# Speech Phoneme Analysis and Classification

Aiden Williams

372001L

Department of Artificial Intelligence

Faculty of ICT

University of Malta

June 2021

# Contents

# Table of Figures

Introduction

Word recognition is a common use for NLP application. Having computer devices understand the phrase ‘Ok Google’ or the word ‘Siri’ is expected. Undoubtedly phoneme recognition is part of the model behind recognition and classification of these words. It is with this perspective that in this report I will go over the data collection and classification process for the words ‘heed’, ‘hid’, ‘head’. All of the diagrams presented in this report where generated using the artefact submitted.

# Objective

The goal of this report is to outline the methods used to collect and process data consisting of the following feature labels:

Speaker, Gender, Word, Phoneme, Time, Formant1, Formant2, Formant3

It is also within the scope of this report, to describe and analyze this collected data. Finally, I will go over the methods used to create a KNN classifier using the sklearn python library.

# Methodology

This task was split into two parts: data collection and building a KNN classifier. To outline the process, I first gathered the data, then I analyzed and did some pre-processing on the data, then I build a KNN classifier and finally trained and tested the classifier using the pre-processed data.

## Data Collection

Data collection was done manually using the Praat software and Microsoft Excel, however any type of spreadsheet software would suffice. The process was as follows:

1. Load up the cwa_CT.wav file for each speaker.
2. Listen to the audio.
3. Identify the parts where the words ‘heed’, ‘hid’, ‘head’.
4. Select the start and end time and use the Zoom to selection feature.
5. Select 3-time ranges that correspond to the vowel phoneme of the 3 words.
6. For each selected range use the Formant Listing feature to see the formants.
7. Select the median listing.
8. Fill the Time, Formant1, Formant2 and Formant3 features using this listing.

The spreadsheet has the following features:

Speaker, Gender, Word, Phoneme, Time, Formant1, Formant2, Formant3

The speaker, gender fields are filled in 3 at a time (one for each word). These fields are taken from the file structure of the dataset.

The word and phoneme fields are filled with each run of the above process as it was decided that even if the actual phoneme of the word being said is changed from the official phoneme (due to the accent) I should still label each word phoneme with its official phoneme.

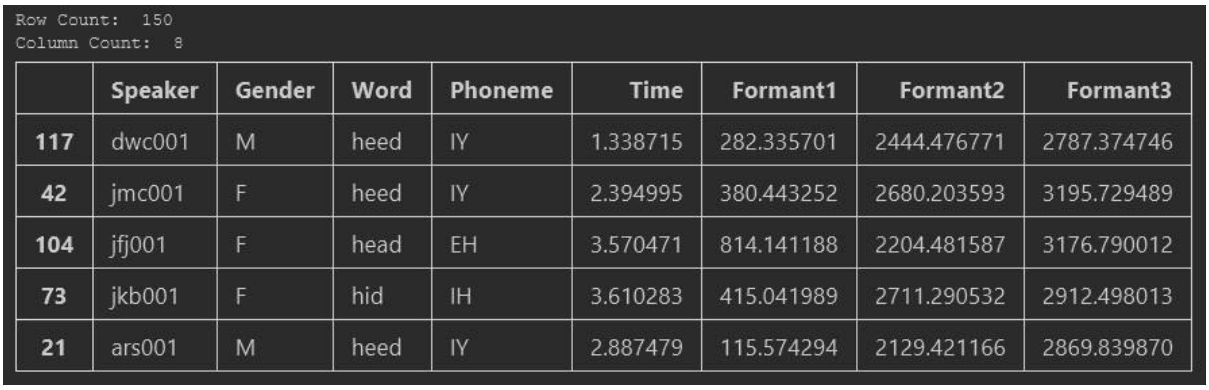
To speed up the process I would copy the listing from Praat and paste this into the Excel spreadsheet, into the Time field. Then I set up the column so that it can be separated, by space, into the next 3 fields.

The final step is to save the completed dataset as a .csv file.

## Data Analysis

In this step I load in the collected dataset into an IPython notebook and analyze it.

1. Using the pandas and sklearn libraries I load in and shuffle the dataset. Using the head function, I can confirm that the dataset has been loaded in correctly.

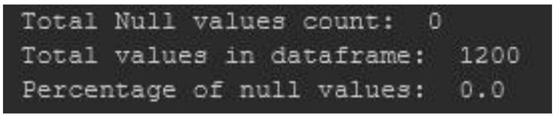


```
Row Count:  150
Column Count:  8
```

| | Speaker | Gender | Word | Phoneme | Time | Formant1 | Formant2 | Formant3 |
|---|---|---|---|---|---|---|---|---|
| **117** | dwc001 | M | heed | IY | 1.338715 | 282.335701 | 2444.476771 | 2787.374746 |
| **42** | jmc001 | F | heed | IY | 2.394995 | 380.443252 | 2680.203593 | 3195.729489 |
| **104** | jfj001 | F | head | EH | 3.570471 | 814.141188 | 2204.481587 | 3176.790012 |
| **73** | jkb001 | F | hid | IH | 3.610283 | 415.041989 | 2711.290532 | 2912.498013 |
| **21** | ars001 | M | heed | IY | 2.887479 | 115.574294 | 2129.421166 | 2869.839870 |

*Figure 1: Dataset head*

2. Then I check for any null values to make sure everything in the data set is computable.



```
Total Null values count:  0
Total values in dataframe:  1200
Percentage of null values:  0.0
```

*Figure 2: Null Value Check*

3. Then I use the matplotlib library to plot the distribution of the 3 formants.

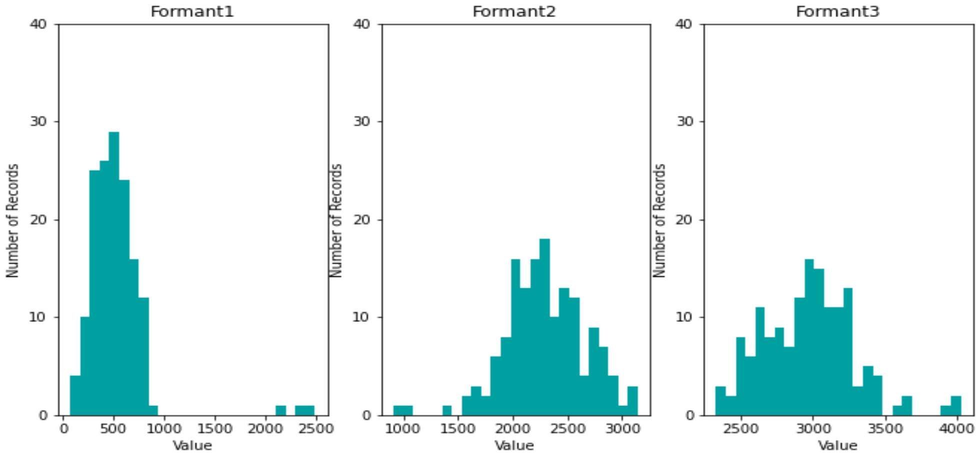


*Figure 3: Formant Distribution*

- From these plots I can see that the distribution is quite standard with a few outliers. I decided to keep these outliers since I was conscious that some entries are actual outliers and I wanted to see whether the KNN classifier would be able still predict these cases.

4. Finally, I plot the gender, word, and phoneme distribution to confirm I collected the data with the proper ratios.

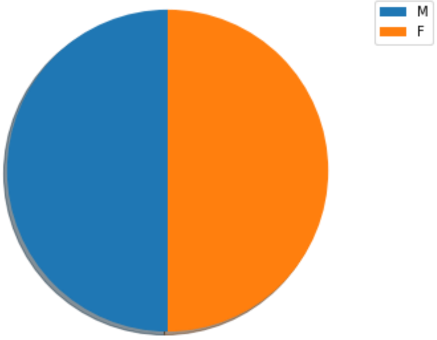


*Figure 4: Gender Distribution*

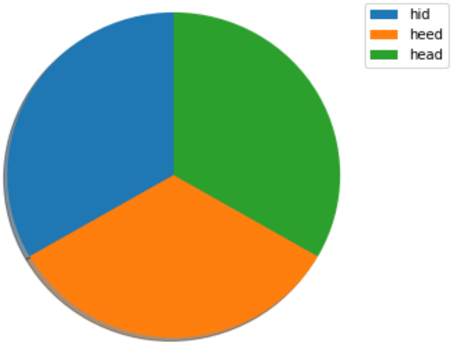


*Figure 5: Word Distribution*

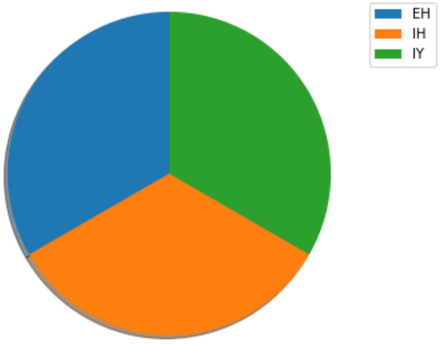


*Figure 6: Phoneme Distribution*

## KNN Classifier

To get a KNN-Classifier I used the sklearn library. From this library I also used a number of metrics.

The process was as follows:

1. Split the dataset into training and testing data
2. Initialize a `sklearn.neighbors.KNeighborsClassifier` object
3. Define several parameters
    a. From the documentation page I saw that by changing the p value I would be able to test out several distance metrics
4. Use the grid search tuning technique to get the best parameter set
5. Fit the training data
6. Predict the test data
7. Visualize the results

# Evaluation

The last step i.e., the KNN Classifier step was run a number of times. Due to page constraints, I will include 3 runs, 3 from the complete data set and 3 from the female set. For each run I kept the best parameters, the confusion matrix, and 4 metrics: accuracy score, fbeta_score, precision, and recall. In the notebook I show both the optimized model and unoptimized model's results for these 4 metrics, however in this report I am only including the optimized results due to page constraints.

These where the results:

*Table 1: Run 1 Results*

| Dataset | Complete | Female |
|---|---|---|
| Best Params | n_neighbors: 11, p: 1 | n_neighbors: 1, p: 2 |
| Accuracy | 73.68% | 52.63% |
| F-Score | 72.23% | 54.28% |
| Precision | 72.72% | 55.55% |
| Recall | 73.68% | 52.63% |
| Train/Test Split | 85/15 | 85/15 |

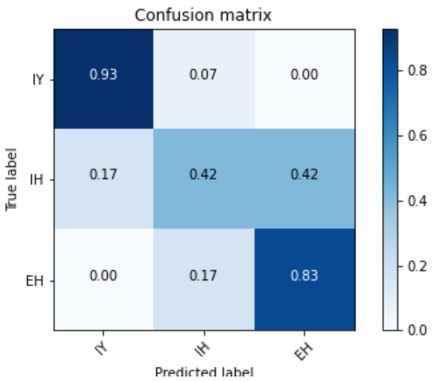


*Figure 7: Run 1 Full Dataset Confusion Matrix*

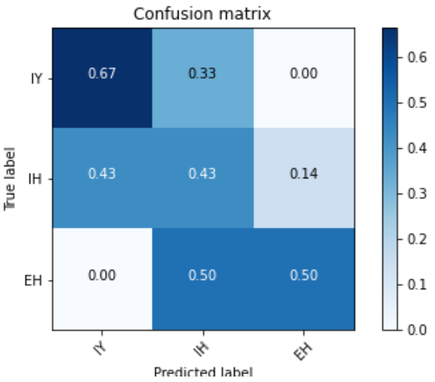


*Figure 8: Run 1 Female Dataset Confusion Matrix*

*Table 2: Run 2 Results*

| Dataset | Complete | Female |
|---|---|---|
| Best Params | n_neighbors: 3, p: 2 | n_neighbors: 9, p: 1 |
| Accuracy | 63.16% | 68.42% |
| F-Score | 62.23% | 66.45% |
| Precision | 65.07% | 66.84% |
| Recall | 63.16% | 68.42% |
| Train/Test Split | 80/20 | 80/20 |

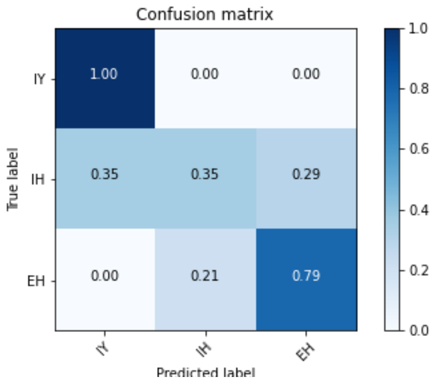


*Figure 9: Run 2 Full Dataset Confusion Matrix*

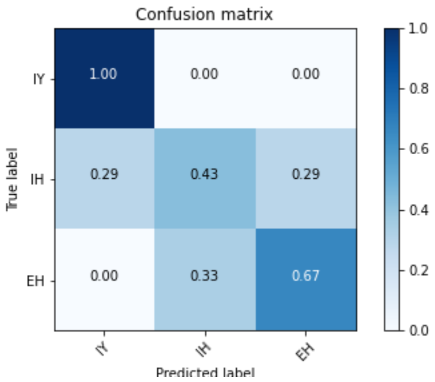


*Figure 10: Run 2 Full Dataset Confusion Matrix*

*Table 3: Run 3 Results*

| Dataset | Complete | Female |
|---|---|---|
| Best Params | n_neighbors: 7, p: 1 | n_neighbors: 7, p: 1 |
| Accuracy | 71.05% | 89.47% |
| F-Score | 68.81% | 89.47% |
| Precision | 68.69% | 89.47% |
| Recall | 71.05% | 89.47% |
| Train/Test Split | 75/25 | 75/25 |

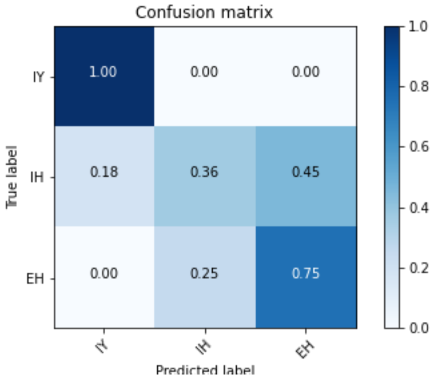


*Figure 11: Run 3 Full Dataset Confusion Matrix*

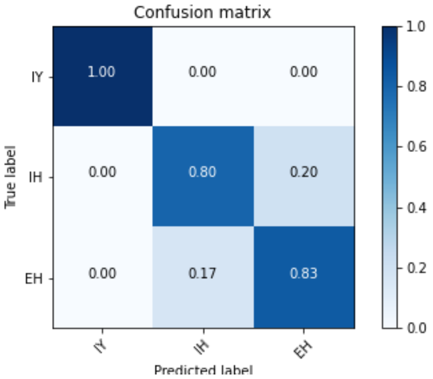


*Figure 12: Run 3 Female Dataset Confusion Matrix*

# Conclusion

From the number of tests, I ran I can see that p=1 (Manhattan Distance) results in the best results. For k, the results where more varied ranging from 1 to 11, with the best results coming from when k was 7. From the results I can also observe that the IH phoneme was the least predictable of the 3 phonemes. This phoneme was being attributed to the word ‘hid’ which during data collection I noted was the most variable between the accents gathered. I also noted that overall, the results between the entire dataset and the split female dataset were not significant. I expected the female dataset to achieve worse results due to less training data, but this did not happen. I also noted that as long the training/ testing split did not negatively affect the results as long as it was kept between 75/25 and 85/15. Overall, considering I knowingly collected and left in audible outliers the model functioned well and was always able to predict the phoneme better than at random chance (33%).