

LAPTOP RECOMMENDATION SYSTEM USING CONTENT-BASED FILTERING REPORT.

1. Introduction

The rapid growth in the technology sector has influenced consumers on electronics and gadgets needed for both personal and professional use. The rise of multiple electronic brands, Specifically, laptops, has made it quite difficult to select a suitable laptop according to their needs. A recommendation system is known to make the decision process easier by filtering and presenting relevant options (Acharya et al., 2023). These recommendation systems improve the user's search experience by reducing the amount of effort and time needs to find an ideal laptop Nguyen et al. (2019).

Today, recommendation systems are essential in retail, e-commerce and all online services. For movies, Netflix can easily recommend your next watch and for products, Amazon can easily recommend a product because of an established recommendation system. However, on the technology section, there is little knowledge regarding recommendation on what is the most preferred laptop hence the need to make a recommendation system.

In the retail industry, Laptop vendors encounter challenges in helping customers find the best-suited laptop from the variety. A user-friendly recommendation system can enhance the consumers' shopping experience by offering personalized suggestions depending on their preference. Our project aims to support this by developing a Laptop Retailer's Recommendation System that will provide customers with tailored laptop options. This will ensure retailers can smoothen the decision-making process of consumer and overall improve customer satisfaction.

Our project employs an AI-powered recommendation system known as content-based filtering (CBF). Content-based filtering provides recommendations based on the items the user interacts with or shows interest in Sridhar et al. (2022). Studies have also proven that CBF is transparent in giving recommendations (Naveed & Ziegler, 2019), while performing highly and accurately making it the ideal choice of our laptop recommendation system. According to Chunhapran (2022), CBF is applicable to many domains, and this assures us to use it for our recommendation system.

This goal of this project is to create a content-based laptop recommendation system that uses the laptop specifications and similarity matrix to generate suggestions to users. It will examine the specification of laptops and find laptops like the one a user had previously shown in.

2.0 Problem Statement

As the variation of laptops keeps increasing, it becomes increasingly difficult for users to find the laptop that best suits their preference. When users search for a laptop online, they will select an interesting laptop model and explore its details. However, a problem arises when they wish to discover similar laptops to the one that they are currently viewing.

Traditionally, users may go to the search filter provided by an online platform to look for similar laptops. However, this approach requires users to have a basic understanding of the specification of the laptop they are currently viewing. This approach was considered as inefficient and time-consuming, as users must remember the laptop specification and manually adjust the search filter to get similar laptops. Moreover, some of the online shopping platforms do not offer a search filter for their customers, forcing them to rely on general keyword search, which is not only time-consuming but also results in low-quality matches.

Due to the problem mentioned earlier, it is essential to provide a list of recommended laptops like the one that users are currently viewing. When users are interested in the laptop that they are currently viewing, they are naturally curious about other similar options. Therefore, by displaying a recommendation of similar laptops list directly under the viewed laptop, users can easily explore and compare the similar alternative options without any extra effort. This improves the user experience by eliminating the need for users to perform manual searching, making the comparison process smooth and more efficient.

Our team aims to develop a recommendation system that can recommend 10 similar laptops based on the laptop that the user is currently viewing. These top 10 recommended laptops will be displayed beneath the laptop that the user is currently viewing, enabling users to easily compare similar options in a smooth and efficient manner.

2.1 Business Objectives

- To build a recommendation system to provide users with highly relevant laptop recommendations based on the one they are currently viewing.
- To simplify product comparison which helps users easily compare laptops by providing 10 similar laptops to the one they are currently viewing.
- To improve user satisfaction and enhance the overall shopping experience by offering accurate, AI-driven recommendations that align with user preferences.

2.2 Problem and challenges without AI

- There are thousands of laptop models available on the online shopping platform, making users having difficulty to manually compare similar laptops that match their interests.
- Without a good recommendation system, displaying random laptop suggestions will waste screen display space, as users are likely to ignore items that they have no interest in.
- If users spend too much time searching for items they want to buy on an online shopping platform, they may feel frustrated and leave the site, resulting in a decline in sales.
- Traditional search filters developed for users to filter out their desired item often require a huge amount of time and effort for manual filtering.
- Users might miss better laptop options that suit their needs if there is absence of an AI-driven recommendation system.

3.0. Elaboration of the AI Solution

We implemented a content-based filtering approach to recommend laptops similar to the one a user is interested in. To enhance the recommendation process, users can input their desired laptop using text. Text processing and content-based filtering techniques were applied to identify the user's desired laptop from our dataset. After identifying the target laptop, the system again applies content-based filtering to fetch 10 similar laptops based on its attributes. To improve accuracy, we combined two similarity techniques: String-Based Similarity (using Cosine Similarity) and Numerical Similarity (using Euclidean Distance). A hybrid model integrating these methods was developed to optimize recommendation accuracy.

3.1. Solution Methodology Lifecycle

To develop a laptop recommendation system, our team has taken the following steps.

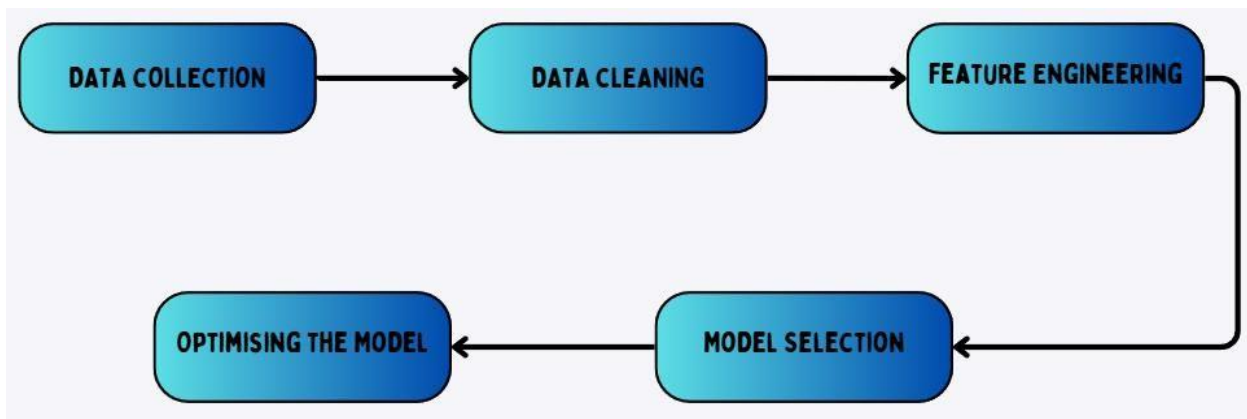


Figure 1: Methodology Life Cycle

3.1.1 Data Collection

During the data collection phase, it is essential to consider the attributes of the laptop that should be included in the dataset, as well as the dataset's size. After reviewing existing literature on the key features users prioritize when purchasing a laptop, we gathered relevant data from open sources. A laptop dataset containing these attributes was obtained from Kaggle ([Laptop Data link](#)) as it is sufficiently large to represent a diverse range of popular laptops, enhancing both the model's presentation and validity, while remaining manageable in terms of computational efficiency. The laptop price dataset contains 1303 entries and 13 columns. The key attributes of the dataset are shown in Table 1 below:

Table 1: Laptop Dataset

Variable	Type	Description
Laptop_ID	Integer	Identifier of each laptop
Company	Categorical	The laptop Manufacturer
Product	Categorical	The brand and model of the laptop
TypeName	Categorical	Type of Laptop(Notebook, Ultrabook, Gaming)
Inches	Numerical	The screen size of the laptop in inches
ScreenResolution	Categorical	Describes screen resolution of the laptop
Cpu	Categorical	Details of the CPU specifications
Ram	Categorical	Information about the Laptops RAM
Memory	Categorical	Indicates the hard disk or SSD memory capacity
Gpu	Categorical	Specifications for the Graphic Processing Unit details
OpSys	Categorical	Operating System installed in the laptop
Weight	Numerical	Describes the weight of the laptop

Price_euros	Numerical	Price of the laptop in Euros.
-------------	-----------	-------------------------------

3.1.2 Data Cleaning

After collecting the dataset from Kaggle, we need to perform data cleaning to ensure the data is complete, accurate, and useful for analysis. The cleaning process includes handling missing values, identifying and removing duplicate entries, converting data into appropriate formats, and eliminating unnecessary units from numerical values. Additionally, irrelevant columns, such as the dataID column, have been removed as they do not contribute to building the model.

3.1.3 Feature Engineering

Since our dataset includes a diverse range of laptops, feature engineering is essential to eliminate irrelevant or redundant data, improving the model's ability to identify similarities between laptop attributes. In this phase, several techniques have been applied. First, feature transformation was performed on numerical data by converting values from string format to numerical form. Additionally, feature extraction was conducted by extracting general categories from laptop attributes. Lastly, feature construction was implemented by creating new meaningful features from the raw data.

3.1.4 Model Selection

First, since we have an additional feature that allows users to input their preferred laptop using text, text processing and content-based filtering with TF-IDF (Term Frequency-Inverse Document Frequency) will be utilized to analyze the user's textual laptop descriptions and identify the most similar laptop from our database.

To recommend the 10 most relevant laptops based on user interest, content-based filtering is applied by comparing laptop attributes and calculating similarity scores. Two techniques are used in this model:

String-Based Similarity, which measures similarity between laptop attributes stored as text data. This is achieved using Cosine Similarity.

Numerical Similarity, which computes similarity between laptop attributes represented as numerical values. This is done using Euclidean Distance.

Both techniques are then combined to develop a model capable of comparing laptops based on both text-based and numerical attributes, ensuring more accurate recommendations.

3.1.5 Optimising the Model

Two different similarity measure techniques were used to create a model that is able to compare laptops based on both text-based and numerical attributes. To distinguish the relative importance of each technique, a weighting has been added to the model. The sum of the weightage for these two techniques were set to be 1.

To evaluate the model's accuracy to recommend laptops that are similar to the laptop that users are interested in, an accuracy test has been developed. This accuracy test can be used to assess the accuracy of the model using different combinations of weight. An optimisation technique called Grid Search was implemented that is able to compute the accuracy of the model based on different combinations of weightage. The accuracy of the model can be optimised by using the combinations of weightage found from the Grid Search that gives the highest model's accuracy.

3.2 Solution Design

Content-based filtering was selected as the primary AI algorithm for developing our laptop recommendation system. This method uses the laptop's attributes to recommend other laptops with similar attributes to the user's preferred laptop. The Content-Based filtering allows the recommendation system to recommend laptops based on explicit user preferences, enhancing personalization by optimizing recommendations to individual needs and interests.

The String-Based Similarity using Cosine Similarity and Numerical Similarity using Euclidean Distance were the two main techniques used to compute the similarity between laptops using their attributes. A model was built incorporating both String-Based Similarity and Numerical Similarity, with assigned weightage for each technique. The model was then being optimised by accuracy test and Grid Search algorithm.

There are a few other AI techniques or algorithms selected to develop the system. These techniques are text processing used for retrieving laptop attributes from user's text input, and TF-IDF (Term Frequency-Inverse Document Frequency) for laptop overview matching.

3.2.1 User text input search

In order to increase the user experience, users can input a text description of their preferred laptop to get the top 10 recommendation from the recommendation system. Text processing

will be used to extract key laptop attributes from the user's text input. Text processing is essential as it transforms the user's text input into laptop overview, a structured string that includes laptop attributes that matches the format of the overview data in our dataset. The conversion of the user's text input into a laptop overview is crucial, as it will be used for comparison with other laptop overviews in our dataset using TF-IDF. The reason TF-IDF was used for comparison of laptop overviews is because it can assign lower weights to frequently occurring words, such as "GB" or "Inches", to ensure useful terms are having greater impact.

3.2.1.1 Text processing

To extract key laptop attributes from user input text, the text is first cleaned by tokenizing into smaller words, removing English stop words and converting all words to lowercase. Next, key laptop attributes that present in the overview in our dataset will be identified. These key laptop attributes will be extracted from the cleaned words to generate a laptop overview based on the user's input that has the same format of the laptop overviews in our dataset.

3.2.1.2 Laptop Overview Matching Using TF-IDF

Once the laptop overview string has been generated from the user input by text processing, it will be compared with all laptop overviews in our dataset. This comparison is done by computing cosine similarity scores between laptop overview generated from the user's input and all laptop overview in our dataset. By using a TF-IDF vectorizer with dot product, the similarity score between laptop overview generated from the user's input and all laptop overviews in our dataset can be computed. The laptop from our dataset with the highest similarity score to the user's input laptop will be selected as the best match. This matched laptop will then be used to find 10 similar laptops based on its attributes.

3.2.2 String-Based Similarity (Cosine Similarity)

The first technique used to evaluate the similarity between laptops is String-Based Similarity with Cosine Similarity. This approach will compute similarity scores solely based on the laptop attributes with string type. All string attributes will be undergone by converting text to lowercase and removing the space between words in each string attribute. Next, a new column is created for each laptop entry in the dataset, where all relevant string attributes are combined into a single text string, separated by spaces. The consolidated text in the newly created column then used to measure similarity across laptops. The CountVectorizer technique is applied alongside Cosine Similarity to get similarity scores.

The reason why CountVectorizer was used is because it does not down-weight frequently occurring laptop attributes, ensuring that all attributes contribute equally to the similarity calculation. Cosine similarity was chosen as it is highly effective for comparing string-based data, resulting in it being the best choice for computing similarity between string attributes.

3.2.3 Numerical Similarity (Euclidean Distance)

The second technique used to assess the similarity between laptops is Numerical Similarity with Euclidean Distance. This approach will compute similarity scores exclusively based on the laptop attributes with numerical type. To ensure consistency, all numerical attributes are first cleaned by converting them into float type. These numerical attributes were then normalized using the MinMaxScaler(). Finally, the Euclidean Distance was used to compute the similarity between the normalized numerical attributes to determine similarity scores between laptops.

In this section, since we are dealing with the numerical attributes, Euclidean Distance is most effective technique for comparing numerical data, as it effectively captures differences in magnitude and scale. MinMaxScaler() is applied to normalize the numerical data, to ensure that all numerical attributes contribute equally to the Euclidean distance calculation, preventing any single feature from dominating the results.

3.2.4 Similarity Model

From the above, two different techniques have been developed to analyse similarity between attributes of two different data types. Cosine Similarity is well-suited for string data, while Euclidean Distance is more effective for numerical data. Since content-based filtering cannot compute similarity for both string and numerical attributes using a single technique, a hybrid approach was introduced. By combining both methods, a more robust content-based filtering model was built for the recommendation system. This hybrid model is using both Cosine Similarity and Euclidean Distance, allowing it to do similarity computation across both numerical and text attributes. Additionally, a weighting factor has been added to balance the influence of each technique within the model.

The Figure 2 below shows the proposed model

$$\text{Combined Similarity} = \alpha \times \text{Cosine Similarity} + \beta \times \text{Numerical Similarity}$$

Figure 2: Hybrid- Model

where,

α is the weightage factor for Cosine Similarity

β is the weightage for Numerical similarity

The sum of the weightage for these two techniques were set to be 1.

The combined similarity is the similarity scores that combines similarity between string attributes and similarity between numerical attributes.

With this model it enables the content-based filtering to accurately compute similarity using both string and numerical attributes.

3.2.4.1 Accuracy test

An accuracy test has been developed to assess the accuracy of the recommendations system by comparing the attributes of a target laptop with all recommended laptops. Only the exact matching of terms in the attributes will increase the accuracy value. Any slight difference in the attributes between recommended laptop and target laptop will result in decreases of accuracy. This algorithm works by first calculating the maximum possible matches for both string and numerical attributes. Then it iterates through each of the recommended laptops and counts the number of exact matches with the target laptop for both string and numerical attributes. Finally, it computes accuracy scores as the ratio of matches to maximum possible matches.

3.2.4.2 Optimization with Grid Search

An optimization algorithm was developed to determine the optimal weight values α (alpha) and β (beta) in the similarity model that gives the highest accuracy. This process utilizes Grid search, where a predefined range of α and β values is explored with an increasing step size of 0.02, ensuring high precision and computational efficiency.

For each α and β combination, the combined similarity score is computed and evaluated using an accuracy test to measure the model's performance. After the accuracy has been measured for all α and β combinations, the pair that results in the highest accuracy is selected as the optimal weight values.

Grid search is able to systematically evaluate all possible combinations of weightage. This ensures that every potential pairing is assessed and guarantees that the algorithm consistently finds the optimal solution for the model.

3.2. Solution Implementation

As stated earlier, the goal of this project is to develop an AI-driven Laptop Recommendation system that improves the user experience by providing an easy comparison of laptops that they are interested in.

3.2.1. Importing Libraries and Data Overview

We utilized several Python libraries to handle data processing, similarity measurement and visualization. This allowed us to develop a robust content-based laptop recommendation system. The libraries utilized are shown below:

```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import MinMaxScaler
from scipy.spatial.distance import euclidean
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords

pip install nltk
```

Figure 3: Importing Libraries

The dataset consists of 1303 laptop entries and 13 columns, both categorical (object) and numerical (float64, int64). The presence of both numerical and categorical attributes allows for a detailed similarity analysis based on the laptop features.

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 13 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   laptop_ID           1303 non-null   int64
1   Company             1303 non-null   object
2   Product             1303 non-null   object
3   TypeName            1303 non-null   object
4   Inches              1303 non-null   float64
5   ScreenResolution    1303 non-null   object
6   Cpu                 1303 non-null   object
7   Ram                 1303 non-null   object
8   Memory              1303 non-null   object
9   Gpu                 1303 non-null   object
10  OpSys               1303 non-null   object
11  Weight              1303 non-null   object
12  Price_euros         1303 non-null   float64
dtypes: float64(2), int64(1), object(10)
memory usage: 132.5+ KB
```

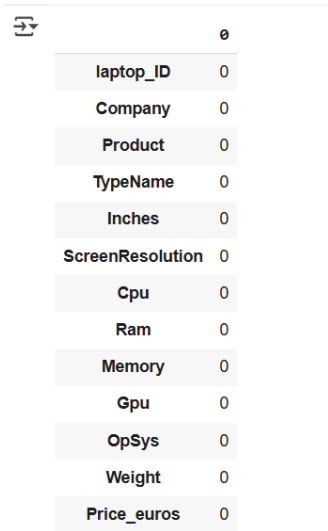
Figure 4: Dataset Information

3.2.2. Data Cleaning & Data Optimization

These are critical steps taken to ensure the quality and reliability of the data, which directly impact the accuracy and validity of the results (Morr et al., 2022).

3.2.2.1. Checking and Handling Missing Values.

This step is crucial for identifying any missing values in the dataset using the `df.isnull().sum()` command. The outcome shown in Figure 5 indicates that this dataset has no missing values; hence, no further handling is needed.



	0
laptop_ID	0
Company	0
Product	0
TypeName	0
Inches	0
ScreenResolution	0
Cpu	0
Ram	0
Memory	0
Gpu	0
OpSys	0
Weight	0
Price_euros	0

Figure 5: Total Missing Values in each laptop attributes

3.2.2.2. Handling Duplicates.

Handling duplicates is needed to ensure that certain values are not overrepresented. When a specific laptop model appears many times, the AI model may over-prioritize it. The duplicates also increase computational overload, which may lead to incorrect recommendations. In our project, a duplicated laptop may be recommended twice in the output, leading to decreased efficiency of the model. Luckily, in our dataset, there were no duplicates found as shown in Figure 6.

```
Total number of duplicate rows: 0
```

Figure 6: Total number of Duplicates

3.2.2.3. Checking and Handling Unique Values.

After ensuring that there are no duplicates, unique values were checked using `df.nunique()` to understand the structure of our data and decide how to handle the features of our data. Figure 7 shows the results.

	e
laptop_ID	1303
Company	19
Product	618
TypeName	6
Inches	18
ScreenResolution	40
Cpu	118
Ram	9
Memory	39
Gpu	110
OpSys	9
Weight	179
Price_euros	791

dtype: int64

Figure 7: Total unique value in each laptop attributes

After checking the dataset, Laptop_ID had 1303 unique values, which means that it does not have any contribution to the recommendation system. It will not help AI learn any useful patterns. The Laptop_ID is like the row numbers in our dataset, which means it has no relationship with the laptop specifications hence, it was dropped. The Figure 8 below shows the remaining columns after removal of the Laptop_ID.

```
False
Remaining columns: Index(['Company', 'Product', 'TypeName', 'Inches', 'ScreenResolution', 'Cpu',
                          'Ram', 'Memory', 'Gpu', 'OpSys', 'Weight', 'Price_euros'],
                          dtype='object')
```

Figure 8: Remaining Attributes Columns

3.2.3. Data Transformation

The formatting of the data was done to ensure data consistency and scale different columns to ensure the proper performance of our recommendation system.

3.2.3.1. Converting RAM to integer

The RAM column originally had values like 4GB,8GB or 16GB, which cannot be processed; hence, the GB part needed to be eliminated as shown in Figure 9. This will help with feature engineering and will allow AI models to correctly analyse the laptop specifications of the RAM.

```

Before Removing 'GB' from Ram:
Ram
0 8GB
1 8GB
2 8GB
3 16GB
4 8GB

After Removing 'GB' from Ram:
Ram
0 8
1 8
2 8
3 16
4 8

```

Figure 9: Removing 'GB' from Ram attributes

3.2.4. Feature Engineering

3.2.4.1. Screen Resolution

This column contains the quality features that enable a laptop screen to deliver quality images, pixel density, and the number of pixels on the screen. To ensure the column was suitable for our AI model, the display type and the screen resolution were extracted, the screen width and height were separated, the code ensured there were no missing values.

By doing all these steps, the display features would improve similarity-based recommendations and, at the same time, ensure data consistency. Figure 10 shows the top rows of our dataset after running the code.

Laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros	Display_Type	Screen_Width	Screen_Height
0	1	Apple MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1339.69	IPS Panel Retina Display	2560.0	1600.0
1	2	Apple MacBook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	898.94	Others	1440.0	900.0
2	3	HP 250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	575.00	Full HD	1920.0	1080.0
3	4	Apple MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	2537.45	IPS Panel Retina Display	2880.0	1800.0
4	5	Apple MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	1803.60	IPS Panel Retina Display	2560.0	1600.0

Figure 10: Added three extra columns into the dataset, Display type, Screen Width, Screen Height

3.2.4.2. Extracting CPU Brand and Speed

All values in the CPU column were ensured to include a specific GHz (Gigahertz) value, which is important to accurately analyse the processor speed. This was achieved by ensuring there is no incomplete information used in the CPU column and by ensuring model accuracy through standardizing CPU specifications to make similar calculations and ensure more precise recommendations. This was achieved using the code below:

```
# Ensure that all cpu column are having specific GHz value
df["cpu_ends_with_GHz"] = df["Cpu"].astype(str).str.endswith("GHz")
df["cpu_ends_with_GHz"].value_counts()
# Drop the temporary column that used to count number of cup ends with GHz
df.drop(columns=["cpu_ends_with_GHz"], inplace=True)
```

Figure 11: Code for ensuring all CPU ends with “GHz”

In the next step, CPU specifications were extracted into 2 separate features, namely CPU Brand and CPU Speed (GHz), to ensure that the processor details are structured for better analysis and comparison. The CPU brand was extracted by selecting the first word from the CPU Column as it represents the manufacturer; the CPU Speed was then extracted using regular expressions, and the value was converted to float for numerical analysis.

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros	Display_Type	Screen_Width	Screen_Height	Cpu_Brand	Cpu_Speed
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1339.69	IPS Panel Retina Display	2560.0	1600.0	Intel	2.3
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	898.94	Others	1440.0	900.0	Intel	1.8
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	575.00	Full HD	1920.0	1080.0	Intel	2.5
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	2537.45	IPS Panel Retina Display	2880.0	1800.0	Intel	2.7
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	1803.60	IPS Panel Retina Display	2560.0	1600.0	Intel	3.1

Figure 12: Added two extra columns into the dataset, Cpu Brand and Cpu Speed

3.2.4.3. Converting Memory to a standardized GB Format

The memory column was also transformed by standardizing all storage values into Gigabytes because some laptops had their storage listed in Terabytes and other Gigabytes, making the comparison difficult. This was achieved using the self-implement function called `convert_memory(mem)` that extracted and converted all the memory values into GB. This was achieved by multiplying TB by 1024 to convert them to GB. Lastly, all GB values were totaled to display the memory in GB. This step ensures all laptops are using a single unit, GB, which makes it easier to analyze, compare, and use for recommendations.

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros	Display_Type	Screen_Width	Screen_Height	Cpu_Brand	Cpu_Speed	Total_Memory_GB
	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1339.69	IPS Panel Retina Display	2560.0	1600.0	Intel	2.3	128
	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	898.94	Others	1440.0	900.0	Intel	1.8	128
	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	575.00	Full HD	1920.0	1080.0	Intel	2.5	256
	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	2537.45	IPS Panel Retina Display	2880.0	1800.0	Intel	2.7	512
	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	1803.60	IPS Panel Retina Display	2560.0	1600.0	Intel	3.1	256

Figure 13: An extra column added into the dataset, Total memory GB

3.2.4.4. Reducing GPU Variations

A few steps were undertaken to ensure the GPU data is suitable for usage in the recommendation system. First, all values were converted to lower case. The next step confirmed that all GPU names contained names of the 4 major manufacturers: NVIDIA, AMD, Intel, or ARM. The data was then confirmed to show the output in Figure14.

```
All GPUs contain 'NVIDIA', 'AMD', 'Intel', or 'ARM': True
```

Figure 14: Confirmation of 4 GPU types in dataset

The next step involved standardizing GPU information by creating a function `extract_gpu_brand(gpu)` which categorized each GPU into one of the four key brands: **NVIDIA**, **AMD**, **Intel**, or **ARM**. A new column named `Gpu_Brand` was formed to ensure structured classification. Figure 15 shows the count each GPU Brand displayed.

```
Gpu_Brand
Intel      722
NVIDIA    400
AMD       180
ARM         1
Name: count, dtype: int64
```

Figure 15: Total number of laptops for each GPU in dataset

Lastly, GPUs were classified according to performance. This classification helped distinguish between integrated, gaming, and workstation GPUs, which significantly impact a laptop's performance for different use cases. To obtain the analysis of the GPU across the dataset, the count of each GPU tier was displayed (Figure16)

```
Gpu_Tier
Integrated      724
Gaming GPU     369
AMD GPU        174
Workstation GPU   31
AMD Workstation GPU   5
Name: count, dtype: int64
```

Figure 16: Total number of laptops for each GPU tier in dataset

3.2.4.5. Standardizing and Categorizing Operating Systems

Since laptops come with different operating systems, the `OpSys` column was processed to extract the operating system in a structured way. To analyse the distribution of the `OpSys`, the count of each category was displayed.

```

OpSys_Name
Windows    1125
Other       66
Linux       62
Chrome      27
Mac          21
Android      2
Name: count, dtype: int64

```

Figure 17: Total number of laptops for each Operating system in dataset

3.2.4.6. Creating an Overview Column

A new column names overview was created to store the summary of each laptop specification. This was done by combining information from multiple columns (Company, Product, Inches, Total_Memory_GB, OpSys_Name, Cpu_Brand) to create a single laptop text description. The processes added units and separators to make the laptop description readable.

Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	...	Display_Type	Screen_Width	Screen_Height	Cpu_Brand	Cpu_Speed	Total_Memory_GB	Gpu_Brand	Gpu_Tier	OpSys_Name	Overview
MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	...	IPS Panel Retina Display	2560.0	1600.0	Intel	2.3	128	Intel	Integrated	Mac	Apple MacBook Pro 13.3 Inches 128GB Mac Intel
MacBook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.6GHz	8	128GB Flash Storage	Intel HD Graphics 6000	...	Others	1440.0	900.0	Intel	1.8	128	Intel	Integrated	Mac	Apple MacBook Air 13.3 Inches 128GB Mac Intel
250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	...	Full HD	1920.0	1080.0	Intel	2.5	256	Intel	Integrated	Other	HP 250 G6 15.6 Inches 256GB Other Intel
MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	...	IPS Panel Retina Display	2880.0	1800.0	Intel	2.7	512	AMD	AMD GPU	Mac	Apple MacBook Pro 15.4 Inches 512GB Mac Intel
MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	...	IPS Panel Retina Display	2560.0	1600.0	Intel	3.1	256	Intel	Integrated	Mac	Apple MacBook Pro 13.3 Inches 256GB Mac Intel

Figure 18: An extra column added into the dataset, Overview

3.2.5. Implementing Language Processing Laptop Search

To enhance user experience, a feature to allow users to search for laptops using natural language. The goal is to ensure that users can freely describe their preferred laptop in words and the system can automatically extract the relevant specifications to match with available laptops.

For example, if a user inputs a query like, "I want an HP brand laptop with a 15.2-inch notebook with 256GB," the system will perform the following steps:

- It will extract the key laptop features. The `extract_query_details(user_input, df)` function will process the input of the user and identify the important words like brand, type, screen size, storage, RAM, OS and CPU.
- It will then convert the details to a Overview format which is a text string as shown in Figure 19.

```
HP notebook 15.2 Inches 256GB
```

Figure 19: User input converted to Overview format

This format will allow for easy comparison with the dataset in the overview column to find matching laptops.

3.2.5.1. Appending User Input to the Overview Column

To enhance the recommendation system's ability to match laptops based on natural language input, we implemented a process to integrate user queries into the dataset. This allows the system to compare user requirements with existing laptop descriptions for better recommendations.

Overview	
1299	Lenovo Yoga 900-13ISK 13.3 Inches 512GB Window...
1300	Lenovo IdeaPad 100S-14IBR 14.0 Inches 64GB Win...
1301	HP 15-AC110nv (i7-6500U/6GB/1TB/Radeon 15.6 In...
1302	Asus X553SA-XX031T (N3050/4GB/500GB/W10) 15.6 ...
1303	HP notebook 15.2 Inches 256GB

Figure 20: Append user input in overview format as an extra row under overview column

3.2.5.2. Implementing Text-Based Laptop Recommendations Using TF-IDF and Cosine Similarity

To improve the recommendation system, we implemented a text similarity-based approach that allows users to find laptops based on their natural language input. This method uses TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity to match user queries with laptop descriptions.

To convert laptop descriptions into numerical format, we applied TF-IDF Vectorization using `TfidfVectorizer()` from `scikit-learn`. This step assigned higher importance to the unique words while filtering out the common stop words like the, a and. This ensured only meaningful words are considered.

Next, we computed the **dot product** to measure how closely a given laptop description matches others in the dataset.

To find the most relevant laptop based on user input, we:

- Retrieved similarity scores for the user's input compared to all laptops.
- Sorted the laptops by **highest similarity score**.
- Selected the **most relevant** laptop (excluding the input itself).

This process **identifies the best-matching laptop** for the user's query based on its similarity to existing laptop descriptions. In this case, the laptop index was:

After identifying the **best-matching laptop** using **TF-IDF and Dot product**, this section prints the **user's input** and the **matching laptop's overview** from the dataset.

```
User Requested:  
I want a laptop HP brand with notebook 15.2inches with 256gb  
Laptop that match:  
Samsung Notebook 9 15.0 Inches 256GB Windows Intel
```

Figure 21: User input and best matching laptop from dataset

This step confirms the recommendation process as it shows the system is working correctly, it improves transparency so that users can verify that it meets their needs, and it enhances user experience by showing a well-structured laptop description.

3.2.6. STRING-BASED SIMILARITY (COSINE SIMILARITY)

3.2.6.1. Cleaning and preparing for Cosine Similarity

A cosine similarity is used to measure how similar two text-based representations are. It is widely used in text-based recommendations to find similarities in user preferences. It applies mathematics by using texts as vectors in a high-dimensional space, calculating the cosine of the angle between 2 vectors and its bases its answer on how close the angle is to zero degrees. The closer an angle is to zero degrees, the higher the similarity of the two pieces of texts.

The following categories were chosen for cosine similarity.

- Laptop Type (TypeName)
- Operating System (OpSys_Name)
- Display Type (Display_Type)
- CPU Brand (Cpu_Brand)
- GPU Brand (Gpu_Brand)
- GPU Tier (Gpu_Tier)

To apply the cosine similarity in our data, the first step involved standardizing of text data by converting all text to lower case and removing spaces to maintain uniformity. Second step involved combining all relevant text-based attributes into a single column “metadata_text” to allow efficient similarity computations.

	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	...	Screen_Width	Screen_Height	Cpu_Brand	Cpu_Speed	Total_Memory_GB	Gpu_Brand	Gpu_Tier	OpSys_Name	Overview	metadata_text
0	Apple	MacBook Pro	ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8.0	128GB SSD	Intel Iris Plus Graphics 640	macOS	...	2560.0	1600.0	intel	2.3	128.0	intel	integrated	mac	Apple MacBook Pro 13.3 Inches 128GB Mac Intel	ultrabook mac (pspanelretinadisplay intel inte...
1	Apple	Macbook Air	ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8.0	128GB Flash Storage	Intel HD Graphics 6000	macOS	...	1440.0	900.0	intel	1.8	128.0	intel	integrated	mac	Apple Macbook Air 13.3 Inches 128GB Mac Intel	ultrabook mac others intel intel integrated
2	HP	250 G6	notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8.0	256GB SSD	Intel HD Graphics 620	No OS	...	1920.0	1080.0	intel	2.5	256.0	intel	integrated	other	HP 250 G6 15.6 Inches 256GB Other Intel	notebook other fullhd intel intel integrated
3	Apple	MacBook Pro	ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16.0	512GB SSD	AMD Radeon Pro 455	macOS	...	2880.0	1800.0	intel	2.7	512.0	amd	amdgpu	mac	Apple MacBook Pro 15.4 Inches 512GB Mac Intel	ultrabook mac (pspanelretinadisplay intel amd ...
4	Apple	MacBook Pro	ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8.0	256GB SSD	Intel Iris Plus Graphics 650	macOS	...	2560.0	1600.0	intel	3.1	256.0	intel	integrated	mac	Apple MacBook Pro 13.3 Inches 256GB Mac Intel	ultrabook mac (pspanelretinadisplay intel inte...

Figure 22: An extra column added into the dataset, Metadata text

3.2.6.2. Computing Cosine Similarity using CountVectorizer

CountVectorizer transforms text data into numerical data and computes the cosine similarity between the laptops. It does this by converting the metadata_text into a matrix of word counts whereby each row represents a laptop and each column represents a unique word in the dataset and encodes the laptop metadata numerically to make it ready for checking similarity.

After transforming the text into a count-based matrix, a cosine similarity is computed to measure how closely the laptops match each other. This outputs a similarity score between 0 and 1 with 1 being identical and 0 being no similarity.

3.2.6.3. Retrieving Similar Laptop Recommendations

After doing the calculations, a verification of the cosine similarity matrix (cosine_sim2) was done to confirm the similarity calculations were applied correctly. The matrix has 1303 laptops, and the matrix contains similarity scores for every laptop pair, this affirms a successfully computed cosine similarity.

(1303, 1303)

Figure 23: Similarity score matrix size

After doing the cosine similarity matrix, we got the list of the recommended laptop indices using get_recommendations () function. This provided the user with relevant alternatives based on their similarity scores. This allows users to view and compare similar laptops based on the specifications such as Brand, processor, storage, and Operating system.

[793, 226, 13, 16, 27, 48, 55, 60, 72, 100]

Overview

793	Lenovo Yoga 510-15IKB 15.6 Inches 256GB Window...
226	Dell Inspiron 5567 15.6 Inches 1024GB Windows ...
13	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
16	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
27	Dell Inspiron 5570 15.6 Inches 256GB Windows I...
48	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
55	Dell Inspiron 3576 15.6 Inches 256GB Windows I...
60	Dell Inspiron 5770 17.3 Inches 2304GB Windows ...
72	Dell Inspiron 5570 15.6 Inches 256GB Windows I...
100	HP 15-bs017nv (i7-7500U/8GB/256GB/Radeon 15.6 ...

dtype: object

Figure 24: List of top 10 similar laptops using string-based similarity

3.2.7. NUMERICAL SIMILARITY (EUCLIDEAN DISTANCE)

3.2.7.1. Preparing Numerical Features for Euclidean Distance Calculation

This is a metric used in recommendation systems to determine the similarity between numerical data points. In Euclidean distance, a smaller distance means the laptops are more similar while a larger distance indicates less similarity. Euclidean distance was used to compare laptops based on their numerical specifications. By calculating Euclidean distance between laptops, we can identify the models with the closed numerical features to a user's preferred specifications. The following features were selected for the comparison:

- Screen Size (Inches)
- RAM (Ram)
- Screen Resolution (Screen_Width and Screen_Height)
- Processor Speed (Cpu_Speed)
- Total Storage (Total_Memory_GB)

The features were also converted to float format to ensure consistency in the computations. This was done using a function that standardizes the values and prevents errors. The data was now in a format where the system could now measure the closeness between different laptops based on their specifications in a reliable and precise manner.

3.2.7.2. Computing Numerical Similarity using Euclidean Distance

We applied Euclidean distance to normalized numerical variables like RAM, CPU speed, screen size, and storage to compare laptops according to their technical specs.

To provide fair comparisons across various feature ranges, the data was first scaled using

MinMaxScaler. After calculating the Euclidean distance between each laptop, the results were transformed into similarity scores using:

$$\text{Similarity} = 1 / (1 + \text{Euclidean Distance})$$

3.2.7.3. Retrieving Recommended Laptops using Euclidean Distance

Checking of the matrix to ensure the Euclidean distance calculations were applied correctly. Like the cosine similarity matrix, the Euclidean distance matrix has 1303 laptops, and the matrix contains similarity scores for every laptop pair, this affirms a successfully computed cosine similarity.



Figure 25: Similarity score matrix size

To recommend laptops with similar hardware specs, the system retrieves the **top 10 most similar laptops** based on features like **RAM, CPU speed, screen size, and storage**.

The recommended laptops are identified using the `get_recommendations ()` function, which selects models with the **smallest Euclidean distance** to the target laptop. The corresponding laptop descriptions are then displayed for comparison, allowing users to explore alternatives with similar performance characteristics.

The output below shows the laptops recommended:

[874, 235, 451, 770, 163, 583, 736, 923, 679, 1042]	
Overview	
874	Samsung Notebook 9 15.0 Inches 256GB Windows I...
235	Dell Inspiron 5567 15.6 Inches 256GB Linux Intel
451	Dell Precision 3520 15.6 Inches 256GB Windows ...
770	Dell Inspiron 5567 15.6 Inches 256GB Windows I...
163	Lenovo Legion Y520-15IKBN 15.6 Inches 256GB Wi...
583	Dell Latitude 5580 15.6 Inches 256GB Windows I...
736	HP ZBook 15 15.6 Inches 256GB Windows Intel
923	Toshiba Tecra Z50-C-140 15.6 Inches 256GB Wind...
679	Acer Aspire VX5-591G 15.6 Inches 256GB Windows...
1042	Toshiba Tecra A50-C-218 15.6 Inches 256GB Wind...

Figure 26: List of top 10 similar laptops using numerical similarity

3.2.8. COMBINING THE SIMILARITY MATRICES FROM NUMERICAL AND STRING

The previous approaches managed the text-based (cosine similarity) and the numerical (Euclidean distance) independently. To improve the accuracy, we combined both to obtain a

hybrid similarity measure. This ensured a recommendation system that factored in the functional attributes (e.g., CPU brand, OS, GPU type) and the hardware specifications (e.g., RAM, storage, screen size, CPU speed). Using cosine alone would factor in categorical attributes of a laptop only and this completely ignores the performance-based differences. Euclidean distance also ignores the qualitative factors hence combining both would introduce a balanced mix of both technical preferences and user preferences.

3.2.8.1. Optimizing the Text and Numerical Similarity

As there needs to be a method to identify the best way to factor in both cosine similarity and Euclidean distance, we implemented an optimization process to determine the best weight distribution for each.

The steps taken included:

- Initializing the variables that would be used to store the best variables
- Testing different values for alpha ranging from 0 to 1 in small increments. Since both alpha and beta must total to 1, beta was computed dynamically. For each alpha-beta combination, a similarity matrix was computed that weighted cosine and numerical similarity accordingly.
- Measuring how well each pair of alpha-beta performed through overall accuracy by generating multiple laptop recommendations and evaluating how similar they are to the target laptop. The average was then compared against previous best values until the best alpha, beta and accuracy values were obtained and printed for reference as shown below:

Best Alpha: 0.42, Best Beta: 0.5800000000000001, Highest Accuracy: 0.8453

Figure 27: The best combination of weightage for highest model accuracy

After that, it was now time to merge both. The weigh factors (alpha and beta) were used to control the contribution of each whereby the cosine similarity had an alpha of 0.42 and Euclidean distance had 0.58 beta. This assigned 42% weight to a text-based similarity while 58% was assigned to numerical similarity.

Next, a computation similarity score was done by summing the weighted cosine and Euclidean similarity matrices. This step ensured a more accurate recommendation and allowed fine-tuning by adjusting alpha and beta to prioritize different aspects based on user-needs.

3.2.8.2. Retrieving Final Laptop Recommendations Using Combined Similarity

After merging, we used the combined similarity matrix to outline the final list of recommended laptops. Figure 28 presents the user with the best matched laptop based on their input preferences.

```
[793, 770, 1186, 1135, 16, 100, 368, 861, 814, 921]
```

	Overview
793	Lenovo Yoga 510-15IKB 15.6 Inches 256GB Window...
770	Dell Inspiron 5567 15.6 Inches 256GB Windows I...
1186	Dell Inspiron 5578 15.6 Inches 512GB Windows I...
1135	Lenovo IdeaPad 500-15ISK 15.6 Inches 0GB Windo...
16	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
100	HP 15-bs017nv (i7-7500U/8GB/256GB/Radeon 15.6 ...
368	Dell Inspiron 5567 15.6 Inches 256GB Windows I...
861	HP EliteBook 850 15.6 Inches 256GB Windows Intel
814	Dell Vostro 3568 15.6 Inches 256GB Windows Intel
921	HP EliteBook 850 15.6 Inches 256GB Windows Intel

dtype: object

Figure 28: List of top 10 similar laptops using Combined similarity

3.2.8.3. Testing the Accuracy of the Laptop Recommendation System

It is necessary to evaluate the system's accuracy to ensure it is providing relevant laptop recommendations. An accuracy test was developed to measure how well the laptops match the target laptop based on both the categorical and numerical attributes.

To achieve this, 3 accuracy scores were calculated.

- String accuracy for measuring how many categorical features match between the target laptop and the recommended ones
- Numerical accuracy measured the number of numerical features identical between the target laptop and the recommended ones.
- Overall accuracy that combined both numerical and numerical giving the final performance score.

This accuracy test purpose is to validate that the recommended laptops closely match the target laptop. A high accuracy score indicates that the system is making relevant and precise recommendations.

3.2.9. Validating the effectiveness of the Hybrid Similarity Approach (0.42 alpha and 0.58)

To ensure combining both text-based and numerical similarity produces better laptop recommendations, we conducted a comparative analysis by testing 4 extreme cases. These cases include:

Table 2: Different combinations of weightage model

Case 1. Using Text Similarity only where Alpha is 1 and Beta is 0
Case 2: Using only numerical similarity (Alpha is 0 and Beta is 1)
Case 3: Using 0.5 for Alpha and 0.5 for Beta (equal weighted)
Case 4: Using the optimal Alpha of 0.42 and Beta of 0.58 (optimized weight distribution)

Case 1. Using Text Similarity only where Alpha is 1 and Beta is 0.

The recommendation system only relies on categorical attributes such as brand, OS, display type, GPU brand and completely ignores the numerical features. The recommended laptops retrieved and displayed are shown in Figure 29.

[793, 226, 13, 16, 27, 48, 55, 60, 72, 100]

Overview

793	Lenovo Yoga 510-15IKB 15.6 Inches 256GB Window...
226	Dell Inspiron 5567 15.6 Inches 1024GB Windows ...
13	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
16	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
27	Dell Inspiron 5570 15.6 Inches 256GB Windows I...
48	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
55	Dell Inspiron 3576 15.6 Inches 256GB Windows I...
60	Dell Inspiron 5770 17.3 Inches 2304GB Windows ...
72	Dell Inspiron 5570 15.6 Inches 256GB Windows I...
100	HP 15-bs017nv (i7-7500U/8GB/256GB/Radeon 15.6 ...

dtype: object

Figure 29: List of top 10 similar laptops using string-based similarity

Case 2: Using only numerical similarity (Alpha is 0 and Beta is 1)

In this case, only hardware specifications like RAM, screen size, resolution, CPU speed and storage capacity and the categorical attributes were completely ignored. The recommended laptops retrieved and displayed are shown in Figure 30.

[874, 235, 451, 770, 163, 583, 736, 923, 679, 1042]

Overview

874	Samsung Notebook 9 15.0 Inches 256GB Windows I...
235	Dell Inspiron 5567 15.6 Inches 256GB Linux Intel
451	Dell Precision 3520 15.6 Inches 256GB Windows ...
770	Dell Inspiron 5567 15.6 Inches 256GB Windows I...
163	Lenovo Legion Y520-15IKBN 15.6 Inches 256GB Wi...
583	Dell Latitude 5580 15.6 Inches 256GB Windows I...
736	HP ZBook 15 15.6 Inches 256GB Windows Intel
923	Toshiba Tecra Z50-C-140 15.6 Inches 256GB Wind...
679	Acer Aspire VX5-591G 15.6 Inches 256GB Windows...
1042	Toshiba Tecra A50-C-218 15.6 Inches 256GB Wind...

dtype: object

Figure 30: List of top 10 similar laptops using numerical similarity

Case 3: Using 0.5 for Alpha and 0.5 for Beta (equally weighted)

Since it is a common assumption to use equal weights , we introduced a hybrid approach where both text and numerical similarity were given equal weight (0.5 each) to observe its performance. This allows comparison with the optimized approach by determining whether equal weighting is as effective as the optimized one. It also shows that a one-size-fit-all is not always ideal. Here are the top 10 recommendations based on the equal-weight approach.

[793, 770, 1186, 1135, 16, 100, 368, 861, 814, 921]

Overview

793	Lenovo Yoga 510-15IKB 15.6 Inches 256GB Window...
770	Dell Inspiron 5567 15.6 Inches 256GB Windows I...
1186	Dell Inspiron 5578 15.6 Inches 512GB Windows I...
1135	Lenovo IdeaPad 500-15ISK 15.6 Inches 0GB Windo...
16	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
100	HP 15-bs017nv (i7-7500U/8GB/256GB/Radeon 15.6 ...
368	Dell Inspiron 5567 15.6 Inches 256GB Windows I...
861	HP EliteBook 850 15.6 Inches 256GB Windows Intel
814	Dell Vostro 3568 15.6 Inches 256GB Windows Intel
921	HP EliteBook 850 15.6 Inches 256GB Windows Intel

dtype: object

Figure 31: List of top 10 similar laptops using equal weightage model

Case 4: Using the optimal Alpha of 0.42 and Beta of 0.58 (optimized weight distribution)

After testing case 1,2 and 3, we rerun the recommendation system using the optimized hybrid similarity approach using the best performing alpha and beta values that were obtained in the

optimization process. (this is a process already done above). The optimal recommendation displayed the 10 laptop recommendations that considered both text and numerical attributes.

```
[793, 770, 1186, 1135, 16, 100, 368, 861, 814, 921]
```

Overview	
793	Lenovo Yoga 510-15IKB 15.6 Inches 256GB Window...
770	Dell Inspiron 5567 15.6 Inches 256GB Windows I...
1186	Dell Inspiron 5578 15.6 Inches 512GB Windows I...
1135	Lenovo IdeaPad 500-15ISK 15.6 Inches 0GB Windo...
16	Dell Inspiron 3567 15.6 Inches 256GB Windows I...
100	HP 15-bs017nv (i7-7500U/8GB/256GB/Radeon 15.6 ...
368	Dell Inspiron 5567 15.6 Inches 256GB Windows I...
861	HP EliteBook 850 15.6 Inches 256GB Windows Intel
814	Dell Vostro 3568 15.6 Inches 256GB Windows Intel
921	HP EliteBook 850 15.6 Inches 256GB Windows Intel

dtype: object

Figure 32: List of top 10 similar laptops using optimized weightage model

After doing the 4 cases, the last 2 show similar results of the 10-laptop recommendation indicating close similarities between both cases. The next step will involve visualization to identify the differences among the 4 cases visualization.

3.2.10 Challenges faced during implementation

Our dataset contains a wide range of laptop variations, increases the work that needs to be done to standardize and process the attributes effectively. This makes us have extensive feature engineering work to generalize and refine laptop attributes for better model performance.

We encounter the imbalance of datasets which have difficulty getting a well-balanced dataset that has equal distribution of laptop operating systems, as most of our datasets consist of Windows-based laptops. Similar issue with laptop brands, leading to potential bias in recommendations.

The accuracy evaluation implemented by our team relies on an exact match between laptop attributes, which may not fully capture the nuances of similarity. This will result in lowering the model's accuracy.

The Grid Search method used for optimizing the model's weight parameters is computationally expensive and time-consuming.

3.3. Solution Output

3.3.1. Accuracy Comparison Visualization

This is an essential part of evaluating performance. By plotting the accuracy metrics over selected sample ranges in our dataset, we can better understand how the different weighting strategies affect the recommendation accuracy across various segments of the dataset.

The visualization will help identify trends in recommendation across different segments of the dataset, assess how effective different weighing approaches are and validate if the optimized weight distribution is the best in terms of accuracy.

To ensure a diverse evaluation, 3 distinct sampling ranges were picked.

- Range 0-50: the first segment of the dataset.
- Range 600-650: A mid-range segment of laptops.
- Range 1252-1302: A last range segment of laptops.

By systematically analysing different dataset segments and similarity weighting strategies, this approach provides actionable insights that enhance recommendation accuracy and overall system reliability.

The visualization as shown in Figure 33 consists of 3 line plots that compare the accuracy of the 4 cases. Each graph plots the accuracy values of different similarity weighting applied to the recommendation models. The 4 cases include:

Table 3: 4 Cases with different color line in graph

Case 1. Alpha 1, Beta 0	Orange Line
Case 2: Alpha 0, Beta 1	Blue Line
Case 3: Alpha 0.50, Beta 0.50(Equally weighted)	Green Line
Case 4: Alpha 0.42, Beta 0.58 (optimized weight distribution)	Red Line

Key observations made in the graph include:

A higher consistency in accuracy of the red line (Alpha 0.42 and Beta 0.58). This shows a more stable performance across all the 3 plots suggesting that the optimized weight distribution is more reliable and balanced recommendation system.

The accuracy of different laptops fluctuate depending on their features. The optimized weight distribution appears to reduce fluctuations compared to equally weights.

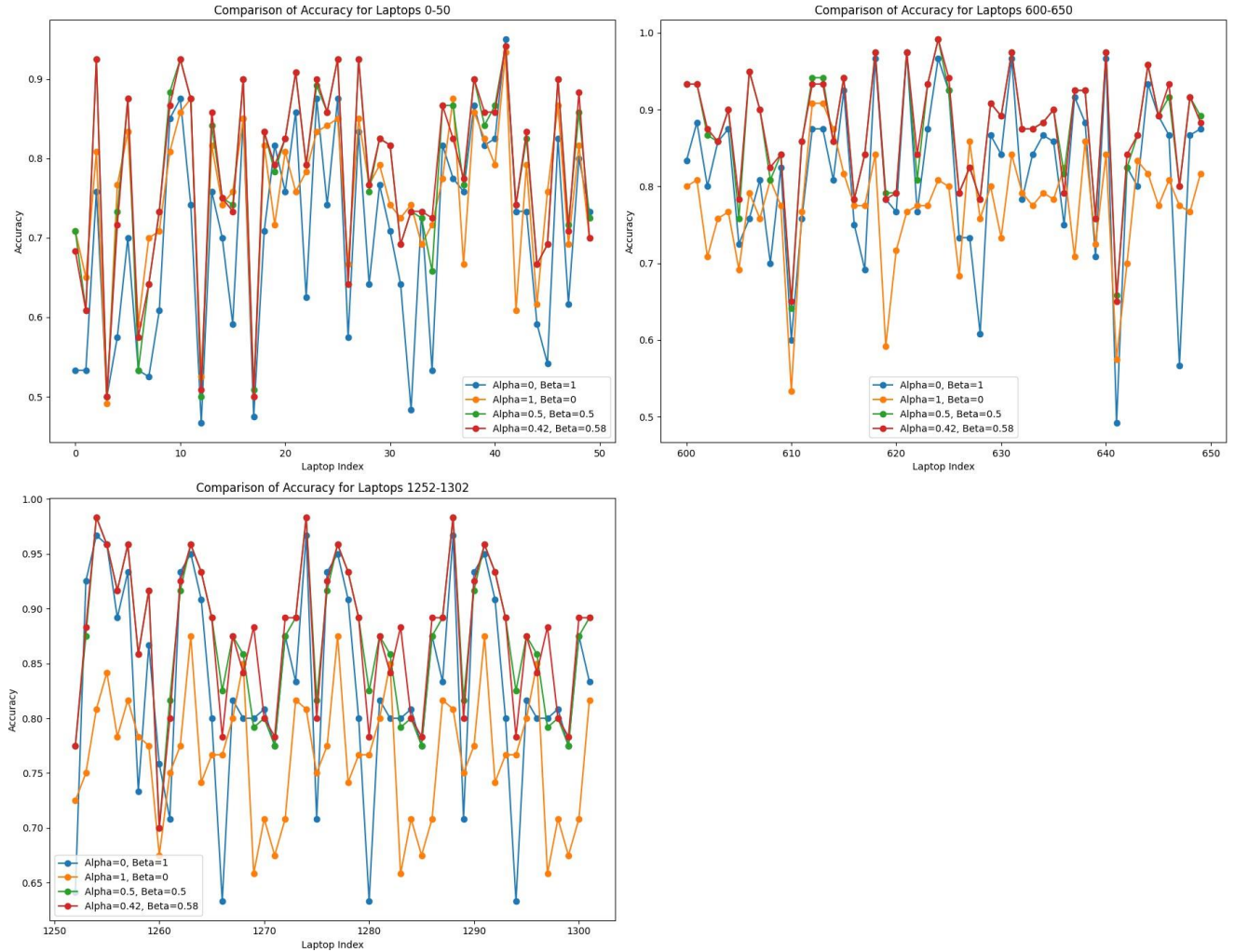


Figure 33: Visualization of 3 line plots that compare the accuracy of the 4 cases

The third plot shows that the redline consistently achieves the highest accuracy and at most times near 01 while green line remains slightly lower. This also suggests that optimal weight combination is more effective in recommending.

In conclusion, the optimized weight distribution (Alpha 0.42 and Beta 0.58) shows higher and more stable accuracy across different laptop indices. It outperforms equal weighting. This shows the importance of finetuning weights in hybrid recommendation models instead of relying on equal contributions from different similarity measures.

3.3.2. Overall Accuracy for Different Alpha and Beta Combinations.

The bar chart shown in Figure 34 visualizes the overall accuracy of the four cases of the different weighting strategies in recommendation system.

The Y-axis represents overall accuracy with values from 0 to 1. The results show that the optimized combination (0.42, 0.58) achieved the highest accuracy (0.8453), slightly outperforming the equal weighting (0.5, 0.5) which had an accuracy of 0.8445.

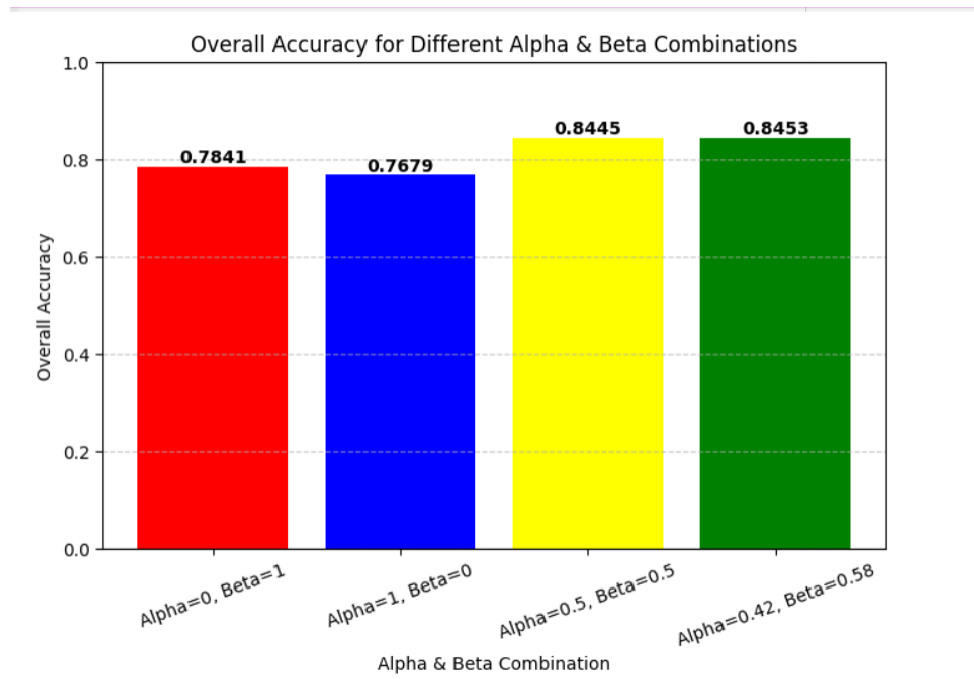


Figure 34: Overall Accuracy for all cases

As the graphs shows, the numerical similarity performed better than cosine. Equal Weighting improved significantly to 0.8445 in comparison to the individual weightings. This confirms that a hybrid approach that is leveraging both similarity measure is beneficial. Optimized Combination provides the highest accuracy of 0.8453 outperforming equal weights. This is a clear indication that fine-tuning weights leads to better recommendation accuracy. In conclusion, optimization yields a marginal but meaningful improvement in accuracy.

3.4. Potential Benefits of The Recommendation System to the Laptop Business

Th results of our analysis provide data driven decision making as the alpha and beta combinations help identify the most effective similarity measure. Increased user satisfaction as our model has an optimized combination that assures the finest results. Through this system, a business gains a competitive advantage as it provides relevant recommendations which will increase sales as customers are more likely to find the right laptop quicker. This system will reduce fatigue for clients which will enhance their shopping experience. Lastly, understanding the recommendation accuracy will help businesses optimize their product offerings by prioritizing the laptops that align with consumer preferences.

4.0 Conclusion

For our laptop dataset, String-Based Similarity using Cosine Similarity for text-based attributes achieved an accuracy of 76.79%, which is less accurate than Numerical Similarity using Euclidean Distance for numerical attributes, which had an accuracy of 78.41%. This indicated that for a laptop dataset, since most of the laptop attributes are containing numerical values, computing similarity using Euclidean Distance yields more accurate results when retrieving the top 10 similar laptops. However, the developed hybrid model, which integrates both similarity measures techniques, shown to have outperformed both standalone techniques. With the optimised weightage combination for the model, the hybrid model achieved the highest accuracy of 84.53%.

The newly developed hybrid model recommendation system by our group has demonstrated improvement in accuracy, increasing from 78.41% up to 84.53%. This improvement will improve the relevance of laptop recommendations to the users, reducing the chance of users receiving irrelevant laptop suggestions in the top 10 recommendation list displayed beneath the laptop they are currently viewing. As a result, user experience has increased, as the AI-recommendation system instantly generates a list of 10 highly relevant laptops, eliminates the need for manual filtering and makes the comparison process smoother and more efficient.

This AI-solution has impacted how the users will be interacting with online shopping platforms during the comparison process. With the AI assistance recommendation system, users can now have easy access and compare similar laptops, which are displayed beneath the laptop they are currently viewing. This enables users to have quicker decision-making, helping users find the most suitable laptop based on their preferences without the need for trial-and-error searches. With the AI-driven recommendation system, it not only simplifies the comparison process but also encourages users to explore alternative similar laptops, increasing the likelihood of purchase.

5.0 References

2. Chunhapran, O., Phromsuthirak, C., Hama, M., & Maliyaem, M. (2022, December). Movie Recommendation System Using Director-Based. In *2022 26th International Computer Science and Engineering Conference (ICSEC)* (pp. 151-155). IEEE.
3. Naveed, S., & Ziegler, J. (2019, September). Feature-Driven Interactive Recommendations and Explanations with Collaborative Filtering Approach. In *ComplexRec@ RecSys* (pp. 10-15).
4. Nguyen, Q., Nguyen, V., Tran, D., Mai, T., & Quan, T. (2019). Star2vec: From Subspace Embedding to Whole-Space Embedding for Intelligent Recommendation System (Extended Abstract). In *Lecture notes in computer science* (pp. 70–71). https://doi.org/10.1007/978-3-030-34980-6_7
5. Sridhar, S., Dhanasekaran, D., & Latha, G. C. P. (2022). Content-Based Movie Recommendation System Using MBO with DBN. *Intelligent Automation & Soft Computing*, 35(3), 3241–3257. <https://doi.org/10.32604/iasc.2023.030361>
6. Morr, C. E., Jammal, M., Ali-Hassan, H., & El-Hallak, W. (2022b). Data Preprocessing. In *International series in management science/operations research/International series in operations research & management science* (pp. 117–163). https://doi.org/10.1007/978-3-031-16990-8_4