DATA 440: Final Project Write-up

Greta Lin Risgin

Automations & Workflows

12 May 2025

Got Coffee?
The Williamsburg Café Recommendation System

Introduction

A good coffee blend, exciting specialty drinks, a welcoming atmosphere—these elements enrich the subculture that has formed around coffee and coffee shops. Many communities around the world have their own unique coffee culture. American coffee culture is particularly popular among students, with coffee shops providing areas to study and caffeine to power through assignments.

Williamsburg offers numerous off-campus coffee shops. However, due to the existence of on-campus coffee shops, many students never expand their coffee shop bubble. On-campus options include locations such as Town Center Cold Pressed and the ISC Starbucks. These locations often prioritize efficiency, but lack the quality and ambiance that often is associated with many independent cafés. For this project, I take user inputs, and output the coffee shop that best fits their unique preferences. It also outputs the recommended coffee shops's signature drink or coffee blend. This introduces the user to an interesting new coffee option to try. This recommendation system allows students a fun way to explore new coffee shops and coffee drinks outside of campus options. The system also provides a potential boost to the Williamsburg economy, as its suggestions prompt students to take their dollars off campus and put them into the local community. For students without dining dollars, there is no cost difference in buying a drink on or off campus. However, since off-campus coffee shops provide a more quality experience, this recommendation system is also introducing students to an improved coffee experience.

Data Collection

I chose to collect my own data for this project by visiting fifteen off-campus coffee shops in Williamsburg. A "coffee shop" is defined in this study as an establishment with an expansive coffee drink menu that serves either light refreshments or full meals. For this reason, chain establishments, such as Dunkin, Starbucks, and Wawa, were included in the study due to their robust coffee menus. However, establishments that serve coffee, but operate more as a restaurant or diner (e.g. Another Broken Egg Café), were excluded from the study. Within the context of this project, the terms "coffee shop" and "café" are used interchangeably.

I began by formulating a list of questions to ask each coffee shop in order to document information important for my recommendation system. This initial dataset included five binary variables and one ordinal variable. For the binary variables, I considered aspects of coffee shops that would be important for students. That is, if a café lacked one of these aspects, it would be enough to exclude it as a recommendation.

The first of these variables was "Study Space," which asked whether or not the establishment had seating for studying. I believed this would be especially important to students looking to both buy a coffee and complete homework in one fell swoop.

The second of these variables was "Vehicle Requirement." For students without cars, some of these coffee shops are not realistically accessible by foot. I evaluated the need for a vehicle for these coffee shops by using the Wren Building as a center point, and deeming any coffee shop farther than one mile away from this point as requiring a vehicle to reach.

The third of these binary variables was "Non-Dairy Charge." Lactose intolerance is fairly common among the population, however, many coffee shops charge extra for non-dairy milks. For example, almond, oat, and soy milk often come with a significant upcharge. This could be a financial burden on students, especially. Therefore, I chose to include this variable such that users could filter out coffee shops that include the upcharge on their drinks.

The fourth binary variable was "Gluten Free." This documented whether or not a cafe includes gluten-free food options. Cafés lacking in gluten-free food options would be excluded as a recommendation for the users this applies to. Even if a café technically had gluten free options, but offerings were incredibly limited (e.g. an apple or a rice crispy treat), these cafés were still marked as "False" for having gluten free options. This is due to the lack of purposeful gluten-free food inclusion the café menu, and therefore potential gluten cross-contamination could be present at these locations. Thus, gluten-free users would not be recommended these locations. Both the "Non-Dairy Charge" and "Gluten Free" were included in the data collection process in order to be mindful of the user's allergens.

The final binary variable was "Food Menu," which is for users seeking to also purchase a meal. Coffee shops with quick bite food options such as brownies or croissants, but lacking in substantial meals were marked as "False." All locations included grab-and-go options, so this was not used as a variable.

The ordinal variable was "Price Point," which included "low," "mid," and "high" distinctions. I based my price point distinctions around the price of a 16oz cup of iced coffee with no additions after tax. The average price of a cup of coffee in Virginia is $3.08. However, I operated under the assumption that the Williamsburg average would be a bit higher. Therefore, a 16oz coffee less than or equal to $3.50 was labeled "low," a 16oz coffee greater than $3.50 and less than $4.00 was labeled "mid," and a 16oz coffee greater than or equal to $4.00 was labeled "high" for price point.

A recreation of the first few rows of my base coffee shop dataset have been included below. "Cafe ID" connects all three of the utilized datasets.

Table 1: Williamsburg Café Dataset

Café ID	Study Space	Vehicle Requirement	Non-Dairy Charge
0	1	0	1
1	1	0	1
2	1	0	1
3	1	0	1

Café ID	Gluten Free	Food Menu	Price Point
0	0	0	High
1	0	1	Mid
2	1	1	Mid
3	1	1	Mid

”Atmosphere Description” and ”Specialty Drink Description” were the two nominal variables collected in my data. I selected thirteen common descriptors for both café atmosphere (e.g. cozy, modern, bright, etc.) and coffee drink flavors (e.g. strong, sweet, fruity, etc.). For each café I visited, I chose three of the descriptors for atmosphere based off my own subjective opinion. For speicalty drink descriptors, I asked each café what their either signature drink or blend was, and to describe it using three of the descriptors from the list. The chosen descriptors were then placed in a descriptor dataset, with each row including the corresponding café ID, one atmosphere descriptor, and one specialty drink descriptor. A recreation of this dataset for the first four cafés has been included below.

Table 2: Café/Drink Description Dataset

Café ID	Atmosphere Description	Specialty Drink Description
0	Chic	Nutty
1	Chic	Fruity
2	Local	Chocolatey
3	Cozy	Spicy

Reference information, such as the name of the coffee shop, the signature drink, and the address of the establishment were placed in a final dataset. After running the recommendation system, the cafe ID was used to pull the string value for the coffee shop name and signature drink for display purposes. The collected addresses were benign data for this current project, however, they were still included for potential future expansion of the project (e.g. the addition of a dynamic map). A recreation of the first lines of this dataset is included below. Note that the addresses have been shortened to just the street address for each establishment.

Table 3: Café Reference Dataset

Café ID	Café Name	Specialty Drink	Address
0	Eleva	Snickerdoodle Dirty Chai Latte	111 S Boundary St
1	Illy	Vanilla Raspberry Cappuccino	435 W Duke of Gloucester St
2	Aromas	S'mores Latte	431 Prince George St
3	Hohl	Housemade Dingo Syrup Latte	219 N Boundary St

Data Handling

Data was stored in CSV files and converted to Pandas dataframes. Due to the small size of the data, Pandas handles the data with efficiency and effectiveness. In order to output the optimal recommendation for each unique user, I used one-hot encoding and multi binary label classification. This way every variable is in binary. Each café had a vector populated with 1's and 0's. Each café vector would then be compared against user's preferences, which were similarly converted into binary form. The variable for "Price Point" was given weights, such that if the user chose a low price point, low price ranked coffee shops would be weighted heavier than mid price ranked shops, which would we weighted heavier than high price ranked shops. The reverse was true for a high price preference. For a mid price preference, low price ranked shops were given heavier weights than high price ranked shops.

For each binary variable, except for "Non-Dairy Charge," the coffee shops that do not meet the user specification are removed from the ranking. I chose not to remove cafés that upcharge for non-dairy milks because only three shops had no extra charge (1607 Coffee Company, Starbucks, and Dunkin'). I ultimately decided that since users who don't drink dairy woul still be able to access non-dairy milk at each location, albeit with an upcharge, the ranking system would only rank these shops lower against the shops without an upcharge.

For the subjective preferences, such as atmosphere and drink flavors, I allow the user to enter their top three desired descriptor for both categories. Multi binary label classification was used to convert the list of categorical preferences to binary. The complete user preference vector is compared against each coffee shop using cosine similarity from the Scikit Learn library. It creates a new column of similarity scores and adds the column to the encoded dataframe. The dataframe is ranked by score, and the top row is taken as the match.

The user preferences were collected using a Dash interface. The interface then feeds the preferences into the café recommender program and returns the top match. This top match is then used with the interface to select a corresponding unique artwork of the recommended signature drink.

Results

Running the main.py file creates a user-friendly Dash interface. The interface successfully prompts the user for their preferences with a mix of radio buttons and dropdown menus for the available options. The app only prompts the user one question at a time, in order to create a structured survey, withou overwhelming the user with all questions at once. Seventeen artworks were created for the interface header, buttons, and recommendation icons. The Dash app layout is utilized for the mechanical setup up the interface, while the created art provides a unique aesthetic for the project. I have been able to produce every possible recommendation with the interface. The outputted recommendation may not fit all of the user's preferences, but it will be the closest out of all the coffee shop options.

Project Expansion

I have already begun the next phase of my project by experimenting with hosting the app on my own website. However, when hosted the website is running on outdated Python libraries, which diminish the effectiveness of the recommendation system. For future project expansion, I would like to work on getting the recommendation system to work as a hosted website with the most updated libraries. Once I get past this step, I have two plans for project expansion.

The first plan is to use the address data for each coffee shop and provide the user with a dynamic map of their café matches. This way the user could see all their matches on the map, and the similarity score to the user's preferences.

The second plan is to move away from a static list of descriptors, and move towards utilizing a LLM, such as Bag of Words, to calculate similarity scores for ideal atmosphere and drink flavors. With the static list of descriptors, vectorizing the descriptors could create an inaccurate recommendation system, as it could be susceptible to telling the program that two unrelated descriptors are related. However, in a future iteration of this project, users could be able to enter any descriptor they want. Descriptors could then be added to the descriptor dataset. The code for creating the descriptor list for atmosphere and signature drink already takes the top three descriptors, and with more additions to the data, this would create for more accuracy for these variables. Utilizing large language models could potentially allow for a more accurate and unrestrained user experience. For example, if the user chooses to enter ”homey” for ideal atmosphere, this would be calculated as closer to ”cozy” than it would to other variables, for example ”modern.”

References

- Daily Dose of Data Science, The Most Overlooked Problem with One-Hot Encoding, https://blog.dailydoseofds.com/p/the-most-overlooked-problem-with
- Dash Bootstrap Components, Layout, https://dash-bootstrap-components.opensource.faculty.ai/docs/components/layout/
- Github, Project Data and Collection Questions, https://github.com/fluffycowfluffy/COFFEE-SHOP-PROJECT/tree/main/data
- Geek for Geeks, Bag of Words Model in NLP, https://www.geeksforgeeks.org/bag-of-words-bow-model-in-nlp/
- Geek for Geeks, One Hot Encoding in Machine Learning, https://www.geeksforgeeks.org/ml-one-hot-encoding/
- Plotly, Advanced Callbacks, https://dash.plotly.com/advanced-callbacks
- Plotly, dcc.RadioItems, https://www.datacamp.com/tutorial/learn-build-dash-python
- Plotly Community, Populating a Dictionary with dcc.Store, https://community.plotly.com/t/populate-a-dictionary-using-dcc-store/82114
- Plotly Community, Using Lists as Values in RadioItems, https://community.plotly.com/t/use-list-as-value-in-radioitems/38499
- Plotly Community, Refresh Page Button, https://community.plotly.com/t/refresh-page-button/17877
- Saving Spot, The Price of a Cup of Coffee in Every State, https://www.cashnetusa.com/blog/price-cup-coffee-every-state/
- Scikit Learn, MultiLabelBinarizer https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html
- Scikit Learn, OneHotEncoder, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
- StackOverflow, How to Change Image Size in Plotly Dash, https://stackoverflow.com/questions/58483221/how-to-change-image-size-in-plotly-dash