Recognising Spoken Digits

A Slidedoc by Alanna Manfredini

Project Background – Summary of Theory and Impact

In modern society there are many ways that speech analysis can be used to improve individuals' daily lives. For example, 'Siri', the digital assistant on the iPhone, is used by activating the Siri program, and speaking a command to it. The iPhone will then analyse the recorded frequencies from the voice, transform it into a command, and execute said command ("What Can I Ask Siri? - Official Apple Support").

Siri and other Digital Assistants must be able to take the frequencies of the spoken words, often spoken with different accents and background noise, and determine which words were spoken. Therefore the program must be able to identify the characteristic elements of a word, no matter the environment in which it was uttered. The desire to identify elements led to the field of speech analysis.

To determine the words uttered, each word may be broken into its constituent phonemes. A phoneme is the minimal contrastive unit of a word, where substituting one phoneme for another changes the meaning of the word (Binder et al., "Phoneme"). Basically, a phoneme describes the unique sounds a word can be broken up into, which can be interpreted as a word depending on the order in which the phonemes are spoken.

As mentioned previously, high accuracy speech analysis programs must not be affected by word variation: allophones, equivalent phonemes that do not change the meaning of a word must be taken into account (Binder et al., "Allophone"). Analysis can be done by finding the probability of a word being represented by a certain set of phonemes and marginalising over the various allophones that could have been substituted into the word. For example, after an utterance of the word river, the machine would combine the probabilities of the word being pronounced in a British accent, /ˈrɪv.əʳ/, or an American accent, /ˈrɪv.ə˞/ (Cambridge University Press). In fact, for higher accuracy models, as well as phonemic variation; temporal structure, the time over which a word is spoken; prosody, the intonations in a voice that distinguish meaning; voice timbre; and quality (Korvel et al.), the emotion of a speakers' voice may affect the utterance sound and hence must be nuisance variables incorporated into the model (Badshah et al.).

There are various approaches to analysing words, such as Hidden Markov Models, Nearest Neighbours, Support Vector Machines and Artificial Neural Networks. These approaches often create a spectrogram of the utterance and do visual analysis to the image. (Korvel et al.)

This project, however, worked with Mel Frequency Cepstral Coefficients (MFCCs). MFCCs are determined by first dividing the recorded sound into analysis windows. Next the data is transformed by a Hamming Window to minimise any bias due spectral leakage: the result when a periodic function is estimated as non periodic (Alam et al.). The equation for a Hamming window is

$$L_F^{(0,1,-1,0)}(u) = \frac{(1-i)[u^2(1-2\beta)+4\pi^2\beta]\sin(\frac{u}{2})}{u\sqrt{\pi}(4\pi^2-u^2)}$$ (Goel and Singh).

Then the Discrete Fourier Transform (DFT) is taken of each analysis window to produce a spectrum of frequencies. The windows must be created since a FFT assumes a stationary signal. Finally, the magnitude spectrum produced by the DFT is warped into a Mel Specturm. The Mel Spectrum represents how humans hear certain frequencies. (Rudd et al.). Finally, an inverse DFT is run to produce the MFCCs.

Project Background – Expectations and Applications

Expectations

This project aimed to created a program which could accurately identify the Arabic digits from 0 through 9 (Table 1).

By determining a variety of models which can accurately represent the clustering of the cepstral coefficients of the data, it is possible to observe the bias variance trade-off between models. A more flexible model requires much more processing power and will create a very detailed model which should fit the training data very accurately. However, when it is tested on the testing data, since there is a strong bias towards the training data, the testing data may not be mapped as accurately. Conversely, with the most rigid model, it will not have much bias because it is set to such rigid constraints. Therefore it will not match the testing data very accurately, due to a broad model causing a poor fit.

By testing a variety of models, it is possible to find the ideal point where there is a flexible but unbiased model, which can accurately map the data. This was especially evidenced in an initial model that is not included in this slidedoc: the single phoneme seven. As can be observed in a future slide, the variation of the MFCCs for digit seven is very slight, making it look like there was only one phoneme and only one cluster. However, if clustering was done with only one cluster, all of the data for all of the test digits would fit inside this cluster. Therefore it was arbitrary which of the test data would fit the model, which resulted in a model accuracy of 9.5%, with the model actually predicting a seven to be four with a probability of 33%.

Applications of these Models

As mentioned previously, the most common application of phonetic analysis is in speech to text programs. This can been seen in digital assistants and dictation packages commonly used on phones and in dictation during radiology diagnoses. More recently there have been breakthroughs in using a combination of audio and video processing to create more accurate speech recognition. This is yet another variable that could be incorporated into the model (Biswas et al.).

Other applications of analysing audio recordings with machine learning include determining the types of sounds in an environment. This could be by analysing the genre of a music to create an enjoyable music app (Rosner and Kostek) or even for biometric authentication (Czyzewski et al.).

Digits	Arabic Word
0	sifir
1	wahad
2	ithnayn
3	thalatha
4	araba'a
5	khamsa
6	sittah
7	seb'a
8	thamanieh
9	tis'ah

Table 1: Digits

Other examples of using machine learning to cluster data and create a model would be in computer vision. By grouping like colours and shapes, through patterns of rasterised pixels, in specific clusters, it would be possible to determine if those colours or shapes appeared in other images.

Another example of where clustering could be used is in encryption decoding. If it is possible to detect clusters within a code it would be possible to find if there is any filler code that was intended to throw off the person trying to decode the messages.

Similarly, genomes have large chunks of repeating DNA sequences. As can be seen with AlphaFold, by grouping junk DNA, it could be extracted from clusters of genes. This could be used to determine an organism's ancestry.

General Overview

The Data

Both testing and training data was downloaded from the UCI Machine Learning Repository (UCI). The data was presented in a .txt file. Each line of the file had 13 MFCCs corresponding to a single analysis window. After approximately 40 analysis windows there was a blank line. This divided the data up into 'paragraphs', where each paragraph corresponded to the data for a single utterance of a digit.

In the training data, the first 660 utterances were for the digit 0, the next 660 for the digit 1, etc. Within those 660 utterances, the first 330 were by males and the second half were by females. Each 10 successive utterances were by the same speaker. There were 6600 utterances by 66 speakers total for the training data.

Similarly, in the test data, the first 220 utterances were for the digit 0, etc. These groups of utterances were likewise divided by 50% male 50% female and each speaker recited each digit 10 times. There were 2200 utterances by 22 speakers total for the testing data.

The Process

The data was parsed into a Python Pandas DataFrame of 16 column with a row for each utterance. The first 3 columns encoded the digit, speaker ID and speaker gender. The next 13 columns corresponded to each of the MFCCs. For example. a single cell in the "MFCC1" column contained all the MFCC1s for a specific utterance.

The MFCCs for each digit were plotted and any sudden change in cepstral coefficient value was noted as a different phoneme. Therefore the number of phonemes were able to be counted. Since, by definition, each utterance of a specific digit should have the same number of phonemes as another utterance of the same digit, the data can be grouped into clusters where each cluster corresponds to a different phoneme.

This clustering was determined using a kmeans and an expectation maximisation method.

Testing data was fitted to these models and the accuracy of the model classifying digits was presented in a confusion matrix

The Models

Six models were run throughout this experiment. They started with the most flexible model and decreased in complexity towards the most rigid model. This resulted in an ideal model which corresponded to the intersection of the bias variance trade-off.

The first model used all 13 MFCCs and full covariance.

The second model had 4 MFCCs, with full covariance.

The third model used initialisation by dividing the data into four rather than the kmeans++ default initialisation.

The fourth model increased the cluster numbers to try to account for the transition periods between clusters

The fifth model divided the training data by gender as well as digits. This was to try to account for the different frequencies of male vs female voices. The probability of the test data being in both the male and female models for each digit were summed.

The sixth model was created with a diagonal covariance. This was the most restrictive of the models.

Determining Phonemes - Initial Plotting

To determine the number of phonemes in each digit, a variety of plots of the MFCCs vs analysis windows were created. Whilst ideally all of the utterances would have been plotted on top of each other, the data was not presented in a temporally scaled form, because the training subjects did not speak the digits in the same amount of time and some of the recordings had silence before and after the digit. Therefore plots of the MFCCs vs time for each utterance were shifted from each other temporally. Hence when all of the utterances were plotted on top of each other, there was too much variation and the graph just became too noisy (a blob). To account for this, only the first 10 utterances were plotted on top of each other. 10 was chosen as it appeared to be enough utterances that trends were able to be determined without being influenced by any errors, but not so many that the temporal shifting obfuscated the delineations between phonemes.

Throughout this presentation, each MFCC is represented by a specific colour (Figure 1). The next three slides depict a plot of the MFCCs vs analysis window as they were given in the data. For many of the plots, it is difficult to tell where the separation in phonemes are, but it is possible to determine the general shape. As the slides progress, the MFCC changes are less obvious since the order of the MFCCs correspond to the Fourier transform and hence the decreasing importance of the sinusoids that makes up the original frequencies. That being said, however, it is important to have multiple MFCCs to analyse to determine the phonemes, because, for some of the digits, the first MFCC does not have any variation over the word, even though it is the most important. Without any variation of the MFCCs, it is impossible to tell the amount of phonemes in a word.

It is also interesting to note that there is no clear distinction between higher MFCC female voices vs lower MFCC male voices. This will be further analysed in the second model.

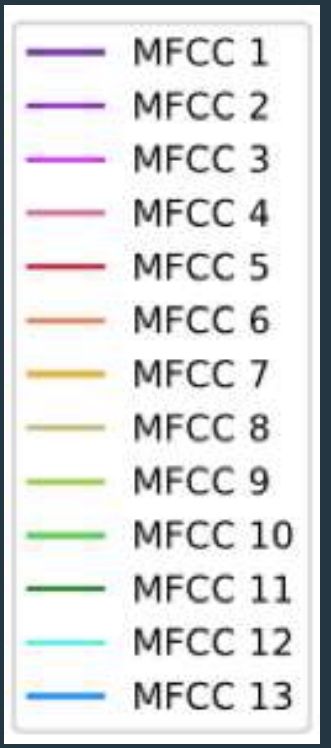
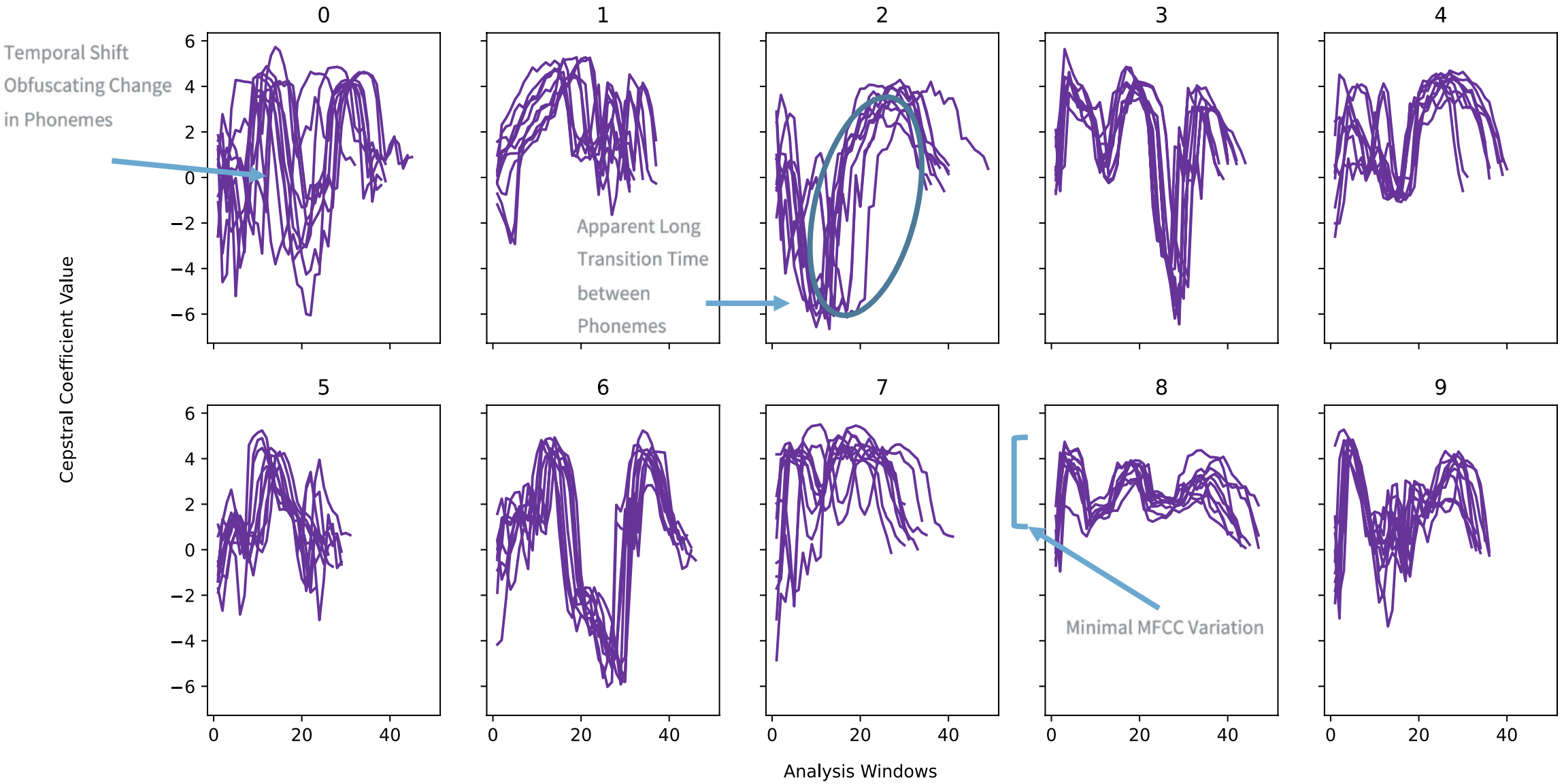
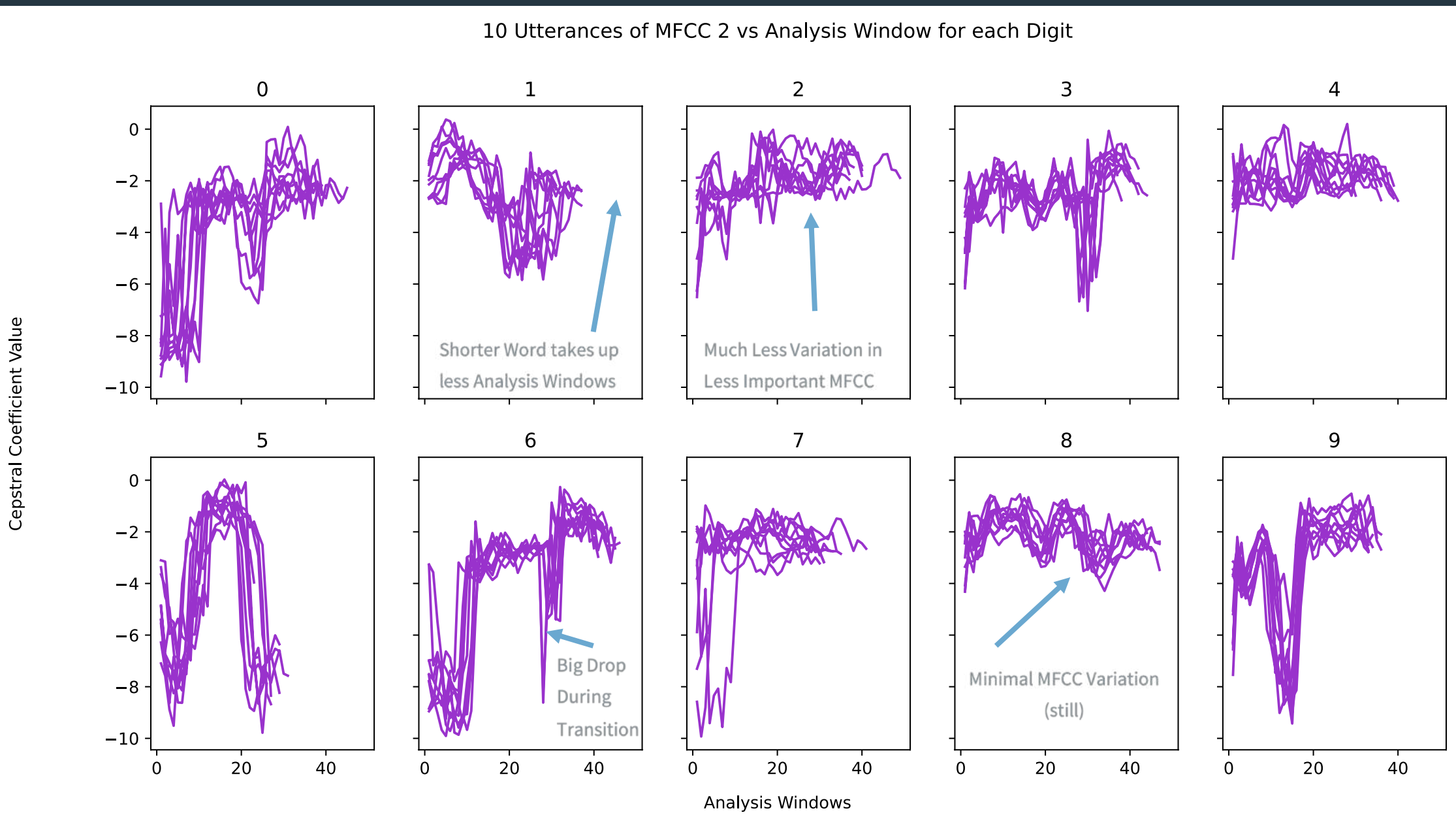


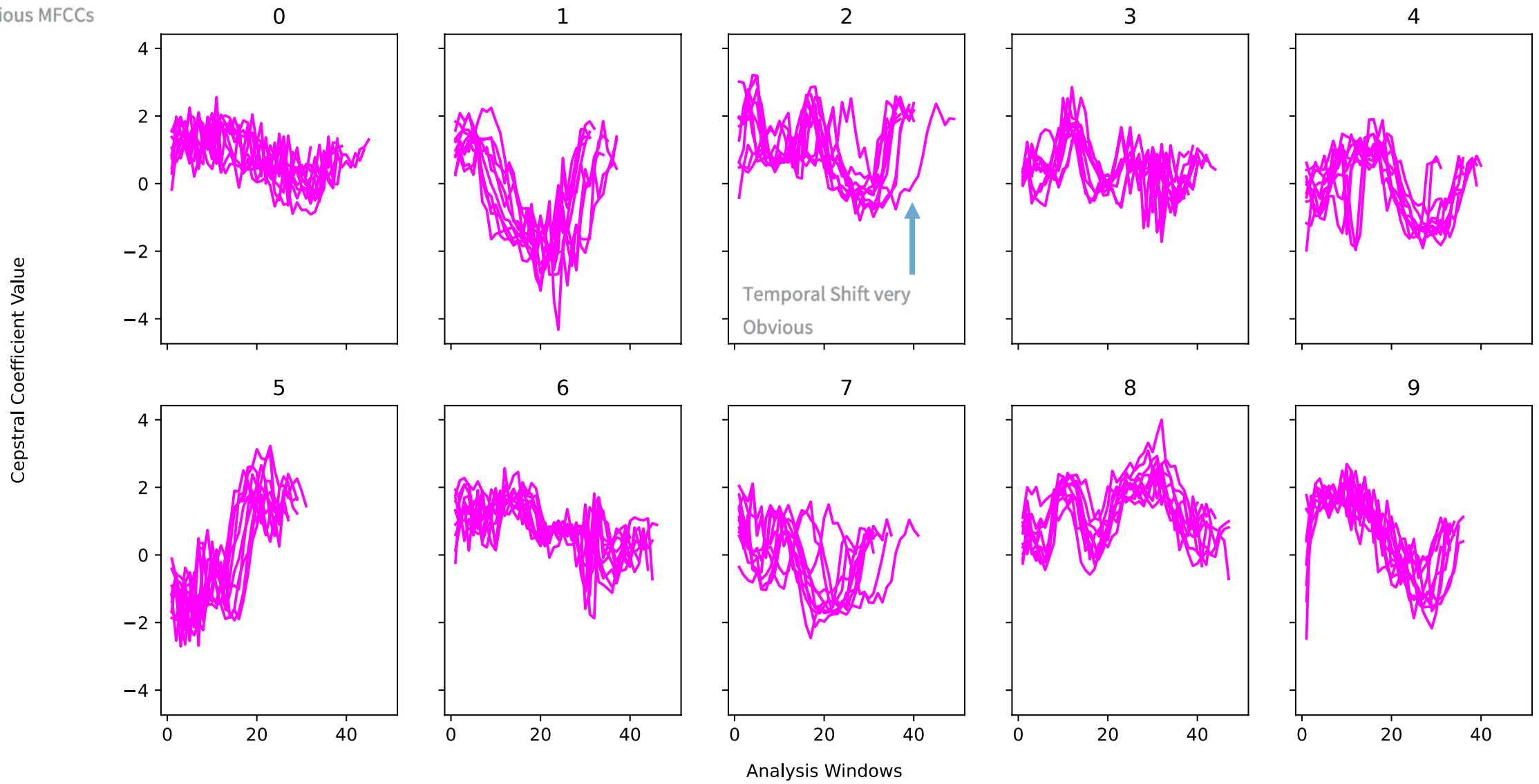
Figure 1. Phoneme Colours used throughout the Presentation


10 Utterances of MFCC 1 vs Analysis Window for each Digit
0
1
2
3
4
5
6
7
8
9
Temporal Shift
Obfuscating Change
in Phonemes
Apparent Long
Transition Time
between
Phonemes
Minimal MFCC Variation
Cepstral Coefficient Value
Analysis Windows
6
4
2
0
−2
−4
−6
0
20
40


10 Utterances of MFCC 2 vs Analysis Window for each Digit
0
1
2
3
4
5
6
7
8
9
Cepstral Coefficient Value
Analysis Windows
Shorter Word takes up
less Analysis Windows
Much Less Variation in
Less Important MFCC
Big Drop
During
Transition
Minimal MFCC Variation
(still)

10 Utterances of MFCC 3 vs Analysis Window for each Digit

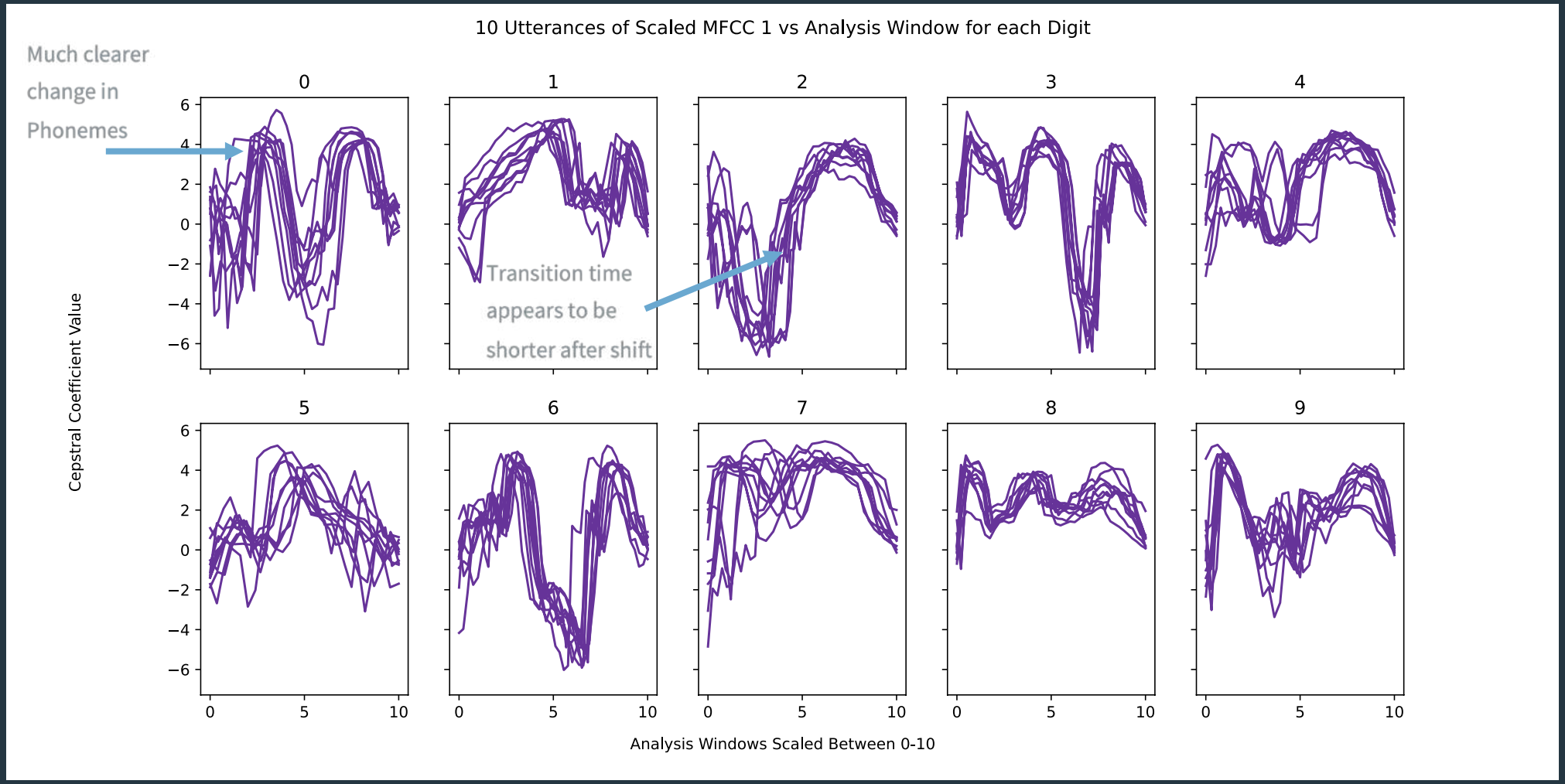
Much Less Variation than Previous MFCCs

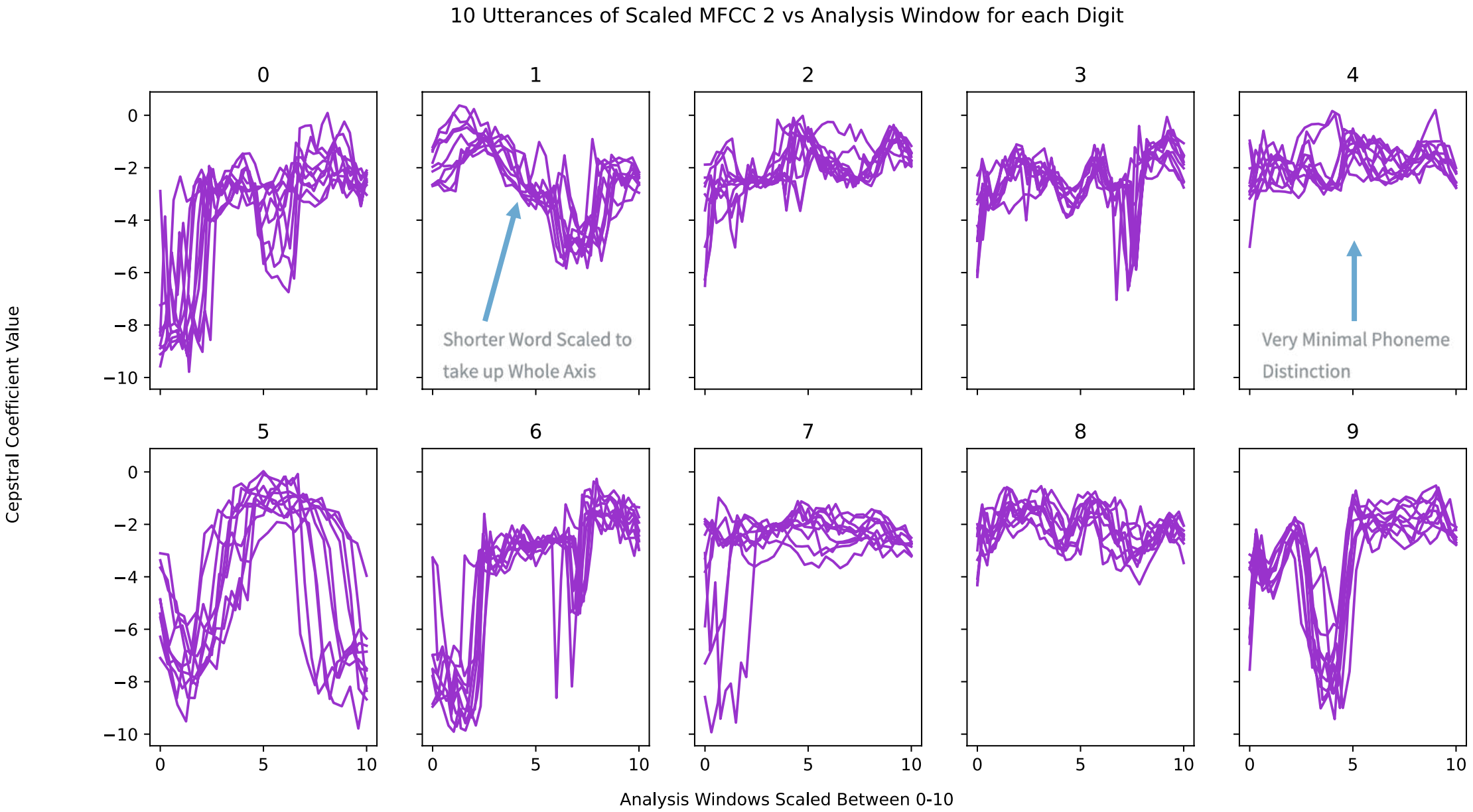


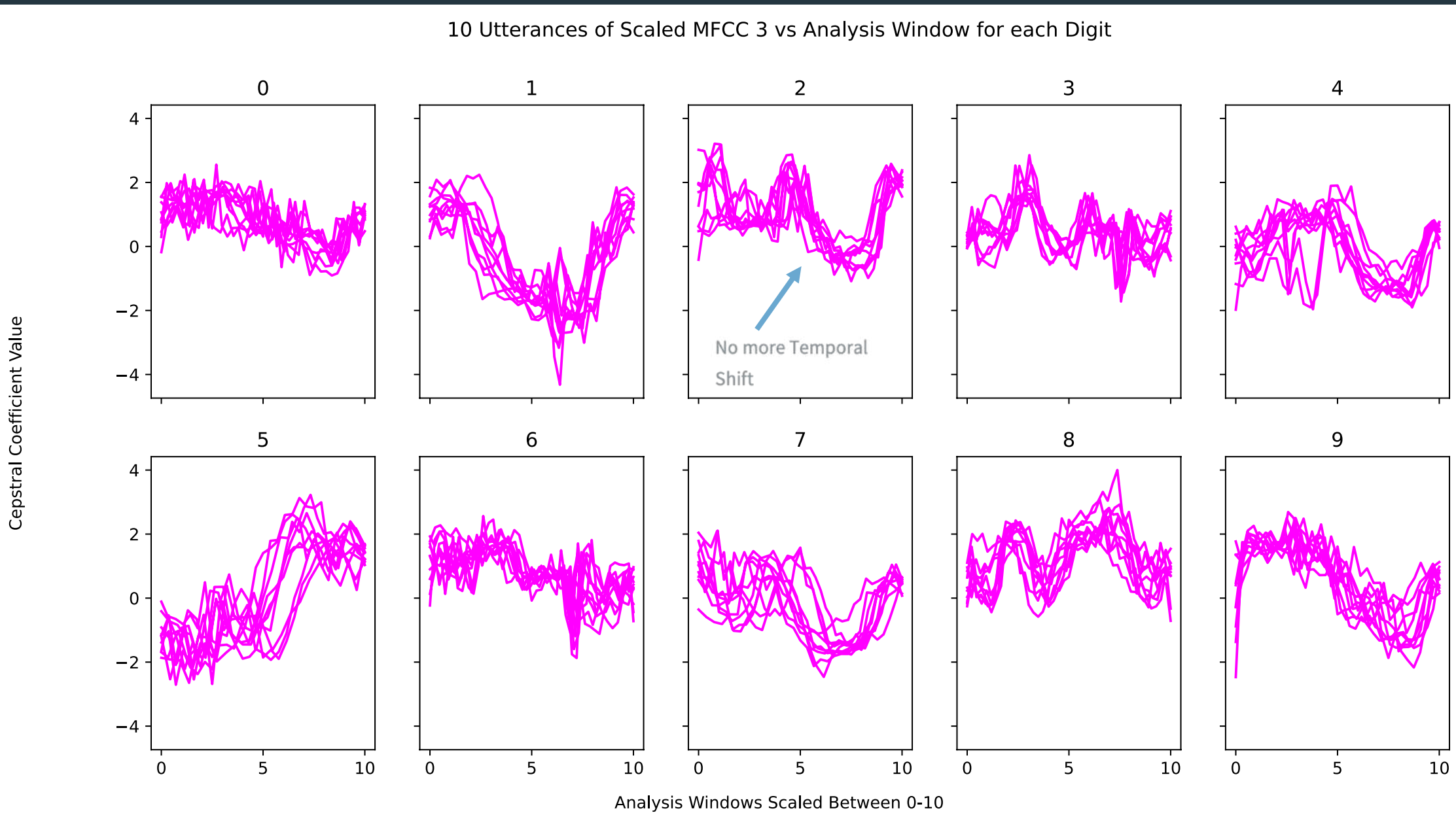
Determining Phonemes – Plotting with a Temporal Shift

Some of the main issues with the previous technique of plotting vs the analysis window were due to the temporal shift, the long transition times and the shorter words being squished into a limited space.

To combat this, the first 10 utterances were plotted again, but each of the utterance times were shifted to stretch between 0 and 10.




10 Utterances of Scaled MFCC 2 vs Analysis Window for each Digit
0
1
2
3
4
5
6
7
8
9
Shorter Word Scaled to take up Whole Axis
Very Minimal Phoneme Distinction
Cepstral Coefficient Value
Analysis Windows Scaled Between 0-10


10 Utterances of Scaled MFCC 3 vs Analysis Window for each Digit
0
1
2
3
4
5
6
7
8
9
No more Temporal
Shift
Cepstral Coefficient Value
Analysis Windows Scaled Between 0-10

Creating Models for the Digits

After improving the plotting of the MFCCs vs analysis windows to more explicitly emphasise the phonemes, each of the MFCCs for a specific digit were plotted in subplots. On these subplots it was possible to see if there were any outliers in the MFCCs which may denote an interesting way to model the data. Then, to find the phonemes, the first 10 utterances of MFCC 1 were plotted vs scaled and unscaled time axes and the first utterance of all the MFCCs were plotted. The distinct phoneme shapes for each of these plots were boxed and compared to ensure the correct number of clusters would be chosen.

Often one of the plots would be much more clear in showing distinct phoneme groups than the others and therefore having three methods increased the accuracy of the choice of phoneme numbers.

By boxing the phonemes, it was possible to predict whether the model would be affected by increasing the cluster numbers to account for transition times. Additionally, it was possible to tell whether certain MFCCs may be dominant in certain digits over others due to more severe changes between phonemes.

Kmeans

Kmeans is a clustering algorithm that is usually used for data mining. (Jung et al.) It is an important mechanism for minimising the distance between datapoints and the cluster centres of the groups of datapoints. These cluster centres are described by $\vec{\mu} = \frac{1}{|\omega|}\Sigma_{\vec{x}\,\epsilon\omega}\,\vec{x}$. (Barber)

In the execution of kmeans, the centres are determined and then the datapoints closest to each centre is assigned to the specific cluster corresponding to said centre. The centres may be determined randomly, or may be assigned. After the cluster assignments are done the first time a new set of cluster centres are determined using the previous formula and the process is repeated. The process may be repeated for a number of iterations or until the optimal centre and cluster assignments is found.

Kmeans is a reliable way of assigning clusters, however it is highly dependent on the original initialisation. If the original initialisation is very poor, datapoints may be assigned to the incorrect clusters and the residual sum of squares may converge to a local, not global, minimum (“K-Means”).

Expectation Maximisation (EM)

Expectation Maximisation is also a clustering algorithm used for data mining, however it involves assigning probabilities datapoints are in a cluster rather than assigning clusters specifically.

To assign these probabilities, the algorithm alternates between guessing a probability distribution for the data based off initial cluster centres (expectation) and re-estimating model parameters using these probabilities (maximisation). More specifically, expectation involves finding the distributions that maximises $\log p(v|\theta) \geq \hat{L}(\{q\},\theta)$ and the maximisation fixes $q(h^n|v^n)$ and maximises the model parameters. Often, as is seen throughout this slidedoc, this involves assuming the datapoints inside a specific cluster fall in a Gaussian distribution. Therefore the probability of data being inside a cluster is computed by the probability of the data being picked from a gaussian distribution described by the algorithms model parameters. The most important model parameters are the cluster means, the cluster covariances and the ”pi” probability values (Barber).

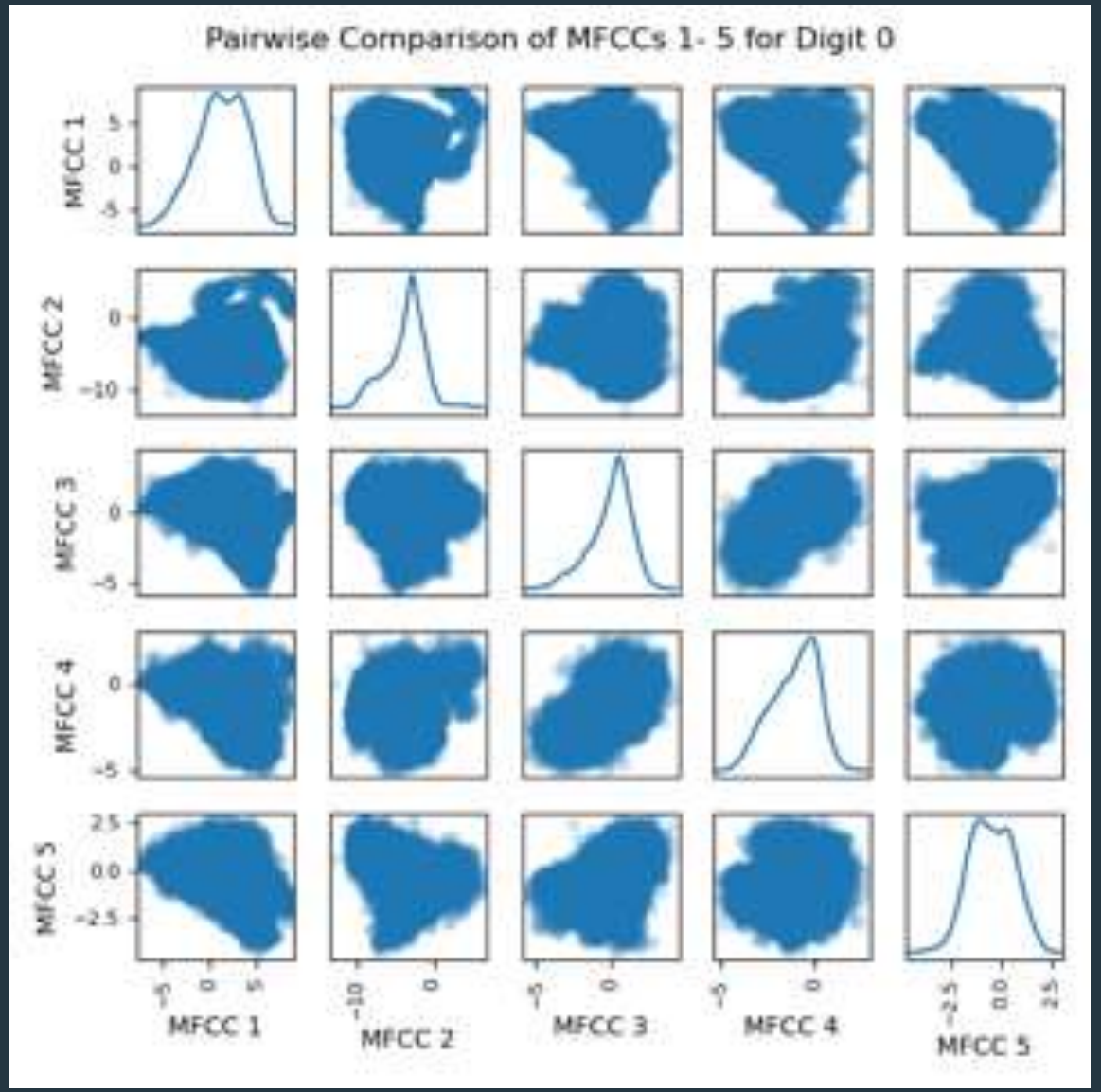
Digit 0 - sifir

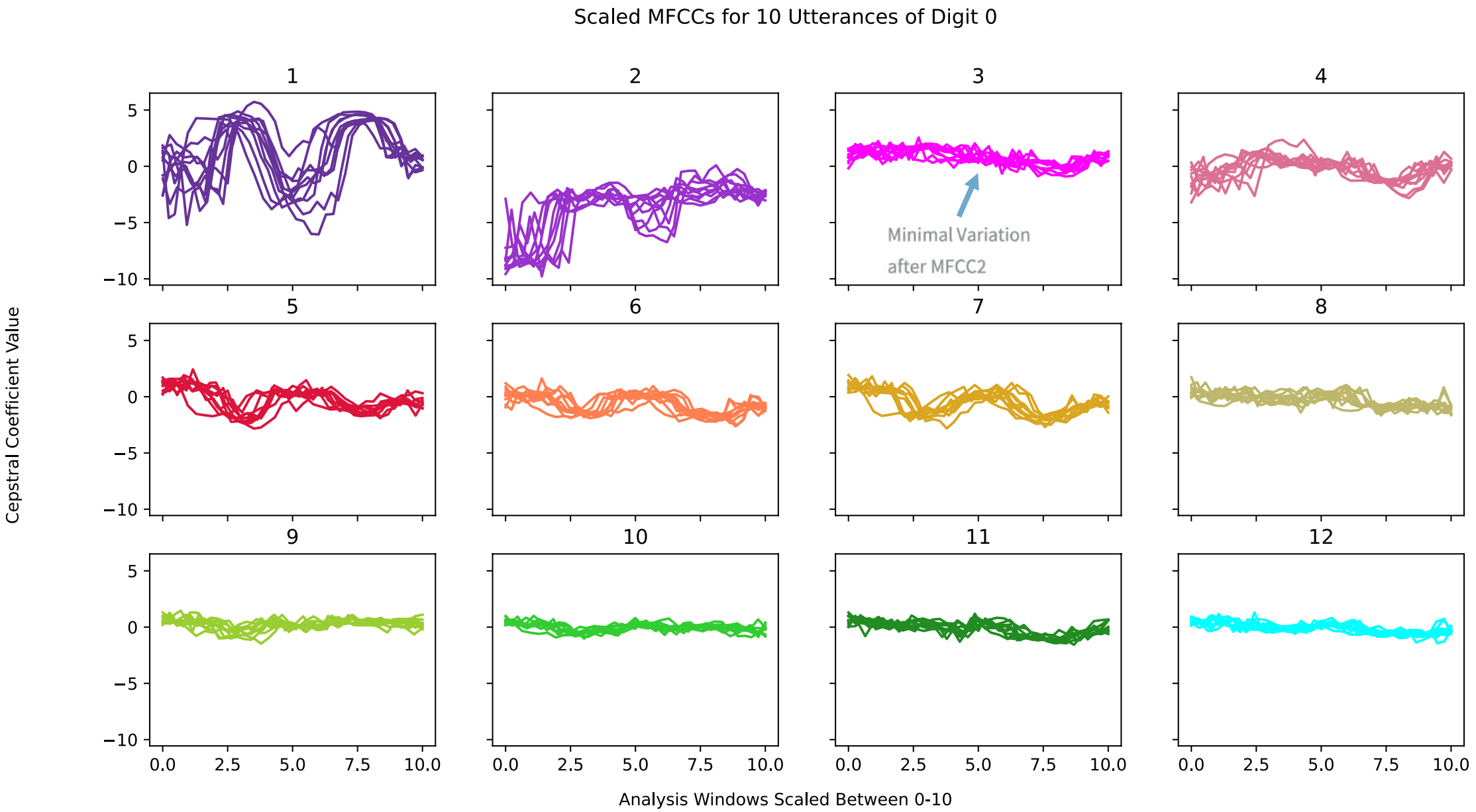
الشفرات sɯ'fɯr

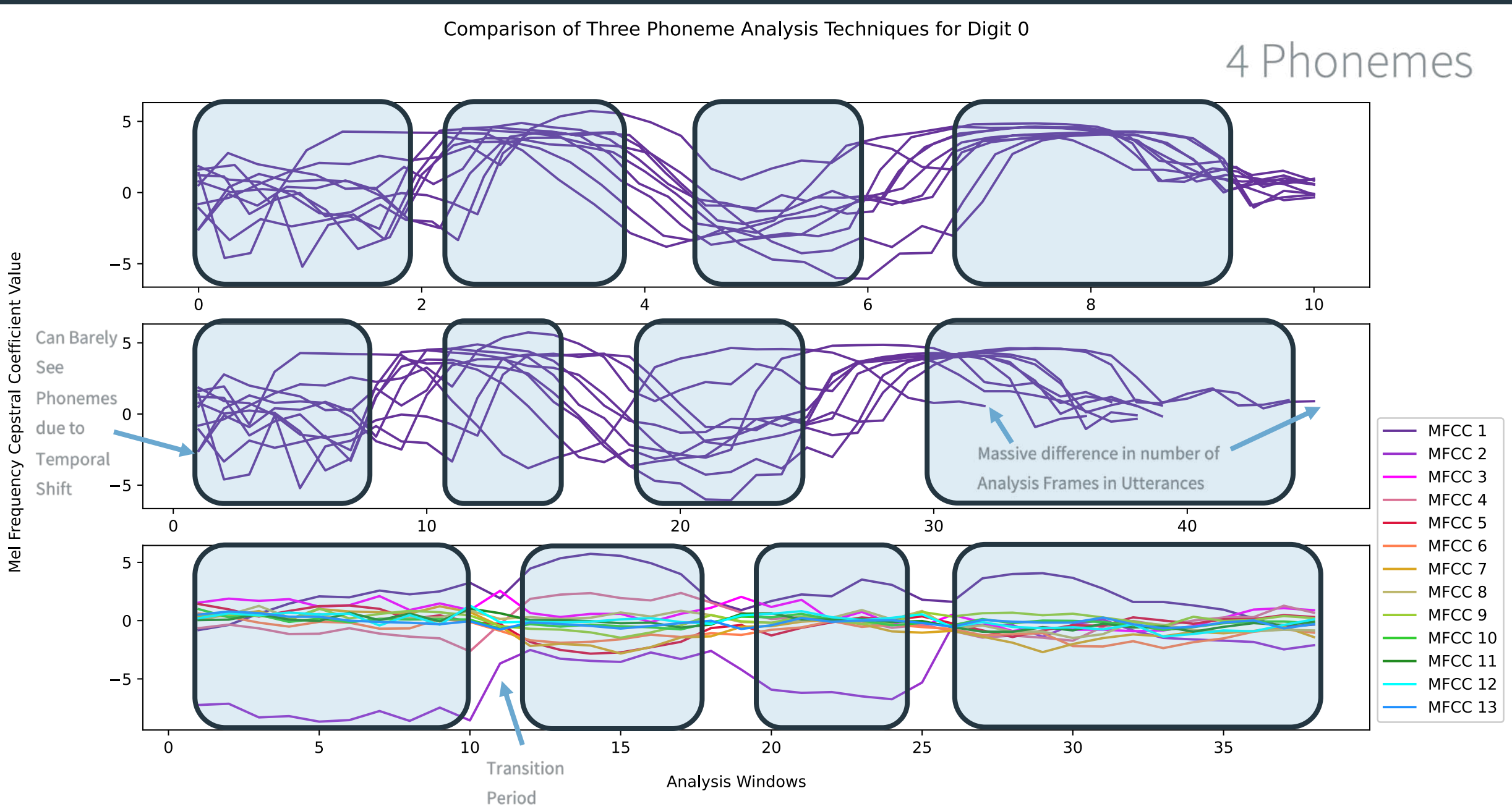
Pairwise Comparison

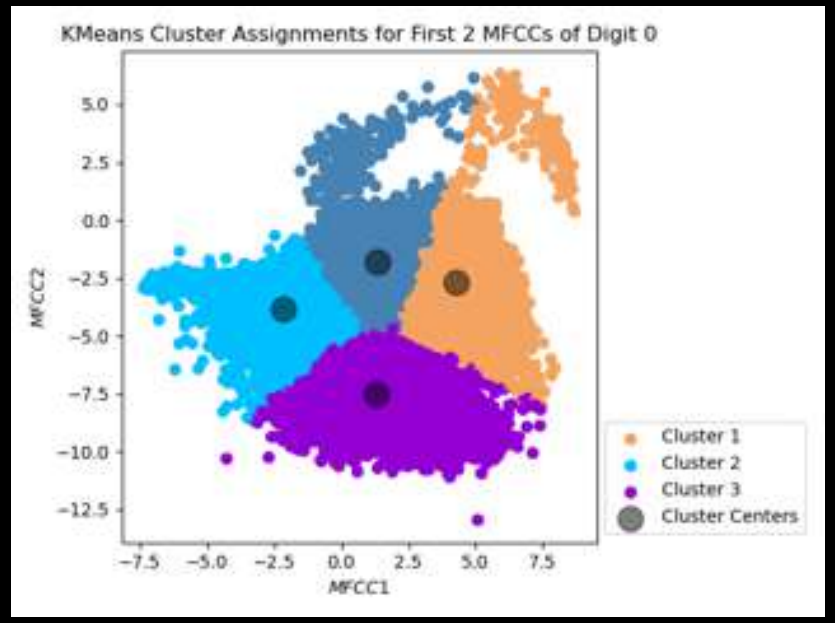
The pairwise comparison plot of MFCCs is one of a few representations that highlights the decreasing importance of the MFCCs. It is possible to observe that the first few comparison plots have very distinct shapes. Then as the MFCC value increases the comparison plots become much more circular, highlighting that there is less information that can be gleaned from the MFCC. Additionally, the histogram down the diagonal highlights that as the MFCC increases it tends towards a gaussian distribution.

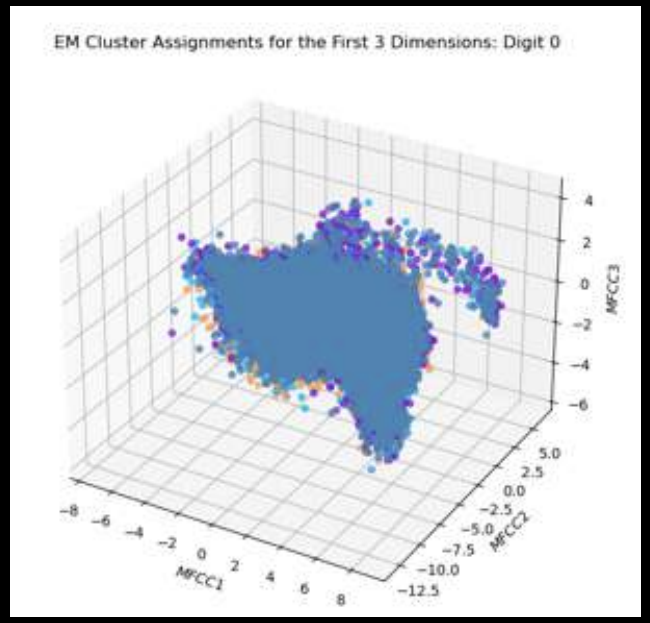
Throughout this project many of the models only use four dimensions of MFCCs, because after this point there is diminishing returns on the accuracy of the model versus the time and processing power needed to produce the clustering. This would be especially evident in models such as the gender division model, where there is not enough data to produce an accurate full covariance matrix for all 13 dimensions.

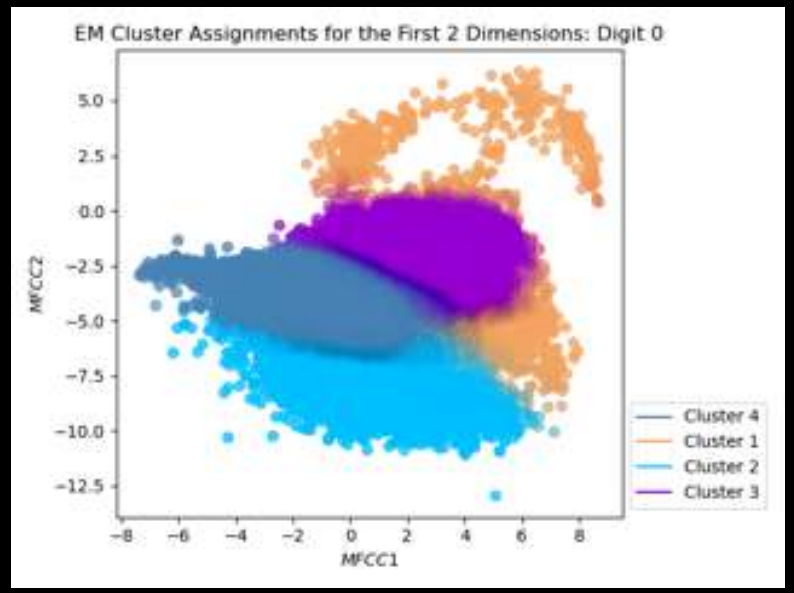


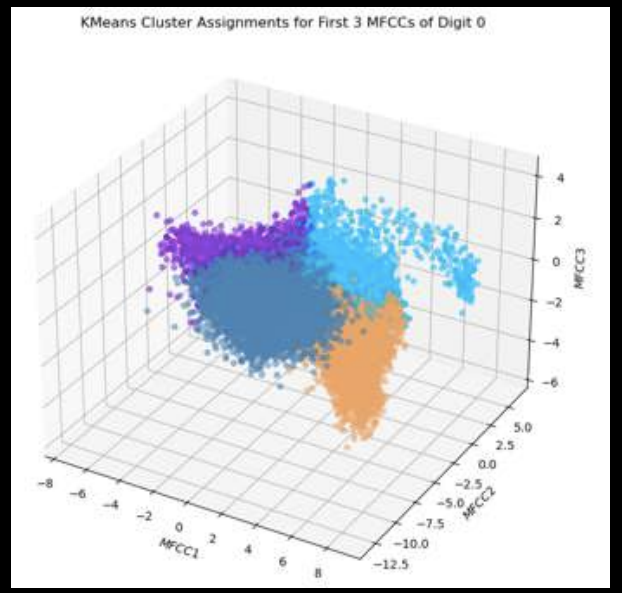

Scaled MFCCs for 10 Utterances of Digit 0
1
2
3
4
5
6
7
8
9
10
11
12
Minimal Variation
after MFCC2
Cepstral Coefficient Value
Analysis Windows Scaled Between 0-10
5
0
−5
−10
0.0
2.5
5.0
7.5
10.0


Comparison of Three Phoneme Analysis Techniques for Digit 0
4 Phonemes
Mel Frequency Cepstral Coefficient Value
Can Barely See Phonemes due to Temporal Shift
Massive difference in number of Analysis Frames in Utterances
Transition Period
Analysis Windows
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13


KMeans Cluster Assignments for First 2 MFCCs of Digit 0
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers


EM Cluster Assignments for the First 3 Dimensions: Digit 0
MFCC1
MFCC2
MFCC3

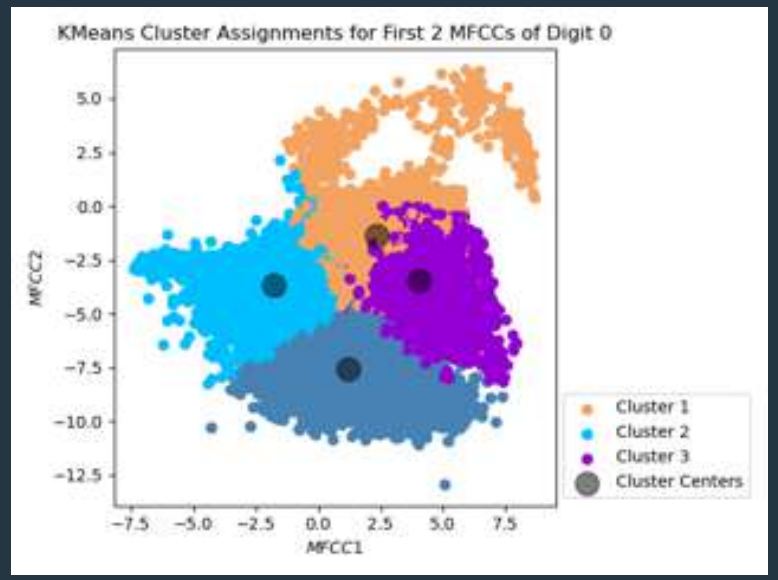

EM Cluster Assignments for the First 2 Dimensions: Digit 0
MFCC1
MFCC2
Cluster 4
Cluster 1
Cluster 2
Cluster 3

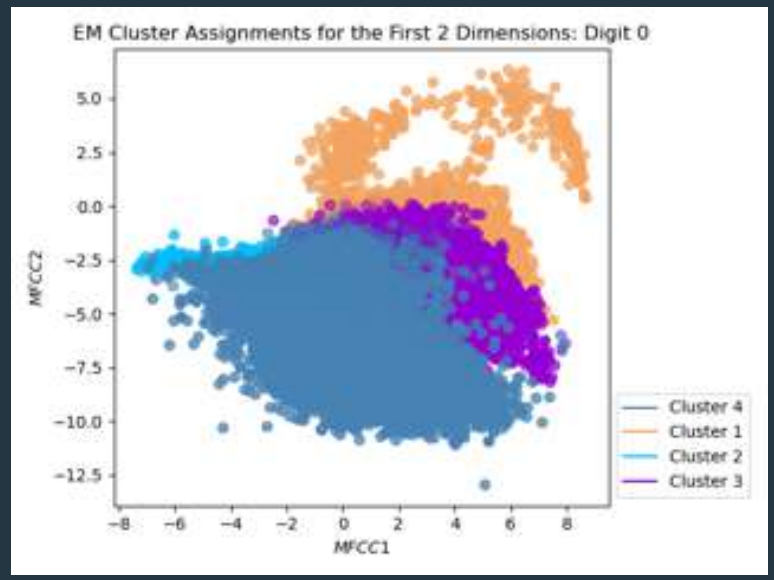

KMeans Cluster Assignments for First 3 MFCCs of Digit 0
MFCC1
MFCC2
MFCC3

Digit 0: 13 Dim Clusters Plotted in 2D

The previous page of plots depicted the cluster divisions of the first 2 or 3 MFCCs. These plots depict a two-dimensional slice of all the MFCCs' cluster divisions. It is possible to tell that in these plots the delineations between the clusters are less clear. This corresponds to the influence that extra dimensions have on the clusters due to their importance. In a more complex model, each of the MFCCs would be scaled according to their importance within the clustering. In this plot it can be observed that Cluster 2 and 3 in the Kmeans have quite strong delineations between them even with the extra dimensions. Alternatively, the EM distributions become a lot more skewed towards cluster 4. The pi value for this cluster would be considerable larger since there are considerably more points within that cluster. This larger cluster corresponds to the larger sized phoneme within the word.

The lack of delineation between the clusters foreshadows that using fewer dimension to determine the clusters may cause a lower accuracy of model.

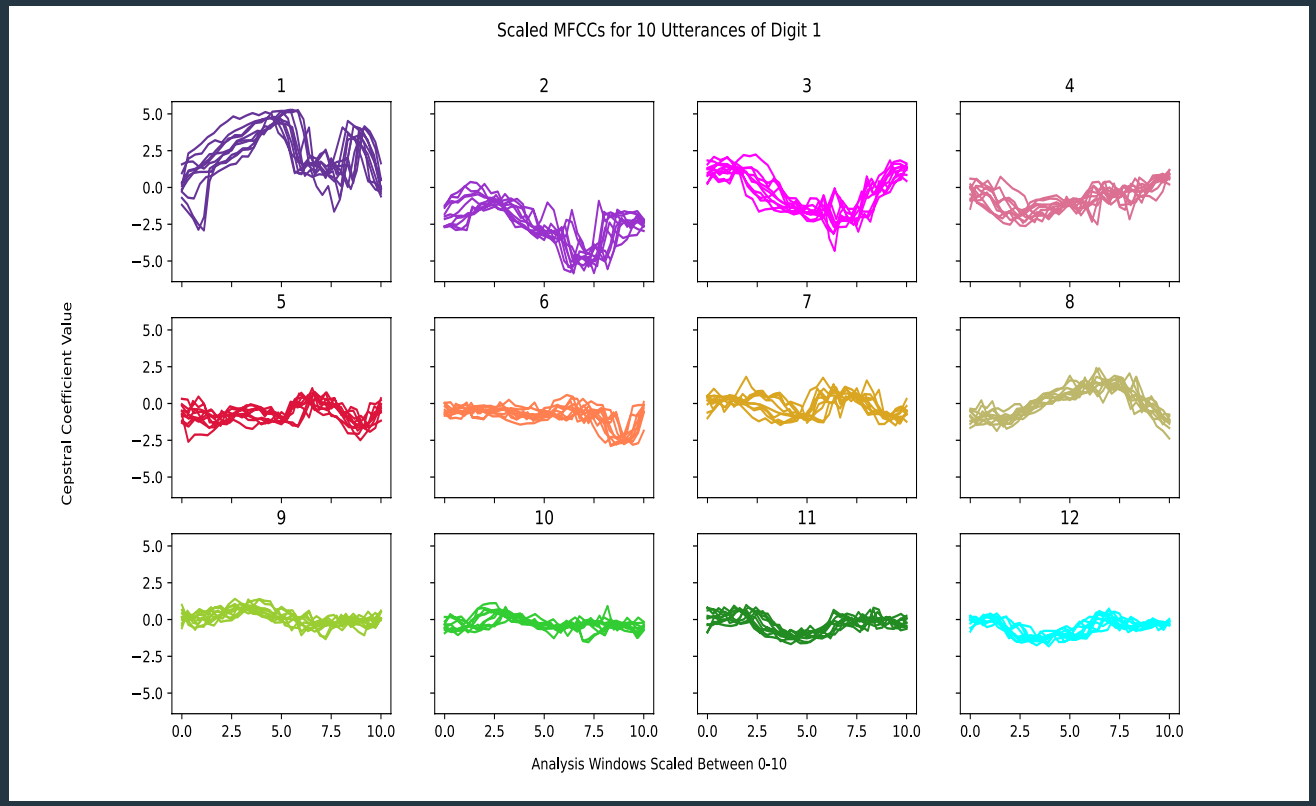


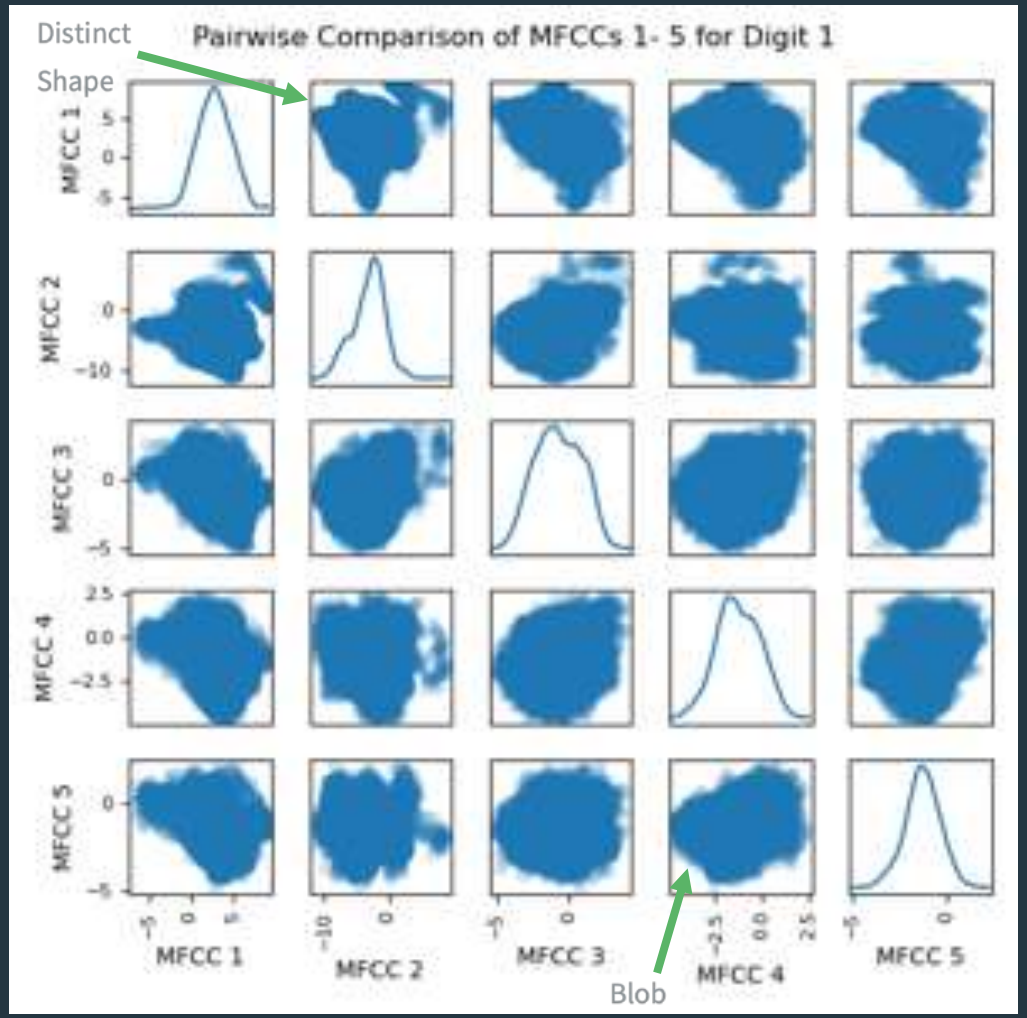


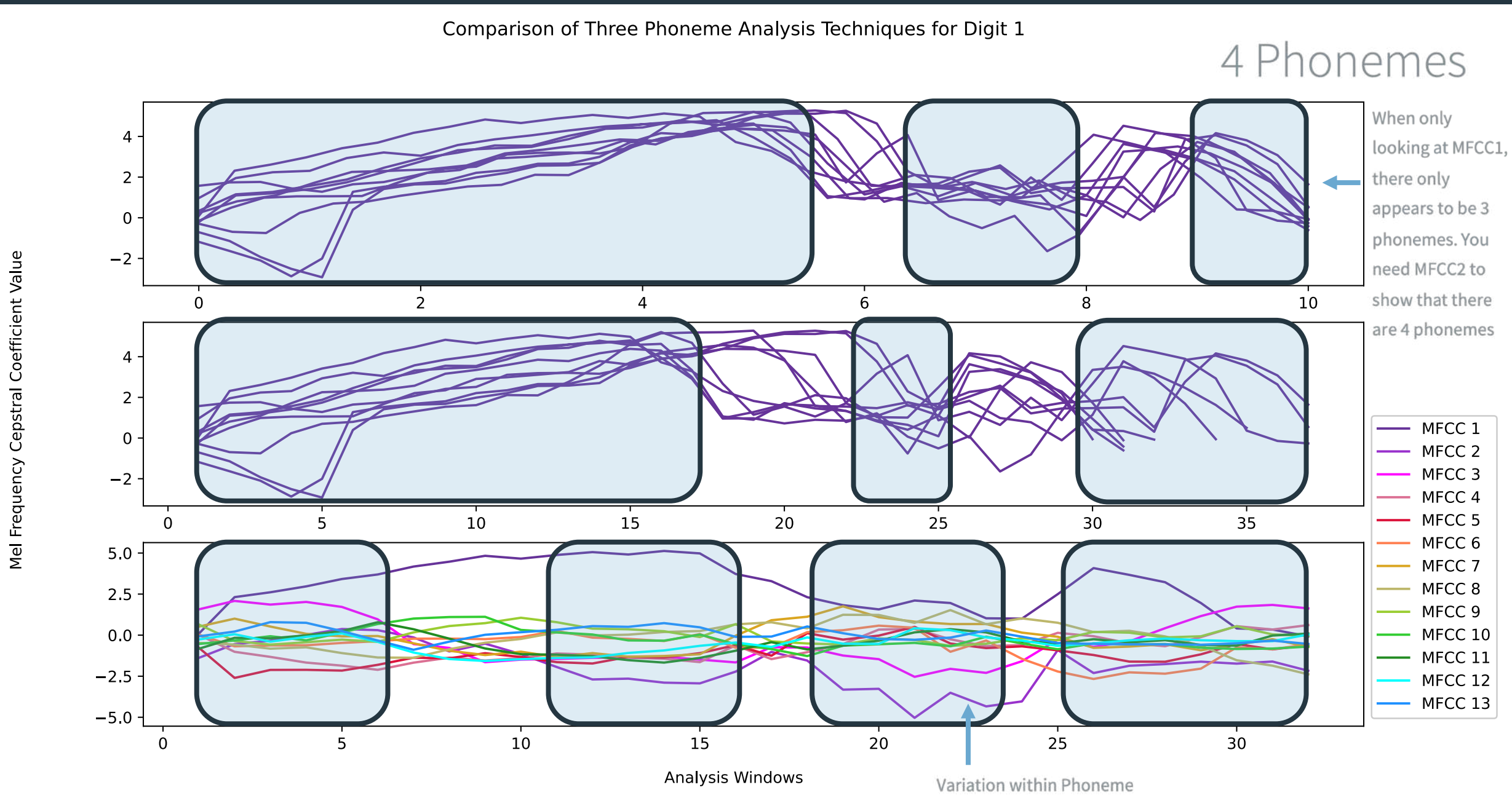
Digit 1 - wahad

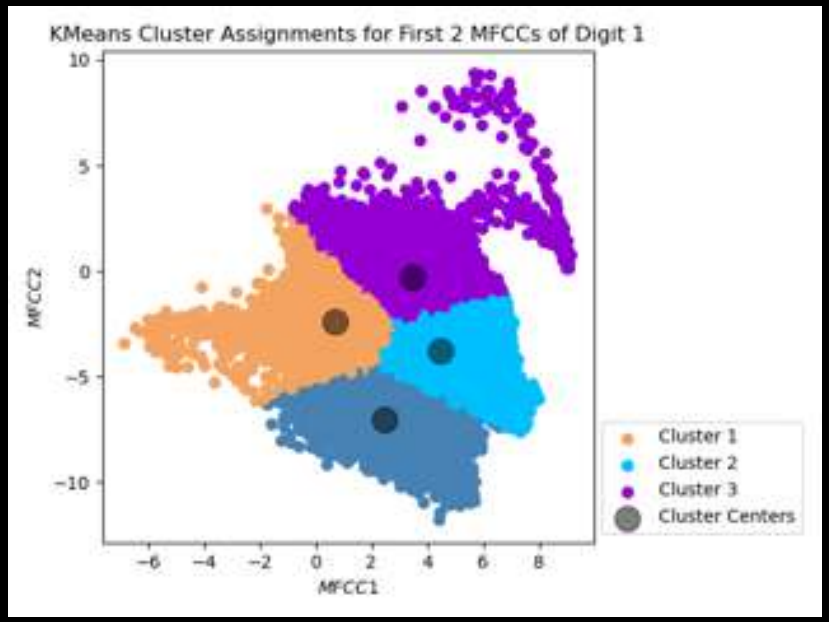
وحد waːˈħid

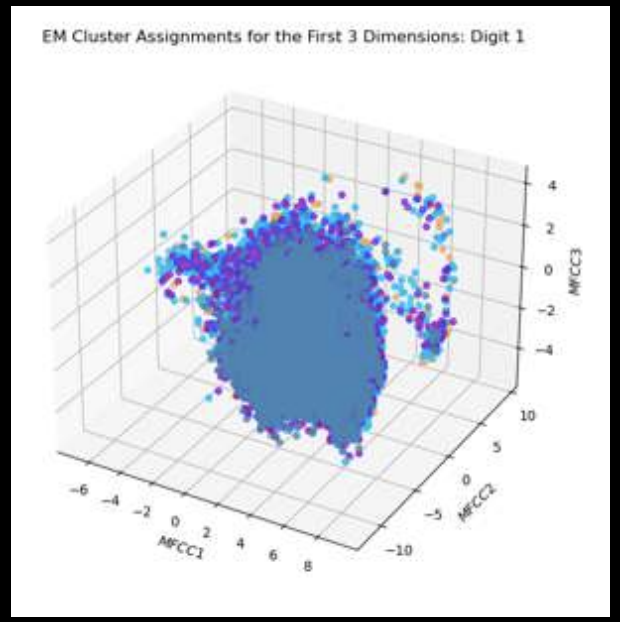
Visualisations of the Importance of Various MFCCs for Digit 1

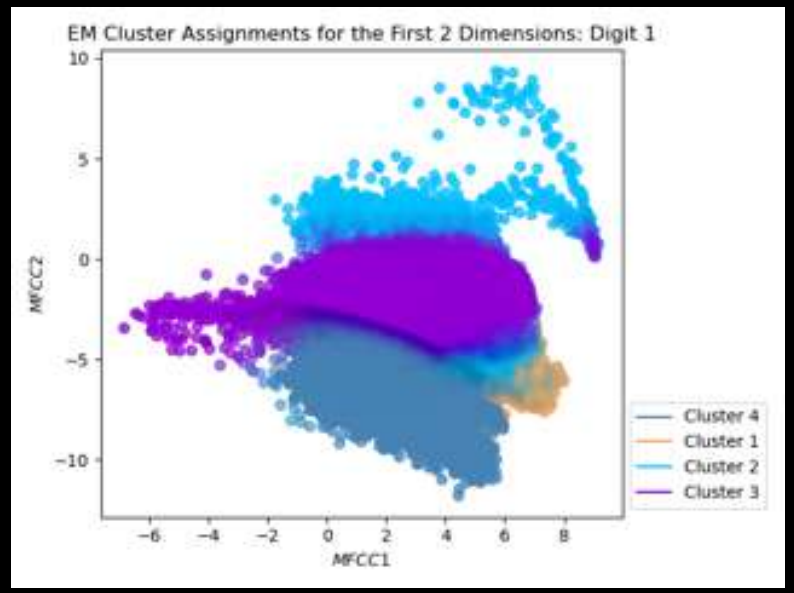


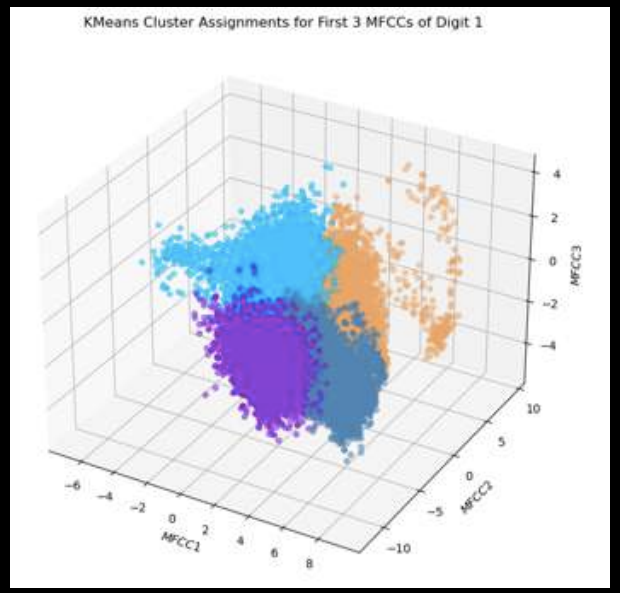



Comparison of Three Phoneme Analysis Techniques for Digit 1
4 Phonemes
When only looking at MFCC1, there only appears to be 3 phonemes. You need MFCC2 to show that there are 4 phonemes
Mel Frequency Cepstral Coefficient Value
Analysis Windows
Variation within Phoneme
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13

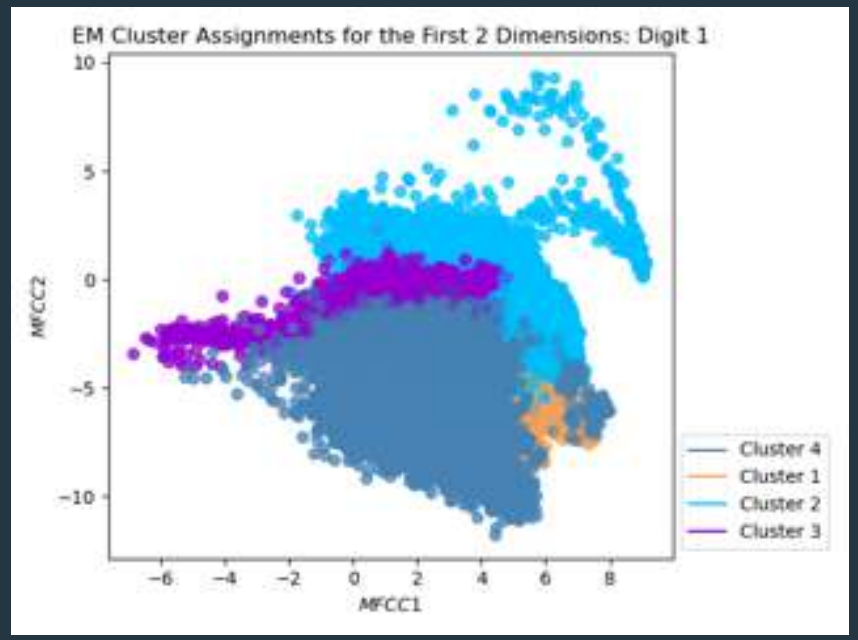

KMeans Cluster Assignments for First 2 MFCCs of Digit 1
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

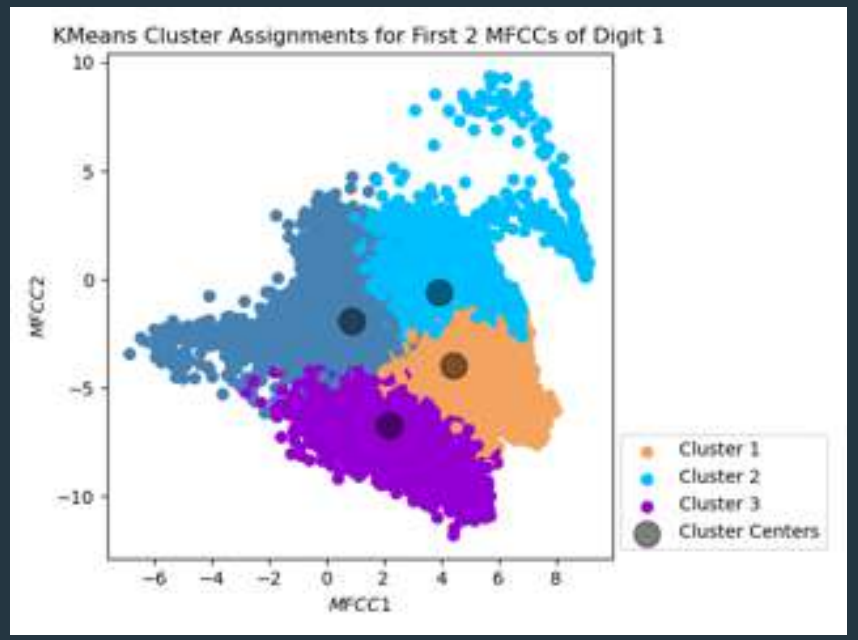

EM Cluster Assignments for the First 3 Dimensions: Digit 1
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 1
MFCC1
MFCC2
Cluster 4
Cluster 1
Cluster 2
Cluster 3


KMeans Cluster Assignments for First 3 MFCCs of Digit 1
MFCC1
MFCC2
MFCC3

Digit 1: 13 Dim Clusters Plotted in 2D

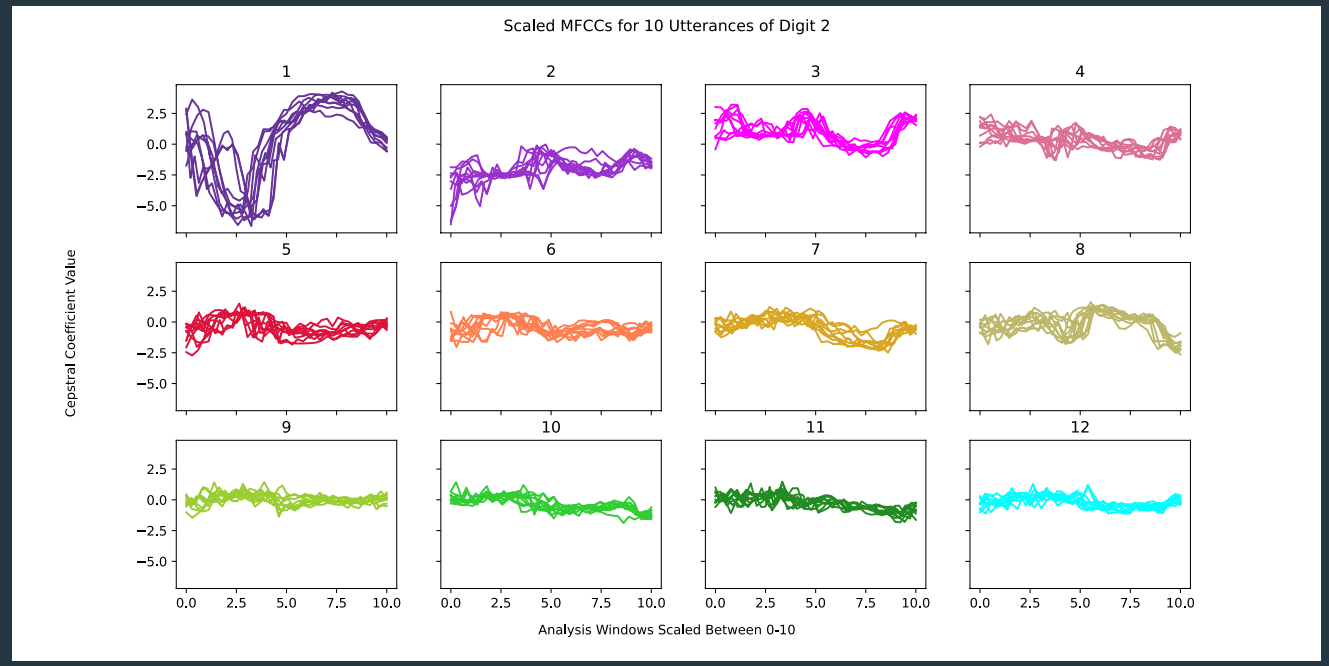


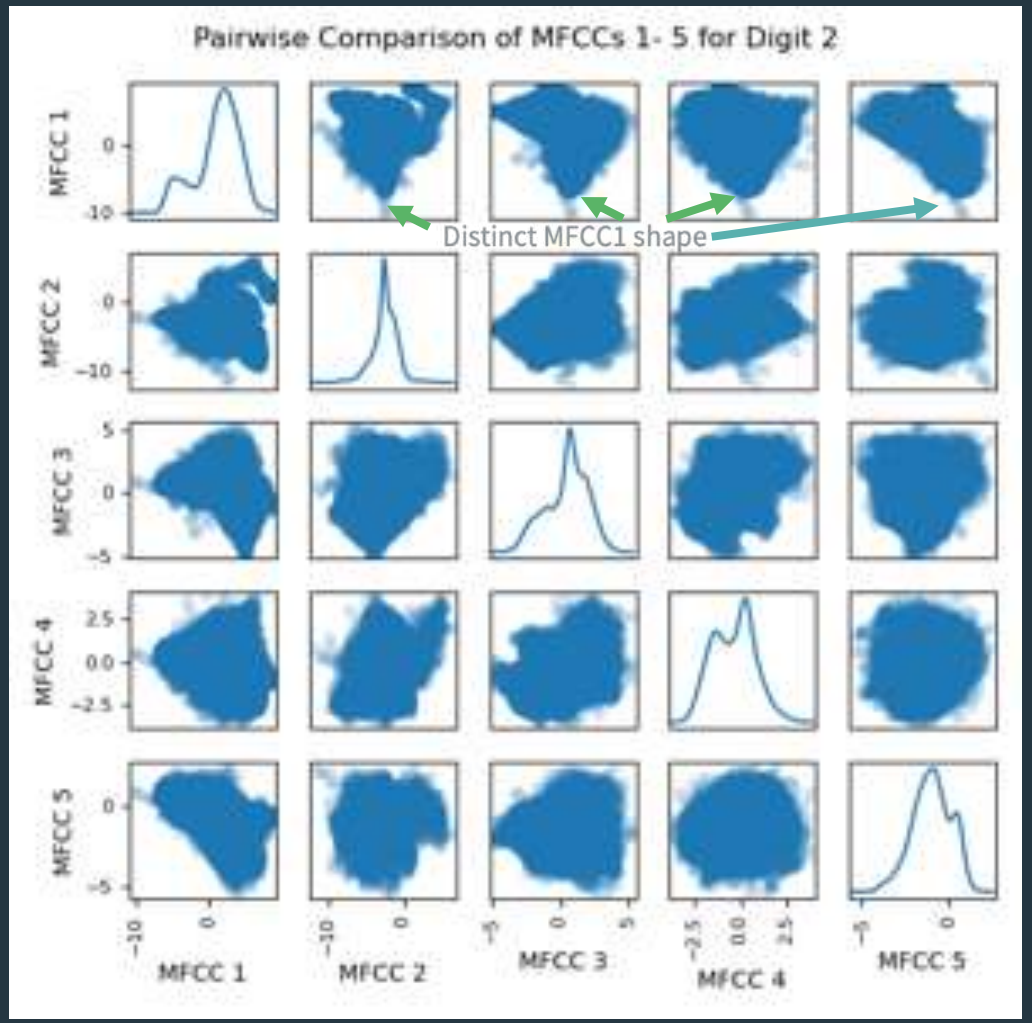


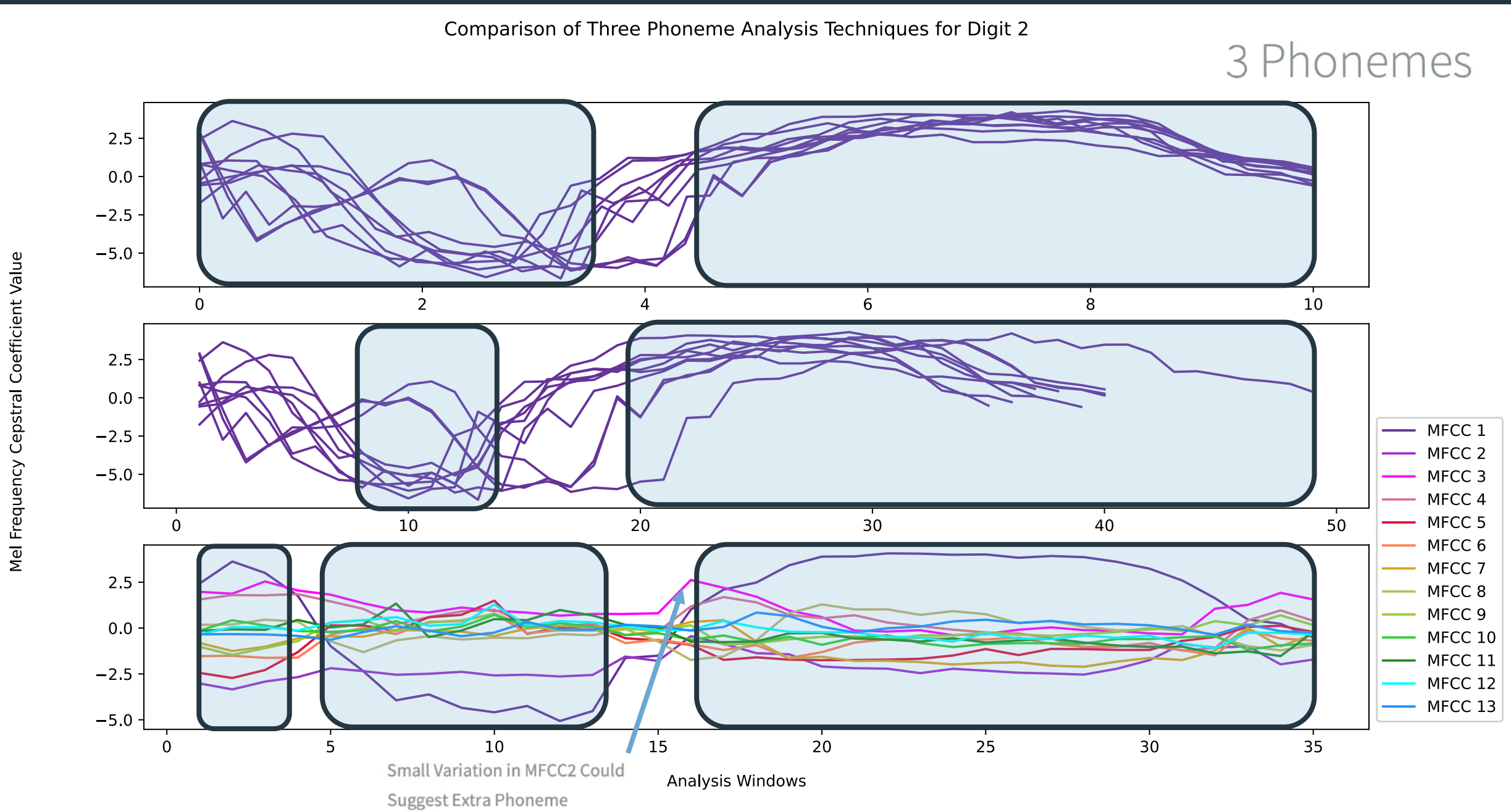
Digit 2 – ithnayn

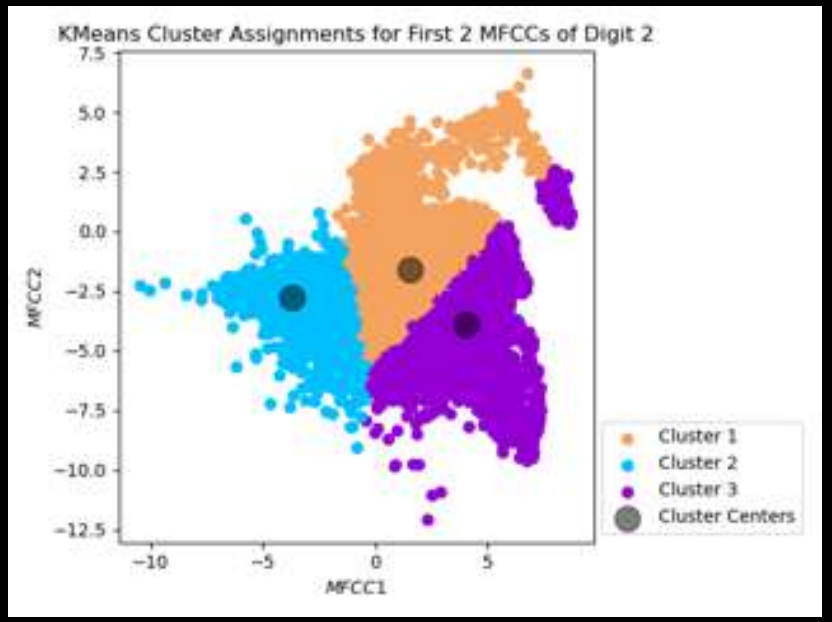
لاثنين

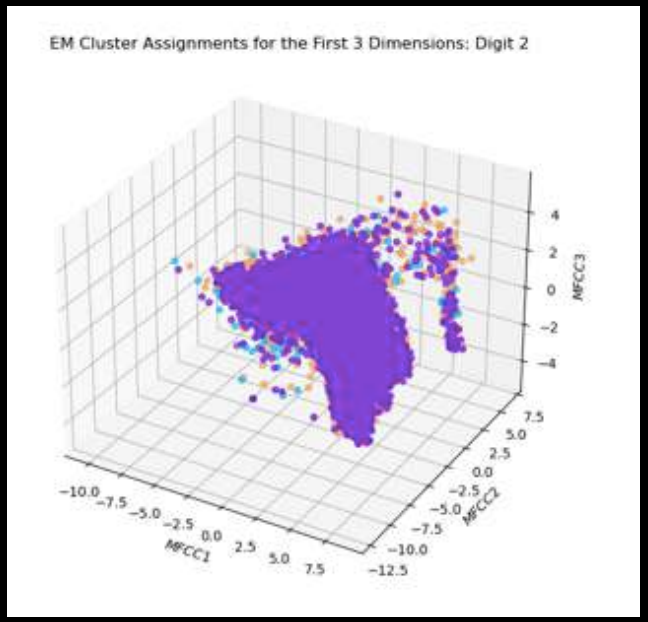
Visualisations of the Importance of Various MFCCs for Digit 2

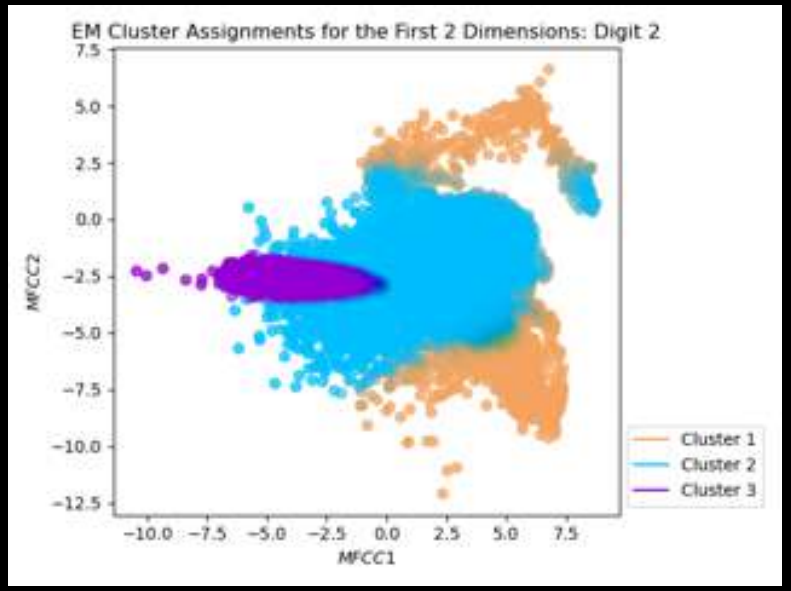


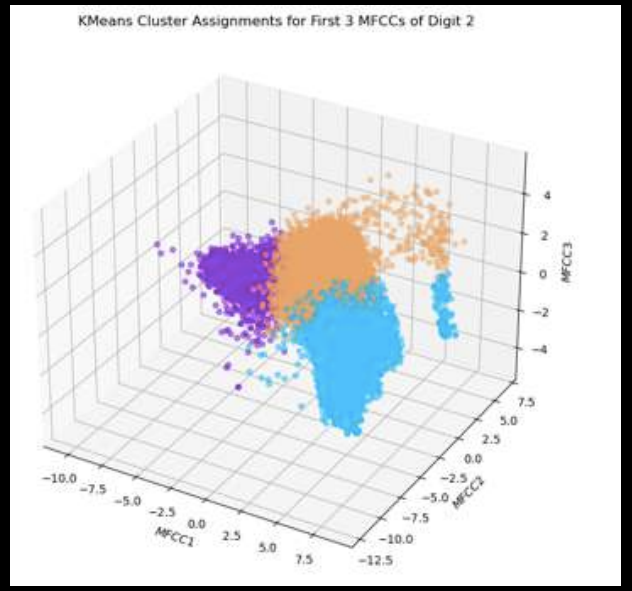



Comparison of Three Phoneme Analysis Techniques for Digit 2
3 Phonemes
Mel Frequency Cepstral Coefficient Value
Analysis Windows
Small Variation in MFCC2 Could
Suggest Extra Phoneme
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13

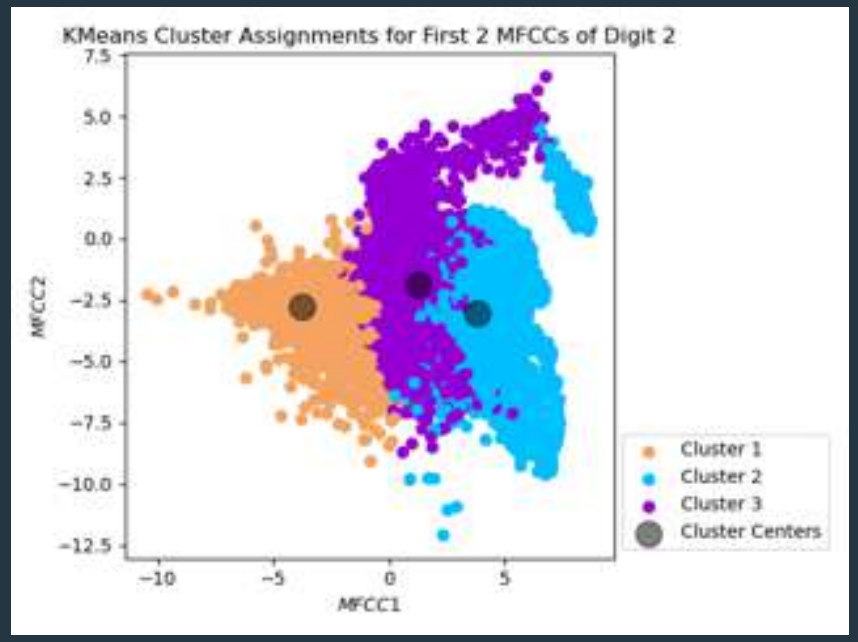

KMeans Cluster Assignments for First 2 MFCCs of Digit 2
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

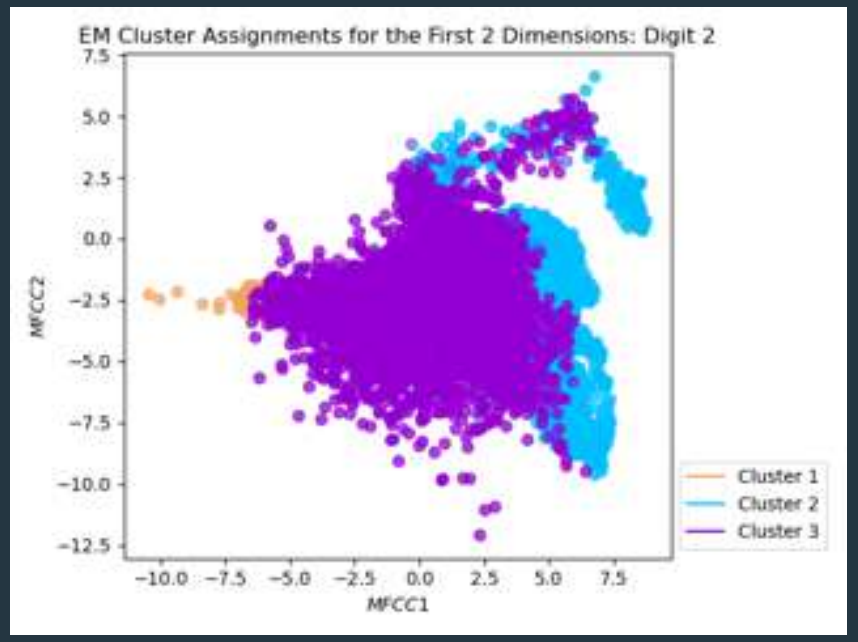

EM Cluster Assignments for the First 3 Dimensions: Digit 2
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 2
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3


KMeans Cluster Assignments for First 3 MFCCs of Digit 2
MFCC1
MFCC2
MFCC3

Digit 2: 13 Dim Clusters Plotted in 2D

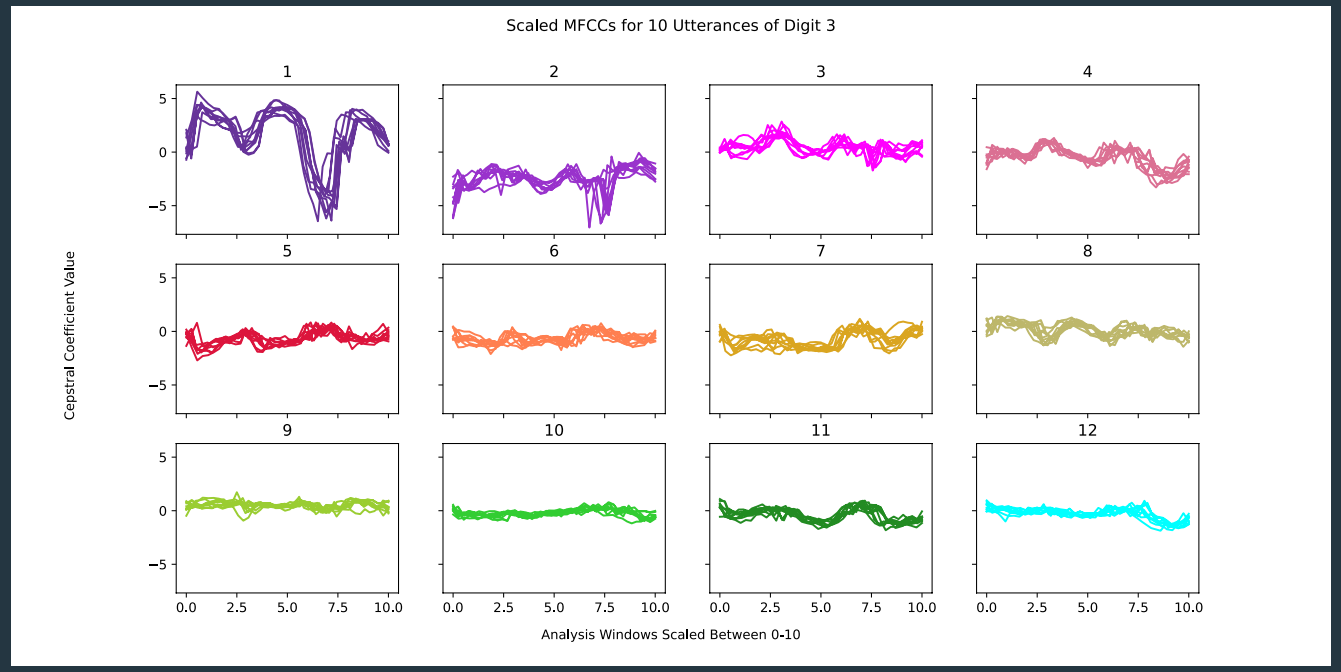


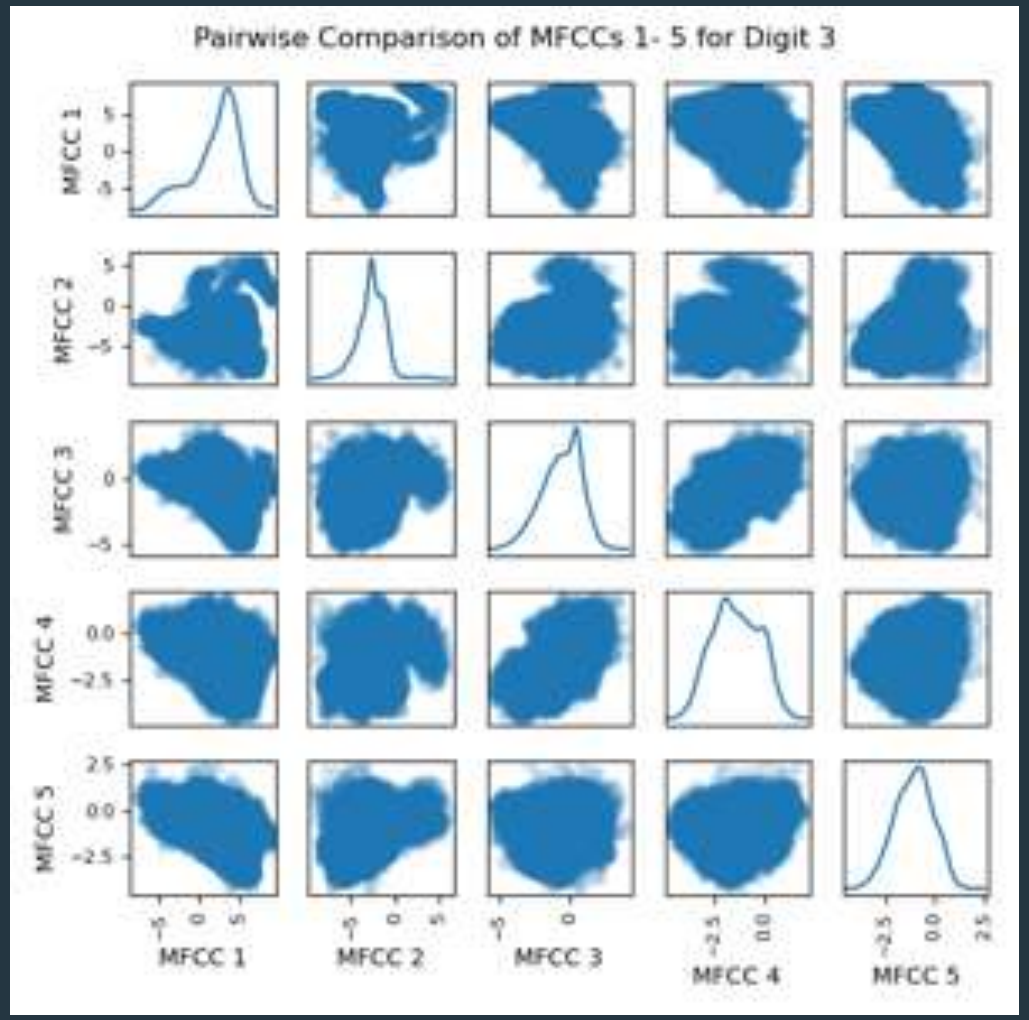


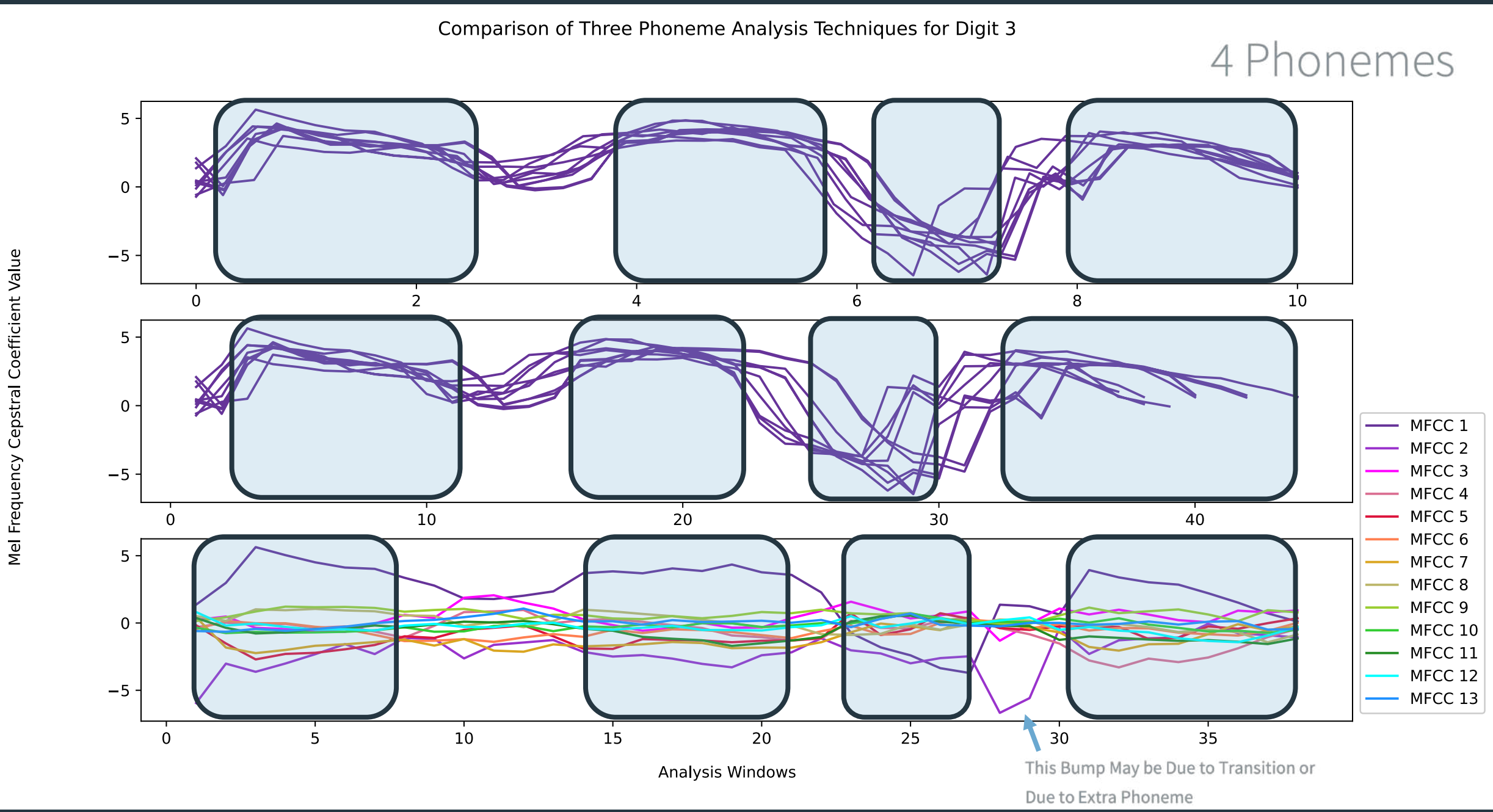
Digit 3 – thalatha

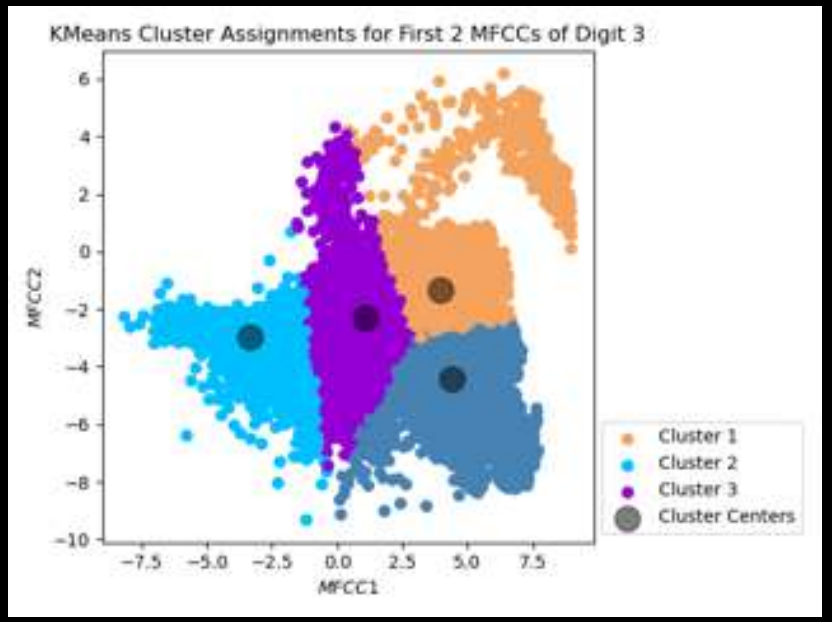
ثَلاَثَة

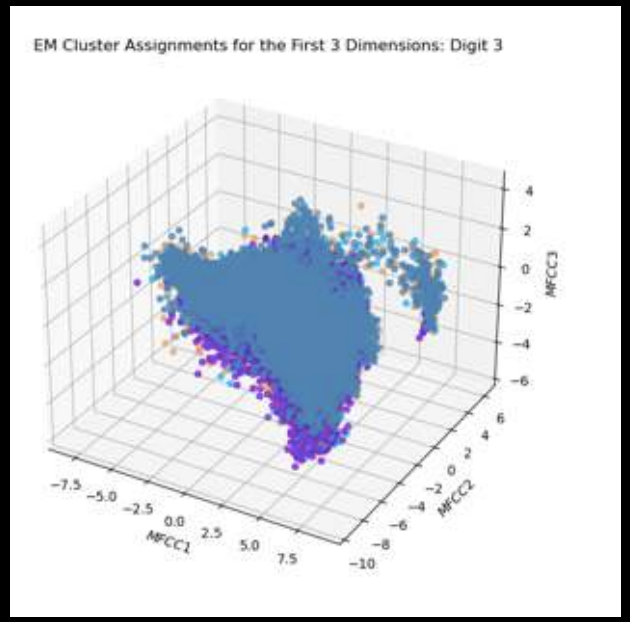
Visualisations of the Importance of Various MFCCs for Digit 3

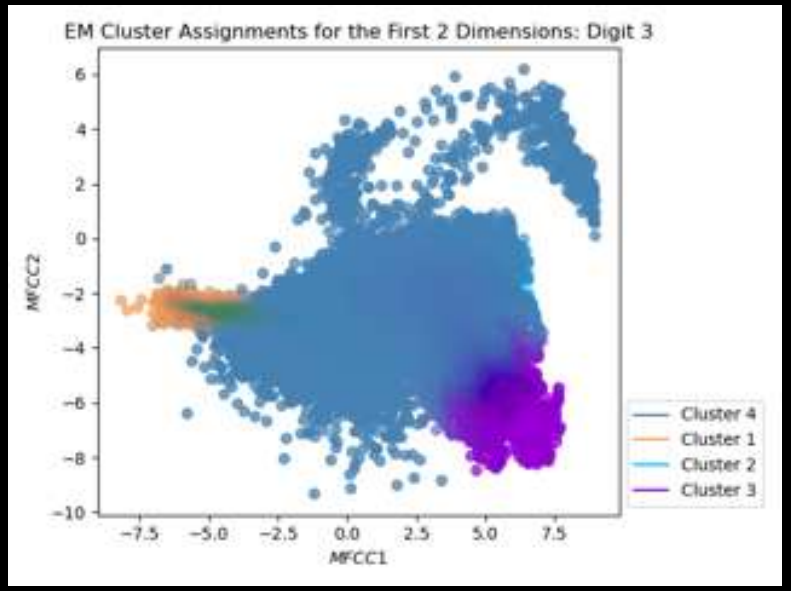


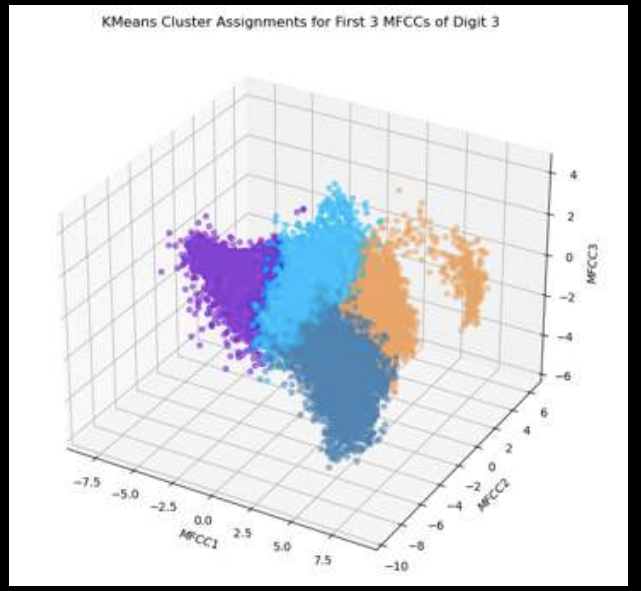



Comparison of Three Phoneme Analysis Techniques for Digit 3
4 Phonemes
Mel Frequency Cepstral Coefficient Value
Analysis Windows
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13
This Bump May be Due to Transition or
Due to Extra Phoneme

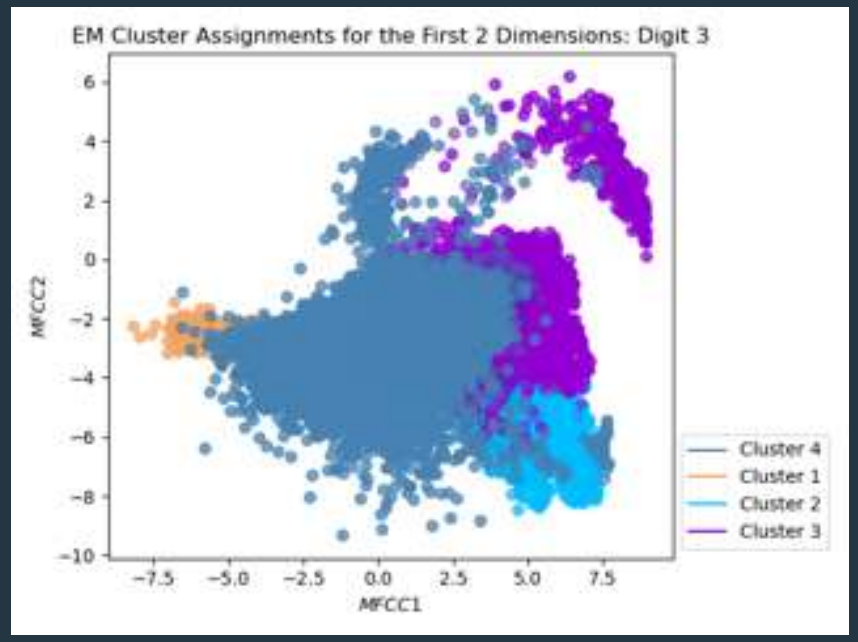

KMeans Cluster Assignments for First 2 MFCCs of Digit 3
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

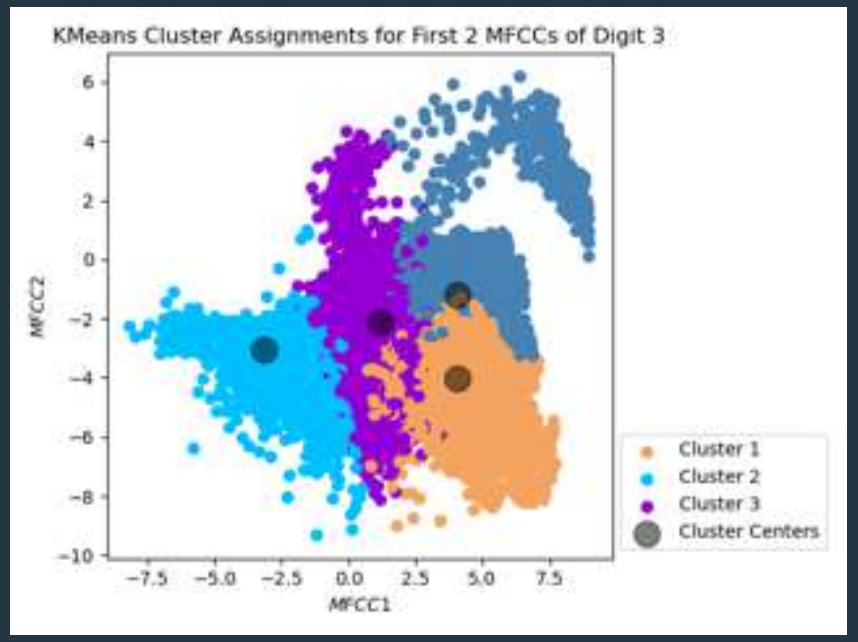

EM Cluster Assignments for the First 3 Dimensions: Digit 3
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 3
MFCC1
MFCC2
Cluster 4
Cluster 1
Cluster 2
Cluster 3


KMeans Cluster Assignments for First 3 MFCCs of Digit 3
MFCC1
MFCC2
MFCC3

Digit 3: 13 Dim Clusters Plotted in 2D

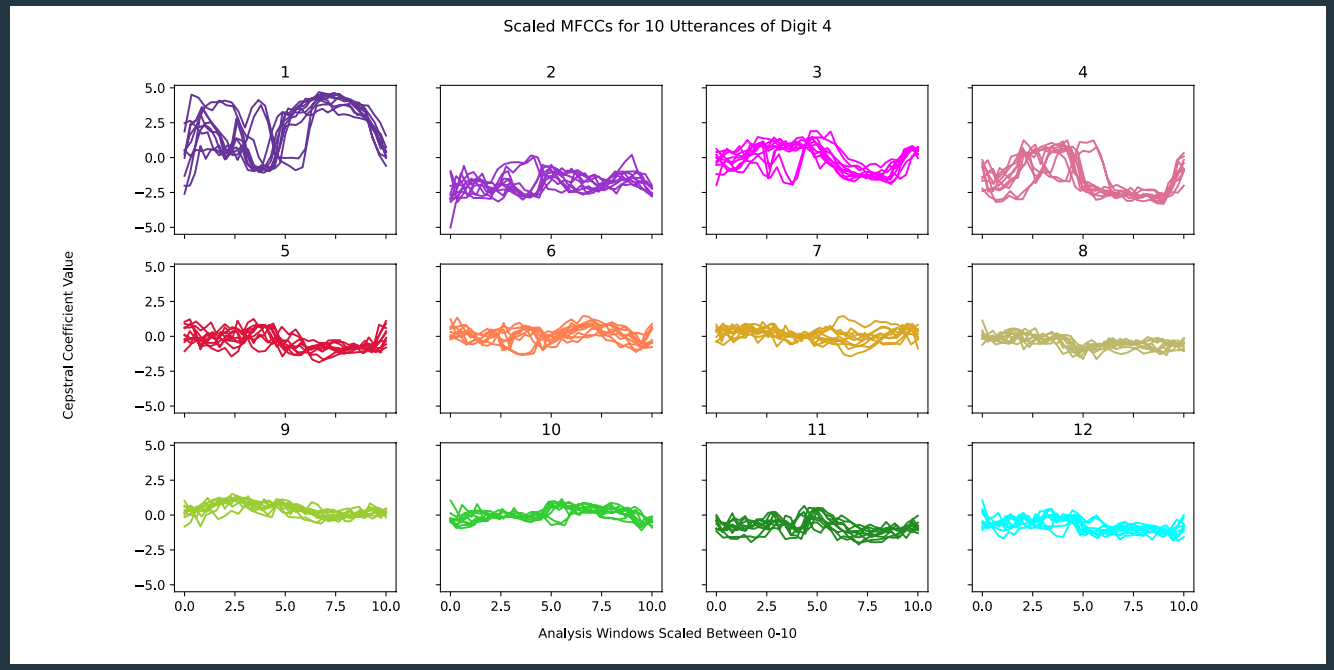


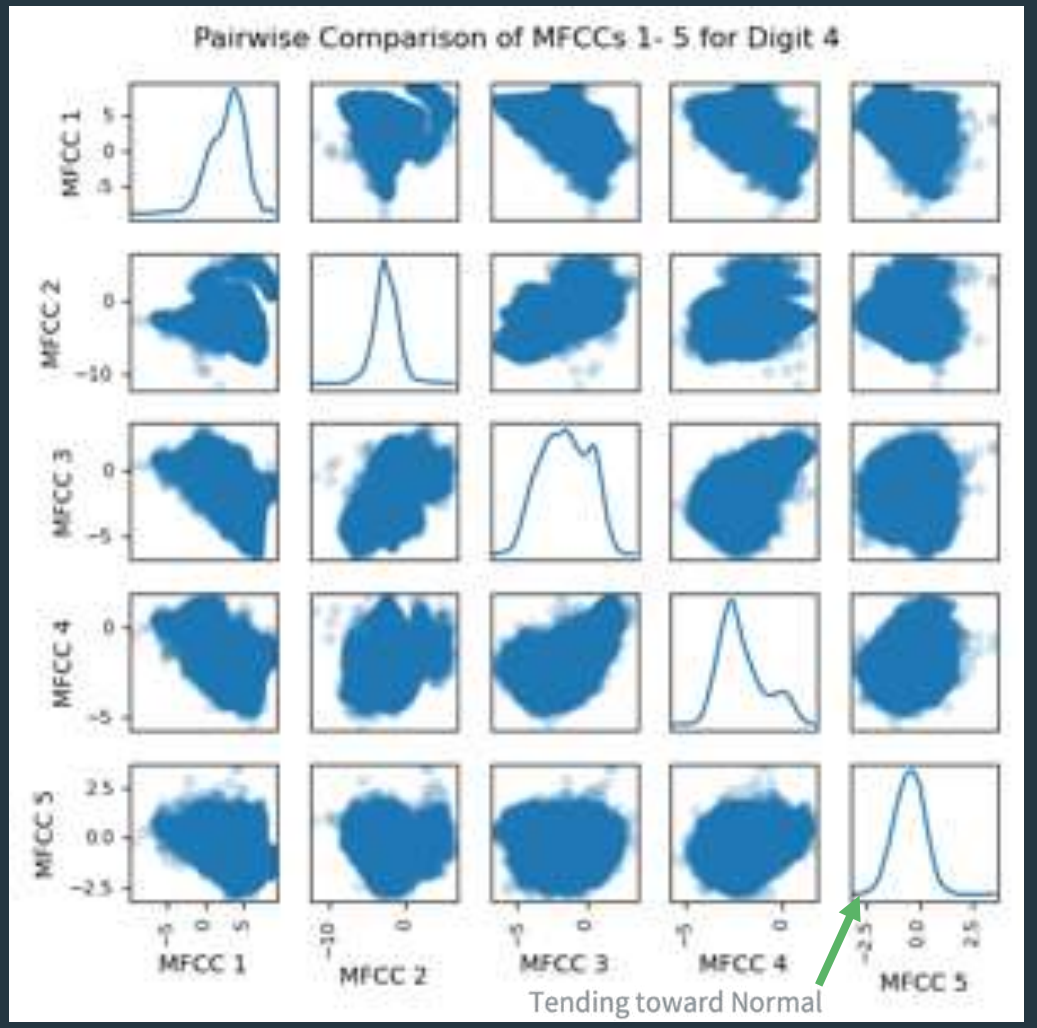


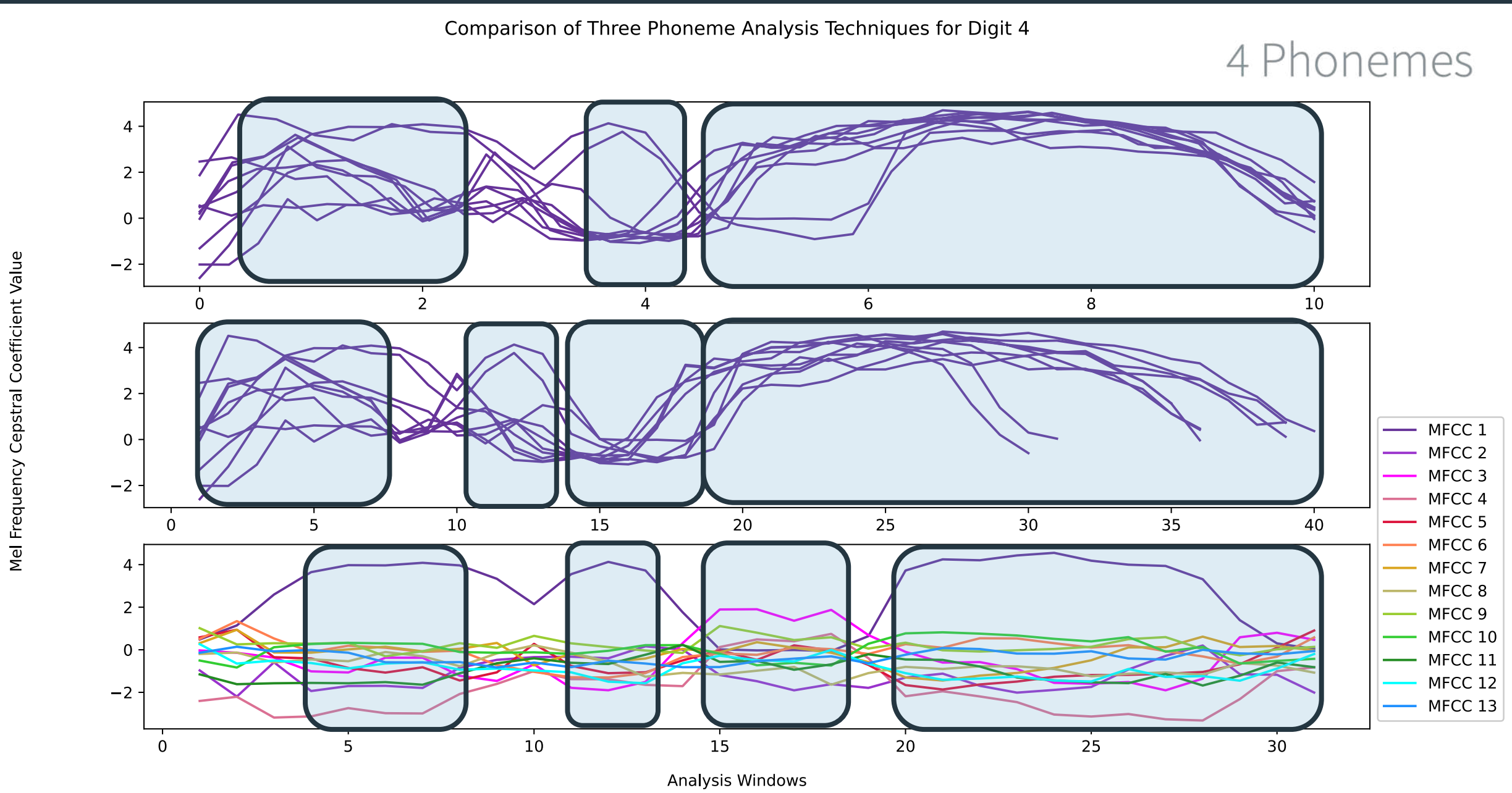
Digit 4 – araba’a

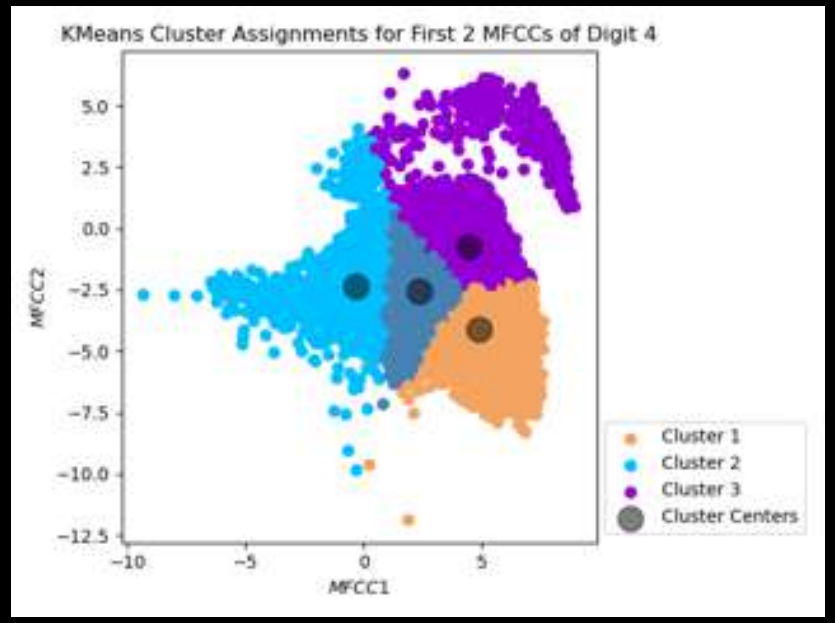
أرابعة

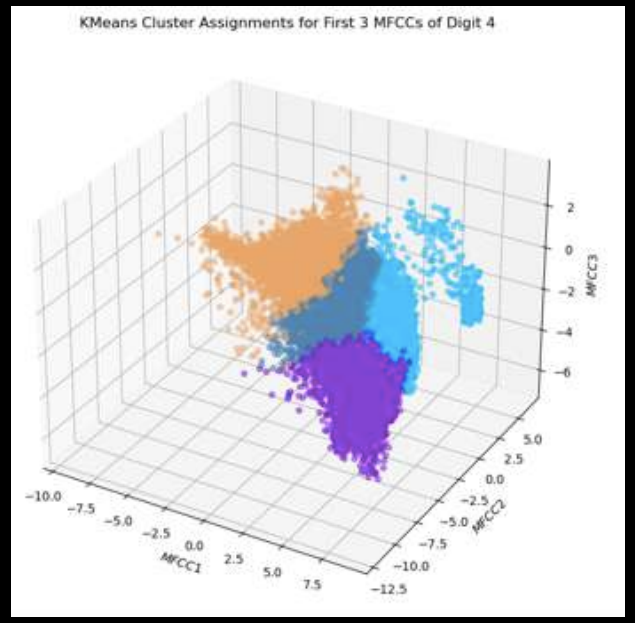
Visualisations of the Importance of Various MFCCs for Digit 4

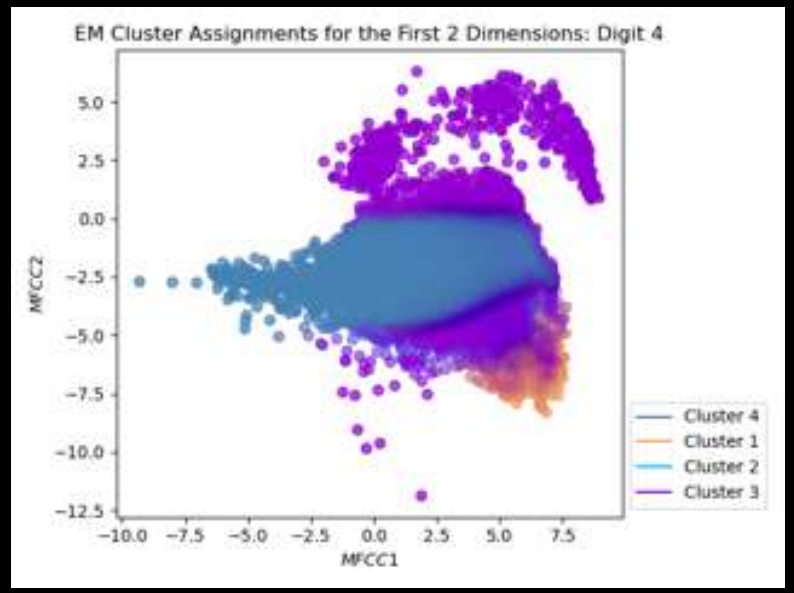


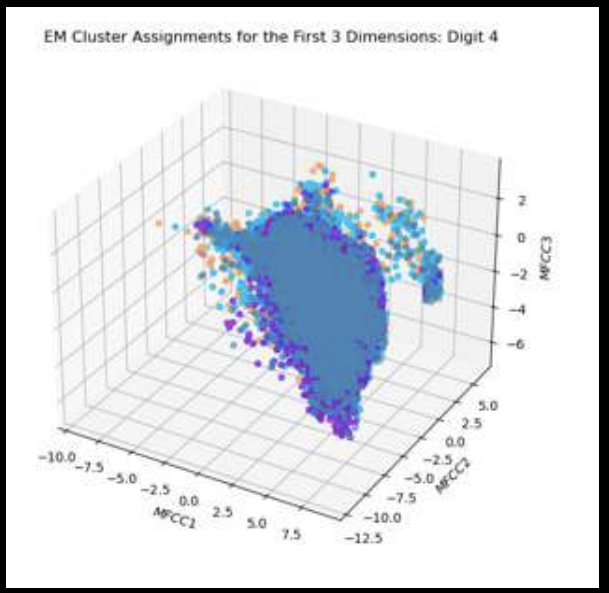



Comparison of Three Phoneme Analysis Techniques for Digit 4
4 Phonemes
Mel Frequency Cepstral Coefficient Value
Analysis Windows
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13

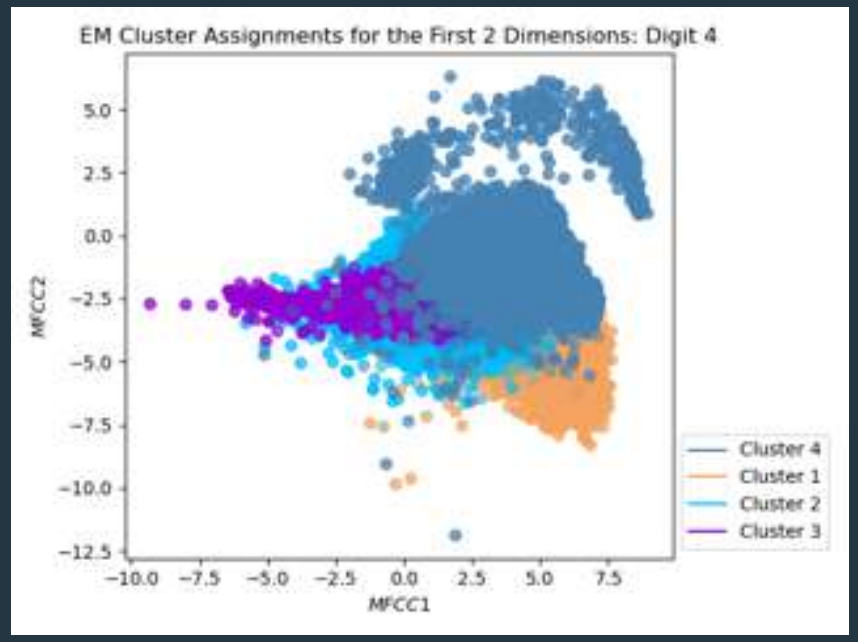

KMeans Cluster Assignments for First 2 MFCCs of Digit 4
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

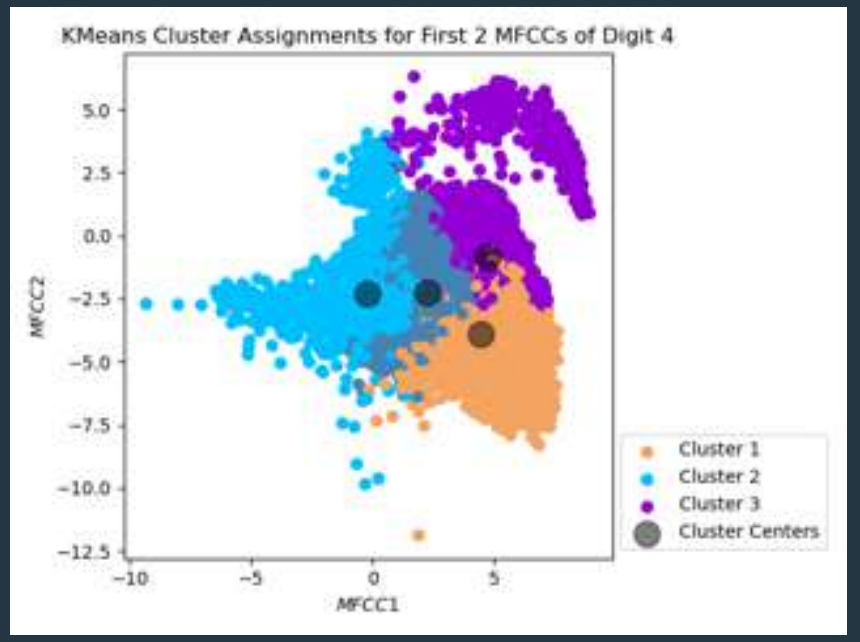

KMeans Cluster Assignments for First 3 MFCCs of Digit 4
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 4
MFCC1
MFCC2
Cluster 4
Cluster 1
Cluster 2
Cluster 3


EM Cluster Assignments for the First 3 Dimensions: Digit 4
MFCC1
MFCC2
MFCC3

Digit 4: 13 Dim Clusters Plotted in 2D

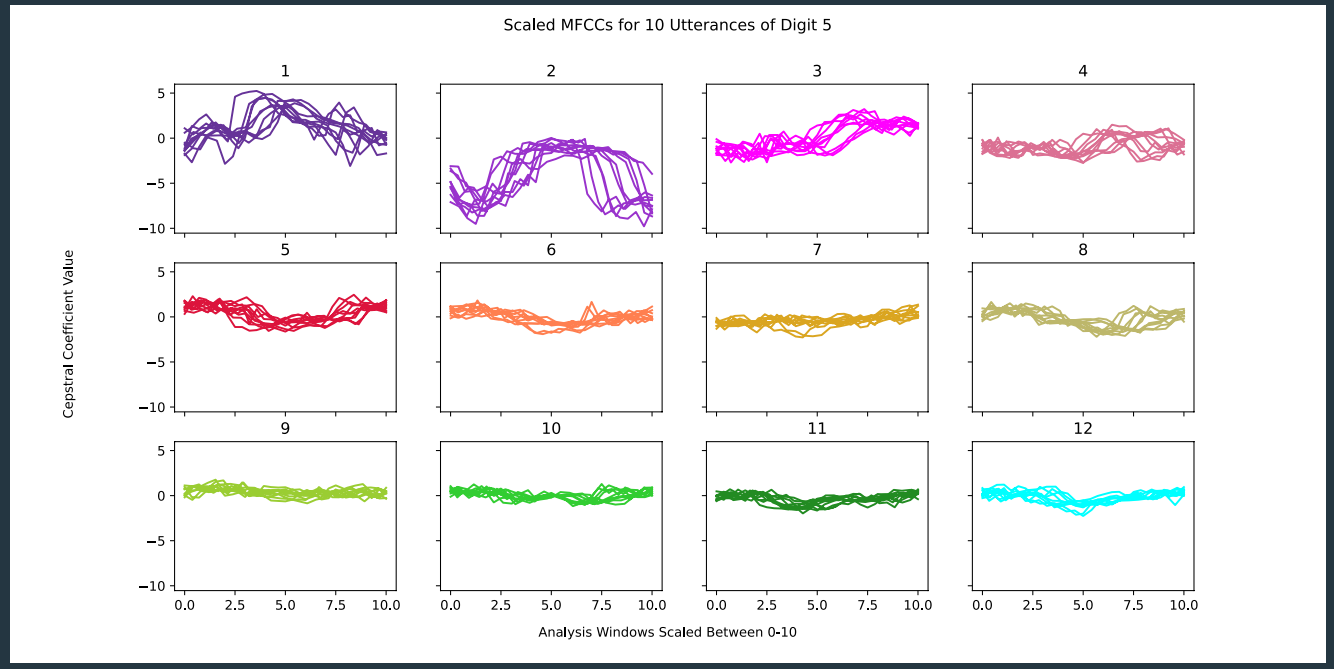


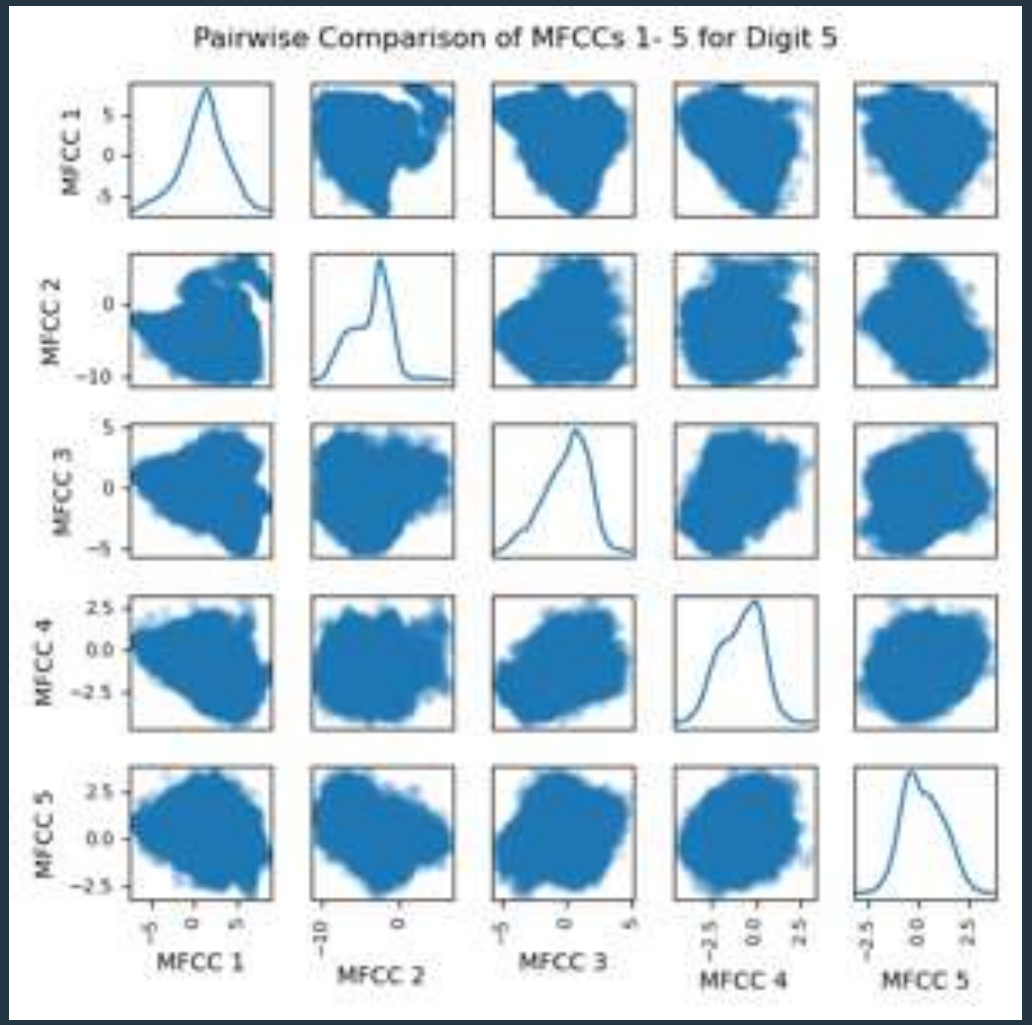


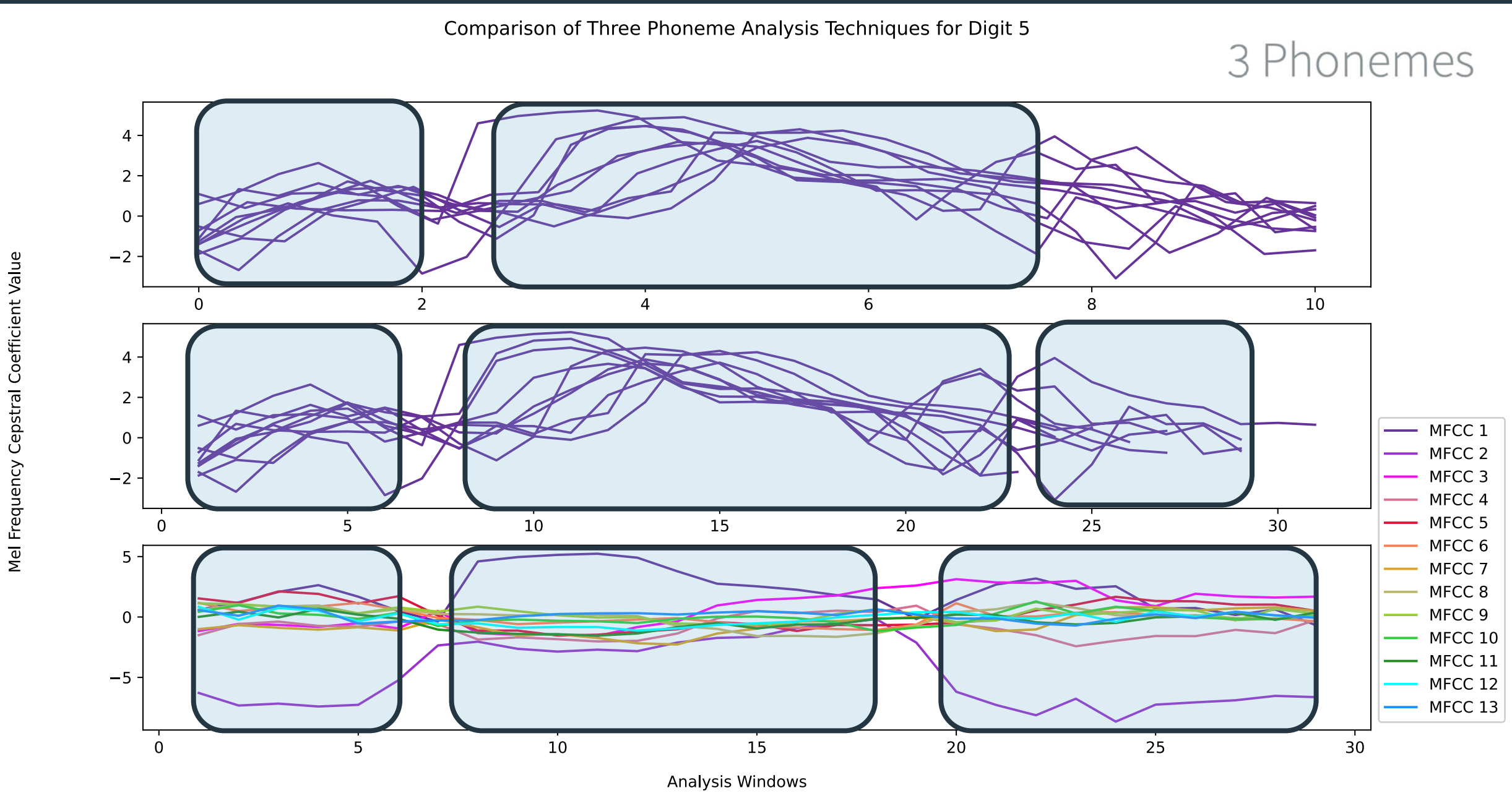
Digit 5 – khamsa

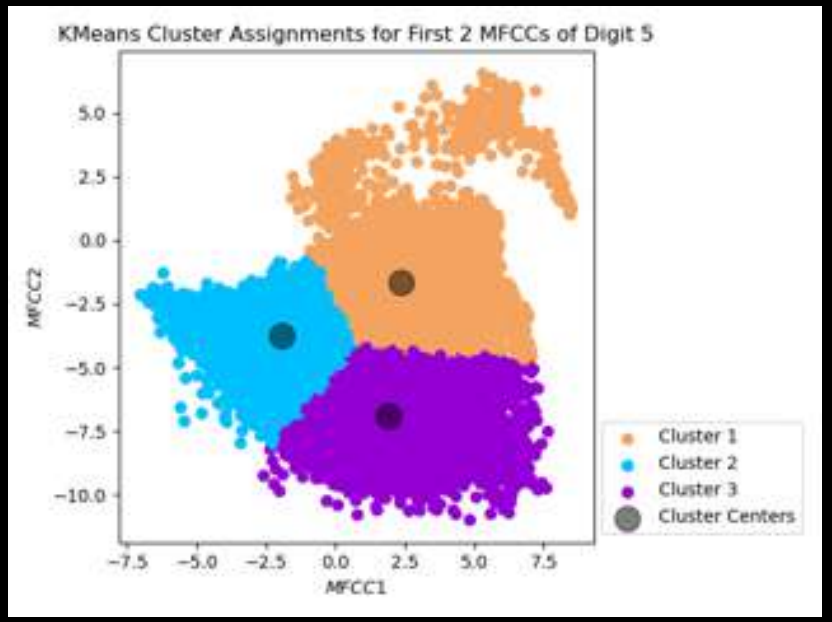
خَمْسَة

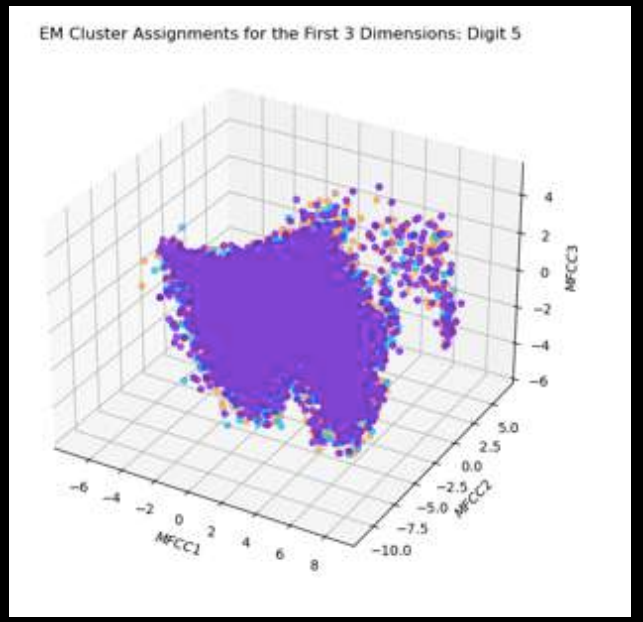
Visualisations of the Importance of Various MFCCs for Digit 5

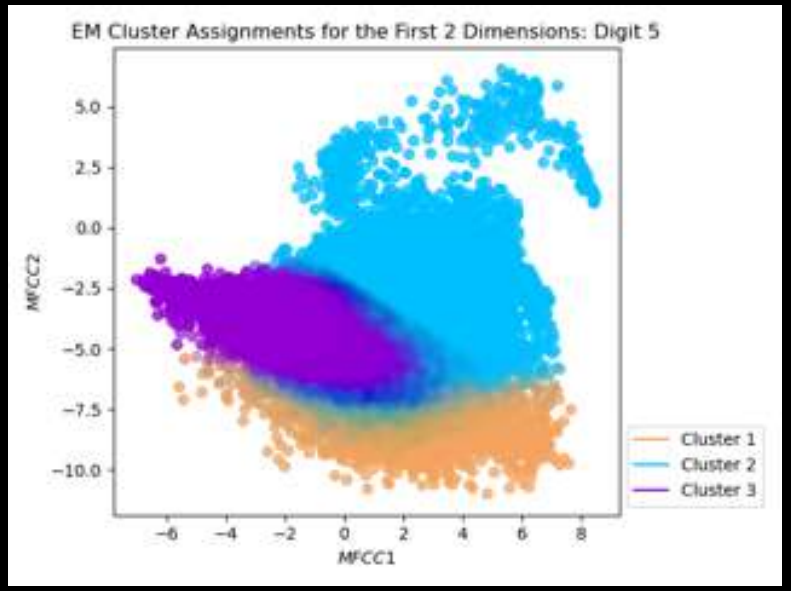


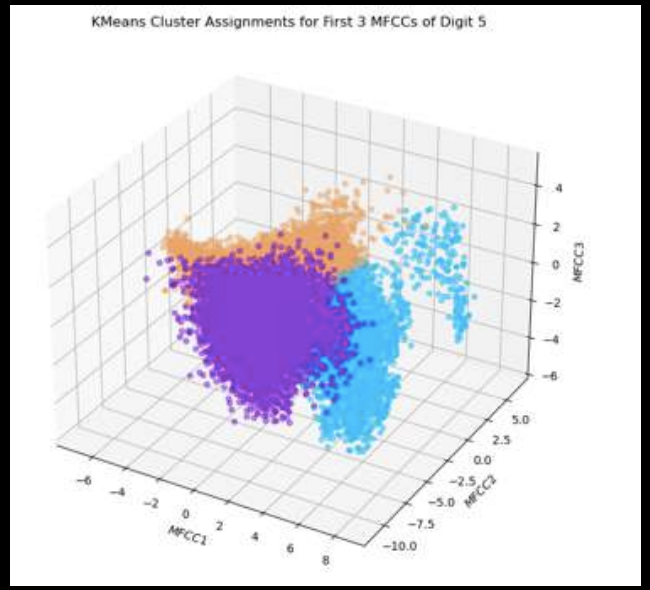



Comparison of Three Phoneme Analysis Techniques for Digit 5
3 Phonemes
Mel Frequency Cepstral Coefficient Value
Analysis Windows
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13

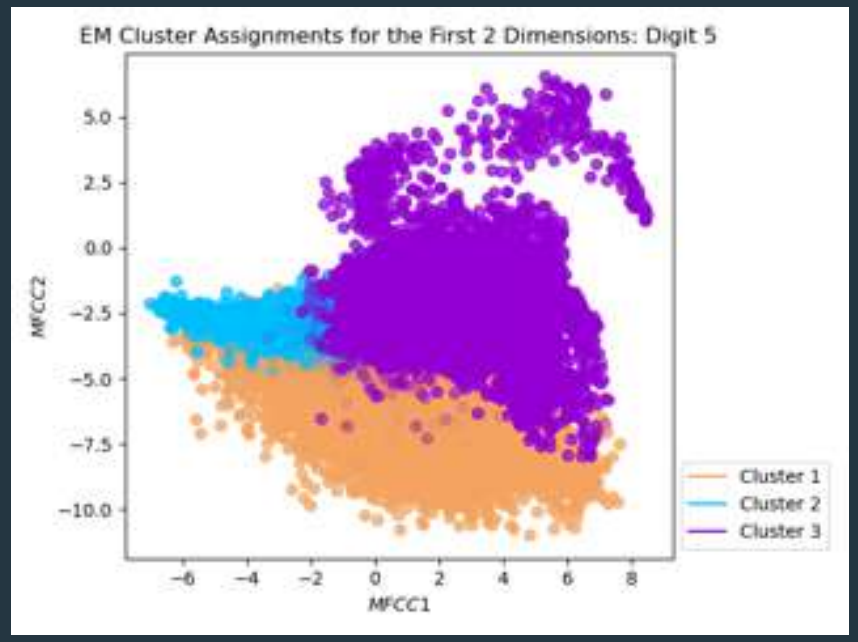

KMeans Cluster Assignments for First 2 MFCCs of Digit 5
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

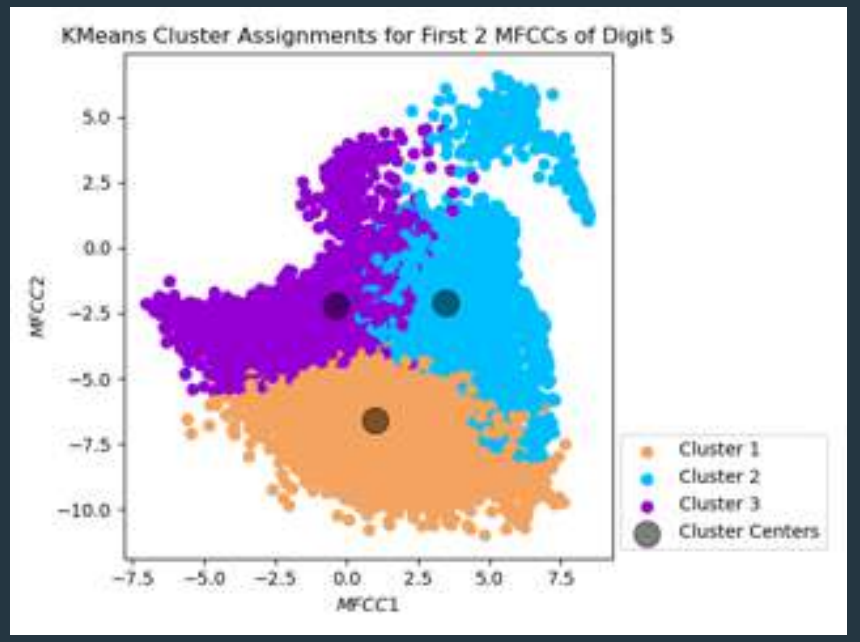

EM Cluster Assignments for the First 3 Dimensions: Digit 5
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 5
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3


KMeans Cluster Assignments for First 3 MFCCs of Digit 5
MFCC1
MFCC2
MFCC3

Digit 5: 13 Dim Clusters Plotted in 2D

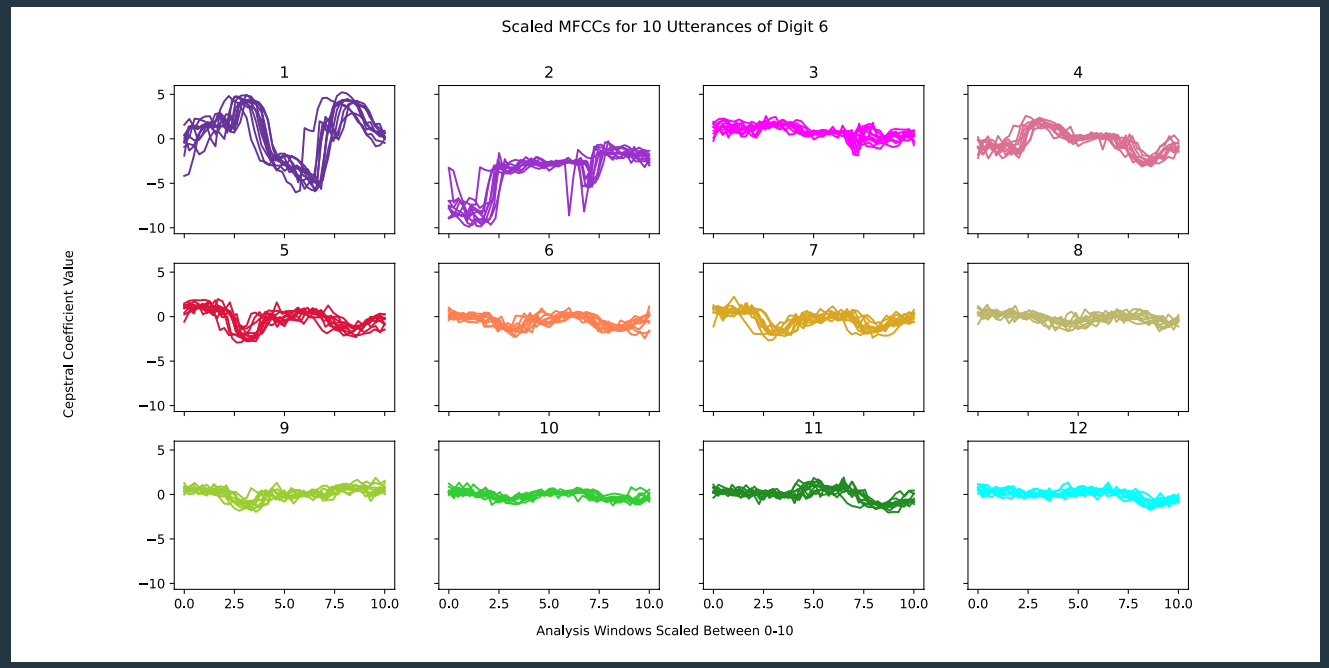


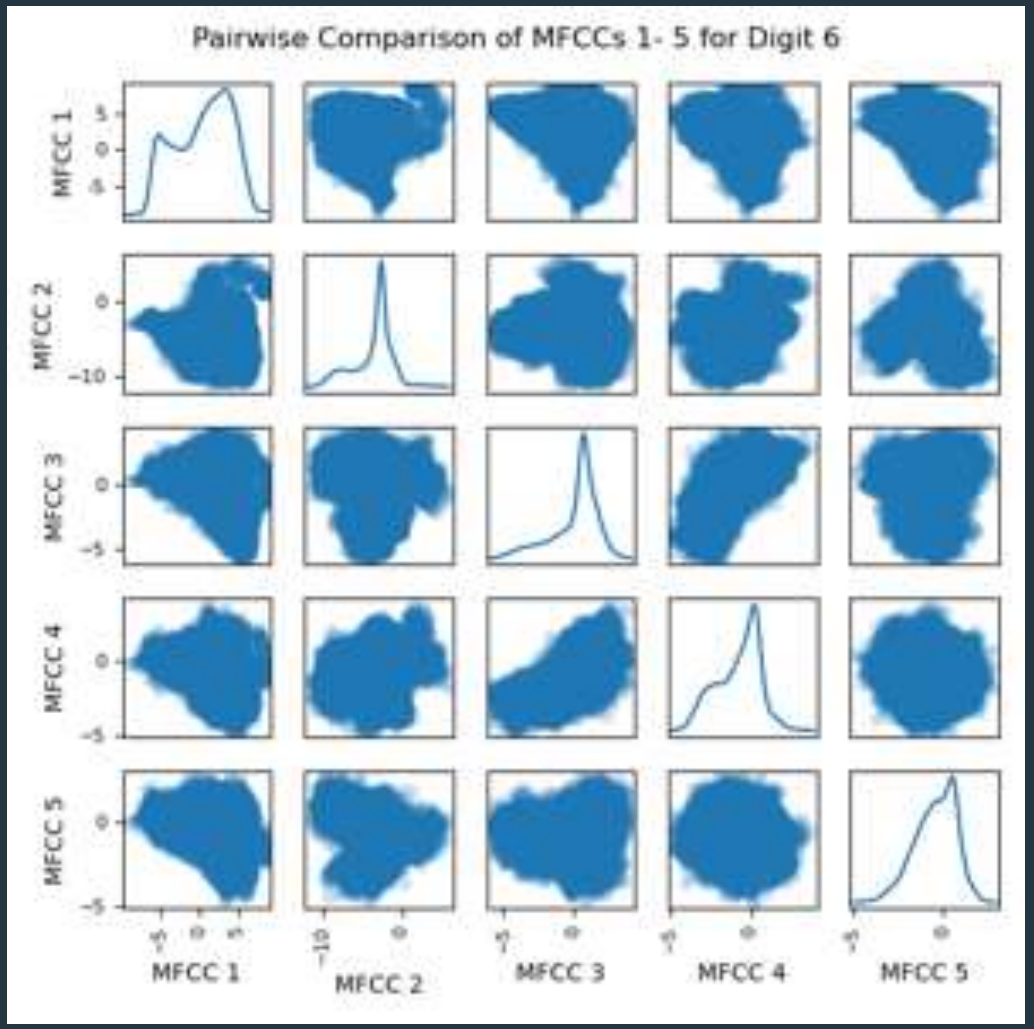


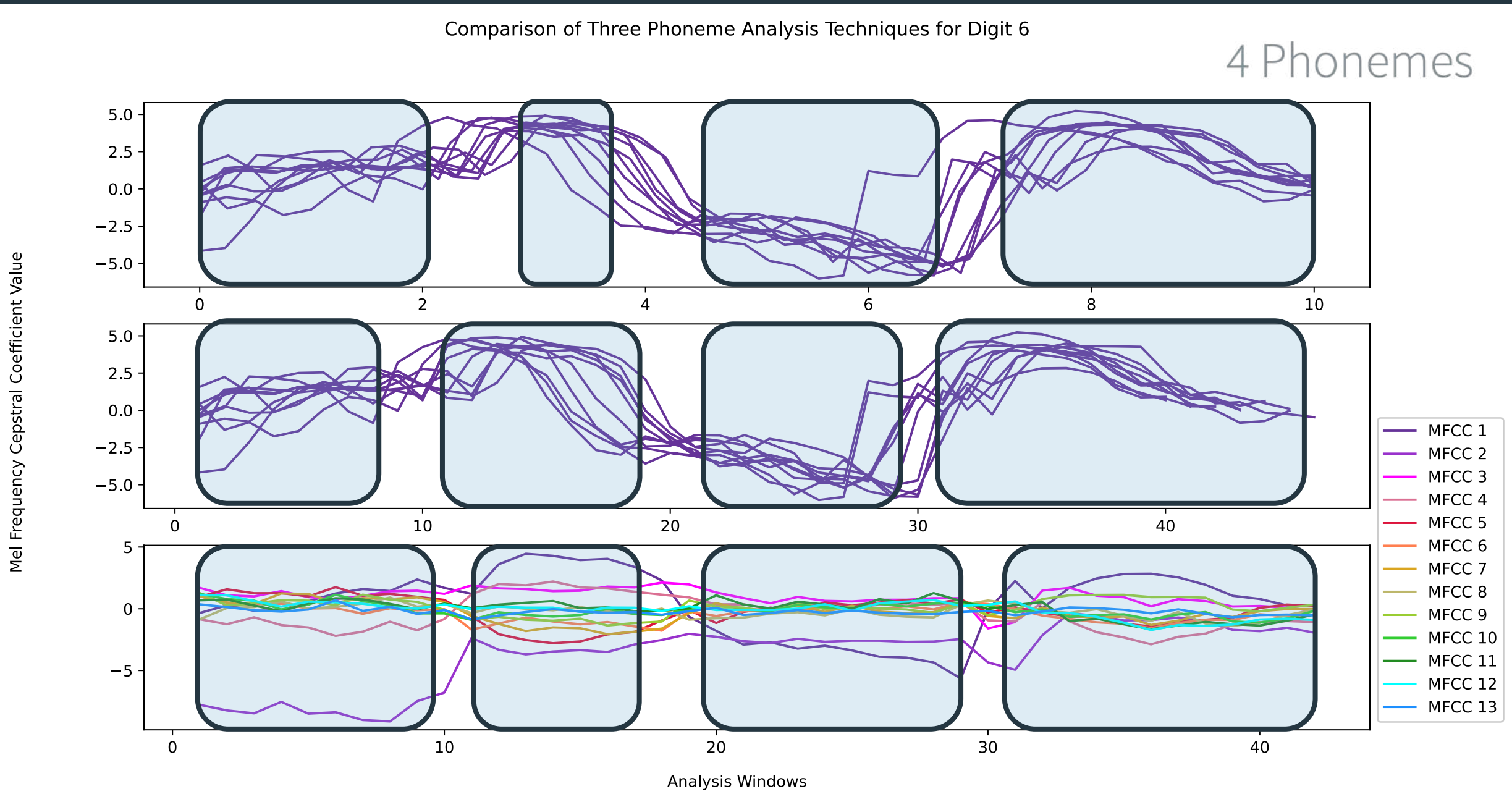
Digit 6 – sittah

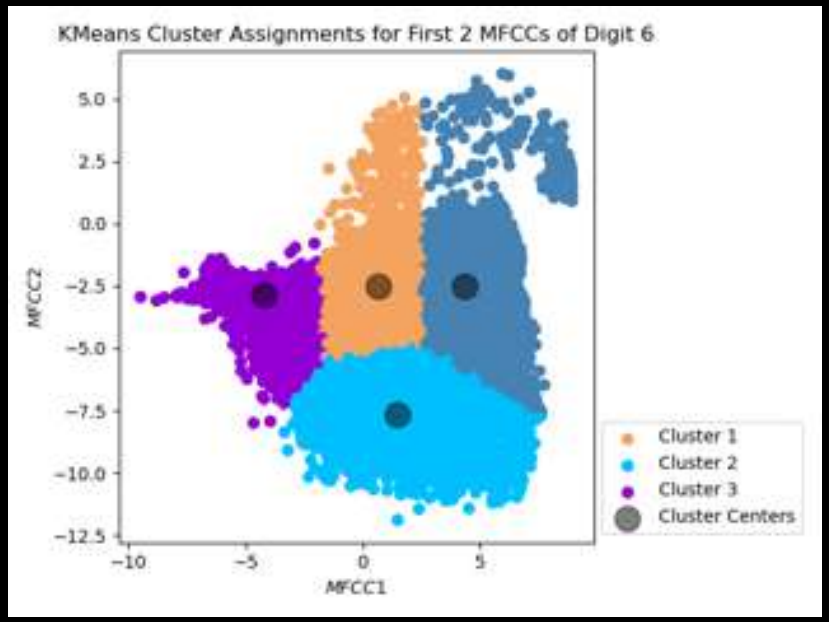
سِتَّه

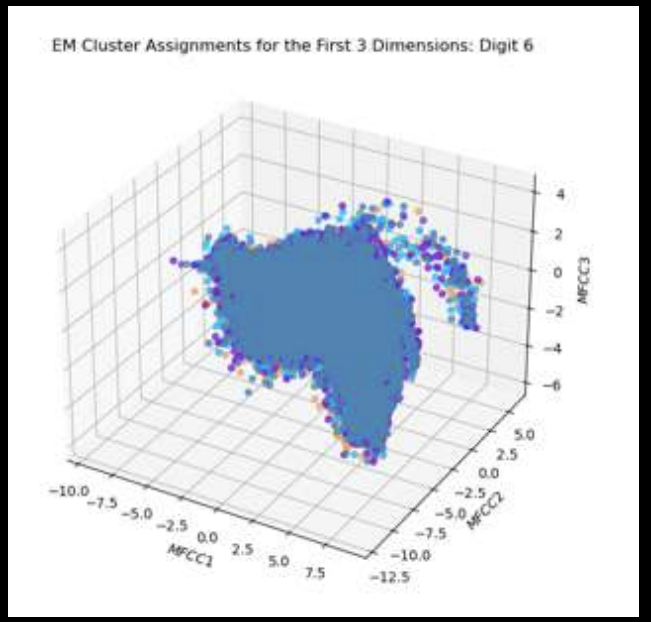
Visualisations of the Importance of Various MFCCs for Digit 6

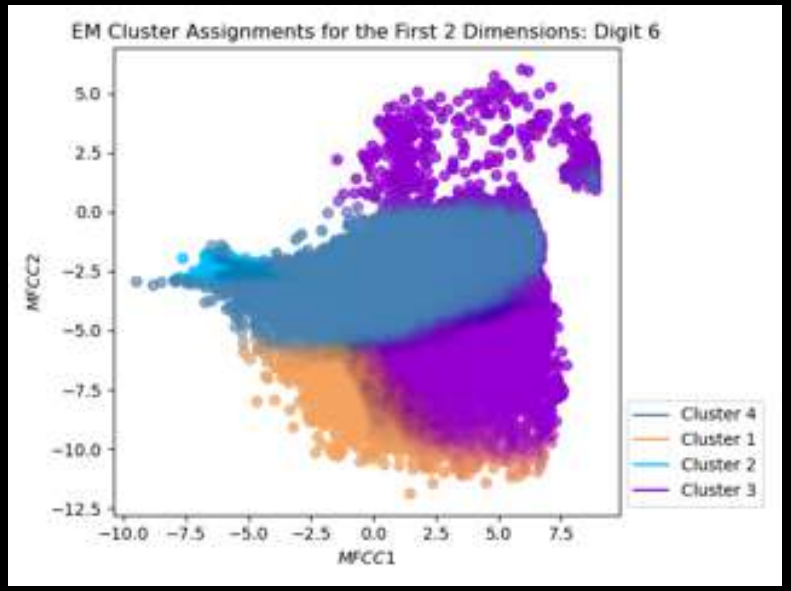


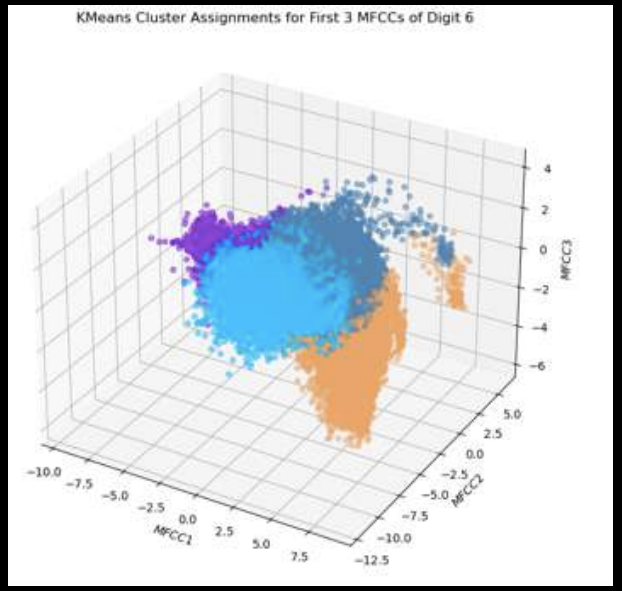



Comparison of Three Phoneme Analysis Techniques for Digit 6
4 Phonemes
Mel Frequency Cepstral Coefficient Value
Analysis Windows
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13

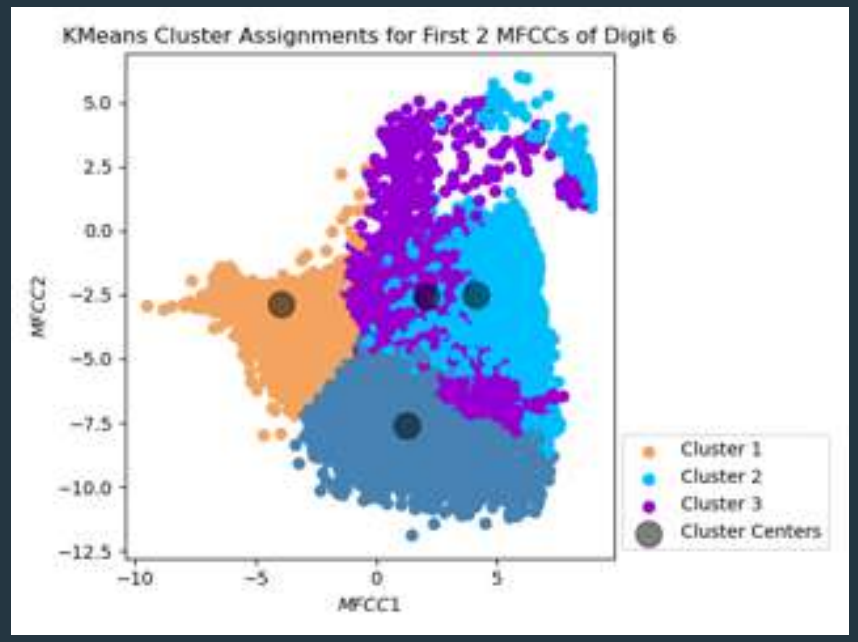

KMeans Cluster Assignments for First 2 MFCCs of Digit 6
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

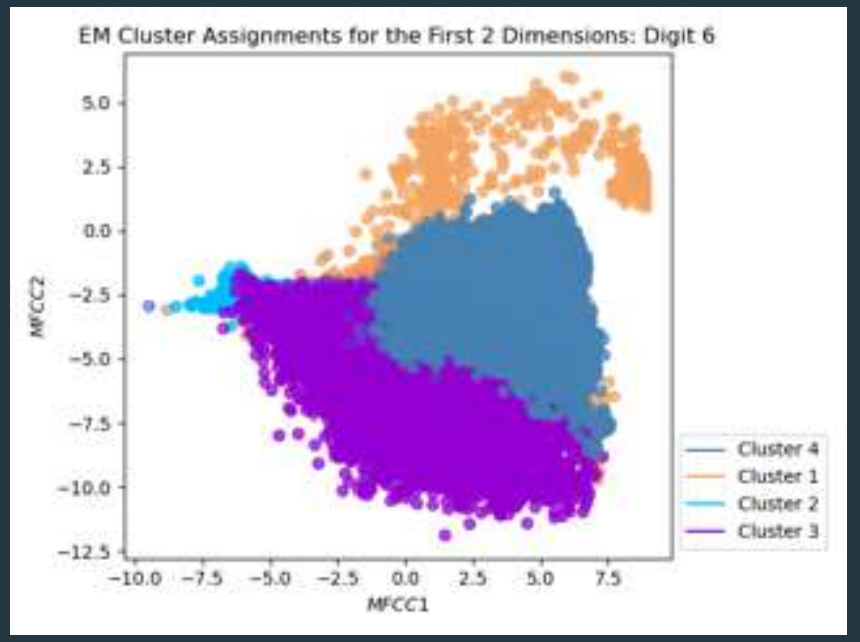

EM Cluster Assignments for the First 3 Dimensions: Digit 6
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 6
MFCC1
MFCC2
Cluster 4
Cluster 1
Cluster 2
Cluster 3


KMeans Cluster Assignments for First 3 MFCCs of Digit 6
MFCC1
MFCC2
MFCC3

Digit 6: 13 Dim Clusters Plotted in 2D

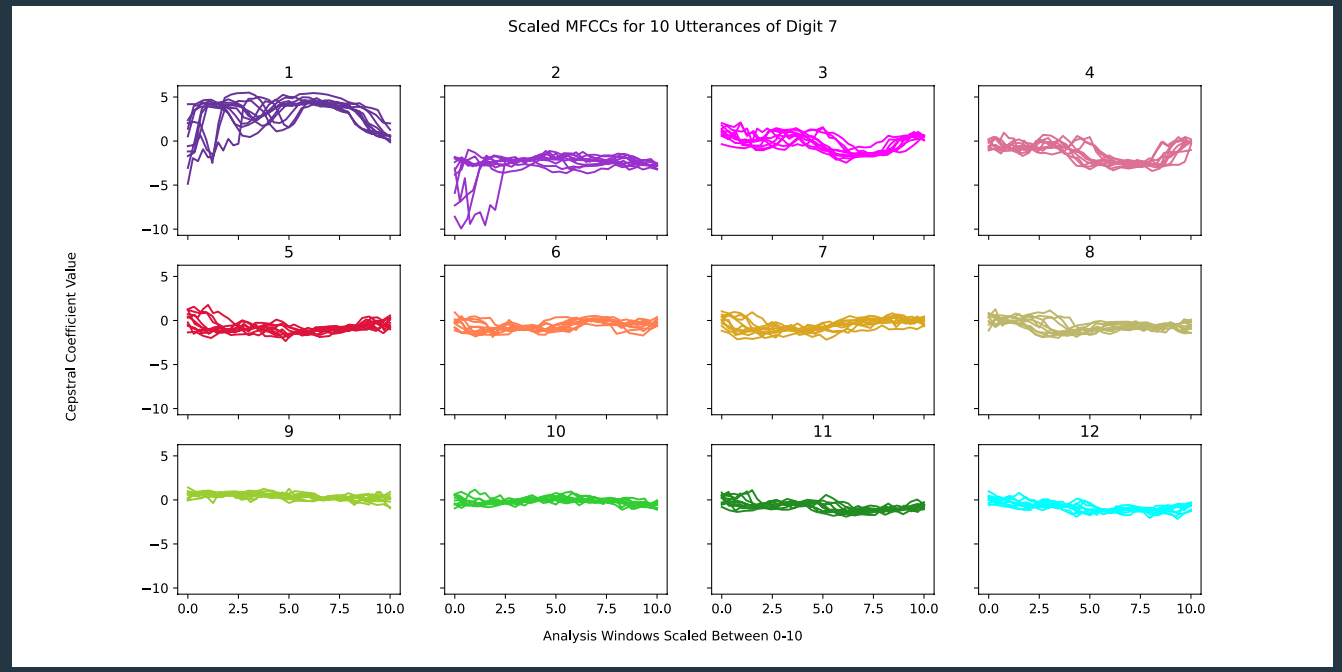


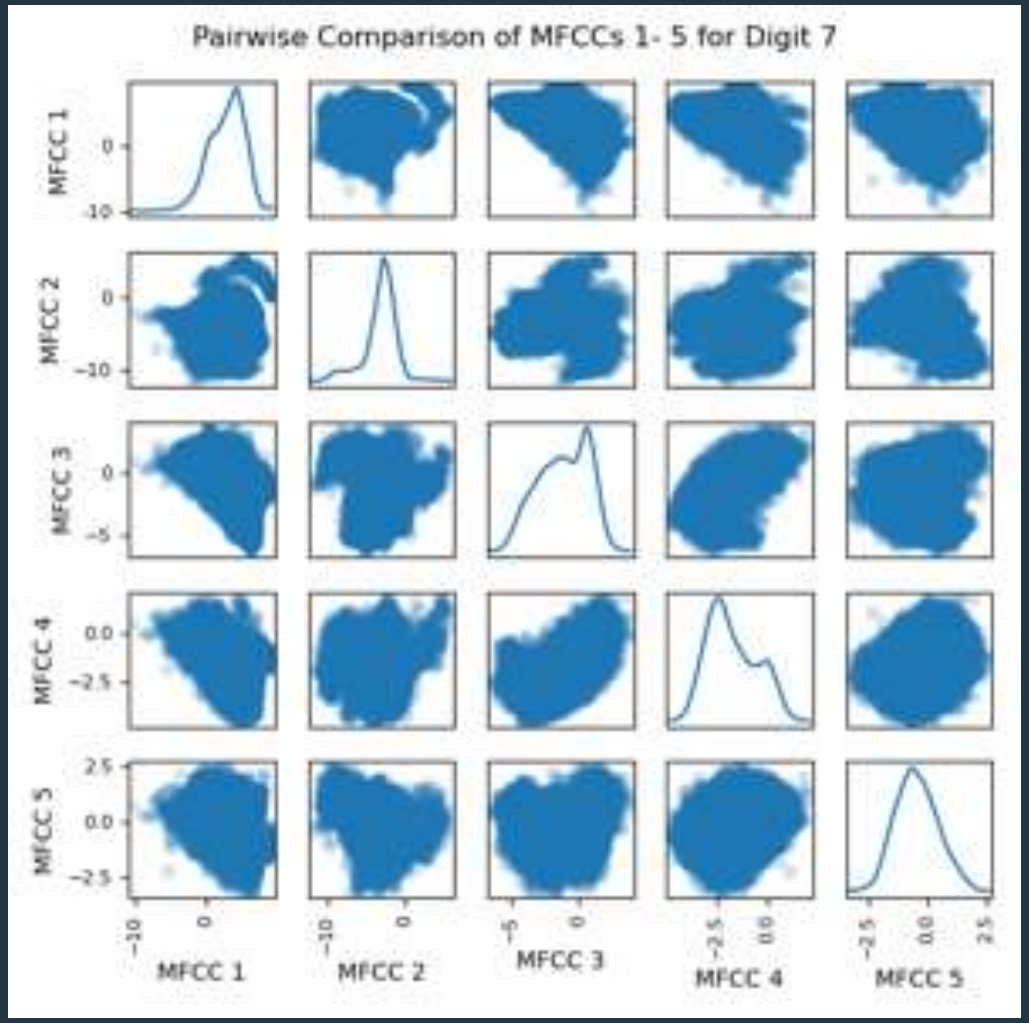


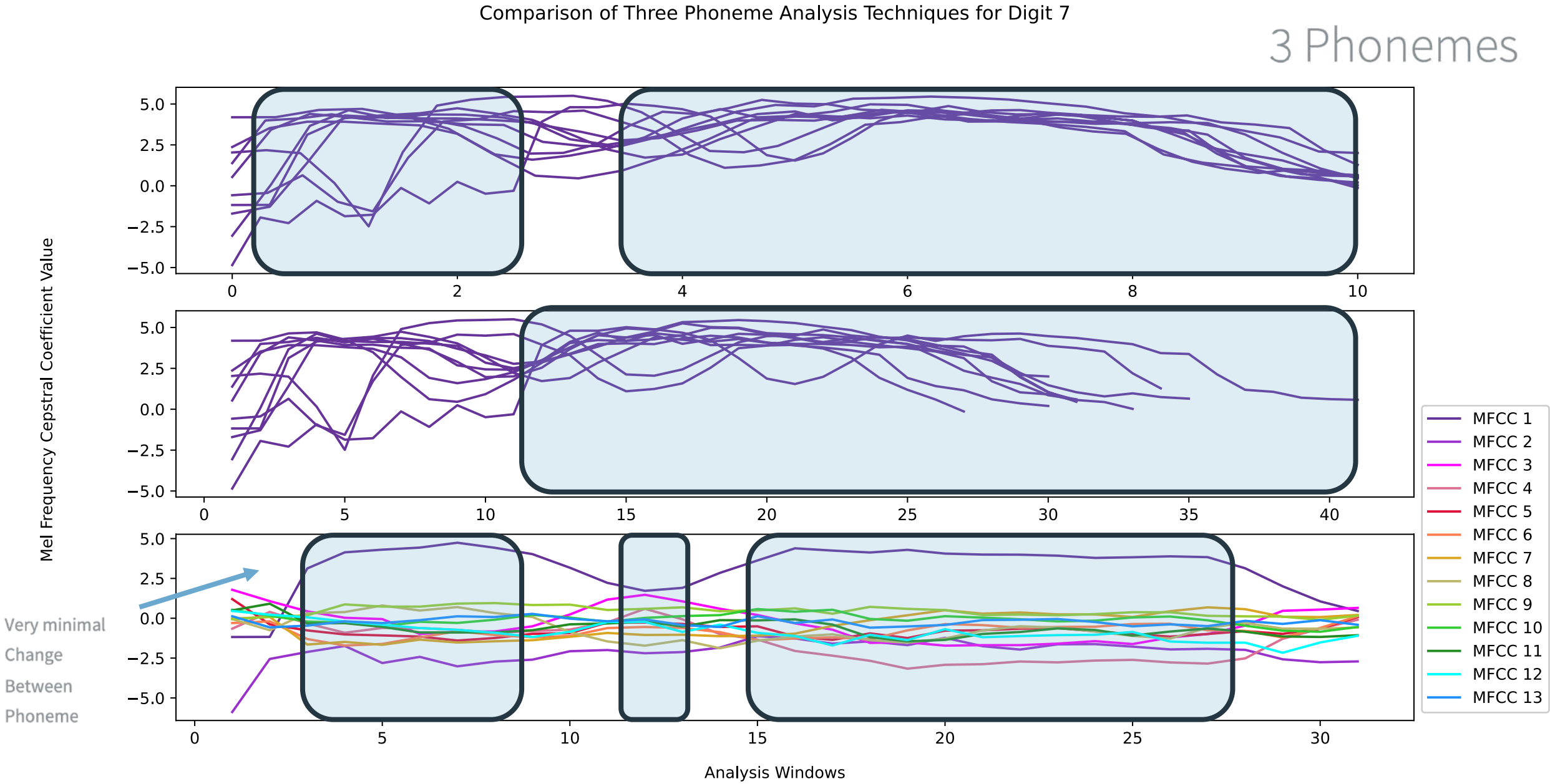
Digit 7 – seb’a

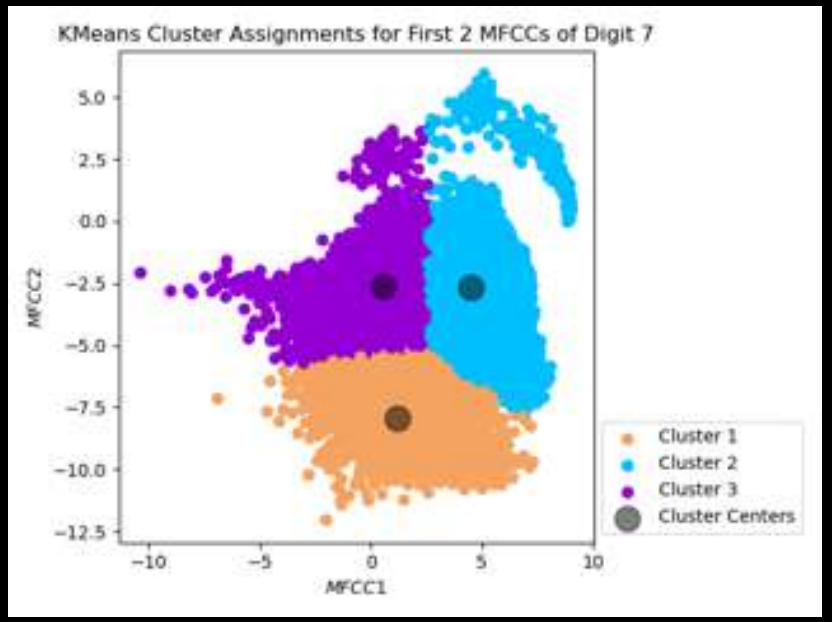
سَبْعَه

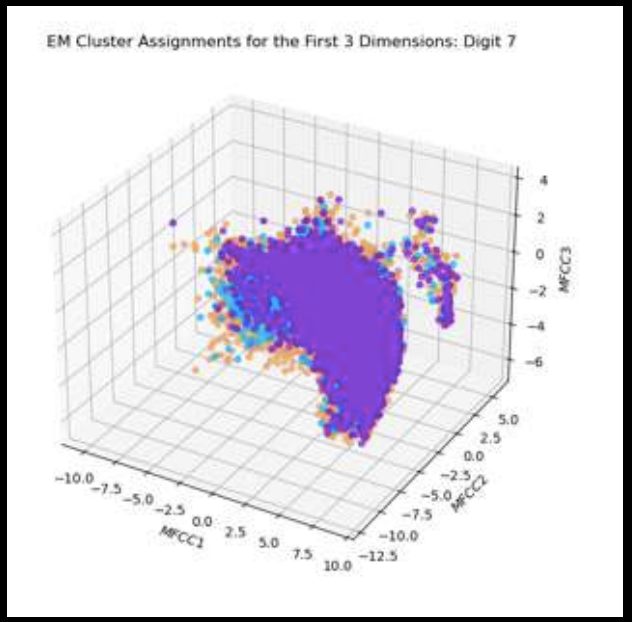
Visualisations of the Importance of Various MFCCs for Digit 7

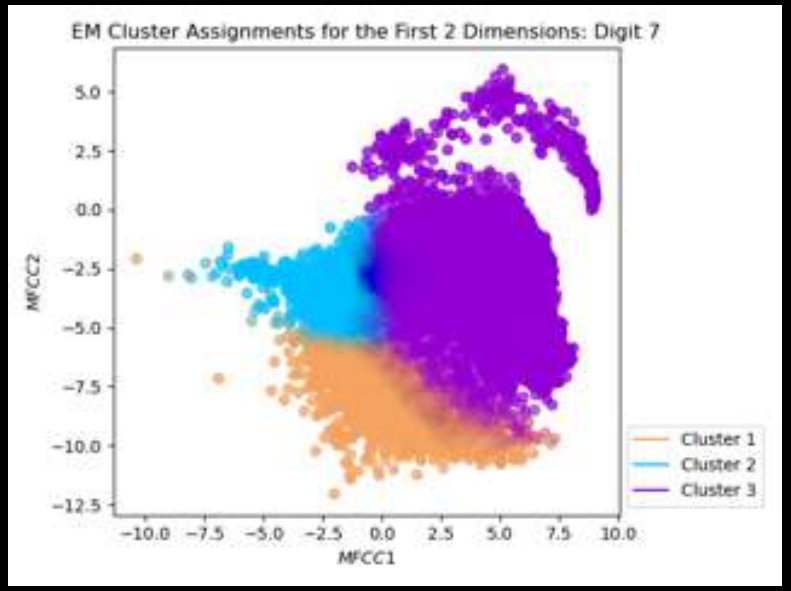


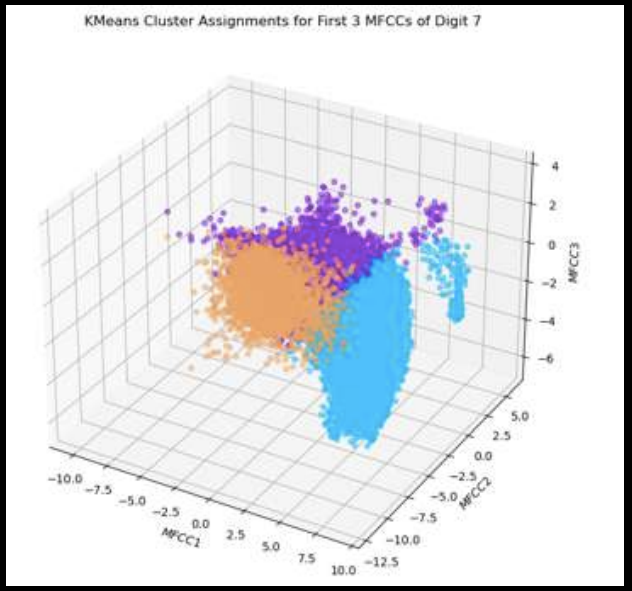



Comparison of Three Phoneme Analysis Techniques for Digit 7
3 Phonemes
Mel Frequency Cepstral Coefficient Value
Very minimal
Change
Between
Phoneme
Analysis Windows
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13

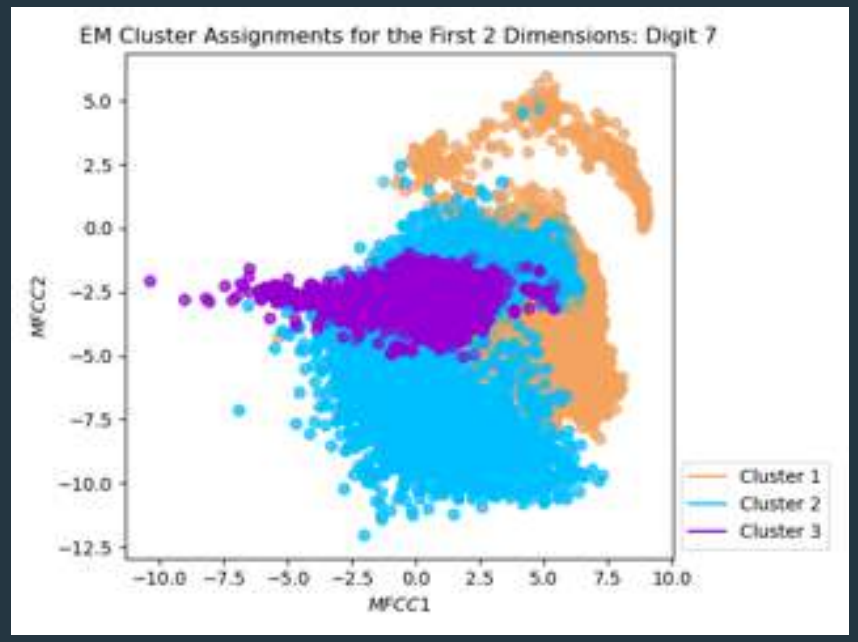

KMeans Cluster Assignments for First 2 MFCCs of Digit 7
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

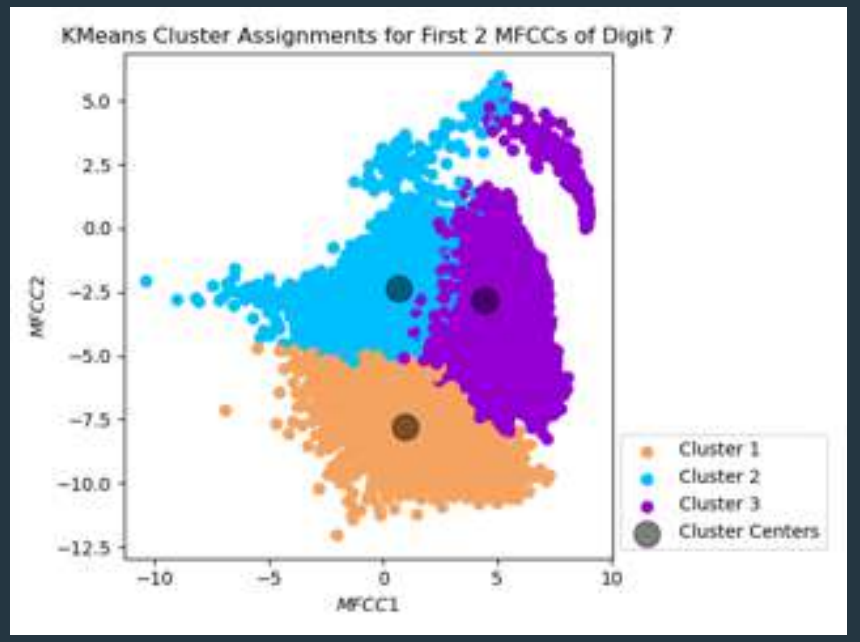

EM Cluster Assignments for the First 3 Dimensions: Digit 7
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 7
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3


KMeans Cluster Assignments for First 3 MFCCs of Digit 7
MFCC1
MFCC2
MFCC3

Digit 7: 13 Dim Clusters Plotted in 2D

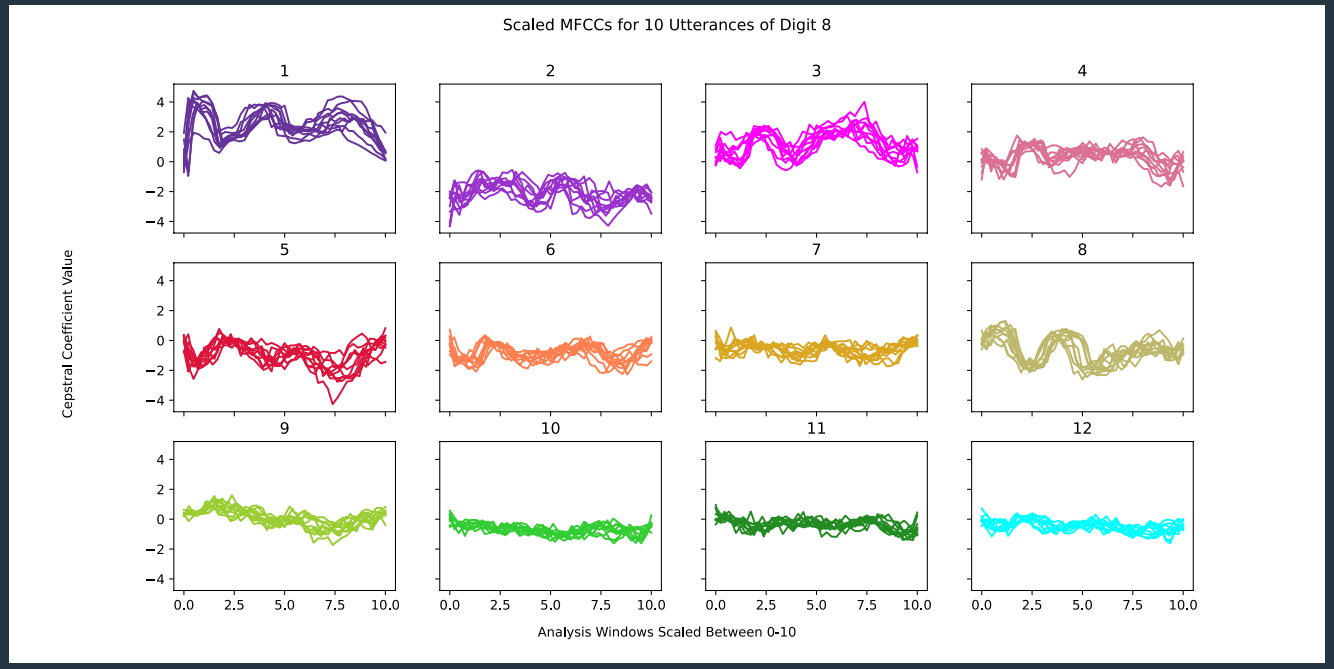


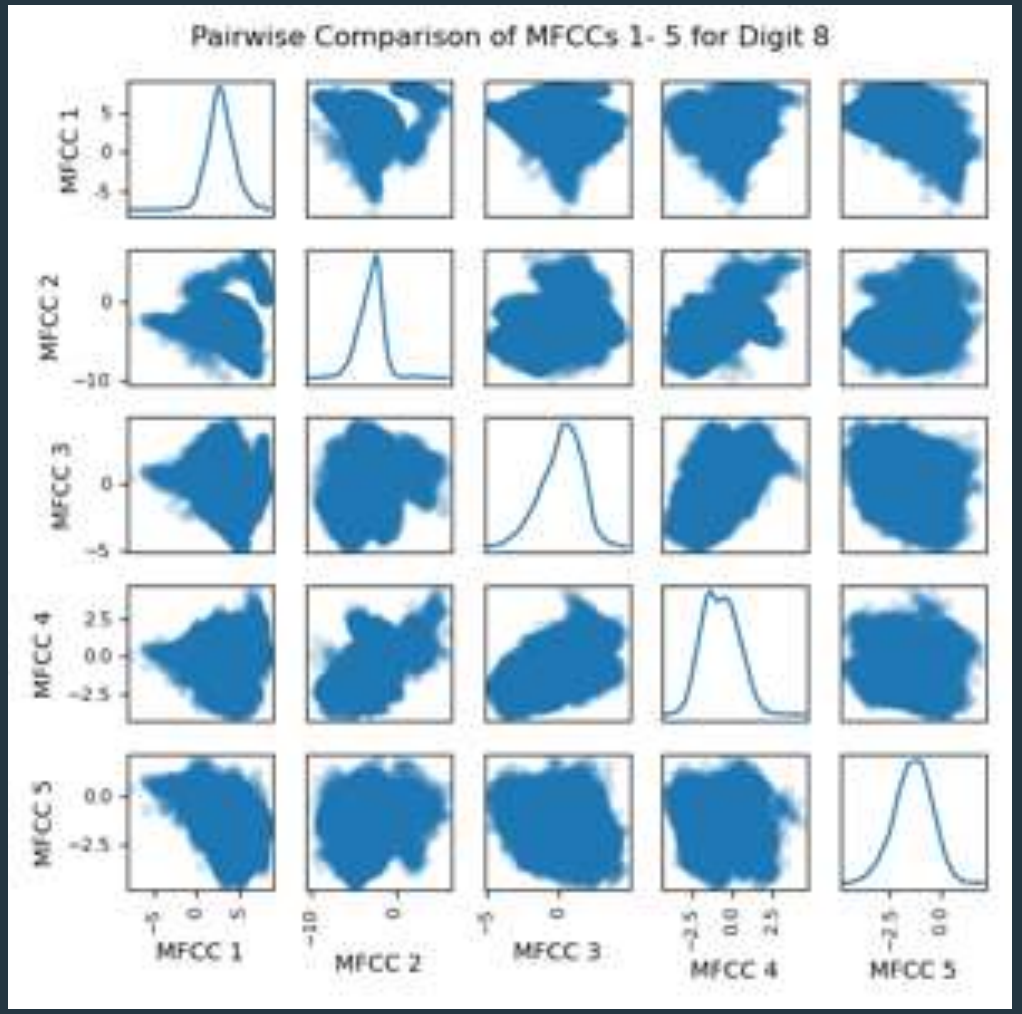


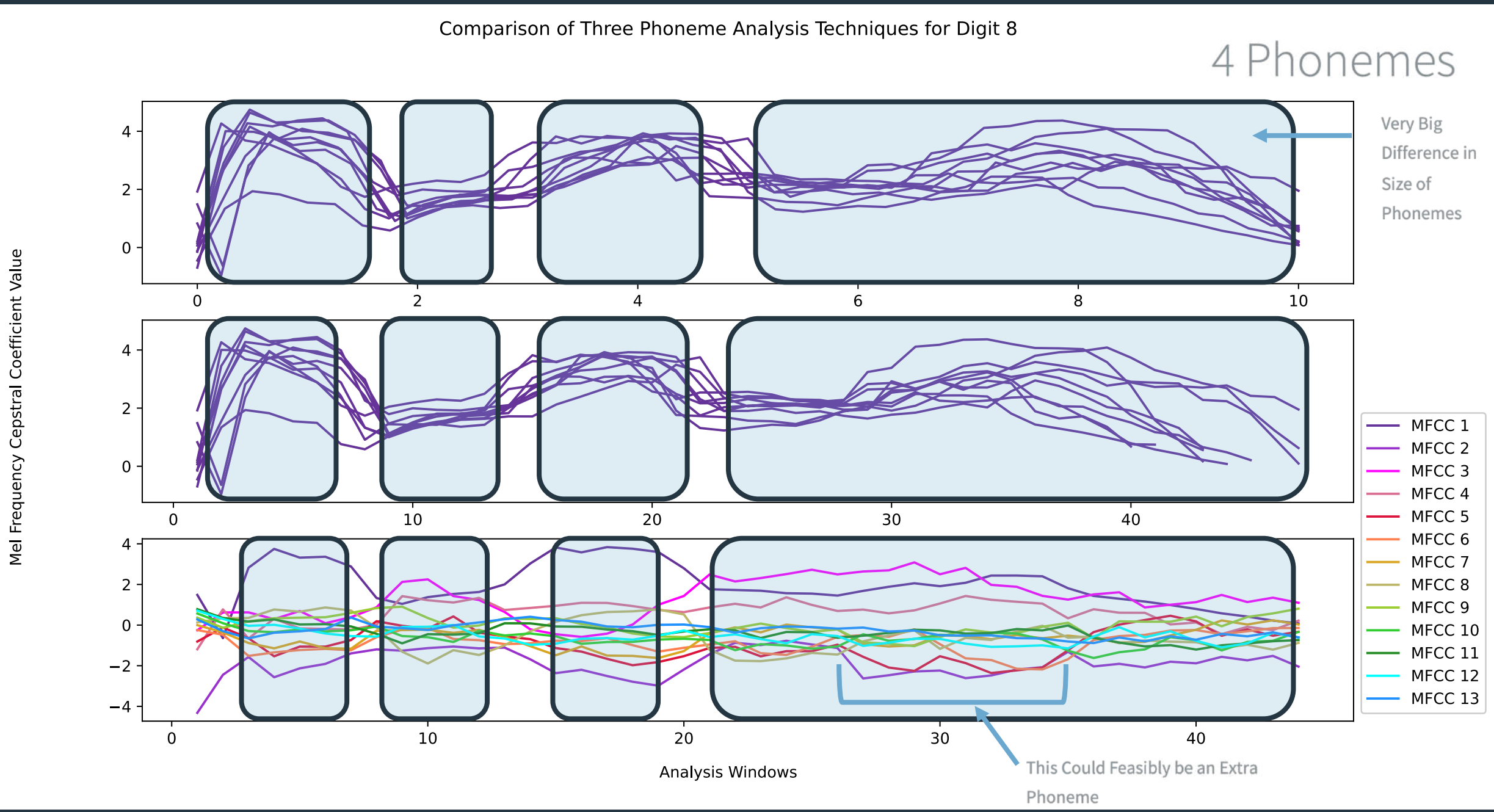
Digit 8 – thamanieh

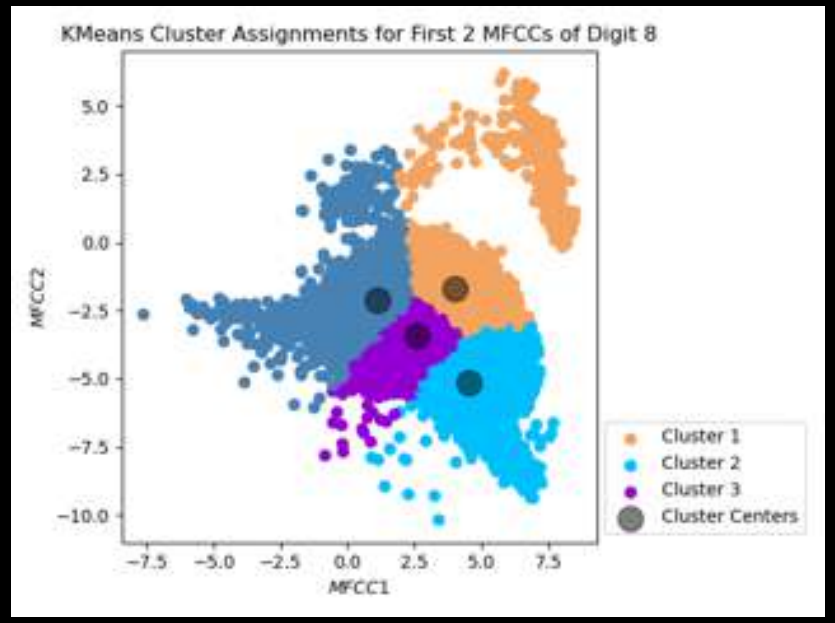
ثَمانِيَه

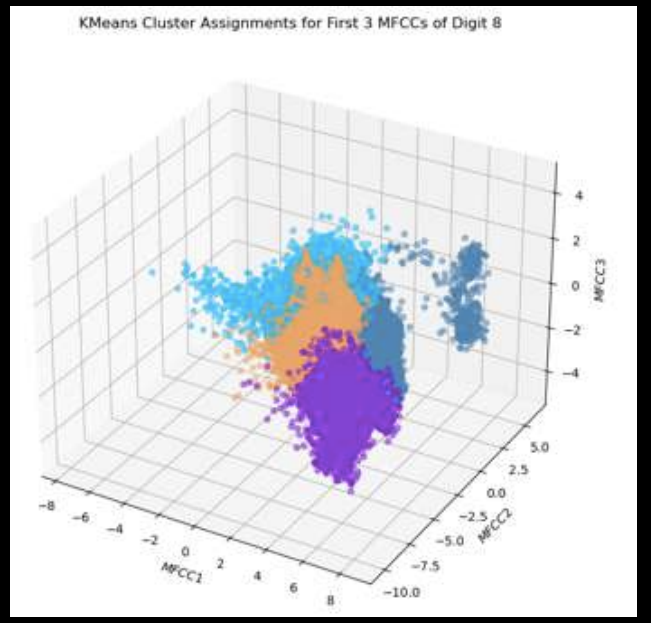
Visualisations of the Importance of Various MFCCs for Digit 8

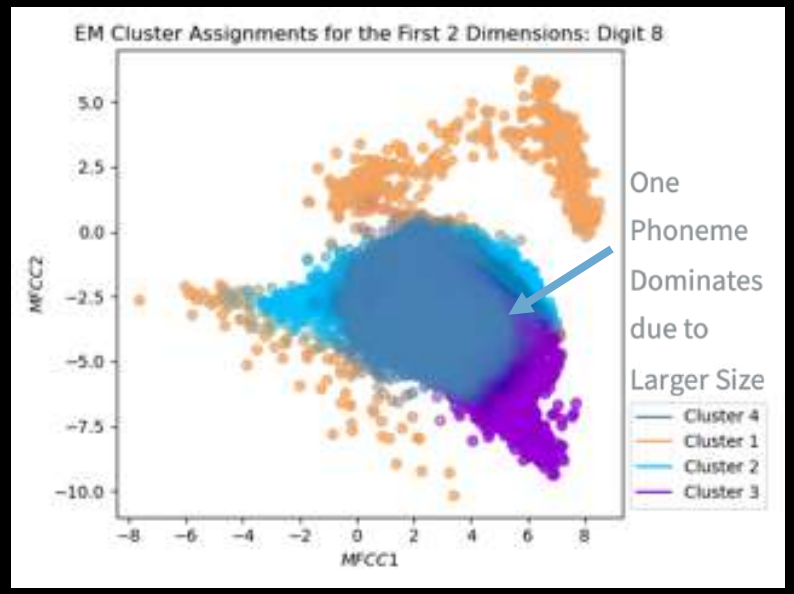


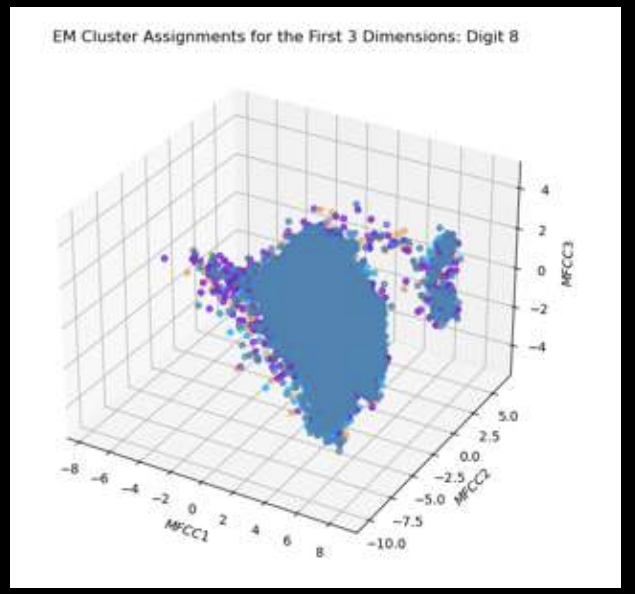



Comparison of Three Phoneme Analysis Techniques for Digit 8
4 Phonemes
Very Big Difference in Size of Phonemes
Mel Frequency Cepstral Coefficient Value
Analysis Windows
This Could Feasibly be an Extra Phoneme
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13

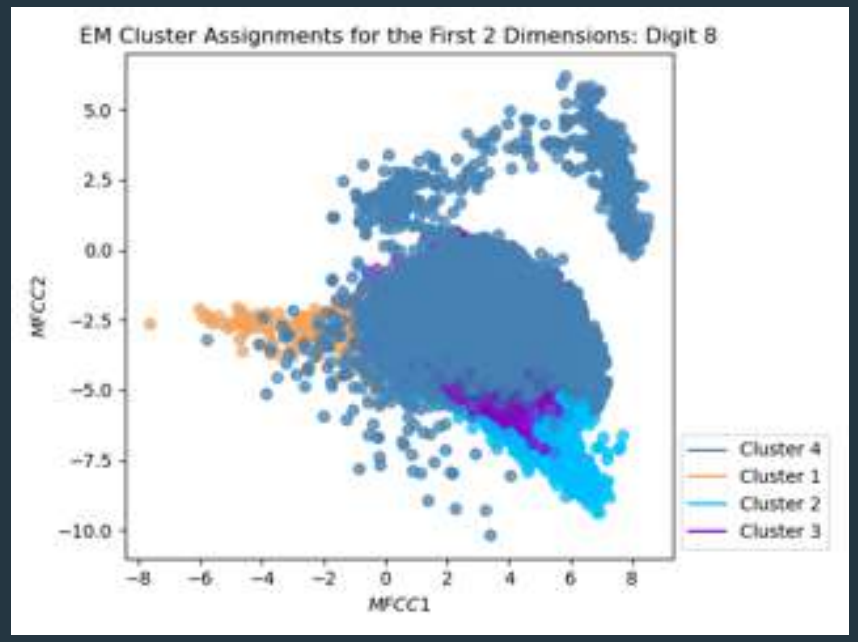

KMeans Cluster Assignments for First 2 MFCCs of Digit 8
MFCC1
MFCC2
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

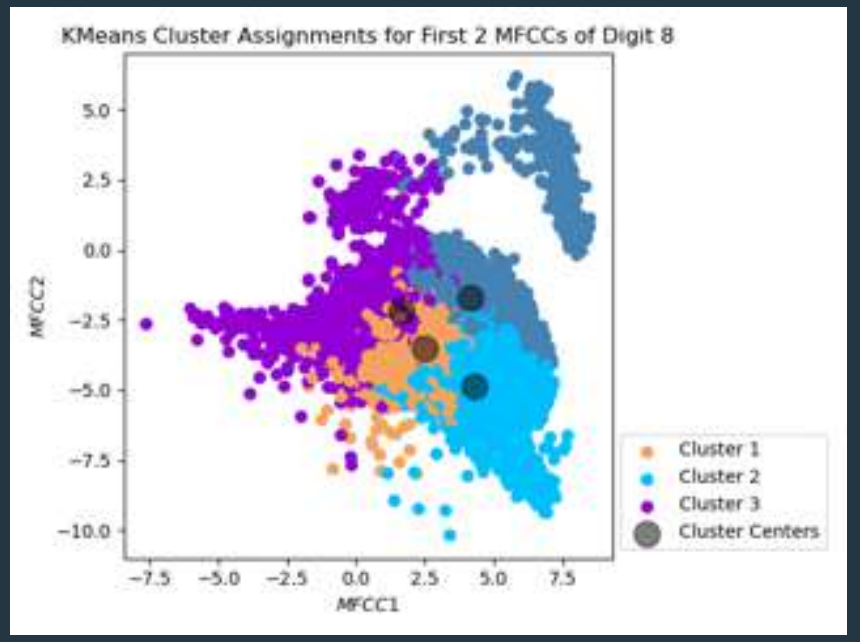

KMeans Cluster Assignments for First 3 MFCCs of Digit 8
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 8
MFCC1
MFCC2
One Phoneme Dominates due to Larger Size
Cluster 4
Cluster 1
Cluster 2
Cluster 3


EM Cluster Assignments for the First 3 Dimensions: Digit 8
MFCC1
MFCC2
MFCC3

Digit 8: 13 Dim Clusters Plotted in 2D

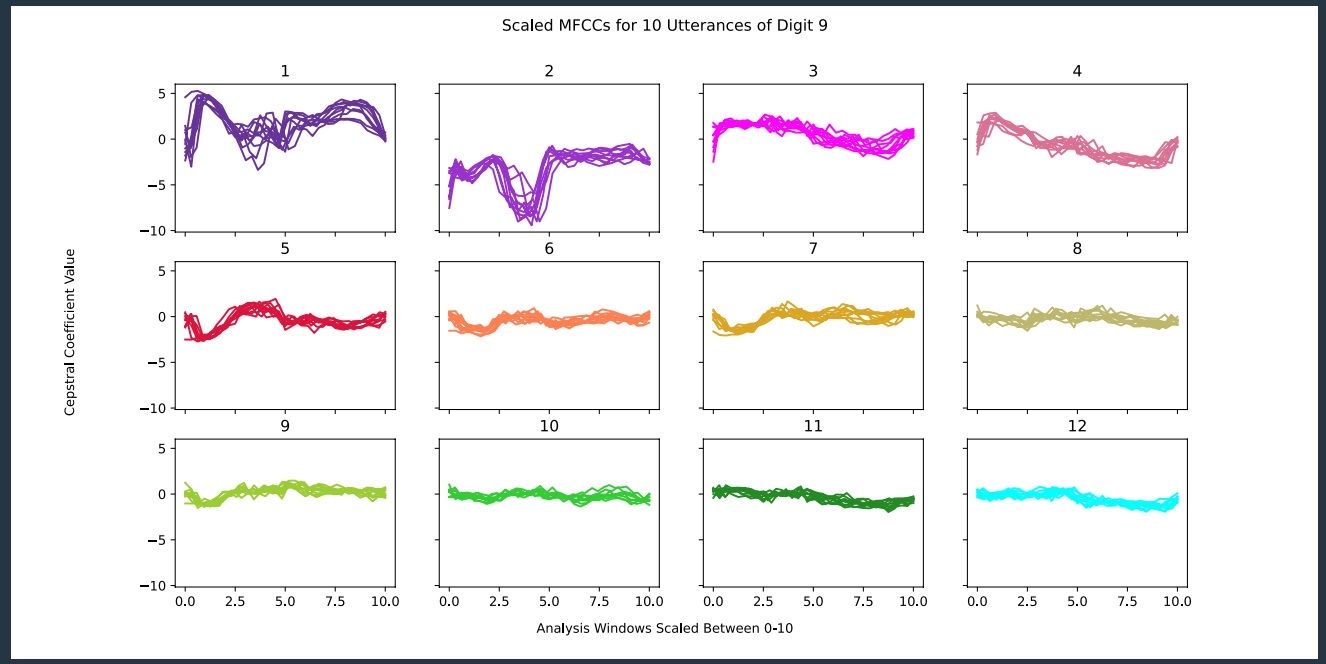


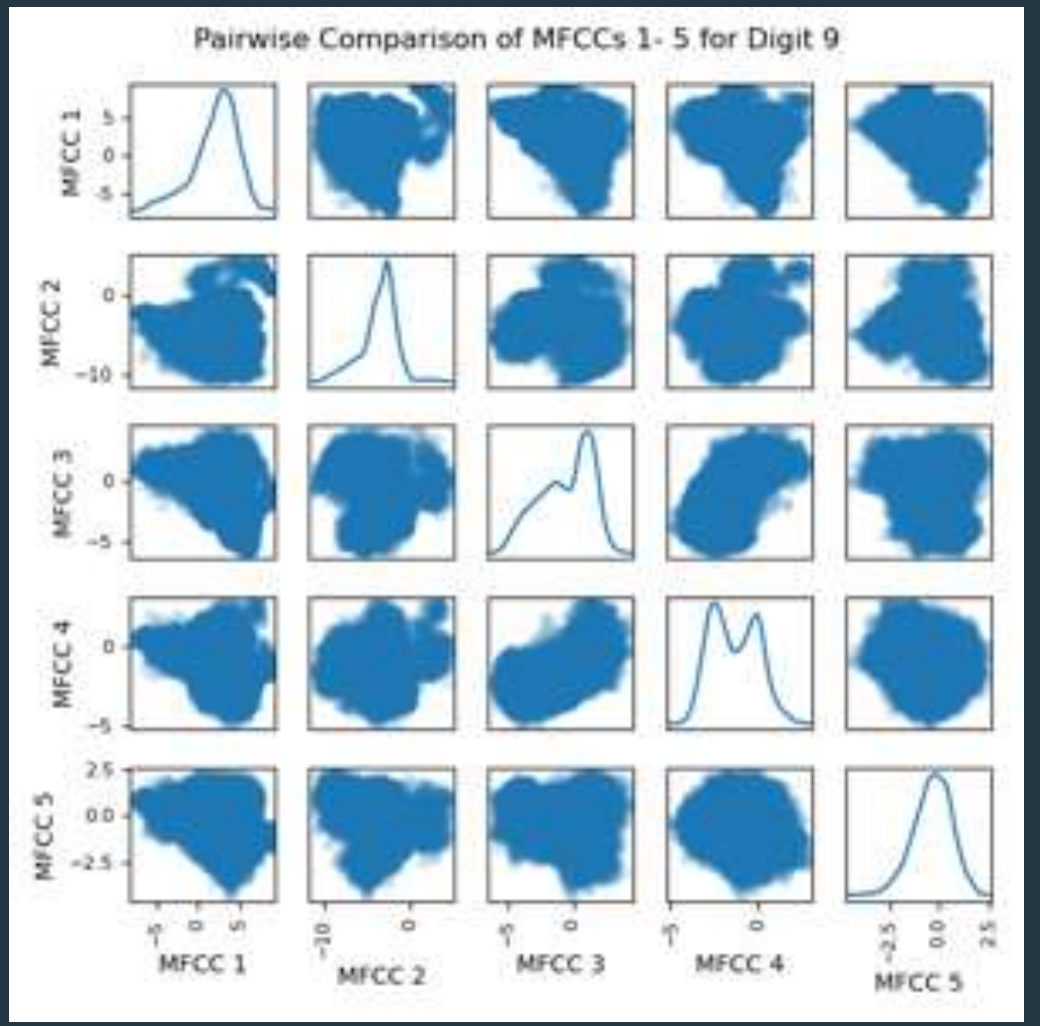


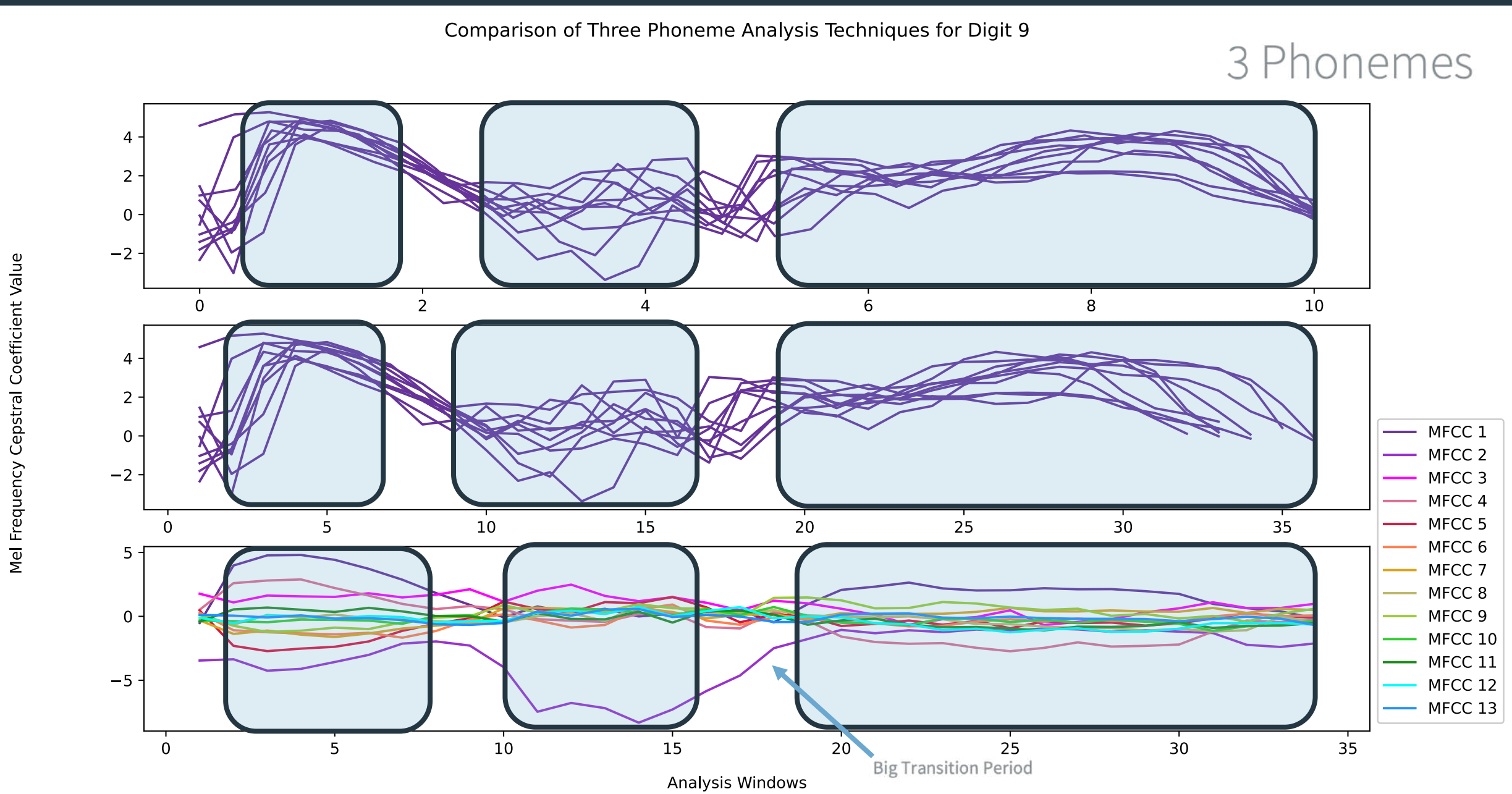
Digit 9 – tis’ah

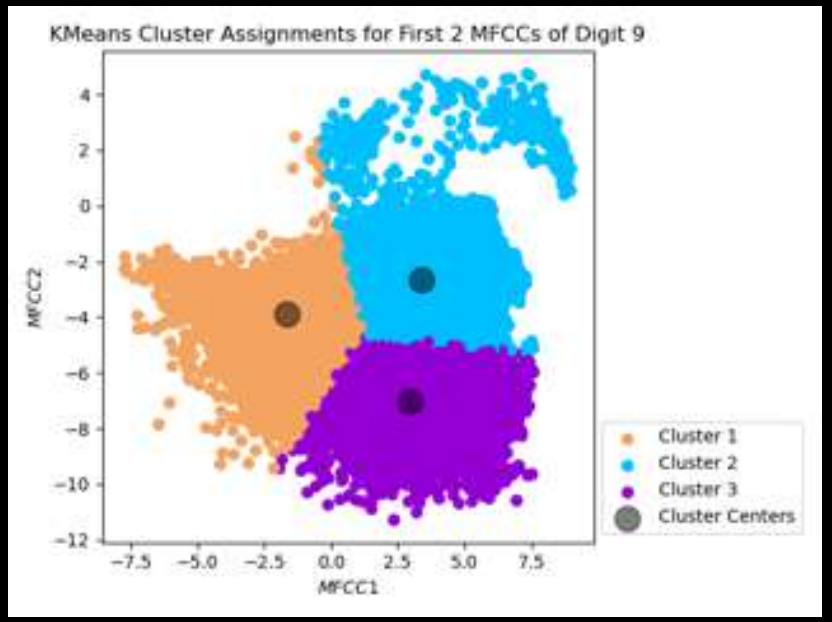
تِسْعَه

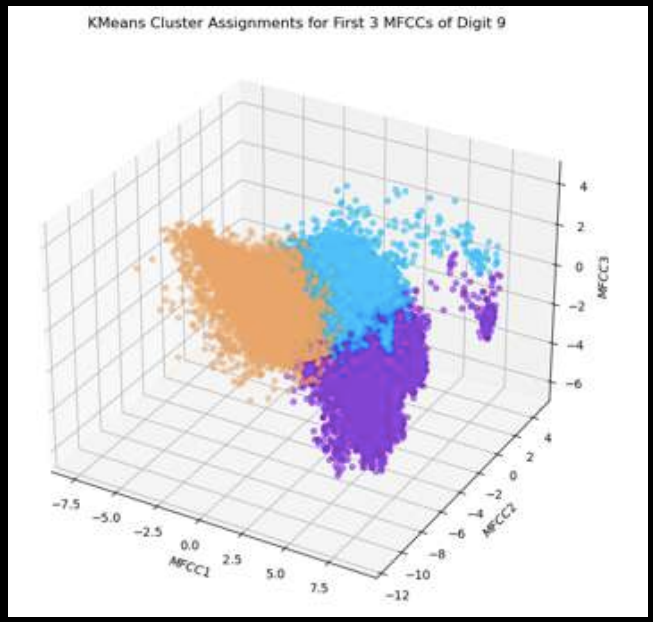
Visualisations of the Importance of Various MFCCs for Digit 8

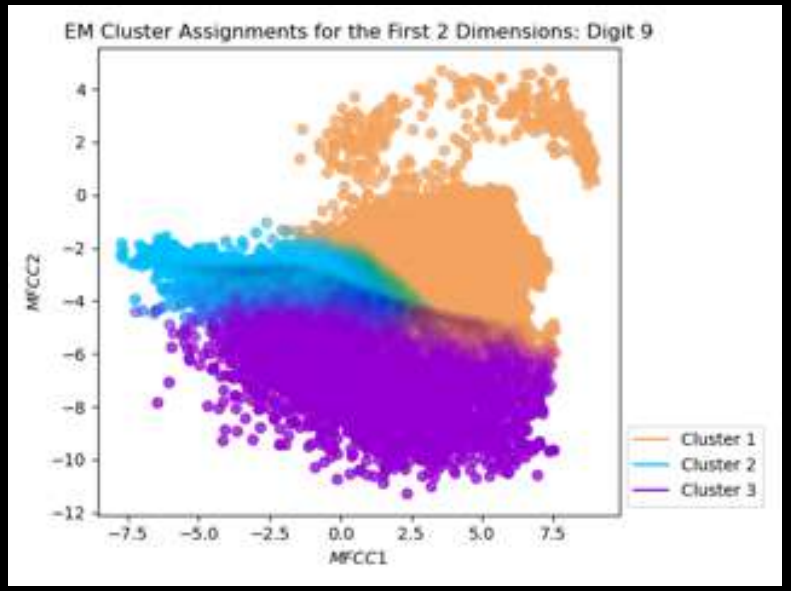


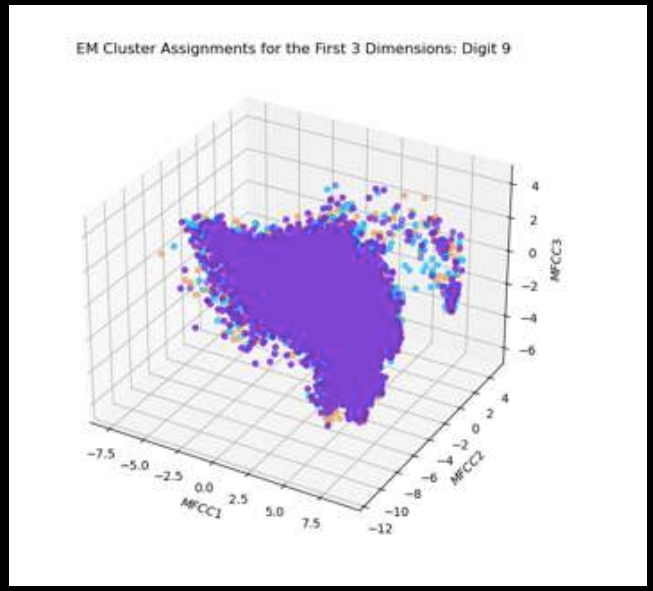



Comparison of Three Phoneme Analysis Techniques for Digit 9
3 Phonemes
Mel Frequency Cepstral Coefficient Value
Analysis Windows
Big Transition Period
MFCC 1
MFCC 2
MFCC 3
MFCC 4
MFCC 5
MFCC 6
MFCC 7
MFCC 8
MFCC 9
MFCC 10
MFCC 11
MFCC 12
MFCC 13

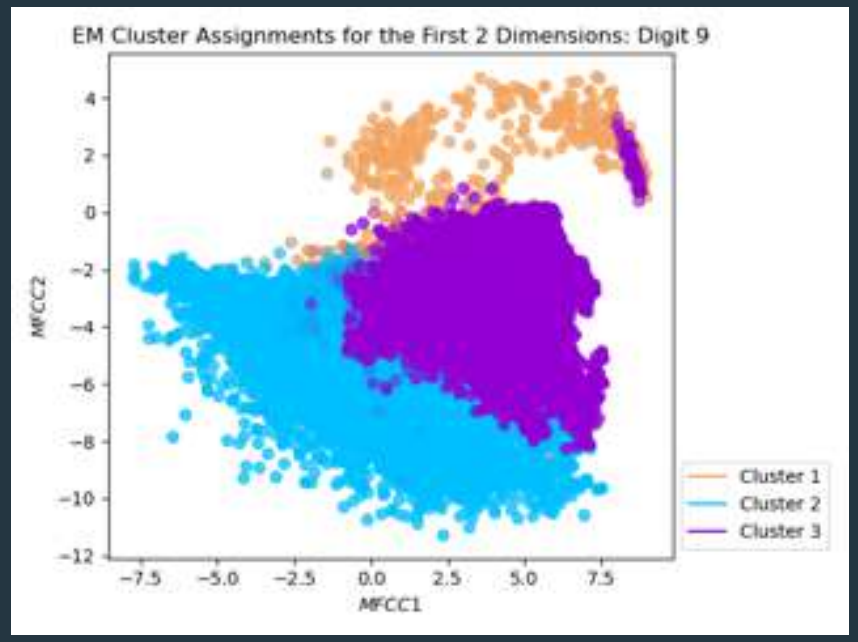

KMeans Cluster Assignments for First 2 MFCCs of Digit 9
MFCC2
MFCC1
Cluster 1
Cluster 2
Cluster 3
Cluster Centers

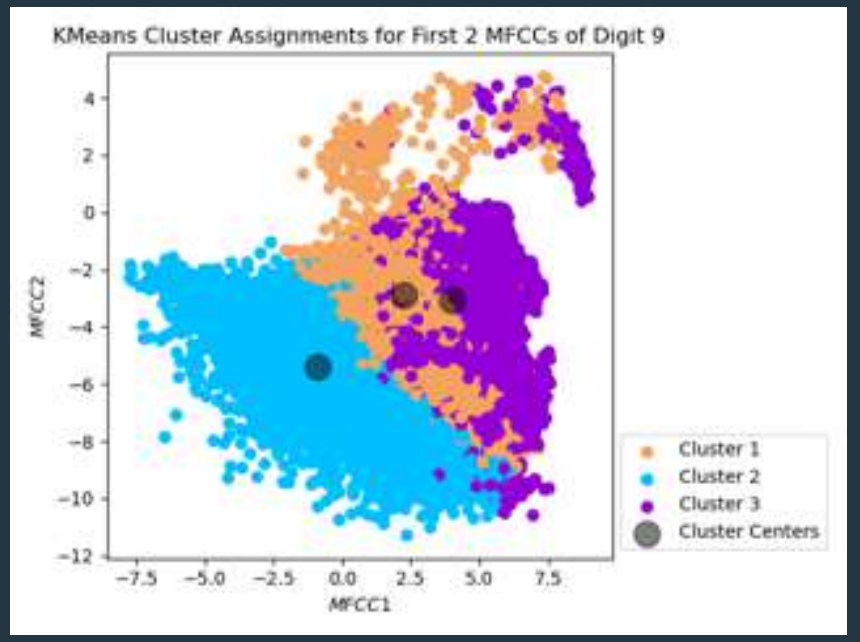

KMeans Cluster Assignments for First 3 MFCCs of Digit 9
MFCC1
MFCC2
MFCC3


EM Cluster Assignments for the First 2 Dimensions: Digit 9
MFCC2
MFCC1
Cluster 1
Cluster 2
Cluster 3


EM Cluster Assignments for the First 3 Dimensions: Digit 9
MFCC1
MFCC2
MFCC3

Digit 9: 13 Dim Clusters Plotted in 2D





Maximum Likelihood Classification (MLE)

Maximum Likelihood Classification is a process by which a parameter is estimated to be the value for which the data is most likely (Li and Jain). This involves creating a variety of models with training data and then fitting testing data to these models to see which matches the most accurately.

In Equation 1, it can be observed that a probability for the data, x_n, is found given that it is fitted to some model, Δ, defined by the mean, m, and covariance, d, of a gaussian mixture model. Each of these probabilities for the is normalised by the likelihood that it is within a specific cluster, π_m. The normalised probabilities for each of the clusters is summed together and then each of these results are multiplied together for each piece of data in a dataset.

This equation shows that the maximum likelihood classification involves finding the probability of the data fitting a specific model, and then it is possible to find the model which has the maximum probability for the data matching it.

$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{N} \Sigma_{m=1}^{M} \pi_{m,d} p(x_n|\Delta_{m,d})$$

Equation 1: Maximum Likelihood Classification (Tantum)

To increase the complexity of a model extra latent or nuisance variables may be added. An example of this within this project is the separation by gender. To ensure the clusters were not being affected by the frequency of the speakers' voice, a model was created from the female testing data that was separate to the model from the male testing data for each digit. This means the testing data was fitted to both models and the probability of it being in the male digit k and the female digit k were summed together. Therefore the equation would have an extra variable to sum over so the right hand side would look like $p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{N} \Sigma_{g=female}^{male} \pi_{m,d} p(x_n|\Delta_{m,d}|gender_g)$.

Maximum likelihood is well suited for this problem, because there are specific, discrete, results that are meant to come out of this problem, but there may be variation to the model which causes inaccuracy.

Bias Variance Trade-off

The Bias Variance Trade-off is an important consideration to make when forming a model. When designing a model, a certain number of parameters must be chosen. With more parameters, a model can account for more of the variables that affect the data, however, when more parameters are added, there is a higher likelihood that the model will be matched very closely with the training data, but may not be able to be extrapolated accurately to the testing data. Herein lies the bias variance trade-off. The aim is to create a model which is complex enough to account for all of the parameters, but which is not so complex that the variance between datasets is not able to fit comfortably within the model.

Bias is categorised by the minimisation of error so that $E\{e^2\} = E\{(y_u - \hat{y})^2\} + E\{n^2\}$ where e is the error, $y_u - \hat{y}$ is the difference between the true output and the model output and n is the noise. Therefore the equation can be written as $(model\ error)^2 = (bias\ error)^2 + variance\ error$.

These equations clearly show that as a models complexity increases the bias error decreases but the variance error increases. Therefore, if a model is too specific, the $(bias\ error)^2$ term will dominate the equation. The bias error is due to the structural inflexibility of the model and can be represented as $y_u - E\{\hat{y}\}$, where y_u is the noise free output and $\hat{y}$ is the model output.

The variance error is due to the model being complex enough that it maps directly to its dataset. For example, if there was a single dataset which could be perfectly approximated by a 3rd order system, but a second dataset looked completely different, to have the same level of zero bias between the two data sets' models the second model would have to be considerably different to the first. This difference between the two models is the variance. To simplify equations needed to calculate the difference between models, the variance error can be calculated as the deviation of the model parameters from their perfect values.

To minimise the model error, it is important to have a very large training set. If the training data is theoretically infinite, the model will not change, no matter the testing data it is given, because the variation would have already been accounted for in one of the previous training datasets. Additionally, if the model has infinite input parameters, the bias error will go to zero , because every point is mapped perfectly. (Nelles)

In a non-theoretical situation these requirements translate to having as many pieces of training data as possible and including as many parameters in the model as is realistic. In this experiment 660 training datasets for each digit were used and various models had different parameters in order to find the optimal model without trading variance and processing time.

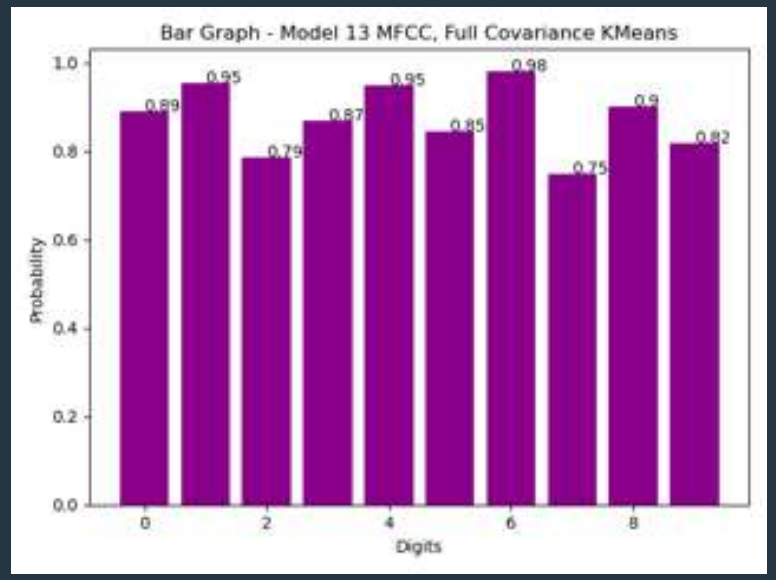
Model 1 – Kmeans - 13 MFCCs with Full Covariance

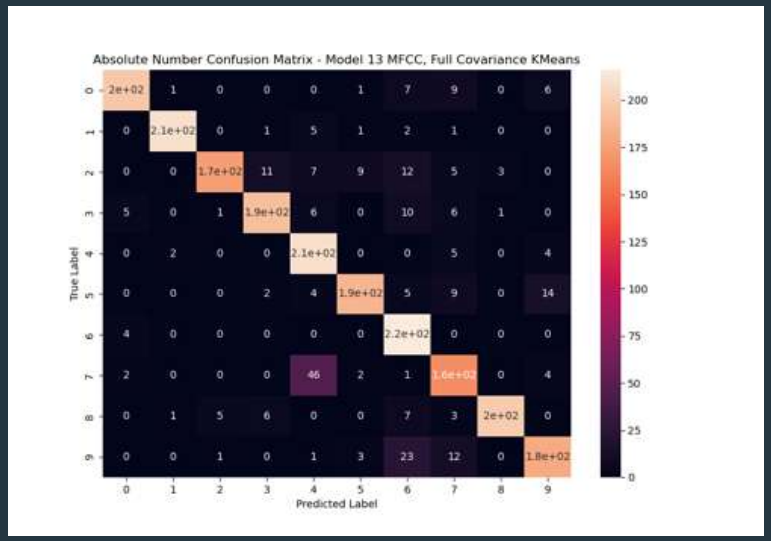
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \pi_{m,d} p(x_n|\Delta_{m,d=full})$$

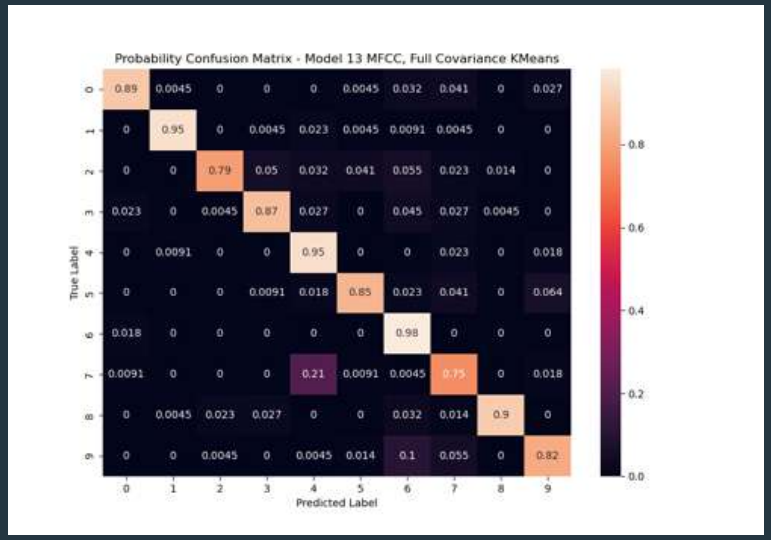
This was a very accurate model with the lowest accuracy rate being 75% for digit 7 and the highest accuracy being 98% for digit 1.

An outlying result of this model is that digit seven was predicted as four at quite a high rate of 21%. As was mentioned previously in this document, seven does not have very distinct phonemes and therefore was the most difficult of the group to categorise in a cluster. It is possible that since the first MFCC is not very distinctive since all of the MFCCs were included at equal weight the other MFCCs were similar to four's MFCCs and therefore dominated the clustering.

This model performs excellently for 2, 4 and 6 and very well for 0, 3 and 8. This may be because the later MFCCs are more distinct for these digits so including them in the clustering resulted in higher accuracy results.







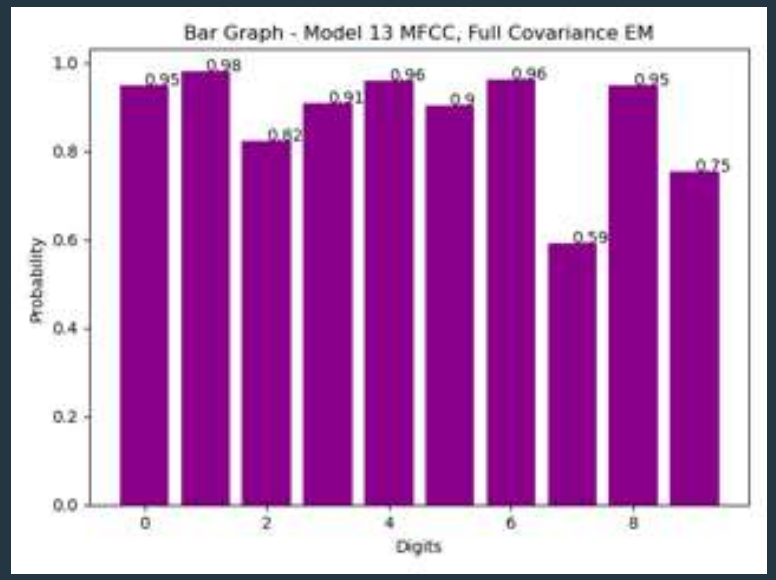
Model 1 – EM - 13 MFCCs with Full Covariance

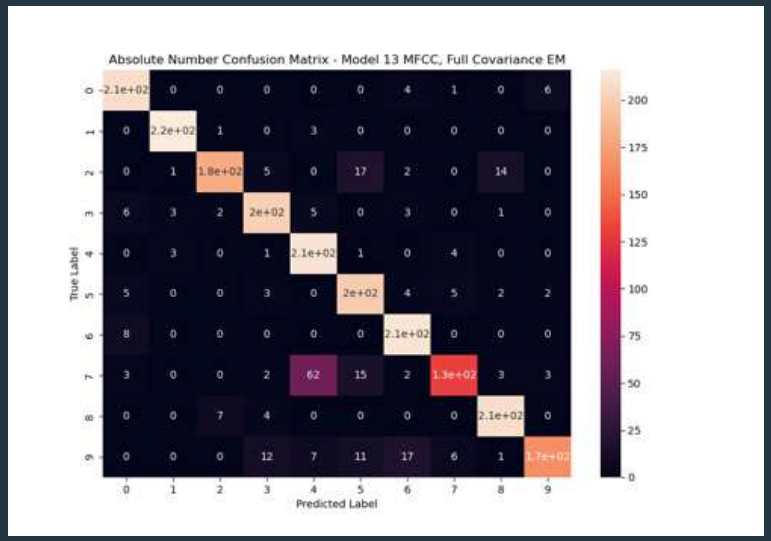
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \pi_{m,d} p(x_n|\Delta_{m,d=full})$$

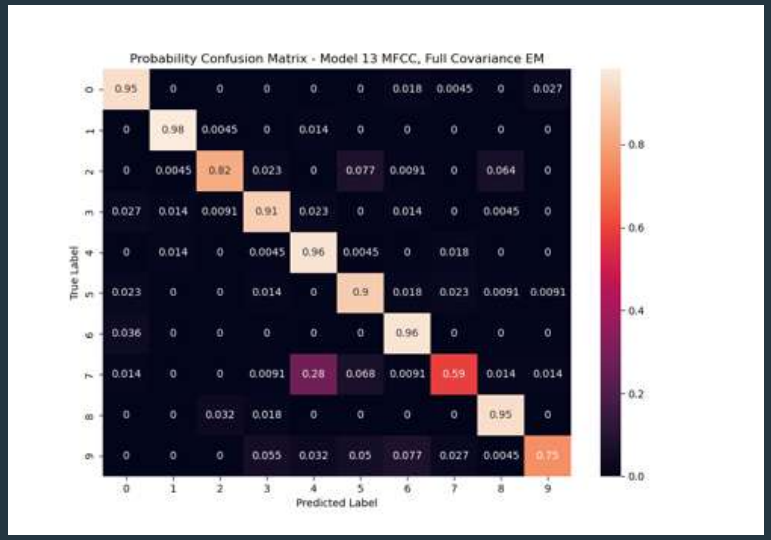
This was an even more accurate model for most digits. The accuracy of every digit except for 6, 7 and 9 increased. Unfortunately, 7, out lowest digit with the previous model decreased using the EM algorithm.

Since Expectation Maximisation involves clustering by finding the probability that each datapoint is in a cluster, perhaps the decrease in 7 is due to the datapoints being very similar because of minimal phoneme variation. Hence the clusters would have much smaller covariances and any of the testing data that was slightly off would appear that it was very unlikely to have come from the 7 model. This would be due to a very well fitted model that would have large variance between datasets. The bias variance trade-off would therefore be very large and dominated by the variance.

Since EM is more flexible than Kmeans, it accounted for more variables in most digits and was a better model.







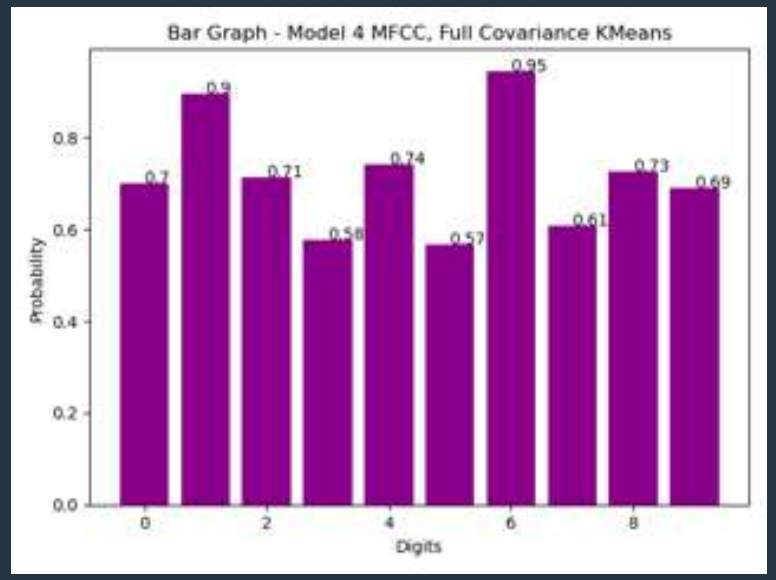
Model 2 – KMeans - 4 MFCCs with Full Covariance

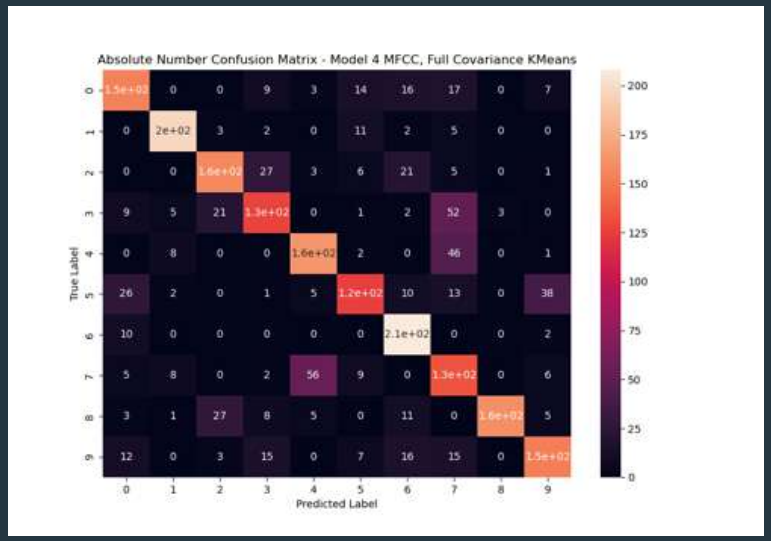
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \pi_{m,d} p(x_n|\Delta_{m,d=full})$$

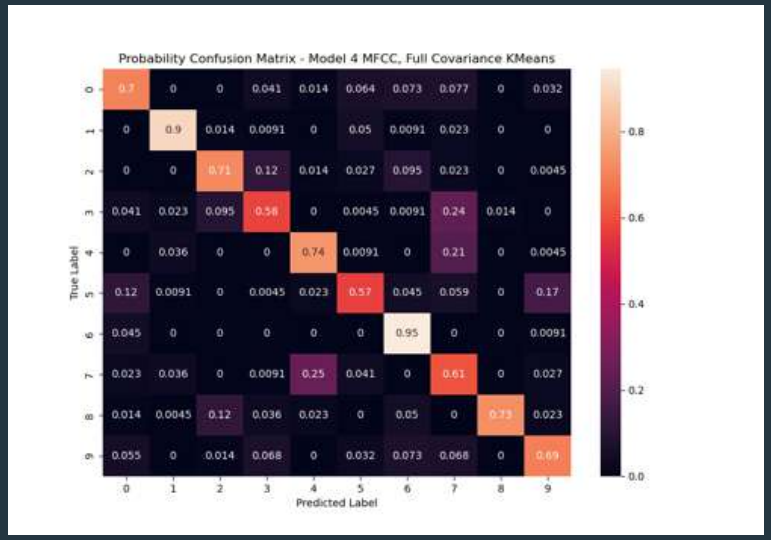
This was not a particularly good model for any of the digits. Digit 6 was an outlier with a probability of 95% and digit 5 had the lowest accuracy model with 57% accuracy.

Clearly, having 4 MFCCs is not enough to accurately predict digits being uttered. This is an interesting moment that shows the importance of having a flexible model to account for the less important dimensions. IN this case the bias variance trade-off is dominated by the bias term.

It is also interesting to note that while digit 8 wasn't predicted very accurately (73% accurate), none of the digits were predicted as digit 8. This indicates that the digit 8 model was a very tight fit and any variance to the inputted utterance would not fit this model.





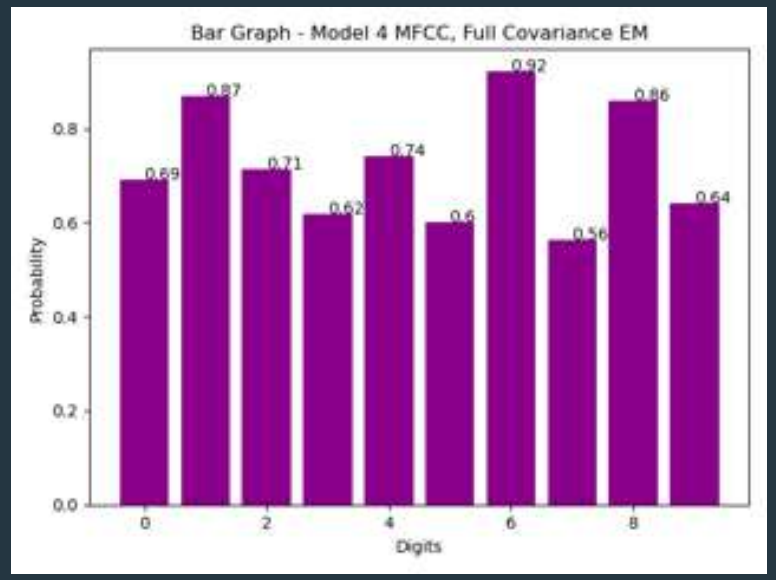


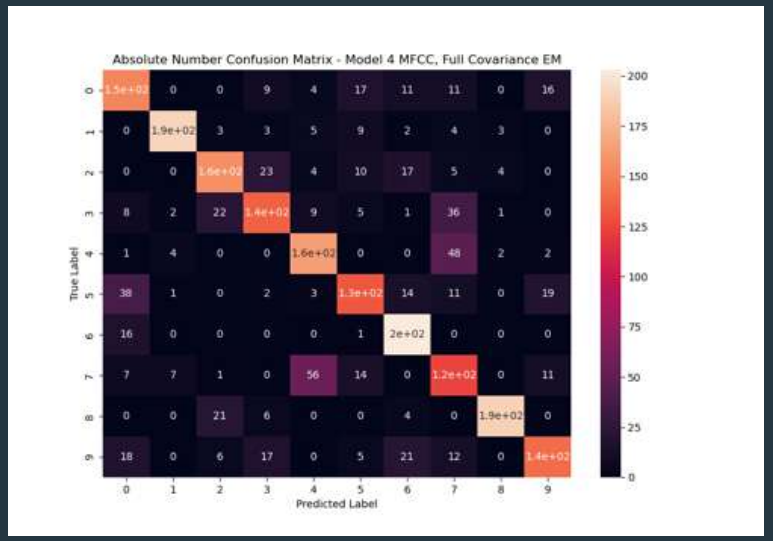
Model 2 – EM - 4 MFCCs with Full Covariance

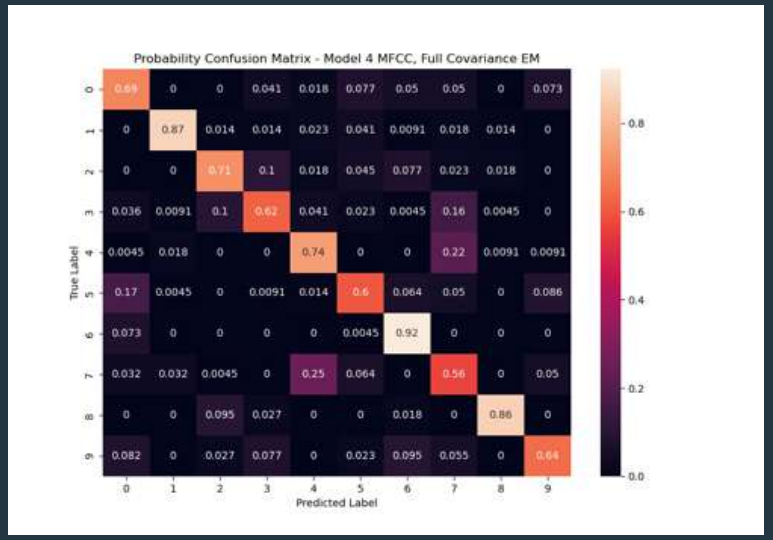
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \pi_{m,d} p(x_n|\Delta_{m,d=full})$$

This model was worse than the previous one for digit 0, 1, 6, 7 and 9, and tied with digit 2 and 4. The digits which improved were digits 3, 4 and 8. These digits also improved between the 13 dimensional kmeans and expectation maximisation.

Similarly to the previous model, digit 8 did not have many digits misclassified as digit 8, but still did not have a high accuracy of being classified correctly when the true value was 8. Although the model for 8 improved between the Kmeans and the EM versions of these models, there were more misclassifications of other numbers as 8. This may have been because the models of most of the other numbers decreased in accuracy leading to less of the other numbers being classified correctly. Alternatively, since it was hypothesised that the Kmeans model was too rigid, perhaps the more flexible EM model had larger covariances which let more 8's fit the criteria, but also more of other numbers as well.





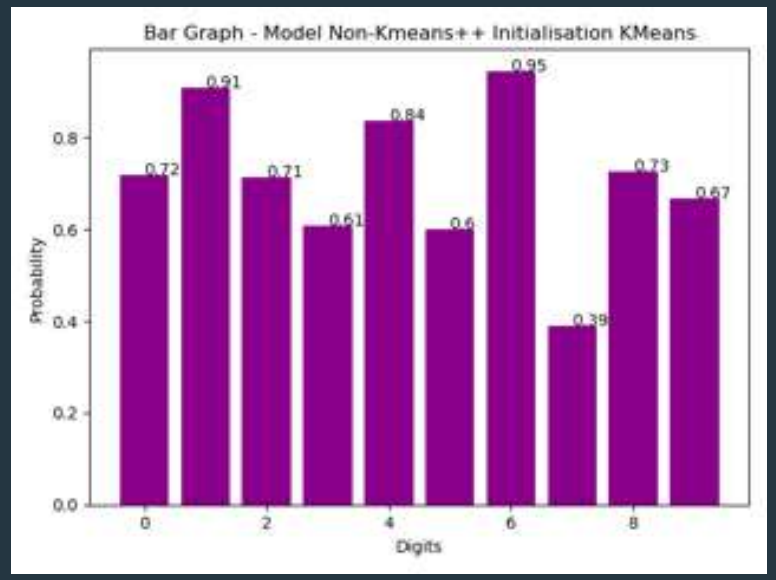


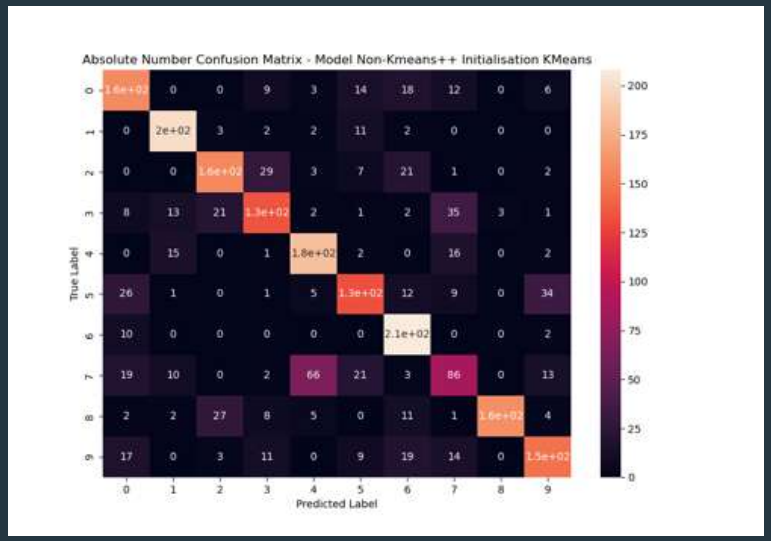
Model 3 – Kmeans - 4 MFCCs , Non-Kmeans++ Initialisation

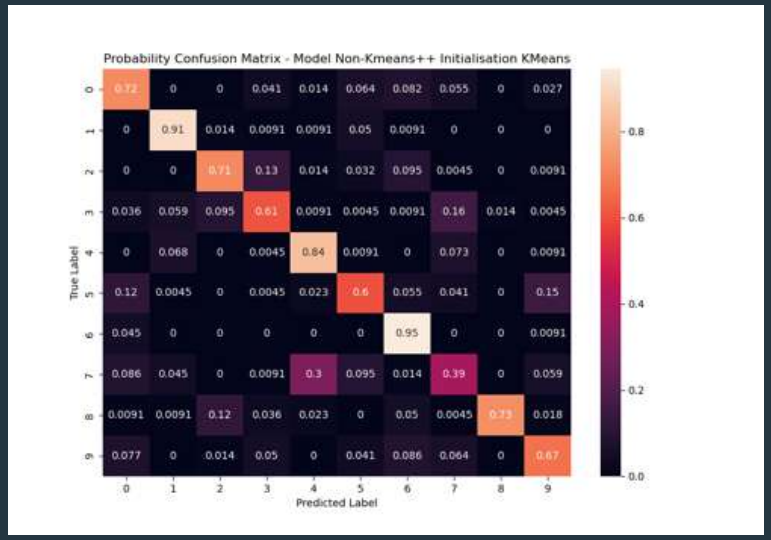
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \pi_{m,d} p(x_n|\Delta_{m=given,d=full})$$

This was a considerable worse model. Unfortunately the version of python this experiment was run on did not have the capability of doing 'random_from_data' for a Gaussian Mixture so there is only the kMeans version of this model. The initialisation points were chosen but dividing the data for one utterance into n segments, where n is the number of clusters. The first entry in each of those segments was chosen as a starting point. If the phonemes were all the same length this would have chosen one initial cluster centre in each phoneme.

This model did not improve from the 13 dimensional clusters, but did improve or was very close to both Model 2s on every number except 7. The initialisation chosen for seven may have caused it to find a local rather than global maximum for the cluster centres. Because the phoneme for 7 did not have much variation, the change of cluster centre may have accidentally caused all the clusters to appear the same rather than demonstrating the variation.







Model 4 – Kmeans - Including Transition Clusters

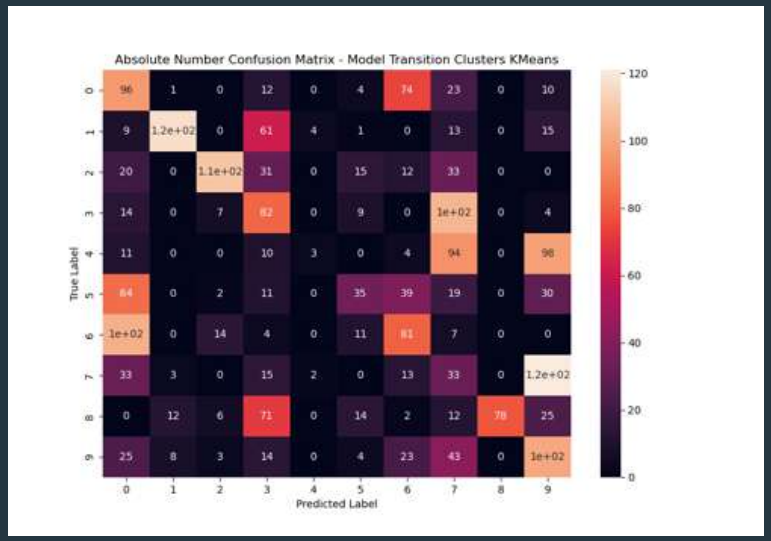
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{2*M+1} \pi_{m,d} p(x_n|\Delta_{m,d=full})$$

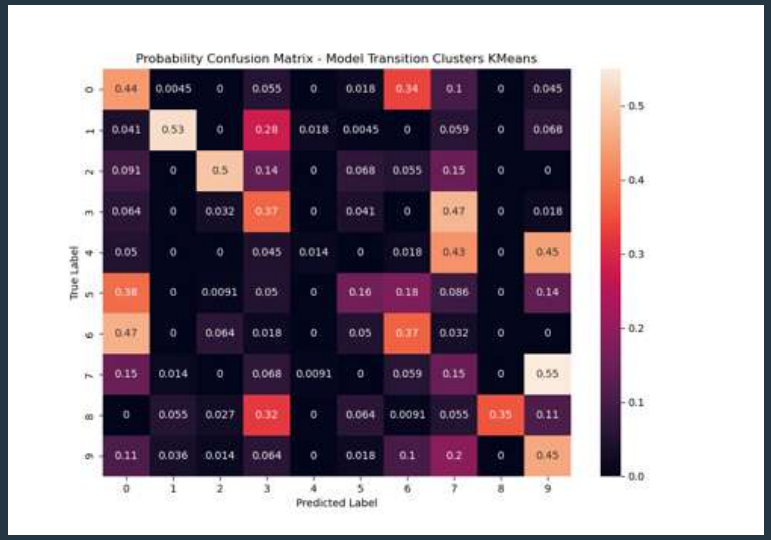
This tapestry of probabilities highlights that adding extra clusters did not have the intended effect of compensating for the transition periods, but in fact created enough clusters that any number could be classified as any other number.

The highest probability was 53% for digit 1 and the lowest probability was 1% for digit 4. Amusingly, a large proportion of the numbers were classified as a 7, which was the inverse of the problem from a previous model.

It can be hypothesised that since 7 had three clusters which spanned the relatively large range of the minimum MFCC = -2 and the maximum MFCC = 4, the five clusters produced in this model were all very different and spanned a large covariance.

4 has many clusters and has MFCCs that span -2 to 4. Therefore the clusters may have been very specific and invariable.



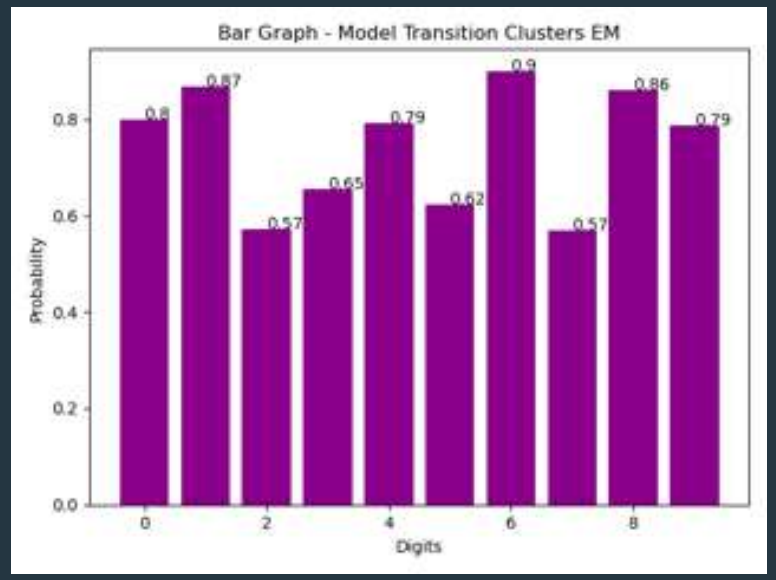


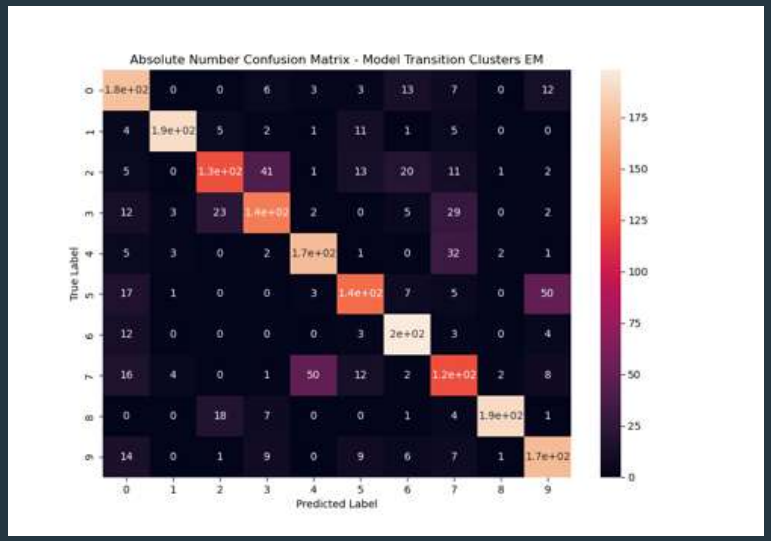
Model 4 – EM - Including Transition Clusters

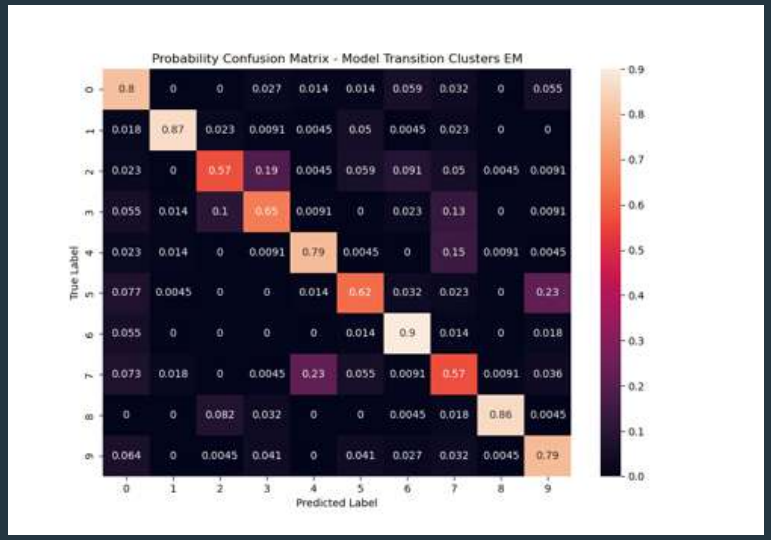
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{2*M+1} \pi_{m,d} p(x_n|\Delta_{m,d=full})$$

With a huge improvement over the previous version of Model 4, this model has a maximum accuracy of 90% and a minimum accuracy of 57%.

This model still has quite a poor 7 and 5 categorisation with accuracies of 57% and 62% respectively, but the accuracy of 4 was 79%, which jumped up from 1% in the previous model. As was hypothesised in the last model, since kmeans produces 'black and white' cluster assignments the model may have been very invariable. Therefore with the addition of the pi value in the expectation maximisation model the probabilities may have all compensated for the rigidity of having extra clusters. This is because two clusters next to each other would have had a shared probability of the data in between, which a test dataset could have fitted into rather than a blatant classification of whether the data was in or out of the cluster. This is an excellent example of bias dominating the model error.







Model 5 – Kmeans - 4 MFCCs with Diagonal Covariance

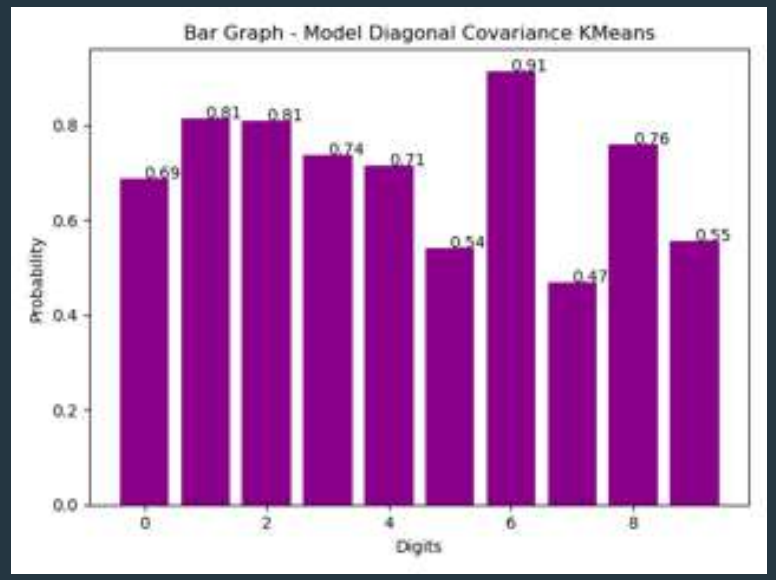
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \pi_{m,d} p(x_n|\Delta_{m,d=diag})$$

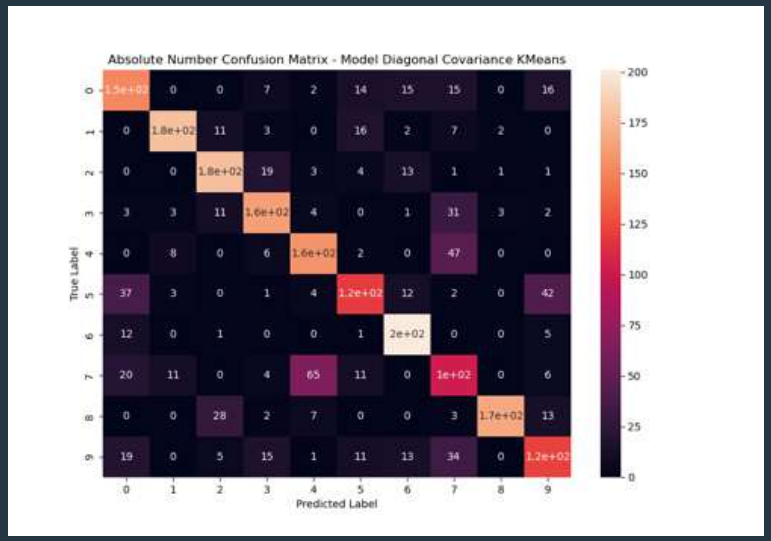
Considering how rigid this model is, this model has accuracies that are comparable to the full covariance matrix with 4 MFCCs.

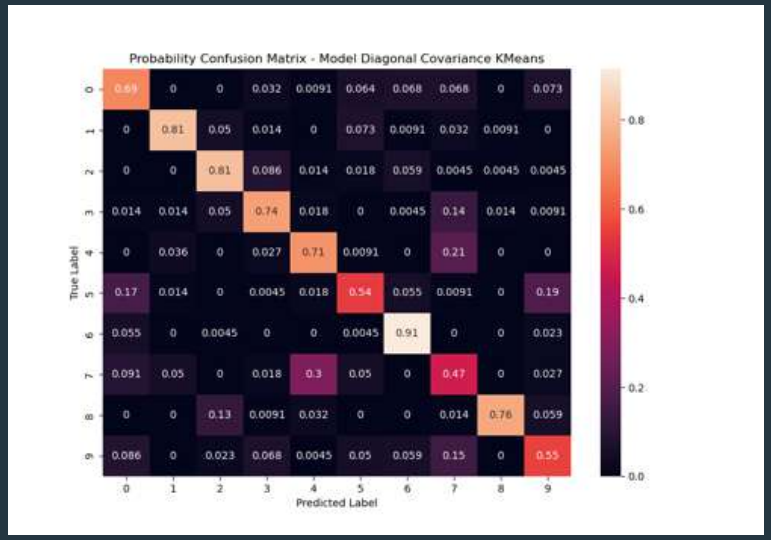
Digit 6 once again stood out as the highest accuracy model with an accuracy of 91% and digit 7 had an accuracy of 47%.

Surprisingly, digit 2 and 3 actually improved by 10% and 16% respectively on the full covariance model. This may have been because the training dataset was a poor representation of the way 2 was uttered and it was better to have a broader cluster to fit the different sounds.

The digits with the largest drop between the full covariance and the diagonal covariance were 7 and 9 with a 14% drop. This indicates the MFCCs were very dependent on each other, which supports the hypothesis that more MFCCs of 7 would have improved the model.







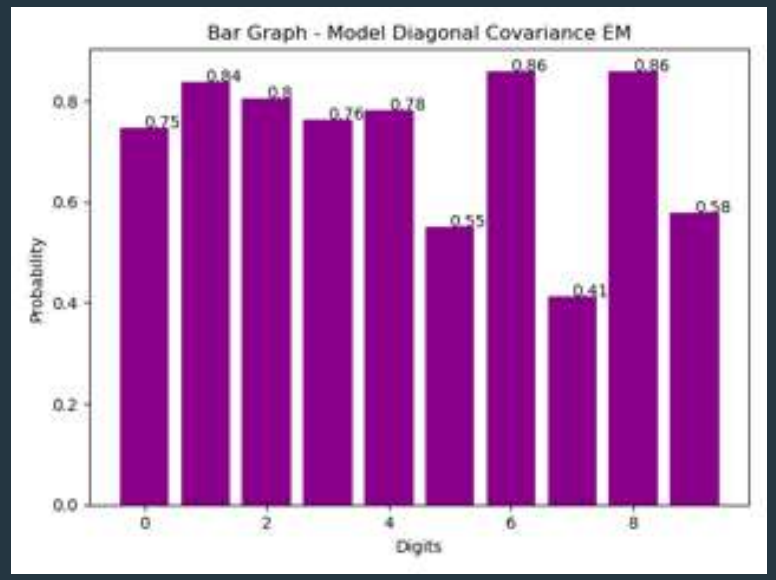
Model 5 – EM - 4 MFCCs with Diagonal Covariance

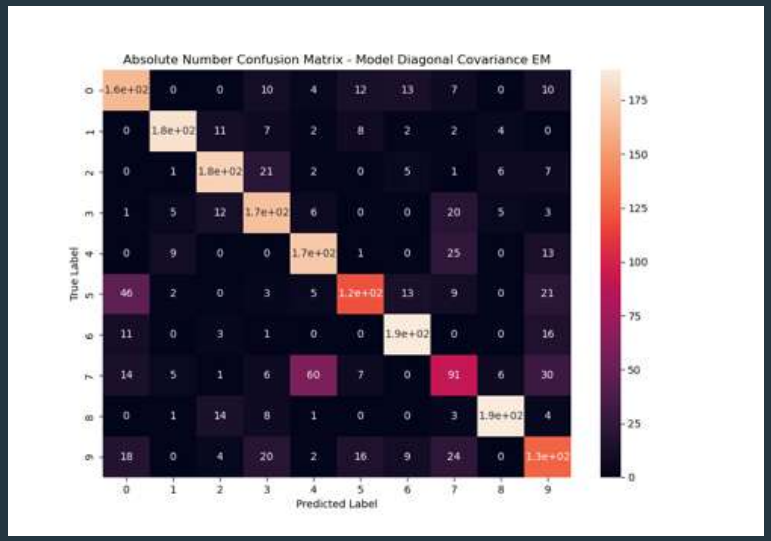
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \pi_{m,d} p(x_n|\Delta_{m,d=diag})$$

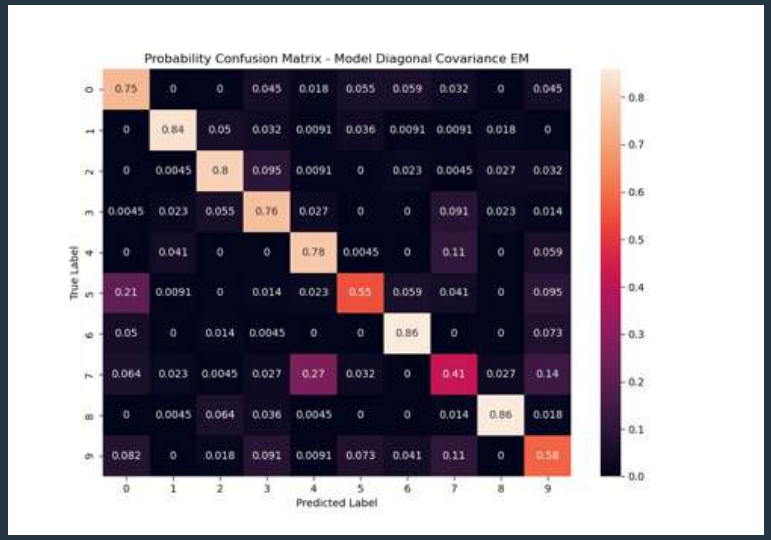
The digits 0, 2, 3, 4 and 8 all improved on their full covariance counterpart and digits 1, 5, and 6 had very similar accuracies. As mentioned in the previous slide, the MFCCs for these components must be quite independent. This leaves room for possible experimentation with reducing the number of MFCCs in the model or making the covariance spherical.

On the other hand, the models for digit 7 and 9 decreased by approximately 10% in accuracy. This further supports the hypothesis that their MFCCs are very dependent. The MLE for the model of digit 7 was almost 4.

Digit 9's accuracy increased from the kmeans version, which is surprising, because it is different to the trend seen in model 1 and 2. It is similar to the trend in model 3, however, that was not an isolated improvement in Digit 9, but one for all digits, so it is difficult to make a well supported comparison.







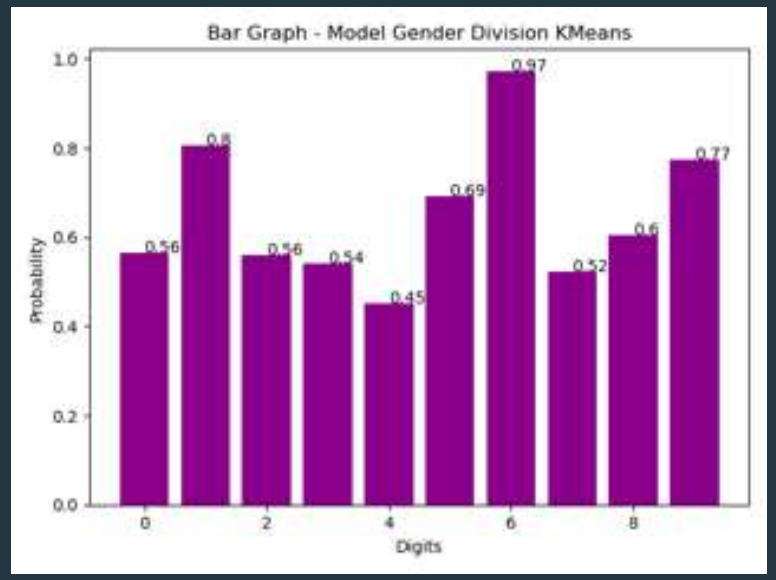
Model 6 – Kmeans - Separation by Gender

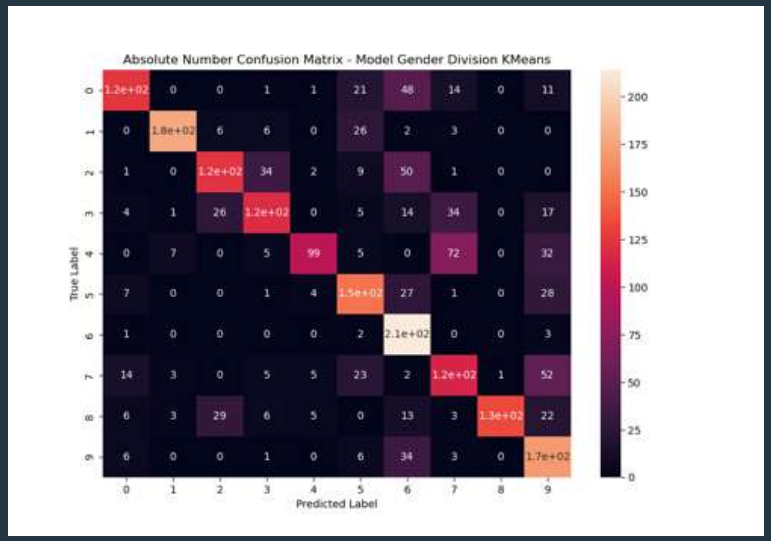
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \Sigma_{g=female}^{male} \pi_{m,d} p(x_n|\Delta_{m,d=full}|\, gender_g)$$

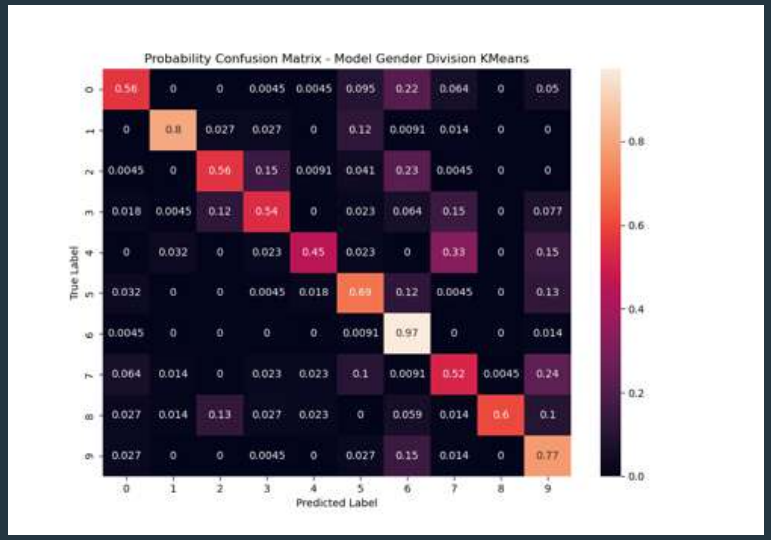
This model was created by separating the training data into male and female as well as by digits, and then summing the probability of the test data fitting in both the male and the female model. This model was surprisingly worse than the full covariance model that wasn't separated by gender. The standout digits were 1 with an accuracy of 80%, 6 with an accuracy of 97% and 9 with an accuracy of 77%.

Digit 6 appears to be the most reliable for creating a model that matches its phonemes.

Digit 9 was an interesting case because it is generally modelled with a much lower accuracy than most of the other digits, but was a stand out digit here. This could have been because the utterance of digit 9 sounded like a different number when spoken at a lower or higher pitch Therefore when the gender was marginalised over the different models the utterances of 8 matched the gender specific model more closely than it did another number which may have sounded like a lower or higher pitch version of 8.







Model 6 – EM - Separation by Gender

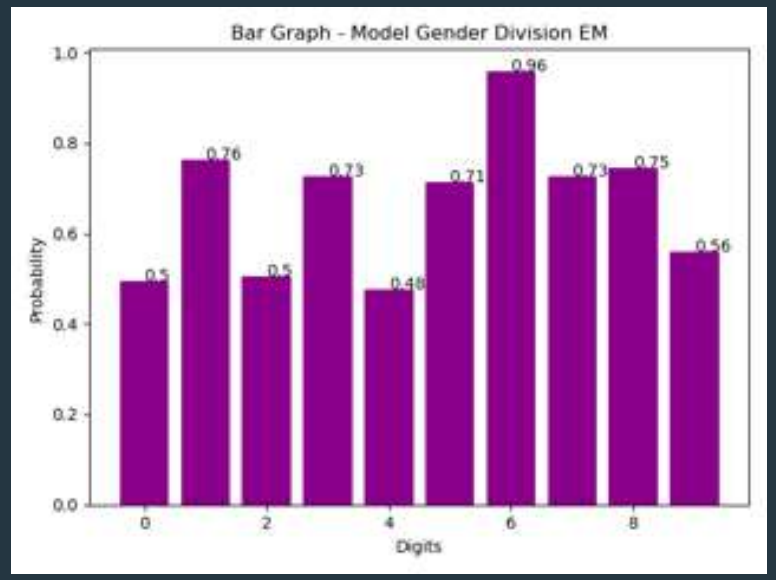
$$p(X|\Delta_d, \Pi_d) = \Pi_{n=1}^{220} \Sigma_{m=1}^{M} \Sigma_{g=female}^{male} \pi_{m,d} p(x_n|\Delta_{m,d=full}|\, gender_g)$$

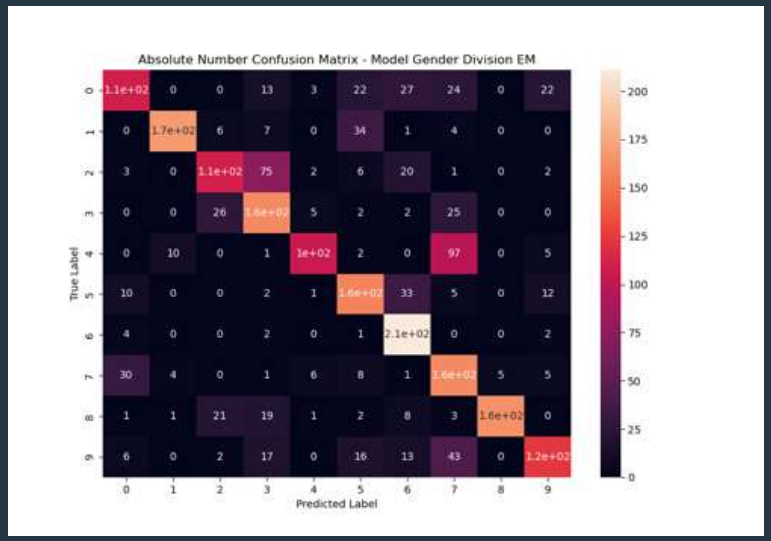
The EM separation by gender was considerably better than the kmeans separation by gender. A standout digit was 7. Its accuracy in this model was 73% and in the best model, the 13 dimensional, full covariance, not gender separated model was merely 75%. This may have been because the digit 7 sounds very different when uttered by higher and lower pitched voices, hence making the cluster very large and difficult to fit testing data to.

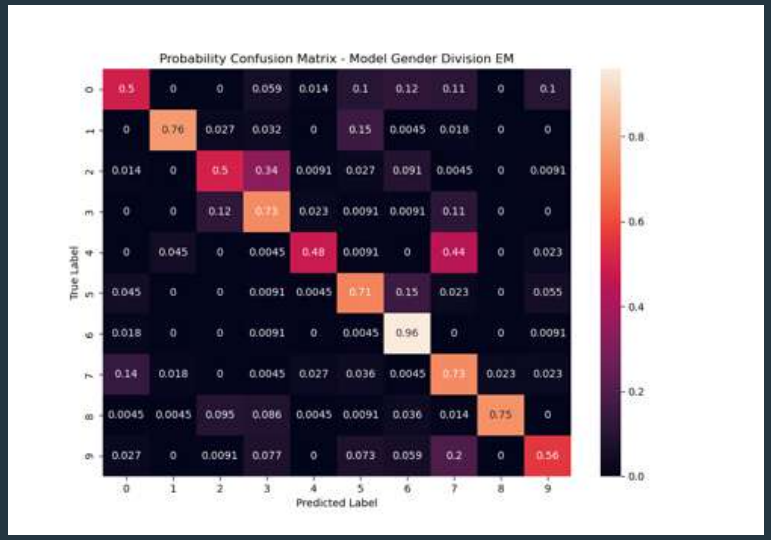
Between the kmeans Model 6 and the EM Model 6, digit , 7 and 8 improved, but digit 9 had a lower accuracy with this model.

Digit 3 and 8 may have improved between the kmeans and the EM because of the same reasoning they improved in previous models; they may have improved because the model is generally more flexible and accounts better for the transition regions between clusters/phonemes.

Digit 9 may need a more rigid model that fits the data more closely.







Overall Model Comparison

In this experiment a model was run with all the cepstral coefficients and four of the cepstral coefficients. Since the four dimensional models were much less accurate than the 13 dimensional models for future work it would be important to find the point where there is diminishing returns by adding an extra coefficient. Due to the shapes of the MFCC graphs this will probably be 7. Various models were run and different models suited different digits better.

In Table 10, it can be observed that the inclusion of more MFCCs results in a better model, which can be seen by the dominance of Model 1. In the third column, however, there are the promising models which all had features which performed well despite having few MFCCs. Most interestingly is Digit 8, which appeared to have success with a rigid model and performed well under full covariance, increasing the cluster numbers and separating by gender, but only when run with an EM algorithm.

These results clearly demonstrate the bias variance trade-off, because some digits were more accurately categorised with a rigid model whereas some where categorised well with a flexible model. This can certainly be linked to the graphs of the MFCCs and show that Digit 7, for example had minimum variation in its phonemes, and thus performed well with a flexible model that could pick up on that small variation. Alternatively digit 6 and 7 performed much better with a rigid model, because they had very distinctive phoneme patterns. In fact, digit 6 performed well under all the models, but excelled under the kmeans models: a model which is much more rigid than the EM model.

Digit	Best Model	Promising Model
0	Model 1 EM	Model 4 EM
1	Model 1 EM	Model 3
2	Model 1 EM	Model 5 KM
3	Model 1 EM	Model 5 EM
4	Model 1 EM	Model 3
5	Model 1 EM	Model 6 EM
6	Model 1 KM	Model 6 KM
7	Model 1 EM	Model 6 EM
8	Model 1 EM	Model 2, 4 and 5 EM
9	Model 1 KM	Model 4 EM and Model 6 KM

Table 1: Model Comparison per Digit

Conclusions

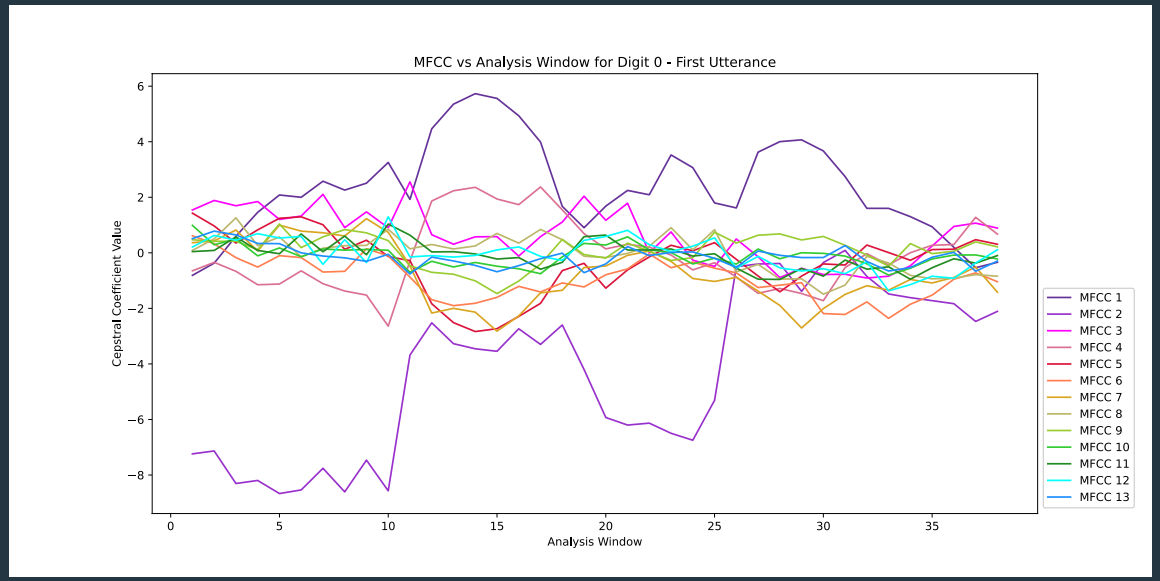
The various models presented different advantages to different digits. This was dependent on the phoneme structure of each digit and whether it had large variation, which needed a rigid model so as to not have clusters that were too encompassing or slight phoneme variation, which had more success with a flexible model.

The most important modelling choices were whether to split by gender and marginalise, how many clusters to include and covariance. When the number of clusters was increased, the model accuracy declined severely, which helped provide another example of bias-variance trade-off. There was minimal distinction between the kmeans vs the EM algorithms, although, from a coding perspective EM was much easier to implement, and slightly more effective.

The most generally effective model was Model 1 EM. It was very flexible and hence was able to match the phonemes of most of the digits. The EM algorithm did a better job than kmeans at accounting for the transition between clusters. The percent accuracy was greater than 90% for almost all the digits, which is a very promising result.

Future modelling could aim to start combining models to see if a more flexible parameter, such as gender separation, could counteract the rigidity from a parameter such as diagonal covariance and therefore create a simple, processing-power-efficient program that had a very high accuracy result. Experimenting with marginalising over different variables such as allophones and taking the MLE of these could also be explored.

Importantly ,future modelling should be kept the same by only changing one latent variable between models, to ensure it is possible to tell which changes were affecting the model in both positive and negative ways.



Research References - Page 1/2

- → Alam, Md.J., Kenny, P. and O'Shaughnessy, D. (2012). Low-variance Multitaper Mel-frequency Cepstral Coefficient Features for Speech and Speaker Recognition Systems. *Cognitive Computation*, 5(4), pp.533–544. doi:10.1007/s12559-012-9197-5.
- → Badshah, A.M., Ahmad, J., Rahim, N. and Baik, S.W. (2017). Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network. *2017 International Conference on Platform Technology and Service (PlatCon)*. doi:10.1109/platcon.2017.7883728.
- → Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge ; New York: Cambridge University Press.
- → Binder, M.D., Hirokawa, N. and Windhorst, U. (2009a). Allophone. *Encyclopedia of Neuroscience*, pp.79–79. doi:10.1007/978-3-540-29678-2_159.
- → Binder, M.D., Hirokawa, N. and Windhorst, U. (2009b). Phoneme. *Encyclopedia of Neuroscience*, pp.3145–3145. doi:10.1007/978-3-540-29678-2_4553.
- → Biswas, A., Sahu, P.K. and Chandra, M. (2015). Multiple camera in car audio–visual speech recognition using phonetic and visemic information. *Computers & Electrical Engineering*, 47, pp.35–50. doi:10.1016/j.compeleceng.2015.08.009.
- → Cambridge University Press (2009). *Cambridge Academic Content Dictionary*. New York: Cambridge University Press.
- → Czyzewski, A., Kostek, B., Bratoszewski, P., Kotus, J. and Szykulski, M. (2017). An audio-visual corpus for multimodal automatic speech recognition. *Journal of Intelligent Information Systems*, 49(2), pp.167–192. doi:10.1007/s10844-016-0438-z.
- → Goel, N. and Singh, K. (2011). Analysis of Dirichlet, Generalized Hamming and Triangular window functions in the linear canonical transform domain. *Signal, Image and Video Processing*, 7(5), pp.911–923. doi:10.1007/s11760-011-0280-2.
- → Jung, Y.G., Kang, M.S. and Heo, J. (2014). Clustering performance comparison usingK-means and expectation maximization algorithms. *Biotechnology & Biotechnological Equipment*, [online] 28(sup1), pp.S44–S48. doi:10.1080/13102818.2014.949045.

Research References – Page 2/2

→ Korvel, G., Kurowski, A., Kostek, B. and Czyzewski, A. (2018). Speech Analytics Based on Machine Learning. *Machine Learning Paradigms*, 149, pp.129–157. doi:10.1007/978-3-319-94030-4_6.

→ Li, S. and Jain, A. (2009). Maximum Likelihood Estimation. *Encyclopedia of Biometrics*, pp.964–964. doi:10.1007/978-0-387-73003-5_598.

→ Nelles, O. (2020). Model Complexity Optimization. *Nonlinear System Identification*, pp.175–231. doi:10.1007/978-3-030-47439-3_7.

→ nlp.stanford.edu. (2009). *K-means*. [online] Available at: https://nlp.stanford.edu/IR-book/html/htmledition/k-means-1.html [Accessed 12 Dec. 2022].

→ Rosner, A. and Kostek, B. (2017). Automatic music genre classification based on musical instrument track separation. *Journal of Intelligent Information Systems*, 50(2), pp.363–384. doi:10.1007/s10844-017-0464-5.

→ Rudd, D.H., Huo, H. and Xu, G. (2022). Leveraged Mel Spectrograms Using Harmonic and Percussive Components in Speech Emotion Recognition. *Advances in Knowledge Discovery and Data Mining*, pp.392–404. doi:10.1007/978-3-031-05936-0_31.

→ support.apple.com. (n.d.). *What can I ask Siri? - Official Apple Support*. [online] Available at: https://support.apple.com/siri [Accessed 10 Dec. 2022].

→ Tantum, S. (2022). Course Project: Recognizing Spoken Digits.

→ UCI (n.d.). *UCI Machine Learning Repository: Spoken Arabic Digit Data Set*. [online] archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Spoken+Arabic+Digit [Accessed 10 Dec. 2022].

Coding Package References

Harris, C.R. et al., 2020. Array programming with NumPy. *Nature*, 585, pp.357–362.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. *Computing in science & engineering*, 9(3), pp.90–95.

McKinney, W. & others, 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*. pp. 51–56.

Pedregosa, F. et al., 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct), pp.2825–2830.

Virtanen, P. et al., 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, pp.261–272.

Waskom, M. et al., 2017. *mwaskom/seaborn: v0.8.1 (September 2017)*, Zenodo. Available at: https://doi.org/10.5281/zenodo.883859.

Collaborations

Who did you share and debate ideas with while working on this project?

Rebecca Edelman and I worked next to each other for some of the project. This involved debating our approaches for the different sections.

Who did you share code with while working on this project?

I did not share lines of code with anyone.

Who did you compare results with while working on this project?

Rebecca Edelman and I compared plots and confusion matrices to ensure we were both making similar progress towards the expected outcomes

Who did you help overcome an obstacle and vice versa while working on this project?

Rebecca Edelman and I would help each other whenever we ran upon an issue. This involved trouble shooting each others' code