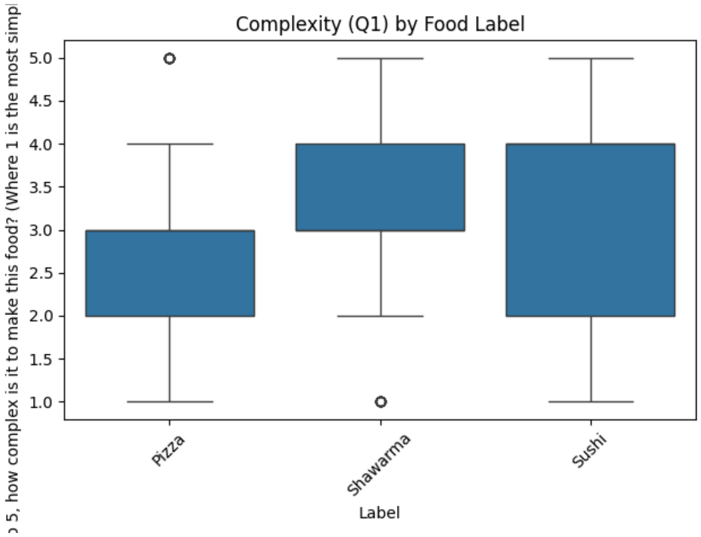
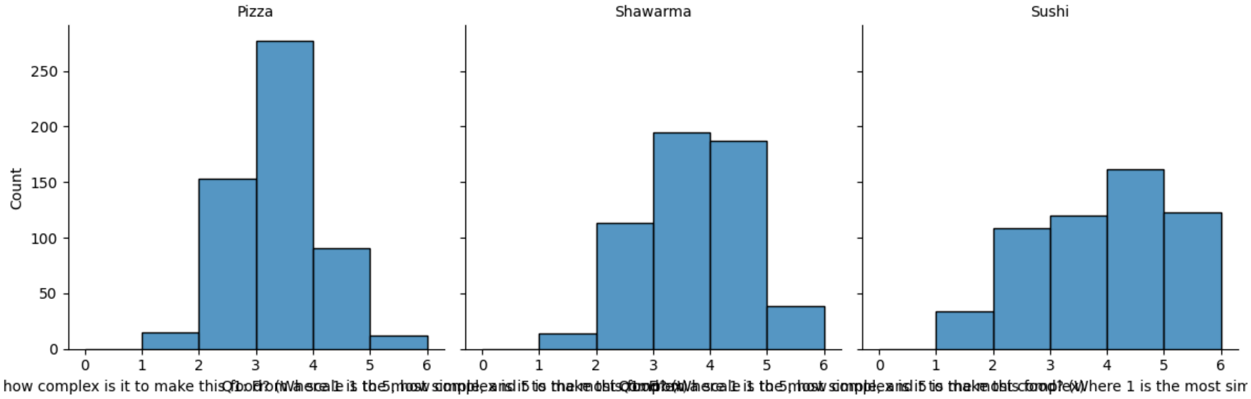
CSC311 Challenge Report

Data

In order to thoroughly explore the data, we plotted many graphs and charts for each feature.

Feature 1: Food Complexity





We notice that there is a higher volume of people voting for 2 and 3 on the complexity of pizza; people also vote for 3 and 4 on the complexity of shawarma more often, while the distribution of votes is more uniform when people are rating the complexity of sushi. We also calculated the means of the complexity of each food as follows:

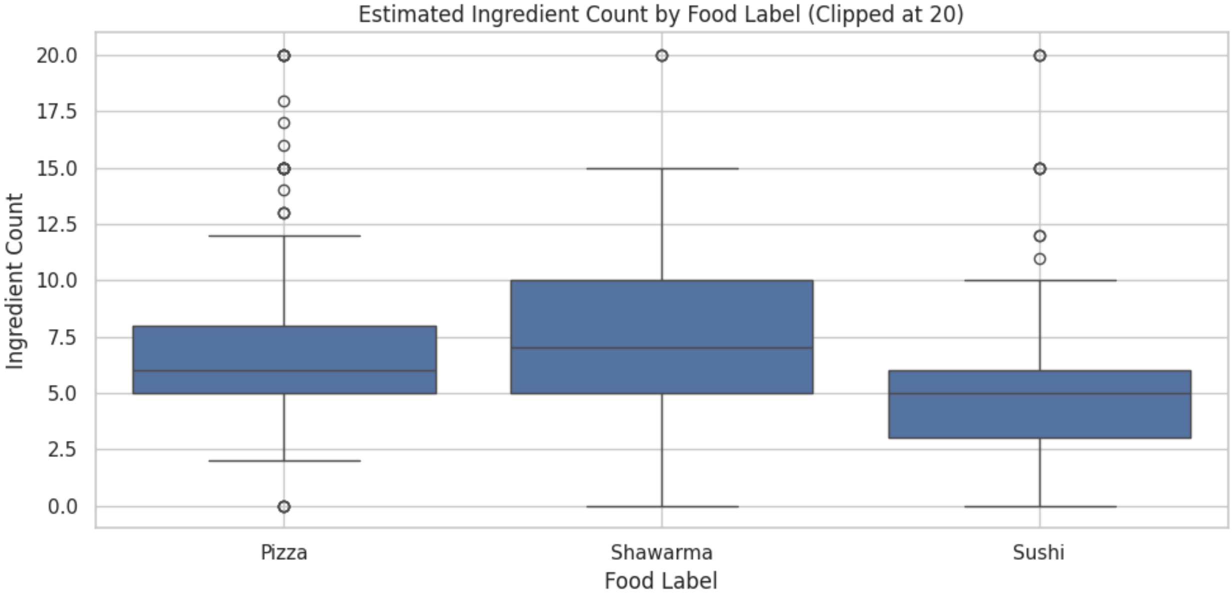
Pizza	Shawarma	Sushi
2.88	3.23	3.42

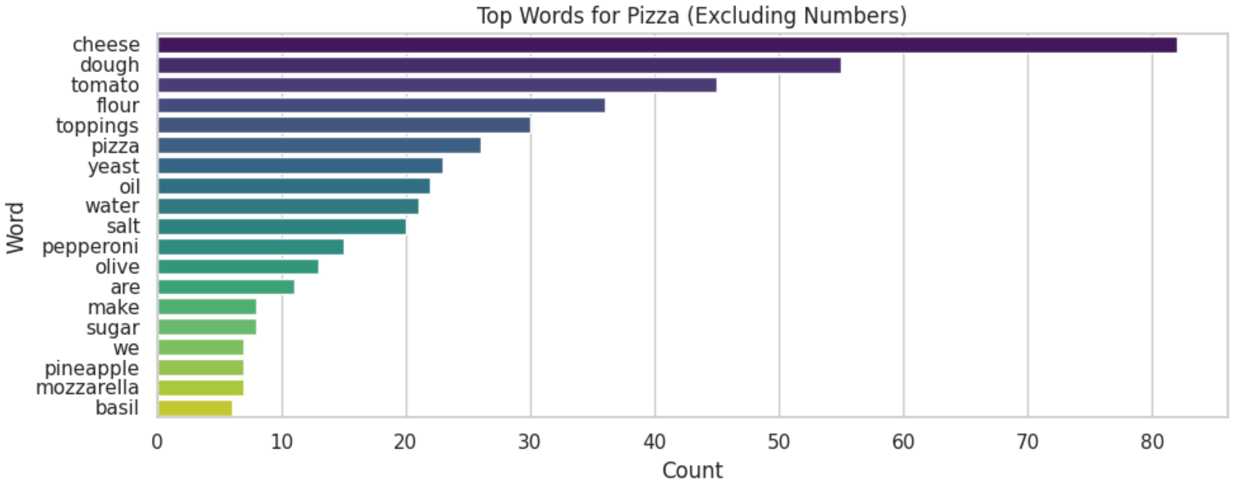
It seems that the means are distinct enough for us to correlate the complexity of a food item to some value, these being pizza having the lowest complexity, sushi having the highest complexity, and shawarma being somewhere in the middle. This feature is especially important to distinguish pizza and shawarma which we found to be a challenge.

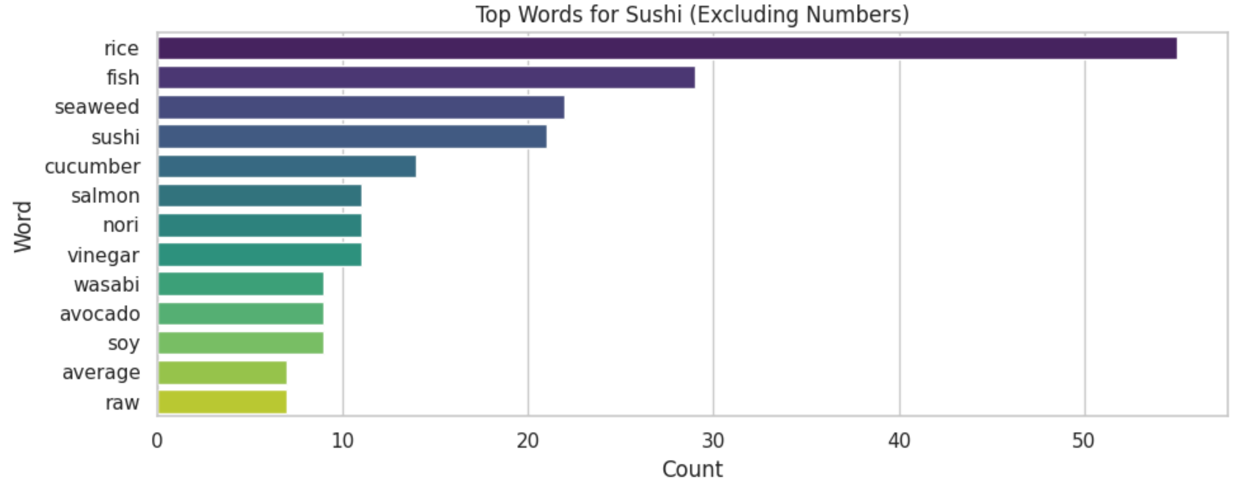
Feature 2: Ingredients

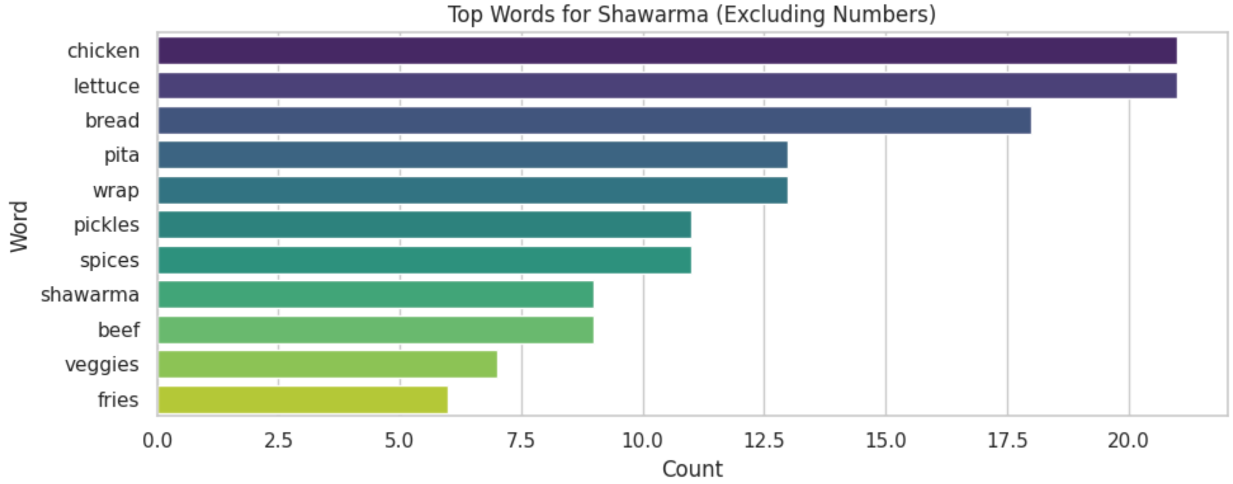
Ingredients being a free text feature which a bias towards numerical values was analyzed as a textual feature, where responses could mention which ingredients are used in producing this food item, as well as a numerical features containing the overall number of ingredients used, by extracting the minimum/average/maximum values of all numbers in the response. This is because some responses give a range of numbers rather than just providing a single value.

Through testing, it was found that maximum values for free value numerical responses consistently produced better results than minimum / average value of ranges, although by a very small margin.

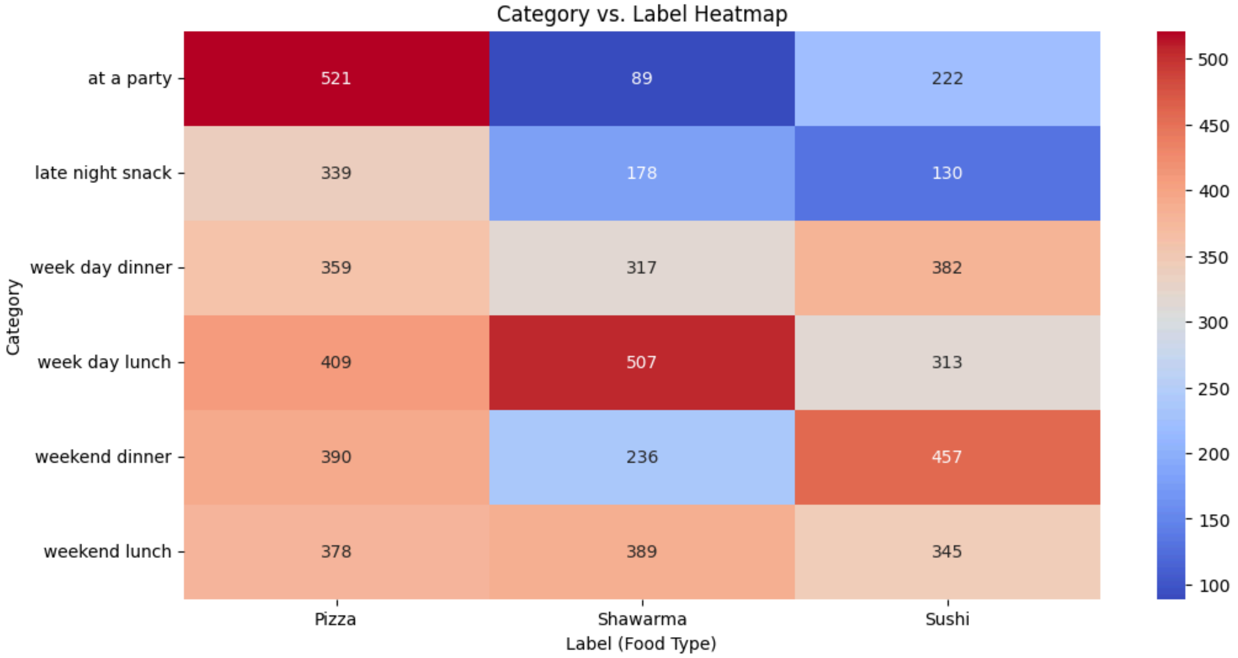



Top Words for Pizza (Excluding Numbers)


Top Words for Sushi (Excluding Numbers)


Top Words for Shawarma (Excluding Numbers)

Feature 3: Meal Setting

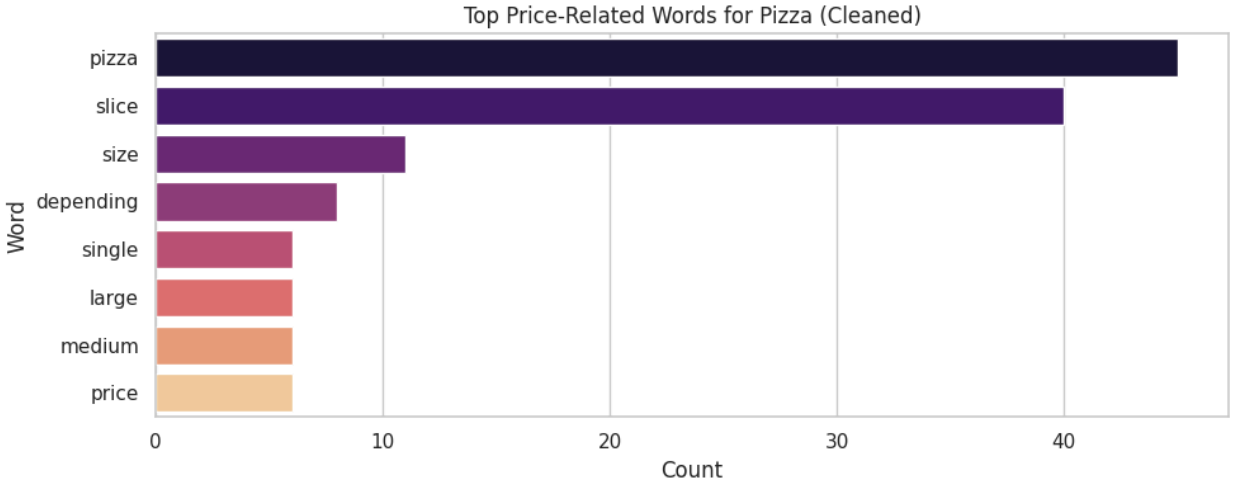


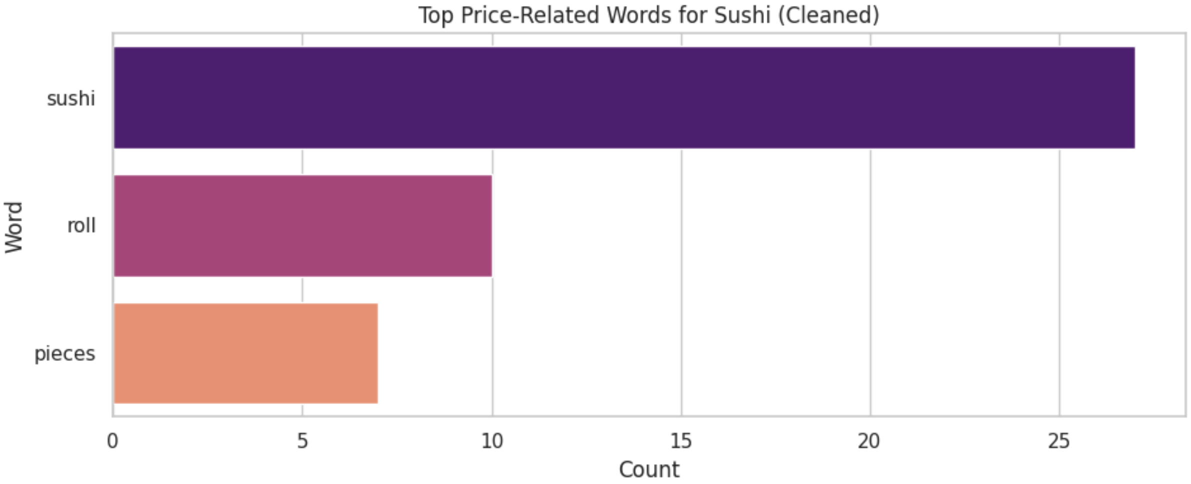
This heat map displays the number of votes that each food item received for each meal setting. We notice that the distribution of appropriate settings to find pizza in seems uniform, with the exception being “at a party” having a higher frequency, which should be expected. The other food items have more varied frequencies, where people expect to find shawarma more often on weekday/weekend lunches, and weekday dinners; people also expect to find sushi more often on dinners and lunches (both weekday and weekend).

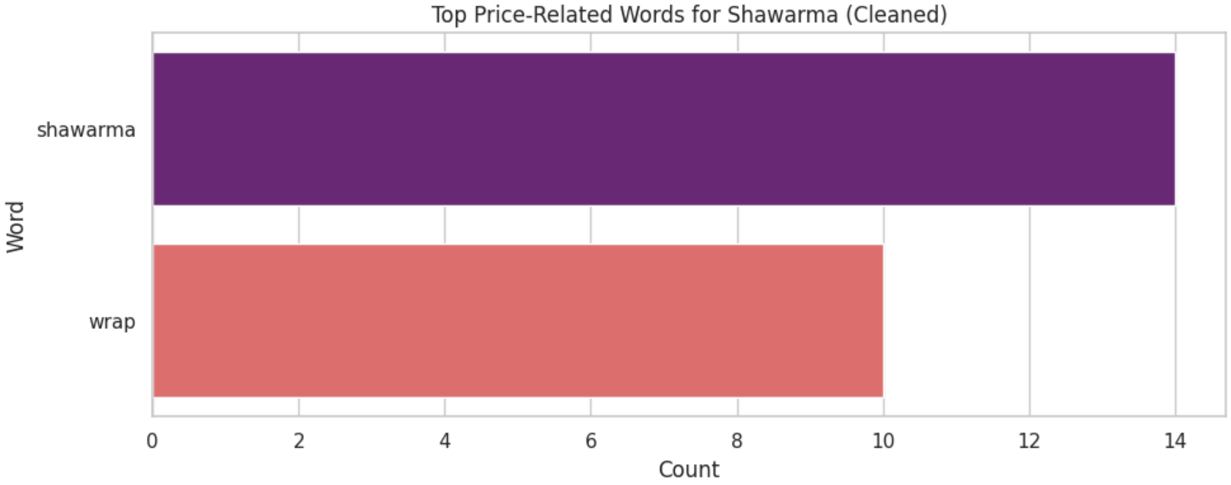
Label category	Pizza	Shawarma	Sushi
At a party	0.217	0.052	0.120
Late night snack	0.141	0.104	0.070
Week day dinner	0.150	0.185	0.207
Week day lunch	0.171	0.295	0.169
Weekend dinner	0.163	0.138	0.247
Weekend lunch	0.158	0.227	0.187

Above is a chart showing the percentage of people who voted for each meal setting, with respect to the food item. We can see that each food item has one meal setting where a food item gets picked significantly more than the others, which have been highlighted. As such, we are able to have a light correlation between the most popular choice of meal setting and its respective food item.

Feature 4: Price





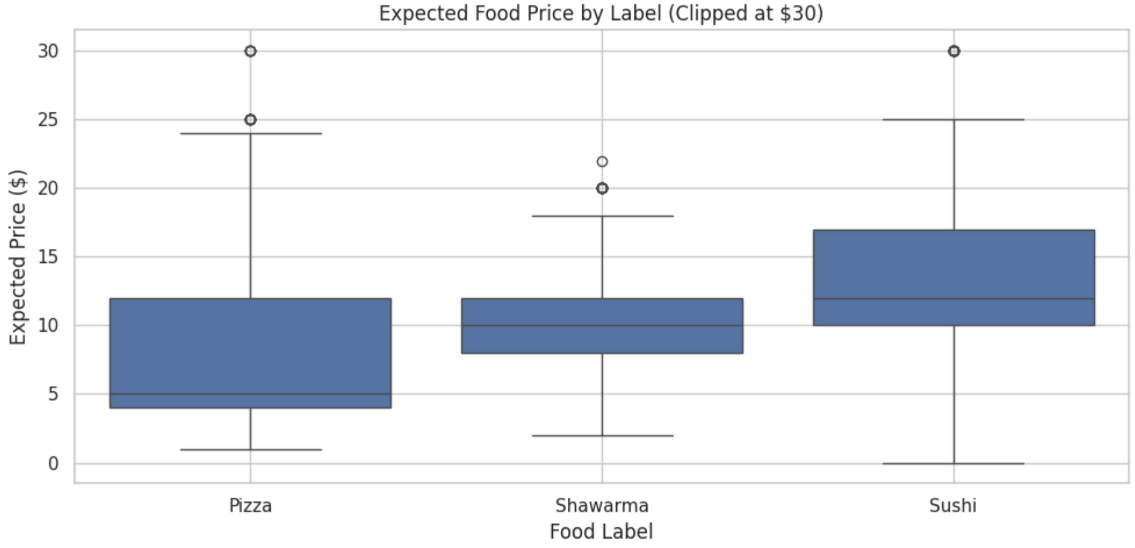


An interesting finding was that the most common text feature for each category was the item itself, so we could have an engineered feature checking if the respondent included the food item in this response. This was followed by a unique word for each category such as:

- Pizza: Slice
- Sushi: Roll
- Shawarma: Wrap

This is because it makes sense to talk about price per serving size in terms of what the serving is called.

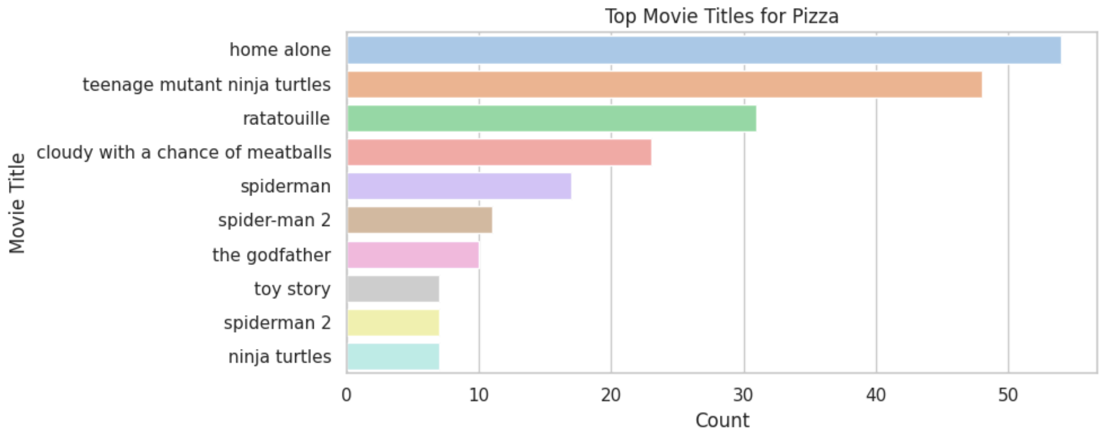
Next, a numerical analysis was performed for each category (Capped at $30 to remove noise):

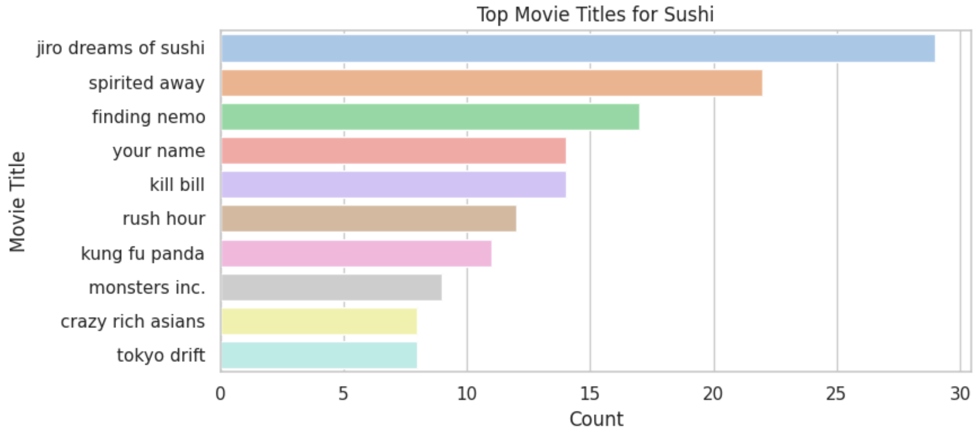


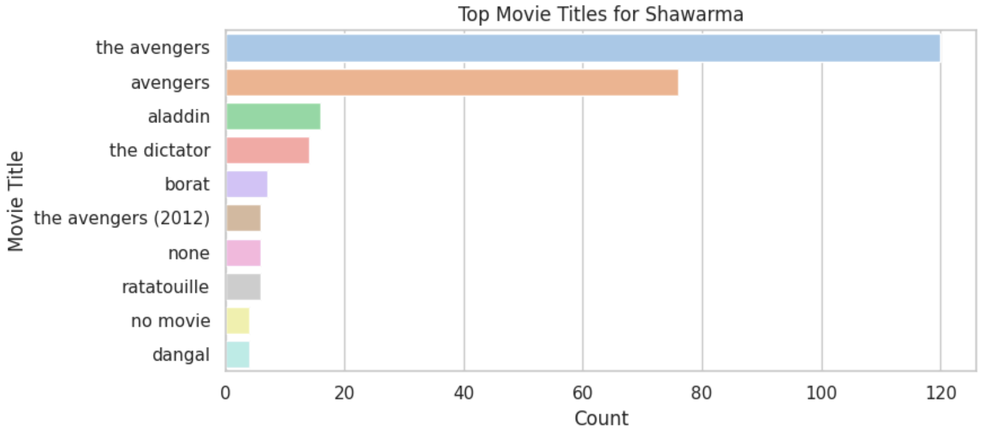
Notable findings are:

- The lowest price was a slice of pizza, followed by a shawarma wrap, followed by a roll of sushi.
- The median for pizza was extremely low compared to the similar median for shawarma and sushi, which may help differentiate pizza from shawarma and sushi which proved to be important
- The averages across all categories were fairly different, making this feature useful for classification of all food items
- Shawarma had a smaller spread than pizza and sushi, which had around the same spread. A reason for this could be respondent interpretation of the question. Some users may have interpreted the question to mean one slice of pizza or one roll of sushi while others may have interpreted it to mean a whole pizza or a small package of sushi similar to the ones they sell in the IB building. But there aren’t as many interpretations for shawarma; it always comes in one wrap.

Feature 5: Movie





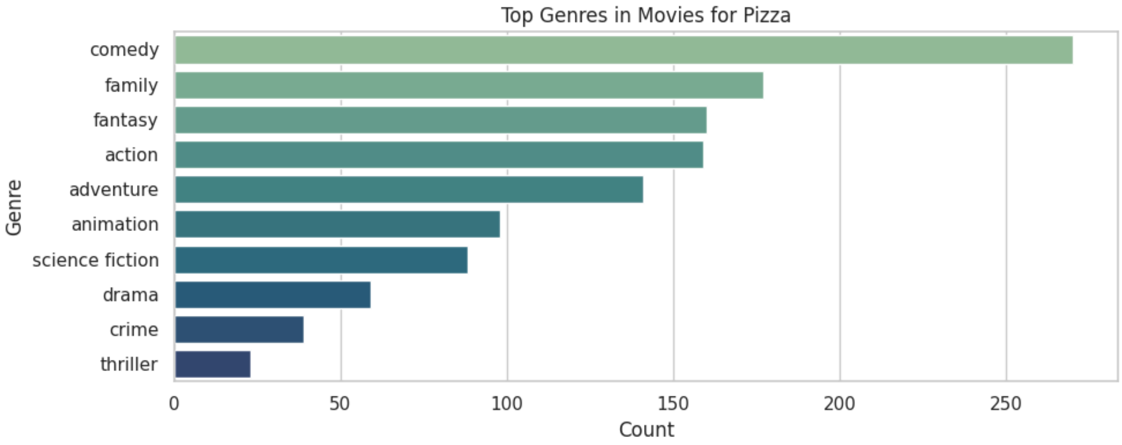


This was a difficult category to work with, since there are potentially hundreds of movies a respondent could think of. Some movies did heavily correlate to a food item:

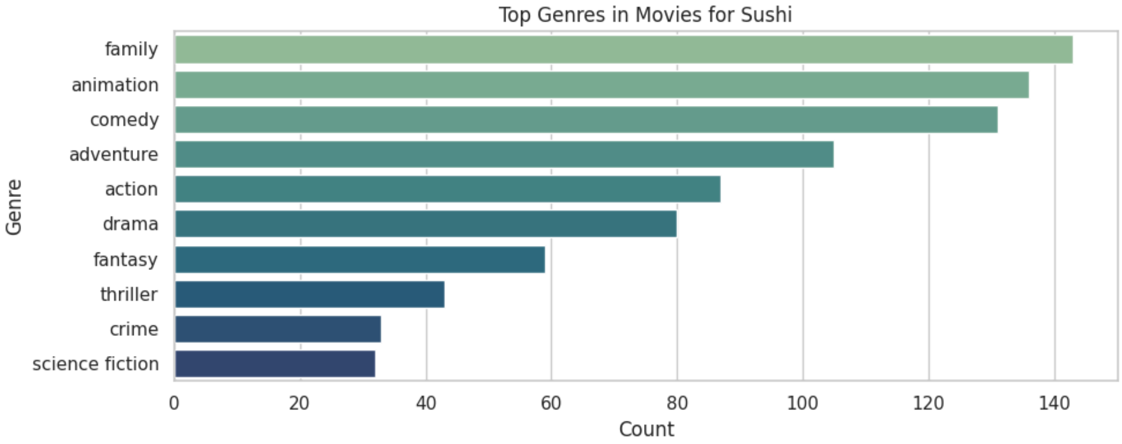
- Avengers: Shawarma
- Home alone / TMNT: Pizza
- Jiro dreams of sushi/ Spirited away: Sushi

A solution for this would be to map the movies to a database that contains other features such as languages/genres/Countries of production

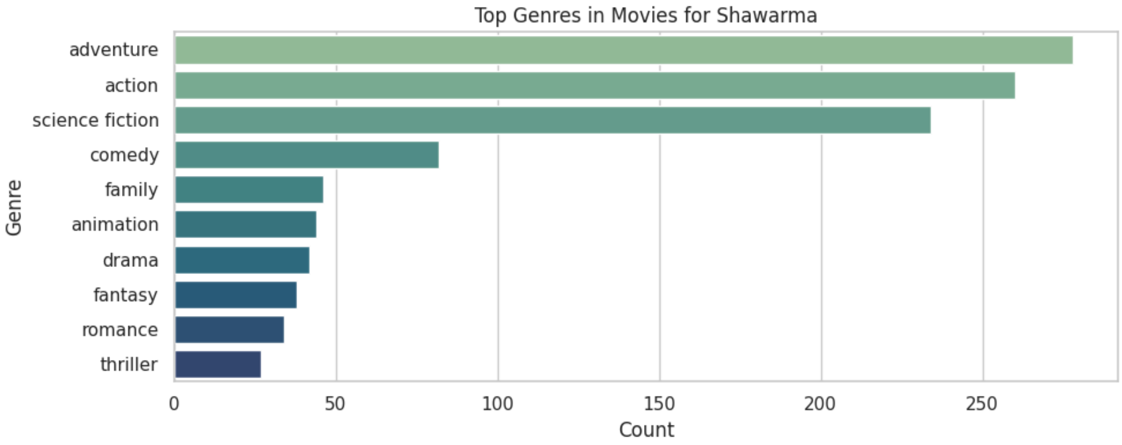
GENRES:



Top Genres in Movies for Pizza



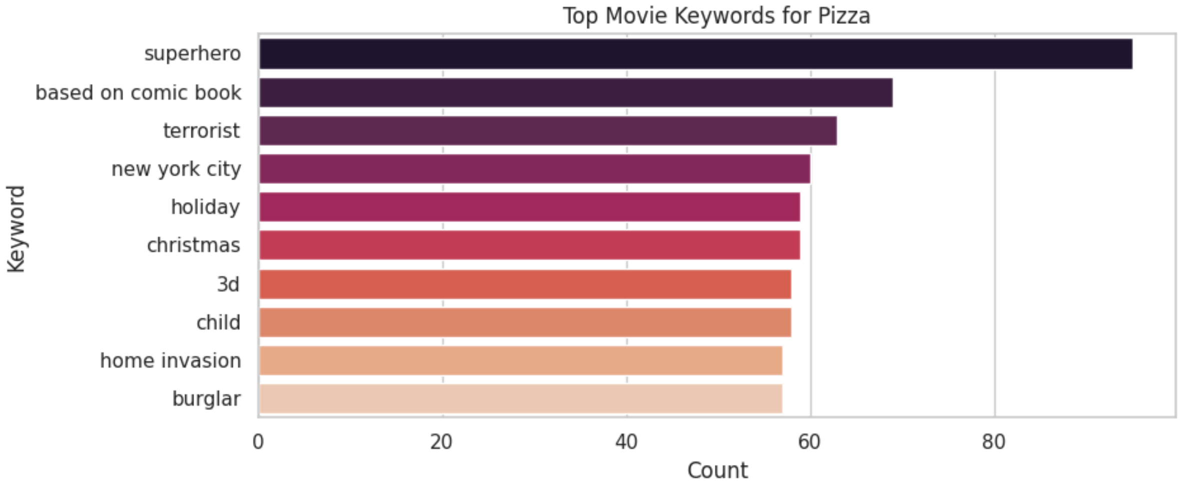
Top Genres in Movies for Sushi



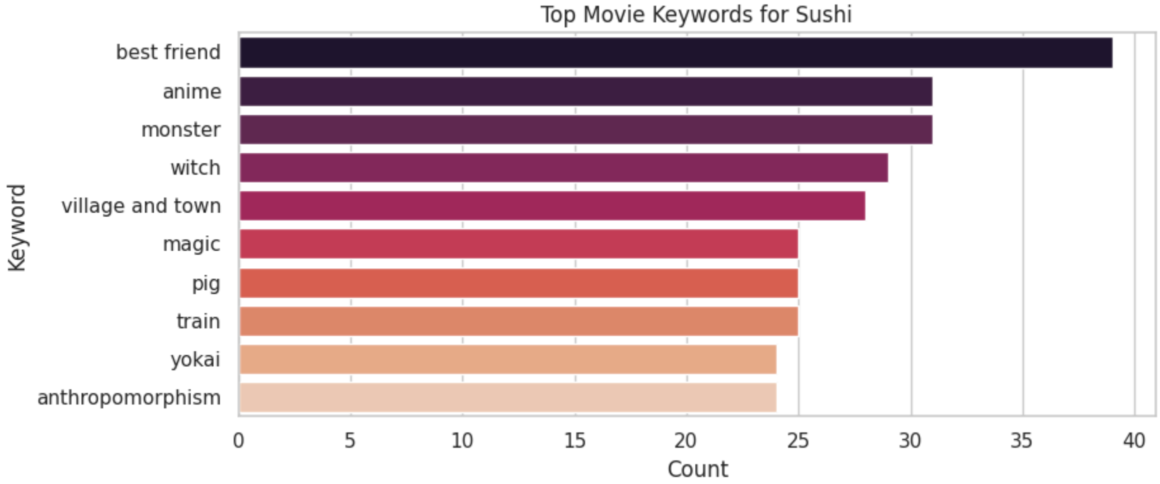
Top Genres in Movies for Shawarma

Surprisingly this was the strongest and most unique choice for a mapping. Certain genres are more closely associated with certain foods:

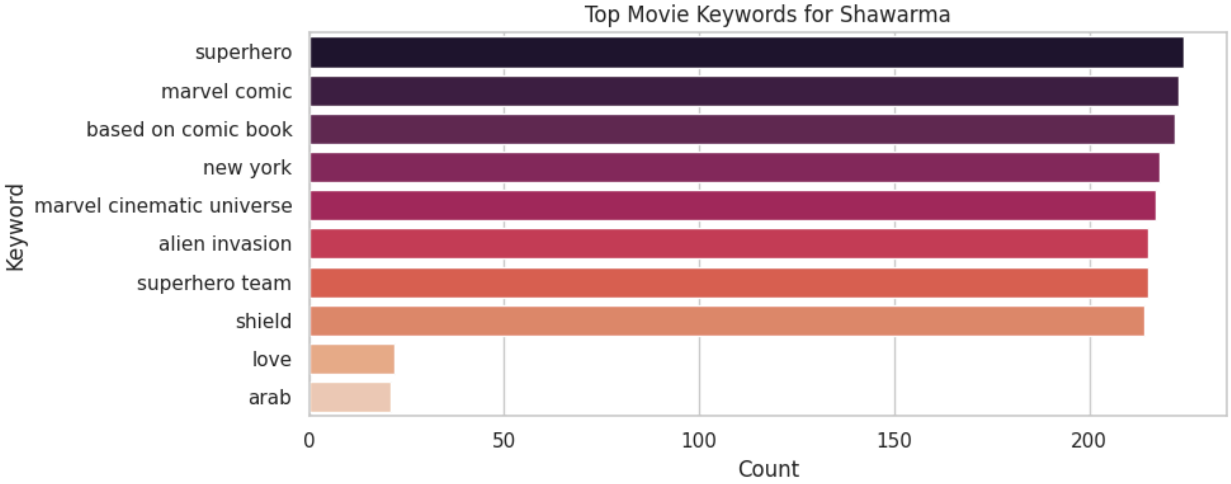
- Comedy/Family/Fantasy: Pizza
- Family/Animation/Comedy: Sushi
- Adventure/Action/Science fiction: Shawarma



Top Movie Keywords for Pizza



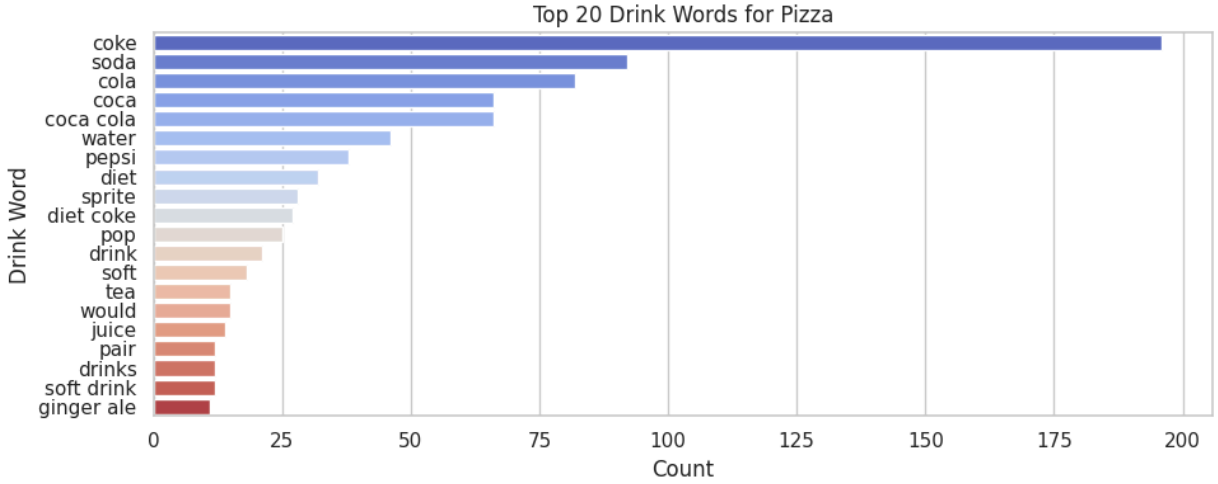
Top Movie Keywords for Sushi

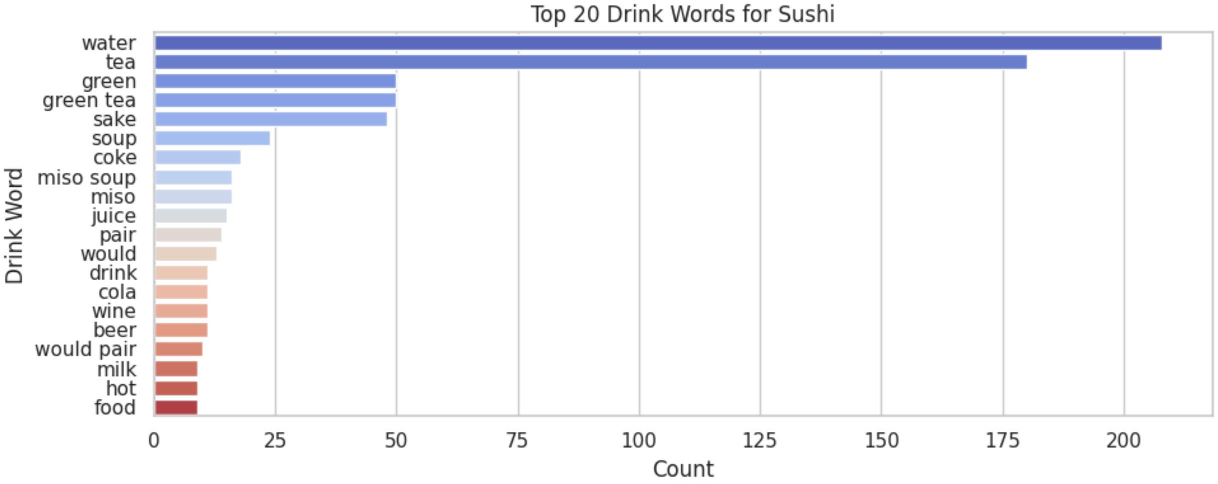


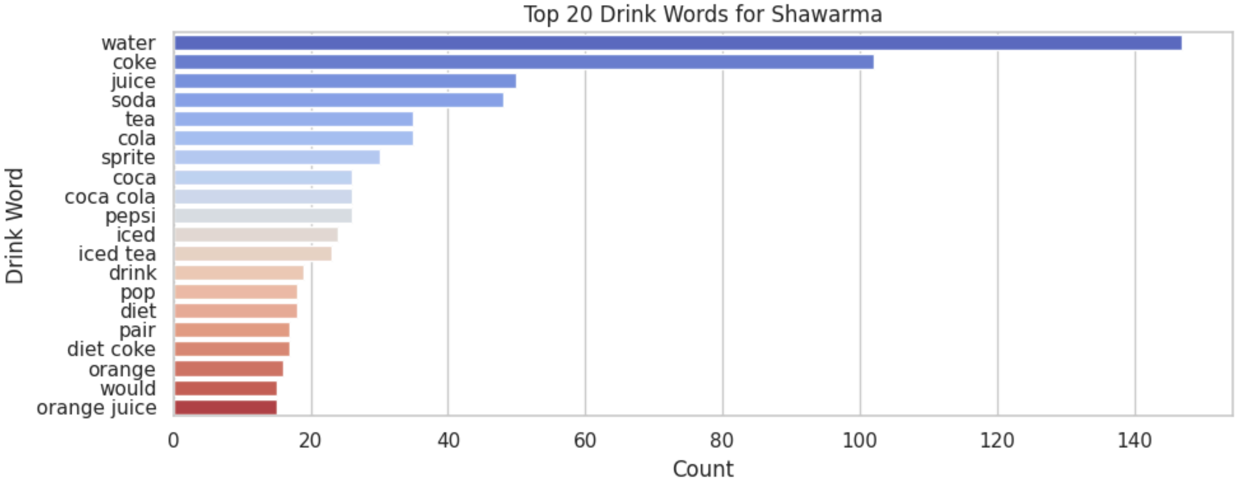
Top Movie Keywords for Shawarma

This is similar to the previous, where pizza and shawarma have extremely similar keywords. They differ heavily from the sushi keywords. While there is some uniqueness with pizza/shawarma it is not enough to warrant applying it to our model.

Feature 6: Drink


Top 20 Drink Words for Pizza


Top 20 Drink Words for Sushi

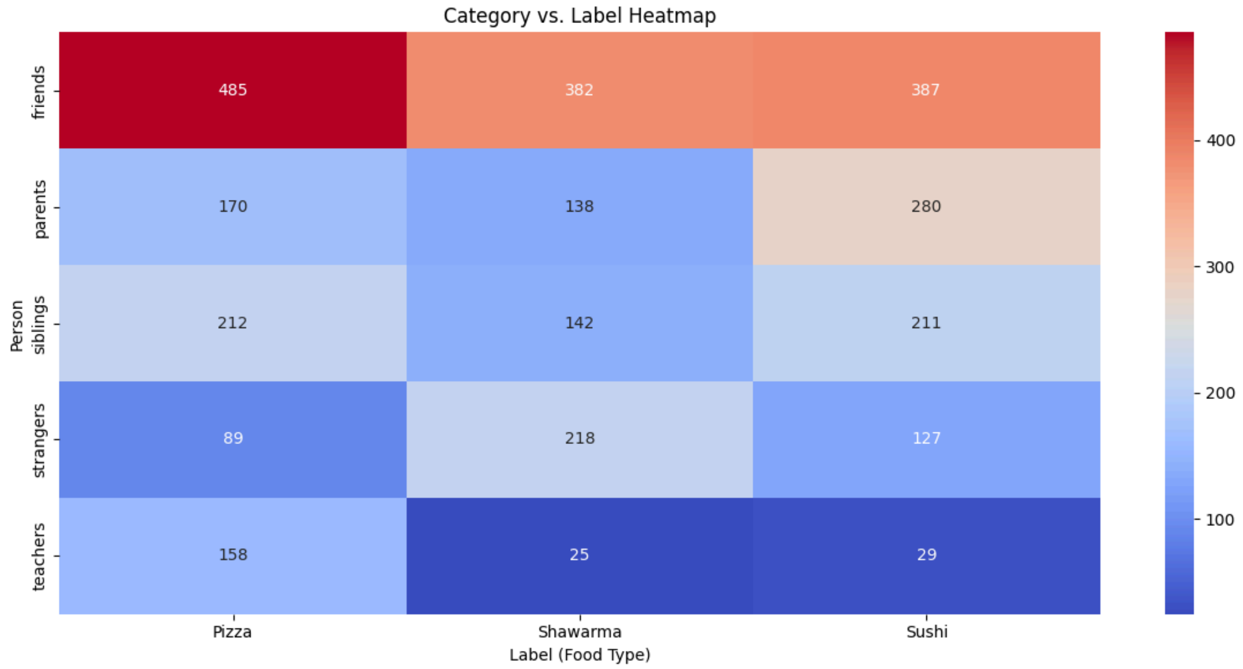

Top 20 Drink Words for Shawarma

Findings:

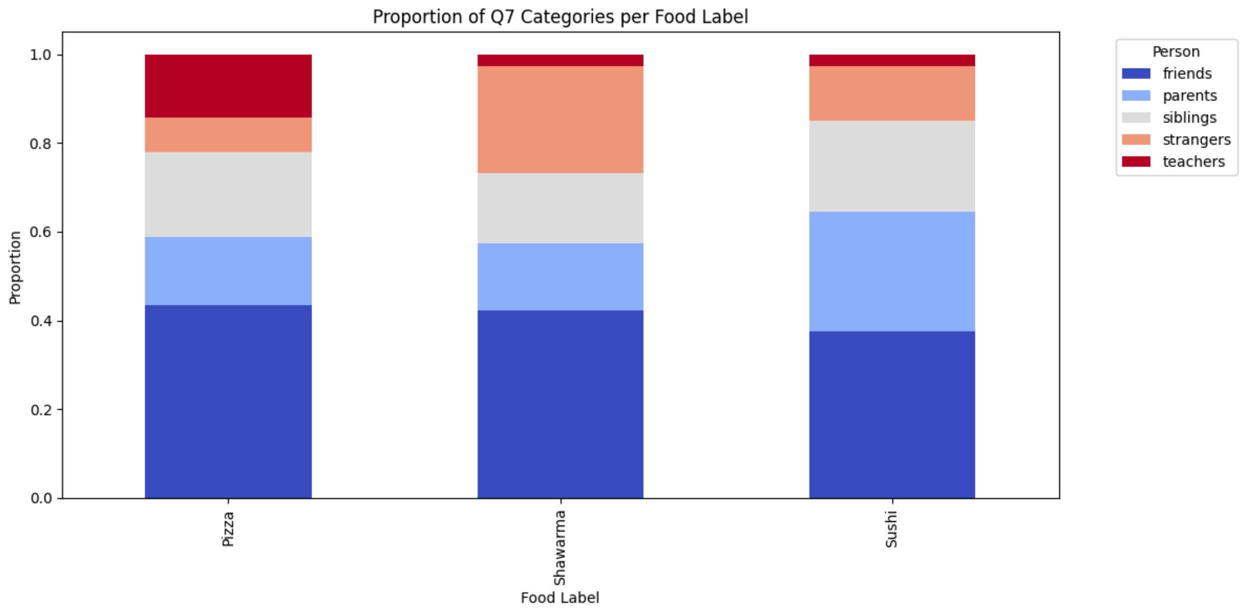
- Water was picked for all food categories and simply serves as noise. Can be removed

- Tea was consistently picked for Sushi followed by Shawarma. Appears very low in Pizza. Can be used to distinguish between Sushi/Shawarma and Pizza
- Sodas such as coke and juices were strongly associated with Shawarma and Pizza while drinks such as Saki and Miso soup were associated with sushi

Feature 7: Relationship



Similar to feature 3, we have a heat map to display the number of votes that each food item received for each relationship. We see that people universally agree that all food items remind them of friends, so nothing significant can be found from the “friends” category. However, we notice that each food item has one relationship that appears more often than in others, which can imply that these relationships correlate to some food item. For more visual clarity, we made a proportional bar graph below.



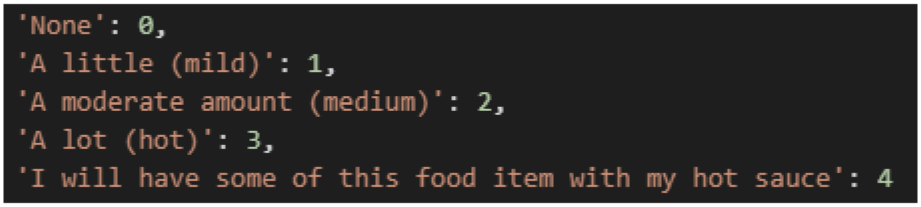
Note that the colours in the heat map do not correlate with the colours in the bar graph. In this bar graph, we observe the following:

- A larger proportion of people think of teachers when they think of pizza
- A larger proportion of people think of strangers when they think of shawarma
- A larger proportion of people think of parents when they think of sushi
- Siblings were consistently picked for all food options at the same rate
- Friends appeared more in shawarma and pizza than sushi

This suggests that we can rely on these relationships to improve our confidence on a model prediction's accuracy.

<u>Feature 8</u>: Spiciness

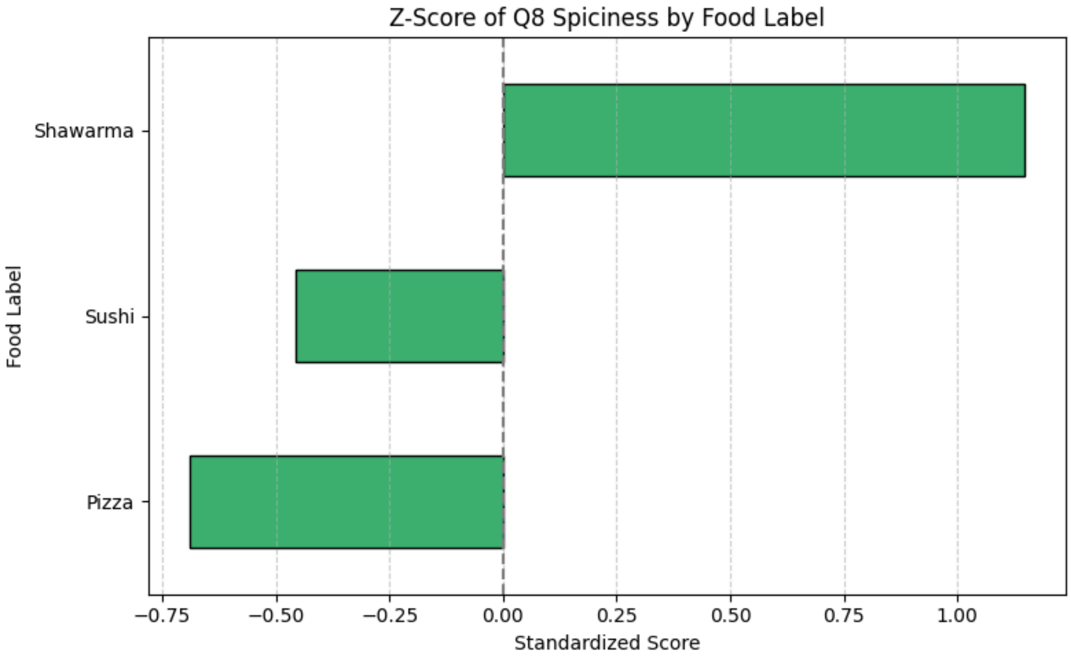
This chart below shows the mean level of spiciness for each food item (ranked from 0 to 4) ordinally based off the following order:



```
'None': 0,
'A little (mild)': 1,
'A moderate amount (medium)': 2,
'A lot (hot)': 3,
'I will have some of this food item with my hot sauce': 4
```

Shawarma	Sushi	Pizza
1.972	1.650	1.603

Although the mean values are similar, if we are to standardize the mean values, we get the following chart showing the Z-scores of each food item's spiciness level.



From this graph, we can infer that a higher spiciness level strongly implies the food item being shawarma, and lower spiciness level implies one of the other two food items. Thus, we can correlate the high levels of spiciness with shawarma and low levels of spiciness with sushi/pizza. However, more features would be required to distinguish between sushi and pizza.

<u>Input Features for Model</u>

In order to determine our input features, we do the following:

- Complexity:
  - we left the values from the dataset as-is
- Ingredient count:
  - Extract Numerical value (max)
- Ingredients:
  - Extract Ingredient words, exclude stop words and numerical values, then apply BOW
- Meal setting: we separated each category into its own binary feature, for a total of 6 features
- Price:
  - Extract numerical value (max)
- Movie title:
  - Extract movie name and apply BOW
- Drink:
  - Remove water and stop words, then apply BOW

- Relationship: similar to the meal setting feature, we separated each category into its own binary feature, for a total of 5 features
- Spiciness: we converted the categories of this feature into numerical values, with 0 indicating the lowest level of spiciness “None”, and 4 indicating the highest level of spiciness “I will have some of this food item with my hot sauce”

With these input features, we can represent a data point as a fixed-length vector in R^D, where D is the total number of input features. In this case, we calculated our D to be 405.
The majority of the input is the BOW representation, which has a length of 389.

NOTE: Additional movie information experimented with was not included due to additional DB lookup on prediction time taking too long (1 prediction / second limit in handout)

Data Splitting
Similar to the dataset splitting found in Lab04, we used the train_test_split() function to split the data at 20% for testing, 16% for validation, and 64% for training.

Model
We explored three families of models, those being: decision tree, kNN, logistic regression.

Decision Tree
We considered the following features for our decision tree model: complexity, ingredient count, meal setting, relationship, price, spiciness. We decided to leave out the “movie” and “drink” features as we wanted to keep the model simple during the beginning of our exploration; after discovering that our simple logistic regression model performed the best, we decided to focus our efforts on that model instead. There were also no adjustments made to the features since normalization does not impact the performance of a decision tree

kNN
We considered the following features for our decision tree model: complexity, ingredient count, meal setting, relationship, price, spiciness. Similarly to our decision tree model, we left out the “movie” and “drink” features for the same reason. Following the steps in the Lab01 notebook, we normalized the numerical features (complexity, ingredient count, price, spiciness) to ensure features with larger values (ingredient count and price) do not disproportionately affect the outcome.

Logistic Regression

We included the features: complexity, ingredient count, meal setting, relationship, price, spiciness, and bag-of-words (BOW) features from the text responses. These included the movie title, genres, production countries, spoken languages, keywords, drink, and more. We used CountVectorizer to convert text to features, and fuzzy matching to map user-written movie titles to real movies from a cleaned dataset. No normalization was needed since logistic regression handled sparse and numeric features fine. We used softmax regression and tuned the learning rate using a validation set.

Model Choice and Hyperparameters

Model Choice

In order for us to test each model using a consistent test set, we decided to call the train_test_split() function with the parameter "random_state=1". This will ensure that the function splits the dataset in the same way.

We also maintained the same percentage of data used in training/validation/testing sets as specified in the "Data - Data Splitting" section above; we assigned 20% of the data to the testing set and 80% to the training and validation sets. Among the remaining data, we assigned 20% of it to the validation set and the remaining 80% to the training set. This gives us a total percentage of data in the training/validation/testing sets at 64%/16%/20%, respectively.

Evaluation Metrics

We mainly focused on test and validation accuracy as a general metric across all models that we have explored. However, for our logistic regression model, we included validation loss as a metric to track as well.

Hyperparameters

In order to determine which model we've explored gives us the highest test accuracy, we must fine-tune each model's respective hyperparameters:

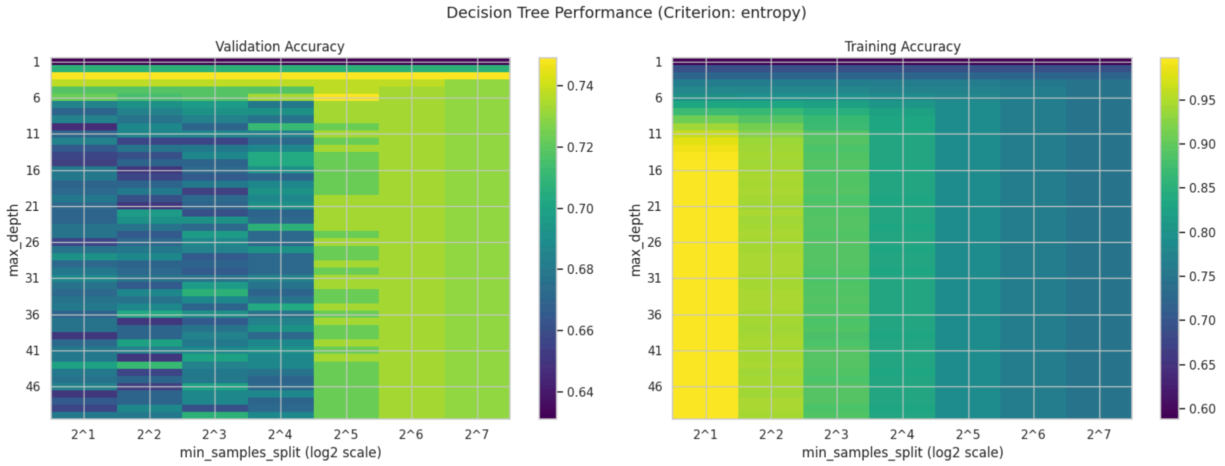
1. Decision Tree

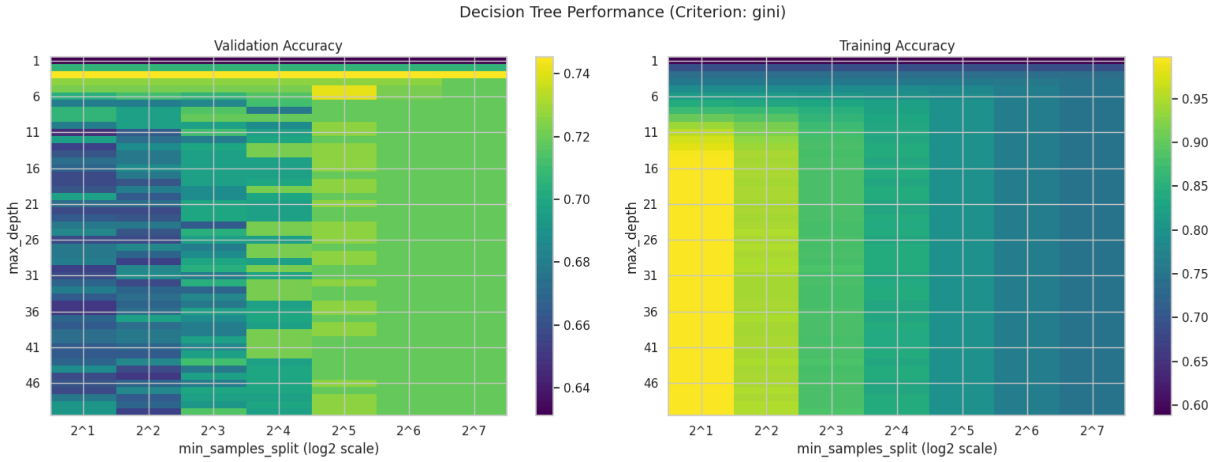
In our decision tree model, we experimented with the following:

- Criterions: entropy, gini
- Maximum depth: 1-50
- Minimum samples split: 2-128 (powers of 2)

Since our dataset contains a total of 1644 data points, we wanted to test our maximum depth and minimum samples split hyperparameters with values that will not clearly overfit/underfit our model.

Using these ranges of hyperparameters, we get the resulting heatmap for our model:





In addition to the heatmaps, we logged the validation accuracy of each combination of hyperparameters and were able to achieve the following accuracy:

Criterion	Max depth	Min samples split	Validation accuracy
Entropy	3	2	0.7490
Gini	3	2	0.7452

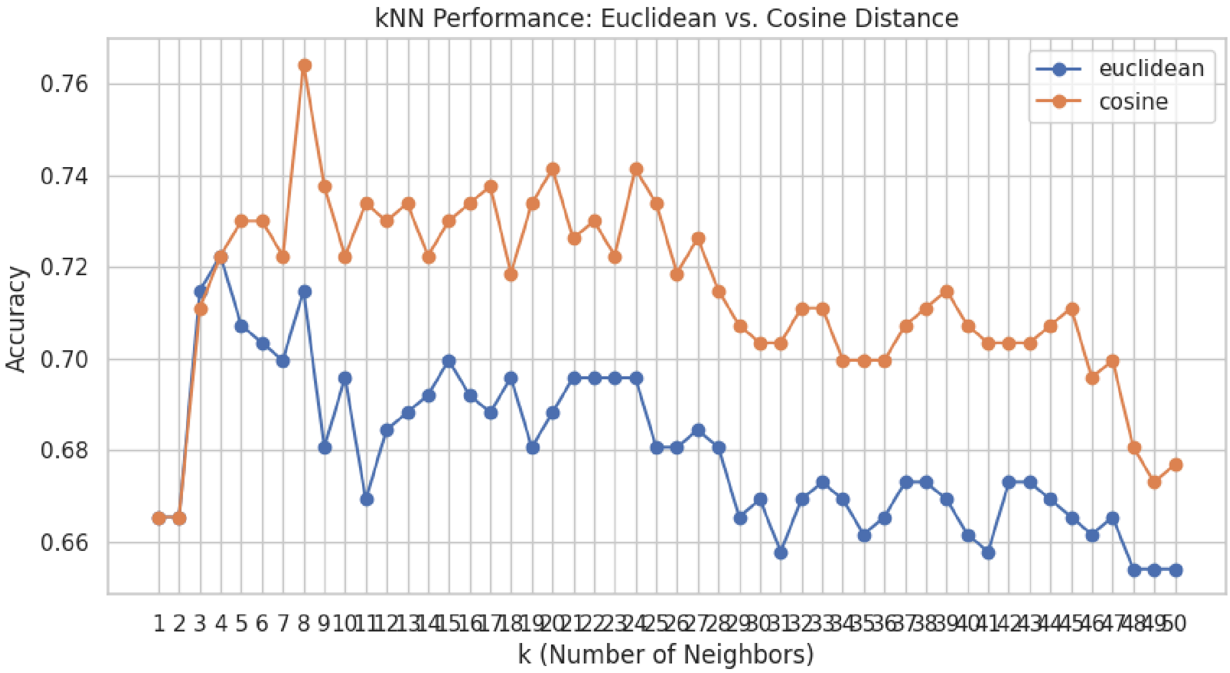
We see that our decision tree performs the best using the criterion “entropy”, with a maximum tree depth of 3 and minimum samples split of 2. If we were to use these hyperparameters to predict our test set, we get a test accuracy of 0.7143.

2. kNN

In our kNN model, we experimented with the following:

- Metric: euclidean distance, cosine similarity
- k-value: 1-50

We also decided to stop our tuning of k-value at 50 since calculating 50 nearest neighbours is clearly underfitting. However, we just wanted to calculate a high enough k-value to show a trend in the decline of accuracy as k increases.
Using these ranges of hyperparameters, we get the resulting line graph for our model:



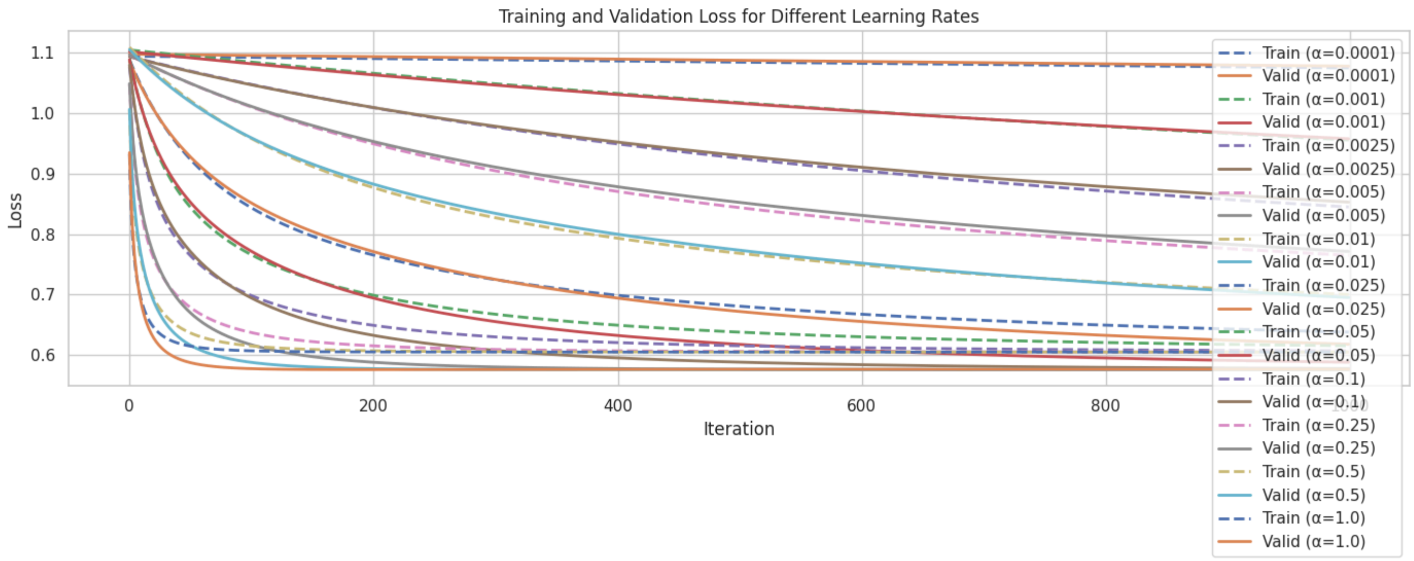
From this line graph, we can observe the highest validation accuracy to be 0.764, at a k-value of 8 using the cosine similarity metric. If we were to use these hyperparameters to predict our test set, we get a test accuracy of 0.748.

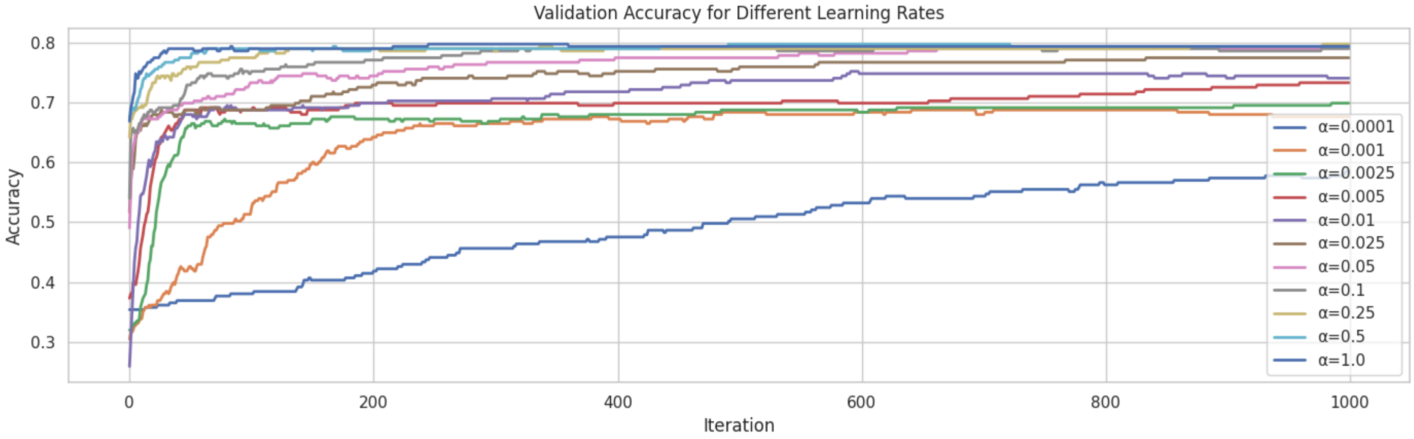
3. Logistic Regression

In our logistic regression model, we are primarily concerned with our learning rate α. Thus, we are testing on the following α-values:

- α: 0.0001. 0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1

We wanted to capture all commonly used values between the two chosen endpoints. Furthermore, we decided to stop testing the α-value at 1 since values greater than 1 may cause the gradient descent to overshoot the minimum loss.





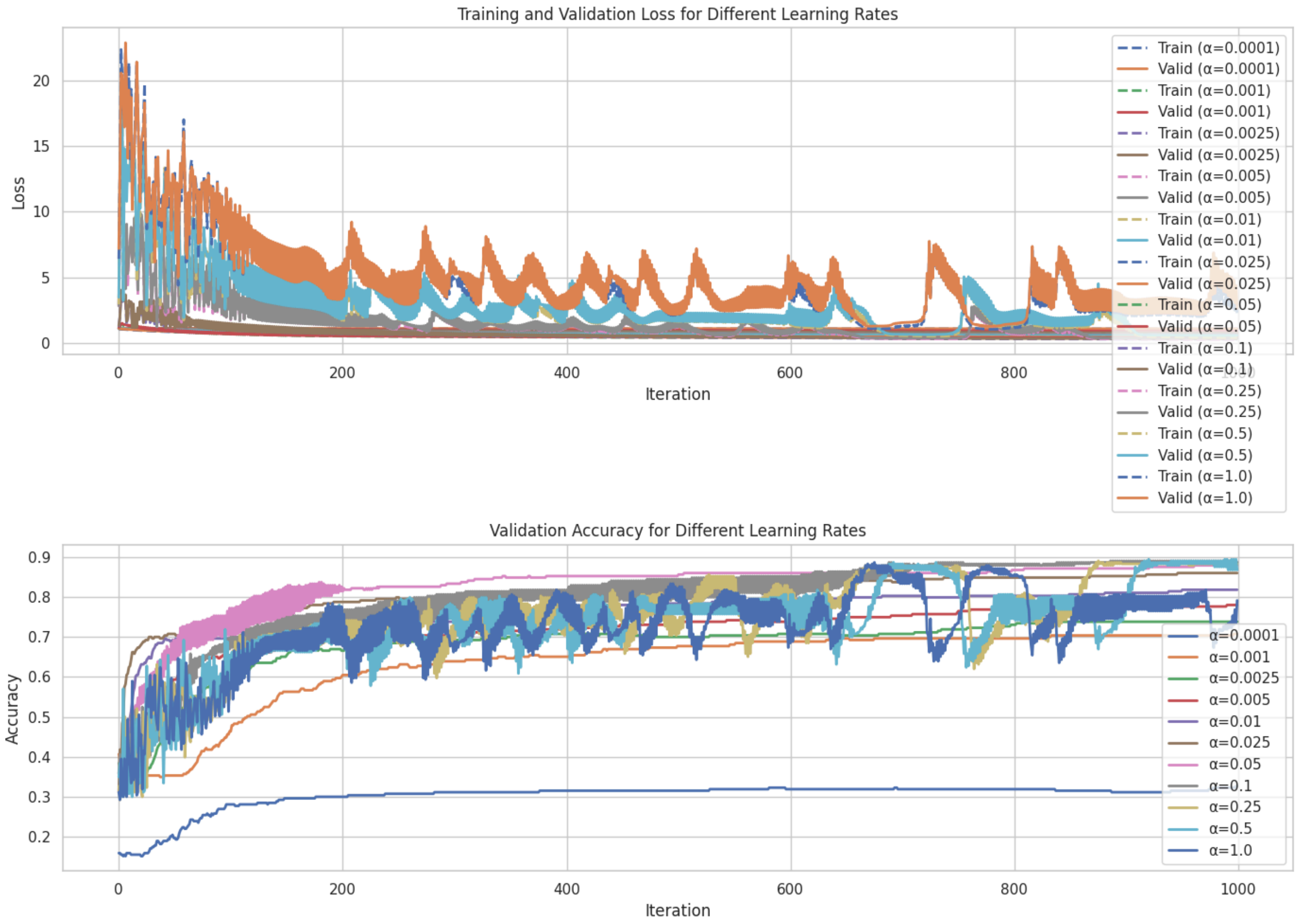
After training for 1000 iterations on each α-value, we are able to log the following data:

α-value	Validation accuracy
0.0001	0.5856
0.001	0.6768
0.0025	0.6996
0.005	0.7338
0.01	0.7414
0.025	0.7757
0.05	0.7909

0.1	0.7909
0.25	0.7985
0.5	0.7947
1	0.7947

From this log, we see that the best α-value is 0.25, giving us a validation accuracy of 0.7985. If we were to use this hyperparameter to predict our test set, we get a test accuracy of 0.8146.

Since we can clearly see that the simple logistic regression model performed the best among the three, we decided to explore more about the model by including bag-of-words features. Using the same set of α-values to tune our hyperparameter, we get the following:



After training for 1000 iterations on each α-value, we are able to log the following data:

α-value	Validation accuracy
0.0001	0.3270
0.001	0.7072
0.0025	0.7376
0.005	0.7795
0.01	0.8175
0.025	0.8593
0.05	0.8783
0.1	0.8859
0.25	0.8821
0.5	0.8669
1	0.7909

From this log, we see that the best α-value is 0.1, giving us a validation accuracy of 0.8859. If we were to use this hyperparameter to predict our test set, we get a test accuracy of 0.8997.

Final Model Choice

After testing each model, we found that our logistic regression model performed the best. Thus, we decided to implement our logistic regression model using bag-of-words. The model will use weights trained during our data exploration from “weights.npy”.

A stopwords set was used containing a few basic English words (such as “the”, “and”, “with”) which simply serve as noise*

*Note: Further explained below

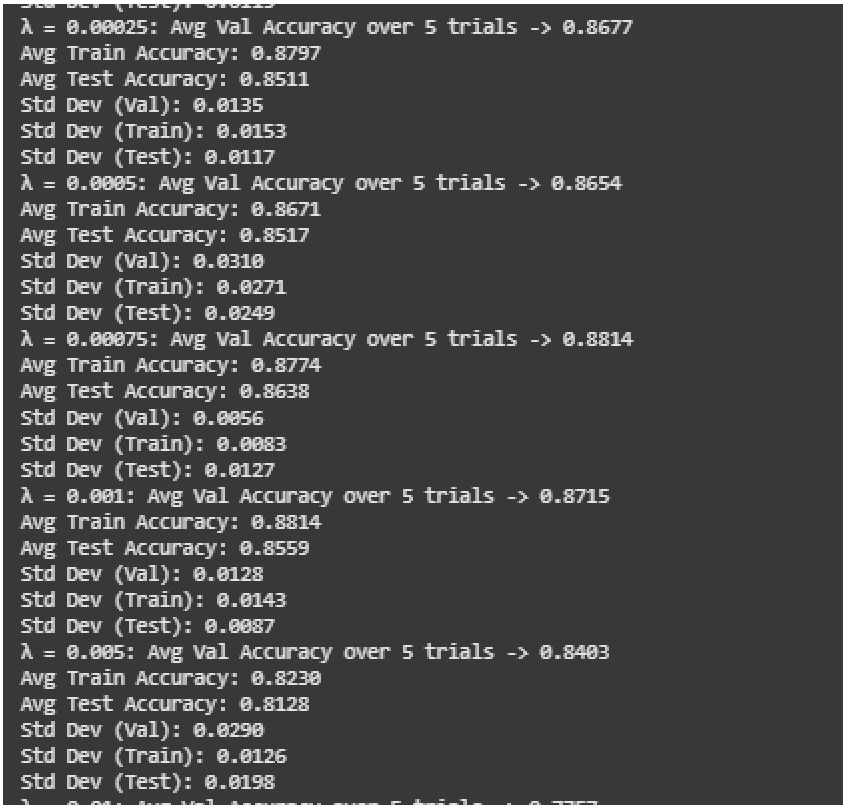
The textual categories used are “ingredients”, “drink”, and “movie title” which were cleaned by removing numbers and stopwords, then assigning values of number of appearances for words that appear that correspond to a dictionary used (stored in the file “vocab.csv”), and 0 for all other words.The file is stored to ensure a consistent input vector for the function on prediction time.

Numerical features were put side by side in a vector, next to the categorical vector, then combined with the BOW vector to produce the final input vector with space

Normalization was experimented with by calculating the mean and standard deviation of each variable, then storing them externally to use for each variable. However, our findings were that this brought down prediction accuracy significantly. Further experimentation is needed to determine why this was the case, so we chose to leave it out of the final model. A potential reason may be the combination of a dense vector (numerical and feature vectors) with a sparse vector (BOW). The model accuracy is high in the validation.

Two techniques for regularization of the model were applied: L2 Regularization and removing uncommon words from vocab.

L2 Regularization:



```
Std Dev (Test): 0.0113
λ = 0.00025: Avg Val Accuracy over 5 trials -> 0.8677
Avg Train Accuracy: 0.8797
Avg Test Accuracy: 0.8511
Std Dev (Val): 0.0135
Std Dev (Train): 0.0153
Std Dev (Test): 0.0117
λ = 0.0005: Avg Val Accuracy over 5 trials -> 0.8654
Avg Train Accuracy: 0.8671
Avg Test Accuracy: 0.8517
Std Dev (Val): 0.0310
Std Dev (Train): 0.0271
Std Dev (Test): 0.0249
λ = 0.00075: Avg Val Accuracy over 5 trials -> 0.8814
Avg Train Accuracy: 0.8774
Avg Test Accuracy: 0.8638
Std Dev (Val): 0.0056
Std Dev (Train): 0.0083
Std Dev (Test): 0.0127
λ = 0.001: Avg Val Accuracy over 5 trials -> 0.8715
Avg Train Accuracy: 0.8814
Avg Test Accuracy: 0.8559
Std Dev (Val): 0.0128
Std Dev (Train): 0.0143
Std Dev (Test): 0.0087
λ = 0.005: Avg Val Accuracy over 5 trials -> 0.8403
Avg Train Accuracy: 0.8230
Avg Test Accuracy: 0.8128
Std Dev (Val): 0.0290
Std Dev (Train): 0.0126
Std Dev (Test): 0.0198
λ = 0.01: Avg Val Accuracy over 5 trials -> 0.7757
```

- An extremely small value for lambda produced the best result, however values after that significantly reduced validation accuracy. This could mean our model was already doing a good job at not overfitting. Our final model used a learning rate of 0.01 and an L2 Regularization lambda of 0.00075

Reducing Vocab:

- Vocab was restricted by only including words that appeared more than 3 times. The logic behind this was that one respondent answered the survey once for each food item, so if more than one respondent included this word, it would be statistically significant
- This reduced our vocab from 1547 terms -> 389 terms, a reduction of ~75%, significantly lowering model complexity while not sacrificing accuracy
- The other technique discussed previously in this section was excluding stopwords, common english words which have no impact on the label, and the drink “water”. Careful deliberation was used for words which may relate to movies. Another important note is that this filtered out the “no” or “none” responses some respondents gave in response to what drink or movie they think of, reducing noise.

The model applies cross entropy loss and a softmax activation function, which are optimal for logistic regression as discussed in the lecture.

Prediction

After finalizing our model, we have reached ~85% accuracy on the validation set, as well as an ~86% accuracy on our test set, with std deviations of ~0.005 and ~0.01 respectively. This means our model generalizes well, since our standard deviations are fairly low. We are expecting our model to perform similarly on the unknown test set.
Final prediction: 85%

Workload Distribution

Amr:

- Assisted in data analysis
- Finding splitting features from forest
- Experimented with movie dataset
- Implemented logistic regression using bag-of-words
- Finalized model encodings to input
- Assisted in writing report
- Testing predictions and fine tuning model
- Finalized model
- Regularization tuning

Jackson:

- Assisted in data analysis
- Implemented decision tree, kNN, and simple logistic regression models
- Assisted in writing report
- Implementing logistic regression and predictions
- Learning rate tuning