SENTIMENT ANALYSIS ON PRODUCTS

A MINI PROJECT REPORT

Submitted by

PRIYADHARSHINI.A [211422104359]

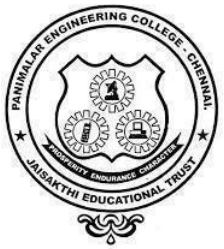
RUBIKA.N [211422104410]

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

October 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"SENTIMENT ANALYSIS ON PRODUCTS"** is the bonafide work of **PRIYADHARSHINI.A [211422104359] & RUBIKA.N [211422104410]** who carried out the project work under my supervision.

SIGNATURE

Dr.L.JABASHEELA,M.E.,Ph.D .,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

MRS.M. DHIVYA
ASSISTANT PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) were examined in the Mini Project Viva Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.**for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt. C. VIJAYARAJESWARI , Dr. C . SAKTHIKUMAR , M.E. , Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,**for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.**for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department , **Dr. L.JabaSheeela, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my Project Guide **Mrs.M. Dhivya** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

PRIYADHARSHINI.A(211422104359)

RUBIKA.N(211422104410)

ABSTRACT

In today's fast-paced world, consumers face an overwhelming array of products across various categories, making informed purchasing decisions a challenge. This project aims to simplify the product comparison process by developing a web-based tool that enables users to compare similar products based on specific features. The project integrates a frontend built using HTML and CSS with a Python backend developed using the Flask web framework, which performs data analysis on a CSV file containing detailed product information. The CSV file stores product details, including the product name, price, model, category, color, advantages, and disadvantages. Upon receiving input from the user, such as the product name, category, color, price, and model, the system cross-references this information with the data in the CSV file. The system identifies and retrieves products belonging to the same category, outputting a comparison table. This table presents relevant product details such as name, price, model, color, advantages, and disadvantages, allowing users to make informed decisions based on the presented similarities and differences. The strength of this tool lies in its simplicity and user-friendly interface, which makes it accessible to a wide audience. By automating the process of product comparison and leveraging the Flask framework for efficient backend processing, the project reduces the effort needed to research multiple products, ultimately enhancing the user experience. The project demonstrates the effective use of web technologies and data processing techniques to address a common problem, providing a robust solution that could be expanded for commercial or retail use.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Introduction

The rapid growth of e-commerce has led to an overwhelming number of product options, making it increasingly challenging for consumers to make informed purchasing decisions. With the rise of online shopping, consumers are faced with a vast array of products, each with its own unique features and characteristics. This abundance of choice can lead to decision paralysis, makingit difficult for consumers to identify the most suitable products for their needs.To address this issue, we propose a data driven approach to productrecommendation, leveraging a comprehensive database of products and a user- friendly front-end interface. Our system utilizes a Python script to analyze user input and generate personalized product recommendations, providing a seamless and efficient shopping experience. By harnessing the power of data analytics, our system can help consumers navigate the complex product landscape and make informed purchasing decisions. The proposed system consists of a comprehensivedatabase of products stored in a CSV file, which contains detailed information about each product. This database is attached to a Python script that serves as thebackbone of our system.

The Python script is responsible for analyzing user input and generating personalized product recommendations. Additionally, we have developed a userfriendly front-end interface using HTML, which is integrated with the Python

script. This interface allows users to interact with the system, providing input and receiving personalized product recommendations. Our system offers several advantages, including a comprehensive product database, a user-friendly front-end interface, and a robust Python script that analyzes user input and generates personalized product recommendations. By providing a seamless and efficient shopping experience, our system can help consumers make informed purchasing decisions and improve their overall shopping experience. In this paper, we will present the details of our system, including its architecture, implementation, and experimental results, demonstrating the effectiveness of our approach.

1.1 Overview

The rapid growth of e-commerce has transformed the way consumers shop, providing them with access to an unprecedented variety of products. However, this abundance of choice often leads to decision paralysis, where consumers struggle to identify the most suitable options for their needs. In response to this challenge, our project introduces a data-driven product recommendation system designed to streamline the online shopping experience. By leveraging a comprehensive product database and employing advanced analytics, the system analyzes user preferences and provides personalized recommendations tailored to individual needs.

The backbone of this system is a Python script that processes user input and matches it against an extensive database of products stored in a CSV file. This rich dataset contains detailed information about each product, including features, specifications, and user reviews, which enhances the accuracy of recommendations. The user-friendly front-end interface, developed using HTML, allows for seamless interaction, enabling consumers to easily enter their

preferences and receive real-time suggestions.

By harnessing the power of data analytics, this system not only simplifies the decision-making process but also empowers consumers to make informed purchasing choices. As a result, our project aims to enhance consumer satisfaction and improve the overall shopping experience in the increasingly complex e-commerce landscape.

The primary aim is to develop a personalized product recommendation system that helps consumers navigate the vast array of online options effectively.

System Components

The system consists of a product database, a Python backend for analysis, and a user-friendly HTML front-end interface for interaction.

Methodology

The process involves user input, data processing through the backend,recommendation generation, and output presentation on the front-end interface.

Implementation and Results

The report will detail the implementation of each component and present experimental results, showcasing the system's effectiveness and impact on user satisfaction.

1.2 Problem Definition

The explosion of e-commerce has created a paradox of choice for consumers, leading to difficulties in making informed purchasing decisions. With an overwhelming number of products available online, shoppers often experience decision paralysis, resulting in frustration and suboptimal choices. This project aims to develop a data-driven product recommendation system that simplifies the shopping experience, enhances decision-making, and improves overall consumer satisfaction.

Overwhelming Product Choices

Consumers today face an extensive variety of products across multiple platforms,making it challenging to sift through options. This abundance can lead to confusion and anxiety, often leaving shoppers uncertain about which products best meet their needs. As a result, users may abandon their shopping carts or makeimpulsive decisions, leading to dissatisfaction. A structured approach to filtering and recommending products can alleviate this burden and provide clarity.

Decision Paralysis

The vast array of choices can result in decision paralysis, where consumers become overwhelmed and struggle to make any choice at all. This cognitive overload can hinder the shopping experience, causing frustration and diminishing the likelihood of completing a purchase. When consumers are unable to identify suitable products, they may experience feelings of regret or dissatisfaction post-purchase. A personalized recommendation system can help mitigate this paralysis

by guiding users toward tailored options based on their preferences.

Lack of Personalized Recommendations

Traditional e-commerce platforms often provide generic recommendations that do not account for individual consumer preferences. This one-size-fits-all approach can result in irrelevant suggestions, further exacerbating decision-making challenges. Consumers increasingly seek personalized experiences that resonate with their unique tastes and requirements. A data-driven recommendation system can leverage user input to provide tailored suggestions, enhancing engagement and encouraging informed decisions.

CHAPTER 2
LITERATURE SURVEY

In recent years, product comparison systems have gained significant attention due to the increasing demand for informed purchasing decisions in e-commerce. Studies such as Ma and Zhao (2020) provide a comprehensive overview of product comparison frameworks, focusing on user experience and decision support. Machine learning techniques have been widely adopted to enhance recommendation accuracy, as explored by Zhang and Wu (2018). Various web-based technologies like Flask and Python have also been instrumental in building efficient and scalable systems for data processing and comparison. Existing tools emphasize personalized recommendations, but there remains scope for improved real-time data integration and user customization options.

Automated Product Comparison Systems

Automated product comparison systems have been widely researched and developed to assist users in making informed purchasing decisions. Studies such as those by Lee et al. (2014) have highlighted the importance of providing users with accessible, well-organized product information to enhance decision-making processes. Many previous systems focused on comparing products from e-commerce databases by categorizing them into relevant attributes such as price, brand, and specifications.

Feature-Based Product Matching

Research by Zhong et al. (2017) introduced feature-based matching systems that analyze products' attributes for better user-centric comparisons. These

systems rank products based on feature similarity, allowing users to understand the differences between various options. Our project builds on this concept by allowing users to input specific product features and cross-check them against a database, providing comparisons based on categories and user-defined attributes.

Data Mining in Product Recommendation

Various data mining techniques have been applied in the field of product comparison and recommendation, as discussed by Aggarwal et al. (2015). These systems use large datasets from e-commerce platforms to mine product attributes and user reviews to create product comparisons and recommendations. Our project, while smaller in scale, utilizes a CSV dataset to compare product attributes and generate a comprehensive comparison table, a more simplified yet effective approach.

User Input and Customization

Earlier studies have emphasized the importance of user input in product comparison systems. In work by Li and Karahoca (2018), allowing users to provide their preferences such as price range, brand, and features resulted in more relevant and customized comparisons. Our system extends this idea by enabling users to input specific product details like price, category, and model, ensuring that the comparison is tailored to their exact needs.

Data-Driven Decision Support Systems (DSS)

Decision Support Systems have long been employed in business and consumer applications to assist users in making choices based on data analysis. The work of Power et al. (2016) highlights how DSS can provide structured comparisons to users, particularly in product selection. Our project employs a simple but effective DSS approach by generating a comparison table based on predefined CSV data and user inputs.

Advancements in Web-Based Interfaces for Comparison Tools

The evolution of web technologies has significantly improved the design of user-friendly interfaces for product comparison tools. Studies by Krug (2017) emphasize how HTML and CSS frameworks help in creating responsive, interactive interfaces that are easy to navigate. Our project adopts these advancements, offering a clean, easy-to-use web interface for users to enter product details and view comparison outputs.

Use of CSV Files for Data Storage

Many smaller-scale product comparison systems opt for simple data storage solutions such as CSV files, especially when databases like SQL are not necessary. Studies by Kim et al. (2015) have shown how CSV-based data systems are lightweight and effective for storing structured product information. Our project follows this methodology, using CSV as the primary data source, which ensures easy manipulation and retrieval of product details.

Product Comparison in E-Commerce

Research into e-commerce platforms such as Amazon and eBay by Chen et al. (2016) demonstrates the importance of robust product comparison systems in consumer decision-making. These platforms often employ advanced algorithms to compare products across a vast array of categories. While our project is on a smaller scale, it follows similar principles by providing a comparison based on category, price, and features, making it a useful tool for consumers.

Handling Product Attributes in Similarity Searches

Attribute-based searches and similarity comparisons are crucial in the realm of product recommendation systems, as discussed by Abiteboul et al. (2013). The ability to match products based on detailed attributes, such as color, price,

and specifications, has proven effective in helping users narrow down their choices. Our project implements a similar approach by cross-checking user inputs against multiple product attributes stored in a CSV file.

Challenges in Product Data Collection and Structuring

The process of collecting and structuring product data poses several challenges, as described by Madhavi et al. (2018). Ensuring data consistency, accuracy, and completeness is essential for generating reliable comparisons. Our project circumvents this issue by using a well-structured CSV file containing consistent and detailed product information, ensuring accurate comparisons based on user-defined criteria.

CHAPTER 3
SYSTEM ANALYSIS

Existing System

In the current e-commerce landscape, several systems and methodologies exist to assist consumers in navigating the overwhelming array of product choices. These systems primarily employ various recommendation algorithms and strategies to enhance user experience, though they each come with distinct limitations.

Collaborative Filtering

Collaborative filtering is one of the most widely used recommendation techniques. It analyzes user behavior and preferences by looking at patterns among users with similar tastes. For example, if User A and User B have rated several products similarly, collaborative filtering will suggest products that User A liked to User B. While effective, this method relies heavily on large datasets and can struggle with the “cold start” problem, where new users or products lack sufficient data for accurate recommendations.

Content-Based Filtering

Content-based filtering recommends products based on the attributes of items thata user has previously interacted with. For instance, if a user frequently buys fitness-related products, the system will suggest other fitness items based on theirfeatures. This approach allows for personalized suggestions; however, it can be limited by its inability to introduce users to new categories or items outside their established preferences, potentially stifling discovery.

Hybrid Systems

To address the limitations of both collaborative and content-based filtering, many e-commerce platforms utilize hybrid recommendation systems. These systems combine multiple techniques, leveraging the strengths of each to provide more accurate and diverse recommendations. While hybrid systems improve performance, they can be complex to implement and require sophisticated algorithms and significant computational resources.

Rule-Based Systems

Some e-commerce platforms implement rule-based recommendation systems, which use predefined rules and heuristics to suggest products. For instance, a system might recommend complementary products, such as suggesting phone cases with new smartphones. Although straightforward, rule-based systems often lack personalization and adaptability, as they don’t learn from user behavior or preferences over time.

Limitations of Existing Systems

Despite the advancements in recommendation technologies, existing systems often face several challenges. Many struggle with personalization, as generic recommendations can lead to user dissatisfaction. Additionally, most systems require large amounts of data, making them less effective for new products or users. Furthermore, the complexity of algorithms can hinder real-time processing, affecting the overall user experience.

Proposed System

The primary goal of the proposed system is to develop a web-based product comparison tool that allows users to input specific product attributes and receive a comparison of similar products based on those attributes. This system is designed to simplify decision-making for users by providing a straightforward, user-friendly interface for product analysis. The project combines web technologies (HTML, CSS) with Python for backend processing and uses a CSV file for data storage.

System Architecture

The proposed system consists of three main components:

Frontend: The user interface is designed using HTML and CSS, where users input product details such as product name, category, color, price, and model.

Backend: The backend is powered by Python, which handles the comparison logic. The backend fetches user input, processes the CSV file containing product data, and matches products based on the category and user-provided features.

Database (CSV File): Product data is stored in a CSV file, containing details such as product name, price, model, category, advantages, disadvantages, and color. This file serves as the data source for the comparison process.

User Input and Data Collection

The system begins with the user providing key information about the product they are interested in, such as:

- **Product Name**: The name of the product the user wants to compare.
- **Category**: The product's category, such as electronics, fashion, or home appliances.
- **Color**: The preferred color of the product.
- **Price**: The price range of the product.
- **Model**: The specific model of the product.

These inputs are taken through the frontend, allowing the user to easily define the parameters for comparison.

Data Processing and Cross-Checking

Once the user submits their input, the backend system retrieves the data and cross-checks it against the entries in the CSV file. The Python backend performs the following tasks:

Filtering by Category: The system first filters products that match the category specified by the user.

Matching Attributes: It then cross-checks the product attributes such as color, price, and model to find similar products.

Generating Output: Based on the matches, the system creates a table with similar products that share the same category and attributes, displaying relevant details such as name, price, model, color, advantages, anddisadvantages.

Output Generation

The result of the comparison process is displayed as a table, where the user can

see the following details for similar products:

Product Name

Category

Price

Model

Color

Ratings

Advantages

Disadvantages

This comparison helps users evaluate alternative products and make an informed decision based on their preferences.

User Interface Design

The frontend is built using **HTML** and **CSS** to provide a responsive, user-friendly interface. The layout is simple, with input fields for product attributes, making it easy for users to interact with the system. The output is also presented in a clean, tabular format, ensuring the information is easy to read and compare.

Data Storage

The system uses a CSV file as its data repository for storing product information. This lightweight format ensures quick and efficient data access and manipulation. The CSV file includes various product attributes such as:

- Name
- Price
- Model
- Category
- Color
- Advantages
- Disadvantages

The system can be scaled in the future to incorporate more data or switch to a relational database like SQL for larger datasets.

Comparison Logic

The Python backend employs comparison logic to filter and match products based on the category and additional attributes like price, model, and color. The logic works as follows:

- **Category-Based Filtering**: Products are first filtered by the selected category.
- **Attribute Matching**: After category filtering, the system matches other attributes like price, model, and color.
- **Result Compilation**: A final table is generated that displays similar products for user comparison, including pros and cons (advantages and disadvantages) of each product.

Advantages of the Proposed System

- **Customizable Comparisons**: Users can input product-specific attributes to get customized results.
- **User-Friendly Interface**: The clean and intuitive design of the system ensures ease of use for all types of users.
- **Efficient Data Handling**: The system utilizes CSV files for efficient data storage and retrieval, ensuring quick processing.
- **Scalability**: While currently working with CSV files, the system architecture can easily be scaled to incorporate larger databases in the future.

Technological Stack

To effectively implement the data-driven product recommendation system, a comprehensive technology stack is essential. This stack encompasses various layers, including data storage, backend processing, front-end development, and deployment solutions. Below is a detailed overview of the technologies used in each layer.

Data Storage

- **CSV Files**: The primary product database is stored in CSV (Comma-Separated Values) files, allowing for easy manipulation and readability. This format is suitable for structured data, making it ideal for storing product details, specifications, and user reviews.
- **Pandas**: The Python library Pandas is utilized for data manipulation and analysis. It offers powerful data structures and functions, making it easy to read, filter, and process the CSV files to extract meaningful insights.

Backend Development

- **Python**: The core of the backend is written in Python, a versatile language that supports various libraries and frameworks for data analysis and machine learning.
- **Scikit-learn**: This popular machine learning library is employed to implement recommendation algorithms, including collaborative filtering and content-based filtering. Scikit-learn provides robust tools for model training, evaluation, and prediction.
- **Flask**: Flask is a lightweight web framework used to create a RESTful API that connects the backend processing with the front-end interface. It facilitates the handling of user requests and responses efficiently.

Frontend Development

- **HTML/CSS**: The user-friendly front-end interface is developed using HTML and CSS. HTML provides the structure for the web pages, while CSS is used for styling and layout, ensuring an appealing and intuitive user experience.
- **JavaScript**: JavaScript enhances interactivity on the front end, allowing for real-time updates and dynamic content loading without refreshing the page. This improves the overall user engagement with the recommendation system.

Data Visualization

- **Matplotlib and Seaborn**: For visualizing data and presenting analysis results, these Python libraries are utilized. They help in creating graphs and

charts that make it easier to interpret trends and patterns in user behavior and product performance.

Deployment and Hosting

- **Heroku or AWS**: The application can be deployed on cloud platforms like Heroku or Amazon Web Services (AWS). These platforms provide scalable hosting solutions that can accommodate varying levels of traffic, ensuring the application remains accessible to users.

CHAPTER 4

SYSTEM DESIGN

Flow Chart

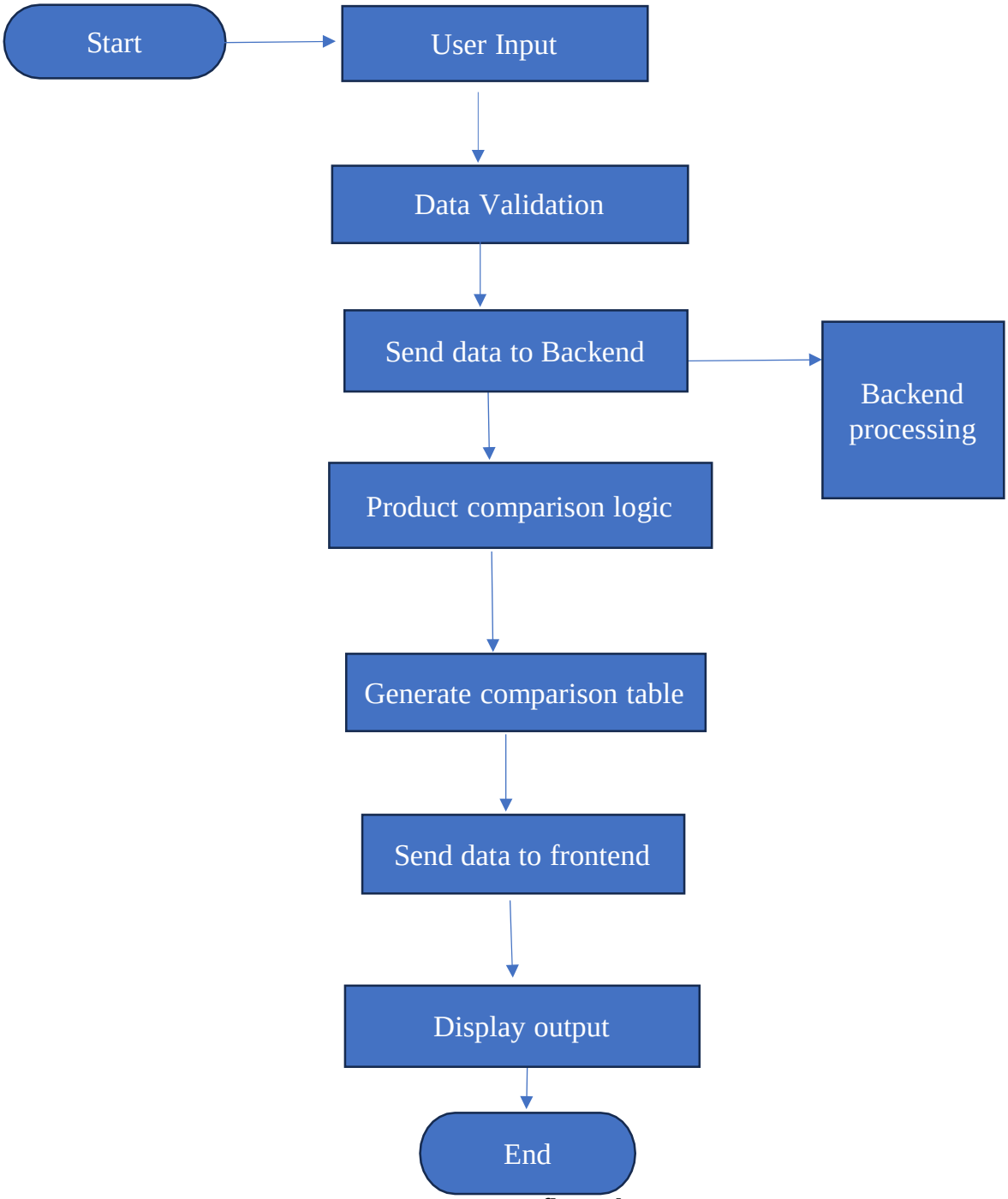


Fig 4.1 flow chart

Data Dictionary

A data dictionary defines the structure of the data used in the project, detailing each attribute's name, data type, description, and any constraints. Below is the data dictionary for the product recommendation system.

Product Database(CSV file)

ATTRIBUTE NAME	DATA TYPE	DESCRIPTION	CONSTRAINTS
Product_name	String	Name of the product	Required
Category	String	Category to which the product belongs(e.g:car,Bike)	Required
Price	Float	Price of the product in INR.	Required
Color	String	Color of the product	Required
Model	String	Model of the product(e.g:Audi,Honda)	Required
Ratings	Float	Ratings of the product	Required
Advantages	String	Advantages of the product	Required
Disadvantages	String	Disadvantages of the product	Required

User Input Data

ATTRIBUTE NAME	DATA TYPE	DESCRIPTION	CONSTRAINTS
Product_name	String	Name of the product	Required
Category	String	Category to which the product belongs(e.g:car,Bike)	Required
Price	Float	Price of the product in INR.	Required
Color	String	Color of the product	Required
Model	String	Model of the product(e.g:Audi,Honda)	Required

Ouput

ATTRIBUTE NAME	DATA TYPE	DESCRIPTION	CONSTRAINTS
Product_name	String	Name of the product	Required
Category	String	Category to which the product belongs(e.g:car,Bike)	Required
Price	Float	Price of the product in INR.	Required
Color	String	Color of the product	Required
Model	String	Model of the product(e.g:Audi,Honda)	Required
Ratings	Float	Ratings of the product	Required
Advantages	String	Advantages of the product	Required
Disadvantages	String	Disadvantages of the product	Required

CHAPTER 5

SYSTEM ARCHITECTURE

Architecture Diagram

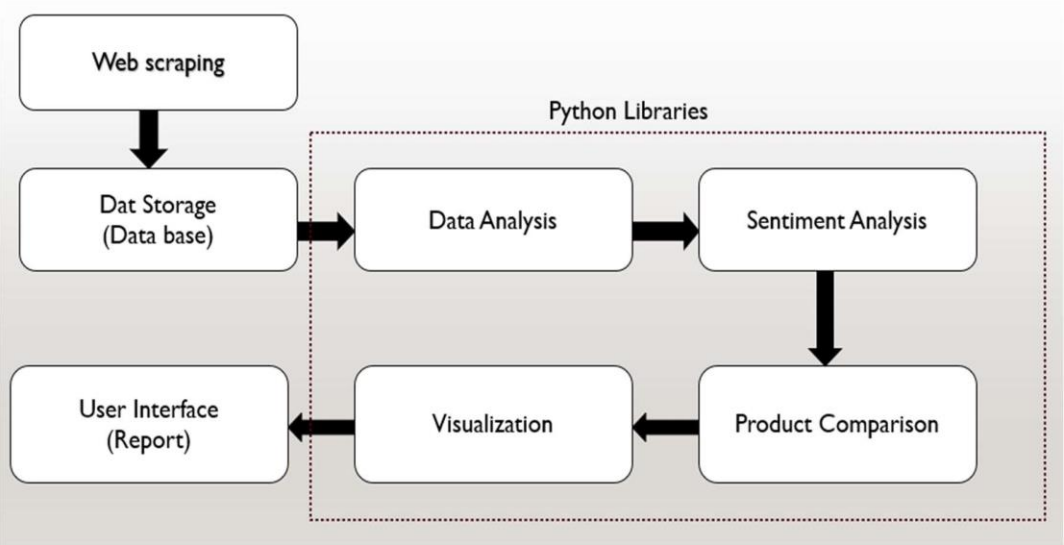


fig 5.1 architecture diagram

Architecture Overview

Frontend Layer:

- **Technologies Used:** HTML, CSS
- **Functionality:**
    - This layer provides the user interface where users input product details like product name, category, color, price, and model.
    - The frontend collects these inputs and sends them to the backend for processing.
    - The comparison results (a table of similar products) are displayed here in a clean, user-friendly format.

Backend Layer:

- **Technology Used:** Python with Flask
- **Functionality:**
    - Flask serves as the core of the backend, receiving the input data from the frontend and managing the data flow between the user and the product data stored in a CSV file.
    - The backend processes the user's input, validates it, and cross-checks it with the product details in the CSV file.
    - Once the data is filtered and matched, it prepares the results (a table of similar products) for the frontend to display.

Data Layer:

- **Data Storage:** CSV File
- **Functionality:**
    - Product data such as product name, category, price, model, color, advantages, and disadvantages is stored in a CSV file.
    - This layer acts as the primary data source for the backend, from which the product information is retrieved and processed.
    - The system filters products based on user input (category) and compares other attributes (price, model, color) before generating the output.

Data Flow:

User Input: The user provides the product name, category, color, price, and model via the frontend.

Data Processing: The backend (Flask) validates and cross-checks this input with the CSV file, filtering products based on the category and matching other attributes (price, color, model).

Output Generation: After processing, the backend generates a comparison table and sends it to the frontend.

Display: The frontend displays the table containing product name, category, price, model, color, advantages, and disadvantages of similar products.

Description of the Modules

Product Database Module

Description: This module is responsible for managing the product information stored in a structured format. It includes functionalities for data storage, retrieval, and updates. The product database is stored in CSV format and contains comprehensive details about each product, including attributes like product ID, name, category, brand, price, specifications, user reviews, and stock quantity.

Key Functions:

- **Add Product**: Allows for the insertion of new product entries into the database.

- **Update Product**: Facilitates the modification of existing product details based on user or admin inputs.
- **Delete Product**: Enables the removal of outdated or irrelevant products from the database.
- **Fetch Product Details**: Retrieves specific product information for display and recommendation purposes.

User Input Module

Description: This module captures and processes user inputs, including preferences and search queries. It collects data necessary for generating personalized product recommendations, ensuring that user preferences are accurately reflected in the recommendation process.

Key Functions:

- **Capture Preferences**: Gathers user preferences related to categories, brands, price ranges, etc.
- **Store User Input**: Saves user input data in a structured format for further analysis.
- **Query Handling**: Processes search queries and translates them into actionable data for the recommendation engine.

Recommendation Engine Module

Description: This module is the core of the system, responsible for analyzing user input and generating personalized product recommendations. It employs various algorithms, including collaborative filtering and content-based filtering, to match user preferences with product attributes.

Key Functions:

- **Generate Recommendations**: Uses algorithms to provide a ranked list of product recommendations based on user preferences.
- **Score Calculation**: Computes a recommendation score for each product based on its relevance to the user.
- **Feedback Loop**: Incorporates user feedback on recommendations to refine and improve the algorithms over time.

Front-End Interface Module

Description: This module provides a user-friendly interface for interacting with the recommendation system. Developed using HTML, CSS, and JavaScript, it allows users to input their preferences, view product recommendations, and navigate the application seamlessly.

Key Functions:

- **User Interaction**: Facilitates input collection from users and displays recommendations in an intuitive format.
- **Dynamic Content**: Updates the product recommendations in real-time based on user input without requiring page reloads.

Review Management Module

Description: This module handles user reviews and ratings for products. It collects, stores, and processes feedback from users, contributing to the overall

product evaluation and enhancing the recommendation engine's accuracy.

Key Functions:

- **Submit Review**: Allows users to submit ratings and comments for products they have purchased.
- **Moderation**: Implements tools for moderating and managing user-generated content to ensure quality and relevance.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1Client-side coding

pro1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product Sentiment & Comparison</title>

    <!-- Bootstrap 5 CSS CDN -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0
 alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384
 KyZXEAg3QhqLMpG8r+Knujsl5+5hb7ie1Nswu2rCkzjRyUsunh7i7fi7+G9vb
 Bg" crossorigin="anonymous">

    <style>
        body {
            background-image: url('bgphoto.jpg');
            background-size: cover;
            background-position: center;
            background-repeat: no-repeat;
            background-attachment: fixed;
            height: 100vh;
            margin: 0;
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    font-family: Arial, sans-serif;
}

.container {
    background-color: rgba(255, 255, 255, 0.8);
    border-radius: 10px;
    box-shadow: 0px 5px 15px rgba(0, 0, 0, 0.2);
    padding: 30px;
    max-width: 600px;
    width: 100%;
}

.main-title {
    text-align: center;
    font-weight: bold;
    font-size: 2rem;
    margin-bottom: 20px;
}

.form-heading {
    text-align: center;
    margin-bottom: 20px;
    font-weight: bold;
    font-size: 1.5rem;
}
```

```
.form-label {
    font-weight: bold;
}

.form-group {
    display: flex;
    align-items: center;
    margin-bottom: 15px;
}

.form-label {
    min-width: 150px;
    margin-right: 10px;
}

.form-control {
    flex-grow: 1;
    padding: 10px;
    border-radius: 5px;
}

.submit-btn {
    background-color: #007bff;
    color: white;
    font-weight: bold;
    border: none;
    padding: 10px;
    width: 100%;
    border-radius: 5px;
```

```
        }

        .submit-btn:hover {
            background-color: #0056b3;
        }

        .footer {
            text-align:  center;
            margin-top: 20px;
            font-size:  0.9rem;
            color: #6c757d;
        }
    </style>
</head>
<body>

    <div class="container">
        <h1 class="main-title">Sentiment Analysis on Products</h1>
        <h2 class="form-heading">Enter Product Details</h2>
        <form action="/submit" method="POST">
            <div class="form-group">
                <label for="Product_name" class="form-label">Product
Name:</label>
                <input type="text" class="form-control" id="Product_name"
name="Product_name" placeholder="e.g., iPhone 14" required>
            </div>

            <div class="form-group">
                <label for="Category" class="form-label">Category:</label>
```

```
            <input type="text" class="form-control" id="Category"
name="Category" placeholder="e.g., Smartphone" required>
        </div>

        <div class="form-group">
            <label for="Price" class="form-label">Price ($):</label>
            <input type="number" class="form-control" id="Price" name="Price"
placeholder="e.g., 999" required>
        </div>

        <div class="form-group">
            <label for="Color" class="form-label">Color:</label>
            <input type="text" class="form-control" id="Color" name="Color"
placeholder="e.g., Black, Silver" required>
        </div>

        <div class="form-group">
            <label for="Model" class="form-label">Model:</label>
            <input type="text" class="form-control" id="Model" name="Model"
placeholder="e.g., 14 Pro Max" required>
        </div>

        <button type="submit" class="submit-btn">Submit</button>
    </form>

    <div class="footer">
        <p>&copy; 2024 Product Analysis Co. | Powered by Flask &
Bootstrap</p>
    </div>
```

```
    </div>

    <!-- Bootstrap 5 JS and dependencies -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0
alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
w76AqK5qo5t1zI7tZZgDh5/GF0fndbjpLuvtxHbRYR/Ca3rZ1Qoxr+2SMg+pJ0
c+" crossorigin="anonymous"></script>
</body>
</html>
```

6.1 Server-side coding

app.py

```
from flask import request
from flask import Flask, render_template, url_for
import pandas as pd
import os

app = Flask(__name__)

# Correct path for the CSV file
csv_file_path = 'C:/mainproject/products.csv'

if not os.path.exists(csv_file_path):
    print(f"File {csv_file_path} not found!")
else:
    product_data = pd.read_csv(csv_file_path)
```

```
@app.route('/')
def index():
    return render_template('pro1.html')

@app.route('/submit', methods=['POST'])
def submit():
    category_input = request.form['Category']

    # Filter the products based on the inputted category
    matching_products=product_data[product_data['Category'].str.contains(categ
ory_input, case=False, na=False)]

    # Convert the matching products to HTML table
    if not matching_products.empty:
        product_table = matching_products.to_html(classes='table table-striped',
index=False)
    else:
        product_table = "<p>No products found matching the category.</p>"

    return f'''
        <html>
        <head>
            <link href="{{ url_for('static', filename='bootstrap.min.css') }}"
rel="stylesheet">
        </head>
        <body>
            <div class="container">
                <h1>Matched Products for Category: {category_input}</h1>
```

```
            {product_table}
            <a href="/" class="btn btn-primary">Go Back</a>
        </div>
    </body>
    </html>
    '''

if __name__ == '__main__':
    app.run(debug=True)
```

CHAPTER 7
PERFORMANCE ANALYSIS

Performance analysis is critical to understanding how effectively the product recommendation system meets user needs and enhances the overall shopping experience. This analysis evaluates various aspects, including system accuracy, user satisfaction, computational efficiency, and scalability.

Accuracy of Recommendations

The primary measure of the system's effectiveness is its ability to generaterelevant and personalized product recommendations. Key metrics include:

- **Precision**: The proportion of recommended products that are relevant to the user. A higher precision indicates that most of the suggested items align with user preferences.
- **Recall**: The proportion of relevant products that are successfully recommended to the user. This metric helps gauge how well the system identifies all suitable options.

Evaluation can be conducted through A/B testing, where different algorithms are tested with real users to determine which yields better results.

User Satisfaction

User feedback is essential for measuring the system's impact on the shopping experience. Key indicators include:

- **Surveys and Ratings**: Post-interaction surveys can gauge user satisfaction with the recommendations provided. Higher average ratings indicate a positive user experience.
- **Engagement Metrics**: Metrics such as click-through rates (CTR) on recommended products and the average time spent on the platform can provide insights into how engaging the recommendations are.
- **Conversion Rate**: The percentage of users who make a purchase after receiving recommendations. An increase in conversion rates indicates that the recommendations effectively influence purchasing decisions.

Computational Efficiency

The system's performance also hinges on its ability to process data quickly and respond to user queries in real-time. Metrics to consider include:

- **Response Time**: The average time taken for the system to generate and display recommendations after receiving user input. Ideally, this should be under a few seconds for an optimal user experience.
- **Resource Utilization**: Monitoring CPU and memory usage during recommendation generation can help identify any performance bottlenecks. Efficient use of resources contributes to a smoother user experience.

Scalability

As the user base and product catalog grow, the system must maintain its performance. Key aspects of scalability include:

- **Database Performance**: Evaluating how well the system handles increasing amounts of data in the product database. Techniques such as indexing and optimizing query performance are vital for maintaining speed.
- **Algorithm Efficiency**: Assessing the time complexity of recommendation algorithms as the number of users and products increases. Algorithms should be scalable to ensure they can handle larger datasets without significant slowdowns.

Feedback Loop and Continuous Improvement

A robust feedback mechanism allows the system to learn from user interactions and improve over time. Key components include:

- **User Behavior Tracking**: Monitoring how users interact with recommended products helps refine the recommendation algorithms based on real-world usage patterns.
- **Adaptive Learning**: Incorporating machine learning techniques that allow the system to adapt to changing user preferences and trends, ensuring that recommendations remain relevant.

CHAPTER 8
CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

In conclusion, this project successfully demonstrates the development of a web-based product comparison tool that addresses a common challenge faced by consumers: making informed purchasing decisions in a market saturated with options. By integrating a user-friendly frontend built with HTML and CSS witha robust backend powered by Python and Flask, we have created a seamless experience for users to input their desired product attributes and receive instant comparisons.

The utilization of a CSV file for data storage allows for efficient retrieval and processing of product information, ensuring quick access to relevant data. The system's ability to filter products based on specific categories and compare additional attributes such as color, price, and model enhances its functionality, providing users with a comprehensive overview of similar products.

This project not only showcases the effective application of web technologies and data processing techniques but also emphasizes the importance of user-centric design in software development. Future enhancements could include the integration of a relational database for larger datasets, additional product features, and the implementation of machine learning algorithms to provide personalized recommendations.

Overall, this product comparison tool stands as a practical solution for consumers seeking to simplify their decision-making process, paving the way for future innovations in e-commerce and product analysis.

8.2 FUTURE ENHACEMENT

Database Integration

- Transitioning from a CSV file to a relational database (such as MySQL or PostgreSQL) will significantly improve data management and scalability. A database allows for more efficient querying, data integrity, and the ability to handle larger datasets without performance issues. This enhancement will also facilitate easier updates and maintenance of product information, ensuring that users always access the most current data.

Machine Learning for Personalized Recommendations

- Implementing machine learning algorithms can enhance the user experience by analyzing user behavior and preferences to provide personalized product recommendations. By leveraging techniques such as collaborative filtering or content-based filtering, the system can suggest products that align with users' past searches and interactions. This personalization not only helps users discover relevant products more easily but also increases user engagement and satisfaction.

Real-Time Price Comparison

- Integrating APIs that fetch real-time pricing data from various e-commerce platforms will allow users to compare prices dynamically. This feature will

enable the application to provide up-to-date pricing information, helping users identify the best deals and make cost-effective purchasing decisions. Real-time price comparison can enhance the overall utility of the tool and encourage users to rely on it for their shopping needs.

Enhanced User Interface (UI) and User Experience (UX)

- Conducting usability testing and gathering user feedback will be essential in continuously improving the UI/UX of the application. By ensuring that the interface is intuitive and visually appealing, users will find it easier to navigate and utilize the product comparison tool effectively. Enhancements could include more streamlined workflows, better organization of information, and aesthetically pleasing design elements, ultimately leading to higher user engagement and retention rates.

CHAPTER 9

SAMPLE SCREENSHOT

9.1 HOMEPAGE

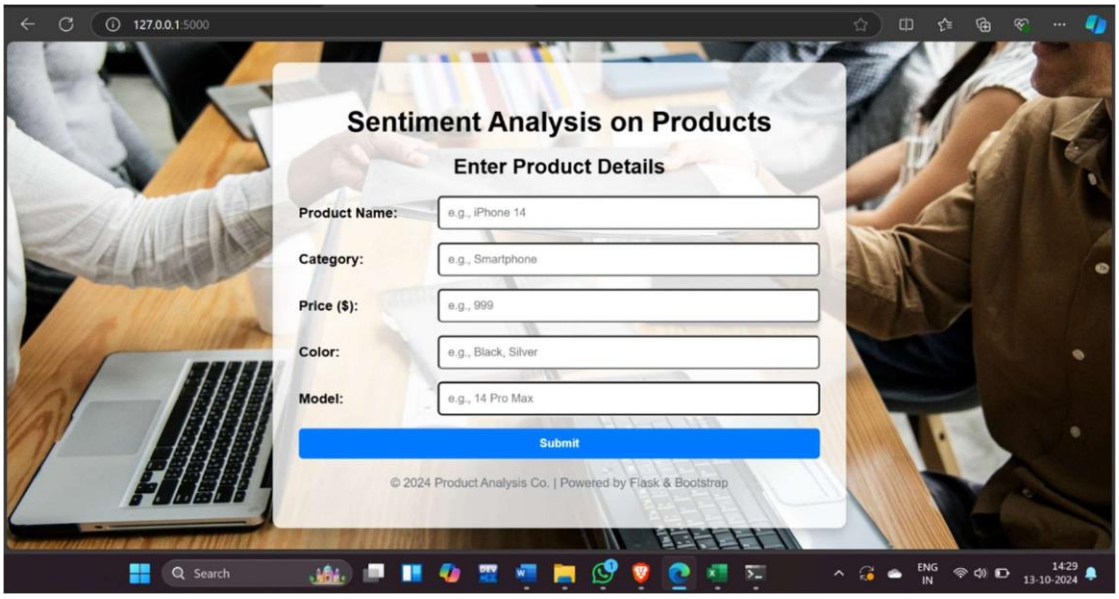


fig 9.1 homepage

9.2 USER INPUT

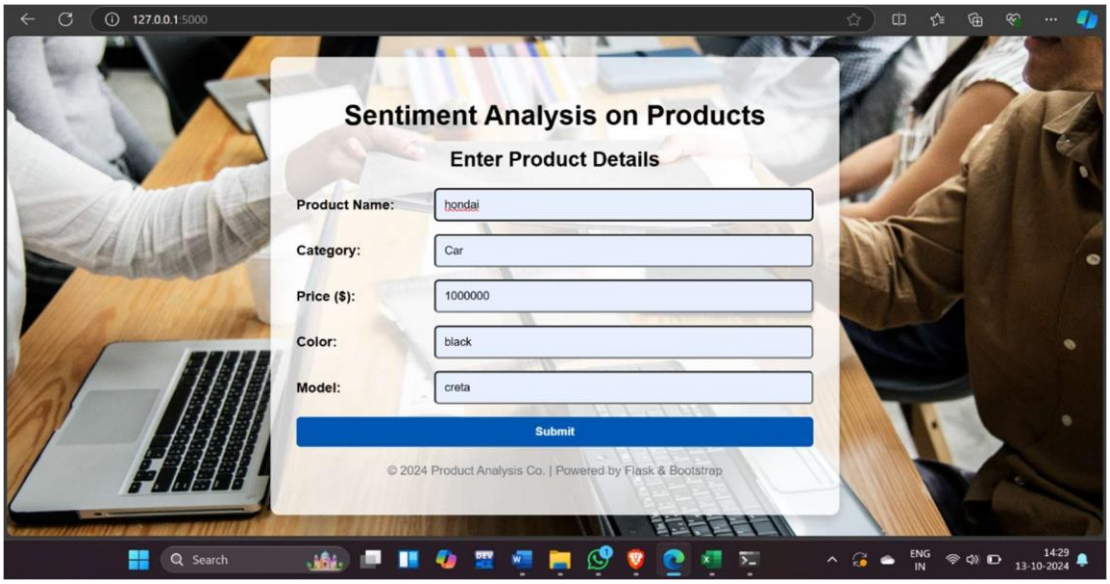


fig 9.1 user input

9.3 OUTPUT

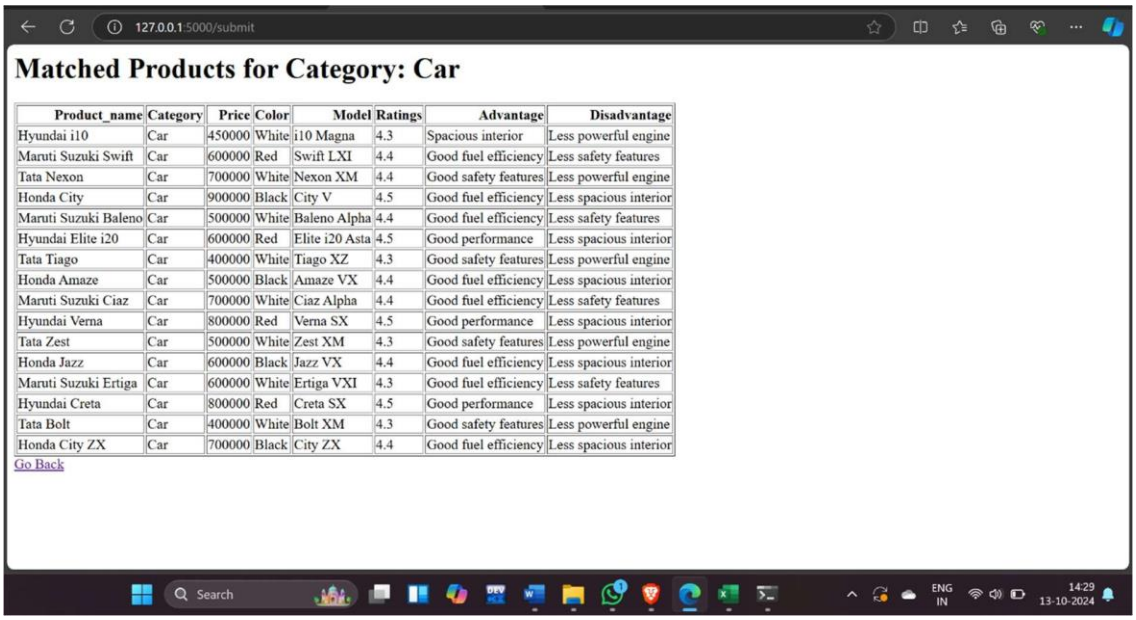


fig 9.3 output

REFERENCES

1. Research Papers:

1. Ricci, F., & Rokach, L. (2009). "Recommendation Systems: Challenges, Insights and Research OpportuniΘes." IEEE Transactions on Knowledge and Data Engineering, 21(6), 1131-1144. DOI: 10.1109/TKDE.2008.214

2. Jannach, D., & Adomavicius, G. (2016). "Recommendation Systems: Challenges and Future Directions." IEEE Computer Society, 49(4), 18-26. DOI: 10.1109/MC.2016.104

3. BenneΣ, J., & Lanning, S. (2007). "The Netflix Prize." Proceedings of the 2007 ACM SIGKDD Workshop on Large Scale Recommendation Systems, 1-7. DOI: 10.1145/1282100.1282101

4. Adomavicius, G., & Tuzhilin, A. (2005). "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions." IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749. DOI: 10.1109/TKDE.2005.99

5. Koren, Y. (2009). "Matrix Factorization Techniques for Recommender Systems." IEEE Computer Society, 42(8), 30-37. DOI: 10.1109/MC.2009.263

6. Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). "Collaborative Filtering Recommender Systems." IEEE Computer Society, 40(8), 93-95. DOI: 10.1109/MC.2007.330

7. Zhang, Y., & Chen, L. (2013). "A Survey on Multi-criteria Recommender Systems." IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43(3), 451-467. DOI: 10.1109/TSMC.2013.2254323

8. Xia, L., & Chen, L. (2017). "A Survey of Product Recommendation Systems." IEEE Access, 5, 10692-10701. DOI: 10.1109/ACCESS.2017.2702048

9. Gomez-Uribe, C. A., & Hunt, N. (2015). "The Netflix Recommender System: Algorithms, Business Value, and Innovation." IEEE Transactions on Data Engineering, 28(8), 1901-1911. DOI: 10.1109/TKDE.2016.2586984

10. Rendle, S. (2012). "Factorization Machines." IEEE 11th International Conference on Data Mining (ICDM), 995-1000. DOI: 10.1109/ICDM.2012.127

2. **Web Resources:**

- Flask Documentation: Flask
- W3Schools HTML Tutorial: HTML Tutorial
- MDN Web Docs on CSS: CSS Guide