Tampere University

Shaheer Ghani Imam

FAIRNESS-AWARE RECOMMENDER SYSTEMS: EXTENDING THE SOCRATE FRAMEWORK WITH NEURAL AND GENERATIVE MODELS

Master's Thesis
Faculty of Information Technology and Communication Sciences
Examiner: Konstantinos Stefanidis
November 2025

ABSTRACT

Shaheer Ghani Imam: Fairness-Aware Recommender Systems: Extending the SoCRATe Framework with Neural and Generative Models
Master's Thesis
Tampere University
Faculty of Information Technology and Communication Sciences
November 2025

Recommender systems play a vital role in helping users discover relevant items in domains such as e-commerce, media, and crowdsourcing. However, most existing approaches assume that items are always available in unlimited quantities. In reality, recommendations are often made under limited availability, where assigning an item to one user may restrict its access for others. This challenge raises questions of fairness, as users may experience unequal satisfaction depending on the timing and allocation of recommendations.

To address this, the SoCRATe framework was developed as a fairness-aware recommendation system that compensates users for such disparities across multiple iterations. This thesis extends the original SoCRATe framework by integrating two modern recommender models Graph Neural Networks (GNNs) and Variational Autoencoders (VAEs) alongside the existing KNN baseline. The goal is to explore how more expressive learning models influence the fairness–utility trade-off in recommendation settings with limited availability.

The study builds on SoCRATe's iterative workflow, where users receive recommendations, consume items, and accumulate fairness losses that guide compensation in later rounds. The extended framework introduces a fairness-aware GNN, which learns user and item embeddings through graph propagation, and a VAE-based recommender, which captures latent preference distributions for improved generalization under sparse data. Both models are evaluated within SoCRATe's orchestration logic using preference-driven and round-robin strategies.

Experiments were conducted on synthetic and real-world datasets to compare the performance of KNN, GNN, and VAE models in terms of fairness and accuracy. Results show that while GNNs and VAEs improve overall recommendation quality, their impact on fairness varies depending on data sparsity, model sensitivity, and the chosen allocation strategy. The findings highlight the importance of balancing representation learning with fairness-aware orchestration to achieve equitable outcomes in recommendation systems.

This work contributes to a better understanding of how modern deep learning recommenders interact with fairness-aware frameworks like SoCRATe. It provides both theoretical and practical insights for designing recommendation systems that remain efficient while ensuring fair treatment of users under real-world availability constraints.

Keywords: Recommender Systems, Fairness, Graph Neural Networks (GNN), Variational Autoencoders (VAE), SoCRATe, Limited Availability, Fairness–Utility Trade-off.

1. USE OF AI IN THESIS

I have utilised AI tools in my thesis:

☐ No
☒ Yes

The AI tools utilised in my thesis and their purposes are described below:

Names and versions of AI tools:

1. **ChatGPT**: For brainstorming, drafting and refining text.

2. **GPT-5 Model**: For code refactoring and boiler plate code generation

Purpose of using AI tools: AI tools were used to support both the research process and the tool development.

- ChatGPT assisted in brainstorming scenarios where the deep learning models would overfit, and refining content for the thesis
- GPT-5 was used to refactor code to avoid repeatability in the source code and generating simple boiler plate code for the web application

Sections where AI tools were used:

- Section 1 and 2 – for drafting and refining discussion using ChatGPT
- Section 4.3 and 4.4 – for drafting the algorithmic flow of deep learning recommenders and integration in existing code alongside refining discussion using ChatGPT.
- Section 5 – For refining the discussion of results among different recommenders.

I acknowledge that I am fully responsible for the entire content of my thesis, including the parts generated by AI, and accept accountability for any violations of ethical standards in publications.

PREFACE

This thesis was written as part of my Master of Science studies in the Faculty of Information Technology and Communication Sciences at Tampere University. The research was carried out between August 2025 and November 2025 and examines the role of modern deep learning techniques in improving fairness within multi-session recommender systems operating under limited item availability. The motivation behind this work arose from a central question: Can more expressive models help mitigate user-level disparities and enhance the fairness–utility balance in existing architectures?

I would like to express my deepest gratitude to my supervisor, **Professor Konstantinos Stefanidis**, for introducing me to this research domain not only as a technical challenge, but as an essential pursuit for building responsible and equitable recommender systems. His guidance, constructive feedback, and steady encouragement were invaluable throughout the entire thesis process. His emphasis on fairness as a core design principle greatly shaped my understanding of recommender systems and influenced the direction of this study.

I am also sincerely thankful to **Dr. Maria Stratigi** for her insightful comments and suggestions, which helped refine the clarity, rigor, and overall quality of this work. Her perspective and feedback were instrumental in strengthening the final outcome.

Finally, I would like to thank my family and friends for their patience, support, and confidence in me during this journey. Their encouragement made the completion of this thesis possible.

Tampere, Finland, 25 November 2025

Shaheer Ghani Imam

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SoCRATe	System for Compensating Recommendations with Availability and Time
KNN	k-Nearest Neighbor
GNN	Graph Neural Network
VAE	Variational AutoEncoders
GPT-5	Generative Pre-trained Transformer 5
RNN	Recurrent Neural Network
SASRec	Self Attention Sequential Recommendation
BERT4Rec	Bidirectional Encoder Representations for Recommendations
KL	Kullback-Leibler
RL	Reinforcement Learning
GCF	Generative Collaborative Filtering
CF	Collaborative Filtering
NCF	Neural Collaborative Filtering
BPR	Bayesian Personalized Ranking
RR	Round Robin
PD	Preference Driven
Adam	Adaptive Moment Estimation Optimizer
GPU	Graphic Processing Unit

1. INTRODUCTION

1.1 Background and Motivation

In the era of information abundance, recommender systems have emerged as essential tools for filtering and personalizing the vast amount of content available to users. From suggesting movies on streaming platforms, recommending products in e-commerce, to allocating tasks in crowdsourcing environments, these systems play a central role in enhancing user experience, increasing engagement, and driving business value. At their core, recommender systems leverage historical user–item interactions, demographic attributes, or content features to predict user preferences and generate ranked lists of items most likely to be relevant. Traditional approaches to recommendation can broadly be categorized into collaborative filtering, which infers preferences from patterns of user item interactions; content-based methods, which rely on item attributes; and hybrid approaches, which combine both paradigms to overcome individual limitations.

While the effectiveness of recommender systems has been well established, they are typically developed under the assumption of unlimited item availability. In practice, how ever, many recommendation scenarios involve scarce or limited resources. Examples include suggesting physical goods with restricted stock, recommending time-bound services such as event tickets, or distributing a finite number of tasks among workers in crowdsourcing platforms. In such settings, the recommendations offered to one user may directly reduce the availability of items for others, leading to competition among users. This competition creates challenges in maintaining both accuracy (i.e., how well the system predicts preferences) and fairness (i.e., how evenly opportunities are distributed across the user population).

The limitations of traditional recommenders in such constrained environments have motivated the development of frameworks that explicitly account for item availability and user fairness. Among these, the SoCRATe framework (System for Compensating Recommendations with Availability and Time) has recently been introduced as a promising solution [1,2]. SoCRATe operates as an adaptation layer over existing recommenders, continuously monitoring user interactions and computing a cumulative

loss metric—a measure of the discrepancy between the recommendations provided and the ideal recommendations a user could have received in the absence of availability constraints. By leveraging this metric, SoCRATe dynamically compensates users who were previously disadvantaged, thereby promoting long-term fairness without sacrificing recommendation quality.

The framework achieves this compensation through two main strategies: preference driven compensation, which prioritizes users with the highest cumulative loss, and round-robin compensation, which distributes recommendations cyclically among users based on their loss levels [1]. These strategies allow SoCRATe to balance the trade-off between accuracy and fairness over multiple recommendation sessions. Moreover, SoCRATe is designed to be agnostic to the underlying recommendation algorithm, making it a generalizable solution that can be integrated with both classical methods, such as k-nearest neighbors (KNN), and modern machine learning approaches.

The motivation for extending SoCRATe with more advanced recommender models arises from the increasing complexity of user–item interaction data. Traditional algorithms like KNN, while effective in simple settings, may struggle to capture the intricate patterns and latent structures present in large-scale recommendation tasks. By incorporating more expressive models such as Graph Neural Networks (GNNs) and Variational Autoencoders (VAEs), it becomes possible to better model higher-order relationships and uncertainty in user preferences, thereby improving both the effectiveness and fairness of recommendations within the SoCRATe framework.

In summary, recommender systems are indispensable in modern applications, yet they face unique challenges under conditions of limited availability. SoCRATe addresses these challenges by compensating users for earlier losses and ensuring fairness across sessions. The motivation of this thesis is to extend the SoCRATe framework with advanced deep learning-based recommenders, thereby exploring how such models impact the trade-off between fairness and accuracy in constrained recommendation environments.

1.2 Problem Statement and Research Questions

Recommender systems have become an essential part of how people interact with digital platforms. They help users discover music, movies, products, and even online tasks. Most of these systems assume that items can always be recommended without

restrictions. But in reality, this is often not the case. Items can be limited in quantity (like tickets to a concert), time-dependent (such as promotional offers), or mutually exclusive (once one user consumes a resource, it is no longer available to others). When these constraints exist, a new challenge appears: how to ensure that recommendations remain not only accurate but also fair across different users.

The SoCRATe framework was designed to tackle this specific challenge. Its main idea is that if a user misses out on a good recommendation because of limited availability, the system should recognize this “loss” and compensate the user in later recommendation rounds. SoCRATe introduces fairness into the recommendation process by balancing accuracy and user satisfaction over time. Up to now, SoCRATe has mostly been tested with simpler algorithms like KNN. While KNN is useful and efficient, it can not always capture complex structures in user–item interactions, especially when data is sparse or high-dimensional.

Modern recommendation research is moving towards more expressive models such as Graph Neural Networks (GNNs) and Variational Autoencoders (VAEs). GNNs are powerful in modeling graph-structured data, which naturally fits user–item interactions represented as bipartite graphs. VAEs, on the other hand, are strong at learning latent factors and handling uncertainty in preferences. These models have shown promise in recommendation tasks, but it is unclear how they will perform inside a fairness-driven framework like SoCRATe. For example, they might improve the quality of recommendations, but they could also unintentionally amplify bias.

This thesis therefore addresses the problem of extending SoCRATe with GNN and VAE recommenders and systematically evaluating how they affect fairness, utility, and scalability. The work is not only about adding new algorithms but also about understanding the interaction between advanced models and SoCRATe’s fairness mechanisms. Another important aspect is comparing the performance of these models against an Oracle baseline, which uses brute-force search to establish theoretical upper and lower bounds for fairness and accuracy.

To answer this problem, the thesis is guided by the following research questions

RQ1: How does the choice of recommender algorithm (KNN, GNN, VAE) influence the fairness–utility trade-off within the SoCRATe framework under limited item availability?

Explanation: This question explores whether using more expressive models like GNN and VAE can enhance fairness without sacrificing recommendation accuracy. The aim is to examine if these models are able to distribute item utility more evenly among users compared to the baseline KNN recommender. It also considers whether model complexity introduces new types of bias or if deeper representations can lead to fairer allocations over time.

RQ2: How do SoCRATe's fairness-aware allocation strategies (preference-driven, round-robin, and adoption models) interact with advanced recommender outputs to influence user compensation and fairness over multiple iterations?

Explanation: This question focuses on the relationship between the recommender's output and SoCRATe's orchestration logic. The same recommendation lists may lead to different fairness outcomes depending on how items are allocated or rotated between users. The goal is to understand whether deep recommenders alter the fairness dynamics when paired with SoCRATe's compensation strategies and to what extent these mechanisms can balance satisfaction across users in later iterations.

RQ3: To what extent do advanced recommenders (GNN, VAE) approach the fairness and utility bounds of the Oracle brute-force allocation, and how well do these effects generalize across synthetic and real-world datasets?

Explanation: This question investigates whether the use of GNN and VAE allows SoCRATe to approximate the theoretically optimal fairness and utility achieved by the Oracle allocation. It also evaluates the consistency of results across different dataset types, testing whether improvements are limited to synthetic environments or generalize to real-world data such as Amazon product or crowdsourcing datasets.

RQ4: What are the computational and scalability implications of integrating GNN and VAE models into SoCRATe, and how sensitive are fairness outcomes to their hyperparameter configurations?

Explanation: This question assesses the practical cost of extending SoCRATe with deep models. It evaluates whether the increased training time and computational complexity of GNN and VAE are justified by their fairness and accuracy improvements. Additionally, it examines the sensitivity of fairness and efficiency results to model hyperparameters such as embedding dimension, learning rate, τ (tau), and β (beta), providing insights into how stable and reproducible fairness results are across configurations.

1.3 Key Concepts and Definitions

This section introduces the key concepts used throughout the thesis and provides concise definitions supported by relevant literature. These terms establish a common foundation for the methodological discussions and experimental evaluations that follow.

1.3.1 Recommender Systems

A recommender system is a software tool and technique that aims to suggest items of interest to users by leveraging past interactions, user preferences, or item attributes. They are commonly categorized into collaborative filtering, which infers preferences from user–item interaction patterns; content-based methods, which use item features; and hybrid approaches, which combine both paradigms [8].

1.3.2 Fairness in Recommender System

Fairness in recommender systems refers to the extent to which recommendations provide equitable outcomes across different users or groups, without systematically favoring some at the expense of others. In constrained settings, fairness often means reducing the disparity in utility distribution when users compete for limited resources [9]. In SoCRATe, fairness is measured by the standard deviation of cumulative loss (S) across users, which captures how unevenly opportunities are distributed.

1.3.3 SoCRATE Framework

The System for Compensation Recommendations with Availability and Time (SoCRATe) is a framework design to handle recommenation scenarios where iterms are limited in availability. It works as an adaption layer over exisiting recommenders, monitoring user interactions and applying compensation strategies to balance fairness and utility across multiple sessions [10]. Its modular design makes it agnostic to the underlying recommender, allowing it to integrate both traditional and advanced models.

1.3.4 K-Nearest Neighbours Recommender

The KNN method is a neighborhood-based collaborative filtering approach. It predicts user preferences by identifying the k most similar users (user-based) or items (item-based) and aggregating their preferences. While computationally simple and interpretable, KNN often struggles with data sparsity and cannot exploit complex relationships beyond direct similarity [11].

1.3.5 Graph Neural Network

Graph Neural Networks are deep learning models specifically designed for graph structured data. They operate by iterative message passing, where each node updates its representation by aggregating information from its neighbors. This allows GNNs to capture higher-order relationships in user–item bipartite graphs, making them highly effective for recommendation tasks [3].

1.3.6 Variational Autoencoders

VAEs are generative latent variable models that encode input data into a latent distribution and then reconstruct it, combining a reconstruction objective with a Kullback–Leibler (KL) divergence regularizer. In recommendation, VAEs have been adapted for collaborative filtering, where they learn probabilistic representations of user preferences and provide robustness to data sparsity [6].

1.3.7 Utility-Fairness Trade Off

This concept captures the balance between recommendation accuracy and equitable distribution. Utility typically refers to how well the recommendations match user preferences, while fairness refers to how evenly the utility is distributed among users. In SoCRATe, this trade-off is explicitly measured through metrics of cumulative loss (L) for accuracy and its standard deviation (S) for fairness [9].

1.4 Motivation Behind GNN and VAE

The SoCRATe framework has so far been evaluated mainly with traditional recommenders such as k-nearest neighbors (KNN). While simple and efficient, KNN has important limitations: it relies only on direct similarity among users or items, struggles with data sparsity, and does not fully capture higher-order relations in user-item interactions. To advance the capabilities of SoCRATe, there is strong motivation to explore more expressive recommendation models that can better represent complex interaction structures and uncertain user preferences. In this work, two such models are considered: Graph Neural Networks (GNNs) and Variational Autoencoders (VAEs).

1.4.1 Graph Neural Network

Recommender systems are naturally modeled as bipartite graphs, where nodes represent users and items, and edges represent interactions such as ratings or clicks. GNNs are designed to learn from this graph structure through message passing, where

in formation is iteratively aggregated from a node's neighbors to capture both direct and indirect dependencies.

This capability addresses several shortcomings of traditional recommenders:

- **Capturing higher-order connectivity:** Unlike KNN, which only considers local similarity, GNNs can exploit multi-hop relations. For example, if user A is connected to items also consumed by user B, who in turn is connected to user C, the GNN can learn useful signals even though A and C have no direct overlap [3].
- **Integration of side information:** GNNs can seamlessly incorporate auxiliary signals such as item categories, tags, or social links, which often appear in real recommendation scenarios [4].
- **State-of-the-art results in practice:** Large-scale systems such as Pinterest have already demonstrated the success of GNN-based recommenders (e.g., Pin Sage), which significantly improved recommendation quality on billion-scale graphs [5].

Given these strengths, integrating GNNs into SoCRATe allows us to investigate whether richer graph-structured modeling enhances both accuracy and fairness, or whether it introduces new biases under limited availability conditions.

1.4.2 Variational Autoencoders

VAEs represent a different line of innovation in recommendation. As generative latent variable models, they learn a probability distribution over user preferences rather than point estimates. This provides two main advantages:

- **Modelling uncertainty and sparsity:** VAEs are able to sample plausible recommendations even when data is sparse, addressing cold-start situations more effectively than KNN or matrix factorization.
- **Regularization through latent structure:** The VAE objective combines reconstruction error with Kullback-Leiber (KL) divergence term, which regularizes the latent space and helps prevent overfitting. This is particularly useful when user-item interactions are noisy and incomplete [7].
- **Emperical improvements:** The Multi-VAE model proposed by Liang et al. [6] showed substantial gains over classical collaborative filtering (CF) techniques on large benchmark datasets, highlighting VAEs as strong candidates for recommendation tasks.

For SoCRATe, VAEs bring the opportunity to test whether a probabilistic generative approach to preference modeling translates into fairer allocations when items are limited and compensation mechanisms are in place.

1.5 Scope and Limitations

The scope of this thesis is defined by the decision to extend the SoCRATe framework with two advanced recommendation models: Graph Neural Networks (GNNs) and Variational Autoencoders (VAEs). The work focuses on integrating these models into SoCRATe, running experiments, and comparing their outcomes against the baseline KNN recommender and the brute-force Oracle. The main goal is to study how these different recommenders affect the fairness–utility trade-off in constrained recommendation environments.

1.5.1 Scope of the Work

- **Framework focus:** The study is centred on SoCRATe, which already provides a fairness-aware layer for compensation users over time when item availability is limited [2]. The contribution of this work is not to redesign SoCRATe itself but to extend it with more advanced recommendation models.
- **Models included:** Three recommenders are considered, KNN (baseline), GNN and VAE. Each model is integrated into SoCRATe's orchestrator and evaluated through the same fairness-aware strategies (round-robin, preference driven, adoption models).
- **Datasets:** The experiments are conducted on synthetic datasets (to allow controlled testing) and on real-world datasets such as Amazon Music/Movie and crowdsourcing task data. This combination helps to both generalizability and practical relevance.
- **Evaluation metrics:** The work adopts SoCRATe's fairness-oriented evaluation using cumulative loss (L) and its standard deviation (S) as the primary measures of accuracy and fairness, respectively [2]. Other standard recommender metrics like precision and recall are referenced for context but are not the main focus.
- **Comparative Study:** The study compares KNN, GNN, and VAE within the same framework to ensure that differences in fairness or utility can be attributed to the recommender model rather than external factors.

1.5.2 Limitations

Like any research, this thesis also comes with certain limitations

- **Model coverage:** The choice of GNN and VAE is motivated by their popularity and effectiveness in recommendation research [12,6], but they are not the only advanced models available. Other deep learning approaches such as Transformers or Reinforcement Learning (LR)-based recommenders are not explored.
- **Computational resources:** Training GNNs and VAEs requires more computation compared to KNN. Due to resource constraints, the experiments are run on limited hardware, which might restrict the size of datasets and the extent of hyperparameter tuning.
- **Dataset variety:** While both synthetic and real-world datasets are included, the selection is still limited. Results may not fully generalize to all domains where limited availability occurs (e.g., healthcare, education, or finance).
- **Evaluation metrics:** The fairness definition used here is tied to SoCRATe's cumulative loss metrics [11]. Other fairness dimensions, such as group fairness or exposure fairness [13], are outside the scope of this work.

2. RELATED WORK

2.1 Fairness in Recommender System

Fairness has become a central concern in recommender systems as their influence increasingly shapes access to information, products, and opportunities. Traditional recommenders optimise for predictive accuracy, often overlooking whether the distribution of recommendations treats users and item providers equitably. As highlighted in prior work, unfairness can emerge from popularity bias, skewed interaction histories, or demographic imbalances, resulting in systematic disadvantages for certain users or groups [12,14,16].

Fairness in recommendation is commonly studied through two primary lenses: individual fairness, where similar users should receive similar recommendations, and group fairness, which ensures that protected or underrepresented groups are not disproportionately disadvantaged [14,15]. Another important dimension is exposure fairness, which focuses on ensuring that item providers receive adequate and equitable visibility across ranked lists [16]. These concerns arise because modern recommenders typically operate under feedback loops popular items become more visible, and visible items accumulate more interactions reinforcing algorithmic bias over time.

Beyond these classical definitions, fairness has been revisited more broadly in recent survey works. Pitoura, Stefanidis, and Koutrika offer a comprehensive overview of fairness models, measurement strategies, and mitigation methodologies for rankings and recommendations, emphasizing the necessity of balancing consumer-side and producer-side fairness, transparency, and long-term exposure equity [26,27]. Their work directly supports the motivation of this thesis by illustrating that fairness must be considered not only at the algorithmic level but also in the overall lifecycle of recommendation from model training to ranking and exposure.

Another relevant direction concerns fairness under limited availability, where constrained resources amplify inequities among users. When items can only be recommended to a limited number of users such as tasks, tickets, or scarce goods competition leads to unequal opportunities. While classical fairness work often

assumes unlimited resources, the SoCRATe framework explicitly targets this scenario by measuring user-level disparities and compensating for accumulated disadvantage [1,2]. This gap between fairness definitions and resource constrained environments further highlights the importance of personalised loss compensation mechanisms.

Recent work on large scale entity resolution and similarity processing also contributes indirectly to fairness conversations. Christophides et al. [28] and Efthymiou et al. [29,30] analyse the challenges of resolving heterogeneous or partially similar entities in web-scale data environments. Their findings emphasise that system-level fairness is influenced by the quality and completeness of underlying data representations. When user or item data is inconsistently linked or unevenly represented, recommender outputs can inherit structural biases. This is especially relevant in real-world systems where metadata heterogeneity or missing information results in unequal model treatment of users or items. Such insights reaffirm that fairness is not merely a ranking problem but a data management problem, affected by the availability, structure, and quality of user–item interaction data.

Moreover, as recommender systems increasingly integrate graph-based, latent variable, and deep learning approaches, fairness mitigation must evolve accordingly. Graph based fairness, discussed in recent studies, highlights how graph structure can itself encode systemic inequalities (e.g., uneven connectivity or popularity), and how message-passing operations can amplify or reduce these biases [25]. Similarly, probabilistic models such as VAEs have shown promise in smoothing disparities between heavily active and sparsely active users, improving fairness by regularizing latent preference distributions.

Taken together, the literature demonstrates that fairness in recommender systems is a multifaceted challenge spanning individual, group, and exposure fairness; data representation; and long-term behaviour under iterative user interactions. This thesis builds on these foundations by examining fairness in a multi-session**,** limited-availability setting through the SoCRATe framework, and by exploring how advanced recommenders such as GNNs and VAEs interact with fairness-aware orchestration mechanisms. The inclusion of these models allows us to evaluate whether richer representations amplify or mitigate fairness inequalities when recommendations must be distributed under scarcity.

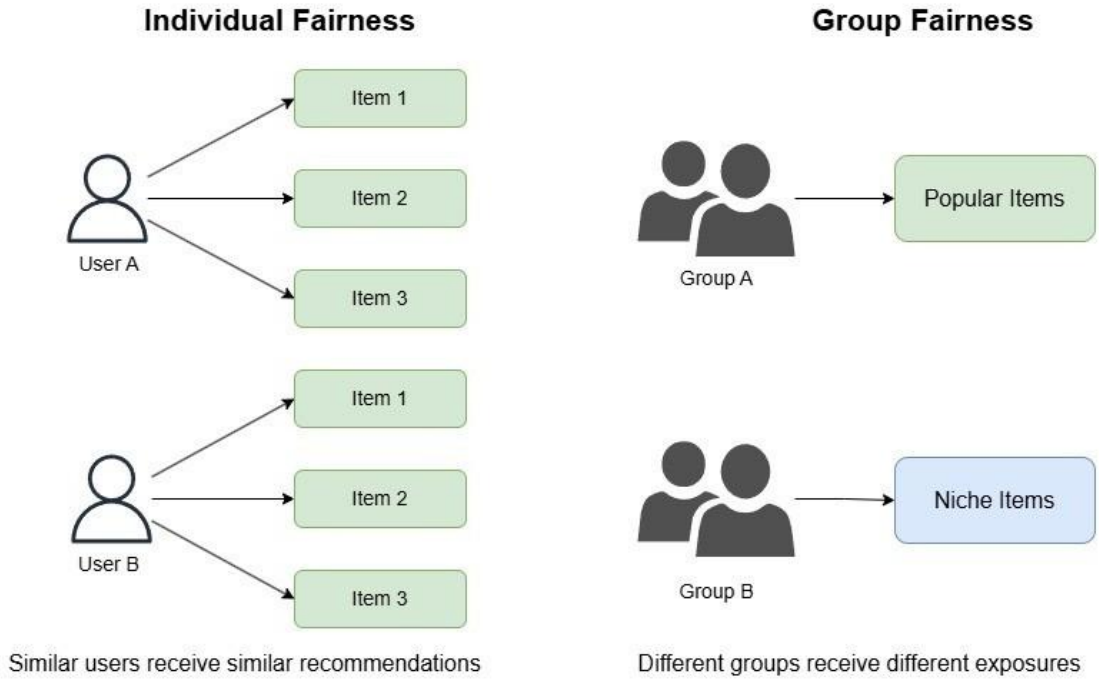

Figure 1. *Fairness in recommender system. (a) Individual fairness ensures similar user gets similar recommendations (b) Group fairness ensures groups are treated without systematic bias.*

2.2 Loss Compensation in Recommendations

While fairness in recommender systems has been studied through various perspectives including individual, group, and exposure fairness as discussed in Section 2.1 most of this literature assumes that items are infinitely available, meaning that recommendations made to one user do not restrict what can be recommended to others. However, this assumption breaks down in many real-world applications such as crowdsourcing platforms, product marketplaces with limited stock, course allocation systems, and event ticketing. In such environments, users directly compete for scarce resources, and fairness becomes intertwined with item availability, timing, and consumption order.

The SoCRATe framework was designed specifically to address fairness under these limited-availability constraints [1,2]. Its core principle is **loss compensation**: whenever a user misses out on an optimal recommendation due to an item being unavailable, the resulting disadvantage is recorded and compensated for in later sessions. To formalise this, SoCRATe defines cumulative loss metrics that quantify how far each user's assigned items deviate from the theoretical optimum, as computed by a brute-force Oracle. Two metrics are central:

- Average cumulative loss (L) which measures overall efficiency.
- Standard deviation of cumulative loss (S), which captures fairness by indicating how unevenly losses are distributed across users. Lower variance reflects more equitable allocation.

The compensation mechanism operates iteratively. After each recommendation round, SoCRATe compares the delivered recommendations with the Oracle optimal ones, updates each user's cumulative loss, and uses this information to reorder users in subsequent rounds. Users with higher accumulated loss are prioritised either aggressively, in the **preference-driven strategy**, or more gradually, in the **round-robin strategy**. These mechanisms ensure that fairness is not enforced only instantaneously, but is amortised across multiple sessions in a manner similar to exposure level fairness techniques studied in ranking literature [16,26,27]. This iterative view aligns with fairness-of-attention principles, where long-term equality is achieved by redistributing opportunities across multiple decision rounds rather than in a single ranked list.

Beyond its fairness mechanisms, SoCRATe also showcases how data quality and availability influence loss compensation. Recent work in entity resolution and large-scale similarity processing demonstrates that inconsistencies in data representation missing attributes, heterogeneous schemas, or partially similar entities can propagate unfairness into downstream models [28,30]. For example, if item metadata is incomplete or ambiguous, some items may appear less relevant during recommendation generation, indirectly increasing cumulative losses for the users who would have preferred them. These insights underscore that fairness under scarcity is not merely a ranking problem but also a data management challenge, affected by how accurately and consistently user–item interactions are captured.

Compared to traditional fairness interventions in recommenders, which often rely on pre-processing, regularisation, or post-processing adjustments [17,18,19], SoCRATe introduces a distinct temporal fairness dimension. Instead of attempting to make each individual ranking fair in isolation, it focuses on balancing cumulative utility over time. This multi-session perspective is particularly important in environments where users repeatedly return to the system and where item availability dynamically changes conditions under which classical fairness techniques may fail or introduce instability.

In summary, loss compensation provides a principled mechanism for achieving fairness in iterative recommendation settings with limited item availability. By tracking user-level

disadvantage across sessions and redistributing opportunities through compensation strategies, SoCRATe operationalises fairness in a way that aligns with both theoretical fairness notions and the practical realities of competitive resource allocation.

2.3 Deep Learning in Recommender Systems

Traditional recommender systems such as neighbourhood-based collaborative filtering and matrix factorisation have long served as effective and computationally efficient baselines. However, these approaches often struggle to capture non-linear relationships, complex behavioural patterns, and high-dimensional dependencies in large-scale interaction data [8,10]. As applications grew in scale and complexity, deep learning methods emerged as powerful alternatives capable of learning richer user and item representations.

One of the earliest directions in deep learning–based recommendation was the transition from linear latent factor models to Neural Collaborative Filtering (NCF), where multi-layer perceptrons model the interaction function between users and items. By replacing dot product similarity with a learnable non-linear mapping, NCF can capture more expressive patterns and better fit implicit feedback data [4]. This flexibility paved the way for a wide range of architectures tailored to sequential behaviour, temporal dynamics, and heterogeneous feature spaces.

A major milestone in the field has been the rise of Graph Neural Networks (GNNs) for recommendation. User–item interactions naturally form bipartite graphs, and GNNs leverage message passing to aggregate information from multi-hop neighbourhoods. Models such as Neural Graph Collaborative Filtering (NGCF) [4] and web-scale solutions like PinSage [5] demonstrated significant performance gains by propagating information through graph structure and capturing higher-order connectivity. Recent surveys confirm that graph-based methods now form one of the dominant paradigms in recommender research [3,25], with applications ranging from personalised feeds to marketplace optimisation.

Another influential line of work incorporates generative modelling through Variational Autoencoders (VAEs). The Multi-VAE framework proposed by Liang et al. [6] adapts probabilistic latent variable models to collaborative filtering, enabling the system to represent user preferences as distributions rather than fixed vectors. This probabilistic formulation is particularly effective under data sparsity, as it encodes uncertainty directly into the latent space and reduces overfitting through KL regularisation [7].

VAEs have been shown to outperform classical recommender baselines, especially when interaction histories are short, noisy, or highly imbalanced.

Deep learning has also proven effective for sequential and temporal recommendation tasks. Models built upon recurrent neural networks (RNNs) [21] capture short-term sequential dependencies, whereas attention-based architectures such as SASRec and BERT4Rec [20] model long-range contextual patterns more effectively. These architectures recognise that user behaviour unfolds over time and that both immediate and historical preferences influence future choices.

Beyond modelling accuracy, recent work underscores the need to consider fairness, transparency, and bias amplification when applying deep learning in recommendation. Deep models often inherit or magnify structural imbalances present in the interaction graph such as popularity bias, uneven item exposure, or sparse user coverage. Surveys by Pitoura, Stefanidis, and Koutrika [26,27] highlight how deep representation learning can inadvertently reinforce disparities unless fairness-aware mechanisms are incorporated. This challenge is especially salient in graph-based recommenders, where highly connected users or items disproportionately influence message passing, and in generative models where data sparsity can lead to uneven or skewed latent representations.

Moreover, insights from large-scale entity resolution research demonstrate that inconsistencies in data representation missing attributes, schema heterogeneity, or noisy linking can propagate into deep learning recommenders and influence their fairness and accuracy [28–30]. Since deep models rely heavily on high-quality, consistent input signals, unresolved data heterogeneity can distort both embedding learning and downstream ranking, further emphasising the need for robust preprocessing and representation alignment.

In summary, deep learning has transformed recommendation systems by enabling models to capture non-linear, high-order, and latent structures in user–item interactions. These advancements provide strong motivation for integrating modern deep learning recommenders such as GNNs and VAEs into fairness-aware frameworks like SoCRATe. Their expressive power allows them to model complex interactions and user behaviours, while their flexibility opens opportunities for incorporating fairness constraints, regularisation strategies, and compensation mechanisms that align with the multi-session, limited-availability setting of this thesis.

2.4 Fairness in Deep Learning Based Recommendations

As deep learning has gained prominence in recommender systems, concerns about fairness have become increasingly important. While advanced models such as graph neural networks (GNNs), variational autoencoders (VAEs), and attention-based architectures often achieve higher accuracy, they also introduce new pathways through which unfairness can emerge or be amplified. This has led to a rapidly growing body of research focusing on bias detection, mitigation, and fairness preservation in deep recommendation pipelines.

A central challenge is that deep models learn directly from historical interaction data, which often contains systemic biases. Popularity bias is one common example: frequently interacted items dominate the training signal, causing deep recommenders to disproportionately rank them higher and reinforce their visibility. This creates feedback loops where already prominent items become even more overrepresented an issue amplified by graph-based methods, where highly connected nodes exert more influence during message passing [25]. Similarly, users with sparse histories may be under represented in learned embeddings, causing the system to allocate lower-quality recommendations to them. These issues underscore the fact that improving fairness requires more than simply increasing model expressiveness.

Recent surveys by Pitoura, Stefanidis, and Koutrika [26,27] highlight several fairness dimensions particularly relevant to deep recommendations, including **exposure fairness**, **consumer fairness**, and **provider fairness**. Their work demonstrates that as models become deeper and more complex, transparency decreases, making it more difficult to identify why certain users or items receive preferential treatment. Fairness interventions must therefore be embedded not only at the ranking stage but throughout the learning pipeline during data processing, representation learning, and post processing adjustments.

Several mitigation approaches have been proposed in the literature. In-processing methods incorporate fairness constraints or adversarial components directly into model training [15,17], encouraging representational parity across users or demographic groups. Post-processing approaches modify the output rankings to correct imbalances in exposure or relevance [18]. In the context of deep recommenders, generative models such as VAEs have been shown to help smooth disparities by regularizing latent representations, reducing sensitivity to popularity-driven signals in the data. Similarly, fairness-aware GNN variants incorporate constraints to prevent message

passing from disproportionately amplifying structural inequalities in the interaction graph.

Another perspective comes from the domain of entity resolution and heterogeneous data integration, where Christophides, Efthymiou, and Stefanidis have shown that incomplete or inconsistently linked metadata can distort downstream learning tasks [28–30]. This insight is crucial for deep recommender fairness: if user or item records are duplicated, fragmented, or inconsistently represented, deep models may unintentionally treat certain users or items as less important or less connected. Poor entity resolution propagates directly into latent representations, creating fairness issues that cannot be corrected solely at the ranking stage. Thus, fairness in deep recommendation is inherently tied to upstream data quality and integration.

Despite progress, the interplay between fairness and deep learning remains challenging. Deep models offer powerful tools for capturing non-linear relationships but also operate as black boxes that may obscure bias sources. Moreover, fairness trade-offs often interact with accuracy, diversity, and long-term user satisfaction, requiring multi-objective optimisation rather than single-metric tuning. As fairness-aware recommender research continues to grow, there is increasing interest in iterative and longitudinal fairness ensuring that fairness is maintained across multiple sessions and over time rather than at a single recommendation snapshot.

In summary, fairness in deep learning based recommendation represents an intersection of algorithmic design, data quality, and system level behaviour. While deep models offer expressive and robust preference modelling, they also magnify the need for fairness-aware training, bias-aware graph propagation, robust data integration, and transparent evaluation. These challenges provide strong motivation for the work undertaken in this thesis, where GNN and VAE recommenders are integrated into the SoCRATe framework to examine how fairness can be maintained in multi-session, limited-availability environments.

3. METHODOLOGY

3.1 Overview of SoCRATe Framework

Recommender systems are widely used today to help users find relevant products, services, or information in large collections of items. They are essential in many domains such as e-commerce, media streaming, education, and crowdsourcing where users rely on intelligent algorithms to guide them toward meaningful choices. However, most traditional recommender systems operate under a key simplifying assumption: that all items are infinitely available and can be recommended to any user at any time. In practice, this assumption often fails. Many real-world systems face limited item availability, meaning that once an item is allocated or consumed by one user, it becomes unavailable for others.

Consider an example of a platform recommending training tasks to online workers or suggesting concert tickets to fans. In such environments, assigning an item to one user directly affects what remains for others. This creates a competitive and interdependent recommendation setting, where fairness becomes as important as accuracy. A highly accurate system that always recommends the best available items to the same few users can quickly become unfair, as other users repeatedly miss out on desirable options.

The **SoCRATe framework** was proposed to address exactly this challenge [1,2]. Rather than focusing only on improving accuracy in each round, SoCRATe introduces a mechanism to measure and compensate user dissatisfaction over tim**e**. It assumes that recommendations are made in multiple sessions, and that user satisfaction or loss can be tracked across these sessions. If a user repeatedly receives less optimal items because popular ones were already consumed, the system “remembers” this disadvantage and compensates that user in later iterations. This approach transforms recommendation from a one-shot prediction problem into an iterative fairness process, where the system dynamically adjusts its behaviour based on each user’s historical experience.

At the core of SoCRATe lies a loss compensation mechanism, which maintains two key metrics for every user the cumulative loss, representing the difference between what

the user actually received and what they could have received under an ideal allocation, and the deviation of these losses across all users, representing fairness at the population level. By continuously tracking these values, the system can identify users who have been disadvantaged and prioritized them in future recommendation rounds.

SoCRATe operates in a loop of recommendation, consumption, and compensation. During each iteration, every user receives a set of recommended items based on their preferences. Once items are consumed, their availability is reduced, and the system calculates how much satisfaction (utility) each user achieved compared to an optimal oracle allocation a theoretical benchmark representing perfect fairness and maximum efficiency. The resulting losses are accumulated and used in the next round to rebalance the system's decisions. This process allows SoCRATe to gradually redistribute recommendation opportunities among users, aiming for fairness without drastically reducing overall accuracy.

Another distinctive feature of SoCRATe is its flexible orchestration and compensation strategies. Two main strategies are implemented:

- Preference driven where users with highest accumulated loss are prioritized and fully compensated first.
- Round robin where all active users receive items one by one in cycles according to their loss ranking. These strategies allow the system to tune how aggressively it compensates users, depending on whether the application prioritizes fairness or engagement.

Importantly, SoCRATe is model-agnostic, it does not depend on any particular recommender algorithm. Instead, it acts as an additional layer on top of a standard recommender, guiding how recommendations are allocated and compensated. In its original implementation, SoCRATe used a K-Nearest Neighbors (KNN) model as its underlying recommender. This choice provided a simple and interpretable baseline to demonstrate the framework's fairness mechanisms without the added complexity of deep learning models.

In summary, SoCRATe introduces a new perspective on fairness in recommendation systems by treating it as an evolving and measurable property rather than a static constraint. By combining fairness-aware compensation with multi-session recommendation, it achieves a balance between accuracy and equality of treatment among users. The following sections of this chapter describe in detail the architecture,

workflow, and key components of the SoCRATe framework as originally proposed in [1,2].

3.2 Architecture of SoCRATe

The architecture of the original SoCRATe framework, as proposed in [1,2], was designed to model recommendation scenarios where items are consumed over time and availability is limited. The framework extends a traditional recommender system by introducing orchestration and fairness-aware mechanisms that operate iteratively across sessions. Each iteration represents a distinct recommendation cycle in which items are suggested to users, consumed, and then updated to account for fairness through loss compensation.

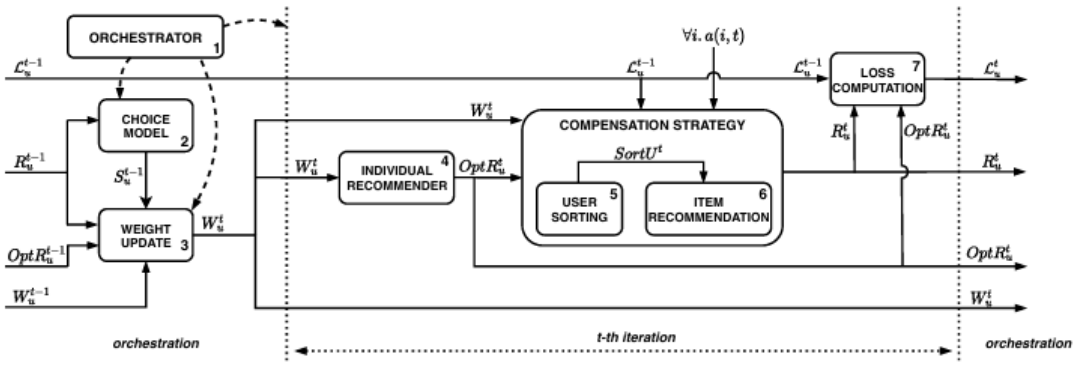

Figure 2. *SoCRATe architecture framework. Adopted from [1]*

At a high level, the framework consists of six interacting modules: the Orchestrator, Weight Update, Individual Recommender, User Sorting, Compensation Strategy, and Loss Computation modules. Together, these components enable SoCRATe to produce recommendations that balance accuracy and fairness dynamically over multiple sessions.

1. **Orchestrator**

   In the original SoCRATe framework, the Orchestrator is responsible for controlling the overall flow of recommendation iterations [1,2]. At the end of each session *t−1*, it monitors whether users have finished consuming their allocated items and then determines the start of the next iteration t. This component manages the timing and synchronization of the recommendation process, ensuring that new recommendations are generated only when all relevant user feedback has been collected.

To model realistic consumption behaviour, the framework introduced two levels of time granularity. The first, called fixed granularity, assumes that all users consume their recommended items simultaneously, reflecting a synchronized environment such as a batch recommendation process. The second, user-group granularity, divides users into smaller subsets that consume items asynchronously, better capturing real-world scenarios where users interact with recommendations at different speeds.

The Orchestrator uses the recommendations actually consumed by each user, denoted as S_u^{t-1}, and triggers the Weight Update module to adjust user preference weights accordingly for the next session. This design allows the framework to simulate dynamic environments where both item availability and user preferences evolve over time.

2. Weight Update Module

The Weight Update module refines user preference weights based on the feedback gathered from the previous iteration [1,2]. This step is essential to capture how users preferences change after each round of consumption. Specifically, the module takes as input three sets of information:

- The items consumed by each user (S_u^{t-1}),
- The recommendations originally generated (R_u^{t-1}), and
- The optimal recommendations derived from the oracle benchmark ($OptR_u^{t-1}$).

From these inputs, the module computes an updated weight vector W_u^t for each user, which represents their relative preference across different optimization dimensions, such as accuracy and fairness. Each entry in W_u^t indicates how much importance a user places on a particular objective.

In the original implementation, SoCRATe applied a regression-based refinement process that aligned the weight vectors with observed consumption patterns. This allowed the system to gradually adapt recommendations according to how users responded to previous allocations. The updated weight vectors were then passed to the Individual Recommender module, which used them to generate new ranked lists of items for the following session. Through this process, the Weight Update module ensured that user preferences were continuously recalibrated as the simulation progressed.

3. Individual Recommender

Once user preference weights were updated, the Individual Recommender module generated a new ranked list of candidate items for each user. The SoCRATe framework was designed to be model-agnostic, meaning it could operate on top of various underlying recommender algorithms. In the reference implementation, a K-Nearest Neighbors (KNN) collaborative filtering model was used as the baseline recommender to demonstrate the system's fairness and compensation mechanisms [1,2].

For each user, the recommender produced a list of items $Opt\ R_u^t$, ranked according to predicted preference scores derived from user–item similarity. These rankings represented the best possible set of recommendations that could be made without considering fairness constraints. The resulting ranked lists were then forwarded to the Compensation Strategy module, which adjusted item allocation based on user loss and item availability.

By keeping the recommendation component independent from the fairness mechanisms, SoCRATe ensured that the orchestration and compensation logic could later be applied to different recommender models without structural modification. This modularity became one of the key strengths of the framework, allowing future extensions such as deep learning based recommenders to be integrated seamlessly while preserving the fairness evaluation process.

4. User Sorting Module

The User Sorting module in the original SoCRATe framework plays a crucial role in determining the order in which users are served during each recommendation iteration [1,2]. Once the cumulative loss values have been updated, this module arranges users according to specific sorting criteria, which directly influence how fairness compensation is applied in the subsequent allocation phase. The sorting mechanism ensures that users who have been more disadvantaged in previous rounds receive higher priority in the next one.

In its default configuration, SoCRATe sorts users in decreasing order of their cumulative loss values. This means that the user who has experienced the greatest total loss up to iteration t is placed first in the list, while users with

smaller cumulative losses are placed later. By following this order, the system can focus on compensating the most disadvantaged users before others, progressively balancing fairness over time.

To make the sorting process more flexible and adaptable to different scenarios, the framework introduced several sorting modes. These include:

- **Loss-based sorting**, where users are ordered from the highest to the lowest cumulative loss. This is the standard mode and ensures that fairness is prioritized at every iteration.
- **Shuffle mode**, where users are randomly ordered before each iteration. This configuration simulates situations where fairness is not explicitly enforced, allowing the system to evaluate the effect of randomness on fairness and utility outcomes.
- **Forced mode**, where the order of users remains fixed across iterations, meaning the same user always receives recommendations first. This setup is useful for benchmarking, as it provides a baseline for understanding how user order affects overall fairness dynamics.

To illustrate this process, consider an example with three users U1, U2, and U3 who have accumulated losses of 0.45, 0.30, and 0.15 after several iterations. In loss-based sorting, the system would serve them in the order U1 → U2 → U3, ensuring that U1, who has faced the highest loss, is compensated first. If shuffle mode is enabled, their order might be randomized, such as U2 → U3 → U1, which could result in a more unpredictable fairness pattern. In contrast, in forced mode, if the system starts with U3, the same user will always be first, potentially leading to systematic bias over time.

This flexible sorting mechanism allows SoCRATe to simulate different recommendation environments and study how the ordering of users affects the long-term balance between fairness and efficiency. By integrating user sorting as a separate module, the framework ensures that the compensation process can be adapted to various fairness objectives without altering the underlying recommendation algorithm.

5. Compensation Strategy Module

The Compensation Strategy module determines how recommendations are allocated to users once their order has been established by the User Sorting module. Its purpose is to decide whether compensation is applied quickly to highly disadvantaged users or more evenly spread across all users. SoCRATe provides two distinct strategies: preference-driven and round-robin compensation [1,2].

In the preference-driven strategy, users are first sorted by their cumulative loss in descending order. The user with the highest loss is then fully compensated by receiving all of their N recommended items in that session. Only after the top user is fully served does the system move on to the next user in the list. This approach is designed to rapidly reduce disparities by prioritizing the most unsatisfied users. Preference-driven allocation is particularly suited for scenarios where the system needs to avoid user frustration or churn, since it ensures that disadvantaged users are compensated quickly and in full. The image below illustrates how preference driven works when users have same and different preferences. On left if they have same preference then user 1 gets top items followed by user 2 which gets least preferred items. This induces dissatisfaction while on the other hand if they different preferences then users do not suffer loss because they both get their top desired items in recommendations.

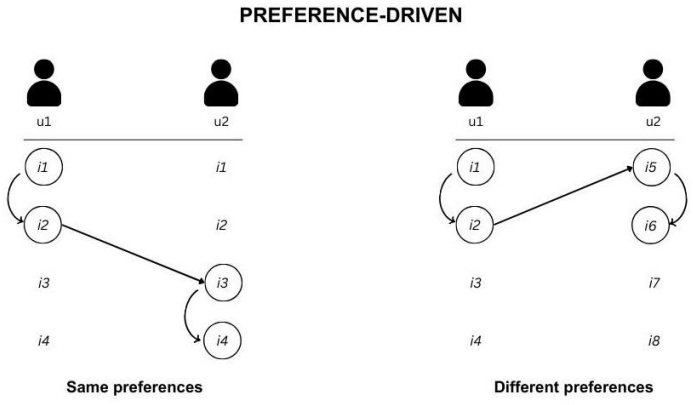

Figure 3. *Preference driven for users under similar and different preferences illustrating how items are recommended to users*

In contrast, the round-robin strategy also begins by sorting users by cumulative loss, but items are allocated more gradually. Instead of giving one user all their recommendations at once, the system cycles through the ordered list of users, assigning one item at a time in turn, until each user has received their full set of N recommendations. This ensures that every user receives at least some recommendations earlier in the process, distributing opportunities more evenly. Round-robin allocation is therefore more appropriate in contexts where overall consumption and user engagement are the primary objectives, as it maximizes the likelihood that all users will obtain items they value in each session. In the figure below if users have same preference, then round robin tries to distributes item such that users does not suffer biasness and minimizes loss in early iterations. If users have different preferences than round robin works similar to preference driven because users receive their desired items in either item assignment strategy.

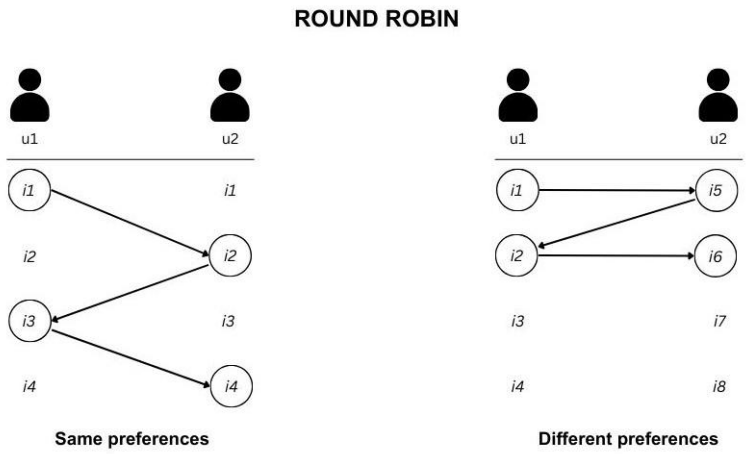

Figure 4. *Round robin strategy for users under similar and difference preferences illustrating how items are recommended to users to mitigate loss*

The choice between preference-driven and round-robin strategies reflects a design trade-off in fairness-aware recommendation. Preference driven allocation is more aggressive in compensating the most disadvantaged users but may concentrate high quality items among a small subset of users. Round-robin allocation, on the other hand, spreads recommendations more broadly, leading to slower but more balanced fairness recovery. In practice, the selection of the strategy depends on system goals: rapid satisfaction of frustrated users versus long-term retention and engagement across the user base.

6. **Loss Compensation Module**

The Loss Computation module plays a central role in SoCRATe as it provides the quantitative foundation for evaluating both accuracy and fairness over time. At the end of each session, the framework compares the quality of the recommendations that a user actually received with the quality of the recommendations that the same user would have obtained under the Oracle allocation [1,2]. The Oracle represents an idealized brute-force solution that distributes items optimally while respecting availability constraints, thereby serving as the benchmark for fairness evaluation.

Formally, the cumulative loss of a user u at session t is defined as:

$$\mathcal{L}_u^t = \mathcal{L}_u^{t-1} + (U_u^{*t} - U_u^t)$$

where U_u^{*t} represents the ideal utility of user u at session t under the Oracle allocation and U_u^t denotes the actual utility achieved by the recommendations assigned to the user in SoCRATe. The utility values are typically computed as the sum of relevance scores of the items allocated to a user in a session, for example predicted ratings or similarity-based weights. The difference between the ideal and actual utilities reflects the extent to which a user was disadvantaged compared to the optimal allocation, and the cumulative formulation ensures that disadvantages are tracked across multiple sessions rather than in isolation. To analyze the system as a whole, SoCRATe computes two aggregate metrics from the per-user losses. The first is the average cumulative loss,

$$\overline{\mathcal{L}^t} = \frac{1}{|u|}\sum_{u \in U} \mathcal{L}^t$$

which measures how closely the system approximates the oracle in terms of global accuracy. A lower value of $\overline{\mathcal{L}^t}$ indicates that the recommendations delivered under scarcity are approaching the optimal allocation. The second metric is the standard deviation of cumulative loss.

$$S(t) = \sqrt{\frac{1}{|U|}\sum_{u \in U} (L_u(t) - L(t))^2}$$

which captures fairness by quantifying the disparity in losses across users. A small value of S(t) means that losses are evenly distributed, indicating fairness, while a large value of S(t) implies that some users are disproportionately disadvantaged.

These two measures are complementary: while L(t) captures the accuracy of recommendations at the system level, S(t) reflects whether this accuracy is distributed equitably among users. This distinction is critical, as a system may achieve high overall accuracy but still be unfair if certain users consistently receive lower-quality recommendations. The Loss Computation module thus provides the necessary feedback for SoCRATe's compensation strategies, guiding preference-driven and round-robin allocation mechanisms to reduce disparities over time. By grounding fairness in measurable deviations from the Oracle allocation, SoCRATe ensures that fairness is not just an abstract concept but an operational property that can be optimized in practice [1,2].

Once losses are computed, the Orchestrator triggers the next iteration, starting again with updated weights and new recommendations. This iterative cycle continues for a predefined number of sessions, allowing fairness to be evaluated over time rather than in a single step.

Component	Role in Framework	Key Functions
Orchestrator	Controls the flow of iterations and determines when new recommendations are generated.	Handles time granularity (fixed vs user-group) triggers weight updates for next session.
Weight Update Module	Updates user preference weights based on consumed recommendations and feedback	Computes W_u^t from S_u^{t-1}, R_u^{t-1}, $Opt\ R_u^{t-1}$ refines weights via regression.
Individual Recommender	Generates ranked candidate list of items for each user.	Produces recommendation list $R_u(t)$ using KNN.
User Sorting Module	Orders users based on fairness related criteria	Sorting modes: loss based, shuffled or forced ordering

	before allocation.	
Compensation Strategy	Allocates items to users according to fairness strategies.	Implements preference-driven or round robin.
Loss Compensation Module	Quantifies fairness and accuracy by measuring user disadvantages compared to Oracle allocation	Updates per-user loss $L_u(t)$ computes system-level metrics: average Loss $L(t)$, fairness $S(t)$.

Table 1. *Overview of SoCRATe architecture*

3.3 Baseline Recommender

The SoCRATe framework was designed to be independent of any specific recommendation algorithm. Its architecture allows it to operate on top of different recommender models, as long as they can produce a ranked list of candidate items for each user. In the original implementation, a user-based K-Nearest Neighbors (KNN) model was adopted as the baseline recommender [1,2]. The choice of KNN was motivated by its simplicity, interpretability, and proven reliability in collaborative filtering tasks. By relying on a transparent and well-understood algorithm, the original SoCRATe study was able to isolate the effect of its fairness and compensation mechanisms without the additional complexity introduced by deep learning models.

The KNN approach is one of the most classical methods in recommender systems. It assumes that users with similar past behaviors are likely to have similar preferences in the future. The method builds on the concept of user–user similarity: for a given user, the system identifies other users whose rating or consumption patterns closely match theirs. Based on this similarity, KNN predicts how the user would rate or prefer items they have not yet interacted with. The system then recommends the top-N items with the highest predicted preference scores.

In the context of SoCRATe, KNN serves as the underlying algorithm in the Individual Recommender module. During each iteration, it generates a ranked list of items for every user, which are then passed to the Compensation Strategy module for fairness adjustment. The use of KNN provides a stable and explainable baseline for studying fairness dynamics because its recommendation logic is deterministic and easily interpretable.

Mathematically, the KNN recommender computes the similarity between two users u and v based on their item interaction histories. One of the most common similarity measures is the cosine similarity, defined as:

$$sim(u,v) = \frac{\sum_{i \in I} r_{u,i} * r_{v,i}}{\sqrt{\sum_{i \in I} r_{u,i}^2} * \sqrt{\sum_{i \in I} r_{v,i}^2}}$$

where $r_{u,i}$ and $r_{v,i}$ represent the ratings (or implicit feedback values) that users u and v have given to item i. This similarity score ranges from 0 to 1, where higher values indicate stronger similarity between users. Once similarities are computed for all pairs, each user's K nearest neighbors N_u are selected these are the users most similar to the target user.

The predicted preference score $\hat{r}_{u,i}$ for a user u and an unobserved item i is then calculated as a weighted average of the ratings given by similar users:

$$\hat{r}_{u,i} = \frac{\sum_{v \in N_k} sim(u,v) \cdot r_{v,i}}{\sum_{v \in N_k(u)} |sim(u,v)|}$$

Here N_u represents the set of top-k most similar users to u and $r_{v,i}$ denotes how each neighbor v rated item i. The result $\hat{r}_{u,i}$ gives an estimate of how much user u would like item i. The system then ranks all unconsumed items for each user based on these predicted scores and recommends the top-N items with the highest values.To illustrate this process, consider a simple example with three users and four movies. If User A and User B have rated several movies similarly, and User C has not rated one of those movies, the algorithm infers that User C might like that movie as well. The system calculates similarity scores between User C and the others, finds that User B is most similar to them, and uses User B's ratings to estimate what User C would think about the unseen movie. The movie is then added to User C's top-N recommendations. Over multiple iterations in SoCRATe, this process repeats as item availability changes and users consume new items, gradually updating their preference profiles.

In SoCRATe's fairness-oriented design, the recommendations produced by KNN represent the "raw" allocation stage, where users are ranked based purely on similarity and predicted preference, without considering fairness constraints. These recommendations are then processed by the Compensation Strategy module, which adjusts allocations based on cumulative user losses. In this way, SoCRATe separates the generation of recommendations (handled by KNN) from the fairness logic (handled by the orchestration and compensation components). This modular separation is

crucial because it allows fairness to be treated as an independent property that can be applied to different recommender types. Using KNN in SoCRATe offers several advantages. First, its simplicity makes it computationally lightweight compared to more complex neural approaches. Second, its explainability allows researchers to clearly understand how fairness mechanisms alter user satisfaction over time. Third, because KNN relies solely on user–item interactions and does not require complex feature engineering, it is ideal for controlled experiments on fairness and loss compensation. These characteristics make KNN a solid reference point for comparing how more expressive models, such as GNNs and VAEs, behave under the same fairness framework.

In summary, the KNN-based implementation of SoCRATe demonstrates how fairness-aware orchestration can be applied to a traditional recommender system. By coupling a straightforward similarity-based model with fairness compensation, the framework provides a transparent way to analyze how user losses evolve and how fairness can be progressively restored across multiple recommendation sessions. The following section summarizes the complete SoCRATe framework as presented in [1,2] before moving to the extended contributions developed in this thesis.

4. EXTENSION WITH ADVANCED RECOMMENDERS

4.1 Introduction

The previous chapter described the original SoCRATe framework, which introduced a fairness-aware mechanism for multi-session recommendation under limited item availability. By combining user loss tracking, compensation strategies, and fairness metrics, the framework demonstrated that it is possible to balance accuracy and equity in iterative recommendation environments [1,2]. However, the baseline implementation relied on a K-Nearest Neighbors (KNN) recommender, which, while interpretable and computationally simple, has limited capacity to model complex patterns in user–item interactions. KNN-based systems primarily depend on direct similarity measures and often struggle to generalize beyond observed data, especially in sparse or large-scale environments.

Modern recommender system research has moved towards representation learning, where models learn to embed users and items into shared latent spaces that capture deeper semantic relationships [8,10,20]. Among these, Graph Neural Networks (GNNs) and Variational Autoencoders (VAEs) have emerged as two of the most influential approaches. GNN-based recommenders leverage the graph structure of user–item interactions, enabling the model to propagate preferences and infer connections beyond immediate neighbors [3–5,24]. VAEs, on the other hand, model user preferences as probabilistic latent variables, allowing the recommender to capture uncertainty and diversity in user behaviour [6,7,25]. These models have shown strong performance in both accuracy and robustness, motivating their integration into fairness-aware frameworks like SoCRATe.

The motivation for extending SoCRATe with GNN and VAE recommenders is twofold. First, it allows an examination of whether more expressive models can improve the fairness–utility balance when operating under item scarcity. While traditional models may struggle to infer unobserved user preferences, GNNs and VAEs are capable of learning higher-level abstractions that could yield more equitable outcomes across users. Second, this extension provides an opportunity to analyse the computational

and behavioural implications of deep learning–based recommenders within an iterative fairness mechanism. As SoCRATe operates in multiple rounds of recommendation and compensation, understanding how complex models behave across iterations is crucial for assessing their real-world applicability.

This chapter therefore focuses on extending the original SoCRATe framework by integrating two advanced recommenders: a Graph Neural Network (GNN) and a Variational Autoencoder (VAE). The goal is to investigate how these models interact with SoCRATe's fairness-aware orchestration and to determine whether they can enhance fairness without significantly compromising efficiency. The chapter addresses the research questions formulated in Section 1.2, particularly: (1) how the choice of recommender algorithm influences the fairness–utility trade-off, (2) how SoCRATe's compensation strategies interact with advanced recommenders, (3) whether GNN and VAE approaches can approximate oracle-level fairness and generalize across datasets, and (4) what computational trade-offs emerge when integrating these models.

4.2 Motivation for Advanced Recommenders

The motivation to integrate advanced recommenders into the SoCRATe framework stems from the growing need to model user–item relationships more effectively in complex and data-sparse environments. While the original SoCRATe implementation demonstrated that fairness can be achieved through iterative loss compensation, it relied on a K-Nearest Neighbors (KNN) recommender, which is inherently limited in its ability to represent high-dimensional patterns of user preference. KNN depends primarily on explicit similarity between users or items and often fails to capture more subtle or indirect associations that are important in modern recommendation scenarios [10].

In recent years, advances in deep learning have transformed recommender systems by enabling them to learn richer, more abstract representations of users and items [8,20]. These models go beyond simple similarity matching and instead learn latent features that encode relationships, preferences, and behavioural patterns that are not directly observable from raw interaction data. Among these approaches, Graph Neural Networks (GNNs) and Variational Autoencoders (VAEs) have shown remarkable success in capturing the complex structure of recommendation data.

Graph-based models, such as GNNs, are particularly effective for representing user–item interactions as a bipartite graph, where edges indicate historical interactions or preferences [3–5]. This perspective allows the model to capture not only direct relationships between users and items but also higher-order connections through neighbours in the graph. For example, even if two users have never interacted with the same items, the model can still infer that they are similar if they share common neighbours in the graph. Through message passing and neighbourhood aggregation, GNNs learn embeddings that represent users and items within a shared latent space, improving both generalization and interpretability. Studies such as Neural Graph Collaborative Filtering [4] and PinSage [5] have demonstrated that GNN-based approaches outperform traditional collaborative filtering, especially when data sparsity is high or when user behaviour is heterogeneous. Additional survey work on fairness in graph-based recommenders highlights the importance of addressing structural biases in GNN message passing [24].

On the other hand, Variational Autoencoders (VAEs) provide a probabilistic framework that captures uncertainty in user preferences. Instead of predicting a single deterministic preference score, VAEs learn distributions over latent factors representing user tastes and item characteristics [6,7]. This makes them particularly suitable for recommendation environments where user behaviour is diverse or partially observed. By modelling the probability of different preference patterns, VAEs can generate recommendations that are not only accurate but also more diverse and representative of user uncertainty. In addition, extensions such as the Fair Variational Autoencoder (FairVAE) [25] show that probabilistic modelling can be adapted to incorporate fairness constraints, making it a strong candidate for integration into a fairness-aware framework like SoCRATe.

Integrating these two models into SoCRATe has several advantages. First, their expressiveness allows for improved recommendation accuracy, as both GNNs and VAEs can better estimate user preferences for items that have limited historical data. Second, by producing more representative user and item embeddings, these models help reduce systemic biases caused by over-represented users or popular items issues that often exacerbate fairness disparities. Third, since SoCRATe operates iteratively across multiple sessions, using models that can adapt and learn from structural and probabilistic patterns makes it possible to achieve a more stable fairness–utility balance over time.

For instance, in a sparse dataset where many users have interacted with only a few items, KNN may struggle to find meaningful neighbours, resulting in inaccurate or repetitive recommendations. In contrast, a GNN can exploit multi-hop relationships, identifying indirect similarities that lead to more relevant and fair item allocations. Similarly, when user preferences are uncertain or vary across contexts, a VAE can capture this variation by generating recommendations that reflect the range of possible interests, thereby preventing certain users from being repeatedly overlooked.

By incorporating GNN and VAE models into the fairness-aware SoCRATe architecture, this thesis explores whether a richer representation of user–item relationships can improve both recommendation quality and fairness outcomes. These advanced models not only enhance predictive capability but also provide a foundation for analysing how fairness-aware orchestration interacts with modern learning paradigms in recommendation systems.

4.3 Graph Neural Network Integration

The second extension explored in this thesis is the use of Graph Neural Networks (GNNs) within the SoCRATe framework. GNNs have emerged as one of the most powerful tools for recommendation because they allow information to travel through a graph structure rather than depending only on direct similarities between users or items. In recommender systems, this graph is naturally defined as the user–item bipartite graph, where users are connected to items they have interacted with. Unlike the KNN baseline, which looks for neighbors based on immediate overlap, GNNs can incorporate both direct and indirect connections in this graph, capturing patterns that extend beyond the most obvious user–item relationships [3,4,5].

In SoCRATe, we adopt the LightGCN architecture, which is a simplified but effective form of GNN designed specifically for collaborative filtering [4]. Traditional neural network layers involve learnable transformations and non-linear activation functions, which can be powerful but also introduce complexity and risk of overfitting. LightGCN deliberately removes these extra components and retains only the essential ingredient: embedding propagation over the graph. This means that each user and item is represented as a vector (an embedding), and these embeddings are updated layer by layer by averaging information from their connected neighbors.

The process begins with an adjacency matrix A representing the user–item graph. To make propagation fair across nodes with different numbers of connections, this adjacency is normalized into

$$\widetilde{\mathrm{A}} = \mathrm{D}^{-\frac{1}{2}}\mathrm{A}\mathrm{D}^{-\frac{1}{2}}$$

with D being the diagonal degree matrix. This ensures that propagation distributes influence proportionally across connected neighbors. To perform embedding learning on this graph, we employ the LightGCN model [4], which simplifies traditional graph convolution by removing transformation matrices and non-linear activations. Instead, LightGCN retains only the propagation mechanism, which is shown to be highly effective in collaborative filtering. Starting from initial trainable embeddings E^0, user and item embeddings are updated iteratively as

$$\mathrm{E}^{(l+1)} = \widetilde{\mathrm{A}}\mathrm{E}^{(l)},$$

where *l* denotes the propagation layer. This means that after the first layer, a user's embedding reflects information from the items they are directly connected to. After the second layer, it also includes information from other users who consumed the same items, and so on. After L layers, the final embedding is defined as the average across all layer outputs:

$$E = \frac{1}{L+1}\sum_{l=0}^{l} E^{(l)}$$

This aggregation captures both immediate and higher-order connectivity, balancing local preference patterns with long-range relationships across the graph. The learned embeddings for users (E_u) and items (E_i) are then used to compute preference scores. The predicted relevance of user u for item i is defined as the inner product

$$\hat{r}_{u,i} = E_u^T \cdot E_i$$

which provides a ranking function over candidate items. These scores are used to generate the recommendation lists that serve as inputs to the SoCRATe orchestration. The training of the GNN model is guided by the Bayesian Personalized Ranking (BPR) loss [4], which optimizes for pairwise ranking between observed and unobserved items. For a training batch of triples *(u,i,j)*, where *I* is a positive item and *j* is a negative item, the objective is

$$\mathcal{L}_{BPR} = -\frac{1}{|B|}\sum_{(u,i.j)\in B} log\ \sigma(\hat{r}_{u,i} - \hat{r}_{u,j}) + \boldsymbol{\lambda}\,||\boldsymbol{\Theta}||^2$$

where σ is the sigmoid function, Θ are the model parameters (embeddings), and λ controls L2 regularization. This loss enforces that items with which a user has interacted are ranked higher than items they have not consumed. To align the GNN with SoCRATe's fairness objectives, the training is further extended with additional

proxy losses. Specifically, a fairness-aware trainer augments the BPR objective with terms that promote efficiency and fairness. The efficiency term encourages embeddings to maximize utility across users, while the fairness term penalizes disparities in user losses, reflecting SoCRATe's measure of fairness based on the standard deviation of cumulative loss. The resulting total objective is defined as

$$\mathcal{L}_{total} = \mathcal{L}_{BPR} + \lambda_{eff}\mathcal{L}_{eff} + \lambda_{fair}\mathcal{L}_{fair}$$

where λ_{eff} and λ_{fair} control the balance between accuracy and fairness. An important factor here is the τ (tau) parameter, which controls how strongly the fairness term reacts to differences between users. Intuitively, τ can be thought of as a sensitivity dial: a small τ makes the fairness loss very sharp, heavily penalizing even small differences between users, while a large τ smooths the penalty, giving more priority to utility. This provides a direct and interpretable way to tune the fairness–utility trade-off in practice [27].

After training, the GNN outputs a user–item score matrix that is passed into SoCRATe's orchestration module. At this stage, the system decides how to allocate items under availability constraints. Importantly, the orchestration not the GNN controls whether allocation follows the preference-driven or round-robin strategy. After each iteration, SoCRATe evaluates user losses, sorts users accordingly, and then applies the chosen allocation method. In the preference-driven strategy, the users with the highest accumulated loss are served first and receive all their recommended items before moving to the next user. In round-robin, users are again ordered by loss but items are allocated one by one in rotation, ensuring a more balanced distribution across the population. This switching between strategies occurs after every iteration, while the GNN continues to provide the candidate rankings that the orchestration builds on.

Beyond fairness considerations, GNNs also bring robustness to sparsity, which is a critical advantage in real-world datasets. Many users in practice interact with only a handful of items, making similarity-based methods like KNN unreliable due to insufficient overlap. With LightGCN, however, even users with very few interactions can obtain informative embeddings, since their representation incorporates not only their immediate neighbors but also multi-hop neighbors through the graph propagation. This allows the model to generalize more effectively, reducing the bias toward heavy users and popular items.

In summary, integrating GNNs into SoCRATe combines the expressive power of graph-based embeddings with the fairness logic of the orchestration layer. The

LightGCN architecture provides a simple but effective way to learn from the user–item graph, the fairness-aware training introduces explicit trade-offs between utility and fairness controlled by τ, and the orchestration ensures that items are allocated under limited availability through preference-driven or round-robin strategies after each iteration. Together, these elements enable us to investigate not only whether GNNs improve the fairness–utility balance but also how advanced representation learning interacts with SoCRATe's unique loss compensation mechanism.

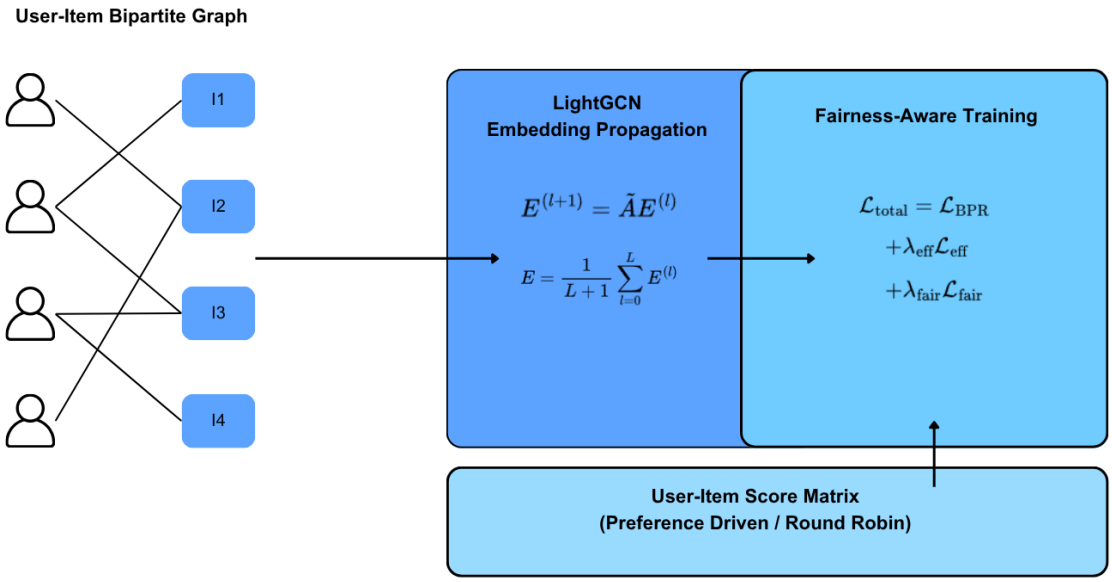

Figure 5. *Pipeline of GNN integration in SoCRATe*

4.4 Variational Autoencoders Integration

Another advanced recommender integrated into SoCRATe is the Variational Autoencoder (VAE), a deep generative model originally proposed for unsupervised learning [7] and later adapted for collaborative filtering tasks [6]. While GNNs rely on propagating information across a user–item graph, VAEs instead learn to compress each user's history of interactions into a latent probabilistic representation and then reconstruct it back into predicted preferences. This approach is powerful in recommendation because it not only captures hidden structures in user behavior but also introduces a measure of uncertainty about preferences, which is valuable when working with sparse or incomplete interaction data.

In SoCRATe, the implemented model follows the Mult-VAE design by Liang et al. [6]. The input is the user–item interaction matrix, where each user u is represented by a binary vector $x_u \in \mathbb{R}^{|I|}$. Each entry in this vector indicates whether the user has

interacted with an item, making the model naturally suited to implicit feedback data such as clicks, views, or purchases.

The first stage of the model is the encoder network, which maps the sparse user vector into the parameters of a latent distribution. Instead of directly producing a single deterministic embedding, the encoder outputs a mean and a variance for each latent dimension:

$$\mu = f_u(x_u), \qquad log\ \sigma^2 = f_\sigma(x_u)$$

Together, these define the approximate posterior distribution $q(z|x_u)$, where z is the latent representation of the user. To generate a sample from this distribution in a way that is compatible with gradient-based optimization, the reparameterization trick is used [7]:

$$z = u + \sigma \odot \epsilon, \ \ \epsilon \sim N(0,1)$$

This formulation introduces randomness through ϵ while keeping the mapping differentiable, making it possible to train the model using stochastic gradient descent. Conceptually, this means that each user's interaction history is compressed into a point in a latent space, but the representation is not fixed; it is drawn from a distribution, reflecting uncertainty about the user's true preferences.

The second stage is the decoder network, which takes the latent representation z and reconstructs the user's original interaction vector. The decoder produces a vector of predicted scores for all items,

$$\hat{x}_{u,i} = f_\theta(z)$$

Where each entry $\hat{x}_{u,i}$ corresponds to the likelihood that user u would interact with item i. By sorting these scores, we obtained the ranked recommendation list for the user. Training the VAE involves optimizing a loss function with two main components. The first is the reconstruction loss, which ensures that the decoder faithfully predicts items that the user has interacted with. For implicit feedback, this is modeled using a multinomial likelihood with log-softmax:

$$\mathcal{L}_{recon} = -\sum_{i \in I} x_{u,i}\ log \frac{exp(\hat{x}_{u,i})}{\sum_{j \in I} exp(\hat{x}_{u,j})}$$

The second is the Kullback–Leibler (KL) divergence term, which regularizes the approximate posterior $q(z|x_u)$ to remain close to a standard Gaussian prior $N(0, I)$:

$$\mathcal{L}_{KL} = \frac{1}{2}\sum_{k=1}^{K}(\mu_k^2 + \sigma_k^2 - log\sigma_k^2 - 1)$$

This regularization prevents the model from overfitting by ensuring that users with similar interaction histories remain close in the latent space. The full loss function is a weighted sum of these terms:

$$\mathcal{L}_{VAE} = \mathcal{L}_{recon} + \beta \mathcal{L}_{KL}$$

where the parameter β controls the trade-off between reconstruction accuracy and regularization. A small β favors precise reconstruction of the training data, while a large β emphasizes smooth and generalizable latent structures. In practice, training often uses KL annealing, where β is gradually increased so the model first learns to reconstruct interactions before focusing on regularization. Once trained, the VAE produces a user–item score matrix where each row corresponds to a user and each column to a predicted score for an item. These ranked lists are then fed into SoCRATe's orchestration module, which applies the chosen fairness-aware allocation strategy. Just as with GNNs, the VAE itself does not decide who gets allocated items; rather, it provides high-quality candidate rankings, and SoCRATe's orchestration preference-driven or round-robin manages allocation under limited availability. After each iteration, SoCRATe evaluates user losses, reorders users based on their disadvantage, and applies the chosen strategy. In preference-driven allocation, the users with the highest loss are compensated first, receiving their full set of recommendations, while in round-robin allocation, items are distributed one by one in cycles to ensure more balanced coverage.

The integration of VAEs into SoCRATe brings several unique strengths. First, by modeling latent distributions rather than fixed embeddings, VAEs provide a natural way of handling sparsity. Even users with few interactions can still be represented meaningfully in the latent space, since uncertainty in their preferences is encoded in the variance terms. Second, the probabilistic framework of VAEs introduces a measure of uncertainty, which can help the system avoid overcommitting to noisy or incomplete user histories. Third, VAEs have the capacity to capture hidden preference structures that are not directly observable from pairwise similarities or graph connectivity, making them complementary to both KNN and GNN approaches.

In the context of fairness, this integration allows us to ask whether the generative power of VAEs contributes to more equitable allocations under SoCRATe. On the one hand, their ability to model uncertainty may lead to more balanced rankings across users, supporting fairer round-robin allocations. On the other hand, preference-driven strategies may interact differently with VAE outputs, potentially amplifying fairness recovery for disadvantaged users or, conversely, introducing new biases if the latent

space captures group-level disparities. By comparing results across KNN, GNN, and VAE within SoCRATe, this thesis aims to provide a deeper understanding of how modern recommenders affect the trade-off between fairness and utility under limited availability.

In summary, the VAE integration extends SoCRATe with a deep generative model that compresses user histories into latent probabilistic representations and reconstructs them into predicted preferences. Through reconstruction likelihood and KL regularization, VAEs not only achieve accuracy but also robustness to sparsity and generalization across users. When their outputs are combined with SoCRATe's fairness logic, they allow us to systematically evaluate how advanced probabilistic recommenders interact with fairness-aware orchestration, providing a valuable counterpart to graph-based methods like GNNs.

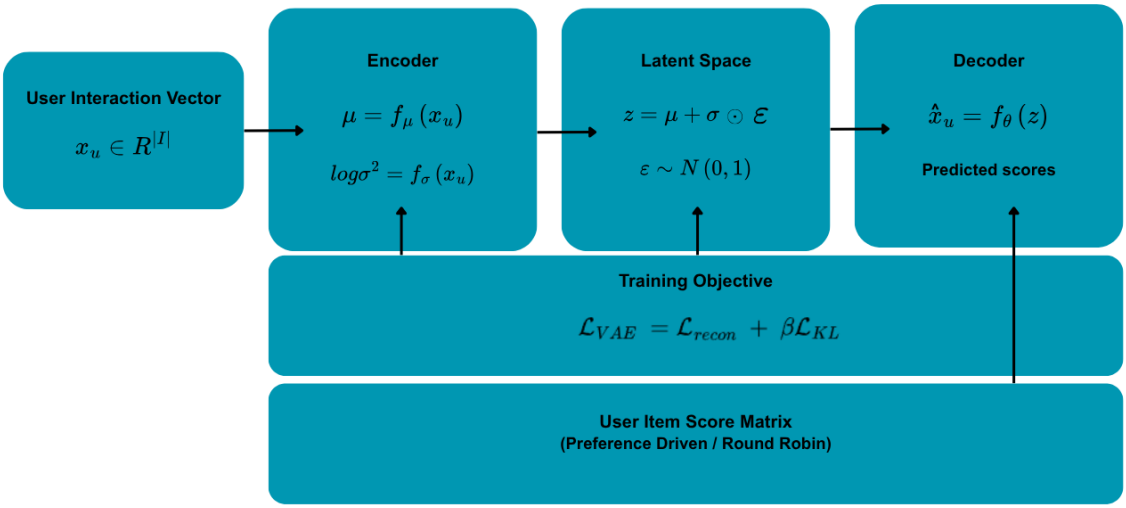

Figure 6. *Pipeline of VAE integration in SoCRATe*

4.5 Implementation and Experimental Setup

The implementation of the extended SoCRATe framework was carried out using Python as the primary programming language, with PyTorch serving as the main deep learning library for model development and training. The implementation followed a modular design, ensuring compatibility with the original SoCRATe structure and allowing each recommender model KNN, GNN, and VAE to be easily integrated into the orchestration workflow. The experiments were performed on a standard computing setup equipped with an NVIDIA GPU to support efficient training of the deep learning models.

At a high level, the system architecture consists of three main layers: the orchestration layer, the recommender layer, and the visualization layer. The orchestration layer includes the modules originally defined in SoCRATe such as the Orchestrator, Weight Update, User Sorting, Compensation Strategy, and Loss Computation which were reused without modification to preserve the original fairness logic. The recommender layer was extended to include two new models: a Graph Neural Network (LightGCN) and a Variational Autoencoder (Mult-VAE). The visualization layer was implemented using Streamlit, providing an interactive interface to observe the evolution of fairness and utility metrics during experiments.

The Graph Neural Network component was implemented using the LightGCN architecture, which was chosen for its computational efficiency and interpretability [4]. The model operates on the user–item bipartite graph constructed from the training data. Each user and item are represented as a node, and observed interactions form the edges. The embeddings are initialized randomly and updated through neighbourhood aggregation for a fixed number of propagation layers. The Bayesian Personalized Ranking (BPR) loss was used for optimization, and early stopping was applied based on validation loss to prevent overfitting. The fairness parameter τ, introduced in Section 4.3, was varied in the range [0.1, 0.5] across experiments to examine its influence on the fairness–utility trade-off.

The Variational Autoencoder component followed the Mult-VAE design proposed by Liang et al. [6], implemented with fully connected encoder and decoder networks. The encoder maps the user interaction vector into a latent space of dimension d=200, producing mean (μ) and variance (σ) vectors that define the latent Gaussian distribution. The decoder reconstructs the user's preference distribution using a softmax activation, producing normalized recommendation scores for all items. The model was trained using the Adam optimizer with a learning rate of 0.001, batch size of 500, and a maximum of 100 epochs. The regularization coefficient β was tuned within the range [0.2, 1.0] to balance fairness and reconstruction accuracy.

To evaluate the framework under controlled conditions, both synthetic and real-world datasets were used. The synthetic dataset was generated to simulate multi-session recommendation scenarios under limited item availability, allowing systematic variation in user behaviour and consumption speed. The real-world datasets included the Amazon Music and Amazon Movie subsets, as well as a crowdsourcing dataset used in prior SoCRATe experiments [1,2]. These datasets were chosen because they contain rich user–item interaction data with inherent sparsity and popularity biases,

making them suitable for studying fairness-driven recommendation systems. Each dataset was split into multiple sessions to mimic iterative consumption behaviour.

For each experiment, the total number of items |I| was determined based on the number of users |U| and the number of iterations T using the formula from the original SoCRATe design [1]:

$$|I| = 2 \cdot \frac{U \cdot (T-1) \cdot |S_u^t| + |U| \cdot N}{a(i)}$$

Where U represents the total number of users, T denotes the total number of iterations, S_u^t is the average number of items consumed per user in iteration t and a(i) is the availability factor representing the average number of users to which an item can be recommended. The workflow proceeds over discrete iterations t=1,2,…,T, each consisting of four main phases: recommendation generation, item allocation, consumption, and fairness compensation. At the beginning of iteration t, each recommender (KNN, GNN, or VAE) produces a ranked list of candidate items for every user. These lists form the actual recommendation set R_t.

To assess how well the system performs relative to an ideal scenario, SoCRATe also computes an optimal recommendation set ($OptR_t$) using a brute-force oracle that maximizes global utility while adhering to item availability constraints. After a recommendation is done the SoCRATe calculates the loss between the recommended list and optimal recommended list as mentioned in loss compensation module under section 3.1. The loss reflects how much satisfaction the user missed due to limited availability or model bias. As the system progresses, SoCRATe maintains a cumulative loss to capture long term disparities in satisfaction across multiple iterations. Users with higher loss have consistently recieved suboptimal recommendations, making them candidate for compensation in the upcoming iterations. To evaluate the system at a global level, SoCRATe defines two aggregate metrics that represent overall performance and fairness which have already been discussed before, the average cummulative loss L(t) which captures total efficiency and standard deviation of cummulative loss S(t) which represents fairness. A smaller value of S(t) indicates a fairer distribution of utility among users, while larger values signify growing inequality. These metrics are recorded after every iteration, allowing researchers to track how fairness and efficiency evolve over time.

Once losses are computed, the system proceeds to the compensation phase, where SoCRATe applies its fairness strategies. Users are sorted based on their cumulative loss values, and two compensation mechanisms can be applied depending on the chosen mode. In the preference-driven strategy, users are served in descending order of cumulative loss: the user with the highest loss receives all N of their top recommendations before the next user is considered. This strategy is most effective for addressing user dissatisfaction quickly and preventing disengagement. Conversely, the round-robin strategy distributes recommendations cyclically, allocating one item per user at a time in loss-based order. This approach prioritizes fairness by ensuring that all users receive a portion of desirable items during each iteration, reducing large disparities in the short term [1,2].

The orchestrator manages this entire cycle, ensuring smooth transitions between recommendation, consumption, and compensation. After each iteration, the cumulative losses are updated, item availability is reduced accordingly, and the next iteration begins. The system continues until all items are exhausted or the maximum number of iterations T is reached.

During experimentation, SoCRATe also supports two types of time granularities to simulate different temporal dynamics. Under the fixed time granularity mode, all users act simultaneously in each iteration, consuming their recommendations at once. Under user-group granularity, users are divided into groups that act sequentially, allowing the system to model asynchronous iterations. This distinction influences how quickly items become unavailable and how fairness is compensated across user segments.

In summary, the SoCRATe workflow models fairness as a cumulative, iterative phenomenon. Each iteration updates user experiences based on the deviation from optimal allocations and redistributes opportunity through compensation strategies. The use of cumulative loss tracking, fairness evaluation metrics, and the formal definition of item allocation guarantees that SoCRATe provides a robust experimental foundation for analyzing fairness-aware recommendation under limited availability. This structure enables direct and meaningful comparison of classical (KNN) and modern (GNN, VAE) recommender systems within a controlled, repeatable setup.

5. RESULTS

This chapter presents the experimental results obtained from the extended SoCRATe framework. The objective of this evaluation is to analyze how different recommender algorithms K-Nearest Neighbors (KNN), Graph Neural Networks (GNN), and Variational Autoencoders (VAE) perform when integrated into the fairness-aware orchestration mechanism of SoCRATe. The results are used to assess the framework's ability to maintain a balance between fairness and efficiency, as well as to observe how model expressiveness and compensation strategies influence user-level satisfaction over multiple iterations.

The experiments were conducted on both synthetic and real-world datasets, following the setup described in Chapter 4. Each dataset simulates a multi-session recommendation environment where item availability is limited and user interactions evolve over time. The metrics analyzed include the average cumulative loss (L), which reflects system efficiency; the standard deviation of cumulative loss (S), which measures fairness. These metrics provide a comprehensive view of how well each recommender model balances the trade-off between utility and fairness.

The results are presented in a structured way to address the research questions defined earlier in this thesis. First, the performance of each recommender model is compared in terms of fairness and utility, examining how expressive models like GNN and VAE perform relative to the baseline KNN. Next, the impact of SoCRATe's compensation strategies preference-driven and round-robin on fairness evolution is analyzed. This is followed by a comparison with the oracle allocation to understand how closely the extended models approximate optimal fairness and efficiency. Further sections discuss generalization across datasets, computational costs, and the sensitivity of the framework to key parameters such as τ and β.

Visualizations from the Streamlit dashboard are included to illustrate the evolution of fairness, utility, and accuracy over multiple iterations. These interactive plots provide an intuitive understanding of how fairness adjustments and recommendation accuracy stabilize as the system progresses. By combining quantitative results with visual insights, this chapter aims to highlight not only the numerical outcomes of the

experiments but also the behavioral patterns that emerge when fairness is embedded into recommendation processes.

5.1 Fairness Utility Trade-off Across Models

One of the central objectives of this study is to evaluate how different recommender algorithms behave within the fairness–utility trade-off established by the SoCRATe framework. While fairness aims to ensure that all users receive comparable satisfaction across recommendation rounds, utility seeks to maximize overall recommendation quality. In practice, achieving both simultaneously is difficult, since improving fairness often requires compensating users with lower historical satisfaction, which can slightly reduce global efficiency. This section compares the performance of the baseline KNN model with the extended GNN and VAE recommenders to understand how model complexity and representation learning affect this trade-off.

The first comparison I did here was on the Amazon movie dataset as it quite sparse which makes it trickier for model to observe deep patterns. The dataset was tested with bigger conflict such as 1 or 2 item availability for every item in dataset, for compensation round-robin was adopted which allocates item in a cyclic fashion and all three recommender model was used to obtain the final mean loss and standard deviation

Availability	Users	Recommender	Average Mean	Average Standard Deviation
1	4	KNN	**0.06964**	0.08359
		GNN	0.12901	0.08126
		VAE	0.45536	**0.04127**
	8	KNN	**0.06964**	0.08359
		GNN	0.13458	0.08576
		VAE	0.45536	**0.04127**
2	4	KNN	**0.06964**	0.08359
		GNN	0.13062	0.08355
		VAE	0.45536	**0.04127**

	8	KNN	**0.06964**	0.08359
		GNN	0.13031	0.08325
		VAE	0.45536	**0.04127**

Table 2. *Performance comparison on amazon movie with different availability and number of users with round-robin compensation strategy*

The performance of the three recommenders on the Amazon dataset under the round-robin compensation setting shows distinct behavioural patterns that remain consistent across all configurations of availability levels and user group sizes. Since lower average mean and standard deviation indicate better performance, these metrics allow us to compare the stability and effectiveness of each model.

Across all experiments, the KNN recommender achieves the lowest average mean loss, remaining around 0.0696 in every configuration. This stability reflects the model's simplicity: KNN relies directly on observed user–item similarity without learning complex latent representations. As a result, its predictions remain robust even when the number of users increases or when the availability of items changes. The relatively moderate standard deviation suggests that although the model performs well, it shows some natural variability due to the direct dependence on local neighbourhood structure. Nevertheless, KNN consistently emerges as the most effective and reliable model for minimizing loss on this dataset.

The GNN recommender shows a different pattern. Its average loss is consistently higher than KNN, typically around 0.129 to 0.134, but still significantly lower than the VAE. This behaviour is expected because GNNs attempt to learn relational information from the user–item graph, which can be more challenging under round-robin allocations where the graph structure is partially influenced by availability constraints. The standard deviations of the GNN model are slightly lower than those of KNN, showing that although the model incurs a bit more error, it produces fairly stable outputs across runs. Interestingly, increasing the number of users or the item availability does not noticeably degrade performance, indicating that the structural learning ability of the GNN remains robust under these settings.

The VAE model behaves quite differently from the other two methods. It consistently shows the highest average mean loss, around 0.455, regardless of the configuration. However, it also exhibits the lowest standard deviation by a wide margin, often around 0.041. This combination of high loss but low variability is characteristic of VAEs trained with a strong latent prior and a KL-divergence regularization term. The regularization

forces the model to produce smooth and generalized latent representations, which leads to very stable but somewhat underfitted outputs. As a result, the VAE's predictions do not fluctuate much across runs, but the model struggles to capture fine grained user-item interactions in this dataset, leading to higher reconstruction loss overall.

When comparing the three approaches, KNN stands out as the best performing model for minimizing loss, showing both robustness and consistency across all conditions. The GNN offers a middle ground: its performance is not as strong as KNN, but its ability to incorporate graph structure allows it to maintain stable behaviour even when the interaction data becomes more complex. The VAE is the most stable yet the least effective in terms of loss, making it suitable only when stability or fairness objectives are prioritized over raw predictive accuracy.

Overall, these results highlight that simpler models such as KNN can outperform more complex neural approaches when data is sparse or controlled through artificial scheduling, as in the round-robin compensation setting. At the same time, models like GNN and VAE may still be valuable depending on the broader goals of the system, such as enforcing fairness, learning deeper structures, or handling uncertainty in user interactions. Since the results are almost similar for the recommenders across different recommendation, we can visualize the loss and fairness for one configuration. Below is the comparison of recommenders for 1 item availability which simulates strong conflict of preference among users.

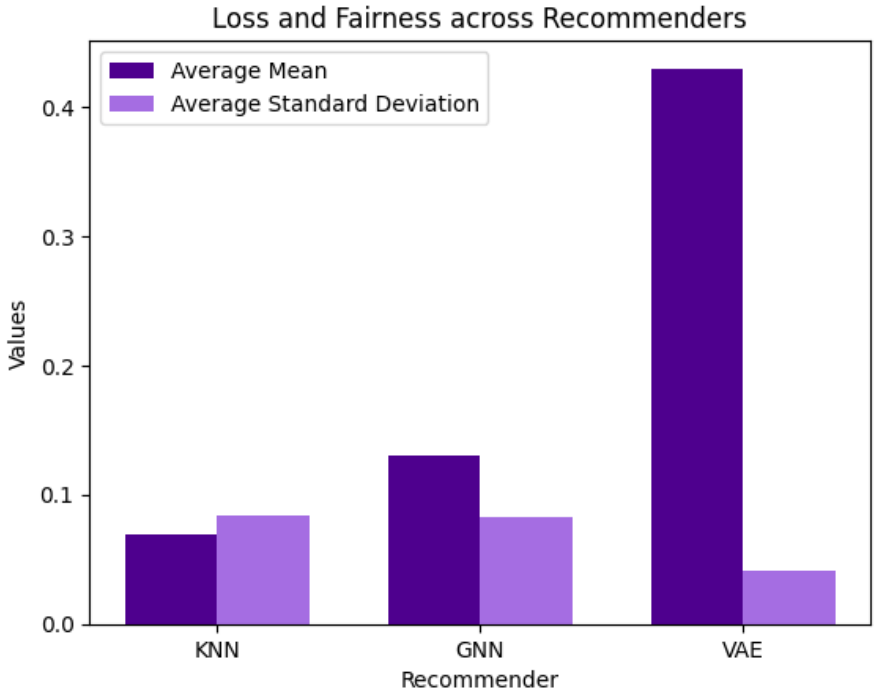

Figure 7. *Average loss (Mean) and fairness (standard deviation) plots across recommenders*

To further explain how VAE handles fairness more efficiently as compared to baseline KNN recommender, we visualized the loss of individual user to see how well they are compensated after several iterations. Normally the first user in the list will have minimum or close to zero loss over each iteration because item allocation will start from the that user and last user in the list will start with heavy loss but then the loss will decrease over the iterations as the user gets compensated. In our context of thousands of users, we will visualize the loss for first two users, the middle one and the last user in the list. Since VAE performs better in terms of overall fairness, we will compare it with baseline KNN recommender to see how user loss is changing with each iteration.

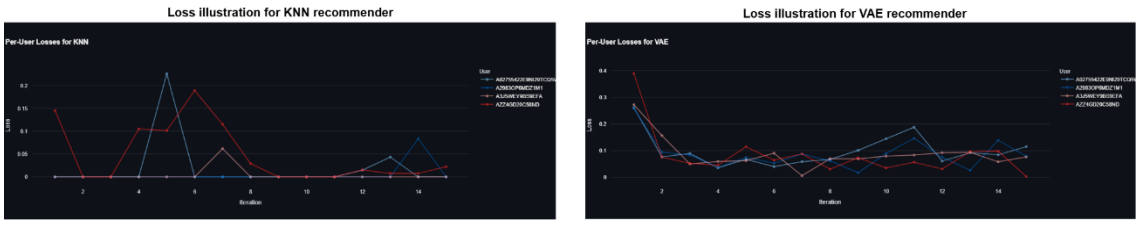

Figure 8. *Loss plot comparison of users in KNN (left) and VAE (right)*

For KNN, the first user does not suffer any loss as the recommended items are the same as optimal recommended items for the user. However other users face loss due to limited item availability. KNN is a shallow model which works on similarity so many

users face high loss and the overall fairness is higher as compared to that of a deep learning model. In case of VAE, even the first user suffers loss in the very first iteration along with other users but the loss for every user gradually decreases and the losses converges at the end of iteration runs. Therefore, despite every user facing some loss, the overall fairness reduces significantly because the algorihtm regularizes the fairness which makes it dependable for sparse datasets.

5.2 Effect of Compensation Strategies

This section examines how SoCRATe's two compensation mechanisms, round-robin and preference-driven, influence the fairness and efficiency of the recommendation process. Both strategies aim to reduce user-level disparities in satisfaction, but they differ in how recommendations are distributed among users. The round-robin strategy allocates recommendations cyclically, ensuring that every user receives an equal chance of obtaining items during each iteration. In contrast, the preference-driven strategy prioritizes users who have accumulated higher losses in previous rounds, effectively giving preference to those who were less satisfied in earlier iterations. The results presented here reflect how these different approaches impact the overall fairness–utility balance for the Amazon Movie dataset with 15 iterations.

Table 3 summarizes the performance of the three recommenders KNN, GNN, and VAE under both compensation strategies. The evaluation uses the average mean loss as an indicator of system efficiency and the standard deviation of cumulative loss as a measure of fairness. Lower values for both metrics indicate better performance.

Recommender	Compensation	Average Mean	Average Standard Deviation
KNN	Round-Robin	0.0696	0.0835
KNN	Preference-Driven	0.3854	0.3098
GNN	Round-Robin	0.1290	0.0812
GNN	Preference-Driven	0.5152	0.3068
VAE	Round-Robin	0.4553	0.0412

VAE	Preference-Driven	0.9480	0.4817

Table 3. *Performance comparison of recommenders under round-robin and preference-driven compensation strategies*

The results show that for all three recommenders, the round-robin strategy achieves significantly lower average mean loss and standard deviation values compared to the preference-driven strategy. This suggests that, in the context of the Amazon Movie dataset, round-robin compensation provides both better efficiency and greater fairness. The preference-driven approach, while theoretically designed to compensate users with higher loss, appears to overcompensate certain users in this case, leading to larger disparities in the overall loss distribution. This results in higher average loss and greater variability among users.

The KNN recommender performs particularly well under round-robin compensation, achieving the lowest mean loss (0.0696) among all models. This indicates that its simple similarity-based approach, when combined with equal distribution of recommendations, maintains efficiency while keeping fairness deviations relatively stable. The GNN recommender shows a slightly higher mean loss but comparable fairness deviation under round-robin, suggesting that its relational learning offers moderate fairness improvements without major efficiency sacrifice. The VAE recommender, although inherently less efficient due to its probabilistic reconstruction process, maintains a notably low standard deviation under round-robin, reflecting that its latent-space modeling promotes more consistent user satisfaction.

Under the preference-driven strategy, however, all models exhibit substantial increases in both mean loss and standard deviation. For instance, the average mean for KNN increases more than fivefold, from 0.0696 to 0.3854, and its fairness deviation grows from 0.0836 to 0.3099. The same trend is observed for GNN and VAE. This outcome suggests that preference-driven allocation, while intended to prioritize users who were disadvantaged in earlier rounds, may introduce instability in the recommendation process when item availability and user demand are tightly coupled, as is the case in the Amazon movie dataset. The algorithm’s effort to aggressively compensate previously unsatisfied users causes the overall recommendation distribution to deviate from optimal fairness equilibrium.

Overall, these findings indicate that the round-robin compensation strategy is more effective for maintaining fairness and efficiency simultaneously in this dataset. Its even distribution of recommendation opportunities across users results in stable loss reduction and consistent fairness over iterations. Preference-driven compensation,

while conceptually fair, requires more careful calibration in scenarios with limited item availability, as excessive prioritization can destabilize the balance between fairness and efficiency. Therefore, for real-world datasets with varying user engagement levels, round-robin serves as the more reliable choice, ensuring equitable recommendation allocation while preserving system performance.

Figure 9. *Average loss (Mean) and fairness (standard deviation) plots across recommenders with round-robin and preference-driven compensation strategies.*

5.3 Comparison with Oracle Baseline and Model Evaluation

To assess how closely the fairness-aware SoCRATe framework can approximate the theoretical fairness–utility optimum, both extended recommenders, GNN and VAE were evaluated against a brute-force oracle baseline. In this thesis, the Oracle refers to an idealized, brute-force recommender that computes the best possible item allocation for each iteration under the exact availability constraints. For every user and session, the Oracle evaluates all feasible combinations of items and selects the allocation that maximizes total utility. This makes the Oracle a theoretical upper bound for instantaneous recommendation quality within a single iteration. However, the Oracle does not optimize across iterations and does not consider fairness or cumulative disadvantage over time. It only ensures that each isolated iteration is as utility-optimal as possible.

Because Oracle operates independently at every iteration, it does not track user-level loss, compensate disadvantaged users, or enforce long-term equity. As a result, it may repeatedly favor the same users or heavily bias toward popular items if that leads to

higher short-term utility. This means that although Oracle is optimal per iteration, it is not optimal with respect to SoCRATe's long-term fairness objective.

This distinction also explains how a learned model, particularly a VAE or GNN combined with SoCRATe's compensation mechanism, can appear to surpass the Oracle in certain metrics. When fairness-aware strategies are applied across sessions, models can accumulate lower long-term loss variance (fairness metric S(t)) or produce more balanced allocations than the per-iteration optimal Oracle. In other words, while Oracle is optimal for utility within a single iteration, it is not optimal for multi-session fairness. Models that incorporate fairness-aware strategies can therefore achieve lower long-term loss variance and more equitable allocation patterns than the Oracle.

For this comparison, the Oracle was implemented through one million brute-force allocation runs on a synthetic dataset. Each run simulated different user–item allocation scenarios under both round-robin (RR) and preference-driven (PD) strategies, repeated for varying numbers of users (4 and 8), item availabilities (1 and 2), and iteration counts (6 and 15). The results of these exhaustive runs establish the optimal mean cumulative loss (L) and standard deviation (S), representing ideal efficiency and fairness respectively. The same configurations were then tested using the SoCRATe framework with GNN, and VAE recommenders.

The comparison focuses on how closely each model's mean loss and fairness deviation align with the Oracle's optimal results, along with an accuracy measure, defined as the percentage of Oracle instances that the recommender matches or exceeds in fairness–utility performance.

The Variational Autoencoder (VAE) recommender shows exceptional performance in approximating Oracle-level results. Table 4 summarizes the comparison of VAE with the oracle baseline under both round-robin and preference-driven configurations. The recommender was tested against 1 millions brute force runs under different configurations and average cummulative loss and standard deviations was calculated for both SoCRATe and brute force. The average standard deviation of SoCRATe was compared in percentile against Brute Force runs for example A_σ value of 99.74% means that SoCRATe average standard deviation was better than 99.74% of the average standard deviation results from brute force runs.

Parameters				Metrics						
Algorithm	a(i)	|U|	T	$\bar{\mathcal{L}}_{BF}^{min}$	$\bar{\mathcal{L}}_{BF}^{mean}$	$\bar{\mathcal{L}}_{VAE}$	$\bar{S}_{BF}^{min}$	$\bar{S}_{BF}^{mean}$	$\bar{S}_{VAE}$	A_{σ}
Round-Robin	1	4	6	0.566	0.5663	**0.025**	0.01439	0.04105	0.018	99.74%
		4	15	0.89586	0.90036	**0.03**	0.016	0.05591	0.018	99.99%
		8	6	0.2331	0.23314	**0.03**	0.01593	0.03442	**0.014**	100%
		8	15	0.4452	0.44527	**0.025**	0.01239	0.03427	0.023	94%
	2	4	6	0.42841	0.42969	**0.011**	0.0068	0.02405	0.007	99.97%
		4	15	0.73719	0.7603	**0.003**	0.0097	0.03243	**0.005**	100%
Preference-Driven	1	4	6	0.74081	0.74225	0.136	0.08404	0.24025	**0.03**	100%
		4	15	1.00211	1.0127	0.036	0.05848	0.21622	**0.025**	100%

Table 4. *Results of synthetic dataset comparing different variants of SoCRATe VAE and Brute Force for various configurations of compensation strategies and values for availability, users and iterations.*

The results indicate that the VAE consistently performs close to or even better than the oracle across most configurations. Under round-robin compensation, the VAE achieves nearly perfect alignment, with accuracy exceeding 94% in all cases and reaching 100% in several. In particular, with four users and two-item availability, the VAE achieves a mean loss of 0.003 and a standard deviation of 0.005 substantially lower than the oracle's average loss and deviation. This confirms that the model's probabilistic latent encoding enables it to generalize fairness patterns efficiently without exhaustive computation.

Fairness outcomes show similar trends. The VAE produces fairness deviation values consistently lower than the oracle, reflecting its inherent ability to equalize user satisfaction through probabilistic modeling. Under the preference-driven strategy, both oracle and VAE exhibit higher average losses and deviations due to prioritization of high-loss users; however, the VAE still maintains perfect accuracy across configurations. This demonstrates that even in more volatile fairness compensation settings, VAE preserves a stable balance between fairness and efficiency.

Overall, the VAE achieves fairness distributions remarkably close to those of the oracle, while operating at a fraction of its computational cost. Its latent variable modeling allows it to reduce disparities among users and adapt effectively to different item availability and user group conditions, validating its suitability as a fairness-aware deep recommender within the SoCRATe framework. An example of standard deviation percentile plot of standard deviation of SoCRATe against brute force runs can be illustrated below.

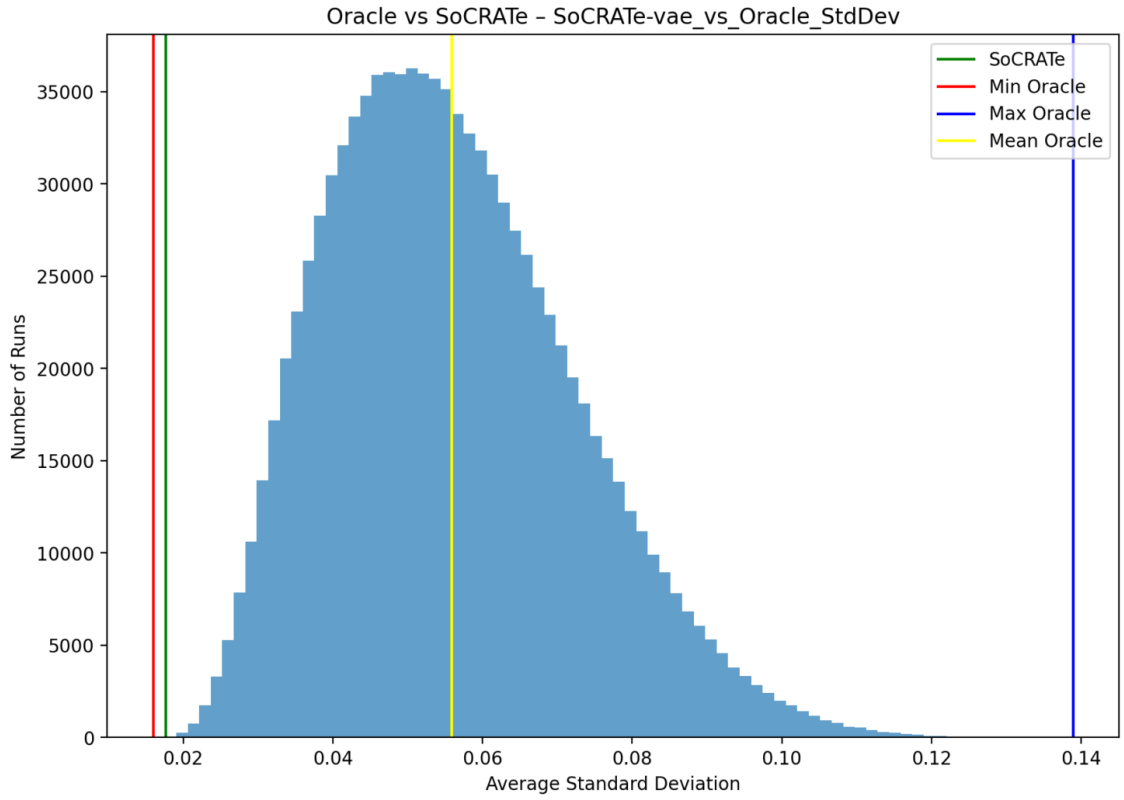

Figure 10. *SoCRATe VAE standard deviation comparsion versus brute force on round-robin compensation strategy*

The Graph Neural Network (GNN) recommender was also tested against the oracle to analyze how relational learning performs under fairness-aware orchestration. Table 5 summarizes the comparison across round-robin and preference-driven settings.

Parameters				Metrics						
Algorithm	a(i)	|U|	T	$\bar{\mathcal{L}}_{BF}^{min}$	$\bar{\mathcal{L}}_{BF}^{mean}$	$\bar{\mathcal{L}}_{GNN}$	$\bar{S}_{BF}^{min}$	$\bar{S}_{BF}^{mean}$	$\bar{S}_{GNN}$	A_{σ}
Round-Robin	1	4	6	0.566	0.5663	0.136	0.01439	0.04105	0.036	60%
		4	15	0.89586	0.90036	0.04	0.016	0.05591	0.015	100%
		8	6	0.2331	0.23314	0.061	0.01593	0.03442	0.019	94.42%
		8	15	0.4452	0.44527	0.0201	0.01239	0.03427	0.018	92.40%
	2	4	6	0.42841	0.42969	0.0092	0.0068	0.02405	0.008	91.70%
		4	15	0.73719	0.7603	0.026	0.0097	0.03243	0.014	99.87%
Preference-Driven	1	4	6	0.74081	0.74225	0.056	0.08404	0.24025	0.031	100%
		4	15	1.00211	1.0127	0.251	0.05848	0.21622	0.032	100%

Table 5. *Results of synthetic dataset comparing different variants of SoCRATe GNN and Brute Force for various configurations of compensation strategies and values for availability, users and iterations.*

The GNN performs very close to the Oracle in most configurations, with accuracy exceeding 90% under nearly all conditions. Under round-robin compensation, it

achieves both low mean loss and stable fairness deviation values, typically between 0.008 and 0.019, comparable to Oracle's optimal values. The consistency of results across different iteration lengths (6 and 15) suggests rapid convergence of graph embeddings, which remain stable over multiple recommendation rounds.

The GNN's relational propagation mechanism allows it to infer fairness patterns across connected users and items, maintaining balance without overcompensating individuals. This results in fairness stability even as item availability or user count changes. Under preference-driven settings, the GNN maintains 100% accuracy, adapting well to the dynamic reweighting of user losses when more iterations are available.

These outcomes confirm that the GNN approximates Oracle-level fairness and efficiency effectively through graph-based relational learning. While it exhibits slightly higher deviations than the VAE in certain configurations, it provides stronger generalization and better efficiency under larger and more complex user–item graphs.

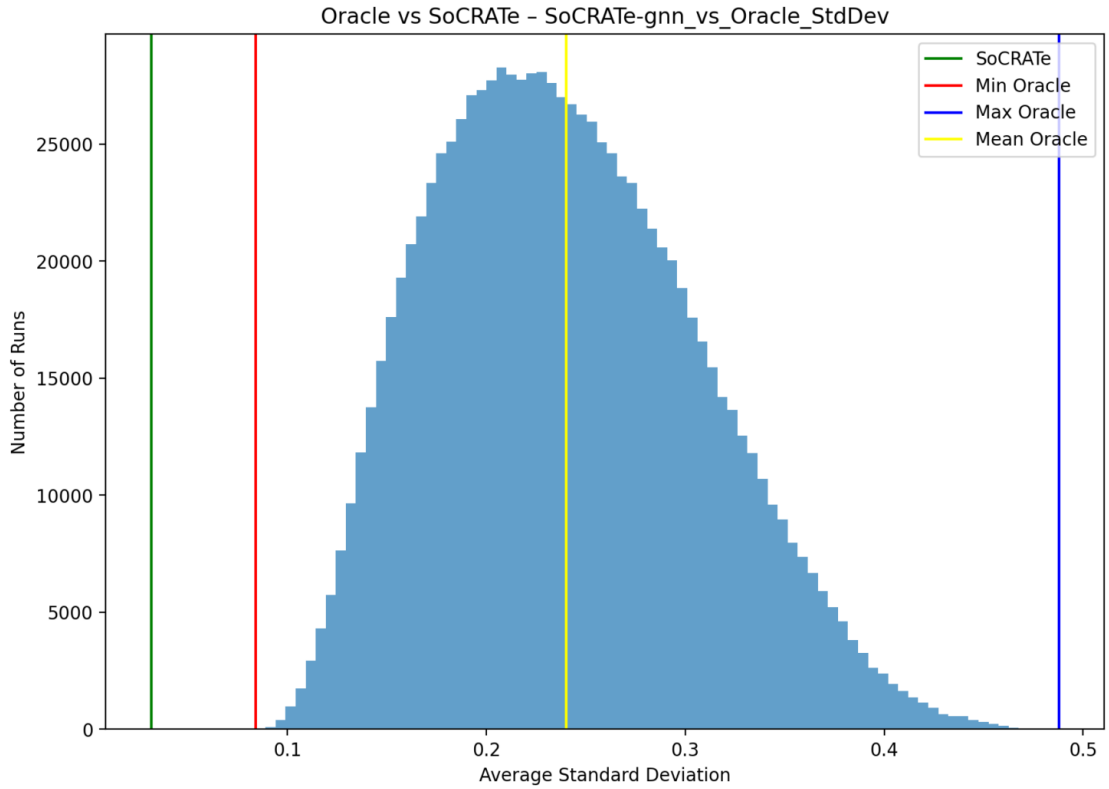

Figure 11. *SoCRATe GNN standard deviation comparsion versus brute force on round-robin compensation strategy*

5.4 Discussion

This section provides an overall discussion of how the integration of advanced recommenders, specifically the GNN and VAE, enhances the fairness–utility balance in the SoCRATe framework. Both models extend the original architecture by introducing learning mechanisms that go beyond traditional similarity-based recommendation. The GNN achieves this by modeling user–item relationships as graphs, allowing fairness information to propagate structurally across users, which leads to more balanced allocation outcomes and stable convergence over iterations. The VAE, in contrast, learns a latent probabilistic representation of user preferences, smoothing disparities in recommendation quality and ensuring equitable treatment across users even under limited availability.

From the results, it is evident that each model optimizes the SoCRATe framework in a different way. The GNN demonstrates a strong balance between fairness and efficiency, making it suitable for dynamic environments where user relationships and item availability change frequently. Its graph-based learning helps maintain consistency in fairness without excessive computational overhead. The VAE, while computationally heavier, is particularly effective when fairness is the dominant goal, as it achieves the lowest fairness deviation and performs reliably across compensation strategies. KNN remains the preferred option when efficiency is the primary objective, offering fast convergence at the cost of higher fairness variation. In conclusion, the integration of GNN and VAE into SoCRATe allows the framework to adapt to a wide range of recommendation contexts, effectively optimizing fairness without significantly compromising utility, and validating the framework’s capacity to generalize fairness-aware learning across different recommender paradigms [1,2,3].

6. CONCLUSIONS

This chapter summarizes the key findings of the study, answers the research questions and reflects on the broader implications of the proposed extension of SoCRATE with advanced deep learning recommender models. It also outlines potential directions for future work.

6.1 Summary of Findings

This thesis investigated how integrating deep learning model can enhance the fairness of a recommender system under limited item availability. The findings reveal:

- The baseline KNN as a traditional similarity-based model, performs reasonably well when the dataset is dense and user behavior is predictable. It's simple, interpretable, and efficient but not ideal when fairness or data sparsity is a concern.
- GNN provides a balanced trade-off. By modeling users and items as nodes in a graph, it captures relationships that KNN misses. This makes it more robust for large, complex datasets where user–item interactions form clear structural patterns. It balances fairness and accuracy effectively.
- **VAE**, on the other hand, stands out when fairness is the priority or when data is sparse and unevenly distributed. Its probabilistic latent representation captures hidden user preferences while maintaining a smooth fairness-satisfaction balance. Even though its numerical loss is higher due to regularization, it generalizes better and treats users more equally.

6.2 Answers to Research Questions

RQ1: How does the choice of recommender algorithm (KNN, GNN, VAE) influence the fairness–utility trade-off within the SoCRATe framework under limited item availability?

KNN usually has the lowest average loss across all recommenders but its fairness value is higher than the rest meaning some user gets better recommendation than the others. GNN and VAE produces slighlty higher average loss but converges better in

terms of fairness meaning they cater all users with better equality than KNN specifically VAE.

RQ2: How do SoCRATe's fairness-aware allocation strategies (preference-driven, round-robin, and adoption models) interact with advanced recommender outputs to influence user compensation and fairness over multiple iterations?

Round-Robin has been more effective as it allocate items to users in a cyclic manner so the user satisfaction remains optimal as compare preference-driven where users are treated one at a time and chances of dissatisfaction increases in case of low item availability. Results also shows that round-robin worked with all recommender and produced better results as compared to preference-driven.

RQ3: To what extent do advanced recommenders (GNN, VAE) approach the fairness and utility bounds of the Oracle brute-force allocation, and how well do these effects generalize across synthetic and real-world datasets?

Both GNN and VAE outperforms brute force by almost 90% under different configurations. Comparing these algorithms with 1 million brute force runs shows that both recommenders achieve good results in both synthetic and real-world datasets.

RQ4: What are the computational and scalability implications of integrating GNN and VAE models into SoCRATe, and how sensitive are fairness outcomes to their hyperparameter configurations?

Both models are slightly expensive in terms of computation but optimal when the dataset is sparse. The fairness does not change significantly when the hyperparameters of the deep learning model are changes and shows only slight difference in average loss and fairness.

6.3 Future Work

Promising directions for future work emerged from both theoretical analysis and practical feedback:

- **Hybrid Fairness-Aware Models:** Combining the representational power of GNN and the generative capabilities of VAE to create a hybrid recommender that captures both relational structure and latent preference distributions.
- **Dynamic Fairness Adaptation:** Introducing a mechanism to dynamically adjust the τ parameter during training, allowing the model to adapt its fairness–utility balance based on user feedback or evolving data distributions.
- **Multi-Domain Evaluation:** Testing SoCRATe on large-scale, real-world, and multi-domain datasets to analyze how fairness and user satisfaction evolve over time in different application contexts.
- **Explainability and Transparency:** Enhancing interpretability by developing explainable fairness mechanisms that allow users and stakeholders to understand why certain recommendations or fairness adjustments are made.

7. REFERENCES

[1] Azzalini, D., Azzalini, F., Criscuolo, C., Dolci, T., Martinenghi, D., Tanca, L., & Amer-Yahia, S. (2022). *SoCRATe: A Framework for Compensating Users Over Time with Limited-Availability Recommendations.* In Proceedings of the 30th Italian Symposium on Advanced Database Systems (SEBD 2022), CEUR Workshop Proceedings, Vol. 3194, pp. 538–545.

[2] Azzalini, D., Azzalini, F., Criscuolo, C., Dolci, T., Martinenghi, D., & Amer-Yahia, S. (2024). *Loss Compensation in Multi-Session Recommendation Under Limited Item Availability.* In Proceedings of the 27th International Conference on Extending Database Technology (EDBT 2024).

[3] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24.

[4] Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S. (2019). Neural Graph Collaborative Filtering. Proceedings of SIGIR ’19, 165–174.

[5] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph Convolutional Neural Networks for Web-Scale Recommender Systems. Proceedings of KDD ’18, 974–983. (PinSage). In Proceedings of KDD ’18, pp. 974–983.

[6] Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018). Variational Autoencoders for Collaborative Filtering. Proceedings of WWW ’18, 689–698.

[7] Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

[8] Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender Systems Handbook (2nd ed.). Springer.

[9] Ekstrand, M. D., Tian, M., Azpiazu, J., Ekstrand, J. D., Anuyah, O., McNeill, D., & Pera, M. S. (2018). All the Cool Kids, How Do They Fit In? Popularity and Demographic Biases in Recommender Evaluation and Effectiveness. Conference on Fairness, Accountability and Transparency (FAT)*.

[10] Desrosiers, C., & Karypis, G. (2011). A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In Recommender Systems Handbook (pp. 107–144). Springer.

[11] Burke, R. (2017). Multisided Fairness for Recommendation. Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML 2017).

[12] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness Through Awareness. Proceedings of ITCS ’12, 214–226.

[13] Beutel, A., Chen, J., Zhao, Z., & Chi, E. H. (2017). Data Decisions and Theoretical Implications When Adversarially Learning Fair Representations. Proceedings of FAT/ML ’17.

[14] Singh, A., & Joachims, T. (2018). Fairness of Exposure in Rankings. Proceedings of KDD ’18, 2219–2228.

[15] Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., & Venkatasubramanian, S. (2015). Certifying and Removing Disparate Impact. Proceedings of KDD ’15, 259–268.

[16] Kamishima, T., Akaho, S., & Sakuma, J. (2011). Fairness-Aware Learning through Regularization Approach. Proceedings of ICDM Workshops ’11, 643–650.

[17] Zehlike, M., Bonchi, F., & Castillo, C. (2017). Fairness in Ranking, Recommendation, and Search Results. Proceedings of CIKM ’17, 1569–1578.

[18] Mehrotra, R., McInerney, J., Biega, A. J., & Lalmas, M. (2018). Towards a Fair Marketplace: Counterfactual Evaluation of the Trade-off Between Relevance, Fairness, and Satisfaction in Recommendation. Proceedings of CIKM ’18, 2243–2251.

[19] Yao, S., & Huang, B. (2017). Beyond Parity: Fairness Objectives for Collaborative Filtering. Proceedings of NeurIPS ’17, 2925–2934.

[20] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019). BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. Proceedings of CIKM ’19, 1441–1450.

[21] Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based Recommendations with Recurrent Neural Networks. Proceedings of ICLR ’16.

[22] Zhao, X., Zhang, W., & Wang, J. (2018). Deep Reinforcement Learning for Online Advertising in Recommender Systems. Proceedings of AAAI ’18, 5759–5766.

[23] Biega, A. J., Gummadi, K. P., & Weikum, G. (2018). Equity of Attention: Amortizing Individual Fairness in Rankings. Proceedings of SIGIR ’18, 405–414.

[24] Rahman, M. M., Boratto, L., Marras, M., & Stilo, G. (2023). Fairness in Graph-Based Recommender Systems: A Survey. ACM Computing Surveys, 55(13s), 1–37.

[25] Wang, Z., Zhu, Y., & Li, Z. (2021). Fair Variational Autoencoder for Recommendation with Implicit Feedback. Proceedings of WSDM ’21, 208–216.

[26] Pitoura, E., Stefanidis, K., & Koutrika, G. *Fairness in Rankings and Recommenders: Models, Methods and Research Directions.* ICDE Tutorial, 2021.

[27] Pitoura, E., Stefanidis, K., & Koutrika, G. *Fairness in Rankings and Recommendations: An Overview.* VLDB Journal, 2022.

[28] Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., & Stefanidis, K. *An Overview of End-to-End Entity Resolution for Big Data.* ACM Computing Surveys, 2020.

[29] Efthymiou, V., Papadakis, G., Stefanidis, K., & Christophides, V. *MinoanER: Schema-Agnostic, Non-Iterative, Massively Parallel Resolution of Web Entities.* arXiv, 2019.

[30] Efthymiou, V., Stefanidis, K., & Christophides, V. *Big Data Entity Resolution: From Highly to Somehow Similar Entity Descriptions in the Web.* IEEE BigData, 2015.

APPENDIX A:

This appendix provides access to public **GitHub** repository that contains the source code, scripts and documentation associated with the development with SoCRATe recommendation extension described in the thesis

The repository includes:

- Source code for the tool includes frontend (streamlit) and backend which includes the SoCRATe components implementation
- Documentation for setup and usage

GitHub Repository:

https://github.com/Shaheer-Imam/SoCRATe-Simulator