Online Clustering Project Final Report

11/8/2024

Kennan Wu

Table of Contents

Overview

The purpose of this project is to be able to cluster 3 different audio signals in an online manner using ART2 clustering. For non PCA, the accuracy lies around 95%-99%, and for PCA 97%-100%.

Important things to note

1. This code was written on a windows machine.
2. This code is optimized to run on the GPU using PyTorch, however if no GPU is detected, then it will run on the CPU, however the GPU greatly increases the processing speed.
3. To get the most accurate results for no PCA, run the smoothing.ipynb file.
4. To get the most accuracy results for PCA, run the no_smoothing.ipynb file.
5. .csv time series information is found in the output folder (this will populate itself after running the code)
6. For the whole dataset, look at all_features.csv for all possible features, and selected_features.csv, for features I deemed the most important
7. For each individual time series, combined_{index}.csv, mfccs_{index}.csv, and melspec_{index}.csv contains the data points..

Getting the code to run

1. cd into the codebase root
2. if you are on a windows machine, run `` `conda env create --file=environments\environment-windows.yaml -n online-clustering-env` ``
3. if you are on macOS, run `` `conda env create --file=environments\environment-macos.yaml -n online-clustering-env` ``
4. In the smoothing.ipynb file, select online-clustering-env as your kernel, and run the file.
    a. The output in cell with the markdown title “Optimal Clustering Results for No PCA"
5. Do the same for no_smoothing.ipynb.
    a. The output cell with the markdown title “Optimal Clustering Results for PCA” is the most optimal results for PCA

General Codebase overview

The codebase follows a fairly simple file structure. To maintain readability, hefty modules were written as classes in python files with five main classes. AudioProcessor, FeatureExtractor, DatasetHandler, PCA, and ART2Processor, all of which are in the *modules* folder. The files that contain the optimal solutions, for both non PCA and PCA are in the smoothing.ipynb file and no_smoothing.ipynb file respectively.

AudioProcessor

This class loads audios using torchaudio. For each waveform extracted from the audio, they are split into frames that overlap based on the specified hop_ratio and frame_size.

FeatureExtractor

This module handles extracting the MFCC and MFSC features from each frame, storing the results in a tensor.

DatasetHandler

This class will generate the csv files that will be used by the clustering model. It will generate all_features.csv, which is a csv file that contains all of the frames with both MFCC and MFSC values. Selected_features.csv, which contains all frames with specified MFCC and MFSC features. This allows for time series to be created with only MFCC, only MFSC, or combined features.

This class also handles creating the time series. It will extract frames within a 15-30 second window from each class and combine them sequentially to make a time series.

ART2Clusterer

This module contains two classes, a clustering model with smoothing, and one without. For the clustering model without, a single point is taken in at a time. The euclidean distance squared is then calculated between this point and existing cluster means. If the nearest mean to this point is larger than the vigilance parameter, a new cluster is created. Otherwise, this point is placed in the winner mean cluster.

The smoothing module works the same as the normal clustering model, except now there is a buffer. When data points are read, they are placed in the buffer until the buffer reaches a max capacity. Once the buffer is full, the distance from the mean of the buffer to the mean of other clusters are compared, and the same vigilance test is done with this distance. The model will then place all data points in the winner cluster.

PCA

This module standardizes the data using sklearn StandardScalar, then computes and projects the PCA data using sklearn PCA.

No PCA Results (Part 1)

Overview

To start, here are the most optimal results from clustering only MFCC features, only MFSC features, and combining the two. All of the time series are randomly generated in the fashion as described above. Each figure shows an accuracy, confusion matrix, and Ground Truth (Blue) vs predicted labels (red) graphs. In the predicted labels, datapoints that are plotted in the class -1 means the clustering algorithm plotted them in a class that is not part of the ground truth classes.

```
Clustering time series 0 - mfccs features
Accuracy: 0.9864406779661017
Confusion Matrix:
[[304   0   0]
 [  3 310   1]
 [  3   0 259]]
```

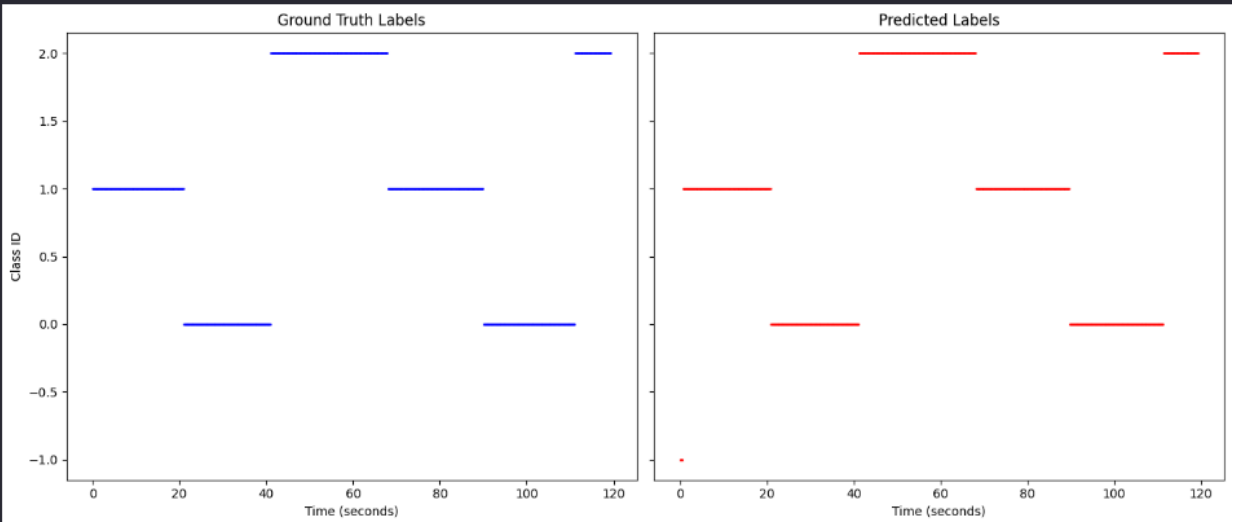

Figure 1.
Clustering results for time series 0 with only MFCC features, smoothing, and normalization

```
Clustering time series 0 - melspec features
Accuracy: 0.6892655367231638
Confusion Matrix:
[[297   0   0]
 [  1 313   0]
 [272   2   0]]
```

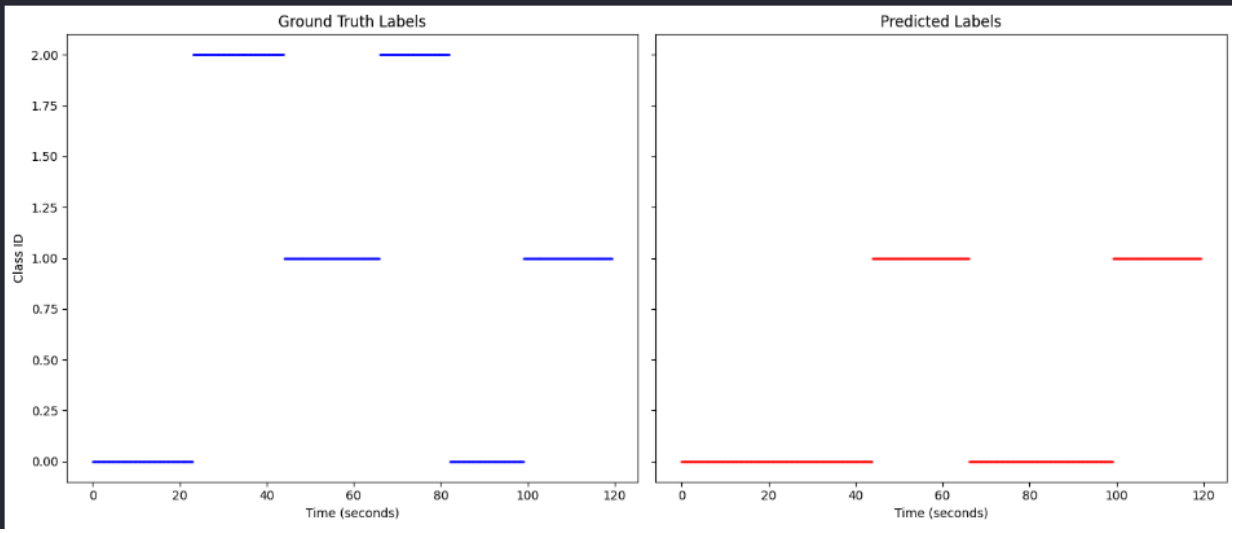

Figure 2.
Clustering results for time series 0 with only MFSC features, smoothing, and normalization

```
Clustering time series 0 - combined features
Accuracy: 0.9943502824858758
Confusion Matrix:
[[379   2   0]
 [  1 293   2]
 [  0   0 208]]
```

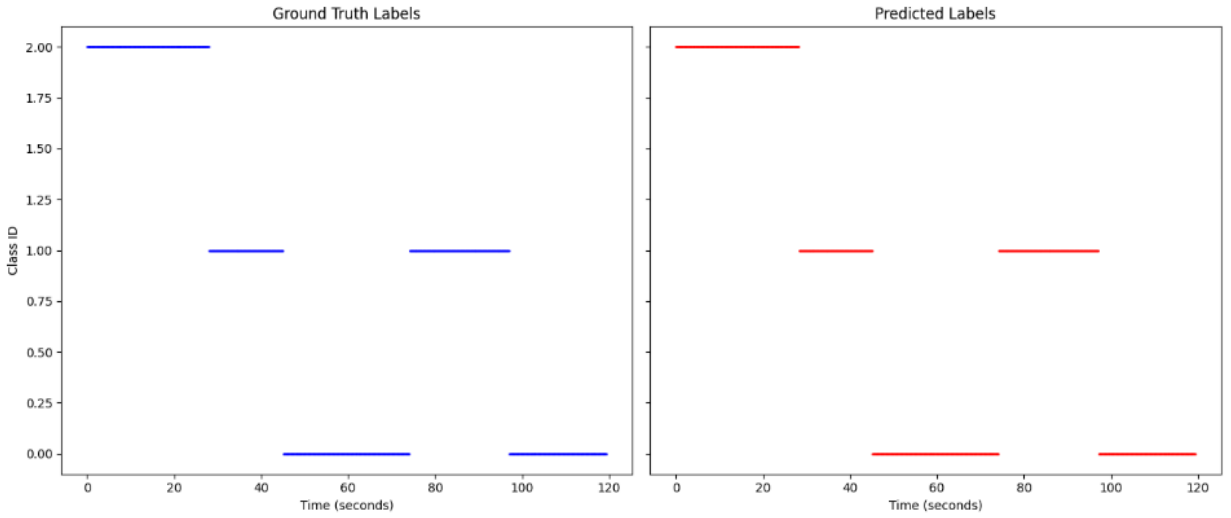

Figure 3.
Clustering results for time series 0 with combined MFCC and MFSC, with smoothing and normalization

```
Clustering time series 1 - mfccs features
Accuracy: 0.9898305084745763
Confusion Matrix:
[[275   0   0]
 [  4 206   0]
 [  1   4 395]]
```

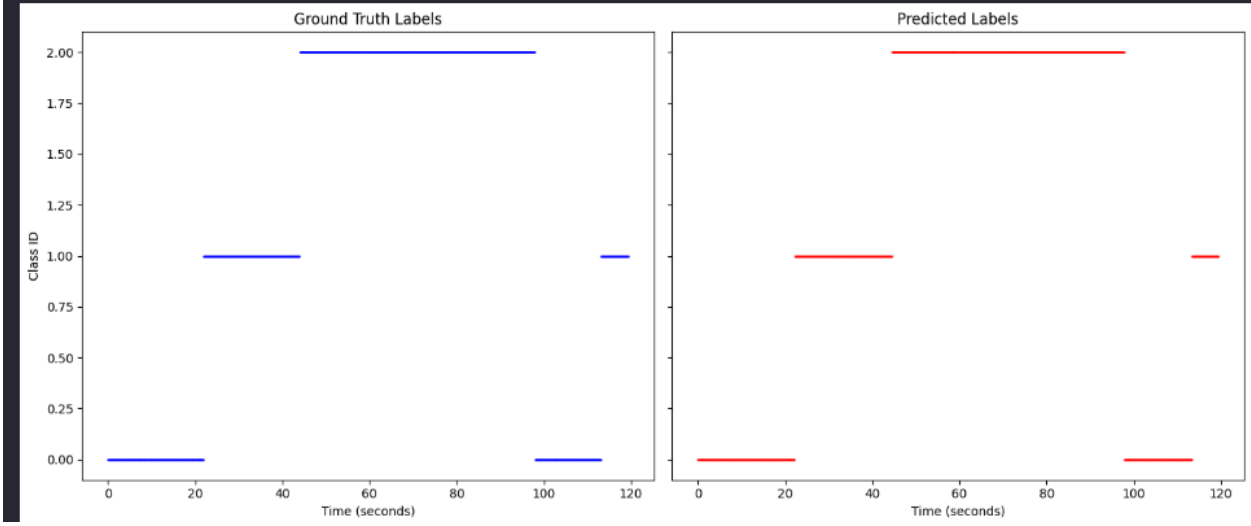

Figure 4.
Clustering results for time series 1 with only MFCC features, smoothing, and normalization

```
Clustering time series 1 - melspec features
Accuracy: 0.7977401129943503
Confusion Matrix:
[[317   0   0]
 [ 10 165   6]
 [148   0 224]]
```

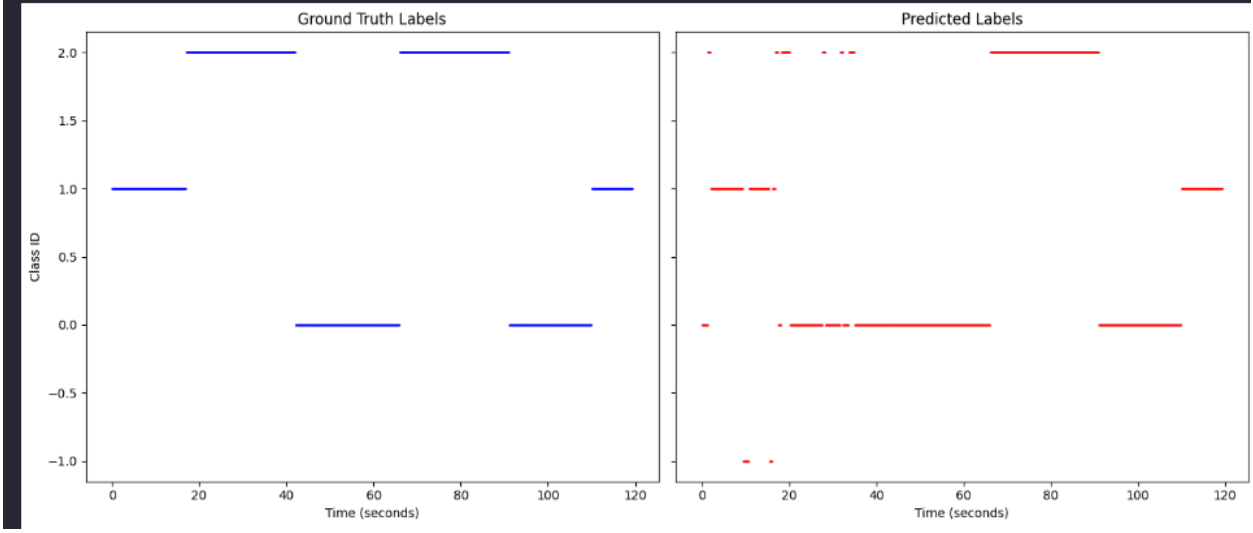

Figure 5.
Clustering results for time series 1 with only MFSC features, smoothing, and normalization

```
Clustering time series 1 - combined features
Accuracy: 0.9932203389830508
Confusion Matrix:
[[328   0   1]
 [  2 323   1]
 [  0   2 228]]
```

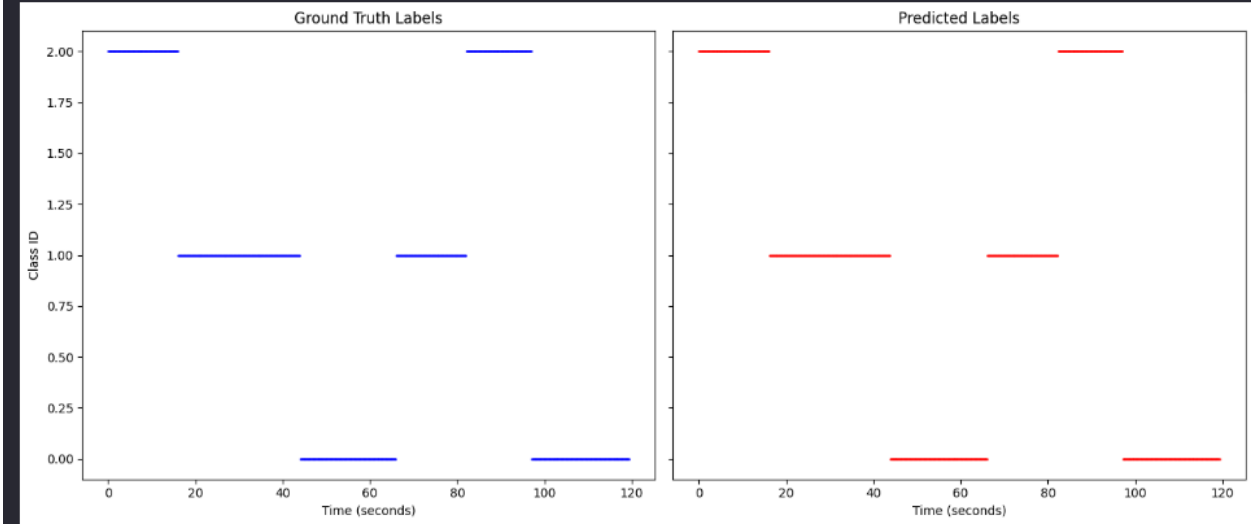

Figure 6.
Clustering results for time series 1 with combined MFCC and MFSC, with smoothing and normalization

```
Clustering time series 2 - mfccs features
Accuracy: 0.7548022598870057
Confusion Matrix:
[[312   2   0]
 [  3 356   5]
 [  0 207   0]]
```

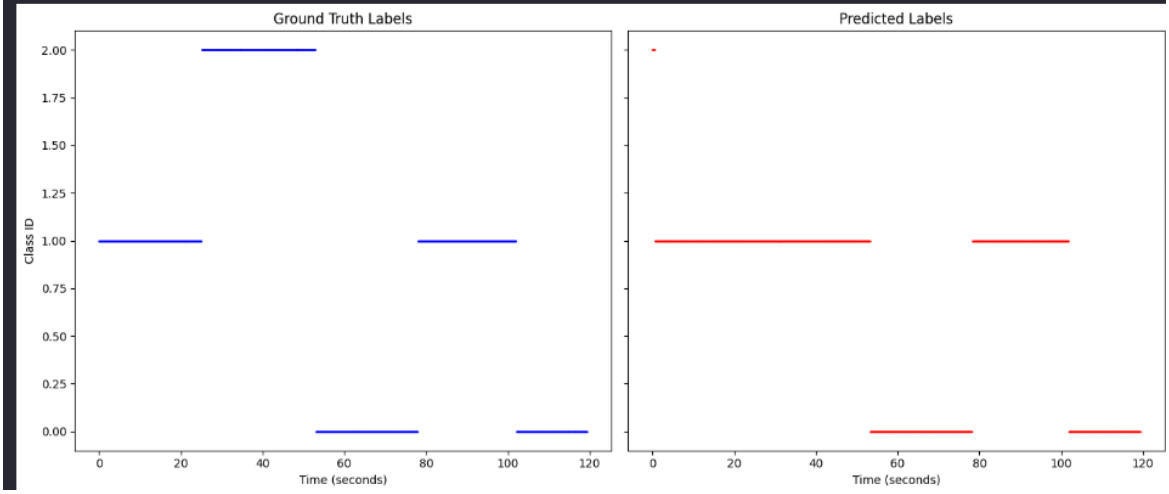

Figure 7.
Clustering results for time series 2 with only MFCC features, smoothing, and normalization

```
Clustering time series 2 - melspec features
Accuracy: 0.7864406779661017
Confusion Matrix:
[[368   2   0]
 [ 11 328   5]
 [156   0   0]]
```

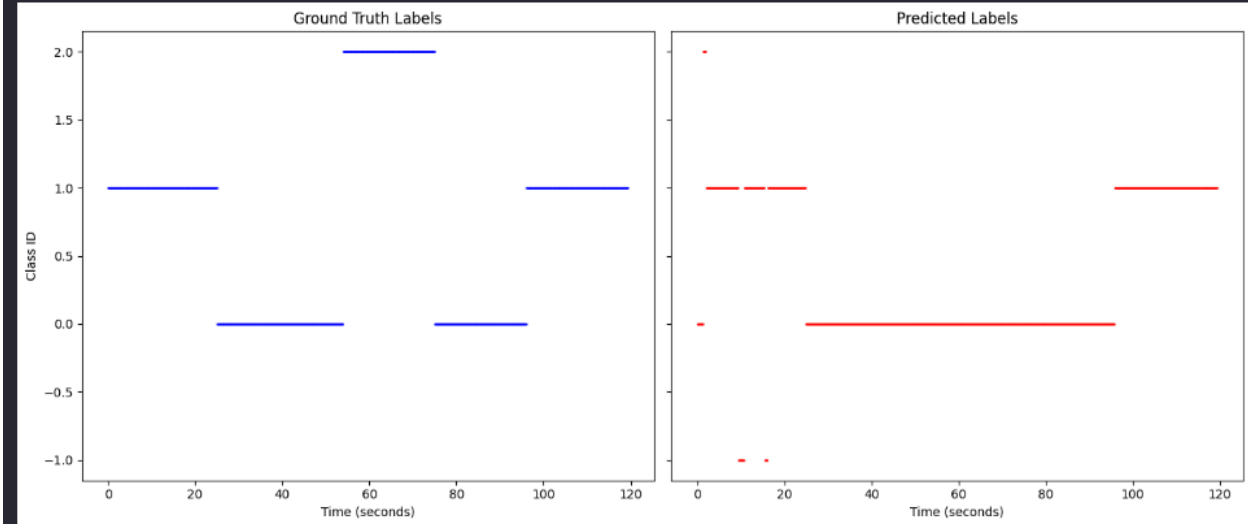

Figure 8.
Clustering results for time series 2 with only MFSC features, smoothing, and normalization

```
Clustering time series 2 - combined features
Accuracy: 0.96045197740113
Confusion Matrix:
[[274   1   2]
 [  1 288   0]
 [  0   1 288]]
```

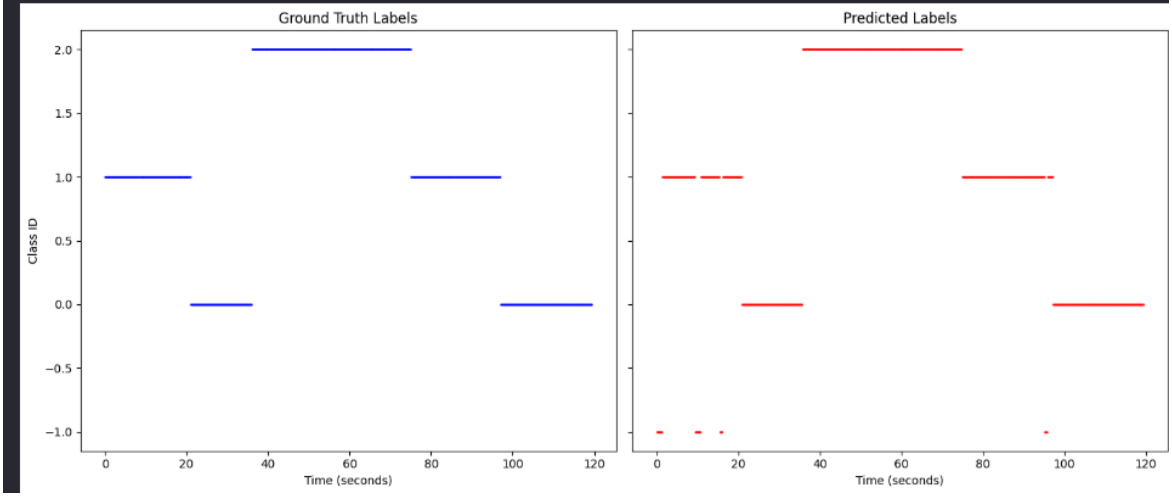

Figure 9.
Clustering results for time series 2 with combined MFCC and MFSC, with smoothing and normalization

```
Clustering time series 3 - mfccs features
Accuracy: 0.703954802259887
Confusion Matrix:
[[337   0   3]
 [  0   0 256]
 [  3   0 286]]
```

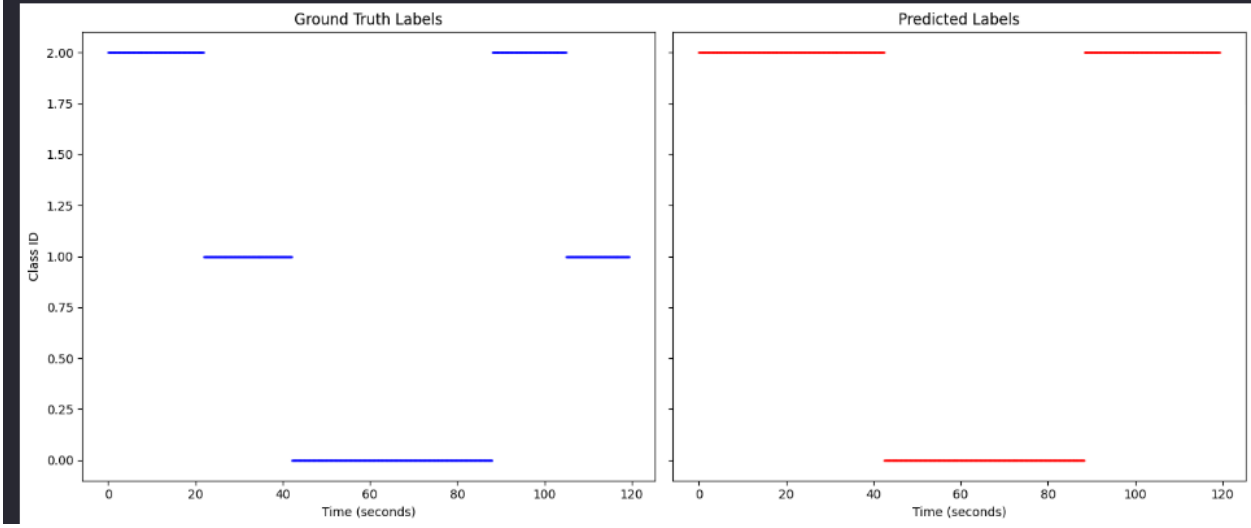

Figure 10.
Clustering results for time series 3 with only MFCC features, smoothing, and normalization

```
Clustering time series 3 - melspec features
Accuracy: 0.7197740112994351
Confusion Matrix:
[[184   1   0]
 [ 10 359   1]
 [236   0  94]]
```

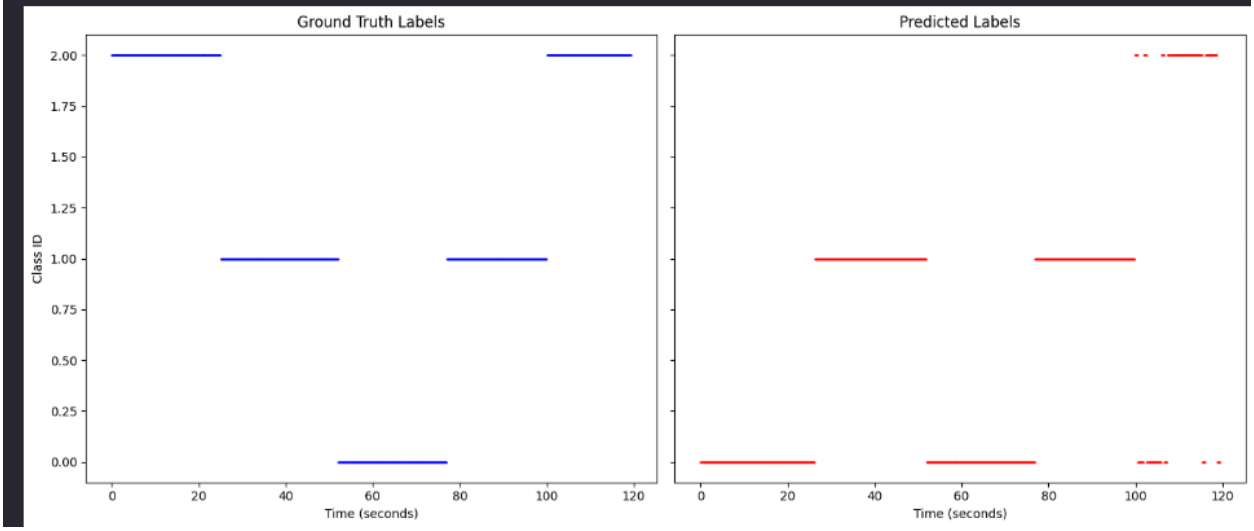

Figure 11.
Clustering results for time series 3 with only MFSC features, smoothing, and normalization

```
Clustering time series 3 - combined features
Accuracy: 0.9954802259887006
Confusion Matrix:
[[208   0   0]
 [  0 333   0]
 [  2   2 340]]
```

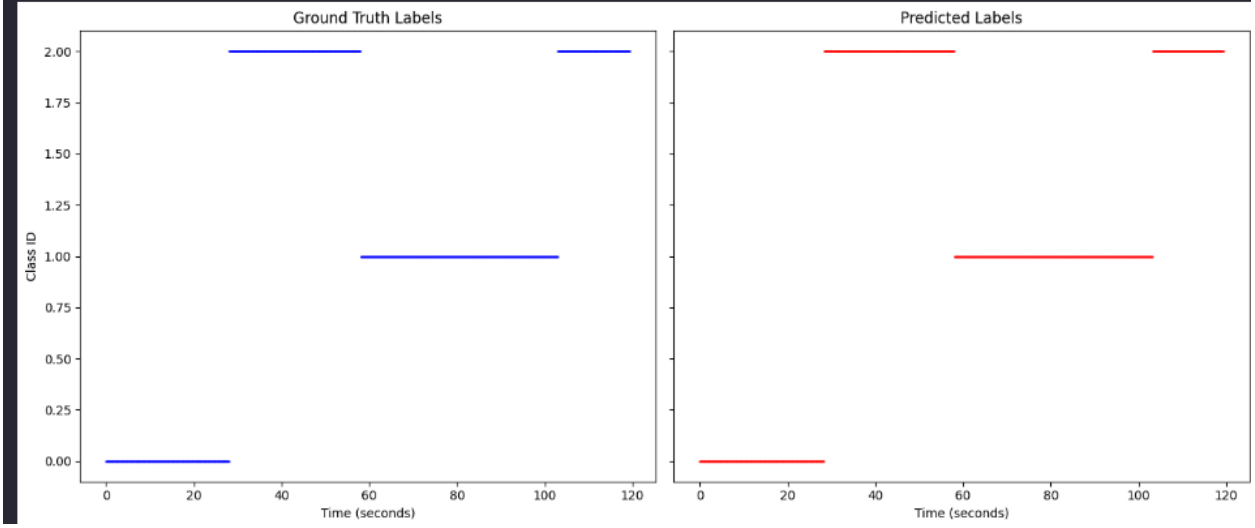

Figure 12.
Clustering results for time series 3 with combined MFCC and MFSC, with smoothing and normalization

```
Clustering time series 4 - mfccs features
Accuracy: 0.8576271186440678
Confusion Matrix:
[[408   0   0]
 [  2  10 124]
 [  0   0 341]]
```

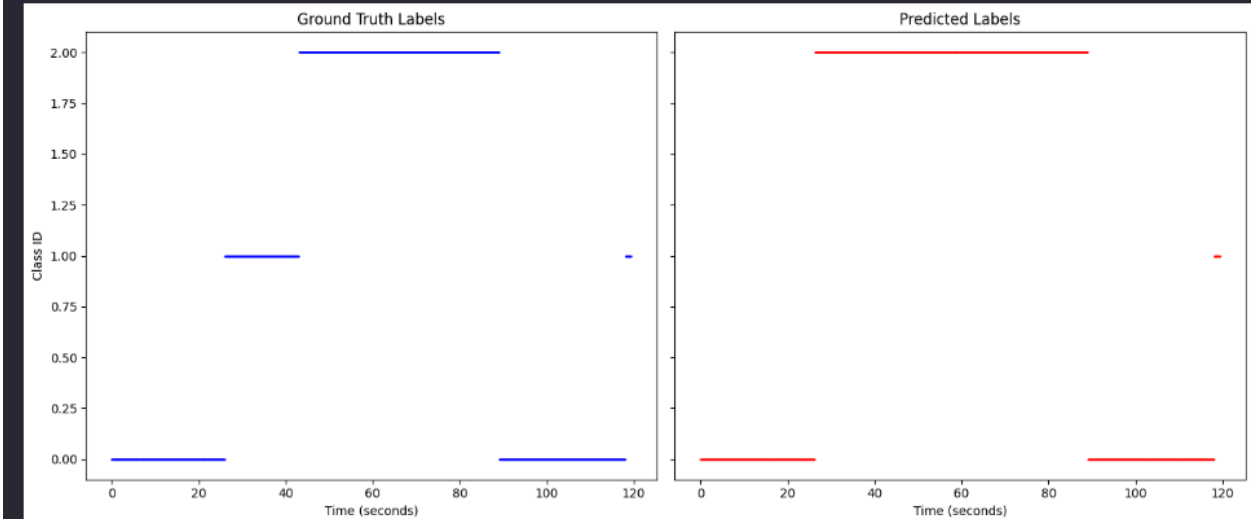

Figure 13.

Clustering results for time series 4 with only MFCC features, smoothing, and normalization

```
Clustering time series 4 - melspec features
Accuracy: 0.6045197740112994
Confusion Matrix:
[[  0   0 345]
 [  0 150   5]
 [  0   0 385]]
```

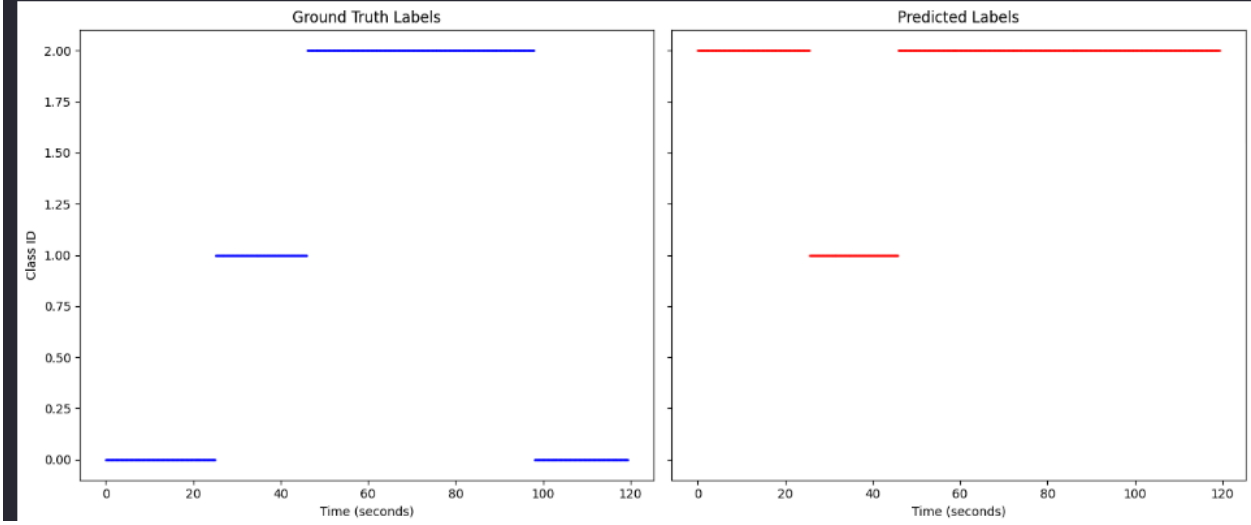

Figure 14.

Clustering results for time series 3 with only MFSC features, smoothing, and normalization

```
Clustering time series 4 - combined features
Accuracy: 0.96045197740113
Confusion Matrix:
[[236   1   0]
 [  2 339   0]
 [  2   0 275]]
```

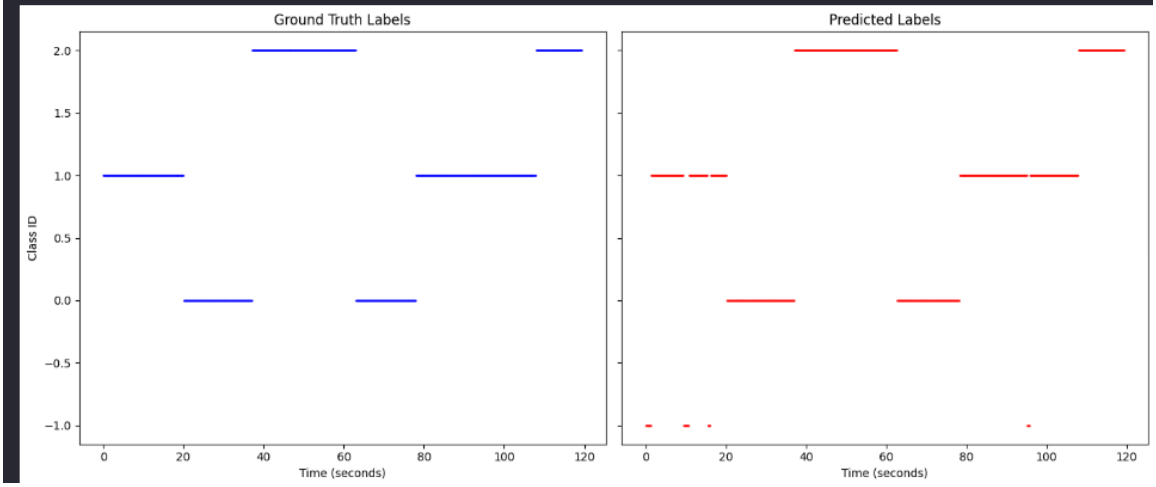

Figure 15.

Clustering results for time series 3 with combined MFCC and MFSC, with smoothing and normalization

The most optimal solution involves combining different parts of the MFCCs and MFSC, smoothing the data, normalizing the data, as well as the following parameters.

1. Frame size (ms): 150
2. Hop ratio: 0.1
3. n_fft: 1024
4. Vigilance (for combined features): 1.4

Using a Squared Distance

As mentioned in the general code overview, the clustering model uses a squared distance to compare to the vigilance parameter. The reason why squared distance is used is because it amplifies larger distances. This means that outliers are much less likely to have an effect on the actual cluster mean during the clustering process. This encourages tighter and more meaningful clusters to be developed, although, this leads to clusters that can just contain a single outlier, however this is addressed in the smoothing section.

Selecting the MFCC and MFSC features

As stated in the general codebase overview, the MFCC and MFSC features were plotted in a heatmap. This heatmap allowed me to see which feature coefficients had more difference when compared to the other classes of data. These MFCC features also changed depending on the frame size, which I will talk about choosing the proper one in the next section.

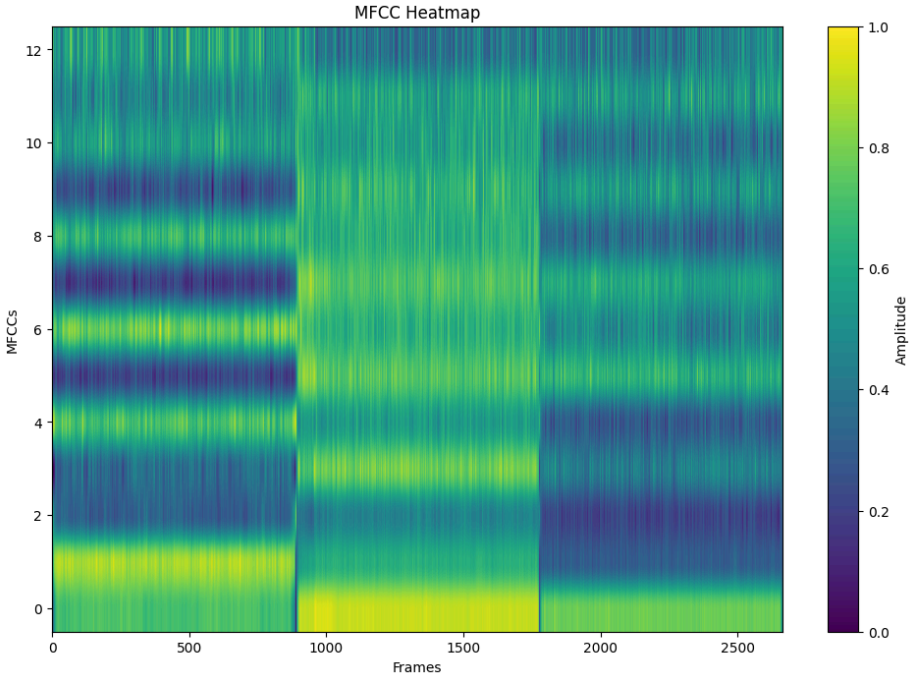

Figure 16.
The normalized MFCC heatmap for a 150ms frame size with 13 MFCC features.

As seen in Figure 16, we can see very distinguished boundaries at about frames 80,000 and 15,500. These boundaries are the changes in different audios. This is good because we want to find features that have the most variation between the differing classes. The goal of this heatmap is to look at which MFCC features have heavy differences between the classes. For example, looking at feature 2, we can see that for all classes, it is mostly the same color of blue, meaning this feature does not change that much depending on the class, and is a feature that will not help us categorize data. The same can be said for feature 12, where the feature is essentially the same color across the entire width.

On the contrary, a very distinct feature would be 5. In the first class it is a dark purple, in the next, a light green, and in the last a dark green. In each class there is a distinct color difference between the classes, and thus this feature is helpful for classification. Reducing the right features can have a fairly significant impact on the accuracy of the clustering. For example, running with 44 features leads to an overall accuracy of 63%, whereas 40 features leads to 93%.

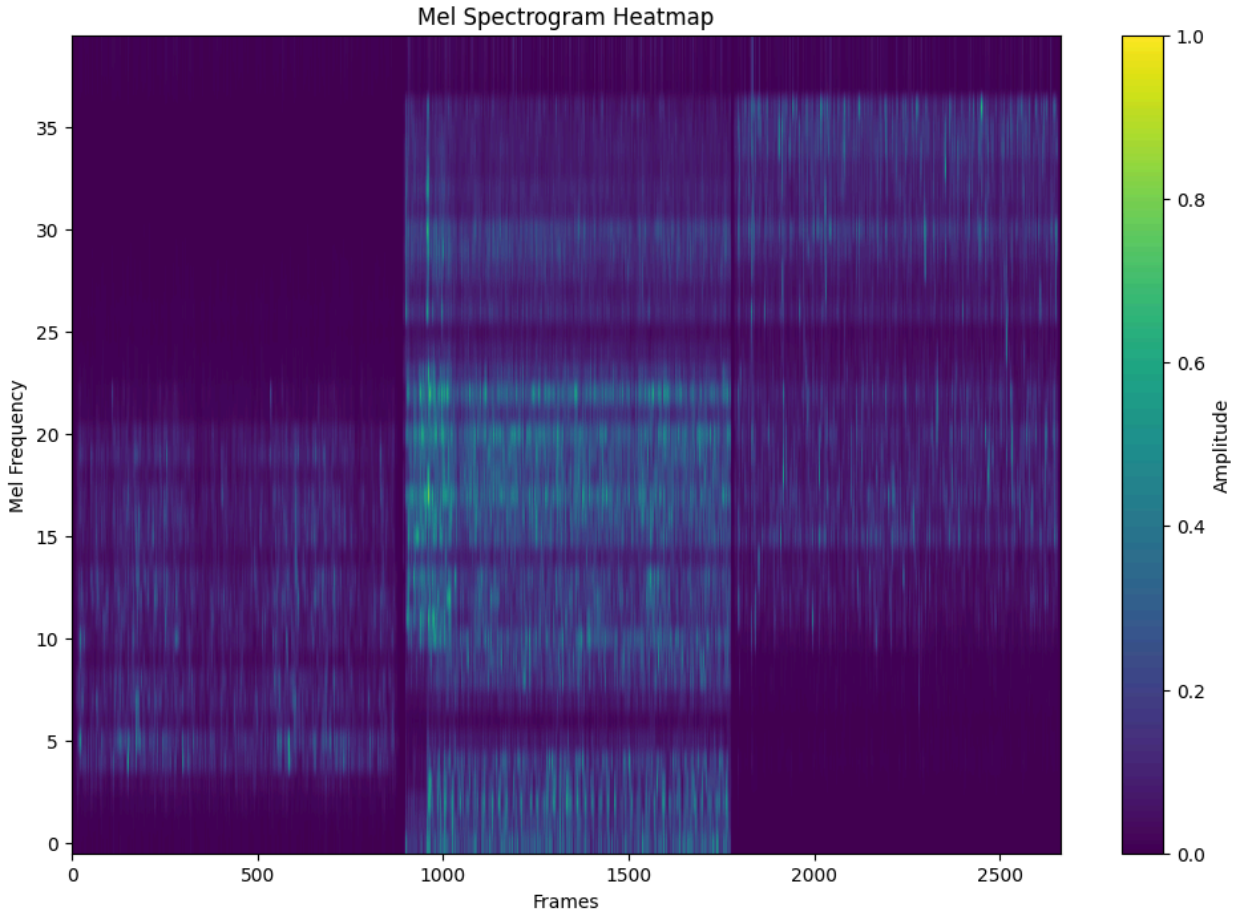

Figure 17.
The normalized heat map of the MFSCs for a 150 ms frame size 40 mels

Similar to the MFCCs, we want to use the features with the highest difference between the classes. In this case, the mels between 0 to 14 and 16 to 35 give the highest difference.

From these two heatmaps, it is easy to infer that the MFCC features give much more variety than the MFSC features, where the MFCC contains a wide variety of colors, and the MFSC contains mostly purple and blue colors. Looking at the clustering results displayed in figures 1-15, we can see that this inference is validated, with the MFCC features consistently achieving an accuracy about 10% higher than the MFSC features alone. Once we combine the two, we are able to get slightly more variation in our features, as well as a slightly improved accuracy.

Frame Size Selection

Frame size also allows for variability in the clustering accuracy. To choose the right frame size, I looked at the MFCC and MFSC variations between smaller and larger frame size selections.

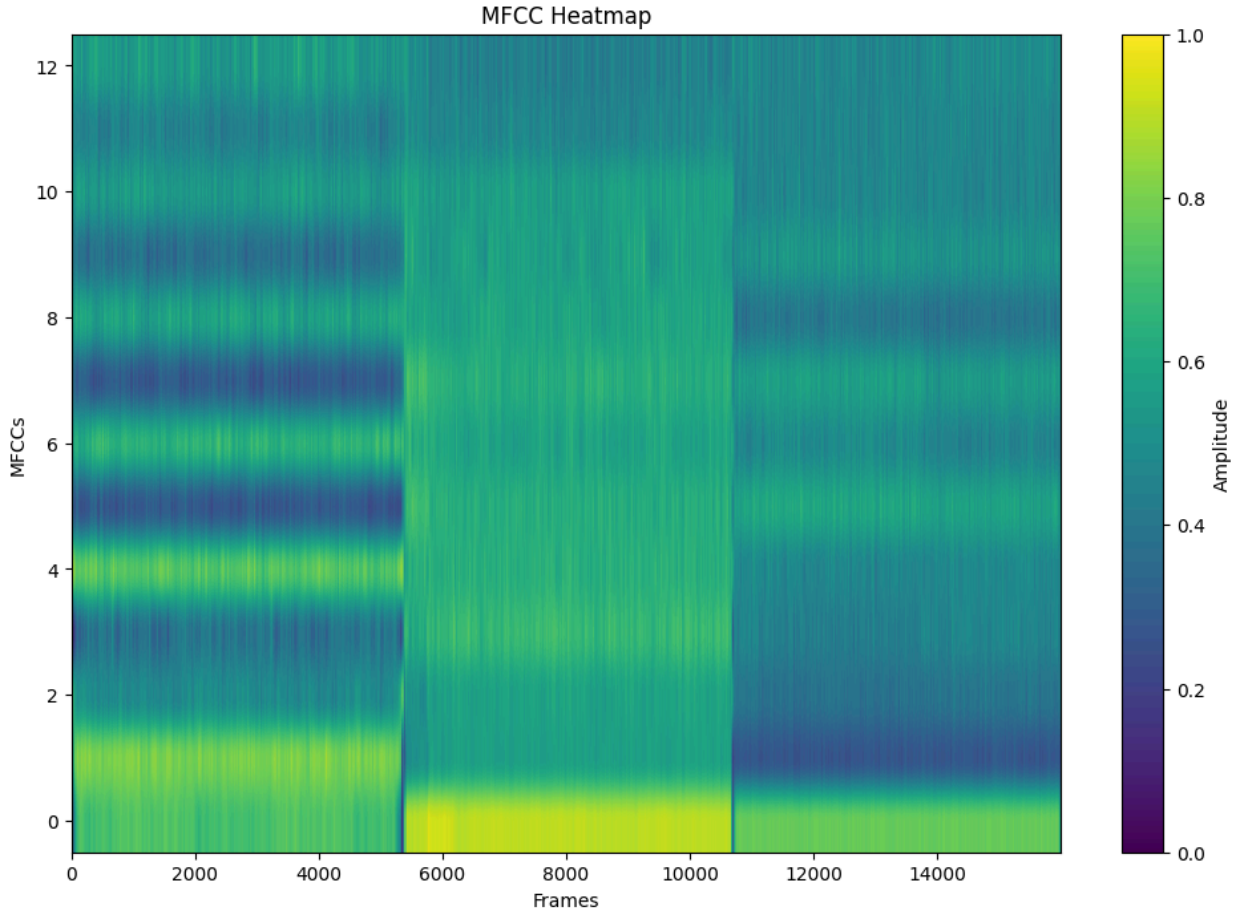

Figure 18.
The normalized MFCC heatmap for a 25 ms frame size with 13 MFCC features.

When comparing Figure 16, there is much less variability between classes, and features, in particular, the 2nd and 3rd class both sharing a deep shade of green. It can be seen that a decrease in frame size means a decrease in variability because there is now less audio to capture and thus less variability between small snippets of audio.

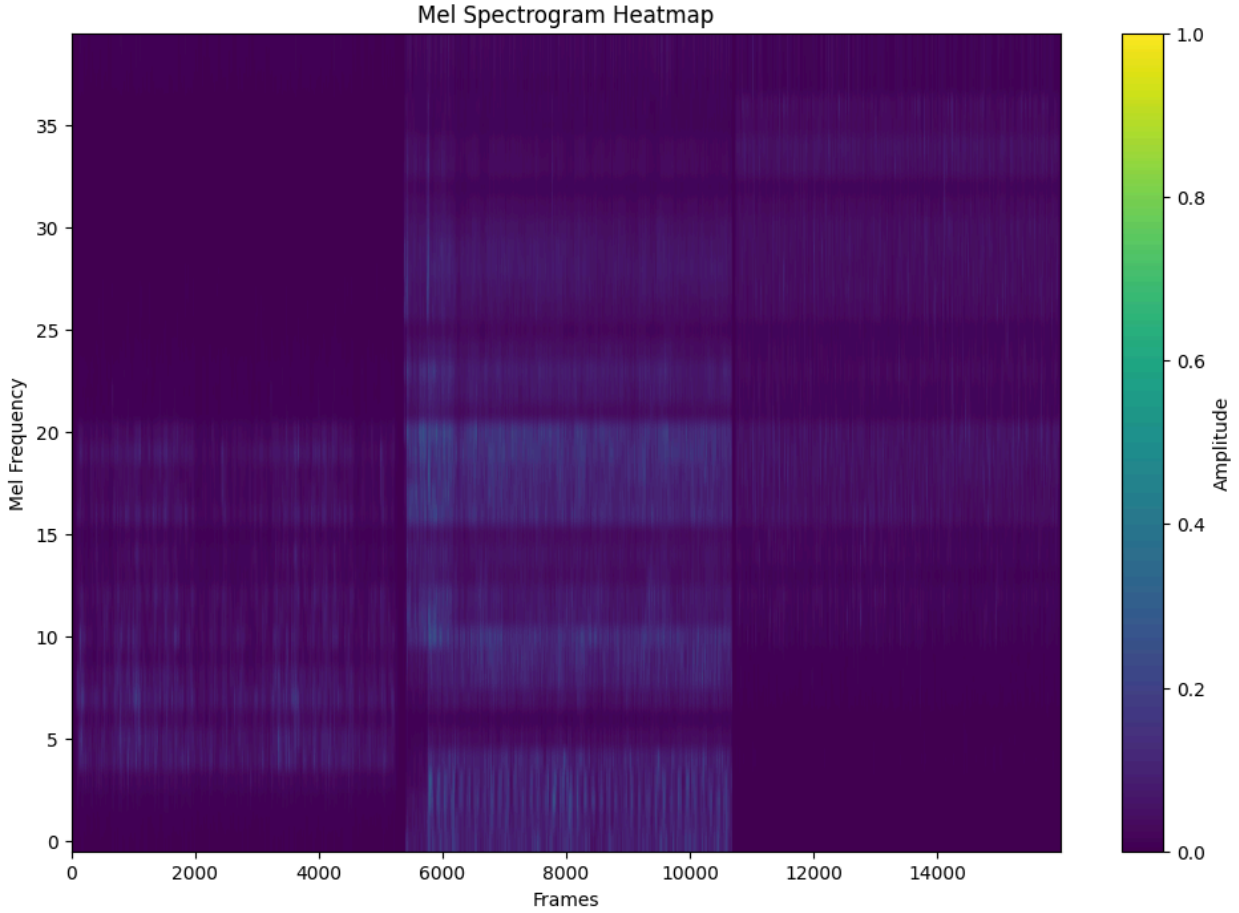

Figure 19.
The normalized heat map of the MFSCs for a 25 ms frame size 40 mels

The same can be said for comparisons between Figure 17 and Figure 19. Figure 19 now seems more like a block of purple, with almost no variability between classes.

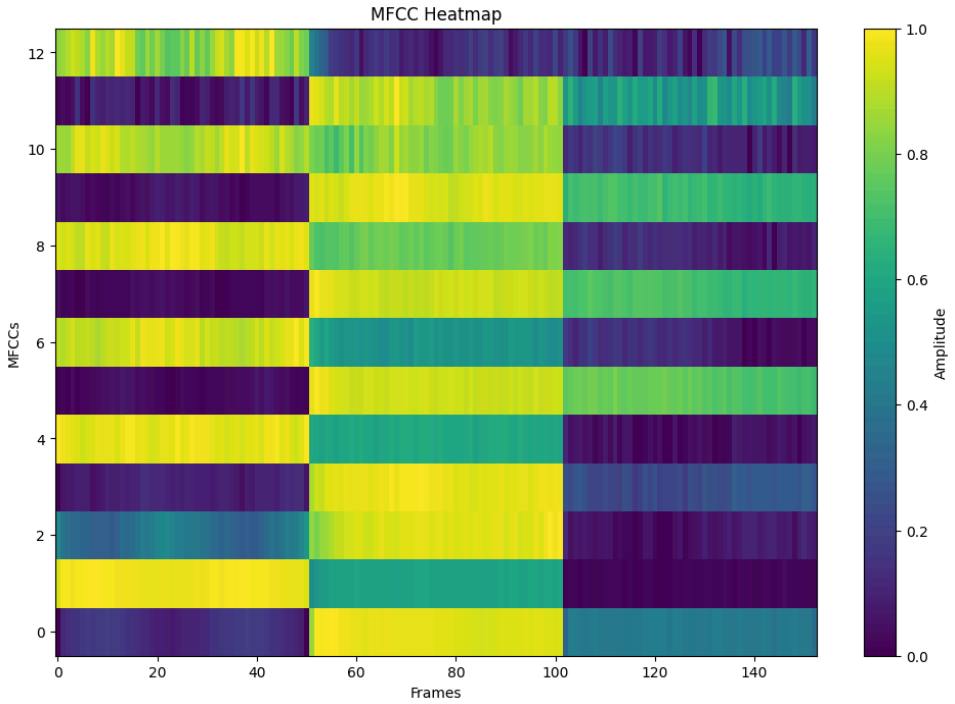

Figure 20.
The normalized MFCC heatmap for a 20,000 ms frame size with 13 MFCC features.

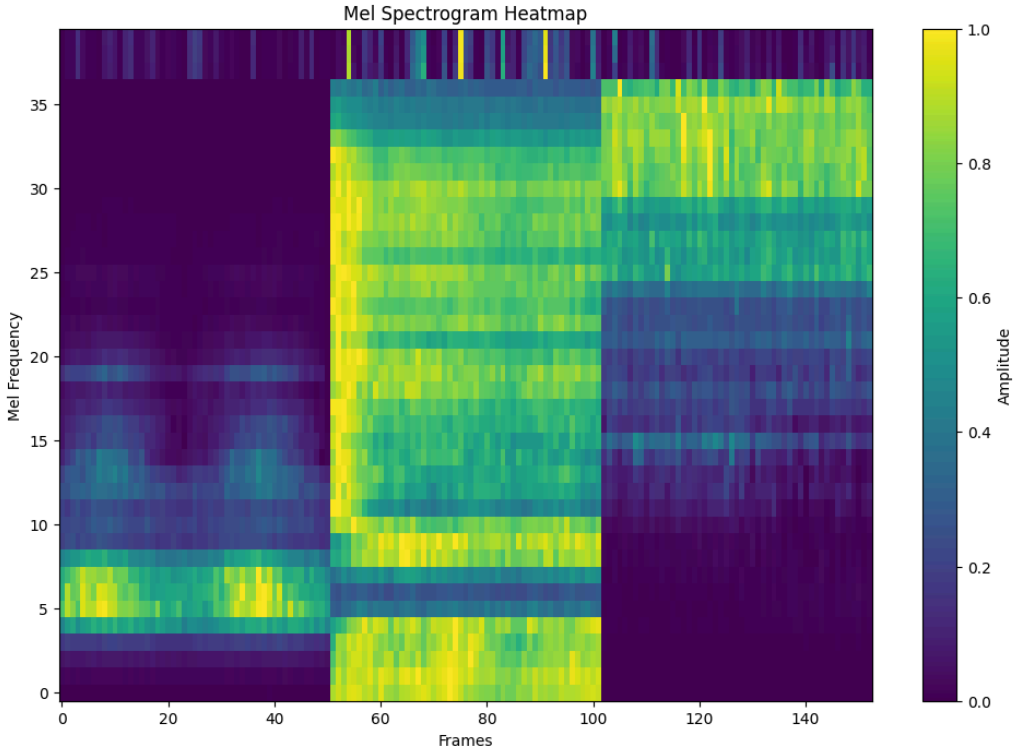

Figure 21.
The normalized heat map of the MFSCs for a 20,000 ms frame size and 40 mels.

On the contrary, creating exceptionally large windows creates very high variability as seen in figures 20 and 21. However, this comes at a cost. High windows means much less data to work with, and larger chunks of data being fed to the actual classifier. Although this might make the accuracy increase at first, when adding multiple audios that could sound similar but are truly different classes, there would be a very large decrease in accuracy. Thus, a middle ground of 150 ms was chosen.

Normalization vs No Normalization

Normalization turns all data points to be between 0 and 1. For this codebase, min max normalization is used, where the smallest feature becomes 0, and the largest becomes 1. From my testing, no normalization has a very large impact on both accuracy and efficiency.

Non normalized values cause features to not have a “weight”. When looking at MFCCs and MFSCs, a MFCC of 30 might have more weight than an MFSC of 100, but since the values aren’t normalized, we don’t know that, and using these plain numbers causes the distances between data points to be exceptionally large.

In my testing, because of the large MFSCs, my computer constantly ran out of storage when computing values and placing these large values into clusters.

Choosing a Vigilance Parameter

The vigilance parameter is the most sensitive to increasing or decreasing the accuracy, since it is the direct number an ART2 clustering algorithm uses to determine whether to create a new cluster or not.

The best way to see how the vigilance parameter was changing was by looking at either how many clusters were being created, or how many data points were being placed in clusters. Generally, if there are too many data points in a single cluster, the vigilance parameter is too large, and if there are too many clusters, the vigilance parameter is too small.

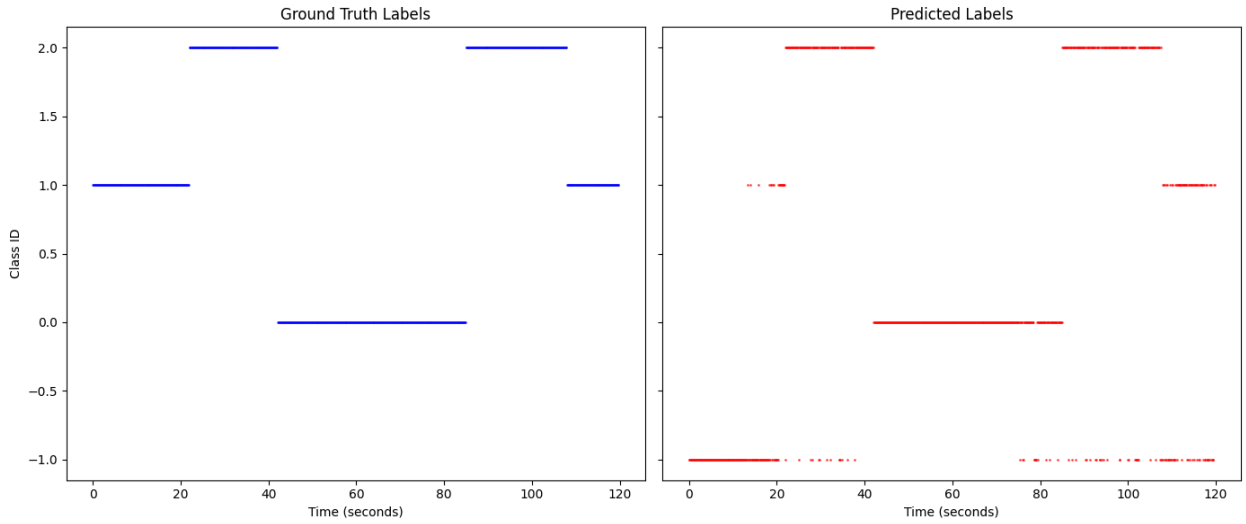

Figure 22.
The clustering results when the vigilance is set to 0.5.

As seen in Figure 22, many elements data points are classified as -1 in the predicted graph, as well as class 1 being much harder to classify. Because the vigilance is set to a smaller number like 0.5, more clusters are being created as data points that are further away from the class means are fed into the clustering algorithm.

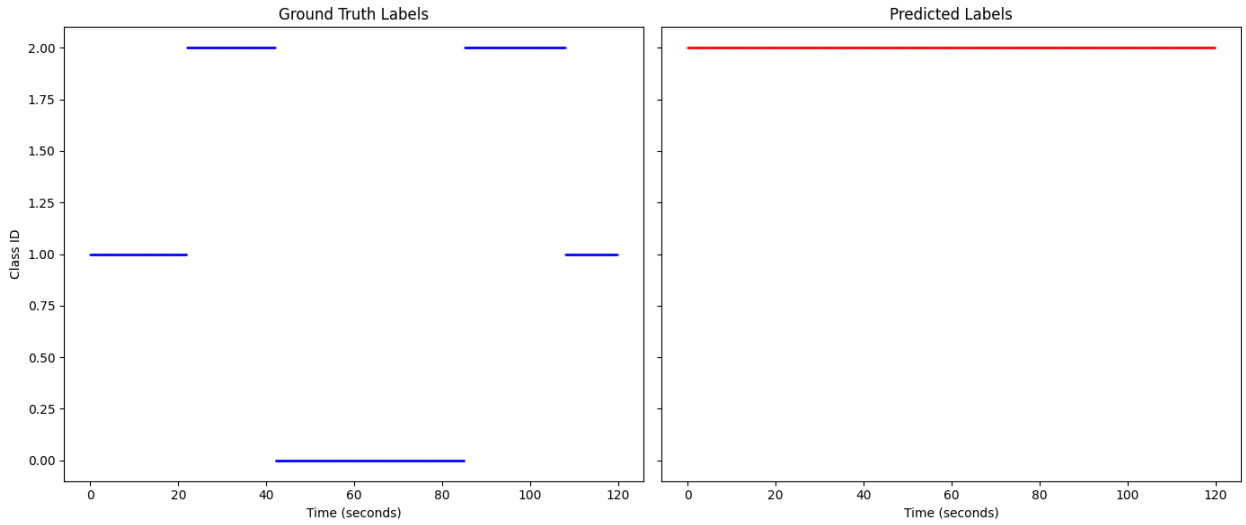

Figure 23.
The ground truth clusters, and the predicted label cluster with a vigilance of 10

On the contrary, in figure 23 we set the vigilance to a very large number 10. Because the tightness of each cluster now becomes so laxed, the clustering algorithm creates a single cluster, which is the class that is first fed into the clustering algorithm.

These graphs show me how to change the vigilance parameter, where long lines of data tell me that the vigilance is too large, and letting in data points from classes that don't belong. Similarly, if too many data points are showing up in class -1, it means I need to increase the vigilance to allow existing clusters to accept more data points.

Smoothing

Adding smoothing to the clustering model adds another variable to adjust that can greatly impact the performance of the model, that being buffer size. Essentially, the buffer works by adding a number of datapoints into the buffer. When the buffer is full, a vigilance test is done on the mean of the buffer rather than a single datapoint, and the data points in the buffer are classified based on their buffer mean. This helps eliminate outlier points, as they are smoothed with other points.

I found that the best buffer size was 5, where a larger amount of data points can offset an outlier. Introducing a buffer also allowed for a smaller vigilance, with a new vigilance for combined data being 0.55. The following figures show the highest achieved accuracy when using a model with no smoothing.

```
Clustering time series 0 - combined features
Accuracy: 0.8648648648648649
Confusion Matrix:
[[163   0   0]
 [  0 248  12]
 [  0   0 357]]
```

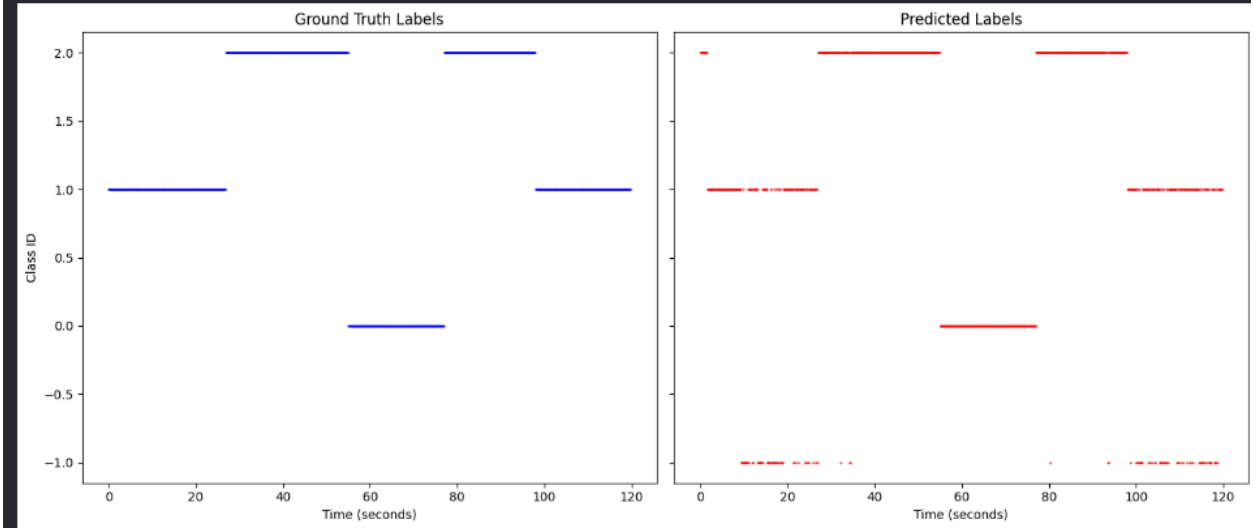

Figure 24.
Clustering results for time series 0 without smoothing

```
Clustering time series 1 - combined features
Accuracy: 0.9921171171171171
Confusion Matrix:
[[268   0   0]
 [  0 303   0]
 [  1   4 310]]
```

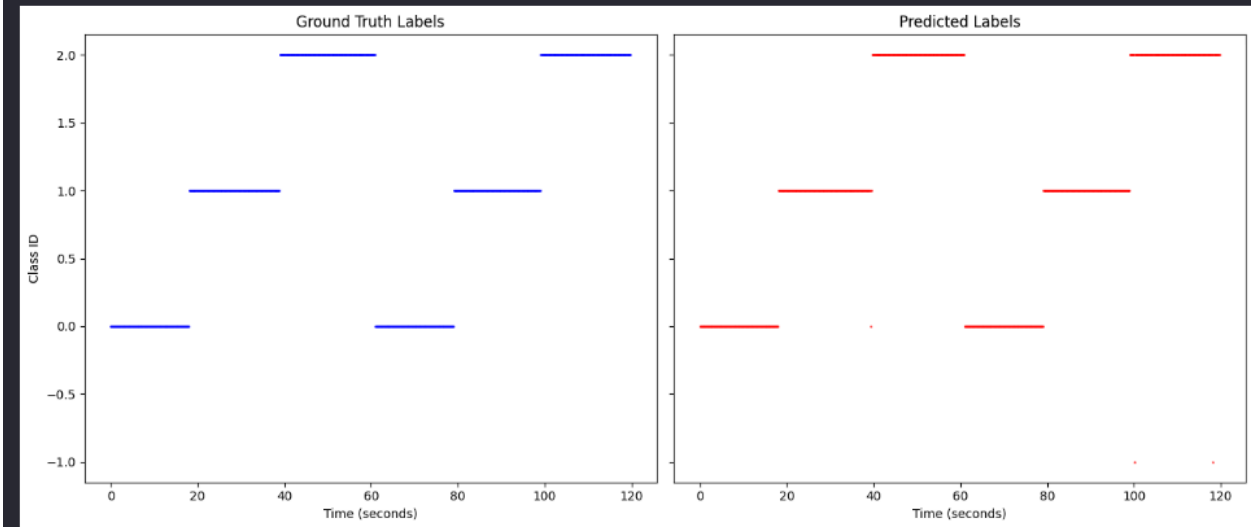

Figure 25.
Clustering results for time series 1 without smoothing

```
Clustering time series 2 - combined features
Accuracy: 0.990990990990991
Confusion Matrix:
[[280   0   0]
 [  0 309   0]
 [  0   0 291]]
```

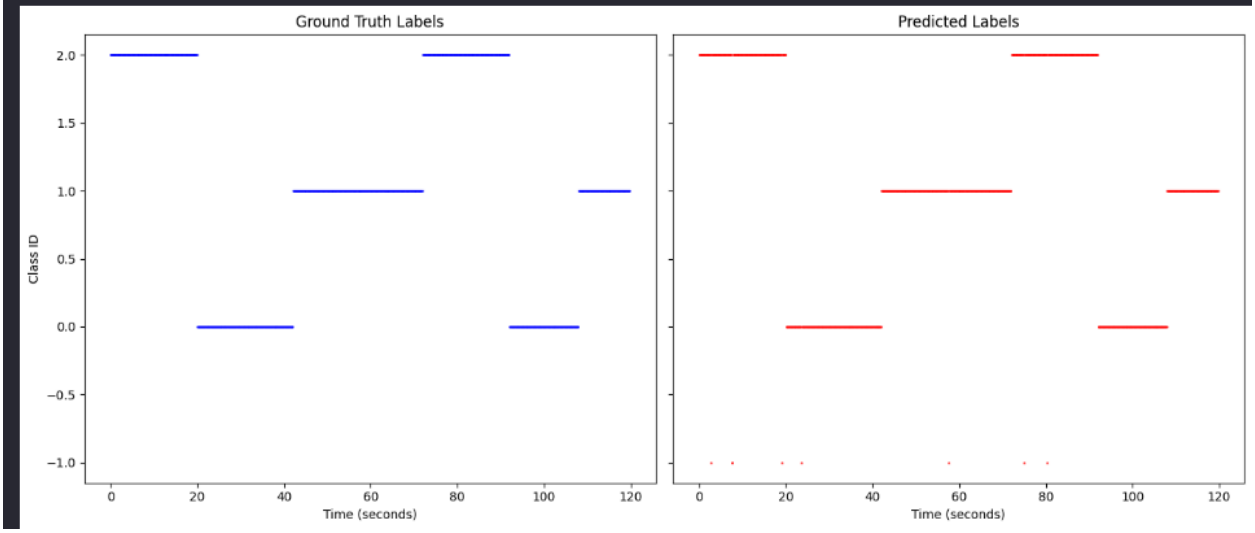

Figure 26. Clustering results for time series 2 without smoothing

```
Clustering time series 3 - combined features
Accuracy: 0.9876126126126126
Confusion Matrix:
[[267   0   0]
 [  0 384   0]
 [  2   8 226]]
```

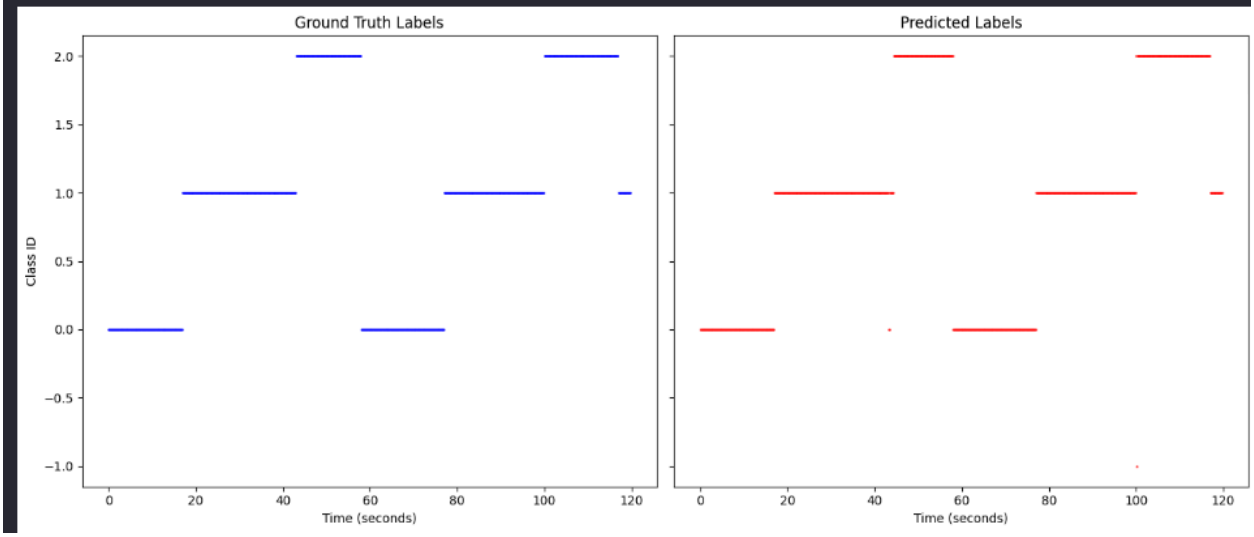

Figure 27. Clustering results for time series 3 without smoothing

```
Clustering time series 4 - combined features
Accuracy: 0.9921171171171171
Confusion Matrix:
[[178   0   0]
 [  0 326   0]
 [  0   0 377]]
```

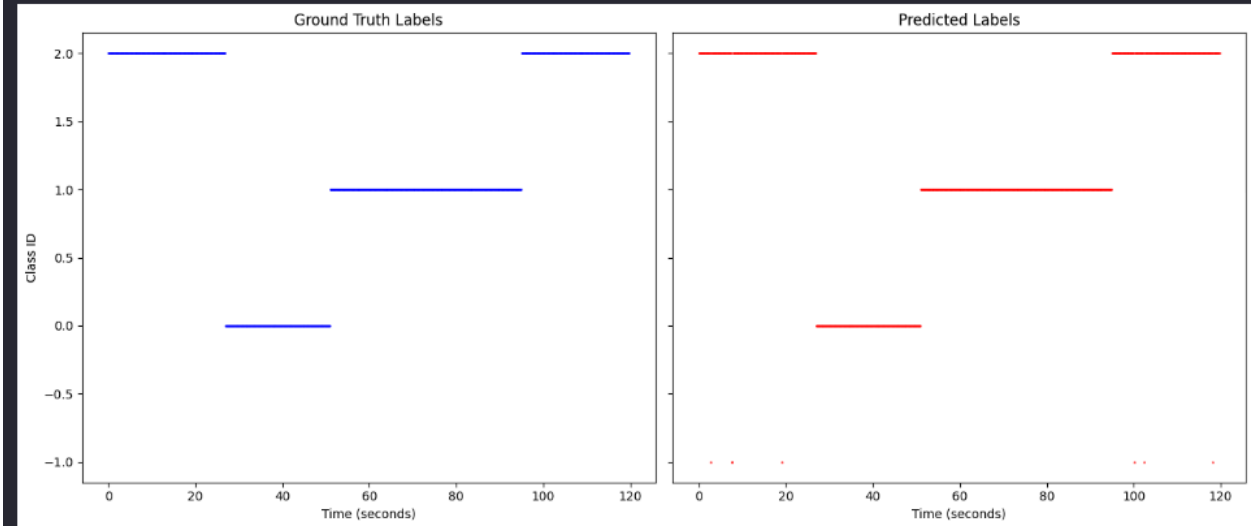

Figure 28. Clustering results for time series 4 without smoothing

From comparing figures 1-15 to figures 24-28, there are a couple things to note. First is the stability of the accuracy, and second is the creation of unclassified classes (the -1 classes in the predicted labels graph).

In regards to accuracy stability, we can see that figures 24-28, although achieving good accuracy results, there are occasional dips to 85% in figure 24. As a matter of fact, these dips occur rather frequently when randomly generating the time series, as these dips can likely be explained with a specific part of a ground truth class containing an outlier. On the other hand, we see that in figures 1-15 with smoothing, the accuracy remains between 95%-99%, as outliers from classes can get smoothed out by other data points before being added to the cluster.

Looking at figure 24-28, we can see that these classes all have some data points plotted at a -1 class, meaning the model creates a cluster that is not part of the ground truth clusters. Again, these points are outliers, and because of their odd distance from the other classes, they misclassified as another class. When looking back to Figures 1-15 (the ones that use a combined feature set), we see that only figure 9 and figure 15 have data points in -1, most of the time, data is classified as 0, 1, or 2, the ground truth classes.

PCA Results

Overview

To further increase the accuracy, we add dimensionality reduction using PCA. PCA adds another parameter to adjust that being the variance when applying the PCA. From my testing, I found that for the most optimal PCA results yield a 99%-100% accuracy. I use only the MFCC features, no smoothing, and the following variables:

1. Frame size (ms): 150
2. Hop ratio: 0.1
3. n_fft: 1024
4. Vigilance (for MFCC only): 1.4

The following figures show the clustering results for MFCC features only (the most optimal result), MFSC features, and combined features:

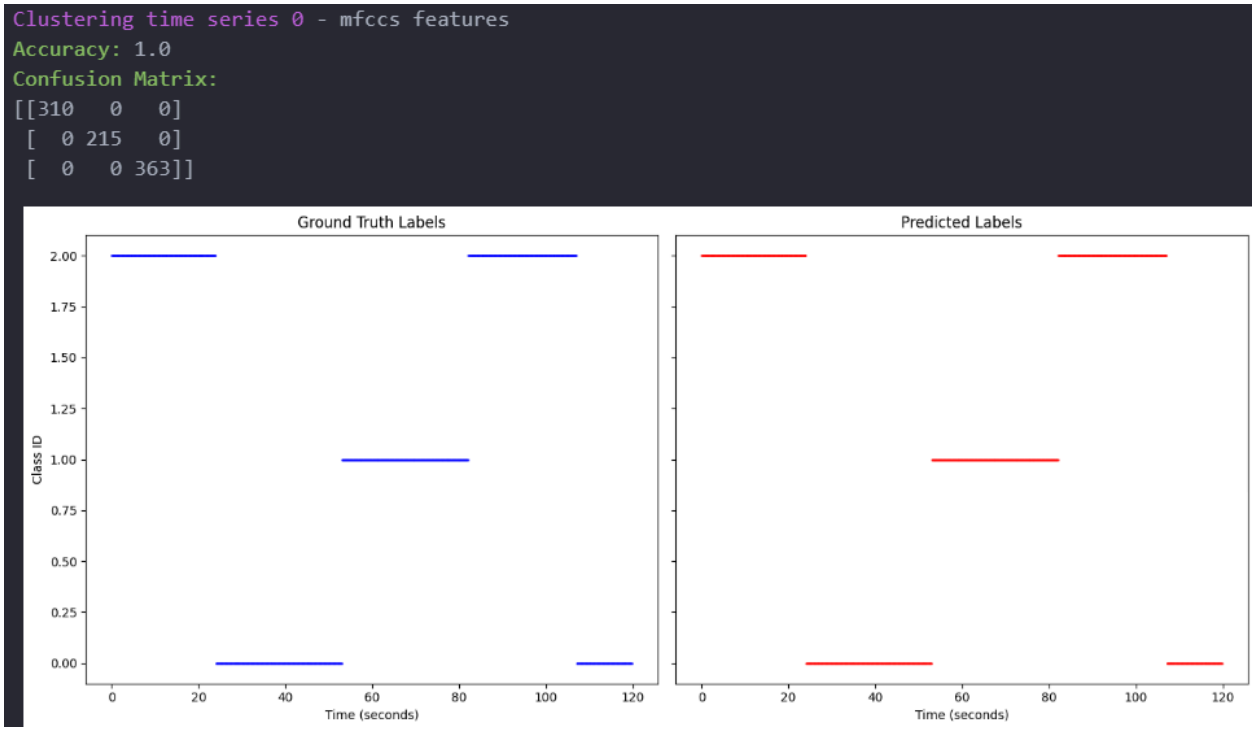

Figure 29.
The clustering results for time series 0 using just MFCC features

```
Clustering time series 0 - melspec features
Accuracy: 0.3952702702702703
Confusion Matrix:
[[  0   0 304]
 [  0   0 228]
 [  0   5 351]]
```

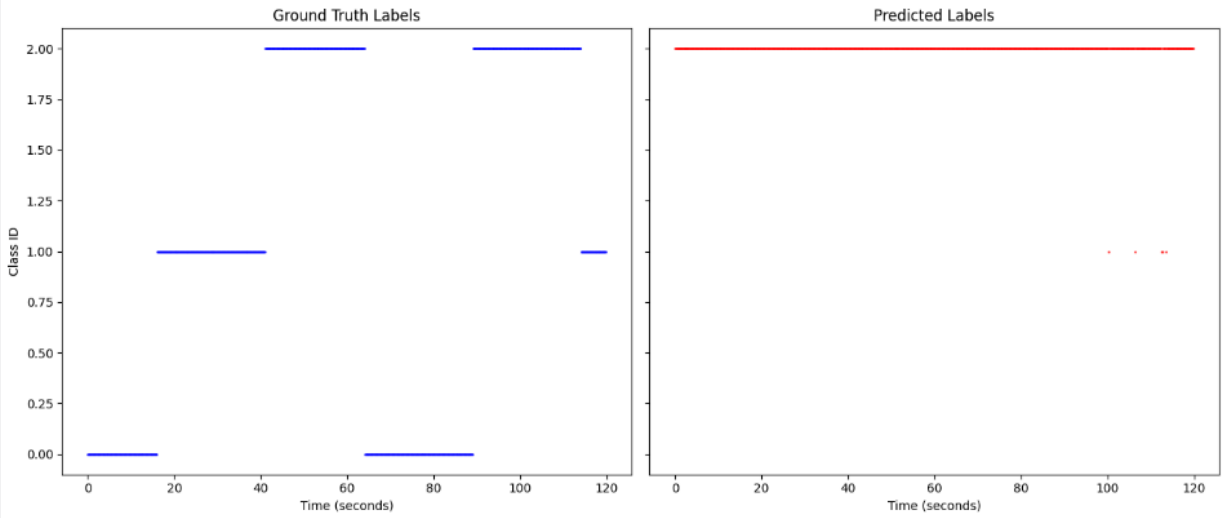

Figure 30.
The clustering results for time series 0 using just MFSC features

```
Clustering time series 0 - combined features
Accuracy: 0.7905405405405406
Confusion Matrix:
[[286   1   0]
 [ 11 380   2]
 [172   0  36]]
```

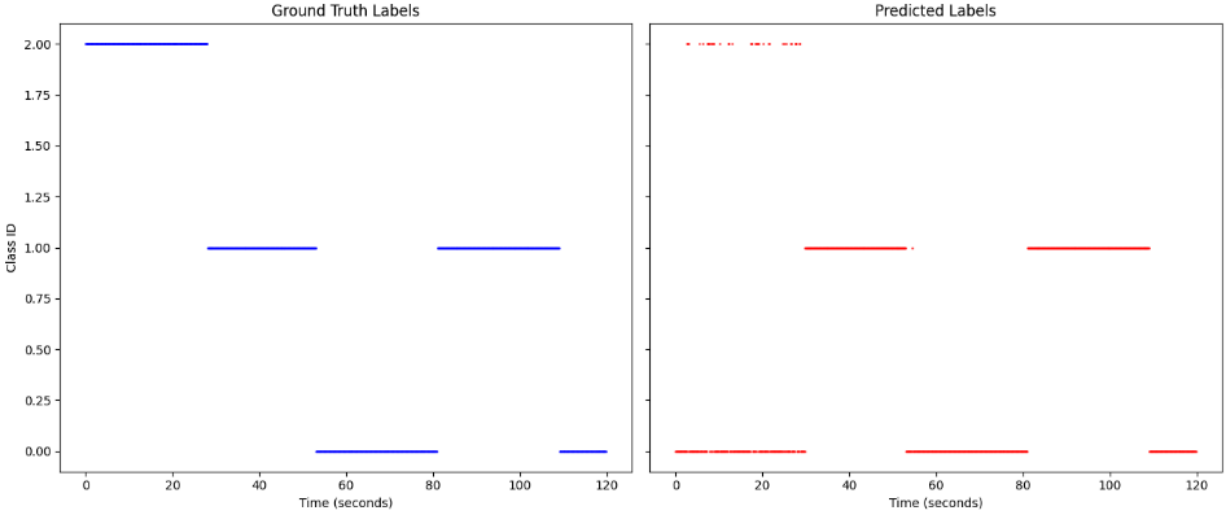

Figure 31.
The clustering results for time series 0 using combined features

```
Clustering time series 1 - mfccs features
Accuracy: 1.0
Confusion Matrix:
[[140   0   0]
 [  0 406   0]
 [  0   0 342]]
```

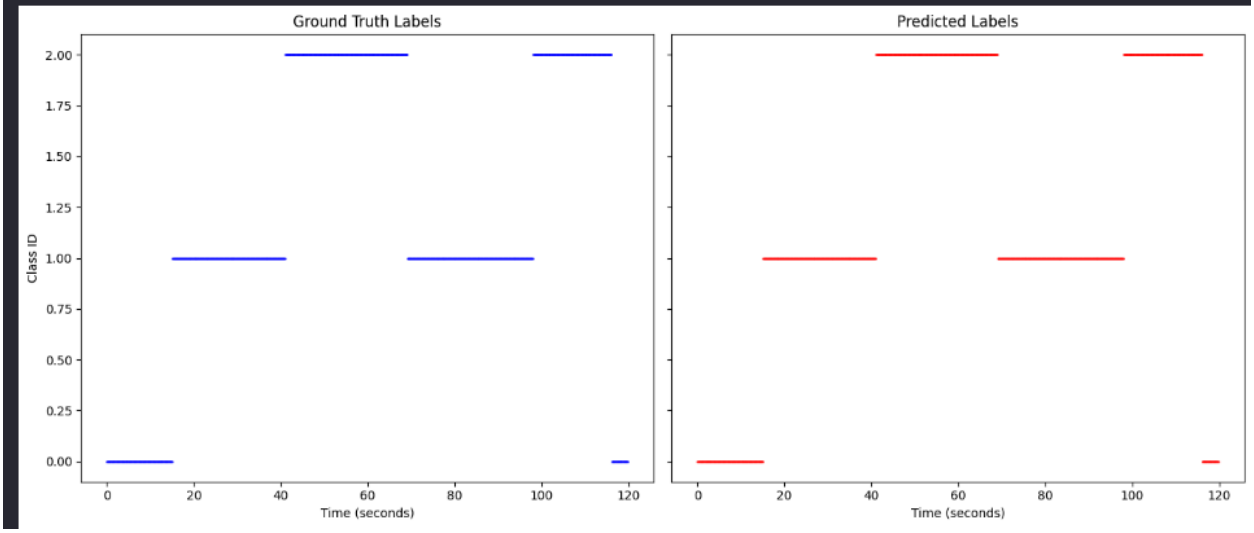

Figure 32.
The clustering results for time series 1 using MFCC features

```
Clustering time series 1 - melspec features
Accuracy: 0.6655405405405406
Confusion Matrix:
[[230   0   0]
 [  2 361   0]
 [168 127   0]]
```

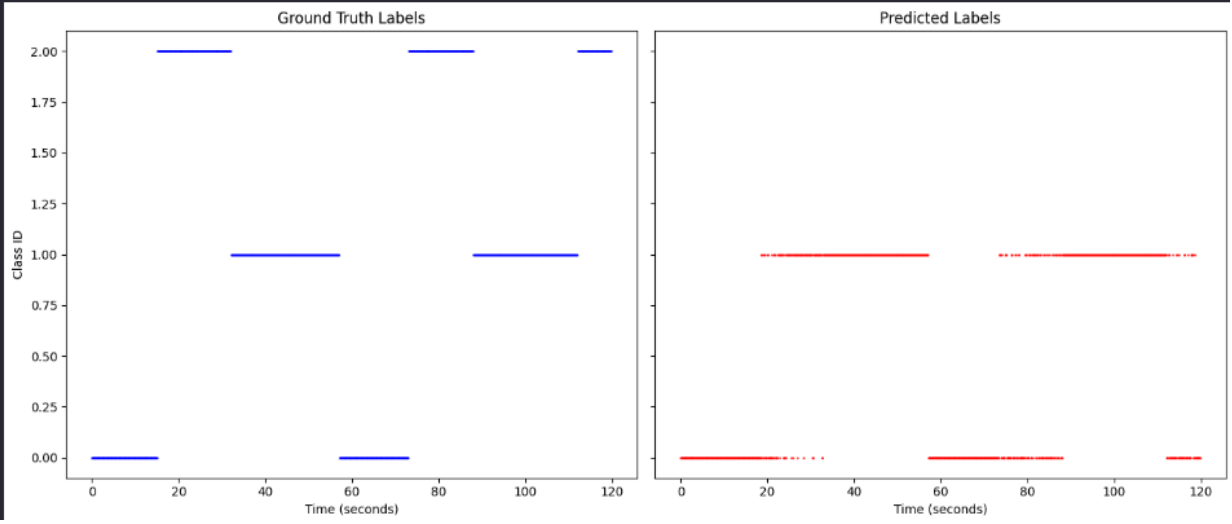

Figure 33.
The clustering results for time series 1 using MFSC features

```
Clustering time series 1 - combined features
Accuracy: 0.8378378378378378
Confusion Matrix:
[[321   5   0]
 [ 18 296   0]
 [117   1 127]]
```

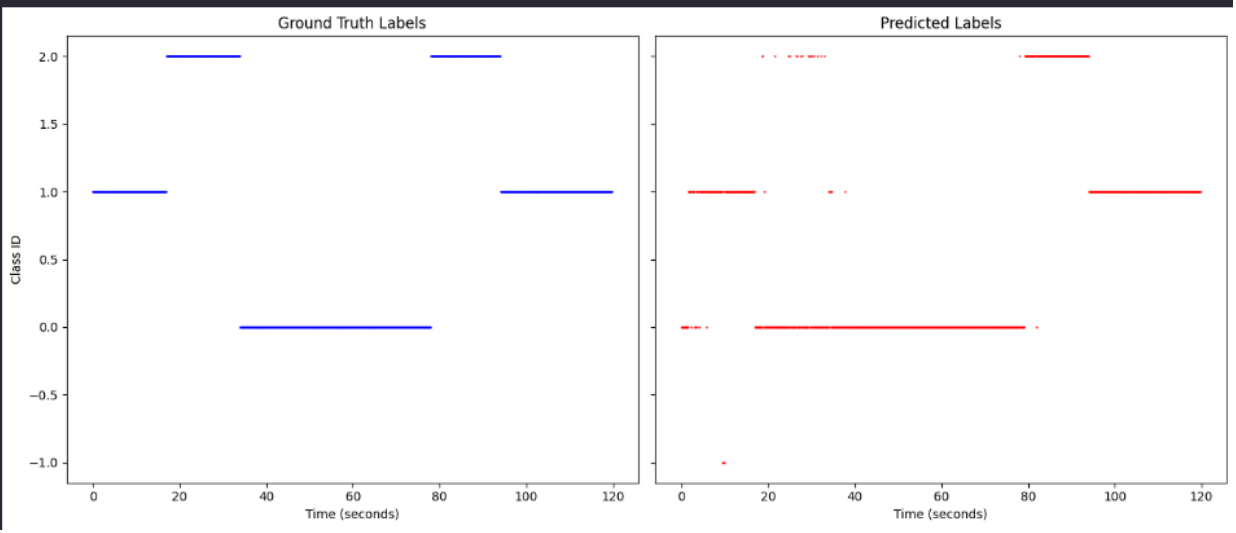

Figure 34.
The clustering results for time series 1 using combined features

```
Clustering time series 2 - mfccs features
Accuracy: 0.9921171171171171
Confusion Matrix:
[[252   0   0]
 [  6 416   0]
 [  0   0 213]]
```

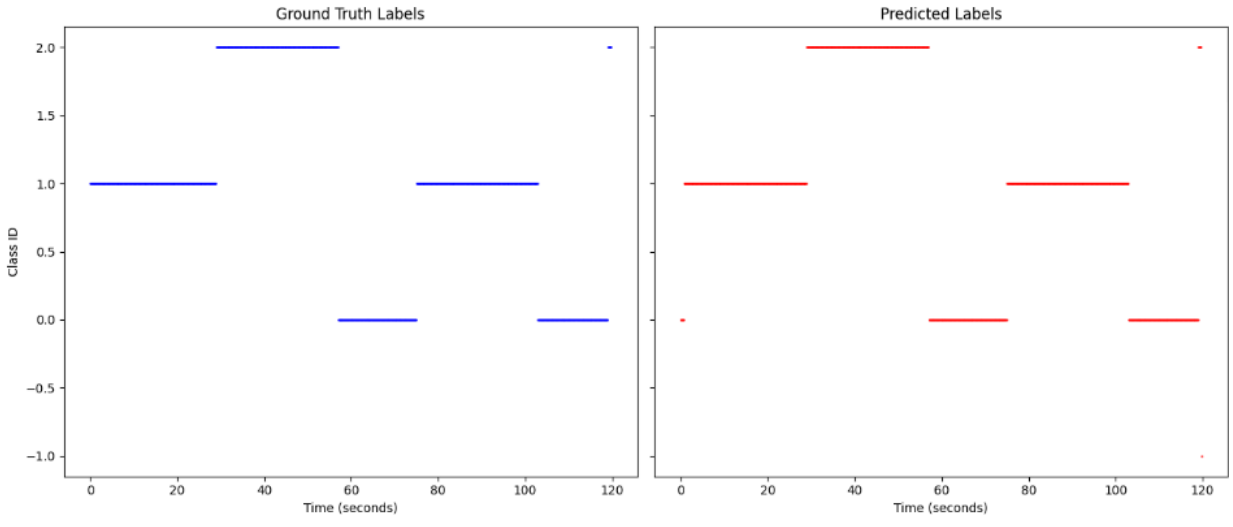

Figure 35. The clustering results for time series 2 using MFCC features

```
Clustering time series 2 - melspec features
Accuracy: 0.6779279279279279
Confusion Matrix:
[[  0   0 244]
 [  2 281  28]
 [  1  11 321]]
```

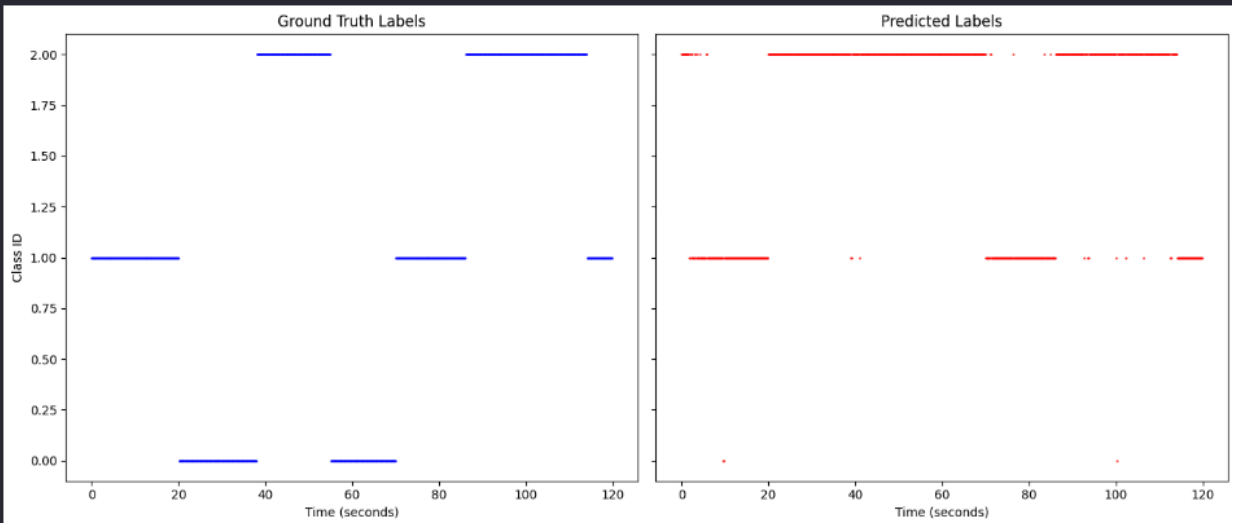

Figure 36.
The clustering results for time series 2 using MFSC features

```
Clustering time series 2 - combined features
Accuracy: 0.6272522522522522
Confusion Matrix:
[[222   0   0]
 [193 176   0]
 [138   0 159]]
```

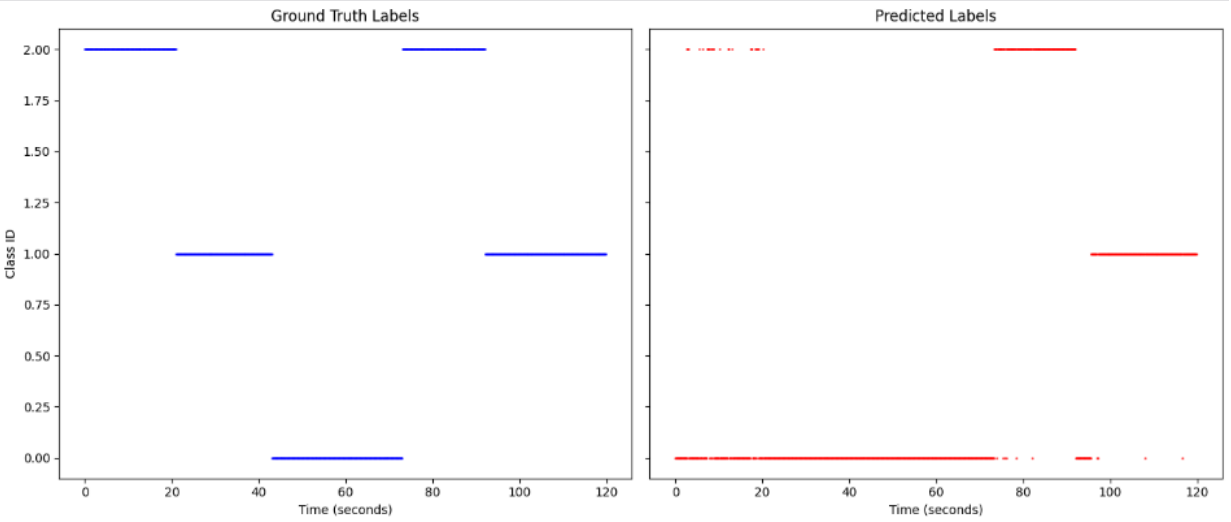

Figure 38.
The clustering results for time series 2 using combined features

```
Clustering time series 3 - mfccs features
Accuracy: 0.9932432432432432
Confusion Matrix:
[[237   0   0]
 [  6 312   0]
 [  0   0 333]]
```

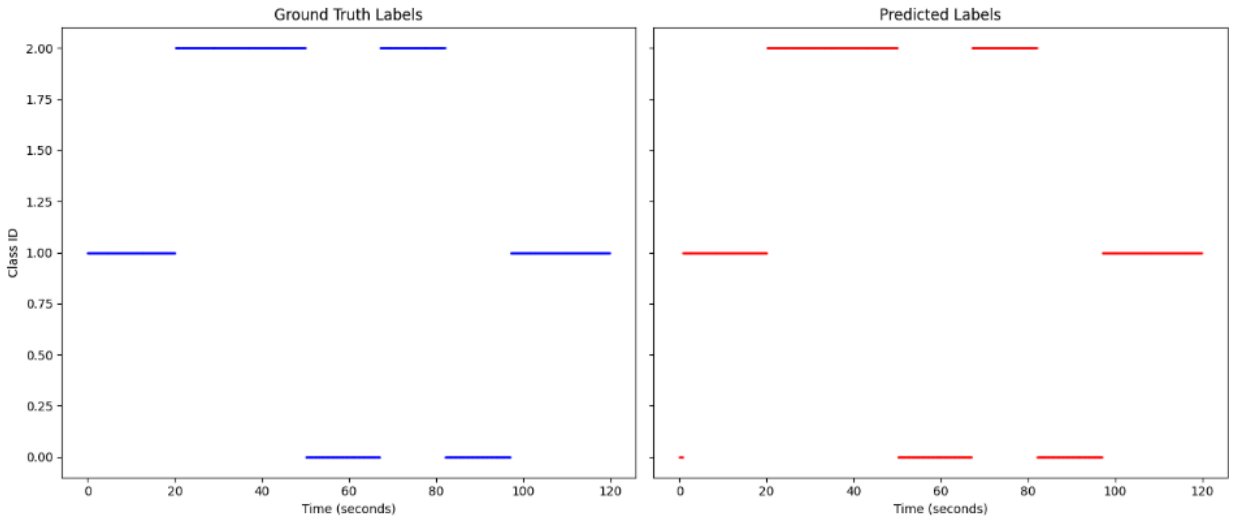

Figure 39.
The clustering results for time series 3 using MFCC features

```
Clustering time series 3 - melspec features
Accuracy: 0.37274774774774777
Confusion Matrix:
[[  0   0 319]
 [  0   0 228]
 [  9   1 331]]
```

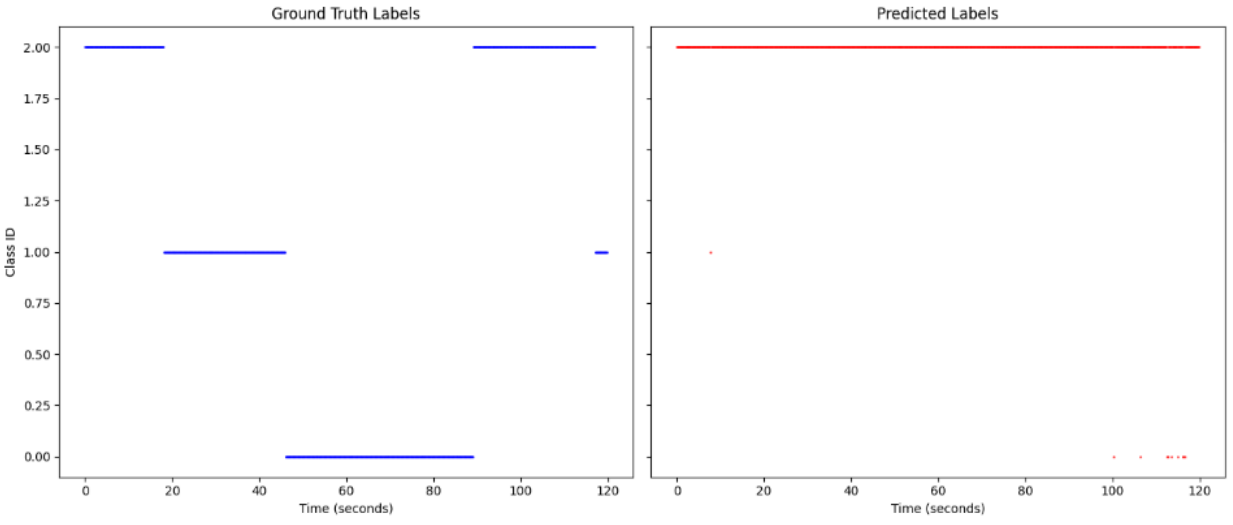

Figure 40.
The clustering results for time series 3 using MFSC features

```
Clustering time series 3 - combined features
Accuracy: 0.7713963963963963
Confusion Matrix:
[[342   0   0]
 [ 31 343   3]
 [153  16   0]]
```

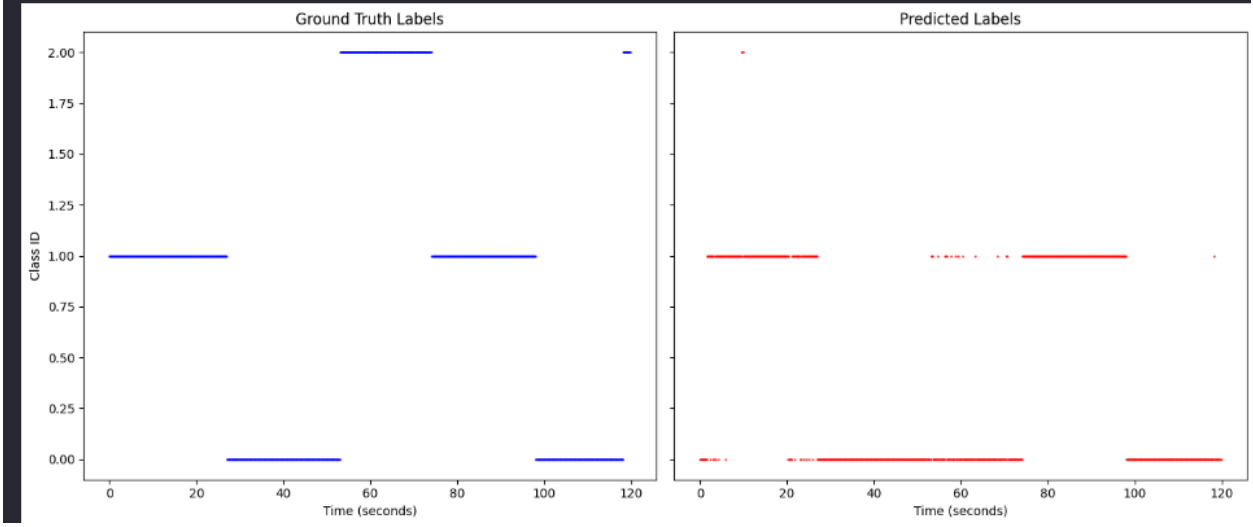

Figure 41.
The clustering results for time series 3 using combined features

```
Clustering time series 4 - mfccs features
Accuracy: 0.9898648648648649
Confusion Matrix:
[[170   0   0]
 [  6 335   0]
 [  0   2 374]]
```

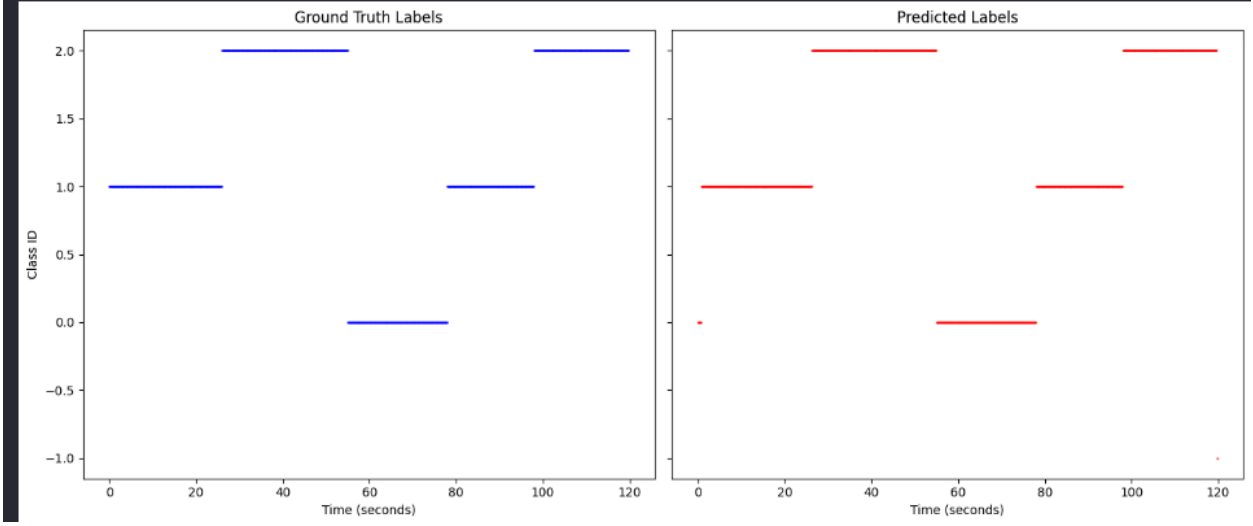

Figure 42. The clustering results for time series 4 using MFCC features

```
Clustering time series 4 - melspec features
Accuracy: 0.6644144144144144
Confusion Matrix:
[[  0 156   0]
 [  0 415   0]
 [  0 142 175]]
```

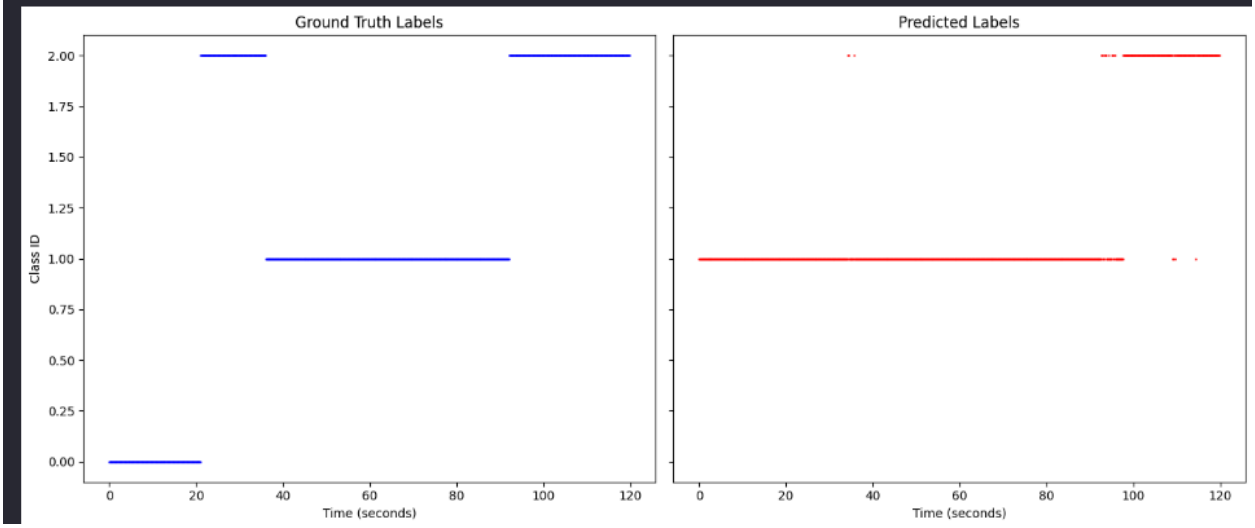

Figure 43.
The clustering results for time series 4 using MFSC features

```
Clustering time series 4 - combined features
Accuracy: 0.6092342342342343
Confusion Matrix:
[[342   0   0]
 [326   0   7]
 [ 14   0 199]]
```

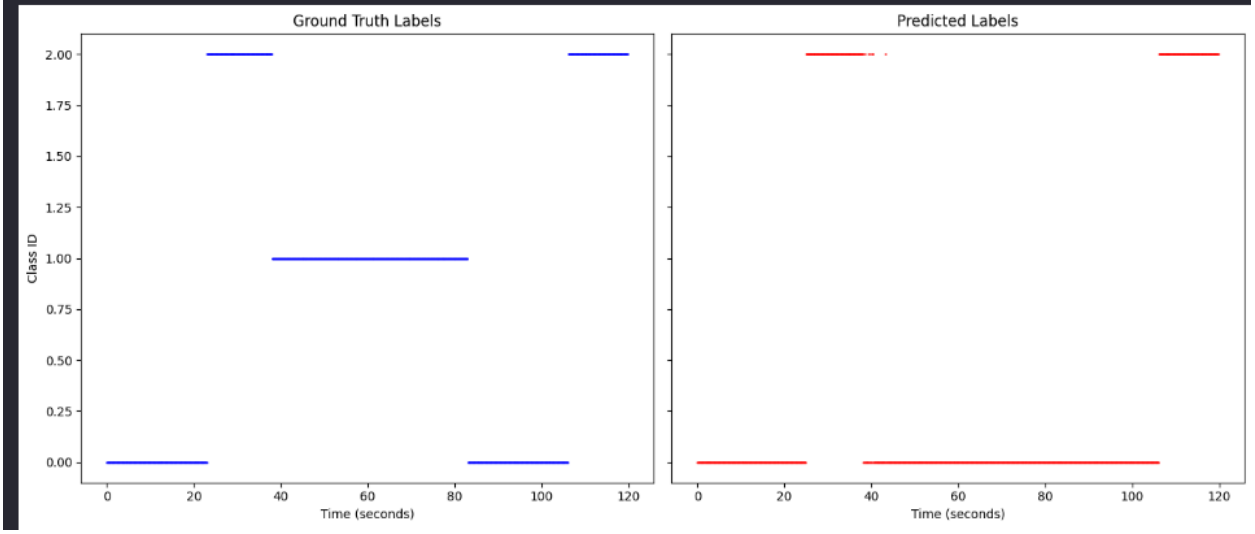

Figure 44.
The clustering results for time series 4 using combined features.

From figures 29 to 44, we can see that the MFCCs produce the best results. This section will go over how the parameters for finding the best values were found, as well as why certain techniques such as smoothing were avoided.

Variance and Feature Selection

The variance of the PCA in this context can be thought of including or reducing noise in the audio samples. A higher variance, such as 0.95, captures more of the feature set and retains more information overall. A lower variance, such as the one I am using, 0.6 takes away more information from the features, which in turn can reduce the noise.

Similar to how finding which MFCC and MFSC features benefited my model the best, I used heatmaps to decipher which features would provide the greatest differences between the classes. The following figures are heatmaps generated of the MFCC, MFSC, and combined features after PCA with a certain variance:

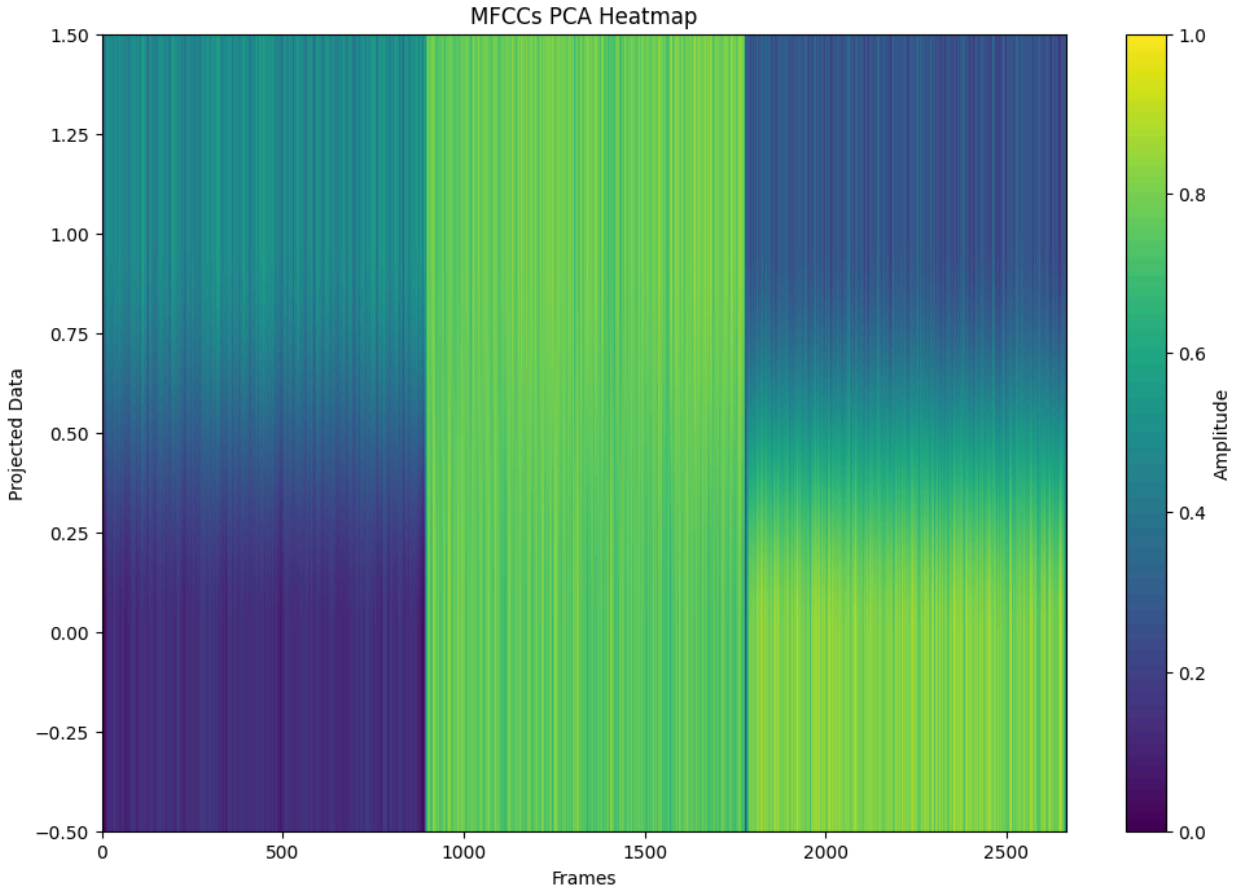

Figure 45.
The heatmap of MFCC features after PCA with a 0.6 variance

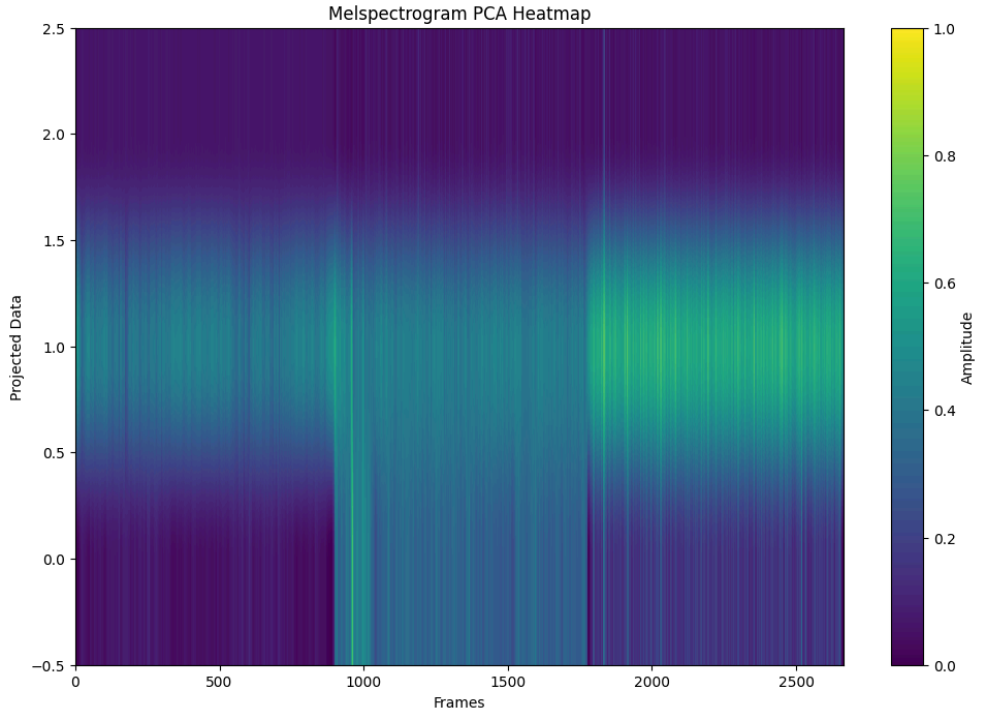

Figure 46.
The heatmap of MFSC features after PCA with a 0.6 variance

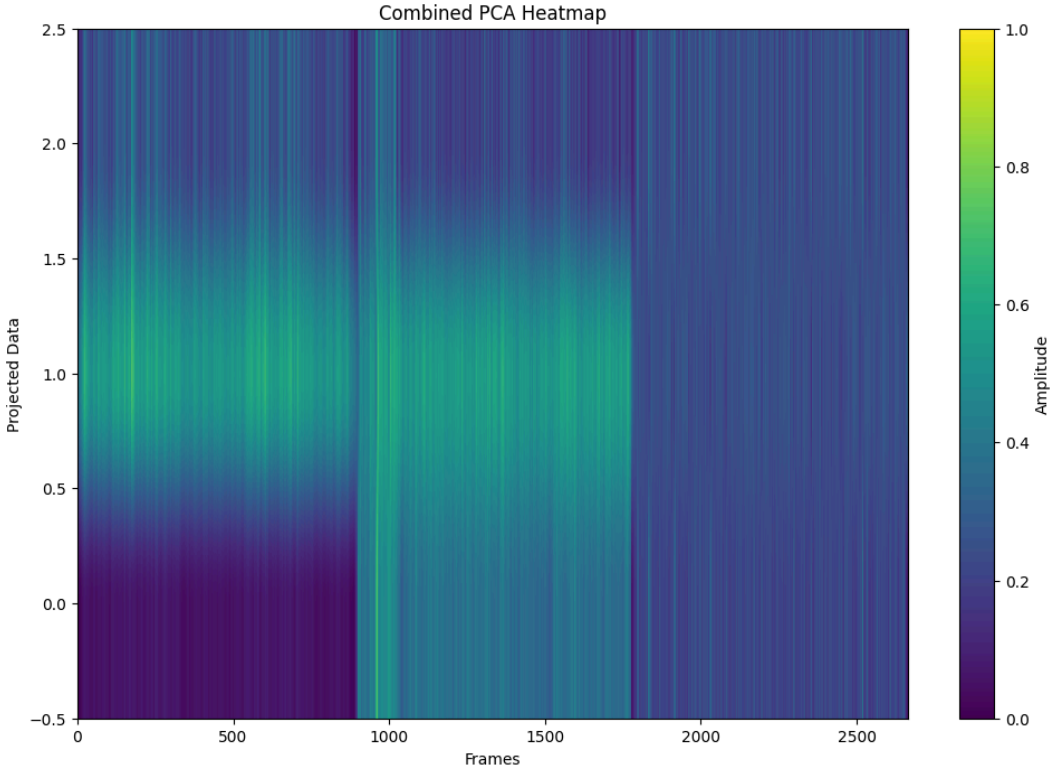

Figure 47.
The heatmap of combined features after PCA with a 0.6 variance

The goal of these heatmaps is to see which classes provide the most variability in color between classes. We can see that figure 45, the heatmap with MFCC features only, has almost completely different colors for each class. Because of this, we can assume that out of the MFCC, MFSC, and combined values, MFCC values will give us the best data to cluster with.

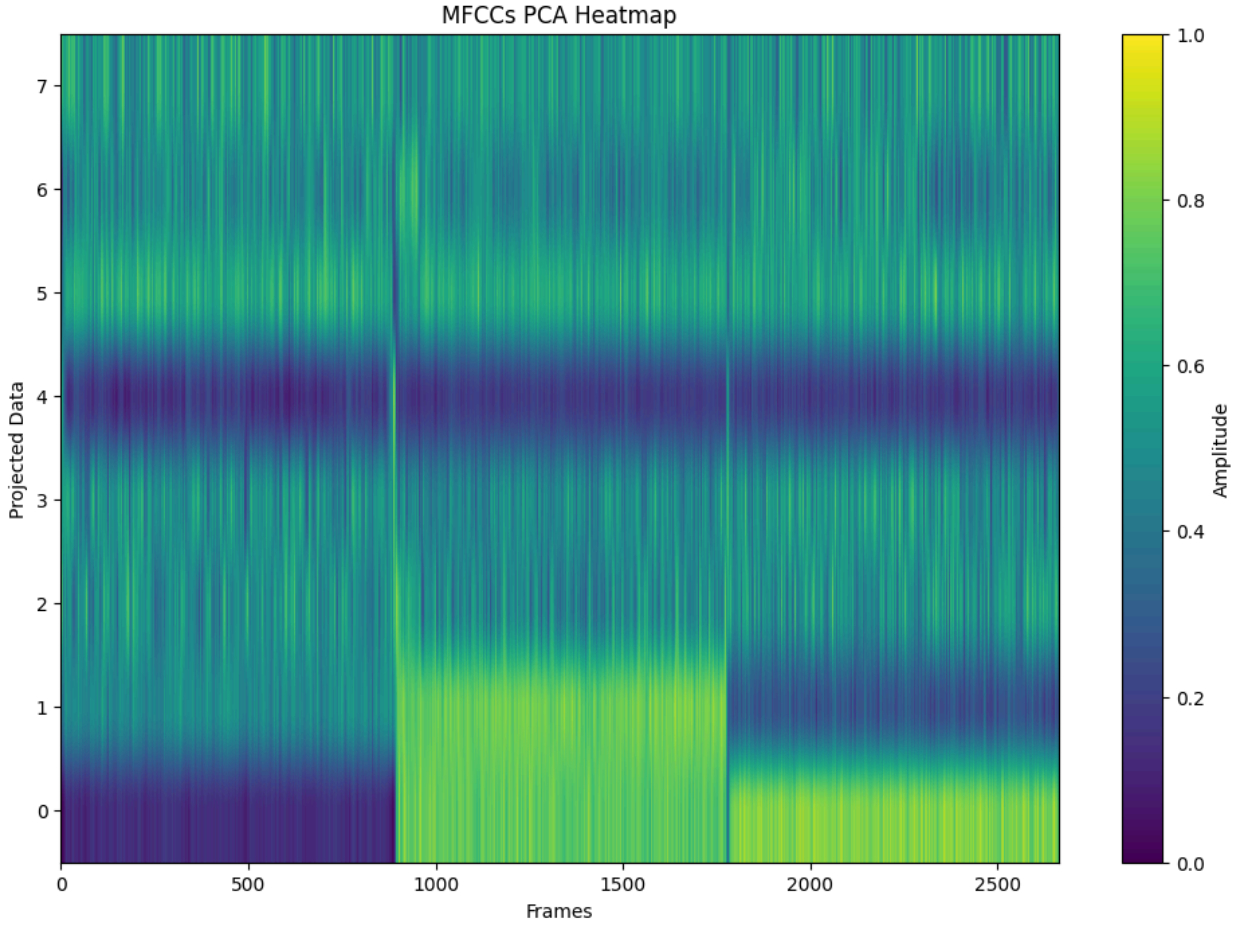

Figure 48.
The heatmap of MFCC features with a 0.95 variability

When comparing figure 48 with figure 45, we can immediately tell there is a lot more noise in figure 48. This is due to more noise being captured with a higher variance when compared to a smaller variance. This is also reflected in the model, where a higher variance drops the accuracy from near 100% to around 85%.

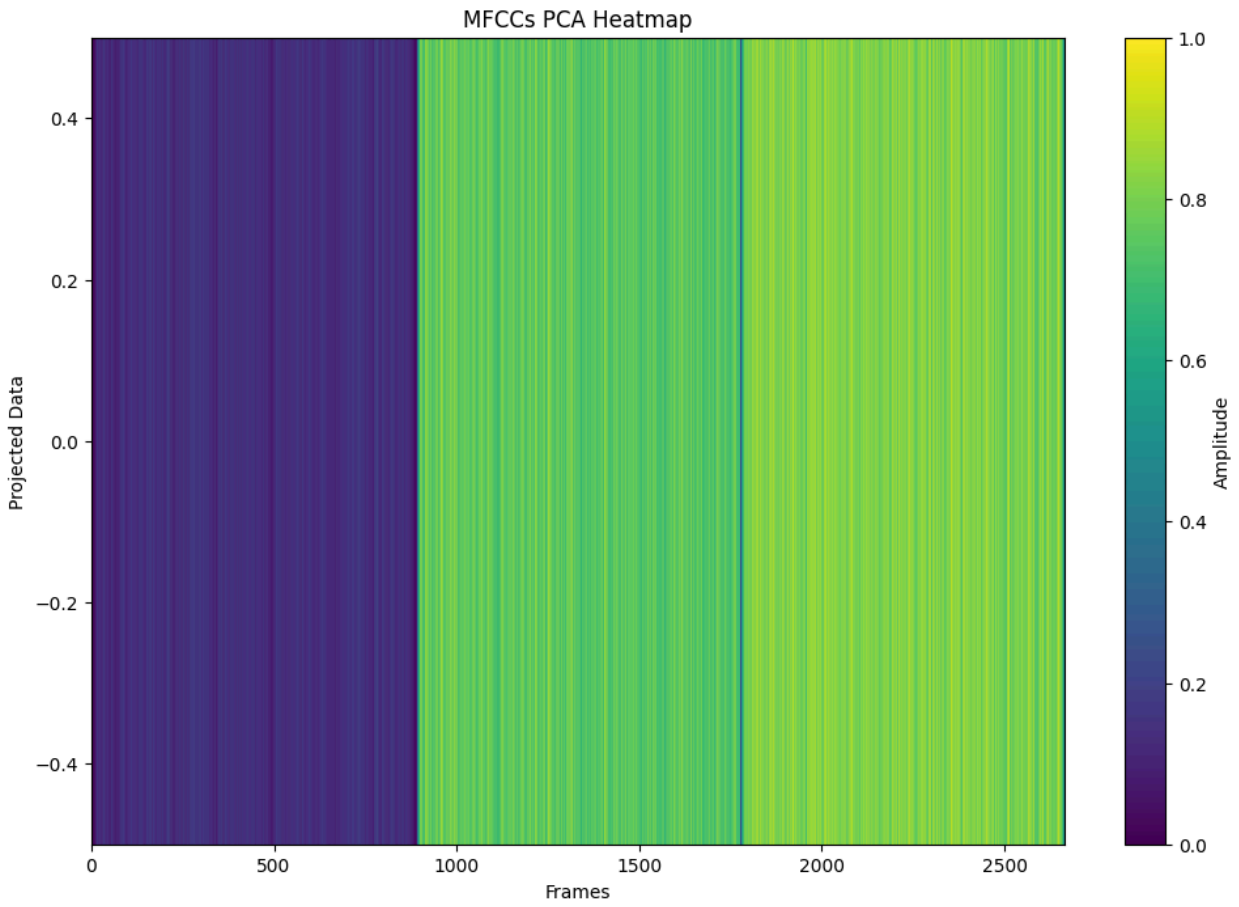

Figure 49.
The heatmap of MFCC features with a 0.1 variance

On the other side of the spectrum, we can see that a low variance causes a nearly identical match between class 1 and 2 because we have reduced the features to almost nothing. Because of this, we need to find a middle ground that can show enough noise such that classes are distinguishable by the clustering algorithm.

Why Smoothing is not used

Smoothing is not used in the optimal solution in the case when the buffer contains data points from multiple classes, some data will be misplaced to the wrong cluster.

```
Clustering time series 0 - mfccs features
Accuracy: 0.992090395480226
Confusion Matrix:
[[268   0   1]
 [  2 305   4]
 [  0   0 305]]
```

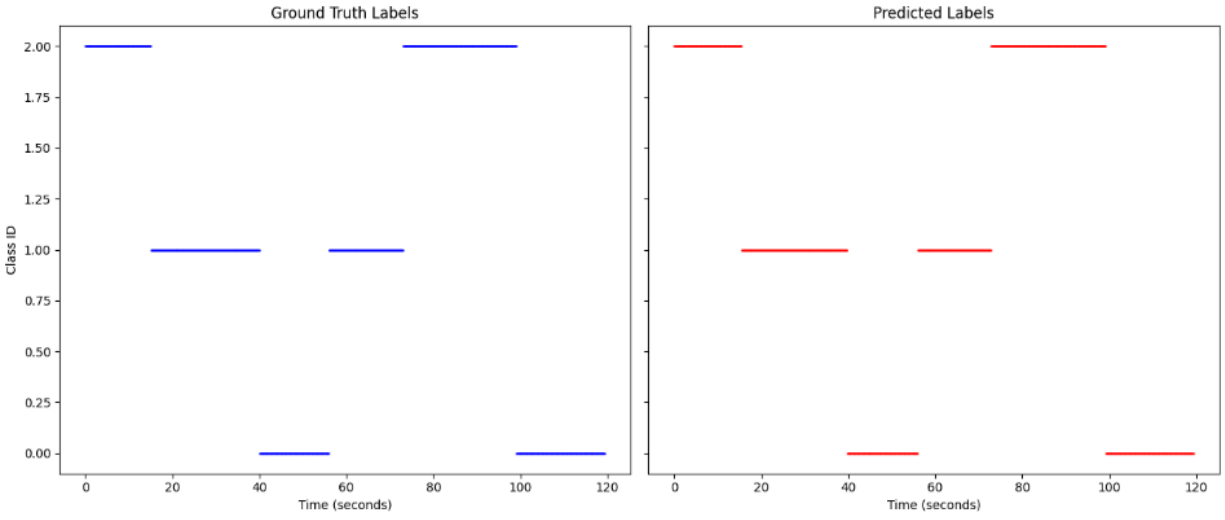

Figure 50.
The clustering result for time series 1 using PCA and smoothing

We can see that although this result is still very good (99%), data points are still being misclassified compared to when we are not using smoothing.

When we used smoothing without PCA, the purpose of it was to get rid of outliers. However, when using PCA we are already getting rid of noise, especially with a lower variance number. Because of this, the usage of smoothing becomes unnecessary, and negatively impacts the accuracy.

Variance

Now that we have a new feature set, a new variance needs to be found. The process of finding an optimal variance is the same as described in the no PCA results. This time, however, since the feature set has reduced, we are dealing with less numbers making the vigilance parameter smaller, as data points are now closer together.

From my testing, I found the most optimal vigilance for MFCC PCA features to be 0.15.

Conclusion

Overall, good results were able to be achieved. For results not using PCA, an accuracy of 95%-99% was able to be achieved with smoothing, normalization, combined features, 1.4 vigilance, and 150 ms frame size. For PCA results, a 99%-100% accuracy was able to be achieved using no smoothing, normalization, MFCC features only, 150ms frame size, 0.6 variance, and 0.15 vigilance.