INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH IN PROGRESSIVE EDUCATION & DEVELOPMENT

OPEN ACCESS JOURNAL

Vol 12, Issue 3, (2023) E-ISSN: 2226-6348

Coffee Recommender System Using Content-based Filtering

Mohd Rahmat Mohd Noordin, Anis Amilah Shari, Mohamad Hafiz Khairuddin, Nurazian Mior Dahalan, Siti Awadah Tajuddin

School of Computing Sciences, College of Computing, Informatics, and Mathematics, Universiti Teknologi MARA Cawangan Melaka Kampus Jasin, 77300 Merlimau, Melaka, Malaysia

Corresponding Author Email: mrahmat.noordin@uitm.edu.my

To Link this Article: http://dx.doi.org/10.6007/IJARPED/v12-i3/19317 DOI:10.6007/IJARPED/v12-i3/19317

Published Online: 28 September, 2023

Abstract

Coffee is a type of beverage that is brewed and made from roasted seeds, or beans from coffee plants. Despite this, there are many coffee drinkers and non-coffee drinkers who do not know much about coffee knowledge. Besides that, there were many cafes that were being affected by the pandemic which caused a business drop. In addition, there are many new cafe owners who do not know much about coffee beans that are suitable to serve their customers. There is a need for a digital method to assist coffee drinkers and café owners either new or old, to receive recommendation about coffee beans that can serve the preferences of avid coffee enthusiasts or specialty coffee service provider. It is important to develop such a recommender system due to the overwhelming variety of coffee available. The goal of this project is to create a recommender system that informs coffee drinkers about various coffee bean varieties based on their personal preferences in coffee and provide a list of cafes that serve speciality coffee in Selangor focusing on Bandar Baru Bangi, Kajang and Serdang. The coffee and beverages that met the users' taste requirements were categorized and presented using this approach. A content-based filtering method that compares items depending on the user's preferences is used to generate the suggestion. In this project, a modified waterfall model of System Development Life Cycle was used to develop the prototype. This system was tested using functionality testing and accuracy testing and showed positive test results and the algorithm of content-based filtering was applied successfully. For future work, further development can be made to propose cafés instead by narrowing down the availability of preferred coffee beans at specific cafes. The recommendation method can also be fine-tuned by exploring the expert system approach in recommendation.

Keywords: Coffee, Recommender System, Content-Based Filtering

Introduction

This project aims to produce a coffee recommender system that provides information about coffee beans based on the drinker's preference of taste and a list of cafes that serve speciality coffee in selected areas. This system classified and presented the coffee beans that fulfilled the drinkers' taste criteria. Many coffee drinkers out there love to drink coffee but do not

really know what kind of coffee they drink. This could lead to difficulty in finding the same taste of coffee but in different places. To solve the problem, this system is designed to help drinkers and beginner café owners find coffee beans that match their preference of taste and cafes that sell speciality coffee drinks so the user can have the best coffee for the day. As a result, this system makes it easier and more convenient for coffee drinkers to know the beans' names based on their preferred flavour and get recommendations on new coffee beans that have similarities with their favourite beans. Besides that, they can also easily look up to the café in selected areas.

Objectives

- To design a web-based system that recommends coffee beans that match their taste preferences and provide a list of cafes that serve speciality coffee drinks in Bandar Baru Bangi and nearby.
- To develop a web-based coffee recommender system for coffee beans using a content-based filtering algorithm.
- To test the functionality and accuracy testing of the system.

Problem Statement

There are many coffee drinkers and non-coffee drinkers that did not know much about coffee knowledge. Based on a survey that was conducted, with 75 respondents, on average there were 56% of the respondents really did not know about basic knowledge of coffee and beans. One of the questions given in the questionnaire is if they knew about four main types of coffee beans and there were only 21.3% of respondents really knew about the fact while the remaining 42.7% and 36% barely know and do not know about it respectively. One of the famous coffee beans came from Brazil and it has been served in many specialty coffee places. Despite this, many of the drinkers did not know what kind of coffee beans were used in their coffee and the taste profile of their drink. Based on the questionnaire, there were 32% and 56% of respondents do not know about Brazil beans and their taste profile where these beans give a classic, mellow, and smooth texture also is taste slightly sweet and have a hint of chocolate. Therefore, this survey has shown that many coffee drinkers did not know what was used in their coffee. Furthermore, they had a difficult time finding the right coffee shops that provided the same coffee likely to match their preferred taste.

Furthermore, the coffee industry is a trend nowadays and a leading sector among other food and beverage markets in Malaysia. A perfect example of a food and beverage brand that has an accredited business operation is Starbucks. At the beginning of 2020, the Covid-19 crisis has had an impact on Starbucks financially and economically. Based on an article by Arnold (2020), even Malaysian Starbucks operator, Berjaya Food Berhad reported a loss of as much as $1.82 million for the latest financial year due to Movement Control Orders (MCO) during the Covid-19 crisis. Therefore, it could also have an impact on the company's strategies and operation where they were not able to operate in its traditional way. Be mindful of the fact, that this could also affect Small and Medium Enterprise (SME) that also sells coffee. If this pandemic has a big impact towards big companies such as Starbucks, those small coffee businesses also might affected and lose their business. Consequently, it causes businesses to drop, and they must set a new strategy to gain back their customers and find new ways of marketing to attract more people.

Finally, in the coffee sector, there would be challenges in having the right coffee bean provider. Best drinks must be served to the customers, and speciality coffee is distinguished from industrial coffee through its high quality, restricted supply, freshness, special flavours, package, or consumption atmosphere (Bacon (2005), Daviron and Ponte (2005), Kusmulyono, et al. (2023)). The café owner must know the best coffee bean supplier that serves the best quality and good price to maintain the quality of their drinks. Edelmann, et al. (2020) found that direct trading started to improve coffee quality and producers' income and establish regular communication. Furthermore, the benefits of an increased number of consumers and their awareness, resulting in a willingness to pay a higher price, go to the roasters or coffee shop owners than to a producer.

Methodology

Waterfall methodology employed a linear or sequential approach in developing software. The project is divided into a series of tasks, with phases designating the highest-level grouping (Sherman, 2015). Sequence phases were required in the waterfall technique which includes planning the project requirements, designing the system design, coding implementation, project testing and maintaining the system. The traditional waterfall model and the modified waterfall model are two types of waterfall models. The Waterfall approach was established in 1970 by Winston W. Royce (Prashant, 2022). However, the waterfall model has its drawbacks which is not suitable for unclear and changing requirements. Furthermore, in the waterfall model, once an application is in the testing phase, it is very difficult to go back and change something that was not correct (Salve et al., 2018). To overcome these problems, the modified waterfall model was created (Prashant, 2022). Hence, the modified waterfall model is suitable for this project. The modified waterfall model allows a return to the previous step overlapping as necessary. Furthermore, the modified waterfall model offers a systematic progression of development procedures with some flexible iterative phases to permit enough documentation and design reviews to assure the quality, dependability, and maintainability of the created custom software development. Figure 1 shows the phases in the modified waterfall model of the System Development Life Cycle (SDLC) that are involved in this project.

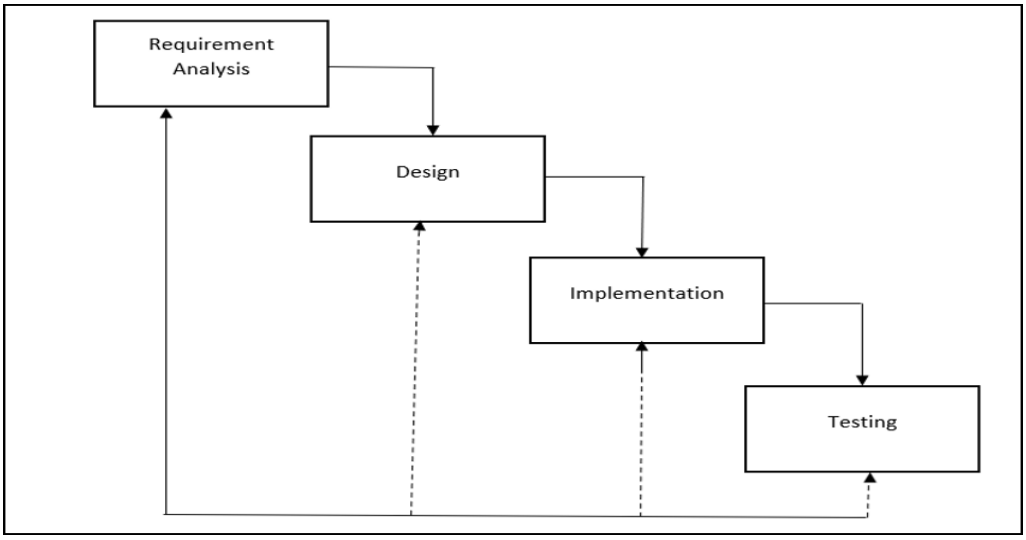


Figure 1 System Development Life Cycle (SDLC) of phases in Modified Waterfall Model

Design and Development

In this section, the coffee recommender systems' design is shown through its system flowchart design, use case diagram design, and system algorithm design which incorporates content-based filtering. As for the development, this section shows the data collection for the system, the development of the recommendation functionalities, and the web development:

a) System Flowchart Design

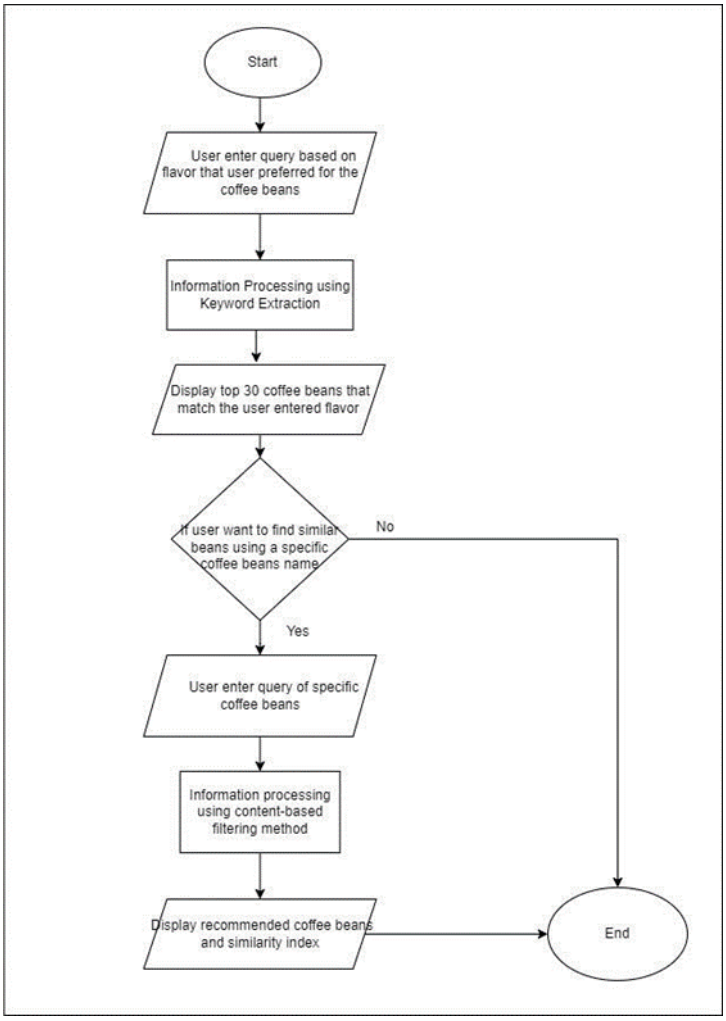


Figure 2 Flowchart of System

Figure 2 shows the flow chart of the system. Firstly, the page asked for the user's choice of coffee preference based on the taste of the coffee. The information from the user's query will then be filtered using the keyword extraction method. The recommended coffee beans based on keyword extraction in the review column were then displayed throughout the page. Furthermore, the option recommender based on similarities appeared on the result page. The recommender page asked for the user's query but in the context of the coffee name. The information from the user's query was then filtered using a content-based filtering technique. The user's query needed to not be empty to proceed with the process.

b) Use Case Diagram Design

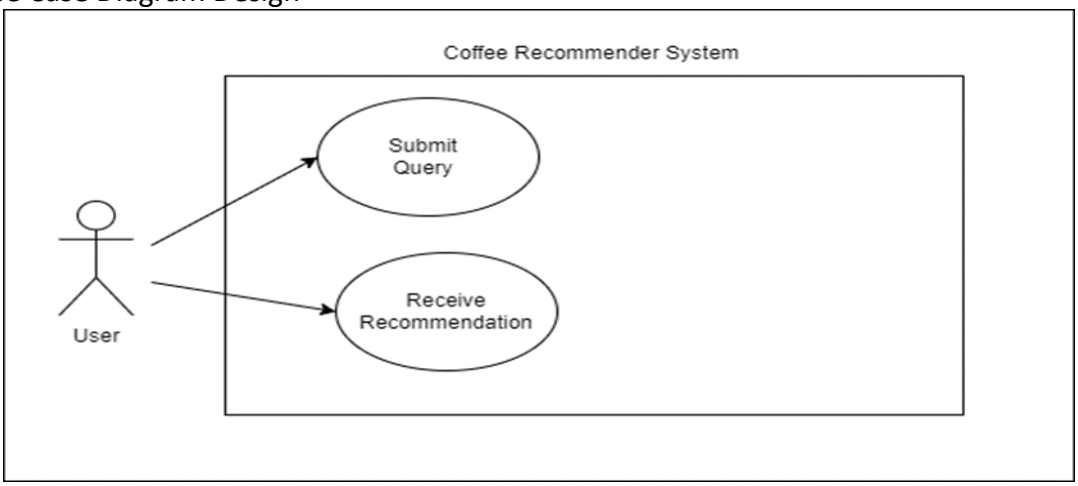


Figure 3 Use Case Diagram of the System

To comprehend the complexity of the data communicated between the system and the user, a use case diagram is next produced. The use case served as a representation of the topic boundary separating the system from the user. Figure 3 displays the use case diagram for this system as the interaction between the user and the system is needed to complete the task. Based on Figure 3 the user could give a query based on their coffee preference to the system and would receive the recommended coffee beans and café.

c) System Algorithm Design

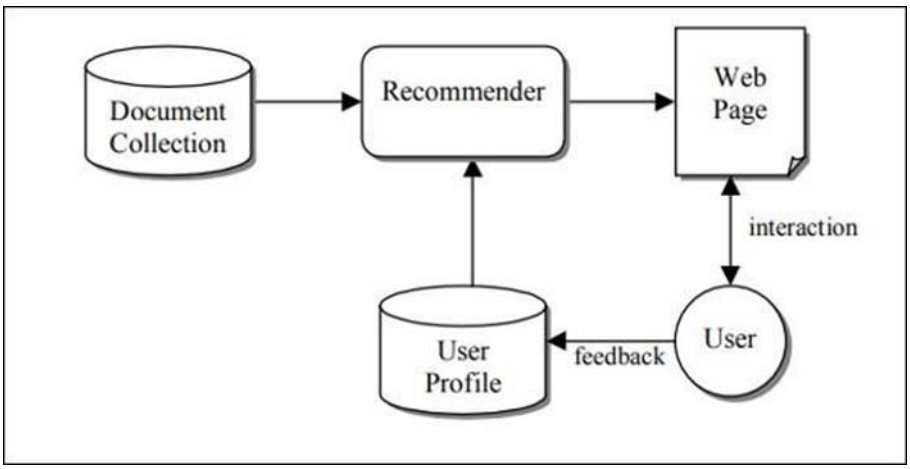


Figure 4 Content-Based Filtering Design

The recommendation system used a content-based method to develop the project. Utilizing keywords and attributes attached to the coffee flavour profile database, content-based filtering generates recommendations. The content-based method filters the user's preferred flavour in coffee based on the description that the user entered. This approach is wholly dependent on matching user preferences to product attributes. Furthermore, the recommended coffee beans are those that share the most characteristics with the user's interests such as the smoothness of the coffee or the bitterness level in the coffee. Figure 4 shows the content-based filtering architecture. The collected data was analyzed by the recommender, and the content-based filtering method was applied. Furthermore, the data

was then evaluated based on various factors, including similarity, uniqueness, closeness, and relevance. The top-ranked pages were shown as hyperlinks on the current web page, and interaction with the user occurred. The user's feedback is used to create a user profile. Lastly, the user profile and the collection of documents are then compared by the recommender system. To evaluate the content-based filtering method, we use the cosine similarity metric to calculate and identify similarities between products and the user's interests. Cosine similarity measures the cosine of the angle between two vectors in a multidimensional space and ranges from -1 to 1. A higher cosine similarity value indicates more shared traits and a higher degree of similarity between the items.

d) Data Collection and Pre-processing

The data used in this project is collected from the GitHub Website. This dataset consists of 5124 data on coffee review, and it consists of name, rating, roaster, regions, type, location, origin, roast, aroma, acid, body, flavour, aftertaste, and review. Figure 5 below shows the data file imported and displays some of the data.

```
In [4]: df = pd.read_csv('coffeeGithub.csv')
        #use slug as unique index id
        df['slug'] = df['slug'].map(lambda x: x[8:])
        df.set_index('slug', inplace=True)
        #remove duplicate descriptions
        df = df.drop_duplicates('desc_1')
        #convert review_date to datetime column
        df['review_date'] = pd.to_datetime(df['review_date'])
        df.head()
```

Out[4]:

slug	all_text	name	rating	roaster	region_africa_arabia	region_caribbean	region_central_america	region_hawaii	region_asia_pacific
ethiopia-deri-kochoha-2	\n\n\n\n \n93\nFlight Coffee Co.\nEthiopia Der...	Ethiopia Deri Kochoha	93	Flight Coffee Co.	1	0	0	0	0
espresso-14	\n\n\n\n\n91\nDoi Chaang Coffee\nEspresso\nLoc...	Espresso	91	Doi Chaang Coffee	0	0	0	0	1
kenya-ruthaka-peaberry	\n\n\n\n \n95\nTemple Coffee and Tea\nKenya Ru...	Kenya Ruthaka Peaberry	95	Temple Coffee and Tea	1	0	0	0	0
ethiopia-gora-kone-sidamo	\n\n\n\n \n93\nTemple Coffee and Tea\nEthiopia...	Ethiopia Gora Kone Sidamo	93	Temple Coffee and Tea	1	0	0	0	0

Figure 5 Data of Coffee Review

Some operations are carried out at the pre-processing stage to enhance the data quality, such as data cleansing and data transformation. Figure 6 below shows the process of data cleaning for the keyword extraction system. The dataset was cleaned by dropping some of the unused data such as regions, slug, aroma, acid, body, and flavour. Furthermore, all the description columns were combined into one column labelled review.

```
In [22]: columns_to_drop =['all_text', 'slug', 'region_africa_arabia', 'region_caribbean', 'region_central_america', 'region_hawaii', 'reg
         df = df.drop(columns=columns_to_drop)

In [24]: df['review'] = df['desc_1'] + df['desc_2'] + df['desc_3'] + df['desc_4'].astype(str)

In [25]: df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 5124 entries, 0 to 5123
         Data columns (total 10 columns):
          #   Column   Non-Null Count  Dtype
         ---  ------   --------------  -----
          0   name     5124 non-null   object
          1   rating   5124 non-null   object
          2   roaster  5124 non-null   object
          3   origin   4529 non-null   object
          4   roast    4696 non-null   object
          5   desc_1   5124 non-null   object
          6   desc_2   5124 non-null   object
          7   desc_3   971 non-null    object
          8   desc_4   4153 non-null   object
          9   review   971 non-null    object
         dtypes: object(10)
         memory usage: 400.4+ KB

In [27]: columns_to_drop =['desc_1', 'desc_2', 'desc_3', 'desc_4']
         df = df.drop(columns=columns_to_drop)

In [28]: df.head()

Out[28]:
```

	name	rating	roaster	origin	roast	review
0	Ethiopia Deri Kochoha	93	Flight Coffee Co.	West Guji Zone, Oromia Region, southeastern Et...	Medium-Light	Bright, crisp, sweetly tart. Citrus medley, ca...
1	Espresso	91	Doi Chaang Coffee	Northern Thailand	Medium	Evaluated as espresso. Deeply rich, sweetly ro...
2	Kenya Ruthaka Peaberry	95	Temple Coffee and Tea	Nyeri growing region, south-central Kenya	Medium	Deeply sweet, richly savory. Dark chocolate, p...

Figure 6 Drop and Combine Data Process

```
In [7]: #instatiate tokenizer, and stemmer
        tokenizer = RegexpTokenizer('\w+')
        p_stemmer = PorterStemmer()

        #create set of stopwords from sklearn and add more words
        stops = set(stopwords.words('english'))

        #add more stop words, uninformative of the coffee, and taster's names
        more_stops = ['cup', 'cupper', 'cupping','reviewer','one','two','three','four',
                      'co','taster','review', 'panel','panelist','rating', 'others', 'found',
                      'nominating','reader','writes', 'ken', 'kenneth', 'davids', 'ron','walters',
                      'kim','westerman', 'jason','sarley','jen', 'apodaca', 'ted','lingle',
                      'teri','bolla', 'willem','boot','ethan','hill','peggy','sue', 'martin', 'freeman',
                      'andrey', 'akselrod', 'heather', 'perry', 'byron', 'holcomb','andy', 'newbom',
                      'sean', 'kohmescher', 'john','diruocco', 'carolina', 'facciani', 'schulz',
                      'john', 'outler', 'monique', 'tam', 'woodard', 'springstube', 'dowling',
                      'des', 'cabigan', 'christopher', 'losa', 'robert', 'bobbs', 'al', 'welker',
                      'jennifer','stone']
        for w in more_stops:
            stops.add(w)

        #function to clean text
        def to_words(raw_text):
            #remove numbers
            raw_text = re.sub('\d+', '', raw_text)
            #tokenize
            words = tokenizer.tokenize(raw_text.lower())
            #remove stop words and stem
            meaningful_words = [p_stemmer.stem(w) for w in words if not w in stops]

            return (" ".join(meaningful_words))

        # Initialize empty lists to hold the clean texts.
        clean_text = []

        # Append clean texts to list.
        for t in df['text']:
            clean_text.append(to_words(t))

        #add column of clean text
        df['clean_text'] = clean_text
```

Figure 7 Cleaned Text Using Stop Words Function

Figure 7 above shows the process of cleaning unused words in the dataset using the stop-word function. The cleaned dataset was used for the filtering and recommendation model using the content-based filtering method. Figure 8 below shows the data cleaned split into two CSV files. The name, roaster, rating, and review date column were saved into the coffee_id.csv file and aroma, acid or milk, body, flavour, type with milk and clean text were saved into the coffee_cleanupdated.csv file. The coffee_cleanedupdated.csv file was used for

filtering and calculating the cosine similarity while the coffee_id.csv file was used for getting the coffee name as input.

```
In [18]: id_columns = ['name', 'roaster', 'rating','review_date']
         coffee_id = df[id_columns]
         coffee_id.to_csv('coffee_id.csv')

In [19]: roasts_df = pd.get_dummies(df.roast,dummy_na=True,prefix='roast')
         roast_names = roasts_df.columns
         roasts_df.columns = [re.sub('[ -]','_', name.lower()) for name in roast_names]
         roasts_df.head()
```

Out[19]:

slug	roast_dark	roast_light	roast_medium	roast_medium_dark	roast_medium_light	roast_very_dark	roast_nan
ethiopia-deri-kochoha-2	0	0	0	0	1	0	0
espresso-14	0	0	1	0	0	0	0
kenya-ruthaka-peaberry	0	0	1	0	0	0	0
ethiopia-gora-kone-sidamo	0	0	0	0	1	0	0
specialty-coffee-blend-espresso	0	0	0	0	1	0	0

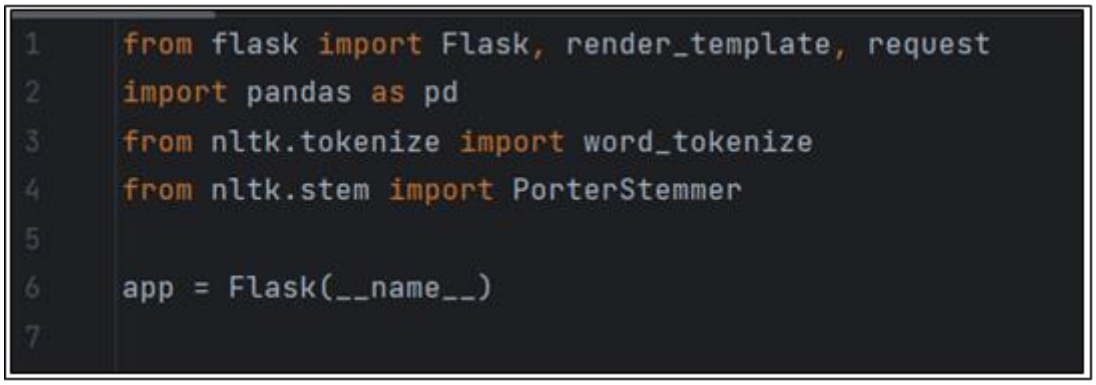
```
In [20]: df = pd.concat([df, roasts_df], axis = 1)

In [21]: regions = ['region_africa_arabia', 'region_caribbean', 'region_central_america',
                    'region_hawaii', 'region_asia_pacific', 'region_south_america']
         types = ['type_espresso', 'type_organic', 'type_fair_trade',
                  'type_decaffeinated', 'type_pod_capsule', 'type_blend', 'type_estate','type_with_milk']

In [23]: df[types].sum(axis=1).value_counts()

Out[23]: 0    2851
         1    1074
         2     688
         3     241
         4      30
         5       3
         dtype: int64

In [24]: clean_columns = ['aroma', 'acid_or_milk', 'body', 'flavor', 'type_with_milk', 'clean_text']
         clean_columns.extend(list(roasts_df.columns) + regions + types)
         coffee_clean = df[clean_columns]
         coffee_clean.to_csv('coffee_cleanupdated.csv')
```

Figure 8 Data Split

e) Recommendation Algorithm Application Development

Recommender systems, a form of machine learning algorithms, are widely used to provide relevant recommendations to users. Whether in apps or search engines, these systems employ a class of algorithms to determine appropriate suggestions for users based on their preferences and interests. Content-based filtering is a popular method within recommender systems, which focuses on the characteristics or content of items that users enjoy. By utilizing user-provided data and available information, the system creates tailored recommendations for individuals. The underlying concept of content-based filtering involves categorizing products using specific keywords (keyword extraction), identifying user preferences, finding products with high similarities (via TF-IDF and Cosine Similarities), and presenting them as recommendations to the user.

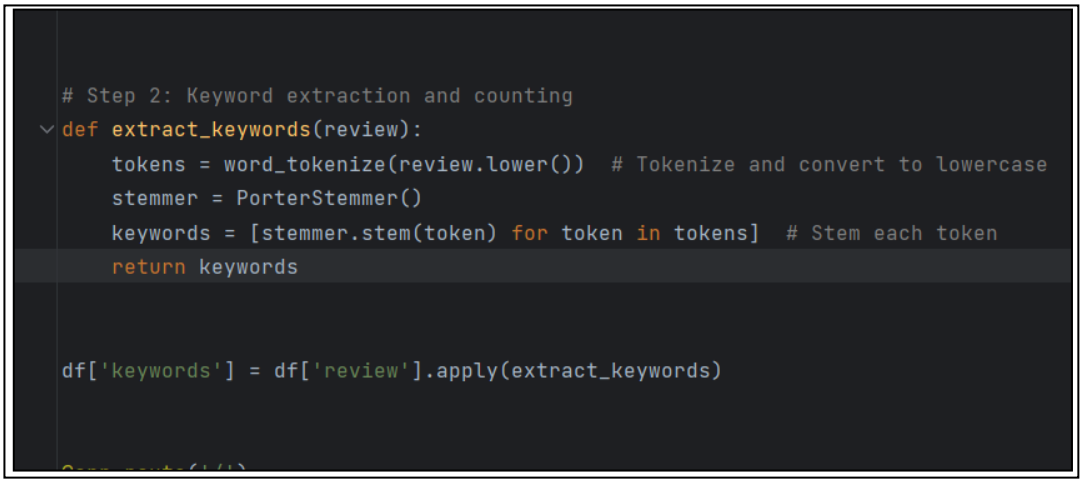
Keyword extraction is a natural language processing (NLP) technique used to identify and extract important words or phrases from a text. These important words, known as keywords, are typically the most relevant and informative terms that represent the main themes or topics in the given text.



```
from flask import Flask, render_template, request
import pandas as pd
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

app = Flask(__name__)

```

Figure 9 Import Libraries on Keyword Extraction

Figure 9 shows the import libraries before starting the code of the algorithm. Here, the tokenizer was used to divide the text into individual units called tokens. The tokens can be words, punctuation marks, or other meaningful elements. Furthermore, the PorterStemmer was also used for the stemming process which reduces words to their base or root form.



```
# Step 2: Keyword extraction and counting
def extract_keywords(review):
    tokens = word_tokenize(review.lower())  # Tokenize and convert to lowercase
    stemmer = PorterStemmer()
    keywords = [stemmer.stem(token) for token in tokens]  # Stem each token
    return keywords


df['keywords'] = df['review'].apply(extract_keywords)
```

Figure 10 Keyword Extraction Function

Figure 10 above shows the process of a function called extract_keywords that performs keyword extraction and counting for coffee reviews stored in the Data Frame. The function takes a review as input, tokenizes the text into individual words (tokens), and converts them to lowercase to ensure consistency. Next, it utilizes the PorterStemmer from the NLTK library to perform stemming on each token. Stemming reduces words to their base form, which helps to group related words together and simplify analysis. The extracted keywords are then stored in a new column called keywords. This process enables the identification of important words in the coffee review column.

The two matrices, Term Frequency (TF) and Inverse Document Frequency (IDF) are closely linked and are valuable in assessing the relevance of a word to a document within a larger corpus of text. The TF represents how often a specific word appears in each document, indicating its prevalence and importance in that context. It measures the ratio of the word's occurrence rate to other terms in the same document. The higher the TF value, the more significant the word is in understanding the document's content. On the other hand, IDF evaluates the rarity of a word across the entire corpus. Together, TF-IDF provides a way to

identify words that are crucial for comprehending the document's content within the broader context of the entire dataset.

```
import pandas as pd
from scipy import sparse
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD, PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import pairwise_distances
import numpy as np
from flask import Flask, request, render_template

app = Flask(__name__)
```

Figure 11 Import Libraries on TF-IDF and Cosine Similarities

Figure 11 above shows the import libraries function before starting to code the algorithm. Here, raw documents are converted to a matrix of TF-IDF characteristics using TfidfVectorizer. The pandas, TruncatedSVD and PCA, StandardScaler, Pairwise Distance and TfidfVectorizer libraries have been imported. After importing the libraries, the CSV data was applied as the data set. To check if the data has been successfully read or not, the data.head() function is applied to display the content in the data sets.

```python
def features_rec(coffees, slugs):
    ss = StandardScaler()
    ss_fitted = ss.fit_transform(coffees)

    pca = PCA(n_components=5)
    pca_fitted = pca.fit_transform(ss_fitted)

    # Calculate cosine similarities and create dataframe of all similarities
    features_recommender = pairwise_distances(pca_fitted, metric='cosine')
    features_recommender_df = pd.DataFrame(features_recommender, index=slugs, columns=slugs)

    # Return dataframe of similarities and the scaled dataframe.
    return features_recommender_df, pca_fitted

def text_rec(coffees, slugs):
    tfidf = TfidfVectorizer(min_df=2, ngram_range=(2, 4), max_features=10000)
    tfidf_fitted = tfidf.fit_transform(coffees)

    # TruncatedSVD transformation, number of components
    tsvd = TruncatedSVD(n_components=150, random_state=36)
    tsvd_fitted = tsvd.fit_transform(tfidf_fitted)

    # Calculate cosine similarities and create dataframe of all similarities
    text_recommender = pairwise_distances(tsvd_fitted, metric='cosine')
    text_recommender_df = pd.DataFrame(text_recommender, index=slugs, columns=slugs)

    return text_recommender_df, tsvd_fitted
```

Figure 12 Applying TF-IDF and Cosine Similarity

Figure 12 illustrates the TF-IDF matrix and cosine similarity were employed to recommend items based on user interests. The TF-IDF function was applied in the text_rec function and cosine similarities were applied to both functions. The matrix represents the similarity between different coffees based on their extracted features and review content. Both functions return their respective cosine similarity matrices and enable content-based recommendations of similar coffees based on their features or review content.

f) Web Development

Website development encompasses all aspects of creating a website, including markup, coding, scripting, and network settings. In this project, the Flask framework is utilized, which is a compact and lightweight Python web framework. Flask offers helpful tools and capabilities, making it easier for developers to build online applications. It provides flexibility and simplicity by enabling the creation of web applications with just one Python file, without the need for a specific directory structure or lengthy boilerplate code. User interface design is the process of creating an interface with the intention of enhancing usability and user experience. User interface design tries to make user interaction as efficient and simple as feasible to help users accomplish their goals. A user interface is necessary for user interaction in any program. Because of the application's user-friendly interface and good design, the user may utilize it with ease. Figure 13 displays the interface layout for the Main Menu Page. On top of the screen, there are four navigation buttons. The user can click on the About button to read a little information on Arabica coffee beans such as the origin and the description of

taste of it in general. In addition, the user was directed to the recommender page upon clicking the "Let's Get Started" hyperlink from the home page.

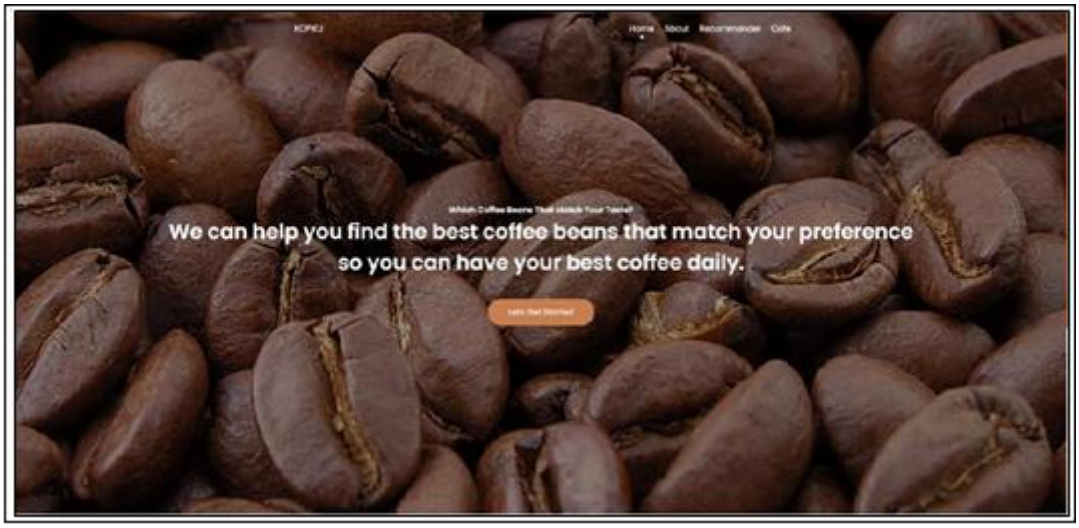


Figure 13 Main Menu Page

The About page is shown in Figure 14. The user can read the information on the website and Arabica coffee beans on this page. The user was directed to the Main Menu page again upon clicking on the home button at the top of the page. Furthermore, the user was immediately directed to the Recommender Page upon clicking on the Recommender Button.

Figure 14 About Page

Figure 15 Coffee Recommender Based on Flavor Page

Figure 15 shows the coffee recommender input page. The interface lets the userfill in the flavour that the user prefers. For example, the user likes coffee with honey, pear, tangerine zest, dark chocolate, and pistachio taste in their coffee beans. The user will be directed to the Results Page upon clicking on the Search button.

Figure 16 Search Results Page

Figure 16 shows the results of user input keywords. On this page, there were thirty results on the list of recommended coffee based on the highest keyword counts in the review column. The information on each coffee includes the coffee name, roast level, roaster, rating, and review. The keywords entered by the user will appear on the top left and the user can easily enter different flavors and click on the search button without having to go back to the recommender page earlier. The name of each coffee can be copied by the user by clicking on the coffee name itself. Furthermore, the user will be directed to a coffee recommender based on similarities for specified coffee beans upon clicking the “Get recommendation for your selected coffee based on similarity” hyperlink from the result page.

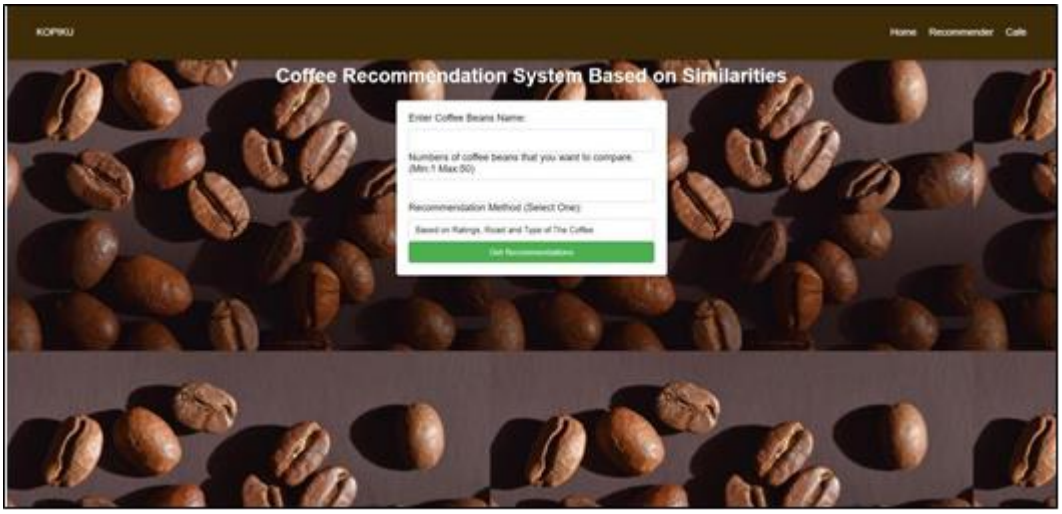


Figure 17

Figure 17 shows the recommendation page that requires users to enter the coffee name and other queries such as the number of nearest coffees so they can get the top recommendation from the entered number and recommendation method. There are three recommendation methods which are based on ratings, roast, type, description and review and combinations of both. The combination method was based on overall features and text. Furthermore, the user will be directed to the recommendation result page upon clicking on the "Get Recommendation" hyperlink on the page.

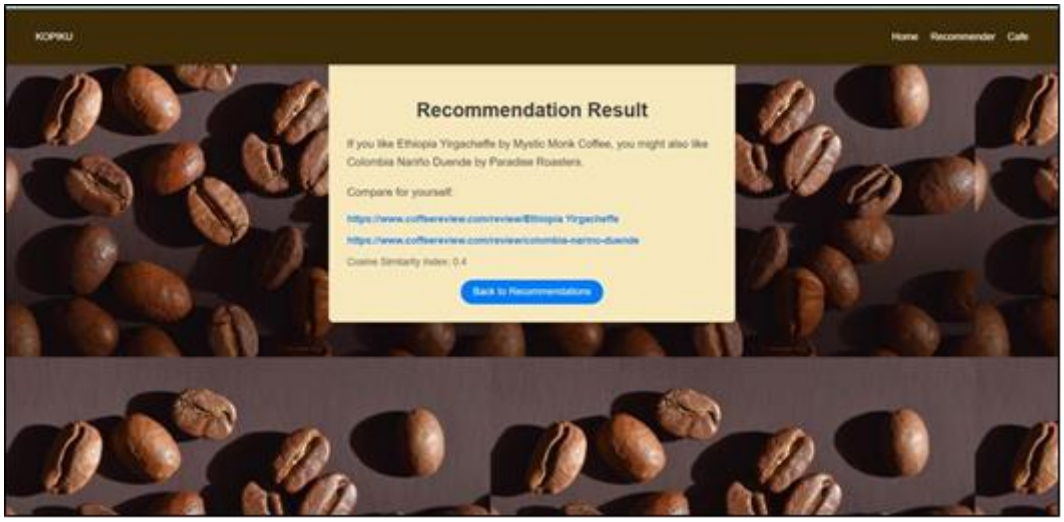


Figure 18 Recommendation Results Page

Figure 18 shows the recommendation result page which provides you with new coffee beans that are most similar within the number of nearest coffees selected before. The user will be directed to the review of coffee page upon clicking on one of each link provided to be compared. Besides that, the user can also know the index of similarities between those two coffee beans on this page. Furthermore, the user will also be directed to the recommendation page earlier by clicking on the "Back to Recommendations" hyperlink on the page.

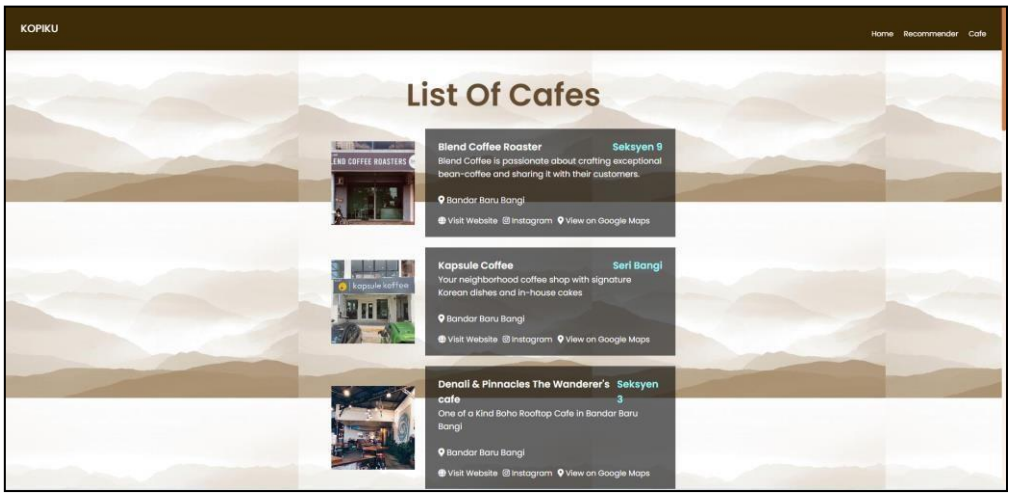


Figure 19 Café List Page

Figure 19 shows the café list page. This page provides a list of cafés that serve speciality coffee in those selected areas which are Bandar Baru Bangi, Kajang and Seri Serdang. The user could go to the website, the social media, and the map location of each cafe by clicking on the button at the bottom of every cafe's details.

```
@app.route('/recommend', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        user_keywords = request.form.get('keywords')
        user_keywords = [keyword.strip() for keyword in user_keywords.split(',')]

        keyword_count = {}
        for coffee_name, review_keywords in zip(df['name'], df['keywords']):
            if coffee_name not in keyword_count:
                keyword_count[coffee_name] = 0
            keyword_count[coffee_name] += sum(keyword in user_keywords for keyword in review_keywords)

        sorted_coffee_names = sorted(keyword_count, key=keyword_count.get, reverse=True)
        top_30_coffee_names = sorted_coffee_names[:30]

        output = df[df['name'].isin(top_30_coffee_names)][['name', 'roast', 'roaster', 'rating', 'review']]
        output['keyword_count'] = output['name'].map(keyword_count)
        output = output.drop_duplicates().sort_values('keyword_count', ascending=False)

        coffee_list = output.values.tolist()

        return render_template('search.html', coffee_list=coffee_list, user_keywords_=_request.form.get('keywords')_)

    return render_template('index.html')
```
Figure 20 Pass Data from Coffee Recommender Based on Flavor Page

The system can connect the model to external data sources such as text files, databases, spreadsheets, and external programs using interface functions. Figure 20 shows the recommendation based on keyword extraction function. This function will retrieve the user keyword input which is the flavour of coffee. The system will then process the keyword and count the keyword. The top thirty coffee lists will then be stored in coffee_list and will be displayed on the search result page called search.html

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        test_name = request.form['test_name']
        pick_best = True if request.form['pick_best'] == 'yes' else False
        n_nearest = int(request.form['n_nearest'])
        how = request.form['how']

        input_slug = name_df.loc[name_df['name'] == test_name, 'slug'].iloc[0]
        input_roaster = name_df[name_df['name'] == test_name]['roaster'].iloc[0]
        slugs = name_df['slug']
        recommender = None

        if how == 'features':
            recommender = features_rec(coffee_df[features], slugs)[0]
        elif how == 'text':
            recommender = text_rec(coffee_df['clean_text'], slugs)[0]
        elif how == 'combination':
            joined = np.concatenate((features_rec(coffee_df[features], slugs)[1],
                                     text_rec(coffee_df['clean_text'], slugs)[1]), axis=1)
            distances = pairwise_distances(joined, metric='cosine')
            recommender = pd.DataFrame(distances, index=slugs, columns=slugs)
        else:
            return render_template('error.html', message="Sorry, that is not a valid recommendation method")

        sims = name_df.join(recommender[input_slug], how='outer', on='slug')
        sorted_sims = sims.drop(sims[sims['slug'] == input_slug].index).sort_values(by=input_slug)

        if pick_best:
```

Figure 21 Pass Data from Coffee Recommender based on Coffee Name Page

Figure 21 shows the recommendation based on the coffee name page. This function will retrieve the user's preferences from the submitted form, such as the coffee name, recommendation method, number of recommendations, and whether to pick the best ones. The most similar coffee will then be displayed on the recommended result page on the results.html page.

Results and Discussions

The coffee recommender system is being tested in three types of testing:

a) System Testing

System testing is a crucial step in software development that verifies the fully integrated and finished software system, ensuring it meets all requirements. The testing process involves exercising the entire computer-based system, as the software is just a part of a larger system that interfaces with other software and hardware components. Functional testing, a type of software testing, evaluates the accuracy of the system's functional requirements or use cases by testing user input, configuration settings, and system data. Accuracy is measured by correctly distinguishing between patients and healthy instances and assessing true positives and true negatives in all analyzed cases. The test cases are aligned with the use-case requirements, and the testing environment and necessary tools are prepared for evaluating each unit of the application or prototype. All entries are ready for testing.

b) Functionality Testing

Functionality testing is a method of testing a web application system. To accomplish the goals of this project, this phase is crucial. The system's functionality has been checked to make sure it functions correctly. The results are expected results of each functionality test case are fulfilled.

c) Accuracy Testing

In this project, the accuracy of the recommendation system is evaluated using similarity scores, with cosine similarity being the chosen metric. Cosine similarity is a mathematical concept used to measure the similarity between two vectors in an inner product space. It determines the cosine of the angle between the two vectors, providing a measure of how closely they align or point in the same direction.

By employing cosine similarity, the recommendation system can effectively assess the likeness between different items, such as coffee products in this case, based on their respective feature representations. The similarity scores generated by the cosine similarity metric enable the system to identify items that share common characteristics or attributes, helping to recommend products that align with the user's preferences or interests.

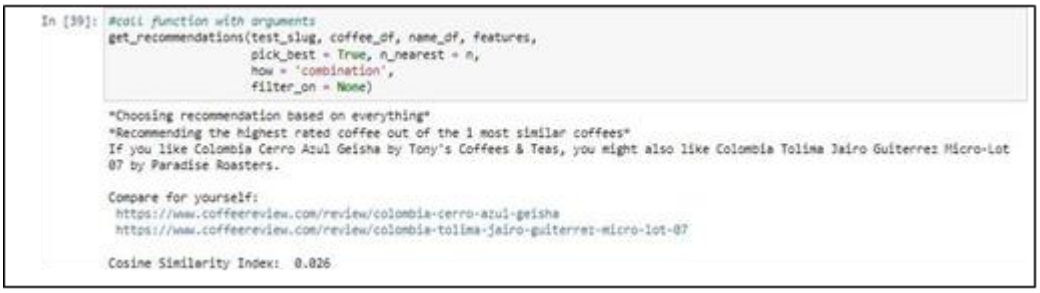


Figure 22 Cosine Similarity Index for Colombia Tolima Jairo Beans

Figure 22 shows the variable similarity representing the cosine similarity scores between the coffee name entered by the user and the recommended coffee beans by the system. The similarity was corresponding to the coffee name and their roast, rating, typing, and reviewing. Higher similarity values Higher similarity values indicate users with more similar preferences while lower values represent different features with the coffee name entered.

Conclusion

In conclusion, the primary objective of this project is to create a coffee bean recommendation system. The prototype has the capability to suggest a list of coffee beans based on the user's flavour preferences and recommend new coffee beans that share similarities with the coffee name provided by the user. The filtering method involves keyword extraction, calculating cosine similarity and applying a content-based algorithm to filter the dataset effectively. The model accurately recommends coffee beans using the attributes entered by the user. This project is particularly valuable for users seeking brand-new coffee beans with similar descriptions to their favourite ones or for comparing multiple coffee beans with the same flavour profile. It simplifies the process of finding the most suitable coffee beans for individual preferences. Additionally, this project can serve as a valuable research and development tool within the coffee community. One potential improvement for this project is to enhance the recommendation system by suggesting multiple coffee beans instead of just one that shares the same description as the user-entered coffee name. By expanding the range of recommended coffee beans, the overall performance and usefulness of the model can be enhanced. Integrating a more diverse set of coffee varieties into the datasets allows the algorithm to learn about a broader range of coffee attributes and features. Consequently, the model may become more comprehensive and accurate in recommending different types of coffee beans with similar traits, providing users with a richer and more diverse selection of coffee options. Further development can be made to propose cafés instead by narrowing

down the availability of preferred coffee beans at specific cafes. The recommendation method can also be fine-tuned by exploring the expert system approach in recommendation where the expertise of a coffee expert(s) can be captured and translated it an algorithm, which can be integrated along with the recommendation algorithm.

References

Arnold, M. (2020). Malaysian Starbucks operator posts loss due to Covid crisis. Retrieved from https://insideretail.asia/2020/08/25/malaysian-starbucks-operator-posts-loss-due-to-covid-crisis/

Bacon, C. (2005). Confronting the Coffee Crisis: Can Fair Trade, Organic, and Specialty Coffees Reduce Small-Scale Farmer Vulnerability in Northern Nicaragua? Retrieved from http://www.sciencedirect.com/science/article/pii/S0305-750X(04)00206-2

Daviron, B. & Ponte, S. (2005). The Coffee Paradox. Global Markets, Commodity Trade and the Elusive Promise of Development. London: Zed Books Ltd

Edelmann, H., Quiñones-Ruiz, X. F., & Penker, M. (2020).Analytic framework to determine proximity in relationship coffee models. Sociologia ruralis, 60(2), 458-481.

Kumuslyono, M. S., Rachman, R. A. & Gunawan, V. P. (2023). Five Forces Analysis Of Coffee Business Industry In Temanggung Regency Of Central Java. MAKER. Jurnal Manajemen, 9(1), 76-89.

Prashant. (2022). What is waterfall model?: Modified waterfall model. https://www.thestudygenius.com/what-is-waterfall-model/

Salve, S. M., Samreen, S. N., & Khatri-Valmik, N. (2018). A Comparative Study on Software Development Life Cycle Models. *International Research Journal of Engineering and Technology (IRJET)*, *5*(2), 696-700.

Sherman, R. (2015). Chapter 18 - Project Management. In R. Sherman (Ed.), *Business Intelligence Guidebook* (pp. 449-492). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-12-411461-6.00018-6