Navya Hooda
Bethany Liang
Angela Liang
Ana Elisa Lopez-Miranda
Dr. Lisa Zhang
CSC311
April 4, 2025

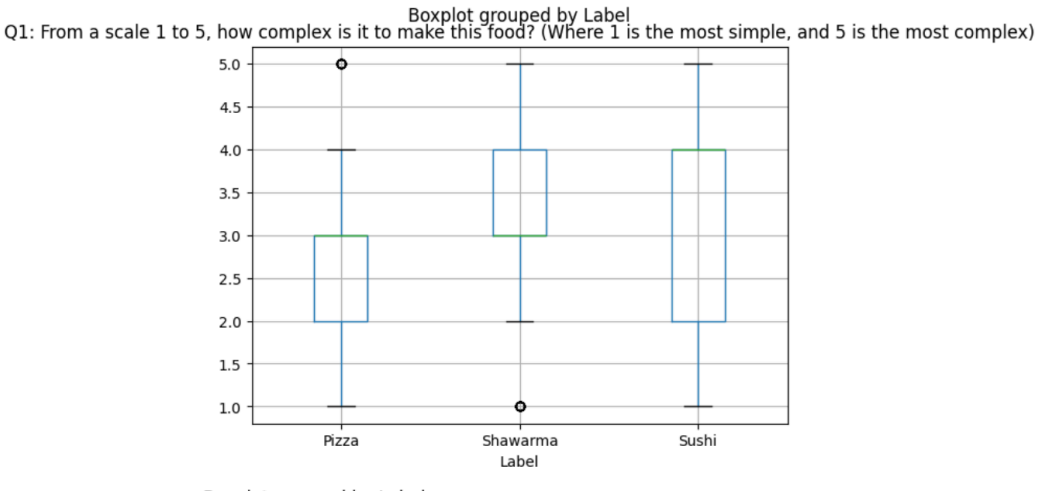
Machine Learning Challenge

Data Exploration

When doing the data exploration we noticed that we had multiple types of data. For example questions Q1,Q2,Q4, Q8 were numerical, Q5-6 were text based, and Q3,Q7 were multi-category based answers. To further understand the trends of these variables and help decide which questions might be more suitable for different models, we collected some empirical evidence on each type of data we have to guide our data selection process. For example, we created boxplots for the numerical ones to see their overall distributions. We used bar plots/histograms to represent the categorical answers

We made graphs showing summary statistics for the different questions. From the graphs, we chose whether to keep the question as a feature.

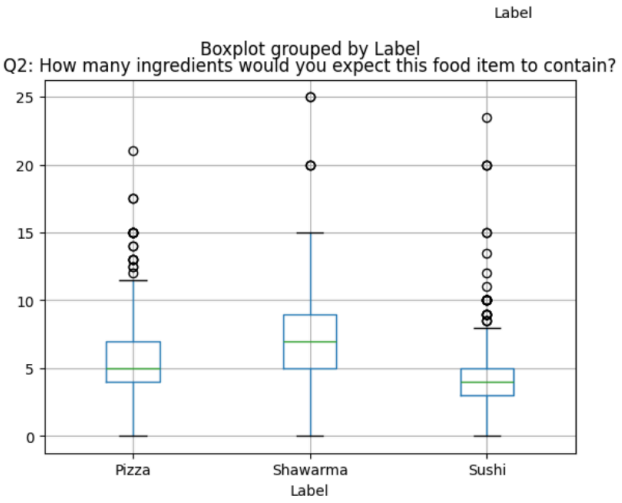
Q1.



As Q1 is numerical, we were able to show the graph using a box plot. The median for pizza is 3.0 while the interquartile range is from 1.0 to 4.0. There is a single outlier at 5.0. Similarly, the median for shawarma is 3.0 with an interquartile range of 2.0 to 5.0. In comparison, sushi had a median of 4.0 with an interquartile range of 1.0 to 5.0. Overall, sushi had a greater variability in perceived difficulty with the highest median of 4.0. All three seem to be mirroring a normal distribution. Because of the different distributions of the boxplot, it

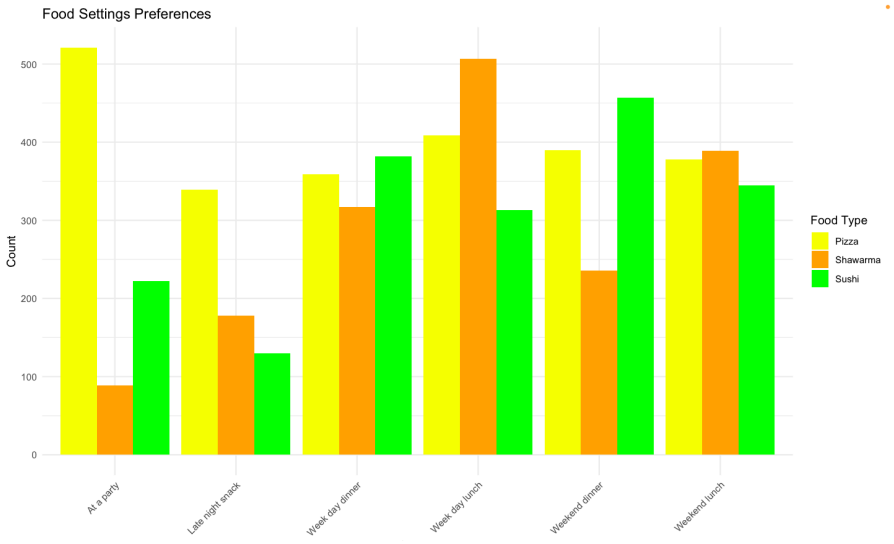
indicates that Q1 is a good indicator for classifying our response due to its differences in the categories.

Q2.



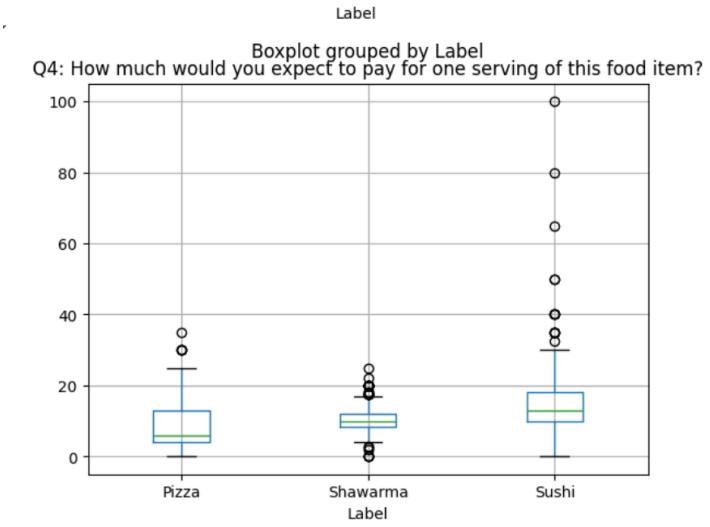
For Q2, Pizza has an interquartile range (IQR) of around 4.0 - 7.5 and a median of 5, Shawarma has an IQR of around 3.0 and a median of 7.5. Pizza has an IQR of about 3.0, and a median of 5.0. All three appear to be right-skewed distributions. Because of the different interquartile ranges of the boxplot and different variability, it indicates that Q2 is a good indicator for classifying our response and has potential as a feature to be used in the models.

Q3.



Q3 had multi-categorical data, to explore this we pick a barplot to visualize which options were most popular in the 3 categories. Pizza has the highest preference at the setting of a party and a late-night snack. Sushi has the highest preference at the setting of a week day dinner and weekend dinner. Shawarma has the highest preference at the sight of a weekday lunch and a weekend lunch. We see that each category had its own popular answer and mostly one category dominates the responses and that makes this question a good indicator to use. As all three are preferred in different settings, it indicates that Q3 could be a good indicator for classifying our response.

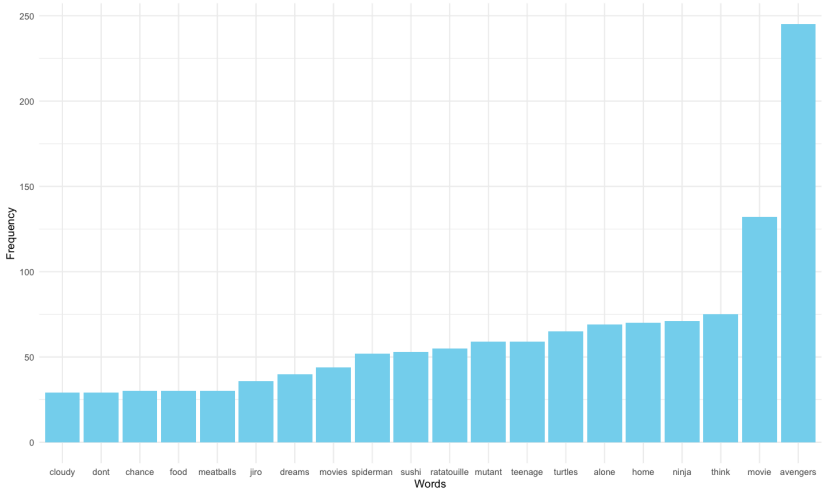
Q4.



Pizza has an IQR from 5.0 to 12.0 and a median of 7.5. Shawarma has an interquartile range of 10.0 to 12.0 and a median of 11.0. Sushi has an interquartile range from 10.0 to 19.0 and a median of 14. Again, because of the different distribution of the boxplots, this also indicates that Q4 is a useful feature for classifying our responses.

Hence, we decided to keep all of the numerical features for our decision tree model.

Q5. (What movie does this food remind you of?)



The above shows the top 20 words that appeared in the question about what movie reminded people of the food item. We first saw the overall trend of this question through a barplot, but as this is an open-ended question, it was hard to gain a meaningful interpretation through it. We decide to look at the word frequencies for each category. We see some words show up that will be handled in our data processing later. The following are the frequencies at which the top four movies appeared within each label to just give a snippet of the data found.

'cloudy with a chance of meatballs'	Frequency Percentage
Pizza	92.0
Shawarma	4.0
Sushi	4.0

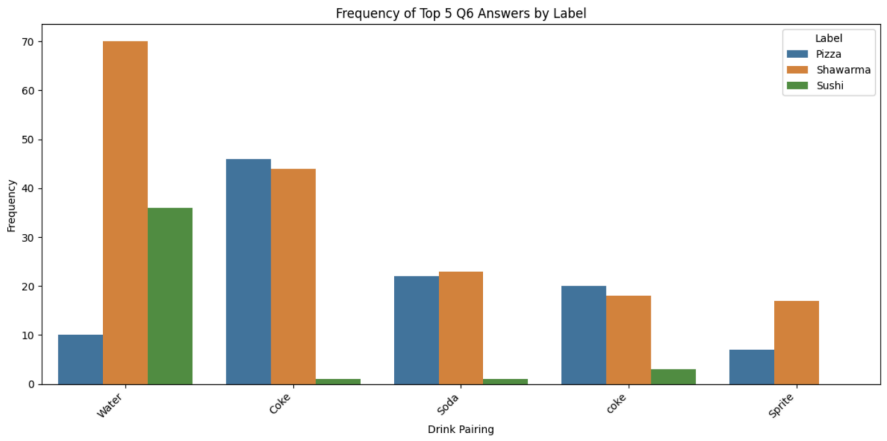
'home alone'	Frequency Percentage
Pizza	98.1818
Shawarma	0.000000
Sushi	1.818182

'the avenger'	Frequency Percentage
Pizza	0.826446
Shawarma	99.173554
Sushi	0.000000

'avengers'	Frequency Percentage
Pizza	4.938272
Shawarma	93.827160
Sushi	1.234568

We see that certain words are tied more to certain categories, and this would be helpful in training our models to learn patterns. Because of the difference in frequencies, it indicates that Q5 could be a good indicator for classifying our response. In addition, we also discovered that some responses from the same question share very similar frequency percentages after we have preprocessed the data with one hot shot encoding.

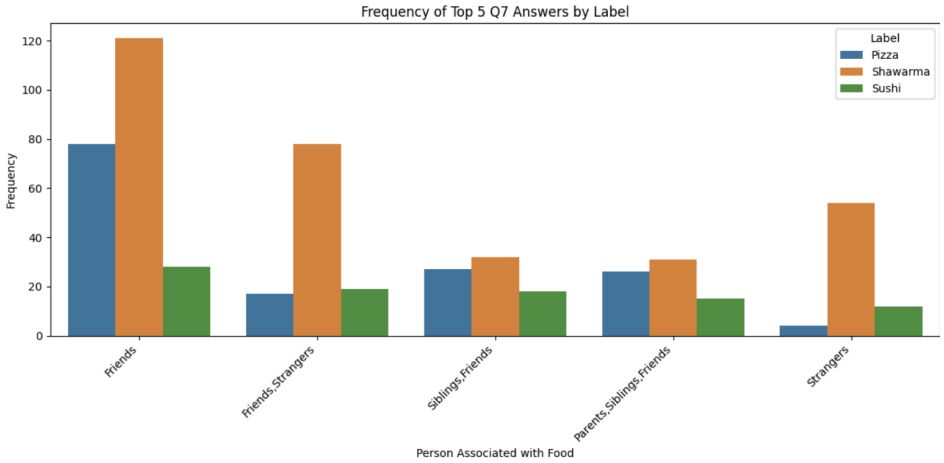
Q6. (Choice of Drink)




```
Top 5 responses for Q6 for Pizza:
    Response  Count
0       Coke     46
1       Soda     22
2       coke     20
3  Coca Cola     12
4      Water     10

Top 5 responses for Q6 for Shawarma:
  Response  Count
0    Water     70
1     Coke     44
2     Soda     23
3     coke     18
4   Sprite     17

Top 5 responses for Q6 for Sushi:
  Response  Count
0    Water     36
1      Tea     12
2     Sake      9
3    water      9
4      tea      5
```

Since Q6 was again a text response we explore the data through frequencies. Water, soda, and Sprite are more frequently chosen for shawarma than for piazza and sushi. Coke is more frequently chosen for Pizza. The barplot doesn't give a detailed insight into the details, so we proceed to a frequency count. Below are the frequencies for the top 5 drinks by category, to look at the frequency trends in more detail. Notice that drinks coke, diet coke, and soda coke are shared across categories but tea and sake seem to be more correlated with sushi. These trends can be useful for training. As the frequencies are not identical across all labels, it indicates that Q6 is a good indicator for classifying responses.

Q7.

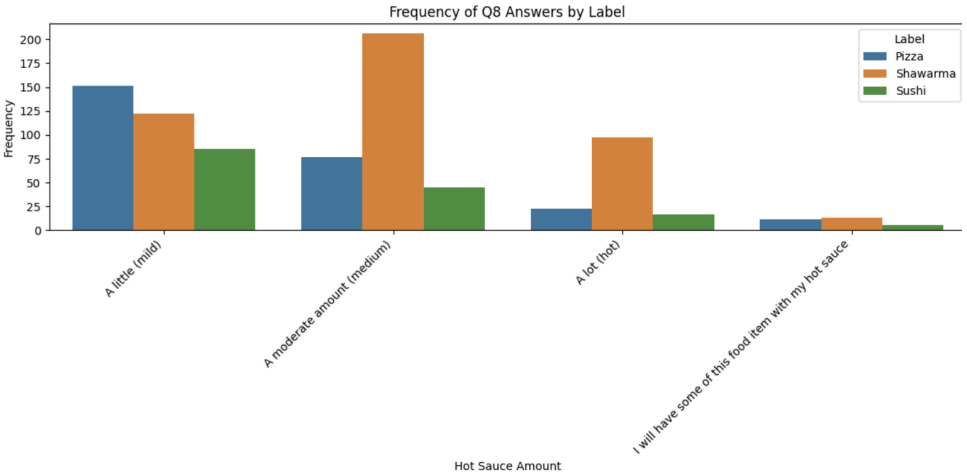


In all categories except strangers and friends/strangers, the three labels follow the same frequency of shawarma being the highest, pizza being the second highest and sushi being the lowest. Because of the variability in frequencies, it indicates that Q7 is a good indicator for classifying responses.

parents	Frequency Percentage
Pizza	14.084507
Shawarma	21.126761
Sushi	64.788732

parents,friends	Frequency Percentage
Pizza	18.918919
Shawarma	23.423423
Sushi	57.657658

Q8.



(Pizza has the highest frequency of containing a little hot sauce. Shawarma is the highest frequency for containing a moderate amount, a lot amount, and I will have some of my food items with my hot sauce. Sushi lags in all four categories as having the least.) Q8 exhibits high variability of responses for different food items. Specifically, responses ("A moderate amount(medium)" and "A lot of (hot)" show the most distinct distribution and are more likely to be labeled Shawarma with these two answers. This indicates Q8 is a good feature to differentiate the labeling. We also look at the frequency of those for more details, and they show good variability in responses, hence Q8 is likely a good predictor.

a lot (hot)	Frequency Percentage
Pizza	16.783217
Shawarma	69.930070
Sushi	13.286713

Medium	Frequency Percentage
Pizza	23.032070
Shawarma	62.973761
Sushi	13.994169

Representation

How we modelled/represented input data generally throughout the preprocessing in our models, certain decisions might be different for some models (explained in models).

- For Q1,2,4,8: These were numerical in nature, but Q2,4 were open text based. For those we ensured that we applied techniques to extract the numerical values, if ranges were given we chose to average them, convert word numbers to numbers, exclude symbols etc. Q1 was numerical by itself since it was on a scale of 1-5. Q8 was technically an ordinal type question, but we converted it to be a numerical scale from 0-4 where needed.
- Q3,7 was a multi-select categorical question, we chose to use indicator encoding denoting something similar to "Q3_friends", "Q7_weekendlunch" for example, to indicate what choice(s) was selected.
- Q5-6 were text based and we used mapping to make features for the word frequency ties we found earlier, and use encoding to indicate whether they were present or not.

Data Splitting

For training and building our model, we will follow the typical data split from our CSV file: 70% of the data for the training set, 15% of the data for the validation set, and 15% of the data for the test set. If we had directly shuffled the responses and split them on the above percentage, it could potentially cause data leakage. For example, a single person has three responses A, B, C, respectively pizza, shawarma, and sushi. If we simply shuffle the responses, response B and A might end up being in the training set and response C in the test set. Since response B, A, and C came from the same person, there is a correlation between

these responses. However, each response must be independent and uncorrelated with each other during training, validation, and final evaluation of the test set.

To ensure that there is no correlation between our training, validation, and test sets, we split the data sets by person IDs (548 unique IDs, so 548 people). We first get the unique person IDs from the CSV file and then randomly shuffle these IDs. Next, we assign the responses from 383 people (548 * 0.7 ≈ 383, 1149 responses) to the training data set, the responses from 82 people (548 * 0.15 ≈ 82, 246 responses) to the validation data set, and 83 people (548 - 383 - 82 = 83, about 15%, 249 responses) to the test data set. By separating the responses by person IDs and shuffling these IDs at the initial stage for splitting our data sets, this effectively prevents data leakage between our training, validation, and test sets. This approach guarantees that the test set contains entirely unseen data during training, which helps avoid overestimating the model's accuracy when evaluating it on the test set.

Models

KNN

We used KNN to compare each input point (x_feature, y_label) with the closest neighbour using the method **predict_knn** and the Manhattan distance to find the closest neighbours to the input data point, and chose the best k using the validation set.

We selected the Manhattan algorithm for computing the distance of the input vector against the training set, as it performs better than using the Euclidean distance. We noticed that we end up having a lower accuracy after normalization. Normalizing our KNN model might reduce accuracy due to feature importance by giving all feature importance equal weights.

For instance, as mentioned in the data exploration section, Q8 is a prominent feature to train our model. Normalizing each feature could reduce the relative importance of Q8, potentially making our model harder to accurately identify the closest neighbors and generate accurate predictions.

For KNN, we decided to omit Q5 ("What movie do you think of when thinking of this food item?") and Q6 ("What drink would you pair with this food item?") to improve our model training.

Q5 can cause noise in the model due to high variability and randomness with a total of 770 unique responses. Including Q5 in our selected features and using it to train our model can cause underfitting, as our model might fail to capture underlying patterns in our data, resulting in poor generalization.

```
**********Question 5 unique Answer**********
770
************************************************
```

Q6 is ambiguous, as many people respond “Coke” as the answer for all three labels, it would potentially overshadow other more relevant features in KNN prediction. This could lead to misclassifications if "Coke" is associated with multiple food labels.

In the below frequency analysis (without preprocessing the data), for any words related to “coke” or “coca cola”(case non-sensitive), we can see 42.70% of responses labelled Pizza, 21.53% of responses labelled Shawarma, and 4.74% labelled Sushi. Similar trend for frequency analysis on the word “soda”, with 16.61% labelled Pizza, 8.76% labelled Shawarma, and 0.18% labelled Sushi.

```
Q6: What drink would you pair with this food item?
Keyword distribution for coke|coca cola

Label
Pizza       234
Shawarma    118
Sushi        26
dtype: int64
Label
Pizza       42.700730
Shawarma    21.532847
Sushi        4.744526
dtype: float64
Keyword distribution for soda
```

```
Keyword distribution for soda

Label
Pizza       91
Shawarma    48
Sushi        1
dtype: int64
Label
Pizza       16.605839
Shawarma     8.759124
Sushi        0.182482
```

Hence, eliminating both questions Q5 and Q6 can help improve our model performance.

Decision Trees

To build our decision tree model, we began by determining a suitable range of features to build the model. Since we have 1644 responses, the lower bound for the number of features was log(1644) = 40 features, and the upper bound was log2 (1644) × 10, which is about 100 features. Hence, for tuning the parameters of the decision tree, we first began by one-hot encoding the top k (a hyperparameter that we set) features for the most common (question_answer) pairs for the questions Q3, Q5, Q6, Q7, and Q8.

To process the data, we extracted the numbers how it was mentioned in the data splitting section. For our categorical questions, we hotshot encoded the columns, which allowed our decision tree to capture important relationships in our data set. Through the data exploration phases, we realized that the frequencies for certain popular answers were the same with some questions. For example in Q6, out of all the responses that answered “sake”, 0 percent came from Pizza, 0 percent came from Shawarma, and 100 percent came from Sushi. Out of all the responses that answered “tea”, about 3% of the responses came from Pizza, 4% of the responses came from Shawarma, and 92% of the responses came from Sushi. In addition, for other drinks, coke and soda also had similar distributions, so we grouped these features into one. In addition, answers like “avengers” and “the avengers” for Q5 are equivalent, so it is also plausible to group them as they share very similar distributions. Grouping features will allow us to reduce the number of features for decision trees, which avoids the problem of overfitting.

Hence, in the preprocessing stage, we grouped the following responses into a single group:
Q2: (week day lunch, week day dinner, weekend dinner)
Q3: (avengers, the avengers)
Q5: (home alone, cloudy with a chance of meatballs)
Q6: (soda, diet coke, coke)
Q8: ("a moderate amount (medium)", "a lot (hot)")

Neural Networks

We tried both a sequential model from TensorFlow and a PyTorch model. Both had two hidden layers with 64 neurons in the first layer and 32 neurons in the second layer. Neural networks was originally chosen for its nonlinearity and its ability to predict from large datasets. ReLU was used in the hidden layers to help with the vanishing gradients. The softmax function was used in the output layer for multi-class classification. We fit the model with the training data and epochs of 50 and a batch size of 4. We used all eight features for this model. The adjustments that were needed were done in the preprocessing file. Inside of preprocessing, we did two things. We split the data into the specified sections above (70% training, 15% validation, and 15% test). Next, we treated each type of question separately. For numerical questions, we cleaned the data so all answers would only contain numeric values alone. For multi-select questions, we created indicator variables to allow multiple selection (as opposed to one-hot encoding). For text-based questions, we made dummy variables (for example _avengers indicates where avengers was one of the answers for q.5) to indicate if the word was in answer or not. The optimal weights were extracted from a neural networks class written on pytorch and a neural networks class written with sequential. The weights were then converted to numpy and manually loaded onto our class. At first, neural networks gave a test score of 89% leading us to believe it was the best model. Unfortunately, after coding it manually and using the optimal weights that were extracted, neural networks only gave a test score of 73%.

Model/Hyperparameter Tuning

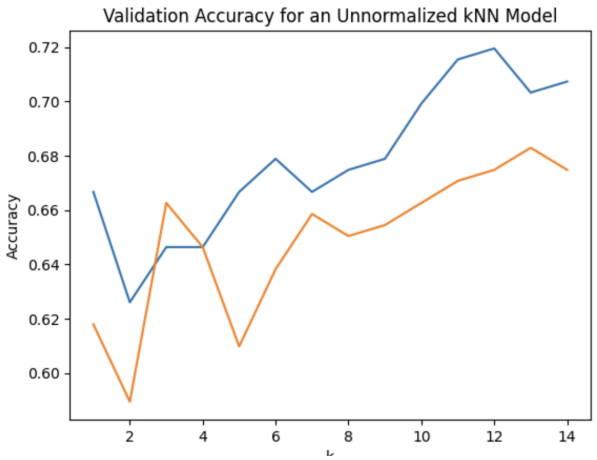
We used a consistent test set among all three models. We all followed the same method of splitting so we used the same set for training, validation, and testing. Thus, ensuring a fair comparison of their performance. We will evaluate our model based on the precision score and hyperparameter tuning to select the best model. Precision score is crucial because it provides insight into the quality of positive predictions made by the model and minimizes false positives. Precision scores were chosen for their simplicity and directness ensuring a consistent metric without potential hidden variables. We will also explain hypermeter tuning for each of our attempted models to improve the performance of our model

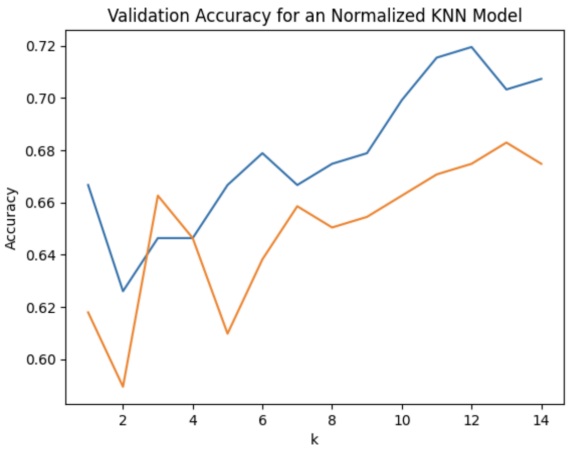
KNN

We want to choose the optimal K for KNN Hyperparameter tuning.

Validation Accuracy Comparison (Euclidean vs. Manhattan)

In KNN, the model accuracy depends on the choice of **distance metric** (evaluation metrics) that we use to find the nearest neighbours. We can see that using the Manhattan distance (**orange**) to determine the k nearest neighbours and predict the label has a higher validation accuracy than using Euclidean distance **(blue**). Picking k=11 suggests having a high validation and test accuracy. As mentioned in the Model section, our KNN model performs better without normalization because with normalization, it might diminish the impact of some important features in our dataset.





Compute Test Set Accuracy (Without Normalization is better)

Without Normalization:

```
[120] compute_accuracy(x_test, y_test, x_train, y_train, manhattan_distance,  11)
```

```
0.7108433734939759
```

With Normalization:

```
[121] compute_accuracy(X_test_norm, y_test, X_train_norm, y_train, manhattan_distance, 11)
```

```
0.6746987951807228
```

Decision Tree

We test on the numbers k = 10, 20, 30, 40, 50, 60, which gives us 53, 103, 153, 203, 253, 303 features in total. We noticed that k = 50 gave us the highest validation accuracy of 78.86

percent. Hence, we choose to encode the top 50 most common features for each of our questions when processing the data for training our model.

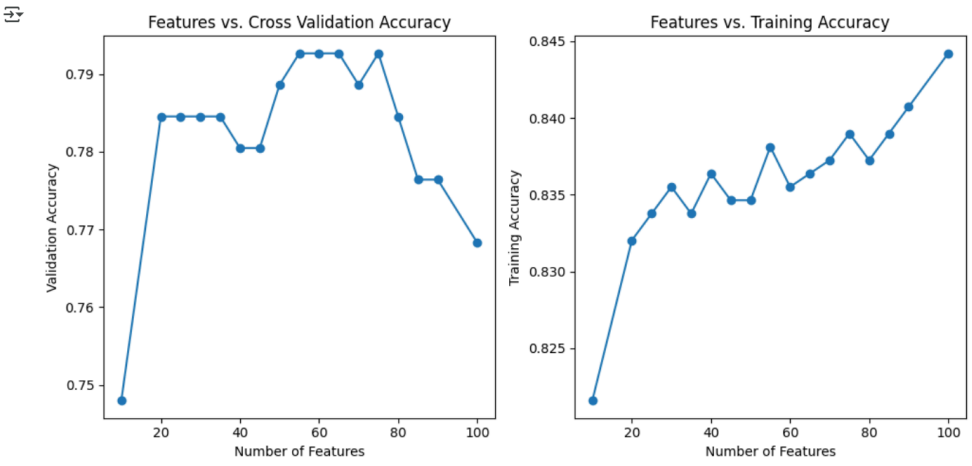
To further tune our hyperparameter settings for the decision tree, we use gridCV from scikit-learn to find the best hyperparameters. We also use a built-in cross-validation score to see how our model performs, as well as running the tune_model with ski-learn on our validation set (15 percent split coming from the person IDs). We used different criteria (entropy and gini), tree max_depth, min_depth, min_samples_split, min_samples_leaf, and min_imprurity_decrease hyperparameters. Although the Sklearn cross-validation method did not split the sets of responses by the person IDs, we believe that it could still be somewhat of a strong indicator of how our model is performing. However, we decided to lean towards focusing on our validation set accuracy.

Best Cross-Validation Parameters
(Top k = 50):
{'criterion': 'entropy', 'max_depth': 7, 'min_impurity_decrease': 0.001, 'min_samples_leaf': 4, 'min_samples_split': 2}
Cross-validation Accuracy (for best model): 0.7389

During the tuning for cross-validation accuracy, we also tested on different heights and saw if the validation accuracy would be higher given higher depths.

(Top K = 50)
Parameters: {'max_depth': 10, 'min_samples_split': 2, 'min_samples_leaf': 4, 'min_impurity_decrease': 0.001, 'criterion': 'gini'}
Validation Accuracy: 0.7927

After we got the hyperparameter settings for our decision trees, we then used the SKI learn **Recursive Feature Elimination (RFE)** built in SKI-learn imports to extract the most prominent features for our decision trees. To avoid an over-complicated model, we ran the decision tree model with the two above parameter settings, and both showed that extracting the top 55 most important features for our decision tree to make predictions gave us the highest accuracy. As shown in the graph below, around 55 - 75 features gave us the highest cross-validation and validation accuracies, respectively.




```
[ ] from sklearn.model_selection import GridSearchCV, StratifiedKFold
```

Using the average cross-validation accuracy model (tuned with cross-validation ski-learn import, with 55 most prominent features and parameters mentioned above), we got a validation accuracy of 0.7683. Using the validation accuracy model (tuned based on our own validation set, also taking the top 55 most prominent features and hyperparameters mentioned above), we had a validation accuracy of 0.7927 (that also had an average cross-validation accuracy of 0.7389). Hence, we decided to go with the validation accuracy model hyperparameter settings for our final decision model.

So our final decision tree model setting was:
K = taking the top 50 most common answers, encoding into features for our categorical features.

Decision Tree Hyperparameters:
Parameters: {'max_depth': 10, 'min_samples_split': 2, 'min_samples_leaf': 4, 'min_impurity_decrease': 0.001, 'criterion': 'gini'}

M = A selected set of 55 features that we extract using the RFE Ski learn imports

Below was our final result:

```
DecisionTreeClassifier train accuracy (with feature selection): 0.8381
DecisionTreeClassifier validation accuracy (with feature selection): 0.7927
Cross-validation scores: [0.7826087  0.69565217 0.7        0.76521739 0.7510917 ]
Average cross-validation score: 0.7389139927852667
selected_feature
['Q1: From a scale 1 to 5, how complex is it to make this food? (Where 1 is the most simple, and 5 is the most complex)',
```

```
X_train, Y_train, X_valid, Y_valid, X_test, Y_test, training_data_columns = preprocess_data('/content
x_test_selected = X_test[:, selected_feature_indices]
best_model.score(x_test_selected, Y_test)
```

```
0.7590361445783133
```

Neural Networks

We fine-tuned our model through hyperparameters like L2 regularisation constant and drop out rate. These 2 values helped our initial model reduce overfitting. Our architecture included two hidden layers with 64 neurons for the first and 32 for the second, with the nature of our smaller dataset. Choosing 32 neurons for the second layers helps it fine tune details in a more generalized manner, and since the hidden layer can learn more than linear relationships these layers help our model learn more through all the features we picked.

We decided to only go with two hidden layers as neural networks are typically done on large test sets but ours was much smaller.

Evaluation Metrics across the models.

To overall compare models, we chose a precision score which details exactly our rate of predicting true positives over the total types of false positives and such. Additionally test accuracies across the set were on the same test sets.

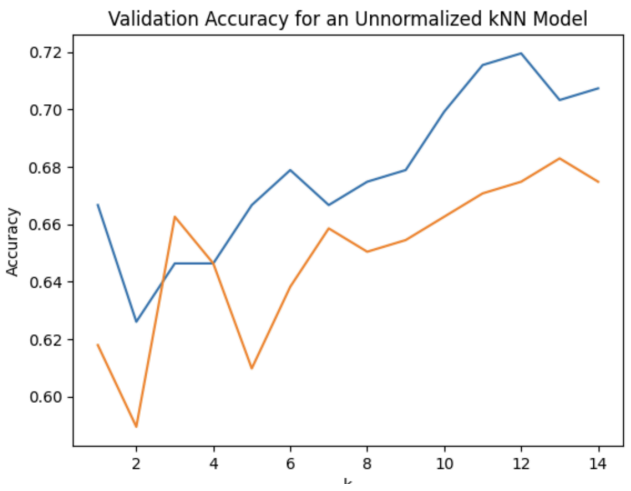
KNN

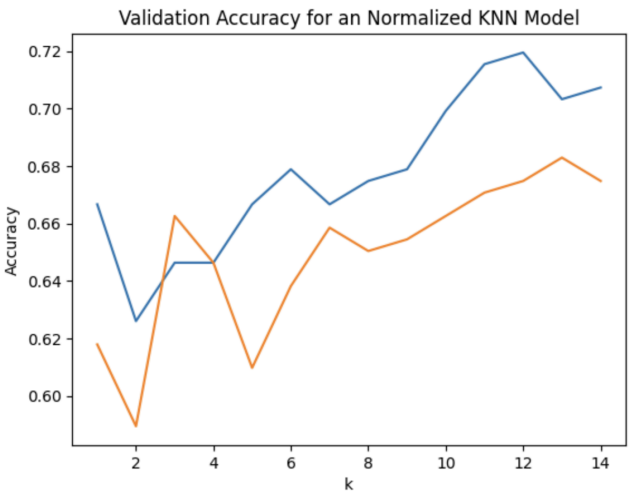
Interpretation: The precision score for label Sushi is approximately 72%, which means out of all data points of the KNN model predicted as Sushi, about 72% were Sushi. The precision score for the label Shawarma is the highest, with 79%. This means our model can identify the label Shawarma well. Label Pizza has the lowest precision percentage of 62.5%. This suggests our KNN model has the most difficulty in identifying the label Pizza.

```
Precision for label (np.int64(0), np.int64(0), np.int64(1)): 0.7228915662650602
Precision for label Sushi: 0.7228915662650602

Precision for label (np.int64(0), np.int64(1), np.int64(0)): 0.7948717948717948
Precision for label Shawarma: 0.7948717948717948

Precision for label (np.int64(1), np.int64(0), np.int64(0)): 0.625
Precision for label Pizza: 0.625
```

Validation Accuracy Comparison (Euclidean vs. Manhattan)

We can see that using the Manhattan distance (**orange**) to determine the k nearest neighbours and predict the label has a higher validation accuracy than using Euclidean distance **(blue**).





Decision Tree

Interpretation: The precision score for the label sushi, pizza and shawarma are approximately 70%, 76%, and 82% respectively, which indicates that 70 percent that predicted as sushi are correct responses, 76 percent that predicted as pizza are correct responses, and 82 percent that predicted as shawarma are correct responses.

```
# Print precision for the desired categor
for i in category_indices:
    print(f"Precision for {label_names[i]
```

```
Overall Test Precision: 0.7633
Precision for Sushi: 0.7010
Precision for Pizza: 0.7639
Precision for Shawarma: 0.8250
```

Neural Networks

Interpretation: The precision score for the label sushi, pizza and shawarma are approximately 83%, 87%, and 84% respectively, which indicates that 83 percent that predicted as sushi are correct responses, 87 percent that predicted as pizza are correct responses, and 84 percent that predicted as shawarma are correct responses.

```
Precision for class pizza: 87
Precision for class shawarma:84
Precision for class sushi: 83
```

Final Decision - Decision Trees (pred.py)

Although neural networks was favored to be the one chosen, ultimately generalizing the model led to the test score dropping from 89% to 73%. Additionally, neural networks had a cross-validation score of 54%. As decision trees had a higher test score of 75% and a higher cross-validation score of 73%, we chose to go with decision trees. Our file is made up of our preprocess function, which standardizes the questions chosen to be kept as features. Additionally, it has the manual code for decision trees that is made up of a multitude of if-statements.

Prediction

We expect our model to perform at 73% accuracy. The test set accuracy is 75.10%, which aligns with expectations as it is close to the validation accuracy. The slight decrease is typical for unseen data, which is normal for a test set. This result also reflects the effectiveness of our data splitting strategy, where careful separation of the training, validation, and test sets ensured there was no cross-contamination. As our cross-validation score was 0.7389 in addition to our test accuracy, this supports the idea that our model will perform at 73% accuracy.

```
DecisionTreeClassifier train accuracy (with feature selection): 0.8381
DecisionTreeClassifier validation accuracy (with feature selection): 0.7927
Cross-validation scores: [0.7826087  0.69565217 0.7        0.76521739 0.7510917 ]
Average cross-validation score: 0.7389139927852667
```

Group Contributions

Ana Elisa:

- Neural Network exploration with PyTorch and weight extraction
- Hyperparameter tuning for neural networks.
- Data exploration
- Prediction and conclusion

Navya:

- Neural Network exploration with Sklearn
- Neural network weight extraction
- Preprocessing data for neural network, encoding and word frequency
- Data exploration
- Hyperparameter tuning for neural nets

Angela: I specifically work on coming up with preprocessing the data for decision trees, splitting the data sets, exploring the decision tree model, and building the decision tree and hyperparameter tuning the model for predicting the categories (pizza, shawarma, and sushi) for the data responses.

Bethany: I specifically work on coming up with the method of preprocessing for kNN, splitting the data, exploring the kNN model, and choosing hyperparameters best for building the kNN Model for predicting the categories (pizza, shawarma, and sushi) for the data responses.