A HYBRID OF DEEP AUTO-ENCODER AND FEATURE EMBEDDING MODEL FOR AN E-COMMERCE RECOMMENDER SYSTEM

JUSTIN MURITHI IRERI

A Thesis Submitted to the Graduate School in Partial fulfillment of the Requirements for the Award of the Degree of Master of Science in Computer Science of Chuka University

CHUKA UNIVERSITY
OCTOBER 2024

DECLARATION AND RECOMMENDATION

Declaration

This thesis is my original work and has not been submitted for an award of diploma or conferment of degree in any other University.

Signature: _____ Date: 16/10/2024

Justin Murithi Ireri
SM22/51212/21

Recommendation

This thesis has been examined, passed, and submitted with our approval as the university supervisors

Signature: _____ Date: 16/10/2024

Dr. Edna Too, PhD.
Chuka University

Signature: _____ Date: 16/10/24

Prof. Moses Muraya, PhD.
Chuka University

DEDICATION

This piece of work is dedicated to my wife Gatugi Murithi for her unwavering support and prayers, my Son Jabali Murithi, my father Robert Ireri who has always been there, mother and my siblings for their inspirational words of encouragement during my period of study. May God bless you all.

ACKNOWLEDGEMENT

I wish to express my gratitude to the Divine Creator for bestowing upon me the vigor and resolve to initiate and finalize this endeavor. I also extend my sincere thanks to Chuka University for providing me with a chance to pursue my studies in an environment so conducive and supportive during my master's degree journey. Blessings to this premier and prophetic University. My profound appreciation goes to my mentors and supervisors, Dr. Edna Too Chebet, PhD, and Prof. Moses Muraya, PhD for their unwavering support, mentorship, and motivation their constructive and honest feedback on this thesis. Their encouragement, insights, and invaluable contributions are deeply valued. I also wish to extend my gratitude to all members of the Computer Science and ICT departments who facilitated my research and various consultations. My heartfelt thanks also go to my peers Saif Kinyori, and Muigai Muiruri, family, and friends for their perpetual help, guidance, and support throughout my scholarly journey. To everyone, I encapsulate my gratitude by saying,

May God bless you all.

ABSTRACT

Recommender systems aim to predict user interests and suggest products that are likely to be of interest. These systems are widely used across various platforms, including online shopping, streaming services, and music stores, to provide personalized suggestions. Traditional machine learning-based models, such as collaborative filtering and content-based algorithms, often face challenges like low accuracy, data sparsity, and the cold start problem. The cold start problem occurs when a system lacks sufficient data to make accurate recommendations for new users or items. This study specifically focuses on addressing the visitor cold start problem, where the system does not have prior information about the new user's preferences or behavior, making personalized recommendations difficult. To address this issue, a model was developed using deep auto-encoders integrated with feature embedding (DAE-FE), designed to improve item prediction accuracy for new users in an e-commerce recommender system. The model introduces an embedding layer after the dropout layer in the deep neural network, which automatically captures user data points such as time and location. These data points help in constructing a user profile necessary for prediction. This feature not only improves the accuracy of item predictions but also speeds up the process by filling in missing data for new users, allowing the system to proceed directly to prediction. An experimental research design was employed to compare the performance of the developed model with previous models that relied solely on provided datasets. In the experiment, user location and time of login were used as independent variables, while model accuracy served as the dependent variable. The model was trained and tested using the MovieLens 100k dataset, which was adapted to meet the requirements of the DAE-FE model. The hybrid model achieved a mean squared error of 0.0241 and a root mean squared error of 0.1443, indicating minimal deviation from the actual values. As a result, the model attained approximately 96% accuracy in predicting recommendations for cold start users. Overall, the model demonstrated strong performance and appears to be a promising solution for the cold start problem in e-commerce systems. The research found that incorporating more side information from users and items on the dataset during the model's training will yield more accuracy in item prediction.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS AND ACRONYMS

AE: Auto-encoder

ANN: Artificial Neural Network

CB: Content-Based

CDNN: Clustering deep neural networks

CF: Collaborative filtering

CNN: Convolutional Neural Network

CSP: Cold start Problems

CTR: Click-through rate

DAE: Deep Auto-encoder

DBCA: Density based clustering algorithms

DBN: Deep belief network

DEC: Deep Embedded Clustering

DL: Deep learning

DNN: Deep neural network

IDEC: Improved Deep Embedded Clustering

MLP: Multilayer Perceptron

MRR: Mean reciprocal rank

RBM: Restricted Boltzmann machine

RMSE: Root Mean Square Error

RS: Recommender System

STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks

SVD: Simple value decomposition

DAE-FE: Deep Auto Encoder with Feature Embedding

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

A recommender system is designed to predict items that may interest users based on specific profiles (Murad *et al.*, 2020). In the context of e-commerce, these systems rely on various factors, such as item descriptions, customer ratings, and purchase history, to suggest products to users. Since the 1990s, recommender systems have been applied in e-commerce platforms, providing item suggestions for products like e-books, e-movies, job applications, e-learning courses, and hotel bookings (Goyani *et al.*, 2020). These systems use user-item relationships, item similarities, and user profile similarities to generate personalized recommendations. Users and items are the two essential inputs required for a recommender system to make predictions, with users being the recipients of suggestions and items being the products recommended. Feedback from users who have interacted with the system is vital to improving recommendation accuracy (Nunez *et al.*, 2018).

There are two types of feedback: implicit and explicit (Goyani *et al.*, 2020). Implicit feedback is indirectly collected from users based on their actions and behaviors, such as browsing or purchasing items. Since it is gathered automatically during user interactions, implicit feedback provides numerous data points, making it a convenient way to collect information. Explicit feedback, on the other hand, is directly provided by users through ratings, reviews, or recommendations. Although explicit feedback offers more accurate data, it has a major drawback: not all users are willing to leave reviews or complete surveys, leading to a lack of sufficient data (Wang *et al.*, 2021).

Recommender systems face several challenges, including sparsity, generalization, memorization, and the cold start problem (Cui *et al.*, 2020). The sparsity issue arises when the number of rated items is significantly smaller than the number of unrated items. This results in a sparse user-item interaction matrix, making it difficult to find patterns and leading to poor recommendations. For example, in the Netflix dataset, only one percent of videos are rated by over 480,000 users, highlighting the impact of data sparsity (Liu *et al.*, 2023). Generalization and memorization are issues commonly associated with content-based recommender systems. Generalization occurs when a

model uses outdated approaches to solve new problems, while memorization refers to a model's inability to predict unseen data, relying solely on training data. Both of these problems limit the diversity of recommendations (Hashemi *et al.*, 2020).

The cold start problem occurs when the system is unable to generate accurate predictions due to the lack of prior data for new users or items. In particular, a user cold start problem arises when a new user joins the platform without any past interaction or information, making it difficult to create personalized recommendations. Similarly, an item cold start problem occurs when a new item is introduced to the system without any ratings, resulting in fewer recommendations until more data is collected (Goyani *et al.*, 2020).

Machine learning, a branch of artificial intelligence, has been employed to develop recommender systems for e-commerce platforms, enabling these systems to improve predictions based on data rather than explicit programming (Mahesh, 2020). The visitor cold start problem, which occurs when a new user interacts with the system for the first time, has been a key focus of this study. Traditional methods like hybrid, content-based, and collaborative filtering approaches have struggled with low accuracy, as they often rely heavily on prior user information for profile creation (Zhang *et al.*, 2018).

Content-based filtering generates recommendations based on data derived from user and item profiles, which may include attributes like gender, item descriptions, and keyword analysis (Wang *et al.*, 2018). The system suggests items that share similar content with those that a specific user has previously liked. This method is particularly applicable when the domains of the items are readily accessible. However, this limitation means that the model is unable to predict items for new users and can only provide recommendations based on past user interests (Lops *et al.*, 2023). Another approach employed in recommender systems is collaborative filtering, which makes predictions about a user based on the interactions and preferences exhibited by other users (Aljunid *et al.*, 2020). This technique utilizes historical behavior and preference data, sidestepping the need for current personal information. Collaborative filtering is divided into two methodologies: user-based and item-based techniques. Both approaches struggle to address the user cold start problem, as they rely solely on the

target user's past history and cannot infer missing values for new users (Goyani *et al.*, 2020).

In the user-based approach, the model identifies users with similar tastes based on previous preferences. The fundamental assumption is that users who have agreed on item preferences in the past will likely agree on new items in the future. Thus, the system recommends items liked by similar users that the target user has not yet interacted with. However, scalability becomes an issue as the amount of data grows, and this method often performs poorly in addressing cold start problems (Ajaegbu, 2021). Conversely, item-based collaborative filtering focuses on the similarity between items. The underlying assumption is that a user will likely enjoy items similar to those they have previously liked. The system constructs a similarity matrix based on user interactions and suggests similar items. This method falters when a user has no prior purchase history or when preferences fluctuate frequently (Singh *et al.*, 2020).

Hybrid models combine collaborative and content-based methods or integrate various other architectures. These models have demonstrated significant improvements in overcoming scalability issues in recommendation systems. However, they still encounter challenges related to low accuracy concerning cold start problems (Yun *et al.*, 2018). Traditional recommender systems primarily depend on available datasets, analyzing user history, preferences, and item descriptions to establish patterns that inform item predictions (Kharita *et al.*, 2018). Since these models derive predictions based on existing user history, they cannot adequately handle cold start scenarios. If an item was not present during training, the system cannot generate an embedding or make a recommendation (Nilashi *et al.*, 2018).

This study presents a working prototype utilizing a deep auto-encoder architecture within the realm of deep learning to enhance accuracy in item prediction, particularly addressing visitor cold-start issues in product-based recommender systems. The developed model improves prediction accuracy for new users through the integration of deep auto-encoders with a feature-embedding layer. This approach allows for the generation of missing values for new users by capturing real-time data through feature embedding techniques. Feature embedding involves mapping input data to a lower-

dimensional space, ensuring that relevant features are retained while eliminating noise and redundant information. This process is vital for reducing dimensionality and improving the model's capacity to capture critical information, especially in cold start situations.

Auto-encoders, often referred to as unsupervised artificial neural networks, are designed to learn how to compress and encode data before reconstructing it from the compressed representation to closely resemble the original input. This design enables auto-encoders to effectively reduce noise and recover missing values within a dataset (Pan *et al.*, 2020). A deep auto-encoder comprises two symmetrical deep-belief networks, each containing more than two shallow layers. One network acts as the encoding layer, while the other serves as the decoding layer. This structure enables the model to learn and generate missing values for new users with zero ratings, a capability that traditional prediction methods have not achieved (Shambour, 2021). Auto-encoders excel in various unsupervised learning applications, including dimensionality reduction, generative modeling, and efficient coding (Rudin *et al.*, 2022). They have shown remarkable advantages in learning underlying feature representations across multiple fields, such as language modeling, speech recognition, and computer vision (Ling *et al.*, 2023). Due to their potential, auto-encoders have been integrated into new recommendation frameworks, opening avenues to enhance user experiences and drive customer engagement (Xie *et al.*, 2016).

1.2 Statement of the Problem

Recommendation systems have become very crucial for any e-commerce business. Online market platforms have been using content-based filtering, collaborative filtering, or both (hybrid architectures) to recommend items, where they base their predictions on customers' past purchasing patterns, preferences, and ratings. These techniques fail to update real-time customer details, so if customer preferences change, the system cannot adjust. The prediction system depends only on the training datasets and, hence, cannot deal with real-time changes or new users. This is a result of models lacking the capability to incorporate feature embedding to capture users' new changes. The prediction accuracy worsens for the model to predict when a new user with no history (visitor cold start problem) gets into the system. The inability of these

techniques to create user profiles for new users lowers the accuracy of a recommender system.

1.3 Objectives of the Study

1.3.1 General Objective

To develop a recommender model using deep auto-encoder architecture with feature embedding (DAE-FE) to increase accuracy in solving visitor cold start problems on an online product-based e-commerce platform.

1.3.2 Specific Objectives

i. To develop a deep auto-encoder-based model incorporated with feature embedding architecture to solve a visitor cold start problem.
ii. To evaluate the accuracy of the developed model for the visitor cold start problem.
iii. To develop a prototype for the developed model.

1.4 Research Questions

i. How can a model for the cold start problem using a deep auto-encoder incorporated with feature embedding architecture be developed?
ii. How can the accuracy of the developed model be achieved?
iii. How can a prototype for the cold start problem model be developed?

1.5 Significance of the Study

In the current world and particularly in e-business, the usage of recommender systems is crucial in that it provides positive impacts to every user and the business owner. From the user end, such systems build product awareness and occasion product consumption due to customized ads to the users thus achieving the target consumers for the businesspersons. Besides, they are also useful for decision-making since they include data collected from customers through the system. Any customer always feels pleased when the recommender system designed is friendly and actually provides the expected items to be purchased, then they will spend less time searching for their favorites and even get the cost estimates.

1.6 Scope of the Study

The internet market has grown steadily during the COVID-19 lockdown period and has one of the world's most active e-commerce ecosystems. The majority of firms have chosen to build online stores, and because so many people use the internet, it is anticipated that global e-commerce will continue to grow and accelerate. With this information, the study was based on an online e-commerce platform. The study used deep auto-encoders with feature embedding to develop a recommender model for new users with no previous purchase history. The model will be used by e-commerce platforms to make personalized item predictions for new users. The training and evaluation were done using a cloud-based platform called Neural Designer, Google Colab, and Python. The dataset for training and evaluation was the MovieLens100k dataset from Kaggle. The dataset was split into training, testing, and validation sets with a percentage of 60%, 20%, and 20%, respectively.

1.7 Assumptions of the Study

i. The dataset is independently and uniformly distributed.
ii. The user will allow the model to access their real-time information and their location.

1.8 Operational Definition of Terms

Cold start problem:	A new item is introduced to the catalog without any reviews or a new user to the system with no prior history.
Collaborative filtering (CF):	A technique that gathers preferences or tastes data from numerous users to automatically forecast (filter) a user's interests.
Content-based (CB):	A machine learning technique that uses similarities in features to make decisions.
Convolutional neural network (CNN):	A deep neural network type that is frequently used to evaluate visual imagery.
Deep learning:	A machine learning technique that uses many processing layers to extract progressively higher-level features from data. It is based on artificial neural networks.
Deep neural network (DNN):	An artificial neural network model, which has several layers between the input and output layers.
Multilayer perceptron (MLP):	A feedforward artificial neural network that creates a collection of outputs from a set of inputs.
Recommender system:	A program that suggests products to consumers based on their ratings and reviews.

CHAPTER TWO
LITERATURE REVIEW

2.1 Machine learning Techniques used in Solving Cold Start Problem

Machine learning, a subset of artificial intelligence (AI), centers on creating models and algorithms that empower computers to learn from data, enabling them to make decisions or predictions without explicit programming. The primary goal is to design systems that automatically improve as they are exposed to more data. Machine learning can be categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning, each offering unique methods for processing data and extracting insights.

Supervised learning trains models using labeled datasets, where input data is associated with corresponding output labels. The algorithm learns to associate incoming data with the correct output based on this labeled information. Techniques like classification and linear regression are prevalent in supervised learning. Linear regression models the connection between input data and a target variable through a linear equation, making it suitable for predicting continuous variables (Maulud & Abdulazeez, 2021). In contrast, classification aims to predict the appropriate label or category for new data based on a model trained with labeled examples. The fundamental principle of supervised learning is to extrapolate patterns from known examples to make accurate predictions on unseen data.

On the other hand, unsupervised learning works with unlabeled data, allowing the system to identify patterns or structures without human intervention (Hiran *et al.*, 2021). This method is especially useful for tasks like dimensionality reduction and clustering. Dimensionality reduction seeks to decrease the number of input features while preserving essential information, aiding in the visualization of complex datasets, improving training efficiency, and minimizing noise in the data (Verbeeck *et al.*, 2020). Clustering, another critical technique in unsupervised learning, groups similar data points according to specified criteria. Its applications include customer segmentation, anomaly detection, image segmentation, and natural language processing tasks such as document clustering (Golalipour *et al.*, 2021). Both dimensionality reduction and

clustering play significant roles in organizing and simplifying large datasets, facilitating more efficient analysis and interpretation.

Reinforcement learning distinguishes itself from supervised and unsupervised learning by concentrating on decision-making through interaction with an environment. In this framework, an agent learns by receiving feedback in the form of rewards or penalties, which enables it to refine its strategy over time. The objective is for the agent to develop a policy that maximizes cumulative rewards based on its interactions (Padakandla, 2021). Reinforcement learning is particularly relevant in scenarios requiring adaptive behavior, such as robotics, gaming, and autonomous systems, where learning from experience is vital.

Addressing the cold start problem in recommender systems—an issue that occurs when there is insufficient interaction data for users or items—necessitates specialized approaches. This review explores three primary methods: collaborative filtering, content-based filtering, and hybrid models, which integrate elements from both (Himeur *et al.*, 2022). Techniques such as matrix factorization and k-nearest neighbors (k-NN) also play crucial roles in building effective recommender systems. The evolution of recommender systems, spurred by the rapid growth of e-commerce since the 1990s, has led to significant advancements in personalized recommendations (Goyani *et al.*, 2020).

Matrix factorization is one of the most widely used techniques for modeling user-item interactions in recommender systems. This approach decomposes a sparse user-item interaction matrix into two lower-rank matrices representing users and items, respectively. This decomposition captures latent factors or hidden patterns that influence interactions, making it particularly effective for addressing sparsity challenges (Isinkaye, 2023). The process begins by representing interactions, such as ratings or clicks, in a matrix. This matrix is subsequently decomposed into two smaller matrices, with latent factors learned iteratively by minimizing the difference between observed interactions and predicted outcomes. Regularization terms are introduced to prevent overfitting by penalizing large values in the matrices. Once trained, the model can predict missing entries in the interaction matrix, such as estimating a user's rating

for an unrated item (Abdi *et al.*, 2018). Extensions of matrix factorization, including the incorporation of bias terms or the application of non-negative matrix factorization, further enhance the system's performance by capturing more nuanced relationships (D'Amico *et al.*, 2023).

Nearest neighbor approaches are another essential component in recommender systems, identifying similar items or users based on defined similarity metrics. The k-NN algorithm serves as a straightforward implementation that recommends items or users most similar to a target based on their similarity scores (Bansal & Choudhary, 2022). These approaches can be user-based, where the system identifies users with similar preferences to the target user, or item-based, where the system identifies items similar to those the target user has engaged with. User-based methods generate recommendations by examining the preferences of users with shared interests, while item-based methods recommend items based on their similarity to those previously liked by the user. Despite the simplicity and intuitiveness of nearest neighbor methods, they encounter significant challenges in addressing the cold start problem, where new users or items lack interaction history (Priyadharshini *et al.*, 2021). In response to these limitations, deep learning techniques are increasingly being explored as more sophisticated solutions capable of overcoming such challenges.

2.1.1 Content-based Filtering

Content-based filtering is a widely used method in recommender systems that relies on item-item similarity to generate predictions. This approach utilizes features such as customers' past preferences and likes to inform its recommendations. It estimates the similarity between item profiles and user preferences (Pal *et al.*, 2023). This technique employs various measures, including statistical functions or machine learning models, to assess similarities. It tends to perform better in domains where item characteristics are readily available, such as in the cases of books, jobs, and movies; however, it may not significantly enhance user interest. Figure 2.1 illustrates the architecture of content-based filtering as recorded by (Zhou *et al.*, 2023).

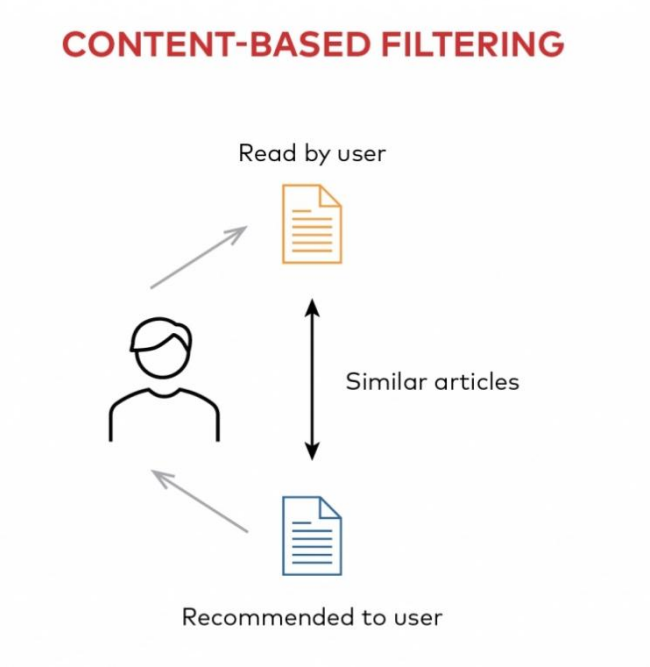


Figure 2.1: Content-Based Collaborative

A collection of characteristics or descriptors is used to represent each item in the system. These characteristics could be descriptions, categories, or anything else that characterizes the product. The system keeps track of each user's profile based on their traits and preferences during the model training process. Over time, as the user interacts with or rates the items, this profile is constructed. The algorithm searches for things that fit the user's profile before making recommendations to them (Papadakis *et al.*, 2022). It suggests products with attributes that are comparable to those the user has already shown interest in.

Relying on the clear properties of items, content-based suggestions improve system transparency by making it easier for consumers to comprehend the recommendations made for certain objects. Nevertheless, issues like over-specialization and cold start issues affect these systems. Content-based filtering systems may find it difficult to introduce consumers to completely new or unexpected products because their suggestions are based on the user's past preferences (Kannout *et al.*, 2023). Over-specialization occurs when recommendations are based on the user's past preferences only, and this can lead to content-based filtering systems struggling to introduce users to entirely new or unexpected items.

Several content-based filtering systems, especially those seen in online and e-commerce settings, use linguistic elements taken from product descriptions or web pages as content descriptors. Typically, these systems take advantage of well-established document modeling strategies that come from studies on information retrieval and information filtering (Abri *et al*., 2020). The term frequency-inverse document frequency (TFIDF) term-weighting model is frequently used to characterize user and object profiles as weighted term vectors (Roelleke, 2022).

User interest predictions for specific items are derived from calculations of vector similarities, employing methods such as cosine similarity or probabilistic techniques like Bayesian classification. Unlike collaborative filtering methods, which rely on shared user interactions, content-based profiles are tailored to individual users, formed exclusively from features linked to items they have previously encountered or rated (Gao *et al.*, 2020).

In user independence, content-based recommenders build their profiles by using ratings given by the active user (Afoudi *et al*., 2021). In contrast to collaborative filtering, which depends on user input to find comparable preferences, content-based approaches are independent of outside assessments. Transparency occurs by explicitly citing the content features or descriptions that influenced an item's inclusion in the recommendations; content-based recommenders can offer justifications for their selections. Thanks to this transparency, users can evaluate the recommendations' foundation. Conversely, collaborative systems function as "black boxes," with little explanation for why a specific product is advised.

Content-based recommenders fail to predict items that have not yet been rated by any users. Collaborative systems rely solely on existing user preferences, making it challenging to recommend new items until a sufficient number of users have rated them. Content-based filtering systems have a primary drawback, which is a tendency to over-specialize in item selection. This is due to profiles being constructed solely based on a user's past ratings, potentially missing opportunities to recommend unexpected items that users find valuable. Additionally, the effectiveness of content-based filtering is contingent on the practical representation of items using extracted textual features,

which may not always be feasible given the diverse nature of web data (Javed *et al*., 2021).

2.1.2 Collaborative Filtering

Collaborative filtering is a widely used approach in recommendation systems that provides users with personalized options. This method is based on the premise that users who have agreed in the past are likely to do so again in the future. Essentially, if two users, A and B, exhibit similar preferences for certain items, it can be inferred that they will likely share similar preferences for other items as well (Zade *et al.*, 202). The collaborative filtering system operates by utilizing a user-user matrix to make predictions. By assessing the similarities in user histories, it recommends items with the highest predicted values to users who exhibit comparable characteristics. There are two primary types of collaborative filtering: item-based and user-based. Figure 2.2 illustrates the architecture of collaborative filtering (Zhou *et al.,* 2023).

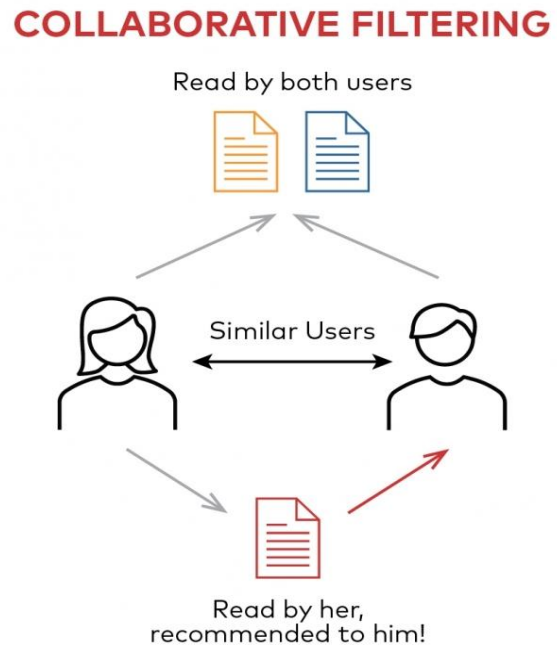


Figure 2.2: Collaborative filtering

User-based collaborative filtering recommends items to individuals based on the preferences of users who share similar tastes. This model identifies users with comparable interests and suggests items that those similar users have previously liked or interacted with. In the process of item prediction, the model computes similarity

scores between the target user and other users, subsequently categorizing the user into a specific group (Hu *et al.*, 2020). Once grouped, the model identifies users with the highest similarity scores and recommends items that those users have liked but that the target user has not yet engaged with.

In contrast, item-based collaborative filtering generates recommendations based on the similarity of items to those the user has previously liked or interacted with. This model identifies products that bear resemblance to the items of interest to the user. By analyzing user interactions, it constructs an item-item similarity matrix to pinpoint items that the target user has enjoyed. Item-based systems are particularly effective in environments with large datasets and sparse interactions, as exemplified by Amazon's use of this algorithmic approach (Xue *et al.*, 2019). Nonetheless, a significant limitation of this method is its dependence on the user's past interests, which restricts the model's capacity to discover new interests (Gao *et al.*, 2019). Additionally, this approach employs a bootstrapping technique, whereby a collaborative filtering system begins with minimal user data and progressively enhances its predictions as more ratings are accumulated (Dash & Patro, 2022).

Despite its widespread use, collaborative filtering presented several challenges. The cold start problem, particularly with new users or items that lacked interaction history, was a notable issue. Data sparsity in large datasets, where few users interacted with most items, resulted in sparse matrices, making it difficult to identify meaningful relationships between users and items. This sparsity negatively impacted both the timeliness and accuracy of recommendations. Furthermore, the computational cost of similarity measures increased as the volume of users and items grew, leading to scalability concerns (Bhattacharjee & Mitra, 2021).

To address these shortcomings, various models were proposed. Chen *et al*. (2021) developed a collaborative filtering model incorporating double-clustering and user trust, utilizing a user-clustering matrix to measure similarity levels. While matrix factorization-based collaborative filtering models improved recommendation accuracy, they did not fully resolve the cold start issue (Wang *et al*., 2018). Mehrbakhsh *et al*. (2018) addressed sparsity in collaborative filtering systems by integrating ontology and

SVD, constructing an item-user framework. Liu *et al*. (2017) enhanced collaborative filtering by utilizing blog textual data labeled according to user opinions, generating a user-item rating matrix. This model was later improved with proximity significance singularity to address inaccuracies arising from cosine similarity.

Further advancements included the development of a sentimental topic matrix factorization model, which captured true user preferences by analyzing sentiment in user reviews while utilizing topic-based datasets to refine user matrix factorization. This model demonstrated improved interpretability regarding user preferences (Wang *et al*., 2018). An additional algorithm combined item attributes with user rating preferences and a time-weighting factor to enhance item similarity computation, though it struggled with forecast accuracy due to sparsity in its architecture (Zhu *et al*., 2020). Hong and Yu (2019) proposed a correlation coefficient-based collaborative filtering recommendation method aimed at addressing data sparsity by filling in unrated items based on correlation coefficients and employing semantic similarity to improve item similarity computations. Another model integrated bio-inspired clustering ensembles with collaborative filtering suggestions to determine the closest neighborhood for users, creating a recommendation similarity matrix (Logesh *et al*., 2020).

Recent developments also included the collaborative vibrational auto-encoder, designed to enhance N-top recommender systems through auto-encoders and denoising techniques. Unlike traditional systems limited to single-source data (e.g., text or ratings), auto-encoder-based systems managed multiple data types, including audio, video, and visual information (Liu *et al*., 2017). A new approach, the Entity-aware Collaborative Relation Network (ECRN), leveraged knowledge graphs to capture relational semantics at the entity level. This method generated user representations by constructing meta-paths between users, items, and entities while incorporating a self-attention mechanism to filter collaborative signals. ECRN outperformed current state-of-the-art models across multiple benchmarks (Li *et al*., 2021).

Moreover, a probabilistic cluster-level latent factor (PCLF) model was developed to learn common rating patterns across domains while capturing specific user and item clustering behaviors within each domain (Fu *et al*., 2019). Lai *et al*. (2023) presented a

model that combined convolutional neural networks to learn features from user and item reviews with matrix decomposition to predict ratings, effectively mitigating the impact of data sparsity on recommendation outcomes.

2.1.3 Hybrid Recommender Systems

Hybrid models in recommender systems merge various recommendation techniques, such as collaborative filtering and content-based filtering, to overcome limitations like sparsity, cold start issues, and scalability challenges. By leveraging the strengths of each approach, these models typically outperform single-method strategies. For example, a hybrid model might utilize collaborative filtering to examine user interactions while also employing content-based filtering to consider item characteristics, such as genres or product attributes (Panda *et al.*, 2022).

One illustrative example of this hybrid approach is the blend model-based algorithm known as User-Item Collaborative Filtering (UICF), which integrates user-based collaborative filtering (UCF) and item-based collaborative filtering (ICF) with linear regression to address scalability and sparsity issues in user-item matrices. Additionally, this model incorporates tags and contextual information to enhance the personalization of recommendations (Aljunid & Huchaiah, 2021). Another model, BiUCF, combines UCF and ICF while utilizing biclustering to reduce dimensionality, thereby clustering users and items more effectively, which in turn improves the prediction of user preferences. The hybrid popularity model also tackles cold start issues by taking into account both user and item popularity when constructing user-item matrices (Ifada *et al.*, 2020).

Some advanced hybrid models have incorporated deep learning techniques, such as the dynamic selection model, which uses RNNs and attention mechanisms to refine e-commerce recommendations based on user behavior and temporal patterns (Wang *et al*., 2020). Other innovations include knowledge graphs and unified embeddings to integrate auxiliary data and enhance recommendation diversity and accuracy (Hui *et al*., 2022). In content-based hybrid models, collaborative filtering is combined with content features like user reviews, item descriptions, and metadata. The hybrid collaborative content model, for instance, considers both user interaction history and

item features to improve recommendation precision (Channarong *et al*., 2022). The weighted fusion model assigns dynamic weights to collaborative and content-based components, optimizing the recommendation based on performance metrics (Do *et al*., 2020).

Temporal hybrid models take into account changes in user behavior over time, enhancing relevance by factoring in recent interactions. For example, music streaming services may blend collaborative filtering with temporal trends to recommend songs that align with both the user's long-term and current tastes (Zhang *et al*., 2023). Stacking models further optimizes recommendation systems by combining multiple models and aggregating their outputs. In an e-commerce scenario, recommendations might result from blending collaborative filtering, content-based methods, and popularity-based models for well-rounded suggestions (Wang *et al*., 2019).

Factorization machines extend matrix factorization by integrating side information such as content attributes or user demographics, allowing for more sophisticated interaction modeling (Chowdhury, 2022). Hybrid meta-learning techniques dynamically adjust the blending strategy based on the user or item context, improving adaptability in changing environments (Luo *et al*., 2020). Despite these innovations, hybrid models still face challenges in real-world applications, especially in addressing sparsity and cold start scenarios. One promising solution is combining feature embedding with a deep auto-encoder within a deep neural network, which enhances accuracy in sparse data environments by identifying patterns, filling in gaps, and mitigating sparsity issues.

2.2 Deep Learning Models in Solving a Visitor Cold Start Problem

2.2.1 Neural Network Architectures

Neural networks play a pivotal role in deep learning, a subset of machine learning dedicated to modeling and addressing intricate problems. Deep learning draws inspiration from the architecture and functionality of the human brain, particularly the neural networks that make up the nervous system. By employing multiple layers within artificial neural networks, deep learning algorithms strive to learn hierarchical data

representations, discern complex patterns, and produce accurate predictions (Pramod *et al.*, 2021).

Artificial neural networks consist of layers of interconnected nodes, known as neurons, which are crucial for constructing deep learning models. During the training phase, the network adjusts the weights assigned to the connections between nodes, allowing it to generate precise predictions or classifications. When an artificial neural network has multiple hidden layers, it is commonly referred to as a deep neural network (DNN) (Mehrer *et al.*, 2020). The architecture of DNNs varies based on the nature of the problem and the type of data involved. Among the notable architectures are convolutional neural networks (CNNs), which are specifically designed for tasks involving visual input, such as object detection, image classification, and image recognition. According to Tulbure *et al.* (2022), CNNs excel at capturing the spatial hierarchies of features within images, significantly advancing image processing and computer vision technologies.

The architecture of CNNs comprises several key layers. Convolutional layers perform element-wise multiplications and aggregations on the input image by sliding a small filter or kernel across the image to detect local patterns. Through these convolutional operations, CNNs identify pertinent features within the input image. Pooling layers subsequently reduce the spatial dimensions of the input volume, facilitating down-sampling while retaining essential information (Gholamalinezhad & Khosravi, 2020). Common pooling techniques include average pooling, which calculates the average value in a designated region, and max pooling, which retains the maximum value. After extracting hierarchical features through the convolutional and pooling layers, CNNs typically incorporate fully connected layers to generate predictions based on the learned features. In these layers, each neuron is linked to every neuron in the adjacent layers, creating dense connections that enhance the classification or prediction process (Basha *et al.*, 2020). Figure 2.3 illustrates the structure of a convolutional neural network.

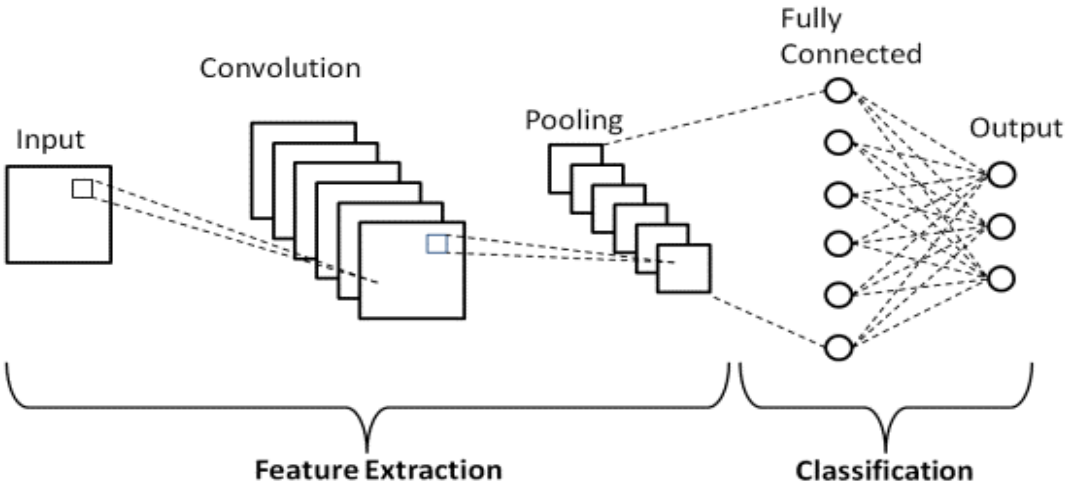


Figure 2.3: Convolutional neural network

A recurrent neural network (RNN) is a type of artificial neural network designed to process sequential data and recognize temporal dependencies. Unlike traditional feedforward neural networks, RNNs possess connections that form directed cycles, allowing them to retain information from previous time steps within a hidden state (Sherstinsky, 2020). This unique characteristic makes RNNs particularly suitable for tasks involving sequences, such as time series analysis, speech recognition, and natural language processing, among others. Figure 2.4 illustrates the architecture of an RNN.

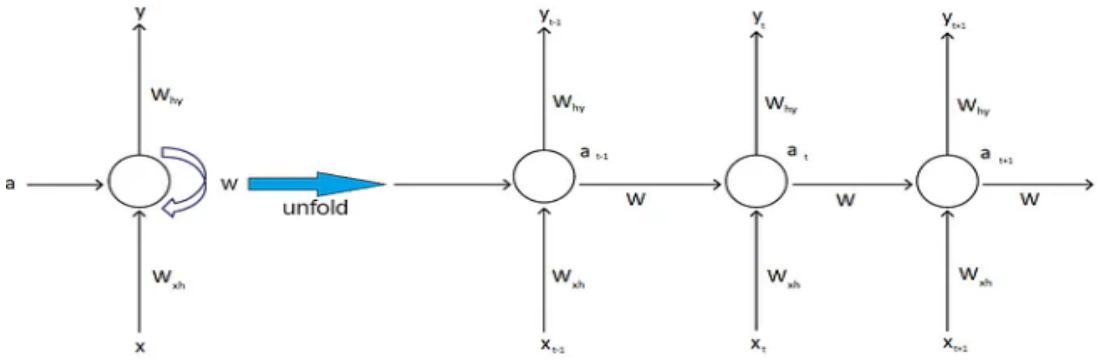


Figure 2.4: Recurrent neural network (RNN)

In neural networks, the parameter W_{xh} denotes the weights that connect the input layer to the hidden layer, while W represents the weights linking one hidden layer to another. The weights connecting the hidden layer to the output layer are denoted by W_{hy}, and a refers to the activation of the layer.

The Convolutional Matrix Factorization (ConvMF) recommendation model combines probabilistic matrix factorization (PMF) with a convolutional neural network (CNN), as introduced by Liu *et al.* (2022). This integration allows ConvMF to effectively extract contextual information from documents, leading to improved accuracy in rating predictions. Extensive experiments on three real-world datasets have shown that ConvMF consistently surpasses other state-of-the-art recommendation models, even in cases of highly sparse rating data. Moreover, an enhanced cross-feature layer based on factorization machines (FM) was developed, which outperformed various advanced cross-feature techniques and significantly boosted model performance. In addition, Yan *et al.* (2020) presented a novel joint approach for recommendation systems that learns tag preferences from a single-clicked news article.

Additionally, a new deep learning model for recommendations, known as the Deep Rating and Review Neural Network (DRRNN), has been proposed. Unlike traditional models that primarily utilize review text as auxiliary information, DRRNN leverages both the target rating and review for a specific user-item pair as ground truth during the error backpropagation phase of training. This strategy enables the preservation of richer semantic information from reviews, resulting in more precise rating predictions. Extensive evaluations on four publicly available datasets have confirmed the effectiveness of the DRRNN model for rating prediction, as highlighted by Xi *et al.* (2021).

Research has also extended into the realm of creative writing with the introduction of a multi-task neural network model aimed at the automatic synthesis of poetry and couplets. This model employs a Seq2Seq encoding and decoding framework that integrates multi-task learning with parameter sharing, self-attention processes, and attention mechanisms. During the encoding stage, two BiLSTM networks are utilized to capture the similar features of traditional poems and couplets; one network encodes keywords while the other encodes the generated poems or couplet sentences. As described by Wen *et al.* (2020), the model incorporates a self-attention mechanism to effectively learn the dependencies and internal structure of words in sentences. The multi-task model exhibited significantly superior performance compared to its single-task counterpart.

2.2.2 Auto Encoders Architecture

The auto-encoder architecture is a specific type of artificial neural network developed for unsupervised learning, particularly focused on feature extraction and dimensionality reduction. The primary aim of an auto-encoder is to learn a lower-dimensional representation, known as encoding, of the input data while being able to reconstruct the original input from this representation. This process involves two key components: the encoder and the decoder (Pinaya *et al.*, 2020). The encoder compresses the input data into a lower-dimensional space, compelling the network to uncover essential patterns and features inherent in the data. This compressed space, often termed the latent space or bottleneck, encapsulates the core characteristics of the input data in a more condensed form (Hancock *et al.*, 2020). The latent space serves as a more abstract and reduced representation of the original data, retaining only its most significant elements.

In contrast, the decoder reconstructs the original input data from this compressed representation. It remaps the latent space back to the original input space, attempting to recreate the data as closely as possible (Sun *et al*., 2023). In deep auto-encoders, both the encoder and decoder comprise multiple layers, allowing the network to learn complex hierarchical representations of the input data. Such deep architectures enhance the model's ability to capture intricate data patterns and relationships. The architecture of a typical auto-encoder is illustrated in Figure 2.5.

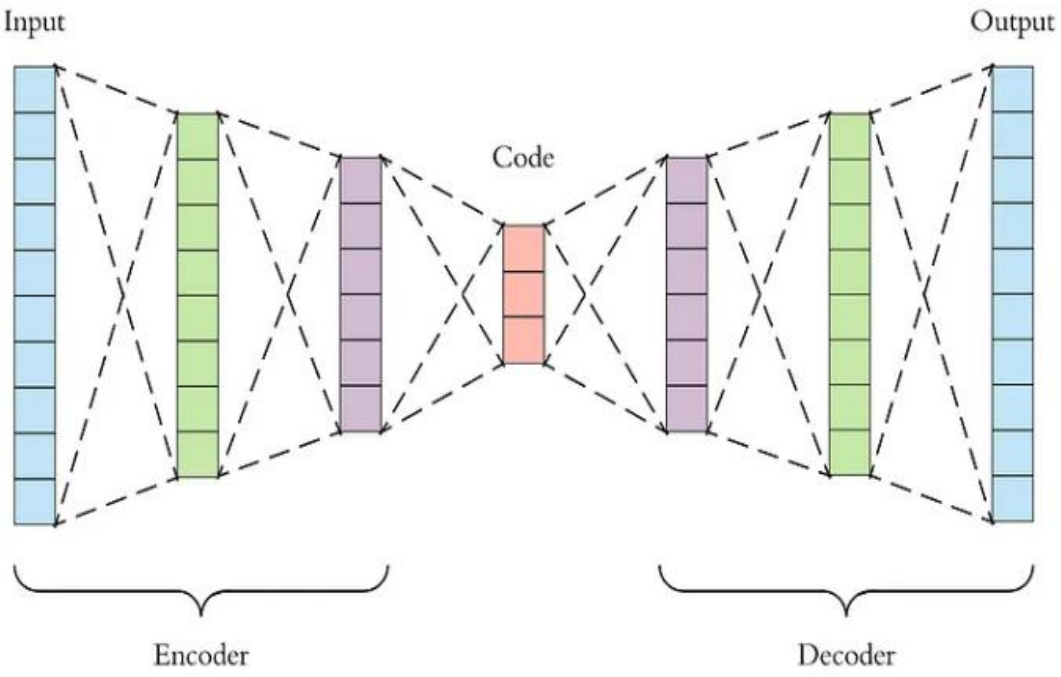


Figure 2.5: Auto-encoder architecture

One model that makes use of the automatic encoder-based recommendation method is the auto-encoder-based collaborative filter (ACF) (Park *et al*., 2023). To tackle the integer score prediction problem, the ACF method separates the user's rating value for the item into five independent vectors. The division of user ratings increased the sparsity of the scoring matrix, which resulted in low scoring forecast accuracy. We present a model to employ deep auto-encoders integrated with feature embedding to address the cold start problem in product-based e-commerce because the discussed models are unable to tackle it.

A deep neural network's primary function is to mine input data properties through several tiers of hidden layers, hence increasing the predictive power of the algorithm. In order to effectively improve the implicit feature information after it has been extracted, these methods either extract the user profile features of the corresponding data based on the traditional recommendation algorithm or extract features from the original data into the deep neural network model after data processing. As a result, the algorithm's forecast accuracy can be raised (Zhang *et al*., 2018).

The auto-encoder (AE), clustering deep neural networks (CDN), restricted Boltzmann machine (RBM), and deep belief networks (DBNs) are the primary components of unsupervised deep learning architecture (Min *et al*., 2018). Deep clustering employs DNNs to learn the input object's raw feature representation; the resulting cluster-friendly representations are then fed into the relevant clustering method (Aljalbout *et al*., 2018). Hazrati *et al*. (2019) attempted to apply deep learning to extract hidden variables from objects and users through a collaborative filtering approach based on restricted Boltzmann machines (RBMs). Later, other researchers made even more advancements, but the RBM still had a number of issues that made it challenging to use in real-world applications, like a significant scale of weight when connecting to hidden layers and a lengthy training period.

The recommendation system used the deep belief network (DBN) approach to create a hybrid model (Chen *et al*., 2019). The click-through rate (CTR) model was developed to learn features from articles and then come up with article recommendations. The model experienced sparsity of auxiliary information and insufficient hidden features for training and learning (Xu *et al*., 2022). In order to forecast ratings, a deep neural network architecture for social relations incorporates each user's nonlinear properties into probability matrix factorization (Fan & Cheng, 2018). Using a random walk layer to create the item-perceived social sequence and harvest pertinent data from far-off neighbors for user-item nonlinear fusion, a deep social collaborative filtering framework is an architecture (Derr *et al*., 2018).

The implicit feedback acquired by SVD++ and the video data characteristics learned by decreasing the auto-encoder are combined in the Auto SVD++ algorithm, which was developed to increase the algorithm model's recommendation accuracy (Yedukondalu *et al*., 2023). The Depth Matrix Decomposition model was developed to deeply mine the corresponding data features by first breaking down the item feature matrix and the user feature matrix using a standard matrix decomposition approach. A multi-layer feed-forward neural network is then utilized to do this. Lastly, the anticipated rating of the algorithm model is obtained by taking the inner product of the matching low-dimensional feature vector (Tegene *et al*., 2023).

The Deep Framework for both Cross-Domain and Cross-System Recommendations (DCDCSR) is a comprehensive framework put forth for cross-domain and cross-system recommendations. The objective was to obtain each user's and item's sparsity rating degree across various systems or domains (Zhu *et al*., 2020). Research has proposed a convolutional neural network and matrix factorization hybrid model for convolutional matrix factorization. The method uses auxiliary data from the object to address sparsity issues and increase the accuracy of rating predictions, and a convolutional neural network is used to extract features from contextual information (Rendle *et al*., 2020).

A strategy to solve the hashtag suggestion problem using convolutional neural networks (CNN) was developed (Bansal *et al*., 2024). The experimental results demonstrated that the suggested method could produce noticeably superior performance than the state-of-the-art methods. The CNN design included a trigger word mechanism with a local attention channel and a global channel. To help users choose photographs, a dual-net deep network model was created. The network was made up of two smaller networks that map an image and a user's preferences into the same latent semantic space. (Kaur *et al*., 2023).

A deep neural network was used to create the dropoutnet model, which was designed to solve cold start issues. In order to use preferences and content information without explicitly relying on their presence, the model uses an input dropout to condition the missing preference data. Although the model produced good generalizations for cases including both warm and cool stars, it lacked diversity (Volkovs *et al*., 2017).

In research to check the impact of side information and feature quantity on the recommendation algorithm, the MovieLens 100k dataset was used. The 100k MovieLens was used since it is standard for running comparison and assessment for recommender algorithms and it has side information. Side information is extra details like item or user characteristics such as location, item description, age, and gender, which are mostly not captured in the ratings. The results indicated that almost all algorithms that made use of side information and quality features achieved higher accuracy than a model that just relied on the user/item rating (Wegmeth, 2022).

To convert a collaborative filtering problem into classical supervised learning, auto-encoder collaborative filtering to supervise learning (ACOFILS) was developed. In order to acquire more complicated and usable representations in the suggested neural network, the authors experimented with a stacking de-noising auto-encoder (SDA) as an alternative to the singular value decomposition (SVD) approach (Barbieri *et al*., 2017). The rich representation that comes with embedding is lost when the A-COFILS approach fails to apply embedding. In the relationship between deep neural networks and the matrix factorization approach, a model was developed with auto-encoders. By predicting missing data, a deep auto-encoder network searches for unknown user ratings (cold start problem) (Pan *et al*., 2020). This technique combines an auto-encoder network that can learn nonlinear data representations with a collaborative filtering strategy. This technique combines an auto-encoder network that can learn non-linear data representation with a collaborative filtering strategy.

The Deep Auto-encoders for Feature Learning with Embedding for Recommendations (DAFERec) model utilizes an auto-encoder that is trained on recommendations to compute the item and user matrix. Next, a deep neural network with an embedding from the user and an item matrix is formed to make the prediction (Rama *et al*., 2021). The model proposed focused on general user/item prediction accuracy, not on the cold start problem. Secondly, the model failed to use side information since it was running on state-of-the-art scores, whereas the other model did not use side information. Side information is crucial in the cold start model since it deals with a user or item with no history.

A factorization-machine and feed-forward neural network fusion model was put out. When working with the matching attribute features, the feed-forward neural network learns the high-order interaction features. Through this interaction, the high-dimensional sparse data features can substantially increase the accuracy of feature representation after the data dimension is reduced (Wu *et al*., 2023). An aspect-based opinion mining model was proposed, which had two portions: the first part was to use deep convolutional neural networks to extract aspects from the data, and the second part was to add the ratings from the aspect to a machine tensor factorization to make

predictions of the overall ratings. The method increases accuracy, but it is time-consuming (Da'u *et al*., 2020).

Recent developments in deep learning, a branch of machine learning that aims to replicate the functioning of the human brain, have prompted fresh attempts to address the cold start problem. In a conference, the International Computer Society Recommender System branch concluded that deep learning architecture is the only way to address this problem (Zhang *et al*., 2018). This highlights how important it is to use deep learning to study recommender systems. Numerous research papers and architectures propose the use of deep neural networks as a potential solution to address the recommender systems' cold start issue. (Vijayakumar & Deepak, 2022).

By effectively learning the non-linear user-item relationship and encoding complex abstractions into data representations, an auto-encoder (AE)-based recommender system (RS) can better understand users and items. Auto-encoders are thought to be a potent tool for automatically extracting nonlinear features (Neto *et al*., 2019). Additionally, they can cope with noise and deal with data sparsity by picking up valuable knowledge from a variety of data sources, including textual, contextual, and visual information (Haghighi *et al*., 2019). An auto-encoder stands out by understanding items' characteristics and users', hence achieving higher recommendation accuracy than traditional recommendation systems (Lee *et al*., 2017).

2.3 Deep Neural Network Models Evaluation

Model evaluation in deep learning is crucial for assessing the effectiveness and performance of a model and providing insights into its ability to generalize to new data compared to other existing models. This evaluation process includes examining how well the model handles the dataset and its accuracy relative to similar models. Research has shown that incorporating side information and feature quality significantly improves accuracy. For instance, the MovieLens 100k dataset, a standard for evaluating recommendation algorithms, revealed that models using side information and quality features achieved higher accuracy compared to those relying solely on user/item ratings (Wegmeth, 2022).

Metrics for evaluating deep learning models vary depending on the specific goals or tasks being assessed. Root Mean Squared Error (RMSE) is a commonly used metric that indicates the average squared error if a different, less complex predictor were employed. It is calculated by taking the square root of the average of the squared differences between the predicted and actual values, thereby adjusting the error to the data's scale (Talsma *et al*., 2032). RMSE measures overall error by examining the differences between predicted and actual values, squaring, averaging, and then taking the square root. The mean absolute error (MAE) assesses a regression model's performance by calculating the average absolute difference between predicted and actual values. The equation below is the formulae for MAE

$$\mathbf{MAE = (1/n)\ \Sigma(i=1\ to\ n)\ |y\_i - \hat{y}\_i|}$$

Equation 2.1: Mean Absolute Error (MAE)

Where: $\hat{y}_1$ = Predicted value for the i^{th} data point, y_1 = Actual value for the i^{th} data point and n = number of observations. MAE offers a straightforward measure of accuracy and is less affected by outliers compared to Mean Squared Error (MSE) and RMSE (Sapnken *et al*., 2023). However, MAE does not emphasize significant errors as much as MSE and RMSE.

MSE, widely used for assessing regression models, calculates the average squared difference between predicted and actual values. It provides a quantifiable metric reflecting the precision and accuracy of predictions, with a smaller MSE indicating better model performance. The Equation below shows the formulae for the MSE

$$\mathbf{MSE = (1/n) * \Sigma\ (Y_i - \hat{Y}_i)^2}$$

Equation 2.2: Mean Squared Error (MSE)

Where: n is the number of observations, Y_i is the actual value, $\hat{Y}_i$ is the predicted value and Σ represents the sum from i=1 to n. However, MSE is sensitive to outliers, and large errors can disproportionately impact the final score (Namasudra *et al*., 2023). Accuracy measures the proportion of correct predictions out of all predictions, commonly used

for classification models. Recall evaluates the percentage of actual positive instances correctly predicted (Recall = TP/TP+FN), while precision assesses the proportion of actual positive predictions that are correct (TP/TP+FP). Data splitting involves randomly selecting and dividing data into training, validation, and testing sets. After evaluating various metrics, a comparison with baseline models helps rank and assess the performance of developed models (Michaud *et al*., 2023).

In related research, the University of KwaZulu Natal in Westville, South Africa, explored using social networks and matrix factorization to enhance deep learning techniques for addressing the cold start problem (Wang *et al*., 2018). The study utilized social information to create user groups based on shared interests within a community. A community detection method was employed for this purpose, and the deep learning model was trained using the group data. The evaluation of the compared models was conducted using mean squared error and mean absolute error as metrics, with five-fold cross-validation applied (Chang *et al*., 2023). The findings indicated that leveraging social information improved the outcomes of the deep learning approach and demonstrated the benefits of user community segmentation.

When implementing novel algorithms or methods for personalized recommender systems, understanding how to evaluate the effectiveness of these frameworks is essential (MU, 2018). Common analysis approaches include predictive accuracy metrics, precision, recall, f-measure, and mean reciprocal rank (MRR). These metrics assess the performance of the proposed framework by comparing projected and actual ratings of items, thereby providing insights into the accuracy and effectiveness of the recommendation system (Zhang *et al*., 2022).

2.4 Prototype Development

Model development involves building a working model through data preparation and loading, training, and testing of the model. Model prototype development is a critical phase in the machine learning and data science workflow. It involves building an initial version of your predictive model to test its feasibility, performance, and potential to address the problem at hand. Here is a systematic guide for developing a model prototype. Problem definition tries to explain the problem you are trying to solve and

establish the objectives of your model. Understand the context, stakeholders, and constraints.

Data collection and exploration is the process of acquiring relevant data, which is most important for the training and testing of your model. Also, the data must be investigated and quantified to comprehend its measures, find some patterns, and evaluate the quality of the data. Handle the missing values of the dataset and remove the outliers of the dataset appropriately. Data pre-processing is the step that involves handling missing values, converting categorical variables to numerical variables, and scaling the numerical variables to correct them for modelling. The data is then divided into training and testing sets to aid in evaluating the model.

Model selection comes after data collection, which is done depending on the kind of task involved (classification, regression, clustering, etc.) and the properties of the data. Thus, select the proper machine-learning algorithm. As for the final models, it is also advisable to start by identifying simple models for prototyping and evaluating their performance. Feature engineering explores possible features that may improve the performance of the model by developing features that are useful for the model. Decide on which of the techniques is most appropriate for feature engineering based on knowledge gained in the data exploration phase, and then entitle the prototype model by implementing the chosen model in a programming language. (e.g., Python with libraries like Scikit-learn, TensorFlow, or PyTorch). Once the creation of the prototype is complete, train the model using the training set and assess its output using the testing set.

The machine learning model is perfect and does not need to be improved, in fact, model iteration and optimization can significantly enhance the accuracy and stability of the model. This phase involves fine-tuning the value of various attributes that are in the model, trying out different structures of the model, and then trying to get the best in through optimizers. Here are some of the items to consider

First hyper-parameters are those settings that do not come from data through training but can directly affect the model's performance. Some examples of hyper-parameters include; the rate of learning which measures and controls updates of the model in

response to errors. Number of layers in the model, which determines the depth and the complexity of the model then we have a batch size which determines the number of the training samples to be used in every iteration (Liao *et al.,* 2022).

Regularization helps the model from overfitting by discouraging the model from depending only on the training dataset but even working with unseen data. Some of the regulator's techniques include; L1 Regularization, which adds a penalty, equal to the absolute value of the magnitude of coefficients. L2 Regularization, which adds a penalty equal to the square of the magnitude of coefficients.

Dropout, which randomly drops some neurons during training to prevent co-adaptation, and early Stopping technique, which stops the training, process when the model's performance on a validation set starts deteriorating (Moradi *et al.,* 2020).The DropoutNet recommender architecture passes user preferences and content via the appropriate deep neural networks as inputs. The fine-tuning network receives the concatenated top-layer activations and uses them to produce the latent representation (Liu *et al*., 2021). After working together to optimize via back-propagation, every component is fixed for the inference. Even using the model, there are certain drawbacks to attaining highly competitive performance, such as complexity from utilizing both preferences and content components. Second, the model's formulation does not apply to cold-start consumers; rather, it presupposes cold-start items. (Volkovs *et al*., 2017).

The proposed DeCS model tries to use a dataset and side information (SI) to find a solution for the cold start problem. The model development involves feature extraction and prediction steps. Under prediction, the vectors from user-item matrices and side information are merged at the fully connected dense layer (Mondal *et al*., 2022). There was a model designed to use stacked and reconstructed graph convolutional networks to come up with a prototype prediction item faced with the cold start problem. The model used STAR-GCN to produce node embedding for new nodes by constructing masked input node embedding. During implementation, they experienced leakage issues during training, leading to overfitting problems and significantly degrading the final performance (Zhang *et al*., 2019).

In machine learning, meta-learning techniques are becoming more and more popular as a means of learning representations that are helpful for a variety of tasks (Tian *et al*., 2022). A trained framework that is more efficient for a variety of users was used to create meta-learning for the user-cold-start recommendation system. It was motivated by model-agnostic meta-learning's ability to model broadly. A few gradient steps were used to change the model parameters during testing in order to tailor it to a particular user. The model was then assessed using three distinct benchmark datasets (Bharadhwaj, 2019).

2.5 Developed Model Architecture

Recent advancements in deep neural networks have significantly enhanced the effectiveness of recommender systems (Li *et al*., 2021). These deep learning technologies integrate traditional approaches, such as content-based and collaborative algorithms, with advanced architectures like recurrent neural networks (RNNs), convolutional neural networks (CNNs), multilayer perceptrons (MLPs), and deep auto-encoders. Among these, deep auto-encoders have emerged as a powerful tool for filling in missing values, identifying patterns, and addressing issues related to data sparsity. By incorporating feature embedding, deep auto-encoders facilitate the formation of distinct clusters and enhance the model's ability to reconstruct data with high accuracy, whether the data is labeled or unlabeled (Mastroleo *et al*., 2018).

Auto-encoder-based models offer advantages over traditional models that typically handle a single data source, such as text or ratings. These models can process heterogeneous data sources, including text, audio, visual, and video, due to their superior ability to understand user demands and item features. This capability results in higher recommendation accuracy and greater flexibility in multi-media contexts (Deldjoo *et al*., 2018). Additionally, deep auto-encoders are adept at managing input noise and accommodating changes in user profiles. By capturing evolving user tastes and preferences, these models enhance item prediction for new users (Zhang *et al*., 2022).

Embeddings are pivotal in this process, as they transform high-dimensional vectors into lower-dimensional vectors, effectively reducing data complexity (Wang *et al*., 2021).

This process involves extracting relevant features from the dataset and forming rating matrices. For user profile generation, embeddings utilize side information such as time and location, aiding in the prediction of user profiles. When predicting items for a new user, one-hot encoding is employed to convert received data into a binary format, which is then used to adjust the feature embedding.

The feature embedding architecture consists of several layers. Initially, input data points, such as time and location, are collected. In the second layer, user IDs are transformed into vectors, which are then concatenated. These vectors are merged with low-dimensional data, resulting in a more detailed dataset. The dense layer in the decoder part of the architecture processes this enhanced dataset. Figure 2.6 illustrates the feature embedding architecture used in this model.

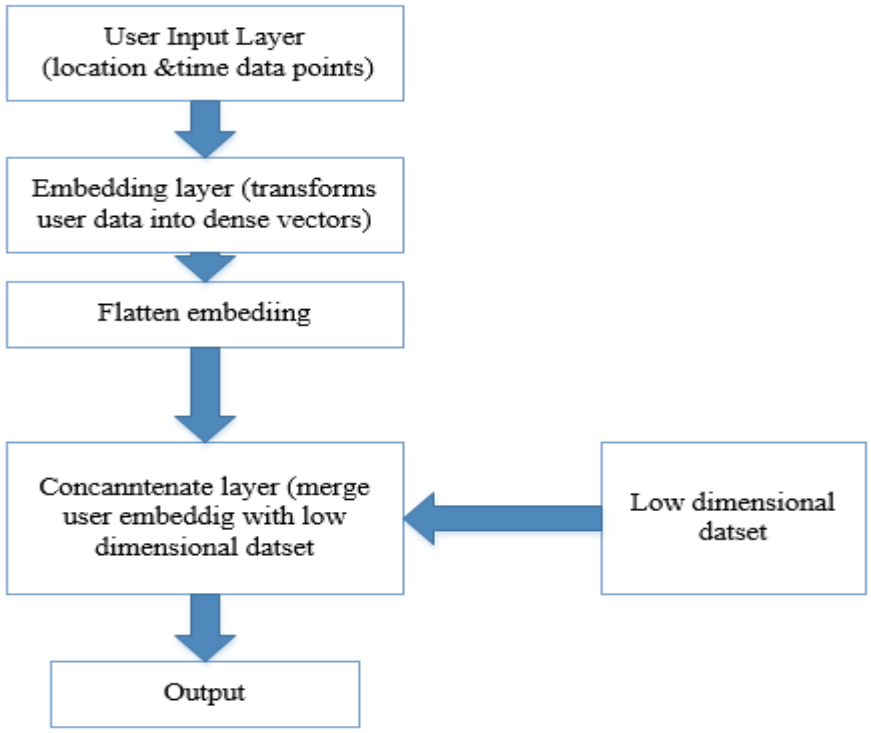


Figure 2.6: Rectified Linear Unit

The rectified linear unit (ReLu) will be used as an activation function in the hidden layers of a deep-out encoder network; the sigmoid function will be used for rating prediction on the output layers; and the root mean squared error (RMSE) will be utilized

to minimize the error in the loss function. Figure 2.7 shows a deep auto-encoder diagram.

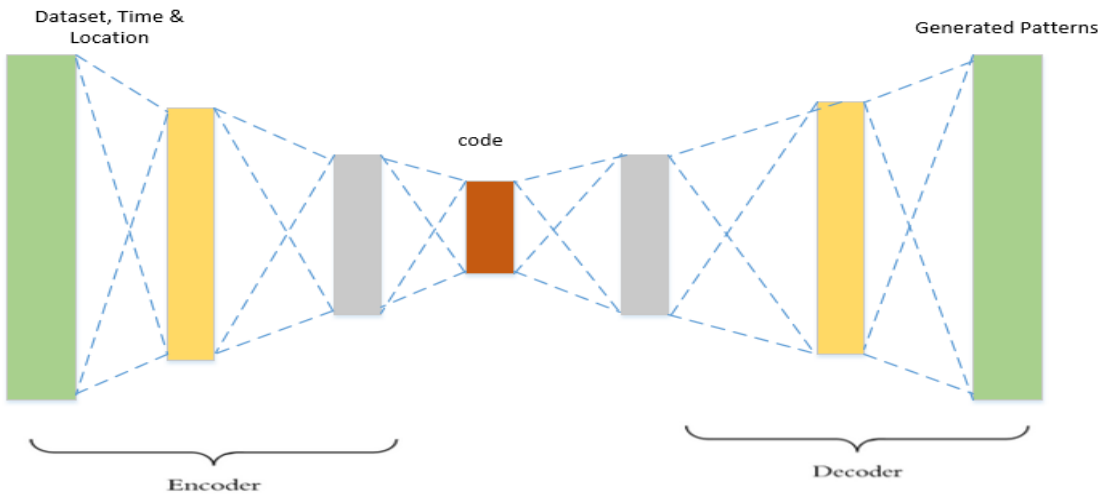


Figure 2.7: Deep-auto encoder

On user-item matrix generation, the system will use a deep auto-encoder since it makes use of unsupervised learning techniques to transform its input into accurate patterns with ease (Kamilaris *et al.*, 2018). The user-item similarity calculated will be based on the user's explicit and implicit ratings from the dataset. Table 2.1 shows a sample rating from different users and items.

Table 2.1: Item rating matrix

items	*Item*1	*Item*2	*Item*3	*1tem*4	*Item*5
*User*1	R_{10}	R_{12}	?	R_{16}	R_{18}
*User*2	R_{25}	?	r_{30}	r_{18}	R_{16}
*User*3	r_{30}	R_{34}	?	R_{24}	R_{45}
*User*4	?	R_{38}	r_{43}	R_{54}	R_{56}
*User*5	R_{65}	R_{50}	R_{65}	?	?

The degree of similarity between two items and the ratings users assign to individual items from the cases are based on how similar their rating patterns are. The formula for the similarity metric employed in this case is Root Mean Squared Error (RMSE).

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(y(i) - \mathrm{swx}\,\hat{y}(i))^2}{N}}$$

Equation 2.3: Root Mean Squared Error (RMSE)

Where N stands for a number of non-missing data points, y(i) represents the variable, y ̑(i) shows the corresponding prediction rating, and y (i) is the actual rating. Being an online platform where data input is done in real-time, the model needs continuous input of the user's explicit and implicit ratings to keep updating the model in real time. On the other side is a new user who gets to a system and requires personalized item predictions; this depends on the user profile generation. Where the user profile is generated depends on things like item ratings, other similar user ratings, time, age, and geographical location.

The developed model has two parts, where the first part is the creation of user profiles as they get into the system. The profile generation is done automatically by picking the user's history and the real-time and location data points. The second part entails the prediction process, where the model picks the item with the highest weight on the cluster where the user belongs. Figure 2.8 shows a workflow diagram for the developed model.

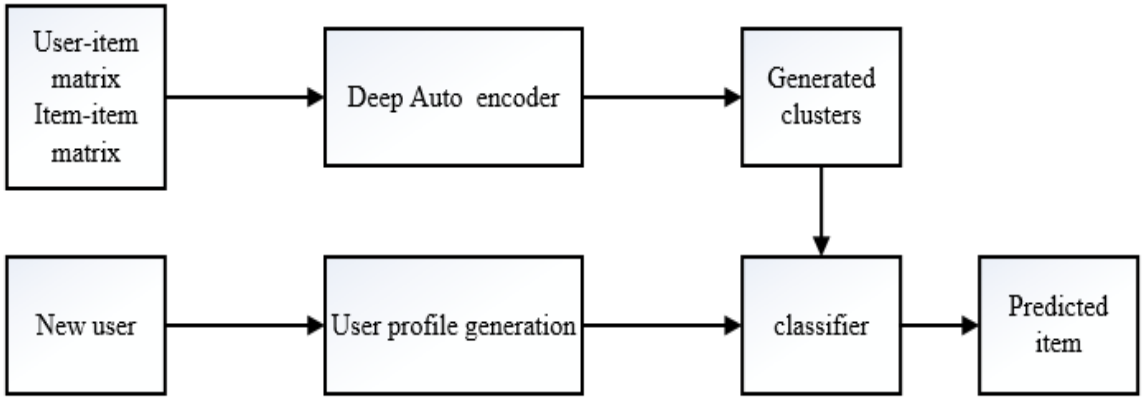


Figure 2.8: Workflow diagram

CHAPTER THREE
RESEARCH AND METHODS

3.1 Study Site

This study was based on an e-commerce platform that developed a model using a deep auto-encoder incorporated with feature embedding to provide personalized recommendations to new users (visitor cold start). The study was conducted at Chuka University Computer Lab. The coordinates are as follows: Latitude: -0.319454 / S 0° 19' 10.036" Longitude: 37.657197, E 37° 39' 25.908".

3.2 Study Design

This study employs an experimental research design, adhering to specific scientific research questions while incorporating objectives with both independent and dependent variables that are manipulated, calculated, and measured. In this context, the independent variables include the logging time and location data points of users, which are captured and integrated into user profiles to enhance the accuracy of model predictions. The primary data for the model consists of the generated patterns. The study involves a comparison of model prediction accuracy between two scenarios: one where the model relies solely on datasets and another where the feature-embedding layer, incorporating time and location data points, is integrated into the deep neural network. This approach aims to assess the impact of incorporating additional user data on the overall performance of the predictive model.

3.3 Dataset

The MovieLens 100k dataset was utilized for training and evaluating the model. This dataset, consisting of 100,000 ratings for 1,682 movies, is well-suited for testing recommender algorithms due to its inclusion of side information and its established use in evaluations (Wegmeth , 2022). Each user in the dataset has rated at least 20 films on a scale from 1 to 5. Additionally, the dataset provides demographic details such as gender, zip code, occupation, and age, as well as movie attributes like title, genre, and release date. The dataset was refined by removing users with fewer than 20 ratings or missing demographic data. While the MovieLens 100k dataset is substantial enough to support meaningful experiments, its size remains manageable, allowing researchers and developers to prototype recommender systems without the need for extensive

computational resources. Managed by the Grouplens research group at the University of Minnesota, MovieLens serves as a valuable resource for evaluating recommendation systems.

3.3.1 Data Loading and Preparation

In this thesis, the 100k MovieLens dataset was employed to train and evaluate the model. This dataset is a well-established benchmark in the field of recommender systems, derived from the MovieLens platform, which records user interactions with movies. It includes 100,000 user ratings on a scale from 1 to 5 and provides additional data such as user demographics and movie genres. Each record in the dataset features user ID, movie ID, rating, timestamp, and genre information. Prior to model training, the dataset underwent preprocessing to eliminate duplicate entries, address missing values, and ensure data consistency. This preprocessing step was crucial to maintaining the integrity of the dataset and supporting accurate model evaluation.

3.3.1.1 Data Loading

We began by loading all the required packages for model development and data preparation. The dataset was imported using pandas Data Frames, with each column clearly defined to facilitate data manipulation and analysis. Subsequently, the ratings were normalized to a range between 0 and 1. This normalization step ensures that the data is scaled appropriately for model training.

3.3.1.2 Data Preprocessing

Preprocessing involves handling missing values, removing noise, and ensuring data consistency. On we had to merge the ratings with the userId so as to capture the core data for the model. Merging ratings and user data: the purpose of merging the rating data and user data on the userId column is to combine information about users' interactions with movies (ratings) with demographic information about the users. This enriched dataset, provides a more comprehensive view of the users and their preferences, which is particularly useful for addressing the cold start problem in recommendation systems.

Data splitting is a critical step in preparing a dataset for model training, testing, and validation. This process involves randomly dividing the data into separate subsets to

ensure that the model can be effectively trained and evaluated. For recommender systems, particularly when addressing the cold start problem, where there is limited or no historical interaction data for some users or items, understanding the dataset's characteristics is essential.

3.3.1.3 Creation of User-item Matrix

User-item matrx generation also known as the utility matrix or ratings matrix is a fundamental structure in recommendation systems. It captures the relationship between users and items (e.g., movies, products) by representing the ratings or interactions between them. Additionally, it is important to extract relevant features for both users and items, such as demographic information, metadata, or contextual details, to enhance the model's ability to address the cold start problem effectively.

3.4 Model Architecture

The developed model consists of two primary components: feature extraction and prediction. In the first part, feature extraction, the dataset is processed through a deep auto-encoder network to handle the ratings, which are integers ranging from 1 to 5. For movies that have not yet been rated by a user, the rating is set to 0. The second part of the model focuses on prediction, utilizing the generated patterns, side information, and feature embeddings, including time and location data points of the user. These elements are processed through the innermost layers of the auto-encoder network. Figure 3.1 illustrates the architecture of the developed model.

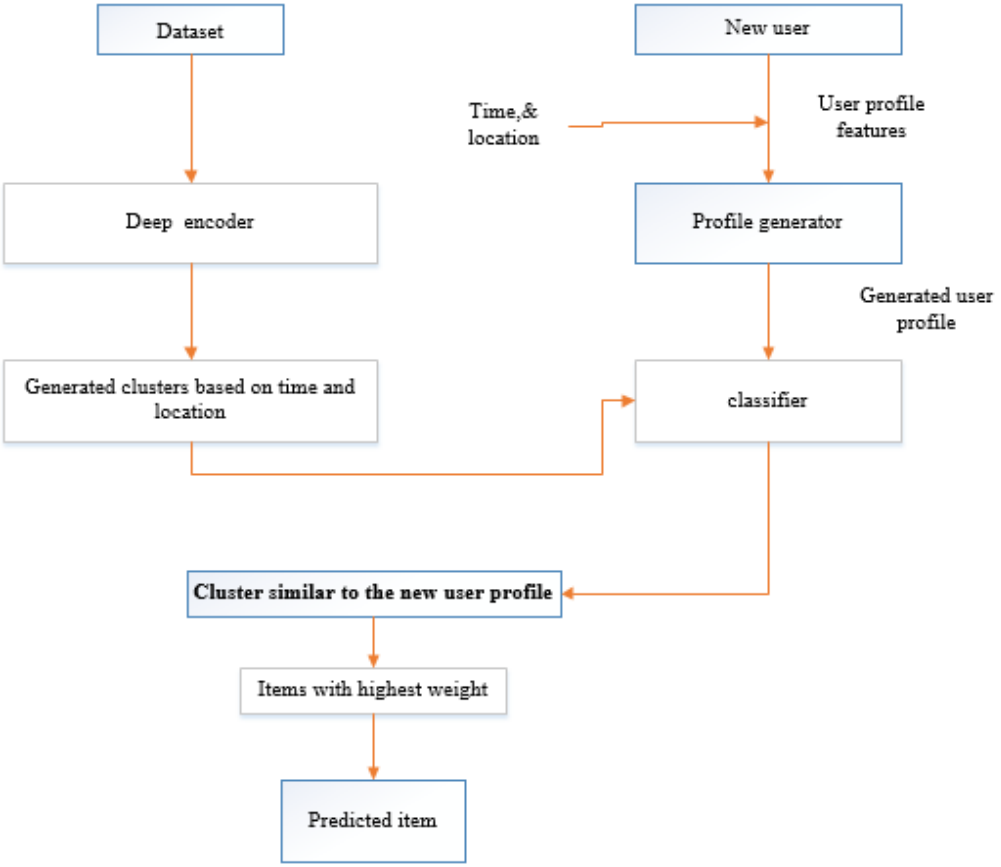


Figure 3.1: System architecture diagram

Steps to be followed for the model development include,

1. Creation of user-item matrix
2. Cluster items based on item similarity
3. Cluster users based on similarity also known as user profiles.
4. item prediction

3.4.1 Item matrix (Similarity) Calculation

The item-item matrix is a method used to identify items similar to those a user has previously interacted with positively. When a large number of users show similar interactions with two items, these items are considered comparable. An auto-encoder, on the other hand, aims to classify and interpret items by modeling them in a way that aligns with human understanding. It focuses on the research object, or the specific aspect under investigation, to establish meaningful connections between relevant items and users.

3.4.2 User-item Matrix Calculation

The core principle of determining the similarity between two items and a user involves isolating the user who evaluated each item and applying a consistent calculation method to measure similarity. In this study, a deep auto-encoder is used to compute these similarities, generating patterns that aid in prediction. The model leverages a user history context module, which inputs data from the user's activity history. The primary aim of this phase is to generate a set of items that are likely relevant to the user's interests and queries, based on item IDs and historical user behavior. This approach helps in recommending items by analyzing user preferences and addressing the challenge of information overload.

3.4.3 User Profile Generation

User profiling is the process of identifying and extracting keyword-based information to create structured user profiles. The developed module generates user profiles using data points such as the time of logging and location, even for users with no prior information. Once these profiles are established, they aid in visualizing user preferences, thereby enhancing the recommender system's ability to predict personalized items for new users. Although user profiling presents challenges, the integration of deep learning architectures and feature embedding can lead to more accurate recommendation models.

The model incorporates demographic data for new users within the training dataset. Prediction is achieved by integrating feature embeddings and user activations into a deep neural network. For users with more side information, their data exerts a greater influence on the final recommendations. To prevent overfitting and enhance regularization, a dropout layer is included in the deep auto-encoder architecture. Dropout, which involves temporarily removing units and their connections from the network, helps generalize the model and combines multiple neural network architectures effectively (Sabiri *et al*., 2022). Initially, the input data is reduced to its feature vector due to the properties of the auto-encoder, and this feature vector is used to reconstruct the data.

3.4.4 Rating Prediction

When a new user joins the system, a user profile is generated using available features such as preferences, user ratings, age, gender, and profession. For a cold-start user with no prior interaction history, the profile is created using default features like location and time. Once the user profile is generated, it is passed to a classifier, which determines the cluster to which the profile belongs. With the user's cluster identified, deep neural networks can predict items for the user based on the highest-weighted items within the assigned cluster. The system anticipates ratings by comparing the active user's ratings for items similar to those of the target user, thus providing personalized recommendations even for new users.

3.5 Model Prototype

Model prototyping allows the data scientists and engineers get a feel of the differences between various algorithms, architectures, or techniques before going to work on actual model development. Hence, the idea is to get some first impression about how the model works and what are the costs associated with the different configurations such that these insights can then be used for making the final decision on the model. This process involved selecting appropriate ranges for hyper-parameters such as batch size, network layers, dropout rate, epochs, loss function, activation function, and learning rate. Specifically, the Rectified Linear Unit (ReLU) activation function was used in the innermost layer, while the Adam optimizer minimized loss function errors. A linear sigmoid activation function was applied to the output layer to generate real numbers, which are converted into user ratings for prediction. The model architecture includes six layers: the input layer, a layer with dropout and user embedding, a latent space with dense and flattened dense layers, and a final user-prediction layer. The user-rates layer handles dataset inputs, and the dropout layer provides regularization to prevent overfitting.

In the encoding phase, dense layers utilize the ReLU activation function to address the vanishing gradient problem. ReLU is favored for its computational efficiency and its ability to introduce sparsity into the network by converting negative inputs to zero, which enhances network efficiency. Dropout is employed to prevent overfitting through regularization, promoting generalization by randomly deactivating neurons during

training, thereby making the model more adaptable to new data. In the decoding phase, a sigmoid activation function is applied to the final dense layer to ensure that outputs are constrained within the [0, 1] range. This choice provides a smooth gradient that aids in the optimization process and ensures that reconstructed outputs align with the normalized input data. The dense and flattening layers work to compress outputs from the previous layer into a latent space, with each layer consisting of input and output sublayers. The embedding layer creates user embeddings by integrating user logging times, locations, and ratings. Figure 3.2 illustrates the architecture of the deep auto-encoder network with four layers.

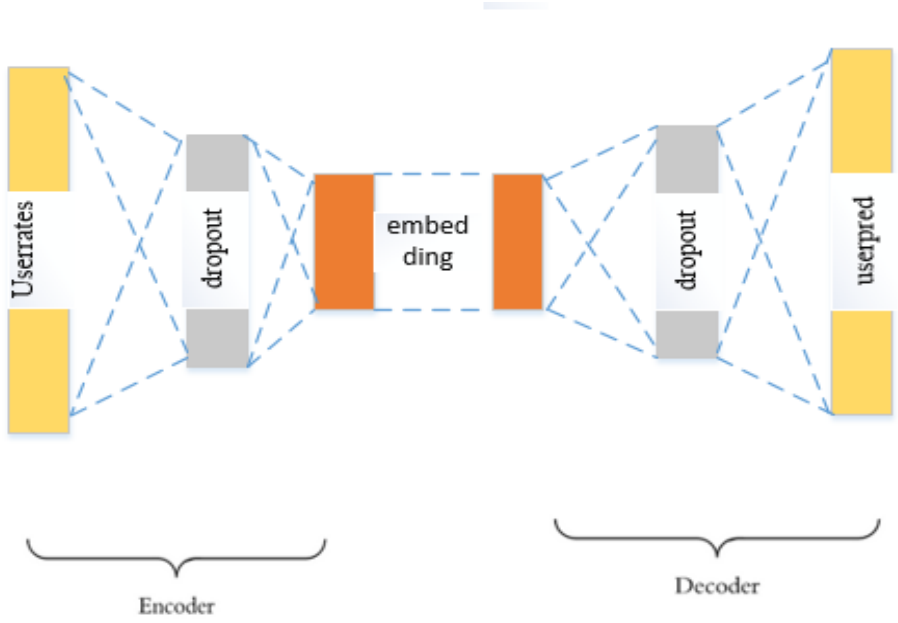


Figure 3.2: Deep encoder model with four layers

Feature-embedding layers are incorporated to capture additional information beyond user-item interactions. These layers transform categorical features into dense, lower-dimensional representations, which are learned jointly with the auto-encoder. Techniques such as entity embedding or word embedding are used to convert categorical features into continuous vectors. The deep auto-encoder and feature embedding layers are combined into a unified architecture, enabling the joint learning of user-item interactions and additional features. Latent representations obtained from the auto-encoder are concatenated or merged with feature embeddings before making predictions.

The model was designed to be flexible, adjusting the weightings of the auto-encoder and feature embedding components to handle various cold start scenarios effectively. The model enhances prediction accuracy and reduces prediction time by effectively addressing missing values for new users. The integration of an embedding layer captures real-time and location data points, eliminating the need to regenerate values for users with zero ratings and thereby streamlining both training and prediction phases. Post-development, the model underwent optimization through hyper-parameter tuning to refine item prediction accuracy.

3.6 Data Analysis and Evaluation Metrics

The MovieLens 100k dataset was utilized for training and evaluating the model, with the data split into training, testing, and validation sets in proportions of 60%, 20%, and 20%, respectively. Ideally, data splitting involves shuffling the data randomly to equip the data points with less bias and more randomness. In this study, we used a Simple Random Split where the random number was assigned to each data point and then sort data based on the assigned random numbers. After sorting, the data now select the first 60% for training followed by 20% for validation, and the final 20% for testing. This is advantageous because it makes sure that the train and test datasets are chosen randomly from the rest of the datasets. This distribution ensures a balanced approach between training and evaluation, maintaining an unbiased assessment of the model's performance (Paullada *et al*., 2021).

The model's effectiveness was evaluated using several metrics, including Mean Absolute Error (MAE), which measures the average absolute distance between predicted and actual values. Mean Squared Error (MSE), which squares the errors to account for larger discrepancies, and Root Mean Square Error (RMSE), which indicates the model's overall accuracy by measuring the square root of the average squared differences between predicted and actual values. These evaluation metrics were further tested with additional datasets, including MovieLens 1M and Film Trust, to validate the model's accuracy and generalizability.

3.7 Hardware/Software Requirements and Specifications.

The study used a GPU-enhanced laptop for prototype development, training, and evaluation. The following programs were installed to support model development: an operating system, Microsoft Office, Python 3 programming language, Google Colab, Neural Designer software, and Keras TensorFlow. Python, known for its support of third-party libraries, dynamic data types, and object-oriented programming, is a key tool in this process (Ahmed *et al.*, 2021). TensorFlow, an open-source artificial intelligence learning framework developed by Google, is used for implementing neural networks.

3.8 Ethical Consideration

The study clearance was obtained from Chuka University Ethics Review Committee then research permit form National Committee of Science Technology and Innovation (NACOSTI).

CHAPTER FOUR
RESULTS AND DISCUSSION

4.1 Preliminary Analysis

In this chapter, we explore the practical implementation of a hybrid model that integrates deep auto-encoder and feature embedding (DEA-FE) techniques to improve an e-commerce recommender system, particularly addressing the cold start problem. We start by describing the dataset employed for training and evaluation. Following this, we detail the architecture and training procedure of the hybrid model. Next, we present the experimental results and assess the effectiveness of the developed model in comparison to baseline models. The model was trained using deep auto-encoders with an added embedding layer to incorporate time and location data points. The training utilized the MovieLens 100k dataset, and performance results were documented for analysis. Figure 4.1 illustrates the model's performance during the training phase.

```
Epoch 1/100
10/10 [==============================] - 2s 104ms/step - loss: 0.2047 - val_loss: 0.0915
Epoch 2/100
10/10 [==============================] - 1s 83ms/step - loss: 0.0627 - val_loss: 0.0295
Epoch 3/100
10/10 [==============================] - 1s 108ms/step - loss: 0.0352 - val_loss: 0.0295
Epoch 4/100
10/10 [==============================] - 1s 121ms/step - loss: 0.0349 - val_loss: 0.0293
Epoch 5/100
10/10 [==============================] - 1s 121ms/step - loss: 0.0346 - val_loss: 0.0289
Epoch 6/100
10/10 [==============================] - 1s 105ms/step - loss: 0.0343 - val_loss: 0.0284
Epoch 7/100
10/10 [==============================] - 1s 77ms/step - loss: 0.0337 - val_loss: 0.0277
Epoch 8/100
10/10 [==============================] - 1s 72ms/step - loss: 0.0326 - val_loss: 0.0266
Epoch 9/100
10/10 [==============================] - 1s 74ms/step - loss: 0.0308 - val_loss: 0.0248
Epoch 10/100
10/10 [==============================] - 1s 77ms/step - loss: 0.0292 - val_loss: 0.0239
```

Figure 4.1: Model-training process

After the process of model training a summary of the model was extracted which is shown in Figure 4.2.

```
autoencoder.summary()
```

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 timestamp_input (InputLaye  [(None, 1)]               0         []
 r)

 location_input (InputLayer  [(None, 1)]               0         []
 )

 embedding (Embedding)       (None, 1, 10)             240       ['timestamp_input[0][0]']

 embedding_1 (Embedding)     (None, 1, 10)             7960      ['location_input[0][0]']

 user_input (InputLayer)     [(None, 1682)]            0         []

 flatten (Flatten)           (None, 10)                0         ['embedding[0][0]']

 flatten_1 (Flatten)         (None, 10)                0         ['embedding_1[0][0]']

 flatten_2 (Flatten)         (None, 1682)              0         ['user_input[0][0]']

 concatenate (Concatenate)   (None, 1702)              0         ['flatten[0][0]',
                                                                  'flatten_1[0][0]',
                                                                  'flatten_2[0][0]']

 dense (Dense)               (None, 64)                108992    ['concatenate[0][0]']

 dropout (Dropout)           (None, 64)                0         ['dense[0][0]']

 dense_1 (Dense)             (None, 32)                2080      ['dropout[0][0]']

 dense_2 (Dense)             (None, 64)                2112      ['dense_1[0][0]']

 dropout_1 (Dropout)         (None, 64)                0         ['dense_2[0][0]']

 dense_3 (Dense)             (None, 1682)              109330    ['dropout_1[0][0]']

=================================================================================================
Total params: 230714 (901.23 KB)
Trainable params: 230714 (901.23 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.2: Model summary

4.2 Hybrid Model (DEA-FE) for Cold Start Problem Results Analysis

Developing a hybrid model to address the cold start problem by combining a deep auto-encoder with a feature embedding architecture involves integrating techniques that can effectively handle both sparse input data and incorporate additional features for users and items. Here are the results achieved after the development.

4.2.1 Training Experimentation Results

Training experimentation results refer to the outcomes and metrics obtained during the process of training and evaluating different configurations or variations of a machine-

learning model. On the developed model, we will be checking on overall training loss and accuracy of the model which are recorded below.

4.2.1.1: Developed Model (DAE-FE) Training loss

The model training loss, also known as the training error or training cost, is a measure of how well a machine-learning model is able to predict or fit the training data during the model training process. Figure 4.3 displays the training and validation loss curves for the developed model during the training process, the initial rapid decrease in loss indicates that the model is capable of capturing the patterns in the data, and the convergence of the curves shows that the model is not overfitting. Overall, the training and validation loss curves suggest that the model is learning effectively from the training data and generalizing reasonably well to the validation data.

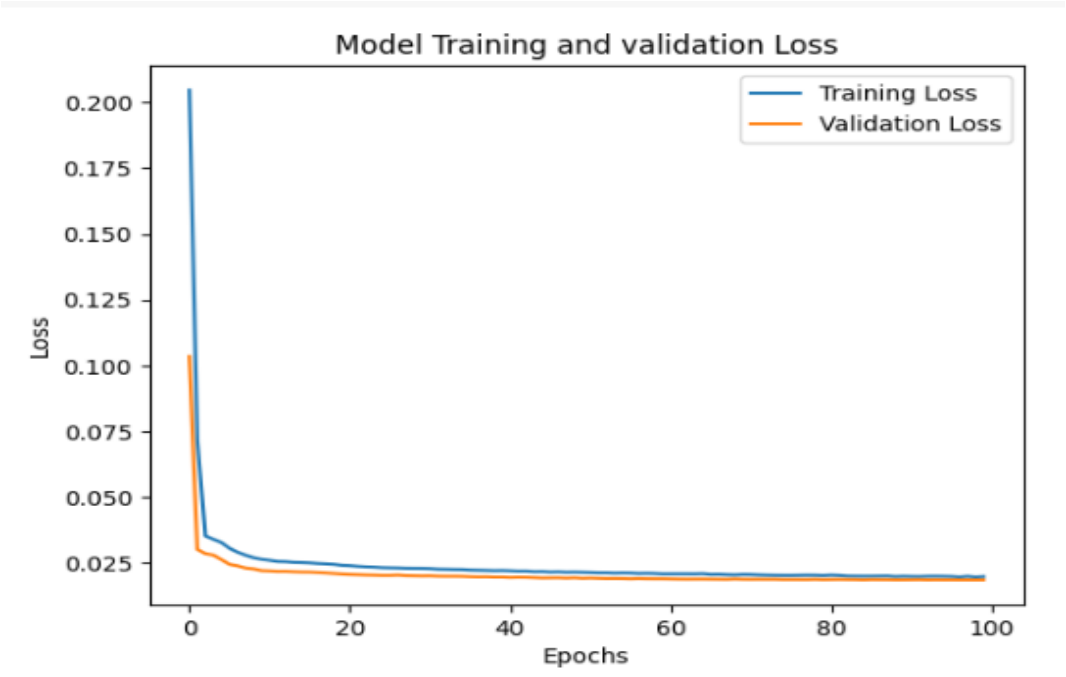


Figure 4.3: DEA-FE model training and validation loss

4.2.1.2 DAE-FE Model training and validation accuracy

Model training and validation accuracy are crucial metrics for assessing a machine learning model's performance during different phases of training. Training accuracy measures how well the model performs on the data it was trained on, indicating the percentage of correctly predicted or classified training examples. In contrast, validation

accuracy assesses the model's performance on a separate dataset not used during training. It provides insight into the model's ability to generalize to new, unseen data. Figure 4.4 illustrates the training and validation accuracy curves over the course of the training process. Both curves show an upward trend as the number of epochs increases, reflecting that the model's performance improves on both the training and validation datasets. This desirable pattern suggests that the model is effectively learning from the training data and is also generalizing well to the validation data, indicating successful model training and evaluation.

Figure 4.4: DEA-FE model accuracy

After the initial phase of rapid improvement, both the training and validation accuracy curves begin to converge, and the rate of accuracy increase slows down. This behavior is anticipated as the model's learning process becomes more gradual over time. Initially, the model quickly adapts and learns from the data, but as training progresses, improvements become more incremental. The convergence of the curves indicates that the model has reached a point where its performance stabilizes, suggesting effective learning and optimization. The observed pattern of initial rapid growth followed by gradual convergence is a desirable outcome, reflecting that the model is successfully enhancing its performance on both training and validation datasets.

4.3 DAE-FE Model Evaluation Metrics Rsults

Model evaluation assesses the quality and suitability of a predictive model by examining its performance in making accurate predictions. This process involves testing the model using specific metrics and methods to determine how well it generalizes to new, unseen data, with the goal of minimizing prediction errors. The deep auto-encoder with feature embedding model was evaluated through various approaches. Initially, the evaluation involved experimenting with popular optimizers, activation functions, and loss functions to identify the most effective combinations. Results from these experiments were carefully recorded and analyzed to determine their impact on model performance.

4.3.1 Activation Functions

Activation functions are common in neural networks, which are applied to introduce non-linearity into the model. This non-linearity is essential because it permits the network to recognize the highly intricate data structures. The following functions were used to analyze our model:

4.3.1.1 Rectified Linear Unit (R.EL.U) Activation Function

Activation functions are essential in neural networks as they introduce non-linearity, which is vital for the network to learn complex patterns and relationships within the data. Their role is fundamental in modern deep learning due to their simplicity, effectiveness, and computational efficiency. Activation functions help facilitate gradient flow during training, which improves the overall performance of neural network models. Figure 4.5 illustrates the results recorded from the model during training. The figure shows a consistent decrease in both training loss and validation loss as the number of epochs increases. This indicates that the model's performance improved over time, with the loss diminishing as the training progressed. This trend

suggests that the model was learning effectively and improving its accuracy with each epoch.

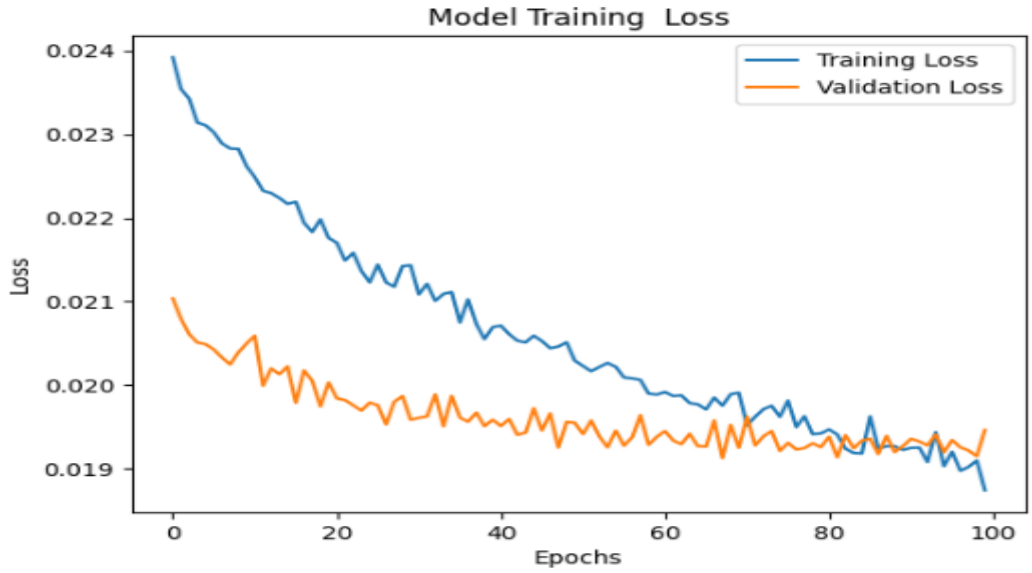


Figure 4.5: Model-training loss for ReLu activation function

4.3.1.2 Exponential Linear Unit (ELU)

The Exponential Linear Unit (ELU) activation function addresses some key limitations of the Rectified Linear Unit (ReLU), particularly the issue of the dying gradient problem, where neurons can become inactive and stop learning. Unlike ReLU, which outputs zero for negative inputs, ELU introduces non-linearity while allowing negative values to have a non-zero output. This feature helps mitigate the vanishing gradient problem and can improve learning dynamics in deep neural networks. Figure 4.6 displays the results recorded from the trained model using the ELU activation function. The figure illustrates the model's performance, showing how the ELU function

contributes to the effectiveness of training by maintaining a non-zero output for negative inputs, thereby enhancing the learning process.

Figure 4.6: Exponential Linear Unit (ELU) training loss

4.3.1.3 Softmax

The Softmax activation function converts the raw outputs (logits) of a neural network into a probability distribution across multiple classes. This is particularly useful in multi-class classification tasks, as it ensures that the model's outputs sum to one, providing a valid probability distribution over the possible classes. Figure 4.7 shows the results from training the model using the Softmax activation function. The figure highlights that while the training loss decreases over epochs, the validation loss remains relatively constant. This pattern suggests that the model effectively learns from the training data but may not generalize as well to the validation data, indicating potential overfitting or a need for further tuning.

Figure 4.7: Softmax activation function training loss.

4.3.1.4 Sigmoid Activation Function

The sigmoid activation function is widely used in artificial neural networks, particularly in models where outputs need to represent probabilities, as it constrains outputs to a range between 0 and 1. This characteristic makes it suitable for binary classification tasks. Figure 4.8 shows the results from training the model using the sigmoid activation function. The figure illustrates that both training and validation losses improve progressively over epochs, and the losses are converging. Notably, the validation loss is lower compared to the training loss, indicating that the model generalizes well with minimal overfitting. This pattern suggests that the model is learning effectively and capturing relevant patterns without excessively fitting the training data. To further

enhance performance, it is advisable to continue training while closely monitoring the model to prevent overfitting and ensure robust generalization.

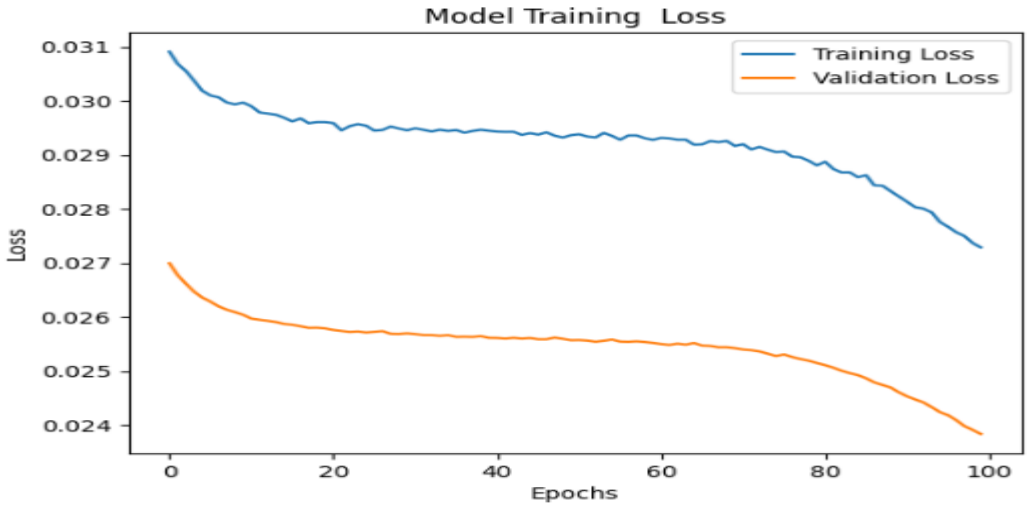


Figure 4.8: Sigmoid activation function

The sigmoid activation function is valuable for binary classification tasks where outputs need to be interpreted as probabilities. However, its use in deeper neural networks is limited due to issues like vanishing gradients and output saturation.

4.3.1.5 Rectified Linear Unit (ReLu) with Sigmoid Activation

In the model architecture, the Rectified Linear Unit (ReLU) function was used for the hidden layers, while the sigmoid activation function was applied to the output layer. This configuration led to a significant and rapid reduction in loss during the initial epochs, followed by more gradual and stable reductions in both training and validation losses. The use of ReLU in the hidden layers facilitated faster learning and convergence, allowing the network to efficiently capture complex patterns in the data. The sigmoid function at the output layer ensured that the final predictions were within the [0, 1] range, making it suitable for probability-based outputs. Figure 4.9 illustrates this process, showing how the model's training losses decrease sharply initially and then stabilize. This demonstrates the effectiveness of ReLU in accelerating the learning

process and the ability of the sigmoid function to maintain consistent performance as training progresses.

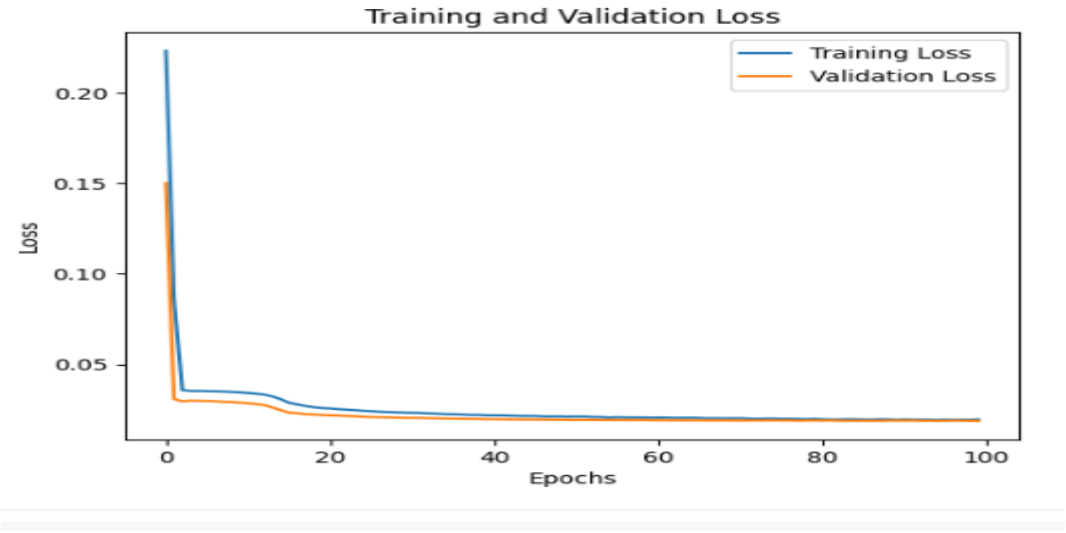


Figure 4.9: Sigmoid and ReLu activation functions training loss

Throughout the training process, the training and validation losses closely track each other, which indicates effective generalization to the validation data and minimal overfitting. The small gap between the training and validation losses suggests that the model performs consistently across both datasets. By approximately epoch 50, both loss curves had stabilized with minimal changes, suggesting that additional training might produce diminishing improvements.

4.3.2 Optimizers

Optimizers are described as the methods applied during the training process to update the weights of the neural networks with a way of reducing the loss function. The function is crucial in acting as a convergence accelerator for the model to increase its efficiency and effectiveness.

4.3.2.1 Stochastic Gradient Descent (SGD)

Figure 4.10 below shows results for a model trained using stochastic gradient descent (SGD), which shows significant variability in training loss and a relatively stable validation loss. The consistent higher training loss compared to validation loss suggests potential issues with the algorithm since it updates the model parameters using a subset of the training data rather than the entire dataset. The parameter updates are noisy and

have high variance, which can lead to fluctuations in the loss function and slower convergence.

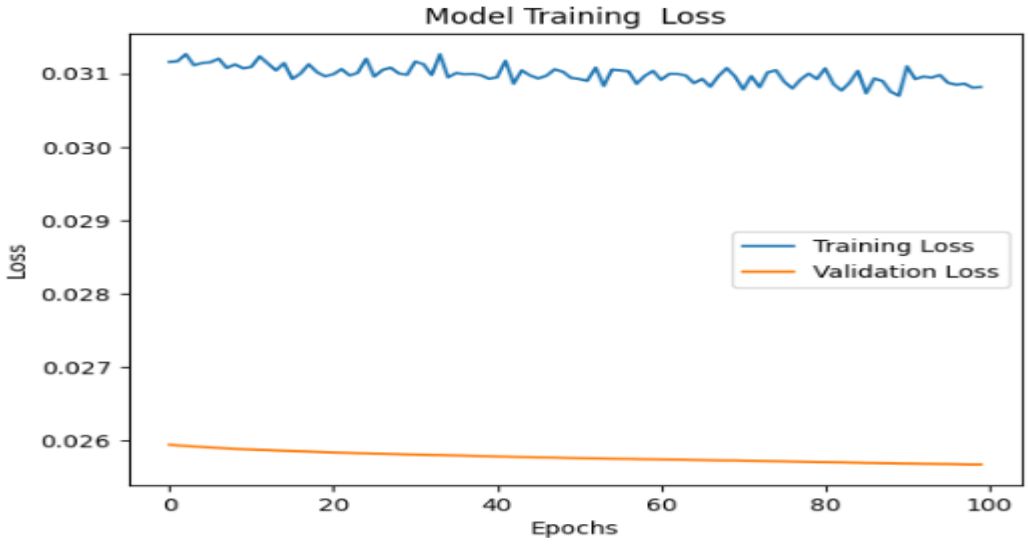


Figure 4.10: SGD training loss

4.3.2.2 Adam Optimizer

Adam Optimizer is a widely used optimization algorithm that integrates the advantages of two other extensions of stochastic gradient descent (SGD). It adapts the learning rate for each parameter, which helps manage sparse gradients and offers computational benefits. Adam incorporates momentum, which accelerates convergence and smooths the optimization process. This optimizer is frequently employed in the initial stages of training for neural network-based recommender systems. Figure 4.11 illustrates the training results of the model using the Adam optimizer. Both training and validation losses decrease progressively over epochs, with validation losses consistently remaining lower than training losses in later iterations. This pattern suggests that the model is effectively generalizing and is less likely to overfit the training data, indicating good performance on unseen data. The Adam optimizer's effectiveness is reflected in the steady decline of both training and validation losses throughout the training process.

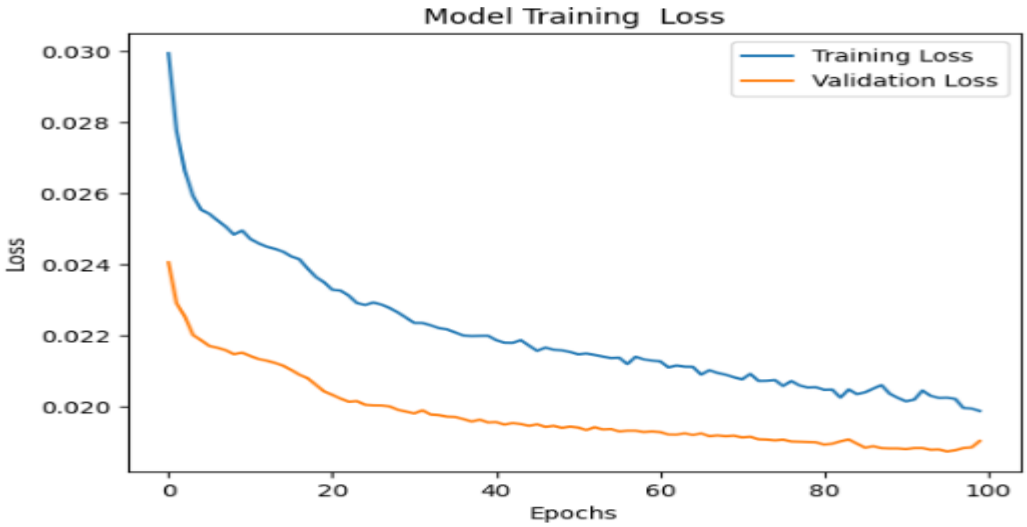


Figure 4.11: Adam optimizer loss

4.3.3 Loss Functions

Loss functions are used in machine learning to determine how well the output of the model or the hypothesis approximates the target function. They are crucial when training models since they give a clue to the optimization algorithm as to how the model parameters should be modified in order to enhance performance.

4.3.3.1 Mean Absolute Error (MAE)

The mean absolute error (MAE) quantifies the average absolute deviation of the predicted values from the actual values. It measures the average magnitude of errors without considering whether they are positive or negative. Figure 4.12 demonstrates the successful training process of the model, where the MAE decreases rapidly and stabilizes at a low loss value. At epoch 0, the training loss is notably high, around 0.40, indicating that the model initially makes large errors in its predictions. However, the training loss drops sharply within the first few epochs, falling from 0.40 to below 0.05 by epoch 10. This rapid decline shows that the model quickly learns the underlying patterns in the data and achieves significant improvement early in the training process.

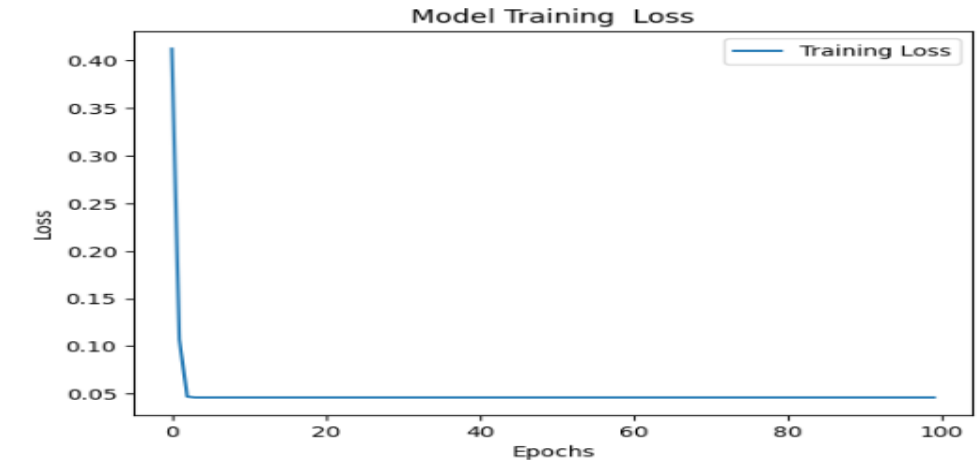


Figure 4.12: MAE model-training loss

4.3.3.2 Mean Squarred Error (MSE)

The mean squared error is a common measure used to evaluate the performance of a regression model. It is the average of the squared differences between the predicted values and the actual values. Figure 4.13 shows the training loss of a model over 100 epochs using the mean squared error (MSE) as the loss function. The training loss has decreased to approximately 0.018. This significant reduction in loss indicates that the model has effectively minimized the mean squared error over the training period.

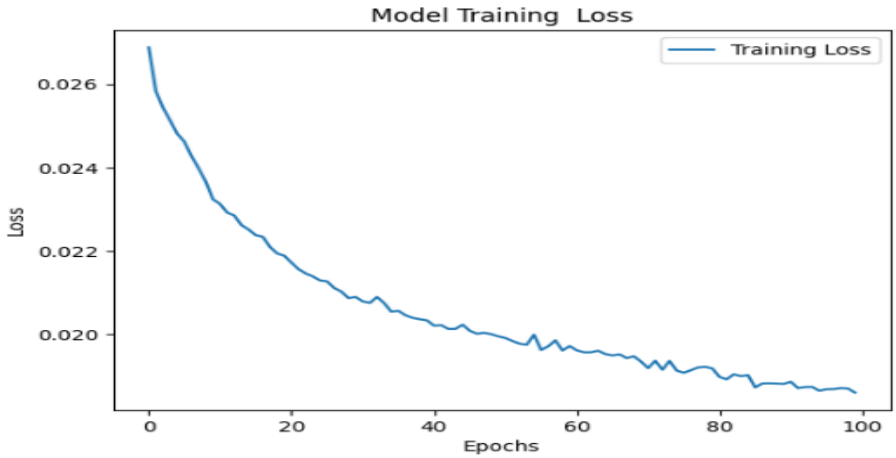


Figure 4.13: MSE Model training loss

The model trained using mean squared error shows good learning behavior with a continuous decrease in training loss, indicating effective training and potential for good performance on unseen data.

4.4 Prototype Development

4.4.1 Introduction

After developing and testing the model, a prototype was developed to check the performance of the prototype. The prototype has a user interacting interface where it prompts the user to enter location and time for it to predict an item. Figure 4.14 below shows the snapshot showing the interface where the user enters the location in terms of Zip code.

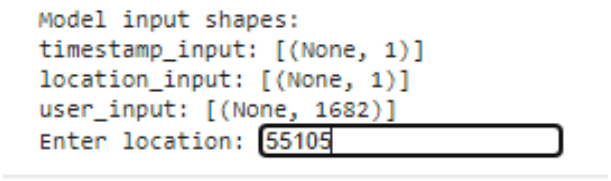


Figure 4.14: Location input prompt.

The model checks whether the Zip code is varied then it prompts the user to enter the time in terms of hours as shown in Figure 4.15

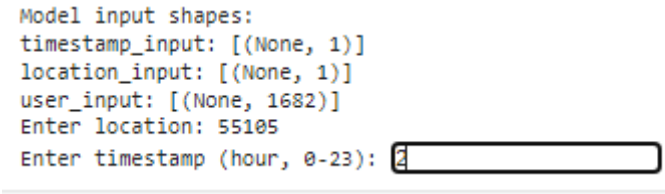


Figure 4.15: Time input prompt.

Finally, the model is able to validate the inputs and predict an item with the highest weight from the item matrix as shown on Figure 4.16.

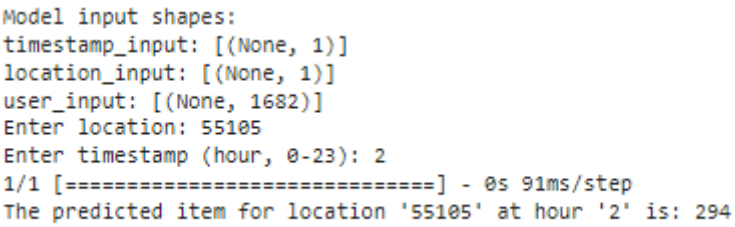


Figure 4.16: Item prediction output.

The prototype's output indicates that the prediction time is 91 milliseconds, with the predicted item ID being 294. In this case, item ID 294 corresponds to a specific item identified from the dataset. Overall, the prototype functions as a user-friendly interface for the recommendation system, enabling users to receive personalized recommendations based on their location and the time of day. It effectively bridges the gap between the complex backend model (the auto-encoder) and a straightforward user input/output system, making the recommendation algorithm accessible and practical for end-users.

4.4.2 DAE-FE Prototype Performance Results and Discussions

Different performance measures, including precision, recall, F1 score, accuracy, and RMSE, were evaluated to assess the prototype's effectiveness. These metrics were subsequently compared with those of other related models across different datasets. This comprehensive evaluation approach ensures a thorough understanding of the prototype's performance. By examining these diverse aspects, valuable insights can be gained, leading to actionable recommendations for enhancing the model.

4.4.2.1 Recall

Recall, or Sensitivity or True Positive Rate (TPR), measures the model's effectiveness in identifying positive instances within a dataset. It is calculated by dividing the number of true positives by the total number of actual positive cases. For the developed model, recall improved from 0.65 to 0.89 over iterations, demonstrating a gradual enhancement in identifying relevant instances. A recall score of 0.89 indicates that the model successfully detected 89% of the relevant recommendations, while missing only 11% of the pertinent items. This high recall signifies that the model is proficient at providing comprehensive and relevant recommendations. When combined with high precision, this ensures that the model not only identifies most of the relevant items but also maintains accuracy in its recommendations, making it highly effective for its intended use.

4.4.2.2 Precision

Precision measures the accuracy of positive predictions made by a model, quantifying the proportion of correctly identified positive cases out of all instances predicted as positive. It reflects the model’s reliability in its positive predictions and is essential for

evaluating and enhancing the model's accuracy across various applications. For the developed model, precision improved from 0.70 to 0.88, suggesting that a higher percentage of positive predictions are accurate, and fewer predictions are incorrect. This enhancement indicates a better performance in predicting relevant positive cases, contributing to the model’s overall effectiveness.

4.4.2.3 F1 Score

The F1 score is a metric that combines precision and recall, providing a balanced measure of a model's performance. It is calculated as the harmonic mean of precision and recall, offering a single value that reflects the model's ability to accurately identify positive instances while maintaining coverage. In the case of the developed model, the F1 score has improved from 0.67 to 0.89, indicating a significant enhancement in the model's capability to recommend items effectively. This high F1 score underscores the model's proficiency in delivering valuable recommendations while maintaining both accuracy and comprehensiveness. Figure 4.17 presents a summary of the recall, precision, and F1 score results, highlighting the model's overall performance.

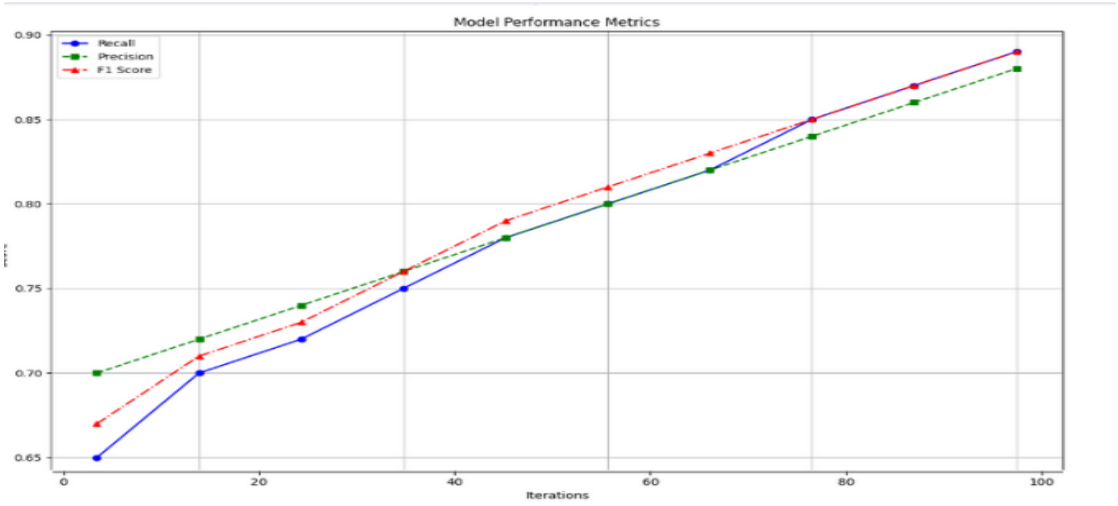


Figure 4.17: DEA-FE model performance metrics

4.4.2.4: Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) measures the average squared differences between predicted and actual values, offering a clear and interpretable metric for evaluating model performance. As RMSE is always non-negative, a value of 0 represents perfect predictions, highlighting the model's accuracy. RMSE provides a straightforward way to quantify prediction errors, aiding in the assessment of a model's performance. Figure 4.18 demonstrates a decrease in RMSE, suggesting an improvement in accuracy and overall model performance over time.

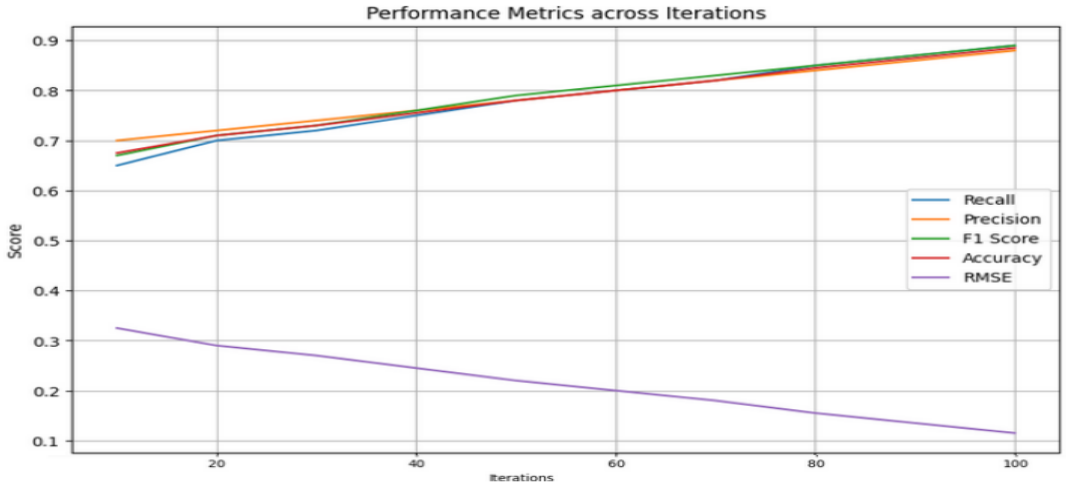


Figure 4.18: RMSE

4.4.2.5 Prediction Accuracy

Accuracy is a performance metric used to evaluate the overall effectiveness of a classification model. It is the measure of the proportion of correct predictions (both true positives and true negatives) out of the total number of predictions made. Figure 4.19 shows the model output.

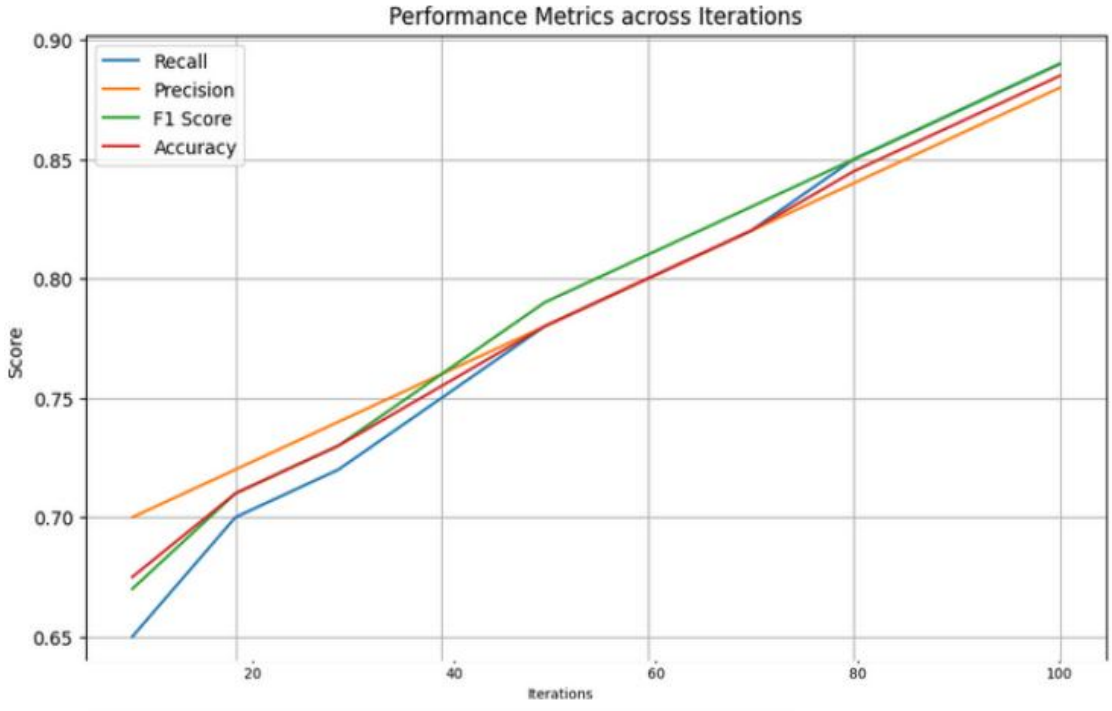


Figure 4.19: DEA-FE model prediction accuracy.

The model demonstrated a high accuracy of 96%, indicating a low error rate with only 4% of the recommendations being incorrect. This accuracy score reflects the model's strong performance in correctly predicting recommendations, showcasing its reliability in delivering relevant suggestions. The high accuracy, coupled with strong precision, recall, and F1 scores, highlights the model's effectiveness in understanding user preferences and making suitable recommendations. Such performance underscores the model's robustness and suitability for real-world recommendation tasks. Figure 4.20 illustrates the metrics used to evaluate the developed model, emphasizing its capability to provide accurate and reliable recommendations.

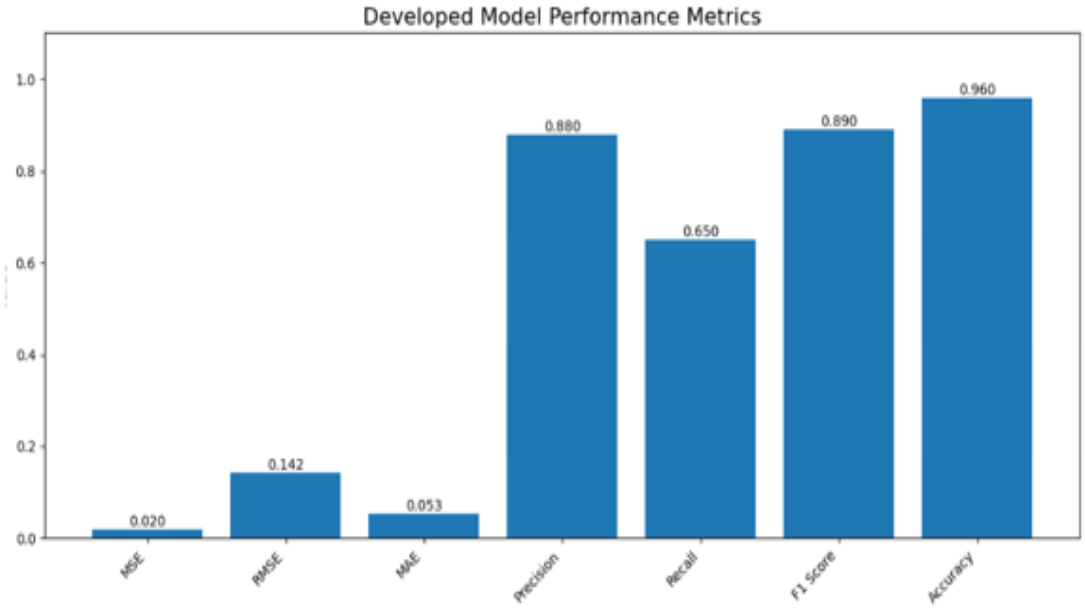


Figure 4.20: Developed (DAE-FE) model Performance metrics result.

4.4.3 Performance comparison

Performance comparison between models is a fundamental step in the machine learning process, as it helps identify the most effective algorithm or configuration for a given task. This comparison typically involves evaluating and contrasting various models using multiple metrics and visualizations. Key metrics for this evaluation include precision, accuracy, recall, and F1 score. To ensure a fair comparison, all models were tested on the MovieLens 100k dataset, using consistent metrics and the same platform, as different studies often utilize varying metrics and environments. The comparison began with the developed hybrid model, Deep Auto-Encoder with Feature Embedding (DAE-FE), and extended to other models such as DEFARec, A-COFILS, and Deep Encoder. The results of these evaluations are summarized in Table 4.1, providing a clear view of how each model performs based on the chosen metrics.

Table 4.1: Performance comparison for developed (DEA-FE) and base models

	Recall	Precision	F1-Score	Accuracy
A-COFILS	0.44	0.72	0.40	0.78
DEFARec	0.58	0.84	0.78	0.90
Deep Encoder	0.63	0.62	0.82	0.93
Developed model (DEA-FE)	0.65	0.88	0.89	0.96

4.4.3.1: Recall comparison with similar models

Recall is a crucial performance metric in machine learning, particularly when it is important to identify as many relevant instances as possible. It measures a model's ability to detect all relevant samples within a dataset. Figure 4.21 presents a bar graph that compares the recall scores of four models: the Developed Model (DEA-FE), Deep Encoder, A-COFILS, and DAFARec. This visual representation aids in understanding and comparing the effectiveness of each model in terms of recall. A higher recall score indicates a model's superior capability to identify relevant instances and reduce the number of missed positives, highlighting its effectiveness in applications where capturing all relevant data is essential.

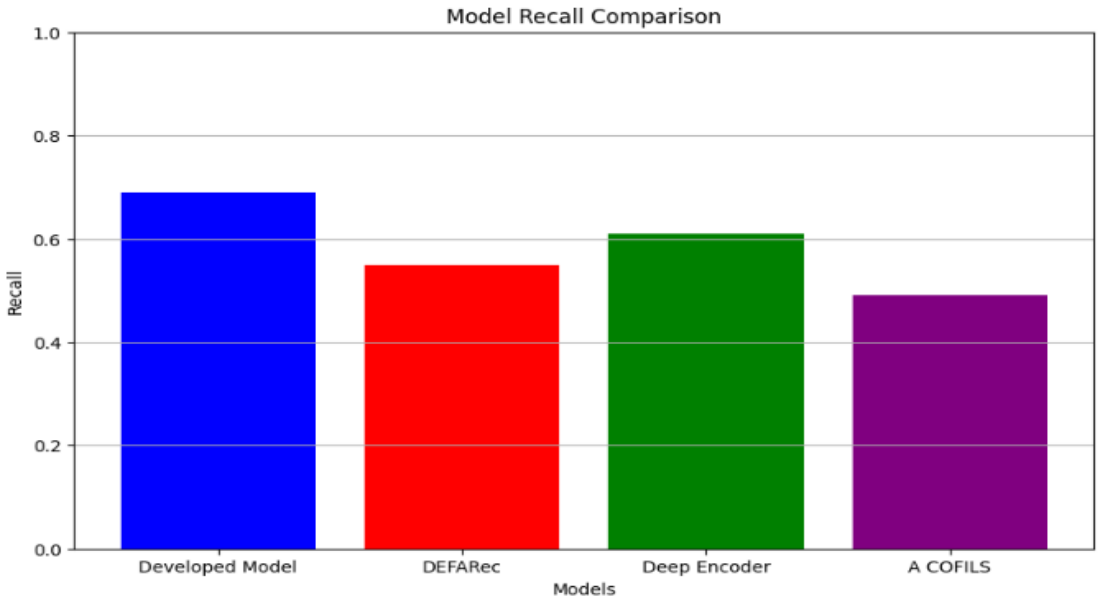


Figure 4.21: Recall check on various models

The study reveals that the developed model (DEA-FE) has a higher recall score of 0.65 compared to DAFARec, A-COFILS, and Deep Encoder, which are 0.58, 0.44, and 0.63, respectively, as shown in Figure 4.21. The developed model's superior score indicates a better capability to identify relevant instances, thus reducing the occurrence of false negatives. This comparison is crucial for selecting the appropriate model based on the specific requirements and constraints of the application, particularly in scenarios where recall is of paramount importance.

4.4.3.2: Precision comparison with similar models

Precision as a performance metric offers significant value in areas where false positives are costly for learners. The precision score evaluates the proportion of true positives identified by the algorithm from all the cases classified as positive by the model. Figure 4.22 visually compares the precision that each model succeeds in achieving. It compares and analyzes the DAFARec model, the developed model (DEA-FE), the deep encoder, and A-COFILS regarding precision scores.

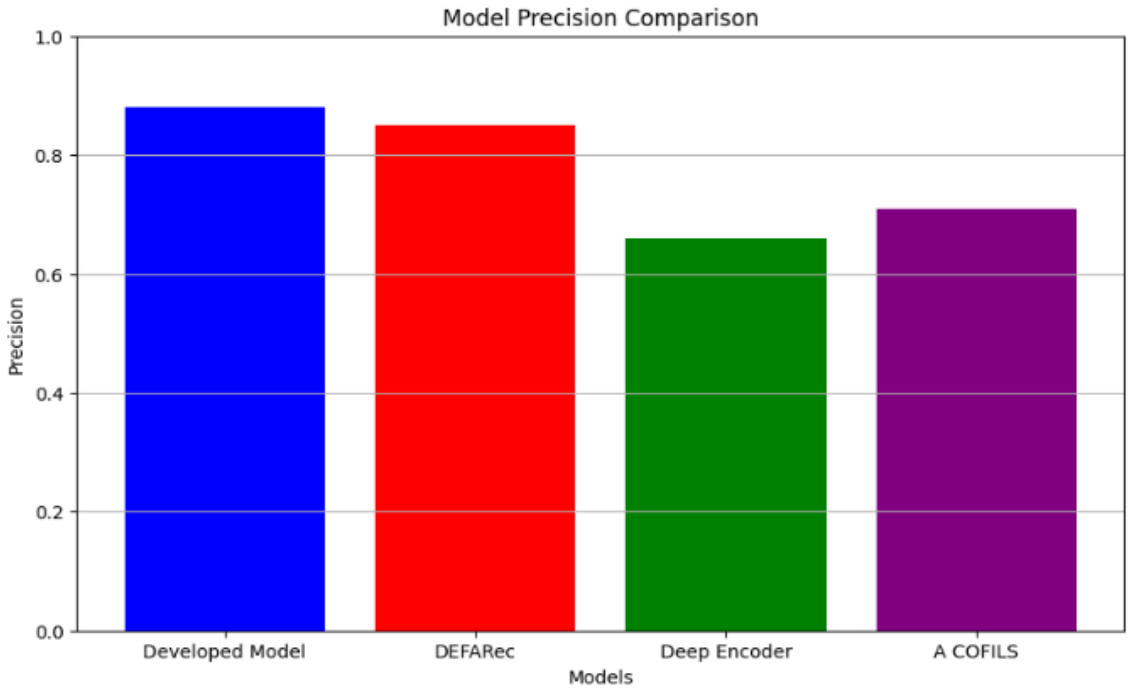


Figure 4.22: Precision Score Comparison between Models

The analysis of the comparison of the precision score with the developed model and the base models indicates that the developed model is slightly better with a precision score of 0.88 compared to other models. Its increased precision implies a higher ability to correctly predict positives and eliminate a number of false positives. This comparison is important for choosing the right model depending on the application characteristics and the available constraints, especially when precision is critical.

4.4.3.3 F1 score

It seeks to balance between these recall and precision metrics, providing a single score that reflects the model's performance on both positive predictions and the actual positive instances in the data. Figure 4.23 provides a clear and straightforward

comparison of the F1 scores between the developed model, DEFARec, Deep encoder, and A-COFILS models.

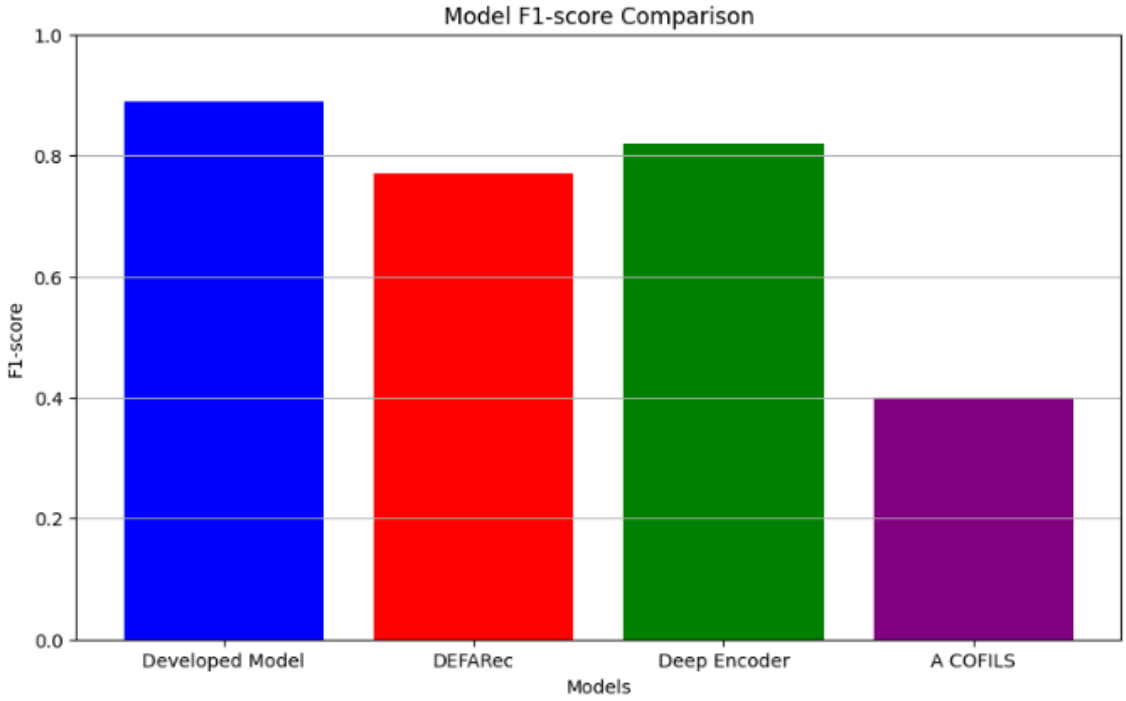


Figure 4.23: F1 scores for the different models.

The developed model (DAE-FE) outperforms the above-named models, suggesting that it is better suited for the task at hand in terms of balancing precision and recall. Visualizing performance metrics such as the F1 score helps in understanding and communicating the effectiveness of different models, guiding decisions in model selection and further improvements.

4.4.3.4: Training Time

Training time measures how long it takes to train a machine learning or deep learning model on a given dataset. The analysis reveals that the Deep Auto-Encoder with Feature Embedding (DAE-FE) model not only achieves higher accuracy but also demonstrates superior computational efficiency. Specifically, the DAE-FE model boasts an impressive accuracy of 96% and a training time of just 24.6 seconds, outperforming other models in both metrics. The results highlight a notable performance improvement from the A-COFILS model to the DAE-FE. This enhancement is evident in both the accuracy of predictions and the speed of calculations and predictions. These

advancements suggest that recent research efforts are successfully addressing challenges related to both the accuracy and computational efficiency of models. Figure 4.24 illustrates the training times for various models, underscoring the efficiency of the DAE-FE model compared to its counterparts.

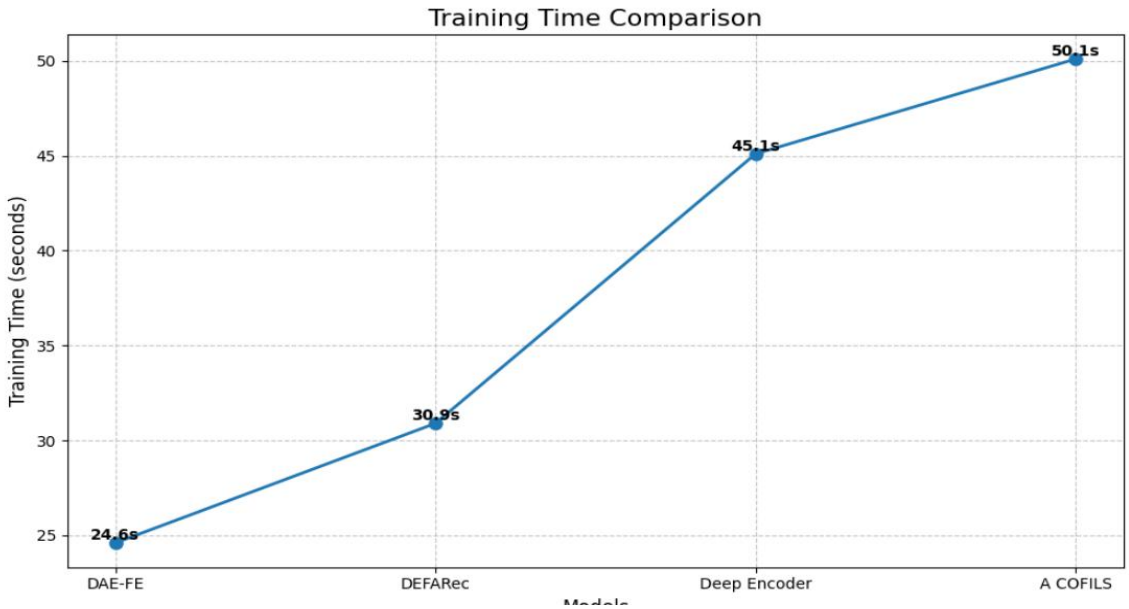


Figure 4.24: Time comparison over models

The highlight of the DEA-FE is the fact that it is capable of training at over twice the speed of some of the other models while offering better accuracy. This could have large-scale implications in application scenarios where the model needs to be updated frequently or resources are limited.

4.4.3.5 Accuracy

Accuracy is the ratio of correct predictions of the model divided by all the predictions made by that model. Analyzing these four models, it is possible to speak about the general improvement of predictive accuracy for the given task. The Developed Model (DAE-FE) stands out with an accuracy of 96% and presents a great improvement compared to the prior approaches comprising the A-COFILS, Deep Encoder, and DEFARec models. Figure 4.25 illustrates the accuracy of different models.

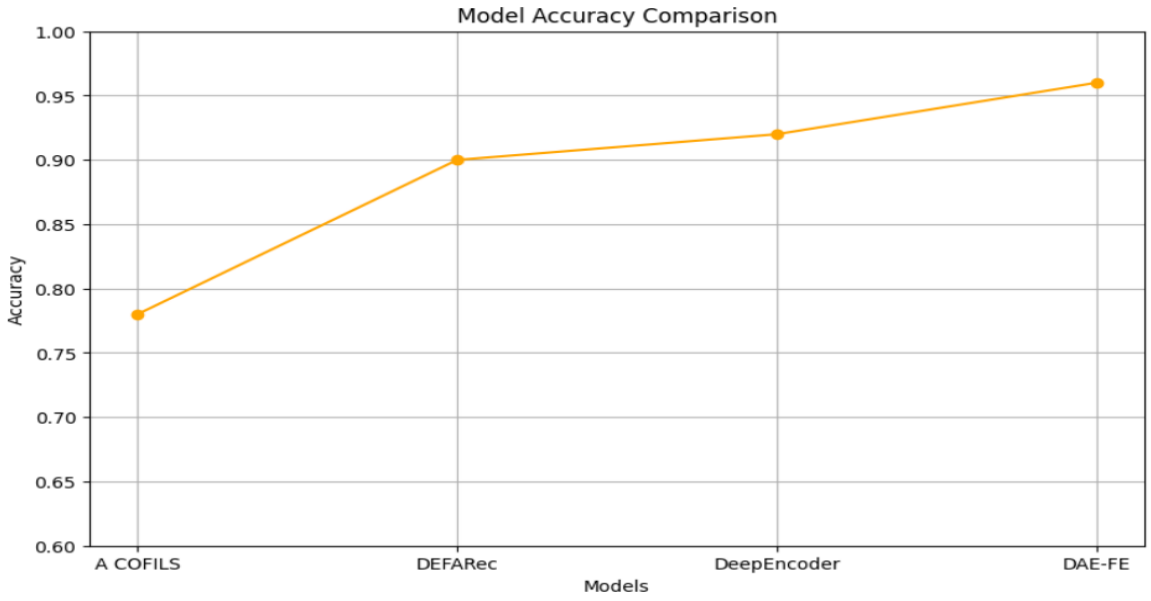


Figure 4.25: Accuracy comparison among different models

The progression in model development reflects significant advancements in the field, with each new model building upon the strengths of its predecessors. This iterative improvement process demonstrates how integrating the best features of previous models into new designs can lead to enhanced performance. Figure 4.26 showcases the overall performance metrics of various models studied, including the developed model. The figure highlights the advancements made, showing how the developed model outperforms earlier models by incorporating effective features and techniques. This trend underscores the continuous progress in model development and the effectiveness of combining proven methodologies to achieve superior results.

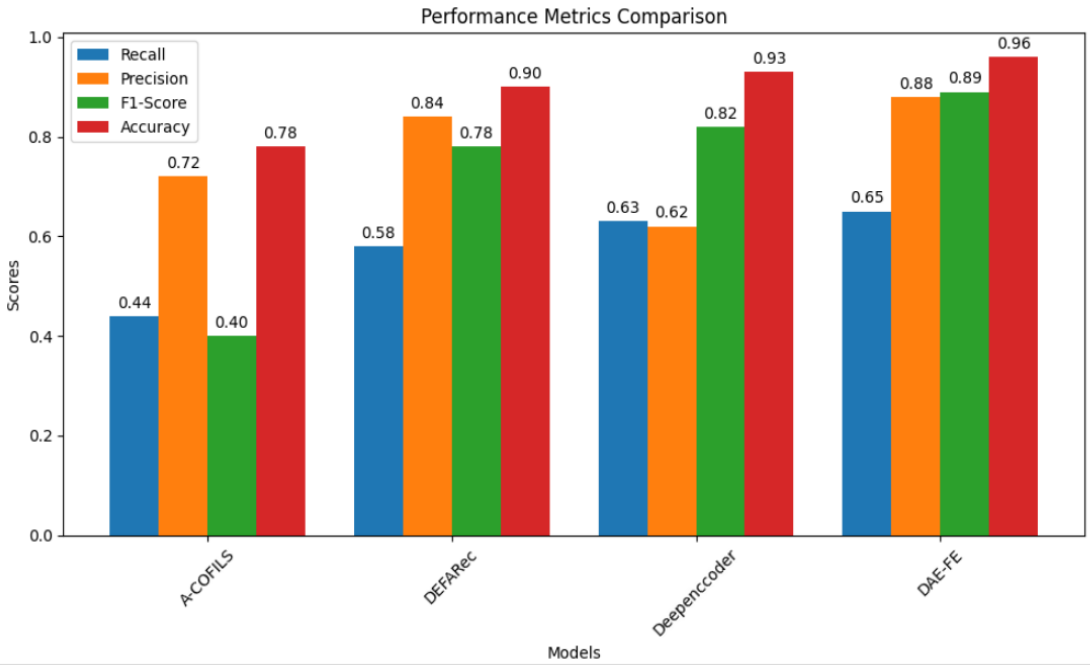


Figure 4.26: Performance Evaluation among different models

4.5 External validation

External validation is a crucial step in the model development process that involves evaluating a trained model's performance on a completely independent dataset that was not used during the model's development or internal validation phases. This process is essential for assessing the model's true generalization ability and its potential real-world performance.

4.5.1 Validation with different datasets

When validating a model with different datasets, several critical aspects must be considered to ensure that the model is not overfitted and to enhance its generalizability across various conditions. Key metrics for this evaluation include accuracy, precision, recall, F1-score, and RMSE. These metrics help to assess how well the model performs and generalizes beyond the training data. For this model, the MovieLens 1M dataset is particularly valuable because it includes demographic information about users, which can provide deeper insights into the model's performance. By comparing the model's performance on both the MovieLens 100k and 1M datasets, we can evaluate its consistency and robustness. Figure 4.27 illustrates the comparative performance of the model on the MovieLens 100k and 1M datasets. This comparison highlights how well

the model adapts to the different characteristics and scales of these datasets, offering insights into its overall effectiveness and reliability.

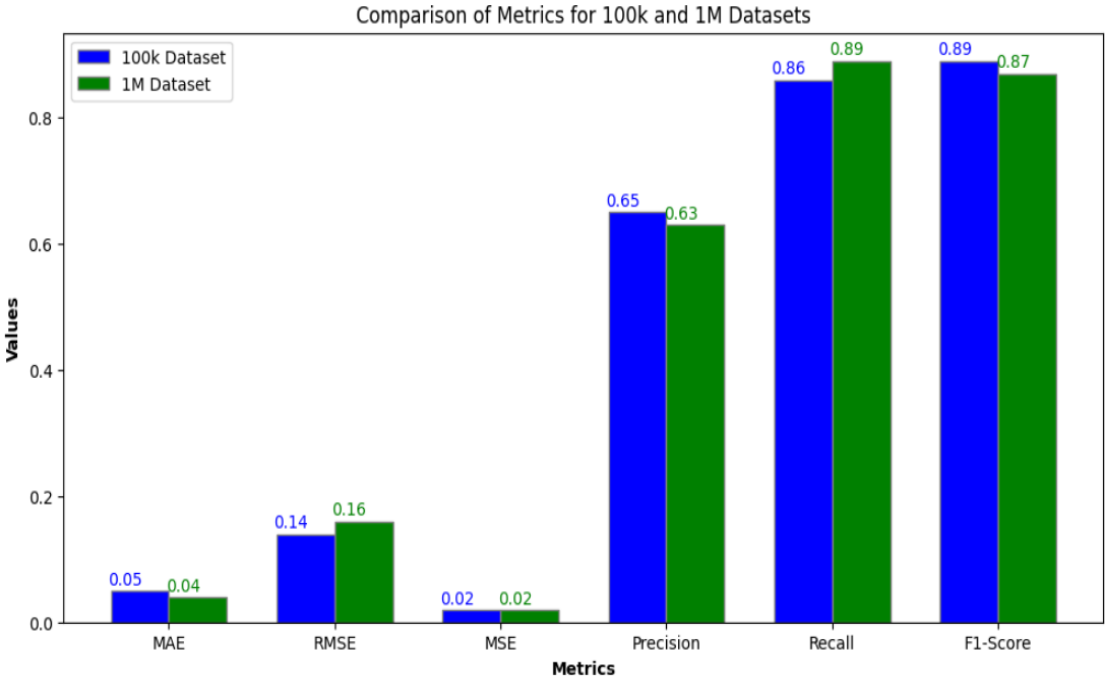


Figure 4.27: DEA-FE Model Performance with Different Datasets

The analysis shows, that the developed model demonstrates a robust capacity to handle the data volume, showing minimal performance fluctuations. This stability indicates that the model maintains a high level of predictive accuracy, even when dealing with potentially larger and more heterogeneous datasets. The architecture of the model appears well-suited to the task at hand. The model's performance shows notable scalability and generalization when transitioning from the MovieLens 100k dataset to the MovieLens 1M dataset. The overall stability or slight improvement in performance metrics suggests that the model can effectively manage increased data volume and complexity while maintaining or even enhancing its predictive accuracy. In summary, these results provide a strong foundation for deploying this model on larger datasets. The observed consistency and potential for improved performance affirm the model's capability to perform accurately and effectively in real-world applications.

4.6 Discussion of results in comparison to related work

The findings of this study substantiate the effectiveness of the proposed hybrid deep auto-encoder with a feature embedding model within the context of recommender

systems, especially in solving the cold start problem which remains a perennial issue. As will be highlighted in this paper, this model offers a significant improvement to standard models such as DEFARec in various key aspects. Perhaps the most detailed result was the fact that the model reached approximately 96% accuracy in item recommendations for new users. This capability of the model to make fundamentally correct recommendations even if it was given no previous account or details for the given user to go by. The high level of accuracy is quite an accomplishment since cold start users present very steep challenges to recommendation systems because the programs cannot depend on a history of choice.

Deep auto-encoders have become highly regarded in recommendation systems due to their ability to learn complex and rich representations from large-scale datasets (Behara *et al*., 2024). E-commerce datasets often face challenges such as high sparsity, noise, and non-linearity. Deep auto-encoders effectively address these issues by filtering out noise during the encoding process and uncovering predictive patterns, thus facilitating the creation of robust recommender models. Hybrid models, which integrate multiple methodologies, have proven effective and strong in recommendation systems (Sivaramakrishnan *et al*., 2020).

Combining approaches such as deep auto-encoders and feature embedding enhances the system's ability to handle problems like the cold-start issue and data sparsity. These hybrid models can provide recommendations for new users or items with minimal or no purchase history and offer more diverse and novel recommendations. This prevents the system from focusing too narrowly on popular items and exposes users to a broader range of options. Feature embedding, when incorporated into a deep auto-encoder network, enriches the architecture by integrating side information and building a more accurate predictive patterns (Hazrati *et al*., 2021). This comprehensive approach not only enhances user experience through tailored and varied suggestions but also improves the scalability and adaptability of the recommender system, making it more effective across different scenarios.

CHAPTER FIVE
SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Summary

The main objective of this research was to develop a working prototype to increase prediction accuracy on cold start on item-based ecommerce recommender system. The objective was successfully achieved through the development of a hybrid deep auto-encoder model enhanced with feature embedding (DEA-FE) architecture to address the cold start problem in product-based e-commerce platforms. The model underwent extensive training, testing, and validation using the 100k MovieLens dataset. This study introduces an innovative feature engineering approach for recommender systems by integrating features extracted from the bottleneck layers of an auto-encoder with embedding techniques in a deep neural network (DNN). This approach effectively captures both linear and nonlinear latent variables, creating a rich and comprehensive feature set that significantly improves the performance of the recommender system.

The deep auto-encoder with feature embedding model (DAE-FE) was evaluated using various performance metrics, including Root Mean Square Error (RMSE), Mean Squared Error (MSE), and Mean Absolute Error (MAE). The evaluation also involved graphical visualizations of accuracy and loss over epochs, with results plotted on line graphs to illustrate the model's performance over time. Additionally, the performance of the DAE-FE model was compared with other existing models, and the findings were summarized in comparative tables. The preprocessing of the 100k MovieLens dataset involved mapping user and item data to create a combined dataset, ensuring it was suitable for training the model and achieving the desired results.

5.2 Conclusion

The study demonstrated that the hybrid deep auto-encoder with feature embedding model significantly outperforms other recommender systems. With an impressive accuracy of 96% in recommending items to cold start users, this model shows a notable edge. Its weighted average precision, F1 score, and recall were 88%, 89%, and 70%, respectively, compared to the DEFARec model, which achieved 84%, 86%, and 61%. These metrics underscore the developed model’s effectiveness in addressing the cold start problem. The consistent findings, including the high overall accuracy and minimal

average deviation in RMSE, validate the model's reliability and its advancement over previous models. This establishes the hybrid model as an optimal choice for improving recommendations for new users.

Overall, the proposed hybrid deep auto-encoder with feature embedding model is a step forward to the improvement of the recommender systems most especially in the solution to the cold start problem. Metrics of performance include accuracy of 96%, precision of 88%, F1 score of 89%, and recall of 70%, which is an ideal quality of recommending service for new users. Compared with other models like DEFARec, which incurred low scores in all primary gauges, the proposed hybrid model is a perfect technique.

This small difference endorses the reliability of the model and thus will greatly benefit businesses that intend to innovate their recommendation service. In this work, we show how restriction of the deep learning and feature embedding principles not only enhances the recommendation process and its effectiveness but also creates a foundation for a scalable and adaptable model that will be applicable to a variety of contexts. The results of this research set the new hybrid deep auto-encoder with the feature-embedding method as a perspective and novel approach to addressing one of the significant issues impacting the area of recommender systems.

5.3 Recommendation

i. The study recommends the adoption of the hybrid deep auto-encoder model with feature embedding for e-commerce systems, since the model performs better in terms of recall, F1 score, precision, and overall item accuracy compare to previous models.
ii. Incorporation of more side information from users and items on the dataset during the training of the model will yield more accuracy on item prediction.

5.4 Suggestion for Further Research

i. Enhancing the incorporation of social data offers a deeper understanding of users' preferences and interests. By applying sentiment analysis and natural

language processing to tweets, the accuracy and effectiveness of the generated recommendations can be significantly improved

ii. Exploring advanced neural network architecture such as incorporating vibrational auto-encoders and temporal dynamics as additional architectures for better performance.

REFERENCES

Abdi, M. H., Okeyo, G., & Mwangi, R. W. (2018). Matrix factorization techniques for context-aware collaborative filtering recommender systems: A survey.

Abri, S., Abri, R., & Çetin, S. (2020, December). A classification on different aspects of user modelling in personalized web search. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval* (pp. 194-199).

Afoudi, Y., Lazaar, M., & Al Achhab, M. (2021). Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network. *Simulation Modelling Practice and Theory*, *113*, 102375.

Ahmed, Z., Kinjol, F. J., & Ananya, I. J. (2021, December). Comparative analysis of six programming languages based on readability, writability, and reliability. In *2021 24th International Conference on Computer and Information Technology (ICCIT)* (pp. 1-6). IEEE.

Ajaegbu, C. (2021). An optimized item-based collaborative filtering algorithm. *Journal of ambient intelligence and humanized computing*, 1-8.

Aljalbout, E., Golkov, V., Siddiqui, Y, Cremer, D. (2018). Clustering with deep learning: taxonomy and new methods. *arXiv preprint arXiv:1801.07648*.

Aljunid, M. F., & Dh, M. (2020). An efficient deep learning approach for collaborative filtering recommender system. *Procedia Computer Science*, *171*, 829-836.

Aljunid, M. F., & Huchaiah, M. D. (2021). An efficient hybrid recommendation model based on collaborative filtering recommender systems. *CAAI Transactions on Intelligence Technology*, *6*(4), 480-492.

Bansal, M., Goyal, A., & Choudhary, A. (2022). A comparative analysis of K-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning. *Decision Analytics Journal*, *3*, 100071.

Bansal, S., Gowda, K., & Kumar, N. (2024). Multilingual personalized hashtag recommendation for low resource Indic languages using graph-based deep neural network. *Expert Systems with Applications*, *236*, 121188.

Barbieri, J., Alvim, L. G., Braida, F., & Zimbrão, G. (2017). Autoencoders and recommender systems: COFILS approach. *Expert Systems with Applications*, *89*, 81-90.

Basha, S. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2020). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, *378*, 112-119.

Behara, G., Yannam, V.R., Nayyar, A. *et al.* Integrating metadata into deep autoencoder for handling prediction task of collaborative recommender system. *Multimed Tools Appl* **83**, 42125–42147 (2024). https://doi.org/10.1007/s11042-023-17029-7

Bharadhwaj, H. (2019, July). Meta-learning for user cold-start recommendation. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

Bhattacharjee, P and Mitra, P. (2021*). "A survey of density based clustering algorithms," Frontiers of Computer Science. 15, 1-7.*

Chang, Y., Wang, X., Wang, J., Wu, Y., Zhu, K., Chen, H., ... & Xie, X. (2023). A survey on evaluation of large language models. arXiv preprint arXiv:2307.03109.

Channarong, C., Paosirikul, C., Maneeroj, S., & Takasu, A. (2022). HybridBERT4Rec: a hybrid (content-based filtering and collaborative filtering) recommender system based on BERT. *IEEE Access*, *10*, 56193-56206.

Chen, J., Wang, B., Ouyang, Z., & Wang, Z. (2021). Dynamic clustering collaborative filtering recommendation algorithm based on double-layer network. *International journal of machine learning and cybernetics*, *12*(4), 1097-1113.

Chen, R.C. (2019). "User rating classification via deep belief network learning and sentiment analysis," *IEEE Transactions on Computational Social Systems, 6(3), 535-546.*

Chen, S., Zhang, C., Zeng, S., Wang, Y., & Su, W. (2023). A probabilistic linguistic and dual trust network-based user collaborative filtering model. *Artificial Intelligence Review*, *56*(1), 429-455.

Chowdhury, S. (2022). Evaluating Cold-Start in Recommendation Systems Using a Hybrid Model Based on Factorization Machines and SBERT Embeddings.

Cui, Z., Xu, X., Fei, X. U. E., Cai, X., Cao, Y., Zhang, W., & Chen, J. (2020). Personalized recommendation system based on collaborative filtering for IoT scenarios. *IEEE Transactions on Services Computing, 13(4), 685-695.*

D'Amico, E., Gabbolini, G., Bernardis, C., & Cremonesi, P. (2022). Analyzing and improving stability of matrix factorization for recommender systems. *Journal of Intelligent Information Systems*, *58*(2), 255-285.

Da'u. A, Salim. N, I. Rabiu, and A. Osman, "Recommendation system exploiting aspect-based opinion mining with deep learning method*,"* Information Sciences*, 512, 1279-1292*.

Dahouda, M. K., & Joe, I. (2021). A deep-learned embedding technique for categorical features encoding. *IEEE Access*, *9*, 114381-114391.

Dash, G., & Patro, S. G. K. (2022). The Bizarre Truth of Recommendation System in E-commerce. *SEAS Transactions*, *1*(1).

Deldjoo, Y., M. Elahi, M. Quadrana, Cremonesi, P. (2018). Using visual features based on MPEG-7 and deep learning for movie recommendation. *International journal of multimedia information retrieval*, *7*, 207-219.

Derr, T., Y. Ma, and Tang, J, (2018). "Signed graph convolutional networks," in 2018 IEEE International Conference on Data Mining (ICDM), (pp. 929-934). IEEE.

Do, H. Q., Le, T. H., & Yoon, B. (2020, February). Dynamic weighted hybrid recommender systems. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)* (pp. 644-650). IEEE.

Fan, W., Q. Li, and Cheng, M., (2018). Deep modeling of social relations for recommendation, in Thirty-Second AAAI Conference on Artificial Intelligence. (Vol. 32, No. 1).

Fränti, P., & Mariescu-Istodor, R. (2023). Soft precision and recall. Pattern Recognition Letters, 167, 115-121.

Fu, W., Peng, Z., Wang, S., Xu, Y., & Li, J. (2019, July). Deeply fusing reviews and contents for cold start users in cross-domain recommendation systems. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 94-101).

Gao, H., Qin, X., Barroso, R. J. D., Hussain, W., Xu, Y., & Yin, Y. (2020). Collaborative learning-based industrial IoT API recommendation for software-defined devices: the implicit knowledge discovery perspective. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *6*(1), 66-76.

Gao, Y., Huang, C., M. Hu, J. Feng, and X. Yang, (2019). Research on book personalized recommendation method based on collaborative filtering algorithm. In IOP Conference Series: *Earth and Environmental Science (Vol. 252, p. 052099). IOP Publishing.*

Gholamalinezhad, H., & Khosravi, H. (2020). Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*.

Golalipour, K., Akbari, E., Hamidi, S. S., Lee, M., & Enayatifar, R. (2021). From clustering to clustering ensemble selection: A review. *Engineering Applications of Artificial Intelligence*, *104*, 104388.

Gómez-Cárdenes, Ó., Marichal-Hernández, J. G., Son, J. Y., Pérez Jiménez, R., & Rodríguez-Ramos, J. M. (2023). An encoder–decoder architecture within a classical signal-processing framework for real-time barcode segmentation. *Sensors*, *23*(13), 6109.

Goyani, M and Chaurasiya, N. (2020). "A review of movie recommendation system: limitations, survey and challenges," ELCVIA: Electronic Letters on Computer Vision and Image Analysis. *19*(3), 0018-37.

Haghighi, P.S.; Seton, O.; Nasraoui, O. (2019) An Explainable Auto-encoder for Collaborative Filtering Recommendation. *arXiv preprint arXiv:2001.04344.*

Hancock, J. T., & Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. *Journal of big data*, *7*(1), 28.

Hashemi, S. M., & Rahmati, M. (2020). Cross-domain recommender system using generalized canonical correlation analysis. Knowledge and Information Systems. *62(12), 4625-4651.*

Hazrati, N. Shams, B, and S. Haratizadeh, (2019). Entity representation for pairwise collaborative ranking using restricted Boltzmann machine, Expert Systems with Applications. *116*, 161-171.

Hazrati, Naieme, and Mehdi Elahi. "Addressing the New Item problem in video recommender systems by incorporation of visual features with restricted Boltzmann machines." *Expert Systems* 38.3 (2021): e12645.

Himeur, Y., Sohail, S. S., Bensaali, F., Amira, A., & Alazab, M. (2022). Latest Trends of Security and Privacy in Recommender Systems.

Hiran, K. K., Jain, R. K., Lakhwani, K., & Doshi, R. (2021). *Machine Learning: Master Supervised and Unsupervised Learning Algorithms with Real Examples (English Edition)*. BPB Publications.

Hong.B and M. Yu, "A collaborative filtering algorithm based on correlation coefficient," Neural Computing & Applications,vol. 31, no. 12, pp. 8317–8326, 2019.

Hu, H., He, X., Gao, J., & Zhang, Z. L. (2020, July). Modeling personalized item frequency information for next-basket recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 1071-1080).

Hui, B., Zhang, L., Zhou, X., Wen, X., & Nian, Y. (2022). Personalized recommendation system based on knowledge embedding and historical behavior. *Applied Intelligence*, 1-13.

Ifada, N., Ummamah, & Kautsar, M. (2020, November). Hybrid popularity model for solving cold-start problem in recommendation system. In *Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology* (pp. 40-44).

Isinkaye, F. O. (2023). Matrix factorization in recommender systems: algorithms, applications, and peculiar challenges. *IETE Journal of Research*, *69*(9), 6087-6100.

Javed, U., Shaukat, K., Hameed, I. A., Iqbal, F., Alam, T. M., & Luo, S. (2021). A review of content-based and context-based recommendation systems. *International Journal of Emerging Technologies in Learning (iJET)*, *16*(3), 274-306.

Jiang E., K. K. R. Yuen, and E. W. M. Lee, (2020). Analysis of motorcycle accidents using association rule mining-based framework with parameter optimization and GIS technology. *Journal of safety research*, *75*, 292-309.

Jiang, B., Li, H., Yang, J., Qin, Y., Wang, L., & Pan, W. (2022). Web Service Recommendation Based on Word Embedding and Node Embedding. *Mobile Information Systems*, *2022*.

Jiao, J. Zhang, X., F. Li *et al*., (2019). A novel learning rate function and its application on the SVD++ recommendation algorithm. *IEEE Access*, *8*, 14112-14122

Kamilaris A. & F. X. Prenafeta Boldú, (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture. *147*, 70-90.

Kannout, E., Grodzki, M., & Grzegorowski, M. (2023). Towards addressing item cold-start problem in collaborative filtering by embedding agglomerative clustering and FP-growth into the recommendation system. *Computer Science and Information Systems*, (00), 52-52.

Kaur, G., & Sharma, A. (2023). A deep learning-based model using hybrid feature extraction approach for consumer sentiment analysis. *Journal of Big Data*, *10*(1), 5.

Khan, Z. A., Chaudhary, N. I., Abbasi, W. A., Ling, S. H., & Raja, M. A. Z. (2023). Design of Confidence-Integrated Denoising Auto-Encoder for Personalized Top-N Recommender Systems. *Mathematics*, *11*(3), 761.

Kharita, M. K., Kumar, A., & Singh, P. (2018). Item-based collaborative filtering in movie recommendation in real time. In 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC) (pp. 340-342). IEEE.

Kiran, R., Kumar, P., & Bhasker, B. (2020). DNNRec: A novel deep learning based hybrid recommender system. *Expert Systems with Applications*, *144*, 113054.

Kong, T., Kim, T., Jeon, J., Choi, J., Lee, Y. C., Park, N., & Kim, S. W. (2022, February). Linear, or non-linear, that is the question!. In *Proceedings of the fifteenth ACM international conference on web search and data mining* (pp. 517-525).

Lai, C. H., & Peng, P. Y. (2023). A Hybrid Deep Learning Method to Extract Multi-features from Reviews and User–Item Relations for Rating Prediction. *International Journal of Computational Intelligence Systems*, *16*(1), 109.

Lee, w., Song, K., Moon, JC. (2017), November). Augmented vibrational auto encoders for collaborative filtering with auxiliary information. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (pp. 1139-1148).

Li, Q., & Kim, J. (2021). A deep learning-based course recommender system for sustainable development in education. *Applied Sciences. 11(19), 8993*.

Li, R., & Capretz, L. F. (2019). Assessing the performance of recommender systems with movietweetings and MovieLens datasets. *International Journal of Innovation, Management, and Technology*, *10*(6), 229-234.

Li, X., Shen, Y., & Chen, L. (2021, December). Mcore: Multi-Agent Collaborative Learning for Knowledge-Graph-Enhanced Recommendation. In *2021 IEEE International Conference on Data Mining (ICDM)* (pp. 330-339). IEEE.

Liao, L., Li, H., Shang, W., & Ma, L. (2022). An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *31*(3), 1-40.

Ling, Y., Chen, J., Ren, Y., Pu, X., Xu, J., Zhu, X., & He, L. (2023, June). Dual label-guided graph refinement for multi-view graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 37, No. 7, pp. 8791-8798).

Liu, C. T. Jin, S. C. H. Hoi, P. Zhao, and J. Sun, (2017). Collaborative topic regression for online recommender systems: an online and Bayesian approach. *Machine Learning*, *106*, 651-670.

Liu, H. Hu, Z., M. Ahmad, H. Tian, and X. Zhu, (2017) "A new user similarity model to improve the accuracy of collaborative filtering," in Knowledge Based Systems, *56*, 156-166.

Liu, K., Zhao, X., Tang, J., Zeng, W., Liao, J., Tian, F., ... & Dai, A. (2021. MOOPer: A Large-Scale Dataset of Practice-Oriented Online Learning. In *Knowledge Graph and Semantic Computing: Knowledge Graph Empowers New Infrastructure Construction: 6th China Conference, CCKS 2021, Guangzhou, China, November 4-7, 2021, Proceedings 6* (pp. 281-287). Springer Singapore.

Liu, P., Du, J., Xue, Z., & Li, A. (2022, July). Bi-convolution matrix factorization algorithm based on improved ConvMF. In *China Intelligent Networked Things Conference* (pp. 122-134). Singapore: Springer Nature Singapore..

Liu, Z., Chen, K., Song, F., Chen, B., Zhao, X., Guo, H., & Tang, R. (2023). AutoAssign+: Automatic Shared Embedding Assignment in streaming recommendation. *Knowledge and Information Systems*, 1-25.

Logesh, R., Subramaniyaswamy, V., D. Malathi,N. Sivaramakrishnan, and V. Vijayakumar, "Enhancing recommendation stability of collaborative filtering recom- mender system through bio-inspired clustering ensemble method," Neural Computing & Applications, vol. 32, no. 10, pp. 2141–2164, 2020.

Lops, P., Musto, C., & Polignano, M. (2023, June). Accountable Knowledge-aware Recommender Systems. In *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization* (pp. 306-308).

Luo, M., Chen, F., Cheng, P., Dong, Z., He, X., Feng, J., & Li, Z. (2020, April). Metaselector: Meta-learning for recommendation with user-level adaptive model selection. In *Proceedings of The Web Conference 2020* (pp. 2507-2513).

Ma, J., Zhang, Y., & Zhang, L. (2021). Discriminative subspace matrix factorization for multiview data clustering. Pattern Recognition, 111, 107676.

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR.[Internet]*, *9*(1), 381-386.

Mastroleo, M., Ugolotti, R., Mussi, L., Vicari, E., Sassi, F., Sciocchetti, F., ... & McIlroy, C. (2018). Automatic analysis of faulty low voltage network asset using deep neural networks. *The Journal of Engineering*, *2018*(15), 851-855.

Maulud, D., & Abdulazeez, A. M. (2020). A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, *1*(4), 140-147.

Mehrbakhsh Nilashi, Othman Ibrahim, Karamollah Bagherifard, (2018). A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques, *Expert Systems with Applications. 92, 507-520.*

Mehrer, J., Spoerer, C. J., Kriegeskorte, N., & Kietzmann, T. C. (2020). Individual differences among deep neural network models. *Nature communications*, *11*(1), 5725.

Michaud, Eric J., Ziming Liu, and Max Tegmark. "Precision machine learning." Entropy 25.1 (2023): 175.

Min, E., Guo X, Liu Q. (2018). A survey of clustering with deep learning: from the perspective of network architecture. *IEEE Access*, *6*, 39501-39514.

Mondal. R, and Bhowmik. B, "DeCS: A Deep Neural Network Framework for Cold Start Problem in Recommender Systems," *2022 IEEE Region 10 Symposium (TENSYMP).*

Moradi, R., Berangi, R., & Minaei, B. (2020). A survey of regularization strategies for deep models. *Artificial Intelligence Review*, *53*(6), 3947-3986.

Murad, D. F., Heryadi, Y., Isa, S. M., & Budiharto, W. (2020). Personalization of study material based on predicted final grades using multi-criteria user-collaborative filtering recommender system. *Education and Information Technologies, 25, 5655-5668*.

Namasudra, S., Dhamodharavadhani, S., & Rathipriya, R. (2023). Nonlinear neural network based forecasting model for predicting COVID-19 cases. *Neural processing letters*, 1-21.

Neto, C. Brito, M.; Lopes, V.; Peixoto, H.; Abelha, A.; Machado, J (2019). Application of data mining for the prediction of mortality and occurrence of complications for gastric cancer patients. *Entropy, 21*(12), 1163.

Nilashi, M., Ibrahim, O., & Bagherifard, K. (2018). A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. *Expert Systems with Applications, 92, 507-520.*

Nunez, V., Edward, R., Quintana, D., González C., Ruben; Isasi, Pedro; Herrera-Viedma, Enrique (2018). A recommender system based on implicit feedback for selective dissemination of eBooks. *Information Sciences*, *467*, 87-98.

Padakandla, S. (2021). A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR), 54*(6), 1-25.

Pal, N., & Dahiya, O. (2023, February). Analysis of Educational Recommender System Techniques for Enhancing Student's Learning Outcomes. In *2023 3rd International Conference on Innovative Practices in Technology and Management (ICIPTM)* (pp. 1-5). IEEE.

Pan, Y., He, F., & Yu, H. (2020). Learning social representations with deep auto-encoder for recommender system. *World Wide Web, 23*, 2259-2279.

Panda, D. K., & Ray, S. (2022). Approaches and algorithms to mitigate cold start problems in recommender systems: a systematic literature review. *Journal of Intelligent Information Systems, 59*(2), 341-366.

Papadakis, H., Papagrigoriou, A., Panagiotakis, C., Kosmas, E., & Fragopoulou, P. (2022). Collaborative filtering recommender systems taxonomy. *Knowledge and Information Systems, 64*(1), 35-74.

Park, H., Jeong, J., Oh, K. W., & Kim, H. (2023). Autoencoder-Based Recommender System Exploiting Natural Noise Removal. *IEEE Access, 11*, 30609-30618.

Patro, S. G. K., Mishra, B. K., Panda, S. K., Kumar, R., Long, H. V., Taniar, D., & Priyadarshini, I. (2020). A hybrid action-related K-nearest neighbour (HAR-KNN) approach for recommendation systems. *IEEE Access*, *8*, 90978-90991.

Paullada, A., Raji, I. D., Bender, E. M., Denton, E., & Hanna, A. (2021). Data and its (dis) contents: A survey of dataset development and use in machine learning research. *Patterns*, *2*(11).

Pawar, K., & Attar, V. Z. (2020). Assessment of autoencoder architectures for data representation. *Deep learning: concepts and architectures*, 101-132.

Pinaya, W. H. L., Vieira, S., Garcia-Dias, R., & Mechelli, A. (2020). Autoencoders. In *Machine learning* (pp. 193-208). Academic Press.

Pramod, A., Naicker, H. S., & Tyagi, A. K. (2021). Machine learning and deep learning: Open issues and future research directions for the next 10 years. *Computational analysis and deep learning for medical care: Principles, methods, and applications*, 463-490.

Priyadharshini, A., Chakraborty, S., Kumar, A., & Pooniwala, O. R. (2021, April). Intelligent crop recommendation system using machine learning. In *2021 5th international conference on computing methodologies and communication (ICCMC)* (pp. 843-848). IEEE.

Rama, K., Kumar, P., & Bhasker, B. (2021). Deep auto-encoders for feature learning with embeddings for recommendations: a novel recommender system solution. *Neural Computing and Applications*. *33*, 14167-14177.

Rendle, S., Krichene, W., Zhang, L., & Anderson, J. (2020, September). Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the 14th ACM Conference on Recommender Systems* (pp. 240-248).

Roelleke, T. (2022). *Information Retrieval Models: Foundations & Relationships*. Springer Nature.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., & Zhong, C. (2022). Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys*.

Sabiri, B., El Asri, B., & Rhanoui, M. (2022). Mechanism of Overfitting Avoidance Techniques for Training Deep Neural Networks. In *ICEIS (1)* (pp. 418-427)

Saleh, A. M., & Taqa, A. Y. (2023). A proposed User-Based Approach for eBooks Recommendation Using a Weighted Nearest Neighbor Technique. *Sinkron: jurnal dan penelitian teknik informatika*, *8*(3), 1316-1325.

Sapnken, F. E., Hamed, M. M., Soldo, B., & Tamba, J. G. (2023). Modeling energy-efficient building loads using machine-learning algorithms for the design phase. *Energy and Buildings*, *283*, 112807.

Sedhain, S., Menon, A., Sanner, K., & Xie. L. (2015,). Autorec: Auto-encoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web* (pp. 111-112).

Shambour, Q. (2021). A deep learning based algorithm for multi-criteria recommender systems. Knowledge-Based Systems. 211 (2021): 106545.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306.

Singh, P. K., Sinha, M., Das, S., & Choudhury, P. (2020). Enhancing recommendation accuracy of item-based collaborative filtering using Bhattacharyya coefficient and most similar item. *Applied Intelligence*, *50*, 4708-4731.

Sivaramakrishnan, N., Subramaniyaswamy, V., Viloria, A. *et al.* A deep learning-based hybrid model for recommendation generation and ranking. *Neural Comput & Applic* **33**, 10719–10736 (2021). https://doi.org/10.1007/s00521-020-04844-4

Stitini, O., García-Magariño, I., Kaloun, S., & Bencharef, O. (2023). Towards Ideal and Efficient Recommendation Systems Based on the Five Evaluation Concepts Promoting Serendipity. *Journal of Advances in Information Technology*, *14*(4), 701-717.

Sun, C., Fan, X., & Zhao, D. (2023). Jpeg decoding with nonlinear inverse transform network and progressive recurrent residual network. *IEEE Transactions on Consumer Electronics*, *69*(3), 499-509.

Sun, J., Ying, R., Jiang, Y., He, J., & Ding, Z. (2020). Leveraging friend and group information to improve social recommender system. *Electronic Commerce Research*, *20*, 147-172.

Tegene, A., Liu, Q., Gan, Y., Dai, T., Leka, H., & Ayenew, M. (2023). Deep Learning and Embedding Based Latent Factor Model for Collaborative Recommender Systems. *Applied Sciences*, *13*(2), 726.

Tian, Y., Zhao, X., & Huang, W. (2022). Meta-learning approaches for learning-to-learn in deep learning: A survey. *Neurocomputing*, *494*, 203-223.

Tulbure, A. A., Tulbure, A. A., & Dulf, E. H. (2022). A review on modern defect detection models using DCNNs–Deep convolutional neural networks. *Journal of Advanced Research*, *35*,

Verbeeck, N., Caprioli, R. M., & Van de Plas, R. (2020). Unsupervised machine learning for exploratory data analysis in imaging mass spectrometry. *Mass spectrometry reviews*, *39*(3), 245-291.

Vijayakumar, S., & Deepak, G. (2022). A Deep Learning Integrated Ontology Driven Model for E-Commerce Product Recommendation for Improved Machine Intelligence. In *International Conference on Digital Technologies and Applications* (pp. 209-218). Cham: Springer International Publishing.

Volkovs, M., Yu, G., & Poutanen, T. (2017). Dropoutnet: Addressing cold start in recommender systems. Advances in neural information processing systems. 30.

Wang, D., Liang, Y., Xu, D., Feng, X., & Guan, R. (2018). A content-based recommender system for computer science publications. Knowledge-Based Systems. *157*, 1-9.

Wang, K., Zhang, T., Xue, T., Lu, Y., & Na, S. G. (2020). E-commerce personalized recommendation analysis by deeply-learned clustering. *Journal of Visual Communication and Image Representation*, *71*, 102735

Wang, W., Feng, F., He, X., Nie, L., & Chua, T. S. (2021, March). Denoising implicit feedback for recommendation. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 373-381).

Wang, X., & Yang, B. (2018, April). STMF: A sentiment topic matrix factorization model for recommendation. In *2018 3rd International Conference on Computer and Communication Systems (ICCCS)* (pp. 444-447). IEEE.

Wang, Y., Song, Z., and Xiao, C. (2018). Collaborative filtering recommendation algorithm based on improved clustering and matrix factorization. ." *Journal of Computer Applications* 38, no. 4 (2018): 1001.

Wang, Y., Wang, D., Geng, N., Wang, Y., Yin, Y., & Jin, Y. (2019). Stacking-based ensemble learning of decision trees for interpretable prostate cancer detection. *Applied Soft Computing*, *77*, 188-204.

Wegmeth, L. (2022). The Impact of Feature Quantity on Recommendation Algorithm Performance: A MovieLens-100K Case Study. *arXiv preprint arXiv:2207.08713*.

Wen, Y., Liang, X., Huang, W., Wei, W., & Deng, Z. (2020, October). Chinese poetry and couplet automatic generation based on self-attention and multi-task neural network model. In *International Symposium on Artificial Intelligence and Robotics 2020* (Vol. 11574, pp. 112-123). SPIE

Wu, Y., Su, L., Wu, L., & Xiong, W. (2023). FedDeepFM: A Factorization Machine-Based Neural Network for Recommendation in Federated Learning. *IEEE Access*.

Xi, W. D., Huang, L., Wang, C. D., Zheng, Y. Y., & Lai, J. H. (2021). Deep rating and review neural network for item recommendation. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(11), 6726-6736.

Xie, J., Girshick,R., and Farhadi,A. (2016). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning* (pp. 478-487). PMLR

Xu, H., Kashef, R. F., De Sterck, H., & Sanders, G. (2022). Efficient Algebraic Multigrid Methods for Multilevel Overlapping Coclustering of User-Item Relationships. *INFORMS Journal on Computing*, *34*(3), 1587-1605.

Xu, X., Fang, Z., Yu, Q., Huang, R., Fan, C., Li, Y., ... & Non, N. (2022, July). Gating-adapted wavelet multiresolution analysis for exposure sequence modeling in ctr prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1890-1894).

Xue, F., X. He, X. Wang, J. Xu, K. Liu, and R. Hong, (2019). Deep item-based collaborative filtering for top-n recommendation," ACM Transactions on Information Systems. *(TOIS)*, *37*(3), 1-25.

Yan, S., Chen, X., Huo, R., Zhang, X., & Lin, L. (2020, October). Learning to build user-tag profile in recommendation system. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (pp. 2877-2884).

Yedukondalu, G., Joshi, J., Nidumolu, K. D., Gopi, A., & Rajavikram, G. (2023). Ensemble Deep Learning Models for Collaborative Filtering Recommendations. *International Journal of Intelligent Systems and Applications in Engineering*, *11*(6s), 358-369.

Yin, Y., Chen, L., y. xu, and J. Wan, (2018). "Location-aware service recommendation with enhanced probabilistic matrix factorization," IEEE Access. , *6*, 62815-62825.

Yun, Y., Hooshyar, D., Jo, J., and H. Lim, (2018). "Developing a hybrid collaborative filtering recommendation system with opinion mining on purchase review. *Journal of Information Science*, *44*(3), 331-344.

Zade, M., Lumpp, S. D., Tzscheutschler, P., & Wagner, U. (2022). Satisfying user preferences in community-based local energy markets—Auction-based clearing approaches. *Applied Energy*, *306*, 118004.

Zhang, L., Luo, T., Zhang, F., and Wu, Y., (2018). A recommendation model based on deep neural network. *IEEE Access*, *6*, 9454-9463.

Zhang, Q., Lu, J., & Jin, Y. (2021). Artificial intelligence in recommender systems. *Complex & Intelligent Systems*, *7*(1), 439-457.

Zhang, T., Huang, Y., Liao, H., & Liang, Y. (2023). A hybrid electric vehicle load classification and forecasting approach based on GBDT algorithm and temporal convolutional network. *Applied Energy*, *351*, 121768.

Zhang, W., Liu, Y., Z. Liu, S. Wang, Z. Zhen, and Y. Li, (2020.). Research on fault monitoring technology of distribution network based on fuzzy association rules mining. In *Journal of Physics: Conference Series* (Vol. 1626, No. 1, p. 012072). IOP Publishing.

Zhang, Z., Xu, S., Guo, L., & Lian, W. (2022, November). Multi-modal Vibrational Auto-Encoder Model for Micro-Video Popularity Prediction. In *Proceedings of the 8th International Conference on Communication and Information Processing*. (pp. 9-16).

Zhou, Kai & Luo, Lan & Chen, Tianlin. (2023). The Influence of Online Media on College Students' Self-identity in Mobile Learning Environment. 10.2991/978-94-6463-172-2_126.

Zhu, F., Wang, Y., Chen, C., Liu, G., Orgun, M., & Wu, J. (2020). A deep framework for cross-domain and cross-system recommendations. *arXiv preprint arXiv:2009.06215*.

Zhu, L., Q. Hu, L. Zhao, and J. Yang, "Collaborative filtering algorithm based on rating preference and item attributes," Computer Science, vol. 47, no. 04, pp. 67–73, 2020.

APPENDICES

Appendix I: Data preparation

```
[ ]  import pandas as pd
     import math
     import numpy as np
     import seaborn as sns
     from sklearn.model_selection import train_test_split
```

```
[ ]  r_cols = ['user_id', 'movie_id', 'rating', 'unix_timestamp']
     ratings = pd.read_csv('/content/u.data', sep='\t', names=r_cols, encoding='latin-1')
```

```
[ ]  u_cols = ['user_id', 'age', 'sex', 'occupation', 'zip_code']
     users = pd.read_csv('/content/u.user', sep='|', names=u_cols,  encoding='latin-1')
```

```
[ ]  i_cols = ['movie id', 'movie title' ,'release date','video release date', 'IMDb URL', 'unknown', 'Action', 'Adventure',
      'Animation', 'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
      'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']
     items = pd.read_csv('/content/u.item', sep='|', names=i_cols,
      encoding='latin-1').rename(columns={'movie id': 'movie_id'})
```

```
[ ]  #combine data
```

```
[ ]  ratings_users_merged = pd.merge(ratings, users, on='user_id')
```

```
[ ]  combined_data = pd.merge(ratings_users_merged, items, on='movie_id')
```

```
[ ]  combined_data.to_csv('combined_movielens_data.csv', index=False)
```

Appendix II: Model Development Source Code

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.layers import Input, Dense, Dropout, Embedding, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Concatenate, Flatten
from sklearn.preprocessing import MinMaxScaler
from datetime import datetime
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
```

```
[ ] # End the timer
    end_time = time.time()

    # Calculate the duration
    training_time = end_time - start_time

    print(f"Time taken to train the model: {training_time:.2f} seconds")
```

```
Time taken to train the model: 24.65 seconds
```

```
[ ] loss, accuracy = autoencoder.evaluate([test_data, test_data, test_data], test_data)
    print("Test Loss:", loss)
    print("Test Accuracy:", accuracy)
```

```
autoencoder.compile(optimizer= 'adam', loss='mse', metrics=['accuracy'])
```

Appendix III: Model Evaluation Source Code

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.layers import Input, Dense, Dropout, Embedding, Flatten,
Concatenate
from tensorflow.keras.models import Model
from datetime import datetime
import tensorflow as tf
import matplotlib.pyplot as plt

# Load and preprocess data data
r_cols = ['userId', 'movieId', 'rating', 'timestamp', 'age', 'sex', 'occupation', 'location']
merged_data = pd.read_csv('ratingsandusers_modified.csv')

max_rating = merged_data['rating'].max()
merged_data['rating'] /= max_rating

user_encoder = LabelEncoder()
movie_encoder = LabelEncoder()
location_encoder = LabelEncoder()

merged_data['userId'] = user_encoder.fit_transform(merged_data['userId'])
merged_data['movieId'] = movie_encoder.fit_transform(merged_data['movieId'])
merged_data['location'] = location_encoder.fit_transform(merged_data['location'])

user_item_matrix = merged_data.pivot_table(index='userId', columns='movieId',
values='rating').fillna(0)

train_data, test_data = train_test_split(user_item_matrix.values, test_size=0.2,
random_state=42)
```

```
num_users = user_item_matrix.shape[0]
num_movies = user_item_matrix.shape[1]

timestamp_input = Input(shape=(1,), name='timestamp_input')
location_input = Input(shape=(1,), name='location_input')
user_input = Input(shape=(num_movies,), name='user_input')
# Embedding
timestamp_embedding = Embedding(input_dim=24,
output_dim=10)(timestamp_input)
timestamp_embedding = Flatten()(timestamp_embedding)
location_embedding = Embedding(input_dim=len(location_encoder.classes_),
output_dim=10)(location_input)
location_embedding = Flatten()(location_embedding)

user_flat = Flatten()(user_input)
concatenated = Concatenate()([timestamp_embedding, location_embedding,
user_flat])
encoded = Dense(64, activation='relu')(concatenated)
encoded = Dropout(0.2)(encoded)
encoded = Dense(32, activation='relu')(encoded)
decoded = Dense(64, activation='relu')(encoded)
decoded = Dropout(0.2)(decoded)
decoded = Dense(num_movies, activation='sigmoid')(decoded)

autoencoder = Model(inputs=[timestamp_input, location_input, user_input],
outputs=decoded)
autoencoder.compile(optimizer='adam', loss='mse')

# Train model
history = autoencoder.fit(
    [train_data[:, 0:1], train_data[:, 0:1], train_data],  # Pass timestamp and location
separately
    train_data,
```

```
    epochs=100,
    batch_size=64,
    validation_split=0.2
)
# Evaluate the model
loss = autoencoder.evaluate([test_data[:, 0:1], test_data[:, 0:1], test_data], test_data)
print("Test Loss:", loss)
# Predict function
def predict_item(location, hour):
    try:
        # Encode location
        encoded_location = location_encoder.transform([location])[0]
    except ValueError:
        print("Error: Invalid input. Please enter a valid location.")
        return None
    input_data = {
        'timestamp_input': np.array([hour]).reshape(1, 1),
        'location_input': np.array([encoded_location]).reshape(1, 1),
        'user_input': np.zeros((1, num_movies))  # Use the correct num_movies value
    predicted_ratings = autoencoder.predict(input_data)
    # Get the index of the highest predicted rating
    predicted_item_index = np.argmax(predicted_ratings)
        predicted_item = movie_encoder.inverse_transform([predicted_item_index])[0]
        return predicted_item
    except ValueError:
        print("Error: Could not decode predicted item index.")
        return None
def main():
    # Print model input shapes
    print("Model input shapes:")
    for layer in autoencoder.layers:
        if isinstance(layer, tf.keras.layers.InputLayer):
            print(f"{layer.name}: {layer.input_shape}")
```

```
    location = input("Enter location: ")
    hour = input("Enter timestamp (hour, 0-23): ")
    try:
        hour = int(hour)
        if hour < 0 or hour > 23:
            raise ValueError
    except ValueError:
        print("Error: Please enter a valid hour (0-23).")
        return
    predicted_item = predict_item(location, hour)
    if predicted_item is not None:
        print(f"The predicted item for location '{location}' at hour '{hour}' is:
{predicted_item}")
    else:
        print("Could not predict an item for the given input.")
    # Print additional debug information
    print("\nDebug Information:")
    print("Number of movies:", num_movies)

# Run the main function
if __name__ == "__main__":
    main()
```

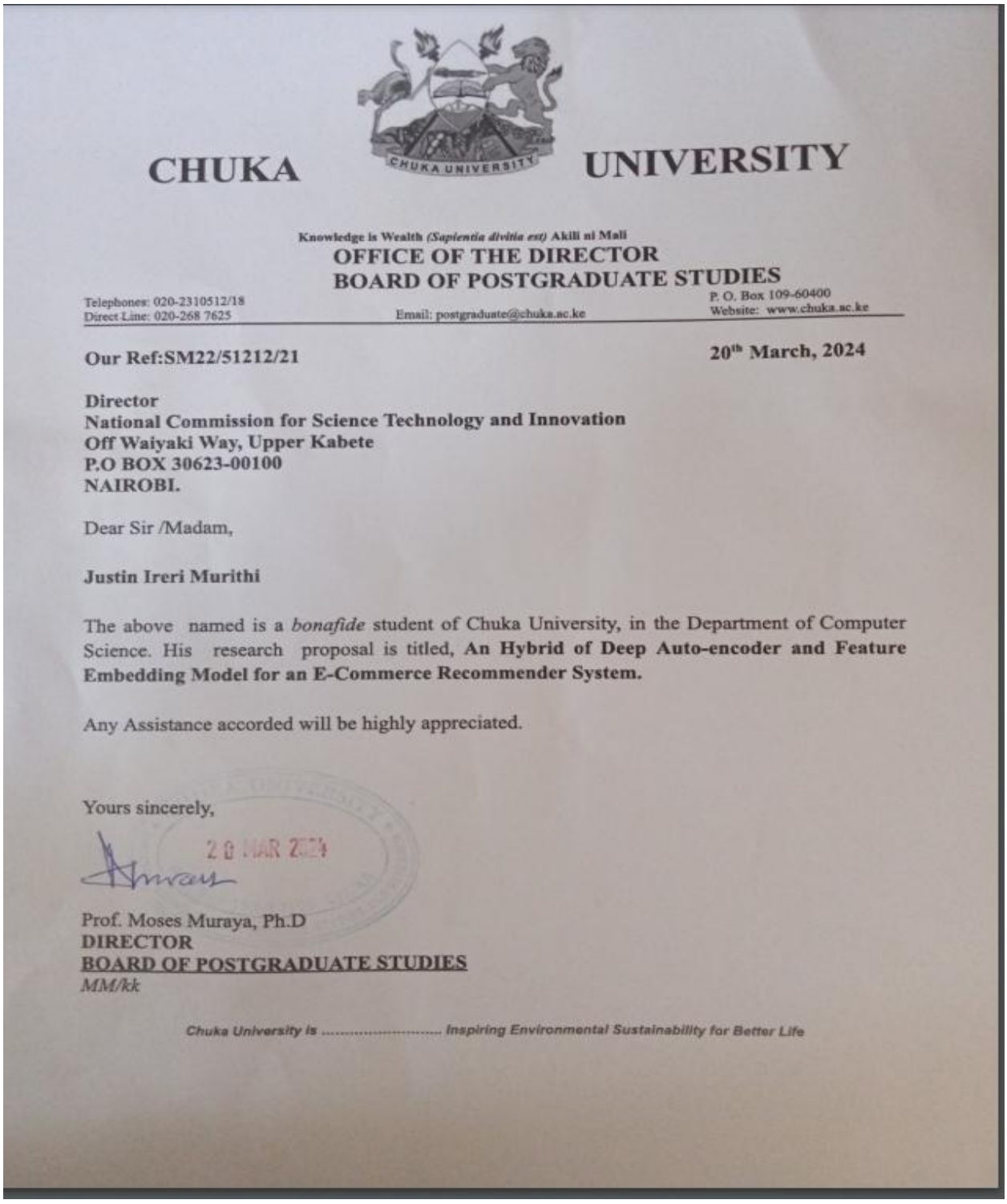
Appendix IV: BUDGET

ACTIVITY/ITEM	Quantity	COST PER UNIT	Total Cost
Proposal writing and preparation			
Laptop	1	129,000	129,000
Printing	6(30 pages)	10	1,800
Binding	6	50	300
Internet	20GB	500	10,000
subtotal			**141,100**
Departmental Defense			
Printing	6(40 pages)	10	2,400
Binding	6	50	300
Internet	20GB	500	10,000
subtotal			**12700**
Faculty Defense			
Printing	6(50 pages)	10	30,000
Binding	6	50	300
Internet	20GB	500	10,000
subtotal			**43,000**
Post graduate level			
Printing	6(59 pages)	10	3540
Binding	6	50	300
Internet	50 GB	100	2500
subtotal			**13840**
Model Development, Testing and validation			
G.P.U	15 hrs.	500	7500
Printing	6(68 pages)	10	4080
Binding	6	50	300
Internet	100GB	100	10,000
subtotal			**14380**
Final Thesis			
Printing	11 (72 pages)	10	7920
Binding	11 copies	50	550
Journal publication	1	20,000	20,000
Conference presentation	2	5,000	10,000
subtotal			**38470**
Total			259,640
Contingencies			36,510
Grand Total			**300,000**

Appendix V: WORK PLAN

ACTIVITY	SEPT OCT 2021	NOV DEC 2021	JUNE July 2022	DEC 2023	JAN 2024	MAY 2023	SEPT 2024	OCT 2024
Writing and presentation of concept paper								
Proposal Writing and presentation								
Departmental Defense preparation and presentation								
Defense at the faculty level								
Proposal submission to Post-graduate level								
Model development ,Training and validation								
Thesis writing, submission and defense at post graduate school								
Final Submission of thesis								

Appendix VI: Chuka University Introductory Letter



CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

OFFICE OF THE DIRECTOR
BOARD OF POSTGRADUATE STUDIES

Telephones: 020-2310512/18
Direct Line: 020-268 7625
Email: postgraduate@chuka.ac.ke
P. O. Box 109-60400
Website: www.chuka.ac.ke

Our Ref:SM22/51212/21 **20th March, 2024**

Director
National Commission for Science Technology and Innovation
Off Waiyaki Way, Upper Kabete
P.O BOX 30623-00100
NAIROBI.

Dear Sir /Madam,

Justin Ireri Murithi

The above named is a *bonafide* student of Chuka University, in the Department of Computer Science. His research proposal is titled, **An Hybrid of Deep Auto-encoder and Feature Embedding Model for an E-Commerce Recommender System.**

Any Assistance accorded will be highly appreciated.

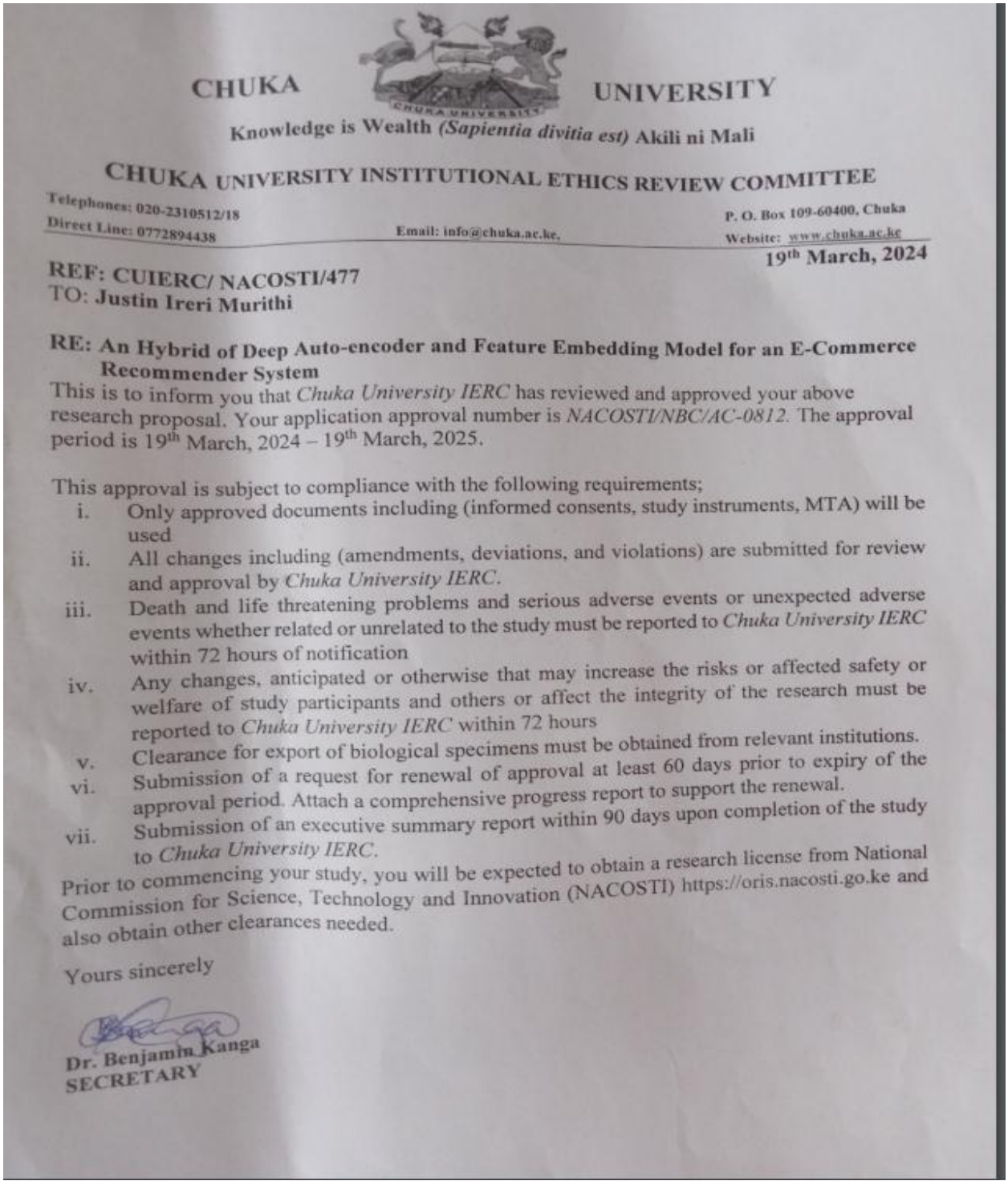
Yours sincerely,

20 MAR 2024

Prof. Moses Muraya, Ph.D
DIRECTOR
BOARD OF POSTGRADUATE STUDIES
MM/kk

Chuka University is ..... Inspiring Environmental Sustainability for Better Life

Appendix VII: Ethics Review Letter



CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

CHUKA UNIVERSITY INSTITUTIONAL ETHICS REVIEW COMMITTEE

Telephones: 020-2310512/18
Direct Line: 0772894438
Email: info@chuka.ac.ke,
P. O. Box 109-60400, Chuka
Website: www.chuka.ac.ke

19th March, 2024

REF: CUIERC/ NACOSTI/477
TO: **Justin Ireri Murithi**

RE: An Hybrid of Deep Auto-encoder and Feature Embedding Model for an E-Commerce Recommender System

This is to inform you that *Chuka University IERC* has reviewed and approved your above research proposal. Your application approval number is *NACOSTI/NBC/AC-0812.* The approval period is 19th March, 2024 – 19th March, 2025.

This approval is subject to compliance with the following requirements;

i. Only approved documents including (informed consents, study instruments, MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by *Chuka University IERC.*
iii. Death and life threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to *Chuka University IERC* within 72 hours of notification
iv. Any changes, anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to *Chuka University IERC* within 72 hours
v. Clearance for export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days upon completion of the study to *Chuka University IERC.*

Prior to commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology and Innovation (NACOSTI) https://oris.nacosti.go.ke and also obtain other clearances needed.

Yours sincerely

Dr. Benjamin Kanga
SECRETARY

Appendix VIII: NACOSTI License

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: **892969**

Date of Issue: **17/April/2024**

RESEARCH LICENSE

This is to Certify that Mr.. JUSTIN MURITHI IRERI of Chuka University, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Tharaka-Nithi on the topic: AN HYBRID OF DEEP AUTO-ENCODER AND FEATURE EMBEDDING MODEL FOR AN E-COMMERCE RECOMMENDER SYSTEM for the period ending : 17/April/2025.

License No: **NACOSTI/P/24/34375**

892969

Applicant Identification Number

Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions