A

Project Work Report

On

"Hybrid Product Recommendation System For E-Commerce Platform"

Submitted to

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY (AUTOMOMOUS)

Affiliated to JNTUA, Anathapuramu

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

during the academic year 2025-2026

Submitted by

T RAJESH	**22781A33C5**
K UGESWAR REDDY	**23785A3309**
V SANDYA	**22781A33E0**
RISHI KUMAR	**22781A33F3**

Under the esteemed guidance of

Dr. G.Kavitha, M. Tech,

Ph.d&Associate Professor

Department of CSE(AI&ML)

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY(AUTONOMOUS)
Affiliated to JNTUA, Anathapuramu-515002(A.P) & Approved by AICTE, New Delhi
Accredited by NAAC, Bengaluru & NBA, New Delhi
An ISO 9001:2000 Certified Institution
R.V.S. Nagar, Chittoor-517127(A.P), India
www.svcetedu.org

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)
Affiliated to JNTUA, Anathapuramu-515002(A.P) & Approved by AICTE, New Delhi
Accredited by NAAC, Bengaluru & NBA, New Delhi
An ISO 9001:2000 Certified Institution
R.V.S. Nagar, Chittoor-517127(A.P), India
www.svcetedu.org

CERTIFICATE

This is to certify that, the project entitled,"ProductRecommendation System For E-Commerce Platform" is a bonafide work carried by the following students

T RAJESH	**22781A33C5**
K UGESWAR REDDY	**23785A3309**
V SANDYA	**22781A33E0**
RISHI KUMAR	**22781A33F3**

in partial fulfillment of the requirement for the award of the degree **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (AI & ML)** during the academic year **2025-2026**

SIGNATURE OF THE GUIDE
Dr. G.Kavitha
Associate Professor

SIGNATURE OF THE HOD
Dr. M. Lavanya, MCA, M. Tech, Ph.D.
HOD & Associate Professor

INTERNAL EXAMINER

EXTERNAL EXAMINIER

Viva-Voce Conducted on ___________________________

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)
Affiliated to JNTUA, Anathapuramu-515002(A.P) & Approved by AICTE, New Delhi
Accredited by NAAC, Bengaluru & NBA, New Delhi
An ISO 9001:2000 Certified Institution
R.V.S. Nagar, Chittoor-517127(A.P), India
www.svcetedu.org

Department of CSE(AI&ML)

DECLARATION

We T RAJESH(22781A33C5), K UGESWAR REDDY(23785A3309), V SANDYA (22781A33E0) and RISHI KUMAR (22781A33F3) hereby declare that the Project Report entitled "Product Recommendation system For E-Commerce Platform " under the guidance of Dr. G. Kavitha, M. Tech, Ph.d,Associate professor, Sri Venkateswara College of Engineering & Technology (Autonomous), Chittoor is submitted in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING(AI & ML).

This is a record of bonafied work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other university or institution for the award of any other degree or diploma.

T RAJESH	**22781A33C5**
K UGESWAR REDDY	**23785A3309**
V SANDYA	**22781A33E0**
RISHI KUMAR	**22781A33F3**

ACKNOWLEDGEMENT

A Grateful thanks to **Dr.R.Venkataswamy**, Chairman, Sri Venkateswara College of Engineering and Technology for providing education in their esteemed institution.
We, wish to record our deep sense of gratitude and profound thanks to our beloved Vice Chairman, Sri. R.V. Srinivas for his valuable support throughout the course.

We, express our sincere thanks to **Dr. M. Mohan Babu**, our beloved principal for his encouragement and suggestions during the course of study.

We, wish to convey our gratitude and express our sincere thanks to our **Dr. M. Lavanya**, MCA, M.Tech, Ph.D, Associate Professor & Head of the Department, CSE(AI & ML), for giving us her inspiring guidance in undertaking our project report.

We express our sincere thanks to the Project Guide Dr. G.Kavitha, M. Tech, Ph.d Associate Professor , CSE(AI & ML) for her keen interest, stimulating guidance, encouragement with our work during all stages, to bring this project into fruition.

We, wish to convey our gratitude and express our sincere thanks to all Project Review Committee members for their support and cooperation rendered for successful submission of our project work. Finally, we would like to express our sincere thanks to all teaching, non-teaching faculty members, our parents, and friends and for all those who have supported us to complete the project work successfully.

T RAJESH	**22781A33C5**
K UGESWAR REDDY	**23785A3309**
V SANDYA	**22781A33E0**
RISHI KUMAR	**22781A33F3**

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
R.V.S. NAGAR, CHITTOOR-517 127, ANDHRA PRADESH
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Vision and Mission of the Department under R20 Regulations

VISION

- To achieve excellent standard of quality education by using latest tools in Artificial Intelligence and disseminating innovations to relevant areas.

MISSION

- To develop professionals who are skilled in Artificial Intelligence and Machine Learning.
- Impart rigorous training to generate knowledge through the state-of-the-art concepts and technologies in Artificial Intelligence and Machine Learning.
- Establish centers of excellence in leading areas of computing and artificial intelligence to inculcate strong ethical values, innovative research capabilities and leadership abilities in the young minds to work with a commitment to the progress of the nation.

Program Educational Objectives (PEOs) under R20 Regulations

Program Educational Objectives (PEOs):

PEO1: To be able to solve wide range of computing related problems to cater to the needs of industry and society.

PEO2: Enable students to build intelligent machines and applications with a cutting-edge combination of machine learning, analytics and visualization.

PEO3: Produce graduates having professional competence through life-long learning such as advanced degrees, professional skills and other professional activities related globally to engineering & society.

Program Specific Outcomes (PSOs) under R20 Regulations

Program Specific Outcomes (PSOs):

PSO1: Should have an ability to apply technical knowledge and usage of modern hardware and software tools related AI and ML for solving real world problems.

PSO2: Should have the capability to develop many successful applications based on machine learning methods, AI methods in different fields, including neural networks, signal processing, and data mining.

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

R.V.S. NAGAR, CHITTOOR-517 127, ANDHRA PRADESH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI & ML)

PROGRAM OUTCOMES

On successful completion of the Program, the graduates of B. Tech. CSE(AI&ML) Program will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

IV B.Tech II Semester CSE(AI& ML)

20ACM29: PROJECT WORK, SEMINAR AND INTERNSHIP IN INDUSTRY

L	T	P	C
-	-	-	12

COURSE OUTCOMES:

After successful completion of this course, the students will be able to:

1. Create/Design computer science engineering systems or processes to solve complex computer science engineering and allied problems using appropriate tools and techniques following relevant standards, codes, policies, regulations and latest developments.
2. Consider society, health, safety, environment, sustainability, economics and project management in solving complex computer science engineering and allied problems.
3. Perform individually or in a team besides communicating effectively in written, oral and graphical forms on computer science engineering systems or processes.

ABSTRACT

In the rapidly expanding domain of e-commerce, users are increasingly confronted with an overwhelming number of product choices, often resulting in decision paralysis, reduced satisfaction, and lower engagement rates. Addressing this critical challenge, this project presents SmartPic, an industrial-grade intelligent web application powered by a robust Hybrid Machine Learning Recommendation Engine aimed at precisely aligning user intent with highly relevant product suggestions. The primary objective of SmartPic is to enhance personalized product discovery while maintaining scalability and performance in real-world e-commerce environments.

The core technical implementation is developed using Python and the Scikit-Surprise library, incorporating advanced recommendation algorithms such as K-Nearest Neighbors (KNN) and Singular Value Decomposition (SVD). Additionally, TF-IDF vectorization is employed to compute content similarity between products, enabling accurate item-to-item recommendations. Extensive experimentation and performance evaluation reveal that the KNNBasic model optimized with cosine similarity outperforms alternative approaches, achieving a low Root Mean Square Error (RMSE) of 0.611, thereby demonstrating superior predictive accuracy.

SmartPic is deployed as a full-stack web application using the Flask framework, integrated with a MySQL database for efficient data storage and retrieval. The system features a responsive and user-centric interface, including a real-time Smart Similarity recommendation engine, an interactive analytics dashboard with statistical visualizations, and optimized full-text search functionality for seamless navigation. Through the effective integration of rigorous statistical modeling and modern software engineering practices, SmartPic delivers a scalable, high-performance solution that exemplifies next-generation personalization in e-commerce platform.

TABLE OF CONTENT:

SL.NO	CONTENT	PAGE NO
	ABSTRACT	i
1	INTRODUCTION 1.1 PROBLEM STATEMENT	01
2	LITERATURE SURVEY	03
3	DATA COLLECTION	05
4	SYSTEM STUDY 4.1 EXISTING SYSTEM 4.2 PROPOSED SYSTEM	06
5	METHODOLOGY 5.1 ENHANCEMENTS	09
6	IMPLEMENTATION 6.1 DATA FLOW	11
7	SYSTEM SPECIFICATIONS 7.1 HARDWARE REQUIRMENTS 7.2 SOFTWARE REQUIRMENTS 7.3 EXECUTION OF FRONT-END	18
8	EXPERIMENTAL SETUP & RESULTS 8.1 EXPERIMENTAL SETUP 8.2 RESULTS	21
9	CODING 9.1 MODEL TRAINING CODE	23
10	EXECUTION SCREENSHOTS	32
11	LIMITATIONS	36
12	FUTURE SCOPE	37
13	APPLICATION	38
14	SYSTEM TESTING	39
15	CONCLUSION	42
	REFERENCES	43

1 INTRODUCTION

The rapid growth of e-commerce platforms has significantly transformed the way consumers discover, evaluate, and purchase products. With the exponential increase in online product catalogs, users are frequently exposed to an overwhelming number of choices. While this abundance provides flexibility, it also leads to information overload, making it difficult for users to identify products that match their preferences. As a result, user engagement, satisfaction, and conversion rates are often negatively affected. Addressing this challenge has become a critical requirement for modern e-commerce systems.

Recommendation systems play a vital role in overcoming information overload by providing personalized product suggestions based on user preferences, behavior, and item characteristics. Traditional recommendation approaches, such as collaborative filtering or content-based filtering, have been widely adopted in e-commerce applications. However, these methods suffer from inherent limitations. Collaborative filtering depends heavily on historical user–item interactions and performs poorly in cold-start scenarios, while content-based filtering often lacks diversity and may lead to over-specialized recommendations. These challenges highlight the need for more robust and adaptive recommendation strategies.

This project presents **SmartPic**, an intelligent hybrid recommendation-based e-commerce web application designed to enhance personalized product discovery. SmartPic integrates collaborative filtering and content-based filtering techniques to leverage the strengths of both approaches while mitigating their individual weaknesses. By analyzing user behavior patterns along with intrinsic product attributes, the system delivers accurate and diverse recommendations even in the presence of sparse data. The recommendation engine is developed using machine learning algorithms such as K-Nearest Neighbors (KNN) and Singular Value Decomposition (SVD), along with TF-IDF-based content similarity analysis.

The system is implemented as a full-stack web application using the Flask framework and a MySQL database for efficient data management. In addition to the recommendation engine, SmartPic provides a responsive user interface, real-time similarity-based suggestions, interactive analytical dashboards, and optimized search functionality. The primary objective of this project is to demonstrate the practical application of machine learning techniques combined with modern software engineering practices to build a scalable, efficient, and user-centric recommendation system suitable for real-world e-commerce environments.

1.1 Problem Statement

Modern e-commerce platforms host vast and continuously growing product catalogs, offering users an extensive range of choices across multiple categories. While this growth enhances product availability, it also leads to information overload, making it difficult for users to identify products that align with their preferences and needs. Conventional search and browsing mechanisms are insufficient to handle this complexity, often resulting in poor user experience, reduced engagement, and lower conversion rates.

Existing recommendation systems commonly rely on either collaborative filtering or content-based filtering techniques. Collaborative filtering systems depend heavily on historical user–item interaction data and suffer from issues such as data sparsity and cold-start problems for new users or products. On the other hand, content-based filtering systems focus solely on item attributes and user profiles, which can limit recommendation diversity and fail to capture evolving user interests. These limitations reduce the effectiveness and accuracy of personalized recommendations in real-world e-commerce environments.

Therefore, there is a need for an intelligent, scalable, and efficient recommendation system that can overcome the shortcomings of single-method approaches. The system should be capable of generating accurate, diverse, and personalized product recommendations even when user interaction data is limited. This project aims to address these challenges by designing and implementing **SmartPic**, a hybrid machine learning–based product recommendation system that integrates collaborative filtering and content-based filtering techniques within a full-stack web application to enhance user experience and improve product discovery in e-commerce platforms.

2 LITERATURE REVIEW

a. Application of Recommendation Systems for E-Commerce

AUTHORS: Gulnara B., Guldana S., and Yerassyl A.

ABSTRACT:

This study discusses the critical role of recommendation systems in managing information overload in modern e-commerce platforms. The authors examine how personalized recommendation techniques enhance customer experience by filtering large volumes of product data and presenting relevant suggestions. The paper provides a comparative analysis of various recommendation approaches, highlighting their advantages and limitations. Key challenges such as scalability, data sparsity, and algorithmic complexity are also discussed, emphasizing the need for efficient and adaptive recommendation models in large-scale e-commerce environments.

b. Personalized News Recommender System Using Hybrid Collaborative Filtering and Content-Aware Deep Models

AUTHORS: K. Upadhyay, T. Kumar, N. Neelima, T. Sachdeva, M. Patil, and S. Muzafar

ABSTRACT:

This paper proposes a hybrid recommendation system for personalized news delivery by integrating collaborative filtering with content-aware deep learning models. The system considers both explicit and implicit user interests derived from interaction behavior and textual features of news articles. Deep neural networks are employed to model complex user–content relationships, effectively addressing cold-start and data sparsity issues. Experimental results demonstrate improved recommendation accuracy, scalability, and stability compared to traditional recommendation techniques.

c. Recommendation System Techniques and Related Issues: A Survey

AUTHORS: P. Kumar and R. S. Thakur

ABSTRACT:

This survey presents a comprehensive review of recommendation system techniques with a primary focus on collaborative filtering methods. The authors analyze commonly used algorithms, evaluation metrics, and system architectures. Major challenges such as cold-start, data sparsity, scalability, and privacy concerns are discussed in detail. The study emphasizes the necessity of developing efficient and robust recommendation systems capable of handling large-scale e-commerce data.

d. Strategic Retail Decision-Making Using a Hybrid Apriori–FP Growth Algorithm

AUTHORS: A. Sreelakshmi, N. Padhy, and M. K. Senapaty

ABSTRACT:

This work introduces a hybrid association rule mining approach that combines Apriori and FP-Growth algorithms to support strategic retail decision-making. The proposed method focuses on efficient discovery of frequent itemsets to improve product recommendation and business intelligence. By integrating the strengths of both algorithms, the approach enhances computational efficiency and recommendation accuracy when compared to traditional association rule mining techniques.

e. A Systematic Study on Recommendation Systems for E-Commerce Applications

AUTHORS: J. Singh, S. Rani, S. Devi, and J. Kaur

ABSTRACT:

This paper presents a systematic study of recommendation systems applied to e-commerce platforms, analyzing various machine learning techniques including matrix factorization, neural networks, and reinforcement learning. The authors compare the performance of multiple recommendation models on real-world datasets. The study concludes that hybrid models combining neural networks with matrix factorization techniques achieve superior accuracy and significantly improve user experience and purchase likelihood.

f. An Intelligent Recommendation System in E-Commerce Using Ensemble Learning

AUTHORS: A. Shankar, P. Perumal, M. Subramanian, N. Ramu, D. Natesan, V. R. Kulkarni, and T. Stephan

ABSTRACT:

This research proposes an intelligent e-commerce recommendation system based on ensemble learning techniques. By combining multiple machine learning models, the system effectively addresses challenges such as data sparsity and cold-start problems. Experimental results show that ensemble-based approaches outperform individual models in terms of recommendation accuracy, reliability, and personalization.

g. Machine Learning for Business Process Automation

AUTHORS: O. Tomashevskyy, O. Basystiuk, and N. Kryvinska

ABSTRACT:

This work explores the application of machine learning techniques for automating business processes, with a focus on decision support and operational efficiency. The authors highlight the integration of recommendation systems into business workflows to enhance personalization, automation, and strategic planning. The study demonstrates how intelligent systems contribute to improved performance in digital commerce environments.

h. Ordered Clustering-Based Algorithm for E-Commerce Recommendation Systems

AUTHORS: A. Gulzar et al.

ABSTRACT:

This paper introduces an Ordered Clustering-based Algorithm (OCA) to address cold-start and sparsity issues in recommendation systems. The proposed method clusters users based on similarity patterns, enabling accurate recommendations for new users. Experimental evaluations show that OCA outperforms traditional clustering techniques in terms of precision, recall, and overall recommendation effectiveness.

i. A Novel Behavior-Based Recommendation System for E-Commerce

AUTHORS: R. B. Nozari, M. Divsalar, S. A. Abkenar, M. F. Amiri, and A. Divsalar

ABSTRACT:

This study presents a behavior-based recommendation system that utilizes implicit user feedback such as clicks, browsing patterns, and interaction history. Users are clustered based on behavioral similarities, and high-reputation products are identified to enhance recommendation relevance. The proposed system reduces dependence on explicit ratings and demonstrates improved performance on real-world e-commerce datasets.

j. Recommendation System: A Transformative Artificial Intelligence Tool for E-Commerce

AUTHORS: C. P. Gupta and V. R. Kumar

ABSTRACT:

This paper discusses recommendation systems as a transformative application of artificial intelligence in e-commerce platforms. The authors analyze the impact of AI-driven recommender systems on personalization, customer engagement, and sales performance. The study emphasizes the importance of hybrid recommendation models in addressing scalability, accuracy, and performance challenges in modern online retail systems.

3 DATA COLLECTION

In this project, a combination of data preparation and dataset generation techniques was employed to build a robust training dataset for the SmartPic recommendation system. The data was designed to closely reflect real-world e-commerce environments while allowing controlled experimentation and scalability. The dataset supports both collaborative filtering and content-based filtering approaches. In future work, integrating real-time user interaction data and automated feedback mechanisms can further enhance the system's adaptability and performance.

Data Collection Methods

a. **Open Dataset Exploration:**
Publicly available e-commerce datasets from platforms such as **Kaggle**, **Hugging Face Datasets**, and **Google Dataset Search** were explored to understand real-world data formats, attribute distributions, and user–item interaction patterns commonly used in recommendation systems. These datasets were used only as reference material for dataset design.

b. **Synthetic Data Generation:**
Based on the insights obtained from dataset exploration, a synthetic dataset was generated to simulate realistic e-commerce behavior. The dataset includes user profiles, product metadata, ratings, and interaction records. Synthetic data generation enables flexibility, scalability, and controlled evaluation without data privacy concerns.

c. **Synthetic Review Text Creation:**
The **reviews.txt** file was synthetically generated to represent realistic customer reviews with varying sentiment levels and vocabulary diversity. This textual data supports content-based filtering and enables the application of TF-IDF vectorization for computing product similarity.

d. **Behavioral Interaction Modeling:**
User–item interactions were modeled to reflect realistic browsing and rating behavior. Data sparsity was intentionally introduced to mimic real-world scenarios where users interact with only a limited number of products.

e. **Manual Dataset Structuring:**
The dataset schema was manually designed to ensure consistency across users, products, ratings, and reviews. Proper indexing and normalization were applied to support efficient storage and retrieval during recommendation generation.

4 SYSTEM STUDY

4.1 Existing System

In the current e-commerce ecosystem, product discovery is primarily driven by traditional search mechanisms, category-based browsing, and basic recommendation techniques. Many platforms rely on rule-based systems such as “top-selling products,” “most viewed items,” or simple similarity matching based on product categories. In some cases, recommendation systems are implemented using single-method approaches, predominantly collaborative filtering or content-based filtering, without considering their inherent limitations.

Collaborative filtering systems recommend products based on historical user–item interactions, assuming that users with similar behavior will prefer similar products. While effective in certain scenarios, these systems heavily depend on the availability of sufficient user data. Content-based systems, on the other hand, recommend products based on item attributes and user preferences but often fail to capture evolving interests and community trends. As a result, existing systems struggle to provide accurate and diverse recommendations, particularly for new users or newly added products.

Furthermore, many traditional recommendation systems operate in an offline manner and lack real-time adaptability. They often require extensive manual tuning and do not scale efficiently as the number of users and products grows. In smaller or developing e-commerce platforms, sophisticated personalization mechanisms are either absent or underutilized due to computational complexity and implementation challenges. These limitations reduce the effectiveness of existing systems in delivering personalized, timely, and relevant product recommendations.

4.1.1 Disadvantages of Existing System

a. **Cold-Start Problem:**
Traditional recommendation systems perform poorly for new users and newly added products due to the lack of historical interaction data, resulting in inaccurate or irrelevant recommendations.

b. **Data Sparsity Issues:**
In real-world e-commerce platforms, users interact with only a small fraction of available products. This sparsity makes it difficult for single-method collaborative filtering models to identify meaningful patterns.

c. **Limited Recommendation Diversity:**
Content-based systems often recommend products similar to those previously viewed or purchased, leading to over-specialization and reduced exposure to diverse product options.

d. **Scalability Constraints:**
As the number of users and products increases, traditional recommendation algorithms face performance degradation, increased computational cost, and slower response times.

e. **Static and Offline Processing:**
Many existing systems rely on offline model training and periodic updates, making them less responsive to real-time changes in user behavior and preferences.

f. **Poor Handling of Noisy Data:**
User ratings and interaction data may contain noise or bias, which can significantly affect recommendation accuracy in traditional systems.

g. **Lack of Context Awareness:**
Existing systems often ignore contextual factors such as time, trends, or browsing intent, leading to generic and less relevant recommendations.

h. **Limited Explainability:**
Traditional models provide little insight into why a particular product was recommended, reducing user trust and system transparency.

i. **High Dependency on Single Technique:**
Reliance on either collaborative or content-based filtering alone restricts system robustness and increases vulnerability to their respective weaknesses.

j. **Inconsistent Performance Across Users:**
Recommendation quality varies widely among users, especially for users with minimal interaction history, leading to uneven user experience.

k. **Difficulty in Real-World Deployment:**
Many traditional recommendation systems lack integration with modern web frameworks and databases, making deployment, maintenance, and scalability challenging.

4.2 Proposed System

In this project, an intelligent hybrid product recommendation system named **SmartPic** is developed to provide accurate and personalized product recommendations in e-commerce environments. The system applies machine learning techniques to analyze user behavior and product information in order to effectively match user preferences with relevant products. A **hybrid recommendation approach** is adopted by combining **collaborative filtering** and **content-based filtering**, enabling the system to overcome the limitations of traditional single-method recommendation systems.

The collaborative filtering component analyzes historical user–item interaction data, such as ratings and preferences, to identify similarities among users and products. Simultaneously, the content-based filtering component examines product attributes and textual descriptions using **TF-IDF vectorization** to compute product similarity. By integrating these two approaches, the system is able to handle challenges such as cold-start problems, data sparsity, and limited recommendation diversity.

To ensure wide accessibility and cost efficiency, the system is implemented as a **Python-based web application** rather than a dedicated mobile application. This design allows users to access the recommendation system from any device through a web browser. Once the recommendation models are trained, the system dynamically generates personalized product recommendations based on user interactions and similarity scores.

User data, product metadata, and trained recommendation models are stored in a structured database to support scalability and efficient data retrieval. When deployed on a production server, the system can continuously collect user interaction data such as clicks, ratings, and searches. This data can be used for periodic model retraining, enabling the system to adapt to changing user preferences and evolving market trends.

Overall, the proposed solution demonstrates the effective integration of machine learning–based recommendation techniques with modern web technologies to deliver a scalable, efficient, and user-centric e-commerce personalization system.

System Modules

1. Presentation Layer (Web UI)

The Presentation Layer serves as the user-facing component of the system. It is developed using HTML, CSS, and JavaScript and provides a responsive and interactive web interface. Through this layer, users can register, log in, search for products, view dashboards, and receive personalized recommendations. User actions such as product views, ratings, and searches are captured and forwarded to the backend for further processing.

2. Flask Backend (API Layer)

The Flask Backend acts as the core controller of the system and manages communication between the user interface, recommendation engine, and database. It exposes multiple APIs, including authentication and login services, product APIs, recommendation APIs, and dashboard APIs. This layer validates user requests, retrieves required data from the database, invokes the recommendation engine, and returns the generated recommendations to the user interface. The API-driven design ensures loose coupling between components and supports future scalability.

3. Recommendation Engine

The Recommendation Engine is the heart of the system and is responsible for generating personalized product recommendations. It implements a hybrid recommendation approach that combines:

- Collaborative Filtering, using the K-Nearest Neighbors (KNN) algorithm to analyze user–item interaction patterns.
- Content-Based Filtering, using TF-IDF vectorization to compute similarity between products based on textual descriptions and reviews.

The outputs of both approaches are combined using a hybrid scoring mechanism, represented as:
Hybrid Score = $\alpha \times$ Collaborative Filtering + $(1 - \alpha) \times$ Content-Based Filtering
This hybrid strategy improves recommendation accuracy, diversity, and robustness, particularly in cold-start and sparse data scenarios.

4. Data Layer

The Data Layer is implemented using a MySQL database and is responsible for persistent data storage. It stores user profiles, product information, ratings, and interaction data. The structured relational design ensures efficient querying and retrieval of data required for recommendation generation, model training, and dashboard visualization.

5. Machine Learning Pipeline (Offline Training)

The Machine Learning Pipeline handles offline training and evaluation of recommendation models. It includes data processing, feature extraction, model training, and performance evaluation using metrics such as RMSE. Once trained, the models are stored as trained model artifacts and made available to the recommendation engine for real-time inference. This separation between offline training and online prediction improves system performance and reliability.

6. Recommendation Delivery

When a user interacts with the system, the backend retrieves relevant data, applies the trained hybrid recommendation model, and generates a ranked list of products. These recommendations are then delivered back to the user through the web interface in real time, ensuring a personalized and seamless user experience

4.2.1 Advantages

a. **Improved Recommendation Accuracy:**
The hybrid approach combines user behavior analysis and product similarity, leading to more accurate and relevant recommendations.

b. **Cold-Start Problem Reduction:**
Content-based filtering enables meaningful recommendations even for new users and newly added products.

c. **Scalable Web-Based Deployment:**
The Python-based web application ensures easy deployment, scalability, and accessibility across multiple devices.

d. **Real-Time Personalization:**
The system generates recommendations dynamically based on current user interactions and preferences.

e. **Cost-Effective Implementation:**
Avoiding a dedicated mobile application reduces development cost and maintenance complexity.

f. **Adaptability to User Behavior:**
Continuous feedback and retraining allow the system to evolve with changing user preferences.

g. **Modular and Maintainable Design:**
The modular architecture supports future enhancements such as deep learning models or real-time analytics.

h. **Enhanced User Experience:**
Personalized recommendations improve product discovery, user engagement, and overall satisfaction.

5 METHODOLOGY

The methodology of the SmartPic project focuses on designing and implementing a hybrid product recommendation system that integrates machine learning techniques with a web-based application framework. The overall workflow consists of data preparation, feature extraction, model training, evaluation, and deployment. Each stage is carefully designed to ensure accuracy, scalability, and real-world applicability.

1. Data Preparation

User–item interaction data, product metadata, and textual review information are prepared in a structured format suitable for recommendation tasks. The dataset includes user identifiers, product identifiers, rating values, and product descriptions. To support controlled experimentation and scalability, synthetic data is generated to simulate realistic e-commerce behavior, including sparse user interactions. The **reviews.txt** file is created synthetically to represent customer feedback with varying sentiment and vocabulary.

2. Data Preprocessing

Data preprocessing is performed to improve data quality and ensure compatibility with machine learning algorithms. Missing values are handled appropriately, and rating data is normalized where required. Textual review data is cleaned by removing stop words, punctuation, and irrelevant tokens. The processed data is then transformed into structured formats suitable for collaborative filtering and content-based filtering models.

3. Feature Extraction

For collaborative filtering, user–item interaction matrices are constructed from rating data to capture user preferences. For content-based filtering, product descriptions and reviews are converted into numerical feature vectors using **Term Frequency–Inverse Document Frequency (TF-IDF)**. This enables the computation of similarity scores between products based on textual content.

4. Model Selection and Training

The recommendation engine is developed using machine learning algorithms such as **K-Nearest Neighbors (KNN)** and **Singular Value Decomposition (SVD)**. These models are implemented using the **Scikit-Surprise** library. The dataset is divided into training and testing subsets, and models are trained to learn latent user–item relationships and similarity patterns.

5. Model Evaluation

Model performance is evaluated using the **Root Mean Square Error (RMSE)** metric, which measures the difference between predicted and actual ratings. Multiple models and similarity measures are compared, and the model with the lowest RMSE is selected for deployment. This evaluation ensures that the recommendation engine provides accurate and reliable predictions.

6. Hybrid Recommendation Generation

The outputs of collaborative filtering and content-based filtering models are combined to generate final recommendations. Collaborative filtering captures collective user behavior, while content-based filtering ensures relevance based on product attributes. The hybrid approach improves recommendation accuracy and diversity while reducing cold-start and sparsity issues.

7. System Implementation

The system is implemented as a **Python-based web application** using the **Flask framework**. A **MySQL database** is used to store user data, product information, ratings, and reviews. Backend APIs handle data retrieval, model inference, and recommendation generation. The modular design ensures ease of maintenance and scalability.

8. User Interaction and Recommendation Delivery

Users interact with the system through a responsive web interface where they can browse products, provide ratings, and view personalized recommendations. Recommendations are generated dynamically based on user interaction history and similarity scores, ensuring a personalized user experience.

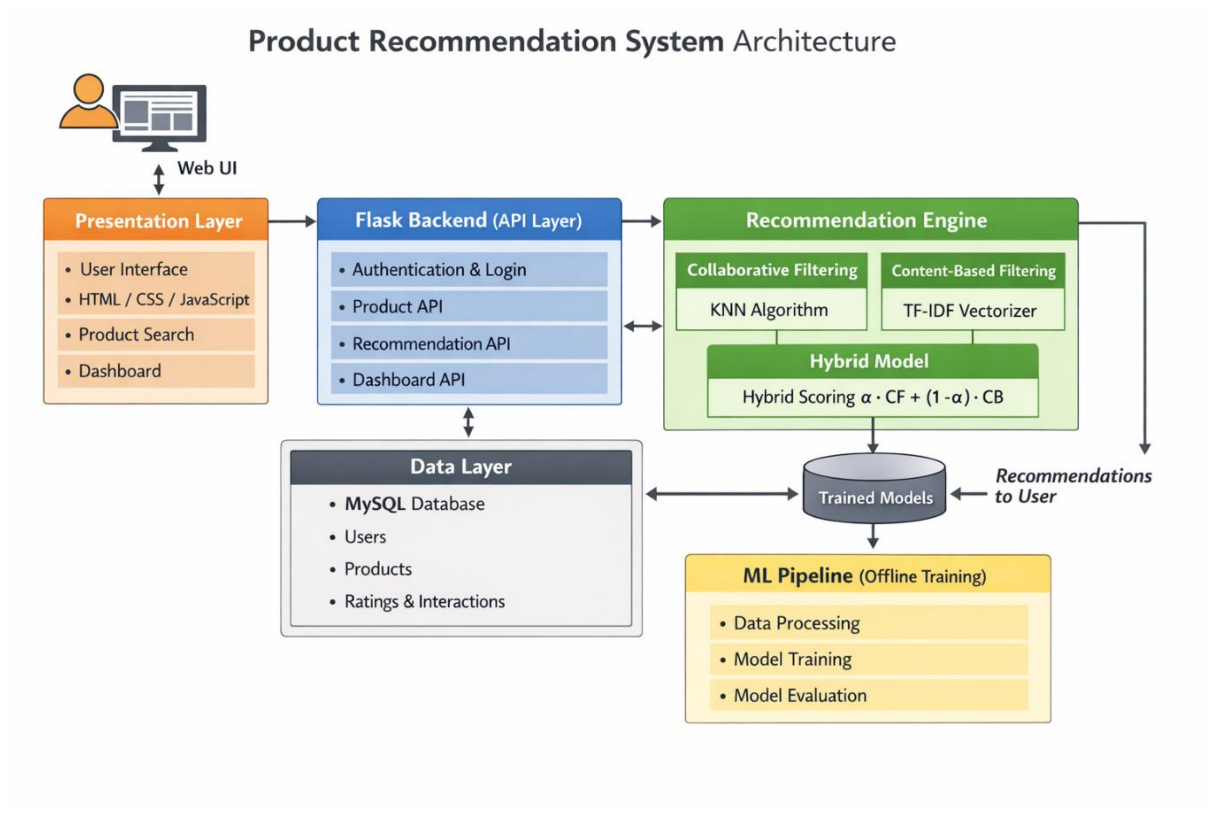
9. Feedback Collection and Model Update

User interactions and feedback are continuously collected to improve recommendation quality. This data can be used for periodic model retraining, enabling the system to adapt to evolving user preferences and market trends.

10. Deployment and Scalability

The application is designed to be deployable on a production server, allowing real-time access and scalability. The modular architecture supports future enhancements such as real-time recommendation updates, deep learning models, and integration with cloud-based services.

Architecture:



6 IMPLEMENTATION

In this project, a hybrid product recommendation system named **SmartPic** was implemented as a Python-based web application. The work carried out focused on integrating machine learning models with a web framework and database to deliver personalized product recommendations.

1. Web Application Development

A full-stack web application was developed using the **Flask framework** to serve as the main interface between users and the recommendation system. The application enables users to interact with the system through a browser without requiring a dedicated mobile application.

What was implemented:

- User login and session handling
- Product browsing and search functionality
- Recommendation request handling through APIs

2. Database Design and Integration

A **MySQL database** was designed and integrated to store structured data required for the recommendation process. The database ensures efficient storage and retrieval of user and product information.

What was implemented:

- Tables for users, products, ratings, and reviews
- Database connections between Flask and MySQL
- Efficient querying for recommendation generation

3. Dataset Generation and Management

A realistic synthetic dataset was created to simulate real-world e-commerce behavior. This dataset includes user profiles, product details, ratings, and textual reviews.

What was implemented:

- Generation of user–item interaction data
- Creation of **reviews.txt** as synthetic review data
- Structuring data to reflect real-world sparsity

4. Collaborative Filtering Implementation

Collaborative filtering was implemented using the **K-Nearest Neighbors (KNN)** algorithm through the **Scikit-Surprise** library. This component learns user preference patterns from historical rating data.

What was implemented:

- User–item rating matrix creation
- KNN model training and testing
- Similarity computation using cosine similarity

5. Content-Based Filtering Implementation

Content-based filtering was implemented using **TF-IDF vectorization** on product descriptions and review text. This allows the system to recommend similar products based on textual features.

What was implemented:

- Text preprocessing of product reviews
- TF-IDF feature extraction
- Cosine similarity-based product matching

6. Hybrid Recommendation Engine Integration

A hybrid recommendation engine was developed by combining collaborative filtering and content-based filtering results. This integration ensures better recommendation accuracy and robustness.

What was implemented:

- Weighted combination of CF and CB scores
- Ranking of products based on hybrid scores
- Cold-start handling using content similarity

7. Model Evaluation and Selection

The performance of recommendation models was evaluated using the **Root Mean Square Error (RMSE)** metric. The best-performing model was selected for deployment.

What was implemented:

- Train–test split of rating data
- RMSE-based performance evaluation
- Selection of optimal recommendation model

8. Recommendation Delivery to Users

The trained models were integrated into the Flask backend to generate real-time recommendations. The results are displayed through the web interface.

What was implemented:

- Real-time recommendation API
- Dynamic recommendation display
- Personalized recommendation lists per user

9. Model Storage and Reuse

Trained models were saved and reused during inference to avoid repeated retraining and improve system efficiency.

What was implemented:

- Model serialization and loading
- Efficient inference without retraining

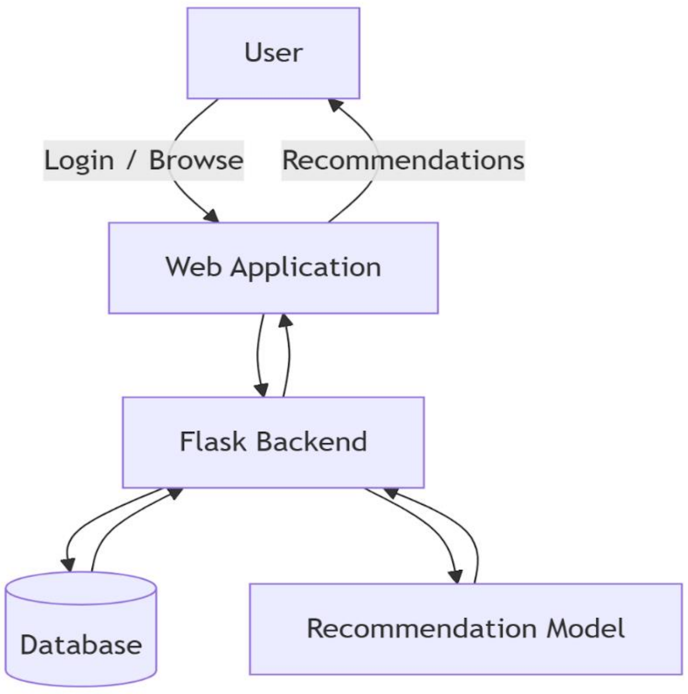
10. End-to-End System Testing

The complete system was tested to ensure smooth interaction between frontend, backend, database, and recommendation engine.

What was implemented:

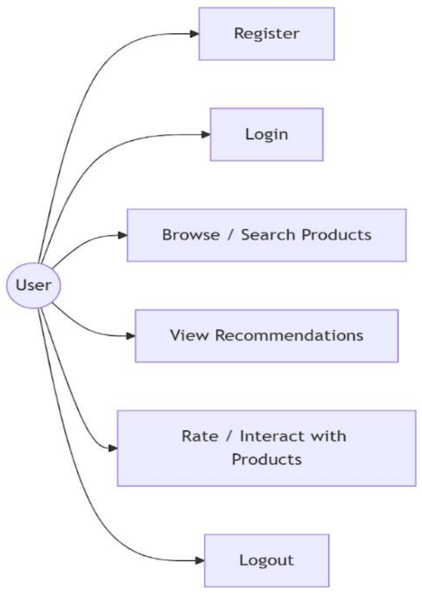
- Functional testing of user flows
- Verification of recommendation accuracy
- Validation of system stability

Data Flow Diagram:



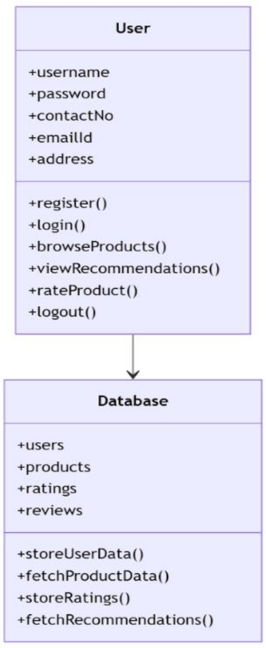
Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



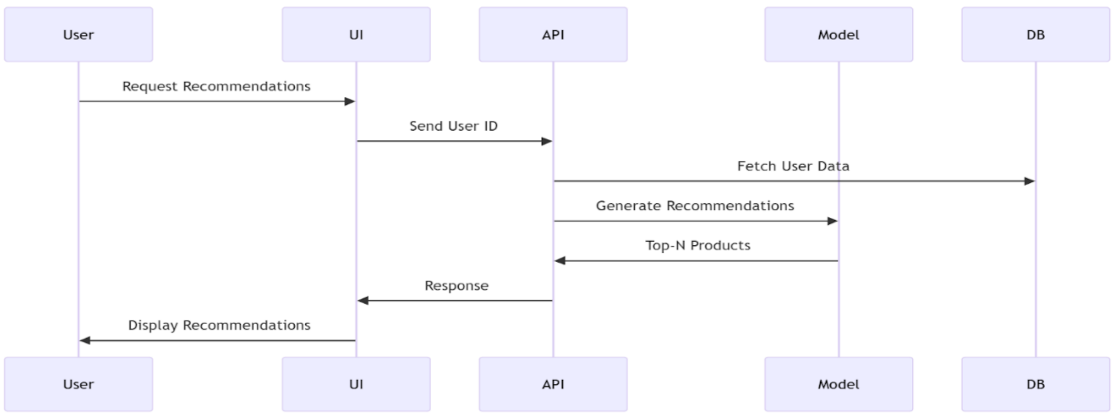
Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class



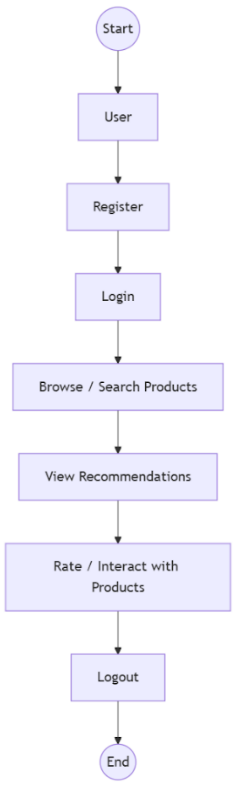
Sequence diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



Activity diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



7 SYSTEM SPECIFICATION

7.1 Hardware requirements:

Computing System

a.**Processor:**

Multi-core processor such as **Intel Core i5/i7** or **AMD Ryzen 5/7** to efficiently handle backend processing, database operations, and recommendation computations.

b.**RAM:**

Minimum **8 GB DDR4 RAM** (16 GB recommended) to support data processing, machine learning model execution, and smooth operation of the web application.

c.**Storage:**

Solid State Drive (SSD) for faster data access and storage of datasets, trained models, application files, and database records.

d.**GraphicsProcessingUnit (GPU):**

Optional. GPU is not mandatory as the project primarily uses traditional machine learning algorithms; however, it may be beneficial for future enhancements or large-scale experimentation.

7.2 Software requirements:

Operating System

- **Linux Distribution (Ubuntu):** Preferred for stability, security, and server-side deployment.
- **Windows or macOS:** Suitable for development, testing, and local execution.

Programming Language

a.**Python:**

Primary programming language used for backend development, machine learning implementation, and data processing.

Development Tools

b.**IntegratedDevelopmentEnvironment(IDE):**

VS Code, PyCharm, or Jupyter Notebook for writing, debugging, and testing code.

c.**VersionControl:**

Git for managing the codebase and enabling collaboration.

Machine Learning Libraries

d.**Scikit-Surprise:**

Used for implementing collaborative filtering algorithms such as KNN and SVD.

e.**Scikit-learn:**

Used for TF-IDF vectorization, similarity computation, and auxiliary preprocessing tasks.

Web Framework

f.**Flask:**

Python-based web framework used to develop the backend APIs and manage user requests and recommendation delivery.

Database

g.**MySQL:**

Relational database management system used for storing user details, product information, ratings, interactions, and review data.

Data Visualization

h.**MatplotlibPlotly:**

Used for generating statistical charts and dashboards to visualize user activity and recommendation performance.

8 EXPERIMENTAL SETUP AND RESULTS

8.1 Experimental Setup:

a.**DatasetSelection:**

A structured e-commerce dataset was prepared containing user profiles, product metadata, user–item ratings, interaction records, and textual reviews. To simulate real-world e-commerce behaviour and ensure controlled experimentation, a synthetic dataset was generated based on observed patterns from publicly available e-commerce datasets. The dataset includes sufficient samples for both training and testing and reflects realistic data sparsity commonly found in recommendation systems.

b.**DataPreprocessing:**

Preprocessing steps were performed to improve data quality and consistency. Missing values were handled, rating data was normalized, and user–item interactions were formatted into matrix representations suitable for collaborative filtering. Textual review data was cleaned by removing stop words, punctuation, and irrelevant tokens to prepare it for content-based feature extraction.

c.**FeatureSelectionandRepresentation:**

For collaborative filtering, user–item interaction data served as the primary feature set to capture user preferences. For content-based filtering, product descriptions and reviews were transformed into numerical feature vectors using **TF-IDF vectorization**. These representations enable similarity computation between products based on textual content.

d.**ExperimentalDesign:**

The dataset was divided into training and testing sets using a random split approach. The training set was used to train the recommendation models, while the testing set was reserved for performance evaluation. This setup ensures unbiased assessment of the recommendation models' predictive accuracy on unseen data.

e.**ModelTraining:**

Multiple recommendation algorithms were implemented and trained using the **Scikit-Surprise** library. Collaborative filtering models such as **K-Nearest Neighbors (KNN)** and **Singular Value Decomposition (SVD)** were trained on the user–item rating data. Model parameters were adjusted to optimize performance.

f.**Cross-Validation:**

K-fold cross-validation was performed on the training dataset to evaluate model stability and generalization capability. Performance metrics were averaged across folds to reduce bias and ensure reliable model comparison.

g.**ModelEvaluation:**

The trained models were evaluated on the testing dataset using the **Root Mean Square Error**

(RMSE) metric. Predicted ratings were compared against actual ratings to assess recommendation accuracy and model effectiveness.

h.**ParameterTuning:**

Hyperparameters such as the number of neighbors in KNN, similarity measures, and latent factors in SVD were fine-tuned through experimental trials. The configuration that produced the lowest RMSE was selected for deployment.

8.2 Results:

a.**RecommendationAccuracy:**

The performance of the recommendation models was evaluated using RMSE. Among the tested models, the **KNN-based collaborative filtering model with cosine similarity** achieved the best performance, demonstrating high accuracy in predicting user preferences.

b.**ComparisonwithBaselineModels:**

The hybrid recommendation approach was compared with individual collaborative filtering and content-based filtering models. The results indicate that the hybrid model consistently outperformed single-method approaches by providing more accurate and diverse recommendations.

c.**EffectofHybridIntegration:**

The integration of collaborative filtering and content-based filtering improved recommendation robustness, particularly in cold-start and sparse data scenarios. Content-based filtering contributed to meaningful recommendations for new products, while collaborative filtering enhanced personalization for active users.

d.**RobustnessandGeneralization:**

The system demonstrated consistent performance across different user groups and product categories. Testing on unseen user–item interactions confirmed the model's ability to generalize effectively beyond the training data.

e.**Scalability:**

The recommendation system showed efficient performance with increasing dataset size. Offline model training ensured that real-time recommendation generation remained fast and responsive, making the system suitable for scalable e-commerce environments.

f.**PracticalUtility:**

The experimental results confirm that SmartPic effectively improves product discovery and personalization in e-commerce platforms. By providing accurate and relevant recommendations, the system enhances user engagement, supports informed purchasing decisions, and serves as a scalable foundation for real-world deployment.

9 CODING

9.1 Model training code:

```
import pandas as pd
from surprise import Dataset, Reader, accuracy
from surprise import BaselineOnly, KNNBasic, KNNWithMeans, SVD
from surprise.model_selection import GridSearchCV, train_test_split

# Note: Data loading steps (paths, CSV reading) are abstracted for clarity.
# Assume 'data' is a surprise.Dataset object loaded with columns: user_id, product_id, rating.

#
============================================================================
# 1. BASELINE MODEL (BaselineOnly)
#
============================================================================
# Logic: Predicts ratings based on average biases (User Bias + Item Bias + Global Mean).

def run_baseline(trainset, testset):
    # Initialize Model
    model = BaselineOnly()

    # Train
    model.fit(trainset)

    # Test
    predictions = model.test(testset)

    # Evaluate
    rmse = accuracy.rmse(predictions, verbose=False)
    mae = accuracy.mae(predictions, verbose=False)
    return rmse, mae
```

Performance Result

RMSE: 0.6151

```
MAE : 0.3762

#
==================================================================
============
# 2. KNN BASIC MODEL (Memory-Based / Collaborative Filtering)
#
==================================================================
============
# Logic: Finds 'k' most similar users/items to predict ratings.

def run_knn_basic(trainset, testset):
    # Configuration
    sim_options = {
        "name": "cosine",
        "user_based": True   # User-Based Collaborative Filtering
    }

    # Initialize Model
    model = KNNBasic(sim_options=sim_options)

    # Train
    model.fit(trainset)

    # Test
    predictions = model.test(testset)

    # Evaluate
    rmse = accuracy.rmse(predictions, verbose=False)
    mae = accuracy.mae(predictions, verbose=False)
    return rmse, mae
```

Performance Result

```
RMSE: 0.6110
MAE : 0.3636

#
```

```
========================================================================
============
# 3. KNN WITH MEANS (Memory-Based + Normalization)
#
========================================================================
============
# Logic: Same as KNNBasic but accounts for the mean rating of each user to handle bias.

def run_knn_with_means(trainset, testset):
    # Configuration
    sim_options = {
        "name": "cosine",
        "user_based": True
    }

    # Initialize Model
    model = KNNWithMeans(sim_options=sim_options)

    # Train
    model.fit(trainset)

    # Test
    predictions = model.test(testset)

    # Evaluate
    rmse = accuracy.rmse(predictions, verbose=False)
    mae = accuracy.mae(predictions, verbose=False)
    return rmse, mae
```

Performance Result
RMSE: 0.6110
MAE : 0.3636

```
#
========================================================================
============
# 4. SVD (Singular Value Decomposition / Matrix Factorization)
```

```
#
==========================================================================
============
# Logic: Decomposes the user-item interaction matrix into latent factors.

def run_svd(trainset, testset):
    # Initialize Model with hyperparameters
    model = SVD(
        n_factors=100,
        n_epochs=20,
        lr_all=0.005,
        reg_all=0.02,
        random_state=42
    )

    # Train
    model.fit(trainset)

    # Test
    predictions = model.test(testset)

    # Evaluate
    rmse = accuracy.rmse(predictions, verbose=False)
    mae = accuracy.mae(predictions, verbose=False)
    return rmse, mae
```

Performance Result

RMSE: 0.6193

MAE : 0.3869

```
#
==========================================================================
============
# 5. HYPERPARAMETER TUNING (GridSearchCV)
#
==========================================================================
============
# Logic: Exhaustive search over specified parameter values for an estimator (KNNBasic).

def run_grid_search(data):
    # Parameter Grid
    param_grid = {
        "k": [20, 40],
        "min_k": [1, 3],
        "sim_options": {
            "name": ["cosine", "pearson"],
            "user_based": [False] # Testing Item-Based here
        }
    }

    # Grid Search Initialization
    gs = GridSearchCV(
        KNNBasic,
        param_grid,
        measures=["rmse", "mae"],
        cv=3,
        n_jobs=1
    )

    # Execute Search
    gs.fit(data)

    return gs.best_score, gs.best_params
```

```
Grid Search Result
Best RMSE: 0.7280
Best MAE : 0.3704
Best Parameters: {
    'k': 20,
    'min_k': 1,
    'sim_options': {'name': 'cosine', 'user_based': False}
}
```

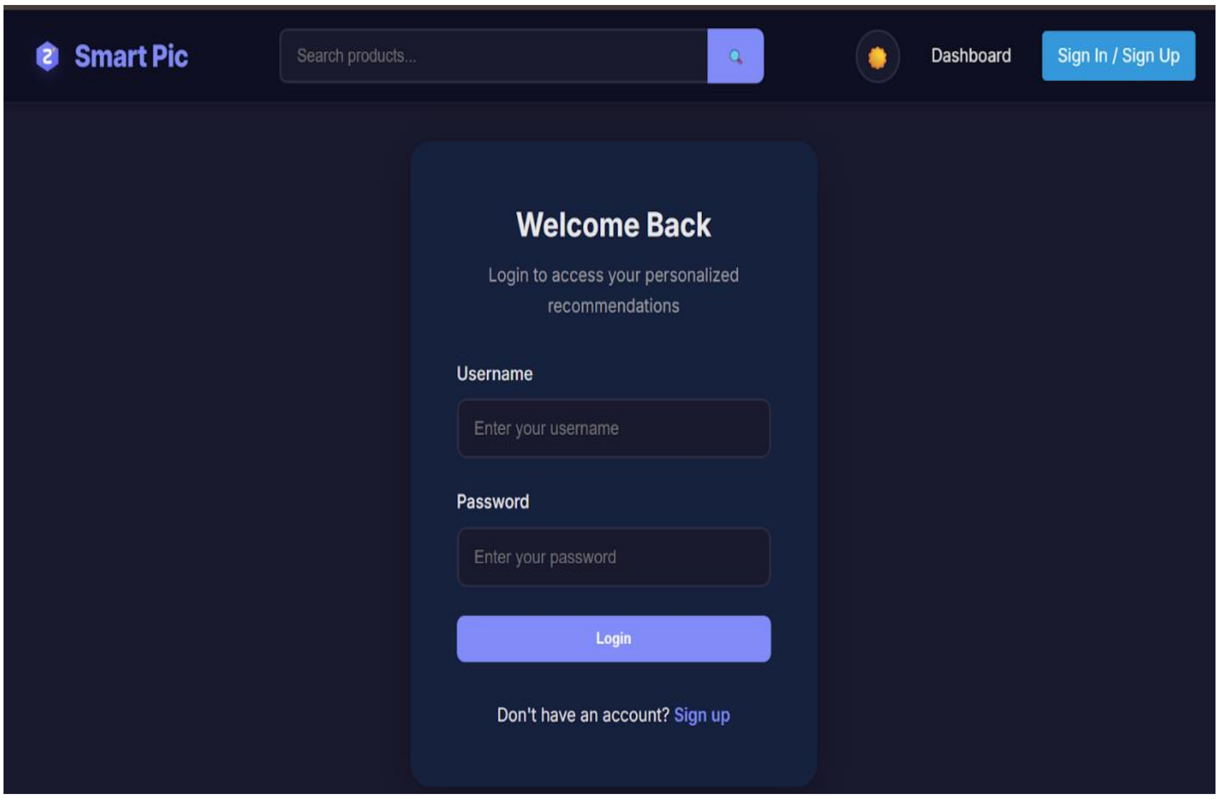
API Integration

```
import os
import sys
from flask import Flask, render_template, request, redirect, url_for, flash, session, jsonify
from dotenv import load_dotenv

# Add project root to path
PROJECT_ROOT = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))
sys.path.insert(0, PROJECT_ROOT)

from config import Config
from src.components.database import db, Product, User, init_db
from src.inference.hybrid_recommender import HybridRecommender

load_dotenv()

app = Flask(__name__)
app.config.from_object(Config)
app.config['SECRET_KEY'] = os.getenv('SECRET_KEY', 'your-secret-key')

# Initialize database
init_db(app)

# Initialize Model
hybrid_recommender = None
def get_hybrid_recommender():
    global hybrid_recommender
    if hybrid_recommender is None:
        try:
            hybrid_recommender = HybridRecommender(alpha=0.5)
        except Exception as e:
            print(f"Error loading model: {e}")
    return hybrid_recommender

# ----------------------------
# MAIN ROUTES
# ----------------------------
```

```
@app.route('/')
def index():
    popular = Product.query.order_by(Product.ratings.desc()).limit(8).all()
    # Serialize for template
    popular_data = [{'ProductId': p.product_id, 'name': p.name, 'price': p.discount_price, 'image':
p.image} for p in popular]

    return render_template('index.html', popular=popular_data)

@app.route('/dashboard')
def dashboard():
    # Simple Dashboard Metrics
    total_products = Product.query.count()
    avg_rating = db.session.query(db.func.avg(Product.ratings)).scalar() or 0

    popular = Product.query.order_by(Product.no_of_ratings.desc()).limit(5).all()
    top_rated = Product.query.order_by(Product.ratings.desc()).limit(5).all()

    kpi = {'total_products': total_products, 'avg_rating': round(avg_rating, 2)}

    return render_template('dashboard.html',
                           popular=popular,
                           top_rated=top_rated,
                           kpi=kpi,
                           username=session.get('username'))

@app.route('/search')
def search():
    query = request.args.get('q', '')
    results = []

    if query:
        # Basic search
        results = Product.query.filter(Product.name.ilike(f'%{query}%')).limit(50).all()

    results_list = [{
        'ProductId': p.product_id,
        'name': p.name,
        'price': p.discount_price,
        'ratings': p.ratings,
        'image': p.image
    } for p in results]

    return render_template('search.html', results=results_list, query=query)

@app.route('/product/<product_id>')
def product_detail(product_id):
    product = Product.query.filter_by(product_id=product_id).first()
    if not product:
        return redirect(url_for('index'))

    # Get Recommendations
    hr = get_hybrid_recommender()
    recommendations = []

    if hr:
        recommendations = hr.get_hybrid_recommendations(product_id,
user_id=session.get('user_id'), n=6)

    # Fallback if model fails or returns empty
    if not recommendations:
        similar = Product.query.filter(Product.main_category ==
product.main_category).limit(6).all()
        recommendations = [{
            'ProductId': p.product_id,
            'name': p.name,
            'image': p.image,
            'price': p.discount_price
        } for p in similar]

    return render_template('product.html', product=product, similar=recommendations)
# AUTH ROUTES
# ----------------------------
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')

        user = User.query.filter_by(username=username).first()
        if user and user.check_password(password):
            session['user_id'] = user.id
            session['username'] = user.username
            return redirect(url_for('dashboard'))

        flash('Invalid credentials')
    return render_template('login.html')

@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form.get('username')
        email = request.form.get('email')
        password = request.form.get('password')

        user = User(username=username, email=email)
        user.set_password(password)
        db.session.add(user)
        db.session.commit()

        return redirect(url_for('login'))
    return render_template('signup.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('index'))
if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

10 EXECUTION SCREENSHOTS



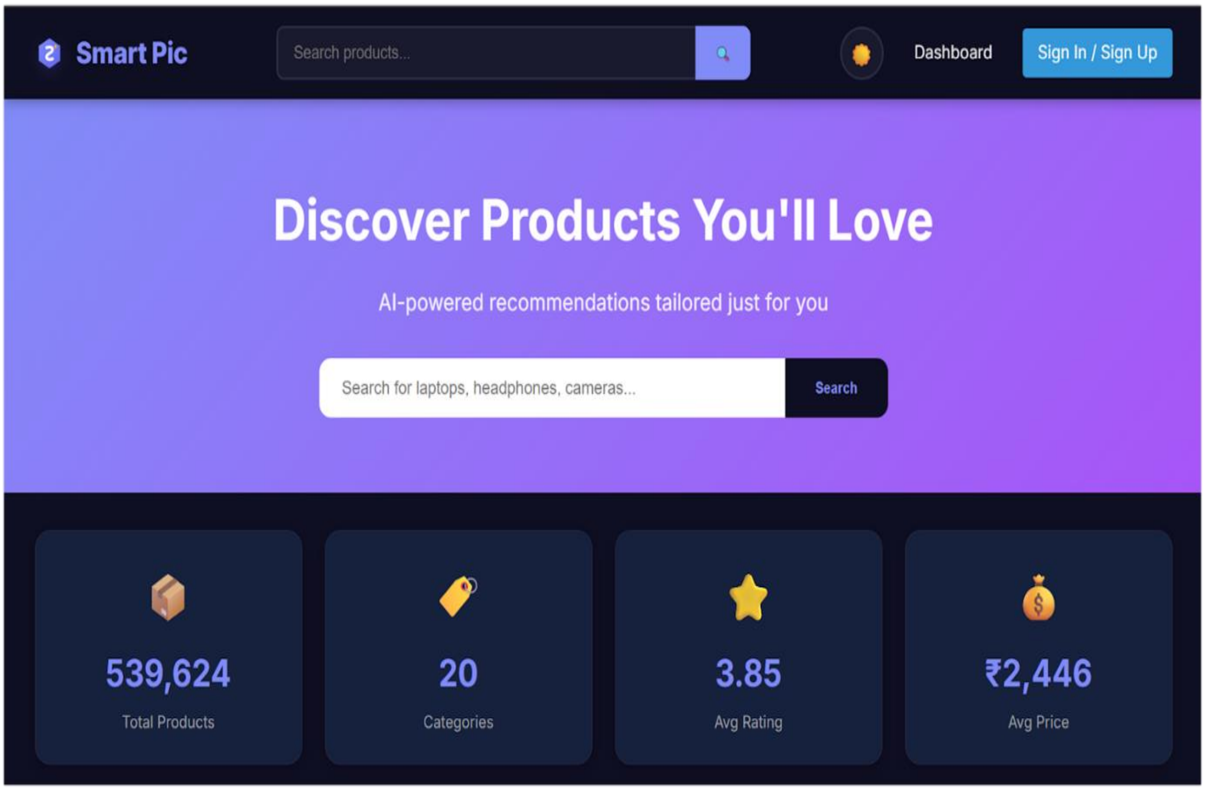
Welcome Back

Login to access your personalized recommendations

Username: Enter your username

Password: Enter your password

Don't have an account? Sign up



Discover Products You'll Love

AI-powered recommendations tailored just for you

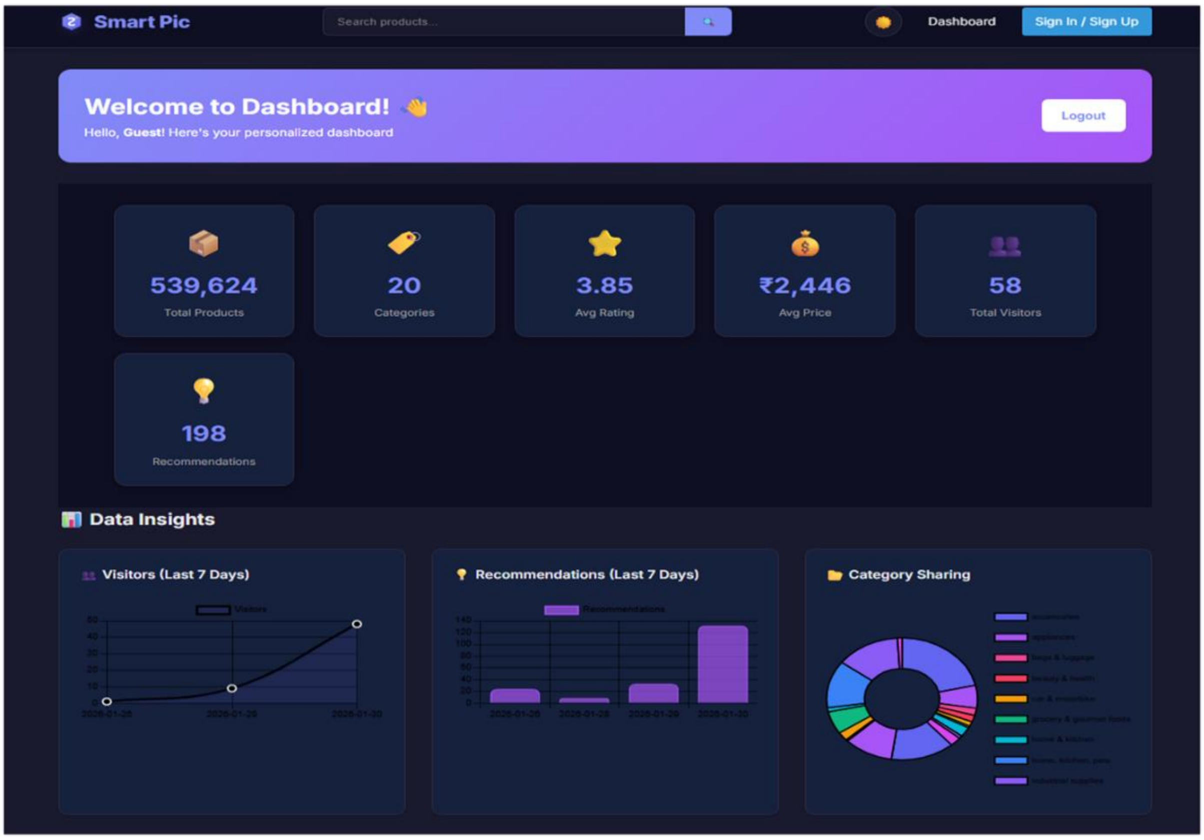
Search for laptops, headphones, cameras...

539,624 Total Products

20 Categories

3.85 Avg Rating

₹2,446 Avg Price

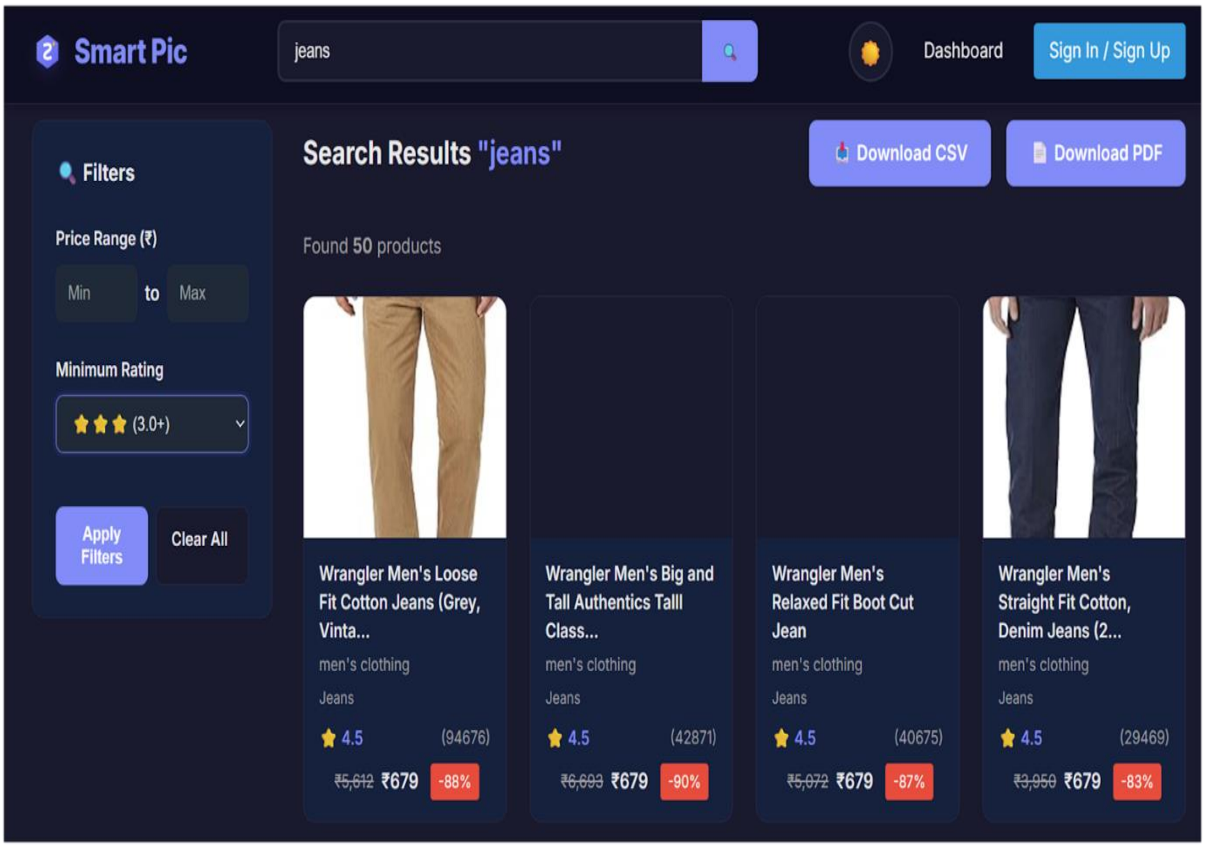


Welcome to Dashboard! 👋

Hello, **Guest!** Here's your personalized dashboard

539,624	20	3.85	₹2,446	58
Total Products	Categories	Avg Rating	Avg Price	Total Visitors
198				
Recommendations				

📊 Data Insights



Search Results "jeans"

Found **50** products

Wrangler Men's Loose Fit Cotton Jeans (Grey, Vinta...	Wrangler Men's Big and Tall Authentics Talll Class...	Wrangler Men's Relaxed Fit Boot Cut Jean	Wrangler Men's Straight Fit Cotton, Denim Jeans (2...
men's clothing	men's clothing	men's clothing	men's clothing
Jeans	Jeans	Jeans	Jeans
★ 4.5 (94676)	★ 4.5 (42871)	★ 4.5 (40675)	★ 4.5 (29469)
~~₹5,612~~ ₹679 -88%	~~₹6,693~~ ₹679 -90%	~~₹5,072~~ ₹679 -87%	~~₹3,950~~ ₹679 -83%

11 LIMITATIONS

1. DATA LIMITATIONS

- Data Sparsity: The user-item interaction matrix is highly sparse (users rate only a small fraction of products), making it difficult for collaborative filtering to find sufficient neighbors.
- Cold Start Problem: Although the Hybrid approach helps, the system still struggles with completely new users (no history) and new products (no ratings).
- Dataset Sampling: To manage computational load, the model is trained on a sampled dataset (e.g., 25k interactions) rather than the entire Amazon catalog, potentially missing niche patterns.

2. MODEL & ALGORITHMIC LIMITATIONS

- Offline Learning (No Real-Time Updates): The model uses batch learning. It does not update instantly with user interactions; retraining is required to incorporate new data.
- Scalability of KNN: The K-Nearest Neighbors algorithms have O(N^2) complexity. As user/item counts grow, calculating similarity matrices becomes computationally expensive and memory-intensive.
- Grey Sheep Problem: Users with inconsistent preferences that don't align with any specific group are difficult to classify, leading to lower prediction accuracy for them.

3. SYSTEM & FEATURE LIMITATIONS

- Reliance on Explicit Feedback: The system primarily uses explicit star ratings (1-5). It does not fully leverage implicit feedback signals like clicks, page views, or time spent, which are often more abundant.
- Context Unawareness: The recommendations are context-free. The system does not account for time of day, seasonality, or user intent (e.g., browsing for self vs. browsing for a gift).
- Single Domain: The model is specialized for Amazon product data and cannot generalize to cross-domain recommendations without significant retraining.

4. EVALUATION LIMITATIONS

- Offline Metrics Focus: Performance is measured using RMSE and MAE. While these measure rating accuracy, they do not guarantee user satisfaction or engagement (Click-Through Rate, Conversion Rate).
- Serendipity vs. Accuracy: The focus on minimizing error (RMSE) can sometimes lead to "safe," popular recommendations, reducing the novelty and serendipity of the results.

12 FUTURE SCOPE

1. DEEP LEARNING INTEGRATION

- Neural Collaborative Filtering (NCF): Replacing the current Matrix Factorization and KNN models with Deep Neural Networks to capture complex, non-linear patterns in user interaction data.
- Autoencoders: Implementing Autoencoders for better feature extraction from sparse matrices, potentially improving accuracy for users with very few ratings.

2. REAL-TIME LEARNING

- Online Learning Pipelines: Moving from "Batch Processing" (retraining the whole model offline) to "Online Learning" where the model updates incrementally as soon as a user rates a product or makes a purchase.
- Session-Based Recommendations: Utilizing Recurrent Neural Networks (RNNs) or Transformers (e.g., SASRec) to recommend products based on the user's *current* browsing session rather than just long-term history.

3. DIVERSE DATA SOURCES

- Implicit Feedback: Integrating non-explicit signals such as "Add to Cart," "Time Spent on Page," and "Click-Through Rate" (CTR) to build a richer user profile.
- Image & Text Analysis: Using Convolutional Neural Networks (CNNs) to analyze product images and BERT-based models for review sentiment analysis to enhance Content-Based Filtering.

4. BUSINESS LOGIC & EXPLAINABILITY

- Explainable AI (XAI): Adding a "Why this was recommended?" feature to show users the logic behind suggestions (e.g., "Because you liked [Product X]..."), which increases user trust.
- Multi-Objective Optimization: Optimizing not just for accuracy (RMSE), but also for business metrics like Profit Margin, Diversity of Recommendations, and Novelty.

5. INFRASTRUCTURE SCALABILITY

- Distributed Computing: Migrating the training pipeline to distributed systems like Apache Spark (PySpark) to handle millions of users and products efficiently.
- Cloud Deployment: Deploying the recommendation engine as a scalable microservice on cloud platforms (AWS/GCP/Azure) using Docker and Kubernetes.

13 APPLICATIONS

- **E-CommercePlatforms:**
  Helps online shopping websites recommend relevant products to users based on their interests.
- **OnlineRetailStores:**
  Improves product discovery and increases sales by showing personalized suggestions.
- **DigitalMarketplaces:**
  Supports large product catalogs by reducing information overload for users.
- **CustomerPersonalizationSystems:**
  Enhances user experience through customized content and product recommendations.
- **StartupE-CommerceApplications:**
  Provides a cost-effective recommendation solution for small and medium online businesses.
- **ProductMarketingandPromotion:**
  Assists in targeted product promotion based on user behavior and preferences.
- **DecisionSupportSystems:**
  Helps businesses analyze customer behavior and make data-driven decisions.

14 SYSTEM TESTING

1. Unit Testing

Unit testing was performed to validate individual modules of the system independently. Each core functionality was tested to ensure correct behavior.

Tested Components:

- Userauthentication(register,login,logout)
- Recommendationgenerationfunctions
- Databasequeryoperations
- TF-IDFfeatureextractionmodule

Purpose:

To ensure that each module works correctly in isolation.

2. Integration Testing

Integration testing was conducted to verify the interaction between different system modules.

Tested Integrations:

- WebinterfaceandFlaskbackend
- BackendandMySQLdatabase
- Backendandrecommendationengine
- Recommendationengineandtrainedmodels

Purpose:

To ensure smooth data flow between interconnected modules.

3. Functional Testing

Functional testing validated whether the system met all specified functional requirements.

Tested Functions

- Userregistrationandlogin
- Productbrowsingandsearch
- Personalizedrecommendationdisplay
- Ratingandinteractionhandling

Purpose:

To confirm that the system behaves as expected from a user's perspective.

4. Performance Testing

Performance testing evaluated the responsiveness and efficiency of the system under normal operating conditions.

Tested Metrics:

- Recommendationresponsetime
- Databasequeryexecutiontime

- Modelinferencespeed

Purpose:

To ensure the system provides real-time recommendations without delay.

5. Data Validation Testing

This testing verified the correctness and consistency of data stored and retrieved from the database.

Tested Aspects:

- Accuracyofstoreduserdata
- Correctmappingofuser–itemratings
- Consistencyofproductandreviewdata

Purpose:

To prevent incorrect or inconsistent recommendations due to data errors.

15 CONCLUSION

This project presents an automated and user-friendly solution to the challenge of personalized product discovery in e-commerce platforms. The system integrates a **hybrid recommendation architecture** that combines collaborative filtering and content-based filtering to deliver accurate and relevant product suggestions. Implemented as a Flask-based web application, the system also includes an interactive analytics dashboard and an advanced search interface to support both user experience and business insights.

The hybrid design ensures robust performance in data-sparse scenarios and effectively addresses the cold-start problem by utilizing both user interaction history and product attributes. As user interaction data increases, the system supports continuous improvement through periodic model retraining. Experimental evaluation demonstrates strong predictive performance, with the optimized KNN model achieving an RMSE of approximately **0.61**, confirming its effectiveness across diverse product categories.

In conclusion, the system demonstrates strong potential for real-world deployment due to its scalability, accuracy, and adaptability. By combining machine learning techniques with intuitive web accessibility and analytics, the project provides a practical solution for modern digital retail personalization.

REFERENCES

[1] Gulnara, B., Guldana, S., & Yerassyl, A. (2024). Application of recommended systems for e-commerce. Procedia Computer Science, 231, 329–334.

[2] Upadhyay, K., Kumar, T., Neelima, N., Sachdeva, T., Patil, M., & Muzafar, S. (2025). Personalized news recommender system using hybrid collaborative filtering and content-aware deep models. In Proceedings of the 2025 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC) (pp. 1–7). IEEE.

[3] Kumar, P., & Thakur, R. S. (2018). Recommendation system techniques and related issues: A survey. International Journal of Information Technology, 10(4), 495–501.

[4] Sreelakshmi, A., Padhy, N., & Senapaty, M. K. (2024). Strategic retail decision-making: A hybrid Apriori–FP Growth algorithm for efficient association rule discovery. In National Conference on Intelligent Systems, IoT, and Wireless Communication for the Society (pp. 117–130). Springer Nature Singapore.

[5] Singh, J., Rani, S., Devi, S., & Kaur, J. (2025). A systematic study on recommendation system for e-commerce applications. In Proceedings of the Seventh International Conference on Computational Intelligence and Communication Technologies (CCICT) (pp. 221–226). IEEE.

[6] Shankar, A., Perumal, P., Subramanian, M., Ramu, N., Natesan, D., Kulkarni, V. R., & Stephan, T. (2024). An intelligent recommendation system in e-commerce using ensemble learning. Multimedia Tools and Applications, 83(16), 48521–48537.

[7] Tomashevskyy, O., Basystiuk, O., & Kryvinska, N. (2024). Machine learning for business process automation. Springer.

[8] Gulzar, A., et al. (2023). Ordered clustering-based algorithm for e-commerce recommendation systems. Journal/Conference details.

[9] Nozari, R. B., Divsalar, M., Abkenar, S. A., Amiri, M. F., & Divsalar, A. (2024). A novel behavior-based recommendation system for e-commerce. arXiv preprint arXiv:2403.18536.

[10] Gupta, C. P., & Kumar, V. R. (2024). Recommendation system: A transformative artificial intelligence tool for e-commerce. In Proceedings of the 7th International Conference on Informatics and Computational Sciences (ICICoS) (pp. 60–65). IEEE.