ML Challenge Report

Siqi Liu Rafay Usman Ryan Shiels Albert Li

I. Data

A. Cleaning

The multiple-choice questions (Q1, Q3, Q7, Q8) were split into individual one-hot features, and Q3 Q7, which both allowed multiple answers, also had each combination represented as a one-hot feature (if they appear in the training data at least 5 times) alongside a feature that represented the number of selected answers.

Q2 and Q4 were converted into numbers using the following logic:

- If there is a range (two numbers with the word “to”, “and” or “-” between), then take the average
- Otherwise, take the first number that appears (it can handle numeric, and also written numbers up to 20)
- Otherwise, if it’s the ingredients question, take the number of commas or line breaks, plus one
- Otherwise, return the average seen in the training

Q5 and Q6 were more complicated free-form text answers. To generalize as broadly to any input, we used a fuzzing library to cluster similar responses into singular features, assuming they appeared frequently enough in the training set. Then, for movie answers we only allowed a singular matched movie (or if nothing matched, “other”), but for drink answers multiple drinks could be listed.

For Q1, Q2, and Q4, another approach we adopted extracting all the numbers from the free-form questionnaire responses and return the average of the numbers. This was done by finding all literal number (number as words) and using a regex pattern to match all numbers in the string, and then converting them to floats. If no numbers were found, we returned 0.

For Q5, Q6, and Q7, another approach we adopted was using a word2vec model on the free-form questionnaire responses. Word2vec learns vector representations for words, assigning similar embeddings to words with related meanings. This captures semantic relationships in the text and allows distances between embeddings to quantify word similarity. For each free-form answer and label (pizza, sushi, shawarma), we calculated the average word vector. Each question had three features, one per food type, that recorded the cosine similarity between the free-form answer’s averaged vector and the averaged vector for each label.

This was all condensed into a function that took in a CSV file, split the data as needed, and output a dataframe of the flattened vectors for every feature.

B. Exploration

We visualized the cleaned data using histograms and boxplots to explore the data. Some questions such as Q3: ”In what setting would you expect this food to be served?” have multiple choice as inputs, and others such as Q5: ”What movie do you think of when thinking of this food item?” allow participants to type their responses. For those questions, we counted the number of occurrence for each input, and only show the top 5 responses in the graph so it is readable while capturing the most important information.

Figure **??** shows a histogram of responses for Q1: From a scale 1 to 5, how complex is it to make this food?. For Sushi, the difficulty is generally higher with more inputs of 4 and 5. For Pizza, we see medium complexity(3) being the highest choice. For Shawarma, we see more votes in the range of 4 and 5.

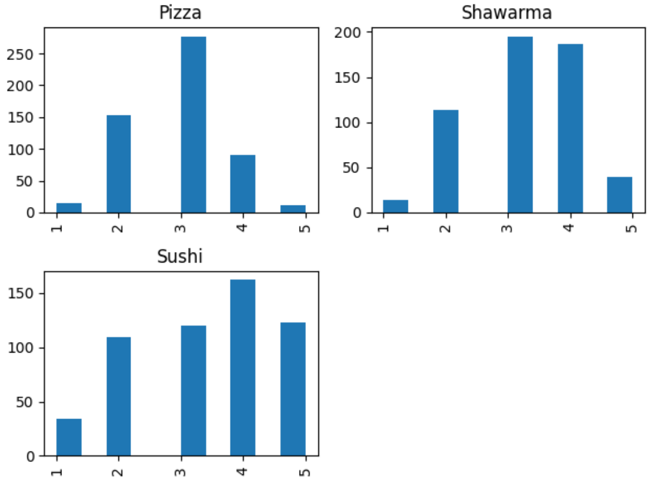

Fig. 1: Histogram for Q1: How complex is it to make the food

In figure **??**, the boxplot illustrates the distribution of expected ingredient counts for each food class. We see the participants generally expect the fewest ingredients in Sushi (median 4), followed by Pizza (median 5), and then Shawarma (median 7). Shawarma shows the greatest variability in expected ingredient counts, while Pizza and Sushi show less variability. All three food types have outliers, suggesting some individuals expect significantly more ingredients than the majority. We see in **??** that the histograms shows a right-skewed distribution for all three food items, indicating most respondents expect a low number of ingredients. However, the skewed pattern suggests a minority of participants anticipate a considerably higher ingredient count.

Question 3 is ”In what setting would you expect this food to be served?”. For Q3: “In what setting would you expect this food to be served?”, we see a trend that Pizza is more appropriate for most situations, and shawarma and sushi are more specific in when they are expected to be served.

In figure **??**, we see a majority of participants expecting to pay a lower price for the food. Interestingly we see much

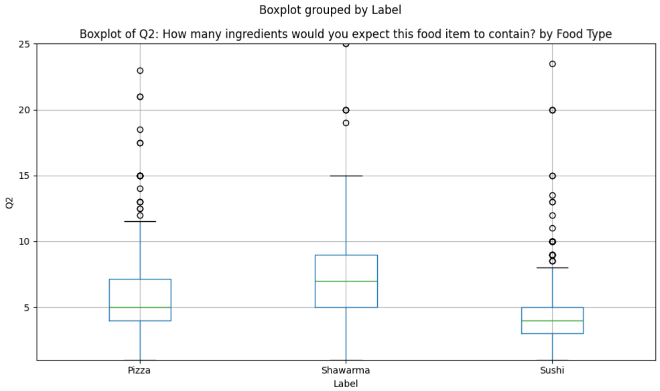

Fig. 2: Boxplot for Q2: How many ingredients would you expect the food to have

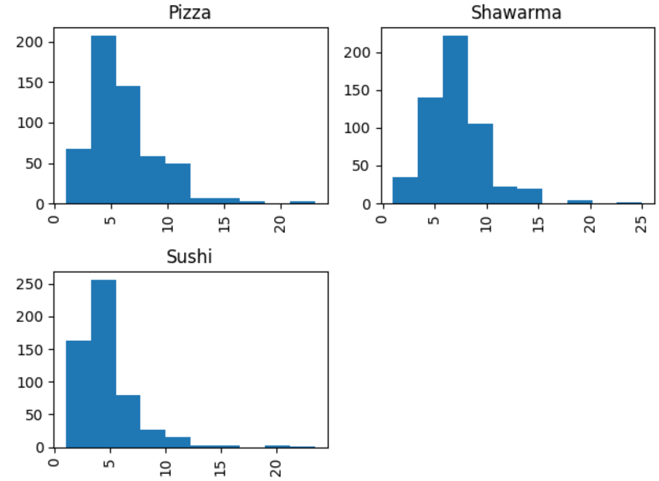

Fig. 3: Histogram for Q2: How many ingredients would you expect the food to have

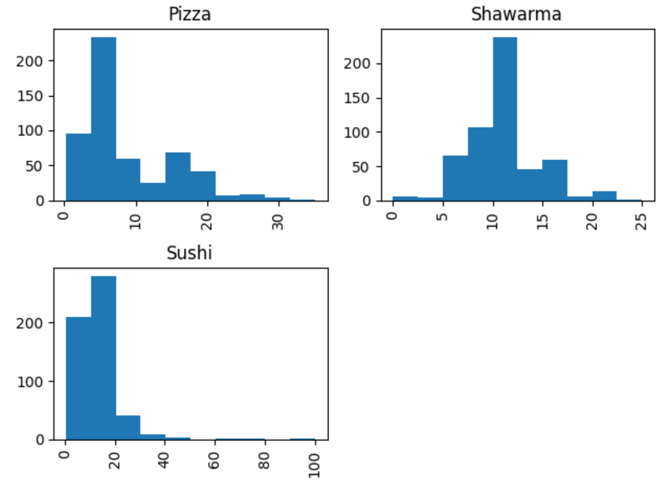

Fig. 4: Histogram for Q4: How much would you expect to pay for one serving of this food item?

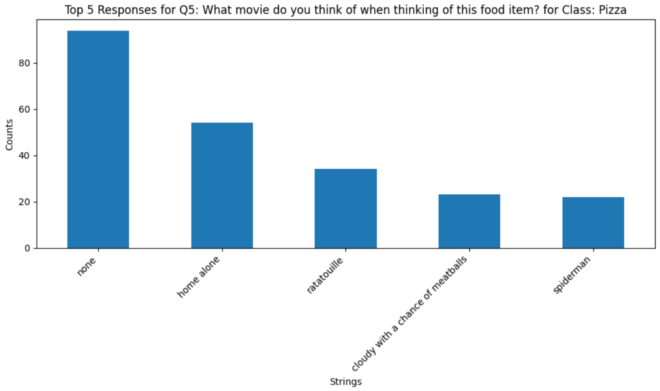

Fig. 5: Counts for Pizza for Q5: What movie do you think of when thinking of this food item?

higher outliers for sushi, with a peak of someone willing to pay 100 dollars for one serving of sushi. We also see different distributions for the three food items - pizza has a peak at 5 dollars, with a tail towards higher prices; shawarma has a normal distribution with mean at 10 dollars.

Figure **??**, **??**, **??** shows the top responses for each class for Q5. For both Pizza and Sushi, we saw “none” as the most popular input. However for Shawarma we saw “Avengers” to be by far the most popular response for Shawarma, suggesting a correlation between ”Avengers” and ”Shawarma”, making Q5 a good indicator.

For Q6: ”What drink would you pair with this food item?” we see a substainal difference between the answers for the three foods. With sushi participants preffered water, tea or sake. With Piazza there was a clear prefference for Cock-cola and other fizzy drinks and Shawarma had a tie between water and coca cola for the most frequent response with a siginficant minority preffering juice or nothing at all.

With regards to Q7: “when you think about this food item, who does this remind you of?”, there are no clear indicating responses between the food classes. For example, the most popular response accross the three foods for Q7 was “Friends”, with the remaining responses being, similar in size and frequency across the three food items. Therefore we decided to remove this Q7 from the parameters.

We split the dataset into 3 sets: 60% training, 20% validation and 20% test. This allowed us sufficient data to train the model as well as data for testing that the models generalizes to unseen data.

With regards to Q8, ”How much hot sauce would you add to this food item?”, the responses for Piazza and Sushi were extremely similar with the responses being exactly the same, but there was a very high preference for a moderate amount of hot sauce when considering Shawarama. As such this was a good feature to include.

II. Models

The setup of our training is similar for all models:

- We used the cleaned and encoded data as described in **??**.
- We split the data into training, validation and test sets using a 60/20/20 split using train_test_split from sklearn.

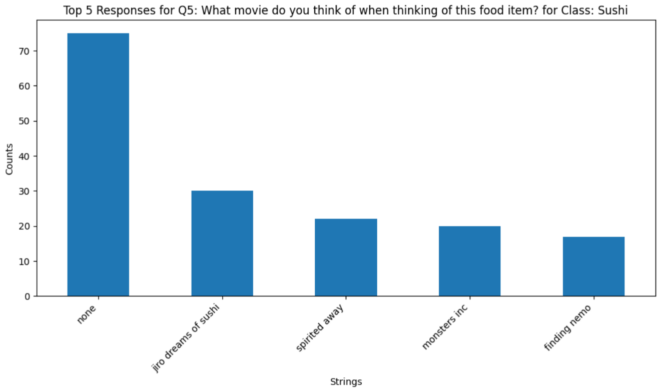

Fig. 6: Counts for Sushi for Q5: What movie do you think of when thinking of this food item?

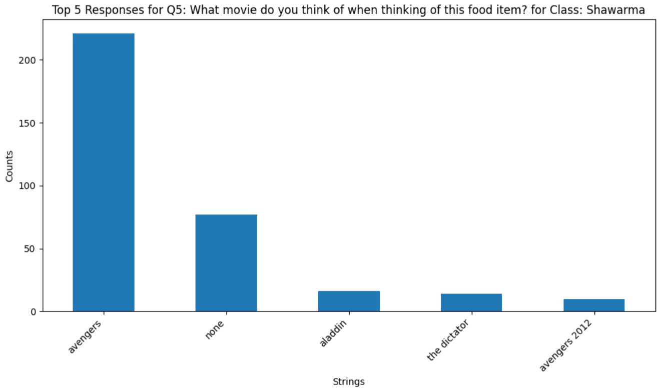

Fig. 7: Counts for Shawarma for Q5: What movie do you think of when thinking of this food item?

We used a consistent random seed of 42 to ensure reproducibility.

- We used stratified sampling to ensure that the distribution of classes is similar in both sets.
- We used the same evaluation metrics for all models: accuracy, precision, recall, and F1-score.
- We used bias_variance_decomp from mlxtend to calculate the bias and variance of our models.
- We used classification_report from sklearn to generate the classification report for our models.

A. Neural Network

We started with training a MLPClassifier on the cleaned and encoded data as described in **??** using the default parameters by sklearn. The default configuration is as follows: Hidden layer: 1 layer, 100 hidden units Activation: ReLu We achieved an 90% validation accuracy with this base configuration without any tuning, and 87% on the test set. We have also tested bagged models and normalization. Specifically, we used the sklearn BaggingClassifier with MLPClassifier as the estimator with 10 estimators. As we learned in class, we expected the bagged model to have lower variance because the prediction is the average of many MLPClassifiers. This is confirmed when we performed variance bias decomposition as seen in **??**, where the bagged model has a lower variance than the non-bagged model. We also tried normalizing the data using the StandardScaler from sklearn. We expected normalization to help the model converge faster and prevent overfitting, but it did not have a significant impact on the model accuracy. We think this is because the data was already cleaned and encoded, and the features were already in a similar range. For example, the numerical features like ”How much would you expect to pay” already have similar ranges (5-20) as the categorical features like ”How many ingredients would you expect to be in the food”.

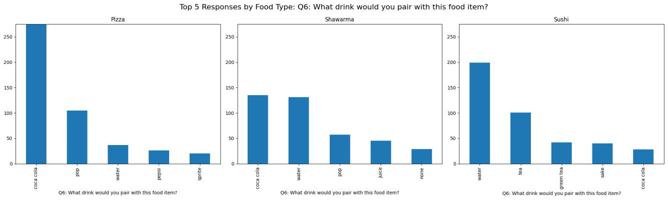

Fig. 8: Top 5 Responses for Q6

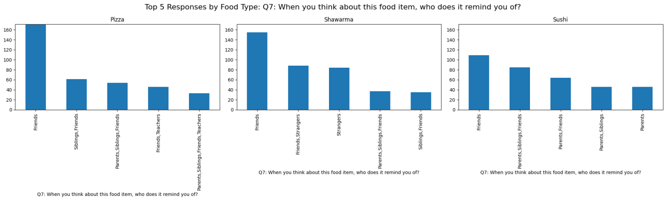

Fig. 9: Top 5 Responses for Q7

Model	Expected Loss	Bias	Variance
MLP	0.1356	0.1155	0.0638
Bagging MLP	0.1362	0.1277	0.0543
Normalized MLP	0.1368	0.1185	0.0628
Normalized Bagging MLP	0.1384	0.1337	0.0530

TABLE I: Loss, Bias, and Variance for Different Models

?? shows the classification report for the MLPClassifier. The model performed well on the validation set, with an accuracy of 90% and a balanced precision and recall across all classes. The F1-score for all classes was around 0.90, indicating a good balance between precision and recall.

Category	Precision	Recall	F1-score	Support
Pizza	0.92	0.90	0.91	88
Shawarma	0.87	0.91	0.89	88
Sushi	0.91	0.89	0.90	87
Accuracy			0.90	263
Macro avg	0.90	0.90	0.90	263
Weighted avg	0.90	0.90	0.90	263

TABLE II: Classification Report for Pizza, Shawarma, and Sushi

B. Decision Trees

I tested a regular decision tree, an ensemble of decision trees, and a Random Forest model, using the built-in Ran-

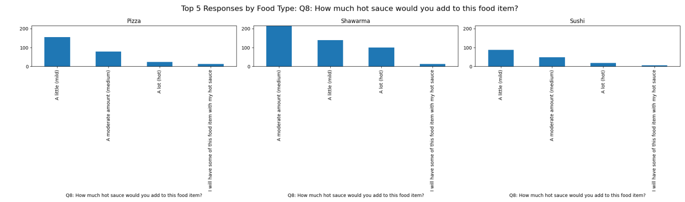

Fig. 10: Top 5 Responses for Q8

domForestClassifier from sklearn. I found the decision tree to perform the worst, and it was easier to manipulate the RandomForestClassifier's parameters, so I decided to explore that further.

I decided to include all of the possible features (even initially 'id' accidentally), as the Random Forest was very robust regardless of the feature choice or hyperparameters. To allow the text-based features to generalize as well as possible, I used a library to cluster text by Levenshtein distance, and to limit overfitting, I only included the categories with enough representatives. This did not seem to have too large an impact on validation accuracy, but would allow it to correctly categorize otherwise unseen data with unique typos or spelling choices. I tested many of the hyperparameter options, but found little change in accuracy aside from increasing the number of estimators to around 250, and limiting the minimum samples split to 10. Some randomness was introduced when generating text clusters, but I found it to still be very consistently accurate regardless of what data it trained on.

It consistently achieved an 87% validation accuracy. The classification report initially showed it was particularly effective at identifying Pizza and Shawarma, but had some difficulty with Sushi, which I would expect if respondents have less familiarity with Sushi. However, after splitting the training data such that there is the same amount of each in the training and validation portions, this was no longer the case.

Category	Precision	Recall	F1-score
Pizza	0.85	0.93	0.89
Shawarma	0.88	0.84	0.86
Sushi	0.90	0.86	0.88
Accuracy	0.88 (263 samples)		
Macro avg	0.88	0.88	0.88
Weighted avg	0.88	0.88	0.88

TABLE III: Classification Report

After comparing the loss, bias, variance decomposition to our other models, I found the expected higher bias, but lower variance that Random Forests are known to have:

Loss	Bias	Variance
0.1380	0.1255	0.0410

C. Word Embedding - Neural Network

We implemented a feedforward neural network with three hidden layers consisting of 64, 64, and 64 neurons, respectively. The hidden layers utilized ReLU activation functions, while the output layer employed a softmax activation function. The model was trained using a learning rate of 0.001 and a batch size of 32, with categorical cross-entropy as the loss function. Training was conducted over 300 epochs, achieving a validation accuracy of approximately 84%. For feature selection, we used word similarity embeddings for questions Q5, Q6, and Q7. For Q1, Q2, and Q4, we computed the average of all numerical values in the text, and for Q3 and Q8, we employed one-hot encoded categorical vectors.

We experimented with various configurations of the model, including adjustments to the number of hidden layers, neurons per layer, and activation functions. For activation functions, we tested leaky ReLU, sigmoid, and tanh. While leaky ReLU and tanh achieved validation accuracies of 82% and 84% respectively, the sigmoid function performed significantly worse, achieving only 65%.

We also explored different hidden layer configurations. Increasing the number of neurons to 128, 64, and 64 per layer slightly reduced the accuracy to 82% and increased training time. Reducing the neurons to 32, 16, and 16 per layer also resulted in a lower accuracy of 82%. A configuration of 32 neurons in each of the three layers yielded a similar accuracy of 82%.

Additionally, we tested varying the number of hidden layers while keeping each layer at 64 neurons. Reducing the number of layers to 1 or 2 led to validation accuracies of 82% and 83%, respectively. Increasing the number of layers to 4 or 5 did not improve performance, maintaining an accuracy of 84%.

D. Word Embedding - Linear Regression

We also evaluated a linear regression model. The implementation leveraged sklearn's Linear Regression module, utilizing the same feature set as the neural network. The model achieved a validation accuracy of approximately 86%, demonstrating competitive performance. This comparison with the neural network revealed that the simpler linear regression model performed slightly better, suggesting that the features representing the dataset may not exhibit significant complexity.

We tested multiple hyperparameter including setting 'fit_intercept' to False and 'positive' to False. Setting 'fit_intercept' to False resulted in a lower accuracy of 0.85%, while setting 'positive' to False yielded the same accuracy as the default setting.

E. Comparison of Word Embedding Models with Other Approaches

The Word Embedding Neural Network yielded an accuracy of about 84%, with a loss of 0.2269, a bias equal to its loss (0.2269), and a variance of 0.0731. The high bias suggests that it does not capture all complexity, while the variance stays relatively moderate. The Word Embedding Linear Regression achieved a slightly higher accuracy at 86%, accompanied by a loss of 0.2515, bias of 0.2515,

Category	Precision	Recall	F1-score
Pizza	0.86	0.83	0.85
Shawarma	0.85	0.85	0.85
Sushi	0.82	0.85	0.83
Accuracy	0.84 (329 samples)		
Macro avg	0.84	0.84	0.84
Weighted avg	0.84	0.84	0.84

Loss	Bias	Variance
0.2269	0.2269	0.0731

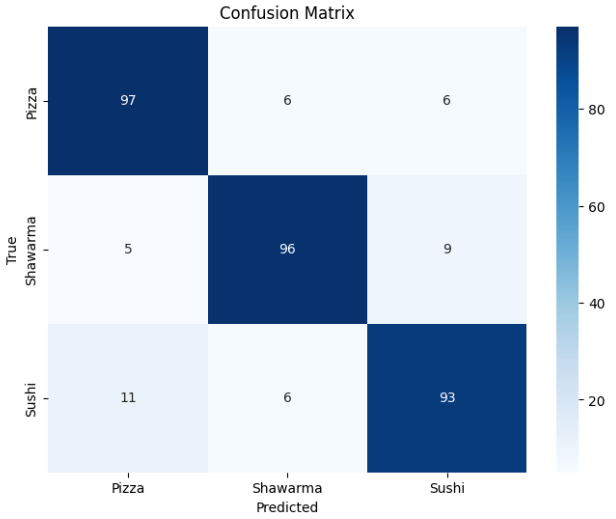

TABLE IV: Word Embedding Neural Network

Category	Precision	Recall	F1-score
Pizza	0.86	0.89	0.87
Shawarma	0.89	0.87	0.88
Sushi	0.86	0.85	0.85
Accuracy	0.87 (329 samples)		
Macro avg	0.87	0.87	0.87
Weighted avg	0.87	0.87	0.87

Loss	Bias	Variance
0.2515	0.2515	0.0088

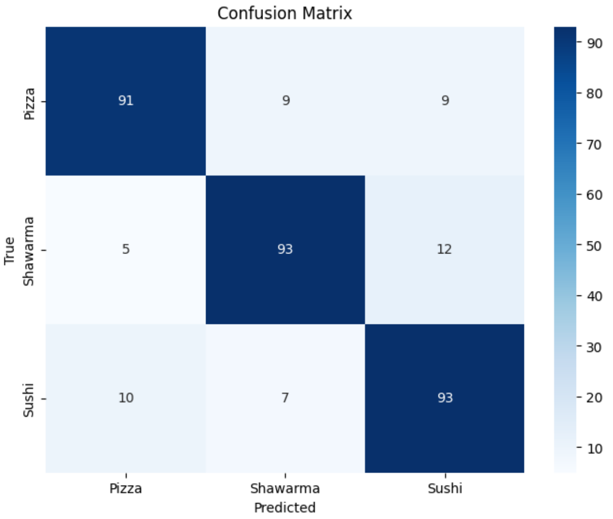

TABLE V: Word Embedding Linear Regression

and a notably lower variance of 0.0088. Though its loss was higher, the overall bias-variance profile indicates more stable predictions compared to the neural network variant.

Compared to other models such as the MLPClassifier (90% accuracy) and RandomForestClassifier (87% accuracy), these word embedding approaches showed somewhat lower accuracies. The neural network had a higher variance component, while the regression model leaned toward higher bias. They both demonstrate that the selected text-based features might not require very deep representations, and simpler linear methods can yield competitive results. The neural network's complexity may not be justified given the representation of the dataset, as indicated by the performance of the linear regression model.

III. Prediction

I would expect our model to perform with 80% accuracy on the test set. Our separated test accuracy resulted in 86%, and the test set the model will be predicting for is from a slightly different population, so it is reasonable to expect less accuracy. The given dataset and the test set however should be similar, and testing how it performs on less correlated data does give an idea as to how it may perform here. We have also tested the model on synthetic datasets generated by AIs and programatically, and found promising results to support the idea that the model has generalized well.

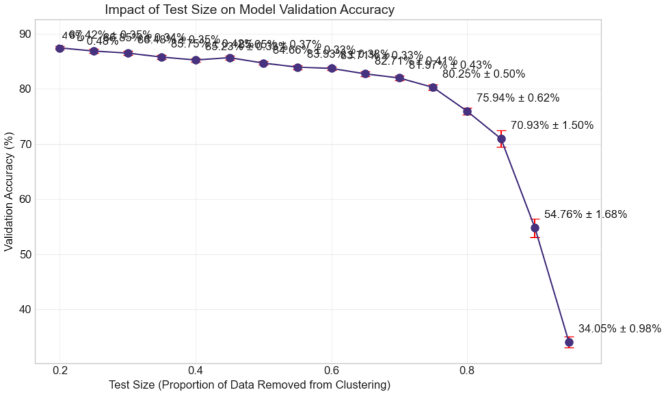

Fig. 11: Impact of smaller dataset on Validation Accuracy

IV. Workload Distribution

Ryan Shiels: Worked on the data cleaning and input functions, and explored Decision Trees/Random Forest models.

Siqi Liu: Data visualization and implementing the MLP Classifier.

Rafay Usman: Data cleaning and implementing/tuning the MLP Classifier.

Albert Li: Worked on Word Embedding models for both Neural Networks and Linear Regression, including Tables and Figures for the report.