Module Code & Module Title

CU6051NP Artificial Intelligence

Academic Semester: Autumn Semester 2025

Credit: 15 credit semester long module

Student Name: Prakrish Agrahari

London Met ID: 23056832

Assignment Submission Date: 01/21/2026

Submitted To: Mr. Jeevan Prakash Pant

GitHub Link: https://github.com/prakrish11/AI-ProducRecommendationSystem

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded

Contents

List of Figures

1. Introduction

In our modern age, AI is everywhere. It’s in how we navigate transportation, how we communicate, and where we get our entertainment, just to name a few. And AI continues to grow, becoming a more integral part of modern societies by the day. Companies invest billions of dollars into the development of more advanced AI.

At the most basic level, AI functions by taking in data and using an iterative processing system and different algorithms to learn from patterns found in the data, and then react to it in a specific way. Advanced AI can also measure its own performance each time this sequence runs and start iterating and improving its own performance. (tableau.com, 2025)

For hundreds of years, humans have been interested in the possibility of artificial intelligence, long before Alan Turing asked the crucial question: “Can machines think?” The concept of “nonhuman intelligence” traces back to ancient Greek philosophers. The concept of robots goes back to the Renaissance.

History of AI

Late in 1982, Edinburgh professor Donald Michie explained the fundamental error that plagued earlier efforts to create artificial intelligence. “The inductive learning of concepts, rules, strategies, etc. from *examples* is what confers on the human problem-solver his power and versatility, and not (as had earlier been supposed) power of calculation.” A minority position in 1982, learning from examples came to dominate artificial intelligence early in the new millennium. In a key 2009 manifesto celebrating the “unreasonable effectiveness of data,” three Google researchers argued “sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics.” We should “embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data.”

The computer scientist John McCarthy coined the term “artificial intelligence” originally in search of funding; in the mid 2010s, the term was dramatically redefined to describe large-scale algorithmic decision-making systems and predictive machine learning “trained” on massive data sets. Through most of the Cold War and beyond, AI researchers focused on “symbolic AI” largely ignored data collected from everyday and military activities. Such empiricism of the quotidian paled in prestige in comparison with logic and numerical computation and the more empirically oriented approaches such as neural networks and pattern recognition were widely lambasted. Learning from data seemed to be the wrong approach for producing intelligence or intelligent behaviors. Alongside this symbolic approach, in the USA, USSR, and beyond, a far less prestigious empiricist stratum developed comprising congeries of techniques for dealing with large-scale military, intelligence, and commercial data. Our contemporary world of AI, with its often-biased algorithmic decision system, owes far more to this empirical strand of inquiry than to the previously higher status and much studied symbolic artificial intelligence. (L.Jones, 2023)

What is An Algorithm?

An algorithm is a set of step-by-step instructions for solving a problem or completing a task. It tells us exactly what to do and how to get the result. Computers use algorithms to help them make decisions, process data, or perform actions automatically. They can be very simple, like sorting a list of numbers, or very complex, like recommending videos on YouTube.

An algorithm needs to be clear, precise, and finish after a certain number of steps. It should not go on forever without reaching an answer. (Kitthu, 2025)

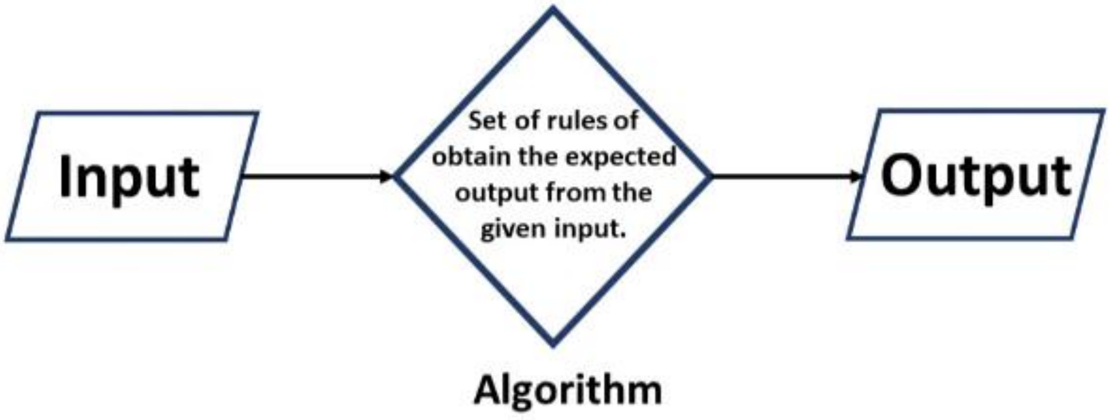

Input
Set of rules of obtain the expected output from the given input.
Output
Algorithm

Figure 1: Algorithm diagram

1.1 Explanation of the topic/AI concepts used

(Product Recommendation System)

Product recommendation systems are an important aspect of retailing because of the improved shopping experience provided for customers. Due to the wide range of products offered by retailers, recommendation systems provide an optimal approach for displaying only relevant products to customers by forming associations that exist between products. Still, it is also important to understand the characteristics of customers connected to different product associations. Conventional approaches for product recommendation systems apply association algorithms and unsupervised classification of customers based on product ratings. (Chibuzor Udokwu, 2023)

AI Concepts Used

- **Artificial Intelligence (AI)**

Artificial Intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. (Copeland, 2025)

AI enables machines to learn from data and make decisions without being explicitly programmed. In this project, AI helps predict which products a user may like.

- **Machine Learning (ML)**

Machine learning (ML) is the subset of artificial intelligence that focuses on building systems that learn and improve as they consume more data. Artificial intelligence is a broader term that refers to systems or machines that mimic human intelligence. (Chen, 2024)

Machine learning allows the system to learn patterns from past user data such as views, purchases, and ratings, and improve recommendations over time.

- **Collaborative Filtering**

Collaborative filtering is like recommending content based on what other people with similar preferences have liked. A collaborative filtering algorithm would analyze the purchase history, browsing behaviour and product ratings of other users with similar interests or shopping habits as the target user. Based on this information, the algorithm will recommend products that similar users have liked or bought, even if the target user has never interacted with these products before. (Kharawl, 2023)

- Recommends products based on **similar users or similar items**.
- Example: If two users like the same products, the system recommends new products liked by one user to the other.
- **Content-Based Filtering**

Content-based filtering is like recommending content based on the content of the movies you like. For example, if a user has previously searched for and purchased running shoes, a content-based filtering algorithm will analyze attributes of those shoes, such as brand, size, colour, and style, and will recommend similar running shoes based on these attributes. (Kharwal, 2023)

- Recommends products based on **product features and user preferences**.
- Example: If a user likes smartphones, the system recommends other smartphones with similar features.

- **Clustering (K-Means Algorithm)**

> Clustering or Cluster analysis is the method of grouping the entities based on similarities. Defined as an unsupervised learning problem that aims to make training data with a given set of inputs but without any target values. (AnalytixLabs, 2022)

- Groups users or products with similar characteristics.
- Helps the system understand different types of users and recommend suitable products.

- **Similarity Measures**
- Techniques such as cosine similarity are used to find how similar users or products are.
- **Model Evaluation**
- Measures how accurate and useful the recommendations are using metrics like accuracy and error rate.

1.2 Explanation/Introduction of the chosen domain/topic

Matching the right customers with the perfect products is an essential task for any ecommerce business, which is why product recommendations are so important for success.

But recommending items online isn’t a simple job. While customers can get tailored recommendations from a sales rep in brick-and-mortar stores, ecommerce brands need to build recommendation systems that determine which popular products are right for their particular audience.

Modern shoppers want personalized interactions, so much so that it can make or break their relationship with a brand. 56% of customers are more likely to return to sites offering recommendations, while 74% of customers feel frustrated by non-personalized content.

With so much riding on the success of your ecommerce site’s recommendations, it’s worth diving into the hows and whys behind product recommendation engines. (Warhurst, 2025)

1.3 Aims and Objectives

Aim of the Project

- To build an AI-based product recommendation system that suggests the right products to users based on their interests and past behavior.

Objectives of the Project

Data and Preparation

- To collect product and user data needed for recommendations.
- To clean the data and remove errors or missing values.
- To prepare the data so it can be used by AI models.

Understanding the Data

- To study how users interact with products.
- To find patterns such as popular products and user preferences.
- To understand which data features are important for recommendations.

Building the Recommendation System

- To create a system that recommends products based on similar users.
- To recommend products based on product features and user interests.
- To group similar users or products using AI techniques like clustering.

Training and Improving the Model

- To train the AI model using past data.
- To improve the model so recommendations become more accurate.
- To reduce incorrect or irrelevant recommendations.

Testing and Evaluation

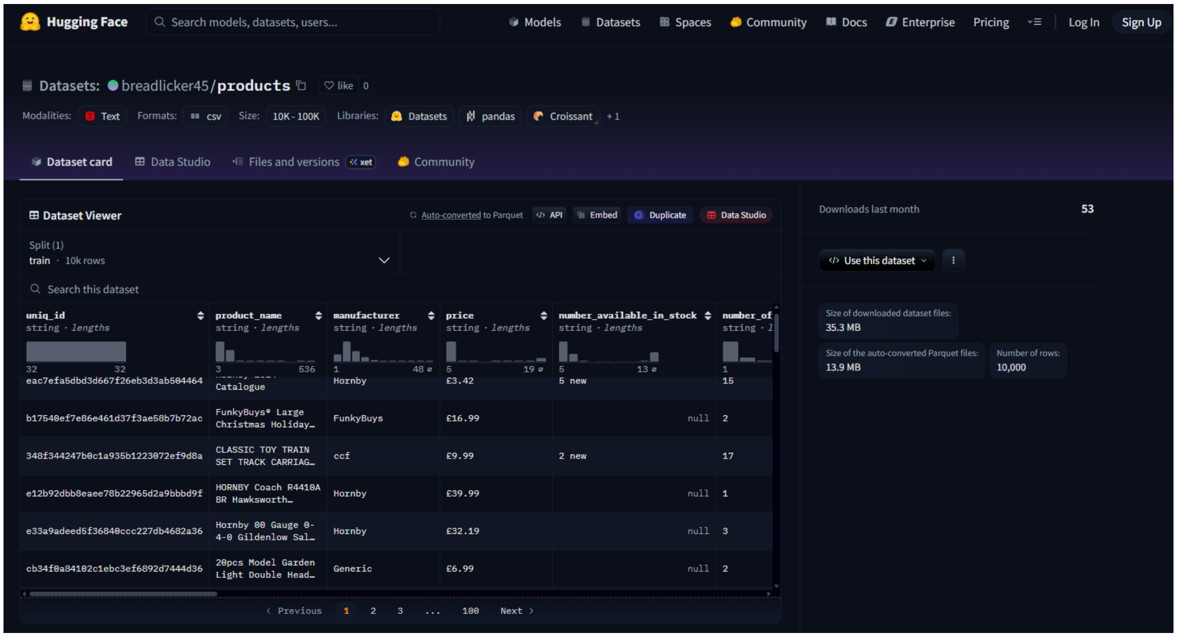
- To test how well the recommendation system works.
- To measure accuracy and usefulness of the recommendations.

Implementation and Usage

- To build and test the system using Jupyter Notebook.
- To show how the system can be used in an online store or application.

1.4 Dataset

Source: https://huggingface.co/datasets/breadlicker45/products



Modalities: Text Formats: csv Size: 10K - 100K Libraries: Datasets pandas Croissant +1

Split (1)
train · 10k rows

uniq_id string · lengths	product_name string · lengths	manufacturer string · lengths	price string · lengths	number_available_in_stock string · lengths	number_of string · l
32 32	3 536	1 48	5 19	5 13	1
eac7efa5dbd3d667f26eb3d3ab504464	Catalogue	Hornby	£3.42	5 new	15
b17540ef7e86e461d37f3ae58b7b72ac	FunkyBuys® Large Christmas Holiday…	FunkyBuys	£16.99	null	2
348f344247b0c1a935b1223072ef9d8a	CLASSIC TOY TRAIN SET TRACK CARRIAG…	ccf	£9.99	2 new	17
e12b92dbb8eaee78b22965d2a9bbbd9f	HORNBY Coach R4410A BR Hawksworth…	Hornby	£39.99	null	1
e33a9adeed5f36840ccc227db4682a36	Hornby 00 Gauge 0-4-0 Gildenlow Sal…	Hornby	£32.19	null	3
cb34f0a84102c1ebc3ef6892d7444d36	20pcs Model Garden Light Double Head…	Generic	£6.99	null	2

Downloads last month 53

Size of downloaded dataset files: 35.3 MB

Size of the auto-converted Parquet files: 13.9 MB

Number of rows: 10,000

Figure 2: Dataset screenshot

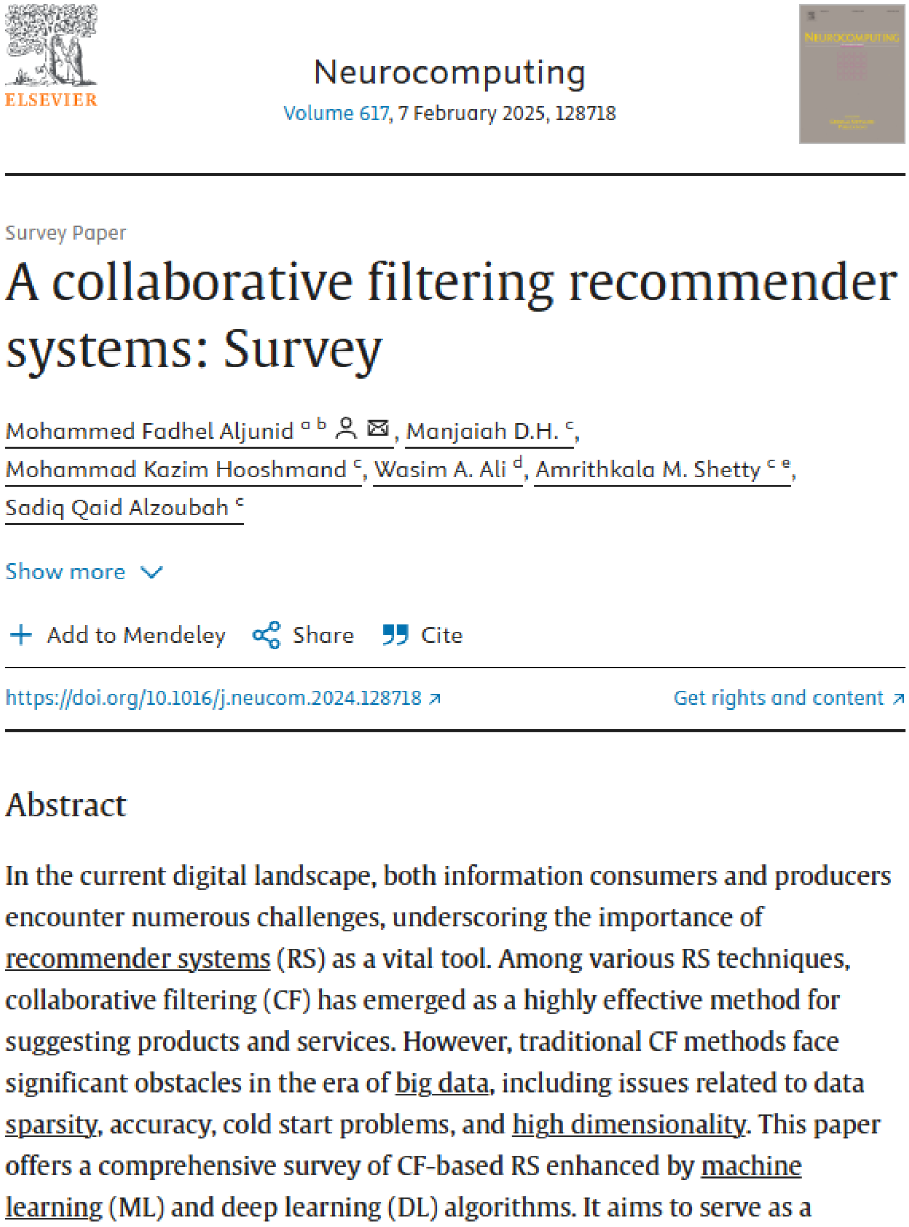
The "products" dataset by user breadlicker45 on Hugging Face is a collection of product-related data, likely scraped or curated for machine learning tasks such as classification, recommendation systems, or natural language processing. It contains textual descriptions, categories, and possibly images of various consumer products (e.g., electronics, clothing, or household items), with metadata including product names, prices, and attributes. The dataset appears to be moderately sized, potentially in the thousands of entries, and is formatted in a tabular structure (e.g., CSV or JSON) suitable for training models on product categorization or sentiment analysis. It's publicly available under an open license, making it useful for researchers and developers working on e-commerce applications, though users should verify data quality and ethical sourcing as it's user-contributed. For more details, check the dataset's page on Hugging Face.

2. Background

To enhance the product recommendation system here are some researches behind the Ai topics, algorithmics and existing work to solve the problem domain.

2.1 Research work done

2.1.1 Collaborative Filtering Recommender Systems: Survey (2025)



ELSEVIER

Neurocomputing

Volume 617, 7 February 2025, 128718

Survey Paper

A collaborative filtering recommender systems: Survey

Mohammed Fadhel Aljunid [a] [b], Manjaiah D.H. [c], Mohammad Kazim Hooshmand [c], Wasim A. Ali [d], Amrithkala M. Shetty [c] [e], Sadiq Qaid Alzoubah [c]

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.neucom.2024.128718 ↗ Get rights and content ↗

Abstract

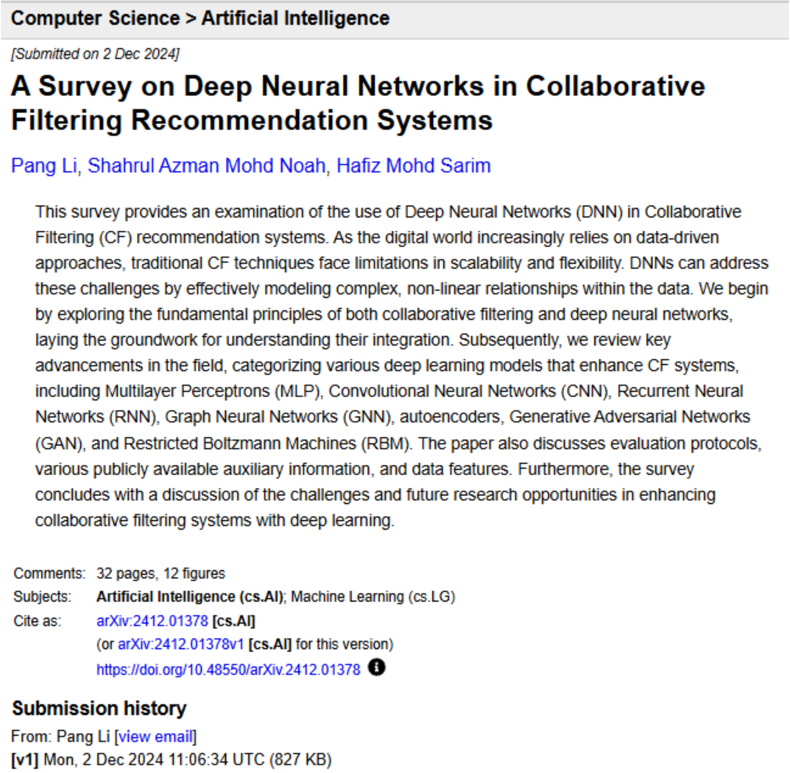
In the current digital landscape, both information consumers and producers encounter numerous challenges, underscoring the importance of recommender systems (RS) as a vital tool. Among various RS techniques, collaborative filtering (CF) has emerged as a highly effective method for suggesting products and services. However, traditional CF methods face significant obstacles in the era of big data, including issues related to data sparsity, accuracy, cold start problems, and high dimensionality. This paper offers a comprehensive survey of CF-based RS enhanced by machine learning (ML) and deep learning (DL) algorithms. It aims to serve as a

Figure 3: Research article 1

(Mohammed Fadhel Aljunid, 2025)

The current 2025 paper offers an extensive literature review of collaborative filtering (CF) recommender systems, namely the efforts to address the classic challenges with the help of modern machine learning (ML) and deep learning (DL) methodology. The survey describes how CF is an effective technique to propose products and services (as with those applied by Amazon or Netflix). Traditional CF techniques have issues with big data such as data sparsity and accuracy, the cold-start problem (challenges with new users/products), and high dimensionality. The paper is organized so as to attract new and experienced researchers first the basic concepts of the recommender systems and then, the specific MC and DL-based recommendations to solve these long-standing CF problems, also, the tasks, metrics, and datasets applied in recent research.

2.1.2 Deep Neural Networks in Collaborative Filtering (2024)



Computer Science > Artificial Intelligence

[Submitted on 2 Dec 2024]

A Survey on Deep Neural Networks in Collaborative Filtering Recommendation Systems

Pang Li, Shahrul Azman Mohd Noah, Hafiz Mohd Sarim

This survey provides an examination of the use of Deep Neural Networks (DNN) in Collaborative Filtering (CF) recommendation systems. As the digital world increasingly relies on data-driven approaches, traditional CF techniques face limitations in scalability and flexibility. DNNs can address these challenges by effectively modeling complex, non-linear relationships within the data. We begin by exploring the fundamental principles of both collaborative filtering and deep neural networks, laying the groundwork for understanding their integration. Subsequently, we review key advancements in the field, categorizing various deep learning models that enhance CF systems, including Multilayer Perceptrons (MLP), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Graph Neural Networks (GNN), autoencoders, Generative Adversarial Networks (GAN), and Restricted Boltzmann Machines (RBM). The paper also discusses evaluation protocols, various publicly available auxiliary information, and data features. Furthermore, the survey concludes with a discussion of the challenges and future research opportunities in enhancing collaborative filtering systems with deep learning.

Comments:	32 pages, 12 figures
Subjects:	**Artificial Intelligence (cs.AI)**; Machine Learning (cs.LG)
Cite as:	arXiv:2412.01378 **[cs.AI]**
	(or arXiv:2412.01378v1 **[cs.AI]** for this version)
	https://doi.org/10.48550/arXiv.2412.01378

Submission history

From: Pang Li [view email]
[v1] Mon, 2 Dec 2024 11:06:34 UTC (827 KB)

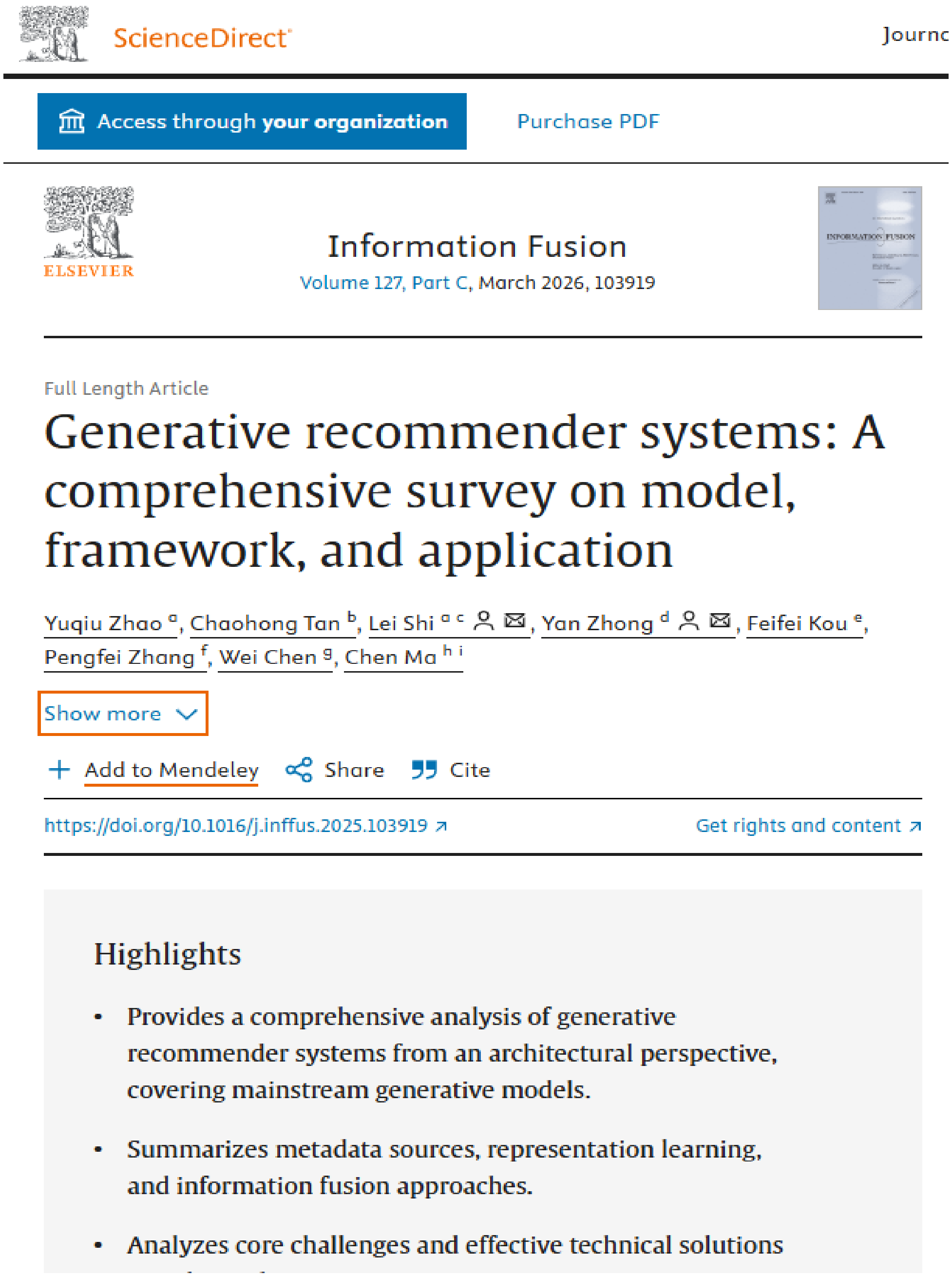
Figure 4: Research article 2

(Pang Li, 2024)

Expanding on the theme of deep learning effects, this 2024 survey by Pang Li et al. provides a technical study of deep neural network architectures to CF, in particular and specifically. It discusses the fact that traditional CF methods tend to be insufficiently scalable and flexible to operate with the complex, non-linear relationships in a modern system and how DNNs can be used to overcome these inefficiencies.

The article gives an in-depth overview of major DNN models implemented to CF systems, such as Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Graph Neural Networks (GNNs). It describes the use of autoencoders in dimensionality reduction and representation learning, and even describes the use of Generative Adversarial Networks (GANs) to generate artificial user-item interaction data to overcome sparsity. The survey wraps up by the need to speak about evaluation protocols and the data available and the need to continue exploring the challenges and future research in this fast-evolving sub-field.

2.1.3 Generative Recommender Systems: A Comprehensive Survey (2025)



Information Fusion

Volume 127, Part C, March 2026, 103919

Full Length Article

Generative recommender systems: A comprehensive survey on model, framework, and application

Yuqiu Zhao [a], Chaohong Tan [b], Lei Shi [a c], Yan Zhong [d], Feifei Kou [e], Pengfei Zhang [f], Wei Chen [g], Chen Ma [h i]

https://doi.org/10.1016/j.inffus.2025.103919 ↗

Highlights

- Provides a comprehensive analysis of generative recommender systems from an architectural perspective, covering mainstream generative models.
- Summarizes metadata sources, representation learning, and information fusion approaches.
- Analyzes core challenges and effective technical solutions

Figure 5: Research article 3

(Yuqiu Zhao, 2026)

Zhao et al. (2025) note a paradigm shift with this particular survey; it is now time to move beyond old-fashioned filtering techniques to Generative Recommender Systems (GRS). It claims that although collaborative and content-based filtering are the pillars, they tend to have problems with the sparsity of data, cold-start issues, and expressing capabilities to provide rich, contextualized recommendations.

The questionnaire examines the integration of Large Language Models (LLMS) and other generative architectures such as Variational Autoencoders (VAEs) and diffusion models into recommendation engines. These models use strong semantic knowledge to learn the query of the user and the description of the item and can directly generate content of the personalized recommendation instead of merely ranking within a fixed set. The paper has given a detailed technical structure of GRS including data representation, model architecture, information fusion strategies and emerging evaluation measures. It is also essential in covering the field engineering issues and implementation plans in the industrial e-commerce environment where business issues and computational efficiency are the key factors.

2.1.4 Hybrid Recommendation System Based on Similar-Price Content (2025)

Applied Sciences

Article 6 October 2025

A Hybrid Recommendation System Based on Similar-Price Content in a Large-Scale E-Commerce Environment

Youngoh Kwon[1], Gwiman Bak[1] and Youngchul Bae[2,*]

[1] Department Electrical and Semiconductor Engineering, Chonnam National University, Yeosu 59626, Republic of Korea
[2] Division Electrical and Computing Engineering, Chonnam National University, Yeosu 59626, Republic of Korea
[*] Author to whom correspondence should be addressed.

Appl. Sci. **2025**, *15*(19), 10758; https://doi.org/10.3390/app151910758
This article belongs to the Special Issue Innovative Data Mining Techniques for Advanced Recommender Systems

Abstract

In large-scale e-commerce, recommendation systems must overcome the shortcomings of conventional models, which often struggle to convert user interest into purchases. This study proposes a revenue-driven recommendation approach that explicitly incorporates user price sensitivity. This study introduces a hybrid recommendation engine that combines collaborative filtering (CF), best match 25 (BM25) for textual relevance, and a price-similarity algorithm. The system is deployed within a scalable three-tier architecture using Elasticsearch and Redis to maintain stability under high-traffic conditions. The system's performance was evaluated through a large-scale A/B test against both a CF-only model and a popular-item baseline. Results showed that while the CF-only model reduced revenue by 5.10%, our hybrid system increased revenue by 5.55% and improved click-through rate (CTR) by 2.55%. These findings demonstrate that integrating price similarity is an effective strategy for developing commercially viable recommendation

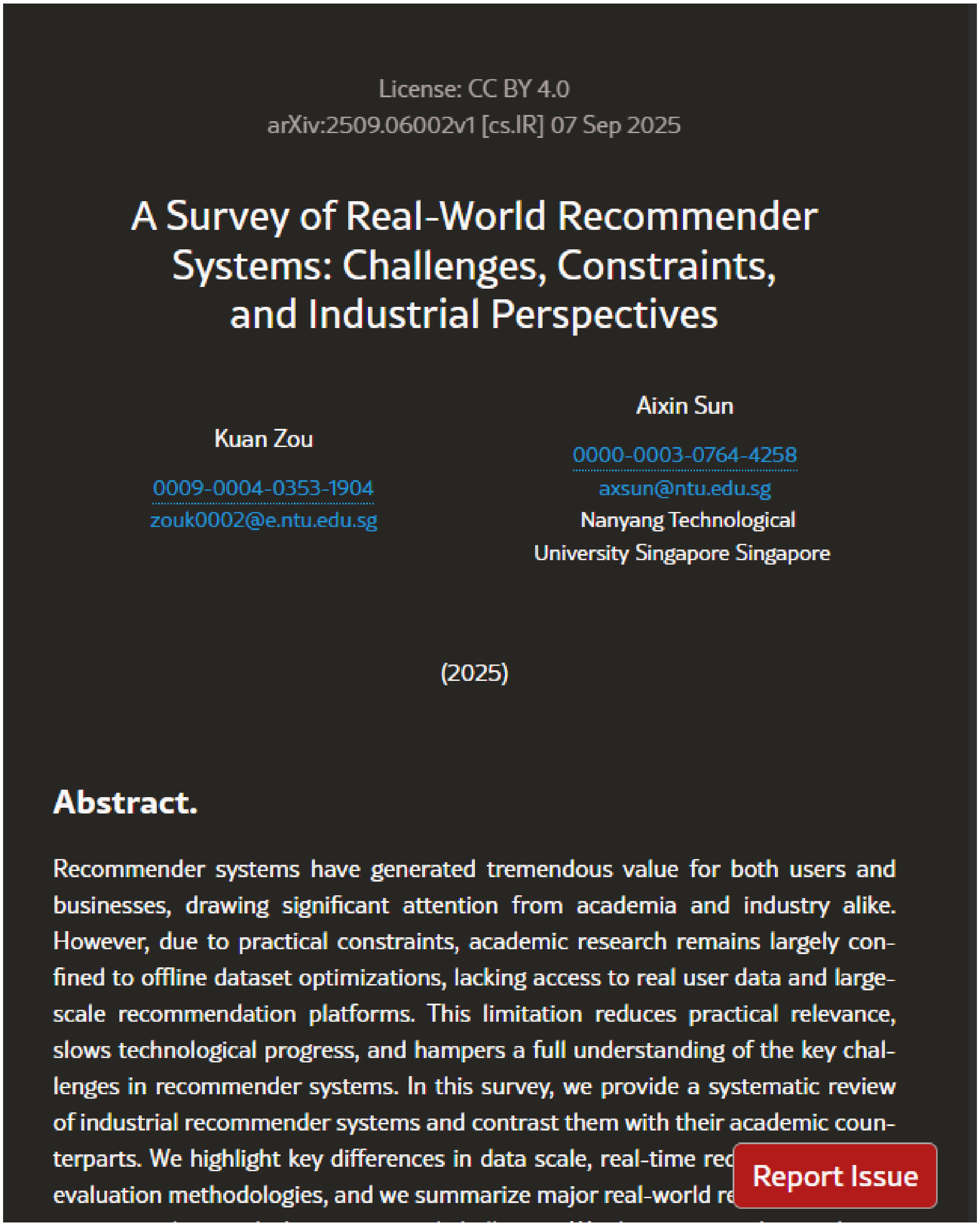
Figure 6: Research article 4

(Youngoh Kwon, 2025)

This 2025 research article is a practical case study on a hybrid system to support the large-scale e-commerce. It shows that the combination of various methods can address certain issues of businesses. The suggested system is a hybrid between collaborative filtering (CF) and content-based filtering (based on the BM25 text-ranking algorithm) and it explicitly includes a price-similarity algorithm.

This investigation also covers one of the main business lessons: despite the correct recommendations, the proposed products cannot work when they do not fit into the apparent affordability of a user. The system is dynamically set to adjust the recommendations to the price sensitivity of a user, which is determined by his purchase history or cart. The model was implemented in a large-scale A/B test on a live platform in a three-tier architecture that is scaled. It was found that it boosted revenue by 5.55% and click-through rate (CTR) by 2.55, and a CF-only model decreased revenue. The work represents the tendency to create interpretable, revenue-driven hybrid models that strike a balance between sophisticated algorithms and business logic to make a difference in the real-world.

2.1.5 Survey of Real-World Recommender Systems



A Survey of Real-World Recommender Systems: Challenges, Constraints, and Industrial Perspectives

Kuan Zou
0009-0004-0353-1904
zouk0002@e.ntu.edu.sg

Aixin Sun
0000-0003-0764-4258
axsun@ntu.edu.sg
Nanyang Technological University Singapore Singapore

(2025)

Abstract.

Recommender systems have generated tremendous value for both users and businesses, drawing significant attention from academia and industry alike. However, due to practical constraints, academic research remains largely confined to offline dataset optimizations, lacking access to real user data and large-scale recommendation platforms. This limitation reduces practical relevance, slows technological progress, and hampers a full understanding of the key challenges in recommender systems. In this survey, we provide a systematic review of industrial recommender systems and contrast them with their academic counterparts. We highlight key differences in data scale, real-time rec
evaluation methodologies, and we summarize major real-world re

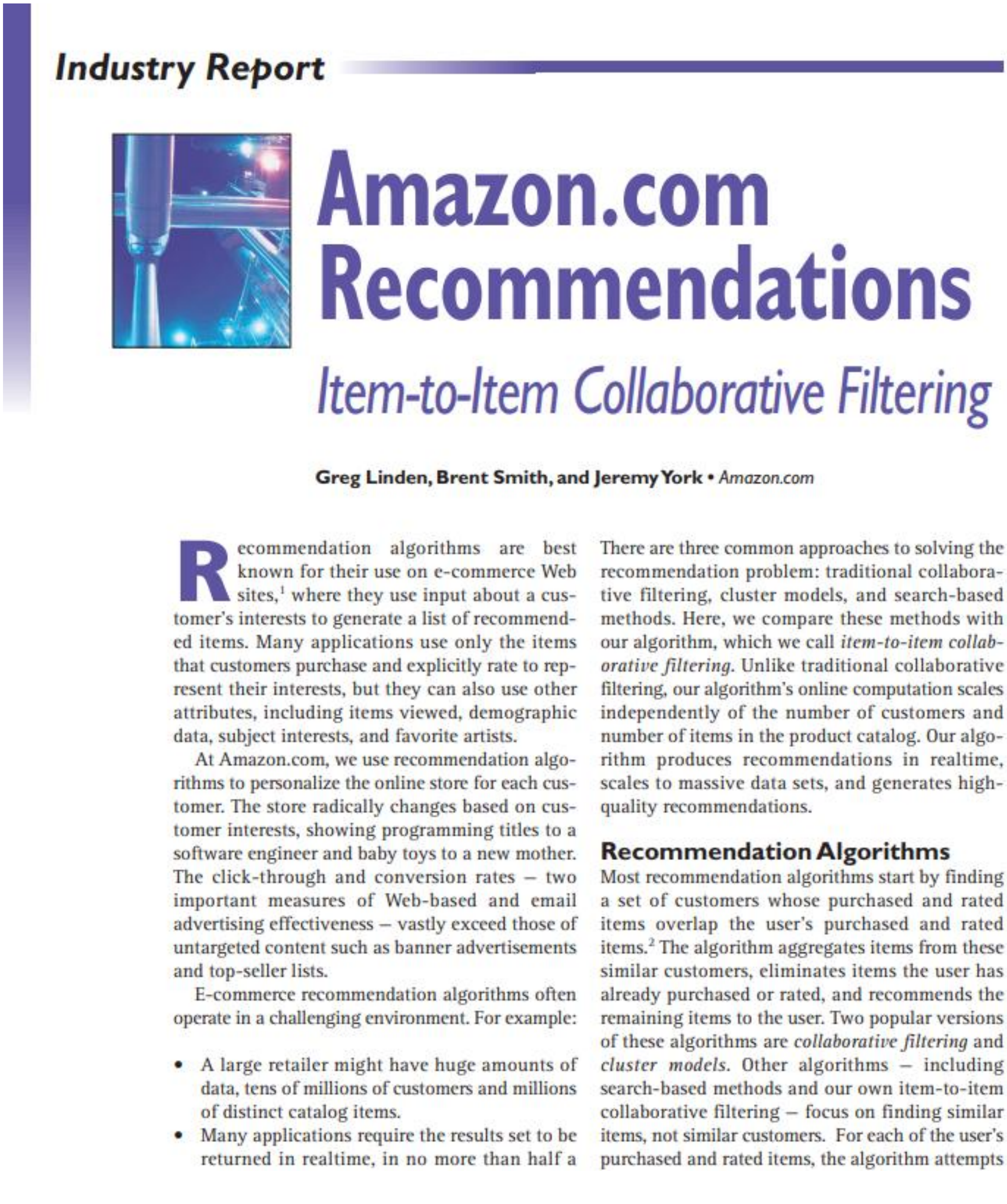
Figure 7: Research article 5

(Kuan Zou, 2025)

This is an academic survey by researchers at Nanyang Technological University (2025) which systematically examines the huge gap between the research and the reality of real-world recommender systems. It states that research has tended to be optimistic to offline datasets and common metrics (such as precision or NDCG), however, does not reflect on the real-world complicated constraints of the business, such as the scale of data, real-time performance needs, intense A/B testing, and business-driven objectives such as revenue and user engagement. To mediate this gap, the authors conduct the analysis of 228 recent industry-validated papers and provide a new framework according to which the systems can be classified as either **Transaction-Oriented (e.g., e-commerce, aimed at generating purchases) or Content-Oriented (e.g., video platforms, aimed at maximizing engagement, such as watch time). The survey identifies the unique issues and remedies in these areas and finishes by describing the future research prospects, such as taking a closer look at how users decide and incorporating the economic theories to enhance the cooperation between the academia and industry.

2.2 Review and analysis of existing projects

2.2.1 Amazon



Industry Report

Amazon.com Recommendations

Item-to-Item Collaborative Filtering

Greg Linden, Brent Smith, and Jeremy York • *Amazon.com*

Recommendation algorithms are best known for their use on e-commerce Web sites,[1] where they use input about a customer's interests to generate a list of recommended items. Many applications use only the items that customers purchase and explicitly rate to represent their interests, but they can also use other attributes, including items viewed, demographic data, subject interests, and favorite artists.

At Amazon.com, we use recommendation algorithms to personalize the online store for each customer. The store radically changes based on customer interests, showing programming titles to a software engineer and baby toys to a new mother. The click-through and conversion rates – two important measures of Web-based and email advertising effectiveness – vastly exceed those of untargeted content such as banner advertisements and top-seller lists.

E-commerce recommendation algorithms often operate in a challenging environment. For example:

- A large retailer might have huge amounts of data, tens of millions of customers and millions of distinct catalog items.
- Many applications require the results set to be returned in realtime, in no more than half a

There are three common approaches to solving the recommendation problem: traditional collaborative filtering, cluster models, and search-based methods. Here, we compare these methods with our algorithm, which we call *item-to-item collaborative filtering*. Unlike traditional collaborative filtering, our algorithm's online computation scales independently of the number of customers and number of items in the product catalog. Our algorithm produces recommendations in realtime, scales to massive data sets, and generates high-quality recommendations.

Recommendation Algorithms

Most recommendation algorithms start by finding a set of customers whose purchased and rated items overlap the user's purchased and rated items.[2] The algorithm aggregates items from these similar customers, eliminates items the user has already purchased or rated, and recommends the remaining items to the user. Two popular versions of these algorithms are *collaborative filtering* and *cluster models*. Other algorithms – including search-based methods and our own item-to-item collaborative filtering – focus on finding similar items, not similar customers. For each of the user's purchased and rated items, the algorithm attempts

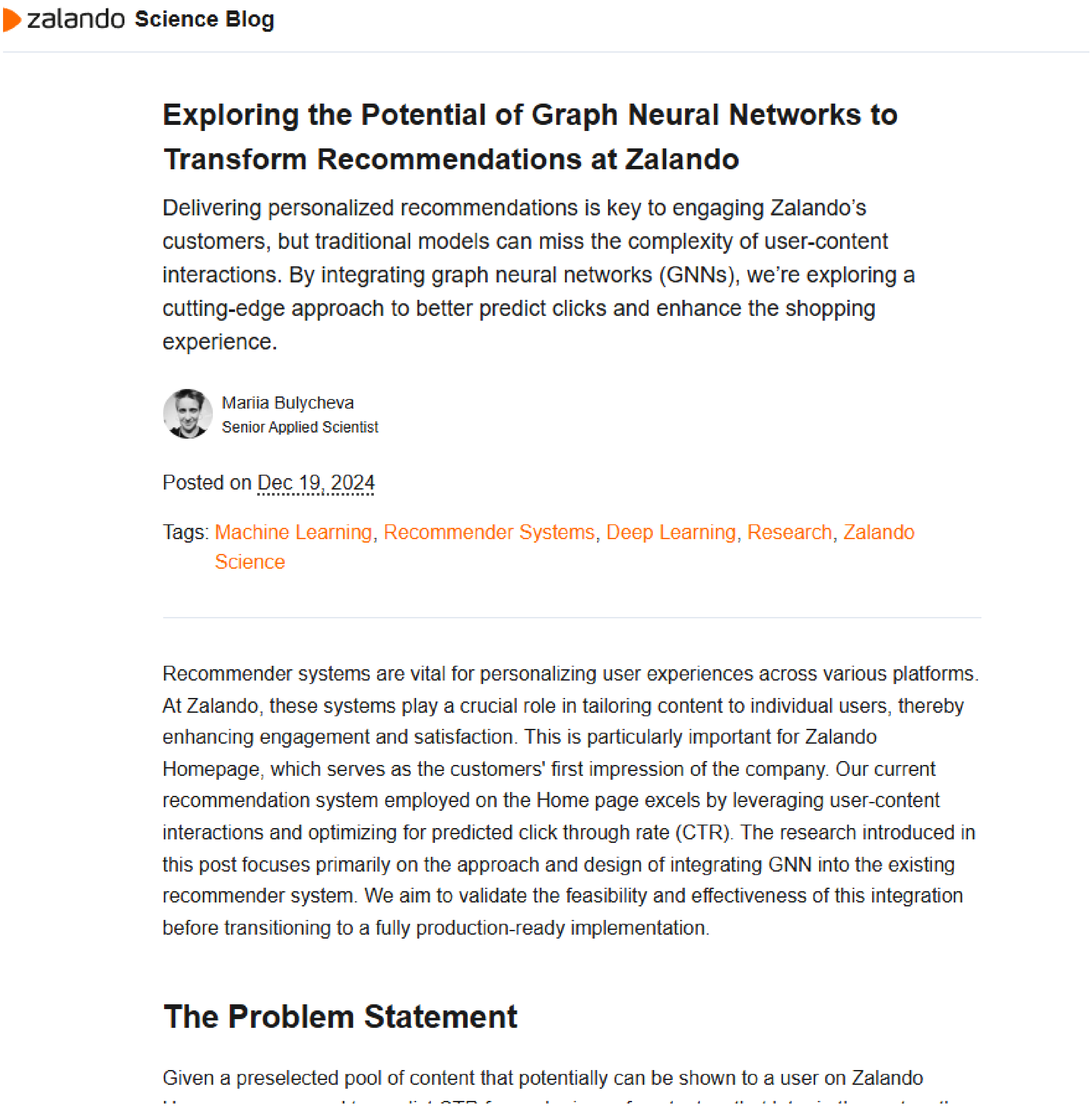
Figure 8: Existing project (Amazon)

(Greg Linden, 2003)

How it's used : Amazon uses a highly engineered item-to-item collaborative filtering approach that looks at which items are bought or viewed together and recommends similar items in real time. This method uses user-item interaction patterns rather than item content, but Amazon also augments recommendations with content or popularity signals in different places.

Problems reported in the article: The Amazon paper explains why classic user-based CF does not scale well and how item-to-item CF solves that; it also notes general limitations of CF such as sparsity and cold-start for new items/users.

2.2.2 Zalando



zalando Science Blog

Exploring the Potential of Graph Neural Networks to Transform Recommendations at Zalando

Delivering personalized recommendations is key to engaging Zalando's customers, but traditional models can miss the complexity of user-content interactions. By integrating graph neural networks (GNNs), we're exploring a cutting-edge approach to better predict clicks and enhance the shopping experience.

Mariia Bulycheva
Senior Applied Scientist

Posted on Dec 19, 2024

Tags: Machine Learning, Recommender Systems, Deep Learning, Research, Zalando Science

Recommender systems are vital for personalizing user experiences across various platforms. At Zalando, these systems play a crucial role in tailoring content to individual users, thereby enhancing engagement and satisfaction. This is particularly important for Zalando Homepage, which serves as the customers' first impression of the company. Our current recommendation system employed on the Home page excels by leveraging user-content interactions and optimizing for predicted click through rate (CTR). The research introduced in this post focuses primarily on the approach and design of integrating GNN into the existing recommender system. We aim to validate the feasibility and effectiveness of this integration before transitioning to a fully production-ready implementation.

The Problem Statement

Given a preselected pool of content that potentially can be shown to a user on Zalando

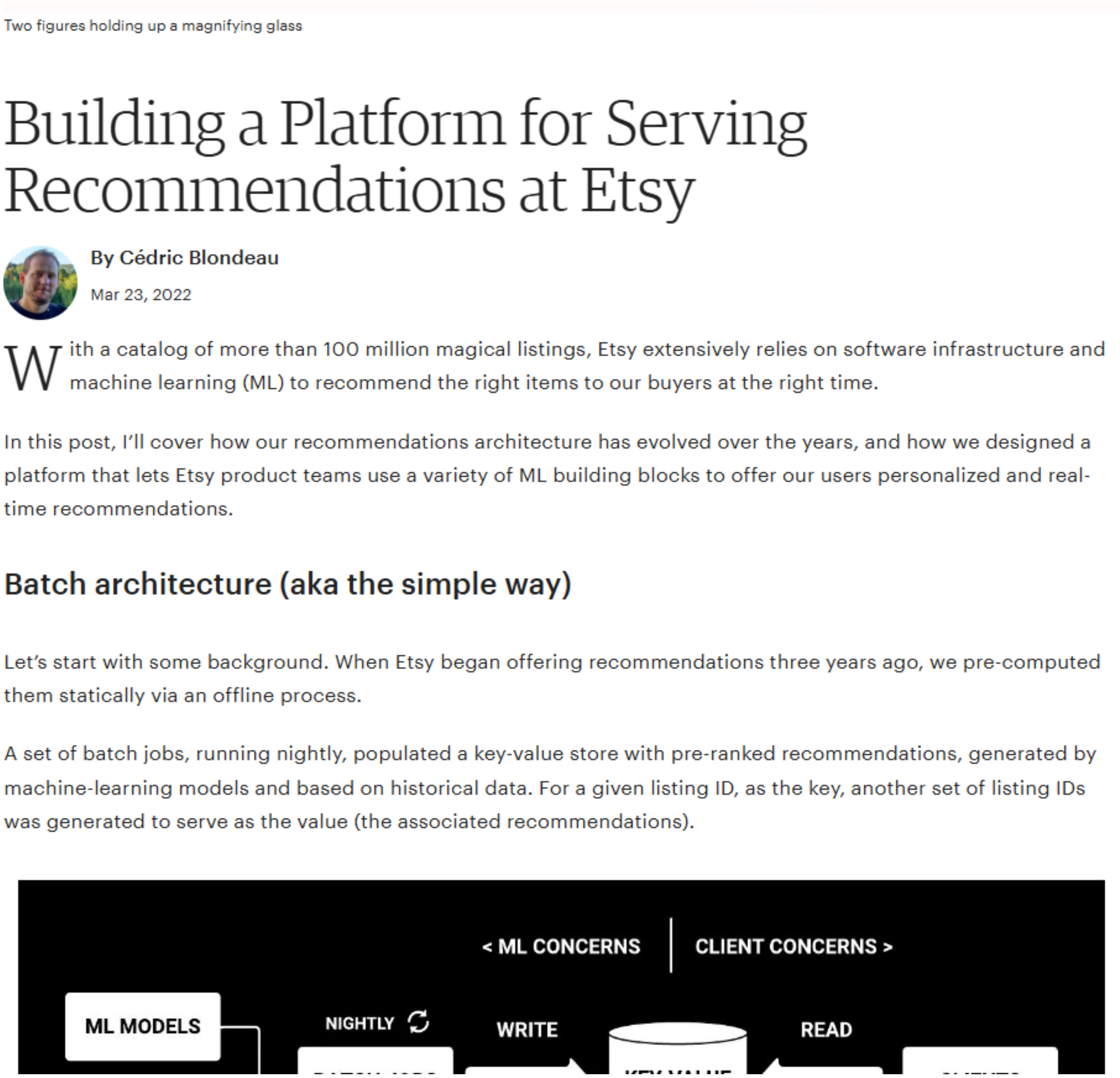
Figure 9: Existing project (Zalando)

(Bulycheva, 2024)

How it’s used (simple): Zalando has published engineering posts on evolving from traditional CF to graph-based and deep models (e.g., GNNs) that combine user–item interaction graphs (collaborative patterns) with item content/attributes to produce embeddings used in ranking. This is effectively a hybrid approach where content and interactions are fused in graph representations.

Problems reported in the article: Zalando’s posts call out challenges such as training and tuning GNNs at scale, limited gains if embeddings aren’t well-tuned, the need for strong negative sampling and evaluation, and remaining issues with hyperparameter tuning and productionization. Cold-start and real-time serving complexity are also discussed.

2.2.3 Etsy



Two figures holding up a magnifying glass

Building a Platform for Serving Recommendations at Etsy

By Cédric Blondeau

Mar 23, 2022

With a catalog of more than 100 million magical listings, Etsy extensively relies on software infrastructure and machine learning (ML) to recommend the right items to our buyers at the right time.

In this post, I'll cover how our recommendations architecture has evolved over the years, and how we designed a platform that lets Etsy product teams use a variety of ML building blocks to offer our users personalized and real-time recommendations.

Batch architecture (aka the simple way)

Let's start with some background. When Etsy began offering recommendations three years ago, we pre-computed them statically via an offline process.

A set of batch jobs, running nightly, populated a key-value store with pre-ranked recommendations, generated by machine-learning models and based on historical data. For a given listing ID, as the key, another set of listing IDs was generated to serve as the value (the associated recommendations).

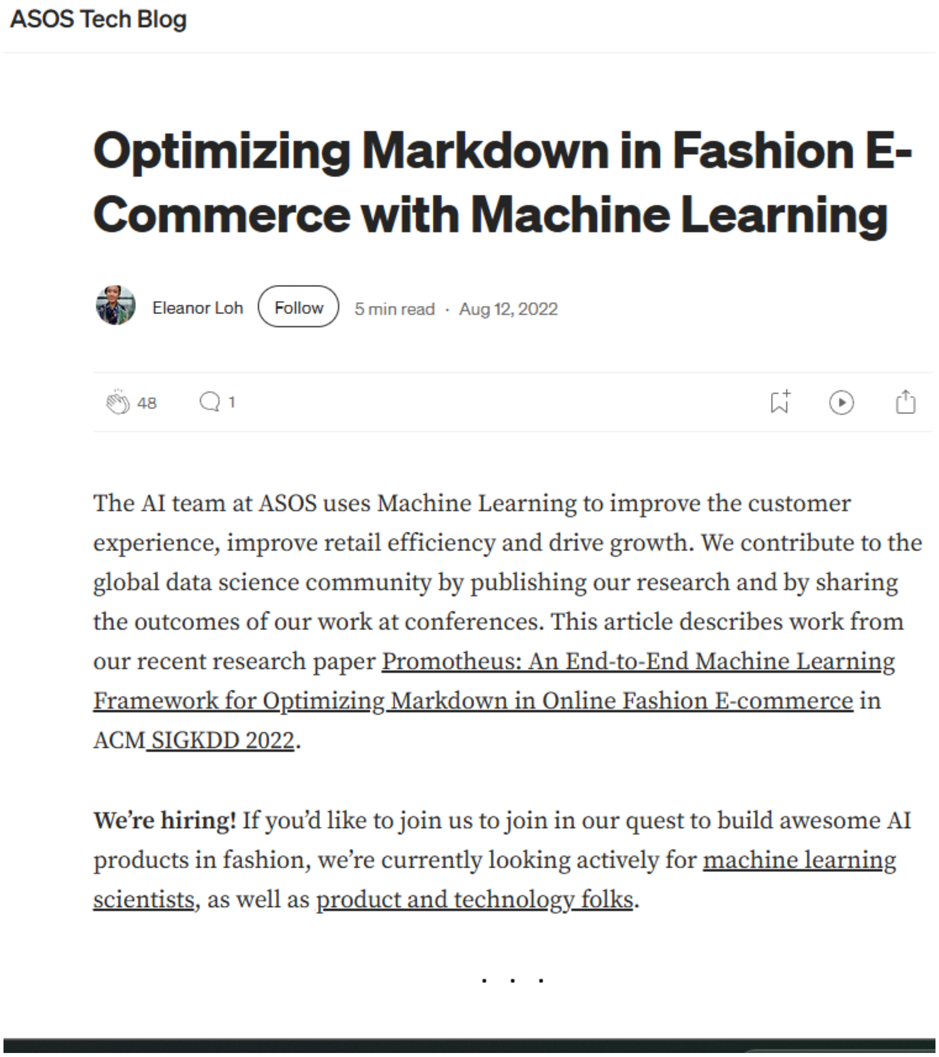
Figure 10: Existing project (Etsy)

(Blondeau, 2022)

How it's used : Etsy runs a recommendation platform that supports multiple approaches collaborative filtering to leverage buyer behaviour and content-based components (item metadata, text features, category tags, and more) to handle new or unique handmade items. Teams combine several recall and ranking blocks so product-level content helps when collaborative signals are weak.

Problems reported in the article: Etsy explains platform problems such as handling sparse interactions for small shops, latency and real-time serving constraints, and the difficulty of making content features work across very diverse, handcrafted product descriptions.

2.2.4 ASOS



ASOS Tech Blog

Optimizing Markdown in Fashion E-Commerce with Machine Learning

Eleanor Loh Follow 5 min read · Aug 12, 2022

The AI team at ASOS uses Machine Learning to improve the customer experience, improve retail efficiency and drive growth. We contribute to the global data science community by publishing our research and by sharing the outcomes of our work at conferences. This article describes work from our recent research paper Promotheus: An End-to-End Machine Learning Framework for Optimizing Markdown in Online Fashion E-commerce in ACM SIGKDD 2022.

We're hiring! If you'd like to join us to join in our quest to build awesome AI products in fashion, we're currently looking actively for machine learning scientists, as well as product and technology folks.

. . .

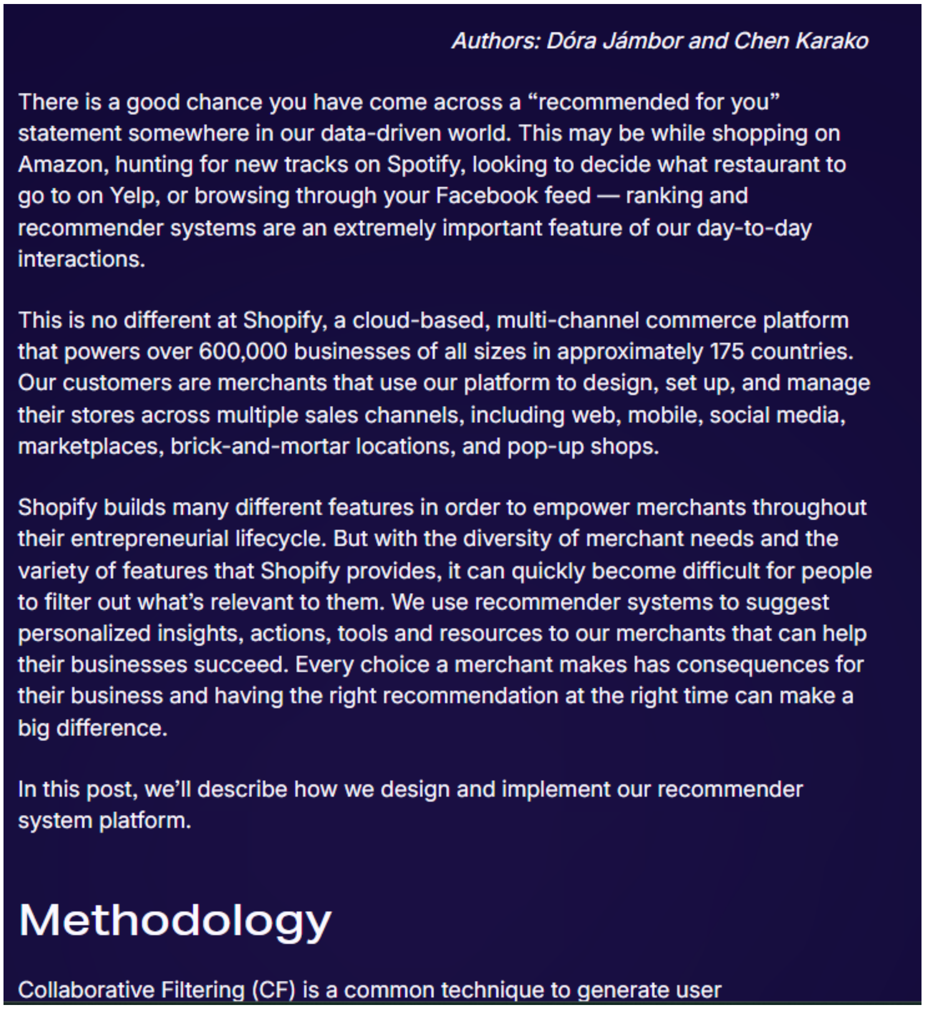
Figure 11: Existing project (ASOS)

(Loh, 2022)

How it's used: ASOS applies hybrid recommenders in fashion: collaborative patterns (what similar customers buy) plus content and visual features (product attributes, category, “complete the look” visual co-occurrence) to suggest outfits and related clothing. Content features are important for fashion (style, fit, color).

Problems reported in the article: Case studies and posts describe challenges like modeling fit and style (subjective attributes), keeping recommendations fresh for seasonal inventory, and avoiding over-personalization that reduces discovery. Cold-start for new fashion items and measuring business impact (beyond clicks) are also discussed.

2.2.5 Shopify



Authors: Dóra Jámbor and Chen Karako

There is a good chance you have come across a “recommended for you” statement somewhere in our data-driven world. This may be while shopping on Amazon, hunting for new tracks on Spotify, looking to decide what restaurant to go to on Yelp, or browsing through your Facebook feed — ranking and recommender systems are an extremely important feature of our day-to-day interactions.

This is no different at Shopify, a cloud-based, multi-channel commerce platform that powers over 600,000 businesses of all sizes in approximately 175 countries. Our customers are merchants that use our platform to design, set up, and manage their stores across multiple sales channels, including web, mobile, social media, marketplaces, brick-and-mortar locations, and pop-up shops.

Shopify builds many different features in order to empower merchants throughout their entrepreneurial lifecycle. But with the diversity of merchant needs and the variety of features that Shopify provides, it can quickly become difficult for people to filter out what’s relevant to them. We use recommender systems to suggest personalized insights, actions, tools and resources to our merchants that can help their businesses succeed. Every choice a merchant makes has consequences for their business and having the right recommendation at the right time can make a big difference.

In this post, we’ll describe how we design and implement our recommender system platform.

Methodology

Collaborative Filtering (CF) is a common technique to generate user

Figure 12: Existing project (Shopify)

(Karako, 2019)

How it's used :

Shopify has product recommendation systems that are primarily supported on the platform level. It offers embedded recommendation facilities (including related products, and frequently bought together) and lets merchants access applications or custom code utilizing collaborative, content-based, and rule-based algorithmic recommendation. The collaborative filtering is applied in those cases when sufficient information about customer contacts is there (purchase or view history within a store), whereas the content-based applies to the product characteristics (category, tags, vendor, price, etc.). Popularity-based and rule-driven recommendations are also applied to Shopify to provide relevant suggestions even with limited information to many small and medium stores.

Problems reported in the articles:

Challenges raised in Shopify engineering blogs and case studies include data sparsity as most merchants do not have sufficiently high traffic and sufficient user interaction data to support effective collaborative filtering. New stores and new products are prone to the cold-start problem thus necessitating content-based and rule-based approaches. Other challenges are to continue with high quality recommendations in very diverse merchant domains, strike a balance between simplicity and customization, and quantify the actual business impact beyond clicks, conversions and revenue lift.

3. Solution

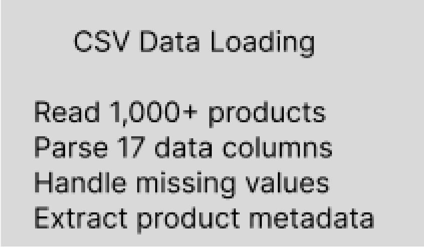
3.1 Explanation of the proposed solution/approach to solving the problem

The main characteristic of the contemporary e-commerce situation is the excessive number of product options people have, which results in a lack of confidence in their decision-making and a lack of customer satisfaction. There are usually many thousands of products that a customer cannot find the relevant ones, and businesses miss the cross-selling opportunities because of the failure of recommendation systems.

The solution/methodology to address this problem is briefly discussed as following:

In the first step, the data is read into the program by the dataset that is provided, i.e. amazon_co-ecommerce_sample.csv.

3.1.1 Data Collection and Loading



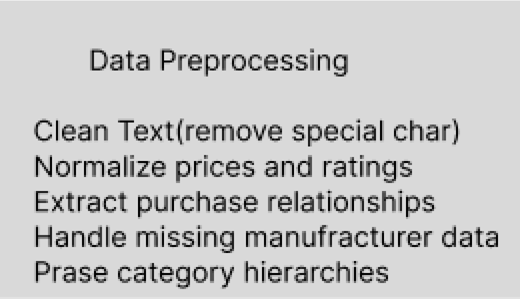
CSV Data Loading

Read 1,000+ products
Parse 17 data columns
Handle missing values
Extract product metadata

Figure 13: Data Collecting and Loading

The system loads the Amazon e-commerce dataset containing comprehensive product information including product names, descriptions, prices, categories, customer reviews, and purchase patterns. Each product entry contains rich metadata that serves as the foundation for both collaborative and content-based filtering approaches.

3.1.2 Data Preprocessing Pipeline



Data Preprocessing

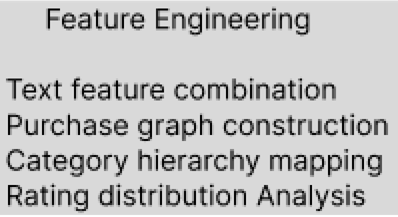
Clean Text(remove special char)
Normalize prices and ratings
Extract purchase relationships
Handle missing manufracturer data
Prase category hierarchies

Figure 14: Data Preprocessing pipeline

The system carries out a few important operations in data preprocessing:

1. Text Cleaning: Deletes HTML tags, special characters and normalizes text on product descriptions and names.
2. Relationship Extraction: Extracts URLs of the type customers who bought this also bought using this to form product relationship graphs.
3. Categorical Encoding: Encodes names of manufacturer and categories as numbers.
4. Normalization of Rating: 4.5 out of 5 stars is changed to numeric values (0-5 scale).
5. Price Standardization: Makes the prices of different products equal so that they can be compared.

3.1.3 Feature Engineering



Feature Engineering

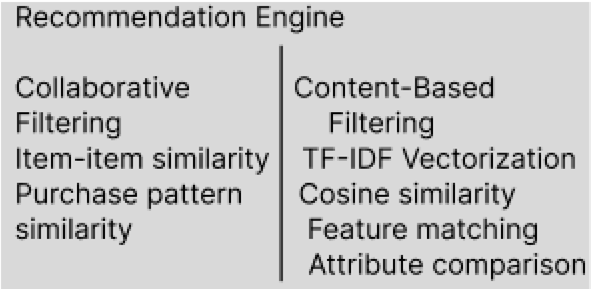
Text feature combination
Purchase graph construction
Category hierarchy mapping
Rating distribution Analysis

Figure 15: Feature Engineering

The system creates two distinct feature sets:

1. Collaborative Features:
   - Extracts co-purchase patterns from product relationship URLs
   - Builds item-item adjacency matrix based on purchase behavior
   - Identifies frequently bought together product pairs
2. Content-Based Features:
   - Combines product name, description, and category into unified text
   - Extracts manufacturer, price range, and review patterns
   - Creates comprehensive product profile vectors

3.1.4 Dual Recommendation Engine Architecture



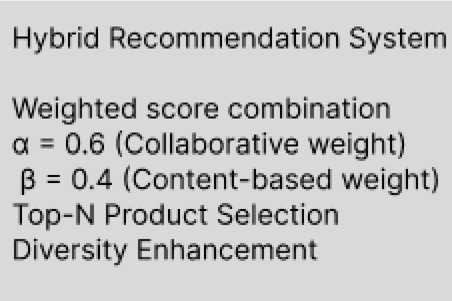
Recommendation Engine

Collaborative Filtering
Item-item similarity
Purchase pattern similarity

Content-Based Filtering
TF-IDF Vectorization
Cosine similarity
Feature matching
Attribute comparison

Figure 16: Dual Recommendation Engine Architecture

3.1.5 Hybrid Recommendation Generation



Hybrid Recommendation System

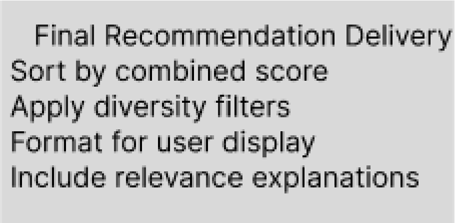
Weighted score combination
α = 0.6 (Collaborative weight)
β = 0.4 (Content-based weight)
Top-N Product Selection
Diversity Enhancement

Figure 17: Hybrid Recommendation Generation

The hybrid approach combines both collaborative and content-based recommendations using a weighted linear combination. This addresses the limitations of each individual approach:

1. Collaborative Filtering leverages actual purchase patterns but suffers from cold-start problems
2. Content-Based Filtering works well for new products but may create recommendation bubbles
3. Hybrid Approach provides balanced recommendations that are both popular and relevant

3.1.6 Recommendation Delivery



Final Recommendation Delivery
Sort by combined score
Apply diversity filters
Format for user display
Include relevance explanations

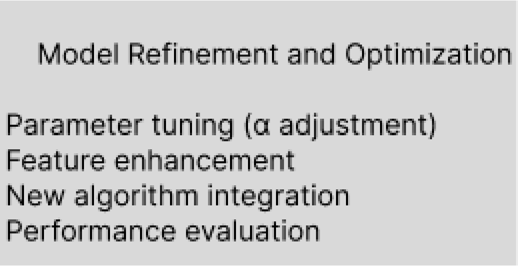
Figure 18: Recommendation Delivery

The system generates final recommendations by:

1. Calculating combined scores for all candidate products

2. Sorting products by descending recommendation score
3. Filtering to ensure category and price diversity
4. Formatting recommendations with clear relevance indicators

3.1.7 Continuous Improvement Loop



Model Refinement and Optimization

Parameter tuning (α adjustment)
Feature enhancement
New algorithm integration
Performance evaluation

Figure 19: Continuous Improvement Loop

Finally, the system includes mechanisms for continuous improvement:

1. Parameter Optimization: Adjusting collaborative vs. content-based weights
2. Feature Refinement: Enhancing text processing and relationship extraction
3. Algorithm Exploration: Testing alternative similarity metrics
4. Performance Monitoring: Tracking recommendation accuracy and relevance

This comprehensive approach ensures that customers receive personalized, relevant, and diverse product recommendations, improving their shopping experience while increasing business conversion rates and customer loyalty.

3.2 Explanation of the AI algorithm/algorithm used

For the E-commerce Product Recommendation System, we use a Hybrid Recommendation Algorithm combining both Item-Based Collaborative Filtering and Content-Based Filtering. The detailed explanation is provided below:

3.2.1 Collaborative Filtering Component: Item-Based Collaborative Filtering

One of the potent personalization technologies powering the adaptive web is collaborative filtering. Collaborative filtering (CF) is the process of filtering or evaluating items through the opinions of other people. CF technology brings together the opinions of large, interconnected communities on the web, supporting filtering of substantial quantities of data. In this chapter we introduce the core concepts of collaborative filtering, its primary uses for users of the adaptive web, the theory and practice of CF algorithms, and design decisions regarding rating systems and acquisition of ratings. We also discuss how to evaluate CF systems, and the evolution of rich interaction interfaces. (Schafer, 2025)

Algorithm Mechanism

Input: Item-User interaction matrix

Output: Item-Item similarity matrix

1. Construct item vectors from user interactions
2. Compute similarity between all item pairs
3. For a given item, find most similar items
4. Recommend items with highest similarity scores

Mathematical Foundation

The similarity between two items i and j is calculated using Cosine Similarity:

$\text{similarity}(i,j) = (\Sigma(u\in U)\ r\_ui \times r\_uj) / (\sqrt{}(\Sigma(u\in U)\ r\_ui^2) \times \sqrt{}(\Sigma(u\in U)\ r\_uj^2))$

Where:

- U = set of all users
- r_ui = interaction of user u with item i
- r_uj = interaction of user u with item j

Implementation in Our System:

In our implementation, we use "customers who bought this item also bought" relationships to create implicit user interactions. Each product pair that appears together in purchase patterns contributes to the similarity score.

3.2.2 Content-Based Filtering Component: TF-IDF Vectorization with Cosine Similarity

Content-Based Filtering is one of the methods used as a Recommendation System. Similarities are calculated over product metadata, and it provides the opportunity to develop recommendations. The products that are most similar to the relevant product are recommended. (ayman, 2022)

TF-IDF Vectorization:

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

text

TF(t,d) = (Number of times term t appears in document d) / (Total number of terms in document d)

IDF(t,D) = log(Total number of documents / Number of documents containing term t)

TF-IDF(t,d,D) = TF(t,d) × IDF(t,D)

Cosine Similarity for Content Matching:

After converting product descriptions to TF-IDF vectors, we compute similarity using:

text

cosine_similarity(A,B) = (A·B) / (||A|| × ||B||)

Where A and B are TF-IDF vectors of two products.

Hybrid Recommendation Algorithm

The hybrid algorithm combines both approaches using a weighted linear combination:

Algorithm Formula:

text

```
final_recommendation_score(i,target) =
  α × collaborative_score(i,target) +
  (1-α) × content_score(i,target)
```

Where:

- α = weight parameter (0 ≤ α ≤ 1)
- collaborative_score = similarity from collaborative filtering
- content_score = similarity from content-based filtering
- i = candidate product
- target = reference product

Reason to Select Hybrid Recommendation Approach:

- **Addresses Cold Start Problem**: Content-based filtering works for new products without purchase history

- **Improves Recommendation Quality**: Combines behavioral patterns with product characteristics
- **Enhances Diversity**: Prevents recommendation bubbles by mixing different recommendation sources
- **Adaptable to Different Scenarios**: Can adjust weights based on available data
- **Scalable Solution**: Handles both sparse and dense interaction data effectively

3.2.3 AI Techniques Used:

a. Cosine Similarity Calculation:

Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space. It's particularly effective for text similarity as it considers both direction and magnitude.

text

Implementation:

```
    def cosine_similarity(vec1, vec2):

        dot_product = np.dot(vec1, vec2)

        norm1 = np.linalg.norm(vec1)

        norm2 = np.linalg.norm(vec2)

        return dot_product / (norm1 * norm2) if norm1 * norm2 != 0 else 0
```

b. TF-IDF Vectorization:

TF-IDF converts text documents into numerical vectors where each dimension represents a term's importance in the document relative to the entire corpus.

text

Implementation Steps:

1. Tokenize product descriptions

2. Compute term frequencies
3. Compute inverse document frequencies
4. Create TF-IDF weighted vectors
5. Normalize vectors for similarity computation

c. Graph-Based Relationship Extraction:

Extracts purchase patterns from product relationship URLs to build implicit user-item interaction graphs.

text

Process:

1. Parse "also_bought" URLs
2. Extract product IDs
3. Build adjacency matrix
4. Compute transitive relationships

d. Weighted Score Combination:

Intelligently combines multiple recommendation scores using tunable weights to balance different recommendation strategies.

text

Advantages:

- Can be adjusted based on data availability
- Allows A/B testing for optimal performance
- Provides explainable recommendations

3.3 Pseudocode of the solution

```
IMPORT pandas, numpy, sklearn, re, math

READ 'amazon_co-ecommerce_sample.csv' INTO data_frame


DROP unnecessary columns: Keep only ['uniq_id', 'product_name', 'manufacturer','price',
'amazon_category_and_sub_category','description','customers_who_bought_this_item_
also_bought']


DISPLAY the cleaned data with shape and first 5 rows


# Data Preprocessing

CREATE function clean_text(text):

   REMOVE special characters and HTML tags

   CONVERT to lowercase

   RETURN cleaned_text


APPLY clean text to 'product_name', 'description', 'amazon_category_and_sub_category'

EXTRACT rating values from 'average_review_rating' string format

NORMALIZE prices to numerical format


# Extract Collaborative Features

CREATE function extract_relationships(data):
    CREATE empty co-purchase matrix
    FOR EACH product in data:
        EXTRACT product IDs from 'also_bought' URLs
        UPDATE co-purchase matrix with relationships
    RETURN co-purchase matrix

co_purchase_matrix = extract_relationships(data)

# Build Collaborative Similarity
CREATE function compute_collaborative_similarity(matrix):
    FOR EACH product_pair in matrix:
        CALCULATE cosine similarity based on co-purchase patterns
    RETURN similarity_matrix

collab_similarity = compute_collaborative_similarity(co_purchase_matrix)

# Prepare Content Features
COMBINE 'product_name', 'description', 'category' into single text feature
NORMALIZE text data using Lemmatization and removing stop words
TRANSFORM text data into numerical features using TF-IDF vectorization
ENCODE categorical features (manufacturer, price_range, rating_bin)

# Build Content Similarity
```

```
CREATE function compute_content_similarity(feature_vectors):
    FOR EACH product_pair:
        CALCULATE cosine similarity between feature vectors
    RETURN content_similarity_matrix


content_similarity = compute_content_similarity(content_vectors)


# Hybrid Recommendation Model
SPLIT product data into training and evaluation sets


CREATE hybrid similarity = alpha * collab_similarity + (1-alpha) * content_similarity


TRAIN recommendation model using hybrid similarity matrix
SAVE the trained model as 'hybrid_recommender.pkl'


# Prediction Function
DEFINE function get_recommendations(product_name, top_n=5):
    FIND product in dataset
    GET product_index


    EXTRACT similarity scores for product_index from hybrid matrix
    SORT products by similarity score (descending)
    REMOVE the query product from results
    SELECT top_n products


    RETURN recommended_products


# Main Recommendation Process
GET user input for product name
CALL get_recommendations with user input


IF recommendations found:
    FOR EACH recommendation in results:
        DISPLAY product_name, manufacturer, price, similarity_score
ELSE:
    DISPLAY "No recommendations found for this product"
END IF


GET user feedback on recommendations
UPDATE model based on user feedback
END
```

3.4 Diagrammatical representation of Flowchart

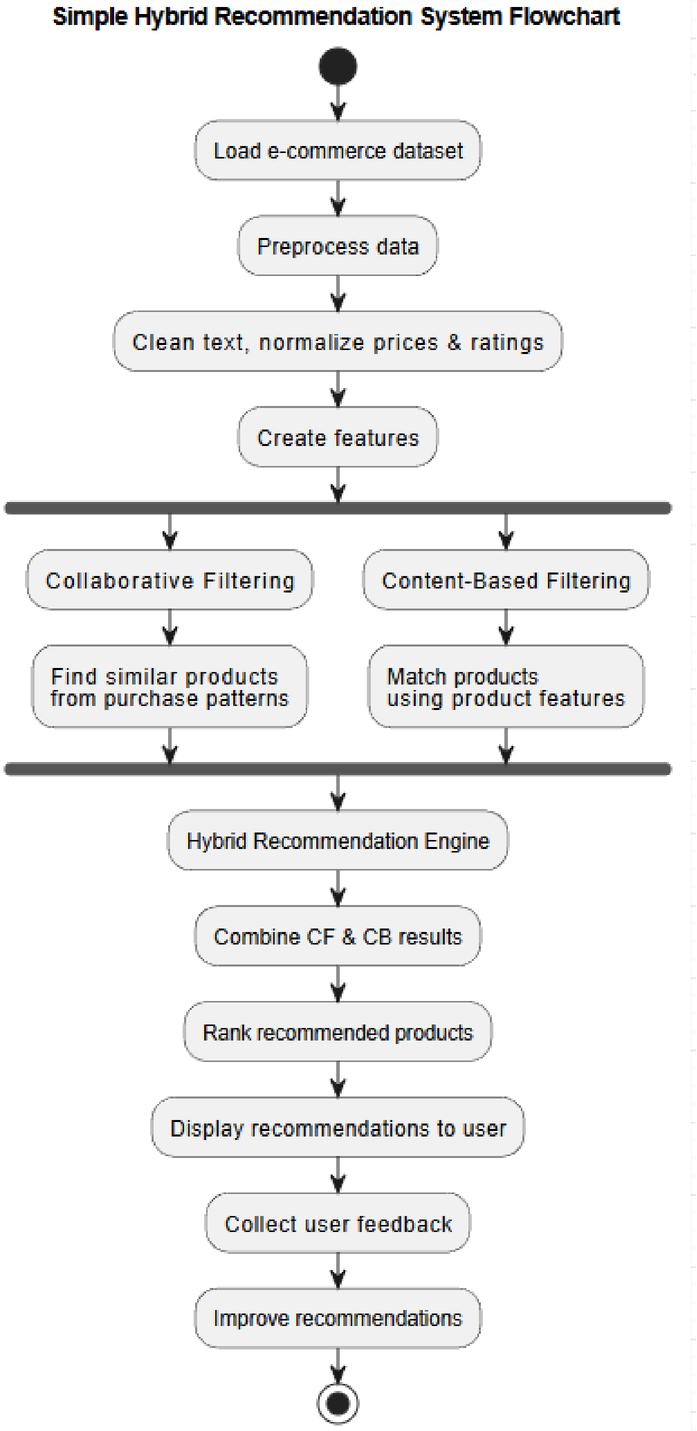

Simple Hybrid Recommendation System Flowchart
Load e-commerce dataset
Preprocess data
Clean text, normalize prices & ratings
Create features
Collaborative Filtering
Content-Based Filtering
Find similar products from purchase patterns
Match products using product features
Hybrid Recommendation Engine
Combine CF & CB results
Rank recommended products
Display recommendations to user
Collect user feedback
Improve recommendations

Figure 20:Flowchart Diagram

3.5 Diagrammatical representation of State Transition Diagram

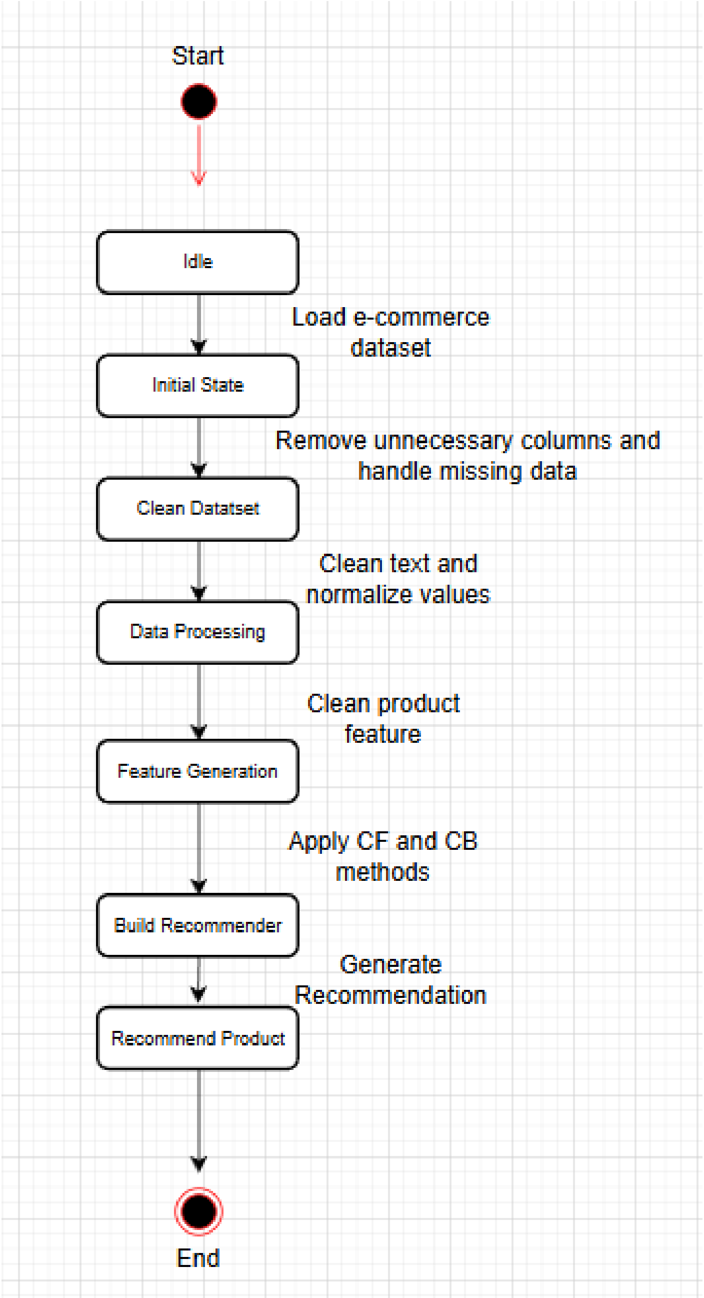
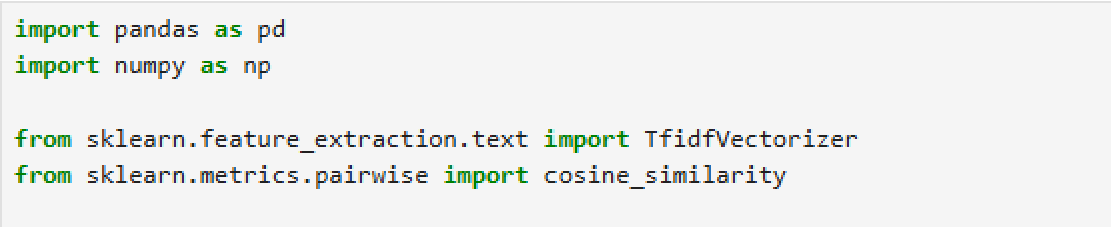


Figure 21: State Transition Diagram

3.6 Explanation of the development process

The steps below are the development phase of the system:

3.6.1 Importing Libraries

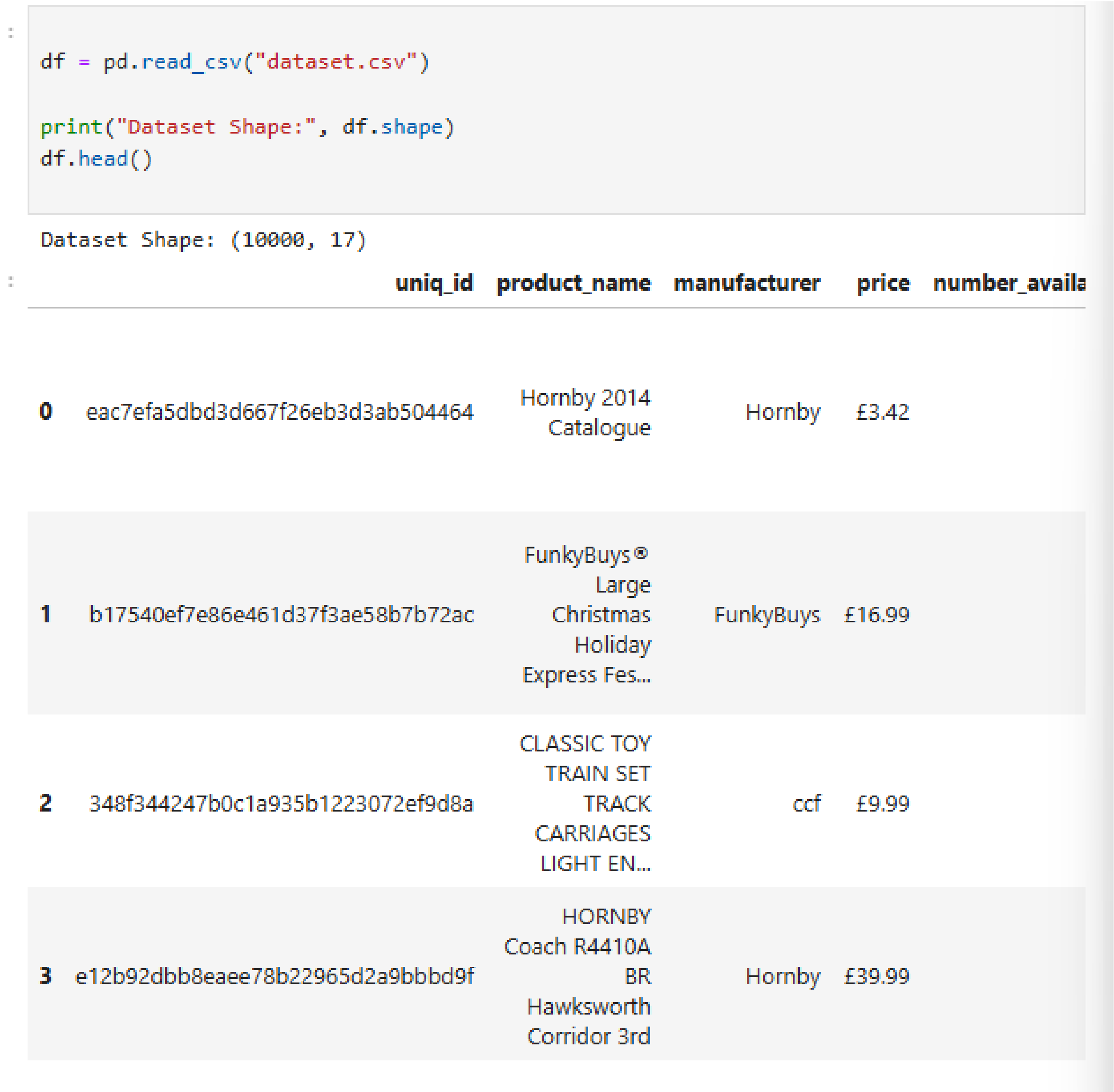


```
import pandas as pd
import numpy as np

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Figure 22: Importing libraries

This image shows the initial Python code setup for a content-based recommendation system. It includes essential imports: pandas for data handling, numpy for numerical operations, and key scikit-learn modules for text processing and similarity measurement. Specifically, TfidfVectorizer is used to convert text into TF-IDF features, while cosine_similarity helps compute how similar different products are based on those features. This setup is typical for building a recommendation engine that suggests items by comparing textual descriptions.

3.6.2 Loading the dataset



```
df = pd.read_csv("dataset.csv")

print("Dataset Shape:", df.shape)
df.head()
```

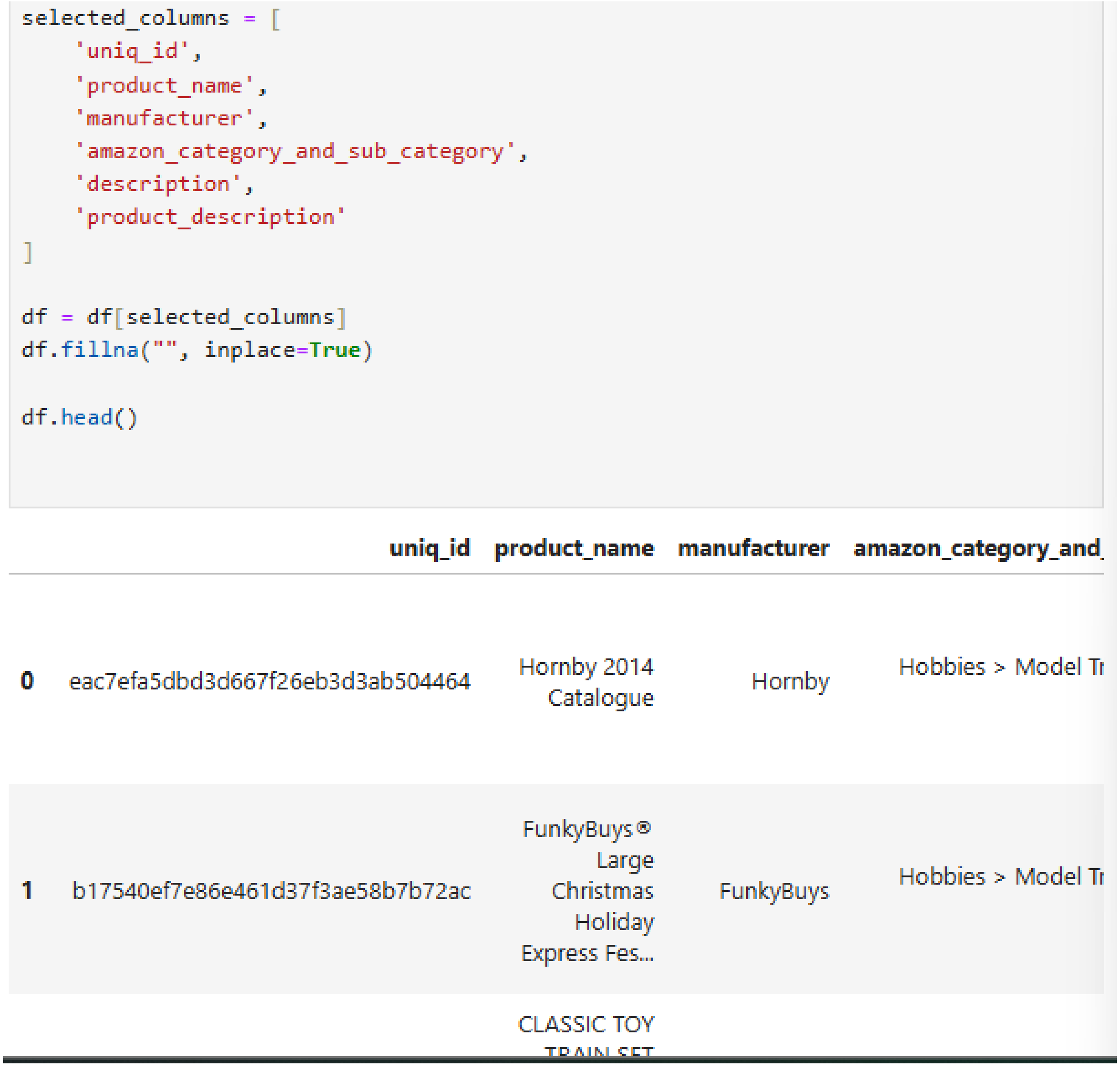
```
Dataset Shape: (10000, 17)
```

	uniq_id	product_name	manufacturer	price	number_availa
0	eac7efa5dbd3d667f26eb3d3ab504464	Hornby 2014 Catalogue	Hornby	£3.42	
1	b17540ef7e86e461d37f3ae58b7b72ac	FunkyBuys® Large Christmas Holiday Express Fes...	FunkyBuys	£16.99	
2	348f344247b0c1a935b1223072ef9d8a	CLASSIC TOY TRAIN SET TRACK CARRIAGES LIGHT EN...	ccf	£9.99	
3	e12b92dbb8eaee78b22965d2a9bbbd9f	HORNBY Coach R4410A BR Hawksworth Corridor 3rd	Hornby	£39.99	

Figure 23: Loading Dataset

The dataset consists of 10,000 rows and 17 columns, though only the first few are shown. Key columns visible include a unique ID, product name, manufacturer, price, and an incomplete column for ratings. The sample entries highlight model train products, like a Hornby catalog and a Christmas holiday train set, indicating this is likely an e-commerce dataset focused on hobby or toy items.

3.6.3 Selecting useful columns



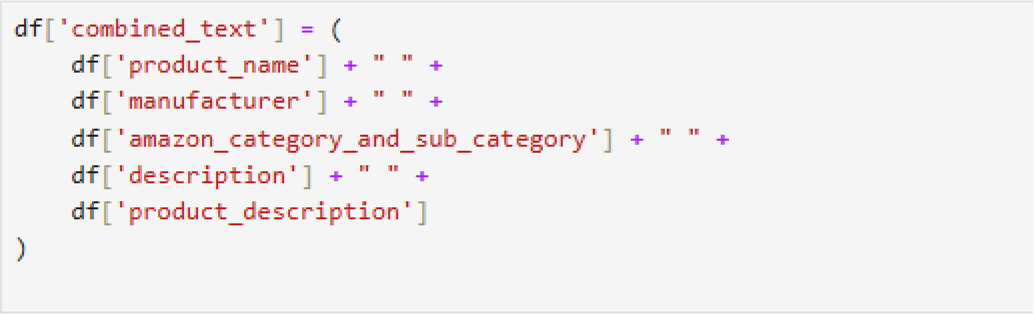
```
selected_columns = [
    'uniq_id',
    'product_name',
    'manufacturer',
    'amazon_category_and_sub_category',
    'description',
    'product_description'
]

df = df[selected_columns]
df.fillna("", inplace=True)

df.head()
```

	uniq_id	product_name	manufacturer	amazon_category_and_
0	eac7efa5dbd3d667f26eb3d3ab504464	Hornby 2014 Catalogue	Hornby	Hobbies > Model Tr
1	b17540ef7e86e461d37f3ae58b7b72ac	FunkyBuys® Large Christmas Holiday Express Fes...	FunkyBuys	Hobbies > Model Tr
		CLASSIC TOY TRAIN SET		

Figure 24: Selecting Useful Columns

Here, the dataset is refined by selecting only relevant columns such as product name, manufacturer, category, description, and product description. Missing values are filled with empty strings to avoid errors during text processing. The preview shows combined text from categories like “Hobbies > Model Tr…” and product names, which will later be used to generate meaningful recommendations.

3.6.4 Create combined text feature

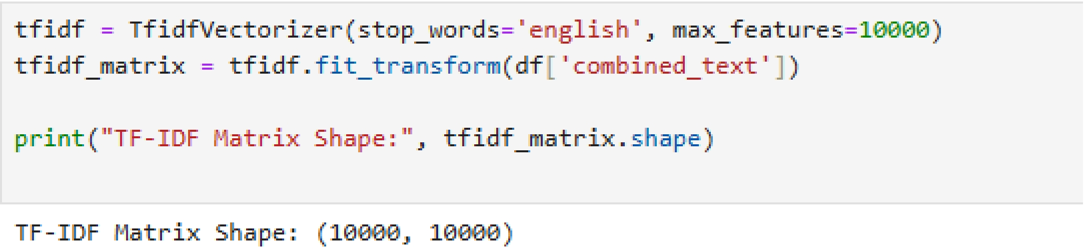


```
df['combined_text'] = (
    df['product_name'] + " " +
    df['manufacturer'] + " " +
    df['amazon_category_and_sub_category'] + " " +
    df['description'] + " " +
    df['product_description']
)
```

Figure 25: Create Combined Text Feature

To improve recommendation quality, multiple text columns are merged into a single field called ‘combined_text’. This includes the product name, manufacturer, category, and both description fields, separated by spaces. Combining these provides a richer textual profile for each product, allowing the system to detect similarities based on various attributes like brand, category, and descriptive details.

3.6.5 Convert text to TF-IDF vectors



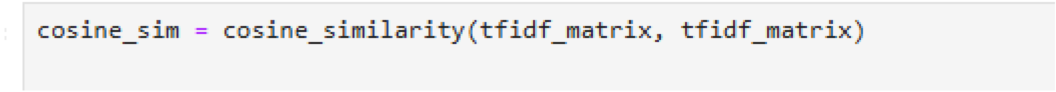
```
tfidf = TfidfVectorizer(stop_words='english', max_features=10000)
tfidf_matrix = tfidf.fit_transform(df['combined_text'])

print("TF-IDF Matrix Shape:", tfidf_matrix.shape)
```

```
TF-IDF Matrix Shape: (10000, 10000)
```

Figure 26: Convert Text to TF-IDF Vectors

The combined text is transformed into numerical form using TF-IDF vectorization, which ignores common English stop words and limits the vocabulary to 10,000 terms. The resulting TF-IDF matrix has a shape of (10,000, 10,000), meaning there are 10,000 products and up to 10,000 distinct word features a large but manageable representation for similarity calculations.

3.6.6 Compute Cosine Similarity Matrix

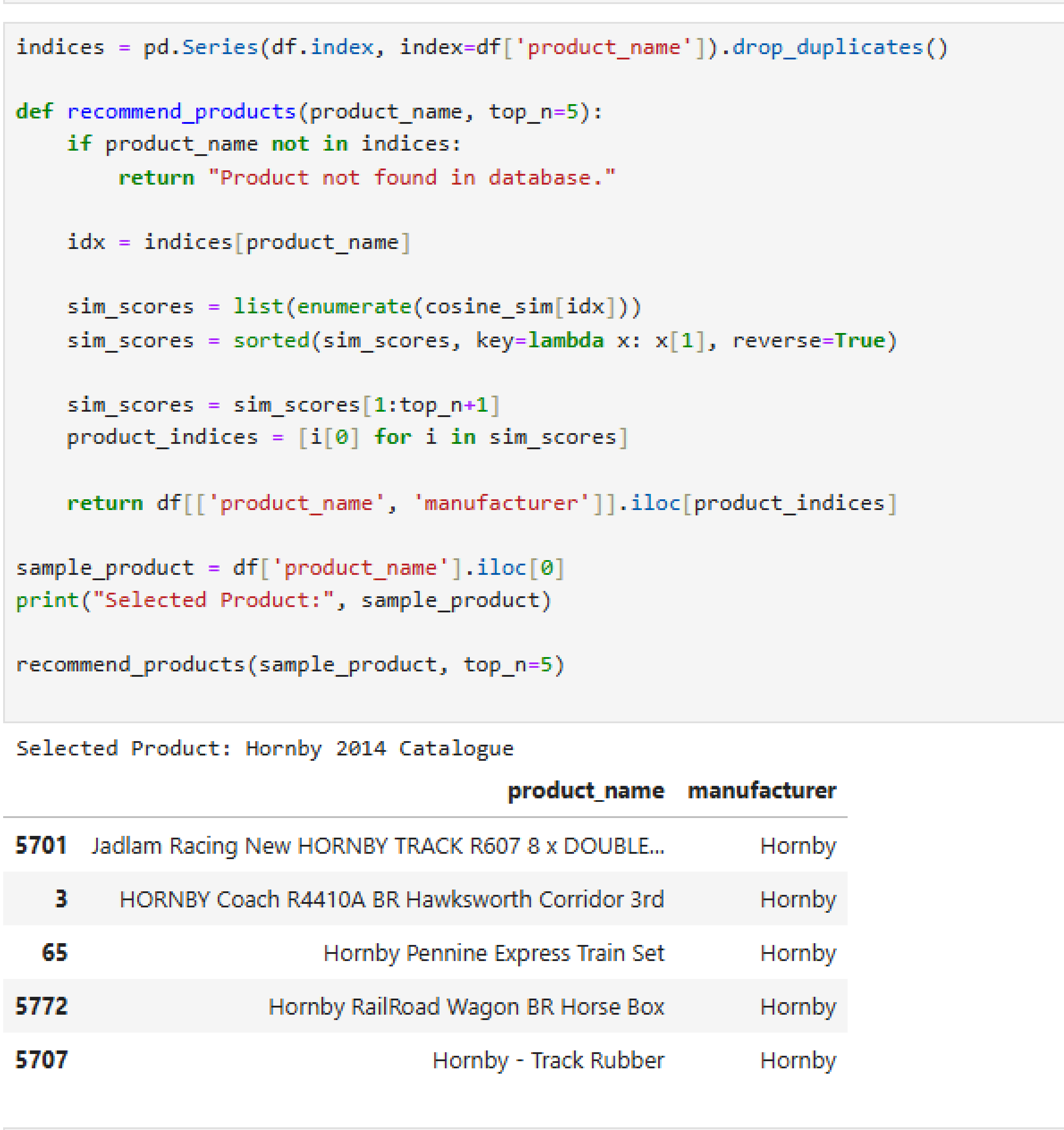


```
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

Figure 27: Computing Cosine Similarity Matrix

Using the TF-IDF matrix, a cosine similarity matrix is computed to measure how similar each product is to every other product. Each cell in this matrix represents the similarity score between two items, ranging from 0 (no similarity) to 1 (identical). This matrix forms the backbone of the recommendation engine, enabling quick lookups of the most similar products.

3.6.7 Create Recommendation Function



```
indices = pd.Series(df.index, index=df['product_name']).drop_duplicates()

def recommend_products(product_name, top_n=5):
    if product_name not in indices:
        return "Product not found in database."

    idx = indices[product_name]

    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    sim_scores = sim_scores[1:top_n+1]
    product_indices = [i[0] for i in sim_scores]

    return df[['product_name', 'manufacturer']].iloc[product_indices]

sample_product = df['product_name'].iloc[0]
print("Selected Product:", sample_product)

recommend_products(sample_product, top_n=5)
```

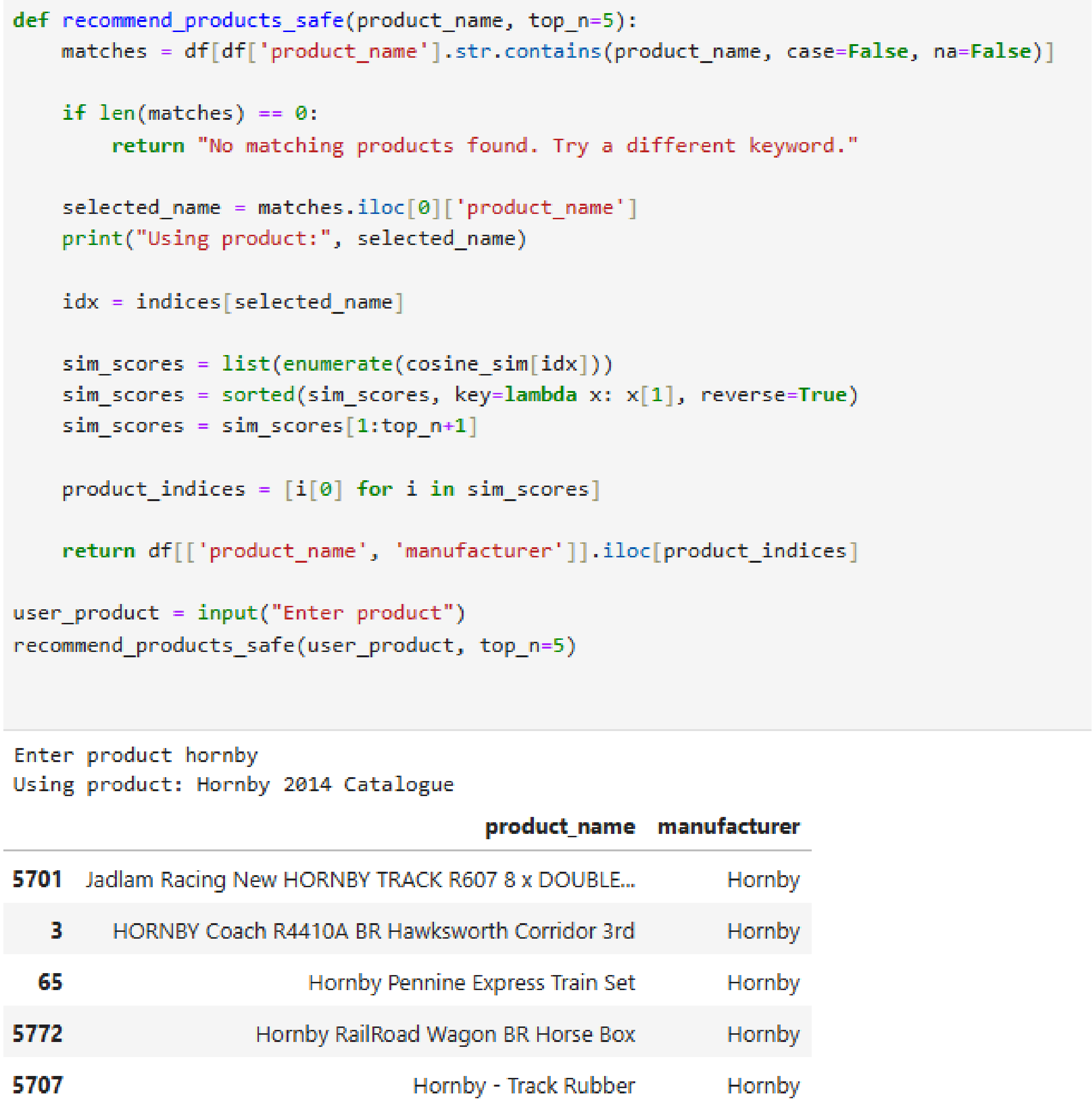
```
Selected Product: Hornby 2014 Catalogue
```

	product_name	manufacturer
5701	Jadlam Racing New HORNBY TRACK R607 8 x DOUBLE...	Hornby
3	HORNBY Coach R4410A BR Hawksworth Corridor 3rd	Hornby
65	Hornby Pennine Express Train Set	Hornby
5772	Hornby RailRoad Wagon BR Horse Box	Hornby
5707	Hornby - Track Rubber	Hornby

Figure 28: Create Recommendation Function

A simple recommendation function is defined: it takes a product name, finds its index, retrieves similarity scores from the cosine matrix, sorts them, and returns the top N similar products excluding the item itself. When tested with “Hornby 2014 Catalogue,” it suggests five other Hornby train-related items, showing the system works well within a known brand.

3.6.8 Requesting User Input for Recommendation



```python
def recommend_products_safe(product_name, top_n=5):
    matches = df[df['product_name'].str.contains(product_name, case=False, na=False)]

    if len(matches) == 0:
        return "No matching products found. Try a different keyword."

    selected_name = matches.iloc[0]['product_name']
    print("Using product:", selected_name)

    idx = indices[selected_name]

    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:top_n+1]

    product_indices = [i[0] for i in sim_scores]

    return df[['product_name', 'manufacturer']].iloc[product_indices]

user_product = input("Enter product")
recommend_products_safe(user_product, top_n=5)
```

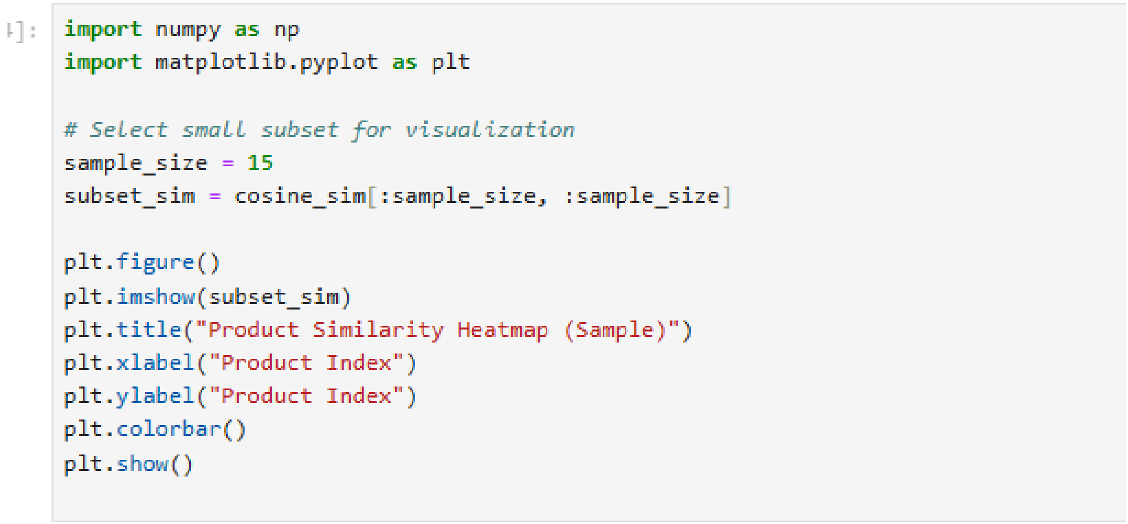
```
Enter product hornby
Using product: Hornby 2014 Catalogue
```

	product_name	manufacturer
5701	Jadlam Racing New HORNBY TRACK R607 8 x DOUBLE...	Hornby
3	HORNBY Coach R4410A BR Hawksworth Corridor 3rd	Hornby
65	Hornby Pennine Express Train Set	Hornby
5772	Hornby RailRoad Wagon BR Horse Box	Hornby
5707	Hornby - Track Rubber	Hornby

Figure 29: Requesting User Input for Recommendation

To handle user input more gracefully, a safer function is introduced. It searches for products containing the user's keyword (case-insensitive) and uses the first match for recommendations. For example, entering "homby" (a possible typo for "Hornby") still returns relevant Hornby products, demonstrating the system's robustness to minor input errors.

3.6.9 Similarity Heatmap



```
import numpy as np
import matplotlib.pyplot as plt

# Select small subset for visualization
sample_size = 15
subset_sim = cosine_sim[:sample_size, :sample_size]

plt.figure()
plt.imshow(subset_sim)
plt.title("Product Similarity Heatmap (Sample)")
plt.xlabel("Product Index")
plt.ylabel("Product Index")
plt.colorbar()
plt.show()
```

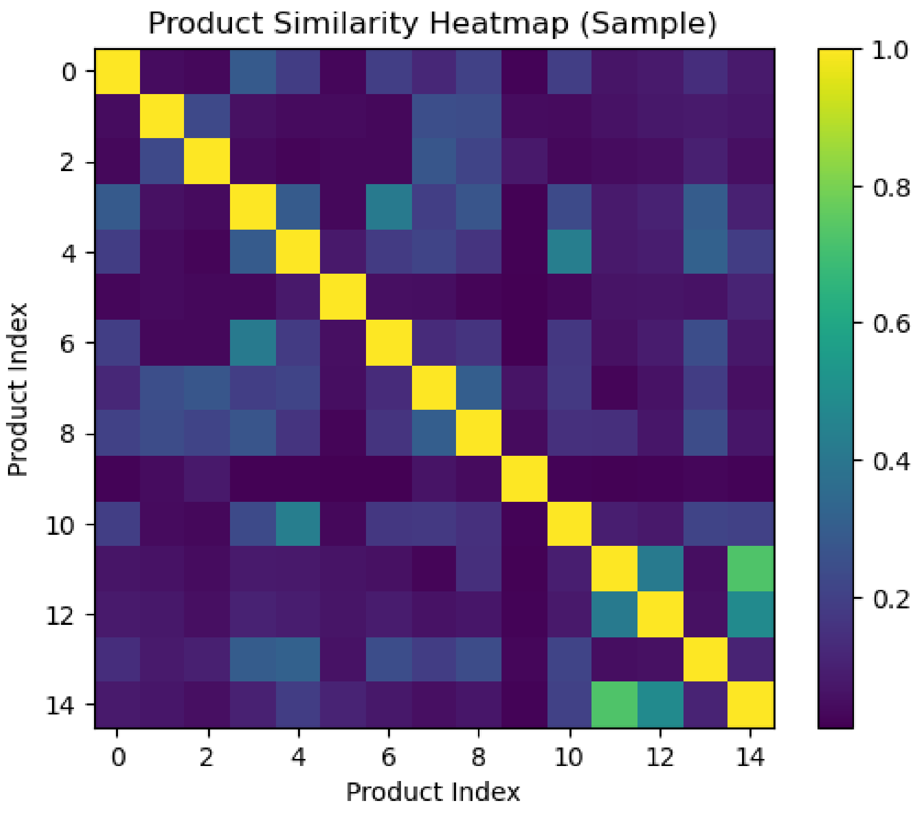
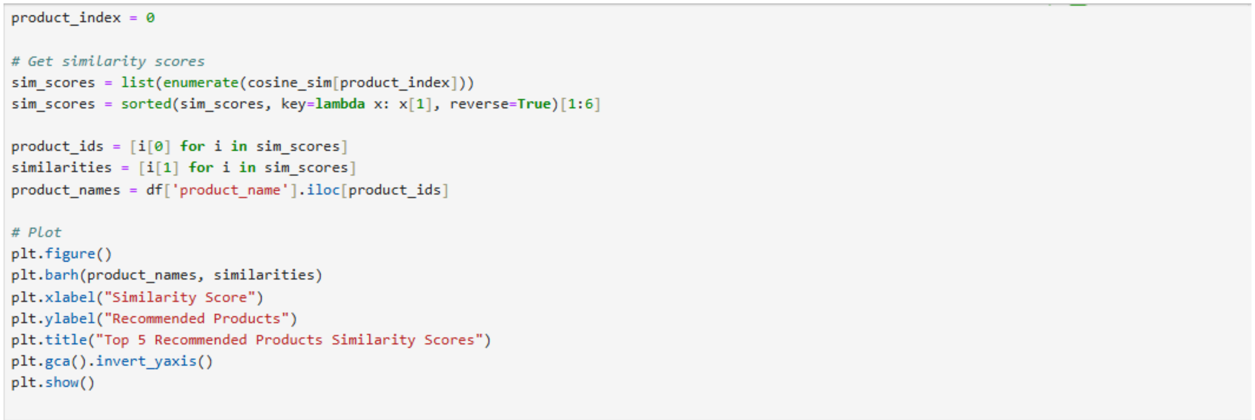


Figure 30: Similarity Heatmap

A heatmap visualizes the cosine similarity matrix for a small sample of 15 products. Lighter colors indicate higher similarity. The diagonal is bright because each product is identical to itself, while off-diagonal patterns reveal clusters of similar items. This helps in qualitatively assessing how well the model groups related products.

3.6.10 Bar Chart - Similarity Scores for Top Recommendations



```
product_index = 0

# Get similarity scores
sim_scores = list(enumerate(cosine_sim[product_index]))
sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)[1:6]

product_ids = [i[0] for i in sim_scores]
similarities = [i[1] for i in sim_scores]
product_names = df['product_name'].iloc[product_ids]

# Plot
plt.figure()
plt.barh(product_names, similarities)
plt.xlabel("Similarity Score")
plt.ylabel("Recommended Products")
plt.title("Top 5 Recommended Products Similarity Scores")
plt.gca().invert_yaxis()
plt.show()
```

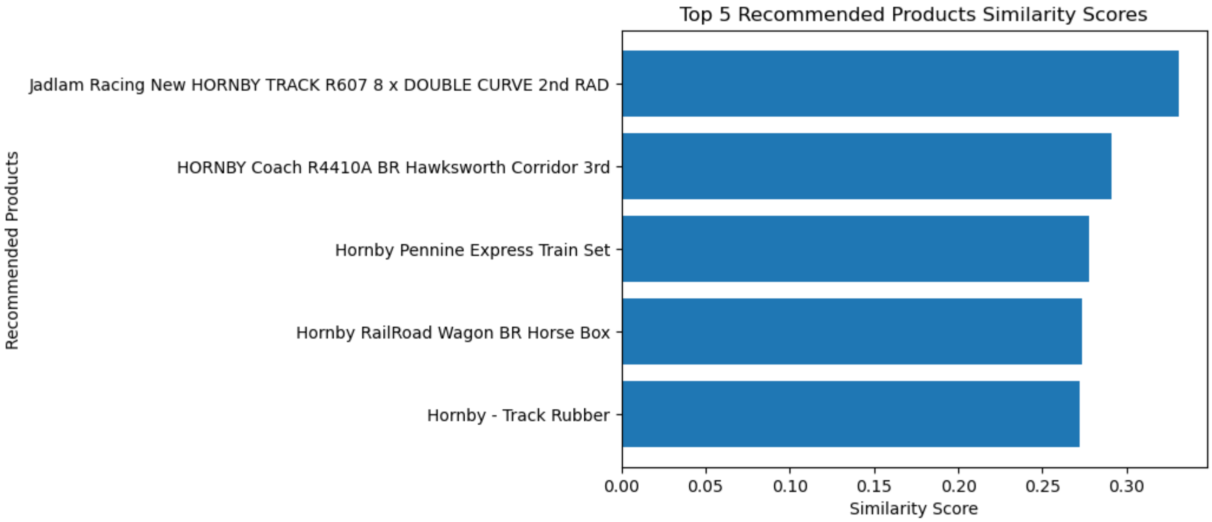


Figure 31: Bar Chart - Similarity Scores for Top Recommendation

A horizontal bar chart displays the top 5 recommended products for a sample item (index 0), along with their similarity scores. The recommendations are all Hornby-branded train accessories, with similar scores ranging from about 0.05 to 0.30. The chart clearly communicates not only which items are suggested but also how strongly they relate to the original product.

4. Conclusion

This coursework built a smart product recommendation system for online shopping. It works by mixing two ways of finding what users might like: one based on what similar customers bought, and the other based on product details like name, description, and category. This helps solve common problems like recommending new products or keeping suggestions fresh and varied.

The system uses simple but powerful AI techniques to match products, and the project includes all the steps from cleaning data to showing results with clear examples and charts. While it works well for its purpose, there’s always room to make it even better in the future, like by trying newer AI models or making it more interactive for real-world use

Based on the research and implementation carried out throughout this coursework, it can be concluded that Machine Learning plays a crucial role in generating personalized and relevant product recommendations. The detailed study of the selected problem domain product recommendation in e-commerce was a vital component of this project, as it enabled the design of a system that balances accuracy, relevance, and diversity in recommendations.

4.1 Analysis of the work done

The coursework involved a lot of research that involved reading journals, books, white papers, online articles and available e-commerce sites. The introduction part of the report gives a summary of Product recommendation Systems, Artificial intelligence and basic concepts of AI. The background section contains an in-depth overview of the studies pertaining to recommendation systems, their collaborative filtering, content-based filtering, and hybrid strategies as well as the results of the practical applications already done.

The analysis of similar recommendation systems utilized by the major e-commerce platforms was reviewed and analyzed comparatively to appreciate the existing practices and limitations. The solution proposed was clarified with reference to the algorithms involved in it. Pseudocode was created to describe the system logic and the general course of work without revealing any superfluous information about the implementation.

To visualize the process of the system, a flowchart was created, and a state transition diagram was created to show the possible states of the system and the events that occur to activate the transitions between them.

4.2 How the solution addresses real world problems

The product recommendation system is a technique used in e commerce today to overcome the issue of information overload where the customers are assisted in identifying the appropriate product out of huge product catalogs and this enhances user experience and business performance. To elaborate, the individualized product recommendation mechanism used by Amazon, which interprets the past history of browsing, purchasing patterns and interaction information of the users to predict and suggest the products that individual customers are likely to purchase, enhances customer satisfaction and increases the rate of sales conversion on large web-based retailers like Amazon; academic and technical explanations of the personalized product recommendation system underscores the fact that customers who are likely to purchase the product are tempted to buy it, which in turn boosts sales conversion rates on large web-based retailers like Amazon. The Narendran M and Syedsafi S technical article on the personalized product recommendation on Amazon describes how machine learning models such as popularity based and collaborative filtering are utilized in product recommendation that matches the user preferences thus improving sales and customer satisfaction as a whole, on the Amazon platform. (Narendran M, 2025)

References

AnalytixLabs. (2022, August 05). *What is Clustering in Machine Learning: Types and Methods*. Retrieved from AnalytixLabs: https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms/

ayman, z. b. (2022, Dec 07). *Recommendation Systems: Content-Based Filtering*. Retrieved from medium.com: https://medium.com/@zbeyza/recommendation-systems-content-based-filtering-e19e3b0a309e

Blondeau, C. (2022, Mar 23). *Building a Platform for Serving Recommendations at Etsy*. Retrieved from etsy.com: https://www.etsy.com/codeascraft/building-a-platform-for-serving-recommendations-at-etsy

Bulycheva, M. (2024, Dec 19). *Exploring the Potential of Graph Neural Networks to Transform Recommendations at Zalando*. Retrieved from zalando Science Blog: https://engineering.zalando.com/posts/2024/12/gnn-recommendations-zalando.html

Chen, M. (2024, Nov 25). *What is Machine Learning*. Retrieved from OCI: https://www.oracle.com/middleeast/artificial-intelligence/machine-learning/what-is-machine-learning/

Chibuzor Udokwu, R. Z. (2023). *Design and Implementation of a Product Recommendation System with Association and Clustering Algorithms*. Retrieved from sciencedirect.com: https://www.sciencedirect.com/science/article/pii/S1877050923003289

Copeland, B. (2025, Dec 15). *Artificial Intelligence*. Retrieved from Britannica: https://www.britannica.com/technology/artificial-intelligence

Greg Linden, B. S. (2003, Feb). *Amazon.com Recommendation Item-to-Item Collaborative Filtering*. Retrieved from cs.umd.edu: https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

Imran Hossain, M. A. (2023, Aug 15). *A Survey of Recommender System Techniques and the Ecommerce Domain*. Retrieved from arxiv.org: https://arxiv.org/abs/2208.07399

Karako, D. J. (2019, Jan 25). *How Shopify Uses Recommender Systems to Empower Entrepreneurs*. Retrieved from shopify.engineering: https://shopify.engineering/how-shopify-uses-recommender-systems-to-empower-entrepreneurs

Kharawl, A. (2023, April 20). *Content Based Filtering and Collaborative Filtering*. Retrieved from https://amanxai.com/2023/04/20/content-based-filtering-and-collaborative-filtering-difference/

Kharwal, A. (2023, April 20). *Content Based Filtering and Collaborative Filtering*. Retrieved from https://amanxai.com/2023/04/20/content-based-filtering-and-collaborative-filtering-difference/

Kitthu, H. A. (2025, May 2). *What is An Algorithm? Definition, Working, and Types*. Retrieved from simplilearn: https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-an-algorithm

Kuan Zou, A. S. (2025, Sep 07). *A Survey of Real-World Recommender Systems: Challenges, Constraints, and Industrial Perspectives*. Retrieved from arxiv.org: https://arxiv.org/html/2509.06002v1

L.Jones, M. (2023, Sep). *AI in History*. Retrieved from OXFORD ACADEMIC: https://academic.oup.com/ahr/article-abstract/128/3/1360/7282241

Loh, E. (2022, Aug 12). *Optimizing Markdown in Fashion E-Commerce with Machine Learning*. Retrieved from medium.com: https://medium.com/asos-techblog/optimizing-markdown-in-fashion-e-commerce-with-machine-learning-9f173be08ace

Mohammed Fadhel Aljunid, M. D. (2025, Feb 07). *A collaborative filtering recommender systems: Survey*. Retrieved from sciencedirect.com: https://www.sciencedirect.com/science/article/abs/pii/S0925231224014899

Narendran M, M. S. (2025, April 09). *Personalized Product Recommendation System for Amazon*. Retrieved from ijraset.com: https://www.ijraset.com/research-paper/personalized-product-recommendation-system-for-amazon

Pang Li, S. A. (2024, Dec 02). *A Survey on Deep Neural Networks in Collaborative Filtering Recommendation Systems*. Retrieved from arxiv.org: https://arxiv.org/abs/2412.01378

Schafer, J. (2025). *Collaborative Filtering Recommender Systems*. Retrieved from link.springer.com: https://link.springer.com/chapter/10.1007/978-3-540-72079-9_9

tableau.com. (2025). *The Ultimate Guide To Artificial Intelligence (AI): Definition, How It Works, Examples, History, & More*. Retrieved from tableau.

tableau.com. (2025). *The Ultimate Guide To Artificial Intelligence (AI): Definition, How It Works, Examples, History, & More*. Retrieved from tableau: https://www.tableau.com/data-insights/ai/what-is

Warhurst, M. (2025, August 27). *The Value of Personalized Product Recommendations in Ecommerce*. Retrieved from bloomreach: https://www.bloomreach.com/en/blog/ecommerce-product-recommendation-engine

Youngoh Kwon, G. B. (2025, Oct 06). *A Hybrid Recommendation System Based on Similar-Price Content in a Large-Scale E-Commerce Environment*. Retrieved from mdpi.com: https://www.mdpi.com/2076-3417/15/19/10758

Yuqiu Zhao, C. T. (2026, March). *Generative recommender systems: A comprehensive survey on model, framework, and application*. Retrieved from sciencedirect.com: https://www.sciencedirect.com/science/article/abs/pii/S1566253525009819