CSC311 report

Christoffer Tan, Janis Joplin, Jingwen Zhong, Tianhui Zhao

March 2025

1 Data

1.1 Exploration

In this section, we conducted a detailed exploratory analysis of survey responses to better understand feature distributions and correlations with our target labels: *Pizza*, *Shawarma*, and *Sushi*. Each question was visualized, analyzed, and evaluated using statistical tests to determine its relevance as a feature.

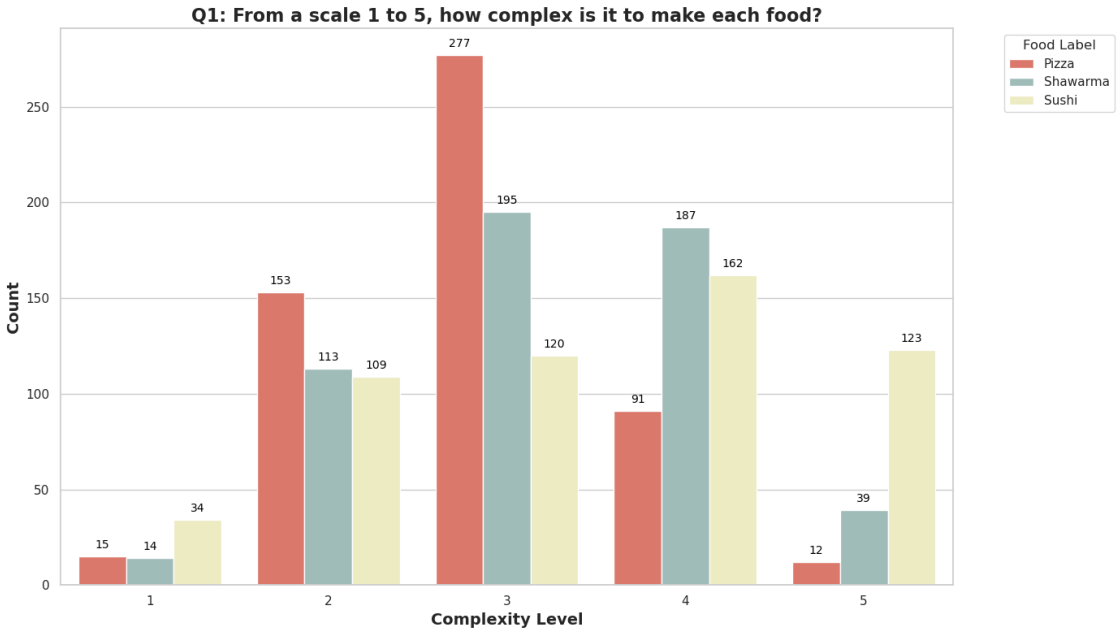


Figure 1: Distribution of perceived complexity levels across food labels. Ratings range from 1 (simplest) to 5 (most complex).

Pizza is generally perceived as moderately simple, primarily receiving ratings between 2 and 3. *Shawarma* exhibits a wider distribution of complexity ratings, typically clustered around medium complexity (ratings 3–4), indicating varied perceptions about its preparation. *Sushi* distinctly stands out with higher complexity ratings, clearly perceived as the most complex among the three. An ANOVA test confirms these observed differences are statistically significant (F-statistic = 42.19, p-value = 1.36e-18), indicating that perceived complexity is a strong distinguishing feature.

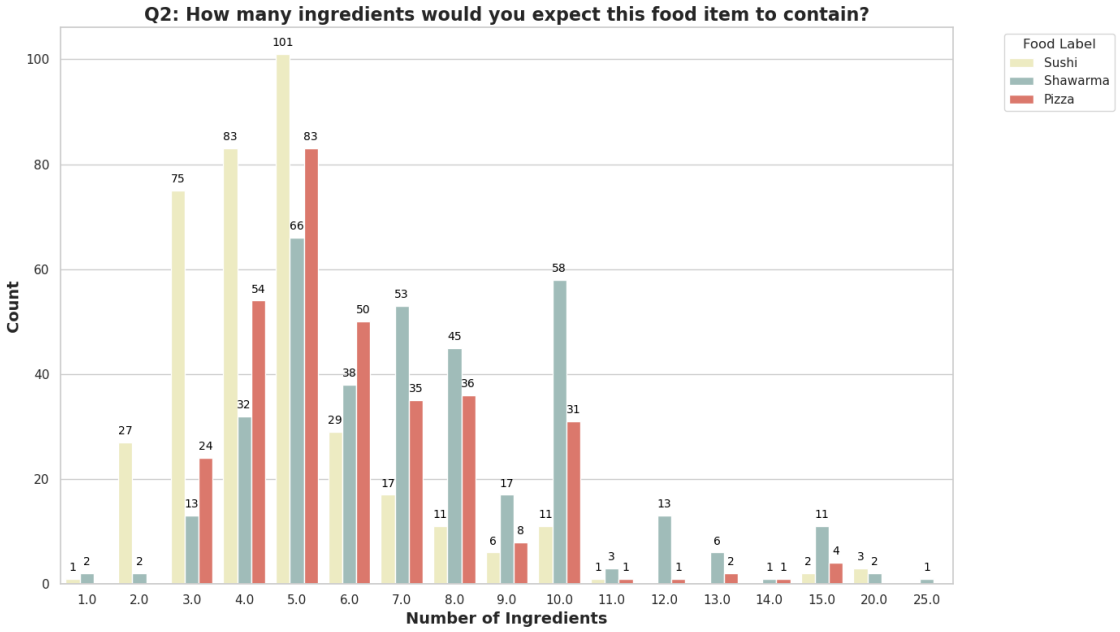
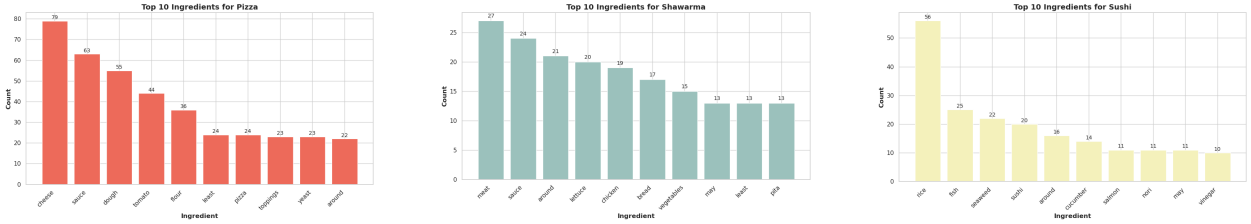


Figure 2: Distribution of perceived number of ingredients across food labels.

For the second question, respondents were asked to estimate how many ingredients each dish contains. Responses varied significantly: some provided direct numeric estimates, while others listed individual ingredients. To systematically analyze this feature, we applied text parsing techniques, specifically tokenization, to extract numerical ingredient counts and identify frequently mentioned ingredients. The analysis revealed distinct patterns: *Sushi* commonly has a narrow ingredient range, typically 3-5 ingredients, while *Shawarma* and *Pizza* showed broader and higher distributions of ingredient counts, suggesting greater variability and complexity in their preparation.



(a) Pizza Label (b) Shawarma Label (c) Sushi Label

Figure 3: Distribution of top 10 perceived ingredients across food labels.

Furthermore, our detailed text parsing highlighted specific ingredients commonly associated with each dish—cheese, dough, and tomato for *Pizza*; meat, lettuce, and sauce for *Shawarma*; and rice, fish, and seaweed for *Sushi*. These ingredients align closely with popular perceptions and enhance our understanding of how respondents categorize each dish based on its core components.

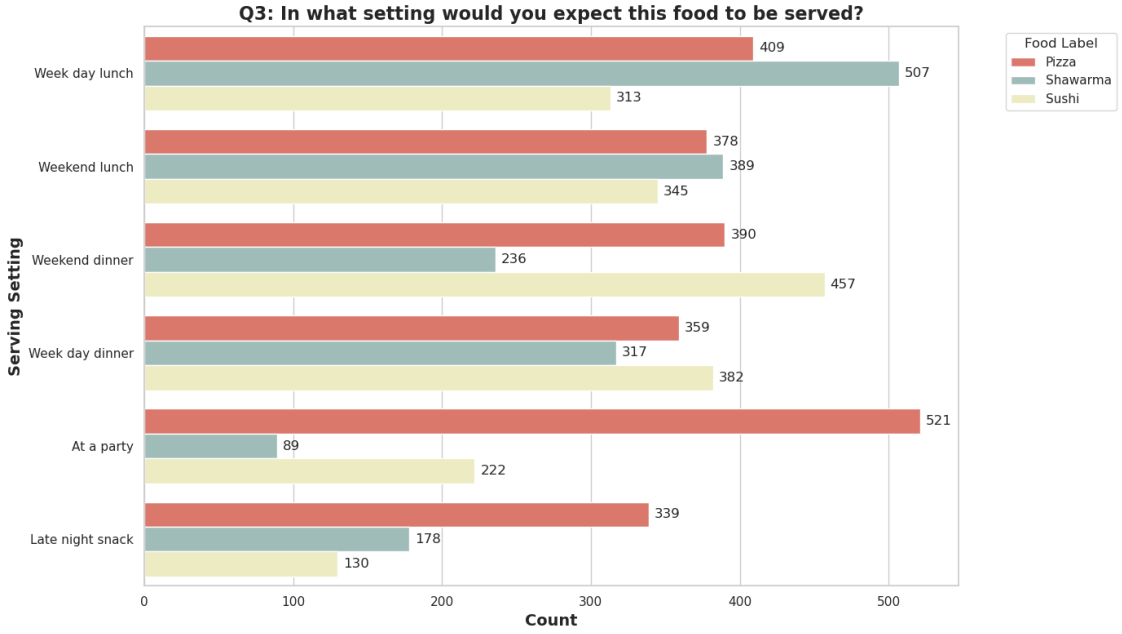


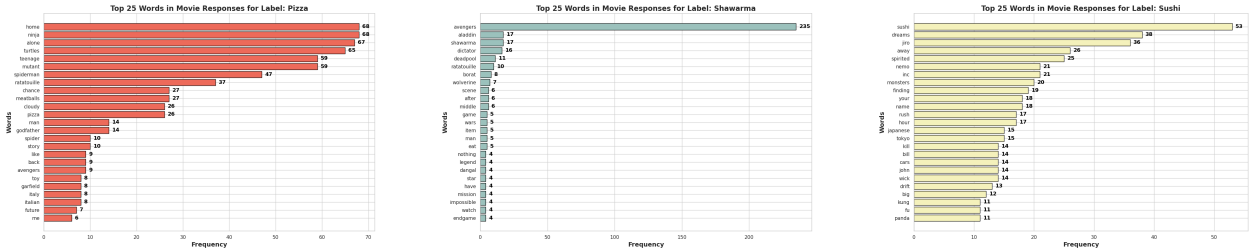
Figure 4: Frequency of each food item expected in various settings.

We further analyzed the settings participants associate with each dish, uncovering distinct associations. *Pizza* is predominantly linked to informal and casual environments, such as parties and late-night snacks. *Shawarma* is commonly associated with weekday lunches, suggesting a perception as an accessible, quick meal option. *Sushi* is notably linked with structured and formal dining occasions, such as weekend dinners. The significance of these observed associations was confirmed by a Chi-square test (Chi-square = 1256.41, p-value ≈ 0), highlighting a strong relationship between dining settings and the food labels.

Figure 5: Frequency of each food item expected in various settings.

Additionally, we examined participants' expectations regarding dish pricing, revealing substantial differences among the food types. *Pizza* is perceived as the most affordable, with most price estimates clustering toward lower values. *Shawarma* holds moderate pricing expectations, typically viewed as moderately priced. In contrast, *Sushi* exhibits a pronounced right-skewed distribution, indicating perceptions of higher pricing.

An ANOVA test confirmed the statistical significance of these price perceptions (F-statistic = 101.04, p-value = 4.15e-42), underscoring price expectation as a highly influential differentiating factor among these dishes.



(a) Pizza Label (b) Shawarma Label (c) Sushi Label

Figure 6: Top 25 movie words mentioned when thinking about each food (Question 5).

After text preprocessing, we identified distinct movie-related associations for each dish: *Pizza* was linked with casual, family-oriented films like Home Alone and Teenage Mutant Ninja Turtles; *Shawarma* strongly evoked the Avengers due to a memorable cultural moment; and *Sushi* was associated with Japanese-themed films like Jiro Dreams of Sushi. These connections reveal how respondents culturally differentiate dishes beyond their ingredients or taste.

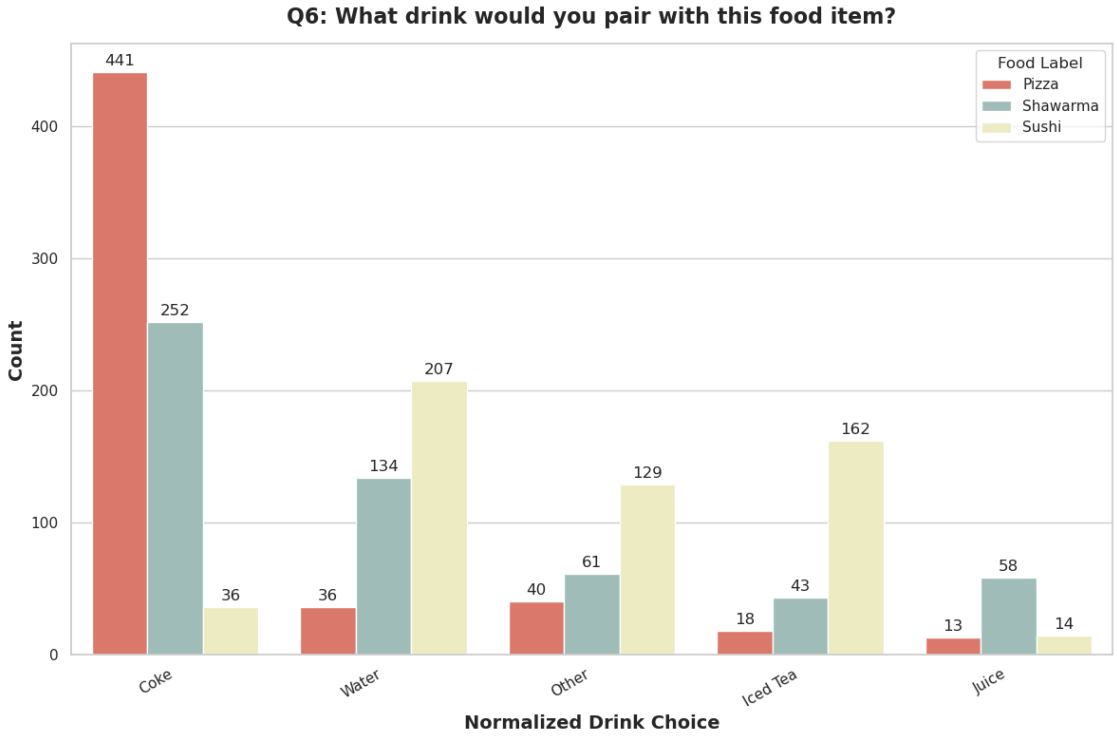


Figure 7: Distribution of normalized drink pairings for each food type.

Responses regarding drink pairings showed notable variation. To handle the diverse textual responses effectively, we applied keyword-based normalization, categorizing responses into clear beverage groups: *Coke*, *Water*, *Juice*, *Iced Tea*, and *Other*. *Pizza* and *Shawarma* primarily paired with Coke, while *Sushi* predominantly associated with Water and Iced Tea, reflecting healthier beverage choices. A Chi-square test (Chi-square = 1573.22, p-value ≈ 0) confirmed a statistically significant relationship between beverage preferences and food types, emphasizing drink pairings as another crucial distinguishing feature.

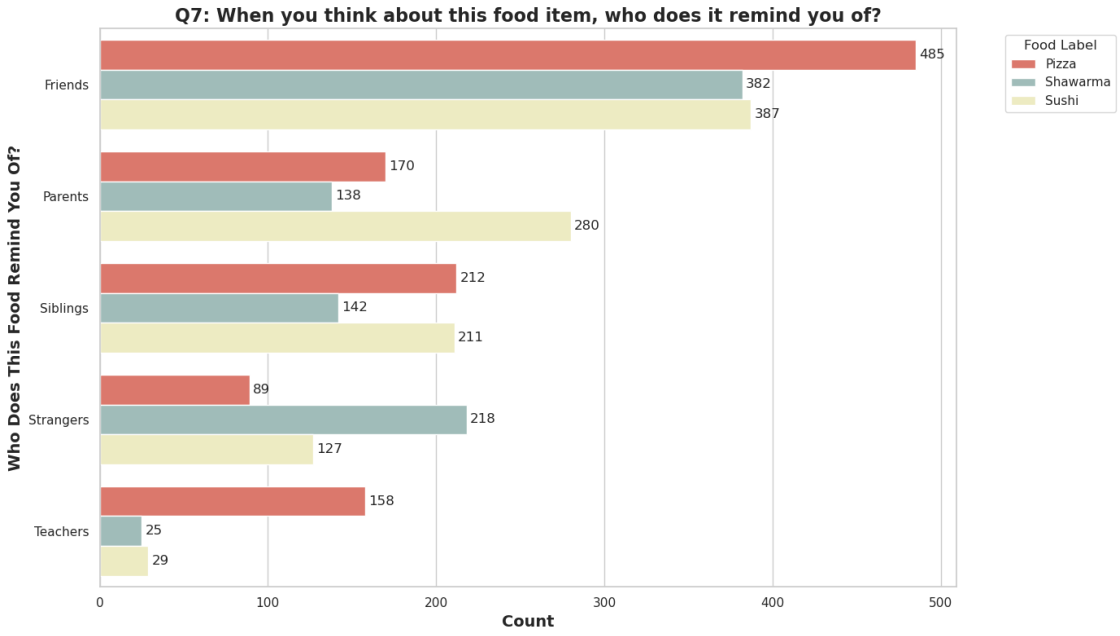


Figure 8: Frequency of person associations for each food type.

Social associations indicated that all dishes were most frequently linked with informal groups, particularly friends. *Pizza* had the strongest association with friends, while *Shawarma* and *Sushi* also showed notable family associations. Teachers had minimal linkage, emphasizing the informal nature of these dishes. These associations significantly differed by dish (Chi-square = 440.94, $p \approx 0$)

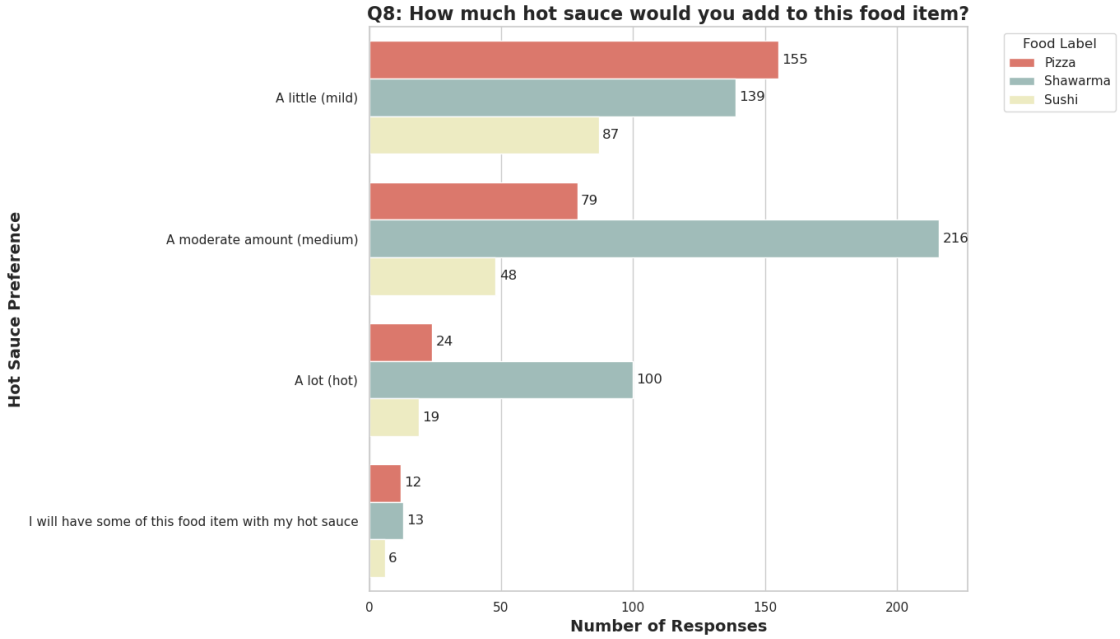


Figure 9: Distribution of respondents' hot sauce preferences for each food type.

Spiciness preferences also showed significant variations. *Shawarma* was distinctly associated with higher spice levels, with strong preferences for moderate to hot spiciness. In contrast, *Pizza* and *Sushi* were predominantly associated with milder spice preferences. A Chi-square test confirmed the statistical significance of these

differences (Chi-square = 73.92, $p \approx 0$), highlighting spiciness as an important differentiating characteristic among these dishes.

1.2 Feature Selection

Our exploratory analysis, combined with statistical tests, showed significant relationships between each feature and the food labels. Specifically, categorical variables like serving settings (Q3), drink pairings (Q6), social associations (Q7), and spice preferences (Q8) had strong associations confirmed by Chi-square tests. Numerical variables, such as perceived complexity (Q1) and expected prices (Q4), also varied significantly across food categories, supported by ANOVA test results.

Additionally, we derived insightful features from open-ended textual responses, such as frequently mentioned ingredients (Q2) and associated movies (Q5). These qualitative features provided valuable context that purely numerical or categorical data could overlook, giving us deeper insight into how respondents differentiated the food types.

Overall, each feature—original or derived—proved logically relevant and statistically meaningful, so we retained all of them to ensure our modeling captured the richness of the original data.

1.3 Data Representation

To ensure the dataset was suitable for machine learning models, we carefully transformed all features into numerical formats such as scalar values, one-hot or multi-hot vectors, or bag-of-words representations. Below is a breakdown of how we represented each question's response in our data matrix:

- **Q1 (Food Complexity)**: This question was already answered on a 1–5 scale, so we retained it as a numeric variable without transformation.
- **Q2 (Number of Ingredients)**: Responses varied widely—some provided exact numbers, others listed ingredients, wrote ranges, or expressed uncertainty. We first extracted numeric values, using the average for ranges when applicable. These values were then one-hot encoded into 25 binary features, each representing ingredient counts from 1 to 25. Additionally, we created bag-of-words features from the remaining text by extracting the top ingredient keywords per food label.
- **Q3 (Occasions for Serving)**: As a multi-choice question, each occasion was converted into its own binary column. A response containing that occasion is encoded with a 1 in the corresponding column.
- **Q4 (Expected Price)**: We extracted numeric values from free-text price responses using regular expressions. The extracted numbers were treated as continuous features.
- **Q5 (Associated Movies)**: We removed punctuation and stop words from the text, tokenized the words, and computed word frequencies separately for each food label. The top 25 most frequent words from each label were then used to create binary bag-of-words features.
- **Q6 (Paired Drinks)**: Many responses were semantically similar but expressed using synonyms (e.g., *Coke*, *Pepsi*, *soda*). We normalized these into a small set of representative categories such as *Coke*, *Juice*, *Water*, *Iced Tea*, and *Other*. These categories were then one-hot encoded.
- **Q7 (Associated People)**: As a multiple-choice question, each possible answer (e.g., *Parents*, *Friends*) was represented by a binary column, where 1 indicates selection.
- **Q8 (Hot Sauce Preference)**: This was an ordinal question. Each level of spiciness was mapped to an integer from 0 to 4, with larger values indicating higher spice levels. This was treated as a numerical feature.
- **Labels**: Each label (*Pizza*, *Shawarma*, *Sushi*) was one-hot encoded to support multiclass classification. In some models (softmax and random forest), we converted this to a single multiclass variable with values 0–2 to use.

1.4 Data Splitting

To evaluate the models fairly, the dataset was first randomly divided into two main subsets: an 80% training set used for model building and hyperparameter tuning, and a 20% test set reserved exclusively for final evaluation.

For the Random Forest and hybrid Naive Bayes + Gaussian Discriminant Analysis (GDA) models, we applied 5-fold cross-validation on the 80% training subset. This method allowed for thorough hyperparameter tuning and accurate estimation of the models' performances during training without overfitting.

For Neural Networks and Softmax Regression models, we used a validation set approach. Specifically, the Neural Networks model divided the 80% subset into 64% training subsets and 16% validation, while the Softmax regression model used a 60% training and 20% validation split. Both models used their respective training subsets to learn model parameters, with hyperparameter tuning guided by performance on the validation subsets. This ensured a clear distinction between parameter estimation and hyperparameter optimization.

Finally, the performance of all models was evaluated in the same 20% test set, facilitating fair comparison between different modeling approaches.

2 Model

In this project, we explored multiple modeling approaches to address the multiclass classification task of predicting food labels (*Pizza*, *Shawarma*, *Sushi*) based on survey-style responses. Our modeling choices were guided by the diverse nature of the input data, which includes both continuous numeric features and high-dimensional binary features.

2.1 Random Forest

Random Forest classifier is a solid choice for the problem because of its robustness and effectiveness in handling datasets containing numerous categorical and numerical features, which matches the structure of our dataset. Random Forest is particularly powerful in managing high-dimensional data, reducing the risk of overfitting through its ensemble approach, which combines predictions from multiple decision trees. Given that our dataset mostly contains categorical variables—such as food settings, drink pairings, and social associations—we converted these into binary features using one-hot encoding. This transformation allowed the Random Forest model to effectively capture complex interactions and relationships among various categorical predictors. Moreover, the Random Forest algorithm inherently mitigates the high variance typically observed in single decision trees by aggregating and averaging outcomes from numerous individual trees, thus improving prediction stability and accuracy.

2.2 Softmax Regression

Softmax regression is a generalization of logistic regression for multi-class classification. While logistic regression is designed for binary classification using the sigmoid function and a threshold, softmax regression replaces the sigmoid with the softmax function, allowing the model to output a probability distribution over multiple classes. For each input, the model computes a weighted sum using a weight matrix W and bias vector b, then applies softmax to produce class probabilities. The class with the highest probability is selected as the final prediction. We used softmax regression because our task involves three distinct classes: Pizza, Shawarma, and Sushi. A model that can directly output a probability for each class is better suited for this setup than a binary classifier.

To train the model, we used gradient descent to minimize the cross-entropy loss between predicted probabilities and the true one-hot labels. We initialized W and b, then updated them iteratively using the gradients. For hyperparameter tuning, we explored a range of learning rates and epoch counts. For each (learning rate, epoch) pair, we trained a model on the training set and computed validation accuracy. We then selected the pair with the highest validation accuracy as the final hyperparameters. After selecting the best hyperparameters, we used them to evaluate the final model on the test set. Softmax regression is

particularly appropriate for our dataset because we transformed all feature data into numerical or vectorized forms. This allows the model to effectively learn linear decision boundaries between the three food classes. Additionally, softmax regression is a relatively simple, interpretable model, making it a strong baseline for comparing more complex models such as neural networks.

2.3 Neural Networks

We considered a custom neural network as one of the candidate models for this task due to its ability to capture complex, nonlinear relationships in the data. Neural networks are particularly effective when the interactions between features are not strictly linear, which made them a promising option given the diverse set of predictors in our dataset.

Our implementation was built from scratch using NumPy, giving us full control over the training process. We implemented forward and backward propagation, softmax output, and supported different activation functions. To prepare the data, we normalized the input features and converted categorical labels into one-hot encoded vectors to match the expected output format.

2.4 Naive Bayes and Gaussian Discriminant Analysis

We adopted a hybrid Naive Bayes + Gaussian Discriminant Analysis (GDA) model because it fits the structure of our dataset, which contains a mixture of binary categorical features and continuous numerical features derived from free-text survey responses.

Most of our processed features are binary, and they are usually high-dimensional, sparse, and noncontinuous, making them ideal for Bernoulli Naive Bayes, which assumes feature independence and efficiently models presence/absence patterns. Naive Bayes is also scalable and interpretable, even with many binary indicators.

For continuous features, such as Q4 (expected price) and Q8 (spice level), we used GDA instead of discretization, which would lose valuable information. GDA assumes that continuous features follow a class-conditional multivariate normal distribution, and it estimates class-specific means and a shared regularized covariance matrix. This allows the model to capture correlations between continuous variables and enhances the separation between classes.

Rather than forcing all features into a single modeling approach, our hybrid method treats binary and continuous variables according to their statistical properties. This leads to more realistic assumptions and better predictive performance.

To further improve robustness, especially under class imbalance and limited sample sizes, we applied bootstrap aggregation (bagging). We trained multiple models on bootstrapped samples and averaged their log-likelihoods at prediction time. This reduces variance and stabilizes predictions, particularly for underrepresented classes.

3 Model Choice and Hyperparameters

3.1 Model Selection

To determine the final model used in `pred.py`, we compared the performance of five approaches: Random Forest, Softmax Regression, Neural Networks, a hybrid Naive Bayes and Gaussian Discriminant Analysis model, and an ensemble that combined all four. Each model was trained and tuned on the same 80% portion of the data and evaluated on a shared 20% test set using accuracy as the main metric. While accuracy on the test set was important, we also looked at how stable each model was across different splits and how well it generalized. In some cases, a model with slightly lower accuracy showed more consistent behavior, which we valued more than a marginal improvement that might not hold up across other datasets. These considerations helped guide our final model selection.

3.2 Evaluation Metrics

We used the following evaluation metrics to guide model selection and assess final performance:

- **Accuracy**: This was our primary metric for evaluating model performance. Accuracy measures the proportion of correctly predicted labels and was used consistently across all models to compare their effectiveness.
- **K-Fold Cross-Validation**: For the Random Forest and the hybrid Naive Bayes and Gaussian Discriminant Analysis models, we applied *5-fold-cross-validation* on the training set to tune hyperparameters and estimate model generalization.
- **Validation Set Performance**: For Neural Networks and Softmax Regression, we split the training data further into dedicated validation sets. These were used during hyperparameter tuning to identify the best configurations before testing.

Regardless of the validation strategy used during training, all models were ultimately evaluated on the same 20% test set. This ensured that our final comparison of model performance was consistent and based on data that had not been seen during training or validation, helping to avoid overfitting.

3.3 Hyperparameter Tuning

3.3.1 Random Forest

For the Random Forest model, we conducted hyperparameter tuning using *scikit-learn's Grid Search* with cross-validation. The hyperparameters we tuned for this model are shown in Table 1.

Hyperparameter	Description	Values Tested	Selected Value
`n_estimators`	Number of decision trees	$5, 10, 20, 50, 100$	50
`max_depth`	Maximum depth of each individual tree	$5, 10, 15, 20,$ `None`	`None`
`min_samples_split`	Minimum number of samples required to split an internal node	$2, 5, 10$	5
`max_features`	Maximum number of features considered when splitting a node	`sqrt`, `log2`, `None`	`log2`

Table 1: Hyperparameter tuning summary for Random Forest.

The final hyperparameters above were chosen based on validation accuracy across folds.

3.3.2 Softmax Regression

For the Softmax Regression model, we tuned two hyperparameters: learning rate and number of epochs. Table 2 summarizes each hyperparameter, its role, the values explored, and the final choice:

Hyperparameter	Description	Tried Values	Final Choice
`learning_rate` (α)	Step size used in gradient descent updates	randomly sampling 20 values from 10^{-4} to 1	0.00483
`epoch`	Number of iterations over the training data	500, 1000, 1500, 1700, 1800, 1900, 2000, 2100, 2500	1900

Table 2: Hyperparameter tuning summary for Softmax Regression.

The final hyperparameters (learning rate and number of epochs) were selected by identifying the combination that achieved the highest validation accuracy, effectively balancing underfitting and overfitting.

3.3.3 Neural Networks

For the Neural Networks model, we tuned four hyperparameters manually: the number of hidden layers and neurons per layer, the activation function, the learning rate, and the regularization strength. To keep the tuning process efficient, we fixed the number of training epochs to 1000 across all configurations. Table 3 provides a summary of each hyperparameter, its purpose, the range of values we experimented with, and the final choice used in our model.

Hyperparameter	Description	Values Tested	Selected Value
`hidden_sizes`	Number of hidden layers and neurons in each layer	$[256], [128, 64], [128, 64, 32]$	$[128, 64]$
`activation`	Activation function in hidden layers	`relu`, `tanh`	`relu`
`learning_rate`	Step size during gradient descent	$0.1, 0.01$	0.01
`lambd`	L2 regularization strength	$0.01, 0.0001$	0.01

Table 3: Hyperparameter tuning summary for Neural Network.

As with the softmax regression model, the final hyperparameters were chosen based on the configuration that achieved the highest validation accuracy. This approach helped us strike a good balance between underfitting and overfitting.

3.3.4 Naive Bayes and Gaussian Discriminant Analysis

For the Naive Bayes + GDA model, we performed hyperparameter tuning using a validation split and *5-fold cross-validation*, implemented using `sklearn.model_selection.KFold` and `accuracy_score`. The goal was to maximize average validation accuracy while maintaining model stability and generality. The hyperparameters we have tuned for this model are shown in Table 4.

Hyperparameter	Description	Tried Values	Final Choice
`B`	Number of bagged models	3, 5, 10	5
`alpha`	Laplace smoothing for Bernoulli NB	0.1, 0.5, 1.0	0.5
`reg_param`	Covariance regularization for GDA	10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}	10^{-6}
`priors_type`	Class prior strategy	`'uniform'`, `'empirical'`	`'empirical'`

Table 4: Hyperparameter tuning summary for Naive Bayes and GDA.

These hyperparameters were chosen based on validation accuracy across folds, as well as consistency across seeds and class balance.

3.4 Final Model

Our final model is an ensemble that combines four independently trained classifiers: Random Forest, Softmax Regression, Neural Networks, and Naive Bayes with GDA. Each model was trained on the same training dataset with its optimal hyperparameters. To make predictions, all four models are applied to the test set, and the final predictions are chosen using a majority vote. In the case of a tie—when two labels each receive votes from two models—one of the tied labels is randomly selected.

Although this choice involved a slight tradeoff in raw test accuracy, the ensemble helped balance the strengths and weaknesses of each model type. It reduced prediction variance and improved generalization, making it a more reliable option for our final model.

The ensemble logic is handled by the `predict_all()` function in `pred.py`. This function reads the input CSV file, applies appropriate pre-processing to the data, and loads the four trained models from their respective `.pkl` files. Each model makes its own prediction for the given input, and the predictions are combined to determine the final label for each instance through majority voting. Tie-breaking is handled by randomly choosing among the tied labels to ensure a valid prediction is always returned.

4 Prediction

4.1 Performance Estimate

The final test accuracy of our ensemble model is **91.49%**. This value was obtained by evaluating the model on the test set that we previously set aside, using the `predict_all()` function defined in `pred.py`, which implements our majority-vote ensemble. Table 5 provides a summary of the individual test accuracies and how frequently each model contributed to the ensemble's final predictions.

Model	Test Accuracy	Votes Contributed
Random Forest	94.83%	316
Softmax Regression	86.63%	305
Neural Network	87.23%	310
Naive Bayes + GDA	90.58%	319

Table 5: Test accuracy and number of final votes contributed by each model (out of 329 test instances).

4.2 Reasoning

Although the Random Forest model achieved the highest individual test accuracy at 94.83%, we selected the ensemble for its improved stability and generalization. The ensemble accuracy of 91.49% remains competitive with the top-performing individual model. By combining predictions from four different model families, the ensemble leverages their individual strengths while mitigating their weaknesses.

As shown in Table 5, all four models contributed significantly to the ensemble's predictions across the 329 test instances. The voting distribution is well-balanced, with each model influencing a large portion of the final outputs. This confirms that the ensemble did not overly rely on any single model, and instead benefited from the diversity provided by different model's approach. Overall, the ensemble offers a strong tradeoff between accuracy and reliability, making it well-suited for our final model.

To conclude, our final model is designed to generalize well and handle unseen data with stability and confidence. Based on its strong and consistent performance on unseen data, we expect the ensemble to achieve similarly high accuracy on the final test set used for grading.

5 Workload Distribution

Christoffer implemented the Random Forest model, Tianhui worked on the Softmax Regression model, Janis developed the Neural Network model, and Jingwen handled the Naive Bayes + GDA model. The remaining components—including the report, prediction script, and ensemble algorithm—were completed collaboratively by all team members.