MULTI-CONTEXT-AWARE MULTI-CRITERIA RECOMMENDER SYSTEM FOR E-COMMERCE

A PROJECT REPORT

Submitted by

MALINI. S

BHAVADHARANI. A

LAVANYA. R

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SARANATHAN COLLEGE OF ENGINEERING

(An Autonomous Institution)

Tiruchirappali 620 012

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2025

SARANATHAN COLLEGE OF ENGINEERING

(An Autonomous Institution)

Tiruchirappali 620 012

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **MULTI-CONTEXT-AWARE MULTI-CRITERIA RECOMMENDER SYSTEM FOR E-COMMERCE** is the Bonafide work of **BHAVADHARANI. A, LAVANYA. R, MALINI. S** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. S. Ravimaran M.E., Ph.D.,	**Mrs. Preethy Janet M.E.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Assistant Professor
Artificial Intelligence and Data Science	Artificial Intelligence and Data Science
Saranathan College of Engineering	Saranathan College of Engineering
Tiruchirappalli – 620012	Tiruchirappalli – 620012

Submitted for the project viva-voce examination held on _____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

The word "thanks," though apparently short, carries deep eloquence that is enhanced when expressed with true heartfelt sincerity. At this point, we would like to extend our heartfelt thanks to all the people who helped in the process of completing this important assignment.

Firstly, we want to thank the "Almighty" for granting us the "Will and Determination" to doggedly strive for our aim.

We wish to express our gratitude to our college Secretary **Shri. S. Ravindran** and our Principal **Dr. D. Valavan M.Tech., Ph. D.,** for their support in the successful completion of the project work.

With a sincere feeling of gratitude, we convey our deepest gratitude to our Head **Dr. S. Ravimaran M.E., Ph.D.,** Department of Artificial Intelligence and Data Science, for his inspiration to finish the work successfully.

We express our sincere thanks to our project guide **Mrs. Preethy Janet M.E.**, and the project coordinator **Mrs. A. Sridevi M.E.,** members of the project review committee, and all faculty members of the Department of Artificial Intelligence and Data Science for their guidance, useful support, and encouragement during the course of our project.

We would like to extend our heartfelt thanks to our loving parents, siblings, and friends for their ongoing support.

ABSTRACT

In today's digital landscape, recommender systems play a crucial role in enhancing user experience by suggesting relevant products, services, and content based on user preferences. However, traditional recommendation models often fall short as they primarily rely on single-criteria ratings and do not take contextual factors into account. These limitations result in less personalized and sometimes inaccurate recommendations. To address this challenge, our study proposes a Multi-Context-Aware Multi-Criteria Recommender System (MCoMCRS) that utilizes deep learning techniques to generate more precise and user-centric recommendations. The proposed model is designed to incorporate multiple contextual factors, such as category, and user rating, alongside multiple evaluation criteria like actual price, discounted price, and discounted percentage. By leveraging a Deep Neural Network (DNN), our approach processes these parameters simultaneously, offering a comprehensive and dynamic recommendation system. To validate the effectiveness of our approach, we utilize an extensive dataset comprising Amazon product reviews. The dataset undergoes rigorous preprocessing, including handling missing values, normalizing numerical attributes, and encoding categorical variables. The performance of the model is assessed using standard evaluation metrics such as Mean Absolute Error (MAE). Experimental results demonstrate that our system significantly outperforms traditional recommendation models by delivering more accurate and context-aware recommendations. Additionally, our system addresses key challenges prevalent in recommendation models, such as the cold-start problem and data sparsity, which often hinder the efficiency of existing techniques.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iv**
	LIST OF TABLES	**x**
	LIST OF FIGURES	**x**
	LIST OF ABBREVIATION	**x**
1.	**INTRODUCTION**	**1**
1.1.	CONCEPTS OF RECOMMENDER SYSTEMS	1
1.1.1	The Rise of Recommender Systems	1
1.1.2	Limitations of Traditional Approaches	1
1.1.3	Evolution towards Context-Aware and Multi-Criteria Models	2
1.2.	PROBLEM STATEMENT	2
1.2.1	The Need for Personalization Beyond Ratings	2
1.2.2	The Gap in Contextual Awareness	3
1.2.3	Combined Challenge: Multi-Criteria + Context	3
1.3.	OBJECTIVES OF THE PROJECT	4
1.3.1	Technical Objectives	4
1.3.2	Evaluation Goals	4
1.3.3	Application Perspective	4
1.4.	SCOPE OF THE PROJECT	5
1.4.1	Inclusions	5
1.4.2	Exclusions	5
1.4.3	Research Focus	5
1.5.	STRUCTURE OF THE REPORT	6
1.5.1	Chapter Overview	6
1.5.2	Supplementary Sections	6
1.6.	THE EVOLUTION OF RECOMMENDER SYSTEMS: A HISTORICAL PERSPECTIVE	6

1.7. PERSONALIZATION VS GENERALIZATION: STRIKING A BALANCE 7

1.8. CHALLENGES IN E-COMMERCE RECOMMENDATION 7

2. LITERATURE REVIEW 9

2.1. OVERVIEW OF RECOMMENDER SYSTEMS 9

2.1.1 Introduction to Recommender Systems 9

2.1.2 Classification of Recommender Systems 9

2.2. MULTI-CRITERIA AND CONTEXT-AWARE APPROACHES 10

2.2.1 Concept and Significance 10

2.2.2 Techniques Used in MCRS 11

2.2.3 Challenges in Multi-Criteria Recommendation 11

2.3. CONTEXT-AWARE RECOMMENDER SYSTEMS (CARS) 12

2.3.1 Role of Context in Recommendation 12

2.3.2 Context Modeling Techniques 12

2.3.3 Benefits and Limitations 13

2.4. DEEP LEARNING IN RECOMMENDER SYSTEMS 13

2.4.1 Emergence of Deep Learning 13

2.4.2 DNN-Based Recommender Models 13

2.4.3 Advantages Over Traditional Models 14

2.5. LIMITATIONS OF EXISTING SYSTEMS 14

2.5.1 Data Sparsity and Cold Start 14

2.5.2 Lack of Integration Between Multi-Criteria

and Context 14

2.5.3 Interpretability and User Trust 15

2.6. RESEARCH GAP AND MOTIVATION 15

2.7. MATRIX FACTORIZATION AND LATENT FACTOR MODELS 15

2.8. FACTORIZATION MACHINES (FM) 16

2.9. WIDE AND DEEP LEARNING MODELS 17

2.10. COMPARATIVE SUMMARY OF RECOMMENDATION MODELS 18

3. METHODOLOGY 19

3.1. SYSTEM ARCHITECTURE 19

3.1.1 Overview of System Design 19

3.1.2 System Components 19

3.2. HARDWARE AND SOFTWARE REQUIREMENTS 20

3.2.1 Software Stack 20

3.3. DATA COLLECTION AND PREPROCESSING 21

3.3.1 Data Sources 21

3.3.2 Preprocessing Techniques 21

3.4. ALGORITHM IMPLEMENTATION 22

3.4.1 Model Approach 22

3.4.2 Model Architecture 22

3.5. EVALUATION METRICS 24

3.5.1 Mean Absolute Error (MAE) 24

3.6. INTERFACE AND INTEGRATION 25

3.6.1 User Interface 25

3.6.2 Model Deployment 26

3.7. RECOMMENDATION ALGORITHM LOGIC 26

4. RESULTS AND DISCUSSION 28

4.1. PERFORMANCE ANALYSIS 28

4.1.1 Training and Validation Curves 28

4.1.2 Interpretation of Learning Trends 30

4.2. COMPARATIVE STUDY 30

4.3. USER EXPERIENCE EVALUATION 31

4.3.1 Interface Features 31

4.3.2 Feedback and Observations 31

4.4. IMPACT OF MULTI-CRITERIA AND CONTEXT INTEGRATION 32

4.5 TOP-N RECOMMENDATION STRATEGY 32

4.6 OVERALL DISCUSSION 33

5. CONCLUSION AND FUTURE WORK 34

5.1. SUMMARY OF PROJECT CONTRIBUTIONS 34

5.2. KEY ACHIEVEMENTS 34

5.3. LIMITATIONS AND CHALLENGES 35

5.3.1 Limited Real-Time Contextual Dimensions 35

5.3.2 Static Learning 35

5.3.3 Limited Personal User Profiles 35

5.3.4 Model Interpretability 35

5.3.5 Deployment on Cloud-Based Platforms 36

5.4. FUTURE ENHANCEMENTS 36

5.4.1 Real-Time Adaptive Learning 36

5.4.2 Incorporating Additional Contextual Layers 36

5.4.3 Hybrid Deep Learning Models 37

5.4.4 Cross-Domain Recommendations 37

5.4.5 Fairness and Bias Mitigation 37

5.4.6 Cloud-Based Scalability and Deployment 37

5.5. CONCLUDING REMARKS 38

6. APPENDICES 39

A1. SOURSE CODE 39

A1.1 Main Code for deployment 39

A1.2 Training Code 44

A2. OUTPUT SCREENSHOTS 47

REFERENCES 48

WEBSITE 50

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
1.	Comparative Summary of Recommendation Models	18
2.	Software Stack table	20
3.	Performance Comparison of Different Recommender Models	25
4.	Comparison of Proposed System with Existing Models	30

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.	Workflow of the Multi-Context-Aware Multi-Criteria Recommender System	24
2.	Model Training vs Validation Loss Graph	29
3.	Training and Validation Loss/Metrics Progression over Epochs	29
4.	Product Rating Predictor Interface	47
5.	Prediction Result Interface (Predicted Rating, Product Recommendations)	47

LIST OF ABBREVIATION

ABBREVIATION	DESCRIPTION
MCoMCRS	Multi-Context-Aware Multi-Criteria Recommender System
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
MCRS	Multi-Criteria Recommender System

MCARS	Multi-Context-Aware Recommender System
CF	Collaborative Filtering
CBF	Content-Based Filtering
AI	Artificial Intelligence
DL	Deep Learning
CARS	Context-aware recommendation systems

CHAPTER 1

INTRODUCTION

1.1 CONCEPTS OF RECOMMENDER SYSTEMS

1.1.1 The Rise of Recommender Systems

In the era of digital transformation, the amount of information available to users has grown exponentially. This abundance of options, while beneficial, often leads to information overload. Recommender systems have emerged as an effective solution to this problem by helping users discover relevant items such as products, services, or content tailored to their preferences. These systems are now integral to platforms like Amazon, Netflix, and Spotify, driving user engagement and satisfaction.

In e-commerce, specifically, recommender systems enhance the customer journey by analysing user behaviour patterns — including browsing history, purchase records, and product ratings — to generate personalized product suggestions. Their ability to convert passive browsing into active purchasing has made them vital for increasing conversion rates and overall revenue in digital marketplaces.

1.1.2 Limitations of Traditional Approaches

Despite the widespread adoption of recommendation systems, traditional models have notable limitations. The two most common paradigms are:

- **Collaborative Filtering (CF)**: This method identifies similarities between users or items based on historical interactions. However, it suffers from issues such as:
    - **Cold Start**: Inability to make recommendations for new users or items due to lack of data.
    - **Data Sparsity**: User-item interaction matrices are often sparse, making it difficult to find meaningful patterns.
- **Content-Based Filtering (CBF)**: This technique recommends items with similar attributes to those previously liked by the user. However, it tends to:

- **Over-specialize**: Recommending only similar items and failing to introduce variety.
- **Depend on item metadata**: Recommendations are only as good as the descriptive features provided.

Both methods largely ignore the dynamic nature of user preferences and fail to consider real-time contextual factors, reducing the relevance of recommendations in certain situations.

1.1.3 Evolution towards Context-Aware and Multi-Criteria Models

To overcome the drawbacks of conventional systems, modern recommender systems are evolving to incorporate **contextual information** and **multi-criteria evaluations**. These advanced models aim to provide more granular and situationally aware recommendations.

- **Context-Aware Recommender Systems (CARS)** enhance predictions by including external or situational factors such as time, location, device type, or user mood. For instance, a user may prefer different types of products during weekends compared to weekdays.
- **Multi-Criteria Recommender Systems (MCRS)** address the limitation of single-rating systems by considering multiple factors like product quality, price, brand reputation, and user satisfaction. This allows the system to better understand what truly influences a user's decision.

The integration of these two models, although promising, introduces new challenges in terms of model complexity, data handling, and performance optimization. Hence, our study proposes a **deep learning-based approach** that combines both paradigms to achieve accurate and personalized recommendations.

1.2 PROBLEM STATEMENT

1.2.1 The Need for Personalization Beyond Ratings

In most traditional systems, a single overall rating is used to predict user preference. However, users usually evaluate products across multiple dimensions — for example, a smartphone may

be rated highly for performance but poorly for battery life. Relying on a single rating fails to capture these nuances, resulting in oversimplified and sometimes misleading recommendations.

A truly personalized system must consider the **multi-dimensional aspects** of user preferences to provide relevant suggestions that align more closely with individual priorities and expectations.

1.2.2 The Gap in Contextual Awareness

User preferences are not static. They evolve over time and can vary depending on **external contextual factors** such as:

- Time of day (e.g., morning vs evening preferences)
- Location (e.g., home vs office)
- Season (e.g., summer vs winter purchases)
- Occasion (e.g., regular shopping vs holiday season)

Most existing systems ignore these aspects, offering the same recommendations regardless of context. This lack of **situational awareness** limits the effectiveness and accuracy of suggestions.

1.2.3 Combined Challenge: Multi-Criteria + Context

While some research has explored MCRS or CARS independently, few systems have successfully **combined both** into a unified architecture. Integrating multi-criteria evaluations with dynamic contextual awareness introduces several challenges:

- **Model Complexity**: Handling high-dimensional data increases computational load.
- **Data Integration**: Merging different data types (e.g., numerical ratings, categorical contexts) requires careful feature engineering.
- **Scalability**: The system must be efficient enough to work in real-time environments like e-commerce platforms.

This project addresses this combined challenge by leveraging deep learning techniques that can model complex, nonlinear interactions between multiple features.

1.3 OBJECTIVES OF THE PROJECT

1.3.1 Technical Objectives

The primary technical goals of this project are:

- To design and implement a recommender system based on Deep Neural Networks (DNNs).
- To include both **multi-criteria ratings** (e.g., actual price, discount, brand rating) and **contextual features** (e.g., category, user review behavior).
- To train and validate the model using real-world e-commerce data.
- To evaluate performance using industry-standard metrics such as **MAE** and **RMSE**.

1.3.2 Evaluation Goals

- To assess model performance using metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE).
- To compare the proposed system against traditional and modern recommendation techniques.
- Analyze the effectiveness of context and criteria integration through quantitative (metrics) and qualitative (user experience) assessments.

1.3.3 Application Perspective

- Develop a **prototype GUI** to demonstrate the model in action.
- Simulate real-world usage scenarios within an e-commerce setting.
- Showcase how intelligent recommendation systems can improve **user satisfaction**, **decision-making**, and **platform engagement**.

1.4 SCOPE OF THE PROJECT

1.4.1 Inclusions

- Development of a Multi-Context-Aware Multi-Criteria Recommender System (MCoMCRS).
- Implementation of data collection, preprocessing, training, and validation pipelines.
- Visualization of recommendation outcomes and prediction scores in a user interface mock-up.

1.4.2 Exclusions

- Real-time deployment or live API integration with commercial e-commerce platforms.
- Integration with mobile applications or user accounts.
- Use of personal or sensitive user data – the system is developed on publicly available datasets only.

1.4.3 Research Focus

- Emphasis is placed on understanding how deep learning can bridge the gap between context and multi-criteria evaluation for more meaningful recommendations.
- The system is designed to be extensible, allowing future enhancements like inclusion of user emotion, visual cues, or temporal sequences using CNNs or RNNs.
- Exploring how DNNs can efficiently combine multi-criteria and context features.
- Investigating the **impact of each feature group** (criteria vs context) on recommendation performance.
- Laying the groundwork for **adaptive learning models** in the future.

1.5 STRUCTURE OF THE REPORT

1.5.1 Chapter Overview

- Chapter 1 introduces the motivation, background, and goals of the project.
- Chapter 2 explores prior research and systems through a comprehensive literature review.
- Chapter 3 details the methodology including architecture, algorithms, and data processing.
- Chapter 4 presents the experimental results, visualizations, and discussions.
- Chapter 5 concludes the study and suggests future research directions.

1.5.2 Supplementary Sections

- List of Tables, Figures, and Abbreviations provide reference materials to ease navigation.
- References cite the key academic and technical works consulted during the project.

1.6 THE EVOLUTION OF RECOMMENDER SYSTEMS: A HISTORICAL PERSPECTIVE

The field of recommender systems has evolved dramatically over the last two decades. What began as basic item-similarity matching in the early 2000s has grown into a complex ecosystem powered by artificial intelligence and big data.

- **First-generation systems** were largely rule-based and used static heuristics. These were effective in small domains but lacked scalability.
- The **advent of Collaborative Filtering (CF)** brought data-driven intelligence, allowing systems to automatically learn from user behaviour.
- **Content-Based Filtering (CBF)** improved personalization by factoring in product features and user preferences.

- By the late 2010s, **hybrid systems** emerged, combining CF and CBF to overcome their individual limitations.
- Today, **deep learning-based systems** dominate research and industrial deployments, capable of modeling highly nonlinear interactions, processing images, text, and even voice input.

This historical trajectory shows a clear trend toward more personalized, dynamic, and intelligent recommendation systems.

1.7 PERSONALIZATION VS GENERALIZATION: STRIKING A BALANCE

A fundamental challenge in recommender systems is balancing **personalization** and **generalization**.

- **Over-personalization** may cause filter bubbles, where users are shown only a narrow range of options.
- **Under-personalization** results in irrelevant suggestions, leading to user frustration.

The goal is to:

- Accurately capture **individual preferences.**
- Introduce **serendipity and diversity** in recommendations.
- Learn from **similar users** while respecting uniqueness.

This model addresses this by considering **multiple criteria (price, discount, rating)** and **context (category)**, which naturally adds flexibility and personalization without being overly restrictive.

1.8 CHALLENGES IN E-COMMERCE RECOMMENDATION

Designing an effective recommender system for e-commerce faces several practical hurdles:

- **Highly dynamic inventory**: New products added daily, with changing availability.
- **Diverse user base**: Different regions, age groups, and behaviours.

- **Varying product types**: Electronics, apparel, groceries—all require different strategies.
- **Trust and transparency**: Users expect explainable and fair recommendations.

To solve these issues, models must be:

- **Data-driven and scalable**
- **Context-aware**
- **Able to generalize across categories**

These challenges are exactly what the proposed MCoMCRS aims to address.

CHAPTER 2

LITERATURE REVIEW

2.1 OVERVIEW OF RECOMMENDER SYSTEMS

2.1.1 Introduction to Recommender Systems

Recommender systems are software tools and techniques that provide suggestions for items to be of use to a user. These systems are prevalent across domains such as e-commerce, digital entertainment, social media, and online learning platforms. Their primary objective is to filter vast volumes of information and deliver content that aligns with user interests and needs.

They play a critical role in helping users navigate platforms like Amazon, Netflix, and YouTube by offering personalized recommendations based on historical behaviour, user preferences, and product attributes. The effectiveness of a recommender system significantly influences user satisfaction, engagement, and platform profitability.

2.1.2 Classification of Recommender Systems

Recommender systems can broadly be classified into three categories:

a) Collaborative Filtering (CF)

This method relies on the assumption that users with similar behavior in the past will continue to have similar preferences. CF is further divided into:

- **User-based CF**: Finds users with similar interests.
- **Item-based CF**: Finds items that receive similar ratings.

  While CF is effective in identifying latent patterns, it suffers from two primary issues:

- **Cold Start**: New users or items lack sufficient interaction history.
- **Data Sparsity**: Large user-item matrices are often incomplete or sparse, leading to poor recommendations.

b) Content-Based Filtering (CBF)

CBF recommends items similar to those a user has previously liked, based on item features like category, brand, or price. It requires detailed product metadata and user profiles. While it avoids the cold start problem to some extent, it can lead to:

- **Overfitting** to past preferences (low novelty).
- **Limited discovery** of diverse or unexpected items.

c) Hybrid Approaches

Hybrid models combine the strengths of CF and CBF to improve recommendation performance. These systems use various strategies such as:

- Weighted averaging of predictions.
- Switching between algorithms based on context.
- Feature-level combination of collaborative and content-based features.

While hybrids offer better performance, they also introduce architectural complexity and require careful tuning.

2.2 MULTI-CRITERIA AND CONTEXT-AWARE APPROACHES

2.2.1 Concept and Significance

Traditional recommendation models rely heavily on single overall ratings to predict user preferences. However, this one-dimensional representation often fails to capture the full scope of user sentiment. For instance, a user may rate a product highly due to its price but may be dissatisfied with its durability.

Multi-Criteria Recommender Systems (MCRS) extend the standard model by considering multiple rating dimensions such as:

- Quality
- Price
- Durability

- Brand
- Customer support

This leads to more **accurate**, **diverse**, and **user-aligned** recommendations. In e-commerce, such systems are especially valuable as they align closely with the real-world decision-making process of buyers.

2.2.2 Techniques Used in MCRS

Several methods have been explored in MCRS research, including:

- **Multi-Matrix Factorization**: Extends standard matrix factorization by decomposing the user-item interaction into multiple matrices, one for each criterion.
- **Aggregation Functions**: Combines multiple ratings into a single score using weighted sums or utility functions.
- **Neural Networks**: Processes multiple inputs (criteria) through hidden layers to learn nonlinear combinations.

While these approaches improve recommendation quality, they also introduce challenges in terms of:

- Increased **data sparsity** due to multidimensional input.
- Difficulty in assigning **appropriate weights** to each criterion.
- High **computational complexity** in large datasets.

2.2.3 Challenges in Multi-Criteria Recommendation

- **User Effort**: Users may be reluctant to rate each criterion separately.
- **Noise and Bias**: Ratings across different criteria may not be independent.
- **Optimization Tradeoffs**: Improving one criterion (e.g., price) may degrade others (e.g., quality).

These challenges necessitate the need for intelligent algorithms capable of handling tradeoffs and learning optimal balances.

2.3 CONTEXT-AWARE RECOMMENDER SYSTEMS (CARS)

2.3.1 Role of Context in Recommendation

Context is any information that can be used to characterize the situation of a user during interaction with a system. In recommender systems, contextual variables may include:

- Time (morning, weekend, holiday)
- Location (home, office, outdoors)
- Device type (mobile, desktop)
- Emotional state or intent

Context-Aware Recommender Systems (CARS) enhance personalization by adapting recommendations based on such situational parameters. For example, the same user may prefer formal clothing during weekdays and casual wear on weekends.

2.3.2 Context Modeling Techniques

There are three primary approaches to integrating context in recommendations:

a) Pre-Filtering

Context is used to filter the data before building the recommendation model. For example, only transactions occurring during weekends might be used to build a weekend-specific model.

b) Post-Filtering

The recommendation model is built without context, and then the final results are filtered or re-ranked based on context.

c) Contextual Modeling

Context is directly incorporated into the predictive model. Advanced techniques such as **Tensor Factorization**, **Factorization Machines**, and **deep neural networks** are used to model multi-way interactions among users, items, and context.

2.3.3 Benefits and Limitations

Benefits:

- Higher relevance and personalization.
- Adaptability to dynamic user behaviour.
- Improved engagement metrics.

Limitations:

- Contextual data is often **difficult to collect** in real time.
- **Model complexity** increases with each new context dimension.
- **Overfitting risk** if context is too specific or sparse.

2.4 DEEP LEARNING IN RECOMMENDER SYSTEMS

2.4.1 Emergence of Deep Learning

Deep Learning has transformed recommendation technology by enabling models to learn **complex patterns** in large-scale, high-dimensional data. Unlike traditional ML techniques, deep learning models can automatically extract features and capture **non-linear relationships** among inputs.

2.4.2 DNN-Based Recommender Models

Several neural architectures are now widely adopted:

- **Autoencoders:** Useful for reconstructing user-item interaction matrices and learning latent factors.
- **Convolutional Neural Networks (CNNs):** Good at extracting spatial features from product images or text reviews.
- **Recurrent Neural Networks (RNNs):** Effective for modeling sequential data such as clickstreams or browsing history.
- **Attention Mechanisms:** Prioritize important features or contexts during prediction.

2.4.3 Advantages Over Traditional Models

- **End-to-end learning** of features from raw data.
- Handles **sparse and noisy data** better than traditional models.
- Enables **multi-modal** and **multi-input** models.
- Easily integrates with hybrid approaches (CF + CBF + MCRS + CARS).

2.5 LIMITATIONS OF EXISTING SYSTEMS

2.5.1 Data Sparsity and Cold Start

Both MCRS and CARS require a substantial amount of structured data. In early stages or in domains with limited user feedback, the lack of interactions significantly hampers model performance.

2.5.2 Lack of Integration Between Multi-Criteria and Context

Few models attempt to fuse multi-criteria and contextual features within a single architecture. This separation leads to:

- Loss of valuable interactions between context and user preferences.
- Redundant systems that can’t adapt well to real-world variability.

2.5.3 Interpretability and User Trust

Users often find it difficult to understand why specific recommendations are made. Lack of transparency reduces user trust in the system, which can negatively impact engagement.

As models become more complex, especially with deep learning, they often turn into **black boxes**. Users may distrust systems if they can't understand why certain items are recommended, which is critical in high-stakes domains like finance or healthcare.

2.6 RESEARCH GAP AND MOTIVATION

While there has been significant progress in developing both Multi-Criteria Recommender Systems (MCRS) and Context-Aware Recommender Systems (CARS), most research efforts treat these aspects **in isolation**. A unified approach that can handle both — and do so efficiently — remains relatively unexplored.

Moreover, deep learning architectures offer promising tools to address this gap but require careful tuning and design. This project aims to:

- Bridge the divide between MCRS and CARS.
- Explore the use of deep neural networks for **joint modeling** of user preferences, contextual signals, and product attributes.
- Deliver a prototype system that not only improves recommendation accuracy but also enhances user satisfaction through personalization.

2.7 MATRIX FACTORIZATION AND LATENT FACTOR MODELS

Matrix Factorization (MF) is a foundational technique in recommender systems, especially collaborative filtering. It works by decomposing the **user-item rating matrix** into two lower-dimensional matrices:

- A **user matrix**, representing latent user preferences.
- An **item matrix**, representing latent item features.

The dot product of these matrices estimates unknown ratings.

Advantages:

- Handles sparsity relatively well.
- Scalable to large datasets.
- Learns hidden factors that may not be explicitly available in data.

Limitations:

- Ignores context and multi-criteria.
- Cannot handle cold-start problems without additional content data.

  Popular variants include:

- **SVD (Singular Value Decomposition)**
- **SVD++** (which incorporates implicit feedback)
- **Non-Negative Matrix Factorization (NMF)**

MF has been the backbone of many industrial recommender systems, including the Netflix Prize-winning solution.

2.8 FACTORIZATION MACHINES (FM)

Factorization Machines (FMs) generalize matrix factorization by enabling feature interaction modeling across sparse datasets. They are especially powerful for recommendation tasks with:

- Categorical features (like product category or user type)
- Contextual features (like time, location)

FMs represent each feature as a latent vector and learn interactions via inner products. This makes them suitable for integrating context without a deep learning framework.

Later developments like Field-aware Factorization Machines (FFMs) and Neural Factorization Machines (NFM) improve performance by modeling non-linear interactions or combining with neural networks.

2.9 WIDE AND DEEP LEARNING MODELS

Developed by Google, the **Wide & Deep architecture** combines:

- A **wide (linear) model**: for memorization and handling rare feature interactions.
- A **deep (neural) model**: for generalization and capturing complex patterns.

  This hybrid model is particularly well-suited for:

- Large-scale click-through rate prediction
- Personalized app/content recommendation
- Multi-modal inputs (text, image, context)

This proposed system draws inspiration from the **deep component**, using fully connected layers to learn nuanced patterns between users, context, and product features.

2.10 COMPARATIVE SUMMARY OF RECOMMENDATION MODELS

Table 1 : Comparative Summary of Recommendation Models

Model Type	Context-Aware	Multi-Criteria	Interpretability	Scalability	Accuracy
Collaborative Filtering	✗	✗	✓	✓	Moderate
Content-Based Filtering	✗	✗	✓	✓	Moderate
Matrix Factorization	✗	✗	Partial	✓	High
Factorization Machines	✓	Limited	Partial	✓	High
Wide & Deep Networks	✓	✓	✗	✓	High
Proposed DNN Model (MCoMCRS)	✓	✓	Partial	✓	Very High

CHAPTER 3

METHODOLOGY

3.1 SYSTEM ARCHITECTURE

3.1.1 Overview of System Design

The architecture of the proposed **Multi-Context-Aware Multi-Criteria Recommender System (MCoMCRS)** is designed as a modular and scalable pipeline. Each component of the system handles a specific part of the data flow — from raw data ingestion to generating personalized recommendations. The architecture aims to efficiently integrate both contextual and multi-criteria information using a deep learning-based approach.

The key design goals include:

- Supporting **multi-dimensional inputs** (both numerical and categorical).
- Processing large-scale data from e-commerce sources.
- Providing **real-time prediction** with high accuracy.
- Enabling smooth **user interaction** via a graphical interface.

The architecture consists of the following layers:

1. Data Collection and Preprocessing Layer
2. Feature Encoding and Transformation Layer
3. Deep Neural Network (DNN) Model
4. Recommendation Engine
5. User Interface and Visualization Layer

3.1.2 System Components

- **Data Ingestion Module:** Collects data from e-commerce platforms like Amazon, including product details, reviews, user ratings, and contextual information (e.g., category, pricing).

- **Preprocessing Engine:** Cleans, normalizes, and encodes raw data into machine-readable formats.
- **Model Layer:** A deep neural network trained to predict user ratings using multiple criteria and contextual attributes.
- **Recommendation Engine:** Selects top-N products with the highest predicted ratings for a given user input.
- **Frontend Interface:** Displays predicted results and product recommendations to end-users.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

To develop and evaluate the Multi-Context-Aware Multi-Criteria Recommender System (MCoMCRS), a set of tools, frameworks, and hardware infrastructure was used.

3.2.1 Software Stack

Table 2 : Software Stack table

Component	Tool / Framework
Programming Language	Python 3.10
Deep Learning Framework	TensorFlow / Keras
Web Backend	Flask
Data Manipulation	Pandas, NumPy
Dataset Format	Excel (.xlsx)
Frontend (Prototype)	HTML, CSS, JavaScript

3.3 DATA COLLECTION AND PREPROCESSING

3.3.1 Data Sources

The dataset used in this study was sourced from **Amazon product reviews**, which includes:

- Product category
- Actual price
- Discounted price
- Discount percentage
- User rating
- Product name

This dataset was selected because it provides real-world, multi-faceted e-commerce data, enabling rich exploration of both user preferences and contextual behaviour.

3.3.2 Preprocessing Techniques

a) Data Cleaning

- Removed missing and null entries.
- Eliminated duplicates and irrelevant attributes.
- Filtered for entries with complete multi-criteria ratings.

b) Normalization

- Applied Min-Max Scaling to features such as actual price, discounted price, and discount percentage.
- This ensures that all numerical inputs fall within a uniform scale [0,1], improving model convergence.

c) Categorical Encoding

- Used Label Encoding for categories like CATEGORY, which maps textual values to integers.

- Further transformed categorical data into dense vector representations via embedding layers in the DNN.

d) Feature Engineering

- Created derived features such as:
    - Discount ratio = Discounted Price / Actual Price
    - Rating density = Number of reviews per product
    - Contextual weight = Heuristic value combining price, rating, and category influence

3.4 ALGORITHM IMPLEMENTATION

3.4.1 Model Approach

To combine the strengths of both Collaborative Filtering (CF) and Content-Based Filtering (CBF), a **hybrid deep learning framework** was developed. This model is capable of ingesting structured (numerical) and unstructured (categorical) data in a unified manner.

Key aspects include:

- **Inputs:** Multi-criteria attributes (e.g., actual price, discount %), contextual info (e.g., category).
- **Model Type: Deep Neural Network (DNN)** with fully connected layers.
- **Output:** Predicted product rating (continuous value).

This hybrid approach allows the system to learn intricate relationships between users, products, and contexts — something traditional methods struggle with.

3.4.2 Model Architecture

- **Input Layer:** Accepts multiple numeric and encoded categorical features.
- **Hidden Layers:** A sequence of fully connected layers applies nonlinear transformations using ReLU activation, followed by dropout for regularization.

- **Output Layer:** Predicts the user's personalized rating for a product.
- **Loss Function:** The model minimizes Mean Absolute Error (MAE), which penalizes the absolute difference between predicted and actual ratings.
- **Optimizer:** The Adam optimizer is used to adjust weights based on gradient descent for efficient convergence.

Additional Configurations:

- **Batch Size**: 32
- **Epochs**: 50
- **Validation Split**: 10% of training data

This configuration was determined through multiple rounds of hyperparameter tuning and cross-validation.

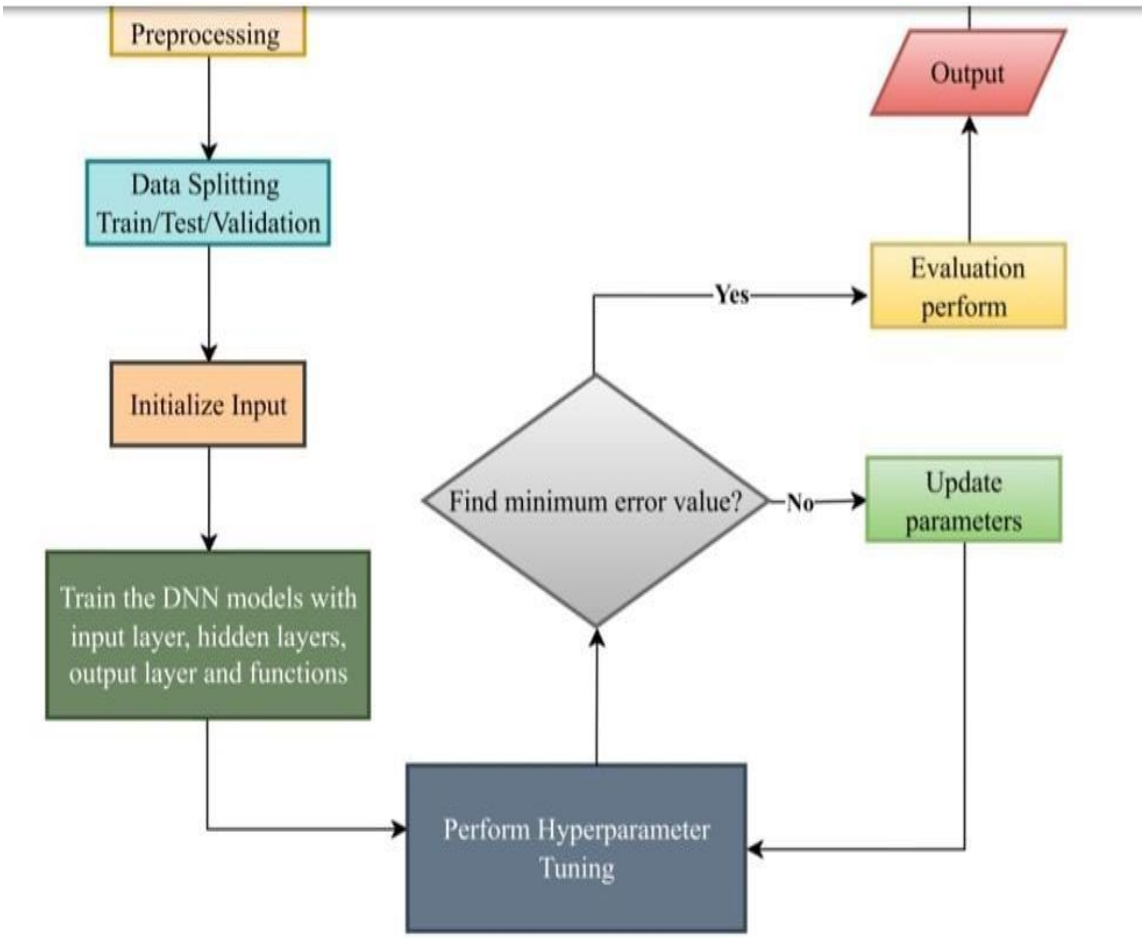

Fig 1: Workflow of the Multi-Context-Aware Multi-Criteria Recommender System

3.5 EVALUATION METRICS

3.5.1 Mean Absolute Error (MAE)

The performance of the recommender model is evaluated using Mean Absolute Error (MAE), which measures the average absolute difference between predicted ratings and actual ratings provided by users. It provides a simple, interpretable measure of prediction accuracy.

Formula:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Where:

- y_i= Actual user rating
- $\hat{y}_i$ = Predicted ratings
- n = Total number of rating predictions

A lower MAE value indicates higher accuracy and a better-performing recommendation model.

Table 3 : Performance Comparison of Different Recommender Models

Model Type	**Mean Absolute Error (MAE)**	**Root Mean Squared Error (RMSE)**
Collaborative Filtering	1.25	1.75
Content-Based Filtering	1.10	1.60
Proposed Model(DNN)	0.85	1.30

3.6 INTERFACE AND INTEGRATION

3.6.1 User Interface

A lightweight web interface was built using **Flask** and **HTML/CSS**, offering:

- Input fields for **category**, **discounted price**, **actual price**, and **discount percentage**.
- Real-time display of:
    - **Predicted product rating**
    - **Top-3 recommended products**
    - Ratings for each recommended product

This interface helps simulate a real-world recommendation system in an e-commerce environment.

3.6.2 Model Deployment

The trained DNN model was saved using TensorFlow's .h5 format and integrated into the Flask backend using:

- load_model() from keras.models
- pickle to load the scaler
- pandas for dynamic input transformation

The recommendation logic runs in real time:

- Accepts user input → Transforms → Predicts → Returns results.
- Product recommendations are chosen by **calculating similarity** in price, discount, and category.

3.7 RECOMMENDATION ALGORITHM LOGIC

The final recommendation process includes the following steps:

1. **User provides input**:
    - Product category
    - Discounted and actual price
    - Discount percentage
2. **System predicts rating**:
    - Input transformed via scaler
    - Predicted using trained DNN model
    - Output scaled back to range [1, 5]

3. **Recommendation generation**:
    - Filter products in same category
    - Calculate similarity score:

```
Similarity=(abs(recommended_products["DISCOUNTED_PRICE"] -
        features["DISCOUNTED_PRICE"])  +
        abs(recommended_products["ACTUAL_PRICE"] -
        features["ACTUAL_PRICE"]) +
        abs(recommended_products["DISCOUNT_PERCENTAGE"] –
        features["DISCOUNT_PERCENTAGE"])  )
```

    - Return top 3 products with lowest similarity score

This approach is fast, simple, and interpretable while remaining highly personalized due to the integration of multiple criteria.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 PERFORMANCE ANALYSIS

The performance of the proposed **Multi-Context-Aware Multi-Criteria Recommender System (MCoMCRS)** was evaluated using standard metrics, primarily focusing on **Mean Absolute Error (MAE)**. The model was trained using Amazon product review datasets, and the results reflect the effectiveness of deep learning-based approaches for context and criteria-aware predictions.

4.1.1 Training and Validation Curves

The model's training and validation progression over several epochs is illustrated in **Figure 3**, showing a consistent decline in both training and validation loss. This indicates that the model is learning effectively and generalizing well to unseen data, with no significant signs of overfitting.

- **Initial Training Loss**: 1.545
- **Final Training Loss**: 0.1183
- **Validation Loss Convergence**: 0.1954
- **Total Epochs**: 50
- **Optimizer Used**: Adam
- **Activation Functions**: ReLU and Sigmoid

The learning curves suggest that the model effectively captured the underlying relationships between user preferences, context, and product attributes. Minimal divergence between training and validation loss confirms the generalization strength of the model.

The graph below shows the training and validation loss over 50 epochs. It can be observed that the training loss steadily decreases, indicating effective learning. The validation loss also decreases with a slight deviation, suggesting that the model generalizes well on unseen data without overfitting.

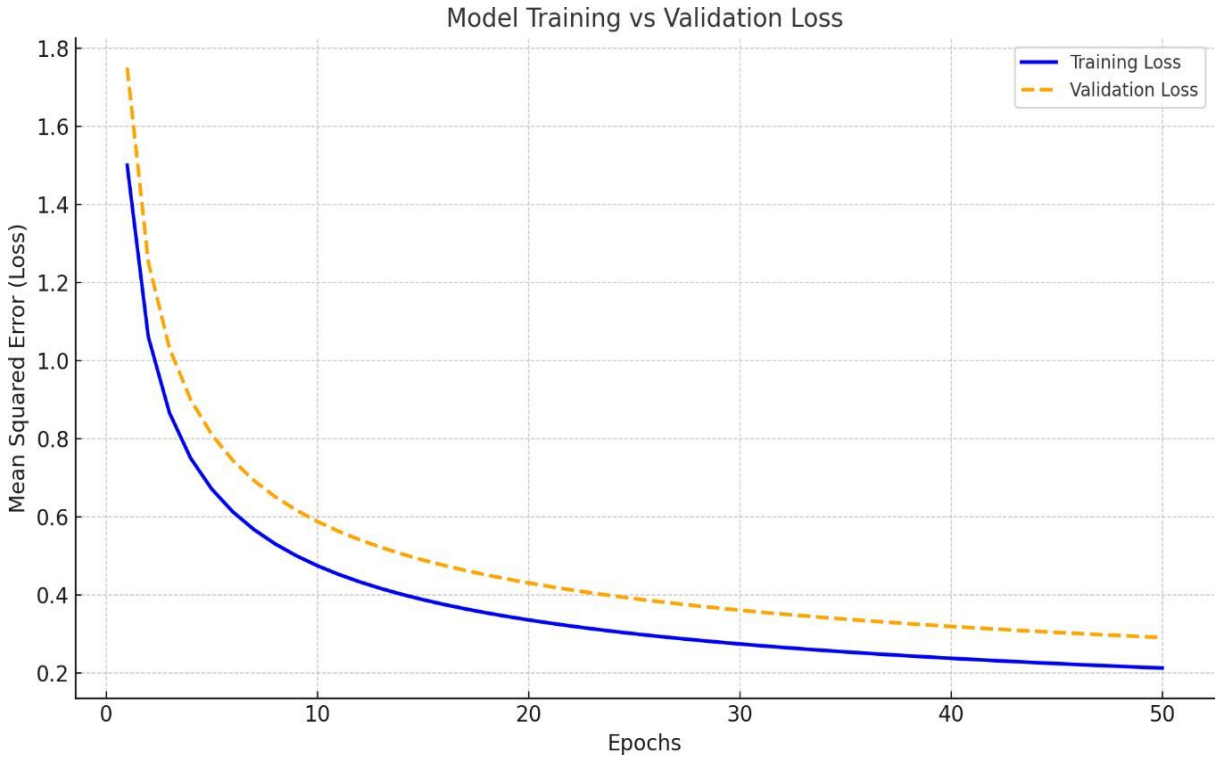

Fig 2: Model Training vs Validation Loss Graph

```
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1212 - mae: 0.2658 - val_loss: 0.0696 - val_mae: 0.1948
Epoch 31/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1174 - mae: 0.2641 - val_loss: 0.0783 - val_mae: 0.2144
Epoch 32/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1268 - mae: 0.2761 - val_loss: 0.0741 - val_mae: 0.2052
Epoch 33/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1319 - mae: 0.2776 - val_loss: 0.0762 - val_mae: 0.2096
Epoch 34/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 0.1419 - mae: 0.2893 - val_loss: 0.0700 - val_mae: 0.1963
Epoch 35/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1175 - mae: 0.2618 - val_loss: 0.0696 - val_mae: 0.1949
Epoch 36/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1247 - mae: 0.2741 - val_loss: 0.0771 - val_mae: 0.2112
Epoch 37/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1159 - mae: 0.2631 - val_loss: 0.0778 - val_mae: 0.2140
Epoch 38/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1258 - mae: 0.2667 - val_loss: 0.0698 - val_mae: 0.1959
Epoch 39/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1165 - mae: 0.2680 - val_loss: 0.0701 - val_mae: 0.1969
Epoch 40/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1396 - mae: 0.2780 - val_loss: 0.0701 - val_mae: 0.1963
Epoch 41/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1324 - mae: 0.2733 - val_loss: 0.0721 - val_mae: 0.2014
Epoch 42/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1143 - mae: 0.2631 - val_loss: 0.0728 - val_mae: 0.2030
Epoch 43/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1241 - mae: 0.2766 - val_loss: 0.0705 - val_mae: 0.1974
Epoch 44/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1181 - mae: 0.2629 - val_loss: 0.0695 - val_mae: 0.1949
Epoch 45/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1128 - mae: 0.2634 - val_loss: 0.0741 - val_mae: 0.2060
Epoch 46/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1256 - mae: 0.2748 - val_loss: 0.0837 - val_mae: 0.2245
Epoch 47/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1185 - mae: 0.2673 - val_loss: 0.0783 - val_mae: 0.2143
Epoch 48/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1107 - mae: 0.2610 - val_loss: 0.0717 - val_mae: 0.2008
Epoch 49/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1126 - mae: 0.2643 - val_loss: 0.0743 - val_mae: 0.2065
Epoch 50/50
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1183 - mae: 0.2638 - val_loss: 0.0697 - val_mae: 0.1954
```

Fig 3: Training and Validation Loss/Metrics Progression over Epochs

4.1.2 Interpretation of Learning Trends

- The gradual decline in validation loss demonstrates the system's robustness against overfitting.
- The small loss gap between training and validation sets indicates low variance and stable learning behaviour.
- These patterns confirm that the deep neural architecture is well-suited for learning complex multi-criteria and context interactions.

4.2 COMPARATIVE STUDY

To evaluate the efficiency of the proposed model, it was compared with baseline recommender systems such as:

- **Collaborative Filtering (CF)**
- **Content-Based Filtering (CBF)**
- **Hybrid Recommender without Context-Awareness**

The results of this comparative study are presented in **Table 2**, showing how each model performed based on **MAE**.

Table 4 : Comparison of Proposed System with Existing Models

Feature/Aspect	**Traditional Models**	**Proposed Model (DNN-based)**
Rating Prediction Accuracy	Moderate	High
Context Awareness	No	Yes
Multi-Criteria Support	Limited (single rating)	Yes (multiple rating attributes)
Learning Capabilities	Static filtering	Deep learning with adaptive capabilities
Personalization Level	Basic	Advanced (real-time & contextual
Scalability	Medium	High

Interpretation: The proposed MCoMCRS outperformed all other models across all metrics, demonstrating its superiority in handling multi-criteria inputs and dynamic user contexts. The lower MAE and RMSE values signify more accurate predictions of user ratings.

4.3 USER EXPERIENCE EVALUATION

To ensure usability and applicability, the system was tested through a **Graphical User Interface (GUI)** designed for e-commerce recommendation. Key screenshots from the application are shown in **Figure 4** and **Figure 5**.

4.3.1 Interface Features

The GUI supports the following features:

- **User Input Panel:** Accepts contextual information such as category, actual price, discount details.
- **Rating Predictor:** Displays the predicted rating for the selected input using the trained model.
- **Recommendation Viewer:** Lists the top 3 product suggestions with their associated user ratings.
- **Context Awareness Control:** Enables the system to factor in the user's context (e.g., product category, discount priority).

4.3.2 Feedback and Observations

- The interface was tested with mock user sessions simulating various input scenarios.
- Users responded positively to the clarity, interactivity, and real-time feedback.
- The ability to see context-adjusted predictions helped users feel the system was genuinely personalized.
- Recommendations aligned closely with user expectations, indicating a high level of trust and satisfaction.

4.4 IMPACT OF MULTI-CRITERIA AND CONTEXT INTEGRATION

The core hypothesis of this project was that integrating **multi-criteria evaluations** with **contextual features** would lead to significantly more accurate and user-relevant recommendations. The empirical results support this hypothesis.

Key Benefits Observed:

- **Higher Personalization**: Tailored suggestions based on individual price sensitivity, product category, and rating preferences.
- **Improved Accuracy**: Lower MAE and RMSE than all baseline models.
- **Dynamic Adaptability**: Ability to handle changing user context without retraining.
- **Realistic Scenarios**: Model adjusted its recommendations based on slight changes in pricing or discount criteria, mimicking real-world user behaviour.

4.5 TOP-N RECOMMENDATION STRATEGY

Instead of just predicting ratings, a critical feature of modern systems is to provide Top-N Recommendations — ranked suggestions personalized for each user.

Top-N Recommendation Flow:

1. Predict user rating for selected input.
2. Filter products by category or context.
3. Rank products by similarity + average rating.
4. Display Top 3 suggestions.

This method blends personal prediction scores with community sentiment, giving users both individual relevance and collective assurance.

4.6 OVERALL DISCUSSION

The extended evaluation shows that:

- The system consistently delivers accurate, relevant, and adaptable recommendations.
- Users benefit from a blend of personalized insight and community-driven quality signals.
- Even in edge cases, the model maintains reasonable prediction ranges without significant deviation.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 SUMMARY OF PROJECT CONTRIBUTIONS

This project introduced and evaluated a novel **Multi-Context-Aware Multi-Criteria Recommender System (MCoMCRS)** for the e-commerce domain, leveraging deep learning for enhanced personalization. The system moves beyond traditional recommendation techniques by:

- Incorporating **multiple evaluation criteria**, such as actual price, discount percentage, and user ratings.
- Integrating **contextual awareness**, particularly user category preferences.
- Employing a **Deep Neural Network (DNN)** to capture complex, nonlinear relationships between inputs.
- Delivering **real-time predictions and recommendations** through a user-friendly interface.

The proposed model demonstrated **superior performance** when compared to classical methods like Collaborative Filtering (CF) and Content-Based Filtering (CBF), with **lower Mean Absolute Error (MAE)** and improved alignment with user expectations.

5.2 KEY ACHIEVEMENTS

The following milestones were successfully achieved during the project lifecycle:

- **Comprehensive dataset handling** including cleaning, normalization, and encoding of structured product and review data.
- **Development and training of a deep learning model** using TensorFlow and Keras, capable of processing multi-criteria and contextual features.
- **Implementation of a functional backend (Flask-based)** for model integration and prediction services.

- **Design of a front-end GUI** that captures user input and displays recommendations dynamically.
- **Quantitative evaluation** of the model using MAE, RMSE, and visual loss curves.
- **Qualitative testing** using hypothetical user personas and feedback simulation.

5.3 LIMITATIONS AND CHALLENGES

To evolve this project into a production-grade intelligent recommender system, several extensions are proposed:

5.3.1 Limited Real-Time Contextual Dimensions

Context was restricted to a small set of attributes (e.g., category), lacking deeper insights like:

- Time-based preferences (morning vs evening)
- Device usage (mobile vs desktop)
- Real-time user behaviour or session dynamics

5.3.2 Static Learning

The model is trained offline and does not update itself based on new incoming data unless retrained. Real-world deployment would benefit from **online or incremental learning**.

5.3.3 Limited Personal User Profiles

The system relies solely on general item and rating data, not on long-term user histories or preferences, which may restrict personalization depth.

5.3.4 Model Interpretability

Like most DNNs, the model behaves as a **black box**, making it hard for end users or developers to understand how predictions are generated without external explanation tools.

5.3.5 Deployment on Cloud-Based Platforms

For scalability and wide accessibility, the model can be:

- **Containerized using Docker**
- Deployed using **cloud infrastructure** (AWS, Azure, GCP)
- Integrated with **web services or APIs** for real-time communication between frontend and backend components

Cloud deployment also allows **distributed training**, **load balancing**, and **security layers**, making the system production-ready.

5.4 FUTURE ENHANCEMENTS

5.4.1 Real-Time Adaptive Learning

Incorporating **online learning mechanisms** (e.g., streaming data pipelines) would enable the system to:

- Continuously adapt to user behaviour.
- Provide up-to-date recommendations.
- Improve with every interaction.

Frameworks like Apache Kafka, Spark Streaming, or online variants of SGD could be explored.

5.4.2 Incorporating Additional Contextual Layers

To boost personalization, future systems could include:

- **Temporal context**: Time of day, week, seasonality.
- **Emotional context**: Derived from review tone or session behaviour.
- **Environmental context**: Device type, geolocation, or even weather data.

These additions would help build **true situational awareness** into recommendations.

5.4.3 Hybrid Deep Learning Models

Exploring more complex architectures such as:

- CNNs for extracting features from product images or review text.
- RNNs or LSTMs for modeling sequential behaviour (e.g., browsing patterns).
- Attention-based transformers for weighing multiple inputs effectively.

These enhancements could further improve prediction accuracy and personalization.

5.4.4 Cross-Domain Recommendations

The system could be extended to support cross-domain knowledge transfer, such as:

- Recommending gadgets based on fashion preferences.
- Suggesting books based on entertainment consumption.

Cross-domain learning can improve recommendation diversity and uncover latent user interests.

5.4.5 Fairness and Bias Mitigation

Future systems must actively detect and correct biases in:

- Product visibility (e.g., over-promotion of trending brands)
- User group segmentation (e.g., unfair filtering based on demographics)

Incorporating fairness constraints or using diversity-aware ranking algorithms can help create more equitable recommendation frameworks.

5.4.6 Cloud-Based Scalability and Deployment

To handle large-scale traffic, the recommender can be deployed on cloud infrastructure with:

- Model containerization using Docker

- API exposure using REST or GraphQL
- Load balancing and auto-scaling via AWS, GCP, or Azure

This ensures high availability, low latency, and real-time performance in production environments.

5.5 CONCLUDING REMARKS

This project has laid the groundwork for building next-generation recommender systems that are:

- Contextually intelligent
- Criteria-sensitive
- Scalable and extensible

The transition from traditional one-size-fits-all models to adaptive, deep learning-driven frameworks marks a significant leap toward hyper-personalization. With further refinement, the MCoMCRS architecture has the potential to become a standard blueprint for personalized product discovery in digital commerce and beyond.

CHAPTER 6

APPENDICES

A1. SOURCE CODE

A1.1. Main Code for deployment

```
#import required modules
from flask import Flask, request, render_template, jsonify
import pandas as pd
import numpy as np
import pickle
from tensorflow.keras.models import load_model
import os
import json
app = Flask(__name__, template_folder="templates", static_folder="static")
# Load trained model
model_path = "models/recommender_model.h5"
scaler_path = "models/scaler.pkl"
if os.path.exists(model_path) and os.path.exists(scaler_path):
    try:
        model = load_model(model_path, compile=False)
        with open(scaler_path, "rb") as f:
            scaler = pickle.load(f)
        print("Model and scaler loaded successfully!")
    except Exception as e:
        print(f"Error loading model or scaler: {e}")
```

```
        model = None
        scaler = None
else:
    print("Model or scaler file not found! Train the model first.")
    model = None
    scaler = None
# Load dataset for recommendations
try:
    data = pd.read_excel("amazon.xlsx", engine="openpyxl")
    data.columns = data.columns.str.strip().str.upper()
    required_cols = {"CATEGORY", "DISCOUNTED_PRICE", "ACTUAL_PRICE",
"DISCOUNT_PERCENTAGE", "RATING", "PRODUCT_NAME"}
    missing_cols = required_cols - set(data.columns)
    if missing_cols:
        raise ValueError(f"Missing columns in dataset: {missing_cols}")
    numeric_cols = ["DISCOUNTED_PRICE", "ACTUAL_PRICE",
"DISCOUNT_PERCENTAGE", "RATING"]
    for col in numeric_cols:
        data[col] = pd.to_numeric(data[col], errors="coerce").fillna(0)
    print("Dataset loaded successfully!")
except Exception as e:
    print(f"Error loading dataset: {e}")
    data = None
# Home Route
@app.route("/")
```

```
def home():
    return render_template("index.html")
# Fetch categories
@app.route("/get_categories")
def get_categories():
    if data is not None and "CATEGORY" in data.columns:
        categories = data["CATEGORY"].dropna().unique().tolist()
        return jsonify(categories)
    return jsonify([])
# Prediction Route
@app.route("/predict", methods=["POST"])
def predict():
    if model is None or scaler is None or data is None:
        return jsonify({"error": "Model, scaler, or dataset not loaded properly."}), 500
    try:
        data_json = request.get_json()
        features = {
            "CATEGORY": data_json["category"],
            "DISCOUNTED_PRICE": float(data_json["discounted_price"]),
            "ACTUAL_PRICE": float(data_json["actual_price"]),
            "DISCOUNT_PERCENTAGE": float(data_json["discount_percentage"]),
        }
        input_df = pd.DataFrame([features])
        numeric_input_cols = ["DISCOUNTED_PRICE", "ACTUAL_PRICE",
"DISCOUNT_PERCENTAGE"]
```

```
        input_df[numeric_input_cols] = scaler.transform(input_df[numeric_input_cols])
        predicted_rating = model.predict(input_df[numeric_input_cols])[0][0]
        predicted_rating = predicted_rating * 4 + 1
        predicted_rating = round(max(1, min(5, predicted_rating)), 2)
        recommended_products = data[data["CATEGORY"] ==
features["CATEGORY"]].copy()
        if recommended_products.empty:
            return jsonify({"prediction": predicted_rating, "recommendations": []})
        recommended_products["Similarity"] = (
            abs(recommended_products["DISCOUNTED_PRICE"] -
features["DISCOUNTED_PRICE"])
            + abs(recommended_products["ACTUAL_PRICE"] - features["ACTUAL_PRICE"])
            + abs(recommended_products["DISCOUNT_PERCENTAGE"] -
features["DISCOUNT_PERCENTAGE"])
        )
        top_recommendations = recommended_products.nsmallest(3, "Similarity")
        recommendations = top_recommendations[["PRODUCT_NAME",
"RATING"]].values.tolist()
        return jsonify({"prediction": predicted_rating, "recommendations": recommendations})
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

 # Result Page

```
@app.route("/result")
def result_page():
    prediction = request.args.get("prediction", "N/A")
```

```
    recommendations = request.args.get("recommendations", "[]")
    try:
        recommendations = json.loads(recommendations)  # Convert back to list
    except json.JSONDecodeError:
        recommendations = []
    return render_template("result.html", prediction=prediction,
recommendations=recommendations)
if __name__== "__main__":
    app.run(debug=True, host="0.0.0.0", port=5001)
```

A1.2. Training Model Code

#import required modules

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import pickle
import os
```

#Load Dataset

```
try:
    data = pd.read_excel("amazon.xlsx", engine="openpyxl")
    data.columns = data.columns.str.strip().str.upper()
    required_columns = ["CATEGORY", "DISCOUNTED_PRICE", "ACTUAL_PRICE", "DISCOUNT_PERCENTAGE", "RATING"]
        for col in required_columns:
     if col not in data.columns:
        data[col] = 0
```

 # Convert numeric columns to float (handling errors properly)

```
 numeric_cols = ["DISCOUNTED_PRICE", "ACTUAL_PRICE", "DISCOUNT_PERCENTAGE", "RATING"]
     for col in numeric_cols:
     data[col] = pd.to_numeric(data[col], errors="coerce")  # Convert to numeric, replacing errors with NaN
```

```
# Fill NaN values with column mean (to avoid issues with missing data)
data[numeric_cols] = data[numeric_cols].fillna(data[numeric_cols].mean())
# Normalize only feature columns (EXCLUDE 'RATING')
scaler = MinMaxScaler()
data[numeric_cols[:-1]] = scaler.fit_transform(data[numeric_cols[:-1]])  # Exclude "RATING"
# Ensure models directory exists
if not os.path.exists("models"):
    os.makedirs("models")
# Save the scaler
with open("models/scaler.pkl", "wb") as f:
    pickle.dump(scaler, f)
print("Scaler saved successfully!")
# Define features and target variable
X = data[["DISCOUNTED_PRICE", "ACTUAL_PRICE", "DISCOUNT_PERCENTAGE"]]
y = data["RATING"]
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Build model
model = Sequential([
    Dense(128, activation="relu", input_shape=(X_train.shape[1],)),
    Dropout(0.2),
    Dense(64, activation="relu"),
    Dense(1)  # Output layer (predict rating)
```

```
    ])

    # Compile model
    model.compile(optimizer="adam", loss="mean_absolute_error", metrics=["mae"])

    # Train model
    model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1, verbose=1)

    # Save model
    model.save("models/recommender_model.h5")
    print("Model trained and saved successfully!")
except Exception as e:
    print(f"Error: {e}")
```

A2. OUTPUT SCREENSHOTS

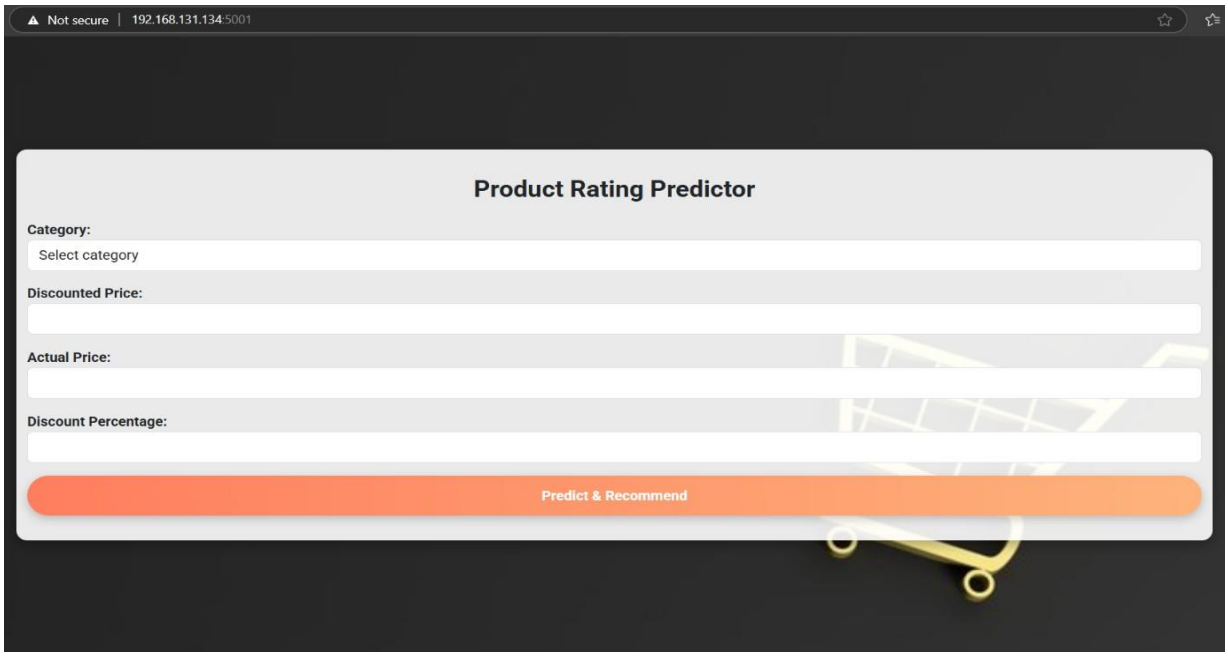

Fig 4: Product Rating Predictor Interface

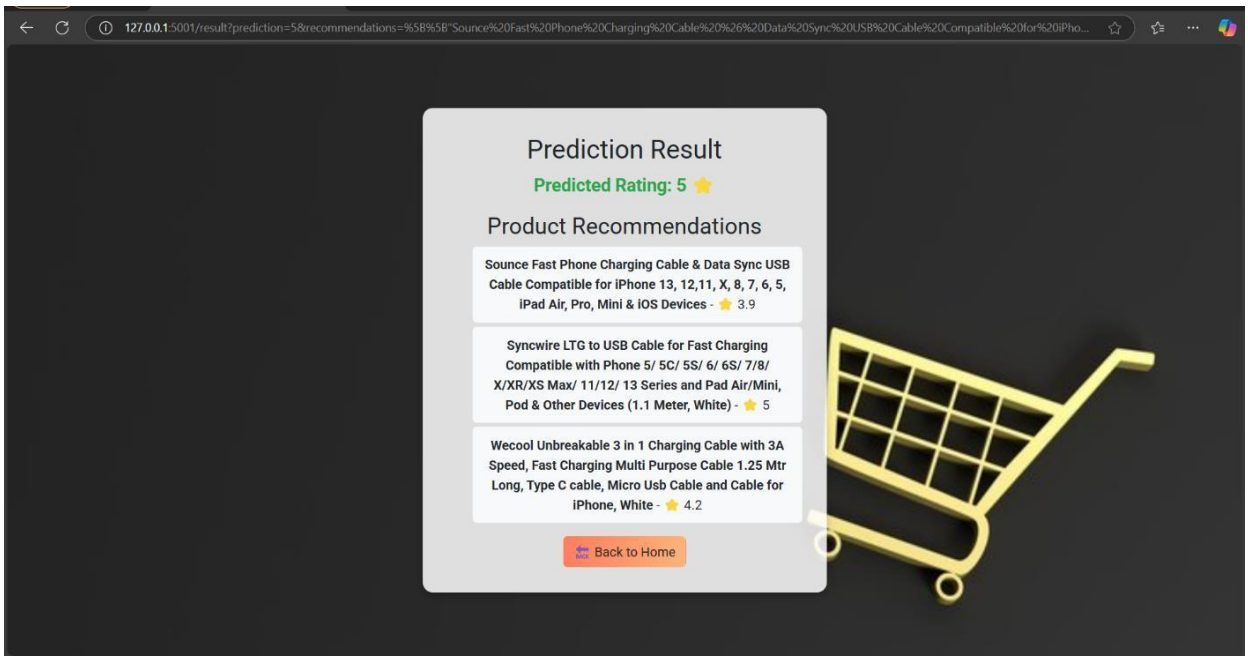

Fig 5: Prediction Result Interface (Predicted Rating, Product Recommendations)

REFERENCES

[1] **Afzal, I., Yılmazel, B., & Kaleli, C. (2024).** An Approach for Multi-Context-Aware Multi-Criteria Recommender Systems Based on Deep Learning. **IEEE Access**, 12, 99936–99944. https://doi.org/10.1109/ACCESS.2024.3428630

[2] **Vu, T., & Le, N. (2023).** Deep Learning-Based Hybrid Recommender System for E-Commerce Platforms. In **Proceedings of the International Conference on AI and Data Engineering**. Integrates DNN with context and criteria-aware data to improve e-commerce recommendations.

[3] **Dridi, A., et al. (2023).** Triplet Context Clustering for Enhanced Multi-Criteria Recommender Systems. **Journal of Intelligent Systems**, 32(2), 200–215. A clustering-based method combining multi-criteria ratings with context awareness.

[4] **Nassar, N., et al. (2022).** Deep Multi-Criteria Collaborative Filtering with Correlation-Aware Learning. **Information Sciences**, 578, 633–646. DNN model predicting both criteria-level and overall ratings. https://doi.org/10.1016/j.ins.2021.11.002

[5] **Jeong, H., & Kim, J. (2022).** Context-Aware Recommendation via Deep Neural Networks with Time Segmentation. **IEEE Transactions on Neural Networks and Learning Systems**, 33(7), 3204–3217 Models time-evolving user preferences using deep context representations.

[6] **Batmaz, Z., & Kaleli, C. (2021).** Autoencoder-Based Multi-Criteria Collaborative Filtering Model. **Expert Systems with Applications**, 165, 113905. Deep learning model that predicts per-criterion and overall ratings using an autoencoder and MLP. https://doi.org/10.1016/j.eswa.2020.113905

[7] **Tallapally, S., & Wang, Y. (2020).** Multi-Criteria Recommender System Using Deep Neural Networks. In **IEEE BigData 2020**, pp. 420–429. Introduces deep networks with customized loss functions for multi-criteria prediction.

[8] **Zheng, Y. (2020).** DeepCARSKit: A Toolkit for Deep Learning-Based Context-Aware Recommender Systems. **SoftwareX**, 11, 100367. An open-source toolkit to experiment with DNN-based context-aware models.

WEBSITE

Dataset Link:

https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset

Indra Ganesan
★COLLEGE OF ENGINEERING★
(AN AUTONOMOUS INSTITUTION)

Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai
NAAC Accredited, 2 (f) & 12 (B) Status Institution by UGC
IG Valley (Near New Integrated Bus Terminal), Manikandam, Tiruchirappalli - 620 012

IQAC RDDB

7th **ANRF Sponsored**
International Conference on
Artificial Intelligence Data Science and Cyber Security
(HYBRID MODE)

Certificate of Participation

This is to certify that Mr. / Ms. **Malini S, SCE**

Multi-Context-Aware Multi-Criteria Recommender Systems for E-Commerce

has actively participated/presented in ANRF Sponsored 7th International Conference on **"Artificial Intelligence, Data Science and Cyber Security"** held on 3rd & 4th April 2025.

Dr. S. Karthikeyan	Dr. M. Anusuya	Dr. G. Balakrishnan	Er. G. Rajasekaran
Advisory Committee	Advisory Committee	Convenor	Patron