WGU C964

Task 2

Computer Science Capstone

Name: Rifatul Karim

Student ID#: 001515900

Date: 12/31/2022

Table of Contents

Letter of Transmittal

December 31, 2022.
Rifatul Karim
System Solution LLC
8976 Middle River Street
Poughkeepsie, NY 12603

Mr. Jones,

Nowadays, there has been continuous growth of a new brand of coffee with new blends being invented all the time. With so many choices available, overall makes it harder to make the decision about which coffee to drink. Such a similar problem, your shop has been experiencing, which enjoyed impressive sales growth in its first year, but the outdated selection process is holding the growth of developing new customers or increasing the average customer lifetime. To solve this problem, it is key to implement a system that could interact and guide people to give relevant offers based on their preferences and as well as being competitive against other coffee shop services this is where the recommender system comes into play, which is an information filtering system.

Using a recommender system will benefit in a way that it will improve the users' experience by enabling them to find coffee items in a couple of clicks. This proposed system will provide the functionality of selecting various types of features of coffee and the system will recommend a coffee based on the cosine similarity of a specified set of features the user selected. If the user likes the recommendation, they will try a different combination or in this way can be introduced to the newly available coffee of their favorite preference. This will increase user interaction with the service and thus will help the business by generating more sales.

I have calculated the resources and labor cost required to build the data product, totaling approximately $75,000. If the data product can be implemented successfully, then there will be additional monthly small fees required depending on which web and SQL database platform will be deployed and hosted respectively. I have great experience developing various software projects and have talented team members who have desired to create high-quality products, and I can promise that we can complete this new proposed system within 3 months.

I hope we will be able to collaborate on this proposed project, and I look forward to hearing from you. If you'd like additional information feel free to contact us and I will be glad to answer any other questions you may have.

Sincerely,
Rifatul Karim

Task 2 - Section A: Project Proposal

Problem Summary

The New York Coffee Shop has been operating its service for almost two years. Their service offers a lot of various coffee items that set them apart compared to other coffee shops. However, it can be very overwhelming for new customers to select from a lot of coffee items and this is evident from their weak sales figures in their second year and also the lack of sales from the new users. To solve this problem, the service needs to introduce a system that can provide guidance to pick coffee by the user preferences amongst the large volume of items. To carry out this idea, a recommender system needs to be implemented within the system which will solve the information overload problem by searching through a large volume of item information and will provide users with their category choice selected. This will work in a way that the user will be given features to select and the recommender system will be able to provide items related to the features that the user selected.

Overall, the coffee recommender system can reduce the information overload problem of the service and will be able to streamline the user's decision with their coffee selection process.

Application Benefits

As the New York Coffee Shop is unique from other coffee shops when it comes to a variety of coffee availability, this new selection process will help the user to make the decision of which coffee to select and perhaps also discover new coffee products that they have never tried before and will lead to more user interactions with the service. For instance, if a user likes a cocoa-featured coffee then he will likely try other coffee with similar featured items with more feature combinations. This increased user interaction with the service will pave the way for more business profits and makes New York Coffee Shop more competitive against other similar coffee shop services.

Application Description

The application will display a list of feature words that will be created from the dataset training. The features will help the user to select what type of coffee feature they would like to try and they will be used as the input for the recommender system. Users will be able to select

more than one feature. Based on the set of features the user has selected, the system will provide a list of coffee recommendations related to the features selected. Each coffee item in the list will include its description and the distribution of the features. Overall, this simple selection process will overall ease the users to make decisions about selecting coffee items.

Data Description

The data that will be used to construct the recommender system will be based on the publicly available specialty coffee reviews scraped from www.coffeereview.com. This site was founded by Kenneth Davids, who and his team have expertly reviewed over 6000 coffee roasts since 1997 and overall has become the world's most widely read coffee buying guide. Each coffee review describes many categories and attributes for each coffee and my most interest will be to look into their blind assessments. The data contains the following:

- Roaster Name (Data Type – String)
- Score (Data Type – Integer)
- Review Date (Data Type – Date)
- Roaster Location (Data Type – String)
- Coffee Origin (Data Type – String)
- Roast Level (Data Type – String)
- Agtron (Data Type – String)
- Est. Price (Data Type – String)
- Aroma (Data Type – Integer)
- Acidity/Structure (Data Type – Integer)
- Body (Data Type – Integer)
- Flavor (Data Type – Integer)
- Aftertaste (Data Type – Integer)
- Blind Assessment (Data Type – String)
- Notes (Data Type – String)
- The Bottom Line (Data Type – String)

The limitation of the dataset would be that some of the coffee roasters might not be available in the shop. Hence, if the recommendation system is successfully implemented into the

service, we will create a database that has your coffee selection and the system will select the items from the database. For now, this dataset will be useful for training purposes and I will use the coffee roaster datasets that are available only in North America as well as from the last five years.

Objective and Hypotheses

The objective of building this recommender system for the service is to make the coffee selection process easier for users to pick amongst the large volume of coffee items. This can be achieved by implementing a recommender system that works in a way that a user will be given a number of coffee feature choices and the system will provide a list of coffee items that are related to the features selected. Overall, the data product hypothesis is that if the recommender system is implemented, then the coffee selection process will get easier.

Project Mythology

The waterfall methodology is the most fitting software development methodology to tackle this project. This methodology works well for this project as the project is not complex and the requirements are precisely known in advance. There are five phases that will have specific deliverables to be met to advance to the next phase to ensure quality and they are:

- Planning: The initial phase is about finding the potential requirements of the application and concerns more about what it should do and not how it should do it and write the information into a formal requirement document.
- Design: The second phase involves analyzing how the application should work and finding any possible models and business logic that will work with the application. Based on the findings, a prototype is created.
- Implementation: This phase is about executing the technical design plan that was created in the prior phases. The data product is fully developed and the business logic is integrated as well.
- Verification: Here the application is tested by the Quality Assurance, beta testers, and all other testers and identifies issues that must be resolved.

- Maintenance: By this phase, the application is fully functional and ready for deployment to the web. The support and maintenance mode is begun to keep the application functional and up-to-date.

Funding Requirements

The overall product to build the recommender system will require funding of $55,000. This funding includes the cost of human resources and development tools. However, if the system can be implemented successfully, there will be required additional monthly or yearly pricing depending on which web platform the system will be deployed on and where the SQL database is hosted.

Stakeholders' Impact

The implementation of this recommendation service will benefit both the stakeholder and the users in their own ways. If the user likes the recommendation, they will try a different combination and discover new available coffee with their favorite preference. In this way, there will be more user interactions with the service and stakeholders can gain profit from the increased user interactions with the service. Stakeholders can have an insight into the visualization of the user interactions using the dashboard where they can check which feature selection is popular with the user and use this data to make any change to the item catalogs to improve business profit.

Data Precautions

The dataset will be used found from the publicly available specialty coffee reviews and they do not contain any sensitive information. Additionally, the application will not require users to create an account to access the recommender service.

Developer’s Expertise

I will work with a small team who are great software developers and have the experience to work on projects using the waterfall methodology. Although they are not proficient in the programming languages that I will use to tackle this project, still the transition to the new language will be easy and I will be heavily involved in this matter for a smoother transition to the

new language. Overall, I have great faith in my team members to bring out success to make this project complete as they have the desire to produce high-quality products.

Task 2 - Section B: Project Proposal

Problem Statement

The current New York Coffee Shop service is following an outdated item selection process that overwhelms the new users to make a decision from a vast combination of coffee available to select from. They can overcome this information overload problem by providing a personalized recommendation system into their system that can generate a small list of coffee items based on the features the user selects.

Customer Summary

New York Coffee Shop features both a point-of-sale system to select and buy coffee in-person and as well a web service as a delivery option. The coffee shop follows the best practice to produce coffee and is highly enjoyed by the users. The shop has enjoyed sales growth in the past year. However, in their second year, the shop's service has not been able to retain old users in the long run and has weak sales figures from the new users with the increased number of coffee items. While increasing the content, they were still following the outdated item selection process which potentially overwhelms the new customers with a vast combination of coffee available to select from. This way the shop will not be able to stay on top of the business in the future. Therefore, the shop's service must implement a recommender system into their system that can give personalized recommendations from the vast combination of items to the users as every user wants to find what they are looking for fast and in a couple of clicks. This personalized selection approach will become increasingly important in the future, as modern businesses are increasingly focused on the long-term value of customers to the business (Peppers & Rogers 1997).

Existing System Analysis

Their current service, while offering a large selection of coffee items to choose from, has very limited categorization that makes it hard to distinguish or find similar coffee items the user likes. This makes it difficult for the users to make a decision to select a coffee item without knowing each coffee's standout features. Therefore, it is essential to implement a system that can generate a list of coffee items based on the coffee features the user would like to try.

Data

The data that will be used to construct the recommender system will be based on the publicly available specialty coffee reviews scraped from www.coffeereview.com. This site was founded by Kenneth Davids, who and his team have expertly reviewed over 6000 coffee roasts since 1997 and overall has become the world's most widely read coffee buying guide. Each coffee review describes many categories and attributes for each coffee and my most interest will be to look into their blind assessments. The data contains the following:

- Roaster Name (Data Type – String)
- Score (Data Type – Integer)
- Review Date (Data Type – Date)
- Roaster Location (Data Type – String)
- Coffee Origin (Data Type – String)
- Roast Level (Data Type – String)
- Agtron (Data Type – String)
- Est. Price (Data Type – String)
- Aroma (Data Type – Integer)
- Acidity/Structure (Data Type – Integer)
- Body (Data Type – Integer)
- Flavor (Data Type – Integer)
- Aftertaste (Data Type – Integer)
- Blind Assessment (Data Type – String)
- Notes (Data Type – String)
- The Bottom Line (Data Type – String)

To obtain the dataset, I will web scrape the site for this task. I have specifically created a command line user interface to complete this task using Java programming language. Firstly, I will need to scrape the list of coffees and retrieve the links to their respective review pages. Afterward, from the list, I will scrape the details of each review page's details and save them into a single file in JSON file format.

From the dataset, I will require a few of the attributes from each data – “roaster name” for the coffee identification and the “blind assessment” that contains the descriptive words of the coffee that will be majorly used for data analysis to extract features from each of the coffee.

The limitation of the dataset would be that some of the coffee roasters might not be available in the shop so once the recommendation system is successfully implemented into the service, we will create a database that matches the shop’s coffee selection and the system will select the items from the database. For now, I will use a coffee review dataset based on the last five years.

Project Methodology

I have reviewed all of the different development processes of machine learning models methodologies and have found that the waterfall methodology will be the most fitting software development methodology to use to tackle this project. It will work well as this project is not complex and the requirements are precisely known in advance. The development process consists of five different steps that will have specific deliverables to be met to advance to the next phase to ensure quality and they are:

- Planning: The initial phase will be about finding the potential requirements of the application and concern more about what it should do and not how it should do it.
- Design: The second phase will involve analyzing how the application should work and finding any possible models and business logic that will work with the application. Based on the findings, a prototype will be created.
- Implementation: This phase is about executing the technical design plan of the project. The data product will be fully developed. Additionally, the business logic will be implemented as well.
- Verification: The application will be tested by the Quality Assurance, beta testers, and all other testers and report any presence of issues that need to be resolved. Furthermore, the application is then verified if it matches the requirement document.
- Maintenance: At this phase, the application will be ready for deployment to the web, and the support and maintenance mode is begun to keep the application functional and up-to-date.

Project Outcomes

The project will have four major deliverables before the application is deployed into the live environment.

1. The first deliverable will be obtaining the dataset.
2. The second deliverable will be about finding the right model that will work well with the dataset and then developing the data product.
3. The third deliverable will be to design a visual for the recommender system.
4. The fourth deliverable will be to test the application and fix the issues if found.

Implementation Plan

The project implementation plan will follow the waterfall methodology which is the process of completing the project deliverables phase by phase, where each phase is dependent on one another. The first phase will be about analyzing and noting down the project requirements and business case for the application into a document that will be used to serve as the basis for communication, planning, and verifying each of the deliverables. The next step will be about obtaining and analyzing the datasets. Afterward, a machine learning model is generated based on the dataset and then the prototype of the data product will be created. Then, the third phase of the project will follow with executing the technical design plan of the project. The visual of the application is worked on to incorporate with the recommender system. The business logic that was planned during the initial phase is implemented into the data product. The next phase will be to verify the deliverables created so far and ensure that they match the project requirements created during the first phase. The data product will be evaluated using various testing such as regression, usability, and A/B hypothesis testing and fix any issues if present. Finally, the project is deployed into the live environment and the maintenance mode is begun to keep the application functional and up-to-date.

Evaluation Plan

The data product will be evaluated once it is fully developed. It will be tested with three different testings that will be executed sequentially and they are:

1. Regression testing will be done by the quality assurance team. Here they will check the recommender system with a different combination of inputs and check if the output results are consistent with the given input.

2. The user acceptance testing is done by the beta testers where the application will be tested on various platforms. They will focus on the user interface if it is intuitive, easy to use, the visuals display correctly, and the function works as expected.

3. Finally, A/B hypothesis testing will be conducted by the coffee connoisseurs where they will determine the effectiveness of the recommender system if the recommended coffee items match the given feature selected. The testing is performed in the following step:

1. A taster is tasked to choose one or more coffee features they would like to try.
2. The recommendation is generated and then the taster is given top three coffee recommendations to try.
3. The taster is then tasked to give the user feedback.
4. Repeat step 1-3 until all testers have tried the system.

Once these tests can pass with mostly positive results, then we can deploy the new system into the live environment with full confidence.

Resources and Costs

Programming Environment

- The data product will be developed under the windows 10 operating system which will not require a cost.
- A simple command line application will be created using Java Programming Language to obtain the dataset from the site. Since the program will be simple, we will select a free Java IDE, such as eclipse, to do this task.
- Machine learning model will be programmed using Python using Pycharm Professional IDE which costs $24.90 per month.
- The web design development will be programmed using React web framework in the Webstorm IDE which costs $15.90 per month.

Environment Costs

During the data product development, we will be running on the render.com cloud service for prototype creation and testing purposes which will not require a cost. However, once the data product is ready to deploy, it will require additional cost to run the server as there will be data required to be stored in a separate database.

Human Resource Requirements –

I will work with a total of eight team members where everyone will play a different specialist role such as programmer, beta tester, and quality assurance and the salary for each member will require an average of $2,500 per month. Besides us, I will hire three coffee connoisseurs to perform the A/B hypothesis testing for the recommender system evaluation and they will cost a total of $15,000. I am estimating that the data product can be completed within three months.

Overall the breakdown of the total cost is:

Resource	**Description**	**Cost**
IDE Cost	Pycharm Professional Edition and Webstorm	$24.90 * 3 months + $15.90 * 3 months = $**122.40**
IT team cost	The team size is 8 and we plan to finish the project within three months.	$2,500 * 8 members * 3 months = $**60,000**
A/B testing	A/B testing will require coffee connoisseurs to do this task.	$5,000 * 3 members = $**15,000**

Total Cost: $75,122.40

Timeline and Milestones –

Task No.	Milestone	Start Date	End Date	Duration	Dependencies	Resources
1	Requirement Gathering and get approval from the stakeholder (**Phase 1**)	01/01/ 2023	01/05 /2023	05 days	None	Stakeholders, Development Team.
2	Complete the data collection process and perform the data quality. (**Phase 2 Starts**)	01/06/ 2023	01/08 /2023	02 days	Task 1	Development Team
3	Identify the machine learning model that will integrate with the dataset.	01/09/ 2023	01/23 /2023	14 days	Task 2	Development Team.
4	Create the data product prototype based on the model.	01/23/ 2023	02/01 /2023	8 days	Task 3	Development Team.
5	Full Production of the data product starts (**Phase 3 Starts**)	02/02/ 2023	02/07 /2023	5 days	Task 4	Development Team.
6	Work with the visual of the service to incorporate the recommender system.	02/08/ 2023	02/15 /2023	7 days	Task 5	Development Team.
7	Implement the business logic of the application.	02/16/ 2023	02/20 /2023	04 days	Task 6	Development Team.
8	Start the Regression Test and resolve any issues within the timeframe. (**Phase 4 Starts**)	02/21/ 2023	02/24 /2023	03 days	Task 7	Development Team
9	Perform usability testing and correct the issues quickly.	02/25/ 2023	03/01 /2023	05 days	Task 8	Development Team.
10	Coffee-tasting experts will come to conduct the A/B testing for the recommender system and take notes of any suggestions or issues.	03/02/ 2023	03/30 /2023	29 days	Task 9	Coffee Testing Experts
11	Final deployment (**Phase 5**)	03/31/ 2023	03/31 /2023	01 days	Task 10	Development Teams

Section D: Post-Implementation Report

Project Purpose

The New York Coffee Shop service is following an outdated item selection process that overwhelms the users to make a decision from a vast combination of coffee available to select from. Therefore, the purpose of implementing the coffee recommender system for the New York Coffee Shop service is to ease their coffee selection process and make the users easier to make decisions about selecting coffee amongst the shop's large volume of coffee items available. This is done by using a machine learning algorithm to learn the coffee items' features and use the features as the input for the user to select coffee items. Based on the feature selected, the system can provide a list of coffee items that are related to the feature selected by the user by utilizing a machine learning algorithm. Overall, this personalized selection approach will ease the users about their coffee selection and find their favorite feature coffee or discover something similar.

Datasets

The data that would be used to construct the recommender system will be based on the publicly available specialty coffee reviews scraped from www.coffeereview.com and stored in JSON file format. I scrapped the dataset using a separate command line application that was created using Java Programming Language. This application is included in the submission named "coffeeReviewScraper.jar". I have only scrapped the coffee data that appeared from the last five years. In total, the dataset contains 1620 coffee reviews. One example of the unmodified coffee review:

```
{
  "Name": "Colombia Castillo Fruit Maceration Series (Passionfruit)",
  "Roaster": "modcup coffee",
  "Location": "Jersey City, New Jersey",
  "Origin": "Armenia, Quindio Department, Colombia",
  "RoasterLevel": "Medium-Light",
  "Score": 95,
  "Agtron": "58/78",
  "Aroma": 9,
  "Acidity": 9,
  "Body": 9,
  "Flavor": 9,
  "AfterTaste": 9,
  "WithMilk": 0,
  "BlindReview": "Powerfully aromatic, fruit-centered. Passionfruit, lychee, ginger flower, almond brittle,
                  kumquat in aroma and cup. Sweet-tart structure with apple-toned, malic acidity; full,
                  syrupy mouthfeel. Exceptionally harmonious and integrated finish into the far long.",
  "Notes": "Produced by Jairo Arcila and Cofinet of Finca Santa Monica entirely of the Castillo variety of
            Arabica, and processed by an experimental fruit maceration process in which the whole fruit is
            first fermented in an anaerobic (limited oxygen) environment with wine yeasts and macerated fruit.
            After fermentation, the coffee cherries are pulped and placed to dry on raised beds with macerated
            fruits until the ideal moisture content is achieved. Founded in 2013, modcup coffee focuses on fresh
            roasting and distinctive natural-processed coffees. Visit modcup.com for more information.",
  "Summary": "High-toned, explicitly and cleanly fruit-forward, leading with notes of passionfruit and
              lychee – tropics in a cup."
}
```


***Figure 4.1**: Example of a particular coffee review scrapped from coffeereviews.com*

Each coffee review describes many categories and attributes for each coffee. For building the recommender system, my most interest was to look at the “Name” for the coffee identification and the “Blind review” that contains the description of the coffee’s features. The dataset file is included in the submission named “coffee_reviews_json.json” which has been put in the backend directory.

Data Product Code –

I have created the data product as a full stack application where the backend was written using Python with its Flask framework to develop a RESTful API to create various endpoints that return data to the frontend. The frontend was written using the React web framework. Additionally, for the backend part, I have incorporated a free Redis database from render.com to cache the data that will be frequently used throughout the program and therefore reduces the loading time. The backend program was created in the following steps:

The first step was to read the uncleaned dataset and cache the name of the coffee roasters and blind review.

```
def readDatasetsAndCache():
   with open("coffee_reviews_details.json", 'r', encoding="cp866") as json_file:
       uncleaned_coffee_reviews = []
       coffee_roasters = []

       for data in json.load(json_file):
           coffee_roasters.append(data["Name"])
           uncleaned_coffee_reviews.append(data["BlindReview"])

       cache('coffee_blind_reviews', json.dumps(uncleaned_coffee_reviews))
       cache('coffee_roasters', json.dumps(coffee_roasters))
```

The next step is to create a vectorized form of the blind review for my Non-Negative Matrix Factorization (NMF) model to train, which will be used to create the feature group and build the recommendation system. Therefore, I chose TF-IDF model that can create a vectorized form of the blind review by training the model. TF-IDF is the ratio of the frequency of a word in each coffee review to the frequency of a word in all of the reviews. Before the blind review datasets are trained into the TF-IDF model, I cleaned the blind review of the dataset by removing the punctuations and numeric values.

```
def clean_blind_reviews(blind_reviews):
    # removes the punctuations and numeric values and lowercase for each reviews
    blind_reviews_cleaned = [re.sub("[^a-zA-Z]", " ", s.lower()) for s in
blind_reviews]

    return blind_reviews_cleaned
```

Using the cleaned blind reviews, the TF-IDF model is initiated, which is created using Python's sklearn library. During the model initiation, I could add different combinations of settings for more relevant feature words to appear on the vector and eliminate the stop words i.e. the common words that are uninformative. To make a decision about the right TF-IDF model setting choice, I created a bar chart plot using Python's Matplotlib library that shows the descriptive side of the data, i.e. the number of frequent words that appear across the review. Using this descriptive visual helped me to identify more stop words appearing across the reviews and gave me an overview of how much the initial setting of the model affects the appearance of the number of unique and meaningful feature words. After finding the right setting combination,

I ran the model and kept removing the stop words that appeared on the top until the top 100 most common words gave me the higher number of useful words:

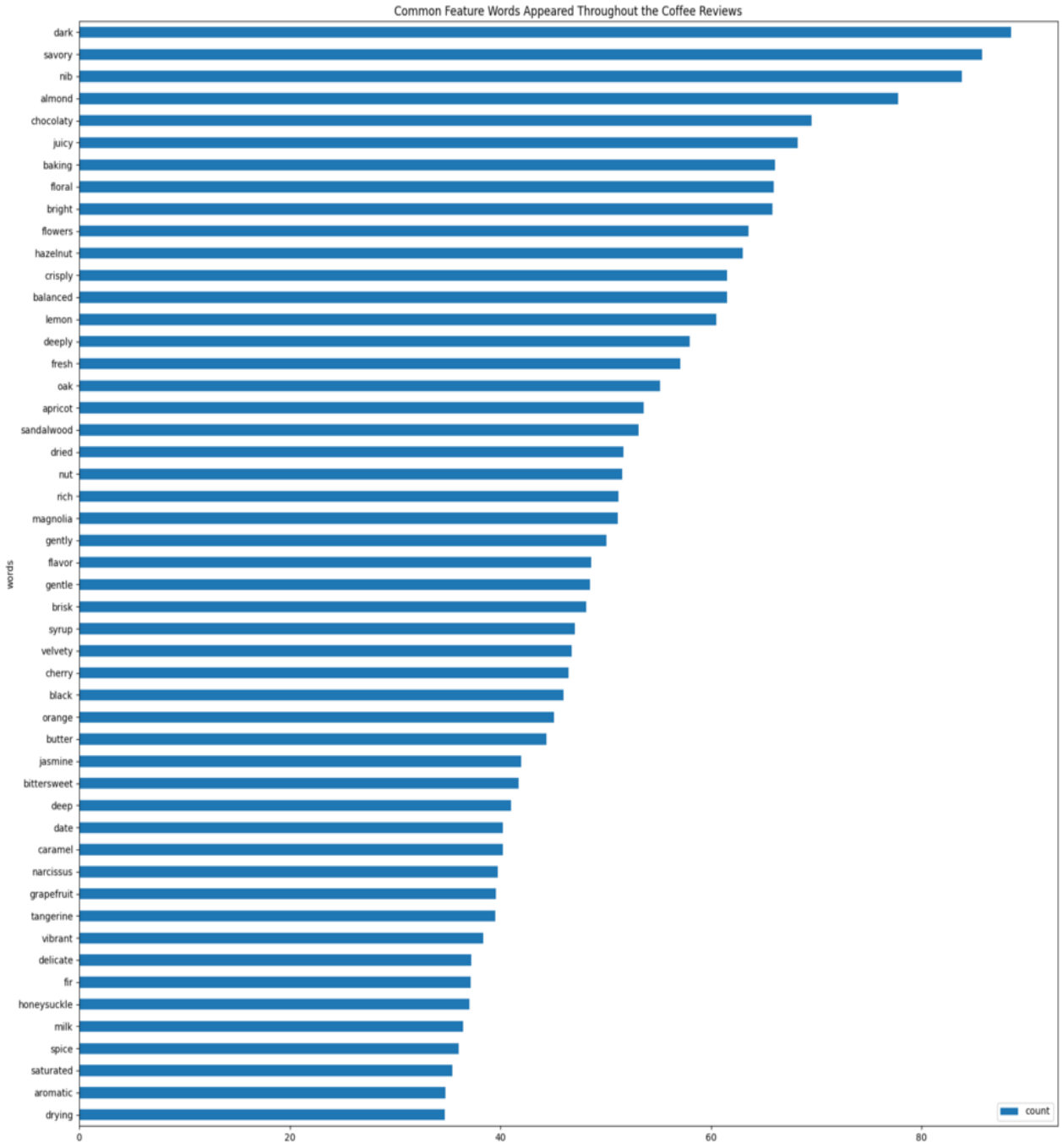


Figure 4.2: The bar chart displays the number of feature words that appeared throughout the coffee review.. This helped me to identify the stop words that appeared on the top and eliminate them during the TF-IDF model initialization to improve the output result. (Descriptive)

Once the TF-IDF model setting is established, it is trained using the cleaned blind review and returns both the vectorized and trained values.

```
def get_tfIdf_for_blind_reviews():
   # load blind assessment part of the coffee reviews from cache
   blind_reviews = load_json_value_from_cache('coffee_blind_reviews')

   # removes the punctuations and numeric values and lowercase for each reviews
   blind_reviews_cleaned = [re.sub("[^a-zA-Z]", " ", s.lower()) for s in
blind_reviews]

   # default stop words list from the sklearn library
   stop_words = text.ENGLISH_STOP_WORDS;

   # some stop words findings added to the default stop words list
   my_stop_words = stop_words.union(get_stop_words())

   # Instantiate the vectorizer class with setting
   tfIdf_vector = TfidfVectorizer(min_df=20,
                                  max_df=0.25,
                                  ngram_range=(1, 1),
                                  stop_words=my_stop_words,
                                  use_idf=True,
                                  smooth_idf=True)

   # Train the model and transform the data
   tfIdf_trained = tfIdf_vector.fit_transform(blind_reviews_cleaned)

   return {'vec': tfIdf_vector, 'trained': tfIdf_trained}
```

Using the large dimension vector of the TF-IDF vector, which represents the features words across the review, my next step would be to reduce the large dimension of the vector and cluster the feature words into a certain number of groups, and then the NMF model is used to achieve this. The NMF model factorizes the high-dimensional vectors into lower-dimensional representations. The NMF model is created using Python's sklearn library. Similar to the TF-IDF vectorizer, the NMF model output varies with the initial number of inputs selected. Therefore, in order to achieve the best NMF output, I have created a visualization for evaluating the model. This helped me to make a decision about selecting the right input for the NMF model. Since the main purpose of the project was to reduce the information overload and make decisions easier for coffee selection, I aimed to have between 6 to 15 groups and then pick the best value. I used

the coherence score to measure each NMF model against the number of features and create a scatter plot based on the two data. Based on the plot, the highest coherence score was found to be 9 and afterward, the line never reached that high.

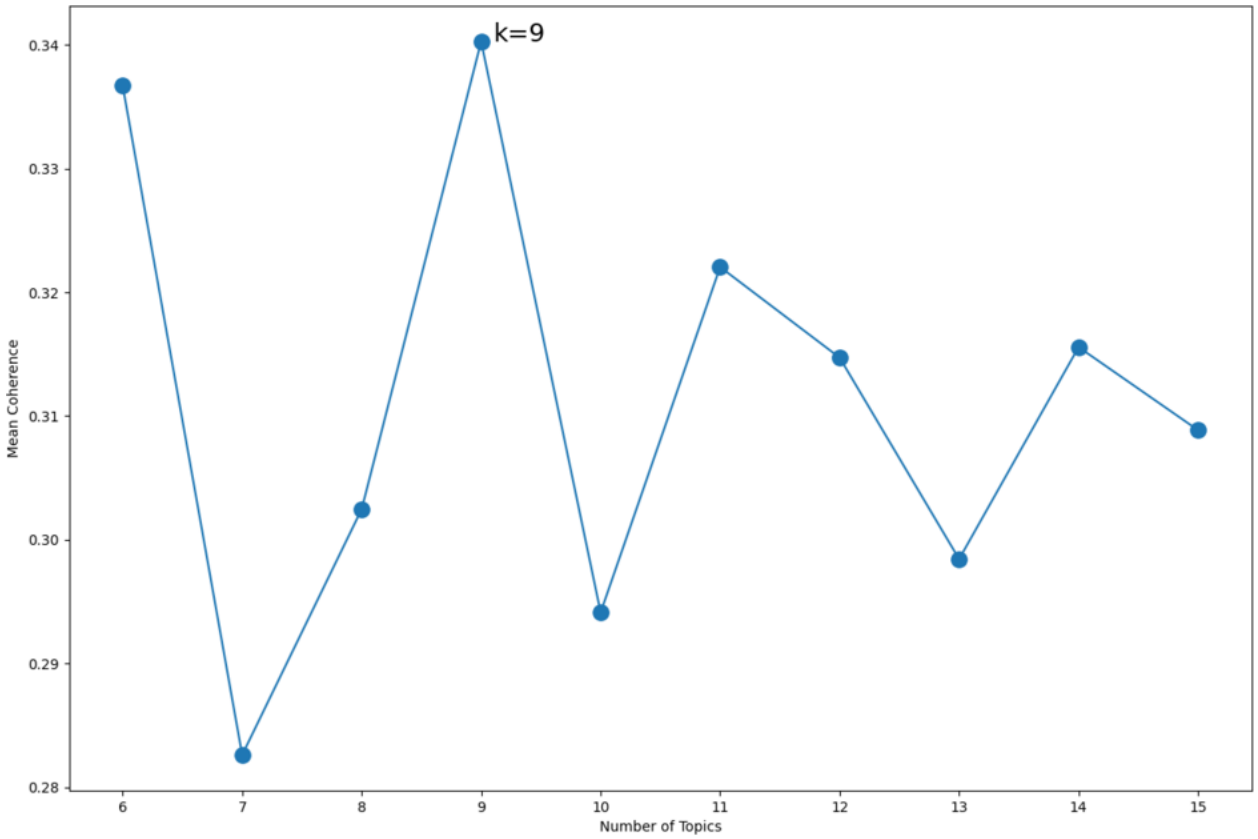


Figure 4.3: The scatter plot visualization was used to identify the best fit of the NMF model input for generating the best model output (Descriptive)

After I have selected the input for the model, I will train the NMF model using the trained TF-IDF model with the number of feature group selected. The model gives two vector outputs - W is the feature words it found and H is the coefficient of the coffee reviews by feature words. Afterward, I will store these values in the Redis database in pickle format to avoid recalculation of the model as these values will be frequently used to calculate the recommendation.

```
def trainNMFModel():
   # obtained the trained tfIdf value
   tfIdf_trained = tfIdf_for_blind_reviews()['trained']

   num_Of_feature_group = 9
   nmf = NMF(n_components=num_Of_feature_group)

   # fit the model to the tfIdf
   H = nmf.fit_transform(tfIdf_trained)
   W = nmf.components_

   cache('nmf_features', pickle.dumps(H))
   cache('nmf_model', pickle.dumps(nmf))
   cache('nmf_components', pickle.dumps(W))
```

Using the trained NMF model, I could work on selecting the feature words based on the model's output. To help me with this process, I created a visualization of the NMF model output as a word-cloud visualization type.

Figure 4.4: Word-cloud is used to visualize the NMF model output and select the standout feature of each group (non-descriptive)

The word-cloud displays the top ten words of each group. This visualization helped to understand how each word in each group dominates the other which can be identified by the larger text. Based on the word cloud, I have picked the feature words that make the most sense that identifies each group and stored them in a separate file. Then my next step will be to create an API endpoint to get the feature words from the frontend. The feature words are returned which are then formatted as JSON strings.

```
@app.route('/get_features', methods=['GET'])
def get_features():
# get selected feature words from the file
   feature_words = get_feature_words()
   return jsonify(feature_words)
```

Lastly, for the recommender system, I created an API endpoint for the recommendation system that will take the parameter of the feature words the user input from the frontend and then returns the list of recommended coffee. During this, “trackFeaturesUserSelected” is called to track the user-selected features for business logic purposes. This can be viewed visually from the Dashboard.

```
@app.route('/get_recommendations', methods=['GET'])
def get_recommendation():
   features_requested = [''.join([(feature + ' ') for feature in
request.args.getlist("features")])]
   trackFeaturesUserSelected(request.args.getlist("features"))
   return jsonify(recommend_coffee_with_features(features_requested))
```

During the “`recommend_coffee_with_features”` function call, firstly, it loads the various data from the cache. Then both TF-IDF and NMF models are separately calculated using the user-selected feature words as the input. With this new NMF model trained with the user’s requested feature, the cosine similarity is computed between the vector of this newly NMF trained model and the original trained NMF model. This is calculated using sklearn’s pairwise_distances function call which computes the distance between two vectors. After the calculation, the cosine distances are then sorted. Based on the cosine distances, the list of coffee item recommendations is created based on the top five cosine similarity distance values.

Full code:

```
def recommend_coffee_with_features(list_of_features_requested):
   # load various data from the redis database
   coffee_roasters = load_json_value_from_cache('coffee_roasters')
   coffee_blind_reviews = load_json_value_from_cache('coffee_reviews')
   nmf_model = load_pickle_value_from_cache('nmf_model')
   nmf_features = load_pickle_value_from_cache('nmf_features')

   # get tfIdf vector
   tfIdf_vec = tfIdf_for_blind_reviews()['vec']

   # tfIdf vector transform using the feature words as the input
   tfIdf_using_feature_words =
tfIdf_vec.transform(list_of_features_requested).todense()
   nmf_using_feature_words = nmf_model.transform(tfIdf_using_feature_words)

   # compute cosine similarities between the nmf using feature words selected
   # and the original nmf feature
   similarities = pairwise_distances(nmf_using_feature_words.reshape
                                     (1, -1),
                                     nmf_features, metric='cosine')

   # sort the similarities calculated by order
   similarities = similarities.argsort()

   # get the highest 5 cosine similarities values
   top_five_recommendations = list(similarities[0][0:5])

   # store the notes section of the unmodified coffee reviews
   coffee_summaries = get_unmodified_coffee_reviews_summary()

   # stores additional information for each recommended items
   recommended_coffees = []
   for rec in top_five_recommendations:
       coffee_blind_review = ["" + coffee_blind_reviews[rec]]
       tfIdf_blind = tfIdf_vec.transform(coffee_blind_review).todense()
       nmf_tfIdf_blind = nmf_model.transform(tfIdf_blind)
       genre_dist_chart_image_bytes = create_pie_chart(nmf_tfIdf_blind.tolist()[0])
       recommended_coffees.append({"coffeeRoasterName": str(coffee_roasters[rec]),
                                   "description": coffee_summaries[rec],
                                   "imagebytes": genre_dist_chart_image_bytes})

   return recommended_coffees
```

For each recommendation, the name of the coffee item, description, and the visual of the distribution of each feature represented as a pie chart is returned. The description of each item is borrowed from the uncleaned coffee review datasets. To understand the distribution of each feature for the particular coffee item, the pie chart visualization type is used and is displayed on the coffee recommendations list.

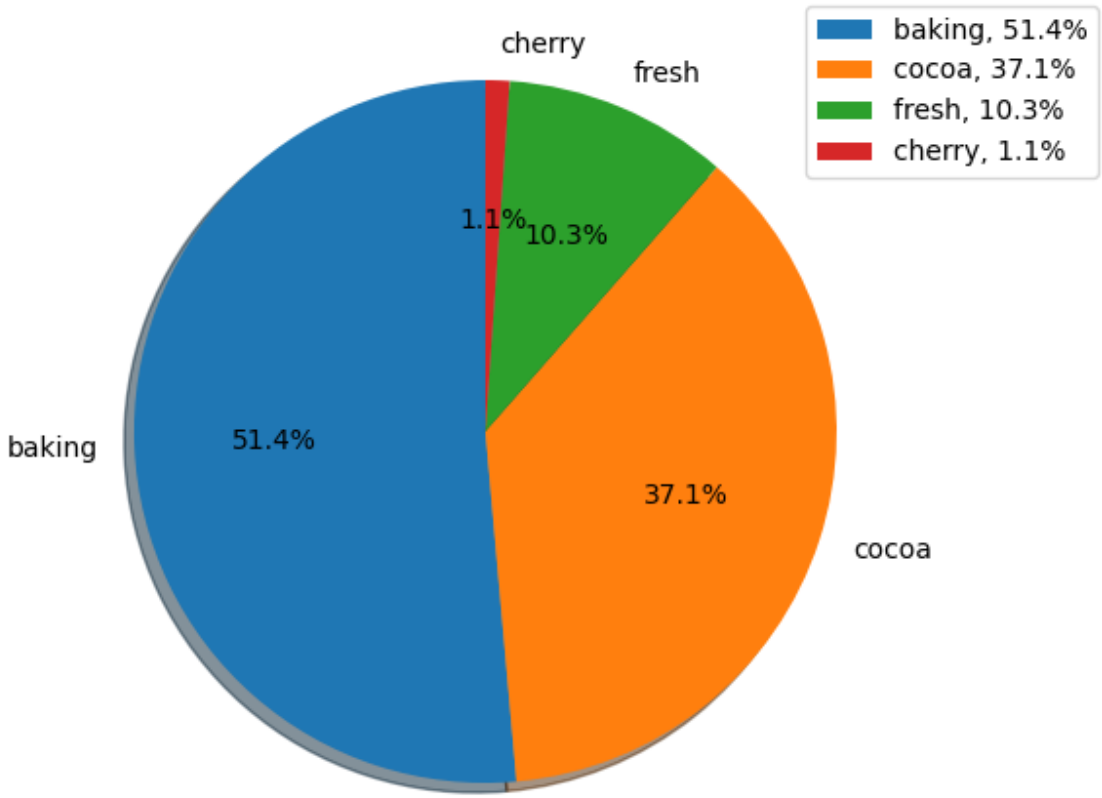


Figure 3.5: The pie chart visualization type is used to understand the feature distribution of each coffee item. (descriptive)

Hypothesis Verification

The initial hypothesis was that if the recommender system is implemented, then the coffee selection process will get easier. However, the hypothesis can only be truly verified once the system goes live deployment. It will be measured using a feedback system, where the user will provide an opinion of the recommendation system if it was helpful to them. However, from the initial testing done by the coffee connoisseur, the result was found mostly positive. This can be improved by expanding the dataset as the current dataset is based only on the last five years. This way more coffee features can be added for the input to select and display more relevant results.

Effective Visualizations and Reporting –

I have implemented various visualizations within the program that helped me to understand the data and make decisions about the model. They are

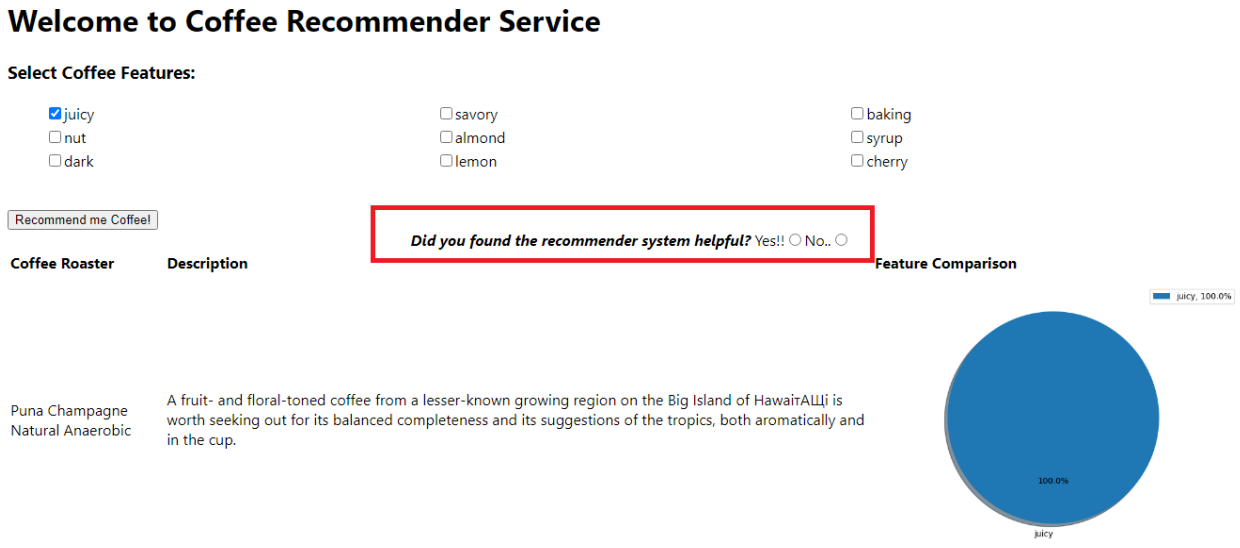
Type	Figure No.	Visualization Type	Description
Descriptive	4.2	Bar chart	The bar chart displays the number of feature words that appeared throughout the coffee review. This visualization provides insight into the frequent word that appeared on the coffee reviews and also helped me to improve the TF-IDF model output by identifying the stop-words that appears on the top 100 and eliminating them during the initialization of the TF-IDF model.
Descriptive	4.3	Scatter Plot	Scatter plot was used to make a decision about selecting a number of feature groups for the NMF model so that the output produces the best result and each group would be as distinct as possible.
Non-Descriptive	4.4	Word Cloud	Word-Cloud was used to visualize the cluster group of the NMF model output and helped me to identify and select the standout feature word of each group. This can be identified by the larger and bolder text appearing on each group that indicates the dominating word of the group.
Descriptive	4.5	Pie Chart	Pie chart is used to display the distribution of each feature for the particular coffee item. This

			visualization is shown when the system generates the coffee recommendation list.

Overall, the first and fourth visualizations help to analyze the meaning of the data representation. (Descriptive). The second visualization helps to understand the measuring of the NMF model (Descriptive) and the third visualization assists to analyze and find the meaning of the model output. (Non-Descriptive)

Accuracy analysis

As mentioned in the "Hypothesis Verification" section, the accuracy of the recommender system can only be truly verified once it goes live and implemented into the service and by collecting the user's feedback about the new system. The feedback option has only two options: Yes and No. The user responses will be sent to the database where each feedback value will be converted to a numeric value, i.e. 1 and 0 respectively. Afterward, the accuracy can be determined by calculating the average value of the feedback values by taking the sum of the feedback values divided by the number of users who clicked. This user feedback system will be monitored continuously once the system is deployed this way accuracy of the coffee recommendation system can be determined and this way we can verify the hypothesis if the recommender system has eased the user's coffee selection process.



However, based on the testing done by the coffee connoisseur, the accuracy of the recommender system was found to be mostly positive. This can be improved by expanding the dataset as the

current dataset is based only on the last five years and adding more features for the input to select and display more relevant results.

Application Testing

Application testing was conducted after the creation of the data product. Firstly, the regression testing was conducted by our quality assurance team where they ensured the consistency of the model's output with the various combinations of the given input. Once the recommender system has been connected to the web, the user acceptance testing was done by the beta testers who focused on whether the user interface was intuitive, easy to use, and visuals displayed correctly as expected. The final testing was the A/B hypothesis testing held by the coffee connoisseurs and the effectiveness of the recommender system output such as if the recommended items matched with the feature selected was determined and the result was mostly positive.

Application Files –

The "coffee-recommender-service" folder contains the project file that mainly consists of two main directories - frontend and backend, and a standalone java command line application.

coffeeReviewScraper.jar The command line application written in Java that was specifically created to scrape data off the coffeereviews.com. The instruction to run this application will be provided in the user guide.

frontend (directory):

node_modules (directory) – contains all the React dependencies files

public (directory) – contains static files such as index.html, a javascript library for the website.

src (directory) –

Index.js – the main component

App.js – the main component that renders the top navigation and displays the home content.

DashBoard.jsx – the component that renders the dashboard contents.

Home.jsx – the component that renders the home content which displays the list of features and recommendations. Additionally, API data is sent or received from the backend.

FeatureSelectionGroup.jsx – the components that render the list of features checkboxes on the home content and track which feature the user has selected.

Table.js – create the recommendation table visualization component with given data

NavBar.jsx – create the top navigation component and display blue text depends which main content the user selected to view.

backend (directory): contain files for running the data product

app.py The main application of the file that starts the server.

nmf.py Contains functions for calculating and getting the TF-IDF model, training the NMF model, and generating recommendations using the trained model

visualizations.py Contains the Python code to create various types of visualization of the data.

redis_util.py Contains reference to load the Redis database and has a function to cache or load the data from key

file_reader.py – Contains functions specifically to read various file contents: datasets, stop word list and feature words.

coffee_reviews_datasets.json the original unmodified dataset that was scrapped from coffeereviews.com and was used to create the data product

stop_words contains a list of stop words that were found during the TF-IDF model analysis. This is then used for the TF-IDF model input to generate a meaningful output of the model

feature_words Contains a list of feature words selected from the NMF model output. This is used as the input for the recommender system.

Profile Configuration file for the render.com

properties Contains the secret key to load the Redis database. This can be hidden when deployed on render.com using environmental variables during the final product.

requirements.txt Contains the list of packages that must be installed to run the application.

data_visualizations (directory) Contains the static images of the data visualizations.

User Guide

The complete full-stack data product application is hosted online at the following address: https://nyc-coffee-recommender-service.onrender.com so no installation is needed. Click on the address or copy this address and paste it into the browser to open the application site.

As the application currently runs using the free version, it takes time to load the initial data to display on the screen due to the backend server takes time to load various libraries and dependencies. Once the list of features is displayed on the screen means that the application has been fully loaded successfully.

- To view the recommendation system functionality,
  click one or more features and then click the “Recommend me Coffee!” button to view the recommended coffee items related to the selected features.

- To access the dashboard, click on the top navigation bar “Dashboard”.
  - Once clicked, the screen will display a navigation to the left where one can select five different types of data visualization to view from.
  - Clicking one of them will load the respective visualization to display the data on the right.
  - Sometimes the data loading is slow due to the slow speed of the backend server and other times wrong images may display clicking the different option again can solve this issue.

To start up the coffee scraper command line application, it is required to install Java 17 or higher versions in windows. If it is not installed, then follow the steps:

1. Download Java JDK from the following link: https://www.oracle.com/java/technologies/downloads/#jdk17-windows, and then install the JDK.
2. After installation completes, right-click on the Computer icon of your desktop and select Properties.
3. Click Advanced system settings.
4. Click Environment variables.
5. Under User variables, click New.
6. Enter JAVA_HOME as the variable name.
7. Enter the path to the JDK as the variable value and then Click OK.
8. Locate the PATH variable.
9. Append the path to the JDK (up to the bin folder) as the PATH value.

To verify if the Java variable was added correctly, open the command prompt and then enter - Java -version. The output will give the version of Java that has been installed into the computer.

If Java is installed on the computer, follow the step to load the application:

1. Open the file explorer and go to the file directory where the application is located.
2. From there, press SHIFT and click the RIGHT mouse together and a popup screen will screen.
3. Option to open a PowerShell or terminal window will appear and click any of one of them.
4. Once the PowerShell window opens, enter “java -jar coffeeReviewScraper.jar” to start the application.

Summation of Learning Experience

This project was really the culmination of my years of computer science experience at WGU and I had to apply various knowledge that I have learned over my WGU journey. The skills I learned from IT Leadership Foundations (D194) gave me the idea about writing proposals and Software Engineering (C188) for learning about the resource and cost management of software development projects and briefly knowing the terms about the various project methodologies which were learned thoroughly on Project Management (C176). Software II (C195) helped me to understand how to approach larger and more complex applications and Data Structures and Algorithms II (C950) gave me the taste of developing Python applications. Largely my success in these two courses together immensely gave me the confidence to develop this ambitious data product and make it to the end. And lastly, the Introduction to Artificial Intelligence (C951) gave me familiarity with the machine learning aspect.

Besides the course resources, I have used other WGU resources such as PluralSight and WGU Udemy. Through these resources, I gained familiarity with other various machine learning topics and tried to implement the models mathematically until I decided to switch to using Python libraries as I felt I need to gain experience through the machine learning libraries. So in this way, I could quickly understand how much adjusting the model's initial setting and the dataset's quality affects the model's output result. However, I think I could improve the model by having more mathematical knowledge and implementing the model in a low-level language. This way I could further tweak the model.

Overall, based on the knowledge I gained through the creation of the data product, I feel confident that this experience will help me in my future computer science career.

Sources:

Maze, J. (2019, December 3). Starbucks' rewards program pushes the chain's growth. Restaurant Business.

https://www.restaurantbusinessonline.com/marketing/starbucks-rewards-program-pusheschains-growth