

**Project Title:**

Building and Optimizing a Personalized Recommendation  
System for E-Commerce using Machine Learning

## **Abstract**

The aim of this project is to develop a recommendation system utilizing machine learning algorithms to analyze customer behavior and product preferences. The models explored in this project are collaborative filtering, content-based filtering and hybrid approach. The models are tested using different performance metrics: accuracy, diversity and engagement. The dataset utilized in this project is from Kaggle and is about E-commerce Purchase History from Electronics Store. To obtain the real-time dataset of customer purchase history, an E-Commerce platform using modern web technologies is also built further to understand the complexities and behavior of real-world recommendation systems. The result shows that content-based filtering performs better in terms of accuracy and collaborative filtering works better regarding diversity and engagement. Further, collaborative filtering depends upon customer and product interaction to provide accuracy and struggle with new users. Content-based filtering depends upon the attributes of products; therefore, it does not struggle with new users but does not provide unexpected recommendations. The hybrid approach minimizes the limitations of both models and offers the best recommendation utilizing the best features of collaborative filtering and content-based filtering.

## Contents

1. Title .....	4
2. Introduction .....	4
2.1 Research Question .....	5
2.2 Project Objectives .....	5
3. Background and Literature Review .....	6
3.1 Introduction .....	6
3.2 Machine Learning .....	6
3.3 Recommendation System .....	6
3.4 Python Programming Language .....	6
3.5 Prototype .....	7
3.6 Web Technologies .....	7
3.7 The Problem of Over Information and Recommender System .....	8
3.8 State-of-the-Art Recommender System .....	8
3.9 Machine Learning and Tree Boosting System .....	8
3.10 Matrix Factorization Techniques for Recommender System .....	8
4. Dataset .....	10
5. Ethical Issues .....	12
6. Methodology .....	13
6.1 Introduction .....	13
6.2 Data Collection and Pre-Processing .....	13
6.3 Model Training .....	17
6.4 Tools and Techniques .....	19
6.5 Machine Learning Algorithm Implementation .....	20
6.5.1 Collaborating Filtering Algorithm .....	20
6.5.2 Content-Based Filtering Algorithm .....	21
6.5.3 Hybrid Approach .....	24
6.6 Prototype of E-Commerce .....	26
7. Results .....	33
7.1 Accuracy .....	33
7.2 Diversity .....	34
7.3 Engagement .....	34
7.4 Results of Performance Metrics .....	35

7.5 RMSE Results.....	36
7.6 Results of Analysis of Dataset .....	36
7.7 Results of E-Commerce Platform .....	43
8. Analysis and Discussion.....	45
9. Conclusion.....	46
9.1 Future Works .....	46
10. References .....	47

## 1. Title

Building and Optimizing a Personalized Recommendation System for E-Commerce using Machine Learning.

## 2. Introduction

In the modern and technological age, E-Commerce has played a significant role in revolutionizing the way people shop. There are varieties of products that users can easily explore, compare, and buy from E-Commerce platforms, pay the cost online, and get delivered to their doorsteps. However, there is an issue. E-commerce platforms offer consumers all possible products, including groceries, electronics, vehicles, clothes, etc. Therefore, it can be difficult for users to find what they need and challenging for the E-Commerce platform to filter and show only relevant products to users to enhance their shopping experience. This issue can be solved by implementing a personalized recommendation system to analyze user purchase history and behavior from past data and provide personalized products they will likely buy. This helps to enhance the user experience while using the e-commerce platform, helps them gain competitive benefits in this highly competitive market, and also helps to grow sales and revenue.

In the competitive e-commerce landscape, personalized recommendation systems have become crucial for enhancing user experience ([Raghuwanshi and Pateriya, 2019](#)) and boosting sales. This project aims to develop a recommendation system utilizing machine learning algorithms to analyze customer behavior and product preferences. By leveraging the e-commerce purchase history from the Electronics Store dataset, the project will deliver accurate and relevant product recommendations to improve customer engagement and conversion rates significantly. The personalized recommendation system is developed using machine learning technology in Python programming language. Various machine learning algorithms, including collaborative filtering, content-based filtering and hybrid approach, are implemented, and their performances are measured using three different performance metrics: accuracy, diversity and engagement.

Apart from the recommendation system, a prototype of an E-Commerce platform will also be developed using modern web technologies for this project. The E-Commerce platform will be developed using PHP programming language. The front end will be created using HTML and CSS technology. The dynamic content is handled by JavaScript programming language. A database will also be implemented to store the customer and their purchase history. The prototype will help to obtain the live system dataset for each year. The live system dataset will be a valuable asset to understanding the complexities and behavior of real-world recommendation systems. This will also help to understand user interactions deeply and optimize the recommendation system accordingly.

## 2.1 Research Question

How can different recommendation system algorithms be optimized to provide personalized product recommendations in an e-commerce platform, and which model performs best regarding accuracy, diversity, and customer engagement?

## 2.2 Project Objectives

The objectives of the project are listed below:

- Conduct a comprehensive literature review on recommendation systems and optimization techniques.
- Acquire and pre-process the E-commerce Purchase History from the Electronics Store dataset.
- Develop and train various machine learning models, including collaborative filtering, content-based, and hybrid models.
- Analyze model performance based on defined metrics: accuracy, diversity, and customer engagement.
- Develop a prototype of an E-Commerce platform using modern web technologies to collect real-world data.
- Document the findings and prepare a final report summarizing the project outcomes and recommendations.

### 3. Background and Literature Review

#### 3.1 Introduction

In modern E-Commerce platforms, multiple products can be displayed to customers. However, each customer has individual requirements and preferences. Therefore, a personalized recommendation system is key for any E-Commerce platform to show customized products and services to customers that they are likely to buy. This will also help enhance customer experience, and they will likely revisit the E-Commerce platform.

#### 3.2 Machine Learning

The personalized recommendation system can be built using machine learning techniques. The technique is one of the parts of AI (Artificial Intelligence) that is widely popular in the current era ([Kaplan, 2016](#)). Machine learning can learn new things from experience and improve themselves over time without needing further coding. It uses different learning techniques like identifying patterns and other available algorithms ([Thomas and Gupta, 2020](#)). It heavily depends upon data size and data quality to provide better results. Machine learning involves training processes where data are provided to the algorithm, and the algorithm uses that data to identify patterns and make decisions.

#### 3.3 Recommendation System

A recommendation system is software powered by machine learning concepts. Based on those analyses, it can analyze user behaviors, past data, and personal preferences and suggest personalized products and services to customers ([Bollen et al., 2010](#)). It makes use of customer and feature data to provide suggestions to the users. The dataset required for the recommendation system can be obtained from trustworthy sources, such as the Kaggle Website, or collected through an E-Commerce prototype system. The main idea of the recommendation system is to encourage customers to buy more products and services by showing them the products they might buy to increase sales and overall revenue.

#### 3.4 Python Programming Language

Python programming language is a high-level and object-oriented programming language designed in 1990 and released in 1991 by Guido van Rossum ([Sanner, 1999](#)). It can be implemented in any platform due to its feature of converting the code to byte code, which is platform independence. Due to this feature, it works on any modern computer available. It also has an active global support community. The community helps people learn programming languages quickly and solves problems or issues. It has a rich set of libraries and frameworks, making it a suitable programming language for implementing complex machine-learning applications. It also makes it easy to implement various machine learning algorithms, clean data and pre-process data using a dataset, visualize data patterns, and evaluate the machine learning algorithm based on different metrics.

### 3.5 Prototype

A prototype is the initial system version that includes a basic working system before the full version of the application is developed ([Ramirez, 2021](#)). It is designed to show some of the system's key features and what to expect from the final version of the system. This is done to get feedback from the stakeholders about the product, validate the assumptions made, collect the required data, and so on. The e-commerce prototype helps test the functionality of e-commerce, such as product display and user purchase activities, to collect yearly data of customer purchase history and implement the recommendation system to the e-commerce.

### 3.6 Web Technologies

Web technology is a technique used to create web applications that run in web browsers ([Bhardwaj, 2021](#)). Web technology helps users communicate with other devices and systems over the internet using markup language and multimedia content. It provides dynamic user experiences and contains a visible frontend for users and a backend where the actual processing occurs. HTML is used to create the web application's front end, which helps to build the structure of the web pages ([Macaulay, 2017](#)). It contains different tags that help to create content for the web pages. CSS is another front-end development technology that defines web application style and layout. It defines how an HTML page should look and separate HTML code from the design of a webpage. Web technology also includes JavaScript, which provides dynamic and interactivity functionality to the web application through its scripting language. JavaScript code is deployed and run on a browser that makes the web pages dynamic, and dynamic updates are possible without the need to load the whole page ([Richards et al., 2010](#)). It works with other front-end technologies like HTML and CSS to make the user experience seamless. Moreover, PHP programming language is also a web technology used to develop the backend of a web application, which helps handle backend functionality. PHP language is used to create dynamic web applications and helps to seamlessly manage server-side tasks ([Sotnik, Manakov and Lyashenko, 2023](#)). PHP language provides various web application functionality, including handling complex web logic, database connection and operations, implementing dynamic forms for user input, handling user authentication processes, handling data, sending and receiving emails, handling sessions and cookies, connection to various API endpoints, etc. MySQL technology is used to create a website database. The database stores customers, products and purchase data for long periods and provides a way to fetch them whenever necessary. It can handle large volumes of data without affecting performance. These web technologies work together to create a web application that provides users with engaging and seamless web experiences. Apart from these, there are other web technologies that help build modern and dynamic web applications.



### 3.7 The Problem of Over Information and Recommender System

According to [\(Zhang et al., 2019\)](#), the excessive growth of information on the internet negatively impacts systems like e-commerce as countless products are suggested to the users without filtering the products. A deep learning-based recommender system can solve the issue by implementing a personalized strategy for a better user experience. These strategies make customers' online experiences pleasant, boost business, and smooth decision-making [\(Ricci et al., 2010\)](#). The recommendation is made by analyzing customer behaviors online, customer past history, customer interactions, customer behaviors and other additional information. Based on the dataset used or the types of data inputted, there are three recommendation models: collaborative filtering, content-based filtering and hybrid recommender systems. The recommender system is used to recommend to customers the products or services they might like by estimating the user's preferences. The collaborative filtering model uses the historical interaction details of customers with the products, including feedback and ratings, to make recommendations. The content-based filtering model uses product auxiliary information to make recommendations. Moreover, a hybrid recommender system uses two or more recommendation strategies to make the recommendation [\(Burke, 2002\)](#).

### 3.8 State-of-the-Art Recommender System

According to [\(Jannach et al., 2010\)](#), a state-of-the-art recommender system provides users with high-quality, personalized, and affordable recommendations. Users may face it complex to survey about the fascinating and valuable products and services and may be unable to make firm decisions. The recommender system solves the complexity, which is one of the key enabling technologies being implemented in all e-commerce systems. It works by analyzing the purchase history of single customers or group of customers. The collaborative filtering model is applied to perform recommendation tasks, which implements the nearest neighbor method to a rating matrix. Further, the content-based filtering model recommends products to the user based on the attributes of the products the user has liked in the past. Content-based filtering does not have the capacity to show diverse or unexpected recommendations to customers, as is the case with a collaborating filtering model. It can only recommend products based on users' historical data. Apart from collaborating and content-based filtering models, many other complex statistical models are being proposed to improve recommendations, such as Bayesian networks, attribute-aware recommender models, knowledge-based approaches, and hybridization strategies.

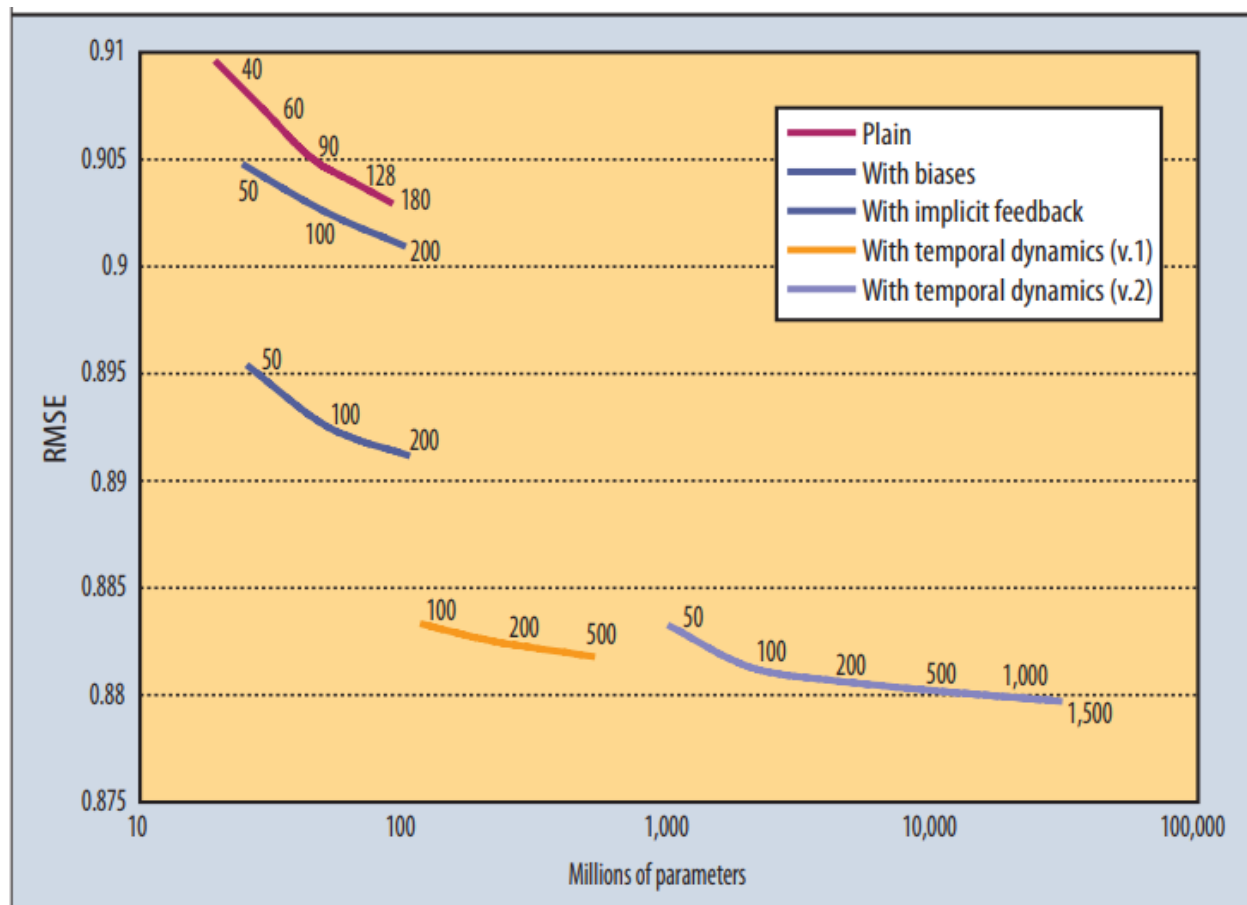
### 3.9 Machine Learning and Tree Boosting System

According to [\(Chen and Guestrin, 2016\)](#), machine learning techniques and implementation are becoming vital in many areas. Machine learning implements a data-driven approach and leans from the massive amount of data provided to it. It implements a gradient tree-boosting technique that provides state-of-the-art results and is useful in many applications. XGBoost is one of the tree-boosting systems that is open-source, scalable and beneficial for many machine-learning applications.

### 3.10 Matrix Factorization Techniques for Recommender System

According to [\(Koren, 2009\)](#), a recommender system analyzes user interest, its pattern, and the user's taste to deliver personalized recommendations. This adds another dimension to the customer experience. Therefore, many companies like Netflix and Amazon implement it on their websites.

Content filtering works by creating user profiles, which help associate customers with matching products. It also requires external data that is difficult to gather or may not be available to process. The alternative to this is collaborative filtering, which works by analyzing customers' past behavior. It can provide recommendations by analyzing customer relationships without needing the user profile. Matrix factorization describes customers and products by vectors of factors gathered from item rating patterns, offering flexibility and demonstrating numerous real-life situations. The method has been popular where predictive accuracy is combined with good scalability. Recommender systems depend on multiple input data. The data are placed in a matrix with one dimension demonstrating customers and other items of interest. In the collaborative filtering model, matrix factorization plays a dominant role. The accuracy of the matrix factorization model is shown in the diagram below:



*Figure 1 Accuracy of Matrix Factorization Model*

[\(Koren, 2009\)](#)

The diagram shows all four-factor models' RMSE (Root Mean Square Error). The lower the value in the plot, the better the accuracy. Accuracy is enhanced when the factor model's dimensionality is increased, and its description has distinct sets of parameters.

## 4. Dataset

The dataset for this project, E-commerce Purchase History from Electronics Store, includes user-item interactions, purchase records, product details, and user demographics. It is designed to provide insights into customer purchasing behavior, enabling the development of personalized recommendations.

The dataset contains data from 2020, which is stored in a kz.csv file. The dataset has eight different types of data, which are shown below:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2633521 entries, 0 to 2633520
Data columns (total 8 columns):
 #   Column          Dtype
---  -
 0   event_time      object
 1   order_id        int64
 2   product_id      int64
 3   category_id     float64
 4   category_code   object
 5   brand           object
 6   price           float64
 7   user_id         float64
dtypes: float64(3), int64(2), object(3)
memory usage: 160.7+ MB
```

*Figure 2 Dataset Description*

**Source:** Kaggle: <https://www.kaggle.com/datasets/mkechinov/ecommerce-purchase-history-from-electronics-store>

The sample of data from the data source:

	A	B	C	D	E	F	G	H
1	event_time	order_id	product_id	category_id	category_code	brand	price	user_id
2	2020-04-24 11:50:39 UTC	2.294E+18	1.52E+18	2.26811E+18	electronics.tablet	samsung	162.01	1.52E+18
3	2020-04-24 11:50:39 UTC	2.294E+18	1.52E+18	2.26811E+18	electronics.tablet	samsung	162.01	1.52E+18
4	2020-04-24 14:37:43 UTC	2.294E+18	2.27E+18	2.26811E+18	electronics.audio.he	huawei	77.52	1.52E+18
5	2020-04-24 14:37:43 UTC	2.294E+18	2.27E+18	2.26811E+18	electronics.audio.he	huawei	77.52	1.52E+18
6	2020-04-24 19:16:21 UTC	2.295E+18	2.27E+18	2.26811E+18		karcher	217.57	1.52E+18
7	2020-04-26 08:45:57 UTC	2.296E+18	1.52E+18	2.26811E+18	furniture.kitchen.ta	maestro	39.33	1.52E+18
8	2020-04-26 09:33:47 UTC	2.296E+18	1.52E+18	2.26811E+18	electronics.smartph	apple	1387.01	1.52E+18
9	2020-04-26 09:33:47 UTC	2.296E+18	1.52E+18	2.26811E+18	electronics.smartph	apple	1387.01	1.52E+18
10	2020-04-26 09:33:47 UTC	2.296E+18	1.52E+18	2.26811E+18	electronics.smartph	apple	1387.01	1.52E+18
11	2020-04-26 09:33:47 UTC	2.296E+18	1.52E+18	2.26811E+18	electronics.smartph	apple	1387.01	1.52E+18
12	2020-04-26 14:55:26 UTC	2.296E+18	2.27E+18	2.26811E+18	appliances.kitchen.r	lg	462.94	1.52E+18
13	2020-04-26 23:35:39 UTC	2.296E+18	1.52E+18	2.26811E+18	appliances.personal	polaris	30.07	1.52E+18
14	2020-04-27 07:24:51 UTC	2.296E+18	2.27E+18	2.3745E+18	electronics.video.tv	samsung	416.64	1.52E+18
15	2020-04-27 14:57:22 UTC	2.297E+18	1.52E+18	2.26811E+18	computers.compon	intel	91.41	1.52E+18
16	2020-04-27 14:57:22 UTC	2.297E+18	1.52E+18	2.26811E+18	computers.compon	intel	91.41	1.52E+18
17	2020-04-27 14:57:22 UTC	2.297E+18	1.52E+18	2.26811E+18	computers.compon	intel	91.41	1.52E+18
18	2020-04-28 02:21:45 UTC	2.297E+18	1.52E+18	2.26811E+18		philips	23.13	1.52E+18
19	2020-04-28 03:47:48 UTC	2.297E+18	1.52E+18	2.26811E+18	computers.noteboc	asus	509.24	1.52E+18
20	2020-04-28 04:25:00 UTC	2.297E+18	1.52E+18	2.26811E+18			6.94	1.52E+18

Figure 3 Sample of Dataset

**Original Purpose:** The data was collected to analyze customer behavior and product performance within an online electronics store.

#### Data Collection:

The dataset will be obtained directly from Kaggle, but Open CDP collected it from a large E-Commerce store in a Middle Eastern country, ensuring that it meets the project requirements. The license detail of the dataset is shown below:

## License

Data files © Original Authors

Figure 4 Dataset License

The author has provided dataset to be used for free as shown below Please [click here](#) for the link to REES46 Marketing Platform.

#### Using datasets in your works, books, education materials

You can use this dataset for free. Just mention the source of it: link to this page and link to [REES46 Marketing Platform](#).

Figure 5 Dataset Permission Set by Author

#### Metadata

- Format: CSV files
- Size: 298.65 MB and 1316,174 records

## 5. Ethical Issues

1. **GDPR Compliance:** The dataset does not contain personally identifiable information, adhering to GDPR.
2. **Compliance with University of Hertfordshire Ethical Standards:** This project adheres to the ethical principles established by the University of Hertfordshire, ensuring responsible conduct in research.
3. **Authorization for Dataset Utilization:** The dataset used in this project is publicly available and has received authorization for both educational and research use.
4. **Ethical Acquisition of Data:** The original organization collected the data ethically, meeting all relevant ethical requirements.

## 6. Methodology

### 6.1 Introduction

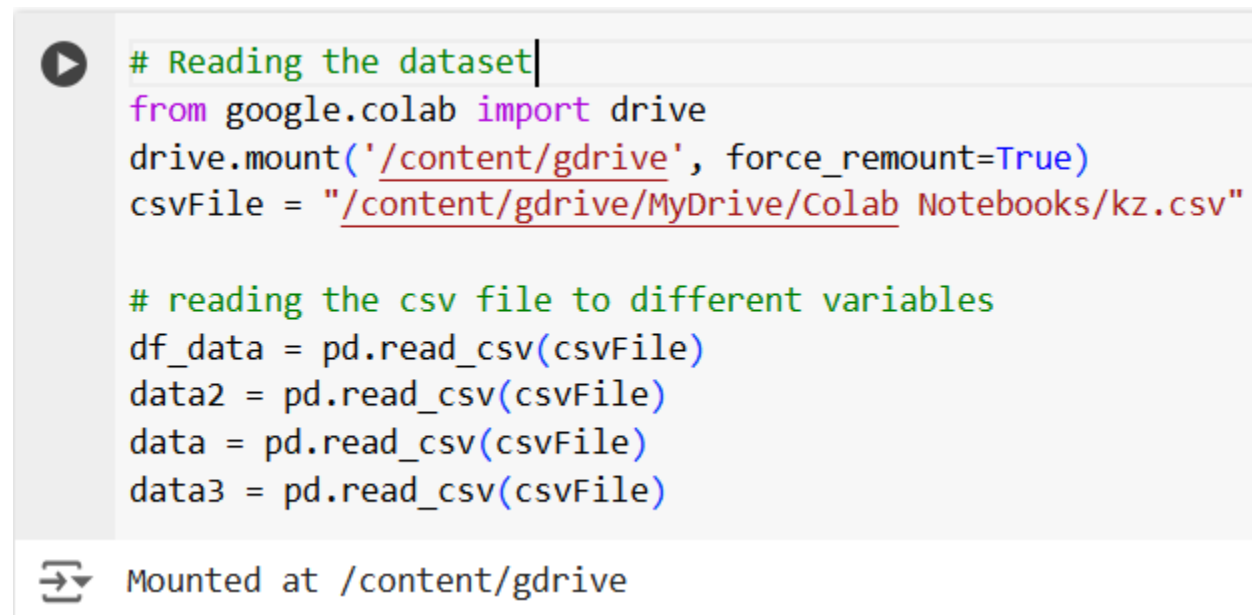
The methodology includes the technical work performed in the project to build the personalized recommendation system. Three machine learning algorithms—collaborative filtering, content-based filtering, and hybrid model — are applied to build the system. This section describes all the algorithms and their implementation methods in detail. It also explains the process and code for applying the dataset to perform machine learning tasks. The performance comparison of the three algorithms and their defined metrics is also done and explained in this section.

The code for the project is written in Jupyter Notebook using Google Colab, and the file name is RecommendationSystem.ipynb. Python programming language is used to write machine learning code using various libraries such as Numpy, Pandas, Seaborn, Matplotlib, etc.

### 6.2 Data Collection and Pre-Processing

As mentioned earlier, the data is collected from the Kaggle website and used in this project to perform machine learning tasks. However, for future implementation, a prototype of an e-commerce website has been built to provide the necessary dataset with real-time user and purchase data. The dataset is uploaded to Google Drive to be implemented by Google Colab. The dataset can have errors, low-quality data or duplicate data. Therefore, data pre-processing is required to improve data quality, remove outliers, remove formatting inconsistencies, and handle missing values ([Kang & Tian, 2018](#)).

The dataset is then read, and different variables are used to store those data as shown below:



```
# Reading the dataset|
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
csvFile = "/content/gdrive/MyDrive/Colab Notebooks/kz.csv"

# reading the csv file to different variables
df_data = pd.read_csv(csvFile)
data2 = pd.read_csv(csvFile)
data = pd.read_csv(csvFile)
data3 = pd.read_csv(csvFile)
```

Mounted at /content/gdrive

*Figure 6 Reading the Dataset*

The description of the data in dataset provides the following result:

```
[ ] data.info()
```

```
↔ <class 'pandas.core.frame.DataFrame'>  
RangeIndex: 2633521 entries, 0 to 2633520  
Data columns (total 8 columns):  
#   Column          Dtype  
---  ---  
0   event_time      object  
1   order_id        int64  
2   product_id      int64  
3   category_id     float64  
4   category_code   object  
5   brand           object  
6   price           float64  
7   user_id         float64  
dtypes: float64(3), int64(2), object(3)  
memory usage: 160.7+ MB
```

*Figure 7 Description of the Dataset*

The data pre-processing is performed where the duplicate data are removed along with the null or empty row using the following code:

```
[ ] # Data preprocessing
    data = data.drop_duplicates()
```

```
[ ] data.isnull().sum()
```



	0
event_time	0
order_id	0
product_id	0
category_id	431953
category_code	612053
brand	505965
price	431953
user_id	2069351


**dtype:** int64

*Figure 8 Data Pre-Processing*

The dataset is for the year 2020, but there are false data from 1970, too. Therefore, those data from 1970 are removed from the dataset using the following code:



```
# exclude rows where 'event_time' contains '1970'
data = data[~data['event_time'].str.contains('1970')]
data.describe(include='all').T
```




	count	unique	top
<b>event_time</b>	2613215	1316173	2020-04-09 16:30:01 UTC
<b>order_id</b>	2613215.00	NaN	NaN
<b>product_id</b>	2613215.00	NaN	NaN
<b>category_id</b>	2185340.00	NaN	NaN
<b>category_code</b>	2006141	509	electronics.smartphone
<b>brand</b>	2111921	22955	samsung
<b>price</b>	2185340.00	NaN	NaN
<b>user_id</b>	562188.00	NaN	NaN

*Figure 9 Removing Data from 1970*

There are other tasks performed for the data cleaning process of the dataset, such as removing the rows with empty brand and user\_id columns using the following code:

```
[ ] # remove empty rows in brand and user_id columns
data = data.dropna(subset=['brand', 'user_id']).reset_index(drop=True)
data.isnull().sum()
```



	0
event_time	0
order_id	0
product_id	0
category_id	0
category_code	115675
brand	0
price	0
user_id	0

**dtype:** int64

*Figure 10 Removing Rows with Empty brand and user\_id Columns*

### 6.3 Model Training

The collaborative filtering algorithm is trained using the SVD (Singular Value Decomposition) model. SVD is a factorization algorithm for collaborative filtering that decomposes a matrix into three simpler matrices ([Zhang et al., 2005](#)). SVD is one of the important tools in linear algebra and has great implementations in systems to generate personalized recommendations, which disclose vital information about the original data. It offers greater ease of implementation and does not impose restrictions on feature values.

The formula for SVD is provided below: ([Pryor, 1998](#))

$$R = U \cdot \Sigma \cdot V^T$$

Where,

$R$  = user-item interaction matrix

$U$  = user feature matrix

$\Sigma$  = diagonal matrix of singular values

$V^T$  = item-feature matrix

Further, the performance of collaborative filtering can be measured using RMSE (Root Mean Square Error) and its formula is shown below:

$$RMSE(P, A) = \sqrt{\frac{\sum_{i=1}^n \sum_{j=1}^m J_{ij} (A_{ij} - P_{ij})^2}{\sum_{i=1}^n \sum_{j=1}^m J_{ij}}}$$

*Figure 11 RMSE Formula*

[\(Ma, 2008\)](#)

Where

P = Prediction

A = answer

Similarly, content-based filtering is trained using TF-IDF vectorization. TF-IDF vectorization transforms text data into numerical representations that analyze the relative significance of data within the given text file [\(Zhang, 2024\)](#).

The TF-IDF is calculated using the formula: [\(Herimanto, Samosir, and Ginting, 2024\)](#)

$$TF - IDF = TF \times IDF$$

$$TF = \frac{freq(i,j)}{max}$$

$$IDF = \log \frac{N}{n(i)}$$

Where,

TF= Term Frequency

$freq(i,j)$ = frequency of occurrence of i in text-file j

max= number of words in text-file

IDF= Inverse Document Frequency

N= total number of text-file

$n(i)$ = number of text-file containing the term

## 6.4 Tools and Techniques

- Python programming language for writing machine language code
- Jupyter notebook to write and edit machine learning code
- Data collection and pre-processing
- Machine learning techniques like collaborating filtering, content-based filtering, and hybrid recommendation models.
- Machine learning libraries like Scikit-learn, TensorFlow, Pandas, NumPy, and SciPy.
- Web development tools and techniques
  - Frontend development: HTML, CSS, Bootstrap, and JavaScript
  - PHP programming language
  - MySQL database
  - Local hosting using the XAMPP Control Panel tool
  - Visual Studio Code IDE
  - Testing web browsers: Google Chrome and Mozilla Firefox
- GitHub for version control
- Microsoft Word for documentation
- Draw.io for drawing.

## 6.5 Machine Learning Algorithm Implementation

### 6.5.1 Collaborating Filtering Algorithm

The collaborating filtering algorithm explains that if two users have similar preferences and tastes, then based on the preferences of the first user, the second user's preferences can be predicted ([Herlocker et al., 2000](#)). This type of filtering is beneficial for systems like recommendation systems for e-commerce, where the behaviors of particular kinds of users can be applied to other similar users. Two types of collaborating filtering are based on similar users or items ([Mustafa et al., 2017](#)). It works by gathering similar types of target user data and their interaction with the target items. Then, the similarities between those users and the items are calculated. Those calculations can then be used to generate recommendations in future. The algorithm has a few limitations: it is hard to provide recommendations if there is less data and fewer user interaction details with the items. Also, it will be hard to recommend items to new users with no interaction history.

The collaborating filtering algorithm is implemented using the Surprise library. For this, the Singular Value Decomposition (SVD) algorithm is applied. First, the necessary libraries are imported. Then, the data is converted into 80% training data and the remaining 20% testing data. The random\_state is set to 42 to ensure the reproducibility of the split, i.e., the same split for every run. The code of library import and data split is shown below:

```
[91] from sklearn.model_selection import train_test_split
      from sklearn.metrics import mean_squared_error
      from surprise import SVD, Dataset, Reader, KNNBasic
      from sklearn.feature_extraction.text import TfidfVectorizer
      from sklearn.metrics.pairwise import linear_kernel
      # Create train-test split for collaborative filtering
      train_data, test_data = train_test_split(df, test_size=0.2, random_state=42)
```

*Figure 12 Library Import and Data Split*

A reader object is created using a price range as a rating scale. The rating range is provided from the minimum value of the price attribute to the maximum value. Using the reader object, the data is loaded into the Surprise library. The Surprise dataset object is created using only three columns from the dataset. After that, a training set is created using the surprise dataset to train the collaborative filtering algorithm. The code for this is provided below:

```
▶ # Collaborative Filtering
  # Convert to Surprise format
  reader = Reader(rating_scale=(df['price'].min(), df['price'].max()))
  data = Dataset.load_from_df(df[['user_id', 'product_id', 'price']], reader)
  trainset = data.build_full_trainset()
```

*Figure 13 Code to Convert to Surprise Object and More*

The SVD algorithm is initiated to perform collaborative filtering. The training dataset is then used to train the model through the use of the SVD algorithm, as shown below:

```
[93] # Use SVD for collaborative filtering
      collab_model = SVD()
      collab_model.fit(trainset)

⇒ <surprise.prediction_algorithms.matrix_factorization.SVD at 0x7ed1e99a3370>
```

*Figure 14 SVD Algorithm Implementation*

A collaborating filtering algorithm predicts the rating for a particular user-product pair. The rating is displayed as 122.66 for the provided user-product pair. The rating of 122.66 estimates that the user with ID 1515915625441990000 can pay 122.66 for the product with ID 1515966223509089906. The rating of other user-product pairs can also be done by providing their values to uid and iid attributes in the code below:

```
▶ # Test on a sample user-product pair
  pred = collab_model.predict(uid=1515915625441990000, iid=1515966223509089906)
  print(f"Collaborative Filtering Prediction: {pred.est}")

⇒ Collaborative Filtering Prediction: 122.66922286190233
```

*Figure 15 Collaborating Filtering Prediction*

The test set is used to test the model, and RMSE is also calculated:

```
testset = [tuple(x) for x in test_data[['user_id', 'product_id', 'price']].to_numpy()]

# Generate predictions for the testset
predictions = collab_model.test(testset)
true_ratings = [pred.r_ui for pred in predictions] # True ratings
predicted_ratings = [pred.est for pred in predictions] # Predicted ratings
rmse = np.sqrt(mean_squared_error(true_ratings, predicted_ratings))
print(f"RMSE: {rmse}")
```

*Figure 16 RMSE Calculation*

### 6.5.2 Content-Based Filtering Algorithm

The content-based filtering algorithm explains that if a user likes an item in the past, they will probably like it again or similar items with equivalent features (B.Thorat et al., 2015). It is based on the analysis of the items themselves rather than the behavior of users with the item, which is considered during collaborating filtering. It makes recommendations based on the user's past interactions with similar items where the attributes of the items are focused. This type of filtering is beneficial for systems like recommendation systems for e-commerce, where the user's behaviors with a particular item can be applied to recommend similar kinds of items in future. It works by gathering the attributes of the items the user has purchased in the past and creating the item

profiling by extracting relevant features. Also, the user profile is created based on the user's past behavior. These profiles are then used to generate recommendations for the future. The algorithm has a few limitations: it is hard to provide recommendations if the item has no or fewer attributes to describe them, only recommend items similar to previous items the user has interacted with, creating a bubble of recommendations of items that the user is already familiar with, leaving the chance to show them with new and interesting products.

The content-based filtering algorithm uses Nearest Neighbors and Term Frequency-Inverse Document Frequency (TF-IDF) vectorization to produce the desired results of similar products. At first, a new column content is created by combining category\_code and brand columns. The newly created content column is a single feature column for content-based analysis. The error is also handled using fillna() method that replaces the missing values of category\_code and brand columns with an empty string. The code for this is provided below:

```
[ ] # Content-Based Filtering
    # Combine category_code and brand for content-based features
    df['content'] = df['category_code'].fillna('') + " " + df['brand'].fillna('')
```

*Figure 17 Creating content column*

The TF-IDF Vectorization object is initiated, which captures the importance of words from the product description. The code transforms the recently created content column into a TF-IDF matrix. A matrix named tfidf\_matrix is created, where products are represented by rows and terms are represented by columns. The code for this is provided below:

```
[ ] # Use TF-IDF to vectorize content
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(df['content'])
```

*Figure 18 Vectorization*

A method is created which takes product\_id and top\_n as input and provides a list of similar product IDs of the top\_n recommendation. The method is called with a product ID, and it provides five other products similar to that product. The code of content\_based\_recommendations\_on\_the\_fly() method is provided below:

```
[ ] # Function to compute similarity on-the-fly
def content_based_recommendations_on_the_fly(product_id, top_n=5):
    # idx = df[df['product_id'] == product_id].index[0]
    if product_id not in df['product_id'].values:
        return []
    indices = df.index[df['product_id'] == product_id].tolist()
    if not indices:
        return []
    idx = indices[0]
    product_vector = tfidf_matrix[idx]
    sim_scores = linear_kernel(product_vector, tfidf_matrix).flatten()
    sim_scores = [(i, score) for i, score in enumerate(sim_scores)]
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:top_n + 1]
    product_indices = [i[0] for i in sim_scores]
    return df.iloc[product_indices]['product_id'].tolist()

# Test content-based filtering
print("Content-Based Recommendations:", content_based_recommendations_on_the_fly(product_id=1515966223509089906))
```

Content-Based Recommendations: [1515966223509106817, 1515966223509106817, 1515966223509089673, 1515966223509089673, 2273948222336532851]

*Figure 19 Content-Based Recommendation Method Code*

The Nearest Neighbor model is trained using a brute force approach on cosine metric to find the most similar products. A method is created which takes product\_id and top\_n as input and provides a list of similar product IDs of the top\_n recommendation using Nearest Neighbor model. The method is called with a product ID, and it provides five other products similar to that product. The code of for this is provided below:

```
from sklearn.neighbors import NearestNeighbors

# Train a nearest neighbor model
nn = NearestNeighbors(metric='cosine', algorithm='brute')
nn.fit(tfidf_matrix)

# Function to get recommendations
def ann_recommendations(product_id, top_n=5):
    idx = df[df['product_id'] == product_id].index[0]
    distances, indices = nn.kneighbors(tfidf_matrix[idx], n_neighbors=top_n + 1)
    return df.iloc[indices.flatten()[1:]]['product_id'].tolist()

# Test ANN recommendations
print("ANN Content-Based Recommendations:", ann_recommendations(product_id=1515966223509089906))
```

ANN Content-Based Recommendations: [1515966223509130229, 1515966223509089906, 2273948222336532851, 1515966223509106817, 1515966223509089673]

*Figure 20 Nearest Neighbors Model*

Further, the RMSE is calculated:



```

# Apply predictions for each user-product pair in the dataset
df['predicted_rating'] = df.apply(
    lambda x: predict_rating(x['user_id'], x['product_id'], tfidf_matrix, df), axis=1
)

# Drop rows with NaN predictions
df_filtered = df.dropna(subset=['predicted_rating'])

# Calculate RMSE
true_ratings = df_filtered['price'].to_numpy()
predicted_ratings = df_filtered['predicted_rating'].to_numpy()
rmse = np.sqrt(mean_squared_error(true_ratings, predicted_ratings))

print(f"RMSE for Content-Based Filtering: {rmse}")

```

*Figure 21 RMSE Calculation*

### 6.5.3 Hybrid Approach

The hybrid approach combines the advantages of collaborating filtering and content-based filtering algorithms. Both algorithms have combined strengths, and the limitations both have are limited when using the hybrid approach. The limitation of collaborating filtering is solved using the hybrid approach, where it can easily show recommendations to new users using the attributes of the items. Further, the limitation that content-based filtering has is solved using the hybrid approach, where users are shown diverse products through collective preferences. However, the hybrid approach has a few limitations, too. It combines two different algorithms, which is why its complexity level is increased, and it also requires a high amount of computational resources to train the data and provide recommendations.

The collaborative filtering and the content-based filtering algorithms implemented above are applied to create a hybrid recommendation system. A method is created that uses the hybrid model to take user\_id, product\_id, and top\_n as input and provides a list of similar product IDs and ratings for the top\_n recommendation. The collab\_model, previously trained for collaborative filtering, predicts the user and product ratings. The method implemented for content-based filtering is then used to get a list of recommended product IDs. Then, a combined recommendation is created and stored in a tuple and returned after being sorted in descending order. The code for this is provided below:

```

# Hybrid Recommendation
# Weighted average of collaborative and content-based recommendations
def hybrid_recommendations(user_id, product_id, top_n=5):
    # Collaborative prediction
    collab_pred = collab_model.predict(uid=user_id, iid=product_id).est

    # Content-based recommendations
    content_recs = content_based_recommendations_on_the_fly(product_id, top_n=top_n)

    # Combine recommendations
    recommendations = [(prod_id, collab_pred) for prod_id in content_recs]
    return sorted(recommendations, key=lambda x: x[1], reverse=True)

# Test hybrid recommendation
print("Hybrid Recommendations:", hybrid_recommendations(user_id=1515915625441990000, product_id=1515966223509089906))

```

Hybrid Recommendations: [(1515966223509106817, 122.64130081685236), (1515966223509106817, 122.64130081685236), (1515966223509089673,

*Figure 22 Hybrid Recommendation*

## 6.6 Prototype of E-Commerce

The prototype of the e-commerce application is developed using web technology. The methodology used to develop the e-commerce web application is the agile method. The agile method is an iterative methodology that helps to develop the required software solution incrementally (Włodarczak, 2023). Currently, only the prototype of the e-commerce web application is needed, which is developed, and the full version will be developed in future based on the user requirements. The agile method makes it easy to develop the initial version of the software and later develop the full version incrementally by collecting user needs through user feedback and collaboration.

The web technologies implemented are HTML and CSS for frontend, JavaScript for dynamic content, PHP for backend programming and MySQL for database implementation. The web application developed is tested in local service that is provided by XAMPP Control Panel, which user interface looks like the following:

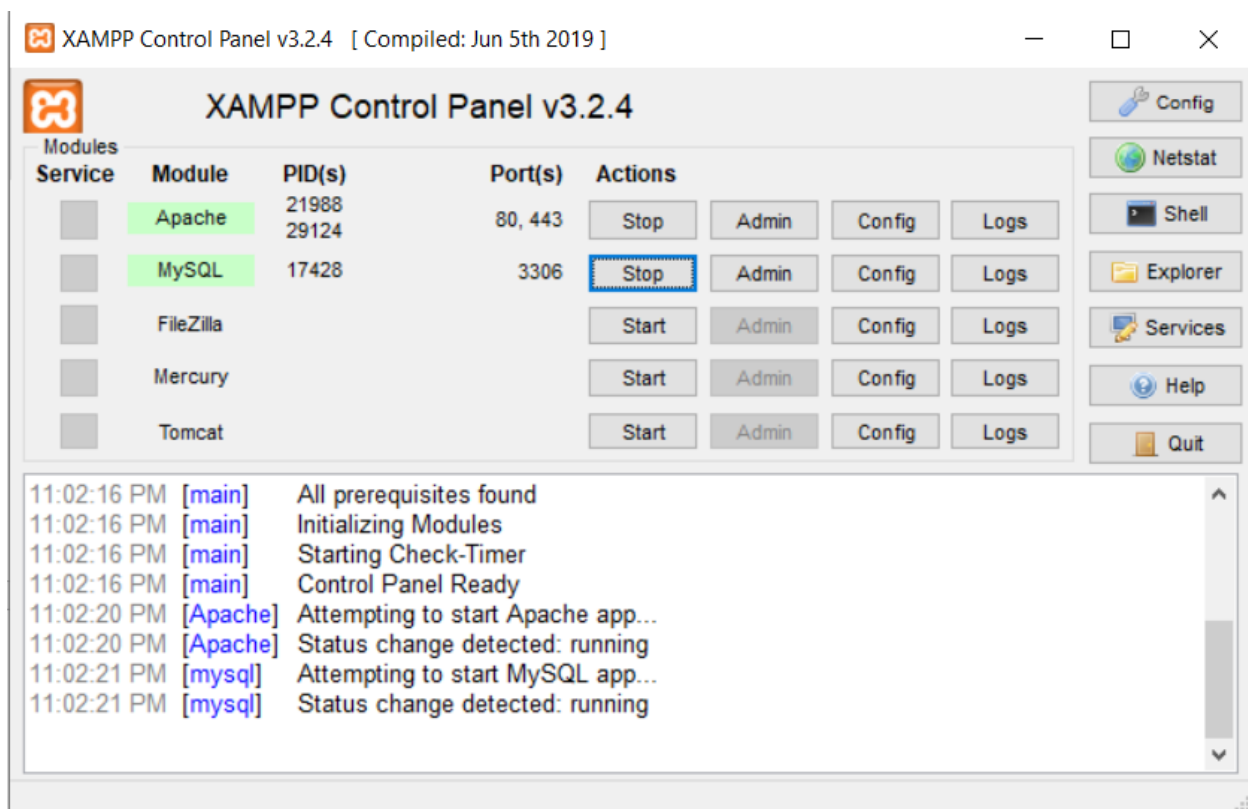


Figure 23 XAMPP Control Panel

The user interface of the e-commerce website looks like the following which shows the customer login page:



*Figure 24 User Interface of E-Commerce Website*

The HTML code to create the header of the website is shown below that contains multiple HTML tags:

```
<html>
<head>
  <title>E-Commerce Platform</title>
  <link href="style.css" rel="stylesheet" type="text/css" />
</head>

<body>
  <div class="wrapper">
    <div class="header">
      <div class="header-logo">
        <a href="index.php"></a>
      </div>
      <div class="header-image">
        
      </div>
    </div>
  </div>
```

*Figure 25 HTML Code of Website Header*

The CSS code to style the HTML tags of the website including the header is:

```

1  body
2  {
3      background-color: #a5edf0;
4  }
5
6  .wrapper
7  {
8      width:75%;
9      margin:auto;
10     background-color: #1cbcc3;
11     color: white;
12 }
13 .header
14 {
15     padding:10px;
16     width:75%;
17     height:120px;
18 }
19 }
20 .header-logo
21 {
22     width:50%;
23     float:left;
24 }
25 .header-image
26 {
27     width:50%;
28     float:right;
29 }
30 }
31 .header-image img{
32     width: 60%;
33     height: 100%;
34 }
35 /*end of header */

```

*Figure 26 CSS Code to Style Header*

JavaScript code is written to create dynamic content including user form to take input. The JavaScript code to validate the form filled by the admin of the web application to add items to the system is:

```
<script>
function confirm(){
    var item_name = document.getElementById('item_name').value;
    var brand = document.getElementById('brand').value;
    var item_price = document.getElementById('item_price').value;
    var item_details = document.getElementById('item_details').value;
    var item_image = document.getElementById('item_image').value;

    if(item_name !== null && item_name !== '' && brand !== null && brand !== '' && item_price !== null
        alert("Item added Successfully.");
    }
}
</script>
```

*Figure 27 JavaScript Code to Validate User Form*

The sample of PHP code that communicates with the database and deletes an item from the item table when item\_id is provided is:

```
1  <?php
2      $item_id = $_GET['item_id'];
3      $conn = mysqli_connect("localhost", "root", "", "ecommerce_db");
4      mysqli_query($conn, "DELETE FROM `item` WHERE item_id = ".$item_id);
5
6
7      header('Location: ../admin-add-item.php');
8  ?>
```

*Figure 28 PHP Code to Delete an Item*

The user interface of the website where customers can view multiple products and make a purchase is:

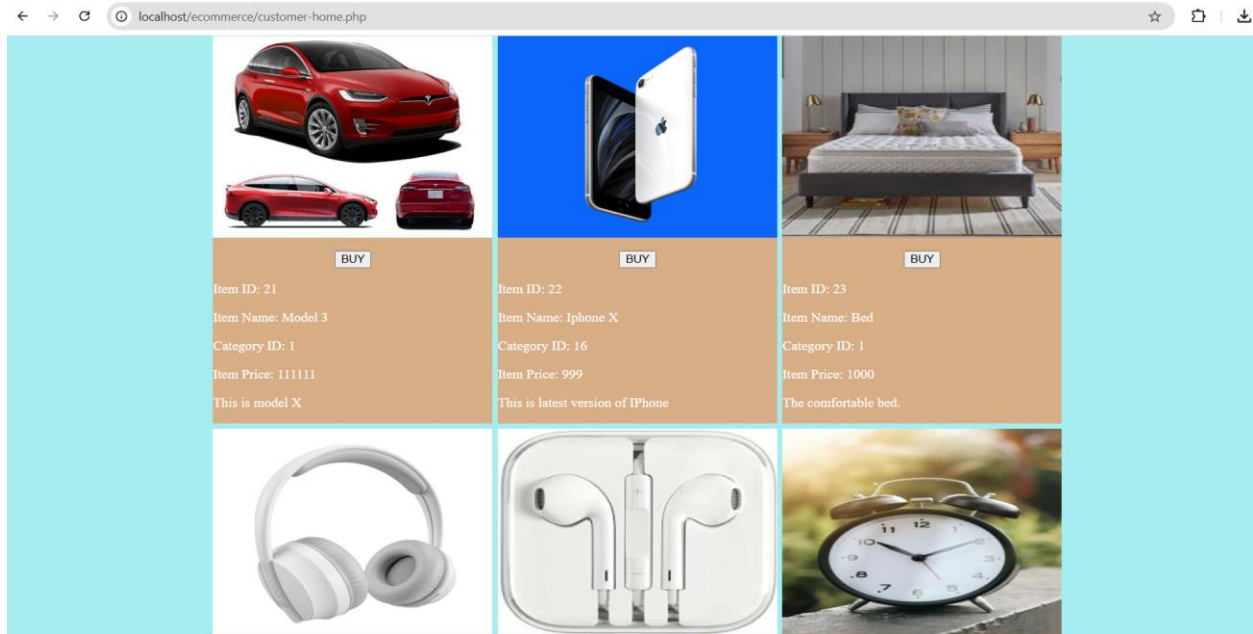


Figure 29 User Interface of Website to Make a Purchase

The entity-relationship (ER) diagram of the database model applied to build the e-commerce platform is:

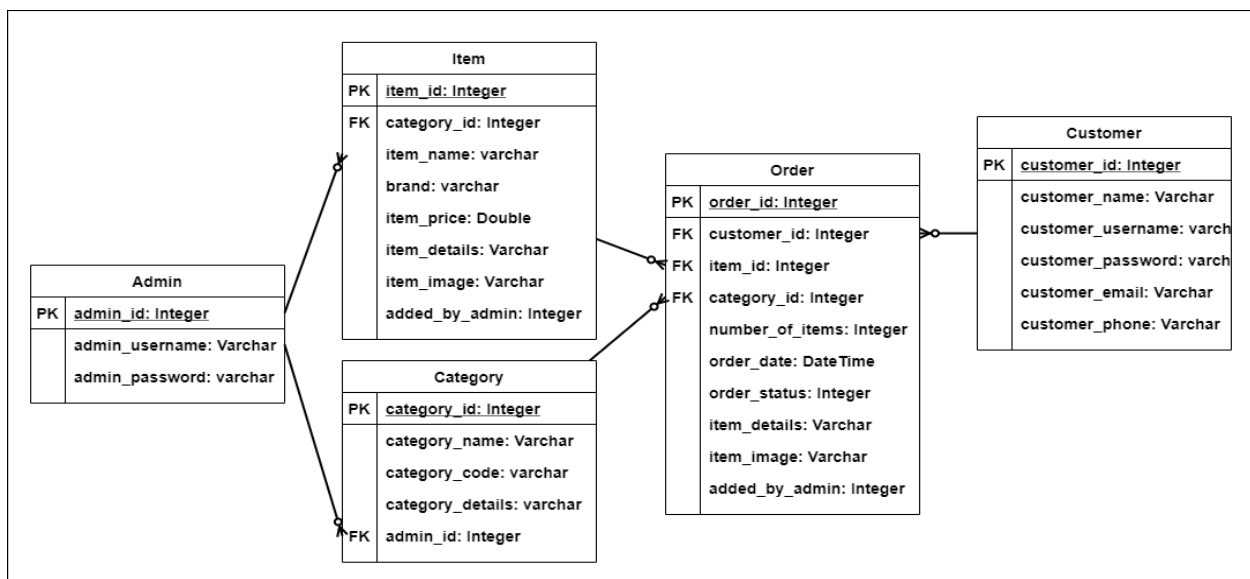
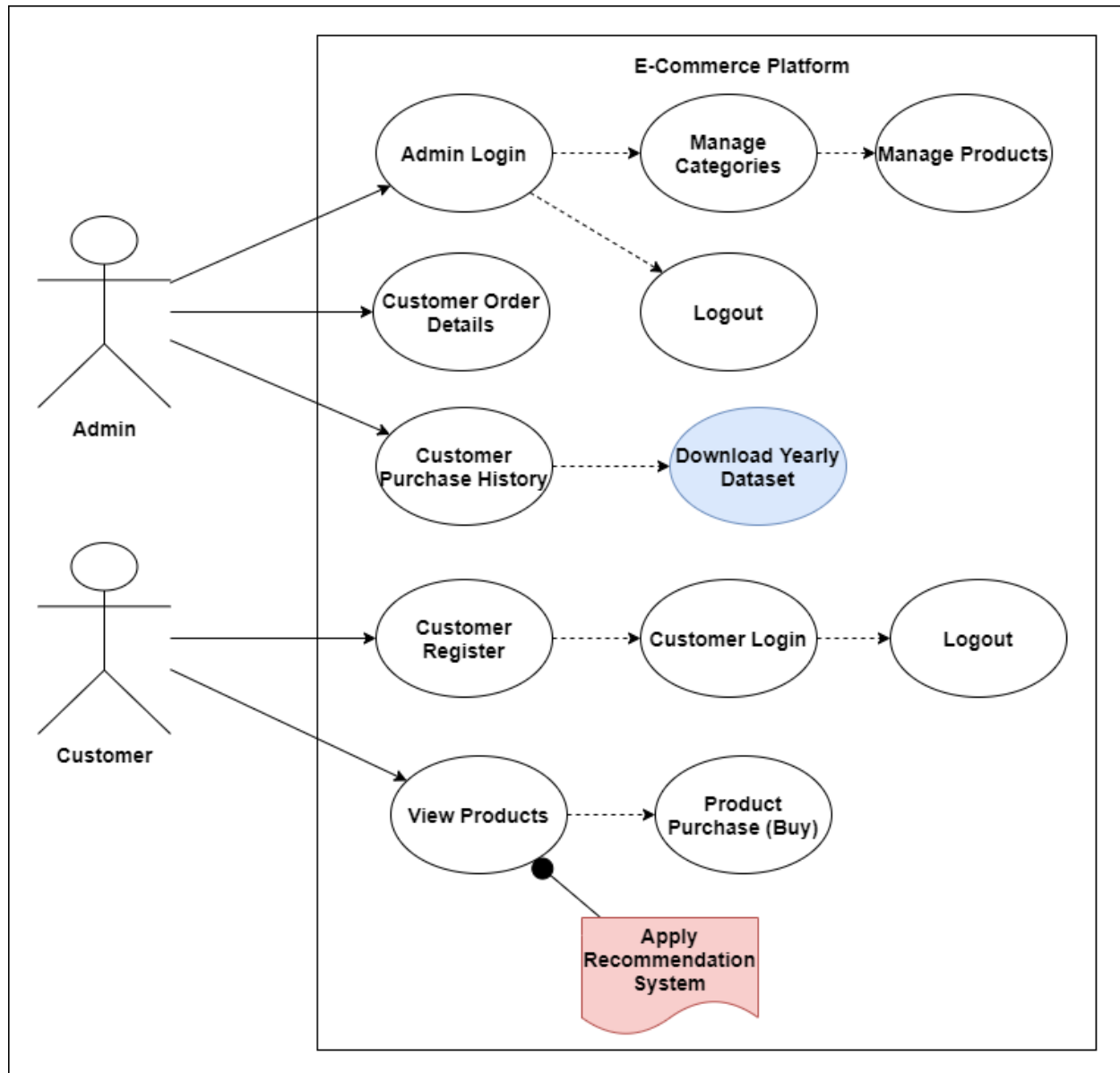


Figure 30 Entity-Relationship Diagram of Database

The entity-relationship diagram shown above describes the table and their relationship in a database model (Li & Chen, 2009). The entity-relationship diagram shown above has five different tables. Each table has a certain number of attributes along with the datatype associated with each of them. The Admin table has three different attributes where admin\_id, which is an integer type, is the Primary Key (PK) attribute, which means it only takes a unique id value. The other two attributes are of varchar datatype and can store alpha-numeric data. Similarly, Item and Category

tables exist with their own attributes and datatype and have a one-to-many relationship with the Admin table indicated by the added\_by\_admin attribute in the Item table as Foreign Key (FK) and the admin\_id attribute in the Category table as FK. The Customer table has six different attributes and is used to store the customer's data. The Order table is used to store the products the customers ordered and retrieve the customer's purchase history in the system. The Order table has multiple attributes and has a one-to-many relationship with the Customer, Item and Category table, all indicated by foreign key.

The use case of an e-commerce platform is:



*Figure 31 Use Case Diagram of E-Commerce Platform*

The use case diagram of the E-Commerce platform is shown above, which provides a detailed explanation of system features and its interaction with the different types of users (Fauzan et al.,



[2019](#)). There are two different types of users in the E-Commerce system, as indicated by the stick figure in the above diagram. The two types of users are the admin and the customer, who interact with the system to perform specific tasks. Admin performs login activity using pre-defined login credentials. The admin is responsible for managing the system's categories and products. Admin can also manage order details placed by the customers. The admin can access the customer purchase history and download the yearly dataset. The customer purchase history data to be downloaded as a dataset is the key feature of the e-commerce system. This feature will assist the recommendation system in future by providing real-time user data to analyze and make decisions. Further, customers can also perform multiple tasks in the system. Customers must register and then login themselves to be able to access the system. The customer can view the available products and make a purchase. The view products feature will also show the available products to the user. In this section, the personalized recommendation system created using machine learning code in Python can be implemented. This will help to show only relevant products to the users which they will be interested in and are likely to buy.

## 7. Results

The performance of the three machine learning algorithms implemented in this project is evaluated using three different metrics: accuracy, diversity, and engagement. These metrics ensure a balanced evaluation of the algorithms' performance and also assess various dimensions of the recommendation system's performance.

### 7.1 Accuracy

The accuracy metric defines how well the recommendation system predicts the user's preferences and historical behaviour. It describes how well the products are predicted that the users will likely find relevant or interact with.

To evaluate the accuracy of the collaborating filtering algorithm, the RMSE (Root Mean Squared Error) is calculated between the predicted prices and the actual prices of the products. To evaluate the accuracy of the content-based filtering algorithm, the test data is passed to the `content-based_recommendation_on_the_fly()` method. The recommended product IDs by the method are compared with the actual product ID that appears in the top recommendation. Further, to evaluate the accuracy of the hybrid model, the user-product pair and top recommendations are noted. The final outcome is calculated as the average proportion of the correct predictions.

The code to calculate the accuracy of all three machine-learning algorithms is provided below:

```
# Performance Metrics
def evaluate_models(test_data, collab_model, top_n=5):
    metrics = {"accuracy": {}, "diversity": {}, "engagement": {}}

    #Accuracy
    # Collaborative Filtering Evaluation
    collab_preds = []
    for _, row in test_data.iterrows():
        pred = collab_model.predict(uid=row['user_id'], iid=row['product_id']).est
        collab_preds.append(pred)
    metrics["accuracy"]["collaborative"] = np.sqrt(mean_squared_error(test_data['price'], collab_preds))

    # Content-Based Evaluation
    content_preds = []
    for _, row in test_data.iterrows():
        recs = content_based_recommendations_on_the_fly(row['product_id'], top_n=top_n)
        content_preds.append(row['product_id'] in recs)
    metrics["accuracy"]["content-based"] = np.mean(content_preds)

    # Hybrid Evaluation
    hybrid_preds = []
    for _, row in test_data.iterrows():
        recs = hybrid_recommendations(row['user_id'], row['product_id'], top_n=top_n)
        hybrid_preds.append(any(r[0] == row['product_id'] for r in recs))
    metrics["accuracy"]["hybrid"] = np.mean(hybrid_preds)
```

*Figure 32 Accuracy Evaluation*

## 7.2 Diversity

The diversity metric measures the variation of the products recommended to users. The variation comes from different brands, categories, or other distinct features. A diverse system recommends new and interesting products to users rather than only suggesting similar-type repeated products.

A random sample of 100 user IDs is taken to evaluate the diversity of all three algorithms. The unique recommendations provided to the users are considered. The distinctive products shown to the users are counted to assess the accuracy of the collaborating filtering algorithm. The same is done for content-based filtering, where the distinctive products shown to the users are counted. Currently, the code to calculate the diversity of the hybrid model is disabled as it is complex to calculate it with the current setup and the dataset. The code is throwing errors and hampering the progress of the other lines of code. The accuracy of the hybrid model is currently set to 0 and will be calculated in a later version of the code.

The code to calculate the diversity of all three machine-learning algorithms is provided below:

```
# # Diversity (unique recommendations for a sample of users)
sample_users = test_data['user_id'].sample(n=100, random_state=42).unique()
diversity_scores = {
    "collaborative": len(set(collab_model.predict(uid=u, iid=i).iid for u in sample_users for i in range(top_n))),
    "content-based": len(set(i for user in sample_users for i in content_based_recommendations_on_the_fly(user, top_n=top_n))),
    "hybrid": 0,
    # "hybrid": len(set(i for user in sample_users for i in hybrid_recommendations(user, i, top_n=top_n))),
}
metrics["diversity"] = diversity_scores
```

*Figure 33 Diversity Evaluation*

## 7.3 Engagement

The engagement metric defines how well the recommendation system drives user interaction or how well the system interacts with the users. It measures how well the system captures user interest, makes them stay in the system longer and makes more purchases than usual.

The engagement metric calculates the average number of recommendations provided to each user using all three machine-learning algorithms.

The code to calculate the engagement of all three machine-learning algorithms is provided below:

```
# Engagement (average number of recommendations per user)
engagement_scores = {
    "collaborative": np.mean([len(collab_model.predict(uid=u, iid=i).iid for i in range(top_n)) for u in sample_users]),
    "content-based": np.mean([len(content_based_recommendations_on_the_fly(i, top_n=top_n)) for i in sample_users]),
    "hybrid": np.mean([len(hybrid_recommendations(u, i, top_n=top_n)) for u in sample_users for i in range(top_n)]),
}
metrics["engagement"] = engagement_scores

return metrics
```

*Figure 34 Engagement Evaluation*

## 7.4 Results of Performance Metrics

The results of performance metrics after running the above mentioned code is as follows:

*Table 1 Result of Performance Metrics*

	<b>Collaborative Filtering</b>	<b>Content-Based Filtering</b>	<b>Hybrid Approach</b>
<b>Accuracy</b>	283.5297920274702	0.5508021390374331	0.5508021390374331
<b>Diversity</b>	5	0	0
<b>Engagement</b>	5.0	0.0	0.0

- The accuracy value for the collaborating filtering algorithm is 283.5297920274702, which means the RMSE between the predicted prices and the actual prices of the products is approximately 283. This number is high, which means that the prediction is sometimes inaccurate and has room for improvement.
- The accuracy value for the content-based filtering algorithm is 0.5508021390374331, which means the portion of the correct recommendations is approximately 55%. This means that for 55% of the cases, the algorithm recommends the proper products to the user, which is quite good.
- The accuracy value for the hybrid approach is 0.5508021390374331, which is the same as the content-based filtering. This means that content-based filtering algorithms dominate the hybrid approach, and collaborative filtering is not helping improve accuracy.
- The diversity of the collaborative filtering algorithm is 5, which means the algorithm shows the user five unique products every time. This number is perfect; the code is set to show five unique products every time, and it works properly.
- The diversity of the content-based filtering is 0, which means the algorithm shows the same products to the user every time without any unique recommendations.
- The diversity of the hybrid model is also 0, which means the algorithm does not show unique recommendations to the user due to being heavily reliant on a content-based filtering algorithm.
- The collaborative filtering algorithm's engagement is 5, which means the algorithm shows the user five unique products every time. This number is perfect; the code is set to show five unique products every time, and it works correctly.
- The engagement of the content-based filtering is 0, which means the algorithm shows the same products to the user every time without any unique recommendations. This can be due to the dataset's lack of enough content data.
- The engagement of the hybrid model is also 0, which means the algorithm does not show unique recommendations to the user due to being heavily reliant on a content-based filtering algorithm.

## 7.5 RMSE Results

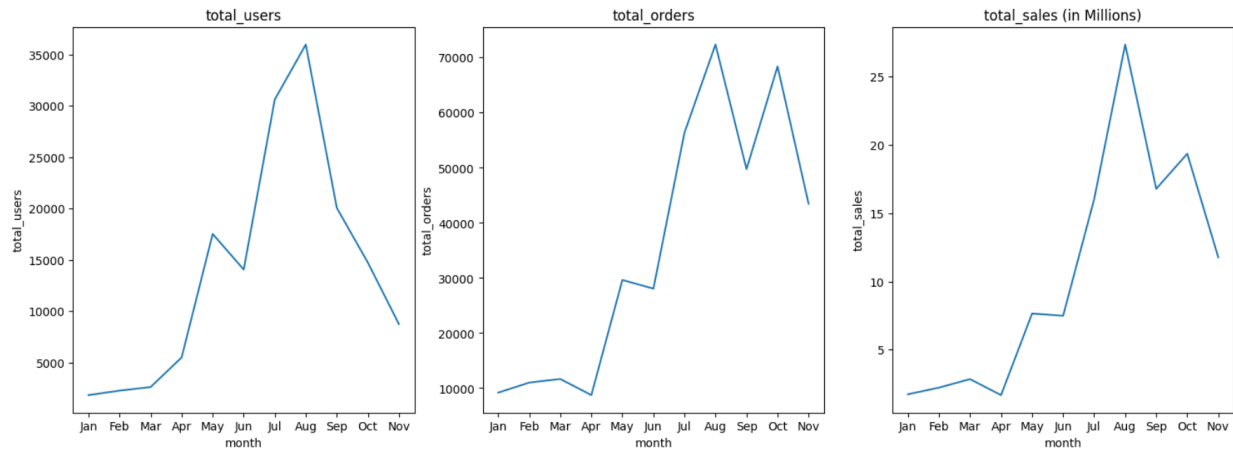
The RMSE of collaborative filtering is 313.03, and that of content-based filtering is 31.12. RMSE provides the average difference between predicted values and the actual values, which means the predictions made by content-based filtering are closer than those of collaborative filtering in the dataset provided. The error value has improved over time with better data training, and the predictions made by algorithms are accurate.

## 7.6 Results of Analysis of Dataset

The electronic store's sales report for the eleven months of 2020 is shown below in table and line bar format. It shows that sales were low for the year's first four months. Sales increased from July until November. August's total number of users, orders, and sales were high.

	month	total_users	total_orders	total_sales
0	Jan	1823	9201	1729464.93
1	Feb	2259	11026	2216672.31
2	Mar	2606	11676	2841015.58
3	Apr	5495	8752	1669080.19
4	May	17527	29644	7644255.82
5	Jun	14059	28073	7486680.81
6	Jul	30628	56363	16019735.90
7	Aug	35989	72370	27362298.79
8	Sep	20062	49759	16785757.14
9	Oct	14736	68405	19361987.48
10	Nov	8744	43473	11764381.12

*Figure 35 Sales Report of Each Month*



*Figure 36 Sales Report of Each Month in Line Diagram*

The sales report of the electronic store for each day of the week is shown below in table and line bar format. It shows that the number of users increased almost every day of the week, which is highest on Monday. The orders are also similar for Sunday to Friday, highest on Saturday. Due to the high number of orders on Saturday, the sales in Million are also highest on Saturday.

	week_day	total_users	total_orders	total_sales
0	Sun	24037	56979	17222096.04
1	Mon	27212	53470	16223122.63
2	Tue	26665	55495	16127420.47
3	Wed	25323	55514	15897601.88
4	Thu	24690	54211	15462231.02
5	Fri	25295	51655	15719070.48
6	Sat	25027	61418	18229787.55

*Figure 37 Sales Report of Each Day of Week*

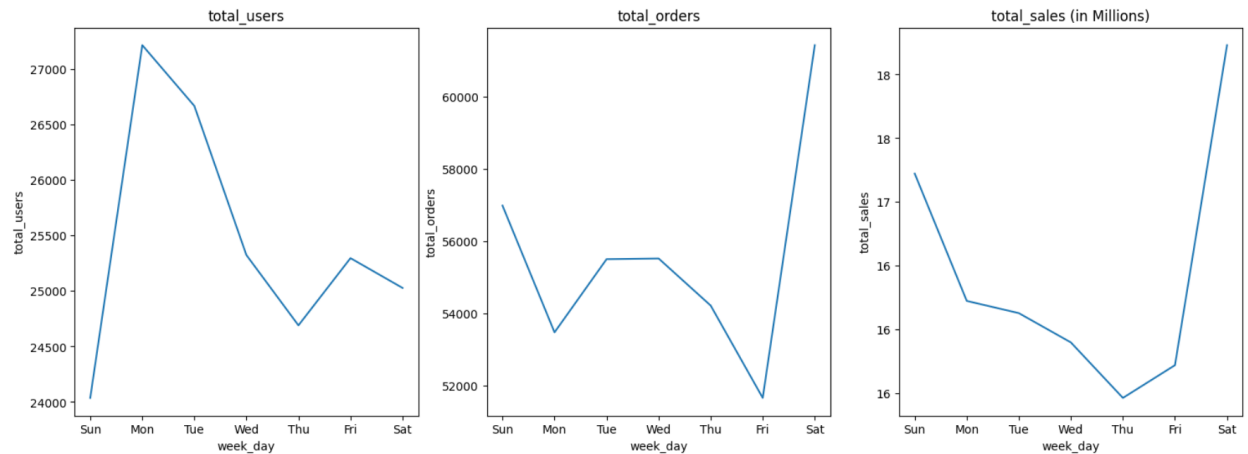


Figure 38 Sales Report of Each Day of Week in Line Diagram

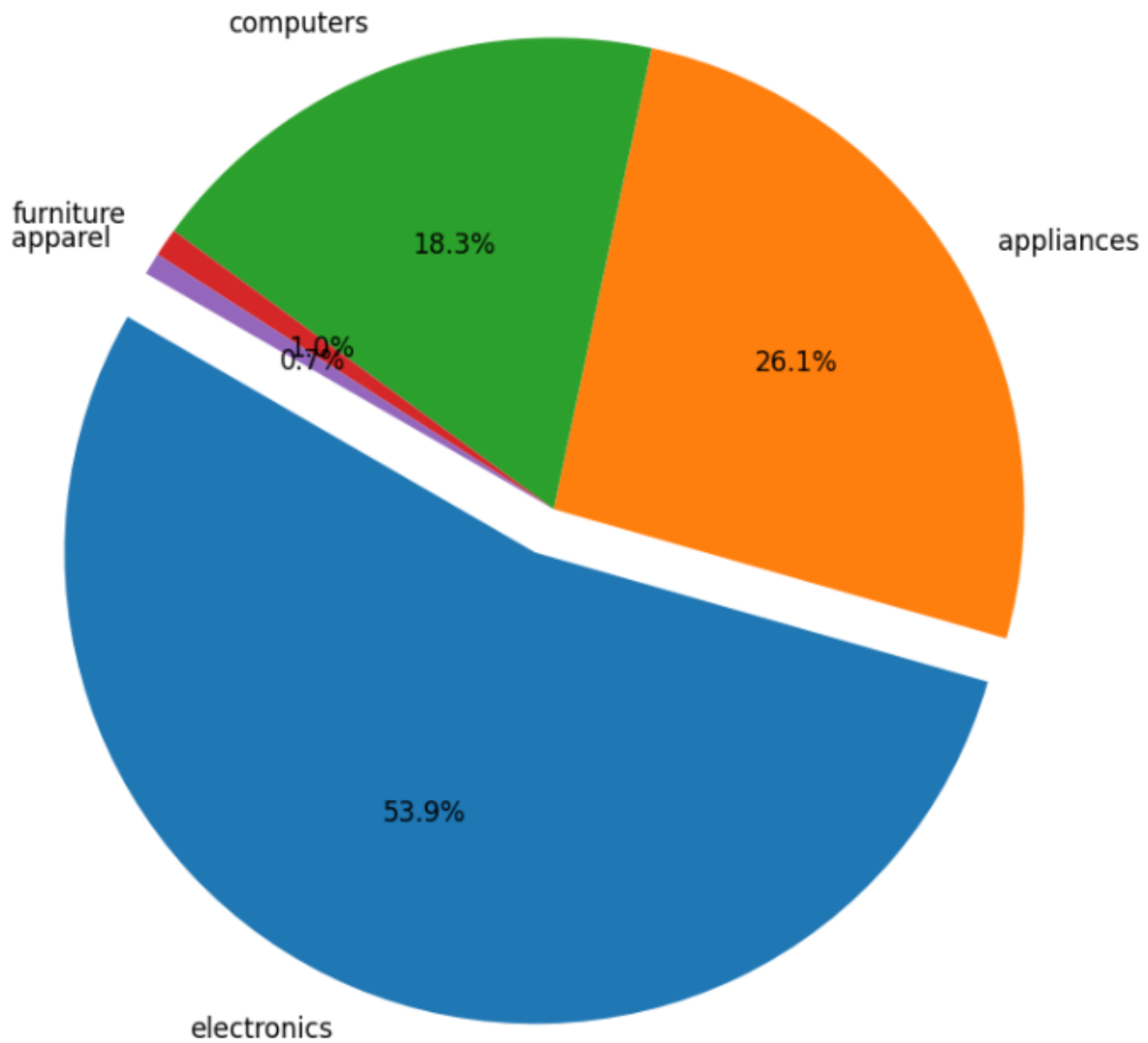
Below are the top 10 electronic store categories and the total number of orders. The electronics category has the highest orders, as well as, appliances and computers.

	category	total_orders
0	electronics	156556
1	appliances	145217
2	computers	72378
3	furniture	21182
4	stationery	8676
5	construction	3959
6	accessories	3019
7	apparel	2664
8	kids	2275
9	auto	1366

Figure 39 Top Orders of Top 10 Categories

The top five categories based on total sales are shown below in pie chart. It shows that electronics has the highest percentage of 53.9% and apparel has the lowest rate of 0.7%, ranking fifth.

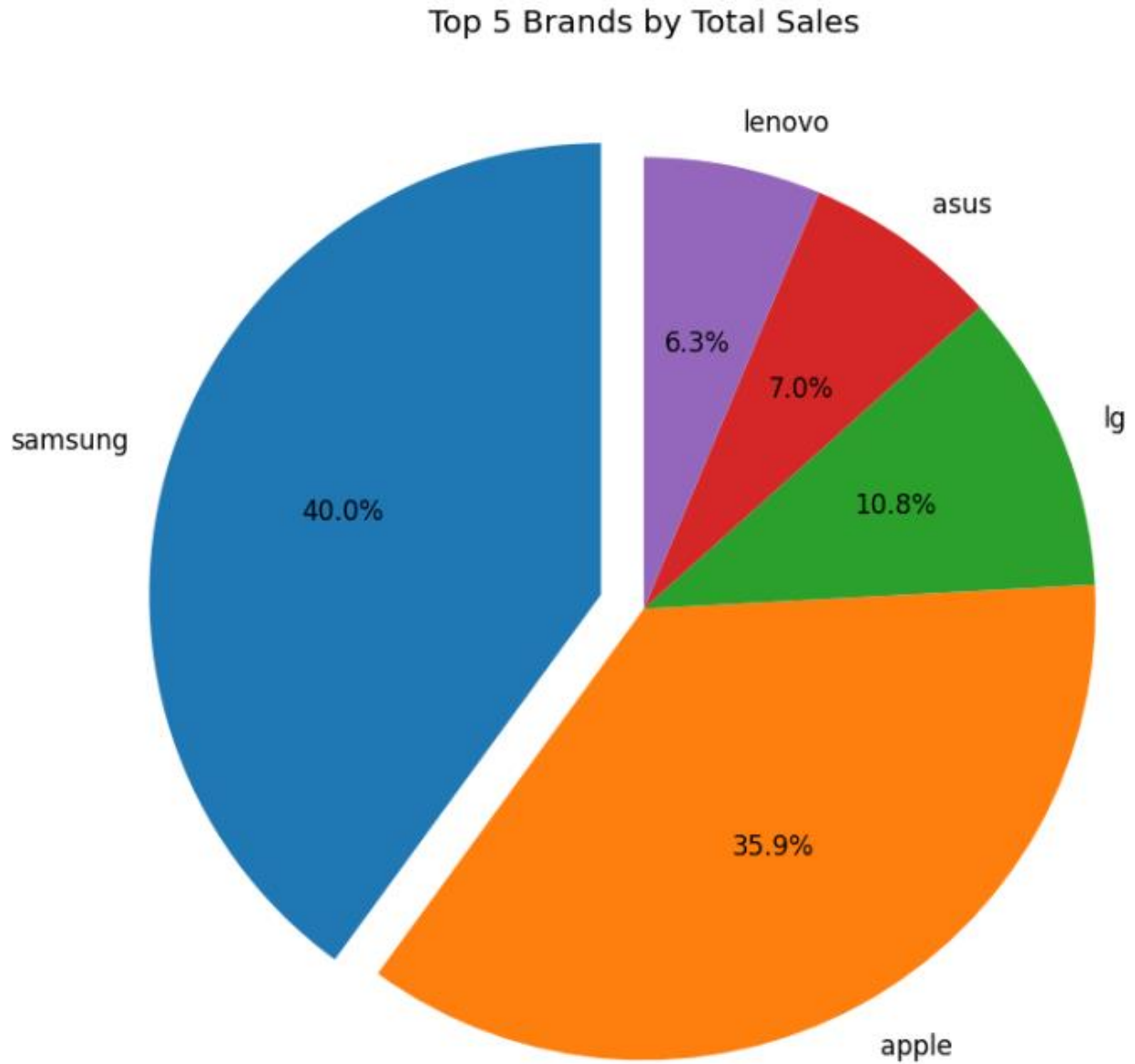
Top 5 Categories by Total Sales



*Figure 40 Top 5 Category by Total Sales in Pie Chart*

The top five brands based on total sales are shown below in a pie chart. Samsung has the highest percentage of 40%, and Lenovo has the lowest rate of 6.3%, ranking fifth.





*Figure 41 Top 5 Brands by Total Sales in Pie Diagram*

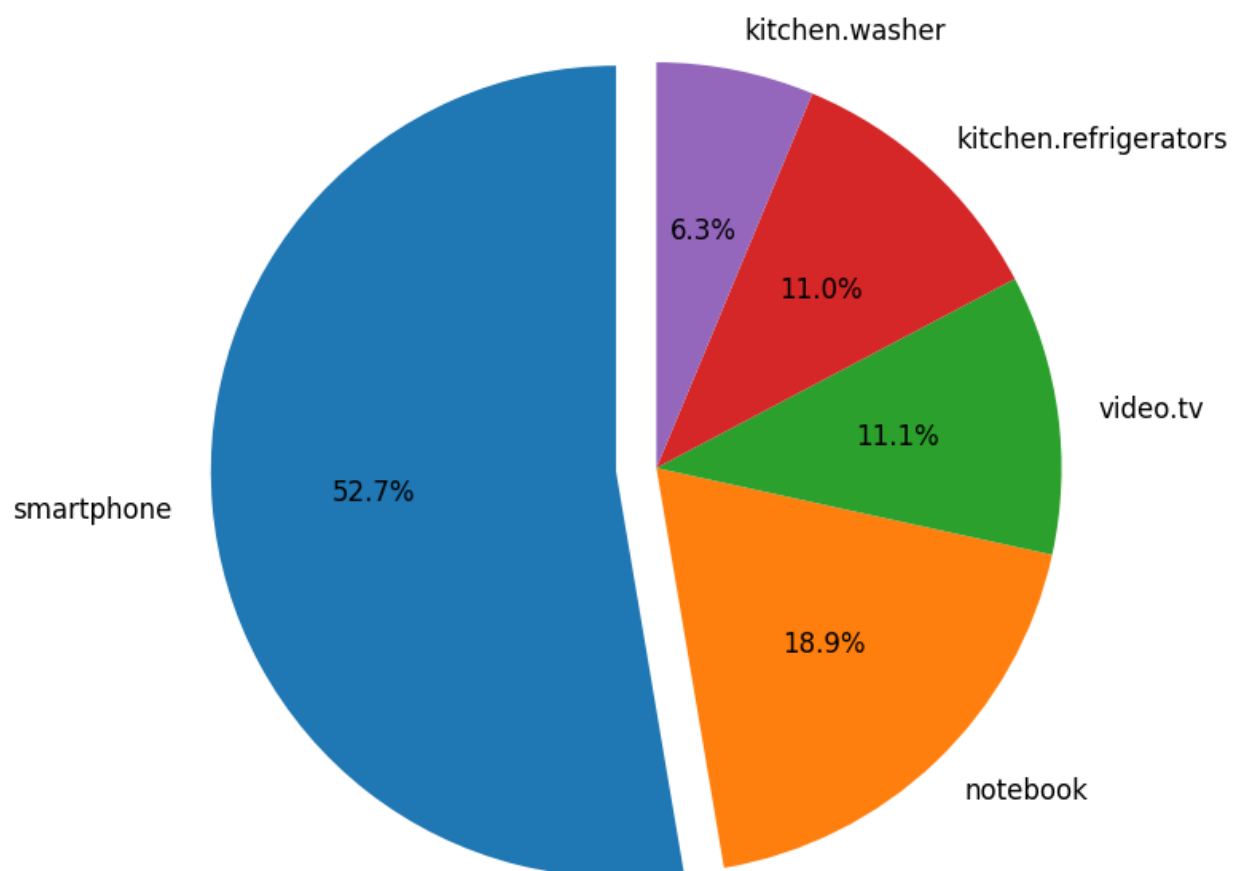
The top 10 brands based on sales are shown below. It shows that the Apple brand has the highest sales along with the Samsung brand.

	product	brand	total_users	total_orders	total_sales
0	smartphone	apple	13795	23043	19163221.51
1	smartphone	samsung	23627	47792	16203110.72
2	notebook	asus	4832	7486	4699457.65
3	notebook	lenovo	4593	7430	4507154.41
4	video.tv	samsung	3640	6080	3903495.59
5	notebook	apple	1456	2249	3292571.10
6	video.tv	lg	3323	5533	3282657.46
7	kitchen.refrigerators	samsung	1995	2973	2191618.54
8	kitchen.washer	samsung	2956	4998	2058715.88
9	kitchen.washer	lg	3262	5327	1950491.25

*Figure 42 Top 10 Brand Based on Sales*

The pie chart below shows the top 5 products based on sales. Smartphone products have the highest percentage of 52.7%, and kitchen washers have the lowest percentage of 6.3%, ranking fifth.

Top 5 Products by Total Sales



*Figure 43 Top 5 Products by Sales in Pie Chart*

## 7.7 Results of E-Commerce Platform

An e-commerce platform has been successfully built using modern web technologies, which are mentioned in this project. The system's front end is built using HTML and CSS, and the back end is developed using PHP. The system runs perfectly without throwing any errors and performing the tasks as expected. The customer, product and purchase details are successfully saved in the database. Customers can view the product list and make a purchase. Admin can manage products and view the order history. The system's main feature, which allows users to view their purchase history, has also been successfully implemented. The purchase history of the user can be viewed in a table by the admin, which looks like the following:



The screenshot shows a web interface with a teal background. At the top, the title "Dataset for Recommendation System" is displayed in white. Below the title is a button labeled "Click to Download Dataset". Underneath the button is a table with 8 columns: event\_time, order\_id, product\_id, category\_id, category\_code, brand, price, and user\_id. The table contains 10 rows of purchase history data.

event_time	order_id	product_id	category_id	category_code	brand	price	user_id
2024-12-20 04:59:24	23	18	1	automobile.car	Tesla	99999999	4
2024-12-20 04:59:48	24	22	16	mobile.phone	Apple	999	5
2024-12-21 06:47:17	25	27	24	electronics.clocks	Titan	121	4
2024-12-22 09:01:25	26	25	23	electronics.audio.headphone	Huawei	111	5
2024-12-22 06:47:32	27	1	16	mobile.phone	Oppo	1234	4
2024-12-24 09:05:22	28	26	23	electronics.audio.headphone	Apple	2099	4
2024-12-24 11:06:09	29	26	23	electronics.audio.headphone	Apple	2099	5
2024-12-28 06:56:25	30	26	23	electronics.audio.headphone	Apple	2099	9
2024-12-28 10:44:32	31	23	1	automobile.car	Yasuda	1000	9
2024-12-28 06:56:41	32	22	16	mobile.phone	Apple	999	9

*Figure 44 Table showing the Purchase History of Customers*

The admin can download the purchase history to create the required dataset by clicking the “Click to Download Dataset” button. After clicking the button, the dataset is downloaded, which looks like the following:

	A	B	C	D	E	F	G	H
1	event_time	order_id	product_id	category_id	category_code	brand	price	user_id
2	12/20/2024 4:59	23	18	1	automobile.car	Tesla	99999999	4
3	12/20/2024 4:59	24	22	16	mobile.phone	Apple	999	5
4	12/21/2024 6:47	25	27	24	electronics.clocks	Titan	121	4
5	12/22/2024 9:01	26	25	23	electronics.audio.headph	Huawei	111	5
6	12/22/2024 6:47	27	1	16	mobile.phone	Oppo	1234	4
7	12/24/2024 9:05	28	26	23	electronics.audio.headph	Apple	2099	4
8	12/24/2024 11:06	29	26	23	electronics.audio.headph	Apple	2099	5
9	12/28/2024 6:56	30	26	23	electronics.audio.headph	Apple	2099	9
10	12/28/2024 10:44	31	23	1	automobile.car	Yasuda	1000	9
11	12/28/2024 6:56	32	22	16	mobile.phone	Apple	999	9

*Figure 45 Dataset after Download*

## 8. Analysis and Discussion

The three machine learning algorithms are implemented, and the performance of all three is measured based on accuracy, diversity, and engagement. The result suggested that collaborative filtering depends upon customer and product interaction data to make further recommendations. In contrast, content-based filtering depends upon the attributes of the products to make further recommendations. In the dataset provided in this project, there is no detail of products; therefore, content-based filtering struggles to give the best recommendation regarding diversity and engagement. Collaborative filtering works better in terms of diversity because it looks beyond customer history and provides diverse product recommendations. However, the accuracy of content-based is better than that of collaborative filtering, and content-based works better for new users as it does not depend upon user and product interaction and can be suggested based on the product description that the new user interacts with. The RMSE of content-based filtering is better than that of collaborative filtering, suggesting that content-based filtering is calculated better. If appropriately applied, a hybrid approach can solve the limitations of these two algorithms. The feature of collaborative filtering should be used to provide recommendations that analyze the patterns of customer and product interactions. For new users, the feature from content-based filtering should be used to show accurate recommendations without being dependent upon customer and product interactions but relying on the products metadata.

The result of the machine learning algorithm and the performance metric are obtained using one dataset with one sort of feature. This is one of the limitations of the result obtained. The machine learning algorithms are not tested using multiple datasets with different sorts of data. The machine learning algorithm works differently with another kind of dataset and provides different results. The other results obtained from various types of datasets are not considered to determine which algorithm works better. For example, a dataset with multiple columns describing a product is not applied, meaning the best result of content-based filtering is not obtained.

The prototype of the e-commerce platform has also been built using modern web technologies. The e-commerce platform successfully provides yearly customer purchase data, which can be used to train and test machine learning algorithms. The limitation mentioned in this project of lacking a diverse dataset to test the machine learning model is also solved by the prototype of the E-Commerce platform. In future, the E-Commerce platform will provide multiple datasets with different features. This dataset can then be applied to the machine learning algorithm, and the performance of each algorithm can be measured. The best algorithm in a particular scenario can also be determined with this.

It can be concluded that the result addresses the research question mentioned above. A recommendation system is successfully created where three different machine learning algorithms are used to analyze the dataset and provide the required recommendation. Applying the performance metrics shows that content-based filtering and the hybrid approach work better for accuracy. For diversity and engagement, collaborative filtering works better. Therefore, all three algorithms have their own features and can be applied to create a recommendation system based on specific user needs.

## 9. Conclusion

A personalized recommendation system is built using three different machine learning algorithms: collaborative filtering, content-based filtering and hybrid approach. A dataset obtained from the Kaggle website is implemented to train and test the algorithms, and their performance is measured using three different metrics: accuracy, diversity and engagement. The result of each algorithm and its performance metrics are obtained and presented. Further, an e-commerce platform using web technologies was also built, and the e-commerce platform successfully provided the required dataset of customer purchase history.

The machine learning code implemented in this project can be applied to create a personalized recommendation system. It can be used in an E-Commerce platform system to show customers relevant and interesting products. The ability of the recommendation system to show relevant products to the customer makes the customer experience of using the E-Commerce platform smoother, increasing their chances of buying many products. Due to this, the revenue of the E-Commerce platform will increase, resulting in a high profit for the company. The prototype of the E-Commerce platform developed in this project can be further developed to create a full-phase E-Commerce system and launch it to the market. The personalized recommendation system that is designed can then be implemented in the e-commerce platform to enhance its overall performance.

### 9.1 Future Works

There are certain things that are remaining in this project that can be developed in future, which are listed below:

- Only three machine learning algorithms are implemented and tested and more can be tested in future such as context-aware or knowledge-based.
- Only three performance metrics are implemented and more can be implemented in future such as scalability.
- The machine learning models are tested using only one dataset. Multiple datasets can be used to test the models for more accurate results.
- The dataset obtained from the E-Commerce platform should be used to train and test the machine learning models. Further, a more diverse dataset should be created from the E-Commerce platform. The dataset should have further information, such as product attributes.
- The personalized recommendation system developed in this project should be implemented in the E-Commerce system, and the performance should be tested.

## 10. References

B.Thorat, P., M. Goudar, R. and Barve, S. (2015) ‘Survey on collaborative filtering, content-based filtering and hybrid recommendation system’, International Journal of Computer Applications, 110(4), pp. 31–36. doi:10.5120/19308-076

Bollen, D. et al. (2010) ‘Understanding choice overload in Recommender Systems’, Proceedings of the fourth ACM conference on Recommender systems. doi:10.1145/1864708.1864724

Bhardwaj, S. (2021) ‘A brief study on web technology’, International Journal of Innovative Research in Computer Science & Technology, pp. 276–280. doi:10.55524/ijircst.2021.9.6.61

Burke, R. (2002) ‘Hybrid Recommender Systems: Survey and Experiments’, User Modeling and User-Adapted Interaction, 12(4), pp. 331–370. doi:10.1023/a:1021240730564

Chen, T. and Guestrin, C. (2016) ‘XGBoost: A Scalable Tree Boosting System’. New York, NY, USA: ACM, pp. 785–794. Available at: <https://dl-acm-org.ezproxy.herts.ac.uk/doi/abs/10.1145/2939672.2939785> (Accessed 13 October, 2024)

Fauzan, R. et al. (2019) ‘Use case diagram similarity measurement: A new approach’, 2019 12th International Conference on Information & Communication Technology and System (ICTS), pp. 3–7. doi:10.1109/icts.2019.8850978

Herimanto, H., Samosir, K. and Ginting, F., 2024. A Comparative Analysis of Content-Based Filtering and TF-IDF Approaches for Enhancing Sports Recommendation Systems. INNOVATICS: Innovation in Research of Informatics, 6(2).

Herlocker, J.L., Konstan, J.A. and Riedl, J. (2000) ‘Explaining collaborative filtering recommendations’, Proceedings of the 2000 ACM conference on Computer supported cooperative work. doi:10.1145/358916.358995

Jannach, D., Felfernig, A., Friedrich, G. and Zanker, M. (2010) Recommender Systems: An Introduction. Cambridge: Cambridge University Press. Available at:



<https://ebookcentral.proquest.com/lib/herts/detail.action?docID=585299> (Accessed 12 October, 2024)

Kang, M. and Tian, J. (2018) 'Machine learning: Data pre-processing', *Prognostics and Health Management of Electronics*, pp. 111–130. doi:10.1002/9781119515326.ch5

Kaplan, J. (2016a) 'Defining artificial intelligence', *Artificial Intelligence*. doi:10.1093/wentk/9780190602383.003.0001

Koren, Y., Bell, R. and Volinsky, C. (2009) 'Matrix Factorization Techniques for Recommender Systems', *Computer* (Long Beach, Calif.), 42(8), pp. 30–37. Available at: <https://ieeexplore-ieee-org.ezproxy.herts.ac.uk/document/5197422> (Accessed 10 October, 2024)

Li, Q. and Chen, Y.-L. (2009) 'Entity-relationship diagram', *Modeling and Analysis of Enterprise and Information Systems*, pp. 125–139. doi:10.1007/978-3-540-89556-5\_6

Ma, C.C., 2008. A guide to singular value decomposition for collaborative filtering. *Computer* (Long Beach, CA), 2008, pp.1-14.

Macaulay, M., 2017. *Introduction to web interaction design: With Html and Css*. Chapman and Hall/CRC.

Mustafa, N. et al. (2017) 'Collaborative filtering: Techniques and applications', 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), pp. 1–6. doi:10.1109/iccccee.2017.7867668

Pryor, M.H., 1998. The effects of singular value decomposition on collaborative filtering.

Raghuwanshi, S.K. and Pateriya, R.K. (2019) 'Collaborative filtering techniques in recommendation systems', *Data, Engineering and Applications*, pp. 11–21. doi:10.1007/978-981-13-6347-4\_2

Ramirez, V. (2021) What is a prototype? Medium. Available at: <https://medium.com/nyc-design/what-is-a-prototype-924ff9400cfd> (Accessed: 12 November 2024).

Ricci, F., Rokach, L. and Shapira, B. (2010) 'Introduction to recommender systems handbook', Recommender Systems Handbook, pp. 1–35. doi:10.1007/978-0-387-85820-3\_1

Richards, G. et al. (2010) 'An analysis of the dynamic behavior of JavaScript programs', Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation. doi:10.1145/1806596.1806598

Sanner, M.F., 1999. Python: a programming language for software integration and development. J Mol Graph Model, 17(1), pp.57-61.

Sotnik, S., Manakov, V. and Lyashenko, V., 2023. Overview: PHP and MySQL features for creating modern web projects.

Thomas, R.N. and Gupta, R. (2020) 'A survey on machine learning approaches and its techniques', 2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1–6. doi:10.1109/sceecs48394.2020.190

Wlodarczak, P. (2023) 'Agile methods', Agile Software Development, pp. 7–13. doi:10.1201/9781003301707-3

Zhang, S., Yao, L., Sun, A. and Tay, Y. (2019) 'Deep learning based recommender system: A survey and new perspectives', ACM computing surveys, 52(1), pp. 1–38. Available at: <https://dl.acm-org.ezproxy.herts.ac.uk/doi/10.1145/3285029> (Accessed 10 October, 2024)

Zhang, S. et al. (2005) 'Using singular value decomposition approximation for collaborative filtering', Seventh IEEE International Conference on E-Commerce Technology (CEC'05), pp. 257–264. doi:10.1109/icect.2005.102

Zhang, S. (2024) 'Restaurant recommendation system based on TF-IDF vectorization: Integrating content-based and collaborative filtering approaches', Advances in Intelligent Systems Research, pp. 610–618. doi:10.2991/978-94-6463-370-2\_62