**CSC311 Project Report**

Thomas Semczyszyn

Qianren Pan

Alan Wang

Albert Li

**Data Exploration**

For Question 1, the distribution of responses for the complexity of each food item is displayed below. Sushi had about half of responses as “4” or “5, indicating it was generally viewed as the most complex. On the other hand, about half of Pizza responses were “3”, suggesting it was viewed as the least complex of the foods.

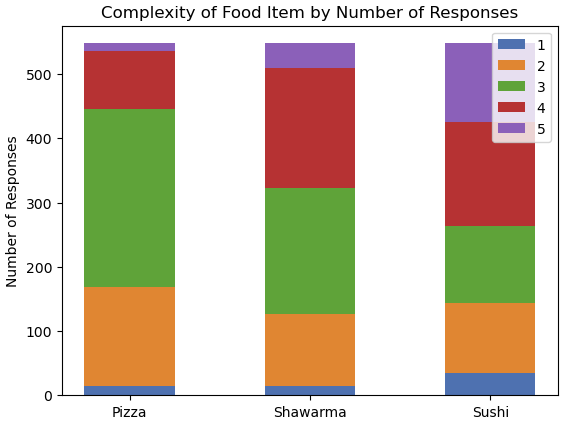


Figure 1: Complexity of Pizza, Shawarma & Sushi

For Question 2, the number of ingredients didn’t seem to provide much information about the food item, except for many of the sushi responses reporting relatively few ingredients.

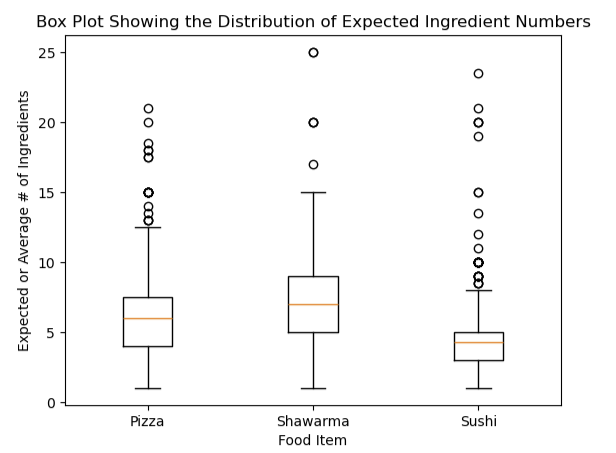


Figure 2. Average Number of Ingredients in Pizza, Shawarma & Pizza

For Question 3, we generated a heatmap (shown below) to analyze the relationship between “Food Setting” and the target food items. The results align with common expectations. “Pizza” is strongly associated with the “At a party” setting, reflecting its popularity as a social and easily shareable food. “Shawarma” appears most frequently in the “Week day lunch” category, which aligns with the presence of numerous Shawarma food trucks near the UofT campus, catering to students grabbing a quick meal between classes. “Sushi” is more commonly linked to “Week day dinner” and “Weekend dinner,” likely due to its lighter nature, making it a preferred option for an evening meal that is easier to digest.

These clear trends suggest that the “Food Setting” feature is highly correlated with the target food labels. Since different food items are strongly associated with specific meal times or social settings, this feature provides valuable information for classification. This strong correlation indicates that incorporating “Food Setting” into our model could significantly improve the prediction accuracy.

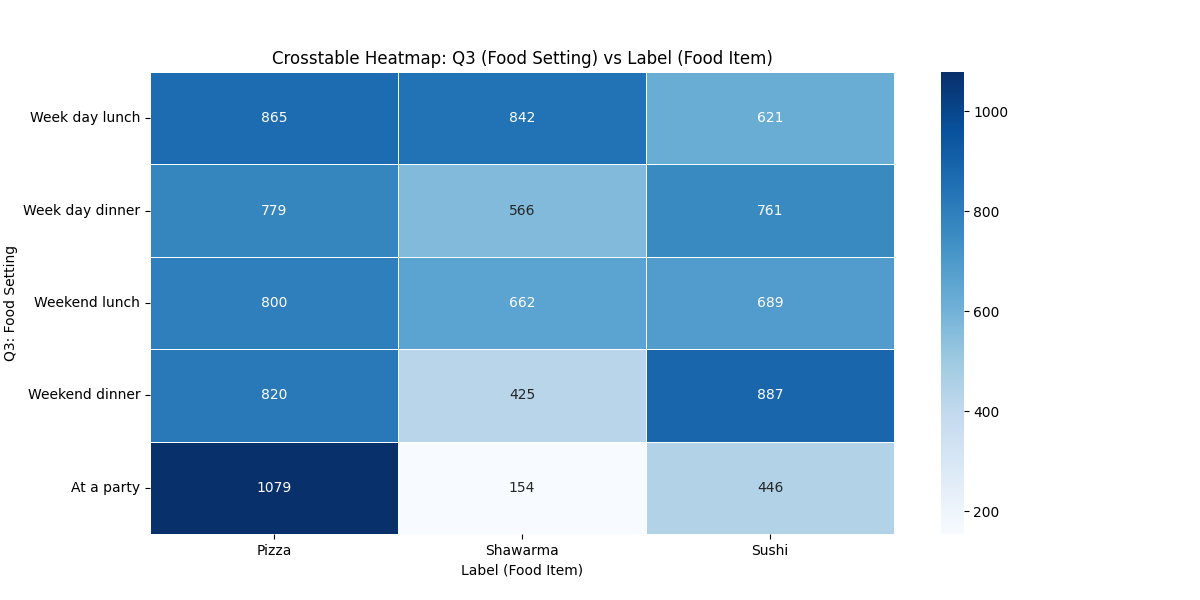


Figure 3: Heatmap of Food Settting vs Food (pizza, sushi, shawarma)

For Question 4 responses, we generated a box plot for each food item for the interpreted price (the price generated after parsing the responses in data preprocessing). Shawarma clearly had the lowest variation in price responses, while sushi varied considerably (which makes sense, as sushi has more high-end dining options). (The 100 response was priced in yen, but we found it fitting to keep as it was consistent with the pattern of some sushi being expensive). It makes sense that both sushi and pizza vary considerably, as the difference in interpretation of serving size (piece/slice vs. whole pizza, roll/dining experience) contrasted with shawarma (whose order sizes don’t vary as much).

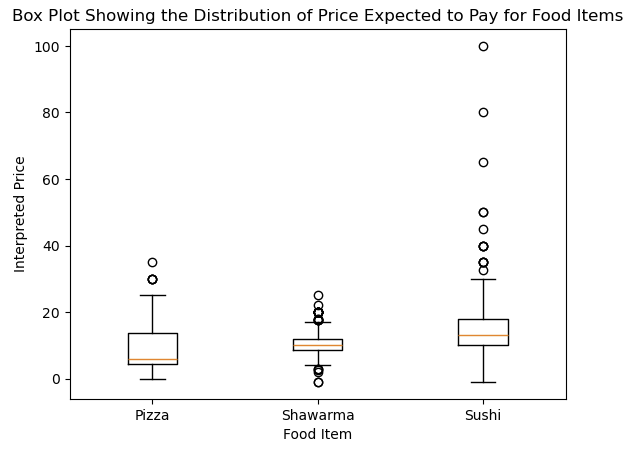


Figure 4: Distribution of Price for Pizza, Shawarma & Sushi

For Question 5 responses, we generated frequency tables for each food item. Unlike the model, we created a new dataframe and only converted responses to lowercase, so movies spelled differently can be repeated. Shawarma was especially concentrated on one movie or movie franchise, the Avengers, while both Pizza and Sushi had some fairly common responses. Overall, not many movies have more than 6 responses. For these most common responses, there does not seem to be overlap (one movie being included in more than one list, implying movies can be particularly useful). However, given the total number of data points (1644, divided into 3 for each item), it seems the majority of responses indicate a movie outside of the top 10 results, so other features likely provide greater predictive power for these other data.

Figure 5: Most Common Movie Responses for Pizza

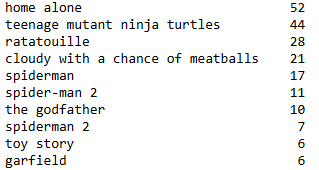
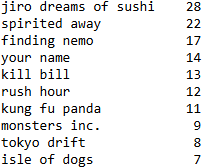


Figure 6: Most Common Movie Responses for Shawarma

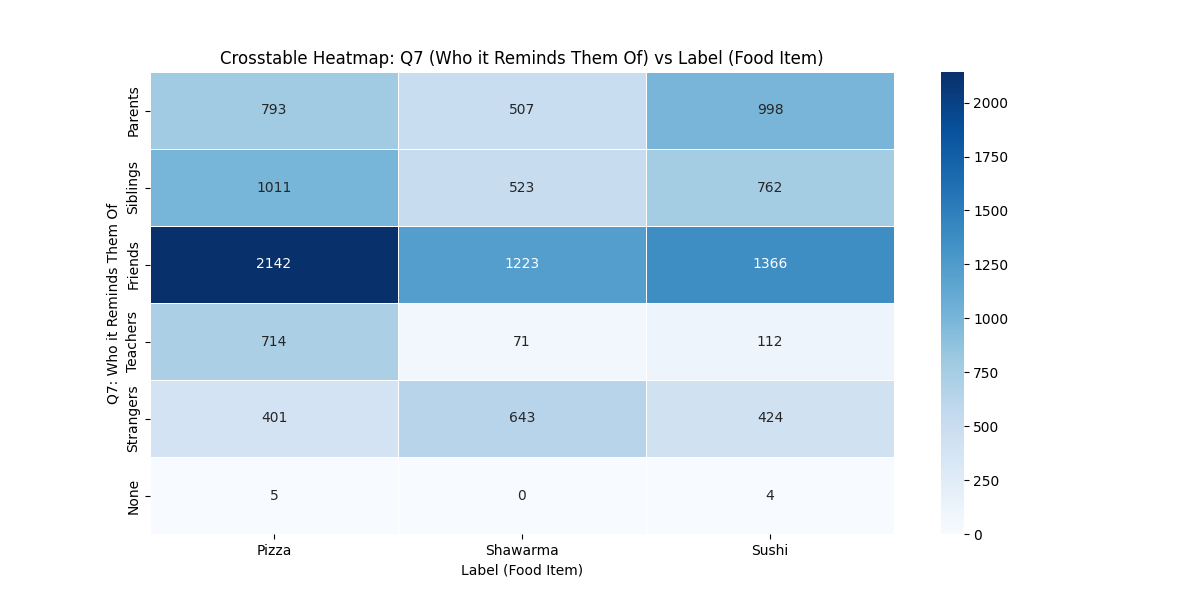


Figure 7: Most Common Movie Responses for Shawarma



For Question 7, we generated a heatmap to analyze the relationship between “Who it Reminds Them Of” and the target food items. The results show a strong association between “Pizza” and the categories “Friends” and “Teachers.” Specifically, the number of entries for “Friends” is nearly double for “Pizza” compared to “Shawarma” and “Sushi.” The connection to “Teachers” is even more pronounced, with “Pizza” having almost ten times the number of entries compared to “Shawarma” and nearly seven times that of “Sushi.”

This trend can be explained by the nature of pizza as a social and widely shared food, often associated with gatherings, school events, and parties—settings where both friends and teachers might be present. Since other food items do not share this strong social connection, “Friends” and “Teachers” emerge as key indicators for identifying “Pizza.” Given these strong patterns, this feature appears to be highly correlated with the target labels and could serve as a useful predictor in our classification model.

Figure 8: Heat Map of Who People Associate the Food With vs Food (pizza, shawarma, sushi)

For Question 8, the distribution of the amount of hot sauce preferred with each food is displayed below. (Where “0” indicates None, “1” indicates a little, “2” indicates a moderate amount, “3” indicates a lot, and “4” indicates more hot sauce than food is preferred.) The distribution makes sense, as shawarma is commonly served with spicy sauce, while sushi is sometimes served with wasabi (which some may interpret as a hot sauce).

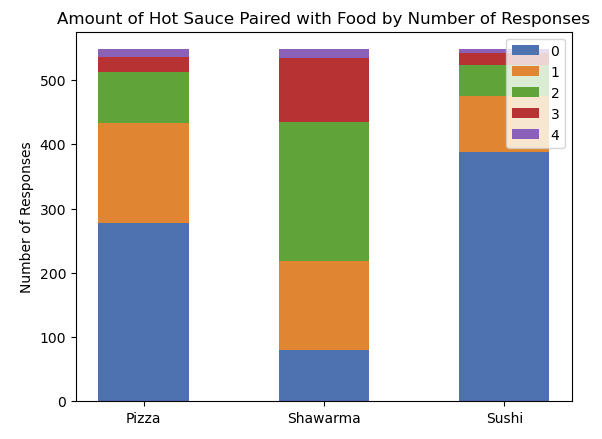


Figure 9: Amount of Hot Sauce Paired with Pizza, Shawarma & Sushi

The initial dataset consists of 1,644 responses with 10 columns corresponding to the index, target and the 8 survey questions which were both open and closed-ended. The target variable is the food item name/label (sushi, shawarma, or pizza).

**Feature Selection**

We felt that generating 8 features from 8 survey questions was insufficient for the purposes of classification: the complexity and scope of the data implied partitioning responses to 8 questions into a larger number of features to differentiate and extract more specific categories of information. For Q2 (“How many ingredients would you expect this food item to contain?”), we extracting four main features. The first two were numerical: an estimated average of the response and an estimated range (e.g. “3 to 8” would be parsed as an average of 5.5 and a range of 5). We added two additional features indicating whether or not the response contained a key word or phrase indicating a stated minimum or maximum number of ingredients (e.g. “3+” or “3 or more”). Finally, we created a considerable number of indicator variables meant to extract the names of ingredients or descriptions from lists (e.g. “flatbread”, “italian”).

We applied similar logic to fully open-ended questions asking respondents about associated movies and food-drink pairings by creating indicator variables from a list of keywords based on the data. For example “sake” for Q6 responses indicated the label was sushi, which makes sense for a Japanese alcohol. “ninja turtle” and “tmnt” both sought to capture responses related to Teenage Mutant Ninja Turtles, whose eponymous creatures adore pizza. Rather than using the full movie name as a keyword, we selected part of the name to minimize effects from typos and abbreviations.

For Q3 and Q7, where the questions asked respondents to check boxes, we created indicator features for each individual box. This allows for greater flexibility in weighting the question information than only having one feature that differentiates based on each combination. Moreover, it ensures that combinations that were unreported in the dataset could still provide predictive power.

**Data Representation**

To enable our models to understand the data, we had to perform thorough preprocessing of the responses. The data was entered using various methods, including text, numerical input, and categorical values in checkboxes, requiring different preprocessing approaches for each question. Additionally, there was variability in the formatting users used to answer, so we had to carefully consider how to extract the key information we needed. We ensured that None or NaN responses were properly handled to ensure the prediction script did not result in errors, and converted all text responses to lowercase to ensure key words or phrases were matched. We also removed the original Q2-Q8 responses to be replaced by their processed analogues.

Question 1 asked the user to rate the complexity of the food on a scale from 1 to 5, and since the form restricted the user to enter an integer in this range, we didn’t have to do any preprocessing and we could simply interpret these values.

Question 2 asked the user to state how many ingredients they would expect the provided food item to contain, and this question took in text input. Due to the lack of input constraints, users used many different formats for the input. We had to develop various strategies to differentiate responses: numbers such as “13” represented the simplest case, while ranges (e.g. “three to eight” were averaged. We converted many numbers represented as text to their numerical forms, e.g. “two” became 2. We made an exception for “one”, as not only did we think it strange for someone to reply that any of the food items had only one ingredient, but that the word was used in different contexts (e.g. “one could also add various toppings”).

For cases that did not include a number, we sorted them into two categories: those that were comma-separated lists containing more than one comma (parsed as the number of commas plus one), and those that did not. We realized it was difficult to systematically count these other cases, so we simply counted the number of words (accurate for single-word space-separated lists of ingredients). However, this system was flawed for lists with single ingredients that were more than one word or lists with explanations and with neither numbers nor commas.

Question 3 asked users to select the setting where they would expect the food to be served, and since this data was obtained through a check box form. Besides the question response that provided a list of settings, we created new indicator features for each setting to better capture distinct patterns associated with a particular setting (as opposed to only differentiating based on the entire list of settings). We also ensured that the data could appropriately process responses where no boxes were checked.

Question 4 asked the users how much they would expect to pay for one serving of this food item. This data was processed in a similar to question 2 to obtain both a single integer and range data from text responses. However, instead of searching for and counting list sizes, we instead use a simplified currency conversion (multiply the answer by 1.5) if the response indicates the cost is in USD, Euros, or Pounds.

Question 5 asked what movie respondents thought of in response to the food item. We created a [vocabulary of keywords](https://docs.google.com/spreadsheets/u/0/d/1fnPPT3sDOkALGgDDcp6EiFr0twV25-yXJOqU9vTyEvQ/edit) so a response consisted of a row of 0s and 1s indicating whether a keyword (part of a movie name) appeared in a given response.

Question 6 asked what drink the respondent would pair with the food item. We employed the same strategy as in Question 5.

Question 7 asked what kinds of people the food item reminded respondents of. Respondents were asked to check boxes, so the number of responses was limited to a relatively small number of strings. We used the same process as Question 7 to create individual features for each group of associated people.

Question 8 asked how much hot sauce the respondent would add to the item, selecting from five descriptions increasing in magnitude. As stated previously, we added a feature to rank these on a numerical scale from 0 to 4 (which could help our decision tree in separating responses with no/a little hot sauce and responses indicating a moderate or greater amount of hot sauce).

We used one-hot encoding to represent the label of the food.

Our data input had a model size of 1644 by 435 (responses by features, with two additional columns for labels and response IDs).

**Data Splitting**

We split the dataset into three sets: a training set composed of 70% of the data (1644 responses), and validation and testing sets composed of 15% of the data each. We randomized the order of the data to ensure each data set’s proportion of pizza, shawarma, and sushi responses would be relatively even. For the purposes of validation and hyperparameter tuning, we randomized the sets for each iteration to ensure our final model was relatively robust. Since the amount of data was somewhat small, we decided to weigh both the test and validation sets a fair amount to ensure sufficient data for each of the three classes.

**Model Exploration**

Our group decided to explore 3 families of models: neural networks, naive bayes, and random forests. Each of these families of models offers distinct strengths and trade-offs, making them valuable for our analysis.

**Neural Network**

The first model we examined was neural networks. Neural networks excel at learning complex patterns and are well-suited for modeling non-linear relationships. They can automatically extract meaningful representations from raw data, and since we do not need to interpret the specifics of the model, we do not need to have a deep understanding of how each weight affects the neural network.

For our model, we implemented a feedforward neural network with two hidden layers, and used the ReLU activation function. We used this function as it generalizes well and is easy to optimize with gradient-based methods. For the output layer, we used the softmax activation function, as it works with multi-class classification problems, allowing our model to predict the food item as being either pizza, sushi or shawarma. To optimize the network, we tuned several hyperparameters, including the number of hidden units, the learning rate, the number of epochs, and the batch size. We also used gradient descent with backpropagation to adjust the network’s weights and ensure the predictions improved over time. Overall, we were able to obtain a validation accuracy of over 91%, with a test accuracy of about 85%.

**Naive Bayes**

Naive Bayes is a probabilistic classifier based on Bayes’ theorem, assuming that features are conditionally independent given the class label. Despite this strong independence assumption, it is often effective for text classification tasks. Given that many of our predictors are categorical and text-based, Naive Bayes is a natural candidate for this problem.

We used the Multinomial Naive Bayes version, which is well suited for text data represented as term frequency vectors. Our dataset contains several categorical responses, such as the complexity, settings where the food is typically eaten, associated movies, drink pairings and spice preferences. These features are mostly independent as knowing a drink won’t necessarily tell us more about the associated movie.

The price feature contained both numerical and text responses making it harder to interpret with models that required numerical features. But in our approach, we considered numerical data as part of the text by treating numbers as categorical values instead of continuous values. We processed them as words instead of numbers which allowed us to capture the additional information as well as fit the data to the model.

Overall, Naive Bayes was explored due to its efficiency, suitability for text heavy datasets and its simplicity. We were able to achieve an accuracy of just over 90%, but we were skeptical that the model may not generalize well to unseen data since the categorical variables were all selected from the given data.

**Random Forest**

Random Forest is a model that is based on existing data processing functions that extract features from textual responses in the survey. Throughout this ensemble learning method, multiple decision trees were built, and their outputs are combined to predict the desired labels. In short, it is a method that trains each tree of randomly selected subsets of data and then aggregates the results to make the final prediction.

For this model, some features that we used are the average number of ingredients, the average price one is willing to pay, listed ingredients, and the key words or phrases associated with a movie. The hyperparameters of our model are number of estimators, maximum tree depth, minimum sample split, and balanced class weight. For clearer and easier manipulation and observation, seeds for randomness are used for reproducibility. GridSearchCV from sklearn.model\_selection to test and optimize the best-fit hyperparameters. Some other methods that were tried on the model to enhance the accuracy are trying to add weight to movie keywords and exploring the relationships and interactions between features, such as multiplication of the two averages; however, the two methods did not work well. On the other hand, discarding the movies that only appeared once in responses marginally improved accuracy, so we kept them in. Ultimately, we were able to achieve an accuracy of 87.7% for our random forest model.

**Model Choice and Hyperparameters**

After tuning various hyperparameters for random forest and neural network models, we compared the best of each of these two types to select as our final model, as measured by the highest average validation accuracy. Given the number of hyperparameters, a comprehensive list varying all hyperparameters would be excessively long. We have displayed a small number of examples below for both our Random Forest and Neural Network. As both models took fairly minimal times to train, we were unconcerned with either model taking too long to predict.

**Hyperparameter Tuning**

Neural Net Hyperparameter Tuning

| hidden\_size | alpha | n\_epochs | batch\_size | Average Validation Accuracy |
| --- | --- | --- | --- | --- |
| 50 | 0.001 | 100 | 32 | 0.8866 |
| 100 | 0.001 | 100 | 32 | 0.8837 |
| 200 | 0.01 | 200 | 64 | 0.9163 |
| 100 | 0.0001 | 100 | 32 | 0.5081 |
| 200 | 0.0001 | 200 | 32 | 0.5772 |
| 50 | 0.001 | 100 | 16 | 0.9069 |
| 100 | 0.001 | 50 | 32 | 0.8085 |

Figure 10 - Hyper Parameter Tuning Results for Neural Network

To find the best hyperparameters for the neural network model, we tested 81 (34) different combinations of parameters, doing 10 runs per set, and determining the average validation accuracy. Some of the combinations of hyperparameters that we tested are included above in **Figure 10**.

The parameter values used in our testing include:

hidden\_size - [50, 100, 200]

learning\_rate - [0.01, 0.001, 0.001]

n\_epochs - [50, 100, 200]

batch\_size - [16, 32, 64].

The neural network parameters that gave us the highest average validation accuracy, a value of 0.9161, were obtained when using a value of 200 for the hidden size, a learning rate of 0.01, running 200 epochs and using a batch size of 64.

Random Forest Hyperparameter Tuning:

| n\_estimators | max\_depth | min\_sample\_split | min\_samples\_leaf | Validation accuracy |
| --- | --- | --- | --- | --- |
| 200 | None | 5 | 1 | 0.9053 |
| 300 | None | 5 | 1 | 0.9112 |
| 400 | None | 5 | 1 | 0.9112 |
| 400 | 30 | 20 | 1 | 0.9112 |
| 400 | 30 | 20 | 5 | 0.8935 |
| 400 | None | 20 | 5 | 0.8935 |
| 300 | 30 | 20 | 5 | 0.8876 |
| 300 | None | 2 | 5 | 0.9053 |
| 400 | 40 | 20 | 5 | 0.8935 |

Figure 11 - Hyper Parameter Tuning Results for Random Forest

To find the best hyperparameter for Random Forest, we used GridSearchCV from sklearn to automate the process of inserting hyperparameters and enumerate the trials. After a sequence of trials that were based on follow values:

'n\_estimators': [100, 200, 300, 400],

'max\_depth': [30, 40, None],

'min\_samples\_split': [5, 10, 20],

'min\_samples\_leaf': [1, 2, 5]

The optimized hyperparameters turn out to be: {'max\_depth': 40, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10, 'n\_estimators': 200}.

**Reasoning for Omitting Naive Bayes**

We decided against using Naive Bayes as our final model for several reasons. First, Naive Bayes relies heavily on the observed frequencies in the training data. Many of our features, such as movie titles and drink names are generated from the responses in our dataset. This means that if new responses include movies or drinks that weren't present in our training set, the model won’t know how to handle them, leading to poor generalization.

Second, our numerical features, like prices or the number of ingredients, suffer from sparsity. In many cases, only a few responses may show a specific value. Naive Bayes treats these numerical responses as discrete categories, which can amplify the issue of sparsity and result in unreliable probability estimates.

Lastly, the assumption of feature independence may not necessarily hold for our dataset. It makes sense that there may be some correlation between feature responses, such as the setting one associates the food item with and the people one associates the food item with. Such relationships are more complex than what Naive Bayes can capture.

Overall, while Naive Bayes is simple and effective in many text classification tasks, its inability to handle unseen features and its sensitivity to sparsity in our numerical data make it a less suitable choice for our problem.

**Final Model**

We selected the neural network to be our final model over the random forest as our best hyperparameter neural network had a validation accuracy of 0.9163, which is higher then that of the random forest having a validation accuracy of 0.9112. The neural network we are using is trained over 200 epochs with a hidden layer size of 200, a learning rate of 0.01, and uses a batch size of 64, as these were found to be the optimal hyperparameters. Our neural network has 2 hidden layers, and uses the ReLU activation function for optimization.

**Prediction**

**Performance Estimate and Rationale**

Provide a point estimate of your model’s performance on the

test set. Providing a range will earn no marks. Provide an explanation of your expected model performance on the test set. Support your explanation with empirical evidence.

We predict that our final model’s accuracy on the unseen test set will be similar to the accuracy on our test set (partitioned from the provided data), with the model. This was 89.82%, so the point estimate of our accuracy on the unseen test set is 89.82 % (a figure obtained by averaging 1000 runs of splitting and then training a neural network with the hyperparameters specified above). We deliberately included a test set in developing our model to both understand how well our model generalizes and to help ensure our model does not overfit. By ensuring the test data does not leak into the training data, we are confident that the model is not hyperfixated on the particularities of a specific set of responses. All operations including backpropogation are vectorized to optimize efficiency and reduce redundant calculations.

**Workload Distribution**

Thomas Semczyszyn: implemented the neural network and hyperparameter testing for it, wrote the neural network & final model description, worked on data representation

Qianren Pan: implemented the Random Forest and tuned the hyperparameters, wrote the model-associated parts in the report.

Alan Wang: implemented naive bayes and analysis of model choice for naive bayes, worked on data exploration.

Albert Li performed data preprocessing, creation and selection of features, most of the data exploration, and wrote most of the report’s sections related to data.