ELECTRONICS PRODUCT RECOMMENDATION SYSTEM

A PROJECT REPORT

21CSC305P –MACHINE LEARNING

(2021 Regulation)

III Year/ V Semester

Academic Year: 2024 -2025

Submitted by

AKHILA TEJASWI [RA2211003011417]
MANOGNA [RA2211003011469]
DIVYASANTOSHI [RA2211003011466]
BHAVANA [RA2211003011439]

Under the Guidance of

Dr. S Shanmugam

Assistant Professor Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY

KATTANKULATHUR- 603 203

NOVEMBER 2024

SRM

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that **21CSC305P - MACHINE LEARNING** project report titled "**ELECTRONIC PRODUCTS RECOMMENDATION SYTSEM**" is the bonafide work of "**AKHILA TEJASWI [RA2211003011417], PAJARLA MANOGNA [RA2211003011469], DIVYA SANTHOSHI [RA2211003011466], BHAVANA GUJJALA [RA2211003011439]**" who carried out the task of completing the project within the allotted time.

Dr. S Shanmugam
Assistant Professor
Department of Computing Technologies
SRM Institute of Science and Technology
Kattankulathur

Dr.G.Niranjana
Professor and Head
Department of Computing Technologies
SRM Institute of Science and Technology
Kattankulathur

Department of Computing Technologies Kattankulathur 603 203 *SRM IST*

ii

ABSTRACT

Recommendation systems help users discover personalized suggestions from a vast selection. Producers use them for cross-selling by recommending additional products, while consumers rely on these systems to find items that match their interests. This value-added relationship fosters customer loyalty. In present e-commerce systems, user pattern search, item, and historical analysis are substantial components of a recommendation system. A better recommendation system based on product specifications and product similarity measures rather than historical data could lead to a progressive change in ecommerce recommendation technologies. This project proposes a model that uses product specifications and various similarity measures to compute user recommendations. The model considers product description and specifications to calculate a similarity measure and then uses these similarity values to form clusters of products. Based on the generated cluster of products, relevant products are recommended to the user. This project presents a method analysis of the various measures and matrices using a sample data set. It also compares the results of our model with the traditionally followed model. The proposed methodology promises to build a user-friendly recommendation system Index Terms—ML.Recommend · Machine learning · ML.NET Recommendation systems · e-commerce recommendation ·ecommerce recommendation systems

The model is designed to address a key goal of recommendation systems: providing personalized and relevant suggestions. By employing machine learning techniques and similarity analysis, it attempts to improve user satisfaction and loyalty while supporting e-commerce strategies like cross-selling.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

RS	-	Recommendation system
CF	-	Collaborative filtering
CBF	-	Content based filtering
RMSE	-	Root Mean Square Error
MAE	-	Mean absolute error
MF	-	Matrix factorization
MBMF	-	Magnitude bounded Matrix factorization
NMF	-	Non-negative Matrix Factorization algorithms
AN	-	Adversarial Networks
SVD	-	Singular Value Decomposition
B-SVD	-	Bounded Singular Value Decomposition
RBM	-	Restricted Boltzmann Machine
CNN	-	Convolutional neural network
RNN	-	Recurrent neural network
DNN	-	Deep neural network
MLP	-	Multilayer Perceptron
AE	-	Autoencoder
NADE	-	Neural Autoregressive

CHAPTER 1
INTRODUCTION

Nowadays, Companies and individual are increasingly trying to buy and sell variety of products online. The most of the transaction are conducted with the help of internet. Not only products, but buying and selling of services are quite popular nowadays. According to Statista, the revenue for ecommerce market was US$1,800,517 globally in 2019 and expected to reach US$2,553,572 million in 2021. Although the digital market like ecommerce companies are not more here in Nepal. Ecommerce development in Nepal is also in growing order which can be seen in figure below. Although the digital market like ecommerce companies are not more here in Nepal. Ecommerce development in Nepal is also in growing order which can be seen in figure below. These figures are projected to increase due to increase in number of mobile internet users which can also be seen in figure above.

The ecommerce market is blooming in advanced and developed country of world. The development of online commerce can also be seen in figure below. (statista, 2020)

MOTIVATION

The idea of building this system is to ease both buyer and seller by recommending products to individual user. Our country Nepal stills lack certain types of ecommerce. There are very limited ecommerce sites in Nepal where user get their desired product.

Nearly all of Nepal's ecommerce platform fails to produce satisfactory revenue from the product till this date. Because of this reason I have built an ecommerce framework for purchasing and selling of numerous products in different categories like books, electronics, fashion, cosmetics, baby products and many more.

The one reason behind the stagnation of growth in the present ecommerce site is due to lack of knowledge about the individual interest of the user and the customer's on-site behavior. Sales are not boosting as expected here in Nepal because it is one of the tough tasks to find and target the right audience in ecommerce. To address this type of problem, I researched the topic of recommendation system to generate the personalized recommendation system for each user who uses our system. This build system uses automated recommendation system which solves the mentioned problem. The developed recommendation system helps the new and existing user to discover relevant and related product recommended from our system based on browsing history, user's behaviors, ratings, demographics and purchase history. The developed system is

Business to customer type of ecommerce where customer may order, buy, rate and review different products online posted by the admin of this site. The web application is very easy to use because of simple and comfortable user-centric interface.

Recommendation systems have had a long history in varying contexts. In this section, we attempt to provide a bird's-eye view. Among the approaches utilized for recommendation technology, collaborative filtering, association rules, and web mining are at the top of the list. Product recommendations in e-commerce usually assess customers' purchasing patterns based on product features. Product recommendation approaches are commonly used to eliminate this additional burden and recommend the scrutinized product to buyers. Systems have been designed for users to predict their preferences for a particular product. These systems recommend an appropriate product based on the customer's tendency and desire. They are the subclasses of information filtering and hence can filter and analyze customer preferences and behaviors. Recommendation systems offer the user an intelligent approach to navigate and search for complex information spaces, which otherwise are not an easy task. Content-based filtering uses item descriptions and user preferences to recommend products that match predefined criteria. In contrast, collaborative filtering relies on the preferences of multiple users, analyzing historical interactions to generate recommendations. It also adapts to individual preference variations. The hybrid method merges content-based and collaborative filtering, recommending items based on both user histories and item descriptions. The concept of similarity is vital in various fields, particularly in recommendation systems, where it compares data objects. Semantic similarity is key for managing information from diverse sources. Common similarity measures include mutual information, cosine coefficient, and Dice coefficient, which work well with categorical data. Euclidean distance calculates similarity by measuring the distance between points and is useful for dense or continuous data. Jaccard similarity evaluates similarity as the ratio of the intersection to the union of sets. While the fundamental concept of similarity remains consistent, the data itself can vary, leading to different data processing methods. Similarity measures are divided into two categories. Formal similarity measures are applied in specific fields such as face verification, speaker verification, visual identity tracking, and recommendation systems. On the other hand, informal similarity measures are broader and more generalized, making them applicable in a wider range of contexts.d to formal measures.

1.1 OBJECTIVES

To develop adaptive e-commerce systems that use AI, semantic agents, and real-time data analytics to deliver dynamic, personalized recommendations, bridging the gap between online and in-store experiences and enhancing customer satisfaction and loyalty. real-time data analytics, and semantic agents, aiming to create dynamic, user-centered shopping experiences that replicate the personalized interaction of traditional markets.

1.2 PROBLEM STATEMENT

The current e-commerce platforms struggle to replicate the personalized, dynamic nature of in-store shopping experiences due to static, predefined recommendation systems. This limitation results in less relevant product suggestions and reduced customer engagement. There is a need for adaptive, AI-driven recommendation models that can analyze real-time data and user context to provide more accurate, personalized recommendations, ultimately enhancing customer satisfaction and loyalty in the digital marketplace.

1.3 SOFTWARE REQUIREMENT SYSTEM

Major Components:

User Interface (UI): Frontend for users to interact with recommendations.

Recommendation Engine: Backend algorithms that process data and generate recommendations.

Data Processing & Storage: Storage for historical data, user behavior, product specifications, and processed insights.

Software and Technology Stack

Sklearn – This module contains multiple libraries having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.

Artifact:

To develop all the artifact given below, Django framework with MySQL database is used. The developed system is ecommerce which can recommend products to similar users of the system.

Artifact 1-Web Application:

The developed system has scalable website design with different features like registration, authentication, search products, product details, categories, sub categories, search bar, user generated review, ratings etc. All the front-end is displayed from the database where only admin can upload all the content. This site has also review system where user can give their view about the specific products and can also give rating from 1 to 5. Admin can add the item to display to other users. The system is also made mobile user-friendly and Search engine optimization for the better performance.

Artifact 2-Product Recommendation:

This site recommends relevant product to users individually based upon the user's behaviors, ratings and history of purchase. Various techniques are used and tested to develop the model. Python is used to develop the AI model for this system.

Scope and limitation of project:

As the main aim of this project is to develop and test the ecommerce system and to integrate recommendation system into web-based application. The scope of the study may include product bought on various part of Nepal but can be used worldwide. The study is not specific to one country or the region. Today's users are demanding very rich system to purchase product online. There is some limitation of the system: I may not work well with newly signed in users due to the fact that the recommendations emerge from a correlation between the intended user and other used focused on collection of ratings.

So, it is almost impossible to judge the user and to categorize a user with very few ratings. It only starts recommending things after user rate some of the products from the system. In similar fashion, new item if not rated by any user, it gets recommended at the last part of the recommendation. Since user only see upper part of the recommendation. The product rated very low also does not appear at the top part of the recommendation. It is also known as "early rater" problem and cold start program. To deal with this problem, homepage of this site does not show recommended things but only shows the latest product from the database and recommended things for the user can only be seen from recommendation webpage. It takes much more time to recommend products to fresh user. The reason behind choosing this project is to show how recommendation can be build, designed and analyzed. The recommendation is that type of

system which can be extended in future and the algorithm used during the development of this project can be improved throughout the period of maintenance of this project. Although the system works perfect in larger dataset.

CHAPTER 2
LITERATURE SURVEY

Recommendation systems have evolved over time, with approaches such as collaborative filtering, association rules, and web mining being at the forefront of technological development. In e-commerce, these systems are designed to assess and predict customers' purchasing patterns, making the process of product selection easier for buyers. By analyzing consumer behaviors, recommendation systems help users navigate vast information spaces, offering suggestions tailored to their preferences. These systems are typically classified into three broad categories: content-based filtering, collaborative filtering, and hybrid methods. Each of these methodologies employs different strategies to recommend products based on user preferences, item descriptions, or a combination of both.

Content-based filtering focuses on recommending items similar to those a user has interacted with before, based on product descriptions and the user's historical preferences. Collaborative filtering, on the other hand, leverages data from multiple users to recommend products that similar users have liked or interacted with. This approach adapts to changing user preferences and can offer personalized suggestions. The hybrid method combines both content-based and collaborative filtering, aiming to harness the strengths of each approach. By considering both item attributes and user histories, hybrid systems can provide more accurate and diverse recommendations.

Central to the success of recommendation systems is the concept of similarity, which helps compare and identify items that are alike. Various similarity measures, including mutual information, cosine similarity, and Jaccard similarity, are used depending on the data type. For example, cosine similarity works well with text-based data, while Euclidean distance is suitable for continuous or dense data. Formal similarity measures are typically applied in specific fields such as face or speaker verification, whereas informal measures are more generalized and can be applied across various domains. Understanding and choosing the right similarity measure is crucial for the effectiveness of recommendation systems in different contexts.

2.1 Types of Recommendation System

Before making recommendation system, one must know the pros and cons of each type of recommender system which are described in this section. Broadly speaking there are two approaches of recommendation system. They are content based filtering, collaborative filtering and hybrid filtering. Each of this type has their own limitation and benefits which is shown in later part of this section. Collaborative filtering is further divided into two type: Model-based filtering techniques and memory-based filtering techniques. Model-based filtering techniques is further divided into clustering, association, Bayesian and neural networks. Memory based filtering technique is further divided into user-user based collaborative filtering and item-item based collaborative filtering. Hybrid filtering is combination of both collaborative and content-based filtering.

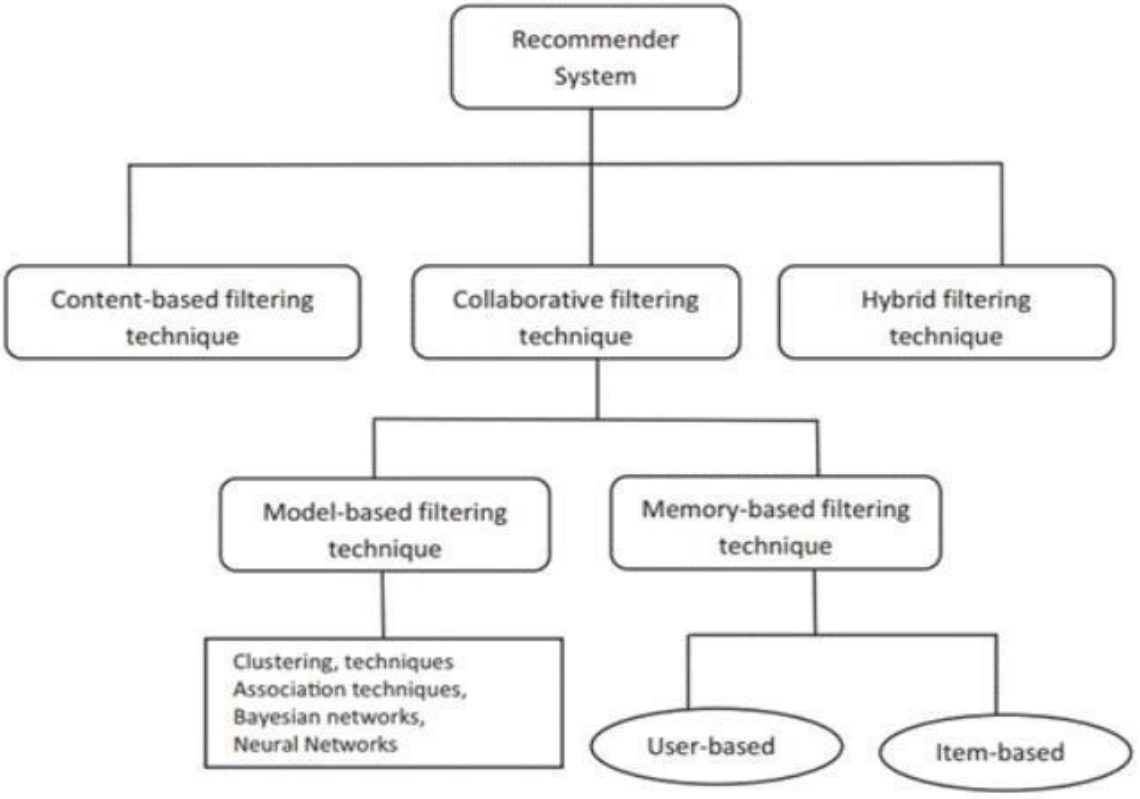

Fig. 2.1 Types of Recommendation System

2.2 Content-based (CB) filtering:

Content-based (CB) filtering, also known as cognitive filtering recommends similar products that the user had already bought from the system. Content-based filtering can also recommend product to user with other factors like age, gender, geography, review, usage pattern etc. The preferences of particular users can be determined by retrieval method such as cosine similarity matrix, term frequency-inverse document frequency, Long Distance Affair etc. or by using machine learning techniques like support vector machines, Naïve Bayes, decision trees etc. (Pazzani & Billsus, 2015)

2.2.1 Term Frequency – Inverse document frequency:

TF-IDF techniques is mostly used in knowledge extraction and processing of data. Term frequency as the name indicates, measures the frequency at which the word occurs which can be calculated as the ratio of total number of particular words occurs in a document and total number of words in the document. (Beel, et al., 2017) Inverse document frequency measures the significance of term in a document and calculated as:

IDF (particular word) = log_e (total no. of documents / no pf documents with particular word in it)

This can be calculated as:

$tfidfweight = \sum_{i \in d} tf_{i,d} * log\ (_{df}{}^{N_i})$

This technique can be used and effective only in the field where there is more textual data.

2.1.1.2 Cosine similarity matrix:

This approach offers the similarity estimation between two items by angle calculation of two vectors. Cosine similarity can be calculated as:

$$similarity = \cos(\theta) = \frac{A \cdot B}{|A||B|}$$

It is the decision which is based on orientation and not on magnitude.

(Sitikhu, et al., 2019)

2.3 Collaborative filtering (CF):

The developers at xerox first coined and developed the term collaborative filtering. Since then, has been enhanced and developed by using different technology and algorithms.

The main goal of recommendation system is to suggest the content based on individuals' desires. Collaborative filtering (CF) suggests the item which are preferred by other related users. There are various types of algorithms that can recommend the item and every algorithm has different set of features. Different algorithm may be great or bad for various datasets depending upon the features. The algorithm can take more time to function better in particular datasets. (HERLOCKER, et al., 2004) Some of the algorithm are described briefly:

2.3.1. Neighborhood-based approach:

Neighborhood-based algorithm can be used to predict the user's desire and opinion. It also analyzes the existing users from the system. In this method, the group of similar users is segregate and their average rating is computed from all users that are close to the actual logged in user. In simple form collaborative filtering is based upon the assumption that similar customers like similar products. So, the main task is to find the similar customer from the set of all the customer available in the database. The one way to find similar customer is from the ratings and reviews given by the customer to different products. Another way is to track all the user's activity in the web application. For the recommendation part first of all, similarity between separated user and active user is calculated by using different techniques which are discussed below:

- . *Pearson Correlation:*

This similarity was introduced in 1994 by Group Lens and is considered as the bestknown algorithm to find the similarity between the users.

Let the profile of the targeted user be x (represented as a vector) and the profile of another user be x (x ∈ T). Here the score of similarity between the target user x and the neighbor y can be calculated by using the following formula.

$$R_{x,y} = \frac{\sum_{i=1}^{n}(xi-x)(yi-\bar{y})}{\sqrt{\sum_{i=1}^{n}(xi-x)^2(yi-\bar{y})^2}}$$

Where,

x_i is the rating given by user to item i, n is the total

number of items and

$\bar{x}$ is the mean rating given by a user and calculated by using the following formula:

$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$

The value of $R_{x,y}$ comes between -1 and +1. If a value is equal to -1, there is a negative correlation between two values and if the value is equal to or near to 1 there is positive correlation. Here x and y are the user whose similarity needs to be computed. (Katarya & verma, 2016)

Constrained Pearson correlation:

(Shardanand & Maes, 1995) proposed constrained Pearson correlation to develop the song's recommendation systems named RINGO. They developed this algorithm from Pearson correlation approach replacing the average rating with average positive and negative level from 1(dislike) to 7(like) and neutral (neither preferred nor hated). The Constrained Pearson can be calculated as:

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - r_z)(r_{v,i} - r_z)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - r_z)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - r_z)^2}}$$

Where u and v are two users whose similarity needs to be computed

$R_{u,I}$ is the rating given by user u to item I and $R_{v,I}$ is the rating given by user v to item i.

2.3.2 Cosine distance:

The user similarity can also be calculated by taking ratings of users and putting two user's rating as vector in an m-dimensional space and compute the cosine of angle between two users as shown in equation below:

$$w_{a,u} = \cos(\vec{r_a}, \vec{r_u}) = \frac{\vec{r_a} \cdot \vec{r_u}}{\|\vec{r_a}\|_2 \times \|\vec{r_u}\|_2} = \frac{\sum_{i=1}^{m} r_{a,i} \, r_{u,i}}{\sqrt{\sum_{i=1}^{m} r_{a,i}^2} \sqrt{\sum_{i=1}^{m} r_{u,i}^2}}$$

where w(a,u) represents the measure of similarity between user a and the active user u.

I is the set of item rated by user a and u

R_{ui} is the rating given by user u to item i and r is the mean rating given by user i

+1(same direction) and -1(opposite direction) represents high correlation and 0(irrelevant) represents no correlation. This approach gives both negative ranking and blank rated product as zero rating. (Liu, et al., 2014)

2.3.3 Adjusted Cosine Similarity:

Adjusted Cosine similarity is necessary to recommend the user based upon the rating of products. The only problem was the user rate products differently to different products. The solution to this problem is to subtract average user rating from rating provided by the user in each product. So, the adjusted cosine similarity is calculated by using the following formula:

$$(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_i)(r_{vi} - \bar{r}_i)}{\sqrt{\sum_{i \in I_u} (r_{ui} - \bar{r}_i)^2 \sum_{i \in I_v} (r_{vi} - \bar{r}_i)^2}}$$

Where, u and v are the two users u is the user to be predicted and v is the active user of the system

2.3.4 Mean squared distance:

Mean squared distance, as name suggests, this approach adds the square difference between the rating of both user u and v alongside their common products which is divided by the total number of common products. This approach underlines the penalizing divergence in taste over resemblance of test. The mean-squared distance can be calculated by using the formula below:

$$s(u, v) = \frac{\sum i \in Iu \cap Iv (ru,i - rv,i)}{|Iu \cap Iv|}$$

Where u is the active user, V is the any other user,
s(u, v) - similarity between active user u and user v,

R_{ui} is the rating given by user u to item i to item I,

R_{vi} is the rating given by user v to item i

(Krishnan & Brahma, n.d.)

2.3.5 Euclidean Distance:

Euclidean Distance can also be used to find the similarity between two users.

The distance between two users can be calculated as:

$$(x, y) = \sqrt{\sum^{n}_{1} (x_i - y_i)^2}$$

Where x_i and y_i are the score of products given by two user u and v for same product n. (Sondur & Chigadani, 2016)

Where fi ∩ fj is intersection between product i and product j for items where users prefer Tanimoto coefficient, sim (i, j) is 1 if two products are completely overlapped and if two products are not overlapped, sim (i, j) is 0. (Lee, 2015)

- ***Spearman Correlation:***

This approach is similar to Pearson correlation, the only difference is instead of finding the similarity between users, this system finds the rank of the user. The spearman correlation can be calculated as:

$$\sum_{i \in Iu \cap Iv}(k_{u,i} \quad (u,v) = \frac{}{\sqrt{\sum_{i \in I_u \cap I_v}(k_{u,i} - \overline{k}_u)2 \sum_{i \in I_u \cap I_v}(k_{v,i} - \overline{k}_v)2}} \quad - k_u)(k_{v,i} - k_v)$$

s

$k_{u,i}$ is the i's ranking for the ratings given by use.
(Sivaramakrishnan, et al., 2018)

- *The clustering based Collaborative filtering:*

This type of recommender system uses cluster of similar users portioned or divided into a different set of data as compared to all other existing users in system. In simpler term, clustering can be called as the grouping of similar user based upon their preference.

There are different types of clustering. Centroid-Based clustering, Distributed-Based Clustering, Connectivity-based Clustering, Density-based clustering are some of them. Any of the mentioned distances can be used for the measurement of user similarity distance. The prediction can be calculated by using the following formula:

$$P_{a,i} = A(\overline{r_{a,i}}) + \frac{\sum_k \{w_{a,k} \times (r_{k,i} - A(\overline{r_{k,i}}))\}}{\sum_k |w_{a,k}|}.$$

Where $P_{a,i}$ is the prediction of user a to item i. $A(\overline{r_{a,i}})$ is the average rating of user a to product I and $A(\overline{r_{k,i}})$ is the average ratings of user k to item I. (Kim & Yang, 2015)

- ***latent factor models:***

Latent factor model is generally called as matrix Factorization models. This model connects user and products in a latent feature space of low dimension $F \in R^d$ where F is the hidden characteristics and d is the latent feature space dimension. For an example, shoes products can

be categorized by their features such as ankle boot, army boots, athletics shoes, ballet shoes, boat shoes, heels etc. A user has hundreds of products could have been bought from 20 category. So, 20 categories are adequate to explain their preferences. The same thing can be applied to products. Singular value decomposition is one of the common techniques for dimension reduction to identify latent factors. The tensor factorization, Bayesian network are other methods for identifying the latent factor.
(Jingying, 2017)

One disadvantage of using content-based recommendation is the system shows always similar type of product all the time. The limitations of content-based recommendation systems are overcome by using collaborative filtering. If the content is not available for the product or if the content is difficult to get still gets recommended to users. No matter what the products look, if purchased user gives good rating to the product it gets recommended to other similar user as well. This system suggests product by looking at the other users(neighbors) that have similar pattern of rating. The neighborhood-based recommendation is favored by various researcher and developer because of its simplicity, efficiency, stability and justifiability. (Desrosiers & Karypis, 2017)

Nowadays, many companies are also trying to use deep neural networks like RBM, Auto encoders, KGE, MLP, AN, AM, RNN, CNN, CF_NADE etc. There are a greater number of hidden layers and better techniques for initialization parameters in Deep neural network. A DNN with large number of hidden units can be more powerful and effective in modeling. Deep neural network in recommendation system is one of the fields that is on the way to perfectionist. Different scientists and researcher are studying and implementing deep neural network for recommendation. Most of the recent work on recommendation are based upon the deep learning-based recommendation system.

I have studied some existing papers, books related to recommendation system and techniques, experiments to complete this project. Some of the techniques, application and evaluation from the papers related to recommendation system are listed below.

One of the journals purposed by (Sharma & Mahajan, 2017) made recommendation system for GitHub. They used collaborative filtering with Pearson coefficient and cosine similarity to find the similar user of GitHub to recommend code to the user. Another paper (Shao, et al., 2020) recommended GitHub repository for paper of academic by using item-item recommender system. They used Graph convolutional Network for the classification purpose on 11,272 papers and 7,516 repositories initially and increased to 32,029 papers to evaluate the

performance in large number of datasets. To recommend better same repository must be stared by multiple user which leads to cold start problem. So, new user repository until and unless stared by other user is not recommended to other user which is the limitation of paper. Despite of this limitations, Graph Convolution Network works much better than other traditional recommender system when there are large number of constrains and features to watch because they can encode both information of node and structure of graph at the same time.

(Wang & Qiang, 2017) have presented a probabilistic framework of neighbors-based recommendation system and multi-layer similarity approach. They also evaluated the model produced from proposed framework using RMSE. A map-based recommender with customer centric interface was done by (Park, et al., 2007) using proposed Bayesian based recommendation system.

In the paper (Jiang, et al., 2018) used novel matrix factorization algorithm and they called their algorithm as Magnitude bounded Matrix factorization to fix user, product prediction fluctuation preceded by conversion of constrained function to unconstrained task. To solve the unconstrained task, they have added a stochastic gradient descent method. They used improved matrix factorization in three real dataset and their algorithm, MBMF indicated strong stability which means this algorithm can be used in larger set of dataset since this algorithm does not repeat the process to determine the best suitable value of k. Bounded matrix factorization, MBMF, Feature-wise updated Cyclic Coordinate Descent method, BMC-ADMM, Bounded SVD gained more accuracy than low level matrix factorization, Non- negative matrix factorization. They used RMSE, MAE and F1score to compare the result.

A research paper published by Phonexay, et al to estimate the performance of movie recommendation. They compared proposed improved k-clique methods with k-nearest neighbors, the maximal clique methods and the k-clique methods in movie lens dataset.
They performed the test in 10 random datasets from the rating of 800 users to the movies. They have taken the user with 200 rated movies, 100 rated movies, 50 rated movies and 20 rated movies in four different parts. To predict the precision in the recommended dataset they used Mean absolute percentage error (MAPE).:
From this experiment compared to than other methods like KNN, Maximal clique, kclique, it was found that the proposed improved k- clique method achieved more accuracy than others. K-clique approach would also increase the precision of the recommendation system but the

only downside is measuring the k-clique approach takes a log time compared to other approach. Right now, KNN is the correct algorithm to use than maximal clique method which is shown in above table. (Phonexay, et al., 2018)

A prototype was made to recommend the users based upon the voice input from the user.by (Kang, et al., 2017) on movie Lens dataset. They collected 347 user's datasets with the help of survey. This is the first recommendation that use natural language processing to detect the desire of the user. They try to solve the real-world problem like what the user really wants from the recommender system not the queries that will work or not in a particular system. The main goal behind this research was to find the subject bias and give the clearer idea to other researcher that the Natural language processing can also be the future to recommendation system. Similar concept was also used in (Chnag, et al., n.d.) where movies are recommended based upon the tag-based preference of users. They recommended the movies based upon the formula:

$$pref_{u,t} = \frac{\sum_{m \in Mu} rating_{u,m} * rel_{m,t} + K * avg_u}{\sum_{m \in Mu} rel_{m,t} + K}$$

Where, m is the movie in set of movies M_u,

rating $_{u,m}$ is the rating given by user u to movie m and $rel_{m,t}$ is the relevance score of movies tag.

In the paper (Zhou, et al., 2018) tried to recommend the display of appropriate advertisement to user in ecommerce company by using very large number of data. They presented the average of five times experimented result from dataset of both Amazon and movie lens dataset. They used deep learning for the prediction of Click Through rate. They developed the base model by using embedding layer to alter high dimensional vectors to low dimensional dense representations followed by pooling layer and Concat layer to deal with various user and their various behavior. To be specific, they used two pooling layers, one is sum pooling for the sum operation and another is average pooling for average operations to embedding vectors. Fully connected layers are used to practice the feature mixing. They used two training techniques: Mini-batch aware regularization and data adaptive activation function. After training with amazon dataset, movie lens dataset and Alibaba dataset. They compared the deep interest network (designed model) model with other existing deep networks like Base Model, Wide and Deep, Deep FM and PNN. The result is better in the developed deep interest network. They

used AUC metric to compare the result with another model and optimizer. They later deployed Deep Interest network (DIN) in Alibaba ecommerce company to display the advertisement to the wide variety of user. Similar to this study, (Zhou, et al., 2019) tries in CTR to estimate the probability of user can click on the product.by using Deep Interest Evolution network(DIEN) and compared this model with other model mentioned above and found out that the Deep Interest evolution Network captures sequential interest efficiently and also models the evolving interest process related to particular product that enforces the performance of large CTR prediction in the advertisement system.

CHAPTER 3
METHODOLOGY

This methodology outlines a systematic approach to developing a recommendation system for e-commerce that provides real-time, personalized, and context-aware product suggestions. The methodology includes phases such as data collection, preprocessing, feature engineering, model development, deployment, and continuous improvement..

3.1 Data Collection and Preparation

Data collection is a critical phase in developing an adaptive e-commerce recommendation system, as it provides the foundation for personalization, context awareness, and model training. The system requires diverse data sources to build accurate, relevant, and dynamic recommendations for users.Data preprocessing is essential to prepare raw data for modeling and ensure accurate predictions. The preprocessing steps involve Analysis of requirement can be defined as to identify system requirements for investigation, documentation all the requirements and analysis of those collected requirement. The major problem behind this project is to build the ecommerce webapp and integrate recommendation system.

In this section we discuss about the fact-finding techniques, function requirement, non-functional requirement and usability requirements.

Sources of data:

The data for the ecommerce site is collected from different existing national ecommerce websites like Daraz, Sasto deal, esewapasal etc. and foreign sites like flipcart, amazon, ebay, Walmart etc. The data for recommendation system is the data provided by amazon.

For making recommendation only, I have also made sample data by myself.

3.2 Design of Modules

The system consists of several modular components that interact to provide real-time, personalized recommendations. Each module is responsible for a specific task, from data processing to recommendation generation and delivery. Below is a high-level design of the core modules involved.

1. Data Collection Module

Purpose: This module collects raw data from various sources, such as user behavior, product attributes, contextual information, and social interactions.

Components:

User Data Tracker: Monitors user interactions (e.g., clicks, searches, views) and gathers session details.

Product Data Aggregator: Collects product information, including specifications, descriptions, images, and prices.

Contextual Data Collector: Uses APIs to gather contextual information, such as location, weather, and device data.

2. Data Preprocessing Module

- **Purpose**: Cleans and prepares raw data for feature extraction and model training.
- **Data Cleaning Unit**: Removes duplicates, handles missing values, and filters irrelevant data.
- **Feature Extraction Processor**: Processes raw product descriptions and user data into usable features (e.g., using NLP for product descriptions).
- **Data Normalizer**: Standardizes data values for consistency across the dataset.

3. Feature Engineering Module

- **Purpose**: Extracts meaningful features from user, product, and contextual data to support accurate recommendations.
- **Components**:
    - **User Feature Generator**: Creates user profiles based on behavior, purchase history, and demographics.
    - **Product Feature Generator**: Transforms product data (e.g., descriptions, images) into vector representations.
    - **Contextual Feature Processor**: Integrates real-time data (e.g., time, location, weather) as contextual features for recommendations.

4. Recommendation Engine Module

- **Purpose**: Generates recommendations based on various algorithms, combining multiple approaches for optimal results.
- **Components**:

a. **Content-Based Filtering Engine**: Recommends products based on user interactions with similar products using vector similarity.
b. **Collaborative Filtering Engine**: Uses user-item interaction data to recommend products that similar users have liked or purchased.
c. **Hybrid Recommendation Engine**: Combines content-based and collaborative filtering, adjusting weights dynamically based on context.
d. **Context-Aware Recommendation Processor**: Adjusts recommendations in real time based on current context (e.g., time, weather).

6. Real-Time Recommendation and Delivery Module

- **Purpose**: Manages the delivery of personalized recommendations to users during their browsing sessions.
- **Components**:

a. **Real-Time Processor**: Uses a low-latency database (e.g., Redis) to fetch and deliver recommendations with minimal delay.
b. **User Interface Integration**: Ensures smooth integration with the front end to display recommendations as users browse..

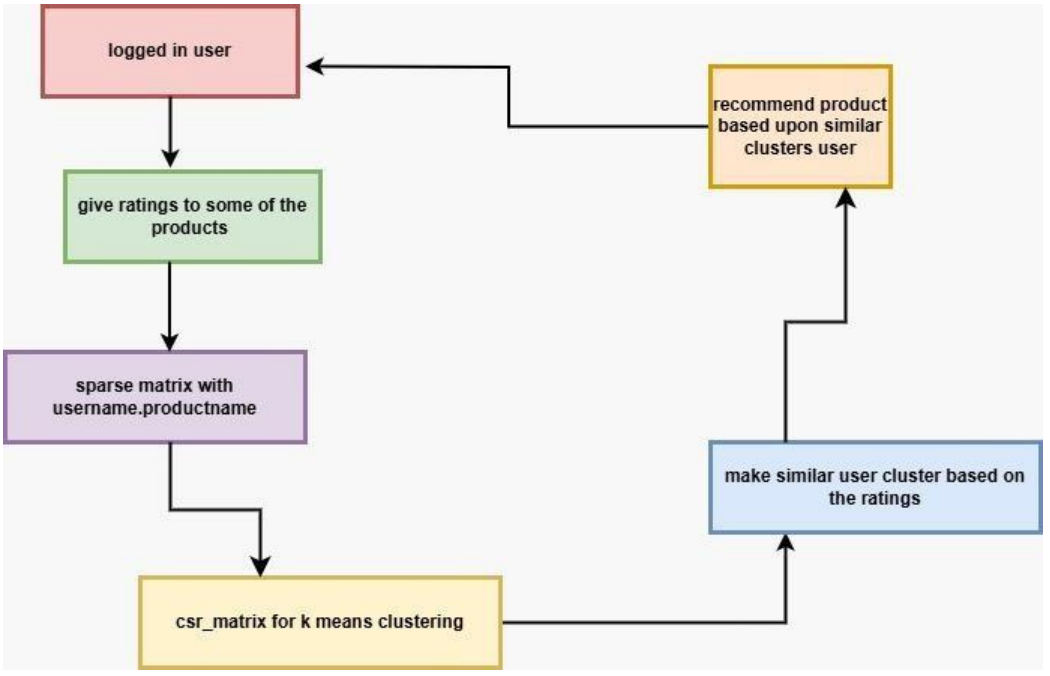

Fig. 3.1 System Architecture

3.2 USECASE DIAGRAM

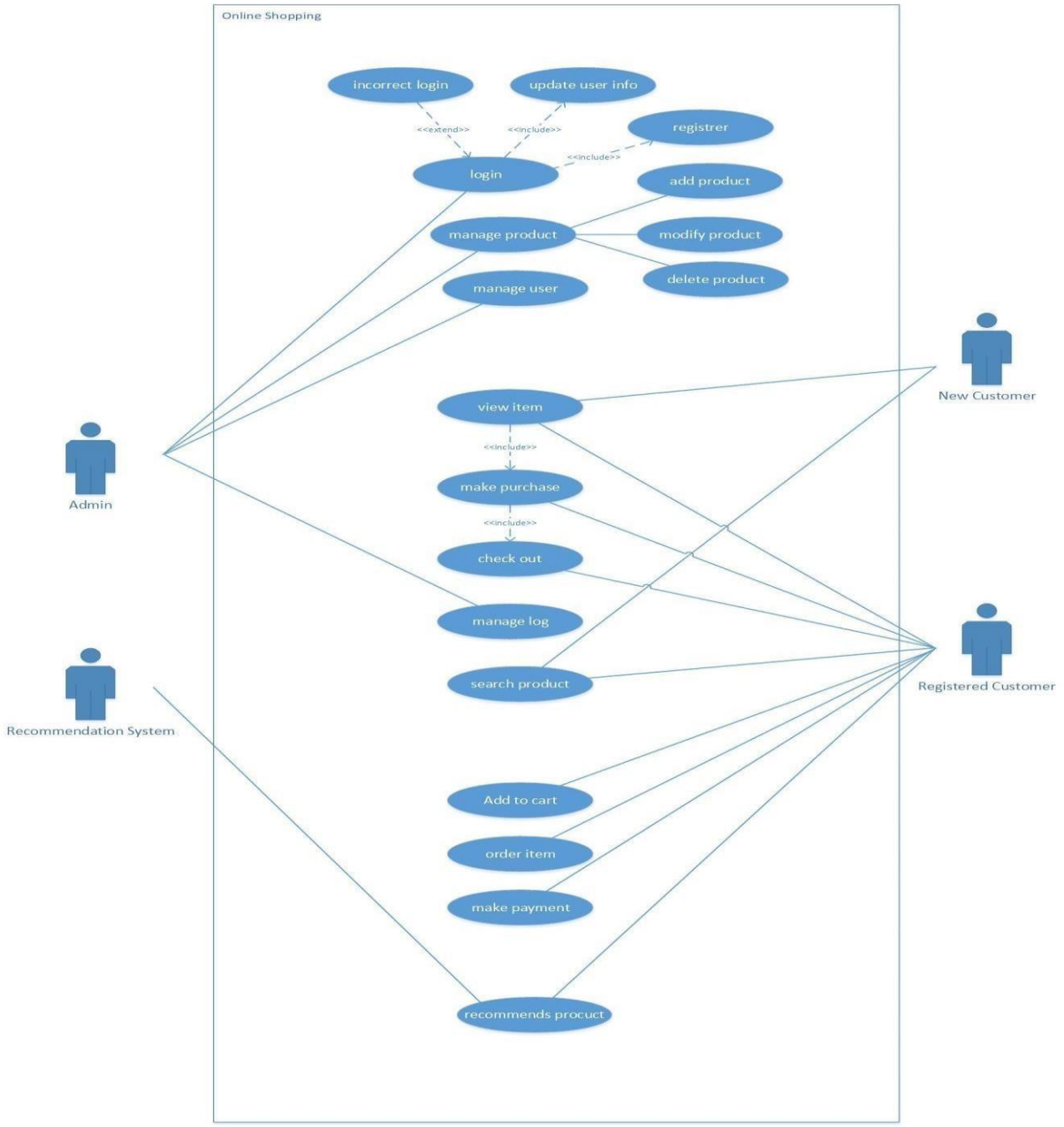

Fig. 3.1 : Usecase Diagram

3.3 FLOWCHART

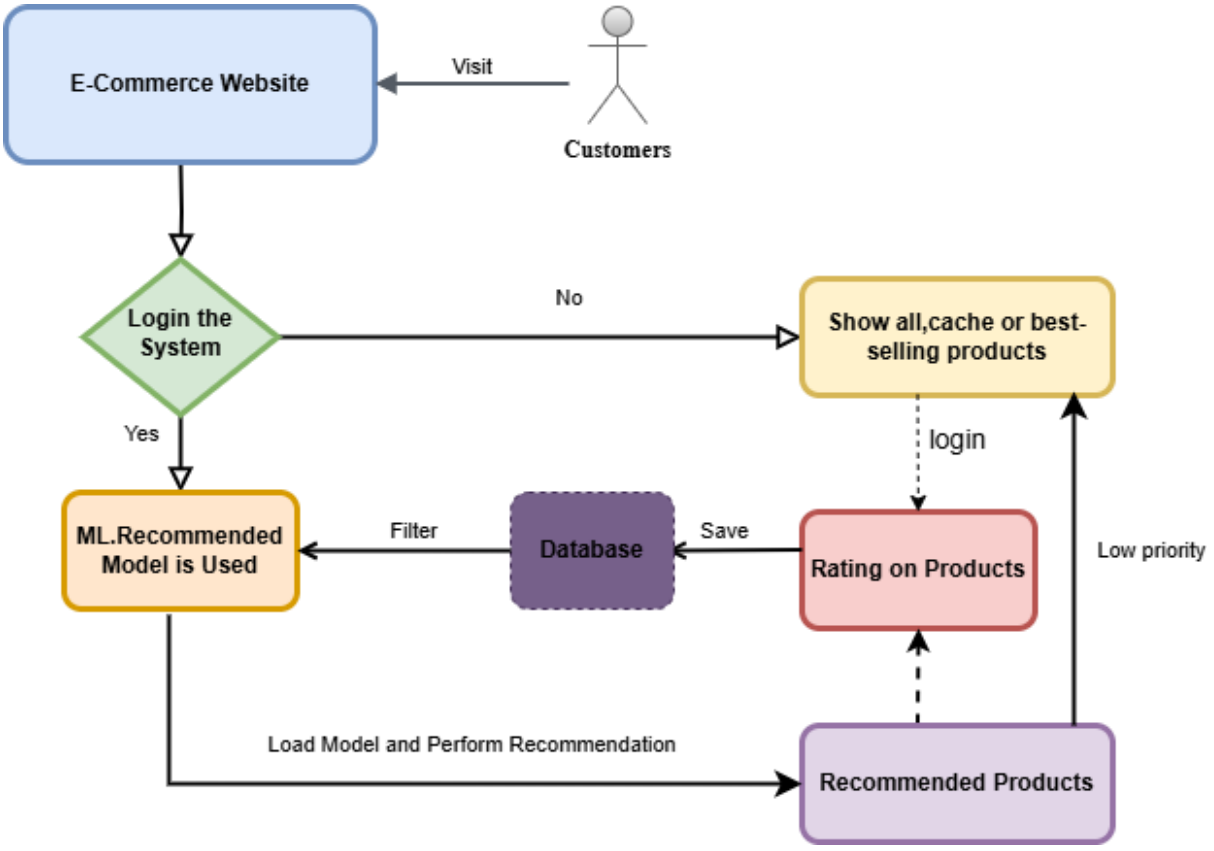

Fig. 4.2 Flowchart

This flowchart represents the process flow of an e-commerce recommendation system integrated with machine learning. Here's a step-by-step explanation:

1. User Visit

E-Commerce Website: A user visits the e-commerce website. This initiates the process of viewing and potentially recommending products.

2. User Authentication

Login the System: The user is prompted to log in, ensuring a personalized experience. If logged in, the system can utilize previous interactions and preferences to enhance recommendations.

3. Product Display

Show All, Cache, or Best-Selling Products: After login, the system initially displays a variety

of products. This can include all products, cached products, or best-selling items to capture the user's interest.

4. Machine Learning Model Application

ML Recommended Model is Used: The machine learning (ML) model for recommendations is loaded and applied. This model filters the available products based on the user's preferences, browsing history, and other criteria, customizing the product list for the individual user.

5. Database Interaction

- Filter: The ML model queries the database to filter products that meet the recommendation criteria.
- Save: The system saves various user interactions, ratings, and preferences in the database for future use in the recommendation model.

6. Product Ratings

Rating on Products: Users can rate products they view or purchase, providing feedback that the ML model can use to refine future recommendations. Product ratings are saved with low priority to avoid affecting the user's current browsing session.

7. Recommendations Generation

- Load Model and Perform Recommendation: The system loads the ML model to generate recommendations based on the saved data, filtering and personalizing products for the user.
- Recommended Products: The recommended products are presented to the user, offering items that are most relevant to their interests and browsing behavior.

Summary

This flowchart depicts a user-centered recommendation system workflow for an e-commerce platform. The process includes initial product displays, ML-driven recommendations, product rating updates, and a continuous database interaction loop for personalized suggestions. This setup aims to enhance the shopping experience by suggesting products that match the user’s interests, potentially increasing engagement and sale

CHAPTER 5

RESULTS AND DISCUSSIONS

In e-commerce systems, it is very important to market products to customers who have needs, which will help the sales of e-commerce systems expand. How can we know what customers need to market the right products to them? When customers go to an e-commerce website, they will be able to post product reviews, based on which customer behavior can be analyzed to market suitable products. The preferences and reviews of customers can change over time, so collaborative filtering of the matrix combined with the time factor to give an appropriate model was researched. In the process, a new nine-step model called ML.Recommend for e-commerce recommendation systems is designed in Fig. The ML.Recommend model was proposed including nine closed steps as shown in Fig. The model is divided into three phases, with phase 1 as data preprocessing and phase 2 as the build model phase In phase 2, the Microsoft ML.NET machine learning model was combined to support the algorithms in the model, which is used in phase 3.

In the data preprocessing stage, dataset was collected from e-commerce sites and queried time filtering along with data cleaning in the first step ①. The data to be processed includes customer data, product data and customer interaction data on products, namely customer reviews for each product data. These data will be extracted over time, and it depends on the needs of the data analyst. Then, the data will be modeled into an object-oriented model in the second step ②, in which the ORM model is used, and the data from the database after the query by time factor are saved in a JSON format. Thus, the ORM mechanism will help the object-oriented modeling system from the JSON format to memory. These classes included will be divided into three namespaces, which are ML.Recommend. Data, that is, a set of data modeling classes for database storage and querying, ML.Recommend Error, that is, a handle exception class and ML.Recommend. Predict, that is, a set of classes to run the model. Details of the class model are presented in the following sections. After having data with the same set of modeling classes, they will be transferred to the third step ③ for processing. In step 3, an algorithm is selected for the model using a collaborative algorithm. Matrix with time-classified data is used as input. Then in the fourth step ④, the dataset will be split in different proportions to train the model. After the model is trained, it will be evaluated in the fifth step ⑤, applying MSE, RMSE and MAE measures to evaluate trained model. From the results of these measures, the user will

decide which model to use. After evaluating the model quality, in the sixth step ⑥, a function is provided to save the model; this function saves the model to the hard drive, so that the user can reuse it later without having to waste computer resources and expended time to re-run the model to help optimize the program. Then in the seventh step ⑦, the user wants to reuse the model, call the model to load function, and the program will restore the model into memory, so that it can be easily used by the user in the eighth step ⑧, where users can optionally select customers and products to recommend, with the results to be used and visualized to customers. Later in the process of using the e-commerce system, customers continue to have other reviews for products at different times in the ninth step ⑨.

The reviews of customers continue to be stored by the system to serve future recommendations; thus, it is a closed and continuously operating system. Through this process, the operating models can be built and reevaluated after a period of time, so that the latest and best recommendations can be given to each customer's preferences. Figure 4 shows detailed flowchart of product recommendation for proposal. Customers often visit the e-commerce system to select products. The system will check if the customer is logged in or not, if a customer logs in to the website, the ML.Recommend model will be used. When the ML.Recommend model is activated, it will load the previously evaluated model file and query the data in the database of the e-commerce platform. The system then runs a recommendation model to find the suitable products for the customer. If customers do not log into the system, a list of new products, previously selected products stored in cache format or best-selling products will be displayed on the website for customers to choose. After the products are displayed to the customer, if any product is viewed and evaluated by the customer, the behaviors on each of these products will be saved back to the database for future recommendations. Also, after the customer logs in, not only the products recommended by the model are shown to the customer, but also new or best-selling products are also displayed but in a descending priority.

USER INTERFACE:

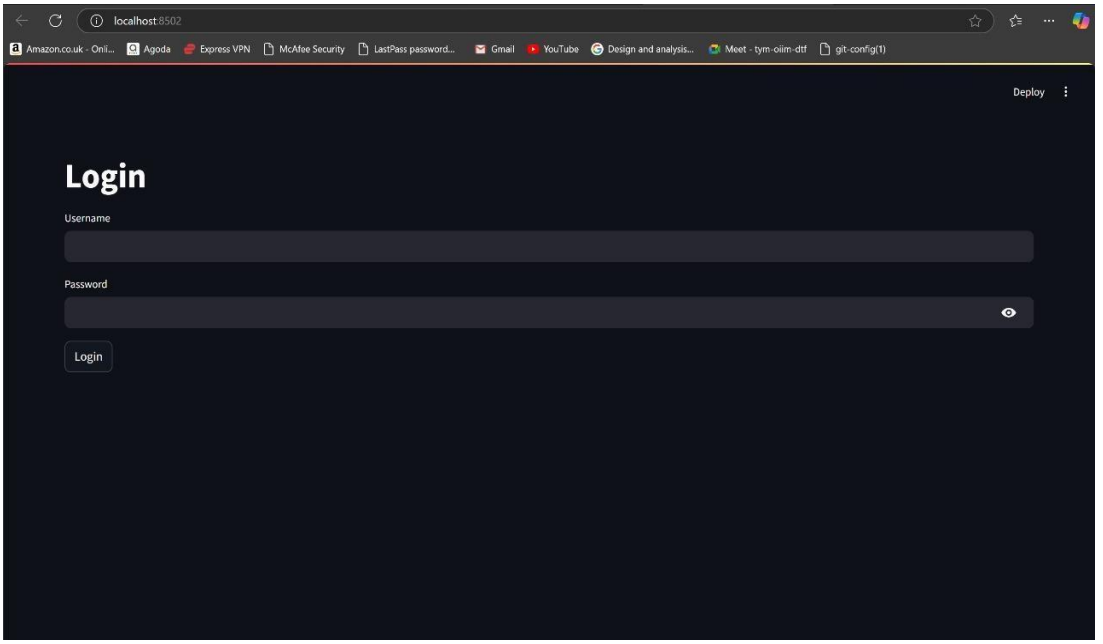

Fig. 5.1 Login Page

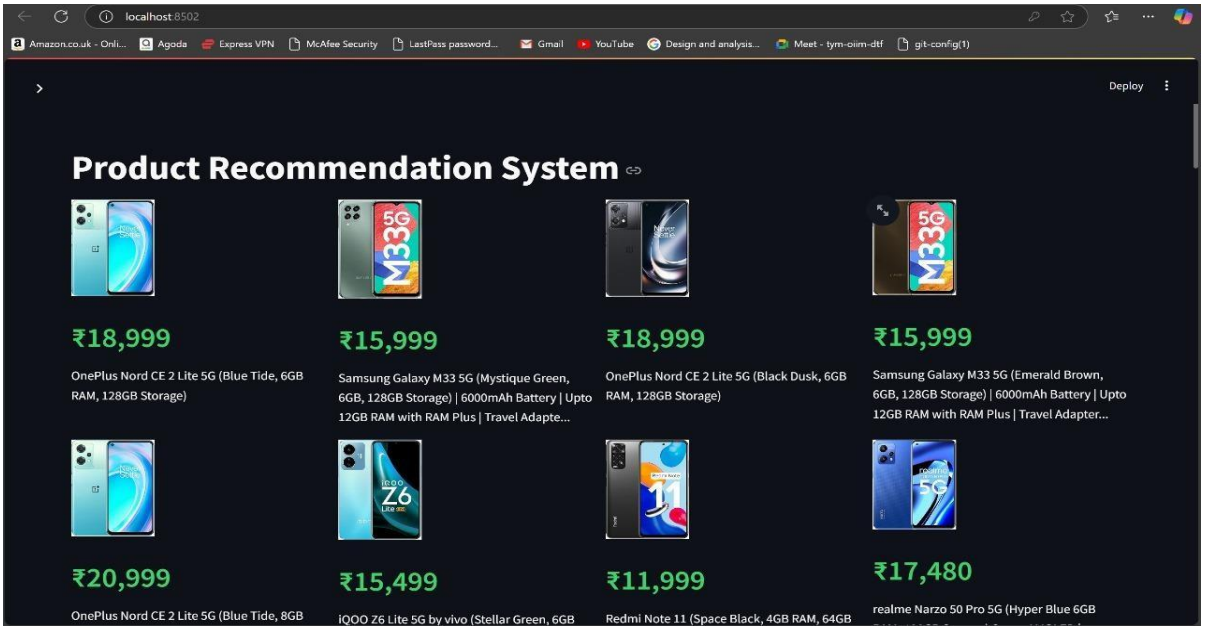

Fig. 5.2: Home Page

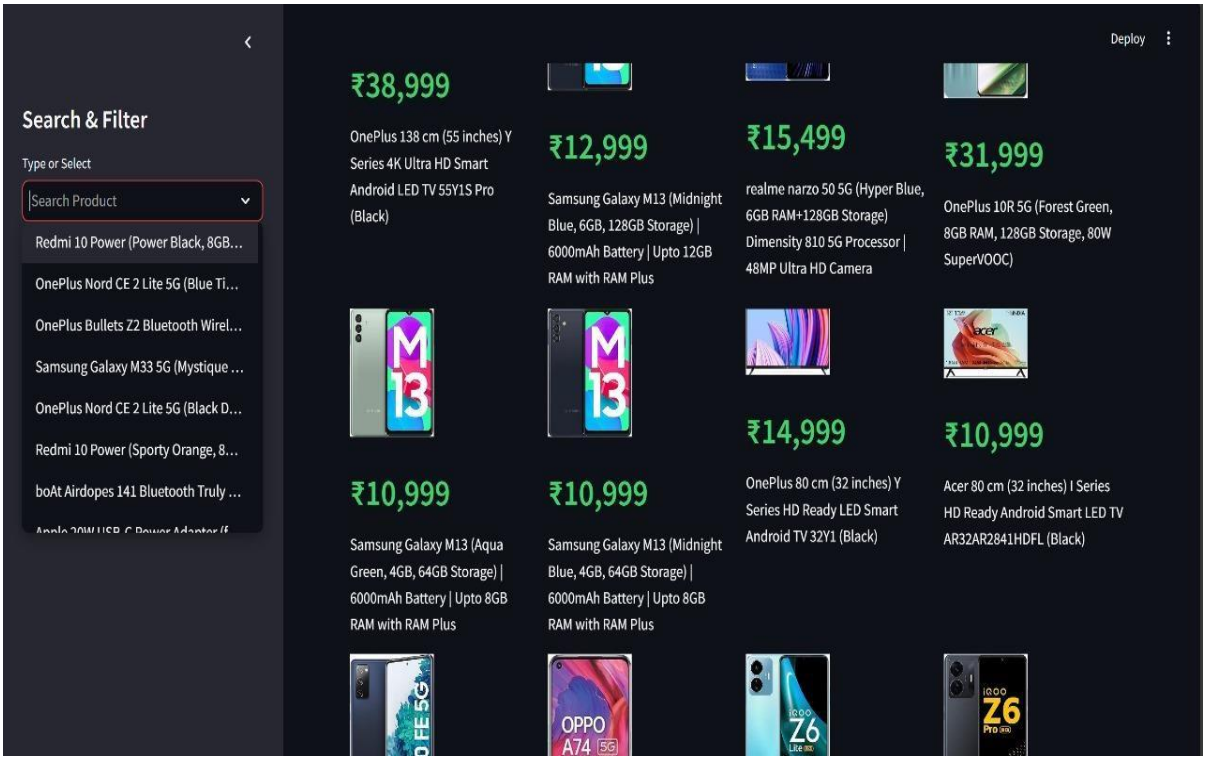

Fig. 3 Search & Filter Tab

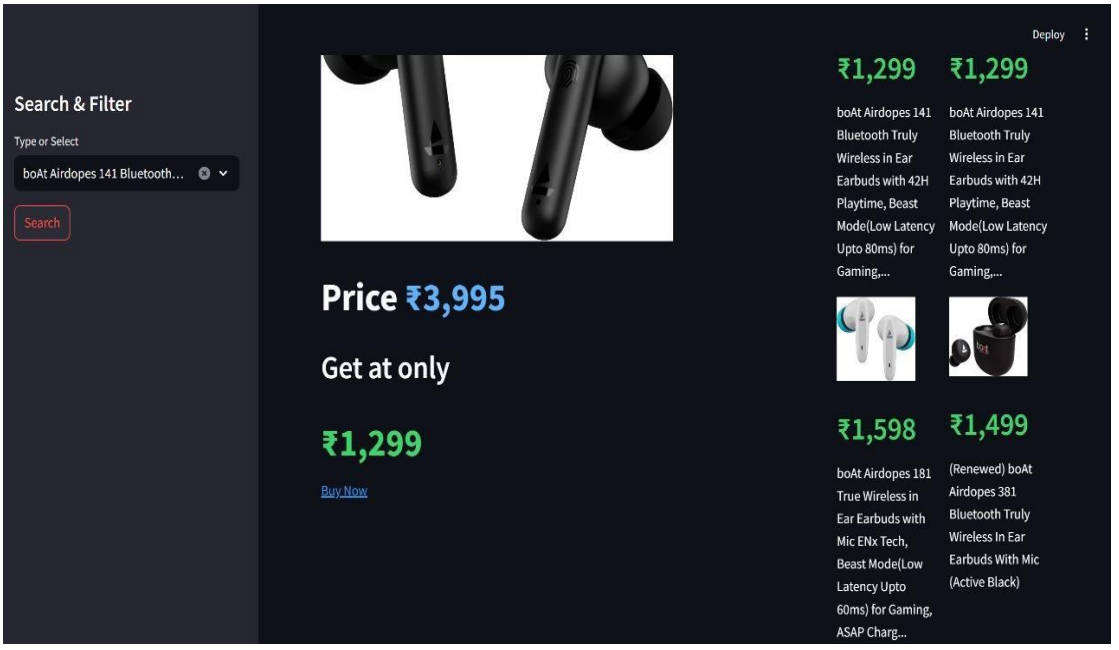

Fig. 4 Recommended Product.

5.2 Clustering Module

- **Purpose**: Groups similar products into clusters to provide category-based recommendations.
- **Components**:

a) **Product Clustering Processor**: Uses algorithms (e.g., K-means, hierarchical clustering) to group products by similarity.

b) **Cluster-Based Recommendation Generator**: Uses clusters to suggest items similar to those the user has shown interest in.

K-Means Clustering

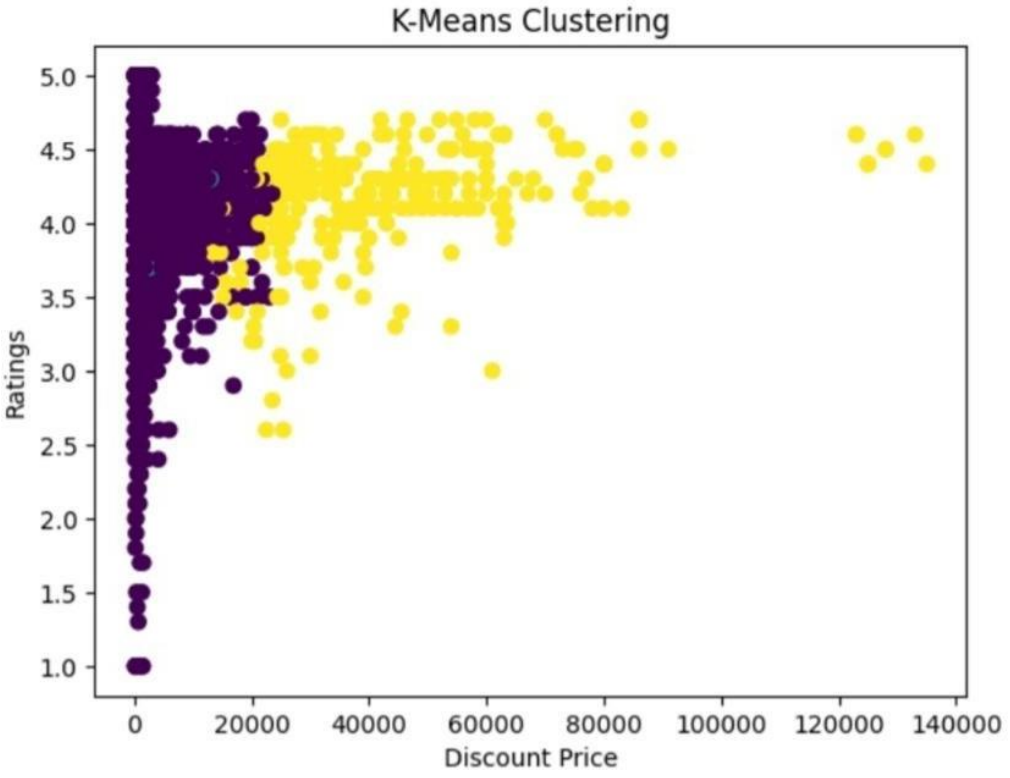

Fig. 3.1: K-Means Clustering

This image is a scatter plot showing the results of K-Means clustering applied to two variables: "Discount Price" on the x-axis and "Ratings" on the y-axis. Here's an analysis of the plot:

1. **Clustering Analysis**: The data points are grouped into two clusters, represented by different colors (purple and yellow). Each cluster indicates a group of data points with similar characteristics in terms of discount price and ratings.
2. **Axes**:
    - **Discount Price (x-axis)**: Represents the price after a discount is applied, ranging from 0 to about 140,000.
    - **Ratings (y-axis)**: Represents customer ratings, ranging from 1.0 to 5.0.
3. **Cluster Characteristics**:
    - **Purple Cluster**: Concentrated on lower discount prices (around 0 to 20,000) and spans a range of ratings, suggesting that lower prices may have varying ratings.
    - **Yellow Cluster**: More spread out along both the x-axis and y-axis, indicating a wider range of both discount prices and ratings.
4. **Interpretation**:
    - Products or services with lower discount prices appear more frequently in the purple cluster, whereas those with higher discounts or varying ratings might fall into the yellow cluster.
    - This clustering can help identify groups of products with similar discounting and rating characteristics, useful for pricing strategies, customer segmentation, or targeted marketing.
5. **Potential Use Cases**:
    - Businesses could use this information to tailor marketing strategies based on cluster characteristics.
    - Insights from the clusters can help determine pricing adjustments or evaluate customer satisfaction relative to discounts.

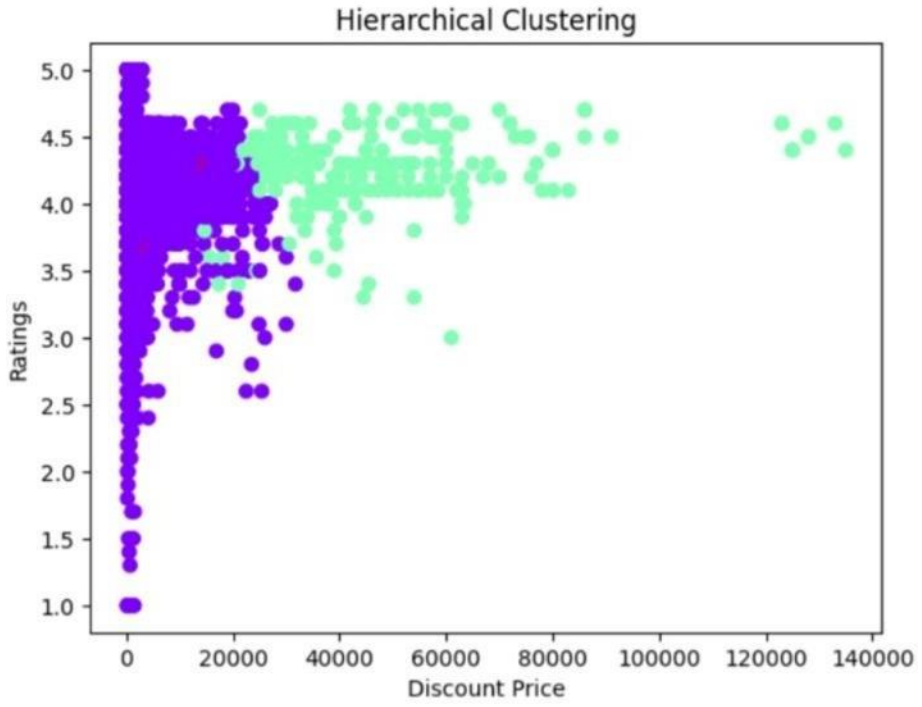

Fig. 3.2 Hierarchical Clustering

These images illustrate hierarchical clustering applied to a dataset with two primary variables: "Discount Price" and "Ratings." Here's a breakdown and interpretation of each plot:

1. Hierarchical Clustering Scatter Plot

- **Description**: The scatter plot shows data points clustered based on two features: "Discount Price" (x-axis) and "Ratings" (y-axis). The points are colored differently to represent two clusters identified by the hierarchical clustering algorithm.
- **Axes**:
  - **Discount Price (x-axis)**: Ranges from 0 to 140,000.
  - **Ratings (y-axis)**: Ranges from 1.0 to 5.0.
- **Cluster Analysis**:
  - The purple cluster is concentrated in the region of lower discount prices, ranging approximately from 0 to 20,000, with a variety of ratings.
  - The light green cluster has a broader distribution across the discount price and ratings scale, indicating that products with higher prices after discounts and variable ratings form a distinct group.
- **Interpretation**:

Hierarchical clustering has separated the data into two main groups based on price and ratings. This can be used to analyze customer preferences for products with different price and rating combinations.

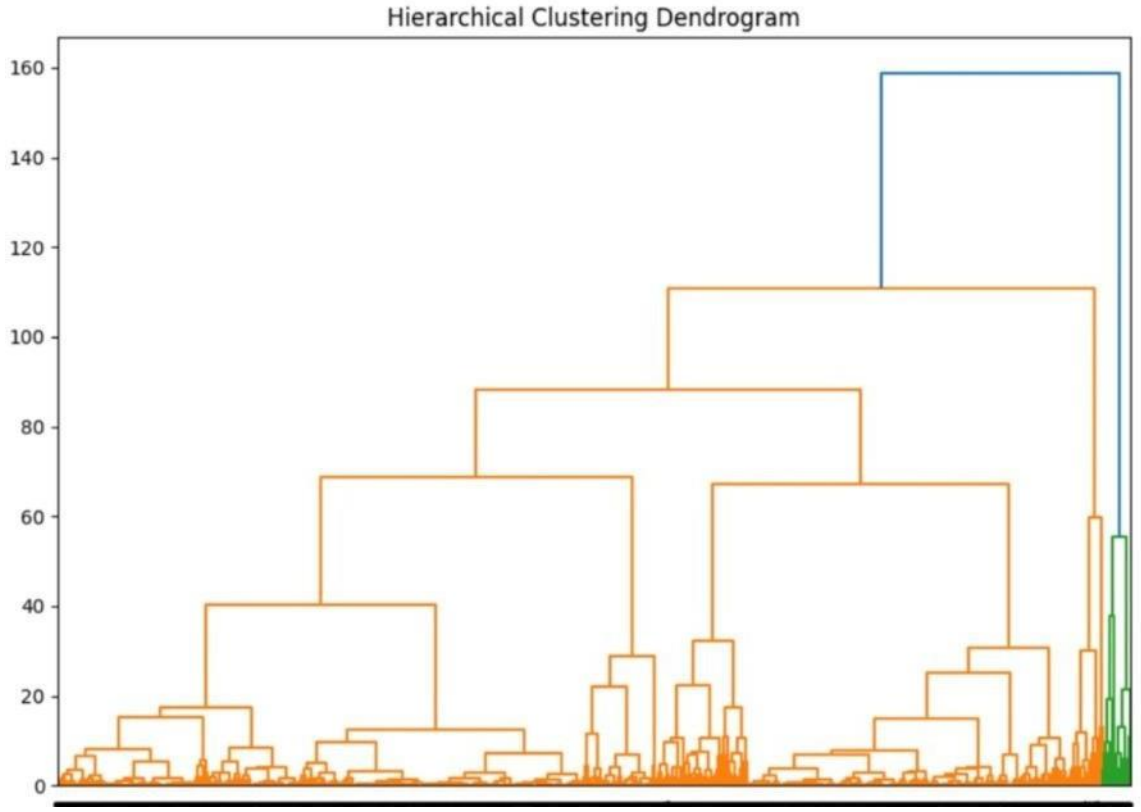

Fig. 3.3 : Hierarchical Clustering Dendrogram

2. Hierarchical Clustering Dendrogram

- **Description**: This dendrogram visually represents the hierarchical clustering process. It shows how individual data points are progressively grouped into clusters.
- **Dendrogram Structure**:
  - The y-axis shows the "distance" or similarity between clusters; larger vertical heights between clusters indicate more distinct groups.
  - At the bottom, each leaf represents a single data point, which are grouped into clusters as you move upward.
  - The large blue line at the top-right separates a major cluster, while smaller green and orange branches indicate sub-clusters within each main group.

- **Interpretation**:
  - The dendrogram helps determine the optimal number of clusters by identifying points where larger gaps in distance occur (e.g., the blue line). This visual can assist in selecting a cluster count, depending on the level of granularity needed for analysis.
  - Cutting the dendrogram at a specific height can yield different numbers of clusters, allowing flexibility in how broadly or narrowly the data is grouped.

Overall Analysis

Hierarchical clustering provides insight into the natural groupings in the dataset. It complements K-Means clustering (from your previous image) by visually representing the data's clustering structure, especially useful for determining an optimal number of clusters.

Applications and Insights

- **Business Decisions**: Clustering helps in identifying different segments of products based on discount price and customer ratings, allowing targeted pricing and marketing strategies.
  - **Customer Segmentation**: The clusters can represent customer preferences, such as a preference for lower-priced products or high-rating items.

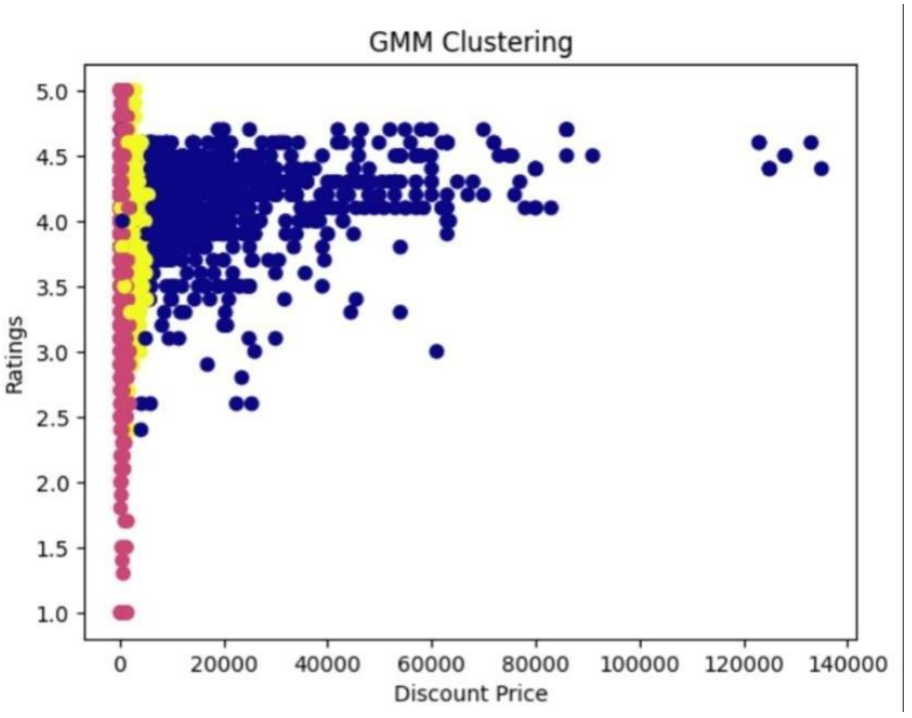

Fig. 3.4 : GMM Clustering

This image shows the results of a Gaussian Mixture Model (GMM) clustering on data with two main features: *Discount Price* on the x-axis and *Ratings* on the y-axis. Here's a breakdown of the clustering and what each element represents:

Axes:

- **Discount Price**: Represents the discounted price of items (presumably in some currency), ranging from 0 to approximately 140,000.
- **Ratings**: Represents customer ratings, ranging from 1.0 to 5.0.

Clusters:

- The data points are grouped into clusters, each represented by a different color (dark blue, yellow, and pink).
    - **Pink Cluster**: Primarily concentrated at the lower end of the *Discount Price* axis. This suggests that items in this cluster have very low or zero discount prices and vary across the entire ratings spectrum (1.0 to 5.0).
    - **Yellow Cluster**: Also concentrated on the left side, close to zero on the *Discount Price* axis. This cluster mostly contains items with ratings above 3.0, indicating a tendency for higher-rated products with low or no discount prices.
    - **Dark Blue Cluster**: Spans across a wide range of *Discount Prices*, from near zero up to 140,000. Items in this cluster generally have ratings above 3.0, but there are some spread out to lower ratings as well.

Observations:

- Most of the data points (in dark blue) are highly concentrated within the lower *Discount Price* range (0–20,000) and have ratings around 3.5 to 5.0, showing a cluster of high-rated items with moderate to high discount prices.
- There are few data points for higher discount prices beyond 60,000, and these are mostly in the dark blue cluster, suggesting that very expensive items may still receive high ratings but are less common in the dataset.

Analysis:

This clustering suggests three distinct groups in the data:

1. Items with low or zero discount prices and a wide range of ratings (pink).
2. Items with low discount prices but generally high ratings (yellow).
3. Items across various discount prices but with higher ratings overall (dark blue).

The GMM clustering helps identify patterns, such as which discount ranges are associated with higher or lower customer ratings. This information can be useful for targeting pricing strategies based on customer satisfaction.

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

As a result of these efforts, a new machine learning model called ML.Recommend for e-commerce product recommendations was built. The model used a matrix factor and time factor combination for product recommendations using ratings and logistic regression for customer comments about products. The ML.Recommend model provides full steps in the machine learning process ranging from data preparation step to initialization, model training, evaluation, storage model, loading model and recommend function, enabling researchers to easily use and save on time on model training when using model saving or loading capabilities. The quality of the model is measured with formulas such as the mean absolute error (MAE), mean square error (MSE), root-mean-square error (RMSE), R-squared and AUC (area under the curve), which are applied to evaluate the model. In addition, ML.Recommend is also compared with other models such as SVD, Slope One, NMF, KNN, Baseline Only and Co-clustering to get general assessments of model quality. The model provide the utilities classes with the following three main namespaces described as: The ML.Recommend.Data namespace contains data model classes of Customer, Product, Rating and the DataUtils class to support data preprocessing; the ML.Recommend.Error namespace contains classes that report errors when performing data preprocessing or running models; and lastly, the ML.Recommend. Predict namespace contains classes that support running the model. The metric class is used to store the quality measure of the model, and the Rating Prediction class is used to store the results of the model execution. Recommend Engine is the class that provides functions to run the model, with this class having important functions of inputting data and splitting the train set to model build, evaluate, save model, load model and predict method in order to find the recommendation products for customers by time factor. The ML.Recommend model was tested with the e-commerce dataset from the website UEL Store and UCI sentiment labeled sentences dataset. The experiment results are presented on desktop software and websites, providing a full range of steps to use the model and provide CPU and RAM performance evaluations. The model produced measurements for an MAE of (0.85) and RMSE of (1.15) in a case training set ratio of 0.1 and an MAE of (0.79) and RMSE of (1.07) in a case training set ratio of 0.9. The recommended results will be updated better by filtering the data by the months closest to the current time to train the model, even though the model quality may be low. Additionally, the ML.Recommend model is published on Microsoft Nuget for researchers to

use and continue to extend the model by adding more features to the open-source code that we provide to serve different research purposes. Nine model files have been trained, and 18 train test set datasets for the model were published for researchers to reuse or verify the quality of the trained model. In the future, the ML.Recommend model will be continued to improve by using Apache Hadoop to support big data analysis and adding more attributes related to a demographic profile, such as gender, age and place of residence, as these attributes have a certain influence on the behavior of evaluating products and services.

6.1 Critical evaluation:

Recommendation systems are the hot topic in online big tech. Constantly, recommendation systems are changing to user's behavior in the system. The recommendation systems are of different types as described in literature review part. The distance between similar user can be calculated by using different approach as shown in Chapter 2. I tried some of the approach for calculating distance. I used latent factor model for recommendation purpose. The matrix is used with user's and their particular rating in product in this system. Most of the ecommerce site uses rating as one of the main components to get the user's preferences on particular topic. So, I did the same thing. I used the rating as the main component for determining or predicting the rating of the particular user. Apart of rating, other implicit features such as like, comment, purchase record, purchase feedback etc. can also be used to recommend products more accurately.

The load time for loading the recommendation is pretty good. To make the system even faster caching can be used. Further enhancements to this system would be to integrate deep learning and hybrid approach for recommending the product. Natural language processing can also be used in comment section of product to detect the user's preferences on certain products.

This system does not have function of editing the review and cannot even reply to the comment. To work recommendation part of this system, the system must not delete any product and users from database since, foreign key of product i.e. product_id and foreign key of user i.e. user_id is used in separate review table of database. Like and dislike system can also be used instead of rating system. Recommendation can also be given to message section. Message replying can also be automated. Chatbot and blog section are one of the main components of ecommerce which can also be integrated.

REFERENCES

[1] S. Demirkol, S. Getir, M. Challenger and G. Kardas, ”Development of an agent based e-barter system,” 2011 International Symposium on Innovations in Intelligent Systems and Applications, 2011.

[2] P . Satheesan, P. S. Haddela and J. Alosius, ”Product Recommendation System for Supermarket,” 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), 2020, pp. 930-935, doi: 10.1109/ICMLA51294.2020.00151.

[3] Kidane, T. Tadele, and R. R. K. Sharma. ”Factors Affecting Consumers’ purchasing Decision through Ecommerce,” Proceedings of the 2016 International Conference on Industrial Engineering and Operations Man agement Kuala Lumpur, Malaysia. vol. 8, no. 10, 2016.

[4] J. Yu et al., ”Collaborative Filtering Recommendation with Fluc tuations of User’ Preference,” 2021 IEEE International Conference on Information Communication and Software Engineering (ICICSE), 2021, pp. 222-226, doi: 10.1109/ICICSE52190.2021.9404120 .

[5] S. H. Choi, S. Kang, Y. J. Jeon, ”Personalized recommendation system based on product specification values,” Expert Systems with Applica tions, vol. 31, Issue 3, pp. 607-616, 2006.

[6] A. R. Lahitani, A. E. Permanasari and N. A. Setiawan, ”Cosine similarity to determine similarity measure: Study case in online essay assessment,” 2016 4th International Conference on Cyber and IT Service Management, 2016, pp. 1-6, doi: 10.1109/CITSM.2016.7577578. [7] S. Temma, M. Sugii and H. Matsuno, ”The Document Similarity Index based on the Jaccard Distance for Mail Filtering,” 2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC

APPENDIX

Clustering

CODE:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
df = pd.read_csv('/content/electronics_product.csv')
df['discount_price'] = df['discount_price'].replace('[₹,]', '', regex=True).astype(float)
df['actual_price'] = df['actual_price'].replace('[₹,]', '', regex=True).astype(float)
# Convert 'ratings' to numeric
df['ratings'] = pd.to_numeric(df['ratings'], errors='coerce')
# Remove non-numeric values in 'no_of_ratings' by setting invalid entries to NaN
df['no_of_ratings'] = df['no_of_ratings'].replace(',', '', regex=True)  # remove commas
df['no_of_ratings'] = pd.to_numeric(df['no_of_ratings'], errors='coerce')
# Select only relevant numeric columns for clustering
numeric_cols = ['ratings', 'no_of_ratings', 'discount_price', 'actual_price']
df = df[numeric_cols]
# Fill missing values with the median for each column
df.fillna(df.median(), inplace=True)

# Feature Scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df)
### 1. K-means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(scaled_features)
```

```
# Plotting the clusters for K-means
plt.scatter(df['discount_price'], df['ratings'], c=df['kmeans_cluster'], cmap='viridis')
plt.xlabel('Discount Price')
plt.ylabel('Ratings')
plt.title('K-Means Clustering')
plt.show()
### 2. Gaussian Mixture Model (GMM) Clustering
gmm = GaussianMixture(n_components=3, random_state=42)
df['gmm_cluster'] = gmm.fit_predict(scaled_features)
# Plotting the clusters for GMM
plt.scatter(df['discount_price'], df['ratings'], c=df['gmm_cluster'], cmap='plasma')
plt.xlabel('Discount Price')
plt.ylabel('Ratings')


plt.title('GMM Clustering')
plt.show()
### 3. Hierarchical Clustering
linked = linkage(scaled_features, method='ward')
# Plot the dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linked, orientation='top', distance_sort='descending',
show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram')
plt.show()

# Apply Agglomerative Clustering
hierarchical_clustering = AgglomerativeClustering(n_clusters=3, affinity='euclidean',
linkage='ward')
df['hierarchical_cluster'] = hierarchical_clustering.fit_predict(scaled_features
# Plotting the clusters for Hierarchical Clustering
plt.scatter(df['discount_price'], df['ratings'], c=df['hierarchical_cluster'],
cmap='rainbow')
plt.xlabel('Discount Price')
```

```
plt.ylabel('Ratings')
plt.title('Hierarchical Clustering')
plt.show()
```

. CODE for Streamlit:

```
import streamlit as st
import pickle as pkl
from helper.recommender_system import recommender, matrix_display

st.set_page_config(layout="wide")

# Sample user data for login check (for demo purposes)
users = {"user": "password", "admin": "admin123"}

# Session state to manage login and cart
if 'is_logged_in' not in st.session_state:
st.session_state.is_logged_in = False
if 'cart' not in st.session_state:
st.session_state.cart = [ ]

# Function to check login credentials
def login(username, password):
return users.get(username) == password

# Login screen
if not st.session_state.is_logged_in:
st.title("Login")
username = st.text_input("Username")
password = st.text_input("Password", type="password")

if st.button("Login"):
if login(username, password):
```

```
        st.session_state.is_logged_in = True
        st.success("Logged in successfully! Please wait...")
        st.experimental_rerun() # Rerun the app to load the product page
        else:
        st.error("Invalid username or password")

        # Product Display Screen (only shown if logged in)
        if st.session_state.is_logged_in:
        st.title("Product Recommendation System")
        st.sidebar.title("Search & Filter")

        # Load product details and similarity matrices
        product_details = pkl.load(open('pickled/version2/Product-details.pkl','rb'))
        similarity_metrix = pkl.load(open('pickled/version2/top10 Similar Products Index
        Matrix.pkl','rb'))
        top_products_index =
        pkl.load(open('pickled/version2/top100_product_indexs.pkl','rb'))

        # Sidebar search functionality
        with st.sidebar:
        selected_option = st.selectbox("Type or Select", product_details['name'], index=None,
        placeholder='Search Product')
        search = st.button('Search')

        # Display the selected product or top products
        if search:
if selected_option is None:
st.warning('Please select a valid option', icon='
else:
        st.header(selected_option)
        product, details = st.columns([0.7, 0.3])
        selected_product_index = product_details[product_details['name'] ==
        selected_option].index[0]
```

```
recommended_ids = recommender(selected_product_index, similarity_metrix)

with product:
purchase, ratings = st.columns(2)
with purchase:
st.image(
'https://m.media-amazon.com/images/' +
product_details.iloc[selected_product_index]['Image url IDS'] + '._AC_UL1500_.jpg',
width=450
)
# Display price and discount information
st.title(f"Price :blue[{product_details.iloc[selected_product_index]['actual_price']}]")
if product_details.iloc[selected_product_index]['discount_price']:
st.header('Get at only')
st.title(f":green[{product_details.iloc[selected_product_index]['discount_price']}]")
url_to_redirect = 'https://www.amazon.in/' +
product_details.iloc[selected_product_index]['product Url Ids']
st.markdown(f"[Buy Now]({url_to_redirect})", unsafe_allow_html=True)

with ratings:
pass

with details:
matrix_display(recommended_ids, product_details, 3, 2)

else:
matrix_display(top_products_index, product_details, 10, 4)
```