Faculty of Computer & Information Technology

Electronic E-Store

Team Member:

NO.	Student Name	Student ID
1	Mohamed Al Farmway	ID: 20-00660
2	Shaker Emad	ID: 20-00336
3	Mohamed Ahmed	ID: 20-00145
4	Abdulrahman Zakaria	ID: 20-01613
5	Khalid Ashraf	ID: 20-01975
6	Philopater Shenouda	ID: 20-00578
7	Ibrahim Abdeltawab	ID: 20-01508

SUPERVISORS

Dr Rodaina Abdelsalam

Assistant professor Faculty of Computers and information Technology EELU

Eng. Aya Magdy

Demonstrator, Department, and Information Technology Egyptian E-Learning University

A Graduation Project Report Submitted for the Partial Fulfillment of the Requirements of the B.SC. Degree

Assiut 2023-2024

Acknowledgement

First and foremost, we thank God Almighty for the blessing of reaching this advanced stage in our education to achieve this result of success.

We put effort into this project. However, this would not have been possible without the support and assistance of many individuals and organizations. I would like to express my sincere thanks to all of them.

Thank you to the Egyptian University for E-Learning for helping us reach this level of awareness.

We would like to express our gratitude to our Project Supervisor Dr. Rodaina Abdelsalam for giving us the opportunity to lead us and provide invaluable guidance during this project. It was a great honor and privilege for us to work under her supervision.

We are especially indebted to express our heartfelt thanks to Eng. Aya Magdy for the guidance, constant supervision, patience, friendship and great sense of humor for all his efforts with us.

Finally, we would like to express our gratitude to everyone who helped us during the graduation project.

Abstract

The graduation project represents an exceptional journey towards redefining the electronic shopping experience, through designing an advanced online store dedicated to buying and selling laptops and accessories. The project aims to find a unique balance between technology and innovation, with a focus on improving the quality and ease of the digital shopping experience.

At the heart of the project is a smart chatbot that relies on machine learning techniques to guide users toward laptop choices that suit their needs. The chatbot enables effective interaction with users, as it asks targeted questions to understand their knowledge of laptops and their preferences.

In addition, the project includes an advanced search system based on artificial intelligence techniques, which facilitates searches and provides accurate results quickly. Advanced digital marketing strategies complement this effort, highlighting the store and increasing social engagement through social media.

The project aims primarily to enhance technical communication between the store and the consumer, and to enable every individual, whether he has technical experience or not, to make smart decisions in choosing the laptop that meets his needs. Overall, this project represents a unique stop to explore the interaction of technology with the field of digital shopping and is an excellent opportunity to develop students' skills in the fields of design, artificial intelligence, and marketing management.

Contents

List of Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
CNN	Convolutional Neural Network
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
NLP	Natural Language Processing
(PCA).	Principal component analysis
ANN	Artificial Neural Network
(HTML)	Hypertext Markup Language
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML

CHAPTER 1

1 Introduction

This avant-garde laptop shopping platform represents a paradigm shift in the realm of online commerce, introducing a cutting-edge chatbot that seeks to redefine the user experience when selecting and purchasing laptops. At the core of this innovation is a revolutionary chatbot, meticulously designed to cater to users of varying technological proficiency. Going beyond the conventional model, this intelligent chatbot not only understands and accommodates the needs of tech novices but also offers sophisticated recommendations tailored for the more tech-savvy individuals.

The platform's commitment to user-centricity is exemplified through its user-friendly interface that seamlessly integrates state-of-the-art technology with personalized assistance. Unlike traditional online shopping experiences, this platform transcends the mere transactional aspect, aiming to create a journey where users not only find the perfect laptop but also relish the entire process. The chatbot serves as a digital guide, engaging users in interactive and intuitive conversations, ensuring that even those with limited technical knowledge can make informed decisions.

Through intricate algorithms and artificial intelligence, the chatbot delves into users' preferences, analyzing nuanced queries to provide detailed guidance. This amalgamation of technological prowess and user-centered design fosters an environment where every user, irrespective of their expertise, can discover a laptop precisely aligned with their unique expectations and requirements.

As a culmination of these advancements, this graduation project marks a significant step forward in the realm of online laptop shopping. By enhancing accessibility and personalization through the innovative use of artificial intelligence, it transforms the shopping experience into a valuable resource for both novices entering the tech arena and experts seeking tailored recommendations. This project aspires not only to simplify the laptop selection process but to elevate it into a holistic and enjoyable journey for all users.

Introducing a chatbot that can cater to users of all technological backgrounds is indeed a great idea. By helping and recommendations tailored to both novices and tech-savvy individuals, you're ensuring that all users can have a seamless and personalized shopping experience.

Having a chatbot that understands user preferences and can recommend laptops based on individual needs and specifications can significantly enhance the overall shopping experience. Whether someone is looking for a simple, user-friendly laptop for basic tasks or a high-performance machine with advanced features, your platform can cater to their needs effectively.

Incorporating cutting-edge technology like natural language processing and machine learning algorithms can further enhance the capabilities of the chatbot, allowing it to understand user queries more accurately and provide more relevant recommendations over time.

Overall, your avant-garde laptop shopping platform has the potential to redefine the way people shop for laptops online, offering a truly personalized and user-friendly experience for customers of all technological backgrounds.

This new initiative represents an important step in the world of online shopping, as it seeks to improve accessibility and personalization through the innovative use of artificial intelligence. By simplifying the laptop

selection process and elevating it to a comprehensive and enjoyable experience for all users, this initiative aims to achieve a major transformation in this field.

Thanks to the use of technology and artificial intelligence, the initiative can provide a unique and innovative shopping experience that is in line with the aspirations of users, whether they are beginners in the world of technology or highly experienced professionals. A sophisticated chatbot can understand each user's unique needs and provide recommendations that are completely tailored to those needs.

In this way, the initiative offers an opportunity for everyone to benefit from the benefits of technology directly and easily, regardless of their level of previous experience with technology. This platform provides an opportunity for users to discover and choose laptops that meet their needs easily and accurately, which enhances the online shopping experience and makes it more comprehensive and enjoyable.

1.1 History

In the early days of the internet, e-commerce platforms emerged as a new and bold experiment. The year 1994 is notably marked by the launch of the first online shopping website by "Net Market," representing a groundbreaking step in the business world. The primary challenge during that period was to build consumer trust in executing online payment transactions, leading to the emergence of some pioneering security technologies.

As time progressed and internet usage increased, several leading platforms in e-commerce appeared, such as "Amazon" founded in 1995 and "eBay" in 1996, introducing new models for electronic commerce. With technology advancing and awareness growing regarding the benefits of online shopping, e-commerce platforms became an essential part of people's lives in the early 21st century. Companies and individuals alike began launching their own online stores to better and faster meet consumer needs.

Our project idea comes as a significant leap in developing the concept of online shopping. We present a sophisticated recommendation system based on artificial intelligence, embodying innovation, and greatly enhancing the device selection process. The intelligent chatbot provides detailed and personalized guidance to users, serving as a digital assistant that accurately understands the needs of everyone.

Our concept embodies a futuristic vision for online shopping, integrating the latest technologies innovatively with personalized assistance to achieve a seamless and enjoyable shopping experience. By analyzing personal preferences and offering precise recommendations, we aim to ensure that each user finds a laptop that precisely aligns with their expectations and requirements.

This project represents a significant step forward in improving the online shopping experience, indicating the importance of adopting modern technology to elevate service levels and cater to the daily needs of consumers.

1.2 Motivation

The driving force behind embarking on this project lies in a deep-seated motivation to address a particular facing substantial challenges when it comes to selecting an appropriate laptop due to a lack of adequate knowledge. This demographic, often overlooked in the realm of technology, encompasses individuals who struggle with the complexities of laptop specifications and features, hindering their ability to make informed choices.

Recognizing this gap, the primary objective of our project is to create a meaningful solution tailored explicitly for this segment of society. We envision a user-friendly platform equipped with an intelligent chatbot designed to serve as a digital companion, providing guidance and recommendations in a manner that is accessible and comprehensible, even for those with limited technological expertise.

The core essence of the project is to empower users who find themselves on the fringes of the tech-savvy spectrum, ensuring that they are not left behind in the rapidly advancing world of laptops and technology. The platform aims to bridge the knowledge gap, offering a simplified yet comprehensive experience that transforms the daunting task of choosing a laptop into an accessible and user-centric process.
Our commitment extends beyond mere accessibility; it's about inclusivity and personalization.

The intelligent chatbot embedded in the platform employs sophisticated algorithms to understand individual preferences, usage patterns, and specific needs. By doing so, it crafts personalized recommendations that go beyond the conventional one-size-fits-all approach, acknowledging the uniqueness of each user within this demographic.

In essence, this project serves as a beacon of technological inclusivity, championing the cause of democratizing access to information and

technology. By making the laptop selection process more user-centric and tailored, we aim not only to simplify the journey for this segment but also to foster a sense of empowerment, ensuring that everyone, regardless of their technological background, can confidently navigate the landscape of laptop choices and find the device that perfectly aligns with their distinct requirements.

1.3 Problem Statement

Problems that a large segment of the population, specifically those who lack sufficient technological knowledge, may face when navigating the complexities of choosing a suitable laptop. Often described as not being tech-savvy, these individuals face enormous obstacles in the ever-evolving landscape of technology options, especially around, laptop, purchasing.

At the heart of this problem lies a deep knowledge gap, as a large portion of the population lacks the necessary understanding of the complex specifications and features associated with laptops. This information asymmetry presents a major obstacle, as these individuals grapple with the daunting task of making informed choices in a sea of technological options.

Our project seeks to correct this disparity by offering a solution that is not only accessible, but also tailored to those with limited technological proficiency. The motivation stems from a commitment to inclusivity, and a recognition of the unique challenges that non-tech-savvy individuals face when venturing into the realm of laptop shopping. We aim to empower this demographic by providing them with an easy-to-use platform equipped with an intelligent chatbot capable of demystifying the complexities of choosing a laptop.

The crux of the matter is to democratize access to technology, ensuring that everyone, regardless of their technical background, can confidently participate in the process of choosing a laptop. Through personalized recommendations, streamlined interfaces, and intuitive interactions facilitated by a chatbot, we aspire to make the laptop selection journey not only understandable, but also enjoyable for individuals who may feel overwhelmed or excluded from the technology landscape.

At its core, our project is a testament to the belief that technology should be inclusive, with the potential to enrich the lives of all individuals, regardless of their level of familiarity with technology. By addressing the challenges non-tech-savvy people face in choosing a laptop, we aspire to promote a more equitable, user-centered technology experience for everyone.

1.4 Problem Solution

Implement an Interactive Chatbot for Personalized Recommendations:
To address the limited user guidance, a solution involves the integration of an interactive chatbot that employs indirect questioning techniques. The chatbot engages users in a conversation, asking non-technical questions to assess their preferences, usage patterns, and specific needs. This approach aims to extract relevant information without overwhelming the user with technical jargon.

User Profiling Questions:
The chatbot can initiate the conversation by asking user-friendly questions to create a profile. For example:

"What activities do you primarily use your laptop for?"

"Do you have a preference for a specific brand or operating system?"

"Are you more focused on portability or performance?"

Preferences and Aesthetics:

To understand aesthetic and usability preferences:

"Do you prefer a sleek and lightweight design or a more robust and powerful build?"

"Are you inclined towards a larger screen for multimedia or a more compact size for mobility?"
Budget Considerations:
Addressing budget constraints without explicitly asking for a specific amount:
"Would you like a budget-friendly option, a mid-range laptop, or a premium device?"
Brand and Feature Perception:
Understanding brands perception and desired features:
"Do you associate with any specific laptop brands or have feature preferences?"
"Are there any must-have features, such as touch screen, long battery life, or specific ports?"

User Feedback Loop:
The chatbot can continuously refine its recommendations based on user feedback, creating an adaptive system that learns from user interactions and improves over time.

Benefits:
User-Centric Approach:
The chatbot employs a user-centric approach, guiding users through a conversational interface that feels natural and tailored to their understanding.

Personalized Recommendations:

By extracting indirect information, the chatbot can provide personalized recommendations, offering laptops that align with the user's preferences, needs, and budget.

Reduced Decision Complexity:

Indirect questioning simplifies the decision-making process by breaking down complex choices into more digestible inquiries, making it accessible for users with varying levels of technological expertise.

By leveraging a chatbot with indirect questioning strategies, users can receive tailored recommendations that address their unique requirements, making the laptop selection process more informed and user-friendly.

1.5 Project Phases

A. Research
B. Analysis
C. Prototyping
D. Implementation
E. Development

CHAPTER 2

2. Related Work

NO.	Year	Algorithm	Accuracy	Dataset
Paper 1	2020	**KNN** &SVM	97.25%	Amazon product data
Paper 2	2021	NLP	94.36%	Laptop Ranke Specifications
Paper 3	2022	classification	91.21%	Laptop Dataset
Paper 4	2021	NLP	86.26%	Laptops Info
Paper 5	2021	Content-Based	91.65%	Laptops Info
Paper 6	2021	Content &collaborative-Based	89.51%	Brand Laptops Dataset
Paper 7	2020	KNN	87.66%	Brand Laptops Dataset
Paper 8	2020	Content-based Filtering &KNN	82.26%	Amazon Reviews

Paper 9	2022	Content-based Filtering & K-means	93.33%	Amazon Canada Products
Paper 10	2020	Content-based &collaborative &Hybrid Filtering	89.66%	Amazon Electronics Products

2.1 Paper 1

Abstract

In recent years, the concept of virtual assistants or chatbots has become commonplace. The leading technology corporations have officially released their virtual assistants such as Cortana of Microsoft, Siri of Apple, and Google Assistant of Google. There are several available tools for building chatbots such as Chat fuel, Mess now, Chatty People and Many Chat. However, these tools may have advertisements and do not support many languages, including the Vietnamese language.

A chatbot can be constructed using the following approaches: using retrieval-based or generative models, supporting short or long conversations, and in closed or open domains. Bots created using the generative model can answer questions which are not in the training dataset, but these answers might be in wrong syntax or have misspelling; whereas bots created using the retrieval-based model can respond answers with correct grammar and spelling, only for questions in the training dataset. The lengths of conversations also affect bots. The longer the text conversations are, the harder it is to construct the bots. Bots constructed in an open domain require a large amount of knowledge to

answer unrestricted questions; whereas bots built in a closed domain can answer questions in that specific domain. Although there are a variety of methods for constructing a chatbot, each method for building chatbot systems needs to handle the following issues: classifying, determining, and extracting intents that users express. In addition, a smart chatbot needs to handle acronyms and misspelled words

Dataset

Amazon product data

Algorithms

KNN &SVM

2.2 Paper 2

Abstract

Chatbot is an application where the computer talks with the user like an human. The software can understand the discussion it is having with the user and respond accordingly.

Chatbot uses the concepts of Natural Language Processing (NLP) and Artificial Intelligence to achieve this task. Chatbots can be useful in various applications. It can be used as a support source for answering questions in

various domains, it can be used as a guide during learning languages and in a User Interface Agent for applications.

The knowledge base is an important part of an Chatbot application. The knowledge base should store all the information of the domain in which the Chatbot is operating.

It should be able to provide information to the application to answer questions asked by the user. Chatbot is used as a guide in language learning can correct the grammatical mistakes done by user in communication with chatbot.

Chatbot will have knowledge stored of grammatically correct sentences. On communication of the user with Chatbot, it will check the correctness of the statement and inform user. Index Terms— Artificial Intelligence, Chatbot, Natural Language Processing.

Algorithms

NLP

Datasets

Laptop Ranke Specifications

2.3 Paper 3

Abstract

This paper synthesizes research on artificial intelligence (AI) in e-commerce and proposes guidelines on how information systems (IS) research could contribute to this research stream.

To this end, the innovative approach of combining bibliometric analysis with an extensive literature review was used. Bibliometric data from 4335 documents were analyses, and 229 articles published in leading IS journals were reviewed. The bibliometric analysis revealed that research on AI in e-commerce focuses primarily on recommender systems. Sentiment analysis, trust, personalization, and optimization were identified as the core research themes.

It also places China based institutions as leaders in this researcher area. Also, most research papers on AI in e-commerce were published in computer science, AI, business, and management outlets. The literature review reveals the main research topics, styles and themes that have been of interest to IS scholars. Proposals for future research are made based on these findings.

This paper presents the first study that attempts to synthesis research on AI in e-commerce. For researchers, it contributes ideas to the way forward in this research area. To practitioners, it provides an organized source of information on how AI can support their e-commerce endeavors.

Algorithms

Classification

Datasets

Laptop Dataset

2.4 Paper 4

Abstract

For software applications, user interfaces that can be used include command line, graphical user interface (GUI), menu driven, form-based, natural language, etc. The mainstream user interfaces include GUI and web-based, but occasionally the need for an alternative user interface arises.

A chatbot based conversational user interface fits into this space. The chatbot is a class of bots that have existed in the chat platforms.

The user can interact with them via graphical interfaces or widgets, and the trend is in this direction. They generally provide a stateful service i.e. the application saves data for each session.

On a college's website, one often doesn't know where to search for some kind of information. It becomes difficult to extract information for a person who is not a student or employee there.

The solution to these comes up with a college inquiry chat bot, a fast, standard, and informative widget to enhance college website's user experience and provide effective information to the user.

Chat bots are an intelligent system being developed using artificial intelligence (AI) and natural language processing (NLP) algorithms.

It has an effective user interface and answers the queries related to examination cell, admission, academics, users' attendance and grade point average, placement cell and other miscellaneous activities.

Algorithms

NLP

Datasets

Laptops Info

2.5 Paper 5

Abstract

Recommendation systems are a popular and beneficial field that can help people make informed decisions automatically.

This technique assists users in selecting relevant information from an overwhelming amount of available data.

When it comes to movie recommendations, two common methods are collaborative filtering, which compares similarities between users, and content-based filtering, which takes a user's specific preferences into account.

However, our study focuses on the collaborative filtering approach, specifically matrix factorization. Various similarity metrics are used to identify user similarities for recommendation purposes.

Our project aims to predict movie ratings for unwatched movies using the Movie Lens rating dataset. We developed a model that uses a matrix factorization algorithm to predict ratings and recommends movies with the highest predicted ratings to users.

The achieved values of RMSE were relatively low, indicating that the system was able to provide accurate recommendations for users.

Algorithms

Content- Based & collaborative- Based.

Datasets

Laptops Info

2.6 Paper 6

Abstract

Abstract Intelligent search is a search with the possibility of linguistic analysis, modern algorithms for parsing and finding words, recommendations based on user preferences. Such a search is a necessity for Internet resources in the field of e-commerce because it is almost impossible to find the right product or data without some help, filtering, or sorting.

One can highlight the following features of the search: spell check (if the search query is misspelt, the search for relevance will try to find close matches based on what was entered); recognition of standard abbreviations and acronyms (this search will be able to recognize standard abbreviations or acronyms, such as MK (Michael Kors), NASA (National Aeronautics and Space Administration)); better understanding (a search for relevance will try to understand the query better, applying well-known knowledge, recognizing frequently used synonyms and limited knowledge of the natural language); filters and sorting (the use of filters by specific categories and sorting will allow to more comfortably and quickly find the necessary information).

The latter will come in handy for almost any type of online resource, from online stores to entertainment sites. Moreover, the paper highlights the topic of recommendation systems. Over the past few decades, with the growth of YouTube, Amazon, Netflix and many other such web services, referral systems have become more and more important in our lives.

Recommendation systems are critical in some areas because they can generate huge revenue or stand out significantly from competitors.

The paper considers the basic methods and approaches used to build search engines and recommendation systems. The implementation of these approaches on examples and natural systems has been considered. A web application based on Java and Elasticsearch has been developed with the performance of a recommendation system based on a collaborative filtering algorithm. The research object is an intelligent search information system with the possibility of recommendations in e-commerce.

The subject of research is the basic principles and requirements for the construction of recommendation systems and intelligent search systems. The study aims to develop a fast and reliable search engine in e-commerce with the possibility of recommendations for users.

Algorithms

Content &collaborative-Based

Datasets

Brand Laptops Dataset

2.7 Paper 7

Abstract

Information technology and the popularity of mobile devices allow for various types of customer data, such as purchase history and behavior patterns, to be collected.

As customer data accumulates, the demand for recommender systems that provide customized services to customers is growing. Global e-commerce companies offer recommender systems to gain a sustainable competitive advantage. Research on recommender systems has consistently suggested that customer satisfaction will be highest when the recommendation algorithm is accurate and recommends a diversity of items.

However, few studies have investigated the impact of accuracy and diversity on customer satisfaction. In this research, we seek to identify the factors determining customer satisfaction when using the recommender system.

To this end, we develop several recommender systems and measure their ability to deliver accurate and diverse recommendations and their ability to generate customer satisfaction with diverse data sets. The results show that accuracy and diversity positively affect customer satisfaction when applying a deep learning-based recommender system.

By contrast, only accuracy positively affects customer satisfaction when applying traditional recommender systems. These results imply that developers or managers of recommender systems need to identify factors that further improve customer satisfaction with the recommender system and promote the sustainable development of e-commerce.

Algorithms

K-Nearest Neighbor (KNN)

Datasets

Brand Laptops Dataset

2.8 Paper 8

Abstract

E-Commerce product features and reviews are the essential factors in real-time e-commerce sites for product recommendation systems. Due to inaccurate decision patterns, in most cases e-commerce users fail to predict the products based on the user ratings and review comments.

Traditional sentiment classification models are independent of data filtering, transformation and sentiment score computing techniques which require high computing memory, time and mostly leading to false-positive rate.

To overcome these issues, a novel sentiment score-based product recommendation model is proposed on the real-time product data.

In this model, a new product ranking score, filtering, and hybrid decision tree classifiers are implemented. Initially, real-time amazon product review data is captured using Document Object Model (DOM) parser. The features from the review comments are extracted using lexicon Feature Dictionary (FD) and AFINN, Normalized Product Review Score (NPRS) are generated to compute the class label for product review sentiment prediction.

Ranked Principal Component Analysis (RPCA) is used as a feature selection measure to overcome the problem of data sparsity. Random Tree, Hoeffding Tree, Adaboost + Random Tree, the three variants of decision tree classifiers are used for product sentiment classification.

The proposed filter-based improved decision tree sentiment classification model for real-time amazon product review data recommends the product based on the user query by prediction using a new novel normalized product review sentiment score and ranked feature selection measure.

The proposed product recommendation, the decision-making system maximizes sentiment classification accuracy. Experimental results are compared against the traditional decision-making classification models in terms of correctly classified instances, error rate, and PRC, F-measure, kappa statistics. The proposed model experimental results show high efficiency.

Algorithms

Content-based Filtering & K-Nearest Neighbor (KNN)

Datasets

Amazon Reviews 2018 (Full Dataset)

2.9 Paper 9

Abstract

This paper offers a detailed explanation of a system that uses sentiment analysis and machine learning algorithms to classify and recommend products on Amazon. Using the idea of Machine Learning, we developed a system that can be used by many e-commerce sites for better product recommendations.

This system employs a machine learning model in which similar and superior products are offered to the customers in order of best to worst based on the product utilized in the past.

The computer will compile a shortlist of all relevant items or products based on user-generated product reviews that meet the user's criteria, considering the product's quality and rating.

The approach we employed was to create a system model that would analyze customer reviews for various products in the same category and then use Natural Language Processing to arrive at a conclusion where the system (model) would be able to assess whether the review is positive or negative.

We've also used the ratings offered on various items to create a technique to combine ratings and reviews to improve the accuracy of the system (model). We employed the Collaborative Algorithm to improve the accuracy of product recommendations.

During the creation of the system, we used the Amazon e-commerce site and its products to simulate a real-world implementation scenario (model).

Our system uses cosine similarity to find the similarities between items on basis of the multiple user's ratings and form a matrix which helps to recommend items to other users.

Algorithms

Content-based Filtering & K-means

Datasets

Amazon Canada Products 2023 (2.1M Products)

2.10 Paper 10

Abstract

The Internet is changing the method of selling and purchasing items. Nowadays online trading replaces offline trading.

The items offered by the online system can influence the nature of buying customers. The recommendation system is one of the basic tools to provide such an environment. Several techniques are used to design and implement the recommendation system.

Every recommendation system passes from two phases similarity computation among the users or items and correlation between target user and items. Collaborative filtering is a common technique used for designing such a system.

The proposed system uses a knowledge base generated from knowledge graph to identify the domain knowledge of users, items, and relationships among these, knowledge graph is a labelled multidimensional directed graph that represents the relationship among the users and the items.

Almost every existing recommendation system is based on one of feature, review, rating, and popularity of the items in which users' involvement is very less or none. The proposed approach uses about 100 percent of users' participation in the form of activities during navigation of the web site. Thus, the system expects under the users' interest that is beneficial for both seller and buyer.

The proposed system relates the category of items, not just specific items that may be interested in the users. We see the effectiveness of this approach in comparison with baseline methods around recommendation system using three parameters precision, recall, and NDCG through online

and offline evaluation studies with user data, and its performance is better than all other baseline systems in all aspects.

Algorithms

Content- based &collaborative &Hybrid Filtering

Datasets

Amazon Electronics Products

CHAPTER 3

3. Background

3.1 website

A website, often referred to simply as a site, is a collection of web pages accessible over the internet. Unlike mobile applications which are tailored for specific devices, websites are designed to be accessed through web browsers on various platforms including personal computers, smartphones, and tablets.

Websites serve a diverse range of purposes, offering users access to information, services, entertainment, and more. They are composed of interconnected pages containing different types of content such as text, images, videos, links, and interactive elements.

Initially, websites were static and provided limited interactivity. However, with advancements in web technologies, modern websites offer dynamic and immersive experiences. They utilize languages like HTML, CSS, and JavaScript to create visually appealing layouts, responsive designs, and interactive features.

Websites are hosted on web servers and can be accessed by users worldwide. They provide businesses, organizations, and individuals with a platform to showcase their products, services, ideas, or creations to a global audience.

Just as mobile applications have their own ecosystems such as the App Store, websites also have platforms for hosting and distribution, such as web hosting services and content management systems. These platforms

enable website owners to manage, update, and optimize their sites efficiently.

In summary, websites are essential components of the digital landscape, offering users access to a vast array of information and services. They continue to evolve alongside technology, providing increasingly sophisticated and engaging experiences to users across the globe.

3.2 Frontend

Frontend, also known as the client-side, is a component of software development that focuses on designing and developing the user interface and user experience. It includes everything that users directly interact with when using a web or software application.

This encompasses visual design, Hypertext Markup Language (HTML) structure, Cascading Style Sheets (CSS) styling, JavaScript programming, and any frameworks or libraries used to facilitate the development process, such as Bootstrap.

3.2.1 Components

HTML (Hypertext Markup Language):

Used to build the page structure and define elements such as paragraphs, images, and links.
CSS (Cascading Style Sheets):

Utilized for formatting and styling the page by specifying colors, fonts, and margins.

JavaScript:

A programming language is used to add dynamic interactivity to the page, providing dynamic styles and effects.

Bootstrap:

A framework that offers a set of design components and tools to expedite and simplify frontend development.

Figma:

A collaborative design tool used for professionally creating UI/UX designs and prototypes in a team environment.

3.2.2 Development Process

Design:

Frontend development begins with the design phase, where UI/UX designers create models to guide the development process.
Coding:

Developers translate the design into reality using HTML, CSS, JavaScript, and Bootstrap.

Testing:

Thorough testing is conducted to ensure good performance and provide a seamless, responsive user experience.
Integration with Backend:

The front end is integrated with the backend to enable interaction with databases and server-side components.
Visual and Interactive Enhancement:

Continuous refinement of visual and interactive aspects to ensure an excellent user experience.
Frontend Benefits:

Enhanced User Experience:
Frontend contributes to creating and improving the user experience through an attractive and comfortable interface.

Responsiveness and Flexibility:
The use of Bootstrap and responsive design enables the frontend to be compatible with various devices.

Accelerated Development Process:
Bootstrap and design tools like Figma help expedite and simplify the frontend development process.

3.3 Backend

Backend, also known as the server-side, is a component of software development that focuses on the non-visible aspects that manage and handle processes behind the scenes. PHP (Server-side Programming Language) is commonly used in backend development.

Components:
Server-side Programming Language (PHP):

Used to execute server-side programs and generate dynamic pages on the server.
Database:

Used to store and retrieve data, including Database Management Systems such as MySQL and PostgreSQL.
Other Programming Languages:

In addition to PHP, other programming languages like Python, Ruby, and Node.js can be used for backend development.
Web Servers:
Web servers such as Apache and Nginx are used to deploy and run software applications on the server.

Server Programming: Involves developing and maintaining software that deals with requests from the client and manages the core logic of the application.

3.3.1 Development Process

Software Development: Involves designing and implementing server-side programs and dealing with databases.

Server Testing: Server software is tested to ensure good responsiveness and application stability.

Performance Optimization: Backend performance is optimized to ensure quick response and avoid common errors.

Integration with Frontend: Backend is integrated with the frontend to achieve smooth interaction between the frontend and backend of the application.

Regular Maintenance: Involves periodic maintenance, security updates, and bug fixes to ensure service continuity.

3.4 Backend Benefits

Data Management: Data management involves organizing, storing, retrieving, and manipulating data systematically. In software development,

especially in web development, data management relies on the backend system to handle these operations on databases.

The backend, or server-side, of a web application manages the interaction between the user interface (frontend) and the database. One of its primary roles is to facilitate data retrieval and storage from and to the database.

Key aspects of data management in the backend include:

Data Retrieval: The backend retrieves data from the database based on user requests or system needs by executing database queries.

Data Storage: It securely stores data in databases, handling insertions, updates, and deletions to maintain data integrity.

Database Operations: Backend developers implement logic to perform various database operations efficiently, such as query optimization and transaction handling.

Data Modeling: They design database schemas and models to represent data structure and relationships.

Data Validation and Sanitization: Backend systems validate and sanitize user input to prevent malicious or erroneous data from entering the database.

Caching: Backend systems use caching mechanisms to temporarily store frequently accessed data in memory to improve performance and reduce database load.

Scaling: Backend systems need to be scalable to handle increased data volume and application usage, which may involve horizontal or vertical scaling.

Effective data management in the backend ensures the reliability, performance, and security of web applications. By implementing robust data handling mechanisms, developers create applications that efficiently manage and leverage data to meet user needs and business objectives.

3.4.2 Execution of Server-side Programs

Execution of server-side programs involves the handling and processing of requests received from client-side applications or users. These programs run on the server and are responsible for performing various tasks, such as data processing, business logic execution, and generating dynamic content to be sent back to the client.

Here are the key aspects of the execution of server-side programs:

Receiving Requests: Server-side programs listen for incoming requests from client-side applications or users. These requests typically include information about the action to be performed and any data associated with it.

Routing: Once a request is received, the server-side program routes it to the appropriate handler based on the request URL or other parameters. Routing ensures that the request is directed to the correct function or module for processing.

Processing Requests: The server-side program processes the received requests by executing the necessary business logic or performing database operations. This may involve querying databases, accessing external APIs, performing calculations, or any other tasks required to fulfill the request.

Generating Responses: After processing the requests, the server-side program generates dynamic content or data to be sent back as a response to the client. This could include HTML content, JSON data, files, or any other type of data based on the nature of the request.

Sending Responses: Finally, the server-side program sends the generated response back to the client over the network. This response is typically formatted according to the HTTP protocol and includes relevant headers and status codes to indicate the outcome of the request.

Error Handling:
Server-side programs also handle errors and exceptions that may occur during request processing. This involves detecting errors, logging relevant information, and sending appropriate error responses back to the client.

Scalability and Performance: Server-side programs should be designed to handle high volumes of requests efficiently and to scale horizontally or vertically as needed to accommodate increasing traffic or resource demands.

Overall, the execution of server-side programs plays a crucial role in the functioning of web applications by processing requests, executing business logic, and generating dynamic content to provide users with the desired functionality and user experience.

3.4.3 Performance Improvement

Performance improvement in software development refers to the process of optimizing the speed, efficiency, and resource utilization of a system or application. This optimization aims to enhance the overall user experience,

reduce latency, and increase throughput. Here are some strategies for performance improvement:

Code Optimization: Review and optimize code to make it more efficient. This includes identifying and refactoring algorithms, reducing unnecessary computations, and eliminating code duplication.

Database Optimization: Optimize database queries and schema design to improve data retrieval and storage efficiency. This may involve indexing frequently accessed columns, deformalizing data, and using appropriate database engines or caching mechanisms.

Caching: Implement caching mechanisms to store frequently accessed data in memory. This reduces the need to fetch data from disk or external sources, improving response times and reducing server load.

Concurrency and Parallelism: Utilize concurrency and parallel processing techniques to execute tasks concurrently and utilize multi-core processors efficiently. This can be achieved through asynchronous programming, threading, or distributed computing frameworks.

Network Optimization: Optimize network communication by minimizing the size of data payloads, reducing the number of network requests, and leveraging compression and caching techniques. This reduces latency and improves overall responsiveness.

Resource Management: Efficiently manage system resources such as memory, CPU, and disk I/O to prevent resource contention and bottlenecks. This includes optimizing resource allocation, garbage collection strategies, and minimizing resource leaks.

Performance Monitoring and Profiling: Continuously monitor application performance using profiling tools and metrics to identify performance bottlenecks and areas for improvement. This allows developers to prioritize optimization efforts effectively.

Load Testing and Scalability: Conduct load testing to simulate real-world usage scenarios and identify performance limits. Design applications to be scalable, allowing them to handle increased load by adding resources or scaling horizontally.

UI and Frontend Optimization: Optimize frontend assets such as images, scripts, and stylesheets to reduce page load times and improve rendering performance. This includes minification, compression, and lazy loading techniques.

Optimization Iteration: Performance optimization is an iterative process. Continuously measure, analyze, and optimize different aspects of the system to achieve incremental improvements over time.
By applying these performance improvement strategies, developers can enhance the efficiency and responsiveness of software systems, resulting in a better user experience and higher overall system performance.

3.4.4 Integration with Frontend

Integration with frontend refers to the process of connecting and coordinating the backend and frontend components of a web application. This integration ensures seamless communication between the server-side and client-side of the application, enabling the exchange of data, requests, and responses. Here are the key aspects of integration with

Frontend:

API Design: Backend developers design and implement APIs (Application Programming Interfaces) that define how frontend components can interact with backend services. APIs specify endpoints, request formats, response structures, and authentication mechanisms.

RESTful Services: REST (Representational State Transfer) is a commonly used architectural style for designing web APIs. RESTful services enable stateless communication between the frontend and backend through standard HTTP methods (GET, POST, PUT, DELETE) and resource-based URLs.

Data Transfer Formats: Define data transfer formats such as JSON (JavaScript Object Notation) or XML (extensible Markup Language) for exchanging data between the frontend and backend. JSON is widely preferred due to its simplicity and compatibility with JavaScript.

AJAX (Asynchronous JavaScript and XML): Utilize AJAX to perform asynchronous requests from the frontend to the backend without reloading the entire web page. AJAX enables dynamic content updates and improves user experience by fetching data in the background.

Frontend Frameworks: Integrate frontend frameworks such as Angular, React, or Vue.js with the backend to create interactive and responsive user interfaces. These frameworks facilitate the development of single-page applications (SPAs) that communicate with the backend via APIs.

Authentication and Authorization: Implement authentication and authorization mechanisms to secure backend resources and endpoints. Use techniques like JSON Web Tokens (JWT) or session-based authentication to validate user identity and permissions.

Cross-Origin Resource Sharing (CORS): Configure CORS policies on the backend to allow or restrict access to resources from different origins (domains). CORS prevents unauthorized cross-origin requests and ensures secure communication between frontend and backend components.

Error Handling: Define error handling mechanisms to communicate error messages and status codes from the backend to the frontend. Handle errors gracefully on the frontend to provide informative feedback to users and maintain application stability.

Testing and Debugging: Test integration between frontend and backend components thoroughly to ensure compatibility, reliability, and performance. Use tools like Postman, Jest, or Selenium for automated testing and debugging.

Continuous Integration and Deployment (CI/CD): Implement CI/CD pipelines to automate the integration, testing, and deployment of frontend and backend changes. Continuous integration ensures that updates to both components are synchronized and deployed smoothly.

By effectively integrating frontend and backend components, developers can create cohesive and interactive web applications that deliver a seamless user experience and meet business requirements.

3.5 Chatbot design stages

A comprehensive guide to chatbot design: from idea to implementation:

In the age of modern technology, digital and interactive applications have become an integral part of our daily lives. Among these applications, Chatbot takes an important place in providing services and improving the user experience across different chat platforms. Designing a chatbot can be an exciting challenge, but with an understanding of the basic foundations and the right steps, anyone can begin the journey of designing an effective and useful chatbot.

The basic principles of chatbot design:

Understanding user needs: How to identify needs and problems that can be solved by a chatbot.
Conversational design: How to design dialogue with a chatbot effectively and attractively.
Artificial Intelligence and Machine Learning: The role of technology such as artificial intelligence and machine learning in improving chatbot performance and making it more intelligent and effective.

How was the chatbot design process done?

Defining closed questions well is a crucial step in chatbot design, as it helps in understanding users' needs and directing them towards appropriate solutions. These questions should be directed directly towards the main aspects of the product to be offered, such as the laptop in this case. Here are some points that can be discussed in detail in this context:

Available budget:

A question about the financial budget that the user intends to allocate to purchase the laptop.
Multiple budget options can be provided, such as "under $1000", "$1000-1500", "more than $1500".

Intended use of the device:

A question about the user's expected use of the laptop, whether for daily use at work, for gaming, or for browsing and entertainment.
This question helps determine the required technical specifications, such as processor power, memory size, and screen size.

the required specifications:

A question about the user's preferred technical specifications, such as screen size, processor type, storage capacity, memory size, and battery range.
A list of options can be provided for each specification, or the user allowed to specify more details.

Favorite brand:

A question about the user's favorite brands if he prefers a specific brand of laptop.

The user may have specific preferences based on their previous experiences or product reviews.

A question about any additional features the user prefers, such as a camera, a USB-C port, or an illuminated keyboard.
Options can also be offered for personal preferences such as device color or case design.
Using these questions, the chatbot can gather accurate information about users' needs and suggest suitable laptops based on their personal requirements and preferences.

Building a chatbot:

We used the JavaScript language to develop and build the chatbot, and then we designed and programmed the chatbot to ask questions and record answers from users. The initialization part of the code sets up the basic structure for an interactive questionnaire or quiz application. Here's a breakdown of each step:

Defining Questions Array:

An array named `questions` is created to hold objects representing each question along with its choices. Each question object contains properties for the question text and an array of choices.

Initializing Variables: Two variables are initialized:

- `questionsAnswers`: This variable will store the user's answers.
- `currentQuestionIndex`: This variable keeps track of the index of the current question being displayed.

Getting References to HTML Elements: References to HTML elements where the chat messages and choices will be displayed are obtained. These elements will be manipulated dynamically to update the interface as the user progresses through the questions.

Next, let's delve into the functions:

- `displayQuestion` Function:

  - Retrieves the current question object from the `questions` array based on the `currentQuestionIndex`.
  - Constructs HTML elements dynamically to display the question text and choices as clickable links or buttons.

  - Appends this HTML to the chat window or message container, allowing the user to view and interact with the question and choices.

```
const questions = [
    { question: "What price do you prefer?", choices: ["Less than 15000", "15000-25000", "25000-50000","More than 50000"] },
    { question: "Choose the company you prefer", choices: ["Apple", "HP", "Lenovo","Dell","Don't prefer any company"] },
    { question: "Do you know what RAM is and how it affects the performance of the device?", choices: ["Yes","No"] },
    { question: "What is the suitable RAM capacity for you?", choices: ["4GB", "8GB", "16GB ","32GB"] },
    { question: "Do you use heavy applications, Photoshop, or programming primarily?", choices: ["Yes","No"] },
    { question: "Do you prefer a specific operating system?", choices: ["Yes","No"] },
    { question: "What is your preferred operating system?", choices: ["Windows", "Mac OS", "Chrome OS", "No preference"] },
    { question: "What screen size do you prefer?", choices: ["Small", "Medium", "Large"] },
    { question: "What will you use the laptop for?", choices: ["Browsing", "Work", "Gaming or graphic editing"] },
    { question: "Do you prefer that the laptop has a touch screen?", choices: ["Yes","No"] },
];
```

Figure 1

- `saveAnswer` Function:
  - This function is a placeholder intended to handle the saving of user answers to a server-side script for further processing and storage.

```
function saveAnswer(answers) {
    // Use AJAX to send the answer to a PHP script for processing and database storage
    const xhr = new XMLHttpRequest();
    xhr.open('POST', 'save_answer.php', true);
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhr.send(`answers=${answers}`);
}
```

Figure 2

- It sets up an XMLHttpRequest object to send the user's answers to a server-side script (in this case, a PHP script) via AJAX.

- The function sends the answers to the server-side script for database storage or other operations.

- `selectChoice` Function:
- This function is triggered when a user selects a choice for a question.

- It adds the selected choice along with the index of the current question to the `questionsAnswers` array.

- Updates the chat interface to display the user's choice and hides the current question.

- Disables further interaction with choices for the current question to prevent multiple selections.

- If there are more questions, it increments the `currentQuestionIndex` and calls `displayQuestion` again to move to the next question. Otherwise, it proceeds to the end of the chat.
At the end of the chat:

- The code encodes the "questionsAnswers" array using Base64 and sends it to the `saveAnswer` function for further processing.
- The choices container is cleared, and a final message is displayed thanking the user for participating in the questionnaire or quiz.

This last demo presents the user with a series of questions designed to understand their needs. After collecting the answers, the chatbot will analyze them and make suitable suggestions for laptops according to the user's preferences.

Great! Now, could you please tell me if you have any preference for a laptop brand?

Preferred Brand:
Which laptop brand do you prefer?
Apple
HP
Lenovo
Dell
No preference

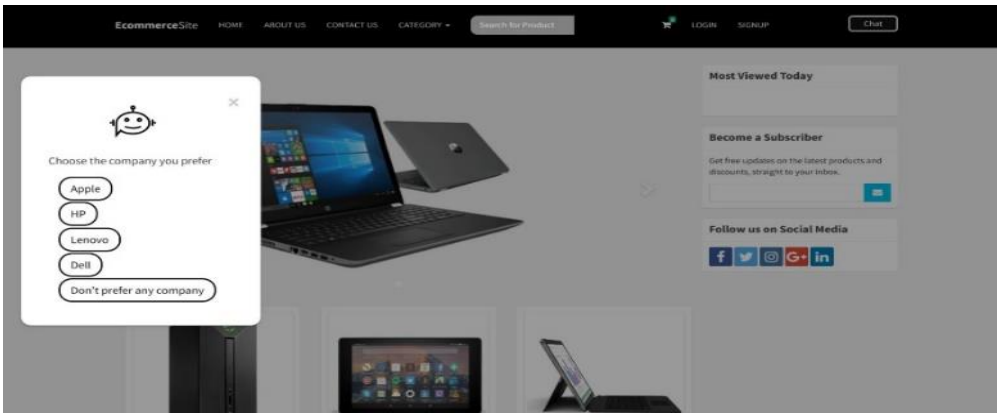


Figure 3

Thus, a successive series of questions is conducted for the user to get the best suggestion.

Until we finally reach the best suggestion based on the user's answers:

Based on the user answers provided through the questions asked by the chatbot, here's how it makes the best laptop suggestion:

Yes, I will start by asking you questions so we can determine which laptop best suits your needs and budget. Please answer the following questions:

1. What is your budget range?

- Less than $500
- Between 500 and 1000 dollars
- Between 1000 and 1500 dollars
-More than $1500

2. What is the primary purpose of using a laptop?

- Browsing the Internet and using office applications
- the games
- Graphic design and video editing
- Programming and development
- Multi-purpose use

3. How important is battery life to you?

- not important
- Average importance
- Important
- very important

4. What screen size do you prefer?

- Less than 13 inches
- Between 13 and 15 inches
- Between 15 and 17 inches
-More than 17 inches

5. Do you prefer a specific operating system?

- Windows
- Mac
-Linux
- No specific best operating system

6. How important is weight to you?

- not important
- Average importance
- Important
- very important

7. Do you need specific specifications?

- Powerful processor (Intel i7/i9 or AMD Ryzen 7/9)
- Large RAM (16 GB or more)
- Powerful graphics card (NVIDIA or AMD)
- Large storage space (SSD)

Based on your answers to these questions, the system can filter out a group of laptops that fit your needs and budget. Would you like to answer these questions now so that the system can provide specific recommendations?

Figure 4

CHAPTER 4

4. Machine Learning (ML)

is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and techniques that enable computers to learn from and make predictions or decisions based on data. The primary goal of machine learning is to develop models that can generalize patterns from past data to make accurate predictions or decisions on new, unseen data.

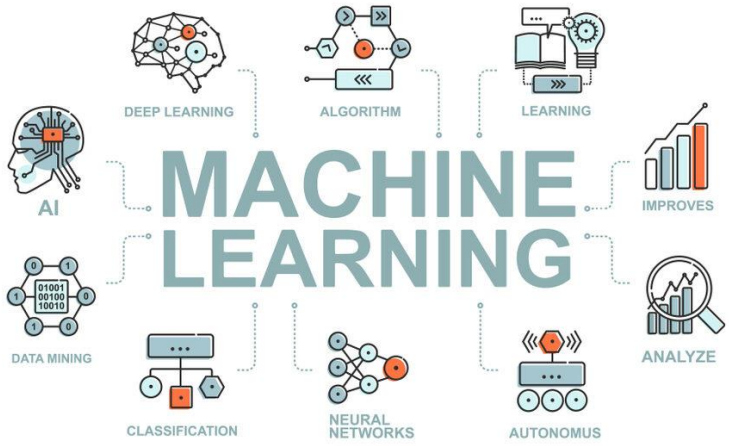


Figure 5

4.1Here's a detailed overview of machine learning:

4.1.1 Learning from Data

Learning from data is the fundamental concept of machine learning. Instead of manually programming computers with specific instructions to perform tasks, machine learning algorithms are trained to recognize patterns and relationships directly from data. This process involves

iteratively processing large datasets, extracting meaningful insights, and identifying patterns that can then be used to make predictions or decisions.

By leveraging machine learning, computers can automatically improve their performance on a task as they are exposed to more data. This ability to learn from data enables machines to handle complex problems and tasks that may be difficult or impossible to solve with traditional rule-based programming.

Machine learning algorithms come in various forms, such as supervised learning, unsupervised learning, and reinforcement learning, each suited for different types of tasks and data. Regardless of the specific algorithm used, the underlying principle remains the same: learning from data to extract valuable insights and make informed decisions.

4.2 Types of Machine Learning

Machine learning can be broadly categorized into three main types:

Supervised Learning:

In supervised learning, the algorithm learns from labeled data, meaning each input data point is associated with a corresponding target label. The algorithm aims to learn a mapping function from the input variables to the output variable, based on the labeled training data. Common supervised learning algorithms include linear regression, logistic regression, support vector machines, decision trees, and neural networks.

Unsupervised Learning:

Unsupervised learning involves training algorithms on data without labeled responses. The algorithm tries to learn the underlying structure or distribution in the data. Clustering and dimensionality reduction are common tasks in unsupervised learning. Examples of unsupervised learning algorithms include K-means clustering, hierarchical clustering, principal component analysis (PCA), and autoencoders.

Reinforcement Learning:

In reinforcement learning, an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on its actions, and its goal is to learn the optimal strategy or policy to maximize cumulative reward over time. Reinforcement learning is often used in scenarios where an agent must make sequential decisions, such as game playing, robotics, and autonomous driving.

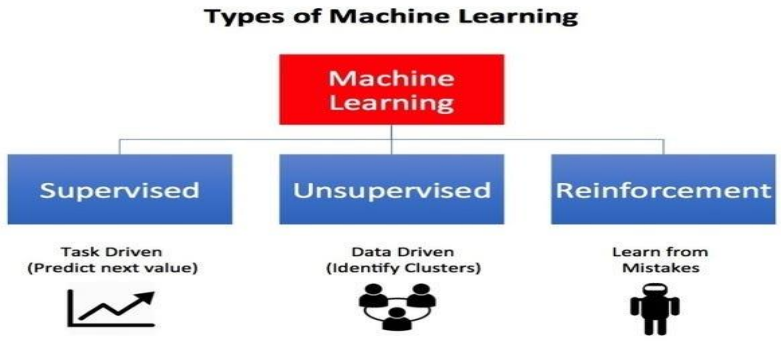


Figure 6

4.2.1 Supervised Learning

certainly! In supervised learning, the algorithm is provided with a data set consisting of input-output pairs, where the input is typically represented by features, and the output is the target score or value. For example, if you wanted to build a system to classify images into different categories such as “dog” or “cat,” the images would be the input data and the categories “dog” or “cat” would be the output.

During the training phase, the algorithm uses the given data set to learn the relationship between features and target labels. It tries to understand how the sign changes based on the features provided. For example, in an image classification problem, an algorithm can learn the distinctive features of each class of images such as shapes, colors, and textures, and how to represent these features mathematically to differentiate between different classes.

After the training process is complete, the trained model can be used to make predictions or classifications on new data that the algorithm has not seen before. When a model is used to classify new data, it takes features as input and generates an output that makes a classification or prediction based on the relationship it has learned from the training data.

Essentially, supervised learning relies on the ability to represent complex relationships between features and target labels, allowing the model to achieve good performance when dealing with new, previously unknown data.

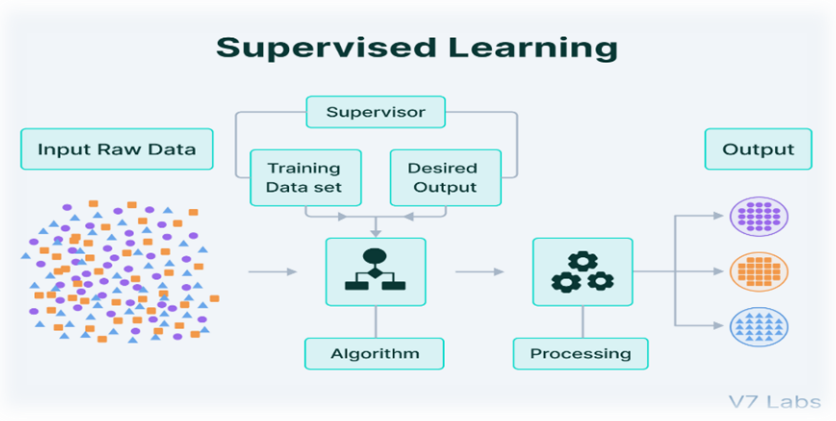


Figure 7

4.2.2 Unsupervised Learning

In unsupervised learning, algorithms explore data without needing guidance from training data. Instead, these models aim to discover hidden patterns and important analyzes within the data provided. This can help in better understanding data and extracting valuable information from it, like how the human brain learns while acquiring new knowledge.

Simply put, unsupervised learning is a type of self-learning, where algorithms can identify latent patterns in unlabeled data sets and provide appropriate output without interference from outside.

Finding these underlying patterns can help group data into groups that share common characteristics, discover relationships between data, and uncover anomalies and errors in the groups.

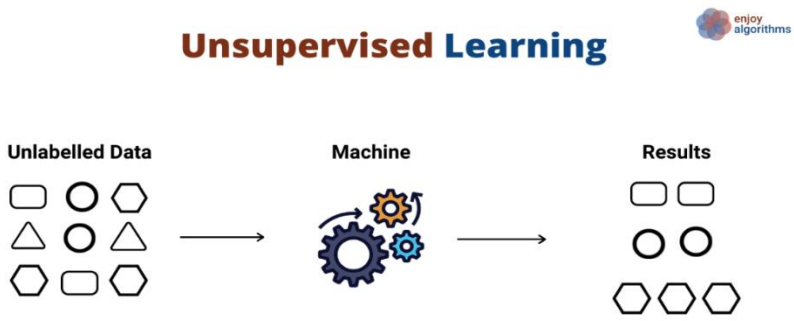


Figure 8

4.2.3 Reinforcement Learning

Reinforcement learning (RL) is a machine learning (ML) technique that trains software to make decisions to achieve the most optimal results. It mimics the trial-and-error learning process that humans use to achieve their goals. Software actions that work towards your goal are reinforced, while actions that detract from the goal are ignored.

RL algorithms use a reward-and-punishment paradigm as they process data. They learn from the feedback of each action and self-discover the best processing paths to achieve final outcomes. The algorithms are also capable of delayed gratification. The best overall strategy may require short-term sacrifices, so the best approach they discover may include some punishments or backtracking along the way. RL is a powerful method to help artificial intelligence (AI) systems achieve optimal outcomes in unseen environments.

CHAPTER 5

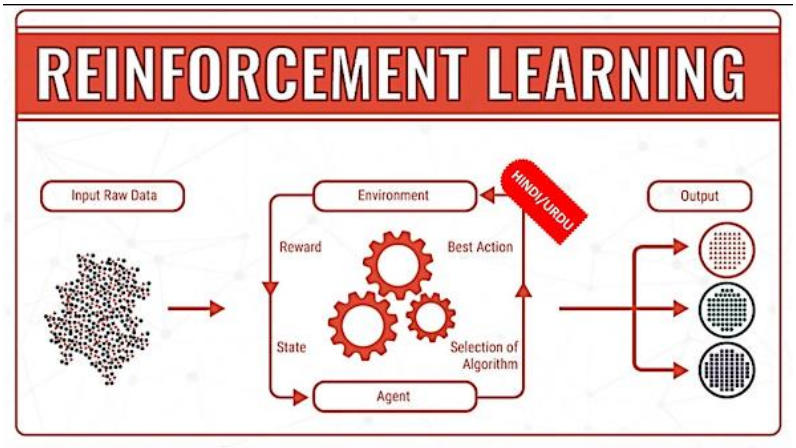


Figure 9

5. Recommendation System

A recommendation system is a software tool or algorithm that analyzes user data and item characteristics to provide personalized suggestions or recommendations to users. These recommendations are tailored to individual preferences and help users discover relevant items they may be interested in. Recommendation systems are widely used in various domains, including e-commerce, social media, content streaming platforms, and more. They aim to enhance user experience, increase engagement, and drive user satisfaction and loyalty.

Recommendation systems utilize various techniques and algorithms to generate recommendations, including collaborative filtering, content-based filtering, and hybrid approaches that combine multiple methods. These systems typically analyze user interactions such as purchases, ratings, clicks, and browsing history, as well as item attributes such as genre, category, or features, to understand user preferences and item characteristics.

Based on this analysis, recommendation systems generate personalized recommendations for users, suggesting items that they are likely to find interesting or relevant. These recommendations can take various forms, including product recommendations on e-commerce websites, movie or music suggestions on streaming platforms, friend suggestions on social media platforms, and more.

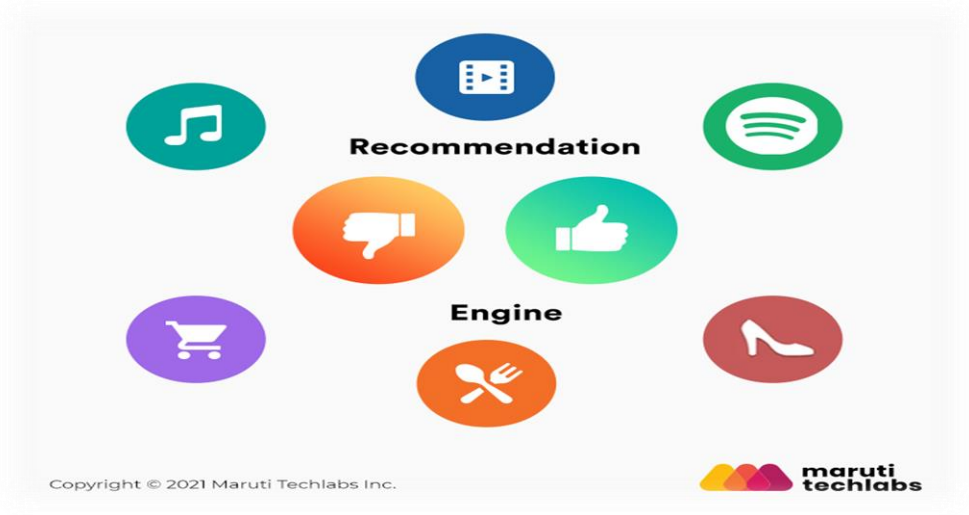


Figure 10

Overall, recommendation systems play a crucial role in helping users discover new content, products, or connections, thereby enhancing their overall experience and satisfaction with the platform. They also provide valuable insights for businesses, helping them improve user engagement, drive sales, and build customer loyalty.

5.1 Components of a Recommendation System

Recommendation systems typically consist of several key components that work together to generate personalized recommendations for users. Here are the main components:

User Interface: This is the part of the system that interacts with users, presenting recommendations in a user-friendly format such as a website, mobile app, or email. The user interface allows users to provide feedback on recommendations and adjust their preferences if necessary.

Input Data: Recommendation systems rely on various types of input data to generate recommendations. This data may include user preferences, historical interactions (such as purchases or ratings), item attributes (such as genre for movies or specifications for electronics), and contextual information (such as time of day or location).

Data Preprocessing: Before recommendations can be generated, input data often needs to be preprocessed to ensure it is in a suitable format for analysis. This may involve tasks such as cleaning the data, removing duplicates, handling missing values, and encoding categorical variables.

Recommendation Algorithm: This is the core component of the recommendation system that generates recommendations based on the input data.

There are several types of recommendation algorithms, including collaborative filtering, content-based filtering, matrix factorization, and hybrid approaches. The algorithm analyzes user behavior and item characteristics to identify patterns and make predictions about which items a user is likely to be interested in.

Model Training: In many recommendation systems, the recommendation algorithm needs to be trained using historical data before it can generate accurate recommendations. This involves using machine learning techniques to learn patterns and relationships in the data. The trained model is then used to make predictions for new users and items.

Recommendation Generation: Once the recommendation algorithm is trained, it can generate recommendations for users in real-time. Recommendations are typically generated by selecting the top-N items that the algorithm predicts the user will be most interested in, based on their preferences and behavior.

Evaluation Metrics: To assess the performance of the recommendation system, evaluation metrics are used to measure how well the recommendations align with user preferences. Common evaluation metrics include precision, recall, accuracy, and mean average precision.

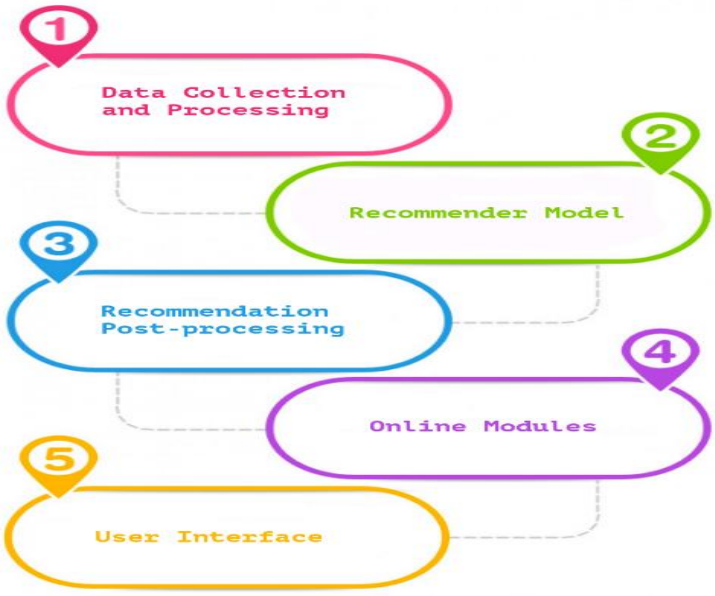


Figure 11

Feedback Loop: Recommendation systems often incorporate a feedback loop to continuously improve the quality of recommendations over time. This may involve collecting feedback from users on the recommendations they receive, monitoring user interactions with recommended items, and updating the recommendation algorithm based on this feedback.

Content Representation: In content-based recommendation systems, items are typically represented using feature vectors that capture their attributes or characteristics. These feature vectors may be generated using techniques such as term frequency-inverse document frequency (TF-IDF) for text-based items or feature extraction for multimedia content.

User Profile: Recommendation systems often maintain user profiles that store information about each user's preferences, behavior, and interactions with the system. User profiles are continuously updated based on new interactions and feedback, and they are used to personalize recommendations for individual users.

Item Catalog: The item catalog contains information about all the items available in the system, including their attributes, metadata, and descriptions. This catalog serves as the basis for generating recommendations and may be organized into categories or tags to facilitate browsing and discovery.

Scalability and Efficiency: As recommendation systems often deal with large volumes of data and need to process real-time requests, scalability and efficiency are crucial considerations. This involves designing the system architecture and algorithms to handle high loads efficiently, potentially using techniques such as distributed computing and parallel processing.

By integrating these components, recommendation systems can effectively analyze user behavior and preferences to provide personalized recommendations that enhance the user experience and drive engagement.

5.2 Types of algorithms used in Recommendation systems:

Indeed, recommendation systems rely on sophisticated algorithms to analyze user data and item characteristics to generate accurate and relevant recommendations. Let's explore some common algorithms used in recommendation systems:

Collaborative Filtering: Collaborative filtering is one of the most popular and widely used recommendation algorithms. It works by analyzing the behavior of multiple users and identifying patterns or similarities among their preferences.

There are two main types of collaborative filtering:

User-Based Collaborative Filtering: This approach recommends items to a user based on the preferences of users who like them. It identifies users with similar behavior or tastes and recommends items that those users have liked or interacted with.

Item-Based Collaborative Filtering: In this approach, recommendations are made based on the similarity between items. It identifies items that are like those that a user has already interacted with or liked and recommends them.

Content-Based Filtering: Content-based filtering recommends items to users based on the features or attributes of the items and the preferences of the user. It analyzes the content or characteristics of items, such as keywords, genres, or tags, and recommends items that are like those that the user has previously liked or interacted with.

Matrix Factorization: Matrix factorization is a technique used to decompose a user-item interaction matrix into lower-dimensional matrices. These lower-dimensional matrices represent latent factors or features that capture underlying patterns in the data. Matrix factorization algorithms, such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS), are commonly used in recommendation systems to generate personalized recommendations based on these latent factors.

Deep Learning Models: Deep learning models, such as neural networks, have also been applied to recommendation systems. These models can learn complex patterns and representations from large-scale user-item interaction data. Deep learning-based recommendation systems often incorporate techniques like embeddings, convolutional neural networks (CNNs), recurrent neural networks (RNNs), or attention mechanisms to capture intricate relationships between users and items.

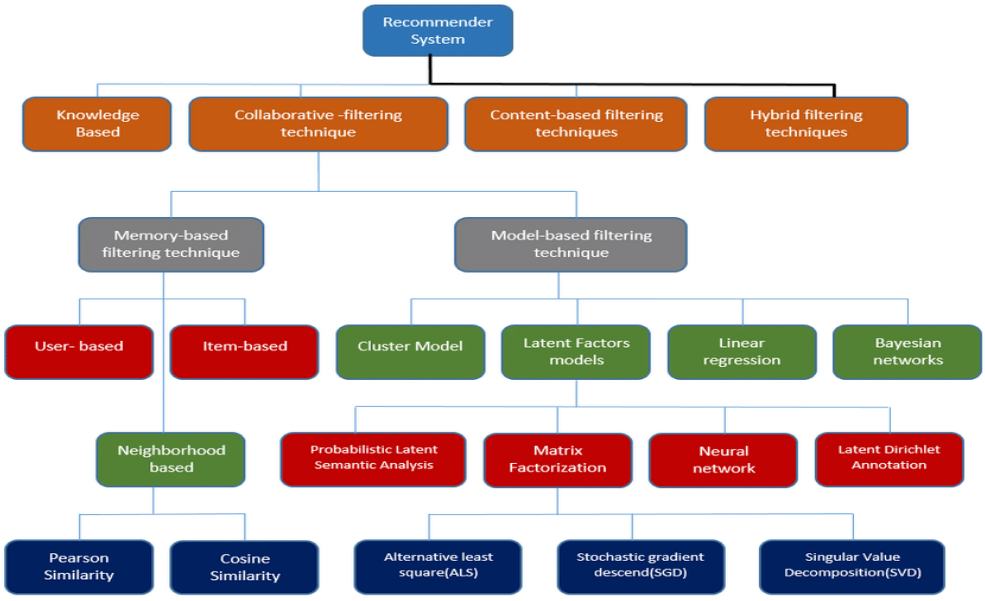


Figure 12

Hybrid Approaches: Hybrid recommendation systems combine multiple recommendation algorithms to leverage their strengths and overcome their weaknesses. For example, a hybrid system may integrate collaborative filtering with content-based filtering or combine matrix factorization with deep learning models to improve recommendation accuracy and coverage.

These are just a few examples of the algorithms used in recommendation systems. Each algorithm has its strengths and limitations, and the choice of algorithm depends on factors such as the characteristics of the data, the goals of the recommendation system, and the available computational resources.

5.3 Algorithms used in the Recommendation process:

5.3.1 Content-based Filtering

Content-based filtering is a recommendation algorithm that suggests items to users based on the attributes or characteristics of the items themselves, as well as the preferences of the user. It focuses on analyzing the content or features of items, such as text descriptions, keywords, genres, or metadata, to generate recommendations that match the user's interests.

Here's how content-based filtering works:

In the process of recommending electronic products such as laptops, the Content-Based Filtering algorithm is used to recommend products that match the preferences and interests of users. Here's how it can be implemented in this context:

Representing Products: Initially, products (such as laptops) are represented using features or attributes. These features can include technical specifications such as processor, memory, screen, storage, etc., in addition to other attributes such as brand, price, and ratings.

User profile: The user submits a profile that contains information about his personal preferences and the specifications he prefers in the laptop, such as brand, size, performance, etc.

Calculating similarity: Using user profile and product profiles, the similarity between the user profile and each product is calculated using a measure such as Cassian similarity. Products that most closely resemble the user's profile are selected.

Generating Recommendations: After calculating similarity, recommendations are generated by ranking the products based on the degree of similarity with the user profile. Priority is given to products that most closely match user preferences.

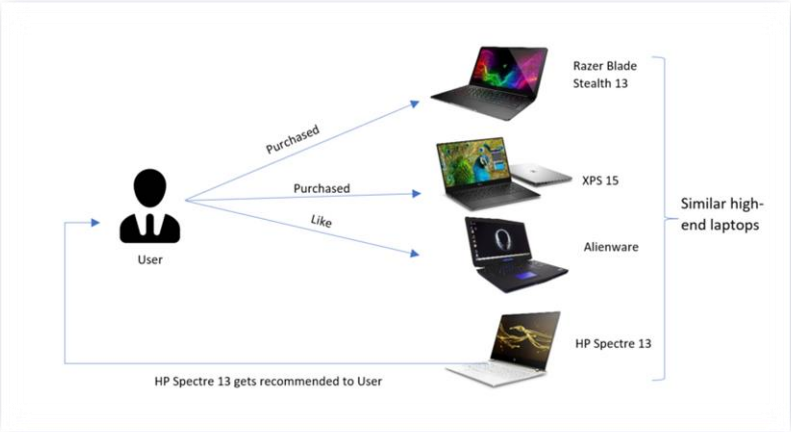


Figure 13

Content-based filtering has several advantages, including:

Personalization: Content-based filtering provides personalized recommendations based on the user's preferences and interests.

Transparency: The recommendations are generated based on explicit features or attributes of the items, making the process more transparent and interpretable.

Cold-start Problem: Content-based filtering can alleviate the cold-start problem, where new items or users have limited interaction data, by relying on item attributes for recommendation.

However, content-based filtering also has limitations, such as:

Limited Serendipity: Recommendations may be limited to items that are like those the user has already interacted with, leading to limited serendipity or exposure to new items.

Over-Specialization: The recommendations may become overly specialized or biased towards the user's past preferences, potentially missing out on novel or diverse recommendations.

Overall, content-based filtering is a powerful recommendation algorithm that can provide personalized recommendations based on the content or features of items. It is commonly used in various domains, including e-commerce, music streaming, news aggregation, and more.

5.3.2 Collaborative Filtering Recommendation Systems

Collaborative Filtering Recommendation Systems are methods used to recommend content to users by analyzing users' behaviors and preferences and comparing them with the behaviors and preferences of a group of other users. These systems are based on the basic idea that users with similar preferences will have similar recommendations. Below is a detailed explanation of the principles and types of collaborative filtering recommendation systems:

Understanding the Recommendation Principle: The basic principle of collaborative filtering recommender systems is to use prior knowledge about users' preferences to recommend content that matches those preferences.

User-based classification: This type of recommender system identifies users who have similar preferences to the current user, and then suggests content that the current user might like based on these users' similar preferences.

Item-based rating: It identifies similarities between items themselves, such as books or movies, and then recommends items that are like those for which the user has already provided positive reviews.

Item-User Pairing: Combines the previous two approaches, where information about users' preferences as well as information about items is used to recommend appropriate items to users.

System Advantages: Advantages of collaborative filtering recommender systems include their ability to precisely adapt to user preferences without requiring specific prior knowledge about the items. It also allows new or unknown items to be recommended to the user.

System Challenges: Challenges facing collaborative filtering recommender systems include the problem of changes in user preferences over time and the difficulty of dealing with new items that do not have sufficient ratings.

Implementation techniques: Implementation techniques for these systems include many methods such as cluster-based classification techniques, factor analysis techniques, and machine learning.

Collaborative filtering recommendation systems play a vital role in recommending electronic products such as laptops and other electronic devices. Here is a detailed explanation of the role of this system, its advantages, and disadvantages, in addition to how it works:

His role in electronic product recommendations:
Provides personalized recommendations: The system analyzes the user's previous behavior and preferences, then recommends products that meet his needs and suit his personal taste.

Improving user experience: By providing accurate and appropriate recommendations, the system helps improve the online shopping experience and increases user satisfaction.

Increase sales: By targeting users with products that suit their interests, recommendation systems can contribute to increasing sales rates for companies.

Advantages of collaborative filtering recommendation systems:

Effectiveness: Recommendations can be accurate and relevant thanks to the analysis of user behavior and preferences.

Low cost: It usually does not require a high cost of implementation compared to some other systems such as content-based classification.

Adaptability: The system can adapt to changes in user preferences continuously and improve its recommendations based on new changes.

Disadvantages of collaborative filtering recommendation systems:

Disconnection problem: Recommender systems may suffer from a disconnection problem where users are recommended products that are like their previous preferences without introducing new and innovative products.

Footnote effect: Users may be exposed to recommendations centered around products they have already purchased or browsed, without adequately accounting for their current preferences.

How do collaborative filtering recommendation systems work:

Collaborative filtering recommendation systems work by analyzing the preferences and behaviors of multiple users to generate personalized recommendations. There are two main approaches to collaborative filtering: user-based and item-based.

User-Based Collaborative Filtering:

Step 1: Similarity Calculation: The system calculates the similarity between users based on their past interactions with items. This can be done using similarity measures such as cosine similarity or Pearson correlation coefficient.

Step 2: Neighborhood Selection: Once the similarity between users is calculated, the system selects a neighborhood of similar users for the target user. This neighborhood typically consists of the k most similar users.

Step 3: Prediction Generation: For items that the target user has not interacted with, the system predicts their ratings or preferences based on the ratings of similar users. This prediction is usually a weighted average of the ratings given by the similar users, with weights determined by their similarity to the target user.

Step 4: Recommendation Generation: Finally, the system recommends the top-N items with the highest predicted ratings to the target user.

Item-Based Collaborative Filtering:

Step 1: Item Similarity Calculation: Instead of calculating similarities between users, item-based collaborative filtering calculates similarities between items based on the ratings given to them by users. Similarity measures such as cosine similarity or Pearson correlation coefficient are commonly used.

Step 2: Neighborhood Selection: The system selects a neighborhood of similar items for each item in the catalog. This neighborhood typically consists of the k most similar items.

Step 3: Prediction Generation: When making recommendations for a user, the system identifies the items they have already interacted with and selects similar items from the neighborhood. The predicted ratings for these items are then calculated based on the user's past ratings for similar items.

Step 4: Recommendation Generation: The system recommends the top-N items with the highest predicted ratings to the user.

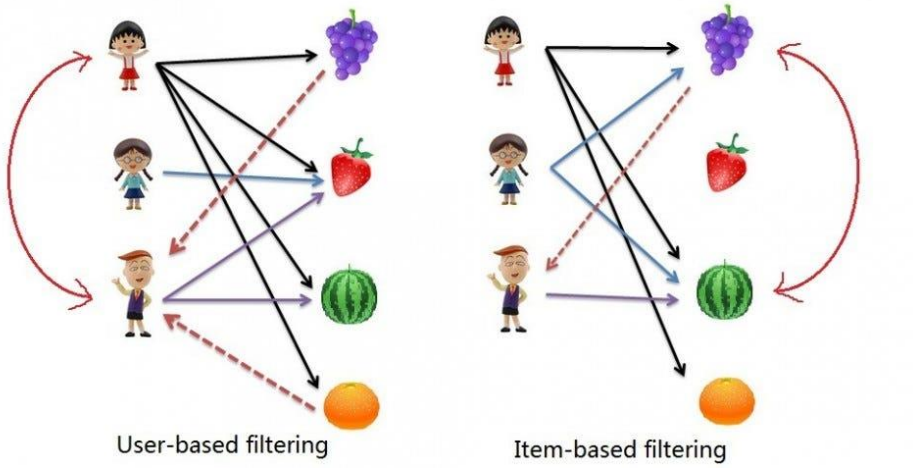


Figure 14

Both user-based and item-based collaborative filtering rely on the principle that users who have similar preferences in the past are likely to have similar preferences in the future, and items that are like those a user has liked in the past are likely to be of interest to them.

One advantage of collaborative filtering is that it does not require explicit knowledge of item attributes or user preferences, making it suitable for a wide range of recommendation scenarios. However, it can suffer from the cold start problem for new users or items with limited data, and it may also struggle with sparsity in the user-item interaction matrix.

In summary, collaborative filtering recommender systems are a powerful tool for improving the online shopping experience and increasing sales rates, but they also require addressing challenges such as the disconnection problem and the entourage effect to get the most benefit from them.

5.3.3 Hybrid based filtering Recommendation systems:

Hybrid recommendation systems combine multiple recommendation techniques, such as collaborative filtering, content-based filtering, and sometimes other approaches like knowledge-based systems or demographic-based filtering. The goal is to leverage the strengths of different methods while mitigating their weaknesses, ultimately providing more accurate and diverse recommendations. Here's how hybrid recommendation systems typically work:

Feature Combination: In hybrid systems, features from different recommendation techniques are combined to enhance recommendation quality. For example, in a collaborative-content hybrid system, user preferences learned from collaborative filtering (e.g., user-item interactions) are combined with item features learned from content-based filtering (e.g., item attributes or descriptions).

Weighted Fusion: Hybrid systems often assign weights to different recommendation techniques based on their performance or relevance to the user and item context. These weights are used to blend recommendations from multiple methods into a single ranked list. The weights can be static or dynamic, adjusting over time based on user feedback or system performance.

Switching Strategies: Some hybrid systems use switching strategies to select the most suitable recommendation technique for each user or item. For example, the system may switch between collaborative filtering and content-based filtering based on the availability of user data or item

attributes. This dynamic switching helps address the cold start problem and improves recommendation accuracy.

Cascade or Meta-Level Fusion: In cascade or meta-level fusion approaches, recommendations from different methods are generated independently and then combined at a higher level. For instance, recommendations from collaborative filtering and content-based filtering can be merged using a meta-classifier or ranking algorithm to produce the final recommendation list.

Contextual Adaptation: Hybrid systems may also consider contextual information, such as time, location, or user context, to tailor recommendations to specific situations. Contextual adaptation enables the system to deliver more relevant recommendations that align with the user's current needs and preferences.

Ensemble Methods: Ensemble methods, such as bagging or boosting, can be applied in hybrid recommendation systems to aggregate predictions from multiple recommendation techniques. By leveraging the diversity of different models, ensemble methods often yield more robust and accurate recommendations compared to individual methods alone.

Hybrid recommendation systems offer several advantages over single-method recommendation systems:

Improved Recommendation Quality: By combining multiple recommendation techniques, hybrid systems can leverage the strengths of each method while mitigating their weaknesses. This often leads to

more accurate and diverse recommendations that better match the preferences and interests of users.

Enhanced Coverage: Hybrid systems can overcome limitations associated with data sparsity and the cold start problem by incorporating multiple sources of information. They can provide recommendations for a wider range of items, including those with limited user interaction data or sparse content attributes.

Robustness and Adaptability: Hybrid systems are more robust to changes in user behavior, item availability, and system dynamics. They can adapt to evolving user preferences and environmental factors by dynamically adjusting the weights or switching between recommendation techniques based on real-time feedback and contextual information.

Addressing User Heterogeneity: Users have diverse preferences, and no single recommendation method can cater to all users equally well. Hybrid systems can accommodate user heterogeneity by offering personalized recommendations tailored to individual preferences, demographic characteristics, or contextual factors.
Reduction of Biases and Overfitting: By combining recommendations from multiple methods, hybrid systems can reduce biases inherent in individual recommendation techniques and mitigate the risk of overfitting to specific user-item interactions. This helps produce more balanced and generalizable recommendations.

Flexibility and Scalability: Hybrid recommendation systems provide flexibility in integrating new recommendation techniques or adapting existing ones to changing requirements and preferences. They can scale effectively to handle large datasets and accommodate diverse application domains and user populations.

Increased Serendipity: Hybrid systems can introduce serendipity into recommendations by combining techniques that may lead to unexpected but relevant suggestions. This enhances user satisfaction and discovery by exposing users to new and diverse content that aligns with their interests.

Overall, the versatility and effectiveness of hybrid recommendation systems make them well-suited for a wide range of recommendation scenarios, from e-commerce and entertainment platforms to content streaming services and personalized advertising. They offer a holistic approach to recommendation that maximizes user satisfaction and engagement while addressing the inherent challenges of recommendation tasks.

While hybrid recommendation systems offer numerous benefits, they also have some drawbacks and challenges:

Complexity: Hybrid recommendation systems tend to be more complex than single-method systems due to the integration of multiple recommendation techniques. Managing the interactions between different methods, determining optimal weighting schemes, and handling data fusion can increase system complexity and development effort.

Increased Computational Overhead: Integrating multiple recommendation techniques often requires additional computational resources and processing time. The need to calculate similarities, feature representations, or predictions from multiple sources can result in higher

computational overhead, especially for large-scale recommendation tasks.

Difficulty in Model Interpretability: Hybrid systems may lack transparency and interpretability compared to single-method systems, making it challenging to explain the rationale behind recommendations to users. Understanding how different recommendation techniques contribute to the final recommendations can be difficult for users and stakeholders.

Weighting and Parameter Tuning: Determining the optimal weighting or parameterization of different recommendation techniques in hybrid systems can be non-trivial. Finding the right balance between methods, adjusting weights dynamically, and optimizing hyperparameters may require extensive experimentation and tuning.

Data Integration Challenges: Integrating heterogeneous data sources and representations in hybrid systems can be challenging, particularly when dealing with diverse types of data such as user interactions, item attributes, and contextual information. Ensuring data consistency, quality, and compatibility across different sources is essential but can be difficult to achieve.

Dependency on Component Performance: The overall performance of hybrid recommendation systems heavily relies on the individual performance of each recommendation component. If one component performs poorly or introduces biases, it can negatively impact the quality of recommendations produced by the entire system.

Potential for Overfitting: Hybrid systems may be susceptible to overfitting, especially if the integration of multiple recommendation techniques is not carefully controlled. Over-reliance on specific methods or features can lead to biased recommendations and reduce the system's ability to generalize to new users or items.

Despite these challenges, hybrid recommendation systems remain a popular and effective approach for addressing the limitations of single-method systems and improving recommendation quality in various application domains. Careful design, robust evaluation, and ongoing optimization are essential for mitigating these drawbacks and maximizing the benefits of hybrid recommendation approaches.

To understand how a hybrid recommendation system works for electronic products such as laptops, we can explain the processes carried out by this system:

Data Collection: Data about users and products is collected from multiple sources. This data includes users' interactions with products (such as ratings and reviews), product attributes (such as brand, technical specifications, price), and any additional information that may be available (such as user ratings, demographic information).

Data Analysis: Data is analyzed using multiple techniques, including collaborative classification and content classification. Collaborative classification is used for basic analytics to see how users interact with similar products and to identify similar groups of users, while content classification is used to analyze attributes of products and determine similarity between them.

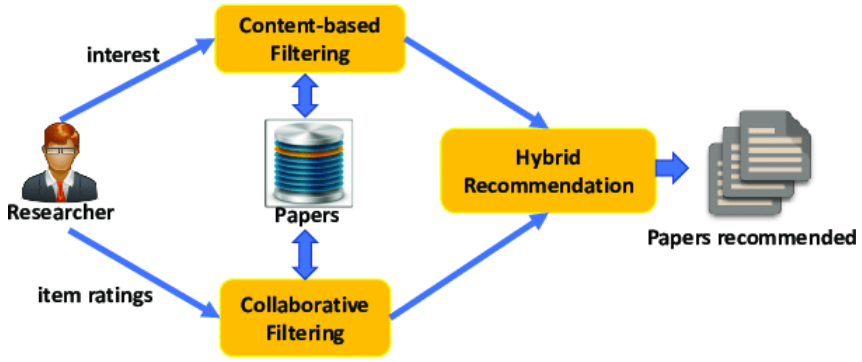


Figure 15

Recommendation generation: Based on data and analytics, personalized recommendations are generated for each user. Hybrid recommendation models can be used to combine recommendations from multiple sources. For example, popular laptops can be recommended based on the preferences of similar users, according to collaborative rating analysis, while more personalized recommendations can be driven based on specific product features using content rating.

System improvement: The system is continually improved by evaluating the performance of recommendations and collecting feedback from users. This information can be used to modify and improve hybrid recommendation models, including adjusting the weights used to combine recommendations from multiple sources or improving rating criteria.

Providing Recommendations: Personalized recommendations are provided to users via the appropriate user interface, whether it is via the website, mobile app, or emails. Users can browse recommendations and go to product pages to get more information or make purchases.

Feedback analysis and system improvement: Users' reactions to the recommendations provided are followed, whether positive or negative. In

response to these responses, the system is continually improved by fine-tuning the algorithms and criteria used for classification and improving the overall user experience.

Dealing with technical challenges: A hybrid recommender system faces technical challenges such as managing large data volumes, providing real-time recommendations, and ensuring security and data protection. Addressing these challenges requires designing a robust and failure-resistant system, as well as following appropriate security and privacy practices.

Providing an integrated user experience: The hybrid recommendation system aims to provide an integrated and harmonious user experience. This is done by designing an intuitive and attractive user interface, providing high-quality and diverse content, and effectively guiding users to find the products they are looking for.

Achieving business goals: Achieving business goals, such as increasing sales and improving user experience, is a key goal of a hybrid recommender system. System performance is regularly evaluated based on key performance metrics, and strategies are adjusted according to the results to get the most out of the system.

Using these processes, a hybrid recommendation system for electronic products such as laptops can provide personalized and accurate recommendations to users, which contributes to improving the online shopping experience and increasing sales rates for companies.

5.3.4 K-Means clustering algorithm:

As every customer has individual preferences, col lectin datasets are the most important and challenging task, while giving recommendations about any product, the Loy alti of the customer needs to be taken into consideration, In the last decade, recommendation systems have become an integral part of e-commerce business to promote product sales and thus become a popular research field in the present era. E-commerce business is electronic transactions over the internet which is focusing on products as well as services. The sustainability of an e-commerce business depends on the financial progress of the enterprises.

It has been observed that though the e-commerce businesses have enhanced their performance in the short term through investment and financing, there are some market risks involved which may even lead to bankruptcy. The prime motto of online business is increasing sales and maximizing profit. To maximize sales, one method employed by an online business is called the conversion of browsers into customers.

It is a process wherein a customer purchases or does some action apart from a sim plea visit. To increase the conversion rate of browsers into customers, some strategies are employed by the website. In, page visits, purchase history, duration of the visit, conversion rate, and number of orders made by the customer are analyzed for the purpose of recommendation.

Customers' buying habits can be analyzed by the relationships of different products they put into their shopping basket while shopping. This can help retailers to know which products consumers most frequently

purchase so that they can develop a better marketing strategy. It has been observed that a certain category of customers has an inclination toward a particular lar brand.

The recommendation system plays an important role by generating suggestions about products to the users. It can also suggest a particular brand of product according to the customers' requirements. It takes the feedback of the customers about the item, which a user has already purchased earlier.

According to the input provided by the user, recommendation systems apply certain algorithms to generate users' ratings on a particular product/item.

In general, the recommendation system suffers from sparsity and scalability problems which reduces the quality of predictions. Another major challenge in present e-commerce business is the cat erotization of the products precisely and efficiently.

Clustering the products into different groups or effectively reducing the dimensionality of data by applying PCA can solve these problems. By applying PCA, the dimension reduction is largely done, and hence algorithmic complexity is reduced.

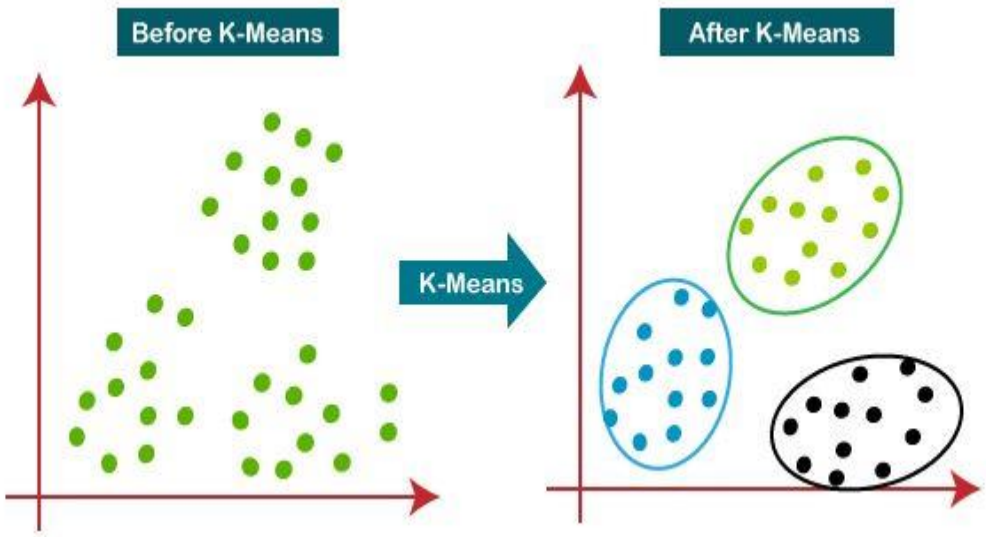


Figure 16

In this work, two unsupervised machine learning techniques PCA and K-Means algorithm have been applied to cluster the customers who purchase apparel online. K-Means clustering has been applied to segment the customer using the monthly income of the customer and the price of the item which they have purchased. This model can recommend apparel products to customers using an online recommender system.

Using PCA, all the information contents of the user have been summarized as the principal components and the clustering of customer have been done based on that.

By comparing these two results, it has been observed that similar clusters were formed. In addition to that, a segmentation model was prepared to cluster the items based on the stock keeping unit (SKU).

The model can run through the sales volume of the item and revenue generated, and the sale price of the item and revenue generated.

This model will help the retailer in identifying the maximum selling item, the most revenue generated item, or the item which can make loss of business.

Furthermore, this model will help the retailer in identifying which item needs more marketing focus. SKU is a unique numerical identifying number that is used to identify the product, product size or type, and the manufacturer in the retail industry.

It is a part of a backend inventory control system that can help the retailer to get low-inventory alerts and restock items that are forecasted by the model. Though we have collected the data from the customers only, the model was prepared to be recommended to the retailer also.

We have prepared the stock keeping unit according to the product, price, and brand information. So, this model can be extended to all categories of customers as well as retailers too.

The subsequent sections of this paper are organized as follows. In Sect. 2, the related research work has been discussed. Our proposed model and details of implementation have been described in Sect. 3. We have discussed the dataset in Sect. 4. In Sect. 5, result analysis has been done. In Sect. 6, the conclusion and future work have been summarized.

5.3.4.1 Proposed model and implementation

In the present scenario, it has been observed that e-commerce product classification and recommendation of products to the customers are tough and challenging tasks, and its dominance in the research area is increasing manifold. In this work, the database of different products/items purchased by customers is taken into consideration. The block diagram of the proposed model for recommendation aimed at customers is shown in Fig.

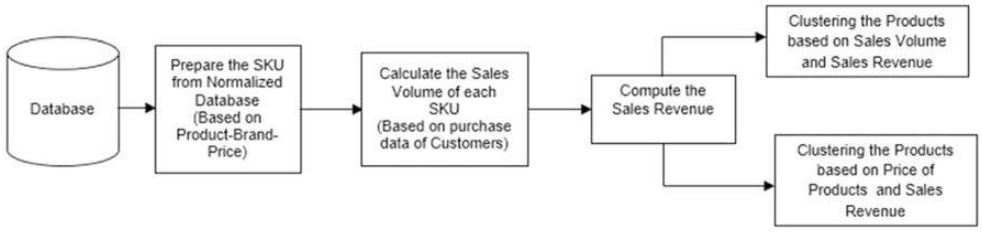


Figure 17

The block diagram of the proposed model for the recommendation of the retailer is shown in Fig. 2. In this proposed model, a unique product–brand–price was modeled as stock keeping unit. The product frequency count of a unique product–brand–price has been modeled as sales volume. The product price of a unique product–brand has been modeled as the sale price. Then, sales volume was multiplied by sale price to compute revenue generation.

During the festive season, customers normally buy different types of apparel products from various brands and the buying pattern of a person is different from that of other periods of the year and the choice of

products and brand varies from person to person. Customers choose different brands as status symbols, some customers trust the quality, and others may choose products based on the prices offered. If the retailer can categorize the products, it will help them to fix their marketing strategy.

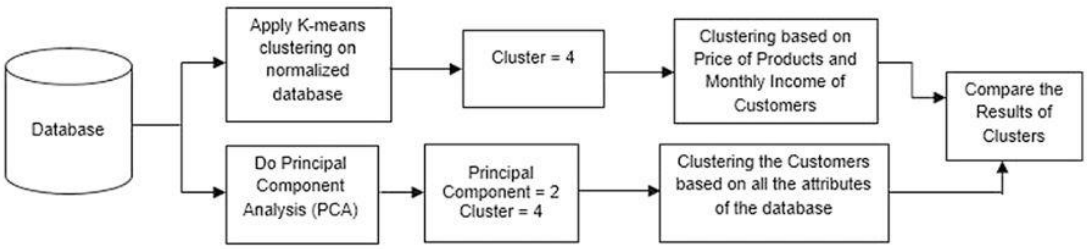


Figure 18

5.3.4.2 The basic step of K-Means clustering is discussed below

- Step 1 Determine the number of clusters K at the beginning.
- Step 2 Determine the coordinates of the centroids/centers of these clusters. Any random data point can be taken as the initial centroid or first K data points can also be used as the initial centroids.
- Step 3 Determine the distance of each data point to the centroids.
- Step 4 Group the object based on minimum distance (find the closest centroid). Each record is assigned to the nearest cluster using a measure of distance (Euclidean distance).

- Step 5 Keeping the same number of clusters, the new centroid of each cluster is calculated, by taking the arithmetic mean of all the data points which belong to that cluster.

- Step 6 Repeat step (2) to step (5) until the clusters stop changing or until the data points stay in the same cluster. The major work of this algorithm is customer segmentation and having knowledge about customers' needs. This will play an important role in building a successful venture.

Customer segmentation is the process where the segregation of data is performed based on different inter sets of potential customers. The whole process involves various actions that help to predict future actions. We can get a huge amount of unstructured data; so, once these data are provided to the K-Means algorithm,

it effectively anal yeses and integrates the data to give a structured appear acne, and if the user requests for a particular set of information, then properly analyzed data are provided.

5.3.4.3 Principal component analysis (PCA):

PCA is a statistical technique, which is used to reduce the dimensionality of a dataset consisting of many correlated variables. It retains the variation present in the dataset, up to the maximum extent, and uses the ideas of variances and co-variances. The following steps are performed to reduce the dimensionality of the dataset.

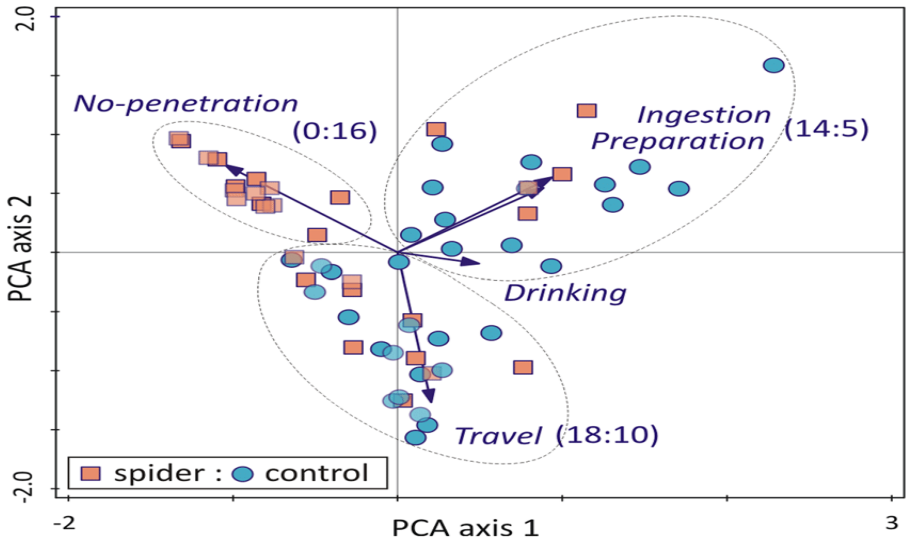


Figure 19

- Step 1 Standardization or scaling the data into a com parable range.
- Step 2 Computing the covariance matrix to identify the correlation and dependencies among the features in a dataset.
- Step 3 Calculating eigenvectors and eigenvalues from the covariance matrix to determine the largest to smallest eigenvalues of the eigenvectors.
- Step 4 Computing the principal component. (The eigen vector with the highest eigenvalue is most significant and taken into consideration and thus the first principal com ponent is formed. The principal components which are less significant are removed to reduce the dimension of the data.)
- Step 5 Rearrange the original data with the final principal components and thus reduce the dimension of the dataset. PCA

helps to identify the correlation and dependencies among the features in a dataset.

A covariance matrix expresses the correlation between the different variables in the dataset. It is essential to identify heavily dependent variables because they contain biased and redundant information, which reduces the overall performance of the model. Mathematically, a covariance matrix is a p × p matrix, where p represents the dimensions of the dataset. Each entry in the matrix represents the covariance of the corresponding variables. Here are the key takeaways from the covariance matrix:

- The covariance value denotes how co-dependent two variables are with respect to each other.
- A negative covariance denotes the respective variables are indirectly proportional to each other.
- A positive covariance denotes that the respective variables are directly proportional to each other.

CHAPTER 6

6. Datasets:

6.1 Description of data:

The dataset is the laptop listings on Flipkart, an ecommerce website in India. It contains 984 unique entries along with the corresponding laptop's CPU ranking and GPU benchmarks. This dataset is a follow up of a previously uploaded dataset. It contains 984 unique entries. The dataset has 26 features (Ex: Laptop name, link to buy, price in Indian rupees, processor, GPU, processor ranking, GPU benchmark, etc.). Please go through the following column description as some of the columns have been **tokenized** (check file column section below or column description in about dataset). This dataset was created by scraping data from the web (dated June).

#	Column	Non-Null Count	Dtype
0	index	984 non-null	int64
1	link	984 non-null	object
2	name	984 non-null	object
3	user rating	690 non-null	float64
4	Price (in Indian Rupees)	984 non-null	int64
5	Type	984 non-null	int64
6	Dedicated Graphic Memory Capacity	984 non-null	float64
7	Processor Brand	984 non-null	int64
8	SSD	984 non-null	int64
9	RAM (in GB)	984 non-null	int64
10	RAM Type	984 non-null	object
11	Expandable Memory	984 non-null	int64
12	Operating System	984 non-null	int64
13	Touchscreen	984 non-null	int64
14	Screen Size (in inch)	984 non-null	float64
15	Weight (in kg)	984 non-null	float64
16	Refresh Rate	984 non-null	int64
17	screen_resolution	984 non-null	int64
18	company	984 non-null	int64
19	Storage	984 non-null	int64
20	Processor name	984 non-null	object
21	CPU_ranking	984 non-null	int64
22	battery_backup	984 non-null	float64
23	gpu name	984 non-null	object
24	gpu_benchmark	984 non-null	float64
25	ram_type_tokenized	984 non-null	int64
26	gpu_processor tokenized	984 non-null	int64

dtypes: float64(6), int64(16), object(5)

Figure 20

6.2 Dataset Overview:

Source: Scraped from Flipkart, an ecommerce website in India.
Date of Collection: June (specific date not mentioned).

Number of Entries: 984 unique entries.
Number of Features: 26 features.

6.3 Sample Features in the Dataset:

Laptop Name: The name or model of the laptop.

Link to Buy: URL link to the product page for purchasing the laptop.

Price in Indian Rupees: The price of the laptop listed in Indian Rupees (INR).

Processor: Information about the laptop's processor (CPU).

GPU: Information about the laptop's graphics processing unit (GPU).

Processor Ranking: Ranking or performance score of the laptop's processor.

GPU Benchmark: Benchmark score of the laptop's GPU.

Other Features: Additional features such as display size, RAM, storage capacity, operating system, etc.

6.4 Column Description (Tokenized Columns):

Some columns may have been tokenized, which means they have been split or processed in some way to extract specific information. Examples of tokenized columns may include:

Processor: May contain information about the CPU brand, model, and specifications.

GPU: May contain details about the graphics card brand, model, and specifications.

6.5 Potential Use Cases:

Market Analysis:
Analyzing trends in laptop specifications, prices, and performance rankings.

Price Optimization:
Identifying pricing strategies based on features and performance.

Product Comparison: Providing insights for consumers comparing different laptop options.

Recommendation Systems: Building recommendation systems based on user preferences and budget constraints.

6.6 Data Preprocessing:

Cleaning and preprocessing steps may include handling missing values, standardizing feature formats, and parsing tokenized columns to extract relevant information.

Data normalization may be necessary for features such as price, processor ranking, and GPU benchmark to ensure consistency in scale. Overall, this dataset offers valuable insights into the laptop market on Flipkart, and with proper preprocessing and analysis, it can provide useful information for various business and consumer applications.

6.7 Analyzing a Dataset:

Laptop Name:
This column contains the name or model of each laptop listed on Flipkart. The laptop name serves as an identifier for each entry in the dataset.

Link to Buy:
This column provides the URL link to the product page on Flipkart where the laptop can be purchased. Users can click on this link to access the product page directly.

Price in Indian Rupees:
This column indicates the price of each laptop listed on Flipkart, expressed in Indian Rupees (INR). The price represents the cost of purchasing the laptop from the ecommerce platform.

6.7.1 Processor:

The "Processor name" column in the dataset likely contains the name or model of the CPU (Central Processing Unit) installed in each laptop. This column provides information about the specific CPU used in the laptop, including the brand, model, and any additional specifications.

Each entry in the "Processor name" column would consist of a string representing the CPU name. This string may include details such as the CPU brand (e.g., Intel, AMD), specific model (e.g., Core i5, Ryzen 7), and possibly other specifications like the number of cores, clock speed, and cache size.

For example:

If a laptop is equipped with an Intel Core i7-10750H processor, the corresponding entry in the "Processor name" column would be "Intel Core i7-10750H".

Understanding the CPU name is crucial for users to assess the processing power and performance capabilities of the laptop, as different CPU models may offer varying levels of performance for computing tasks.

6.7.2 GPU:

The "GPU" column in the dataset provides information about the graphics processing unit (GPU) installed in each laptop. This column typically includes details such as the brand, model, and specifications of the GPU.

Here's what each component of the GPU information might entail:

Brand:
Indicates the manufacturer of the GPU, such as NVIDIA, AMD, or Intel.

Model:
Specifies the specific model or series of the GPU, such as GeForce GTX 1650, Radeon RX 5600M, or Intel Iris Xe Graphics.

Specifications:
Includes additional details about the GPU, such as the VRAM (Video Random Access Memory) size, clock speed, CUDA cores (for NVIDIA GPUs), and other technical specifications relevant to the graphics processing capabilities of the GPU.

For example:

If a laptop has an NVIDIA GeForce GTX 1650 GPU with 4GB of VRAM, the corresponding entry in the "GPU" column might be "NVIDIA GeForce GTX 1650 (4GB VRAM)".
Understanding the GPU information is essential for users who require high graphics performance for tasks such as gaming, video editing, 3D rendering, and other graphical applications. The GPU plays a crucial role

in rendering images, videos, and animations on the laptop's display, so knowing the GPU specifications helps users assess the graphics capabilities of the laptop.

6.7.3 Processor Ranking:

The "Processor Ranking" column in the dataset provides a performance ranking or score for the processor (CPU) installed in each laptop. This ranking or score serves as an indication of the processing power and performance capabilities of the CPU relative to other processors.

Here's how the processor ranking information might be structured:

The ranking or score could be a numerical value assigned to each CPU based on its performance metrics, benchmark scores, or other criteria.

Higher numerical values typically indicate higher performance or better-ranking processors, while lower values signify lower performance or lower-ranking processors.
The ranking may be based on standardized benchmark tests conducted by organizations like Pass Mark, Geek bench, or CPU-Z, which evaluate CPU performance across various tasks and applications.

For example:

A laptop with a processor ranking of "90" might indicate that its CPU is ranked higher in performance compared to a laptop with a processor ranking of "75".
Understanding the processor ranking allows users to gauge the processing power and performance level of the CPU in each laptop relative to other models. This information is valuable for users who

prioritize CPU performance for tasks such as multitasking, gaming, content creation, and other compute-intensive applications.

6.7.4 GPU Benchmark:

The "GPU Benchmark" column in the dataset provides the benchmark score of the laptop's graphics processing unit (GPU). GPU benchmarks measure the graphics processing performance of the GPU, which is crucial for tasks such as gaming, video editing, and 3D rendering.

Here's how the GPU benchmark information might be presented:

The benchmark score is a numerical value that represents the performance of the GPU in various graphics-intensive tasks and applications.

Higher benchmark scores typically indicate better graphics processing performance, while lower scores signify lower performance.

GPU benchmark scores are often obtained through standardized benchmark tests conducted by organizations such as 3DMark, Pass Mark, or Fur Mark. These tests evaluate the GPU's performance across different graphics rendering scenarios, including gaming, graphical simulations, and computational

For example:

A laptop with a GPU benchmark score of "8000" might indicate that its GPU performs better in graphics processing tasks compared to a laptop with a GPU benchmark score of "6000".
Understanding the GPU benchmark score allows users to assess the graphics processing capabilities of the GPU in each laptop. This information is essential for users who require high graphics performance for gaming, multimedia editing, design work, and other visually demanding applications.

6.7.5 Other Features:

The "Other Features" column in the dataset likely encompasses a variety of additional specifications and attributes relevant to laptops beyond those already mentioned. Here's a breakdown of some of the potential features that may be included in this column:

Display Size:
Specifies the size of the laptop's display screen, typically measured diagonally in inches.

RAM (Memory) Capacity:
Indicates the amount of random-access memory (RAM) installed in the laptop, measured in gigabytes (GB) or terabytes (TB).

Storage Type:
Specifies the type of storage device used in the laptop, such as solid-state drive (SSD), hard disk drive (HDD), or hybrid storage configurations.

Storage Capacity:
Provides information about the total storage capacity of the laptop's storage device, typically measured in gigabytes (GB) or terabytes (TB).

Operating System:
Indicates the operating system pre-installed on the laptop, such as Windows, macOS, Chrome OS, or Linux.

Battery Life:
Specifies the estimated battery life of the laptop on a single charge, typically measured in hours under specific usage conditions.

Weight:
Indicates the weight of the laptop, usually measured in kilograms (kg) or pounds (lbs.), to assess portability.

Dimensions:
Provides the physical dimensions of the laptop, including length, width, and height, typically measured in millimeters (mm) or inches (in).

Processor Type:
Specifies the type of processor architecture used in the laptop, such as Intel x86, AMD Ryzen, or ARM-based processors.

Graphics Card:
Provides additional details about the laptop's dedicated graphics card, if applicable, including brand, model, and VRAM size.

Connectivity Options:
Indicates the available connectivity ports and options on the laptop, such as USB ports, HDMI, Thunderbolt, Ethernet, Wi-Fi, Bluetooth, and card readers.

Keyboard and Touchpad:
Specifies features of the keyboard and touchpad, such as backlighting, key travel, and multi-touch gestures.

Audio Features:
Provides information about the laptop's audio system, including speaker configuration, audio enhancements, and microphone features.

These additional features contribute to the overall functionality, performance, and user experience of the laptop, allowing consumers to make informed decisions based on their specific needs and preferences.

6.7.6 Tokenized Columns:

Tokenized columns, such as "Processor" and "GPU," involve breaking down the information into distinct tokens or segments to extract specific details about the CPU and GPU, respectively. Here's how tokenization might be applied to these columns:

Processor Column:

Token 1: CPU Brand: This token represents the brand of the CPU, such as Intel, AMD, Qualcomm, etc.

Token 2: CPU Model: This token identifies the specific model or series of the CPU, such as Core i5, Ryzen 7, Snapdragon, etc.
Token 3 and beyond: CPU Specifications: Additional tokens may include specifications like the number of cores, clock speed, cache size, and other technical details.
Example: "Intel Core i7-10700K (8 cores, 3.80 GHz, 16MB cache)"

GPU Column:

Token 1: GPU Brand: This token denotes the manufacturer of the GPU, such as NVIDIA, AMD, Intel, etc.

Token 2: GPU Model: This token identifies the specific model or series of the GPU, such as GeForce GTX 1650, Radeon RX 5600M, Iris Xe Graphics, etc.

Token 3 and beyond: GPU Specifications: Additional tokens may include details like VRAM size, clock speed, CUDA cores (for NVIDIA GPUs), and other technical specifications.

Example: "NVIDIA GeForce GTX 1650 (4GB VRAM)"
By tokenizing the columns, each piece of information within the "Processor" and "GPU" columns can be separated and analyzed individually. This facilitates data processing, querying, and analysis, allowing users to extract specific attributes and characteristics of the CPU and GPU for further examination or comparison.

6.7.7 Use Cases:

The dataset can be used for various analyses and applications, including market analysis, price optimization, product comparison, and recommendation systems for laptops based on user preferences and budget constraints.

Market Analysis:
Analyze trends in laptop specifications, prices, and performance rankings across different brands and manufacturers.

Identify popular features and configurations among consumers in the laptop market.
Track changes in market demand and preferences over time to inform business strategies and product development.

Price Optimization:

Determine optimal pricing strategies based on laptop features, performance rankings, and market demand.
Identify price points that maximize profitability while remaining competitive in the market.
Analyze price elasticity and consumer response to price changes to adjust pricing strategies accordingly.

Product Comparison:

Facilitate side-by-side comparisons of laptops based on specifications, performance rankings, and price.
Help consumers make informed purchasing decisions by highlighting differences and similarities between laptop models.

Provide insights into the strengths and weaknesses of each laptop model relative to competitors.

Recommendation Systems:
Develop recommendation systems for laptops tailored to individual user preferences and budget constraints.

Utilize machine learning algorithms to analyze user data and recommend laptops that best match the user's requirements.

Incorporate collaborative filtering, content-based filtering, or hybrid recommendation approaches to enhance recommendation accuracy.

Inventory Management:

Optimize inventory management and stock replenishment strategies based on sales trends, demand forecasts, and product performance.
Identify slow-moving or obsolete laptop models for clearance or discontinuation.

Ensure adequate stock levels of popular laptop models to meet customer demand and minimize stockouts.
Marketing Strategies:
Target marketing campaigns and promotions based on consumer preferences, demographics, and buying behavior.
Personalize marketing messages and advertisements to resonate with different customer segments.
Analyze the effectiveness of marketing campaigns in driving sales and brand awareness in the laptop market.

By leveraging the dataset for these analyses and applications, businesses, researchers, and consumers can gain valuable insights into the laptop market and make informed decisions related to pricing, product positioning, marketing, and purchasing.

6.7.8 Data Preprocessing:

Data preprocessing is an essential step in preparing the dataset for analysis and modeling. Here's how various preprocessing steps can be applied to the dataset:

Cleaning the Data:

Identify and handle any inconsistencies, errors, or outliers in the dataset.
Remove duplicate entries or records to ensure data integrity.
Address any formatting issues or anomalies in the data.

Handling Missing Values:
Check for missing values in the dataset and decide on an appropriate strategy for handling them.
Options include imputation (replacing missing values with a calculated or estimated value), deletion (removing rows or columns with missing values) or treating missing values as a separate category.

Standardizing Formats:
Ensure consistency in data formats across different columns.
Convert data into a standardized format if necessary (e.g., converting dates to a uniform format).

Parsing Tokenized Columns:
For tokenized columns like "Processor" and "GPU," split the tokens into separate columns to extract specific information.
Create new columns for each tokenized attribute (e.g., CPU brand, model, specifications for the Processor column).

Normalizing Numerical Features:
Normalize numerical features to bring them to a common scale, especially if they have different units or ranges.
Techniques such as min-max scaling or z-score normalization can be applied to ensure consistency in numerical feature ranges.
Encoding Categorical Variables:
Convert categorical variables into numerical representations that can be understood by machine learning algorithms.
Techniques include one-hot encoding, label encoding, or ordinal encoding depending on the nature of the categorical variables.
Feature Engineering:
Create new features or derive additional insights from existing features to enhance model performance.
Example: Creating a new feature to represent the age of a laptop based on its release date.

Data Splitting:
Split the dataset into training, validation, and test sets for model training, evaluation, and testing purposes.
By performing these preprocessing steps, the dataset will be cleaned, standardized, and ready for analysis or modeling tasks such as market analysis, price optimization, recommendation systems, and more.

Preprocessing ensures that the data is reliable, consistent, and suitable for downstream tasks.

6.7.9 Insights and Analysis:

Analyzing the dataset can indeed yield valuable insights into various aspects of the laptop market on Flipkart. Here are some potential insights that businesses, researchers, and consumers can derive from exploring and analyzing the dataset:

Market Trends:
Identify trends in laptop sales volume, revenue, and popularity over time.
Determine which brands, manufacturers, or laptop types are gaining or losing market share.
Track shifts in consumer preferences for specific features, specifications, or form factors.

Pricing Strategies:
Analyze price distribution and trends across different laptop models, brands, and configurations.
Identify price segments or price ranges that are most prevalent in the market.
Determine optimal pricing strategies based on factors such as features, performance rankings, and competitive landscape.

Popular Laptop Configurations:
Identify the most popular laptop configurations based on factors like CPU brand, GPU model, RAM capacity, storage type, and display size.
Determine which combinations of features and specifications are in high demand among consumers.

Performance Benchmarks:
Evaluate the performance benchmarks of GPUs and CPUs across different laptop models and brands.

Compare the graphics processing performance and CPU performance of various laptops to identify top performers.
Determine which laptops offer the best value proposition in terms of performance relative to price.

Consumer Preferences:
Gain insights into consumer preferences for specific brands, manufacturers, or laptop types (e.g., gaming laptops, thin and light laptops).
Understand which features and specifications are most important to consumers when making purchasing decisions.
Identify emerging trends or shifts in consumer preferences that may impact future market dynamics.
By leveraging these insights, businesses can make informed decisions related to product development, marketing strategies, pricing adjustments, and inventory management. Researchers can use the dataset to conduct market studies and academic research on the laptop market. Consumers can benefit from the information to make educated purchasing decisions based on their preferences and requirements. Overall, exploring and analyzing the dataset can provide valuable intelligence for stakeholders in the laptop market ecosystem.

CHAPTER 7

7. Unveiling the Model: Understanding its Mechanics and Application

7.1 Introduction to the Model:

In this section, we delve into the intricacies of a data-driven model designed to enhance user experience and provide personalized recommendations. The model encompasses a series of steps, from data preprocessing to the application of machine learning algorithms, culminating in the delivery of tailored suggestions based on user preferences.

The primary objective of this model is to analyze and extract insights from a dataset containing information about various laptop specifications, including CPU ranking, RAM capacity, GPU benchmark, price, and manufacturer. By leveraging techniques such as clustering and logistic regression, the model aims to categorize laptops into distinct groups and predict user preferences with a high degree of accuracy.

Through a detailed exploration of each function within the model, we aim to elucidate its underlying mechanics and practical applications. From data cleaning and feature engineering to algorithmic training and recommendation generation, each step plays a crucial role in optimizing the model's performance and relevance to end-users.

Join us as we unravel the complexities of this model, shedding light on its methodology, functionality, and real-world implications. Whether you're a data enthusiast, a machine learning practitioner, or simply curious about the inner workings of recommendation systems, this journey promises valuable insights and practical knowledge.

Let's embark on this exploration of the model, where data science meets personalized user experiences.

The journey begins with the Importing Libraries:

7.2 Importing Libraries

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

import json
```

Figure 21

pandas (pd):

Explanation: Pandas is a powerful Python library used for data manipulation and analysis. It offers data structures and functions designed to make working with structured data easy and intuitive.
Usage: In our code, pandas is primarily used for reading and manipulating datasets, selecting specific features for training, and handling missing values.

NumPy (np):

Explanation: NumPy is a fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.
Usage: In our code, NumPy is utilized for numerical operations and transformations, such as scaling numerical features and concatenating arrays.

sklearn.cluster.K-Means:

Explanation: K-Means is a popular clustering algorithm used for partitioning data into clusters based on similarities. It aims to divide the data points into distinct groups, where each group (or cluster) contains data points that are similar to each other.
Usage: We use K-Means from the scikit-learn library to perform clustering on the dataset, grouping laptops with similar specifications into clusters.

sklearn.preprocessing.StandardScaler:

Explanation: StandardScaler is a preprocessing technique used to standardize features by removing the mean and scaling them to unit variance. It ensures that the features are on the same scale, which can improve the performance of machine learning algorithms.

Usage: In our code, StandardScaler is employed to scale numerical features before training the machine learning model.

sklearn.preprocessing.OneHotEncoder:

Explanation: OneHotEncoder is used to convert categorical features into a numerical format. It creates binary columns for each category in the feature, where each column indicates the presence or absence of that category.

Usage: We utilize OneHotEncoder to encode categorical features, such as laptop manufacturers, into a format suitable for machine learning algorithms.
sklearn.model_selection.train_test_split:

Explanation: train_test_split is a function used to split datasets into training and testing sets. It allows for evaluation of machine learning models on unseen data, helping to assess their generalization performance.
Usage: In our code, train_test_split is employed to partition the dataset into separate training and testing sets before model training.
sklearn.linear_model.LogisticRegression:

Explanation: LogisticRegression is a classification algorithm used to model the probability of a binary outcome. It estimates the probabilities for each class label and predicts the class with the highest probability.
Usage: We utilize LogisticRegression to train a classification model for predicting the cluster labels of laptops based on their features.
These libraries and modules form the backbone of our data preprocessing and machine learning pipeline, enabling us to manipulate data effectively, apply clustering algorithms, and train predictive models for personalized recommendations.

After that, you use the (data cleaning and preprocessing) function.
The data is carefully organized and converted into a format suitable for analysis.

```
def clean_and_preprocess_data(file_path):
    # Clean and preprocess data
    df = pd.read_csv(file_path, encoding='latin1')

    # Selecting features and target for training
    features = df[['CPU_ranking', 'RAM (in GB)', 'gpu_benchmark', 'Price (in Indian Rupees)', 'company']]
    features = features.dropna()

    # Encoding categorical features
    encoder = OneHotEncoder(handle_unknown='ignore')
    features_encoded = encoder.fit_transform(features[['company']])

    # Scaling numerical features
    scaler = StandardScaler()
    numerical_features = features[['CPU_ranking', 'RAM (in GB)', 'gpu_benchmark', 'Price (in Indian Rupees)']]
    features_scaled = scaler.fit_transform(numerical_features)

    # Combining encoded and scaled features
    features_final = np.concatenate((features_scaled, features_encoded.toarray()), axis=1)
    print("Team chatbotch ")

    return df, features_final, encoder, scaler
```

Figure 22

7.3 Reading the Data:

Explanation: This function begins by reading the dataset from a CSV file specified by the file_path. The pd.read_csv() function from the Pandas library is used for this purpose. The encoding='latin1' parameter is provided to handle any potential encoding issues in the dataset.
Selecting Features:

Explanation: After reading the data, the function selects specific features and the target variable (company) for training. These features include CPU_ranking, RAM (in GB), gpu_benchmark, and Price (in Indian Rupees). The dropna() function is then applied to remove any rows with missing values (NaN) in the selected features.
Encoding Categorical Features:

Explanation: Categorical features, such as the company column, need to be converted into a numerical format before they can be used in machine learning models. The OneHotEncoder from scikit-learn is utilized for this purpose. It creates binary columns for each category in the company feature, effectively encoding it into a numerical format.
Scaling Numerical Features:

Explanation: To ensure that numerical features are on the same scale and have similar ranges, they are scaled using the StandardScaler from scikit-learn. This preprocessing step standardizes the numerical features by removing the mean and scaling to unit variance. It helps prevent features with larger scales from dominating the learning process.
Combining Encoded and Scaled Features:

Explanation: Finally, the encoded categorical features and the scaled numerical features are combined into a single feature matrix

(features_final). This matrix contains all the features necessary for training machine learning models. The np.concatenate() function from NumPy is used for this purpose.
Printing Message (Optional):

Explanation: An optional message, "Team chatbotch", is printed to indicate that the preprocessing step has been completed.
Returning Processed Data and Preprocessing Objects:

Explanation: The function returns four objects: the original DataFrame df, the processed feature matrix features_final, the encoder object encoder (used for encoding categorical features), and the scaler object scaler (used for scaling numerical features). These objects are essential for further analysis and model training.

After that, you use the (Apply_kmeans_clustering)

7.4 Apply_kmeans_clustering.

Clustering algorithm to classify laptops into distinct groups based on their features. After that, you do the job.

```
def apply_kmeans_clustering(df, features_final, n_clusters=3, random_state=42):
    # Apply K-means clustering
    kmeans = KMeans(n_clusters=n_clusters, random_state=random_state)
    df['cluster'] = kmeans.fit_predict(features_final)
    return df, kmeans
```

Figure 23

Explanation: This function applies the K-means clustering algorithm to the processed features (features final) in order to group similar data points into clusters.
Usage: It initializes a K-Means object with the specified number of clusters (n_clusters) and a random state for reproducibility. Then, it fits the KMeans model to the data using the fit_predict() method, which assigns each data point to a cluster and returns the cluster labels.
Assigning Cluster Labels to DataFrame:

Explanation: The cluster labels obtained from the KMeans model are assigned to a new column named 'cluster' in the DataFrame df. Each row in the DataFrame corresponds to a data point, and the assigned cluster label indicates the cluster to which the data point belongs.
Usage: The cluster labels are added to the DataFrame using bracket notation (df['cluster']). This allows for easy access and manipulation of the cluster assignments within the DataFrame.
Returning Updated DataFrame and KMeans Model:

Explanation: The function returns two objects: the updated DataFrame with cluster labels (df) and the trained KMeans model (kmeans). These objects can be further analyzed or used for making predictions.
Usage: The DataFrame with cluster labels is returned to allow for downstream analysis, such as visualizing cluster assignments or performing cluster-specific operations. Additionally, the trained K-Means model is returned in case it needs to be reused or evaluated later in the analysis pipeline.

After that, you use the (Train_logic_regression)

7.5 Train_logic_regression.

We train a logistic regression model to predict which group the new laptop belongs to.

```python
def train_logistic_regression(features_final, df_cluster, test_size=0.2, random_state=63):
    # Splitting the data into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(features_final, df_cluster, test_size=test_size, random_state=random_state)

    # Training the Logistic Regression model
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)

    # Evaluating the model's performance on the testing set
    accuracy = model.score(X_test, y_test)
    print("Accuracy on testing set: {:.2f}%".format(accuracy * 100))

    return model


file_path = 'dataset.csv'
df, features_final, encoder, scaler = clean_and_preprocess_data(file_path)

df, kmeans = apply_kmeans_clustering(df, features_final)
model = train_logistic_regression(features_final, df['cluster'])
# Exchange rate of Indian Rupee to Egyptian Pound
exchange_rate_inr_to_egp = 0.37 # You can change this number based on the current exchange rate
# Convert the price of the laptop from the Indian rupee to the Egyptian pound
df['Price (in Egyptian Pounds)'] = df['Price (in Indian Rupees)'] * exchange_rate_inr_to_egp
```

Figure 24

Training Logistic Regression Model:

Explanation: This function trains a logistic regression model using the features (features_final) and cluster labels (df_cluster) obtained from the preprocessing steps. Logistic regression is a supervised learning algorithm used for binary classification tasks.

Usage: The data is split into training and testing sets using the train_test_split function from scikit-learn. The logistic regression model is then instantiated with a maximum iteration limit (max_iter) set to 1000 to ensure convergence. The model is trained on the training data using the fit method.

Evaluating Model Performance:

Explanation: After training the model, its performance is evaluated on the testing set. The score method of the logistic regression model computes the accuracy of the model on the testing data.
Usage: The accuracy score is printed to the console, indicating how well the model performs in predicting the cluster labels on unseen data.
Returning Trained Model:

Explanation: The trained logistic regression model (model) is returned from the function.
Usage: The trained model can be further utilized for making predictions or analyzing the relationship between features and cluster labels.
Additional Code:

Explanation: After defining the function, the subsequent code reads the dataset from a CSV file, preprocesses the data, applies K-means clustering, and trains a logistic regression model.
Usage: The exchange rate between the Indian Rupee and the Egyptian Pound is defined (exchange_rate_inr_to_egp). The price of laptops in the DataFrame df is then converted from Indian Rupees to Egyptian Pounds using this exchange rate.

After that, you use the (get_user_profile_cluster)

When a user profile is provided. This function encodes and measures a user's preferences, predicts which group best matches his or her profile,

and lays the foundation for personalized recommendations. Finally, a function

```python
def get_user_profile_cluster(user_profile_data, kmeans, scaler, encoder):
    # Encoding and scaling user profile data
    company_encoded = encoder.transform(user_profile_data[['company']])
    numerical_data = user_profile_data[['CPU_ranking', 'RAM (in GB)', 'gpu_benchmark', 'Price (in Indian Rupees)']]
    numerical_data_scaled = scaler.transform(numerical_data)

    # Combining encoded and scaled user profile data
    user_profile_final = np.concatenate((numerical_data_scaled, company_encoded.toarray()), axis=1)

    # Predict the cluster for user profile data
    user_profile_cluster = kmeans.predict(user_profile_final)
    return user_profile_cluster[0]
```

Figure 25

Encoding and Scaling User Profile Data:

Explanation: This function takes user profile data as input and performs encoding and scaling operations like the preprocessing steps applied to the original dataset. Categorical features (e.g., company) are encoded using the same encoder object used during preprocessing, and numerical features are scaled using the scaler object.

Usage: The user profile data is first encoded using the transform method of the encoder object to convert categorical features into numerical format. Then, numerical features are scaled using the transform method of the scaler object to ensure consistency with the training data.
Combining Encoded and Scaled User Profile Data:

Explanation: After encoding and scaling, the encoded and scaled user profile data are combined into a single feature matrix (user_profile_final) using NumPy's concatenate function. This matrix contains all the features necessary for predicting the user's cluster.

Usage: The concatenate function is used to combine the scaled numerical features (numerical_data_scaled) with the encoded categorical features converted to a dense array (company_encoded.toarray()).
Predicting the Cluster for User Profile Data:

Explanation: Once the user profile data is prepared, the function uses the trained K-means model (kmeans) to predict the cluster label for the user profile. The predict method of the K-means model assigns the user profile data to one of the predefined clusters.
Usage: The predicted cluster label for the user profile data is returned as the output of the function. The [0] indexing is used to extract the predicted cluster label from the array returned by predict.
Returning Predicted Cluster Label:

Explanation: The function returns the predicted cluster label (user_profile_cluster) as the output.
Usage: The predicted cluster label can be used to retrieve cluster-specific recommendations or to analyze user preferences based on their profile data.
Overall, this function prepares and predicts the cluster label for user profile data, allowing for personalized recommendations based on the user's characteristics and preferences. It ensures that the user's profile is processed in a manner consistent with the preprocessing steps applied to the training data.

After that, you use the (content_based_recommendation)

It calculates content-based similarities within a user's group and recommends top-rated laptops tailored to their preferences.

```python
def content_based_recommendation(user_profile, cluster_data):
    # Calculate content-based similarity within the selected cluster
    similarity_list_content = (
        2 * (user_profile['Ram'] == cluster_data['RAM (in GB)']).astype(int) +
        (user_profile['Touch'] == cluster_data['Touchscreen']).astype(int) +
        1 / (1 + np.abs(user_profile['Inches_max'] - cluster_data['Screen Size (in inch)'])) +
        1 / (1 + np.linalg.norm(user_profile['cpu_max'] - cluster_data['CPU_ranking'])) +
        1 / (1 + np.linalg.norm(user_profile['GPU_max'] - cluster_data['gpu_benchmark'])) +
        5 / (1 + np.abs(user_profile['min_price'] - cluster_data['Price (in Indian Rupees)'])) +  # Update: Higher weight for price
        5 / (1 + np.abs(user_profile['max_price'] - cluster_data['Price (in Indian Rupees)'])) +  # Update: Higher weight for price
        (user_profile['OS'] == cluster_data['Operating System']).astype(int) +
       1 * (user_profile['company'] == cluster_data['company']).astype(int)
    )
    cluster_data['similarity_content'] = similarity_list_content
    top_matches_content = cluster_data.sort_values(by=['similarity_content', 'user rating'], ascending=[False, False]).head(5)
    return top_matches_content
```

Figure 26

Calculating Content-Based Similarity:

Explanation: This function calculates the content-based similarity between the user profile and laptops within the selected cluster. It computes a similarity score for each laptop based on various features such as RAM, touchscreen, screen size, CPU ranking, GPU benchmark, price, operating system, and company.
Usage: The similarity score is calculated using a weighted combination of feature comparisons. Different features contribute differently to the similarity score, with some features (such as price) given higher weights. The similarity score reflects how closely each laptop matches the user's preferences.
Sorting and Filtering Top Matches:

Explanation: After computing the similarity scores, the function sorts the laptops in the cluster based on their similarity to the user profile. It sorts the laptops in descending order of similarity and, in case of ties, by user rating. It then selects the top 5 matches as recommendations for the user.
Usage: The sorted DataFrame of laptops (cluster_data) is filtered to retrieve the top 5 matches based on similarity and user rating. These top

matches represent the most suitable recommendations for the user's preferences.

Adding Similarity Score to Data Frame:

Explanation: The similarity scores computed for each laptop are added as a new column ('similarity_content') to the DataFrame cluster_data. This allows for easy access to and visualization of the similarity scores.
Usage: The similarity scores provide additional context when presenting the recommendations to the user, helping them understand why certain laptops were selected as top matches.
Returning Top Matches:

Explanation: The function returns a DataFrame (top_matches_content) containing the top 5 recommended laptops based on content-based similarity.
Usage: The returned DataFrame can be further processed or presented to the user as personalized recommendations, assisting them in making informed decisions about laptop selection.
Overall, this function enables the generation of content-based recommendations tailored to the user's preferences, leveraging similarity calculations between the user profile and laptops within a selected cluster.

After that, you use the (main)

The main function coordinates personalized laptop recommendations by preparing user profile data, predicting user clusters, and generating content-based suggestions based on cluster characteristics, ultimately delivering tailored recommendations for improved user experience.

```python
def main(user_profile):

    user_profile_data = pd.DataFrame({
        'CPU_ranking': [user_profile['cpu_max']],
        'RAM (in GB)': [user_profile['Ram']],
        'gpu_benchmark': [user_profile['GPU_max']],
        'Price (in Indian Rupees)': [user_profile['min_price']],
        'company': [user_profile['company']]
    })
    # Convert the company variable to dummy variables
    user_profile_data = pd.get_dummies(user_profile_data, columns=['company'], drop_first=True)

    # Ensure the user profile data has the same feature names as the training data
    user_profile_data = user_profile_data.reindex(columns=df.columns, fill_value=0)

    user_profile_cluster = get_user_profile_cluster(user_profile_data, kmeans, scaler, encoder)
    cluster_data = df[df['cluster'] == user_profile_cluster]
    recommendations = content_based_recommendation(user_profile, cluster_data)
    print("------------- Recommended Laptops----------------")
    print(recommendations)
    json_ = recommendations.to_json()
```

Figure 27

Creating User Profile DataFrame:

Explanation: This function starts by constructing a DataFrame (user_profile_data) containing the user's profile information, such as CPU ranking, RAM, GPU benchmark, minimum price, and company preference. This DataFrame serves as input for generating recommendations tailored to the user's preferences.
Usage: The user's profile information is extracted from the user_profile dictionary passed as an argument to the function. The information is then structured into a DataFrame format for further processing.
Encoding Categorical Features:

Explanation: The function converts the categorical feature company into dummy variables using one-hot encoding. This ensures compatibility with the training data, where the company feature was also one-hot encoded during preprocessing.
Usage: The pd.get_dummies function is applied to the company column of the user profile DataFrame, creating binary indicator variables for each category.
Aligning Feature Names:

Explanation: To ensure consistency with the training data, the function aligns the feature names of the user profile DataFrame with those of the original DataFrame (df) used for training the model. This step ensures that the user profile data has the same features as the training data for compatibility during prediction.
Usage: The reindex method is used to align the columns of the user profile DataFrame with the columns of the training DataFrame (df). Missing columns are filled with zeros.
Predicting User Profile Cluster:

Explanation: The function predicts the cluster to which the user profile belongs using the get_user_profile_cluster function. This step assigns the user profile to a specific cluster based on its features and the clustering model (kmeans) trained on the training data.
Usage: The predicted cluster label (user_profile_cluster) is obtained by calling the get_user_profile_cluster function with the user profile DataFrame, along with the scaler, encoder, and KMeans model.
Generating Content-Based Recommendations:

Explanation: Based on the predicted cluster label, the function retrieves the laptops belonging to the same cluster from the original DataFrame (df). It then generates content-based recommendations for the user using the content_based_recommendation function.
Usage: The function calls content_based_recommendation with the user profile and cluster data to obtain personalized recommendations tailored to the user's preferences and the characteristics of laptops within the predicted cluster.
Displaying Recommendations:

Explanation: The function prints the top 5 recommended laptops along with their similarity scores, user ratings, and prices in Egyptian pounds. It also converts the recommendations DataFrame to JSON format.
Usage: The recommendations are printed to the console for the user to review. Additionally, the recommendations are returned as JSON data, which can be further processed or utilized in other applications.

Overall, the main function orchestrates the entire recommendation process, from preparing the user profile data to generating personalized recommendations based on clustering and content-based similarity. It provides users with relevant laptop suggestions tailored to their preferences and ensures a seamless user experience.

7.6 Accuracy:

Accuracy is a crucial metric in the field of algorithms and data science in general. It measures the consistency between predicted or classified values and the actual values in the data. In the context of algorithms, accuracy is used to gauge how close a model's predictions are to the true values.

It's important to understand that accuracy isn't the sole metric considered when evaluating a model's performance. There are other important performance metrics such as Precision and Recall, and understanding these aspects and how to achieve a balance between them is also crucial.

There are several ways to calculate accuracy depending on the type of problem and data. For example, in classification problems, accuracy can be calculated by measuring the proportion of correctly classified samples to the total number of samples. In the case of predicting continuous values, metrics such as Mean Absolute Error or Mean Squared Error can be used.

Achieving high accuracy in algorithms generally depends on several factors such as the quality of the data used in training, selecting the appropriate model and tuning its parameters correctly, and providing suitable evaluation techniques. By working on improving these factors and ensuring a balance between model accuracy and other performance metrics, the efficiency and effectiveness of algorithms can be enhanced across various applications.

However, it's essential to recognize that accuracy alone might not provide a comprehensive understanding of a model's performance, especially in scenarios where the dataset is imbalanced, or the costs associated with misclassifications vary. Therefore, alongside accuracy, other metrics such as Precision, Recall, F1 Score, and Area Under the ROC Curve (AUC-ROC) are often employed to provide a more nuanced assessment of a model's effectiveness.

Precision reflects the proportion of correctly predicted positive cases out of all predicted positive cases, while Recall represents the proportion of correctly predicted positive cases out of all actual positive cases. F1 Score is the harmonic mean of Precision and Recall and is particularly useful when there's an uneven class distribution.

In addition to these metrics, the Confusion Matrix is a valuable tool for visualizing the performance of a classification model. It displays the counts of true positives, true negatives, false positives, and false negatives, enabling a more granular examination of a model's strengths and weaknesses.

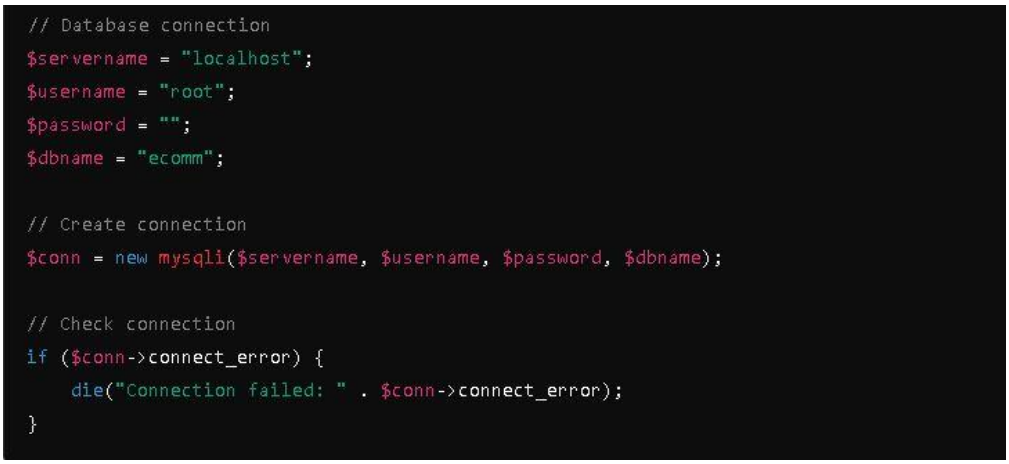
To optimize accuracy, several strategies can be employed, including feature engineering to enhance the quality of input data, model selection and hyperparameter tuning to identify the most suitable algorithm and configurations, and cross-validation techniques to ensure the robustness of the model's performance across different subsets of data.

Moreover, continuous monitoring and refinement of the model are essential to adapt to evolving datasets and changing requirements. By iteratively improving the accuracy of algorithms through these methods and considering a holistic range of performance metrics, data scientists can develop more reliable and effective predictive models across various domains and applications.

Accuracy, or precision, is used to measure the performance of a machine learning model, representing the ratio of correctly classified items among all items in the test data. If the accuracy is 97%, it means the model can classify data correctly 97% of the time, indicating excellent performance and high confidence in classification.

7.7 Integrating AI into the Backend:

Database Connection:



```php
// Database connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "ecomm";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

Figure 28

The code sets the database connection details: server name (localhost), username (root), password (- empty in this case), and database name (ecomm).

It creates a connection to the database using mysqli, creating a conn object.

It checks if the connection was successful. If the connection fails, the script ends with an error message.

Fetch Data from the Database:

```php
// Fetch data from the database
$sql = "SELECT * FROM user_answers";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Initialize an empty array to hold all rows
    $data_array = [];

    // Fetch each row from the result and add it to the array
    while ($row = $result->fetch_assoc()) {
        // Convert numeric values to integers if they are numeric strings
        foreach ($row as $key => $value) {
            if (is_numeric($value)) {
                $row[$key] = intval($value);
            }
        }
        // Add the row to the data array
        $data_array[] = $row;
    }

    // Close the database connection
    $conn->close();
```

Figure 29

- The code defines a SQL query to fetch all data from the user_answers table.
- It executes the query using query and stores the result in result.
- It checks if the result contains any rows (num_rows > 0).
- It initializes an empty array data_array to store the rows.
- It fetches each row from the result using fetch_assoc and converts numeric values to integers.
- It adds each row to the data_array.
- It closes the database connection.

Prepare JSON Data and Send it Using cURL:

```php
    // Set the Content-Type header to application/json
    header('Content-Type: application/json');

    // Encode the data array as JSON
    $jsonData = json_encode($data_array);

    // Initialize cURL
    $curl = curl_init();

    // Set cURL options
    curl_setopt_array($curl, [
        CURLOPT_URL => 'http://127.0.0.1:5000/',
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_POST => true,
        CURLOPT_POSTFIELDS => $jsonData,
        CURLOPT_HTTPHEADER => [
            'Content-Type: application/json',
            'Content-Length: ' . strlen($jsonData)
        ]
    ]);

    // Execute the cURL request
    $response = curl_exec($curl);

    // Check for errors
    if ($response === false) {
        echo "cURL Error: " . curl_error($curl);
    } else {
        // Output the response
        echo "Response: " . $response;
    }

    // Close cURL
    curl_close($curl);
} else {
    echo "No data found.";
}
?>
```

Figure 30

- The code sets the content type to application/json.
- It encodes the data_array as JSON using json_encode.
- It initializes a cURL session using curl_init.
- CURLOPT_URL: specifies the URL to send the request to (localhost in this case).

- CURLOPT_RETURNTRANSFER: ensures cURL returns the result instead of printing it.
- CURLOPT_POST: specifies that the request is a POST request.
- CURLOPT_POSTFIELDS: specifies the JSON data to be sent.
- CURLOPT_HTTPHEADER: sets the content type and the content length of the JSON data.
- It executes the cURL request using curl_exec and stores the result in response
- It checks for any errors in the cURL request, and if an error occurs, it prints the error message.
- If no error occurs, it prints the response.
- It closes the cURL session using curl_close.
- If no data is found in the query, it prints "No data found."

Purpose of the Code

The code retrieves data from the user_answers table in the ecomm database, converts the data to JSON format, and sends it to another server using cURL. The goal is to integrate this data with another system (such as an AI system) for analysis or other purposes.

CHAPTER 8

8 Conclusion

The graduation project represents an exceptional journey towards redefining the electronic shopping experience through the design of an advanced online store dedicated to buying and selling laptops and accessories. It aims to strike a unique balance between technology and innovation, with a focus on enhancing the quality and ease of the digital shopping experience.

At the heart of this project is a smart chatbot that leverages machine learning techniques to guide users toward laptop choices that suit their needs. This chatbot enables effective interaction with users, asking targeted questions to understand their knowledge of laptops and their preferences.

Additionally, the project includes an advanced search system based on artificial intelligence techniques, which facilitates searches and provides accurate results quickly. Complemented by advanced digital marketing strategies, this effort highlights the store and increases social engagement through social media.

The project primarily aims to enhance technical communication between the store and the consumer, enabling every individual, regardless of their technical expertise, to make smart decisions in choosing the laptop that meets their needs. Overall, this project represents a unique exploration of the interaction between technology and digital shopping. It is an excellent opportunity for students to develop skills in design, artificial intelligence, and marketing management.

This avant-garde laptop shopping platform represents a paradigm shift in online commerce, introducing a cutting-edge chatbot that seeks to redefine the user experience when selecting and purchasing laptops. At the core of this innovation is a revolutionary chatbot, meticulously designed to cater to users of varying technological proficiency. Going beyond the conventional model, this intelligent chatbot not only understands and accommodates the needs of tech novices but also offers sophisticated recommendations tailored for more tech-savvy individuals.

The platform's commitment to user-centricity is exemplified through its user-friendly interface that seamlessly integrates state-of-the-art technology with personalized assistance. Unlike traditional online shopping experiences, this platform transcends mere transactional aspects, creating a journey where users not only find the perfect laptop but also enjoy the entire process. The chatbot serves as a digital guide, engaging users in interactive and intuitive conversations, ensuring that even those with limited technical knowledge can make informed decisions.

Through intricate algorithms and artificial intelligence, the chatbot delves into users' preferences, analyzing nuanced queries to provide detailed guidance. This blend of technological prowess and user-centered design fosters an environment where every user, regardless of their expertise, can discover a laptop precisely aligned with their unique expectations and requirements.

As a culmination of these advancements, this graduation project marks a significant step forward in the realm of online laptop shopping. By enhancing accessibility and personalization through the innovative use of artificial intelligence, it transforms the shopping experience into a valuable

resource for both novices entering the tech arena and experts seeking tailored recommendations. This project aspires not only to simplify the laptop selection process but to elevate it into a holistic and enjoyable journey for all users.

The driving force behind this project is a deep-seated motivation to address challenges faced by individuals with limited knowledge in selecting appropriate laptops. This often-overlooked demographic struggles with the complexities of laptop specifications and features, hindering their ability to make informed choices.

Recognizing this gap, the primary objective of our project is to create a meaningful solution tailored explicitly for this segment of society. We envision a user-friendly platform equipped with an intelligent chatbot designed to serve as a digital companion, providing guidance and recommendations in an accessible and comprehensible manner, even for those with limited technological expertise.

The essence of the project is to empower users on the fringes of the tech-savvy spectrum, ensuring they are not left behind in the rapidly advancing world of laptops and technology. The platform aims to bridge the knowledge gap, offering a simplified yet comprehensive experience that transforms the daunting task of choosing a laptop into an accessible and user-centric process.

Our commitment extends beyond mere accessibility; it's about inclusivity and personalization. The intelligent chatbot embedded in the platform employs sophisticated algorithms to understand individual preferences, usage patterns, and specific needs. By doing so, it crafts personalized

recommendations that go beyond the conventional one-size-fits-all approach, acknowledging the uniqueness of each user within this demographic.

In essence, this project serves as a beacon of technological inclusivity, championing the cause of democratizing access to information and technology. By making the laptop selection process more user-centric and tailored, we aim not only to simplify the journey for this segment but also to foster a sense of empowerment, ensuring that everyone, regardless of their technological background, can confidently navigate the landscape of laptop choices and find the device that perfectly aligns with their distinct requirements.

Reference

[1] Lam, Khang Nhut, Nam Nhat Le, and Jugal Kalita. "Building a Chatbot on a Closed Domain using RASA." Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval. 2020.

[2] Prayitno, Philip Indra, et al. "Health chatbot using natural language processing for disease prediction and treatment." 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI). Vol. 1. IEEE, 2021.

[3] Bawack, Ransome Epie, et al. "Artificial intelligence in E-Commerce: a bibliometric study and literature review." Electronic markets 32.1 (2022): 297-338.

[4] Nithuna, S., and C. A. Laseena. "Review on implementation techniques of chatbot." 2020 International Conference on Communication and Signal Processing (ICCSP). IEEE, 2020.

[5] Goyani, Mahesh, and Neha Chaurasiya. "A review of movie recommendation system: Limitations, Survey and Challenges." ELCVIA: electronic letters on computer vision and image analysis 19.3 (2020): 0018-37.

[6] Balush, Illia, Victoria Vysotska, and Solomiia Albota. "Recommendation System Development Based on Intelligent Search, NLP and Machine Learning Methods." MoMLeT+ DS. 2021.

[7] Kim, Jaekyeong, Ilyoung Choi, and Qinglong Li. "Customer satisfaction of recommender system: Examining accuracy and diversity in several types of recommendation approaches." *Sustainability* 13.11 (2021): 6165.

[8] Syamala, Maganti, and Nattanmai Jeganathan Nalini. "A Filter Based Improved Decision Tree Sentiment Classification Model for Real-Time Amazon Product Review Data." *International Journal of Intelligent Engineering & Systems* 13.1 (2020).

[9] Ahmed, M. Z., Singh, A., Paul, A., Ghosh, S., & Chaudhuri, A. K. (2022). Amazon Product Recommendation System.

[10] Singh, Mahesh Kumar, and Om Prakash Rishi. "Event driven recommendation system for E-commerce using knowledge based collaborative filtering technique." *Scalable Computing: Practice and Experience* 21.3 (2020): 369-378.]

[11] Islek, Irem, and Sule Gunduz Oguducu. "A hierarchical recommendation system for E-commerce using online user reviews." *Electronic Commerce Research and Applications* 52 (2022): 101131.

[12] Goyani, Mahesh, and Neha Chaurasiya. "A review of movie recommendation system: Limitations, Survey and Challenges." *ELCVIA: electronic letters on computer vision and image analysis* 19.3 (2020): 0018-37.

[13] Da'u, Aminu, Naomie Salim, and Rabiu Idris. "An adaptive deep learning method for item recommendation system." *Knowledge-Based Systems* 213 (2021): 106681.

[14] Dadhich, Anjali, and Blessy Thankachan. "Sentiment analysis of amazon product reviews using hybrid rule-based approach." *Smart Systems: Innovations in Computing: Proceedings of SSIC 2021*. Springer Singapore, 2022.

[15] Shahbazi, Zeinab, et al. "Toward improving the prediction accuracy of product recommendation system using extreme gradient boosting and encoding approaches." *Symmetry* 12.9 (2020): 1566.

[16] Xu, Kangming, et al. "Intelligent Classification and Personalized Recommendation of E-commerce Products Based on Machine Learning." *arXiv preprint arXiv:2403.19345* (2024).

[17] Marchand, André, and Paul Marx. "Automated product recommendations with preference-based explanations." *Journal of retailing* 96.3 (2020): 328-343.

[18] Sharma, Sunny, Vijay Rana, and Manisha Malhotra. "Automatic recommendation system based on hybrid filtering algorithm." *Education and Information Technologies* 27.2 (2022): 1523-1538.

[19] Osman, Nurul Aida, et al. "Integrating contextual sentiment analysis in collaborative recommender systems." *Plos one* 16.3 (2021): e0248695.

[20] Karn, Arodh Lal, et al. "Customer centric hybrid recommendation system for e-commerce applications by integrating hybrid sentiment analysis." *Electronic Commerce Research* 23.1 (2023).

[21] Balush, Illia, Victoria Vysotska, and Solomiia Albota. "Recommendation System Development Based on Intelligent Search, NLP and Machine Learning Methods." *MoMLeT+ DS*. 2021.

[22] Zhang, Xiaofeng, et al. "A novel hybrid deep recommendation system to differentiate user's preference and item's attractiveness." *Information Sciences* 519 (2020): 306-316.

[23] Guo, Yan, et al. "Maximizing E-Tailers' Sales Volume through the Shipping-Fee Discount and Product Recommendation System." *Discrete Dynamics in Nature and Society* 2020.1 (2020): 7349162.

[24] Shafi, Pathan Mohd, et al. "AI—assisted chatbot for e-commerce to address selection of products from multiple products." *Internet of Things, Smart Computing and Technology: A Roadmap Ahead* (2020): 57-80.