Spring 2023

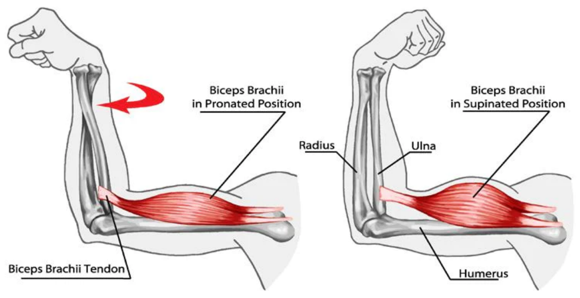
Model for movement classification of the bicep

Youssef Kharrat , '23

E90 Report:
Model for movement classification of the bicep

By : Youssef Kharrat
Advised By : Professor Maggie Delano

Swarthmore College, Spring 2023


Biceps Brachii in Pronated Position
Biceps Brachii Tendon
Radius
Ulna
Biceps Brachii in Supinated Position
Humerus

Abstract

This project introduces a classification model whose purpose is the distinction between the two possible movements done by the bicep; namely flexion and rotation. Surface EMG data was collected using the MIKROE-2621. It was later sampled from Analog to Digital by a Nucleo 401-RE board at 200 Hz. To generate the dataset, I connected the apparatus to my arm and performed the movements a total number of 300 times. This data was later analyzed and a choice of features were made that eventually was fed into MATLAB Classification Learner. A Support Vector Machine performed best and was able to classify the movements at a 94.3 % accuracy rate. This report shows the different techniques used to analyze the data and how I developed an understanding of the principal components that govern this classification using statistical methods.

Introduction

Electromyography is a technique used to detect and measure electrical activity produced by the muscle units during muscle contraction. This signal can be decoded to study muscle function and movement. The ability to accurately classify bicep EMG signals into the appropriate movement could help with sports science, and human-computer interaction.
For instance, such a system could be used to control a prosthetic arm, since in a lot of cases the impairment is not due to neurological degradation. This would allow amputees to control their artificial limbs more naturally and intuitively.
Another potential use case is in sports medicine, where the system could be used to monitor the movements of athletes. By providing insight on the type of movement the muscle is trying to make, the coach could provide more pointed advice and feedback for the athlete.

The rest of this report presents the design and implementation of the model, including data acquisition, and processing, and the development of the machine learning algorithm. I also provide the results of the performance of the model using MATLAB and discuss the future directions for this work.

Theory

In this section, I go over some definitions of different concepts that I use in my report. Firstly sEMG since it is the type of data that I am using all along my project.Secondly, I got over Support Vector Machines as it ends up being the most capable model in terms of accuracy and speed.

Lastly, definitions of some statistical methods are provided. These techniques are utilized as tools to make sense of the data and develop an understanding of principal components of my model.

Surface Electromyography (sEMG):

Surface Electromyography (sEMG) is a non-invasive technique that measures the electrical activity of muscles through electrodes placed on the surface of the skin. These electrical signals are generated by the motor units within the muscle fibers, and they “order” the muscle to act in a certain way. We can extract from these signals different time domain and/or frequency domain features that can be used to infer the muscle activity and the type of movement being performed, such as flexion, extension, or contraction[1].

Support Vector Machines (SVM):

SVMs are a kind of machine learning algorithm that can be used for classification problems. The goal of this technique is to create a hyperplane that separates the different classes while maximizing the distance between the data points of each classification class. SVMs are useful in non-linearly separable problems by mapping the different features to a higher dimensional space.

Receiver Operating Characteristic (ROC):

The ROC curve is a plot of True Positive Rate (TPR) vs the False Positive Rate (FPR) of our intended variable at various classification thresholds. These thresholds are defined differently within the hyperparameters of the different models. A perfect model would have a TPR of 1 and an FPR of 0 which would be defined as a point in the top left corner and two orthogonal lines $x = 0$ and $y = 1$.
By calculating the area under this curve we get a sense of how capable our model is at the intended classification.

Chi squared test (CHI2):

[1] Farina D. Interpretation of the surface electromyogram in dynamic contractions. Exerc Sport Sci Rev. 2006 Jul;34(3):121-7. doi: 10.1249/00003677-200607000-00006. PMID: 16829739.

Chi Squared is a statistical method that is generally used for hypothesis testing. In the context of variable selection, we use it to evaluate how important a variable is to our classification. This is done by defining the NULL hypothesis as : our studied variable is independent of the resulting classification variable. We end up with a score that tells us how relevant the given variable is.

Apparatus and data collection

In order to collect the sEMG data from the bicep, a Nucleo Board 401RE (figure 1) was used. This was connected to a MIKROE-2621 (EMG click) (figure 2) which linked the electrodes to the muscle. The microcontroller board allowed us to control the frequency at which we sampled the data using the analog input. The EMG click on the other hand had all the filters and amplifiers necessary to reduce noise.

In particular this hardware would first apply a High Pass filter pre-amplification, then the signal would get amplified since the voltage is small (in millivolts). Then the signal would go through another high pass filter. This combination of filters is used to limit noise and specifically the DC offset. Finally a third order 60 Hz low pass filter is utilized which attenuates frequencies higher than 60 Hz. It is also important to note that a Driven Right Leg block is part of the circuitry (figure 6). It is used to limit common mode interference which is especially prevalent in biological systems. This phenomenon refers to the fact that there is a lot of radiation around 50-60 Hz due to the grid around us.

I used the 5V pin and my GND pin on my NUCLEO board in order to power my EMG Click. Then I connected the ANALOG output of the EMG device to the ADC converter of my board. Finally, I used MBED studio in order to set up the necessary pins and upload the necessary code to my microcontroller. (shown in Appendix 1)

As the embedded system is collecting data, I connected a MATLAB program to the appropriate port in my computer in order to analyze these EMG signals and save them for post analysis. (shown in appendix 2)

Finally for the post processing step, I first uploaded the data file and parsed it into each particular movement. Since we are taking 10 to 20 movements per sample, we need to be able to break it down. The way this was done was by first having a peak detector that saves the starting point once we reach a certain threshold. This marked the beginning of our particular movement. Then, we have a window of 7 points that keeps track of the number of successive points that are within the range of 0 V. Once this condition became true, we concluded that the signal's movement reached its end.

Another processing step is normalizing the data. For this purpose, we use a Z-score normalization where we subtract the mean and divide by the std deviation. It is important to note that the normalization values used for the training data needs to be the same as the one used for the testing data. This means that the mean used for the training data might not be the actual mean of the testing sample. This is especially relevant for the frequency domain analysis where the features that we are using require a DC offset of 0. This leads us to normalize the testing data differently when extracting the time domain features and the frequency domain features.

Finally, I extracted the intended features using the “EMG Feature Extraction Toolbox”[2] on MATLAB. (all the post processing is shown in appendix 3.)

For my classification step, I used the “Classification Learner App” in MATLAB which gave me access to a number of predefined machine learning models (Trees, SVM, KNN etc.).

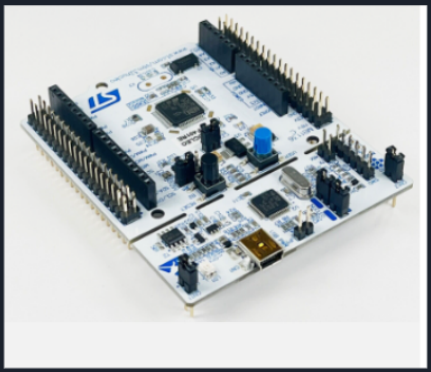


Figure 1: Nucleo Board 401RE

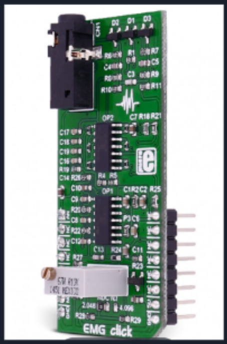


Figure 2: Click EMG

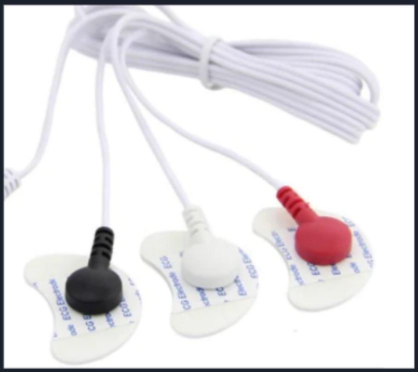


Figure 3 : Surface electrode connectors

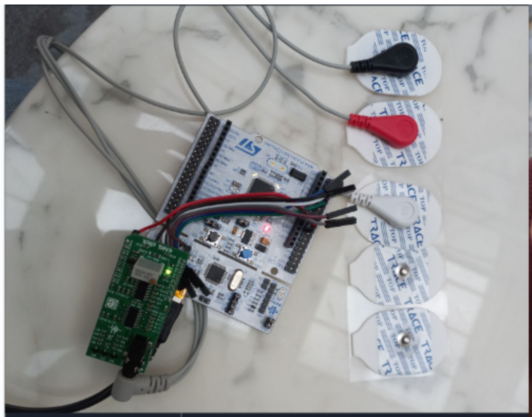


Figure 4 : connected hardware

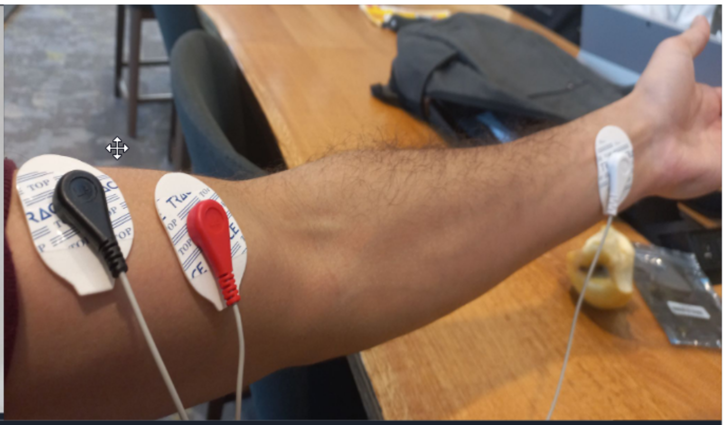


Figure 5 : electrode placement for data collection

[2] Too, Jingwei, et al. “Classification of Hand Movements Based on Discrete Wavelet Transform and Enhanced Feature Extraction.” International Journal of Advanced Computer Science and Applications, vol. 10, no. 6, The Science and Information Organization, 2019, doi:10.14569/ijacsa.2019.0100612. (https://www.mathworks.com/matlabcentral/fileexchange/71514-emg-feature-extraction-toolbox)

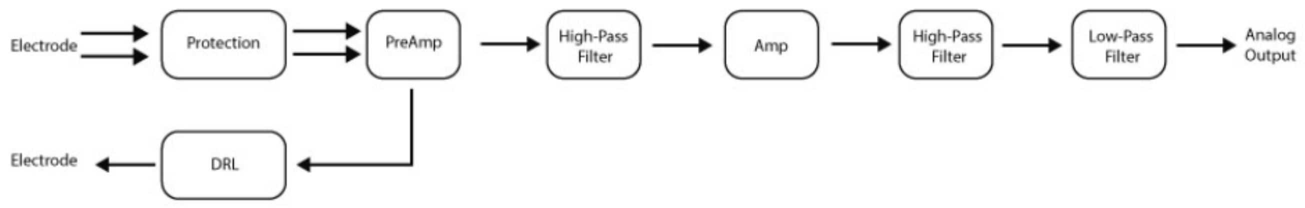


Figure 6 : hardware underlying the EMG click

Health Concerns and Standards

The collection of these EMG signals is usually done through a placement of electrodes close to the muscle group. This is either done on the surface of the skin or using a needle. Needles can be a source of infection and discomfort, which is why I have opted for the non-intrusive skin option.

For electrical safety it is required that the EMG equipment is insulated from the power line and that any accessible part of the instrument is connected to the ground. The electrical hazard for the patient is generally caused by faults in the grounding system.
I am actively respecting this constraint since everything is powered through the USB of my computer which has an isolated ground.

The MIKROE-2621 also comes with Overvoltage protection and Overcurrent protection to add a layer of safety to the respondents. [3]

It is also worth noting that I am not running a clinical study and thus my data will be limited to me. This means that choosing an algorithm that does not require a large amount of data is necessary.

Results:

[3] Extra documentation on the MIKROE-2621 at https://www.mikroe.com/emg-click

Qualitative analysis :

In order for us to choose what features to use in our modeling, we first investigated qualitatively. Firstly, I superimposed the frequency spectrum of both movements in order to get a sense of the effect of frequency. As is shown in figure 7, we can see that for ranges [0,10]Hz, [30,50]Hz and [60,100]Hz there is an apparent difference between the two spectrums. However for other ranges it is harder to make a conclusion. It is also interesting to see here that even though we have a 60Hz low pass filter, there is still some power in the frequencies above it.

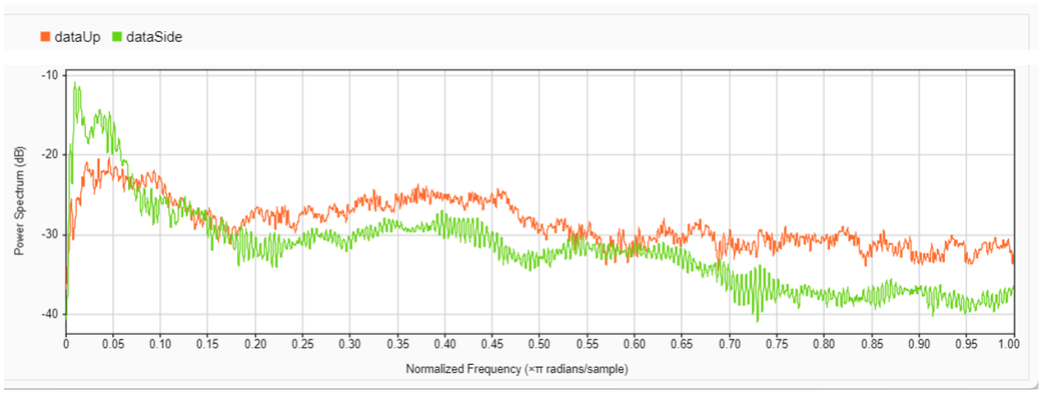


Figure 7 : Frequency domain spectrum of both movements superposed

As of the time domain, we can see in figure 8 that the “UP” movement is generally of longer length and that it had higher peaks at the beginning of the movement. These peaks quickly fall off. Whereas the “SIDE” signals were of lower peaks yet more sustained amplitude later on.

It is important to note that this is not always the case and that we chose examples that emphasized this trend in the report for sake of clarity. In fact these signals depend on the intensity of the movement, whether the muscle is tired or not, how fresh the gel is and a number of other factors that might influence the signal and its acquisition.

This part of the process was a way for us to pick the features that might be most relevant for the model. As we will later see, our intuitions were mainly right but there were still shortcomings.

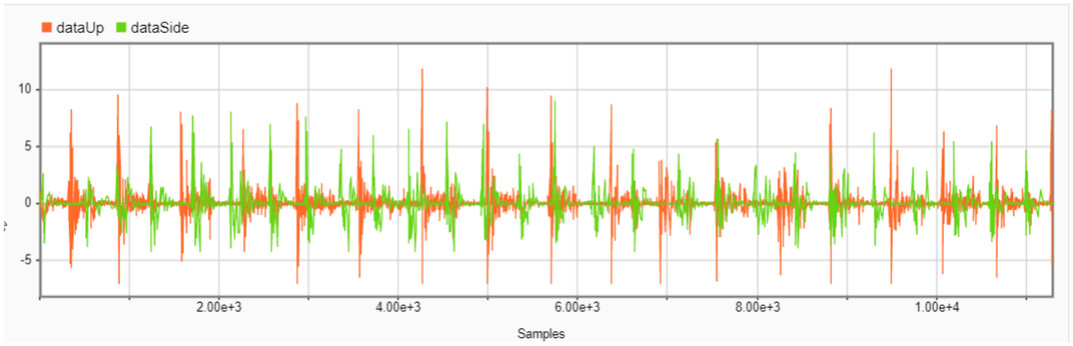
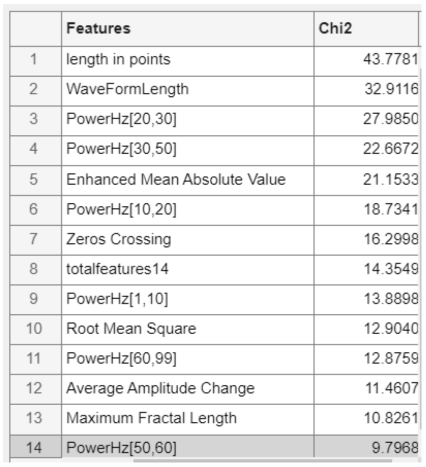


Figure 8 : Superimposed UP and SIDE signals in the Time domain

Quantitative analysis:

This qualitative analysis is what led us to choose the following features on the time and frequency domain. These features are listed below (figure 9) in order of importance. It is just as important to find the right model that delivers the best results as it is important to develop an intuition as to why that is the case. In fact, reducing the number of features can reduce the size of our model and the amount of the needed data for training.



	Features	Chi2
1	length in points	43.7781
2	WaveFormLength	32.9116
3	PowerHz[20,30]	27.9850
4	PowerHz[30,50]	22.6672
5	Enhanced Mean Absolute Value	21.1533
6	PowerHz[10,20]	18.7341
7	Zeros Crossing	16.2998
8	totalfeatures14	14.3549
9	PowerHz[1,10]	13.8898
10	Root Mean Square	12.9040
11	PowerHz[60,99]	12.8759
12	Average Amplitude Change	11.4607
13	Maximum Fractal Length	10.8261
14	PowerHz[50,60]	9.7968

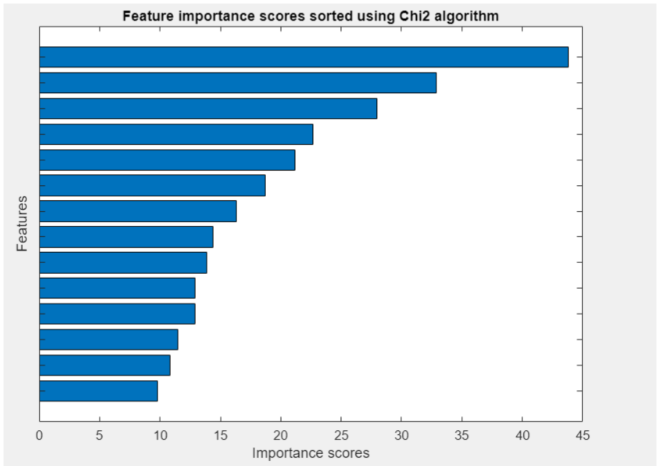


Figure 9 : List of features and their importance using the Chi2 algorithm

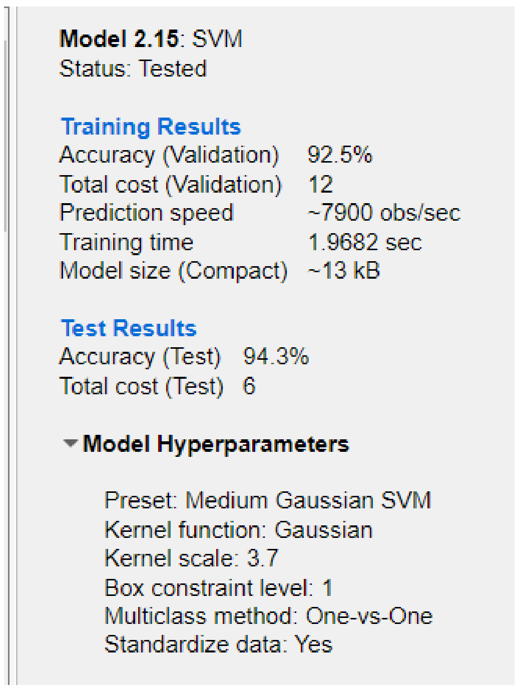
Feature	Description
Waveform Length / Length in points	Length of the signal in seconds or in number of points
Enhanced Mean Absolute Value	The mean absolute value based on the wavelet transform[4]
Zeros Crossing	Number of times the signal goes through the 0 volt point
Root Mean Square	RMS is the effective value of a varying voltage.
Average Amplitude Change	The average delta between every two consecutive points.[5]
Maximum Fractal Length	Similar to waveform length but expressed on a logarithmic scale less susceptible to noise.
PowerHz[x, y]	Power of the signal between x and y divided by the total power

Table 1: The selected features and their description

My models were trained and tested on the classification learner app on Matlab. I trained multiple different models and compared the resulting accuracies. Based on the empirical data, I have concluded that SVM is the best performing model. In fact we have achieved 94.3% accuracy on our testing data. This model is also the fastest performing model among the highly accurate ones when it comes to making predictions, with an estimated prediction speed of 7900 predictions per sec. This number is not to be taken into account as an absolute but as a relative measure among the different models and their summary (figure 10)suggested by MATLAB.
It is important to note that we have split our total data to 60 % training data and 40% testing data. This was done in order to make sure that we don't overfit our model and that our testing is representative.

[4] Jingwei Too, Abdul Rahim Abdullah and Norhashimah Mohd Saad, "Classification of Hand Movements based on Discrete Wavelet Transform and Enhanced Feature Extraction" International Journal of Advanced Computer Science and Applications(IJACSA), 10(6), 2019

[5] R. Barioul, S. Fakhfakh, H. Derbel and O. Kanoun, "Evaluation of EMG Signal Time Domain Features for Hand Gesture Distinction," *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*, Istanbul, Turkey, 2019, pp. 489-493, doi: 10.1109/SSD.2019.8893277.



Model 2.15: SVM
Status: Tested

Training Results

Accuracy (Validation)	92.5%
Total cost (Validation)	12
Prediction speed	~7900 obs/sec
Training time	1.9682 sec
Model size (Compact)	~13 kB

Test Results

Accuracy (Test)	94.3%
Total cost (Test)	6

Model Hyperparameters

Preset: Medium Gaussian SVM
Kernel function: Gaussian
Kernel scale: 3.7
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: Yes

Figure 10: Summary of my final SVM model

As shown in figure 11 and 12, our final model has an AUC on its ROC curve of more than 97% for both validation and testing data. Validation refers to the results on our training data, whereas test refers to the results on our testing data. Our program optimizes systematically for our validation ROC which explains why the actual model is on the point closest to (0,1). However, it is already set when drawing the curve for my test data.

We first observe that our AUCs are fairly high. This means that our model for some hyperparameters is able to solve the problem at hand. The second thing we observe is that for both cases, our Model Operating Point is closest to the (0,1) point which tells us that our training data is representative of what was faced during the testing.

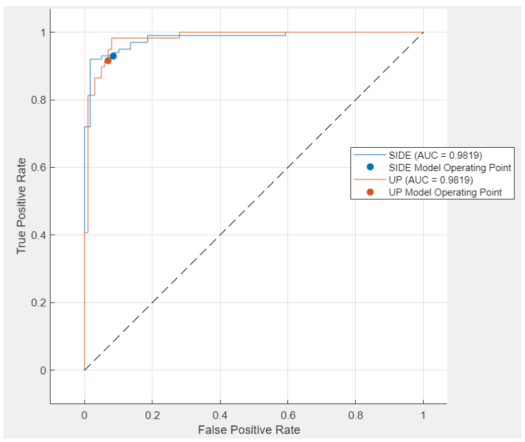


Figure 11 : Validation ROC curve

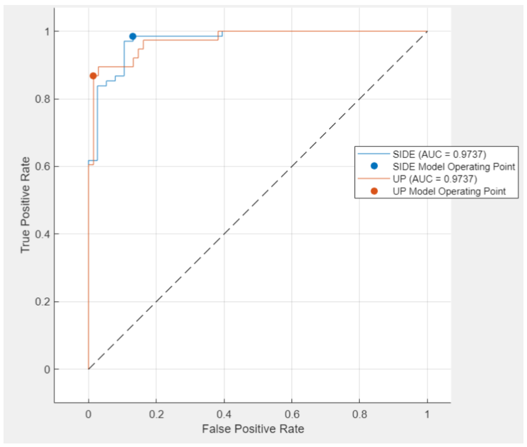


Figure 12: Test ROC curve

As is shown below in the Test Confusion Matrix (figure 13 and 14) , our model successfully categorizes “SIDE” movements with a true Positive Rate of 98.5%. However it doesn’t perform as well with the UP movement, receiving a False Negative Rate of 13.2%.

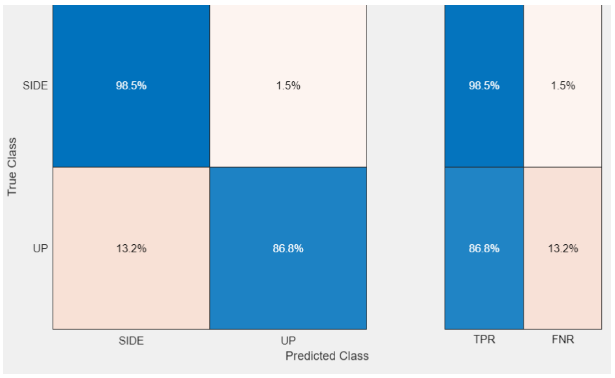


Figure 13: Test Confusion Matrix

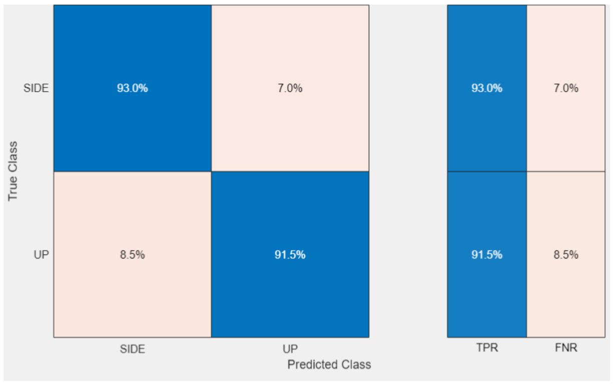


Figure 14: Validation Confusion Matrix

The plot below (figure 15) shows the distribution of the different features using a Z scaling normalization (subtracting mean and dividing by the std deviation) based on our final model. This shows the distribution and the separability of the different movements based on the existing features. This plot suggests that the most relevant features are the length and power in the range [20,30] Hz. This is shown by the darkness of the blue and red on either side of the 0 line and the lack of intersectionality among those two features.

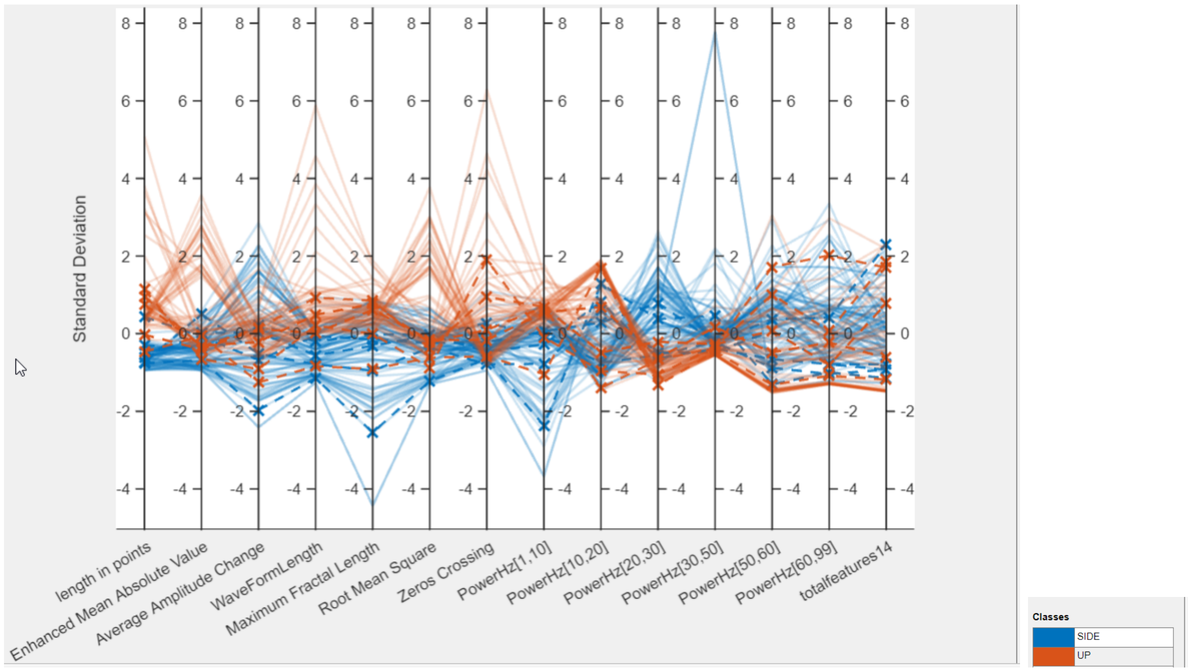


Figure 15 : Parallel Coordinates Plot

As part of the analysis and attempt to understand the relevant information that govern our classification model I investigated the effect of oversampling and bandwidth.I have prepared three different models that work with different sets of data. The first is our original signal, the second is an undersampled signal and finally the third is an oversampled signal. Table 2 shows the different accuracies of the different models.

Model	[0 ,100]Hz bandwidth	[0 ,50]Hz bandwidth Undersampled	[0,50]Hz bandwidth Oversampled
Test Accuracy	94%	86,1 %	85,2 %

Table 2: Accuracies of the oversampled, undersampled and original model

Discussion

Delimiting algorithm:

One of the challenges in this project was to delimit the start and end of each signal accurately. The final version uses the parsing technique described in the methods part. Originally however, I used an algorithm that kept track of a moving average. However this did not yield good results since the EMG signal is highly symmetrical around the 0V point, and thus my moving average would hit this threshold value fairly regularly.
This meant that our delimiting algorithm was setting the end of a sample in the middle of the signal.

Getting more quality data thus came at the cost of trimming down the samples that were not captured in their entirety. This was achieved by setting a set of conditions that were used to determine whether a sample should be used for training. In my case I based this on a minimum signal length and a minimum average amplitude change.
However, as our parsing algorithm improved in performance, so did our classification model as we got higher quality data and increased in data size as less trimming was required.

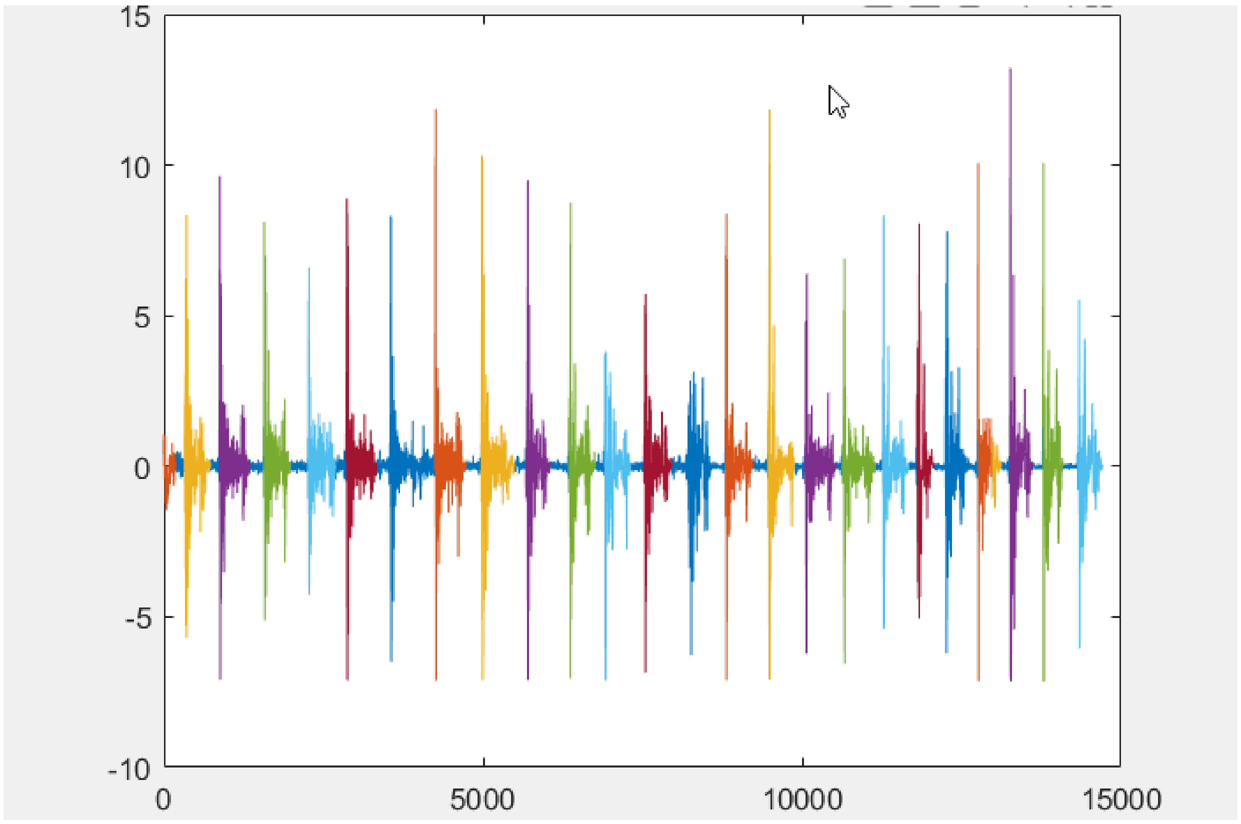


Figure 16: example of partitioned data (different colors for different samples)

We can see in the example above (figure 16) how the algorithm successfully parses the sample with multiple movements of different high peaks and lengths.

Normalization:

The normalization technique that was used was the Z normalization where we subtract the mean from the data and divide by the standard deviation.This technique was used in this project for multiple reasons. The first reason came about when I was dealing with the frequency domain features. One of the features used in our model was the percent energy stored in different frequency ranges. However, every span of frequencies returned values less than 1% other than the [0,10] which was greater than 99%. This meant that the DC value was so high that it stomped the other fractions when dividing by the total power.
By subtracting the mean, I was able to nullify the DC component and thus I had access to much more meaningful values across the whole range of frequencies.

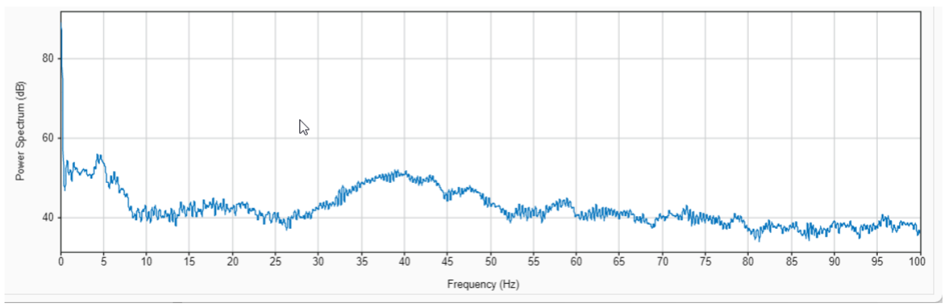


Figure 17 : frequency analysis pre-normalization

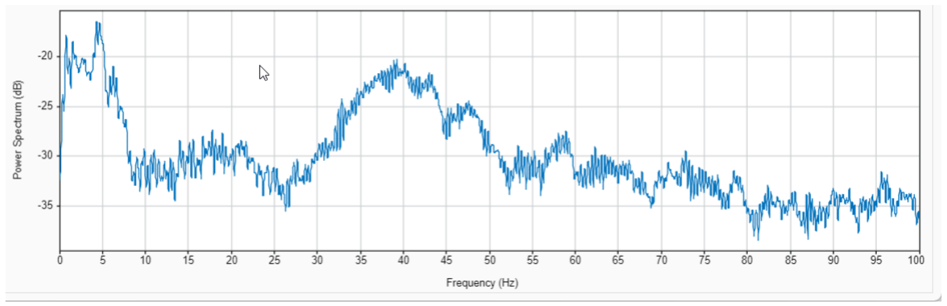
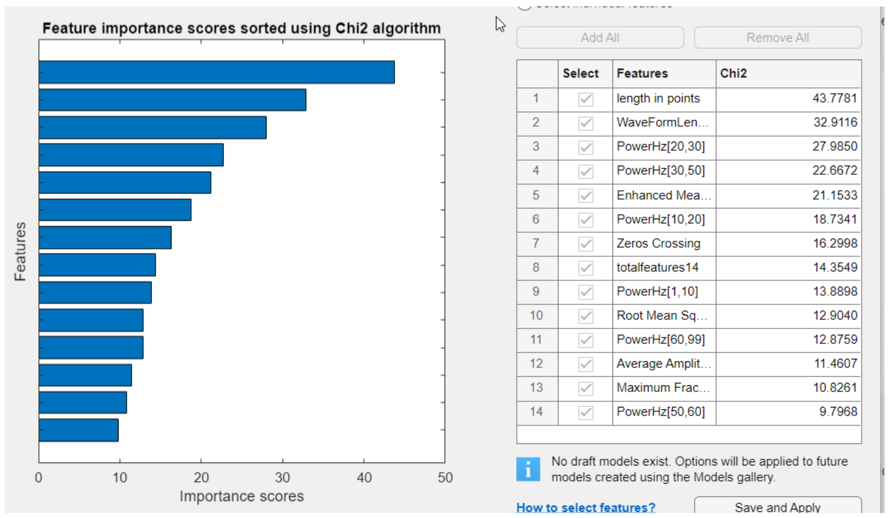


Figure 18: frequency analysis after eliminating the DC component through normalization

I originally theorized that by normalizing the data of my features this would improve the results of my model as suggested in different articles. The idea is that by keeping all training data within a similar range, it wouldn't give more weight to one attribute over another. Nonetheless empirically it dropped our accuracy from 80% to 70%. My theory is that the features that had higher ranges and thus potentially more weight were also the most pronounced and important features, namely the ones proportional to the length of the signal. This can be shown in the table below (figure 19), where the most significant variables are indeed the "length in points" and the "WaveformLength".



	Select	Features	Chi2
1	✓	length in points	43.7781
2	✓	WaveFormLen…	32.9116
3	✓	PowerHz[20,30]	27.9850
4	✓	PowerHz[30,50]	22.6672
5	✓	Enhanced Mea…	21.1533
6	✓	PowerHz[10,20]	18.7341
7	✓	Zeros Crossing	16.2998
8	✓	totalfeatures14	14.3549
9	✓	PowerHz[1,10]	13.8898
10	✓	Root Mean Sq…	12.9040
11	✓	PowerHz[60,99]	12.8759
12	✓	Average Amplit…	11.4607
13	✓	Maximum Frac…	10.8261
14	✓	PowerHz[50,60]	9.7968

No draft models exist. Options will be applied to future models created using the Models gallery.

Figure 19: feature importance scores using CHI2 algorithm

Sampling problem:

An interesting problem that I also tried to look into was the effect of resolution vs the effect of bandwidth. In fact we are able to increase the resolution of the signal by compromising bandwidth. This is accomplished through oversampling. This technique works by averaging every two consecutive points. Effectively, this reduces the bandwidth by half but also reduces the quantization error due to Analog to Digital conversion.

In order to experimentally test this, we created 3 different samples. The first one is our original signal sampled at 200Hz, which according to Nyquist's theorem gives us a bandwidth of 100Hz. The second sample was derived from the first one by dropping every other point (undersampling) . Effectively, this meant that we sampled at 100Hz and thus we had a bandwidth of 50Hz. Finally for the third sample, we averaged every two consecutive points of the original signal (oversampling) . This also results in a bandwidth of 50Hz but with an increased resolution due to the reduction of quantization error.

Finally we ran the 3 different kinds of signals through our training model and compared the results. (Table 2)
When comparing the oversampled model and the undersampled model, we notice that they both yield similar results which leads us to conclude that an increased resolution does not affect the effectiveness of our model.

Then when we compare our model with increased bandwidth to the other two, we notice that we get a significant increase in our accuracy. We conclude that [50;100]Hz bandwidth still holds significant information that the model can use for classification. We also conclude that the increased resolution did not result in much improvement. We theorize that this is due to the fact that oversampling does not affect the time domain features and that its effect on the frequency domain is marginal since we are not facing a problem of aliasing.

Data Distribution:

When dealing with these machine learning models, I have had to constantly make sense of my data and look for patterns or potential problems that might corrupt its distribution. In fact my results are only as good as my data. If it is representative, the model yields good results however if it has some situational pattern then the model might rely on a non-representative set and thus have trouble when facing the testing data.

One of my initial models for example would work well on its training data but then fail with the testing data. This pointed toward overfitting. When looking at the confusion matrix it became apparent that my model overfit to the "UP" movement. After investigating the data used, I theorized that this problem was due to the fact that I had 3 times more examples of "UP" than the "SIDE". Indeed, after acquiring more data of "SIDE", this overfitting vanished and we started getting better results.

Another similar instance showcased the importance of generalization. I indeed had a model that was theoretically capable of solving both testing and training problems yet it was failing at testing.
I found out that the area under the curve of my ROC was higher than 0.9 for both testing and training data yet the predictive performance on test data was around 70%. This meant that my model had the potential of predicting both if it were to adjust its hyperparameters.

Initially, my testing data was acquired under different circumstances than my training data. Meaning that it was taken at a different time with a different gel on a different arm, and that data was preserved as its own testing file. The reasoning behind this was that if I wanted to test how general my solution is, it needed to be facing data that was acquired in different circumstances than the training.
Nonetheless with the limited amount of data that I generally had, this was too ambitious. This led me to mix both sets and instead train on 60% of a random subset of the mix and test on the other 40%.
This eventually allowed my training phase to adjust to these different scenarios and enabled my model to achieve around 95% accuracy on the testing data.

This problem of data distribution made me realize that data acquisition happening under different circumstances might affect the data in different ways. Thus it is important to vary these circumstances during training as much as we can in order to generalize our solution, as we do not know the circumstances under which the device will be used.

This also meant that my results on MATLAB are not sufficient and that further testing in real time was necessary in order to get a more accurate picture of overall performance.

Conclusion and Considerations

Overall, the model worked well and was able to achieve the classification at 94.3%. The model is also compact (13KB) and relatively fast in terms of execution. This makes it a good candidate to translate into a C++ code that would run on the Nucleo Board.
However, it is only as good as the data that was fed to it. Thus, knowing that it was from a single individual and with a limited amount of data, this makes its ability to generalize this task dubious. Further testing in real time would be required in order to understand the conditions that affect performance. On a personal level, this pushed me to be creative in the ways I approach the data and develop an intuition as to what is relevant.

Acknowledgements

Special thanks to my advisor Professor Maggie Delano who guided me through the project and offered valuable suggestions, explanations and advice.

Special thanks to professor Allan Moser who helped me navigate Matlab, approach the algorithmic side of the project and for his general advice.

Special thanks to Cassy Burnett who ordered all the necessary materials for the project.

Appendices

Appendix 1: C++ code used on NUCLEO board for data acquisition.

```
/*
 * Copyright (c) 2017-2020 Arm Limited and affiliates.
 * SPDX-License-Identifier: Apache-2.0
 */

#include "mbed.h"
#include <chrono>
#include <cmath>

// Initialize a pins to perform analog input and digital output functions
AnalogIn   ain(A0);
DigitalOut dout(LED1);
Timer t1;
using namespace std::chrono;

BufferedSerial pc(USBTX,USBRX);
char buff[6]; // largest value is 65535, new line

Ticker sample;

void sampleADC(){
    unsigned short value = ain.read_u16();

    buff[0] = value/10000 + 0x30;
    value = value - (buff[0]-0x30)*10000;

    buff[1] = value/1000 + 0x30;
    value = value - (buff[1]-0x30)*1000;

    buff[2] = value/100 + 0x30;
    value = value - (buff[2]-0x30)*100;

    buff[3] = value/10 + 0x30;
    value = value - (buff[3]-0x30)*10;

    buff[4] = value/1 + 0x30;

    pc.write(&buff,6);
}

int main(void)
{
  buff[5] = '\n';

  sample.attach(&sampleADC,5ms);

  while (1) {
    thread_sleep_for(100);

  }
}
```

Appendix 2 : MATLAB code used for visualizing and saving data a the time of data acquisition.

```
function plotAndOutputSignal(portIn, Fs, windowWidth)
%plotAndOutputIHR Plot Incoming ECG Waveform and Output Instantaneous Heart
% Rate. Fs in Hz (set to 1/Ticker period) and windowWidth in seconds.
   port = serialport(portIn,14400); % initialize serial port
   datapoints = []; % array of data
   time = []; % array of times
   numPoints = 0; % counting number of points
   feat = []; % keeping track of different features of each signal
   xWidth = Fs*windowWidth; % calculate windowWidth in indices using Fs
   figure % new figure each time, can comment out
   % loop forever, hit ctrl+c in MATLAB window to cancel

   firstIndex = 0;
   lastIndex = 0;
   buffSize = 100; %size of the moving window
   buff = zeros(buffSize,1);
   indexBuff=mod(-1,buffSize) ;
   average = 0;
   confLevel = 2 / sqrt(buffSize);
   thresholdHigh = 22000;
   thresholdLow = 18000;
   counting = false;
   while(true)
       datapoint = str2double(port.readline); % get datapoint, one per line

       datapoints = [datapoints,datapoint]; % save datapoint
```

```
        time = [time,numPoints*1/(Fs)]; % update time array
        numPoints = numPoints + 1; % increment number of points
        % adjust to window width
        if (xWidth < numPoints)
            xlim([time(numPoints - xWidth) time(numPoints)])
        else
            xlim([0 windowWidth]);
        end

        % maintain full y axis range, feel free to adjust
        ylim([0 70000]);


%          % plot peaks
%          hold on
%          plot(peakTimes,peaks,'Color','red','Marker','o','LineStyle','none');
%          hold off

        % Code seems to slow down a lot with these on...feel free to try it
        % xlabel('Time (seconds');
        % ylabel('ADC count');
         save('fileName','datapoints','feat');
    end
End
```

Appendix 3: MATLAB code used for data post processing.

```
function [yUP, featuresTest, Class] = LoadAndAnalyzeNormalizedMod(filename,
Class,featuresTest)
load(filename,"datapoints");
yUP = datapoints(1,2:length(datapoints));
yUP = yUP';
% Normalization
meanV = mean(yUP);
stdV = std(yUP);
yUP = (yUP - meanV) ./stdV;

firstIndex = 0;
lastIndex = 0;
endSize = 7; %size of the inactive points for the end of the signals
buffSize = 100; %size of the moving window
buff = zeros(buffSize,1);
indexBuff=mod(-1,buffSize) ;
average = 0;
```

```matlab
thresholdHigh = 2000 ./ stdV;
thresholdLow = -2000 ./ stdV;
counting = false;
or= 1;
% hold on;
plot(yUP);
for j = 1:length(yUP)
   if (((yUP(j)> thresholdHigh) || (yUP(j)<thresholdLow)) && (counting ==
false))
       firstIndex = j;
       buff = zeros(buffSize,1);
       counting = true;
       indexBuff = 1;
       average = 0;
       full = false;
   end

   if (counting==true)
         average = average - buff(indexBuff)./buffSize + yUP(j)./buffSize;
         buff(indexBuff) = yUP(j);
         if (indexBuff == buffSize)
           full = true;
         end
         indexBuff = mod((indexBuff),buffSize) + 1;
       closeCondition = true;
       if (full)
           for s = 1:endSize
               if ( ( yUP(j-s) < (-250 /stdV) )||( yUP(j-s) > (250 /stdV)) )
                   closeCondition = false;
               end
           end
       end

       if ( full && closeCondition)
           or = or * -1;
           counting = false;
           lastIndex = j;
%           plotting = true;
           firstIndex
           lastIndex
           hold on
           plot(firstIndex:lastIndex,yUP(firstIndex:lastIndex,1));
           hold off
           % Enhanced Mean Absolute Value
           f1 = jfemg('emav', yUP(firstIndex:lastIndex,1));
           % Average Amplitude Change
           f2 = jfemg('aac', yUP(firstIndex:lastIndex,1));
           % Waveform Length
           f3 = jfemg('wl', yUP(firstIndex:lastIndex,1));
```

```
            % Maximum Fractal Length
            f4 = jfemg('mfl', yUP(firstIndex:lastIndex,1));
            % Root Mean Square
            f5 = jfemg('rms', yUP(firstIndex:lastIndex,1));
            % Zeros Crossing
            opts.thres = 0.01;
            f6 = jfemg('zc', yUP(firstIndex:lastIndex,1), opts);
            % LogTeagerKaiserEnergyOperator
            f7 = jfemg('ltkeo', yUP(firstIndex:lastIndex,1));

            % Feature vector
            totalPower = bandpower(yUP(firstIndex:lastIndex,1),200,[1,99]);
            hzP10 = bandpower(yUP(firstIndex:lastIndex,1),200,[1,10]);
            hzP20 = bandpower(yUP(firstIndex:lastIndex,1),200,[10,20]);
            hzP50 =  bandpower(yUP(firstIndex:lastIndex,1),200,[30,50]);
            hzP60 =  bandpower(yUP(firstIndex:lastIndex,1),200,[50,60]);
            hzP100 =  bandpower(yUP(firstIndex:lastIndex,1),200,[60,99]);
            featuresTest = [featuresTest ;lastIndex - firstIndex ,f1, f2, f3,
f4, f5, f6,f7, hzP10/totalPower, hzP20/totalPower, hzP30/totalPower,
hzP50/totalPower, hzP60/totalPower, hzP100/totalPower];
            Class = [Class; "SIDE"]; % adjust SIDE or UP depending on the data
you are reading
       end
   end


end
```