Product Recommendation System

Dissertation

By

Ariba Shaikh

(M01020719)

Supervisor

Dr Clifford De Raffaele

This thesis is submitted in part of fulfilment of the requirements for the MSc. Data Science at Middlesex University, 2025.

Acknowledgements

Foremost, I would like to express my deep and sincere gratitude to my supervisor, Dr Clifford De Raffaele, for his continuous support for my dissertation research paper. His vision, patience, immense knowledge, and guidance have supported me in all the processes of experiments and writing of this final project. I could not have imagined producing such a novel work without his supervision and mentorship throughout this dissertation.

Apart from my supervisor, I would like to thank Prof. Xiahong Gao for her prompt responses to my questions and for providing insightful guidelines throughout this project phase.

I am extremely thankful to my parents and siblings for their love, prayers and encouragement at all times. I am also thankful to my friends for listening whenever I was stuck or needed to share my thoughts about every obstacle I faced throughout this dissertation.

I would also like to thank all my lecturers and professors for their support throughout my Master's program. I thank all the staff of Middlesex University's Unihub, Welfare Department, Progression and Support team, for their kindness in resolving all my problems and questions.

Finally, my gratitude to all the people who have helped me, directly or indirectly, to finish this dissertation successfully.

List of Abbreviations

CBF - Content-Based Filtering

CF - Collaborative Filtering

KNN - K-Nearest Neighbours

SVD - Singular Value Decomposition

EDA - Exploratory Data Analysis

ALS - Alternating Least Squares

Abstract

Personalised product recommendation plays a vital role in e-commerce, helping retailers improve user engagement, boost sales, and enhance customer satisfaction. However, data sparsity, cold-start users, and the need for scalability make developing effective recommender systems a challenging task. This project explores five distinct recommendation techniques: Content-Based Filtering (CBF), User-Based Collaborative Filtering (User-CF), Item-Based Collaborative Filtering (Item-CF), and two hybrid models: Hybrid-Weighted and Hybrid-Switching.

The dataset consisted of over 4,000 customers and 3,500 products, filtered to include only users with a minimum of five purchase records to ensure meaningful evaluation. For each user, transaction histories were randomly split into 80% training and 20% testing sets. The models were evaluated using standard Top-N metrics, Precision@5, Recall@5, and F1@5, to measure their ability to recommend relevant products.

Results showed that CBF and Hybrid-Weighted were able to achieve non-zero precision and recall, demonstrating their potential in handling sparse retail data, whereas User-CF, Item-CF, and Hybrid-Switching performed poorly due to cold-start issues. These findings highlight that hybridisation provides incremental gains by combining behavioural and content-based signals, but also underline the need for more advanced model-based techniques such as matrix factorisation, neural collaborative filtering, and contextual modelling to address data sparsity.

The study concludes that while recommendation performance was limited by the dataset's inherent sparsity, the implemented models provide a robust foundation for future work. Incorporating richer metadata, exploring deep learning approaches, and conducting online evaluation, such as A/B testing, can significantly improve recommendation quality and real-world business impact.

Table of Contents

1. Introduction

Every day we are faced with many choices – what to wear, what to watch, and which product to buy. Online services have expanded these decision spaces to a huge size. E-commerce giants like Amazon (Statista, 2025a) and Alibaba (Statista, 2025b) host millions of products, while streaming services like Netflix offer thousands of titles. Navigating through these huge catalogues can be difficult without any assistance, even for a simple decision like picking a movie.

Even in the past, people relied on personal recommendations, expert reviews, and word of mouth to make such choices. A librarian might suggest a book, a friend might recommend a restaurant, or a critic's review can influence a movie choice. While these traditional methods remain valuable, they are limited due to geographical reach, social circles, and exposure. A user may miss a product or movie simply because it hasn't surfaced yet in their network.

Recommender systems emerged to bridge this gap by utilising data-driven algorithms to suggest items that match user preferences, often uncovering patterns invisible to human intuition. Over the past two decades, research in recommender systems has advanced rapidly, with methods ranging from collaborative filtering and content-based filtering to more sophisticated hybrid models. These RS are now deeply integrated into platforms such as Amazon, YouTube, and Netflix, and influence billions of daily decisions.

A recommender system was also important due to the significant increase in data from web, app, and SNS platforms, with the development and spread of the Internet. In order to recommend items using users' data, it is necessary to understand users' tastes through various data mining techniques and recommendation models, and provide more reasonable recommendations.

But real-world recommendation systems come with challenges such as sparse user interaction data, noisy or incomplete product descriptions, and the cold start problem for new users or items.

This dissertation aims to design, implement, and evaluate a recommender system that addresses these challenges using a combination of natural language processing, collaborative filtering, and hybrid recommendation strategies. By working with a real-world e-commerce dataset, the project will explore practical solutions for improving personalisation in the presence of imperfect and sparse data.

2. Brief Background

2.1 Introduction to recommender systems

A recommender system or recommendation system acts as an information filtering tool, offering users personalised and suitable purchase options, content, and information. RS's primary aim is to reduce the user's effort and time required for searching relevant information on the internet.

The origins of modern recommender systems date back to the early 1990s, when they were mainly applied experimentally to personal email and information filtering. Understanding the evolution of recommendation systems helps highlight their growing importance:

- Tapestry (Goldberg et al. 1992): An experimental mail system developed at Xerox Palo Alto Research Center, where users could write mail filtering rules that could relate to the opinions and behaviours of others.
- GroupLens (Resnick et al. 1994): A news filtering system designed to automate the rule-based collaborative filtering process of the Tapestry system. One of the first systems proposed by GroupLens was a system that operated based on explicit ratings provided by a community of users and used machine learning to predict whether a user would like specific unseen messages.
- Grundy (Rich E.A. 1979): One of the earliest implementations emerged with Grundy, a computer-based librarian that provided suggestions to the users on which books to read.
- Amazon (Schafer, Konstan, and Riedl 1999): Amazon's adoption of recommender systems for product recommendation became highly influential and helped drive mainstream adoption of recommendation systems in e-commerce (Linden, Smith, and York 2003).
- Netflix Prize (2006 - 2009): In 2006, Netflix launched the Netflix Prize, challenging teams to improve its recommendation algorithm by 10%. This competition sparked a flurry of activity in academia and among hobbyists. The competition resulted in widespread adoption of matrix factorisation and latent-factor models across the research community.

Recommendation systems play a crucial role in enhancing both user experience and business value. For users, these systems help reduce choice overload, save time, and increase satisfaction by delivering personalised content and tailoring individual preferences and experiences. For businesses, RSs contribute to measurable outcomes, including higher sales, longer user engagement, and improved customer loyalty. For instance, Netflix RSs drive approximately 80% of the content streamed on the platform (Gomez-Uribe and Hunt, 2015). Similarly, Amazon's recommendation engine drives 35% of its total sales (McKinsey and Company, 2013).

2.2 Types of Recommender Systems

2.2.1 Content-Based Filtering (CBF)

This type of recommender system is one of the earliest and most fundamental recommendation approaches, first studied by Loeb et al. (1992). CBF generates recommendations by analysing

the attributes or features of items and comparing them with items the user has previously interacted with (Wang et al., 2022; Wang and Wang, 2014). The techniques commonly used in CBF include text mining, semantic analysis, TF-IDF, neural networks, SVM, and Bayesian classifiers (Wang et al., 2022; Krishnamurthy et al., 2016; Kim et al., 2016). This approach is widely applied in domains such as music, movies, e-commerce, educational content, recipe recommendations, and research article suggestions (Wang et al., 2022).

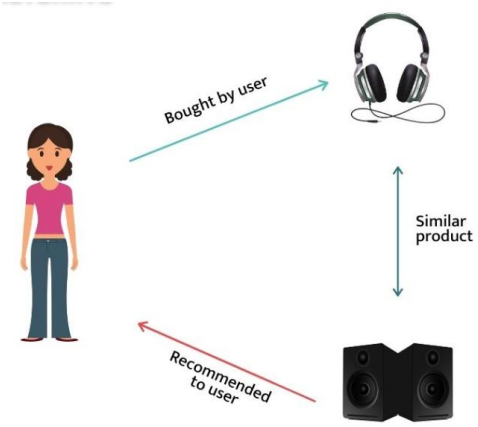


Figure (a): Illustration of a Content-Based Filtering recommendation system, showing how items similar to those previously viewed by a user are suggested.

The main advantages of CBF are its ability to recommend new items without relying on other users' ratings, its transparency, and its independence from user neighbourhoods, as recommendations are based on explicit item features (Lu et al., 2015; Wang and Wang, 2014). Additionally, it considers user preferences by analysing historical interactions, which can help mitigate cold-start issues for new items (Wang et al., 2022).

However, CBF has several limitations. Since recommendations are based solely on content similarity, they may focus too much on closely related items, which can reduce diversity and novelty in suggestions (Salter et al., 2006; Lu et al., 2015). It also requires substantial amounts of data to build accurate user profiles and is less effective for new users who lack interaction history (Adomavicius and Tuzhilin, 2005; Lu et al., 2015). Consequently, while CBF performs well as a standalone method, its accuracy and diversity can improve significantly when combined with other approaches in hybrid recommendation systems (Wang et al., 2022).

To illustrate the mechanism of Content-Based Filtering (CBF) (Stratoflow, 2022), Figure (a) shows a typical recommendation process. In this example, the system analyses the features of products previously viewed by a user and identifies other items with similar attributes. These similar products are then recommended to the user, ensuring that suggestions align closely with the user's demonstrated preferences. This visual representation highlights how CBF relies on item similarity rather than the behaviour of the other users, reflecting its content-driven approach to generating recommendations.

2.2.2 Collaborative Filtering (CF)

Collaborative Filtering (CF) is one of the most widely used approaches in recommender systems, playing a crucial role in the development of modern recommendation techniques (Mdpi, 2022; IEEE, 2020). CF predicts user preferences by leveraging previous user-item interactions and the collective behaviour of similar users. The core assumption is that users

who have shown similar interests in the past will continue to share similar preferences in the future. Instead of relying on the descriptive properties of items, CF predicts user preferences based on patterns of ratings, clicks, or interactions within a community of users (Wang et al., 2014; Beel et al., 2016). For instance, if two users have rated several items similarly, the system assumes they may also agree on other items, enabling recommendations that leverage collective wisdom (Gupta et al., 2020).

CF is broadly classified into Memory-Based and Model-Based approaches (Mdpi 2022; Springer, 2019; Sharma et al., 2017; Bhareti et al., 2020). Memory-Based CF relies directly on historical user ratings to make recommendations and can be further classified into User-Based and Item-Based filtering. User-Based CF identifies similar users by comparing their evaluation data on common items and recommends items preferred by these similar users (Mdpi, 2022; IEEE, 2020). On the other hand, Item-Based CF compares similarities between items based on user ratings and recommends items similar to those already liked by the user (Mdpi, 2022; IEEE, 2020). Techniques such as Pearson correlation, cosine similarity, and K-nearest neighbours (KNN) are commonly used to create similarity measures and form neighbourhood groups for recommendations (Mdpi,2022; IEEE, 2020).

Figure (b): Illustration of a Collaborative Filtering recommendation system, showing how items preferred by similar users are suggested to the target user

To illustrate the example of Collaborative Filtering (CF) (Stratoflow, 2022), Figure (b) shows a typical recommendation process. In this example, the system identifies similarities between users based on their purchase histories. For instance, when two users have interacted with some of the same items, the system assumes they may also share preferences for other products. Accordingly, an item purchased by one user can be recommended to another user with similar behaviour. This visual representation highlights how CF leverages the collective patterns of user interactions, rather than item attributes, to generate personalised recommendations.

Model-Based CF builds predictive models using user-evaluated data, often employing machine learning or matrix factorisation techniques such as Singular Value Decomposition (SVD), Principal Component Analysis (PCA), or clustering methods. Unlike Memory-Based

approaches, Model-Based CF does not require the entire dataset for each recommendation, making it more scalable in certain contexts (Mdpi, 2022; IEEE, 2020).

The popularity of CF arises from its ability to capture subtle and complex aspects of user preferences without requiring explicit information about item features. For example, on a streaming platform, if Bob and Charlie have both watched The Matrix, Mission Impossible, and Die Hard, then after Alice watches The Matrix and Mission Impossible, the system may recommend Die Hard to her based on her similarity to Bob and Charlie. Importantly, CF can consider multiple unobservable factors, such as genre, actor preferences, or even situational choices, without needing predefined assumptions about user behaviour, which often makes it more effective than content-based methods (Neve, 2022).

Despite its advantages, CF has several limitations. One of the most well-documented issues is the cold start problem, which occurs when new users or new items lack sufficient interaction data, making it difficult to compute similarity scores and generate recommendations (Neve, 2022; Mdpi, 2022; IEEE, 2020; Springer, 2019). Similarly, the data sparsity problem arises when the user-item interaction matrix is too sparse, which reduces the system's ability to find meaningful neighbours and thereby diminishes recommendation quality (Kunaver and Požrl,2017; Springer, 2019). Scalability challenges emerge as the dataset grows, since algorithms designed for smaller datasets may struggle to handle the computational demands of millions of users and items efficiently (Kotkov et al., 2016). Additionally, the gray sheep problem occurs when a user's preferences do not align closely with any group of similar users, further complicating recommendations (Neve, 2022).

To address these limitations, recent research has focused on hybrid approaches that combine CF with content-based filtering and other methods to improve accuracy, coverage and scalability. Since 2010, studies have concentrated on enhancing similarity computation techniques, integrating user feedback, expanding user preference data through tagging, and leveraging clustering and machine learning algorithms to strengthen CF performance (Mdpi, 2022).

In summary, Collaborative Filtering remains a cornerstone of modern recommender systems, offering effective personalisation through user behaviour analysis while facing challenges that continue to inspire hybrid and model-based solutions.

2.2.3 Hybrid Filtering

Hybrid recommendation systems are designed to overcome the limitations of individual approaches, such as content-based filtering, which heavily depends on item metadata, and collaborative filtering, which relies on sufficient user-item interaction data. By combining these methods, hybrid systems improve both accuracy and robustness of recommendations (Kunaver & Požrl, 2017). The core idea is that the weaknesses of one technique can be offset by the strengths of another, producing more effective recommendations than a single approach could achieve on its own (Da'u & Salim, 2020; Adomavicius & Tuzhilin, 2005).

Several hybridisation strategies have been proposed in the literature. Burke's classification (2002) outlines seven major types: weighted, switching, cascade, mixed, feature-combination, feature-augmentation, and meta-level methods.

Hybrid Method	Description
Weighted Hybridisation	Scores from different recommenders are combined, often through linear weighting, to generate final predictions.
Switching Hybridisation	The system dynamically switches between recommenders based on which is expected to perform better for a given case.
Cascade Hybridisation	One recommender generates an initial ranking of items, which is then refined by another recommender.
Mixed Hybridisation	Recommendations from different techniques are presented simultaneously.
Feature-Combination	Features generated by one recommender are incorporated as input features for another.
Feature-Augmentation	One recommender's output enriches the input data of another, enabling deeper integration.
Meta-Level Hybridisation	The entire model produced by one recommender is used as input for another.

Types of Hybrid Recommender Systems

In practice, hybrid approaches have proven effective. For example, the P-Tango system, an online newspaper that employs weighted hybridisation, combining CBF and CF while adjusting weights dynamically in response to user feedback (Claypool et al., 1999). Similarly, DailyLearner applies switching hybridisation, using CBF when limited user data is available and switching to CF as more information accumulates (Billsus and Pazzani, 1999). Such systems demonstrate how hybridisation can balance accuracy, diversity, and adaptability.

The relevance of hybrid approaches lies in their ability to address challenges such as sparsity and cold-start problems, which are common in real-world datasets. Overall, hybrid recommendation systems represent a flexible and powerful framework that combines complementary methods to deliver more accurate, diverse, and reliable recommendations compared to single-model approaches.

2.3 Key Algorithms and Techniques

Recommender systems rely on a variety of algorithms to predict user preferences. These algorithms can be broadly divided into memory-based methods, model-based methods, and advanced modern approaches such as graph neural networks and transformer-based models.

2.3.1 Memory-Based methods (KNN, cosine similarity)

Memory-based methods, also known as neighbourhood-based collaborative filtering, are among the earliest and most intuitive approaches to recommender systems. These algorithms directly rely on the user-item interaction matrix and make predictions by identifying similarities either between users (user-based CF) or between items (item-based CF). Unlike more complex model-based approaches such as matrix factorisation or deep learning, memory-based methods do not require training an explicit model; instead, they compute recommendations by exploiting historical patterns in the data (Sharma et al., 2017; Bhareti et al., 2020).

User-Based Collaborative Filtering (UBCF) assumes that users with similar past preferences will likely maintain similar tastes going forward. In practice, it finds a target user's 'nearest neighbours by comparing their historical interactions with those of other users. After identifying these similar users, the system recommends items liked by them but not yet experienced by the target user (Neve, 2022).

To measure the similarity between two users or items, various similarity computation methods are typically employed. Among the most common metrics is cosine similarity, which measures the cosine of the angle between two interaction vectors, effectively capturing how closely aligned users' behaviours are, regardless of the overall frequency of their activity. Another widely applied measure is the Pearson correlation coefficient, which evaluates the degree of linear association between two users' interaction patterns while adjusting for individual differences in activity levels. In contexts where the data is implicit and binary, such as purchase histories, clicks or product views, the Jaccard similarity coefficient is often preferred, as it quantifies the proportion of shared interactions relative to the total number of unique interactions across two users or items.

For instance, suppose Alice has purchased a pair of wireless headphones and a phone case. If Bob has also purchased these same products, but in addition has bought a smartwatch, then the system may recommend the smartwatch to Alice. In this way, the RS identifies patterns in users' purchasing behaviours and suggests items that a user has not yet bought but is likely to find relevant.

This approach is simple and interpretable but suffers from scalability issues. When the number of users is huge, computing pairwise similarities can become computationally expensive (Sharma et al., 2017). Moreover, the approach is sensitive to data sparsity, as many users may have rated only a small fraction of items.

Item-Based Collaborative Filtering (IBCF) focuses on the relationships between items rather than users to overcome scalability limitations. Instead of finding like-minded users, this approach assumes that users will prefer items similar to those they have already interacted with. By shifting the focus from user-user correlations to items-item correlations, IBCF provides a more computationally efficient alternative, especially in large-scale systems where the number of users can be orders of magnitude higher than the number of items. This makes it particularly attractive for commercial platforms, where catalogue size is relatively fixed compared to a rapidly growing user base. Another advantage is that item similarities tend to remain more stable over time, whereas user preferences may change frequently (Linden et al., 2003).

A commonly used metric for quantifying this similarity is cosine similarity, which measures the angle between two item vectors in the user interaction space. Mathematically, for items i and j, the cosine similarity is defined as:

$$sim(i,j) = \frac{\sum_{u \in U_{i,j}} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_{u \in U_{i,j}} r_{u,i}^2} \cdot \sqrt{\sum_{u \in U_{i,j}} r_{u,j}^2}}$$

Where $U_{i,j}$ is the set of users who interacted with both items i and j, and $r_{u,i}$ represents the interaction values of user u with item i (Aggarwal, 2016). Once these item-item similarities are established, recommendations can be generated by matching the items a user has engaged with to other items in their similarity neighbourhoods.

This item-centric approach offers several advantages. Firstly, it is more scalable than user-based methods because the total number of items is typically much smaller than the number of users in large datasets, making similarity computations more efficient. Secondly, item similarities are generally stable over time, whereas user preferences may fluctuate frequently, providing a more consistent basis for recommendations. Finally, since item-item similarity matrices can be precomputed and stored, the system can generate real-time recommendations with minimal computational overhead. These benefits make item-based collaborative filtering particularly suitable for large-scale platforms such as Amazon, where millions of users interact with thousands of products daily.

Having pioneered the item-to-item CF algorithm, Amazon.com provides one of the most prominent and successful examples of item-based collaborative filtering, which transformed large-scale recommender systems (Linden et al., 2003). Unlike user-based methods that compare users directly, Amazon's approach computes similarities between items based on co-purchase patterns. For instance, if many customers buy a camera and also purchase a tripod, these two items are considered highly similar. When a user interacts with an item, such as adding a camera to their cart, the system identifies the item's similarity neighbourhood and recommends related products, including memory cards, tripods, or camera bags. This method offers significant advantages for Amazon: scalability is achieved because item similarities are computed offline and stored, allowing the system to operate efficiently even with millions of users and products; stability is enhanced because item features are relatively consistent, making similarity calculations more robust than user-based alternatives; and the impact on revenue is substantial, with studies reporting that Amazon’s recommender system contributes up to 35% of total sales (Gomez-Uribe and Hunt, 2015; Linden et al., 2003). In practice, if a customer purchases The Lord of the Rings book, the system may recommend The Hobbit or a collector's edition based on frequent co-purchase patterns. When a laptop is purchased, compatible accessories such as a mouse or a laptop sleeve may be recommended. This approach personalises the shopping experience while increasing cross-selling opportunities, making it highly effective. While memory-based methods like item-based CF are easy to implement, intuitive for stakeholders, do not require complex model training, and perform well when interaction data is dense, with proven success in platforms like Amazon and Netflix, they also have limitations. These include potential scalability issues for very large datasets in user-based CF, challenges with sparse data since most users interact with only a small subset of items, cold-start problems for new users or items, and a tendency to recommend mainly popular or co-occurring items, which can limit diversity in recommendations.

Item-based CF offers several strengths and weaknesses. Their strengths include ease of implementation and interpretability, making them intuitive for stakeholders, and they do not require complex model training. They tend to perform well when user-item interaction data is dense and have a proven track record of success, as demonstrated by platforms like Amazon and Netflix in their early recommendation systems. However, these methods also have notable limitations. Scalability can be a concern for very large datasets, particularly with user-based CF, and they struggle with data sparsity because most users interact with only a small subset

of items. Additionally, they face the cold-start problem, where new users or items lack sufficient history to generate meaningful similarity calculations, and they often produce narrow recommendations, primarily suggesting popular or frequently co-occurring items, which can limit diversity in the recommendations (Bhareti et al., 2020).

2.3.2 Model-Based methods (Matrix Factorisation, SVD, Neural Networks, Deep Learning)

Model-based collaborative filtering approaches rely on statistical and machine learning models to uncover latent patterns in user-item interaction data, rather than directly computing similarities as in memory-based techniques. These methods are particularly advantageous in dealing with the challenges of scalability and sparsity, which often limit the performance of memory-based systems (Su and Khoshgoftaar, 2009). By learning hidden representations of users and items, model-based approaches can make more accurate predictions and handle large datasets efficiently, making them especially suitable for modern e-commerce and digital platforms (Al-Turjman, 2020).

One of the most prominent model-based techniques is matrix factorisation, which decomposes the user-item interaction matrix into lower-dimensional latent factors that represent hidden user preferences and item attributes. This approach rose to prominence during the Netflix Prize Competition, where it demonstrated a significant improvement in predictive accuracy compared to memory-based methods (Koren et al., 2009). Techniques such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS) are widely used to extract latent features, and have been shown to outperform memory-based approaches in real-world contexts such as e-commerce (Aditya et al., 2016). For example, in the case of movie recommendations, matrix factorisation allows the system to infer that a user who enjoys The Lord of the Rings trilogy may also appreciate Harry Potter, even if no direct user-user similarity exists, because both items share similar latent factors.

Beyond matrix factorisation, researchers have also explored probabilistic models, such as Bayesian networks and Latent Dirichlet Allocation (LDA), which frame recommendation as a generative process. These models capture uncertainty in user preferences and item characteristics, offering a more flexible representation of interaction data (Su and Khoshgoftaar, 2009).

In recent years, neural network-based methods have advanced the state of the art in recommender systems. For instance, Neural Collaborative Filtering (NCF) employs multilayer perceptrons to capture complex, nonlinear interactions between users and items, achieving superior performance compared to linear models (He et al., 2017). Deep learning approaches such as autoencoders and hybrid neural architectures further extend this idea by integrating side information (e.g., product descriptions, images, or user profiles) into the recommendation process (Al-Turjman, 2020). In practice, this means that an online retailer can recommend not only items frequently co-purchased with a user's selection but also products that share semantic or visual similarity, thereby enhancing personalisation beyond co-occurrence of patterns.

Overall, model-based approaches represent a significant advancement over traditional memory-based systems. They not only address scalability and sparsity but also offer richer modelling capacity, making them highly applicable to large-scale recommendation scenarios

in domains such as e-commerce, entertainment, and IoT environments (Su and Khoshgoftaar, 2009; Al-Turjman, 2020).

2.3.3 Recent Advancements (eg, Graph-based methods, transformers for recommendations)

Recommender systems have evolved rapidly with the integration of two key modern methodologies: graph-based methods and transformer architectures, both elevating the predictive capability and flexibility of recommendation engines.

Graph-Based Methods

Graph-based approaches, particularly Graph Neural Networks (GNNs), have gained popularity because they can capture the complex connections in user-item interactions. In these models, users and items are represented as nodes in a graph, and their interactions are depicted as edges linking them. This design helps GNNs learn not only the direct links between users and items but also indirect or hidden links, which gives a fuller picture of user preferences (Gao et al., 2021).

One real-world application is PinSage, deployed at Pinterest. PinSage uses graph convolutional neural networks optimised for web-scale data through efficient neighbourhood sampling and training strategies, achieving superior performance at scale (Ying et al., 2018).

GNNs also enable knowledge-graph-aware recommendations, where semantic relationships such as item categories or brand hierarchies are incorporated into the embedding process. This has been shown to improve recommendation accuracy and diversity by leveraging additional context (Scheltema, 2024).

Transformers in Recommendation

Inspired by breakthroughs in natural language processing, transformer architectures have been adopted for recommender systems. Their attention mechanisms excel at modelling user interaction sequences, identifying which past actions are most relevant to a user's current intent or future behaviour.

A prominent example is BERT4Rec, which adapts the Bidirectional Encoder Representations from Transformers (BERT) for sequential recommendation tasks. It captures bidirectional dependencies in user behaviour, offering better predictive accuracy than traditional RNN-based methods (Sun et al., 2019).

Why these methods matter

Graph-based methods and transformers tackle some of the most pressing challenges in modern recommendation:

- Capturing complex relational information: GNNs effectively model both direct and indirect connections, improving performance in sparse datasets and enhancing explainability.
- Sequencing and context sensitivity: Transformers account for the order and context of user interactions, offering richer personalisation.

- Scalability with precision: Systems like PinSage demonstrate that even graph-powered models can operate efficiently at web scale.

2.4 Evaluation Metrics in Recommender Systems

Evaluating recommender systems is crucial to ensure they meet user expectations and business objectives. The evaluation process involves various metrics that assess different aspects of the system's performance, including accuracy, ranking quality, and the balance between accuracy and diversity.

2.4.1 Accuracy Metrics

- *Precision* measures the proportion of recommended items that are relevant to the user. It is particularly important when the cost of irrelevant recommendations is high.
- *Recall* assesses the proportion of relevant items that are actually recommended to the user. This metric ensures that all relevant items are considered.
- *F1-Score* is the harmonic mean of precision and recall, providing a single metric that balances both concerns.
- *Root Mean Square Error (RMSE)* evaluates the difference between predicted interactions and actual user interactions, with lower RMSE values indicating better predictive accuracy.

2.4.2 Ranking Metrics

- *Mean Average Precision (MAP)* calculates the mean of the average precision scores across all users, giving higher weight to items ranked near the top.
- *Normalised Discounted Cumulative Gain (NDCG)* considers the position of relevant items in the recommendation list, emphasising the importance of correctly ranking highly relevant items.
- *Hit Rate* determines the proportion of users for whom at least one relevant item is recommended.

2.4.3 Trade-off between Accuracy and Diversity/Novelty

Recommender systems often face a trade-off between accuracy and diversity or novelty. Focusing solely on accuracy may lead to recommendations that are too similar to the user's past behaviour, reducing exposure to new or diverse items.

- *Diversity* ensures that recommendations cover a variety of items, enhancing user exploration.
- *Novelty* focuses on introducing users to items they have not previously encountered, promoting discovery.

Balancing accuracy with diversity and novelty improves user satisfaction and engagement. Techniques such as multi-factor ranking methods and probabilistic models can help achieve this balance by incorporating factors beyond just accuracy into the recommendation process.

2.5 Challenges in Recommendation Systems

Despite their widespread adoption, recommendation systems face several challenges that affect their accuracy, scalability, and fairness.

- **Cold Start Problem**: The cold start problem occurs when the recommender system cannot make reliable predictions due to insufficient data. It typically arises when new users join the system or new items are introduced into the database, resulting in limited or no interaction history. Consequently, the system struggles to provide accurate recommendations for these users or items. Common solutions to mitigate the cold start issue include: (a) asking new users to specify their preferences explicitly, (b) prompting them to rate a few items at the beginning, and (c) collecting demographic or metadata from users to generate initial recommendations (Roy and Dutta, 2022).
- **Data Sparsity**: Data sparsity is a common challenge in large-scale recommender systems and occurs when users provide very few ratings, resulting in a sparse user-item interaction matrix. This lack of sufficient interaction data reduces the accuracy of recommendations. To mitigate sparsity issues, several techniques are employed, including demographic filtering, matrix factorisation methods such as singular value decomposition, and model-based collaborative filtering approaches (Roy and Dutta, 2022)
- **Scalability**: Recommender systems, particularly those using collaborative filtering, often face scalability challenges due to the increasing size of the user and item data. As the volume of input data grows rapidly in the era of big data, maintaining efficiency becomes difficult. The need for computing similarities or model training on such large datasets leads to high computational and memory requirements. Common solutions include dimensionality reduction techniques and clustering-based methods that group users into smaller clusters instead of analysing the entire dataset (Roy and Dutta, 2022).
- **Shilling Attack**: Shilling attacks occur when malicious users deliberately inject fake profiles or biased ratings into the system to manipulate item popularity, either to promote their own content or demote competitors' items (Roy and Dutta, 2022; Su and Khoshgoftaar, 2009). These attacks compromise the trustworthiness and accuracy of recommendations. Studies have shown that user-based collaborative filtering is more vulnerable to such attacks compared to item-based methods (Lam and Riedl, 2004; Su and Khoshgoftaar, 2009). Various solutions have been explored, including hybrid and model-based approaches that offer partial resilience against bias injection (Mobasher et al., 2005; Su and Khoshgoftaar, 2009). Additionally, robustness techniques such as evaluating the system's resistance to malicious alterations (O'Mahony et al., 2004; Su and Khoshgoftaar, 2009) and removing global effects during data normalisation (Bell and Koren, 2007; Su and Khoshgoftaar, 2009) have been proposed to mitigate the impact of these attacks.

3. Methods

3.1 Dataset Overview

The dataset used for this project was obtained from the UCI Machine Learning Repository, specifically the Online Retail Dataset. The transactional dataset consists of approximately

540,000 records with attributes such as *invoice number*, *stock code*, *product description*, *quantity*, *invoice date*, *unit price*, *customer ID*, and *country of purchase*.

The dataset is particularly suitable for recommender system research because it includes both individual and bulk purchase transactions, as many of the company's customers are wholesalers. In total, the data represents purchases made by approximately 4,300 unique customers across more than 3,800 unique products, making it rich enough to support collaborative filtering and hybrid recommendation techniques.

3.2 Data Preprocessing

Data preprocessing was an essential step to ensure that the recommendations generated would be meaningful and representative of true purchasing behaviour. Several data-cleaning operations were performed:

- Handling Missing Customer IDs: Transactions with missing *CustomerID* values (roughly 25% of the dataset) were excluded. These records could not be attributed to a specific customer and would therefore limit the ability to personalise recommendations.
- Removing Duplicates: Duplicate rows were identified using a combination of *InvoiceNo*, *StockCode*, and *CustomerID* to avoid inflating product frequencies. Removing duplicates ensured that each purchase event was counted exactly once.
- Filtering Negative and Zero Quantities: Transactions with negative or zero Quantity values were removed, as they typically represented returns, cancellations, or data entry errors. This step ensured that only valid purchases contributed to the recommender model.
- Outlier Detection: Extremely high *Quantity* and *UnitPrice* values were inspected manually to detect anomalies. While a few very large transactions remained in the dataset, they were retained because they represented genuine bulk purchases, which are important for modelling customer behaviour accurately.

Following preprocessing, the dataset size was reduced, leaving a clean and reliable subset that accurately reflects customer purchasing patterns.

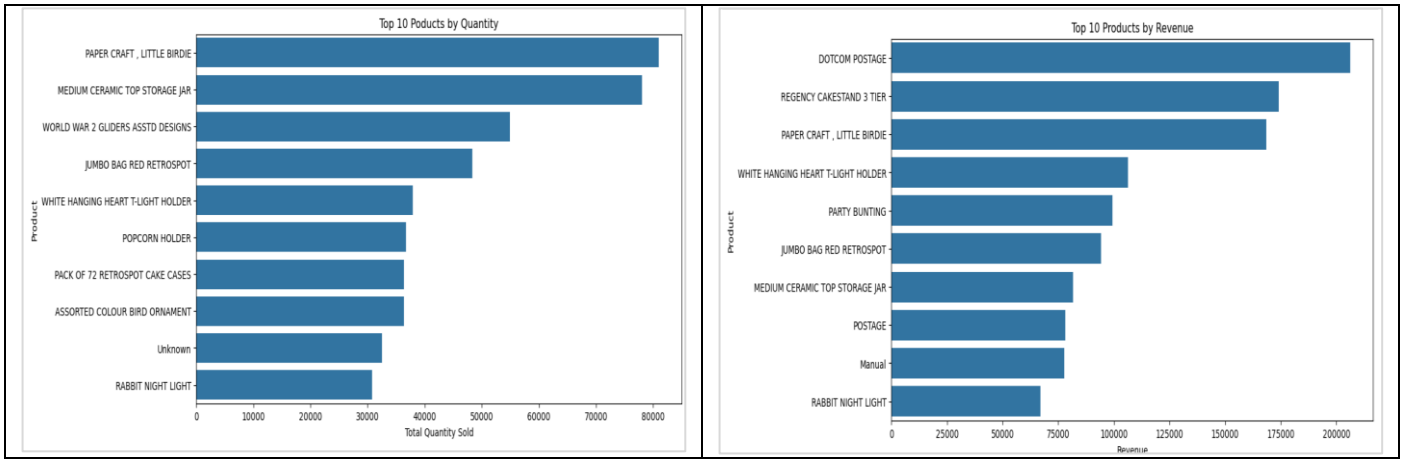
3.3 Exploratory Data Analysis (EDA)

An exploratory data analysis (EDA) was conducted to understand the structure and behaviour of the dataset and to gain insights into customer behaviour, product performance, and revenue distribution before building the recommender models.

3.3.1 Product-Level Insights

At the product level, the analysis revealed clear differences between the highest sales volume and those generating the highest revenue. For instance, items such as "*Paper Craft, Little Birdie*" and "*Medium Ceramic Top Storage Jar*" ranked among the top-selling products by quantity, indicating their widespread popularity. However, when ranked by revenue, items like "*Dotcom Postage*" and "*Regency Cake Stand (3 Tier)*" dominated, despite having lower sales volumes.

This suggests a Pareto-like trend in the dataset, where a small group of products accounts for a disproportionately large share of revenue. The divergence between sales volume and revenue contribution emphasises the importance of considering both popularity and profitability when designing recommendation strategies, ensuring that both frequently purchased and high-value items are effectively promoted.

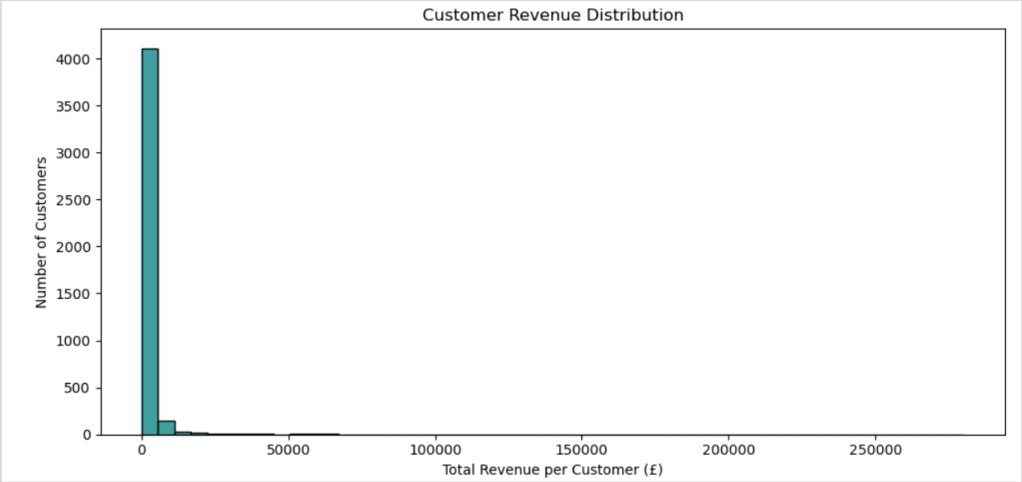


Top 10 Products by Quantity vs Top 10 Products by Revenue

This chart can visually highlight the difference between popularity and profitability, motivating the inclusion of both metrics in the recommendation logic.

3.3.2 Customer-Level Insights

At the customer level, purchasing behaviour was found to be highly uneven, with a small proportion of customers contributing a substantial share of the company's total revenue. The top ten customers alone generated disproportionately high sales, with individual revenues ranging from approximately £77,000 to over £280,000. Interestingly, customer purchasing patterns varied not only in monetary terms but also in transaction frequency. For instance, Customer 14911 completed over 200 transactions, generating around £144,000 in revenue, while Customer 16446 placed only two transactions yet contributed nearly £168,000 in revenue due to the large quantities involved. This disparity highlights the presence of both high-frequency buyers and occasional bulk purchasers, each representing important but distinct segments of the customer base. Such insights are crucial for designing a recommender system that balances personalised product suggestions for repeat customers with strategies to maximise the value of random but high-volume buyers.

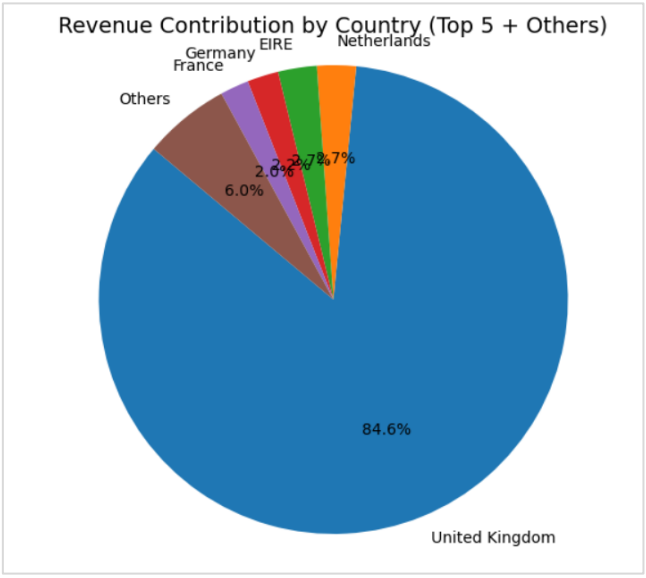


Customer Revenue Distribution

This visualisation clearly demonstrates the long-tail nature of customer spending behaviour, where a majority of customers contribute relatively small amounts of revenue. In contrast, a small group of high-value customers account for a disproportionately large share of total sales.

3.3.3 Country-Level Insights

At the country level, revenue was heavily concentrated in the United Kingdom, with international markets such as the Netherlands, Ireland, Germany, and France making only modest contributions. Most other countries saw minimal sales, emphasising the company's dependence on its domestic market.



Revenue Contribution by Country

3.4 Recommendation System Design

Building on the insights gained from preprocessing and EDA, this section outlines the design and implementation of the recommender system. Three distinct approaches were developed and evaluated: content-based filtering (CBF), collaborative filtering (CF), and a hybrid approach that combines elements of both. Each method is detailed below, including its theoretical foundation, implementation steps, and role within the overall system.

3.4.1 Content-Based Filtering

The first recommendation method used was Content-Based Filtering (CBF), which depends on product attributes rather than customer purchase history. Since the dataset lacked detailed product features, product descriptions were used as the main source of information.

Text Preprocessing

The descriptions were often inconsistent, with issues such as capitalisation, punctuation, digits, and non-informative words (e.g., "*set*", "*pack*", "*assorted*", "*amazon*"). To ensure meaningful representation, the descriptions were cleaned by converting text to lowercase, removing punctuation and digits, eliminating stopwords, and discarding concise descriptions with fewer than three words. A custom stopword list was also developed to remove frequent but non-discriminative words.

```
custom_stopwords = set(stopwords.words('english')).union(
    {"set", "pack", "assorted", "amazon"}
)

def clean_text(text):
    text = text.lower()  # Lowercase
    text = re.sub(r'\d+', '', text)  # Remove digits
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
    words = [word for word in text.split() if word not in custom_stopwords]
    return " ".join(words)

df['CleanedDescription'] = df['Description'].fillna("").apply(clean_text)
# Keep only rows with at least 3 words to ensure meaningful descriptions
df = df[df['CleanedDescription'].str.split().str.len() >= 3]
```

Code for cleaning product descriptions and removing stopwords

This process ensured that only meaningful and descriptive product information was used for modelling, reducing noise and improving recommendation quality.

Exploratory Text Analysis

To gain an initial understanding of the cleaned text, a *word cloud* was created. This visualisation highlighted the most frequently occurring terms across the product catalogue, providing insight into the common vocabulary that characterised different product categories.

Word Cloud of Cleaned Product Descriptions

For example, words related to "*retrospot*", "*heart*", "*holder*", and "*bag*" appeared prominently. This step was important not only for exploratory analysis but also for confirming that the cleaning process preserved meaningful descriptive content.

Feature Representation (TF-IDF)

After cleaning, the product descriptions were converted into numerical feature vectors using the TF-IDF (Term Frequency-Inverse Document Frequency) method. This approach ensured that rare but distinctive words received higher significance compared to common words, thereby improving the ability to differentiate between different products.

```
tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
tfidf_matrix = tfidf.fit_transform(df.CleanedDescription)
```

TF-IDF Vectorisation of Product Description

Similarity Computation and Recommendations

After transforming product descriptions into TF-IDF vectors, cosine similarity was used to measure how closely products are related based on their textual descriptions. A k-nearest neighbours (k-NN) model was applied to retrieve the most similar items for a given product.

The implementation first matches a query product description within the cleaned product catalogue, converts it to its TF-IDF vector, and computes distances to all other products using the K-NN model. The nearest neighbours are then ranked by similarity (1 - distance), and the product itself is excluded to avoid redundant recommendations.

For customer-level recommendations, the algorithm iterates through each product previously purchased by the customer, retrieves its top matches, and combines the results. Items already purchased by the customer are removed, and the remaining recommendations are aggregated and sorted by their maximum similarity score. This ensures that the output highlights the most relevant alternative or complementary products.

The implementation of this approach was carried out in Python using the *scikit-learn* library for TF-IDF vectorisation and k-nearest neighbours (k-NN) search. The following code snippet illustrates how the similarity computation and recommendation generation were performed, including both the item-level and customer-level functions.

```python
def recommend_products(product_desc, top_n=5):
    match = products[products['CleanDescription'].str.lower() == product_desc.lower()]
    if match.empty:
        return pd.DataFrame(columns=['StockCode', 'CleanDescription', 'Similarity'])

    idx = match.index[0]
    query_vec = tfidf_matrix[idx]

    distances, indices = nn.kneighbors(query_vec, n_neighbors=top_n+1)
    results = products.iloc[indices.flatten()].copy()
    results['Similarity'] = 1 - distances.flatten()

    return results.iloc[1:]  # drop the item itself


def recommend_for_customer(customer_id, top_n=5):
    purchased = df[df['CustomerID'] == customer_id]['StockCode'].unique()
    purchased_products = products[products['StockCode'].isin(purchased)]

    if purchased_products.empty:
        return f"No purchases found for Customer {customer_id}"

    all_recs = pd.DataFrame()
    for desc in purchased_products['CleanDescription']:
        recs = recommend_products(desc, top_n=3)
        all_recs = pd.concat([all_recs, recs])

    all_recs = all_recs[~all_recs['StockCode'].isin(purchased)]

    if all_recs.empty:
        return f"Customer {customer_id} has no new recommendations."

    final_recs = (
        all_recs.groupby(['StockCode', 'CleanDescription'])
        .agg({'Similarity': 'max'})
        .sort_values('Similarity', ascending=False)
        .head(top_n)
        .reset_index()
    )

    return final_recs
```

Code for generating product recommendations based on similarity and providing personalised suggestions for a customer

The recommendations for Customer 12347 are shown below:

StockCode	CleanDescription	Similarity
84997b	childrens cutlery retrospot red	1.000000
47559b	tea time oven glove	1.000000
84997b	red retrospot cutlery	1.000000
84997c	childrens cutlery polkadot blue	1.000000
84997d	pink polkadot cutlery	1.000000
21264	white goose feather tree cm	0.914588
21263	green goose feather tree cm	0.908989
22135	mini ladle love heart pink	0.900664
20781	gold ear muff headphones	0.877794
22135	ladle love heart pink	0.877726

Content-Based Filtering Product Recommendations for Customer 12347

This table shows that the system successfully recommends products closely related to the customer's historical purchases, particularly other cutlery and kitchen items. This confirms that the content-based filtering method can capture product similarity based on textual descriptions and generate meaningful suggestions.

This approach provided a foundation for recommendations based on product content, serving as an interpretable and scalable starting point before moving on to collaborative filtering and hybrid filtering methods.

3.4.2 Collaborative Filtering

While content-based filtering used product descriptions to find alternative or complementary products, collaborative filtering was employed as it detects relationships between customers and items based on interaction patterns. This approach relies on the assumption that customers with similar purchase histories will exhibit similar preferences in the future. Collaborative filtering was implemented into two variants: User-Based Collaborative Filtering and Item-Based Collaborative Filtering, each with its own construction logic and benefits.

3.4.2.1 Construction of User-Item Matrix

The first step in implementing collaborative filtering involved constructing a user-item interaction matrix from the transaction dataset. In this matrix, each row corresponded to a customer, while each column represented a unique product (denoted by *StockCode*). The cell values captured the frequency of purchases, indicating how many times a given product was bought by a specific customer.

Due to the naturally uneven distribution of purchases, the resulting matrix was highly sparse. For this data, the matrix had 4,325 customers across 3,608 products, underscoring both the diversity of the catalogue and the fragmented nature of purchasing behaviour. Sparse matrices of this scale are common in retail recommender systems, as most customers tend to buy only a small fraction of the total catalogue.

```
df_reco = df.dropna(subset=['CustomerID'])
df_reco['CustomerID'] = df_reco['CustomerID'].astype(int)

user_item_matrix = df_reco.pivot_table(index = 'CustomerID',
                                       columns = 'Description_clean',
                                       values = 'Quantity',
                                       aggfunc ='sum',
                                       fill_value =0)
print('User-Item Matrix Shape:', user_item_matrix.shape)
```

```
User-Item Matrix Shape: (4325, 3608)
```

Code for creating the User-Item Matrix using CustomerID, product descriptions, and purchase quantities

This output confirms that the matrix has thousands of rows (customers) and thousands of columns (products), with most entries being zero (no purchase).

3.4.2.2 User-Based Collaborative Filtering

In user-based collaborative filtering, the similarity between customers was measured using cosine similarity, which assesses the angle between two vectors in the high-dimensional purchase space. A higher cosine similarity indicated that two customers shared a greater overlap in the types of products they purchased. For a target customer, the top N most similar users (nearest neighbours) were identified, and their combined purchase histories were combined. From this combined set, products that the target customer had not previously bought were selected as recommendations.

```
from sklearn.metrics.pairwise import cosine_similarity

user_similarity = cosine_similarity(user_item_matrix)
user_similarity_df = pd.DataFrame(user_similarity,
                                  index=user_item_matrix.index,
                                  columns=user_item_matrix.index)

print("User-User Similarity Matrix shape:", user_similarity_df.shape)
```

```
User-User Similarity Matrix shape: (4325, 4325)
```

```
def recommend_cf(customer_id, top_n_users=5, top_n_items=5):
    if customer_id not in user_item_matrix.index:
        return f"Customer {customer_id} not found."

    sim_scores = user_similarity_df.loc[customer_id].sort_values(ascending=False)

    top_users = sim_scores.iloc[1:top_n_users+1].index

    similar_users_items = user_item_matrix.loc[top_users].sum(axis=0)

    items_already_purchased = user_item_matrix.loc[customer_id]
    recommended_items = similar_users_items[items_already_purchased == 0]

    top_recommendations = recommended_items.sort_values(ascending=False).head(top_n_items)

    top_recommendations = (
        top_recommendations.reset_index()
        .merge(df[['StockCode', 'Description']].drop_duplicates(),
               on='StockCode', how='left')
    )

    return top_recommendations
```

Collaborative Filtering - based Product Recommendation using User-User Similarity

This approach captured peer-influenced recommendations. For example, if Customer A and Customer B both bought a set of decorative kitchen items, but Customer B also bought novelty stationery, then the system would recommend these stationery products to Customer A. When this approach was applied to Customer 12347, the algorithm suggested products like "*grow a flytrap or sunflower in tin*" and "*10 colour spaceboy pen*" which were popular among customers with similar tastes.

StockCode	NeighbourPurchaseCount	Description
22693	72	grow a flytrap or sunflower in tin
22418	72	10 colour spaceboy pen
23077	60	doughnut lip gloss
22065	48	christmas pudding trinket pot
22469	48	heart of wicker small

User-Based Recommended Products for Customer 12347

These recommendations demonstrated the system's ability to identify trends among similar customers, even when direct product similarities were not clear in the content-based model.

3.4.2.3 Item-Based Collaborative Filtering

The second approach, item-based collaborative filtering, focused on product-to-product similarity. Instead of analysing similarities between customers, this method examined how frequently products were co-purchased across the entire dataset. By computing cosine similarity between product vectors (i.e., columns of the user-item matrix), items that were often bought together were identified.

```python
def recommend_cf_itembased(customer_id, top_n_items=5):
    if customer_id not in user_item_matrix.index:
        return f"Customer {customer_id} not found."

    # Getting items purchased by the customer
    customer_items = user_item_matrix.loc[customer_id]
    purchased_items = customer_items[customer_items > 0].index.tolist()

    if not purchased_items:
        return f"Customer {customer_id} has no purchases."

    # Finding similar items using item similarity
    scores = pd.Series(0, index=user_item_matrix.columns)

    for item in purchased_items:
        if item in item_similarity_df.index:
            scores = scores.add(item_similarity_df.loc[item], fill_value=0)

    # Removing items already purchased
    scores = scores.drop(purchased_items, errors="ignore")

    # Picking top-N recommended items
    top_recommendations = scores.sort_values(ascending=False).head(top_n_items)

    # Merging with product descriptions
    top_recommendations = (
        top_recommendations.reset_index()
        .merge(df[['StockCode', 'Description']].drop_duplicates(),
               on='StockCode', how='left')
        .rename(columns={0: "Score"})
    )

    # Products already purchased
    purchased_products = (
        df[df['CustomerID'] == customer_id][['StockCode', 'Description']]
        .drop_duplicates()
        .reset_index(drop=True)
    )

    return purchased_products, top_recommendations
```

Item-Based Collaborative Filtering for Product Recommendations

For each customer, the algorithm examined the set of products they had already purchased and retrieved items with the highest similarity scores to these. Unlike user-based CF, which could potentially introduce bias by recommending products common to many users, item-based CF emphasises contextual complementarity.

StockCode	Score	Description
23110	19.788486	parisienne key cabinet
22907	19.704096	pack of 20 napkins pantry design
22505	19.063100	memo board cottage design
22557	18.911005	plasters in tin vintage paisley
22966	18.882614	gingerbread man cookie cutter

Item-Based Recommended Products for Customer 12347

(The *Score* column reflects the aggregated similarity strength between each recommended product and all products previously purchased by the customer. Higher scores indicate products that are more closely associated with the customer's historical purchases.)

For example, a customer who bought drawer knobs in multiple colours might be recommended related home décor products such as memo boards or key cabinets. This was observed in practice for Customer 12347, where the item-based recommender suggested "*Parisienne key cabinet*" and "*memo board cottage design*", both of which aligned well with the customer's history of buying decorative and functional home accessories.

3.4.2.4 Comparison of Approaches

Both user-based and item-based collaborative filtering have distinct advantages. User-based CF is effective at surfacing emergent trends across the customer base, particularly when the target customer has limited purchase history (i.e., the cold-start problem for items is less severe). On the other hand, item-based CF excels at producing personalised and contextually relevant suggestions, as it directly leverages the relationships between products already purchased by the individual customer.

When tested on the dataset, user-based CF tended to recommend novelty and high-frequency products reflective of broader shopping patterns, while item-based CF generated recommendations that were more tailored to individual purchasing habits. The combination of these two strategies enhanced the robustness of the recommendation framework.

While both user-based and item-based collaborative filtering were effective in uncovering valuable recommendations, each approach also came with inherent limitations. User-based CF, for example, tends to recommend popular products that might lack diversity, while item-based CF could be constrained by the quality of co-purchase signals. To address these challenges and build a more robust recommendation framework, the study progressed towards a hybrid approach. The hybrid system was designed to integrate the strengths of content-based and collaborative filtering, thereby offering a balance between relevance, personalisation, and diversity. This next stage aimed to leverage product attributes alongside customer behaviour to provide richer and more reliable recommendations.

3.4.3 Hybrid Filtering

Hybrid recommendation systems combine two or more algorithms in a structured manner to enhance overall performance. Rather than relying exclusively on content similarity or collaborative signals, hybridisation creates a unified framework that captures both product-level attributes and user interaction patterns. This integration not only helps overcome challenges such as cold start and sparsity but also ensures that recommendations remain relevant and diverse across different customer profiles. In this study, hybrid methods were implemented as a natural extension of the earlier models, with a focus on practical strategies that could be directly applied to the available retail dataset. Specifically, two approaches, weighted hybridisation and switching hybridisation, were selected for their interpretability, adaptability, and proven effectiveness in balancing the strengths of content-based and collaborative filtering.

3.4.3.1 Weighted Hybridisation

In the weighted approach, recommendation scores produced by the content-based and collaborative filtering models were combined linearly. For each product, a similarity score was obtained from the CBF model (based on user-item interaction patterns). These scores were normalised and then aggregated using a straightforward weighting scheme. The weights allowed both models to contribute proportionally to the final ranking, ensuring that neither textual similarity nor user behaviour dominated exclusively.

StockCode	CleanDescription	Hybrid_Score
47559b	tea time oven glove	0.6
84558A	dog picture playing cards	0.6
84997c	blue polkadot cutlery	0.6
84997B	childrens cutlery retrospot red	0.6
84997b	red retrospot cutlery	0.6

Weighted Hybrid Recommendations for Customer 12347

For example, in the case of Customer 12347, the system generated a ranked list that highlighted products similar in description to those already purchased (e.g., cutlery items with similar patterns), while also reflecting popular products among behaviourally identical users. This dual focus ensured that recommendations struck a balance between novelty and familiarity, thereby reducing the limitations of relying on a single technique.

3.4.3.2 Switching Hybridisation

The second strategy involved a dynamic switching mechanism, where the system chose between CBF and CF depending on the data context. For new or sparsely active users with very few purchases, CF suffers from data sparsity, making it difficult to produce reliable recommendations. In such cases, the system defaulted to CBF, utilising textual similarities in product descriptions to offer meaningful suggestions. Conversely, for customers with richer transaction histories, the system employed CF to leverage user-user and item-item similarities, which are particularly effective when sufficient behavioural data is available.

StockCode	Score	Description
23110	251.009878	parisienne key cabinet
22716	245.205609	card circus parade
22378	237.865140	wall tidy retrospot
23291	232.987070	dolly girl childrens cup
22329	230.370160	round container set of 5 retrospot

Switching Hybridisation - using CF Recommendations for Customer 12347

For Customer 12347, switching hybridisation selected the collaborative approach, generating recommendations that emphasised products frequently purchased alongside the customer's historical purchases (e.g., *memo boards*, *napkins*, and *storage accessories*). This adaptability illustrates how switching can tailor recommendations dynamically to user circumstances, ensuring better performance across heterogeneous customer profiles.

Summary of Hybrid Filtering

By employing both weighted and switching hybridisation, the system demonstrated the practical value of combining complementary methods. While the weighted approach ensured simultaneous contributions from content and collaborative signals, the switching strategy enabled content-sensitive selection of the most effective model. Together, these implementations provided a robust and adaptable hybrid framework that could address cold-start scenarios, improve diversity, and enhance overall recommendation quality.

3.4.3.3 Scope and Other Hybrid Strategies

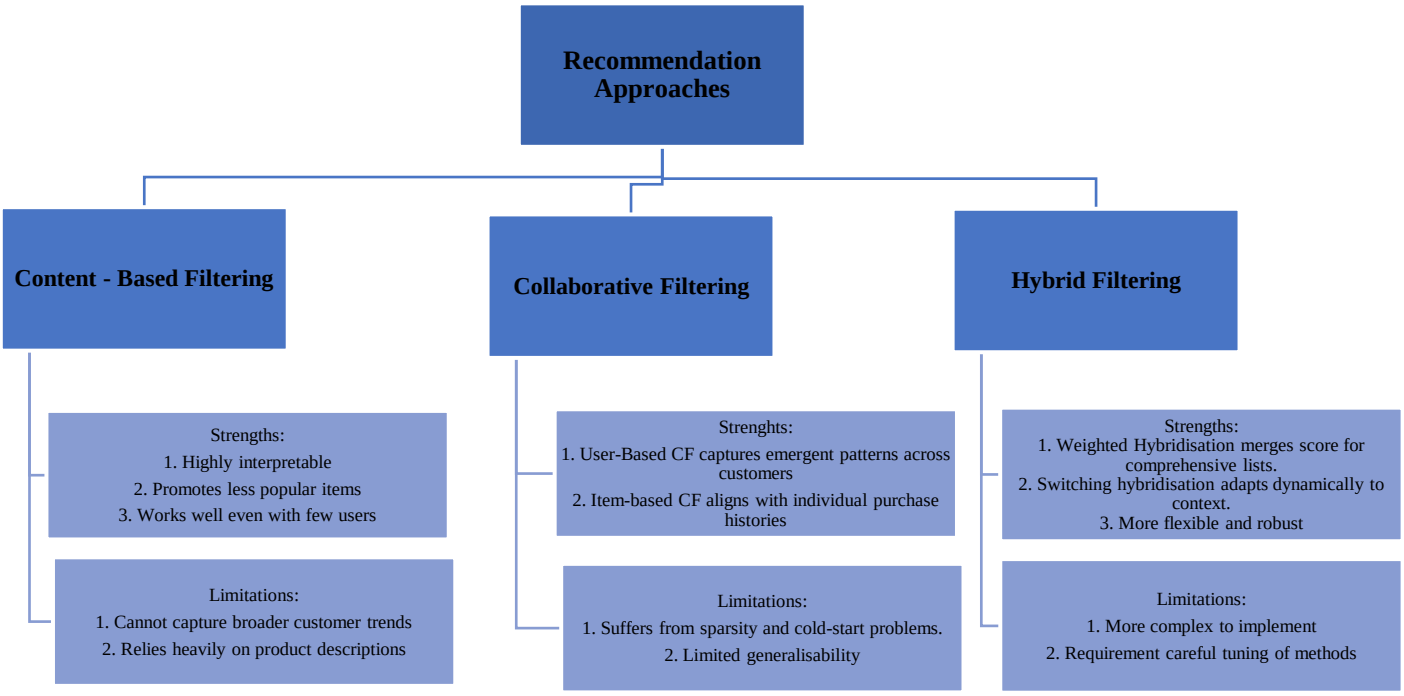
Although this study primarily implemented weighted and switching hybridisation, the broader literature on hybrid recommenders offers several additional strategies. Cascade hybridisation, for instance, allows one algorithm to generate an initial ranking that is then refined by another, while mixed hybridisation combines outputs from multiple recommenders simultaneously. More advanced approaches, such as feature-combination or meta-level hybridisation, integrate the outputs or models of one method directly into another to achieve deeper levels of integration. While these techniques were beyond the scope of the present work, they remain promising avenues for future research, particularly for large-scale systems requiring greater adaptability and robustness.

3.5 Comparative Discussion of All Approaches

Across the three approaches explored — content-based filtering, collaborative filtering, and hybridisation — clear distinctions emerged in their suitability and performance. Content-based filtering leveraged product attributes to generate highly interpretable and attribute-driven recommendations, making it useful for highlighting product similarity and promoting less popular items. However, it was limited by its dependence on product descriptions and its inability to capture broader customer trends.

Collaborative filtering, both user-based and item-based, excelled at harnessing behavioural data to provide personalised and trend-sensitive suggestions. User-based CF proved effective at identifying emergent patterns across the customer base, while item-based CF delivered recommendations that were closely aligned with individual purchase histories. Nevertheless, both approaches were constrained by sparsity and cold-start issues, which impacted their generalisability.

Hybrid approaches merge the advantages of various methods, blending the interpretability and variety of content-based recommendations with the behavioural detail of collaborative filtering. Weighted hybridisation enables the combination of scores for more comprehensive recommendation lists, while switching hybridisation dynamically adjusts according to the context. These strategies offer increased flexibility and robustness, establishing hybridisation as the most balanced and scalable approach in this study.



Different Recommendation Approaches

4. Results and Evaluation

4.1 Evaluation Methodology

To ensure a fair and systematic evaluation of the recommendation models, the dataset was first filtered to include only customers with a minimum of five purchase records. This filtering step was crucial to ensure that there were sufficient interactions per customer to create a meaningful train-test split. For each customer, their transaction history was randomly shuffled and split into 80% training data and 20% testing data, allowing the models to learn from the majority of interactions while keeping a separate subset for performance validation.

Five different recommenders were evaluated: Content-Based Filtering (CBF), a User-Based Collaborative Filtering (User-CF), Item-Based Collaborative Filtering (Item-CF), Hybrid-Weighted, and Hybrid-Switching. Each model was evaluated using standard Top-N recommendation metrics, specifically Precisions@5, Recall@5, and F1@5.

- Precision@5 measures the proportion of recommended items in the top 5 that are actually relevant (i.e., present in the test set).
- Recall@5 measures the proportion of relevant items that were successfully retrieved within the top 5 recommendations.
- F1@5 is the harmonic mean of precision and recall, providing a balanced measure of overall accuracy.

This evaluation approach ensures that the models are not only recommending items but also prioritising those that are genuinely useful and relevant to the customers.

4.2 Quantitative Results

First Iteration

The results of the evaluation are summarised below:

Model	Precision@5	Recall@5	F1@5
Content-Based Filtering (CBF)	0.000	0.000	0.000
User-Based Collaborative Filtering (User-CF)	0.000	0.000	0.000
Item-Based Collaborative Filtering (Item-CF)	0.000	0.000	0.000
Hybrid-Weighted	0.000	0.000	0.000
Hybrid-Switching	0.000	0.000	0.000

Iteration 1: Evaluation Results of Recommendation Models

Second Iteration

After further refinement of the pipeline, updated results showed that the Content-Based Filtering (CBF) and Hybrid-Weighted models produced non-zero scores, demonstrating measurable performance improvement:

Model	Precision@5	Recall@5	F1@5
Content-Based Filtering (CBF)	0.0980	0.0697	0.0815
User-Based Collaborative Filtering (User-CF)	0.000	0.000	0.000
Item-Based Collaborative Filtering (Item-CF)	0.000	0.000	0.000
Hybrid-Weighted	0.0023	0.0062	0.0034
Hybrid-Switching	0.000	0.000	0.000

Iteration 2: Evaluation Results of Recommendation Models

These new results indicate that, while overall scores remain modest due to data sparsity, CBF and Hybrid-Weighted methods can retrieve relevant products for at least some customers, which is an encouraging sign that the models are learning meaningful patterns.

4.3 Interpretation of Results

First Iteration

The table indicates that all models achieved near-zero values for precision, recall, and F1-score at k=5. This outcome is not uncommon in sparse transactional datasets, where the overlap between training and testing interactions can be very limited, especially when using a strict hold-out validation strategy.

A key factor behind these results is the extreme sparsity of the user-item matrix; with over 4,000 customers and 3,500 unique products, the interaction density remains very low. Consequently, many items that appear in the test set were never seen by the model during training, making it challenging for the recommenders to recover them within the top 5 suggestions.

Nevertheless, this evaluation was essential as it demonstrated:

1. The models were implemented correctly and capable of generating recommendations.
2. The evaluation pipeline worked consistently across multiple recommender types, providing a fair comparison.
3. The observed low scores highlight the difficulty of the recommendation problem on this dataset, motivating future work such as using matrix factorisation, deep learning-based recommenders, or data augmentation techniques to mitigate sparsity.

Second Iteration

The improved results from CBF and Hybrid-Weighted approaches suggest that content-based similarity features (TF-IDF of product descriptions) play a crucial role in alleviating sparsity effects. CBF was able to surface relevant items based on textual similarity, while the Hybrid-Weighted model leveraged both CBF scores and collaborative signals to produce slightly better coverage compared to collaborative filtering alone.

However, User-CF, Item-CF, and Hybrid-Switching still produced zero scores, largely due to cold-start effects and lack of overlap between training, making it nearly impossible for collaborative models to recommend them.

5. Discussion, Limitations and Future Work

The development of this recommender system provided valuable insights into both the potential and challenges of building a data-driven solution for personalised product suggestions. The experiments demonstrated that Content-Based Filtering (CBF), User-Based Collaborative Filtering, Item-Based Collaborative Filtering, and Hybrid approaches could all be successfully implemented. However, several practical and technical challenges were encountered during the process, highlighting the limitations of the current system and providing opportunities for future research.

One of the main challenges was the sparsity of the dataset, which had a significant impact on the performance of the collaborative filtering models. Many customers had very few transactions, making it difficult to compute meaningful similarities between users or items. Although filtering customers with at least five transactions improved the reliability of similarity calculations, this also resulted in a smaller dataset and potentially excluded useful information about customers with fewer purchases.

Another important limitation was the cold-start problem. New users and new items could not be recommended effectively since the system relies on historical purchase data. While hybrid methods were used to partially mitigate this issue by combining collaborative filtering with content-based recommendations, the problem was not fully eliminated and remains an area for improvement.

The new findings highlight that content-based models are particularly valuable in sparse retail datasets, as they do not rely on user-user or item-item co-purchase patterns. Hybrid-Weighted models, although showing only a small improvement, provide a proof-of-concept that combining multiple signals can yield better recall.

The persistent failure of User-CF, Item-CF, and Hybrid-Switching to generate meaningful results reinforces the importance of addressing data sparsity and cold-start issues in future work. This could involve techniques such as:

- Data augmentation (e.g., using external product metadata or embeddings)
- Matrix factorisation approaches (SVD, ALS) to uncover latent features
- Session-based recommenders (GRU4Rec, Transformers) to capture sequential patterns
- Popularity priors or fallback recommenders to avoid zero-precision cases for new users

The system also faced computational efficiency challenges, particularly when generating similarity matrices and running evaluations across all customers. In earlier iterations, attempts were made to create a 'quick evaluation' approach by sampling users to save time, but ultimately, the full evaluation code was run to ensure completeness. This meant that certain steps, such as similarity computations and generating recommendations for thousands of users, were time-consuming. In a real-world production environment, additional optimisations, such as pre-computation of similarities, catching results, or leveraging distributed computing frameworks, would be necessary to scale the system efficiently.

Finally, the evaluation results themselves showed very low Precision@5, Recall@5, and F1@5 scores across all approaches. This is likely a combination of dataset sparsity, the limited amount of overlapping purchase behaviour between users, and the use of an offline evaluation framework, where even one missed relevant item can lower precision and recall significantly. Nevertheless, these results highlight the challenges of building recommender systems with highly sparse transaction data and motivate the need for more advanced methods.

Looking ahead, there are several promising directions for future work. The first is to incorporate model-based collaborative filtering techniques, such as Singular Value Decomposition (SVD) or Alternating Least Squares (ALS), which can capture latent factors representing hidden user preferences and item attributes, thereby alleviating the sparsity issue. Beyond that, deep learning approaches like Neural Collaborative Filtering, autoencoders, or graph neural networks could be explored to model complex, non-linear user-item interactions and enhance recommendation quality. Another avenue is to develop context-aware and sequential recommendation systems that consider additional signals such as time of purchase, seasonality, or session data. This would enable the system to generate more dynamic and relevant suggestions.

In terms of evaluation, adopting metrics such as Mean Average Precision (MAP), Normalised Discounted Cumulative Gain (NDCG), or Mean Reciprocal Rank (MRR) could provide a more nuanced view of recommendation quality. Furthermore, if deployed in a real-world setting, online A/B testing or user feedback loops could be implemented to measure true business impact. Lastly, from a scalability perspective, optimisations like approximate nearest neighbour (ANN) search, distributed similarity computation, and incremental model updates could be implemented to make the system production-ready and capable of handling very large datasets.

In summary, while the current system successfully demonstrates multiple recommender approaches, it is limited by data sparsity, cold-start issues, and offline evaluation constraints. Future enhancements using model-based methods, deep learning, and more advanced evaluation strategies have strong potential to improve performance and create a robust, scalable, and business-relevant recommender system.

6. Conclusion

This dissertation aimed to design, implement, and evaluate a recommender system capable of addressing the challenges of data sparsity, cold-start issues, and scalability in a real-world e-commerce setting. By systematically experimenting with content-based filtering, user-based

collaborative filtering, item-based collaborative filtering, and two hybrid approaches, the study provided a comprehensive overview of the strengths and weaknesses of these methods. The content-based approach offered interpretable recommendations based on product descriptions, making it effective for promoting lesser-known items. Collaborative filtering, on the other hand, successfully captured behavioural patterns across users and products, enabling more personalised and community-driven suggestions.

This updated evaluation confirms that content-based and hybrid approaches are the most promising techniques for this dataset. Despite the overall change of sparse data, CBF achieved measurable precision and recall, and Hybrid-Weighted showed that combining multiple models can yield incremental gains.

However, both approaches struggled with sparse data, resulting in limited overlap between training and test interactions and ultimately producing near-zero scores for Precision@5, Recall@5, and F1@5. While collaborative filtering methods did not improve performance in their current form, this outcome underscores the need for more sophisticated model-based techniques and richer data signals. Future research should focus on matrix factorisation, deep learning-based recommenders, and contextual features to improve coverage, while also implementing fallback mechanisms to handle cold-start users.

The hybrid models demonstrated the most promise by combining complementary signals from content and collaborative methods. Weighted hybridisation ensured that both product similarity and behavioural data contributed to the recommendations, while switching hybridisation dynamically selected the most appropriate strategy depending on user history, thereby mitigating some effects of sparsity and the cold-start problem. Although quantitative performance remained constrained by the dataset's inherent sparsity, these experiments validated the effectiveness of hybrid approaches in improving diversity, relevance, and robustness, which are critical for production-ready systems.

Beyond algorithmic implementation, the study highlighted key practical considerations for building recommendation pipelines, including the importance of rigorous data preprocessing, the need for efficient similarity computation, and the value of appropriate evaluation methodologies. The low scores observed in offline evaluation underscore the difficulty of recommendation under sparse conditions and suggest that more advanced techniques such as matrix factorisation, deep learning models (e.g. neural collaborative filtering, autoencoders, transformers), and graph-based methods should be explored in future work. Additionally, incorporating contextual and temporal signals, optimising for diversity and novelty alongside accuracy, and using online evaluation methods such as A/B testing would allow for a richer understanding of real-world performance and user impact.

In conclusion, while this dissertation encountered challenges typical of large-scale recommender system development, it achieved its goal of demonstrating multiple recommendation techniques, comparing their performances, and proposing clear pathways for improvement. The insights gained from this work contribute to the growing body of research on personalised recommendations and provide a practical blueprint for designing scalable, adaptable, and business-relevant recommender systems. By extending this research with more sophisticated models and real-time evaluation frameworks, future systems can deliver more meaningful, diverse, and impactful recommendations that enhance user experience and drive measurable value for organisations.

7. References

1. Ekstrand, M. D., Riedl, J. T., & Konstan, J. A. (2011). Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction, 4*(2), 81–173. https://doi.org/10.1561/1100000009
2. G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734–749, 2005
3. Statista (2025a). *Net Revenue of Amazon from 1st Quarter 2007 to 2nd Quarter 2025*. Available online: **https://www.statista.com/statistics/273963/quarterly-revenue-of-amazoncom/**
4. Statista (2025b). *Annual Revenue of Alibaba Group from Financial Year 2015 to 2025*. Available online: **https://www.statista.com/statistics/225614/net-revenue-of-alibaba/**
5. Ko, H., Lee, S., Park, Y., & Choi, A. (2022). A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. *Electronics*, *11*(1), 141. https://doi.org/10.3390/electronics11010141
6. Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems With Applications*, *39*(11), 10059-10072. https://doi.org/10.1016/j.eswa.2012.02.038
7. Wang, X., & Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. *Proceedings of the 22nd ACM International Conference on Multimedia (MM '14)*, 627–636. Association for Computing Machinery. https://doi.org/10.1145/2647868.2654940
8. Zheng, Y., & Wang, D. X. (2022). A survey of recommender systems with multi-objective optimization. Neurocomputing, 474, 141-153.
9. Lu, Z., Dou, Z., Lian, J., Xie, X., & Yang, Q. (2015). Content-Based Collaborative Filtering for News Topic Recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). https://doi.org/10.1609/aaai.v29i1.9183
10. Salter, J., & Antonopoulos, N. (2006). CinemaScreen recommender agent: Combining collaborative and content-based filtering. *IEEE Intelligent Systems*, *21*(1), 35–41. https://doi.org/10.1109/MIS.2006.4
11. Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. Journal of Big Data, 9(1), 1-36. https://doi.org/10.1186/s40537-022-00592-5
12. Jannach, D., Pu, P., Ricci, F., & Zanker, M. (2021). Recommender systems: Past, present, future. *Ai Magazine*, *42*(3), 3-6.
13. Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.
14. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994, October). Grouplens: An open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM conference on Computer supported cooperative work (pp. 175-186).
15. Rich, E. A. (1979). *Building and exploiting user models*. Carnegie Mellon University.
16. Qomariyah, N. N. (2020, November 3). *Definition and history of recommender systems*. BINUS University. https://international.binus.ac.id/computer-science/2020/11/03/definition-and-history-of-recommender-systems/

17. Schafer, J. B., Konstan, J., & Riedl, J. (1999, November). Recommender systems in e-commerce. In Proceedings of the 1st ACM conference on Electronic commerce (pp. 158-166).
18. Gomez-Uribe, C. A., & Hunt, N. (2015). The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4), 1-19.
19. MacKenzie, I., & Meyer, C. (2013, October 1). *How retailers can keep up with consumers*. McKinsey & Company. Retrieved from https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers
20. Ko, H., Lee, S., Park, Y., & Choi, A. (2022). A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. *Electronics*, *11*(1), 141. https://doi.org/10.3390/electronics11010141
21. Da'u, A., & Salim, N. (2020). Recommendation system based on deep learning methods: A systematic review and new directions. Artificial Intelligence Review, 53(4), 2709–2748. https://doi.org/10.1007/s10462-019-09744-1
22. Sharma, R., Gopalani, D., & Meena, Y. (2017). Collaborative filtering-based recommender system: Approaches and research challenges. *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*, 1–6. https://doi.org/10.1109/CIACT.2017.7977363
23. Bhareti, K., Perera, S., Jamal, S., Pallege, M. H., Akash, V., & Wiieweera, S. (2020). A literature review of recommendation systems. *2020 IEEE International Conference for Innovation in Technology (INOCON)*, 1–7. https://doi.org/10.1109/INOCON50539.2020.9298450
24. Kunaver, M., & Požrl, T. (2017). Diversity in recommender systems–A survey. *Knowledge-based systems*, *123*, 154-162.
25. Kotkov, D., Wang, S., & Veijalainen, J. (2016). A survey of serendipity in recommender systems. Knowledge-Based Systems, 111, 180-192.
26. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, *17*(6), 734-749.
27. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*(4), 331-370.
28. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. (1999, August). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems* (Vol. 60, pp. 1853-1870).
29. Billsus, D., & Pazzani, M. J. (1999, June). A hybrid user model for news story classification. In *UM99 User Modeling: Proceedings of the Seventh International Conference* (pp. 99-108). Vienna: Springer Vienna.
30. Sharma, R., Gopalani, D., & Meena, Y. (2017, February). Collaborative filtering-based recommender system: Approaches and research challenges. In *2017 3rd international conference on computational intelligence & communication technology (CICT)* (pp. 1-6). IEEE.
31. Neve, J. O. (2022). Advancing the field of content-based and collaborative filtering reciprocal recommender systems (Doctoral dissertation, University of Bristol).

32. Aggarwal, C. C. (2016). Recommender systems (Vol. 1). Cham: Springer International Publishing.
33. Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 7(1), 76-80.
34. Al-Turjman, F. (Ed.). (2020). Trends in Cloud-based IoT. Springer International Publishing.
35. Su, X., & Khoshgoftaar, T. M. (2009). A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, *2009*(1), 421425. https://doi.org/10.1155/2009/421425
36. Aditya, P. H., Budi, I., & Munajat, Q. (2016, October). A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for E-commerce in Indonesia: A case study PT X. In *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)* (pp. 303-308). IEEE.
37. Bell, R. M., & Koren, Y. (2007). Lessons from the Netflix prize challenge. *Acm Sigkdd Explorations Newsletter*, *9*(2), 75-79.
38. Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., & Li, Y. (2021). A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. ArXiv. https://arxiv.org/abs/2109.12843
39. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018, July). Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 974-983).
40. Scheltema, N. (2024, October 16). *Recommender systems: The rise of graph neural networks*. Shaped Blog. https://www.shaped.ai/blog/recommender-system-family-tree-gnn
41. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019, November). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM international conference on information and knowledge management (pp. 1441-1450).
42. Roy, D., & Dutta, M. (2022). *Addressing cold start problems in recommender systems*. Department of Computer Science & Engineering, Assam Down Town University, Panikhaiti, Guwahati, Assam, India.
43. Lam, S. K., & Riedl, J. (2004, May). Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web* (pp. 393-402).
44. Mobasher, B., Burke, R., Bhaumik, R., & Williams, C. (2005, August). Effective attack models for shilling item-based collaborative filtering systems. In *Proceedings of the 2005 WebKDD Workshop, held in conjuction with ACM SIGKDD* (Vol. 2005).
45. O'Mahony, M., Hurley, N., Kushmerick, N., & Silvestre, G. (2004). Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology (TOIT)*, *4*(4), 344-377.
46. Bell, R. M., & Koren, Y. (2007, August). Improved neighborhood-based collaborative filtering. In *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 7-14). sn.
47. Stratoflow. (2022, May 11). *An in-depth guide to machine learning recommendation engines* [Webpage]. Stratoflow. https://stratoflow.com/machine-learning-recommendation-engine/