommender techniques make up the other group. Contextbased 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, in e-learning, this means that these systems use relational knowledge to map learner-relevant learning resources [4]

On the other hand, generative AI Intelligence (GAI) is powered by foundation models (large AI models), enabling them to handle a variety of tasks beyond conventional boundaries seamlessly, including summarization, Q&A, classification, and others. GAI can create customised 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 recommendations that are more contextually relevant. Generative AI algorithms are employed by e-commerce platforms like Amazon to provide tailored product recommendations that are predicated on consumers' past purchases and browsing patterns.

The objectives of this paper are to explore various of Generative AI in business process. Second, to design and implement the selected Generative AI. Lastly, to evaluate the algorithm through user testing or performance evaluation.

At the beginning of this project too, some problems were addressed and needed to be researched which are including what are the commonly used recommendation techniques people use nowadays. Second, what are the recommendation phases needed to be carried out to develop an e-commerce recommender system and finally what is the evaluation metrics that can be used to evaluate model's performance.

II. The Material 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 both explicit (such as ratings) and implicit (such as clicks) input. After that, enter the learning phase, during which the system makes use of this information to comprehend user preferences. To do this, profiles that reflect user preferences and the features that objects offer 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 yet seen. By ensuring that recommendations are customized to each user's individual preferences, this procedure enhances user satisfaction. The implementations underlying the recommendation phases are depicted in Fig. 1.

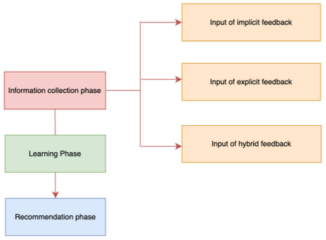


Fig. 1 Phases in RS

Information Gathering Phase requires a strong user profile or model. This first step collects relevant user data, including the user's attributes, behaviours, and the content of the resources they visit, in order to build a user profile or model for the prediction of taste. To successfully make recommendations that are appropriate for the user's preferences, the system requires as much information from the user as possible. Therefore, the model's capacity to represent users' current choices or preferences is a key factor in the effectiveness of a recommender system or recommendation. User's input data can be gathered in three ways, which include 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 easier and less demanding for the user [5]. Since explicit feedback solicits the user's direct input regarding product preferences, its efficacy depends on its accuracy. This kind of input requests ratings for multiple products from users via the system interface, which helps build and improve the recommendation model. The quantity and calibre of these user-provided ratings directly affect how accurate the recommendations are. 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 is still regarded as providing more reliable data even though it necessitates more work from users. Additionally, because it offers transparency into the recommendation process, it raises perceived recommendation quality and increases confidence in the recommendations [5].

Additionally, Users who wish to show their interest directly can only provide feedback through hybrid feedback, which combines explicit and implicit feedback rating. This approach empowers users by allowing them to provide feedback explicitly, emphasising 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 feedback 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 in this phase. This phase's trained data offers specific patterns that are used 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].

In the recommendation phase, the phase of recommendation suggests the kinds of products that a customer or user would find appealing. Recommendations may be given directly from the dataset gathered in the information gathering stage (which may be 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 revolutionise 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 shown an example of architecture generative AI in recommender system [6].

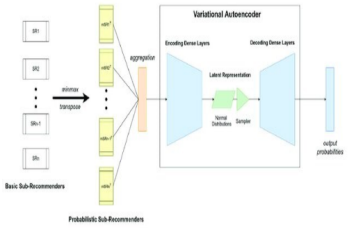


Fig. 2 Architecture of Generative AI in recommender system

In the Data Collection and Pre-processing Step, first, 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 for it to be effectively used in later stages. 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 needs to be cleaned, formatted, and converted in this initial step. To correspond with the intended output domain, a varied and representative dataset—such as high-quality images—must be gathered. Later, the data will undergo preprocessing to guarantee consistency, eliminate noise, and get ready for model training.

To get the desired results in the Model Architecture Selection Phase, choosing an appropriate model is crucial. 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 model that is selected and usually involves methods like regularization, gradient descent, and backpropagation to maximize model parameters. Reducing the difference between the output of the model and the real data, or ground truth, is the main goal of training. The model can generate material that closely resembles the patterns found in the training data thanks to this optimization process. As a result, the model can produce material that is similar to the training set.

This is accomplished by comparing it to predetermined criteria and benchmarks during the Evaluate and Refine process. You can evaluate the quality, coherence, and realism of the created output with the use of these measures. If the outcomes don't live up to 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 required level of satisfaction with the model's output.

In the Test and Validate phase, after being satisfied on how the generative AI model is performing, it becomes important 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 generalise to new data. Additionally, assess the model's capability to produce varied and coherent outputs in diverse scenarios. Lastly, compare the generated results against human judgement and domain-specific criteria to ensure the model's reliability and effectiveness.

In the Deploy and Integrate phase, the deployment and integration layer is where the generative AI process ends. The improved model is incorporated into useful 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 feedback and opinions from users, monitor model's performance and make changes in the meantime. Continuous improvement will always help to keep the model relevant and effective. Fig. 3 shows the end-to-end process of generative AI.

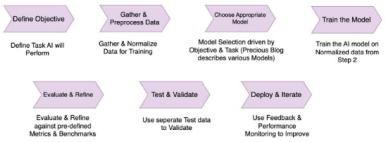


Fig. 3 End-to-end process of generative AI

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 key factor. For instance, it is assumed that a user prefers a certain author or brand if they have interacted with that author's book or brand's product in the past. Additionally, there's a chance they'll purchase a comparable item later.

The three components of CB RS's architecture are a content analyser, a profile learner (generator), and a filtering component. Fig. 4 illustrates this architecture. Using feature extraction techniques, the Content Analyzer gathers item content from various information sources and derives item representations. The Profile Learner generates a user profile based on previous likes and dislikes by generalising 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 [7].

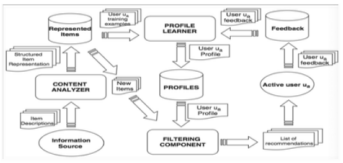


Fig. 4 Architecture of Content-based

Additionally shown is a content-based book recommendation engine that uses online book information. The system creates a user profile by analysing data taken from the web using a Naïve Bayes classifier. With the use of user-provided training examples, this profile plays a crucial role in producing a prioritised 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:* To find new user-item correlations, collaborative filtering examines user relationships and product interdependencies. User behaviour or user evaluations of recommended things serve as the foundation for CF recommendations. It investigates a variety of potential material and suggests items loved by users who are similar to you [8]. 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 are those that 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 personalised recommendations. User-based and item-based collaborative filtering are the two primary categories into which memory-based recommenders fall.

The user-based approach involves suggesting items to the target user by identifying others with similar behaviour or preferences. This entails finding users who closely resemble the target user based on their 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 scifi 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, machine learning models are used by model-based recommenders to produce recommendations. These systems carefully extract patterns, correlations, and linkages from past user-item interaction data. The model-based recommenders can forecast a user's preferences for products they haven't before interacted with by absorbing insights from this data. 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 of which has a unique advantage in producing precise and customized recommendations.

Despite this, a popular method for carrying out collaborative filtering is matrix factorization. The models are better than conventional nearest-neighbour 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 that are 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 there isn't any explicit feedback available. Indirect feedback is the expression of opinions through user behaviour, 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 personalised, differed, and effective recommendations is one of the best ways to increase sales and improve client retention. Especially in instances where recommendations are made in real life, they might generate more dependable, precise, and adaptable ideas. Content based and collaborative filtering are combined, and both filtering 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 utilising 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 films based on user ratings that are similar (content-based filtering) and by 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 [5]. The metalevel hybrid technique learns a second recommendation algorithm from a first algorithm by using the full model as input [9] which is shown in Fig. 5 below on how the HB filtering works.

Some of the good consequences that we can gain from using hybrid recommenders are that it is frequently robust in managing different suggestion scenarios. They can adjust to various user behaviours, data features, and recommendation difficulties. Such adaptability is useful 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.

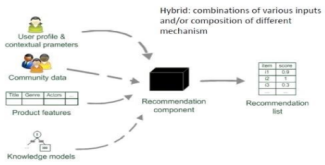


Fig. 5 HB filtering

4) *Generative AI:* Recently, recommender systems have benefited from the application of various AI techniques, which have improved user happiness and the overall user experience. AI makes recommendations that are of a higher calibre than those made with traditional techniques. Automation of intelligent behaviour is the aim of AI technology development. These six domains include knowledge engineering, reasoning, planning, communication, perception, and motion [10]. Hence, the next step will explain techniques that are applied to achieve the desired outcome. Fig. 6 explains the AI areas and the techniques used.

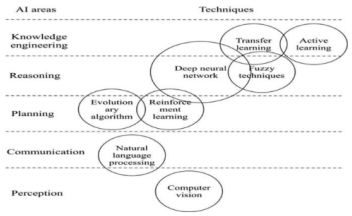


Fig. 6 The AI areas and the techniques used.

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 [11]. According to this definition, transfer learning is using 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 purpose in order to maximise machine learning performance with minimal information requirements. Users may be asked to name occurrences that are not labelled by an active learning system. Active learning has useful applications in different AI disciplines, especially well-suited for online systems, since labelling 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 be used to replicate real-world concepts that cannot be properly described, they are frequently used in the field of artificial intelligence (AI). Fuzzy approaches have drawn a lot of interest in the literature. For example, fuzzy distance has been used to describe similar instances, and fuzzy sets have been utilised by researchers to represent linguistic variables when feature values cannot be adequately expressed in numerical values [11].

Afterwards, reinforcement learning, and evolutionary algorithms participate in the planning domain. Evolutionary algorithms (EA) are a subclass of population-based search algorithms for global optimisation in artificial intelligence that are inspired by natural processes. A few 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 put an end to it.

The recommender system is seen as a learning agent in reinforcement learning, user behaviours 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. Finding a value function or method that enables consumers to maximise long-term advantages is the aim.

In order to address the problem of data sparsity in communication, the majority of recommender systems also enrich the rating matrix with review data that is acquired via natural language processing. In extreme cases, virtual ratings are generated utilising the emotion polarity that is obtained from review classification in the absence of 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 [12].

In summary, the direct application of computer vision in picture recommendation—that is, the mapping of 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 [10]. This was extended in later study by using low-level data, like colour qualities, that mimicked parts of the human visual system 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-ofinterest recommendations have also made use of visual content, taking use of the wealth of landmarks included in pictures and images uploaded by users.

C. Related Works

In the study by [13], 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 a number of 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 greatest choices and is renowned for its speed, low memory usage, and ability to handle big datasets had an impact on the optimizer selection—a strategy that is consistent with this article. The evolution of loss and validation loss (val_loss) show a declining trend over the course of each epoch, ultimately stabilising 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 achieved 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 dataset and 0.010 for the second dataset.

In the following study, [14] 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. Characterised by unique features and tuneable 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 it easier to identify the best performance parameters for a particular dataset. The ADAM optimizer, the SELU activation function, and one round of dense refeeding with mean normalisation enabled were among the common 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 DAE model's prediction accuracy is greatly impacted by the parameters selected. 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, [11] emphasised 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 important 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, [15] 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). The innermost layer activations of a deep autoencoder are used by DAFERec 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, thereby 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 tenfold cross-validation technique was employed by the authors to confirm the dependability of their findings.

Furthermore, based on [16], they have succeeded 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 really good results, with a high chance that customers will actually buy the next product suggested by the system. The focus of the study is 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 next suggested product. The evaluation metric serves as 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 are investing more money in social media, e-commerce, mobile, and internet channels to improve customer engagement and communication with both current and potential customers and increase sales. Thus, studies conducted by [17] explains in detail how predictive analytics is useful for estimating people's product preferences, price sensitivity, and expected next steps on the customer journey—all of which are critical for improving business activities. This is accomplished by utilising 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 recommendation system on Amazon discussed by [18] mentioned in this study. The use of neural networks in a personalised 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, the usage of offline assessment metrics—like Product Converted Coverage (PCC) at K and Precision at K—that are frequently employed in real-world recommender system applications is presented in this work. The usage of a deep learning package that facilitates multi-GPU model parallel training 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 utilisation 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 it increased accuracy metrics such as the research has demonstrated 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, [19] 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. Due to the fact that the ratings are typically broad, this method guarantees strong categorization performance. The authors utilise 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, the mapping relationship between the output ratings and the high-level representation is described by a softmax model. To assist in creating a highlevel, 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, [20] recommendation approach shown in this work encodes sessions inside the job domain using several autoencoder architectures. To recommend jobs within a session, the inferred latent session representations are used in a k-nearest neighbour fashion. Three autoencoder types are used in the study: variational autoencoder (VAE), denoising autoencoder (DAE), and classical autoencoder (AE). The input and output of the most basic AE are separated by a single hidden layer. 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, variational inference is used by VAE 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, [21] 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 through without taking temporal dependencies into account. In this configuration, the sequence passes through a number of fully connected layers at each RNN time-step. The probability distribution of the most likely future preferences is modelled 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 with respect to 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 a number of criteria. The evaluation concentrates on top-N recommendation while taking implicit preferences into account, 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 are designed for efficient coding, dimensionality reduction, and generative modelling are good candidates for unsupervised learning tasks, such as auto-encoders. It has been useful in learning underlying feature representation in a number of 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 [22]. The output layer is where the output data exits.

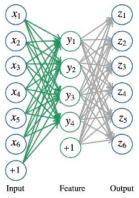


Fig. 7 Autoencoder architecture

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. More meaningful features can be learned by the auto-encoder 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 minimise the reconstruction error between x and x'.

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 [23] as shown in Fig. 8. The bottleneck will have a sampling layer rather than a straightforward dense layer. In order to create a Gaussian sample that will be used as input for the decoder, this layer will employ the mean and variance from the previous encoder layer. The first layer of the AE also uses dropout.

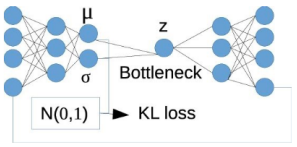


Figure 8 VAE Architecture

In sampling layer, latent vector Z by sampling from a Gaussian distribution with mean mu and standard deviation sigma will be generated. This is the reparameterization trick, which allows the network to be trained using gradient descent. Equation (4) show 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 denoted by *Loss* as per shown in Equation (5).

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

Next, dropout is a regularisation 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 that has been obtained from Kaggle Website provided by Datafiniti's Product Database [24]. The dataset is loaded into a panda DataFrame after importing the necessary libraries for developing the recommendation system. This dataset consisted of 22 columns features and only five columns being used for further exploration which is listed in Table 1.

Table 1 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.username	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
manufacturer	The company or entity responsible for manufacturing the product.
reviews.date	The date when a review was posted.
reviews.dateAdded	The date when the review was added to the system or dataset.
reviews.dataSeen	The date when the review data was last seen or accessed.
reviews.didPurchase	Indicates whether the reviewer claims to have purchased the product.
reviews.doRecommend	Indicates whether the reviewer recommends the product.
reviews.id	A unique identifier for each review.
reviews.numHelpful	The number of users who found the review helpful.
reviews.rating	The specific rating given by the reviewer in the review.
reviews.sourceURLs	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.userCity	The city or location associated with the reviewer.

C. Data Cleaning and Data Preprocessing

After loading the dataset in the prototype, the process of data cleaning and data preprocessing are carried out to remove the errors and inconsistencies that occur in the datset as well as improves its quality. Pandas.drop() method is used to remove unnecessary columns before continuing further analysis. Columns selected were "name", "asins", "categories", "reviews.rating" and "reviews.username".

After that, process of cleaning data subsequently resumed with remove the missing values (NaNs). This delicate process is essential because it enables the models to extract more significant insights from the dataset, providing a strong basis for further analysis. By filling in these gaps, the dataset becomes more complete, which enables the models to produce more detailed representations from the data that is available. 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 dataframe is a bit messy and need to cleanup, unique() method has been used first to identify unique product name available in the dataset. Then, to standardize it, str.strip() and str.replace() also being used to remove leading and trailling 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.

Fig. 10 Covert categorical data into numerical formats

D. E-commerce Products Recommender System

After the preprocessing part are done, cleaned dataset are load using surprise library which import 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 then splitted into training and testing data where it appears to be initializing matrices to represent training and test data in a recommendation system. The initialization with zeros indicates that, initially, 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, autoencoder model using the specified training data which is the 'train_data' so that the model 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 is repeated for 30 epochs, with each epoch consisting of batches of 52 samples.

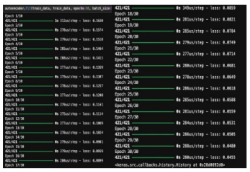


Fig. 11 Training process repeated for 30 epochs consisting of 52 batches samples.

The user's attributes or characteristics are captured in the subsequent function by the vector that represents the user in the learnt latent space, which is derived from the autoencoder model. The representation of an item in the learned latent space is represented 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 an avector of similarity scores, where each score denotes how similar the user is to a particular object. Lastly, they are being sorted based on similarity scores in descending order, in dication the objects that are most like the user. Recommendations are based on the top-N items with the highest similarity scores.

Presumably, each item in the learnt 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 they are recommended accordingly. Lastly, for a real visualisation, 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 item bought by user id prompted and recommend new products.

Fig. 12 Example of product recommendation

E. User Interface

Graphical User Interface (GUI) is also included in this prototype. The prototype can be started by opening the "Autoencoder in E-commerce RecSys" in jupyter notebook or another online or local applications that support python and its libraries, as shown in Fig. 13.

Fig. 13 IPYNB file of the prototype

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

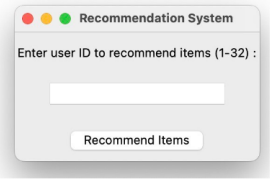


Fig. 14 GUI to prompt User ID

After prompted User ID, another window will be popped up to show the recommendation results where the window will firstly show the User ID and their previous item bought. Then, top recommendation for the user id will be displayed along with the similarity scores as shown in Fig. 15.

Fig. 15 Recommendation Results

Previous Purchase	Recommended Items	Similarity Score
Fire Tablet 7 Display Wi-Fi 8 GB includes special offer Magenta	Echo Black Amazon 9W PowerFast Official OEM USB Charger and Power Adapter for Fire Tablets and Kindle eReaders	0.4070
	Amazon Standing Protective Case for Fire HD 6 4th Generation Black Amazon	0.3151
	Amazon Fire Tv	0.3079
	Amazon Kindle Touch Leather Case 4th Generation 2011 Realise Olive Grean	0.2983
	Amazon Kindle Voyage 4GB WiFi 3G Black Fire HD 8 Tablet with Alexa	0.2012
	Certified Refurbished Amazon Fire Stick Previous Generation	0.2006
	Amazon Fire HD 8 in Tablet 18GB Black 6th Gen 2016 Android	0.1759
	Amazon 5W USB Official OEM Charger and Power Adapter for Fire Tablets and Kindle eReaders Amazon	0.1333
	Amazon Kindle Paperwhite eBook reader 4GB monochrome Paperwhite touchscreen WiFi Black	0.1220
	Amazon Kindle Lighted Leather Cover Kindle Keyboard	0.1144

F. Evaluation Results

A popular metric for calculating the average absolute difference between expected and actual values is mean absolute error (MAE). It offers an easy-to-understand method for determining how accurate a predictive model is.

The MAE is computed by adding together all the absolute differences between each observed and forecasted value, 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. Better model accuracy is indicated by a lower MAE.

On the other hand, the loss function represented graphically by loss curves, which are usually plotted against the total number of training epochs. As the optimisation 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 performance of the model, loss curves are a crucial tool. The training loss and validation loss 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 indicates how well the model generalises to new data. A diminishing training loss is expected as the model gets better at reducing the error on the training set. The loss function 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 that minimize the 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, and 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 in terms of preferences or behaviour. It is possible to employ a variety of similarity metrics, such as Jaccard similarity, Pearson correlation, and cosine similarity [24],[25],[26]. 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.

The MAE result was collected for test set for both collaborative filtering and generative AI that use autoencoder for test set. The result show is MAE score when using generative AI is more lower that use the traditional collaborative filtering. Table 2 shows the score obtained for both test set mentioned.

Table 2 MAE score

Evaluation Metric	Test Set (Collaborative Filtering)	Test Set (Generative AI Recommender System)
MAE Score	0.83	0.23

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 in detecting overfitting, finding the optimal epoch for early

stopping, and understanding the convergence of the model. The result is as shown in Fig. 16.

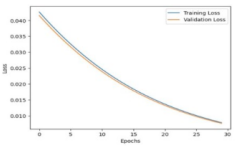


Fig. 16 Loss curves

A decreasing training loss is as expected as the model learns to minimize the error on the training set. However, the validation loss helps identifying 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 brought the spotlight on the advantages and disadvantages of three well-known approaches: CB, CF, and hybrid. Apart from these well-known methods, a new method that incorporates 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 programme. Metrics like MAE, Loss Curves, and Similarity Score will be used to measure how well the models perform in terms of predicting ratings within the dataset.

To improve the recommendation system even further, the next step will involve implementing new models, such as VAE. Metrics like precision and recall will be included to the evaluation to fully examine the models' performances. Concurrently, efforts will be focused on finishing and improving the system prototype and GUI development. The goal of this iterative approach is 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] 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.

[3] 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.

[4] J. K. Tarus, Z. Niu, and G. Mustafa, "Knowledgebased recommendation: a review of ontology-based recommender systems for e-learning," Artif Intell Rev, vol. 50, no. 1, 2018, doi: 10.1007/s10462-0179539-5.

[5] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," Egyptian Informatics Journal, vol. 16, no. 3. 2015. doi: 10.1016/j.eij.2015.06.005.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Q. Zhang, J. Lu, and Y. Jin, "Artificial intelligence in recommender systems," Complex and Intelligent Systems, vol. 7, no. 1, 2021, doi: 10.1007/s40747020-00212-w.

[11] G. Zhang, Y. Liu, and X. Jin, "A survey of autoencoder-based recommender systems," Frontiers of Computer Science, vol. 14, no. 2. 2020. doi: 10.1007/s11704-018-8052-6.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] D. Grewal, J. Hulland, P. K. Kopalle, and E. Karahanna, "The future of technology and marketing: a multidisciplinary perspective," Journal of the Academy of Marketing Science, vol. 48, no. 1. 2020. doi: 10.1007/s11747-019-00711-4.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] S. Chen and W. Guo, "Auto-Encoders in Deep Learning—A Review with New Perspectives," Mathematics, vol. 11, no. 8. 2023. doi: 10.3390/math11081777.

[23] 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.

[24] DataFiniti. (2019). Consumer Reviews of Amazon Products [Data set]. Kaggle. https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products

[25] Y.Y. Chow, S.C. Haw, P. Naveen, E.A. Anaam, H.B. Mahdin, "Food Recommender System: A Review on Techniques, Datasets and Evaluation Metrics", Journal of System and Management Sciences, vol. 13, no. 5, pp. 153-168, 2023.
[26] S.T. Lim, J.Y. Yuan, K.W. Khaw, 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, pp. 43-58, 2023.
[27] P.M. LeBlanc, D. Banks, L. Fu, M. Li, Z. Tang, Q. Wu, "Recommender Systems: A Review". Journal of the American Statistical Association, vol. 119, no. 545, pp. 773-785, 2024.

Appendix B : FYP2 Meeting Logs

Meeting Log 1

TPT3101 Project (FYP2) Meeting Log
Trimester: March 2024 (Trimester ID:2410)

Meeting Date: 10/5/2024	**Meeting No.:** 1
Meeting Mode: (~~In-person~~ / Online)	
Project ID: 2632	**Project Type:** Research-based / Application-based
Project Title : Generative AI Recommender System in E-commerce	
Student ID : 1211303587	**Student Name:** Nur Anis Nabila bt Mohd Romzi
Student Programme and Specialisation: Bachelor of Computer Science B. CS (Hons) Data Science	
Supervisor Name: Haw Su Cheng	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: ~~Implementation~~ / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) ~~/ Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Paper conference regarding FYP1 have been modified and submitted to be review.

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Implementation / Testing (Application-based projects) or Evaluation of Findings and~~ Research Contribution (Research-based projects) ~~/ Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Adding new model which is LSTM.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- None

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- None

PROF. Ts. DR. HAW SU CHENG
Professor
Faculty of Computing and Informatics
Multimedia University, Persiaran Multimedia
63100 Cyberjaya, Selangor Darul Ehsan

……
Supervisor's Signature

……..
Co-Supervisor's Signature
(if applicable)

……...
Student's Signature

……
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student must upload the soft copies of the meeting logs and attach them along with final (FYP2) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 4 to Week 14). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and for checking the attendance requirement of the student, by the FYP Committee.

   This also provides the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provides the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

Meeting Log 2

TPT3101 Project (FYP2) Meeting Log
Trimester: March 2024 (Trimester ID:2410)

Meeting Date: 17/5/2024	**Meeting No.:** 2
Meeting Mode: (~~In-person~~ / Online)	
Project ID: 2632	**Project Type:** Research-based / ~~Application-based~~
Project Title : Generative AI Recommender System in E-commerce	
Student ID : 1211303587	**Student Name:** Nur Anis Nabila bt Mohd Romzi
Student Programme and Specialisation: Bachelor of Computer Science B. CS (Hons) Data Science	
Supervisor Name: Haw Su Cheng	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation / ~~Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Done implement new model which is LSTM and evaluate the model

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation ~~/ Testing (Application-based projects) or Evaluation of Findings and~~ Research Contribution (Research-based projects) ~~/ Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Add more evaluation metrics and optimize them to be more precise

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- None

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- None

PROF. Ts. DR. HAW SU CHENG
Professor
Faculty of Computing and Informatics
Multimedia University, Persiaran Multimedia
. 63100 Cyberjaya, Selangor Darul Ehsan . . .

……
Supervisor's Signature

anstmZ

……..
Student's Signature

……..
Co-Supervisor's Signature
(if applicable)

……
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student must upload the soft copies of the meeting logs and attach them along with final (FYP2) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 4 to Week 14). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and for checking the attendance requirement of the student, by the FYP Committee.

   This also provides the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provides the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

Meeting Log 3

MMU MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

TPT3101 Project (FYP2) Meeting Log
Trimester: March 2024 (Trimester ID:2410)

Meeting Date: 31/5/2024	**Meeting No.:** 3
Meeting Mode: (~~In-person~~ / Online)	
Project ID: 2632	**Project Type:** Research-based / Application-based
Project Title : Generative AI Recommender System in E-commerce	
Student ID : 1211303587	**Student Name:** Nur Anis Nabila bt Mohd Romzi
Student Programme and Specialisation: Bachelor of Computer Science B. CS (Hons) Data Science	
Supervisor Name: Haw Su Cheng	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: ~~Implementation / Testing (Application-based projects) or~~ Evaluation of Findings ~~and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Done implement evaluation metrics and add more of them which are Precision, Recall, F1-Score and NDCG

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation ~~/ Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Test and evaluate LSTM and Autoencoder model.
- Apply the most optimize generative AI in the streamlit.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- None

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- None

PROF. TS. DR. HAW SU CHENG
Professor
Faculty of Computing and Informatics
Multimedia University, Persiaran Multimedia
. 63100 Cyberjaya, Selangor Darul Ehsan . . .

……
Supervisor's Signature

anstnZ

……..
Student's Signature

……..
Co-Supervisor's Signature
(if applicable)

……
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student must upload the soft copies of the meeting logs and attach them along with final (FYP2) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 4 to Week 14). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and for checking the attendance requirement of the student, by the FYP Committee.

   This also provides the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provides the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

Meeting Log 4

TPT3101 Project (FYP2) Meeting Log
Trimester: March 2024 (Trimester ID:2410)

Meeting Date: 7/6/2024	**Meeting No.:** 4
Meeting Mode: (~~In-person~~ / Online)	
Project ID: 2632	**Project Type:** Research-based / Application-based
Project Title : Generative AI Recommender System in E-commerce	
Student ID : 1211303587	**Student Name:** Nur Anis Nabila bt Mohd Romzi
Student Programme and Specialisation: Bachelor of Computer Science B. CS (Hons) Data Science	
Supervisor Name: Haw Su Cheng	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation ~~/ Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Done deploy streamlit of the recommendation system.

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Test the streamlit app and optimize it so that it can run smoothly.
- Start final draft report.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- None

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- None

PROF. Ts. DR. HAW SU CHENG
Professor
Faculty of Computing and Informatics
Multimedia University, Persiaran Multimedia
. 63100 Cyberjaya, Selangor Darul Ehsan . . .

……
Supervisor's Signature

ansrmZ

……..
Student's Signature

……..
Co-Supervisor's Signature
(if applicable)

……
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student must upload the soft copies of the meeting logs and attach them along with final (FYP2) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 4 to Week 14). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and for checking the attendance requirement of the student, by the FYP Committee.

   This also provides the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provides the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

Meeting Log 5

TPT3101 Project (FYP2) Meeting Log
Trimester: March 2024 (Trimester ID:2410)

Meeting Date: 18/6/2024	**Meeting No.:** 5
Meeting Mode: (~~In-person~~ / Online)	
Project ID: 2632	**Project Type:** Research-based / Application-based
Project Title : Generative AI Recommender System in E-commerce	
Student ID : 1211303587	**Student Name:** Nur Anis Nabila bt Mohd Romzi
Student Programme and Specialisation: Bachelor of Computer Science B. CS (Hons) Data Science	
Supervisor Name: Haw Su Cheng	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation ~~/ Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Submit draft report to madam.
- Optimize code and add image for better representation in the streamlit for product recommendation.

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Writing final fyp report

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- None

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- None

PROF. Ts. DR. HAW SU CHENG
Professor
Faculty of Computing and Informatics
Multimedia University, Persiaran Multimedia
. . . .

……
Supervisor's Signature

……..
Student's Signature

……..
Co-Supervisor's Signature
(if applicable)

……
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student must upload the soft copies of the meeting logs and attach them along with final (FYP2) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 4 to Week 14). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and for checking the attendance requirement of the student, by the FYP Committee.

   This also provides the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provides the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

Meeting Log 6

TPT3101 Project (FYP2) Meeting Log
Trimester: March 2024 (Trimester ID:2410)

Meeting Date: 27/6/2024	**Meeting No.:** 6
Meeting Mode: (~~In-person~~ / Online)	
Project ID: 2632	**Project Type:** Research-based / Application-based
Project Title : Generative AI Recommender System in E-commerce	
Student ID : 1211303587	**Student Name:** Nur Anis Nabila bt Mohd Romzi
Student Programme and Specialisation: Bachelor of Computer Science B. CS (Hons) Data Science	
Supervisor Name: Haw Su Cheng	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: ~~Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) /~~ Draft Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finalize report.
- Finalize streamlit process.
- Finalize slide and video for demonstration.

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

None

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- None

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- None

PROF. Ts. DR. HAW SU CHENG
Professor
Faculty of Computing and Informatics
Multimedia University, Persiaran Multimedia
. 63100 Cyberjaya, Selangor Darul Ehsan . . .

……
Supervisor's Signature

anstmz

……..
Student's Signature

……..
Co-Supervisor's Signature
(if applicable)

……
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

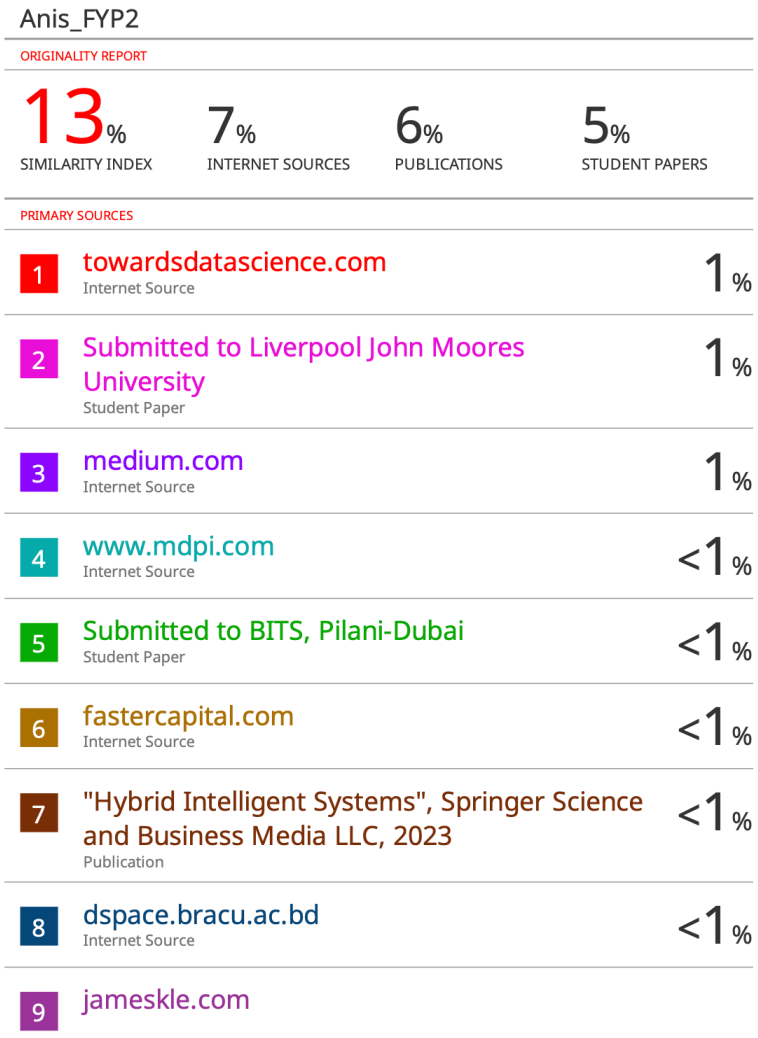
2. Student must upload the soft copies of the meeting logs and attach them along with final (FYP2) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 4 to Week 14). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and for checking the attendance requirement of the student, by the FYP Committee.

   This also provides the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provides the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

Appendix C : Turnitin Similarity Index Page



Anis_FYP2

ORIGINALITY REPORT

13%	7%	6%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	towardsdatascience.com Internet Source	1%
2	Submitted to Liverpool John Moores University Student Paper	1%
3	medium.com Internet Source	1%
4	www.mdpi.com Internet Source	<1%
5	Submitted to BITS, Pilani-Dubai Student Paper	<1%
6	fastercapital.com Internet Source	<1%
7	"Hybrid Intelligent Systems", Springer Science and Business Media LLC, 2023 Publication	<1%
8	dspace.bracu.ac.bd Internet Source	<1%
9	jameskle.com	

Appendix D: Github Link for Code

Generative-AI-in-Recommender-System-for-e-Commerce